TUGAS AKHIR – KS 141501

PERANCANGAN APLIKASI PENCATATAN KEHADIRAN KULIAH PADA SMARTPHONE ANDROID DENGAN METODE PENGEMBANGAN ITERATIVE PARALLEL PROTOTYPING DAN ICONIX PROCESS

DESIGNING ATTENDANCE MARKING APPLICATION FOR ANDROID SMARTPHONE USING ITERATIVE PARALLEL PROTOYPING AND ICONIX PROCESS AS DEVELOPING METHOD

ASHR HAFIIZH TANTRI

NRP 5211 100 061

TUGAS AKHIR – KS14 1501

PERANCANGAN APLIKASI PENCATATAN KEHADIRAN KULIAH PADA SMARTPHONE ANDROID DENGAN METODE PENGEMBANGAN ITERATIVE PARALLEL PROTOTYPING DAN ICONIX PROCESS

Ashr Hafiizh Tantri
NRP 5211100061

Dosen Pembimbing I
Radityo Prasetianto Wibowo, S. Kom., M. Kom.

Dosen Pembimbing II
Rully Agus Hendrawan, S. Kom., M. Eng.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KS14 1501

DESIGNING ATTENDANCE MARKING APPLICATION FOR ANDROID SMARTPHONE USING ITERATIVE PARALLEL PROTOYPING AND ICONIX PROCESS AS DEVELOPING METHOD

Ashr Hafiizh Tantri
NRP 5211100061

Academic Supervisor I
Radityo Prasetianto Wibowo, S. Kom., M. Kom.

Academic Supervisor II
Rully Agus Hendrawan, S. Kom., M. Eng.

DEPARTEMENT OF INFORMATION SYSTEM
Faculty of Information Technology
Institute of Technology Sepuluh Nopember
Surabaya 2017

LEMBAR PENGESAHAN

PERANCANGAN APLIKASI PENCATATAN KEHADIRAN KULIAH PADA SMARTPHONE ANDROID DENGAN METODE PENGEMBANGAN ITERATIVE PARALLEL PROTOTYPING DAN ICONIX PROCESS

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
Pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
Oleh :

Ashr Hafiizh Tantri
NRP. 5211 100 061

Surabaya, 19 Juli 2017

KETUA
JURUSAN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M.Kom.
NIP. 19650310 199102 1 001

LEMBAR PERSETUJUAN

PERANCANGAN APLIKASI PENCATATAN KEHADIRAN KULIAH PADA SMARTPHONE ANDROID DENGAN METODE PENGEMBANGAN ITERATIVE PARALLEL PROTOTYPING DAN ICONIX PROCESS

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

Ashr Hafiizh Tantri
NRP 5211 100 061

Disetujui Tim Penguji : Tanggal Ujian : 5 Juli 2017
Periode Wisuda : September2017

Radityo Prasetianto Wibowo, S.Kom, M.Kom **(Pembimbing I)**

Rully Agus Hendrawan, S.Kom, M.Eng **(Pembimbing II)**

Nur Aini Rakhmawati, S.Kom., M. Sc. Eng. **(Penguji I)**

Faisal Johan Atletiko, S.Kom., M. T. **(Penguji II)**

PERANCANGAN APLIKASI PENCATATAN KEHADIRAN KULIAH PADA SMARTPHONE ANDROID DENGAN METODE PENGEMBANGAN ITERATIVE PARALLEL PROTOTYPING DAN ICONIX PROCESS

Nama Mahasiswa : Ashr Hafiizh Tantri
NRP : 5211 100 061
Jurusan : Sistem Informasi FTIf- ITS
Dosen Pembimbing I : Radityo Prasetianto Wibowo, S. Kom., M. Kom.
Dosen Pembimbing II : Rully Agus Hendrawan, S. Kom., M. Eng.

ABSTRAK

Aplikasi pencatatan kehadiran berbasis smartphone android sebenarnya sudah cukup banyak dikembangkan, namun belum ada yang menunjukkan alasan yang kuat terkait pemilihan user interface untuk aplikasi. padahal perancangan user interface yang buruk bisa merugikan pengguna aplikasi.

Oleh karena itu, pada tugas akhir ini diajukan perancangan aplikasi yang berfokus ke user interface dengan menggunakan metode iterative parallel prototyping yang dijabarkan menjadi 3 tahapan, yaitu : low fidelity prototyping yang menghasilkan 3 prototipe UI, medium fidelity prototyping yang menghasilkan 2 prototipe UI, dan high fidelity prototyping yang menghasilkan 1 prototipe UI. Prototipe UI yang ada selanjutnya diuji dengan melibatkan calon pengguna menggunakan usability study serta melibatkan pakar UI menggunakan heuristic evaluation. Selanjutnya digunakan ICONIX Process untuk menjembatani antara desain UI dengan pembuatan baris kode.

Dengan luaran berupa mock up user interface serta beberapa dokumen pendukung, diharapkan pengembang aplikasi

pencatatan kehadiran bisa menggunakan tugas akhir ini sebagai acuan dari pengembangan user interface aplikasinya.

Kata kunci : Android, ICONIX Process, Iterative Parallel Prototyping, Pencatatan Kehadiran, User Interface

DESIGNING ATTENDANCE MARKING APPLICATION FOR ANDROID SMARTPHONE USING ITERATIVE PARALLEL PROTOTYPING AND ICONIX PROCESS AS DEVELOPING METHOD

Student's Name	**: Ashr Hafiizh Tantri**
NRP	**: 5211 100 061**
Departement	**: Sistem Informasi FTIf- ITS**
Supervisor I	**: Radityo Prasetianto Wibowo, S. Kom., M. Kom.**
Supervisor II	**: Rully Agus Hendrawan, S. Kom., M. Eng.**

ABSTRACT

There are a lot of attendance marking application for android smartphone that has been developed, but almost none of them explain the reason for choosing specific user interface for the application. Even though bad user interface might have adverse effects for the user.

Hence, this undergraduate thesis is proposing an aplication designing using iterative parallel prototyping as user interface developing method which is detailed into 3 steps : low fidelity prototyping which produce 3 UI prototype, medium fidelity prototyping which produce 2 UI prototype, and high fidelity prototyping which produce 1 UI prototype. The existing prototype then tested with usability study which involving user participant and heuristic evaluation which involving UI expert. After that, ICONIX Process is used to bridge between UI Design and writing source code.

The output of this research ,user interface mock up and additional documents, hopefully can aid another attendance marking application developer when developing user interface for their application.

Keyword : Android, Attendance Marking, ICONIX Process, Iterative Parallel Prototyping, User Interface

KATA PENGANTAR

Puji syukur atas rahmat dan karunia Nya, penulis dapat menyelesaikan Tugas Akhir serta laporannya. Dalam Tugas Akhir ini, penulis mengambil topik yang berjudul

"Perancangan Aplikasi Pencatatan Kehadiran Kuliah Pada Smartphone Android Dengan Metode Pengembangan Iterative Parallel Prototyping dan Iconix Process"

Penulis berharap pada Tugas Akhir ini dapat memberikan informasi yang bermanfaat bagi pembaca, menjadi batu loncatan bagi penulis untuk terus menghasilkan karya, serta tidak cepat berpuas diri dan memberikan sumbangsih bagi ilmu pengetahuan.

Pada kesempatan ini, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah membantu dan membimbing pengerjaan Tugas Akhir, yaitu:

1. Keluarga yang selalu memotivasi, memberikan semangat, dan mendoakan penulis selama pengerjaan dan penyusunan laporan.
2. Bapak Radityo Prasetianto Wibowo, S. Kom, M. Kom. selaku pembimbing I dan Bapak Rully Agus Hendrawan, S. Kom, M. Eng. selaku pembimbing II senantiasa meluangkan waktunya serta memberikan ilmunya dan mengarahkan penulis dalam pengerjaan tugas akhir.
3. Bapak Dr. Apol Pribadi S., S. T., M. T. selaku dosen wali, terimakasih atas waktunya dan arahan yang diberikan selama penulis menjadi mahasiswa di Jurusan Sistem Informasi.
4. Kepada Ibu Nur Aini Rakhmawati, S. Kom., M. Sc. Eng. dan Bapak Faisal Johan Atletiko S.Kom, M.T. yang telah memberikan waktunya untuk menguji tugas akhir ini.
5. Mas Ricky Asrul Sani yang telah meluangkan waktunya untuk memberikan informasi akademik dan mengatur waktu sidang di laboratorium ADDI.
6. Bapak, Ibu Dosen dan Staff di Jurusan Sistem Informasi, ITS yang telah bersedia memberikan ilmu-ilmunya,

pengalaman serta memastikan kelancaran dan terselesainya urusan akademik.
7. Aldo Kelvianto dan Muhammad Furqon Haq yang telah bersedia menjadi mentor dalam pemrograman Android.
8. Teman-teman penulis, Hufadz, Rogeri, Fajar, Izzano, Vilat, Radhifan, Hafidz yang selalu senantiasa memberikan motivasi hingga penyusunan lapotan
9. Penghuni ADDI, IKTI, MSI, RDIB, dan SE yang telah mau menyediakan waktunya untuk berdiskusi terkait bahasan Tugas Akhir.

Terima kasih sebesar-besarnya kepada seluruh pihak yang tidak dapat disebutkan satu persatu secara langsung maupun tidak langsung selama pengerjaan laporan Tugas Akhir ini. Semoga kebaikan, waktu dan tenaga yang telah diberikan kepada penulis dibalas oleh Yang Maha Kuasa.

Surabaya, 17 Juli 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada Bab ini akan dipaparkan latar belakang, rumusan masalah, batasan permasalahan, tujuan, relevansi atau manfaat dari tugas akhir, keterkaitan dengan road map laboratorium Akusisi Data dan Diseminasi Informasi, serta target luaran yang diharapkan.

1.1. Latar Belakang

Pencatatan kehadiran adalah salah satu kegiatan yang umum dilakukan pada kegiatan perkuliahan. Setiap instansi pendidikan memiliki metode masing - masing dalam melakukan pencatatan kehadiran, mulai dari pencatatan kehadiran manual berbasis kertas, penggunaan *smart card* [1], pencatatan kehadiran berbasis *fingerprint* [2], penggunaan *smartphone* Android sebagai pencatat kehadiran [3] [4] [5] [6], dan lain sebagainya.

Penggunaan smartphone Android dalam pencatatan kehadiran sendiri bisa dikatakan teknologi yang cukup baru karena sistem operasi Android mulai populer pada tahun 2010. [7] Banyak pengembang aplikasi yang lebih suka mengembangkan aplikasi untuk *smartphone* Android dibandingkan *smartphone* dengan sistem operasi lainnya, hal ini terbukti dengan jumlah konten di Google Play Store dengan kata kunci "*attendance*" menampilkan 250 hasil. Sementara iTunes menampilkan 100 hasil dan Windows Store menampilkan 73 hasil dengan kata kunci yang sama.

Dari beberapa pengembangan aplikasi pencatatan kehadiran berbasis *smartphone* Android [3] [4] [5] [6], belum ada yang menjelaskan proses pembuatan *user interface* untuk aplikasi. Padahal, desain *user interface* yang buruk bisa membuang waktu, memboroskan uang, atau bahkan menghilangkan nyawa seseorang. [8] [9] [10] [11]

Untuk itu, pada tugas akhir ini menggunakan metode Iterative Parallel Prototyping untuk pengembangan *user interface* aplikasi. Metode Iterative Parallel Prototyping sendiri merupak-

an penggabungan dari 2 metode yang ada, yaitu *iterative prototyping* dan *parallel prototyping*. *Iterative prototyping* dipilih karena berdasar penelitian, luaran yang dihasilkan akan lebih baik jika dilakukan iterasi. [12] Sementara *parallel prototyping* dipilih karena berdasar penelitian, *parallel prototyping* mendorong adanya eksplorasi desain dan menghasilkan luaran yang bervariatif dan memiliki kualitas cukup baik. [13]

Namun dalam pengembangan *user interface* masih diperlukan tahapan validasi / uji untuk memastikan bahwa *user interface* yang dihasilkan sudah cukup baik, sehingga juga perlu dilakukan pengujian dengan menggunakan metode *usability study* dan *heuristic evaluation* untuk menguji prototipe *user interface* yang dihasilkan.

Dengan *user interface* yang sudah dikembangkan dan diuji, bukan berarti *user interface* yang dihasilkan akan bisa diimplementasikan menjadi program yang utuh, karena pengguna seringkali kesulitan menguraikan kebutuhannya secara rinci, sehingga terjadi *gap* antara desain dengan analisis. Untuk itu perlu dilakukan desain sistem untuk menjembatani *gap* tersebut.

Salah satu metode yang bisa digunakan untuk menjembatani *gap* tersebut adalah ICONIX Process. ICONIX Process digunakan dengan pertimbangan bahwa tahapan awal ICONIX Process adalah GUI *storyboarding* dimana luaran yang dihasilkan adalah prototipe GUI. Selain itu ICONIX Process memiliki keuntungan lain yaitu memaksa menghilangkan ambiguitas pada kebutuhan pengguna, menghindari terjadinya *analysis paralysis*, *lightweight* dan *highly iterative* namun tetap memfokuskan ke tahapan pembuatan kode secepat mungkin tanpa perlu mengesampingkan keuntungan dari tahap analisis *up-front* dan proses perancangan. [14]

1.2. Perumusan Masalah

Rumusan masalah yang diangkat dalam pengerjaan tugas akhir ini adalah:

1. Bagaimanakah cara merancang aplikasi pencatatan kehadiran yang memiliki UI yang memenuhi kaidah interaksi manusia komputer
2. Bagaimanakan cara melakukan desain sistem untuk aplikasi pencatatan kehadiran dengan menggunakan ICONIX Process

1.3. Batasan Masalah

Pengerjaan tugas akhir ini memiliki beberapa batasan masalah / Ruang Lingkup sebagai berikut :

1. Studi kasus dilakukan pada Jurusan Sistem Informasi ITS.
2. Responden yang dilibatkan dalam *usability study* adalah karyawan Jurusan Sistem Informasi ITS yang terlibat dalam aktivitas pencatatan kehadiran, sampel dari mahasiswa Jurusan Sistem Informasi ITS, dan sampel dari dosen Jurusan Sistem Informasi ITS.
3. Tahapan ICONIX Process hanya dilakukan sampai dengan tahap pembuatan *source code* yang dilakukan hanya sebatas pembuatan *mock up*
4. Pengerjaan Tugas akhir ini tidak meliputi bahasan keamanan infrastruktur teknologi informasi.

1.4. Tujuan Penelitian

Berdasarkan hasil perumusan masalah dan batasan masalah yang telah disebutkan sebelumnya, Tujuan dari pengerjaan tugas akhir ini adalah :

1. Merancang UI untuk aplikasi pencatatan kehadiran menggunakan metode Iterative Parallel Prototyping.
2. Melakukan desain sistem untuk aplikasi pencatatan kehadiran dengan menggunakan ICONIX Process.

1.5. Manfaat Penelitian

Manfaat dari implementasi tugas akhir ini yaitu sebagai acuan untuk mengermbangkan aplikasi pencatatan kehadiran

1.6. Relevansi

Tugas akhir ini berkaitan dengan mata kuliah Interaksi Manusia Komputer, Rancang Bangun Perangkat Lunak.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan membahas mengenai teori-teori yang mendukung dan mendasari penelitian tugas akhir ini serta beberapa studi literatur yang dapat menunjang Tugas Akhir yang diambil.

2.1. Studi Sebelumnya

Pada Subbab ini ditunjukkan beberapa penelitian yang menjadi dasar dari pengembangan perangkat lunak yang dilakukan pada tugas akhir ini.

2.1.1. Mobile Attendance Checking System on Android Platform for Kazakhstani University – N. Saparkhojayev, E. Shakhov, Y. Mailybayev (2016)

Paper ini membahas mengenai sistem presensi yang dikembangkan di universitas kazhakhstan. Sistem yang dikembangkan tidak fokus hanya di pencatatan kehadiran namun juga ada fitur untuk mengatur grup mahasiswa, *grading*, dan lain sebagainya. Cara kerja pencatatan kehadiran pada paper ini adalah menggunakan 1 *device* yang dipegang oleh dosen, dimana dosen berperan aktif untuk mengambil absen mahasiswa.

Pada *paper* ini ada metode *benchmarking* UI berdasar beberapa aplikasi yang sudah ada, meskipun UI yang dibuat masih kurang dijelaskan proses pembuatan dan validasinya. *Benchmarking* UI digunakan secara tidak langsung pada tugas akhir ini, dimana tahapan desain UI dilakukan dengan mempelajari beberapa aplikasi yang sudah ada. Selain itu, pada *paper* ini ada rancangan *database* yang bisa digunakan sebagai pertimbangan untuk pembuatan *domain model*, serta beberapa kebutuhan fungsional yang bisa diadaptasi untuk tugas akhir ini.

2.1.2. Development of The Online Student Attendance Monitoring System (SAMS™) Based on QR-Codes and Mobile Devices – A. A. Abd. Rahni, N. Zainal, M. F. Zainal Adna, N. E. Othman, M. F. Bukhori (2015)

Paper ini membahas aplikasi yang pencatatan kehadiran yang mengkombinasikan *QR-Code* dengan *smartphone* Android. Perancangan yang dilakukan pada jurnal ini adalah perancangan sistem, sehingga tidak terbatas pada aplikasi melainkan juga pengembangan *web apps* yang digunakan untuk pengirimian *e-mail* notifikasi, validasi data, dan fungsi lainnya.

Alur pencatatan kehadiran dimulai sejak mahasiswa didaftarkan di sistem, dimana sistem akan memberikan kode unik untuk setiap mahasiswa yang didaftarkan dan mengenkripsikan kode tersebut menjadi *QR-Code*. Untuk setiap kegiatan perkuliahan, mahasiswa diharuskan mendownload *QR-Code* terebut ke dalam *device* yang dimilikinya, dimana nanti dosen akan mencatat kehadiran dengan mengambil gambar *QR-Code* tersebut melalui *device* milik dosen itu. Aplikasi yang ada di *device* milik dosen nantinya melakukan proses dekripsi *QR-Code* dan melakukan validasi kode unik mahasiswa kepada *server*.

Tampilan GUI pada jurnal ini sudah cukup enak dipandang sehingga bisa digunakan sebagai pertimbangan dalam tahapan perancangan desain UI. Meskipun belum dijelaskan metode validasi GUI yang dihasilkan, namun sudah ada pengujian penerimaan responden dan hasilnya cukup baik.

2.1.3. Android Based Smart Learning and Attendance Management System – Rakhi Joshi, V. V. Shete, S. B. Somani (2015)

Paper ini membahas aplikasi yang pencatatan kehadiran yang meliputi *server* dan pemrosesan data yang ada pada *web apps*. Tahapan perancangan yang dibahas pada *paper* ini kurang rinci, hanya disebut sebagai *technical steps*.

Pengguna yang berperan aktif dalam proses pencatatan kehadiran adalah dosen, dengan pertimbangan mengurangi kecurangan yang terjadi, sehingga hanya ada 1 *device* yang digunakan dalam proses tersebut.

Pada *paper* ini disebutkan ada beberapa modul yang digunakan, yaitu ; *authentication module*, *student attendance module*, dan *database module* yang bisa digunakan sebagai referensi untuk tahapan desain sistem.

Selain itu ada beberapa komponen UI yang bisa digunakan sebagai pertimbangan dalam tahapan desain UI, meskipun tidak dijelaskan secara rinci bagaimana UI tersebut dihasilkan pada *paper* ini.

2.1.4. Parallel & Iterative Design + Competitive Testing = High Usability – Jakob Nielsen (2011)

Paper ini meneliti 3 metode yang bisa digunakan dalam mengembangkan *user interface*, yaitu *iterative design*, *parallel design*, dan *competitive testing*. Pada *paper* ini juga diberikan contoh penggunaan metode *parallel design* yang digabungkan dengan *iterative design*.

Metode *parallel design* dan *iterative* design pada *paper* ini diambil dan ditambahkan metode *parallel prototyping* yang dicontohkan Steven P.Dow er al. menjadi *iterative parallel prototyping* untuk digunakan dalam tahapan pengembangan desain antar-muka pengguna pada aplikasi yang dihasilkan di penelitian ini.

2.1.5. Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-Efficacy – Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L- Schwartz, Scott R. Klemmer (2010)

Paper ini membandingkan dampak *parallel prototyping* dengan *serial prototyping* dalam mengembangkan sebuah iklan grafis dengan batasan waktu yang sama. Hasil penelitian ini menunjukan bahwa metode *parallel prototyping* menghasilkan iklan grafis yang lebih menarik daripada *serial prototyping*.

Metode *parallel prototyping* pada *paper* ini digunakan dalam mengembangkan antar-muka pengguna pada aplikasi yang dihasilkan di penelitian ini.

Tabel 2-1 Rangkuman Studi Sebelumnya

No	Judul	Yang diambil untuk Tugas Akhir ini
1	Mobile Attendance Checking System on Android Platform for Kazakhstani University – N. Saparkhojayev, E. Shakhov, Y. Mailybayev (2016)	- Tahapan *benchmarking* UI - Kebutuhan fungsional dijadikan referensi dalam tahapan desain sistem
2	Development of The Online Student Attendance Monitoring System (SAMS™) Based on QR-Codes and Mobile Devices – A. A. Abd. Rahni, N. Zainal, M. F. Zainal Adna, N. E. Othman, M. F. Bukhori (2015)	- Penggunaan kamera untuk aplikasi - Komponen UI dijadikan sebagai referensi pada tahapan desain UI
3	Android Based Smart Learning and Attendance Management System – Rakhi Jo shi, V. V. Shete, S. B. Somani (2015)	- Modul yang ada dijadikan sebagai referensi pada tahapan desain sistem - Komponen UI dijadikan referensi pada tahapan desain UI
4	Parallel & Iterative Design + Competitive Testing = High Usability – Jakob Nielsen (2011)	- Metode *iterative design* dan *parallel design* diadaptasi dan digunakan dalam tahapan desain UI
5	Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-Efficacy – Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L- Schwartz, Scott R. Klemmer (2010)	- Metode *parallel prototyping* yang diajukan digunakan dalam tahapan desain UI

2.2. Dasar Teori

2.2.1.Pencatatan Kehadiran

Sejarah pencatatan kehadiran dimulai pada tahun 1889, ketika Willard L. Bundy menciptakan mesin pengukur waktu kerja untuk karyawan. [15] namun pada saat itu metode pencatatan kehadiran sebagian besar masih menggunakan pencatatan manual menggunakan lembaran kertas dan tulisan tangan.

Lalu pada tahun 1970an muncul mesin pencatatan kehadiran digital dimana karyawan memasukkan PIN / *password* masing masing untuk melakukan presensi.

Pada tahun 1964, IBM mengembangkan kartu pita magnetik yang selanjutnya digunakan sebagai dasar dari pengembangan mesin pencatatan kehadiran berbasis pita magnetik. [16]

Dengan dikenalkannya dunia *internet* dan *web* pada tahun 1990an, maka pengembangan sistem pencatatan kehadiran berbasis *web* mulai dikembangkan. [17]

Lalu pada tahun 2000an, penelitian mengenai biometrik dan aplikasinya mulai populer. [18] Sehingga ada beberapa produsen yang mulai mengembangkan mesin presensi berbasis identifikasi biometrik, terutama *fingerprint*, yang hingga sekarang masih sering dijumpai di beberapa instansi.

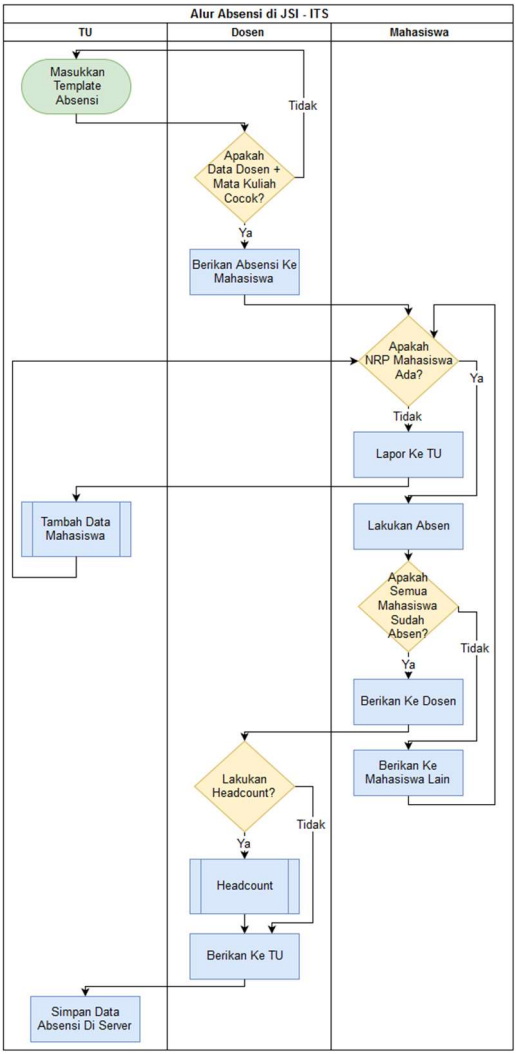
Yang paling baru adalah sistem pencatatan kehadiran berbasis *smartphone*, terutama dengan sistem operasi Android yang populer sejak tahun 2010. [7]

2.2.2.Jurusan Sistem Informasi – Institut Teknologi Sepuluh November (JSI – ITS)

Jurusan Sistem Informasi adalah salah satu Jurusan yang berada di bawah naungan Fakultas Teknologi Informasi pada Institut Teknologi Sepuluh November. Sebagaimana jurusan lain, pada JSI-ITS juga ada proses presensi / pencatatan kehadiran.

Sebenarnya sudah ada beberapa implementasi terkait presensi / pencatatan kehadiran yang ada di JSI – ITS, contohnya : *web apps* dan *fingerprint*. Namun karena sebab tertentu, JSI – ITS saat ini kembali menggunakan pencatatan kehadiran manual.

Dalam kegiatan pencatatan kehadiran di JSI-ITS, ada 3 pihak yang terlibat, yaitu: mahasiswa, dosen, dan pegawai TU. Berdasar hasil pengamatan awal, berikut ini alur yang terjadi dalam proses pencatatan kehadiran yang ada di JSI ITS:



Gambar 2-1 Swimlane Diagram Alur Pencatatan Kehadiran JSI-ITS

2.2.3. Android

Pada tahun 2007, Google Inc., T-Mobile, HTC, Qualcomm, Motorola dan pihak – pihak lain yang tergabung dalam Open Handset Alliance mengenalkan sistem operasi Android kepada publik. [19] Android diluncurkan dengan lisensi *open-source* untuk menyaingi iOs yang diluncurkan oleh Apple. [20] Dengan lisensi *open-source* maka publik bisa mengembangkan Android untuk tujuan mereka masing – masing sehingga perkembangan Android cukup pesat.

Fitur yang kemungkinan akan digunakan pada perancangan ini adalah GPS / Global Positioning System, Kamera, dan penyimpanan data. Dengan pertimbangan bahwa GPS digunakan untuk memastikan bahwa kegiatan pencatatan kehadiran memang dilakukan pada JSI-ITS, kamera digunakan untuk mengambil gambar mahasiswa yang bisa digunakan untuk memastikan kehadiran mahasiswa, dan penyimpanan data untuk menyimpan kamera dan data pencatatan kehadiran yang digunakan ataupun didapat dalam kegiatan tersebut.

2.2.4. User Interface

User interface adalah salah satu bahasan yang ada di bawah lingkup ilmu interaksi manusia dan komputer. Tujuan perancangan *user interface* adalah untuk mengantisipasi apa yang dibutuhkan oleh pengguna dan memastikan bahwa tampilan antarmuka yang ada memiliki elemen yang mudah untuk diakses, dipahami, dan bisa digunakan untuk memfasilitasi apa yang akan dilakukan oleh pengguna. [21]

Desain *user interface* penting karena ada beberapa kasus bahwa *user interface* yang buruk bisa membuang waktu, memboroskan uang, atau bahkan menghilangkan nyawa seseorang. [9] [10] [11]

Untuk perancangan user interface pada sistem operasi Android, Google telah menerbitkan panduan bernama “Material Design” sehingga desainer tidak memulai dengan tangan kosong. [22]

2.2.5.Iterative Parallel Prototyping

Iterative Parallel Prototyping yang dilakukan pada tugas akhir ini merupakan metode pengembangan *prototype* yang memadukan *iterative design* serta *parallel design* yang dibahas oleh Jakob Nielsen. [23]

Pada penelitian tugas akhir ini, penggunaan *iterative parallel prototyping* difokuskan pada pengembangan antarmuka pengguna / *user interface*. Dimana proses tersebut dibagi menjadi 3 tahap, yaitu: *low fidelity prototyping*, *medium fidelity prototyping*, dan *high fidelity prototyping*.

Low fidelity prototyping memiliki ciri khas tampilan yang masih sederhana dan bisa berubah ubah secara drastis dari setiap iterasi namun waktu yang dibutuhkan untuk membuat prototipe relatif cepat, sehingga bisa mendukung proses pengumpulan kebutuhan, dimana umumnya hal tersebut terjadi pada tahap awal perancangan.

High fidelity prototyping memiliki ciri khas tampilan yang sudah seperti program jadi namun memakan waktu yang tidak sedikit untuk membuat prototipe, sehingga lebih cocok untuk dilakukan ketika kebutuhan user sudah pasti, dimana umumnya terletak pada tahap akhir perancangan.

Sementara itu, *medium fidelity prototyping* digunakan untuk menjembatani antara *low fidelity prototyping* dan *high fidelity prototyping* dimana prototipe yang dihasilkan pada tahapan ini tampilannya sedang, tidak begitu baik, namun tidak begitu buruk. dan waktu yang dibutuhkan juga bisa dikatakan tidak terlalu cepat dan tidak terlalu lambat. Pada tahapan ini juga dilakukan validasi akhir untuk kebutuhan pengguna sehingga pada tahapan *high fidelity prototyping*, tampilan yang diajukan tidak berubah terlalu drastis.

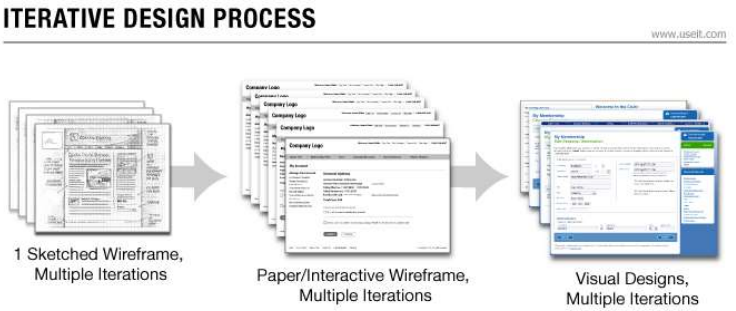
2.2.5.1. Iterative Design

Iterative Design adalah model proses pengembangan yang berfokus pada adanya iterasi dalam pembuatan desain. adanya iterasi dalam proses pengembangan memungkinkan luaran

yang dihasilkan lebih baik dibandingkan hasil luaran dari pengembangan tanpa iterasi. [12]

Secara sederhana, proses *iterative design* dilakukan dengan cara membuat sebuah versi desain lalu dilakukan pengujian untuk melihat kekurangan dari versi desain tersebut dimana kekurangan dari versi desain tersebut diperbaiki pada versi selanjutnya lalu dilakukan pengujian lagi untuk versi berikutnya dan seterusnya hingga proses iterasi dirasa cukup.

Ada kalanya proses iterasi dilakukan secara lintas medium, seperti yang dicontohkan oleh Jakob Nielsen pada gambar berikut :



Gambar 2-2 Contoh *Iterative Design Process* (Jakob Nielsen, 2011)

Seperti yang bisa dilihat pada Gambar 2-2 Contoh *Iterative Design Process* (Jakob Nielsen, 2011)iterasi yang dicontohkan dibagi menjadi 3 medium, yaitu : *sketched wireframe*, *paper/interactive wireframe*, dan *visual design*.

Selain itu, Jakob Nielsen juga menyarankan *Iterative Design* dilakukan dengan beberapa syarat berikut : [23]

- Iterasi dilakukan setidaknya 2 kali, namun lebih banyak iterasi lebih bagus
- Setidaknya dilakukan *simple user testing* (5 pengguna atau kurang) untuk menguji desain UI yang dihasilkan.

2.2.5.2. Parallel Design

Iterative design memiliki sebuah kelemahan fatal, yaitu *iterative design* membatasi pengembang untuk memperbaiki 1 desain saja, dengan kata lain jika desain awal salah maka berujung pada produk akhir yang tidak sesuai dengan yang diharapkan.

Pada pengerjaan *parallel design*, dibuat beberapa alternatif desain pada waktu yang sama dimana desain yang dihasilkan umumnya menunjukkan adanya perbedaan pada beberapa halaman / interaksi kunci.

Jakob Nielsen menyarankan setidaknya ada 3 alternatif desain yang dikembangkan.

Setelah alternatif desain dihasilkan, dilakukan pengujian dengan melibatkan pengguna. Menurut Jakob Nielsen, pengguna sebaiknya hanya diperbolehkan menguji 2 atau 3 alternatif desain karena jika terlalu banyak akan menyebabkan pengguna kesulitan mengungkapkan perbedaan antara alternatif desain yang ada. [23]

Jakob Nielsen juga menyarankan bahwa urutan alternatif desain yang diujikan sebaiknya juga digilir bergantian karena umumnya pengguna lebih fokus dengan desain yang mereka uji pertamakali. [23]

Setelah dilakukan pengujian, dilakukan penggabungan desain, mengambil ide terbaik dari setiap alternatif desain yang ada. *usability study* yang dilakukan tidak dilakukan untuk memilih mana alternatif desain yang dikembangkan, melainkan untuk menemukan bagian mana yang bagus dan bagian mana yang buruk dari setiap alternatif desain yang ada.

2.2.6. Usability Study

Usability study adalah salah satu pendekatan yang bisa digunakan untuk mendapatkan desain antarmuka pengguna yang baik. *Usability study* memiliki beberapa kelebihan dibandingkan *A/B Testing* dan *radical innovation*, berikut ini tabel perbandingan antara 3 pendekatan yang ditulis oleh Jakob Nielsen. [23]

Tabel 2-2 Perbandingan 3 Pendekatan Untuk Pengembangan Desain Interface (Jakob Nielsen, 2012)

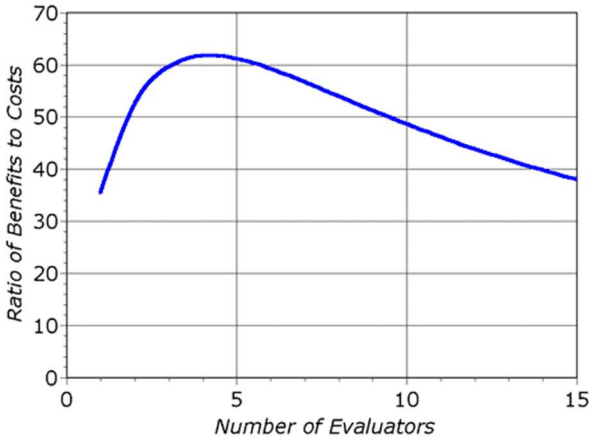
	A/B	***Usability***	***Radical***
Biaya	Rendah	Rendah–sedang	Tinggi
Keuntungan	1–10%	10–100%	100–1,000%
Resiko	Tidak ada	Rendah	Tinggi
Siapa yang bisa melakukan?	Semua orang	Semua orang	Orang Jenius
Seberapa sering?	Perminggu	Perbulan	10 tahun sekali
Dampak GDP	Sedang	Tinggi	Sedang

Menurut Jakob Nielsen, ada 4 matriks dasar yang bisa digunakan untuk mengukur *usability*, yaitu : [24]

- *Succes* r*ate*
- Waktu yang dibutuhkan untuk menyelesaikan t*ask / completion time*
- *Error rate*
- Kepuasan pengguna

Pada tugas akhir ini, keempat matrik itu digunakan secara bertahap. *Success rate* dan *completion time* digunakan pada tahap *low fidelity prototyping*, lalu pada tahap medium fidelity prototyping ditambahkan matriks kepuasan pengguna serta *error rate* secara bertahap berdasarkan iterasi yang dilakukan.

Untuk penentuan jumlah *evaluator* / responden yang digunakan pada tahapan *usability study*, digunakan teori yang diajukan oleh Jakob Nielsen, dimana Jakob Nielsen menyarankan bahwa jumlah *evaluator* yang optimal adalah 3-5 orang dengan menimbang rasio keuntungan terhadap biaya. [25]



Gambar 2-3 Rasio Keuntungan Terhadap Biaya Berdasarkan Jumlah Evaluator Yang Dilibatkan Dalam Usability Testing (Jakob Nielsen, 1995)

Pada *usability study* yang dilakukan pada tugas akhir ini, mahasiswa adalah kelompok pengguna utama dari pencatatan kehadiran, sehingga jika menggunakan rasio keuntungan biaya yang diajukan oleh Jakob Nielsen maka dibutuhkan 3 – 5 responden dari kelompok pengguna mahasiswa. Sementara untuk pegawai TU, hanya ada 1 orang yang terlibat dalam proses pencatatan kehadiran sehingga untuk setiap iterasi, digunakan pegawai TU yang sama untuk mengukur matrik *usability* dari sudut pandang pegawai TU. Sementara untuk dosen, diambil sampel 1 – 2 orang untuk setiap iterasi.

Ada beberapa langkah yang dilakukan dalam *usability study*, yaitu: [27]

1. Menjelaskan mengenai apa yang harus dilakukan oleh partisipan.
2. Menjelaskan kompensasi / reward yang diberikan kepada partisipan setelah selesai melakukan proses yang ada.
3. Menanyakan beberapa hal untuk memastikan bahwa partisipan memiliki kualifikasi untuk melakukan *usability study* dan mengetahui karakteristik partisipan.

4. Partisipan dipersilahkan mencoba prototipe dimana penguji mendampingi partisipan sambil mengajukan pertanyaan terkait elemen UI pada prototipe untuk mendapat *feedback* dari partisipan.
5. Menanyakan apa yang seharusnya ditambahkan, apa yang seharusnya diperbaiki, dan beberapa hal lain terkait prototipe setelah partisipan selesai mencoba prototipe.

2.2.7. Post-Study System Usability Questionnaire (PSSUQ)

PSSUQ adalah instrumen penelitian yang awalnya dikembangkan untuk menguji usability berdasar skenario di IBM. PSSUQ berisi 19 poin yang ditujukan untuk menjabarkan 5 karakteristik usability, yaitu : penyelesaian kerja dengan cepat, kemudahan untuk mempelajari sistem, dokumentasi kualitas tinggi, dan informasi online. [26]

Setelah responden mengisi kuosioner PSSUQ, contoh kuosioner PSSUQ dilampirkan pada Lampiran A, maka dilakukan proses rekapitulasi data dan mengumpulkan nilai yang didapatkan. Untuk mengartikan data yang didapatkan bisa menggunakan kuartil untuk membagi menjadi 4 nilai (sangat baik, baik, buruk, dan sangat buruk) ataupun bisa menggunakan statistik deskriptif untuk menjelaskan data kuantitatif.

2.2.8. Heuristic Evaluation

Heuristic evaluation adalah metode pemeriksaan pada perangkat lunak yang membantu mengidentifikasikan permasalahan usabilitas yang ada pada desain antarmuka pengguna. *Heuristic evaluation* menitikberatkan pada interaksi dengan *expert* untuk melakukan evaluasi antarmuka pengguna tanpa melibatkan pengguna. Namun pada TA ini, ‘expert’ yang digunakan adalah penulis, dengan merujuk pada *heuristic* yang diajukan oleh Rosa Yanez Gomez et al. yang berisi 230 subheuristik dan terbagi menjadi 13 heuristik. [27] Berikut ini tabel yang menunjukan 13 heuristik yang diajukan oleh Rosa Yanez Gomez et al.:

Tabel 2-3 13 Heuristik untuk Mobile (Rosa Yanez Gomez et al., 2014)

No	Heuristik
1	Visibility of system status
2	Match between system and the real world
3	User control and freedom
4	Consistency and standards
5	Error Prevention
6	Recognition rather than recall
7	Flexibility and efficiency of use
8	Aesthetic and minimalist design
9	Help users recognize, diagnose, and recover from errors
10	Help and documentation
11	Skills
12	Pleasurable and respectful interaction with the user
13	Privacy

Untuk daftar subheuristik yang digunakan bisa dilihat pada lampiran.

Ada beberapa langkah untuk melakukan *Heruristic Evaluation*, yaitu : [29]

1. Menentukan daftar heuristik yang akan digunakan
2. Menentukan evaluator yang akan digunakan
3. Memberi pengarahan untuk evaluator
4. Fase pengujian pertama
5. Fase pengujian kedua
6. Merekam permasalahan
7. Sesi diskusi / tanya jawab antar evaluator

2.2.9.Severity Rating

Severity rating bisa digunakan untuk mendapatkan perkiraan kasar dari bagian UI yang membutuhkan *usability* lebih sehingga penguji bisa mengalokasikan sumbar daya terbanyak untuk memperbaiki masalah paling serius. [28]

Severity rating yang digunakan dalam tahapan Heuristic Evaluation adalah yang diajukan oleh Jakob Nielsen dimana severity dibedakan berdasarkan dampak dan tingkat kesulitan untuk memperbaikinya. berikut ini adalah tabel yang menjelaskan *severity rating* yang digunakan :

Tabel 2-4 Severity Rating (Jakob Nielsen, 1995)

Severity Ranking	
Rating	**Penjelasan**
0	Hal ini bukan merupakan permasalahan *usability*
1	Masalah kosmetis : tidak perlu diperbaiki kecuali ada waktu ekstra dalam pengerjaan proyek
2	Masalah kecil *usability* : prioritas rendah untuk memperbaiki ini
3	Masalah besar *usability* : penting untuk diperbaiki, sehingga mendapat prioritas tinggi
4	Bencana *usability* : mendesak untuk diperbaiki sebelum produk dirilis

2.2.10. ICONIX Process

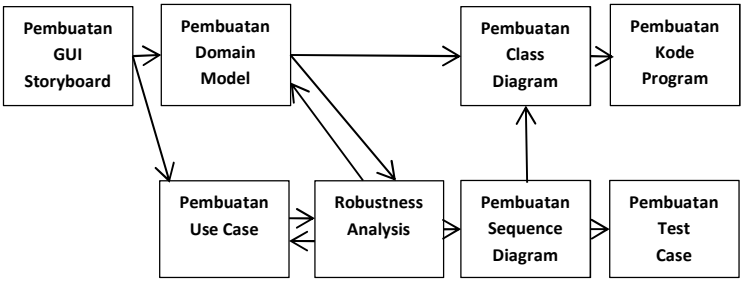
ICONIX Process adalah metode yang diajukan oleh Doug Rosenberg dan Matt Stephens sebagai jawaban dari pertanyaan bagaimana cara menuju ke *source code* melalui *use case*. Dimana pendekatan yang digunakan minimal tapi *sufficient* berdasarkan UML.

Tujuan utama dari ICONIX *Process* adalah melakukan pendekatan minimalis sehingga proses yang dilalui dari *use case* menuju *source code* dengan waktu yang relatif singkat.

Namun jika dirasa kurang, perancang / pengembang aplikasi bisa menambahkan elemen UML lain yang dirasa perlu.

Kelebihan iconix process adalah: memaksa menghilangkan ambiguitas pada kebutuhan pengguna, menghindari terjadinya *analysis paralysis*, *lightweight* dan *highly iterative* namun tetap memfokuskan ke tahapan pembuatan kode secepat mungkin tanpa perlu mengesampingkan keuntungan dari tahap analisis *up-front* dan proses perancangan. [14]

Pada ICONIX Process ada beberapa tahapan yang dilakukan, Berikut ini adalah tahapan yang ada di ICONIX Process secara garis besar:



Gambar 2-4 Tahapan ICONIX Process

Tahapan yang dilakukan pada ICONIX memang terlihat rumit, namun jika diruntut dari penjelasan di buku, tahapan paling awal adalah pembuatan GUI *Storyboard*, dimana pada tugas akhir ini dihasilkan dari metode Iterative Parallel Prototyping.

Selanjutnya dilakukan 2 hal hampir secara bersamaan, yaitu pembuatan *domain model* awal dan pembuatan *use case*. Dimana pembuatan 2 hal tersebut didasarkan dari prototipe antar muka pengguna yang sudah dibuat dan kebutuhan awal yang telah didapatkan.

Setelah itu dilakukan *robustness analysis* yang digunakan untuk memeriksa apakah use case sudah benar atau belum, dimana *robustness analysis* dilakukan secara iteratif hingga tidak

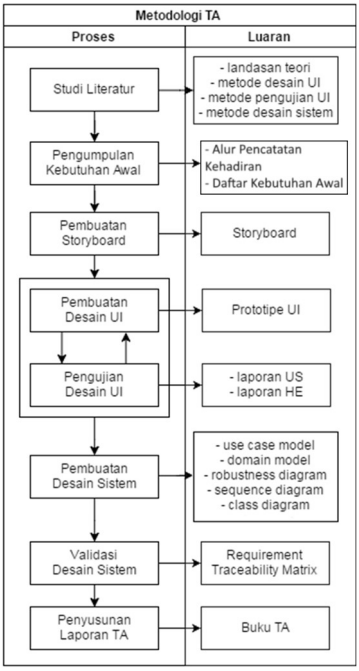
ditemukan kesalahan pada *use case* tersebut. Dan selama tahapan *robustness analysis* juga dilakukan pengecekan *domain model* berdasarkan elemen *entity* yang ada di *robustness diagram*. Setelah robustness analysis sudah selesai, maka dilakukan pembaruan domain model yang ditambahi juga dengan atribut untuk domain yang mungkin juga ditemukan sewaktu melakukan *robustness analysis*.

Setelah itu dilakukan pembuatan *sequence diagram* dan setelah itu dilakukan penyempurnaan *domain model* menjadi *class diagram* dengan cara ditambah elemen *method* yang kemungkinan juga ditemukan pada tahapan pembuatan *sequence diagram*.

BAB III
METODE PENELITIAN

3.1. Tahapan Pelaksanaan Tugas Akhir

Bab ini menjelaskan tentang metode yang dipakai dalam melaksanakan Tugas Akhir ini. Metode yang digunakan secara garis besar sesuai dengan gambar berikut ini.



Gambar 3-1 Metodologi Penelitian

3.1.1.Studi Literatur

Pada tahapan ini dilakukan penggalian informasi yang berhubungan dengan penelitian tugas akhir. Adapun input dari tahapan ini adalah *paper*, jurnal ilmiah, buku, dan media lainnya. Sementara luaran dari tahapan ini adalah : landasan teori, metode desain UI, metode pengujian UI, metode desain sistem.

3.1.2.Pengumpulan Kebutuhan Awal

Pengumpulan kebutuhan awal dilakukan dengan tujuan untuk mengetahui proses bisnis / *flow* yang terjadi seputar pencatatan kehadiran yang ada di Jurusan Sistem Informasi ITS, mengetahui pihak yang terlibat dan peran masing – masing dalam kegiatan pencatatan kehadiran, serta mengetahui kebutuhan dari masing – masing pihak yang terlibat dalam pencatatan kehadiran. Untuk mengumpulkan informasi tersebut, dilakukan observasi serta wawancara.

Luaran dari tahapan ini ada 2, yaitu alur pencatatan kehadiran yang telah diperbarui dan tabel daftar kebutuhan awal.

3.1.3.Pembuatan Storyboard

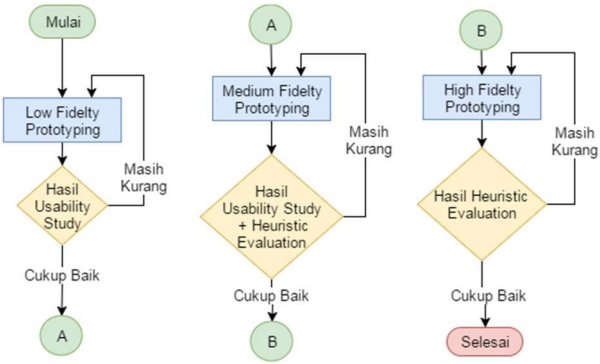
Setelah melakukan pengumpulan kebutuhan awal, tahapan selanjutnya adalah pembuatan *storyboard*. Di tahapan ini *storyboard* digunakan untuk membuat perkiraan awal bagaimana nanti aplikasi ini digunakan dalam pencatatan kehadiran jurusan sistem informasi.

Pembuatan *storyboard* didasarkan pada alur pencatatan kehadiran yang didapat dari tahapan sebelumnya. Proses pembuatan *storyboard* dilakukan dengan menggunakan kertas dan alat tulis yang dibutuhkan.

Luaran dari tahapan ini adalah *storyboard*.

3.1.4.Pembuatan Desain User Interface

Tahapan ini adalah pengembangan dari ICONIX *Process* pada bagian GUI *storyboarding*. Seperti namanya, pada tahapan ini dilakukan pembuatan purwarupa tampilan antarmuka untuk aplikasi. Berikut adalah gambar yang menjelaskan proses ini secara lebih rinci.



Gambar 3-2 Tahapan Desain User Interface

3.1.4.1. Low Fidelity Prototyping

Pada tahapan ini dilakukan pembuatan purwarupa yang didasarkan pada storyboard serta alur yang didapat dari tahapan pengumpulan kebutuhan awal. Purwarupa yang ada dibuat dengan menggunakan metode *paper and pen prototyping* dan *iterative parallel prototyping*, dimana nantinya akan dihasilkan 3 buah purwarupa.

Luaran dari tahapan ini berupa purwarupa antarmuka pengguna. Untuk tahapan ini, dilakukan iterasi berdasarkan hasil pengujian desain UI, dimana jika hasil pengujian dirasa kurang akan dilakukan perbaikan *prototipe*.

3.1.4.2. Medium Fidelity Prototyping

Pada tahapan ini dilakukan pembuatan purwarupa yang didasarkan pada purwarupa serta hasil pengujian yang didapat dari tahapan sebelumnya pada iterasi terakhir. Purwarupa yang ada dibuat dengan menggunakan *software* Balsamiq dalam mode *sketch* dengan metode *iterative parallel prototyping*, dimana nantinya akan dihasilkan 2 buah purwarupa.

Luaran dari tahapan ini berupa purwarupa antarmuka pengguna. Untuk tahapan ini, dilakukan iterasi berdasarkan hasil pengujian desain UI, dimana jika hasil pengujian dirasa kurang akan dilakukan perbaikan *prototipe*.

3.1.4.3. High Fidelity Prototyping

Pada tahapan ini dilakukan pembuatan purwarupa yang didasarkan pada purwarupa serta hasil pengujian yang didapat dari tahapan sebelumnya pada iterasi terakhir. Purwarupa yang ada dibuat dengan menggunakan *software* Balsamiq dalam mode *wireframe* dengan metode *iterative parallel prototyping*, dimana nantinya akan dihasilkan 1 buah purwarupa.

Luaran dari tahapan ini berupa purwarupa antarmuka pengguna. Untuk tahapan ini, dilakukan iterasi berdasarkan hasil pengujian desain UI, dimana jika hasil pengujian dirasa kurang akan dilakukan perbaikan *prototipe*.

3.1.5.Pengujian Desain User Interface

Tahapan ini berjalan berkesinambungan dengan tahapan pembuatan desain UI, dimana hasil dari pengujian desain *user interface* dijadikan sebagai pertimbangan apakah perlu dilakukan iterasi dalam tahapan pembuatan desain UI. Pengujian desain *user interface* dilakukan dengan menggunakan 2 metode, yaitu *usability study* dan h*euristic evaluation*.

3.1.5.1. Usability Study

Tahapan ini dilakukan untuk menguji prototipe yang dihasilkan pada tahapan *low fidelity prototyping* dan *medium fidelity prototyping*.

Tahapan ini dilakukan dengan cara melakukan pengujian berupa *usability study* dimana sampel dari TU, dosen, dan mahasiswa dipersilakan untuk mencoba purwarupa yang dihasilkan dan memberikan *feedback* terkait purwarupa yang mereka coba.

Luaran yang dihasilkan berupa laporan *usability study*.

3.1.5.2. Heuristic Evaluation

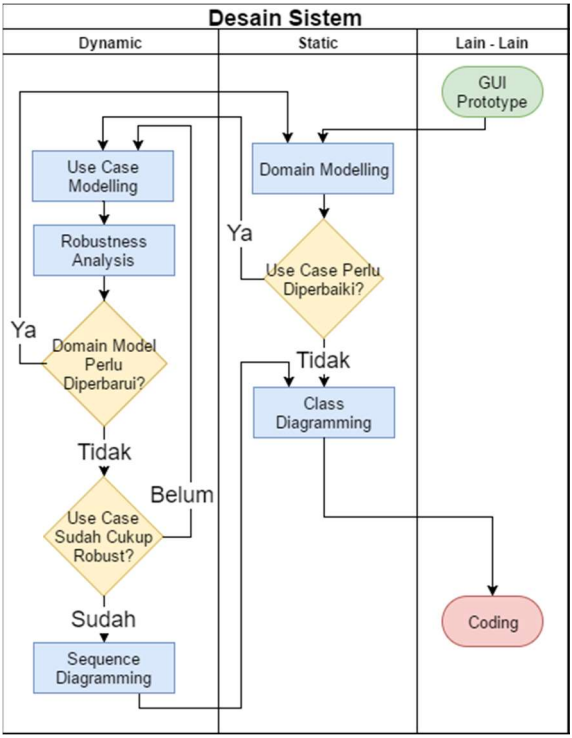
Tahapan ini dilakukan untuk menguji prototipe yang dihasilkan pada tahapan *medium fidelity prototyping* dan *high fidelity prototyping*.

Tahapan ini dilakukan dengan cara memeriksa purwarupa yang ada dengan menggunakan heuristik yang diajukan oleh Rosa Yanez Gomez et al.

Luaran yang dihasilkan berupa laporan *heuristic evaluation*.

3.1.6. Pembuatan Desain Sistem.

Tahapan ini adalah bagian dari ICONIX Process selain bagian GUI *storyboarding*. Pada tahapan ini dibuat beberapa diagram yang digunakan untuk 'menerjemahkan' purwarupa tampilan antar muka sehingga bisa lebih mudah untuk dijadikan kode program. Meskipun tahapan di ICONIX Process berupa tahapan *iterative*, namun pada tugas akhir ini diterjemahkan menjadi gambar berikut agar lebih mudah dipahami.



Gambar 3-3 Tahapan Desain Sistem

3.1.6.1. Domain Modelling

Pembuatan *domain model* dilakukan dengan cara melakukan ekstraksi dari *noun phrase* yang didapat dari kebutuhan fungsional, *task*, *goal*, dan beberapa referensi terkait dengan aplikasi yang sedang dikembangkan. *noun phrase* yang didapat lantas diseleksi dan dijadikan *class* dalam *domain model* yang dihasilkan. *Domain model* yang dihasilkan masih berupa *class* yang tidak memiliki atribut dan operasi serta *relationship* antar *class* masih bertipe *generalization* atau *aggregation*.

3.1.6.2. Use Case Modelling

Seperti namanya, tahapan ini digunakan untuk menghasilkan *use case*, namun *use case* yang dihasilkan tidak terlalu detail. Luaran yang dihasilkan berupa *use case model* dimana beberapa class yang ada di *domain model* digunakan dalam *use case* dan juga *use case scenario*.

3.1.6.3. Robustness Analysis

Robustness analysis dilakukan untuk memeriksa kekurangan dari *use case* dan *domain model* yang dihasilkan. Pada tahapan *robustness analysis*, digunakan *robustness diagram* untuk menganalisa setiap *use case* yang ada.

3.1.6.4. Sequence Diagramming

Sesuai dengan namanya, tahapan ini menghasilkan *sequence diagram*. Tahapan ini penting dilakukan untuk menghubungkan antara *domain model* yang masih berisi data dan entitas namun belum ada penjelasan bagaimana entitas yang ada saling bertukar data sesuai dengan skenario *use case*.

3.1.6.5. Class Diagramming

Class diagram dibuat dengan cara memperbaiki *domain model* yang dihasilkan pada tahapan sebelumnya dan ditambahkan komponen operasi untuk *class* yang ada.

3.1.7. Validasi Desain Sistem

Pada tahapan ini, dilakukan validasi desain sistem yang telah dihasilkan berdasarkan kebutuhan dengan menggunakan *traceability matrix*.

Luaran dari tahapan ini berupa *requirement traceability matrix*.

3.1.8. Penyusunan Laporan Tugas Akhir

Tahapan yang terakhir adalah penyusunan buku tugas akhir. Seluruh rangkaian dan hasil penelitian yang dilakukan pada tahapan – tahapan sebelumnya digunakan sebagai input pada tahapan penyusunan laporan tugas akhir.

Luaran dari tahapan ini adalah buku tugas akhir yang telah disusun sesuai dengan standard.

Halaman ini sengaja dikosongkan

BAB IV
PERANCANGAN

Pada Bab ini dijelaskan proses analisa dan pembuatan desain aplikasi pencatatan kehadiran berbasis smartphone android. Sesuai yang telah ditunjukkan pada metodologi, ada beberapa tahapan pengerjaan tugas akhir yang dilakukan dimana tahapan pengumpulan kebutuhan awal, pembuatan *storyboard*, pembuatan desain UI, pengujian desain UI, pembuatan desain sistem dan validasi desain sistem masuk ke dalam tahap perancangan.

4.1. Pengumpulan Kebutuhan Awal

Pada tahapan ini dilakukan 2 hal yang dilakukan, yaitu : observasi dam wawancara untuk memahami alur pencatatan kehadiran yang ada di JSI-ITS, lalu disusun kebutuhan fungsional awal yang digunakan sebagai dasar dari pembuatan *storyboard* dan UI.

Observasi dilakukan dengan cara mengamati alur pencatatan kehadiran dari sudut pandang mahasiswa, mulai dari pengambilan kertas pencatatan kehadiran dari pegawai TU bagian pencatatan kehadiran, proses pencatatan kehadiran di dalam kelas, hingga kertas pencatatan kehadiran kembali lagi ke pegawai TU bagian pencatatan kehadiran.

Wawancara dilakukan kepada kepala program studi untuk memahami proses pencatatan kehadiran dari sudut pandang manajemen dan informasi lain serta 3 aktor utama, yaitu : mahasiswa, dosen, dan pegawai TU bagian pencatatan kehadiran, untuk memahami apa yang mereka lakukan pada proses pencatatan kehadiran yang ada.

Berikut ini daftar pertanyaan yang diajukan kepada kepala program studi :

1. Berdasarkan pengamatan saya, ada 3 pihak yang terlibat secara langsung dalam proses pencatatan kehadiran di jurusan sistem informasi ITS, yaitu :

mahasiswa, dosen, dan TU. Sepengetahuan bapak, apakah ada pihak lain yang juga terlibat langsung dalam proses pencatatan kehadiran di JSI – ITS?
2. Apakah peran dosen, mahasiswa, dan TU dalam pencatatan kehadiran di JSI – ITS?
3. Apakah urgensi dari presensi mahasiswa?
4. Apa dampak jika pencatatan kehadiran mahasiswa ditiadakan?
5. Adakah format tertentu dari ITS terkait pencatatan kehadiran? Misal, informasi apa saja yang harus ada di kertas pencatatan kehadiran?
6. Ketika tingkat kehadiran mahasiswa tidak sampai 80%, ada kebijakan bahwa mahasiswa tersebut secara otomatis mendapat nilai E. untuk kebijakan tersebut, apakah merupakan kebijakan dari ITS atau kebijakan dari JSI?
7. Untuk kertas presensi, kenapa di ruang kelas JSI – ITS umumnya ada 2 macam kertas presensi, 1 presensi untuk dosen dan 1 presensi untuk mahasiswa?

Sementara itu, untuk Mahasiswa, berikut ini daftar pertanyaan yang diajukan :

1. Berdasarkan pemahamanmu, jelaskan alur pencatatan kehadiran secara keseluruhan, mulai dari TU hingga kembali ke TU?
2. Apa saja yang kamu lakukan setelah kamu mendapat kertas pencatatan kehadiran dari mahasiswa lain / dosen?
3. Apa saja informasi yang kamu butuhkan dari kertas presensi?
4. Pernahkan kamu mengalami kejadian yangtidak termasuk dalam *normal case scenario* dalam proses pencatatan kehadiran? (misal: nama tidak ada)
5. Adakah hal lain yang kamu lakukan dengan kertas presensi selain melakukan pencatatan kehadiran? (misal: mencari NRP teman satu kelompok)

Sementara itu, untuk Dosen, berikut ini daftar pertanyaan yang diajukan :

1. Berdasarkan pemahaman anda, apa saja kegiatan yang dilakukan dalam proses presensi / pencatatan kehadiran?
2. Apa saja yang anda lakukan, terkait dengan proses presensi, setelah mendapat kertas presensi dari TU?
3. Apa saja yang anda lakukan, terkait dengan proses presensi, sebelum anda memberikan kertas presensi kepada mahasiswa?
4. Apa saja yang anda lakukan, terkait dengan proses presensi, setelah mendapat kertas presensi dari mahasiswa terakhir?
5. Apa saja informasi yang anda butuhkan dari kertas presensi?
6. Pernahkah anda mengalami kejadian yang umumnya jarang dijumpai dalam proses presensi?
7. Adakah hal lain yang anda lakukan dengan kertas presensi selain melakukan presensi?

Sementara itu, untuk pegawai TU bagian pencatatan kehadiran, berikut ini daftar pertanyaan yang diajukan :

1. Berdasarkan pemahaman anda, apa saja kegiatan yang dilakukan dalam proses presensi?
2. Apa saja yang anda lakukan, terkait dengan proses presensi, sebelum memberikan kertas presensi ke dosen?
3. Apa saja yang anda lakukan, terkait dengan proses presensi, setelah mendapat kertas presensi dari dosen?
4. Apa saja Informasi yang anda butuhkan dari kertas presensi?
5. Pernahkah anda mengalami kejadian yang umumnya jarang dijumpai dalam proses presensi?
6. Adakah hal lain yang anda lakukan dengan kertas presensi selain melakukan presensi?

Untuk hasil wawancara yang dilakukan, terlampir file hasil rekaman, dimana untuk narasumber dari dosen tidak dilakukan wawancara secara langsung dengan pertimbangan bahwa dosen hampir tidak memiliki peran dalam kegiatan pencatatan kehadiran, namun pada tahapan *usability study* nantinya diajukan beberapa pertanyaan terkait peran dosen kepada responden yang dilibatkan.

4.1.1.Alur Pencatatan Kehadiran

Berdasarkan hasil observasi, alur presensi / pencatatan kehadiran yang ada di jurusan sistem informasi melibatkan 3 kelompok pengguna utama, yaitu : pegawai TU bagian presensi (disingkat TU), dosen pengampu mata kuliah (disingkat Dosen), dan mahasiswa yang terdaftar dalam mata kuliah tersebut (disingkat Mahasiswa).

Presensi dimulai dari TU yang memasukan template presensi ke dalam sistem. Pada proses bisnis yang terjadi di JSI – ITS, umumnya TU mengambil data *template* presensi dari Integra ITS lalu menambahkan beberapa data tambahan sesuai kebutuhan yang selanjutnya dimasukkan ke dalam *database* di JSI ITS. Setelah proses itu selesai, TU akan mencetak template presensi menjadi kertas Presensi.

Sebelum kegiatan perkuliahan sebuah mata kuliah dimulai, Dosen akan meminta kertas presensi kepada TU. TU melakukan verifikasi untuk memastikan bahwa Dosen yang meminta kertas presensi tersebut adalah dosen pengampu mata kuliah, jika iya maka TU akan menyerahkan kertas presensi ke Dosen. Jika tidak maka TU akan mencari kertas presensi yang sesuai dengan Dosen. Dosen selanjutnya membawa kertas presensi ke kelas, lalu memberikan ke Mahasiswa.

Mahasiswa yang menerima kertas presensi selanjutnya memeriksa apakah NRPnya ada di kertas presensi, jika tidak ada maka Mahasiswa akan lapor ke TU dan akan ditindaklanjuti oleh TU dengan tindakan menambah data mahasiswa. Jika NRP Mahasiswa ada di kertas presensi, maka Mahasiswa tersebut

akan melakukan presensi. Setelah melakukan presensi, maka Mahasiswa tersebut memberikan kertas presensi ke Mahasiswa lainnya.

Setelah semua Mahasiswa sudah melakukan presensi, maka kertas presensi akan diserahkan kembali ke Dosen. Dosen bisa melakukan headcount untuk memastikan bahwa data di kertas presensi sudah cocok dengan mahasiswa yang ada di kelas. Headcount umumnya dilakukan dengan cara memanggil nama mahasiswa satu persatu. Setelah itu kertas presensi diberikan kembali ke TU dimana TU akan menyimpan data yang di dapat dari kertas presensi ke dalam database di JSI ITS.

Untuk Swimlane Diagram alur pencatatan kehadiran JSI-ITS bisa dilihat pada Lampiran B.

4.1.2.Kebutuhan Fungsional Awal

Fungsi dari kebutuhan awal adalah membantu pembentukan skenario yang diujikan pada *usability study* sehingga bersifat fleksibel, bisa berubah berdasarkan proses desain yang ada.

Tabel 4-1 Daftar Kebutuhan Fungsional Awal

Nomor	Kebutuhan Fungsional Awal
TU001	TU bisa melakukan transfer template presensi dari server ke android dan sebaliknya
TU002	TU bisa menambahkan data tambahan untuk template presensi
TU003	TU bisa melihat informasi kelas untuk mencocokan data dosen pengampu dengan dosen yang meminta template presensi
TU004	TU bisa menambahkan data mahasiswa baru
MHS001	Mahasiswa bisa memeriksa apakah NRPnya ada atau tidak
MHS002	Mahasiswa bisa melakukan presensi / pencatatan kehadiran
DSN001	Dosen bisa memeriksa data presensi mahasiswa
DSN002	Dosen bisa melakukan headcount

4.2. Pembuatan Storyboard

Pembuatan *storyboard* dimaksudkan untuk merancang bagaimana nantinya aplikasi akan digunakan.

Storyboard yang di hasilkan ada 2 versi, versi pertama menunjukan bagaimana alur pencatatan kehadiran secara garis besar jika aplikasi digunakan di JSI – ITS, sementara versi kedua menunjukkan bagaimana aplikasi digunakan pada saat kegiatan perkuliahan secara lebih rinci jika dibandingkan dengan versi pertama.

Hasil pembuatan *storyboard* bisa dilihat pada Lampiran C.

4.3. Pembuatan & Pengujian Desain User Interface

Tahapan pembuatan desain *user interface* dan pengujian desain *user interface* berjalan secara rekursif, dimana hasil pengujian desain *user interface* akan dijadikan sebagai dasar dari pembuatan desain *user interface* pada iterasi / tingkatan selanjutnya.

Tahapan ini menitik beratkan pada pembuatan desain *user interface* yang cocok untuk pengguna. Pengembangan prototipe dilakukan dalam 3 tahapan, yaitu *low fidelity prototyping*, *medium fidelity prototyping*, dan *high fidelity prototyping*. Lalu untuk menguji desain prototipe UI yang ada, digunakan *usability study* dan *heuristic evaluation*.

4.3.1. Usability Study

Usability study dilakukan secara bertahap, yaitu pada LFP dan MFP. Dimana pada LFP, *usability study* yang dilakukan bersifat non-formal, lebih ke mengarah ke pengumpulan kebutuhan perangkat lunak.

Sementara untuk *usability study* pada MFP, pengujian dilakukan secara semi-formal dimana juga digunakan kuosioner PSSUQ yang diajukan.

Selama proses *usability study* dilakukan, untuk setiap prototipe yang diujikan kepada responden, jika ada kesalahan prototipe /

responden kesulitan dalam memahami *interface*, maka penguji mengambil alih sejenak dan mengarahkan responden ke halaman yang dituju.

Selain itu, penguji juga mencatat *feedback* dari responden selama pengujian berlangsung, baik berupa masukan lisan, maupun matriks usabilitas, yaitu : *completion time* dan *success rate* untuk *low fidelity prototypying*, serta ditambahkan matriks *error rate* untuk *medium fidelity prototyping*.

Berikut ini adalah prosedur yang dilakukan dalam tahapan *usability study* pada tugas akhir ini:

1. Penguji menjelaskan secara umum apa itu *usability study*, metode *iterative parallel prototyping*, *paper prototyping* (untuk tahapan *low fidelity prototyping*), w*ireframe* / *mock-up* (untuk tahapan *middle fidelity prototyping*) dan sebagainya.
2. Penguji menjelaskan *task* dan obyektif dalam pengujian.
3. Penguji mempersilakan responden untuk bertanya.
4. Secara acak, penguji akan memberikan prototipe A atau prototipe B, ataupun prototipe C (jika ada) kepada responden
5. Penguji mempersilakan responden untuk melakukan *usability study* untuk prototipe pertama
6. Responden mengisi kuosioner untuk prototipe pertama
7. Penguji mempersilakan responden untuk memberi *feedback* terkait prototipe pertama
8. Penguji mempersilakan responden untuk melakukan *usability study* untuk prototipe kedua
9. Responden mengisi kuosioner untuk prototipe kedua
10. Penguji mempersilakan Responden untuk memberi *feedback* terkait prototipe kedua
11. Penguji mempersilakan responden untuk melakukan *usability study* untuk prototipe ketiga (jika ada)
12. Responden mengisi kuosioner untuk prototipe ketiga (jika ada)

13. Penguji mempersilakan Responden untuk memberi *feedback* terkait prototipe ketiga (jika ada)
14. Penguji memberikan Reward kepada responden (Ucapan terima kasih, bingkisan, dsb.)

Setelah *usability study* selesai dilakukan, penguji melakukan proses rekapitulasi kuosioner yang sudah diisi oleh pengguna. Ada 2 macam kuosioner yang digunakan pada *usability study*, yaitu : kuosioner *feedback* tambahan yang digunakan pada tahapan *low fidelity prototyping* dan kuosioner PSSUQ yang digunakan pada tahapan *medium fidelity prototyping*. Dimana kuosioner PSSUQ digunakan untuk mengukur matriks usability *user satisfaction*.

Data yang didapat dari hasil rekap matriks *usability*, yaitu : *success rate*, *completion time*, *error rate*, dan *user satisfaction*, merupakan data kuantitatif yang menunjukkan nilai mentah / *rough value* dari prototipe UI yang ada. Nilai tersebut bisa menunjukkan prototipe mana yang lebih unggul namun belum bisa menjelaskan kenapa prototipe UI yang satu lebih unggul dibandingkan prototipe UI lainnya.

Sementara itu, data yang didapat dari *feedback* lisan responden dan kuosioner *feedback* tambahan berupa data kualitatif, dimana data tersebut digunakan untuk mendeskripsikan data kuantitatif yang didapat untuk prototipe UI yang ada.

Syarat dari prototipe UI yang dihasilkan dianggap layak, sehingga proses pengerjaan bisa dilanjutkan ke tahapan berikutnya, adalah :

1. Prototipe UI diujikan kepada semua kelompok pengguna yang terlibat langsung dalam proses pencatatan kehadiran (mahasiswa, dosen, dan pegawai TU bagian presensi / pencatatan kehadiran).
2. Untuk setiap *task* yang diajukan, ada prototipe yang memiliki *success rate* 100% untuk memastikan bahwa prototipe itu bisa digunakan pengguna dalam penyelesaian *task* mereka.

4.3.2. Heuristic Evaluation

Heuristic evaluation dilakukan dengan cara mencoba prototipe UI yang ada sambil membandingkannya dengan heuristik yang ada, untuk memastikan bahwa ada atau tidaknya *heuristic violation* dari prototipe UI tersebut.

Pada tugas akhir ini, daftar heuristik yang digunakan adalah daftar heuristik yang diajukan oleh Rosa Yanez Gometz et al pada jurnal yang ia tulis dengan judul "*Heurisitic Evaluation on Mobile Interface: A New Checklist*". Daftar subheuristik / heuristik bisa dilihat pada lampiran. Sementara itu, *evaluator* pada tugas akhir ini adalah penulis.

Pengujian ini dilakukan untuk prototipe yang dihasilkan pada tahapan *medium fidelity prototyping* serta *high fidelity prototyping*, dimana prosedur pengujian yang dilakukan tidak jauh berbeda. Berikut ini prosedur pengujian yang dilakukan :

1. Penguji membuka prototipe pertama dan mencoba semua interaksi yang ada.
2. Jika ditemukan adanya komponen UI yang tidak sesuai dengan heuristik / *heuristic violation* pada prototipe pertama, maka dilakukan pencatatan.
3. Penguji membuka prototipe kedua dan mencoba semua interaksi yang ada (jika ada prototipe kedua)
4. Jika ditemukan adanya komponen UI yang tidak sesuai dengan heuristik / *heuristic violation* pada prototipe kedua, maka dilakukan pencatatan. (jika ada prototipe kedua)
5. Setelah semua interaksi pada semua prototipe selesai dicoba dan dibandingkan dengan setiap subheuristik yang ada, dilakukan pemberian peringkat *severity rating* untuk melihat *heuristic violation* mana yang lebih penting untuk diperbaiki lebih dahulu.

Pemberian *severity rating* dilakukan secara subjektif, dengan kata lain penguji *heuristic evaluation* memiliki kebebasan untuk menentukan *severity rating* terhadap semua *heuristic violation*

yang ia temukan. Agar hasil *heuristic evaluation* lebih mudah dipahami, digunakan *severity rating* yang diajukan oleh Jakob Nielsen.

Untuk melihat apakah prototipe sudah cukup layak, bisa dilihat dari *heuristic violation* yang ada. Jika ada *heuristic violation* dengan *severity rating* 3 maupun *severity rating* 4, maka prototipe itu bisa dikatakan masih belum layak.

Untuk tahapan *high fidelity prototyping* dimana hasil luaran dianggap sebagai versi akhir, maka prototipe yang ada seharusnya memang tidak memiliki *heuristic violation* dengan *severity rating* 3 maupun *severity rating* 4.

Sementara itu, untuk tahapan *medium fidelity prototyping*, karena masih menggunakan pengembangan paralel, ada sedikit penyesuaian. Prototipe yang ada dianggap layak apabila *heuristic violation* dengan *severity rating* 3 tidak lebih dari 5 *heuristic violation*, namun tidak boleh ada *heuristic violation* dengan *severity rating* 4.

4.3.3.Low Fidelity Prototyping

Pada tahapan ini dilakukan pembuatan antar-muka pengguna dengan metode *iterative parallel prototyping* dimana prototipe dihasilkan dengan metode *paper prototyping*. Prototipe yang dihasilkan pada tahapan ini ada 3 buah untuk setiap iterasi yang dilakukan.

Pengujian prototipe dilakukan dengan menggunakan metode *usability study*.

Proses iterasi pada tahapan ini dilakukan apabila tidak ada prototipe yang berhasil mencapai *success rate* 100% untuk 3 kelompok pengguna yang dilibatkan (mahasiswa, dosen, dan TU).

Untuk luaran berupa prototipe UI, bisa dilihat pada lampiran.

4.3.4. Medium Fidelity Prototyping

Pada tahapan ini dilakukan pembuatan antar-muka pengguna dengan metode *iterative parallel prototyping* dimana prototipe dihasilkan dengan aplikasi Balsamiq *mode sketch*. Prototipe yang dihasilkan pada tahapan ini ada 2 buah untuk setiap iterasi yang dilakukan.

Pengujian prototipe dilakukan dengan menggunakan metode *usability study* dan *heuristic evaluation*. Proses iterasi pada tahapan ini dilakukan apabila keadaan berikut tidak terpenuhi :

- ada prototipe yang berhasil mencapai *success rate* 100% untuk 3 kelompok pengguna yang dilibatkan (mahasiswa, dosen, dan TU)
- ada prototipe yang berhasil mendapat *avg. user satisfaction* di bawah 4 untuk setiap kelompok pengguna yang dilibatkan (mahasiswa, dosen, dan TU)
- ada prototipe yang mendapatkan hasil yang baik dalam *heuristic evaluation* (*heuristic violation* dengan *severity rating* 4 = 0 dan *severity rating* 3 < 5)

Untuk luaran berupa prototipe UI, bisa dilihat pada lampiran.

4.3.5. High Fidelity Prototyping

Pada tahapan ini dilakukan pembuatan prototipe antar-muka pengguna dengan menggunakan aplikasi Balsamiq *mode wireframe*. Prototipe yang dihasilkan pada tahapan ini ada 1 buah untuk setiap iterasi yang dilakukan.

Pengujian prototipe dilakukan dengan menggunakan metode *heuristic evaluation* dimana proses iterasi dilakukan apabila prototipe yang dihasilkan masih memiliki *heuristic violation* dengan *severity rating* 3.

Untuk luaran berupa prototipe UI, bisa dilihat pada lampiran.

4.4. Pembuatan Desain Sistem

ICONIX Process menggunakan pendekatan minimalis untuk memproduksi beberapa diagram yang dibutuhkan dalam tahap perancangan program. Diagram – diagram tersebut antara lain : *use case diagram*, *domain model*, *sequence diagram*, dan *robustness diagram*. Jika dirasa kurang, ICONIX memperbolehkan pengembang perangkat lunak untuk menambahkan diagram lain. Namun pada tugas akhir ini, peneliti merasa bahwa diagram - diagram yang diajukan sudah cukup untuk membantu proses perancangan. Selain itu, peneliti melakukan beberapa penyesuaian terhadap tahapan ICONIX agar lebih mudah untuk dipahami. Berikut ini adalah tahapan – tahapan yang dilakukan terkait dengan Desain Sistem.

4.4.1. Kebutuhan Fungsional

Kebutuhan fungsional yang dicantumkan didapat dari pembaruan Tabel 4-1 Daftar Kebutuhan Fungsional Awal serta masukan pada tahapan desain prototipe UI. Fungsi dari kebutuhan fungsional pada desain sistem adalah untuk membantu mendapatkan *domain model* tahap awal dengan melakukan seleksi *noun phrase* yang ada di kebutuhan fungsional.

Tabel 4-2 Daftar Kebutuhan Fungsional

Nomor	Kebutuhan Fungsional
FR-101	Mahasiswa bisa melakukan presensi / pencatatan kehadiran
FR-102	Mahasiswa bisa melihat data ketidakhadirannya
FR-103	Mahasiswa bisa melihat daftar mahasiswa yang satu kelas mata kuliah dengan dirinya
FR-104	Mahasiswa bisa melihat informasi kelas mata kuliah yang sedang diikuti
FR-201	Dosen bisa melihat daftar mahasiswa pada kelas mata kuliah yang sedang ia ajar
FR-202	Dosen bisa mengubah status kehadiran mahasiswa

Nomor	Kebutuhan Fungsional
FR-203	Dosen bisa memilih kelas mana yang akan dilakukan proses pencatatan kehadiran berdasarkan daftar kelas yang berisi mata kuliah yang ia ajar
FR-204	Dosen bisa melihat informasi kelas mata kuliah yang sedang ia ajar
FR-301	Pegawai TU bisa memasukkan data mahasiswa ke dalam kelas mata kuliah yang belum diambilnya
FR-302	Pegawai TU bisa menghapus data mahasiswa dari dalam kelas mata kuliah yang sedang diambilnya
FR-303	Pegawai TU bisa memindahkan mahasiswa dari kelas mata kuliah yang diambil oleh mahasiswa ke kelas lain
FR-304	Pegawai TU bisa mengubah status kehadiran mahasiswa
FR-305	Pegawai TU bisa menambahkan data mahasiswa baru
FR-306	Pegawai TU bisa menghapus data mahasiswa
FR-307	Pegawai TU bisa menambahkan kelas mata kuliah baru
FR-308	Pegawai TU bisa memilih direktori pada handset android yang digunakan untuk memuat template presensi

4.4.2. Seleksi Noun Phrase

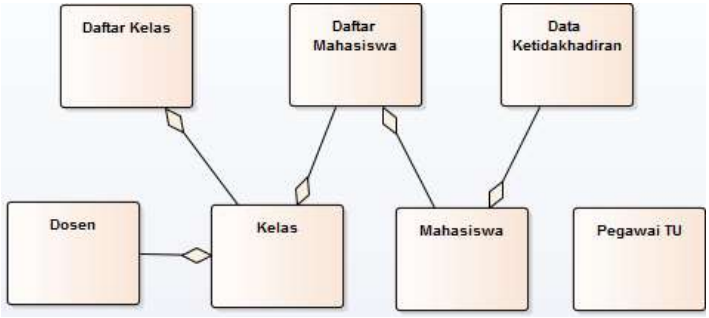
Setelah dibuat daftar kebutuhan fungsional, maka dilakukan seleksi *noun phrase*. Berikut ini tabel yang menunjukan hasil seleksi *noun phrase* yang telah diurutkan berdasar abjad :

Tabel 4-3 Daftar Noun Phrase

No	Noun Phrase
1.	Daftar Kelas
2.	Daftar Mahasiswa
3.	Data Ketidakhadiran
4.	Data Mahasiswa
5.	Dosen
6.	Informasi Kelas
7.	Kelas
8.	Mahasiswa
9.	Mata Kuliah
10.	Pegawai TU
11.	Status Kehadiran Mahasiswa

4.4.3. Domain Model

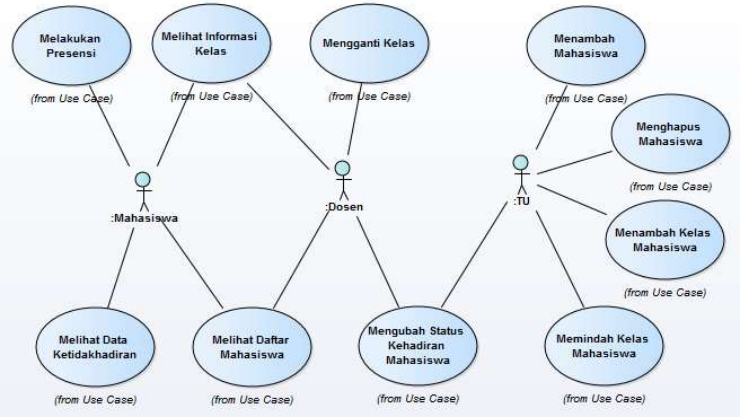
Dari *noun phrase* yang telah didapat pada tahapan sebelumnya, dilakukan proses seleksi untuk memastikan apakah *noun phrase* tersebut termasuk objek atau bukan. Apabila *noun phrase* tersebut termasuk objek maka dimasukkan ke dalam *domain model*. Berikut ini *domain model* yang dibuat berdasarkan proses tesebut :



Gambar 4-1 Domain Model

4.4.4. Use Case Modelling

Pada tahapan ini, *use case* yang dicantumkan pada *use case model* belum bersifat final, pada proses *robustness analysis* bisa saja ditemukan *use case* baru ataupun ada *use case* yang dihapus. *Use case model* yang dihasilkan hanya sebagai panduan dalam tahapan berikutnya.



Gambar 4-2 Use Case Model

4.4.5. Robustness Analysis

Pada *robustness analysis* dilakukan analisa *use case* berdasarkan *robustness diagram*, berikut ini hasil dari *robustness analysis* yang dilakukan untuk setiap *use case*.

Pada tahapan ini ada beberapa *use case* yang ditemukan, dihapus, ataupun diperbaiki. Untuk mempermudah proses dokumentasi, hanya *use case* versi akhir yang menggunakan awalan UC, sementara *use case* dengan awalan DUC adalah *use case* yang dihapus karena alasan tertentu.

Tahapan iterasi dilakukan dalam robustness analysis secara bergantian untuk tiap *use case* hingga dirasa *use case* tersebut cukup *robust* / tidak ambigu.

Rincian *robustness analysis* bisa dilihat pada Lampiran D.

4.4.5.1. UC-101 Mahasiswa Membuka Halaman Informasi Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada tahapan pada tahapan *robustness analysis*, asal dari *use case* ini adalah pelebaran dari *use case* UC-102, UC-103, dan UC-104 yang membutuhkan pengguna mahasiswa untuk melakukan "*login*" terlebih dahulu. Selain itu, *use case* ini juga merupakan gabungan dari DUC – 101 yang sudah dihapus dari daftar *use case*. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Penggunaan *controller* "tampilkan halaman" dirasa perlu ditambahkan sehingga dilakukan iterasi ini.
- Penggunaan "Data Mahasiswa" sebagai *entity* dirasa lebih sesuai dengan *use case description* dibandingkan "Mahasiswa".

Hal yang ditemukan ketika dilakukan iterasi 2:

- Sesuai dengan aturan *robustness diagram*, dimana *entity* tidak boleh terhubung langsung dengan *boundary* maka ditambahkan *controller* untuk menghubungkan beberapa *entity* dan *boundary* yang terhubung secara langsung pada iterasi sebelumnya.
- Penggunaan "Mahasiswa" sebagai *entity* dirasa lebih sesuai dengan karakteristik *entity* dibandingkan "Data Mahasiswa" sehingga dikembalikan seperti iterasi sebelumnya.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Perbaikan prototipe *user interface* menyebabkan beberapa elemen yang ada di *robustness diagram* menjadi kurang tepat untuk mewakili alur yang terjadi di prototipe *user interface* yang baru sehingga dilakukan penambahan beberapa *entity* dan juga *alternate scenario* agar sesuai dengan prototipe *user interface* yang baru.

Hal yang ditemukan ketika dilakukan iterasi 4:

- *Use case desription* dirasa kurang tepat sehingga dilakukan revisi *use case description*.
- Ada beberapa *entity* yang belum terwakili pada iterasi sebelumnya sehingga ditambahkan pada iterasi ini.
- Hubungan antar beberapa elemen ada yang dirasai kurang tepat sehingga diperbaiki pada iterasi ini.

4.4.5.2. DUC-101 Mahasiswa Melihat Informasi Kelas

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 3 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Perubahan *use case description* yang terjadi karena adanya pemisahan *use case* menjadi *use case* baru, yaitu UC-101, menyebabkan elemen *robustness* yang ada harus disesuaikan ulang.
- Ada *domain* pada *domain model* yang belum dimasukkan menjadi *entity* pada *robustness diagram* yang ada.
- *Alternate scenario* yang ada tidak fokus ke interaksi aplikasi / sistem ke pengguna sehingga dihapus.

Hal yang ditemukan ketika dilakukan iterasi 2:

- *Entity* data mahasiswa dirasa tidak perlu dicantumkan pada *use case* ini karena tidak berhubungan secara langsung dan sudah diwakili oleh *use case* UC-101 sehingga dihapus.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Dengan pertimbangan bahwa info kelas sudah ditampilkan pada UC-101, maka *use case* ini dihapus dan digabung dengan UC-101.

4.4.5.3. UC-102 Mahasiswa Melakukan Presensi

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 7 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.
- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Adanya penambahan UC-101 menyebabkan alur *use case* sedikit berubah, sehingga dilakukan penyesuaian.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Alur yang ada dirasa masih menyederhanakan *use case* dan berpotensi menyebabkan *use case* menjadi ambigu, sehingga ditambahkan beberapa elemen *robustness diagram* lainnya agar lebih representatif.

Hal yang ditemukan ketika dilakukan iterasi 4:

- Mengarahkan pengguna secara langsung ke 'Halaman Input NRP dirasa janggal karena pada prototipe UI yang ada, pengguna harus melewati 'Halaman Awal' terlebih dahulu.
- Pemunculan *toast* pada *alternate scenario* dinilai kurang cocok sehingga dilakukan perbaikan alur *alternate scenario*.
- Alur yang ada masih dirasa ambigu sehingga dilakukan perbaikan beberapa elemen *robustness diagram* yang ada.

Hal yang ditemukan ketika dilakukan iterasi 5:

- Ada *alternate scenario* yang dirasa perlu ditambahkan, yaitu ketika NRP belum diisi.
- Alur yang ada dirasa masih kurang representatif, sehingga dilakukan beberapa perbaikan.

Hal yang ditemukan ketika dilakukan iterasi 6:

- Dengan munculnya UC-101, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan *use case* yang di-*invoke*.

Hal yang ditemukan ketika dilakukan iterasi 7:

- Perbaikan kecil untuk menyesuaikan alur yang ada.

4.4.5.4. UC-103 Mahasiswa Melihat Data Ketidakhadiran

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Dengan munculnya UC-101, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan *use case* yang di-*invoke*.
- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Sesuai dengan aturan *robustness diagram*, dimana *entity* tidak boleh terhubung langsung dengan *boundary* maka ditambahkan *controller* untuk menghubungkan beberapa *entity* dan *boundary* yang terhubung secara langsung pada iterasi sebelumnya.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Ada *alternate scenario* opsional yang dirasa perlu ditambahkan, yaitu ketika mahasiswa mencari data ketidakhadiran.

Hal yang ditemukan ketika dilakukan iterasi 4:

- Fungsi *search* yang diajukan pada prototipe *user interface* tidak terlalu rumit sehingga *entity* "Hasil Pencarian Data Ketidakhadiran" tidak perlu dimasukkan ke dalam diagram yang ada. Selain itu juga dilakukan sedikit penyesuaian untuk sebagai dampak dari dihapusnya *entity* hasil ketidakhadiran.

4.4.5.5. UC-104 Mahasiswa Melihat Daftar Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Dengan munculnya UC-101, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan use case yang di-*invoke*.
- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.
- *Use case description* dirasa terlalu kurang rinci sehingga diperjelas lagi dalam iterasi ini.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Sesuai dengan aturan *robustness diagram*, dimana *entity* tidak boleh terhubung langsung dengan *boundary* maka ditambahkan *controller* untuk menghubungkan beberapa *entity* dan *boundary* yang terhubung secara langsung pada iterasi sebelumnya.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Ada *alternate scenario* opsional yang dirasa perlu ditambahkan, yaitu ketika mahasiswa mencari mahasiswa lain menggunakan *search box*.

Hal yang ditemukan ketika dilakukan iterasi 4:

- Fungsi *search* yang diajukan pada prototipe *user interface* tidak terlalu rumit sehingga *entity* "Hasil Pencarian Mahasiswa" tidak perlu dimasukkan ke dalam diagram yang ada. Selain itu juga dilakukan sedikit penyesuaian untuk sebagai dampak dari dihapusnya *entity* hasil ketidakhadiran.

4.4.5.6. UC-201 Dosen Membuka Halaman Presensi Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada tahapan pada tahapan *robustness analysis*, asal dari *use case* ini adalah pelebaran dari *use case* UC-202, UC-203, dan UC-204 yang membutuhkan pengguna mahasiswa untuk melakukan "*login*" terlebih dahulu. Selain itu, *use case* ini juga merupakan gabungan dari *use case* DUC – 201 yang sudah dihapus dari daftar *use case*. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 1 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Penggunaan "*Database*" sebagai *entity* dalam alur yang ada dirasa terlalu ambigu, sehingga diubah menjadi entity "Akun Pengguna" untuk menghilangkan ambiguitas.
- *Use case description* dirasa terlalu kurang rinci sehingga diperjelas lagi dalam iterasi ini dan dilakukan perbaikan diagram agar sesuai dengan *use case description*.

4.4.5.7. DUC-201 Dosen Melihat Daftar Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 1 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Dengan munculnya UC-201, dan pertimbangan bahwa daftar mahasiswa juga ditampilkan di UC-201, maka *use case i*ni dihapus dan digabung dengan UC-201.

4.4.5.8. UC-202 Dosen Mengubah Status Kehadiran Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 3 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Dengan munculnya UC-201, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan *use case* yang di invoke.
- *Use case description* dirasa terlalu kurang rinci sehingga diperjelas lagi dalam iterasi ini.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Sesuai dengan aturan *robustness diagram*, dimana *entity* tidak boleh terhubung langsung dengan *boundary* maka ditambahkan *controller* untuk menghubungkan beberapa *entity* dan *boundary* yang terhubung secara langsung pada iterasi sebelumnya.

Hal yang ditemukan ketika dilakukan iterasi 3:

- *Use case description* dirasa terlalu kurang rinci sehingga diperjelas lagi dalam iterasi ini.

4.4.5.9. UC-203 Dosen Melihat Informasi Kelas

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.
- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Dengan munculnya UC-201, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan *use case* yang dimasukan ke dalam diagram yang ada.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Tidak adanya penjelasan *relationship* macam apa dari UC-201 yang dihubungkan ke controller "Halaman Presensi Mahasiswa" berpotensi memunculkan ambiguitas, oleh karena itu *relationship* UC-201 terhadap *controller* "Halaman Presensi Mahasiswa" didefinisikan sebagai "*invoke*" agar sesuai dengan *use case description* yang ada.

Hal yang ditemukan ketika dilakukan iterasi 4:

- Sesuai dengan aturan *robustness diagram*, dimana *entity* tidak boleh terhubung langsung dengan *boundary* maka ditambahkan *controller* untuk menghubungkan beberapa *entity* dan *boundary* yang terhubung secara langsung pada iterasi sebelumnya.

4.4.5.10. UC-204 Dosen Mengganti Kelas

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 7 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Dengan munculnya UC-201, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan *use case* yang di-*invoke*.
- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Ada *alternate scenario* yang dirasa perlu ditambahkan, yaitu ketika dosen ingin mengganti kelas yang tidak ada pada "Halaman Ganti Kelas Dosen".
- Prototipe *user interface* belum mewadahi skenario ketika dosen ingin mengganti kelas yang tidak ada pada "Halaman Ganti Kelas Dosen" sehingga dilakukan perbaikan prototipe *user interface* untuk memfasilitasi hal tersebut serta dilakukan beberapa penyesuaian elemen *robustness diagram* yang ada.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Sesuai dengan aturan *robustness diagram*, dimana *entity* tidak boleh terhubung langsung dengan *boundary* maka ditambahkan *controller* untuk menghubungkan beberapa *entity* dan *boundary* yang terhubung secara langsung pada iterasi sebelumnya.

Hal yang ditemukan ketika dilakukan iterasi 4:

- Alur yang ada dirasa masih kurang representatif, sehingga dilakukan beberapa perbaikan.

Hal yang ditemukan ketika dilakukan iterasi 5:

- Ada *alternate scenario* yang dirasa perlu ditambahkan, yaitu ketika dosen ingin mencari kelas menggunakan *search box*.

- *Alternate scenario* "kelas tidak ada" dirasa kurang sesuai dengan prototipe *user interface* yang ada sehingga dilakukan beberapa penyesuaian.

Hal yang ditemukan ketika dilakukan iterasi 6:

- Dengan munculnya *use case* DUC-202, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan use case yang di-*invoke*.

Hal yang ditemukan ketika dilakukan iterasi 7:

- Dengan dihapusnya *use case* DUC-202, maka dilakukan beberapa penambahan dan penyesuaian untuk memfasilitasi *alternate scenario* opsional ketika dosen mencari data kelas.

4.4.5.11. DUC-202 Dosen Mencari Kelas

Use case ini merupakan salah satu *use case* yang ditemukan pada tahapan pada tahapan *robustness analysis*, asal dari *use case* ini adalah pelebaran dari *use case* UC-204 dimana ada *alternate scenario* opsional ketika dosen mencari kelas menggunakan *search box* yang ada. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 1 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Dengan pertimbangan bahwa fungsi *search* yang diajukan pada prototipe *user interface* tidak terlalu rumit maka keberadaan *use case* ini tidak diperlukan sehingga DUC-202 dihapus dan dilakukan penyesuaian untuk *use case* yang memanggil DUC-202.

4.4.5.12. UC-301 TU Membuka Halaman TU Awal

Use case ini merupakan salah satu *use case* yang ditemukan pada tahapan pada tahapan *robustness analysis*, asal dari *use case* ini adalah pelebaran dari *use case* UC-302, UC-308, UC309, dan UC-310 yang membutuhkan pengguna TU untuk melakukan "*login*" terlebih dahulu. Selain itu, UC-301 adalah

salah satu dari 2 *use case* yang merupakan pelebaran dari *use case* UC-303, UC-304, UC-305, UC-306, dan UC-307 yang membutuhkan pengguna TU untuk melakukan "*login*" + mengakses halaman mahasiswa terlebih dahulu (UC-302). Tidak ada iterasi yang dilakukan untuk *use case* ini karena UC-301 dirasa sudah cukup *robust*.

4.4.5.13. UC-302 TU Membuka Halaman Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada tahapan pada tahapan *robustness analysis*, asal dari *use case* ini adalah pelebaran dari *use case* UC-303, UC-304, UC-305, UC-306, dan UC-307 yang membutuhkan pengguna TU untuk melakukan "*login*" (UC-301) + mengakses halaman mahasiswa terlebih dahulu. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Hubungan antara *boundary* "Halaman Mahasiswa" dengan *entity* "Data Mahasiswa" dirasa masih kurang tepat sehingga ditambahkan *controller* "Muat Data Mahasiswa" yang menghubungkan antara *entity* "Data Mahasiswa" dengan *controller* "Tampilkan Halaman Mahasiswa".

Hal yang ditemukan ketika dilakukan iterasi 2:

- Ada 2 *alternate scenario* yang dirasa perlu ditambahkan, yaitu ketika pengguna TU ingin mencari data mahasiswa melalui *search box* dan ketika pengguna TU ingin mengakses "Halaman Mahasiswa" melalui "Halaman Daftar Kelas".

Hal yang ditemukan ketika dilakukan iterasi 3:

- Dengan munculnya *use case* DUC-301 dan DUC-302, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan kedua *use case* yang di-*invoke*.

Hal yang ditemukan ketika dilakukan iterasi 4:

- Dengan dihapusnya *use case* DUC-301 dan DUC-302, maka dilakukan beberapa penambahan dan penyesuaian untuk memfasilitasi *alternate scenario* opsional ketika pengguna TU mencari data kelas maupun data mahasiswa.

4.4.5.14. UC-303 TU Mengubah Status Kehadiran Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.
- Alternate scenario yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Dengan munculnya UC-302, maka sebagian besar elemen *robustness diagram* yang ada dihapus dan disubstitusi dengan menggunakan *use case* yang di-*invoke*.

Hal yang ditemukan ketika dilakukan iterasi 3:

- Hubungan antara *boundary* “Halaman Mahasiswa” serta *boundary* “Tampilan Ubah Status Kehadiran Mahasiswa” dengan *entity* “Data Mahasiswa” dirasa masih kurang tepat sehingga ditambahkan *controller* “Muat Data Mahasiswa” dan *controller* “Tampilkan Status Kehadiran Mahasiswa” agar alur *use case* lebih jelas.

- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.

Hal yang ditemukan ketika dilakukan iterasi 3:

- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.
- Penggunaan "Mahasiswa" sebagai *entity* dirasa lebih sesuai dengan karakteristik *entity* dibandingkan "Data Mahasiswa"
- Prototipe *user interface* dirasa kurang memfasilitasi *use case* yang ada sehingga dilakukan sedikit revisi terhadap prototipe *user interface* yang ada.

4.4.5.15. UC-304 TU Menghapus Data Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 1 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.
- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.
- Penggunaan "Mahasiswa" sebagai *entity* dirasa lebih sesuai dengan karakteristik *entity* dibandingkan "Data Mahasiswa".

4.4.5.16. UC-305 TU Menambah Kelas Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.
- *Alternate scenario* yang ada tidak fokus ke interaksi aplikasi / sistem ke pengguna sehingga dihapus.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Hubungan antara *boundary* "Halaman Tambah Kelas" dengan *entity* "Daftar Kelas" dirasa masih kurang tepat sehingga ditambahkan *controller* "Muat Daftar Kelas" yang menghubungkan antara *entity* "Daftar Kelas" dengan *controller* "Tampilkan Halaman Tambah Kelas".

Hal yang ditemukan ketika dilakukan iterasi 3:

- Ada *alternate scenario* opsional yang dirasa perlu ditambahkan, yaitu ketika TU mencari data kelas menggunakan *search box* yang ada.

Hal yang ditemukan ketika dilakukan iterasi 4:

- Dengan dihapusnya *use case* DUC-301, maka dilakukan beberapa penyesuaian untuk memfasilitasi *alternate scenario* opsional ketika TU mencari data kelas.

4.4.5.17. UC-306 TU Memindah Kelas Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 4 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.

- *Alternate scenario* yang ada tidak fokus ke interaksi aplikasi / sistem ke pengguna sehingga dihapus.

Hal yang ditemukan ketika dilakukan iterasi 2:

- Ada *alternate scenario* yang dirasa perlu ditambahkan, yaitu ketika sistem gagal mengubah kelas.
- Pemunculan *toast* pada alur *normal scenario* dinilai kurang cocok sehingga dipindahkan ke alur *alternate scenario*.

4.4.5.18. UC-307 TU Menghapus Kelas Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tidak ada iterasi yang dilakukan untuk use case ini karena UC-306 dirasa sudah cukup *robust* namun pada tahapan awal (iterasi 0) ditemukan bahwa prototipe *user interface* dirasa kurang memfasilitasi *use case* yang ada sehingga dilakukan sedikit revisi terhadap prototipe *user interface* yang ada.

4.4.5.19. UC-308 TU Menambah Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 5 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Alur yang diajukan pada *use case description* masih kurang sesuai dengan prototipe *user interface* yang diajukan sehingga dilakukan beberapa penyesuaian.
- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.
- Prototipe *user interface* dirasa kurang memfasilitasi *use case* yang ada sehingga dilakukan sedikit revisi terhadap prototipe *user interface* yang ada.

Hal yang ditemukan ketika dilakukan iterasi 2:

- *Entity* "Daftar Kelas" dirasa perlu dicantumkan pada *use case* ini karena halaman tambah kelas menampilkan daftar kelas.

Hal yang ditemukan ketika dilakukan iterasi 3:

- *Alternate scenario* yang ada kurang fokus ke interaksi aplikasi / sistem ke pengguna sehingga dilakukan revisi *alternate scenario*.
- Ada *alternate scenario* yang dirasa perlu ditambahkan, yaitu ketika TU ingin mencari kelas menggunakan *search box*.
- Penggunaan "Mahasiswa" sebagai *entity* dirasa lebih sesuai dengan karakteristik *entity* dibandingkan "Data Mahasiswa".

Hal yang ditemukan ketika dilakukan iterasi 4:

- Dengan dihapusnya DUC-301, maka dilakukan beberapa penambahan dan penyesuaian untuk memfasilitasi *alternate scenario* opsional ketika TU mencari data kelas.

Hal yang ditemukan ketika dilakukan iterasi 5:

- Perbaikan prototipe *user interface* menyebabkan beberapa elemen yang ada di *robustness diagram* menjadi kurang tepat untuk mewakili alur yang terjadi di prototipe *user interface* yang baru sehingga dilakukan beberapa penyesuaian.

4.4.5.20. UC-309 TU Menambah Kelas Mata Kuliah Baru

Use case ini merupakan salah satu *use case* yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Pada tahapan awal (iterasi 0) ditemukan bahwa prototipe *user interface* dirasa kurang memfasilitasi *use case* yang ada sehingga dilakukan sedikit revisi terhadap prototipe *user interface* yang ada. Namun ternyata prototipe *user interface* yang ada masih memerlukan revisi sehingga masih

dibutuhkan iterasi agar *use case* ini *robust*. Tahapan iterasi untuk use case ini dilakukan sebanyak 1 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Perbaikan prototipe *user interface* menyebabkan beberapa elemen yang ada di *robustness diagram* menjadi kurang tepat untuk mewakili alur yang terjadi di prototipe *user interface* yang baru sehingga dilakukan beberapa penyesuaian.

4.4.5.21. UC-310 TU Memilih Lokasi Template Presensi

Use case ini merupakan salah satu use case yang ditemukan pada waktu pengumpulan kebutuhan fungsional yang ada di desain sistem. Pada tahapan awal (iterasi 0) ditemukan bahwa prototipe *user interface* dirasa kurang memfasilitasi *use case* yang ada sehingga dilakukan sedikit revisi terhadap prototipe *user interface* yang ada. Namun ternyata prototipe *user interface* yang ada masih memerlukan revisi sehingga masih dibutuhkan iterasi agar *use case* ini *robust*. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 1 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Perbaikan prototipe *user interface* menyebabkan beberapa elemen yang ada di *robustness diagram* menjadi kurang tepat untuk mewakili alur yang terjadi di prototipe *user interface* yang baru sehingga dilakukan beberapa penyesuaian.

4.4.5.22. DUC-301 TU Mencari Kelas

Use case ini merupakan salah satu *use case* yang ditemukan pada tahapan pada tahapan *robustness analysis*, asal dari *use case* ini adalah pelebaran dari *use case* UC-302, UC-305, dan UC-308 dimana ada *alternate scenario* opsional ketika TU mencari kelas menggunakan *search box* yang ada. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 2 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Fungsi *search* yang diajukan pada prototipe *user interface* tidak terlalu rumit sehingga *entity* "Hasil Pencarian Kelas" tidak perlu dimasukkan ke dalam diagram yang ada. Selain itu juga dilakukan sedikit penyesuaian untuk sebagai dampak dari dihapusnya *entity* "Hasil Pencarian Kelas".

Hal yang ditemukan ketika dilakukan iterasi 2:

- Dengan pertimbangan bahwa fungsi *search* yang diajukan pada prototipe *user interface* tidak terlalu rumit maka keberadaan *use case* ini tidak diperlukan sehingga DUC-301 dihapus dan dilakukan penyesuaian untuk *use case* yang memanggil DUC-301.

4.4.5.23. DUC-302 TU Mencari Mahasiswa

Use case ini merupakan salah satu *use case* yang ditemukan pada tahapan pada tahapan *robustness analysis*, asal dari *use case* ini adalah pelebaran dari *use case* UC-302 dimana ada *alternate scenario* opsional ketika TU mencari mahasiswa menggunakan *search box* yang ada. Tahapan iterasi untuk *use case* ini dilakukan sebanyak 2 kali.

Hal yang ditemukan ketika dilakukan iterasi 1:

- Fungsi *search* yang diajukan pada prototipe *user interface* tidak terlalu rumit sehingga *entity* "Hasil Pencarian Mahasiswa" tidak perlu dimasukkan ke dalam diagram yang ada. Selain itu juga dilakukan sedikit penyesuaian untuk sebagai dampak dari dihapusnya *entity* "Hasil Pencarian Mahasiswa".

Hal yang ditemukan ketika dilakukan iterasi 2:

- Dengan pertimbangan bahwa fungsi *search* yang diajukan pada prototipe *user interface* tidak terlalu rumit maka keberadaan *use case* ini tidak diperlukan sehingga DUC-

302 dihapus dan dilakukan penyesuaian untuk *use case* yang memanggil DUC-302

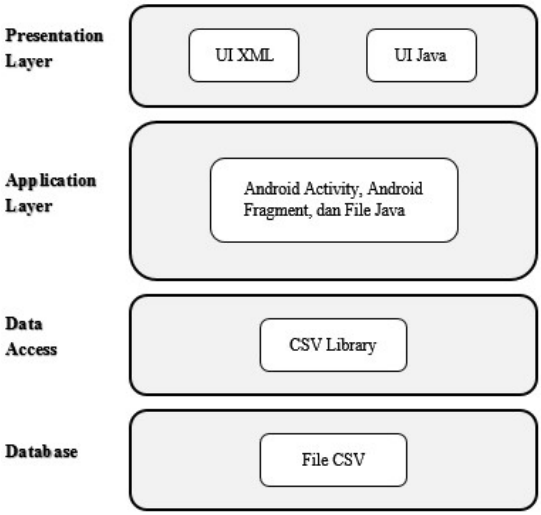
4.4.6. Updated Domain Model

Berdasarkan *robustness analysis* yang sudah dilakukan, ada beberapa *entity* yang ditemukan dimana *entity* tersebut dimasukkan menjadi *domain* ke dalam *domain model* yang ada. Selain itu, beberapa *controller* yang ada pada tahapan *robustness analysis* menunjukkan bahwa ada beberapa atribut yang juga harus dimasukan ke dalam *domain* yang ada. Hasil dari tahapan ini bisa dilihat pada Lampiran E.

4.4.7. Technical Architecture

Technical architecture dilakukan untuk mempermudah proses pembuatan *sequence diagram*. Karena tahapan *sequence diagramming* dimaksudkan untuk menggambarkan bagaimana setiap objek dalam *use case* berinteraksi sehingga diperlukan pemahaman bagaimana *framework* Android bekerja.

Untuk mempermudah pemahaman, berikut ini pemetaan komponen *framework* Android yang dibagi ke dalam 4 *layer*.



Gambar 4-3 Technical Architecture AttendJSI

Berikut ini komponen yang nantinya akan dimunculkan pada kode program:

4.4.7.1. Activity

Activity adalah komponen android yang menyediakan tampilan di dalam sebuah aplikasi dimana pengguna akan berinteraksi dengannya. [29] *Activity* adalah komponen utama dalam pembuatan aplikasi android karena tanpa adanya *activity*, akan sulit untuk menjalankan komponen komponen lainnya untuk menjadi aplikasi android yang utuh.

Pada *sequence diagram*, *method* yang diimplementasikan adalah :

1. StartActivity(Intent) : yang digunakan hampir semua *activity* yang ada pada rancangan attendJSI. *Method* ini diubah sedikit menjadi StartActivity() sehingga lebih mudah dibaca namun untuk implementasinya di kode program tetap menggunakan StartActivity(Intent) dengan *method* bernama StartActivity() untuk *class* yang menggunakannya.
2. StartActivityForResult(Intent, int) : yang digunakan untuk beberapa *activity* khusus, seperti proses pengambilan gambar menggunakan kamera. *Method* ini diubah sedikit menjadi StartActivity() sehingga lebih mudah dibaca namun untuk implementasinya di kode program tetap menggunakan StartActivityForResult(Intent, int) dengan *method* bernama StartActivityForResult() untuk *class* yang menggunakannya.

Dalam pembuatan kode program, *activity* yang digunakan tidak terbatas hanya pada activity, namun juga bisa menggunakan *fragmentActivity*, *listActivity*, *appCompatActivity* dan lain sebagainya, sesuai dengan kebutuhan.

4.4.7.2. Fragment

*Fragmen*t adalah komponen android yang berbentuk pecahan, dimana komponen itu bisa digunakan oleh banyak *activity*

ataupun dijadikan *nested fragment*. Umumnya, *fragment* tidak bisa digunakan secara langsung, untuk menampilkannya dibutuhkan untuk dilakukan *embed* ke *activity* terlebih dahulu.

Dalam pembuatan kode program, *fragment* yang digunakan tidak terbatas hanya pada *fragment*, namun juga bisa menggunakan *listFragment* atau yang lain, sesuai kebutuhan.

4.4.7.3. Dialog

Dialog adalah layar kecil yang mendorong pengguna untuk membuat keputusan atau memasukan informasi tambahan. *Dialog* pada umumnya tidak memenuhi layar dan biasanya digunakan untuk *modal event* (suatu kondisi yang membutuhkan pengguna untuk melakukan sesuatu sebelum mereka bisa melanjutkan). [30]

Pada *sequence diagram*, ada 2 jenis dialog yang digunakan, yaitu:

1. *Default dialog* : digunakan ketika ada sebuah kejadian yang membutuhkan tampilan *dialog* sederhana, misal : salah input NRP. Sesuai dengan kebutuhan, maka pemunculan *dialog* dalam kode program hanya memanggil *dialog default* yang tidak membutuhkan UI ataupun komponen khusus.
2. *Custom dialog* : digunakan ketika ada sebuah kejadian yang membutuhkan tampilan *dialog* yang sedikit rumit, misal : mengganti status kehadiran mahasiswa. Sesuai dengan kebutuhan, maka pemunculan *dialog* dalam kode program juga membutuhkan UI *custom*, sehingga dibuat xml khusus dan juga *file* java untuk UI jika perlu.

Komponen ini tidak dimasukkan menjadi *class* dengan pertimbangan bahwa pemanggilan komponen bisa dilakukan pada *activity* atau *fragment* sehingga tidak dimunculkan pada *sequence diagram* ataupun *class diagram*.

4.4.7.4. Toast

Toast adalah sebuah tampilan yang berisi pesan singkat untuk *user*. [31] Tujuan dari digunakannya *Toast* adalah agar pesan yang ditampilkan tidak terlalu merepotkan pengguna sebagaimana *dialog*.

Komponen ini tidak dimasukkan menjadi *class* dengan pertimbangan bahwa pemanggilan komponen bisa dilakukan pada *activity* atau *fragment* sehingga tidak dimunculkan pada *sequence diagram* ataupun *class diagram*.

4.4.7.5. Snackbar

Snackbar menyediakan umpan balik yang ringan terhadap sebuah perilaku user. *Snackbar* menampilkan pesan singkat pada bagian bawah layar *handset*. Snackbar ditampilkan diatas, menutupi, elemen elemen lainnya dan hanya bisa tampil satu *snackbar* pada satu waktu. [32]

Secara sederhana, *snackbar* adalah gabungan / *hybrid* dari *toast* dengan *dialog*. Pada AttendJSI, *snackbar* akan digunakan untuk beberapa kasus yang membutuhkan tombol *undo*, seperti : menghapus data.

Komponen ini tidak dimasukkan menjadi *class* dengan pertimbangan bahwa pemanggilan komponen bisa dilakukan pada *activity* atau *fragment* sehingga tidak dimunculkan pada *sequence diagram* ataupun *class diagram*.

4.4.7.6. Camera

Framework Android menyertakan dukungan untuk berbagai jenis kamera dan macam - macam fitur dari kamera yang ada pada *handset* / *device*, yang memungkinkan pengguna untuk mengambil foto ataupun video pada aplikasi. [33]

Berhubung aplikasi hanya membutuhkan fungsi sederhana dari *Camera*, yaitu mengambil foto dan menyimpan gambar yang diambil, maka pada *sequence diagram* tidak perlu ditambahkan *class* khusus. Agar alur *sequence diagram* tidak rancu, ditambahkan *boundary* bernama "*Android Camera*" untuk

mewakili *Camera*. Namun *boundary* "*Android Camera*" tidak akan dimasukkan ke dalam *class diagram*.

4.4.8. Sequence Diagramming

Use case yang sudah melalui proses *robustness analysis* akan dijadikan sebagai dasar dari pembuatan *sequence diagram*. Selain itu, juga dilakukan penyesuaian komponen yang ada di *sequence diagram* sesuai dengan *technical architecture* dari *software* dan *framework* yang akan digunakan untuk membangun aplikasi.

Sequence diagram adalah diagram yang menggambarkan hubungan antar objek dalam waktu yang berurutan untuk memenuhi permintaan dari pengguna, sesuai dengan skenario yang digambarkan pada *use case*, sesuai dengan proses *logic* yang ada pada aplikasi.

Untuk tahapan ini, hasil akhir dari tahapan *sequence diagramming* bisa dilihat di Lampiran F.

4.4.9. Class Diagramming

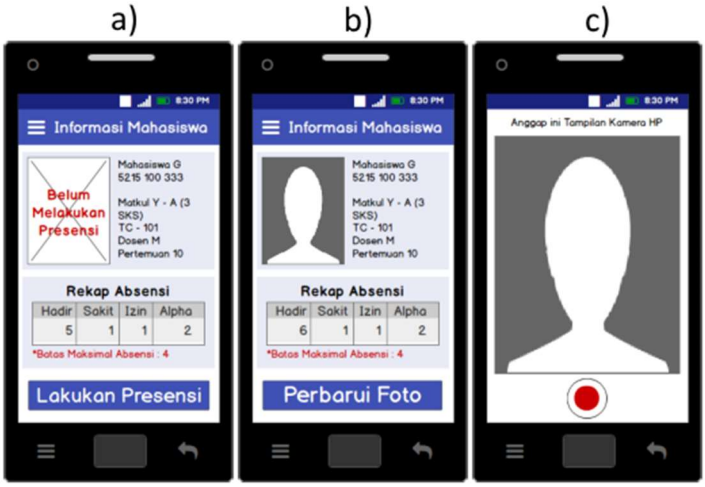
Class diagram berisikan fungsi fungsi yang ditemukan pada tahapan *sequence diagramming*. *Class diagram* ini nantinya dijadikan sebagai dasar dari pembuatan kode program.

Untuk tahapan ini, hasil akhir dari tahapan *class diagramming* bisa dilihat di Lampiran G.

BAB V
IMPLEMENTASI

Implementasi yang dilakukan pada tugas akhir ini berupa pembuatan baris kode sesuai dengan perancangan yang telah dilakukan. Agar penjelasan tidak terlalu bertele - tele, maka dipilih salah satu kebutuhan fungsional yang dijelaskan proses implementasinya menjadi baris kode, yaitu : FR-101 mahasiswa melakukan presensi / pencatatan kehadiran.

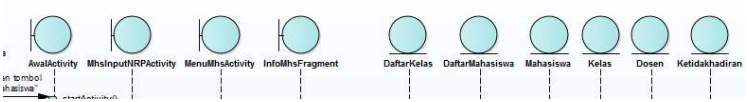
Untuk mendapatkan baris kode, pertama kali dilihat tampilan prototipe *user interface* mana saja yang digunakan dalam kebutuhan fungsional ini. tampilan prototipe UI yang digunakan pada kebutuhan fungsional ini adalah : UI001, UI101, UI102, UI103, UI104, UI105, UI106. Dimana tampilan inti adalah UI104 yang digunakan untuk menampilkan halaman informasi mahasiswa, UI106 yang merupakan tampilan kamera ketika tombol "Lakukan Presensi" pada UI104 ditekan, serta UI105 yang menunjukkan perubahan dari halaman informasi mahasiswa ketika sudah dilakukan pengambilan foto mahasiswa melalui kamera.



Gambar 5-1 tampilan UI104 (a), tampilan UI105 (b), tampilan UI106 (c)

Setelah itu, dilakukan identifikasi apa saja kira kira elemen UI pada android yang dibutuhkan untuk menampilkan UI104, UI105, dan UI106. Untuk itu, bisa digunakan bantuan dari *sequence diagram* yang berkorelasi dengan FR-101, yaitu SD101 dan SD102.

SD101 berasal dari UC-101 yang di-*invoke* untuk menjalankan UC-102 yang merupakan fungsi inti dari FR-101 sehingga referensi yang digunakan adalah SD102. Pertama – tama dilihat *class* apa saja yang terdapat pada *sequence diagram*.



Gambar 5-2 Class yang terdapat pada SD102

Dari Gambar 5-2 Class yang terdapat pada SD102, bisa dilihat ada beberapa *boundary class* dan ada beberapa *entity class* yang terlibat dalam proses. Berikut ini daftar *boundary class* yang ada dalam SD102 :

- AwalActivity
- MhsInputNRPActivity
- MenuMhsActivity
- MhsInfoFragment

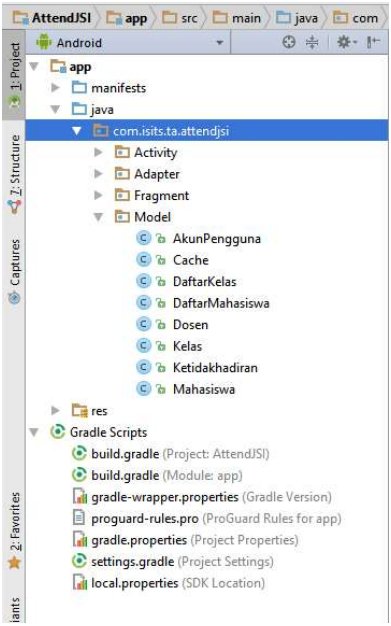
Lalu juga ada *entity class* yang ada dalam SD102, yaitu :

- DaftarKelas
- Daftar Mahasiswa
- Mahasiswa
- Kelas
- Dosen
- Ketidakhadiran

Dari *boundary class*, bisa dilihat bahwa akan digunakan 2 *class* bawaan android, *activity* dan *fragment*, yang nantinya dipanggil

dengan cara di-*extend*. Sehingga pada Android Studio, digunakan *wizard* untuk membuat keempat *boundary class* tersebut di *project* android yang sudah ada.

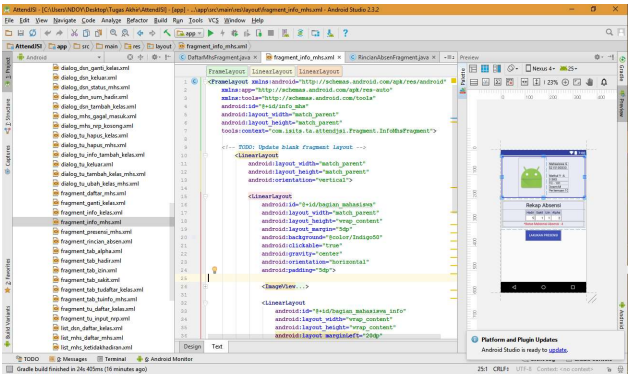
Untuk mempermudah proses pembuatan baris kode, *class* yang ada dimasukkan ke dalam *package* masing – masing. misalnya *boundary class* yang meng-*extend android activity* dimasukkan ke dalam *package activity*, *boundary class* yang meng-*extend android fragment* dimasukkan ke dalam *package fragment*, *entity class* dimasukkan ke dalam *package model*.



Gambar 5-3 Tampilan hirarki package pada project android studio

Setelah semua *class* yang berhubungan selesai di-*build*, dilakukan penyesuaian tampilan sesuai dengan tampilan prototipe yang ada dengan cara mengubah isi dari file xml yang terhubung ke dalam *activity* / *fragment* yang ada. InfoMhsFragment adalah *class* yang mewakili tampilan UI104

dan UI105, dimana *file* xml yang terhubung adalah fragment_info_mhs.xml.



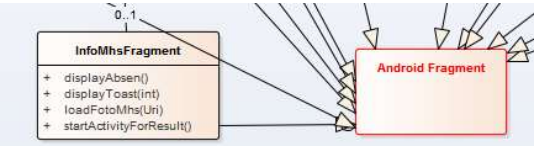
Gambar 5-4 Tampilan file fragment_info_mhs.xml di Android Studio

Fungsi dari *file* fragment_info.mhs.xml adalah membentuk UI untuk halaman info mahasiswa, atau dalam konteks android bisa disebut sebagai *fragment* info mahasiswa. Berikut ini beberapa komponen UI yang digunakan untuk membentuk tampilan yang digunakan oleh class InfoMhsFragment :

- *FrameLayout* digunakan sebagai *layout* dasar karena *FrameLayout* dirancang untuk menampilkan tampilan tunggal. Selain itu, untuk menampilkan *fragment*, umumnya digunakan *FrameLayout* pada Android.
- Ada *ScrollView* dengan tujuan agar *layout* yang dihasilkan bisa di-*scroll* ketika dijalankan di *handset* yang memiliki resolusi layar kecil.
- Sementara untuk membuat tabel digunakan *LinearLayout* dengan id header_tabel. Pertimbangan dari penggunaan *LinearLayout* adalah karena lebih bisa mewakili elemen UI yang diajukan pada prototipe dibandingkan *TableLayout*.

Setelah tampilan UI pada *file* xml sudah mirip dengan prototipe UI yang ada, selanjutnya dilengkapi dengan mengisi baris kode pada *class* sesuai dengan yang ada pada *class* diagram. Untuk

class InfoMhsFragment, ada 4 *method*, yaitu : displayAbsen(), displayToast(int), loadFotoMhs(Uri), dan StartActivityForResult()



Gambar 5-5 InfoMhsFragment pada class diagram

Setelah itu baris kode yang ada dilengkapi sehingga bisa berfungsi sebagai mana mestinya. Berikut ini daftar *method* yang ada pada class InfoMhsFragment setelah dilakukan penulisan baris kode lebih lanjut :

onCreateView : salah satu *method* bawaan dari *class fragment* yang ada di *framework* android. Pada file ini, *method* ini berisi beberapa pengaturan respon elemen UI terhadap interaksi pengguna, contoh : setOnClickListener yang digunakan untuk mendefinisikan apa yang terjadi ketika pengguna melakukan *click* pada elemen UI tersebut.

```
41    @Nullable
42    @Override
43    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
44                             @Nullable Bundle savedInstanceState) {
45        View v = inflater.inflate(R.layout.fragment_info_mhs, container, false);
46
47        fotoMhs = (ImageView) v.findViewById(R.id.mhs_foto);
48        lokasi = (TextView) v.findViewById(R.id.batas_absen);
49
50        Button presensi = (Button) v.findViewById(R.id.tombol_presensi);
51
52        presensi.setOnClickListener((v) -> { takePhoto(v); });
58
59        fotoMhs.setOnClickListener((v) -> { takePhoto(v); });
65
66        if (new DaftarMahasiswa().loadMhsFoto(nrpMhs) != null) {
67            imageUri = new DaftarMahasiswa().loadMhsFoto(nrpMhs);
68            loadFotoMhs(imageUri);
69        }
70
71        LinearLayout shortcut = (LinearLayout) v.findViewById(R.id.bagian_presensi);
72        shortcut.setOnClickListener((v) -> {
75                RincianAbsenFragment fragment = new RincianAbsenFragment();
76                FragmentTransaction ft = getActivity().getSupportFragmentManager().beginTransaction();
77                ft.replace(R.id.frame, fragment);
78                ft.commit();
79        });
81
82        return v;
83    }
```

Gambar 5-6 method onCreateView

onViewCreated : salah satu *method* bawaan dari *class fragment* yang ada di *framework* android. Pada file ini, *method* ini hanya memiliki satu baris kode, yaitu getActivity.setTitle("Informasi Mahasiswa") yang digunakan untuk mengatur tulisan yang ditampilkan pada *title bar* aplikasi.

```
85      @Override
86      public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
87          super.onViewCreated(view, savedInstanceState);
88          getActivity().setTitle("Informasi Mahasiswa");
89      }
```

Gambar 5-7 method onViewCreated

takePhoto : salah satu *method* yang diajukan pada *class diagram*. *Method* ini berisi kode yang digunakan untuk pengambilan gambar / foto.

```
92      private void takePhoto(View v) {
93          String filename = kodekelas + "-" + Integer.toString(nrpMhs) + ".jpg";
94          if (Build.VERSION.SDK_INT >= 23) {
95              ActivityCompat.requestPermissions(getActivity(), new String[]{
96                      Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
97              ActivityCompat.requestPermissions(getActivity(), new String[]{
98                      Manifest.permission.READ_EXTERNAL_STORAGE}, 1);
99          }
100
101         Intent intent = new Intent("android.media.action.IMAGE_CAPTURE");
102         File photo = new File(Environment.getExternalStoragePublicDirectory(
103                 Environment.DIRECTORY_PICTURES), filename);
104         imageUri = Uri.fromFile(photo); //doesn't work in API 24 or higher
105         intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
106         startActivityForResult(intent, TAKE_PICTURE);
107     }
```

Gambar 5-8 method takePhoto

onActivityResult : *method* bawaan framework android yang digunakan untuk menerima hasil dari fungsi startActivityForResult. Dalam file ini, *method* ini digunakan untuk menerima hasil dari pengambilan gambar / foto yang dipanggil di *method* takePhoto.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);

    if (resultCode == Activity.RESULT_OK) {
        imageUri = new DaftarMahasiswa().loadMhsFoto(nrpMhs);
        new DaftarMahasiswa().updateMhsURIFoto(imageUri);
        loadFotoMhs(imageUri);
    }

}
```

Gambar 5-9 method onActivityResult

displayAbsen : salah satu *method* yang diajukan pada *class diagram*. Untuk sementara *method* ini kosong karena baris kode yang ada sudah cukup untuk menampilkan prototipe UI.

```
public void displayAbsen(){

}
```

Gambar 5-10 method displayAbsen

displayToast : salah satu *method* yang diajukan pada *class diagram*. Untuk sementara *method* ini kosong karena baris kode yang ada sudah cukup untuk menampilkan prototipe UI.

```
public void displayToast(int idtoast){

}
```

Gambar 5-11 method displayToast

.**loadFotoMhs :** salah satu *method* yang diajukan pada *class diagram*. *Method* ini berisi kode yang digunakan untuk menampilkan foto mahasiswa.

```
public void loadFotoMhs(Uri URIFoto) {
    Uri selectedImage = URIFoto;
    getActivity().getContentResolver().notifyChange(selectedImage, null);

    ContentResolver cn = getActivity().getContentResolver();
    Bitmap bitmap;

    try {
        bitmap = MediaStore.Images.Media.getBitmap(cn, selectedImage);
        lokasi.setText(imageUri.toString());
        fotoMhs.setImageBitmap(bitmap);

    } catch (Exception e) {
        Log.e("error:", e.toString());
    }
}
```

Gambar 5-12 method loadFotoMhs

startActivityforResult : salah satu *method* yang diajukan pada *class diagram*. Untuk sementara *method* ini kosong karena baris kode yang ada sudah cukup untuk menampilkan prototipe UI.

```
146     public void startActivityforResult(){
147
148     }
```

Gambar 5-13 method startActivityforResult

Setelah semua *class* yang ada di class diagram dan semua tampilan dari prototipe UI versi akhir ditulis menjadi baris kode. Aplikasi selanjutnya akan di-*build* menjadi file berekstensi *.apk sehingga aplikasi bisa dijalankan ke dalam *handset* android yang ada.

BAB VI
HASIL DAN PEMBAHASAN

Bab ini akan dijelaskan mengenai hasil dan pembahasan dari analisis, hasil penelitian dan usulan berupa rekomendasi.

6.1. Verifikasi Desain UI

Tahapan verifikasi desain UI dilakukan dengan 2 cara yaitu : *usability study* dan *heuristic evaluation* yang digunakan pada 3 tahapan desain UI (*low fidelity prototyping*, *medium fidelity prototyping*, dan *high fidelity prototyping*).

6.1.1. Usability Study Low Fidelity Prototyping Iterasi 0

Pada tahapan ini, kelompok pengguna yang dilibatkan pada *usability study* adalah mahasiswa. Berikut ini hasil dari *usability study* yang dilakukan :

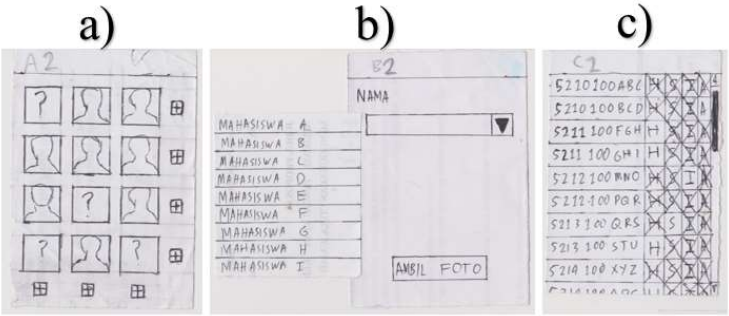
Tabel 6-1 Hasil Usability Study LFP Iterasi 0

Task	**Prototipe A**		**Prototipe B**		**Prototipe C**	
	Avg. Success Rate	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Completion Time***
1. Melakukan Presensi	60%	66,67	100%	32,2	80%	30

Dari Tabel 6-1 Hasil Usability Study LFP Iterasi 0, terlihat bahwa prototipe B adalah yang memiliki *success rate* paling bagus, dengan *success rate* 100% dan rerata *completion time* 32,2 detik. Sementara prototipe C memiliki rerata *completion time* paling bagus dengan *success rate* 80% dan rerata *completion time* 30 detik. Sementara itu prototipe A memiliki *success rate* dan *completion time* paling buruk dengan *success rate* 60% dan rerata *completion time* 66,67 detik.

Perbedaan mendasar dari prototipe A dibandingkan dengan 2 prototipe lainnya adalah prototipe A memiliki tampilan awal berupa *grid* sementara tampilan awal 2 prototipe lainnya berbentuk daftar (prototipe B berbentuk *drop-down list*, semen-

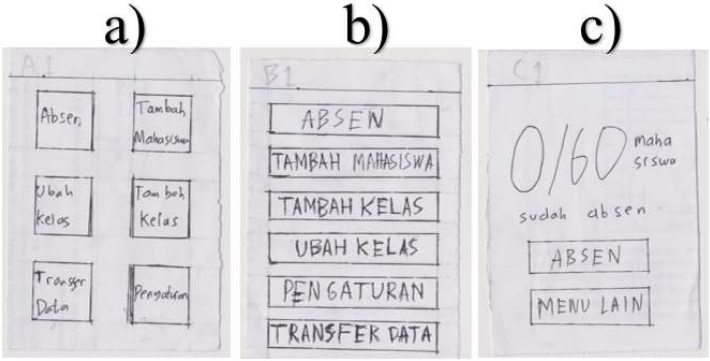
tara prototipe C berbentuk *scroll-able list*).



Gambar 6-1 Perbandingan antara tampilan awal prototipe A (a), tampilan awal prototipe B (b), dengan tampilan awal prototipe C (c)

Berdasarkan hasil *feedback*, tampilan awal prototipe C dan prototipe A memang menyusahkan, dimana partisipan tidak paham dengan apa yang harus dilakukan di tampilan awal prototipe A, sementara di prototipe C partisipan merasa tidak berhasil menemukan 'tombol' yang bisa membawa mereka tampilan berikutnya. Untuk prototipe C, agar bisa berpindah ke tampilan berikutnya, mayoritas partisipan menekan bagian dari daftar mahasiswa secara acak.

Sementara untuk proses presensi pada ketiga prototipe kurang lebih sama (*login* ➔ pilih menu presensi ➔ ambil foto ➔ selesai) dimana menurut beberapa partisipan, alurnya terasa aneh karena kebiasaan mahasiswa di Jurusan Sistem Informasi adalah melakukan presensi tanpa perlu memilih menu presensi ataupun melakukan *login*. Jadi ketika mahasiswa disodori kertas presensi, mereka hanya perlu mencari NRP / nama mereka dari daftar mahasiswa yang disodorkan dan setelah itu menulis tanda tangan untuk melakukan presensi.



Gambar 6-2 tampilan menu awal mahasiswa untuk prototipe A (a), prototipe B (b), dan prototipe C (c)

6.1.2. Usability Study Low Fidelity Prototyping Iterasi 1

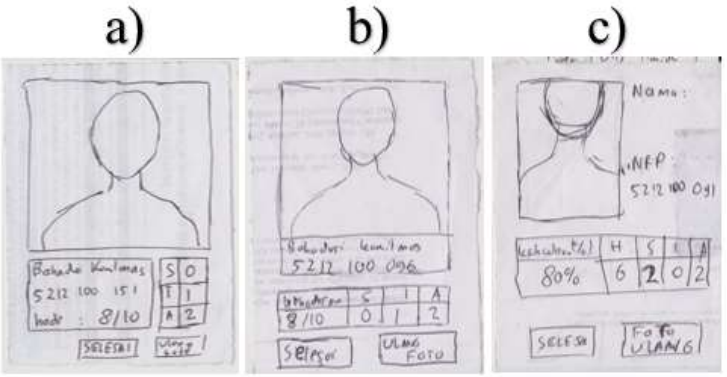
Pada tahapan ini, kelompok pengguna yang dilibatkan pada *usability study* adalah mahasiswa dan dosen. Berikut ini hasil dari pengujian yang dilakukan :

Tabel 6-2 Hasil Usability Study LFP Iterasi 1

Task	**Prototipe A**		**Prototipe B**		**Prototipe C**	
	Avg. Success Rate	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Completion Time***
1. Melakukan Presensi	100%	41,4	80%	30,5	100%	39,4
2. Melihat Minggu Perkuliahan	0%	0	0%	0	20%	22
3. Melihat Batas Maksimal Absensi	0%	0	0%	0	20%	35
4. Memastikan Status Kehadiran Mahasiswa	0%	0	0%	0	0%	0

Berdasarkan Tabel 6-2 Hasil Usability Study LFP Iterasi 1, yang menjadi fokus dari *usability study* kali ini adalah kenapa *task* 2, *task* 3 dan *task* 4 *success ratenya* sangat rendah.

Berdasarkan *feedback*, terutama untuk tampilan setelah pengguna melakukan penggambilan gambar, mayoritas partisipan mengeluhkan tidak ada informasi terkait batas maksimal absensi atau setidaknya informasi SKS untuk mata kuliah. Hal itu menyebabkan mahasiswa kesulitan melakukan *task* 3. Meskipun sudah ditampilkan rasio perbandingan kehadiran dengan ketidakhadiran, namun hal itu tidak banyak membantu.



Gambar 6-3 Tampilan setelah pengguna melakukan pengambilan gambar pada prototipe A (a), prototipe B (b), dan prototipe C (c)

Sementara itu, untuk *task* 2, mayoritas partisipan mengatakan bahwa 'pertemuan ke' tidak sama dengan 'minggu perkuliahan' meskipun ada yang menganggap bahwa keduanya sama.

Untuk *task* 4, berdasar pemaparan partisipan, penyebab *success rate*nya rendah adalah karena tampilan yang harus dilalui oleh partisipan terlalu banyak dan rumit.

6.3.1. Usability Study Low Fidelity Prototyping Iterasi 2

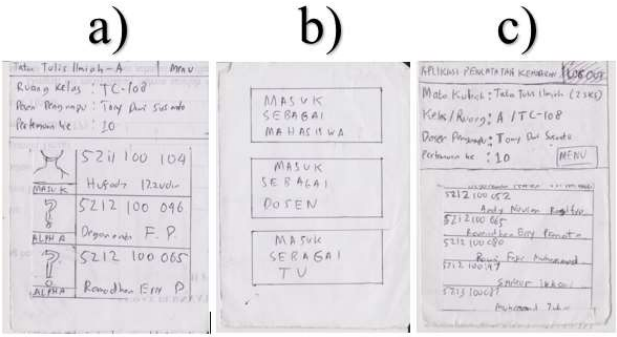
Pada tahapan ini, kelompok pengguna yang dilibatkan pada *usability study* adalah dosen dan TU. Berikut ini hasil dari pengujian yang dilakukan :

Tabel 6-3 Hasil Usability Study LFP Iterasi 2

Task	**Prototipe A**		**Prototipe B**		**Prototipe C**	
	Avg. Success Rate	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Completion Time***
1. Melihat Informasi Perkuliahan	100%	46	100%	116	0%	0
2. Memastikan Status Kehadiran Mahasiswa	100%	20	100%	72	0%	0
3. Mengubah Status Kehadiran Mahasiswa	100%	12	0%	0	0%	0
4. Memastikan Data Dosen Cocok Dengan Data Di Aplikasi	0%	0	0%	0	0%	0
5. Mengatasi Mahasiswa Yang Pindah Kelas	0%	0	0%	0	0%	0

Berdasarkan Tabel 6-3 Hasil Usability Study LFP Iterasi 2, prototipe A menunjukkan hasil terbaik dibandingkan 2 prototipe lainnya, namun *task* 4 dan *task* 5 masih mendapatkan *success rate* 0% sebagaimana 2 prototipe lainnya.

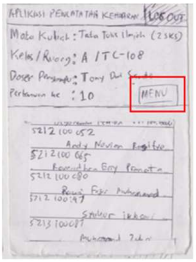
Berdasarkan pemaparan partisipan, alur dari ketiga prototipe tidak seperti yang biasa ia lakukan pada proses pencatatan kehadiran. Tampilan awal seharusnya ditunjukkan daftar kelas terlebih dahulu lantas baru diberi fungsi lain – lain.



Gambar 6-4 tampilan awal prototipe A (a), tampilan awal prototipe B (b), dan tampilan awal prototipe C (c)

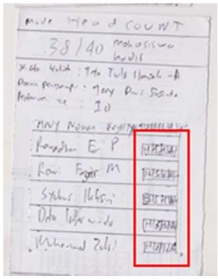
Sementara itu, ada kasus unik untuk prototipe 1, dimana partisipan memilih menggunakan cara lain dari yang disediakan oleh prototipe A untuk mengerjakan *task* 1. Partisipan menunjukkan bahwa sebaiknya ada *pop-up* yang bisa menunjukkan status kehadiran mahasiswa secara langsung.

Untuk prototipe C, partisipan kesulitan menemukan tombol menu yang letaknya tidak biasa sehingga semua *task* tidak berhasil dilakukan.



Gambar 6-5 tombol menu pada tampilan awal prototipe C (ditandai dengan garis merah)

Untuk prototipe B, partisipan tidak paham bahwa kotak kecil di bagian kanan nama mahasiswa adalah tombol yang bisa ditekan untuk mengubah status mahasiswa sehingga partisipan gagal melakukan *task* 3.



Gambar 6-6 tombol untuk mengubah status mahasiswa pada prototipe B

6.1.3. Usability Study Low Fidelity Prototyping Iterasi 3

Pada tahapan ini, ketiga kelompok pengguna dilibatkan pada *usability study*. Berikut ini hasil dari pengujian yang dilakukan:

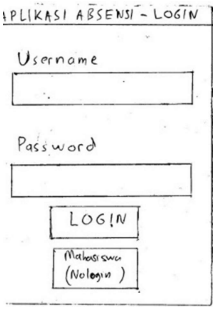
Tabel 6-4 Hasil Usability Study LFP Iterasi 3

Task	Prototipe A		Prototipe B		Prototipe C	
	Avg. Success Rate	*Avg. Completion Time*	*Avg. Success Rate*	*Avg. Completion Time*	*Avg. Success Rate*	*Avg. Completion Time*
1. Memeriksa Informasi Kelas	80%	67	80%	40,5	100%	26,4
2. Melakukan Presensi	80%	40,5	100%	49,6	100%	27,2
3. Melihat Batas Maksimal Absensi	100%	72,2	100%	47,6	100%	90,2
4. Melihat Informasi Perkuliahan	50%	109	100%	19,5	50%	38
5. Memastikan Status Kehadiran Mahasiswa	100%	70	100%	23,5	100%	28
6. Mengubah Status Kehadiran Mahasiswa	100%	73,5	100%	40,5	100%	22,5
7. Memastikan Data Dosen Cocok Dengan Data Di Aplikasi	100%	18	0%	0	100%	36
8. Mengatasi Mahasiswa Yang Pindah Kelas	100%	121	100%	104	100%	155

Berdasarkan Tabel 6-4 Hasil Usability Study LFP Iterasi 3, bisa dilihat bahwa setiap kelompok pengguna memiliki preferensi prototipe masing – masing dalam *usability study* ini.

Partisipan dari kelompok pengguna mahasiswa merasa lebih cocok dengan prototipe C untuk melakukan *task* mereka. Hal itu terbukti dengan *success rate* untuk ketiga *task* yang hasilnya lebih tinggi (*success rate* 100% untuk *task* 1, *success rate* 100% untuk *task* 2, *success rate* 100% untuk *task* 3) dibanding prototipe A(*success rate* 80% untuk *task* 1, *success rate* 80% untuk *task* 2, *success rate* 100% untuk *task* 3), maupun prototipe B (*success rate* 80% untuk *task* 1, *success rate* 100% untuk *task* 2, *success rate* 100% untuk *task* 3).

Berdasarkan pemaparan kebanyakan partisipan dari kelompok pengguna mahasiswa, kelemahan Prototipe A adalah halaman *login* awal dimana secara naluriah partisipan akan mengisi *form login* meskipun berdasarkan alur prototipe seharusnya mahasiswa hanya perlu menekan tombol khusus mahasiswa.

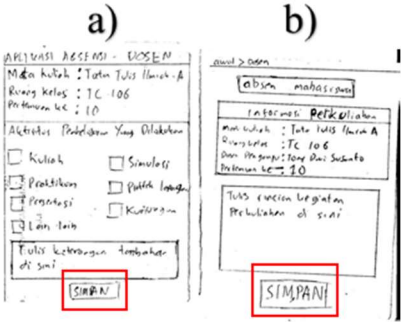


Gambar 6-7 Halaman awal pada prototipe B yang "menjebak" responden mahasiswa

Partisipan dari kelompok pengguna dosen merasa lebih cocok dengan prototipe B untuk melakukan *task* mereka. Hal itu terbukti dengan *success rate* untuk ketiga *task* yang hasilnya lebih tinggi (*success rate* 100% untuk *task* 4, *success rate* 100% untuk *task* 5, *success rate* 100% untuk *task* 6) dibanding prototipe A (*succes rate* 50% untuk *task* 4, *success rate* 100%

untuk *task* 5, *success rate* 100% untuk *task* 6), maupun prototipe C (*succes rate* 50% untuk *task* 4, *success rate* 100% untuk *task* 5, *success rate* 100% untuk *task* 6).

Berdasarkan interaksi yang terjadi pada *usability study*, ada partisipan gagal melakukan *task* 1 pada prototipe A maupun prototipe C karena posisi tombol yang kurang natural. Meskipun partisipan sudah diberi petunjuk untuk menekan tombol tersebut namun partisipan masih belum yakin soal fungsi dari tombol tersebut karena pemilihan kata untuk tombol yang kurang tepat sehingga partisipan menolak untuk menekan tombol tersebut.



Gambar 6-8 Interface tombol yang kurang sesuai untuk task 1 pada prototipe A (a) dan prototipe C (b)

Sementara itu, partisipan dari kelompok pengguna TU merasa lebih cocok dengan prototipe A. Baik prototipe A maupun prototipe C, keduanya sama – sama memiliki *success rate* 100% untuk kedua task, namun nilai *completion time* yang dimiliki oleh prototipe A (*completion time* 18 detik untuk task 1, *completion time* 121 detik untuk *task* 2) lebih baik dibandingkan nilai *completion time* prototipe C (*completion time* 36 detik untuk *task* 1, *completion time* 155 detik untuk *task* 2).

Berdasarkan hasil pengamatan pada sesi *usability study*, partisipan dari kelompok pengguna TU kesulitan untuk melakukan *task* 1 pada prototipe B karena alur yang diajukan di

prototipe B berbeda dengan 2 prototipe lainnya. Prototipe A dan prototipe C secara default memilih kelas mata kuliah sementara prototipe B mempersilakan pengguna untuk memilih kelas terlebih dahulu.

Sementara itu, memisahkan tampilan antara pengguna dari kelompok partisipan TU dengan kelompok partisipan lain, membuat partisipan bisa melakukan *task* lebih cepat. Hal tersebut bisa dibuktikan dengan nilai *completion time* prototipe A (*completion time* 18 detik untuk *task* 1, *completion time* 121 detik untuk *task* 2) yang lebih bagus daripada prototipe C (*completion time* 36 detik untuk *task* 1, *completion time* 155 detik untuk *task* 2).

Gambar 6-9 Tampilan utama untuk pengguna TU pada prototipe A (a), prototipe B (b), dan prototipe C (c)

Selain itu juga ada hasil kuosioner PSSUQ yang menggambarkan *user satisfaction*. Berikut ini hasil dari proses pengolahan data kuosioner PSSUQ :

Tabel 6-5 Hasil Kuosioner PSSUQ LFP Iterasi 3

	Prototipe A	Prototipe B	Prototipe C
Avg. Mahasiswa Satisfaction	3,0	2,9	2,5
Avg. Dosen Satisfaction	3,3	2,9	3,1
Avg. TU Satisfaction	2,4	3,1	3,3
Avg. Satisfaction	3,0	2,9	2,8

Hasil Kuosioner PSSUQ menunjukkan bahwa hasil *usability study* berbanding lurus dengan nilai *user satisfaction*. Meskipun hasil rerata total menunjukkan bahwa prototipe C memiliki nilai *user satisfaction* paling baik (*user satisfaction* 2,8) namun bukan berarti bahwa prototipe C adalah prototipe yang paling unggul dibanding prototipe lainnya. Sebagaimana hasil *usability study* yang dilakukan, setiap kelompok pengguna memiliki preferensi prototipe dalam mengerjakan *task* mereka.

6.1.4. Usability Study Medium Fidelity Prototyping Iterasi 0

Pada tahapan ini, ketiga kelompok pengguna dilibatkan pada *usability study*. Berikut ini hasil dari pengujian yang dilakukan:

Tabel 6-6 Hasil Usability Study MFP Iterasi 0

Task	Prototipe A			Prototipe B		
	Avg. Success Rate	*Avg. Error Per Task*	*Avg. Completion Time*	*Avg. Success Rate*	*Avg. Error Per Task*	*Avg. Completion Time*
1. Memeriksa Informasi Terkait Mata Kuliah	100%	3,2	68,2	100%	2,4	41,6
2. Melakukan Presensi	100%	0,0	53,8	100%	0,0	29,4
3. Melihat Batas Maksimal Absensi	100%	0,0	54,6	80%	0,4	38,5
4. Memperbarui Informasi Perkuliahan	100%	0,0	14,0	100%	0,0	13,0
5. Memastikan Status Kehadiran Mahasiswa	100%	2,0	21,0	100%	0,0	18,0
6. Mengubah Status Kehadiran Mahasiswa	100%	3,0	28,0	100%	1,0	113,0
7. Mengatasi Mahasiswa Pindah Kelas	100%	5,0	130,0	100%	1,0	101,0

Berdasarkan Tabel 6-6 Hasil Usability Study MFP Iterasi 0, bisa dilihat bahwa hasil *usability study* kali ini sedikit mirip dengan *usability study* yang dilakukan untuk *low fidelity*

prototype iterasi ke 3, dimana setiap kelompok pengguna memiliki preferensi prototipe masing – masing.

Berdasarkan *success rate* yang didapat, partisipan dari Kelompok pengguna mahasiswa lebih cocok dengan prototipe A (*succes rate* 100% untuk *task* 1, *success rate* 100% untuk *task* 2, *success rate* 100% untuk *task* 3) dibandingkan prototipe B (*success rate* 100% untuk *task* 1, *success rate* 100% untuk *task* 2, *success rate* untuk *task* 3). Namun jika dilihat berdasarkan *completion time*, prototipe B lebih unggul (*avg. completion time* 41,6 detik untuk *task* 1, *avg. completion time* 29,4 detik untuk *task* 2, *avg. completion time* 38,5 detik untuk *task* 3) jika dibandingkan dengan prototipe A (*avg. completion time* 68,2 detik untuk *task* 1, *avg. completion time* 53,8 detik untuk *task* 2, *avg. completion time* 54,6 detik untuk *task* 3).

Hal tersebut mungkin saja terjadi karena semua partisipan mendapat *pattern* yang sama, yaitu mencoba prototipe A terlebih dahulu dan setelah itu mencoba prototipe B. Dengan penggunaan prototipe A, prototipe B terasa lebih familiar yang berakibat terhadap meningkatnya nilai *completion time* untuk prototipe B dibandingkan prototipe A. Namun jika dilihat lebih teliti, ada beberapa deviasi yang terjadi pada *usability study* ini.

Untuk prototipe A ada beberapa partisipan dari kelompok pengguna mahasiswa yang membutuhkan waktu lebih dari 1 menit untuk melakukan *task*-nya (P2 pada *task* 1, P3 pada *task* 1, P3 pada *task* 2, dan P3 pada *task* 3). Berdasarkan data yang didapat dari *screen-capturing*, P3 seringkali mencoba untuk mengeksplor fungsi – fungsi aplikasi lainnya sebelum melakukan *task* yang dimaksud. Hal tersebut bisa dibuktikan dengan *completion time* P3 untuk prototipe B yang paling lama jika dibandingkan partisipan lainnya (*completion time* 60 detik untuk *task* 1, *completion time* 51 detik untuk *task* 2, dan gagal melakukan *task* 3). Untuk P2 pada *task* 1, *completion time* yang didapatkan melebihi 1 menit karena partisipan beberapa kali mencoba berinteraksi menggunakan tombol di hardware yang memang belum difasilitasi oleh prototipe ini.

Sementara untuk prototipe B hanya ada 1 partisipan dari kelompok pengguna mahasiswa yang membutuhkan waktu lebih dari 1 menit untuk melakukan *task*-nya (P1 pada *task* 3). Pada kasus ini, terjadi kesalahan prototipe dimana tampilan tabel rekap absensi menghilang sehingga partisipan terpaksa mengulangi untuk melakukan *work-around* agar bisa menyelesaikan *task* yang diberikan.

Catatan khusus untuk prototipe A, halaman *login* menyebabkan kebanyakan partisipan dari kelompok pengguna mahasiswa kesulitan melakukan *task* yang diberikan sehingga menghabiskan cukup banyak waktu hanya untuk menyelesaikan interaksi di halaman login.

Gambar 6-10 Halaman Login pada prototipe A

Untuk partisipan dari kelompok pengguna Dosen juga memiliki sedikit anomali, dimana prototipe B unggul di *task* 4 (*completion time* 13 detik, 0 *error per task*) dan *task* 5 (*completion time* 18 detik, 0 *error per task*) dan *task* 6 (*completion time* 113 detik, 1 *error per task*). namun prototipe

A unggul di *task* 6 berdasarkan *completion time* (*completion time* 28 detik, 3 *error per task*). Berdasarkan hasil *screen-capture*, partisipan secara tidak sengaja memilih menu untuk kelompok partisipan TU dan menghabiskan cukup banyak waktu agar bisa kembali ke tampilan awal dan masuk ke dalam menu untuk kelompok partisipan dosen.

Sementara itu kelompok partisipan TU bisa dikatakan lebih nyaman menggunakan prototipe B karena *completion time* (101,0 detik) dan *error per task* (1,0) nilainya lebih baik daripada prototipe A (*completion time* 130, 0 detik dan *error per task* 5,0).

Selain itu juga ada hasil kuosioner PSSUQ yang menggambarkan *user satisfaction*. Berikut ini hasil dari proses pengolahan data kuosioner PSSUQ :

Tabel 6-7 Hasil Kuosioner PSSUQ MFP Iterasi 0

	Prototipe A	Prototipe B
Avg. Mahasiswa Satisfaction	2,8	2,4
Avg. Dosen Satisfaction	2,7	3,2
Avg. TU Satisfaction	3,0	4,3
Avg. Satisfaction	2,8	2,8

Hasil Kuosioner PSSUQ menunjukkan bahwa rerata *user satisfaction* berimbang antara prototipe A (*Avg. Satisfaction* 2,8) dengan prototipe B (*Avg. Satisfaction* 2,8). Dengan P2 sebagai satu – satunya yang memberikan nilai di atas 4,0 untuk prototipe A dimana partisipan memaparkan bahwa prototipe A memiliki banyak masalah, mulai dari halaman *login*, penggunaan dan pemilihan warna, status kehadiran dan beberapa hal lainnya yang dirasa kurang terlihat seperti aplikasi android yang biasa digunakan.

6.1.5. Heuristic Evaluation Medium Fidelity Prototyping Iterasi 0

Berikut ini hasil dari *heuristic evaluation* yang dilakukan terhadap prototipe yang ada :

Tabel 6-8 Hasil Heuristic Evaluation MFP Iterasi 0

Severity Rating	*Heuristic Violation* Pada Prototipe A	*Heuristic Violation* Pada Prototipe B
0	0	0
1	14	10
2	30	25
3	18	18
4	5	1
Total	67	54

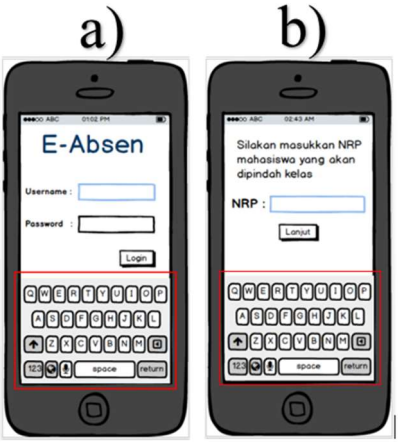
Seperti yang bisa dilihat pada hasil *heuristic evaluation*, prototipe A memiliki 67 *heuristic violation* dengan pembagian sebagai berikut: 5 *heuristic violation* dengan *severity rating* 4, 18 *heuristic violation* dengan *severity rating* 3, 30 *heuristic violation* dengan *severity rating* 2, 14 *heuristic violation* dengan *severity rating* 1, dan 0 *heuristic violation* dengan *severity rating* 0.

Sementara prototipe B memiliki 54 *heuristic violation* dengan pembagian sebagai berikut: 1 *heuristic violation* dengan *severity rating* 4, 18 *heuristic violation* dengan *severity rating* 3, 25 *heuristic violation* dengan *severity rating* 2, 10 *heuristic violation* dengan *severity rating* 1, dan 0 *heuristic violation* dengan *severity rating* 0.

Penyebab dari prototipe A memiliki 5 *heuristic violation* dengan *severity rating* 4 adalah tampilan pada halaman *login* mahasiswa, *breadcrumb* yang seharusnya tidak digunakan dan tampilan *keyboard on-screen* yang tidak sesuai.

Gambar 6-11 Penggunaan breadcrumb pada prototipe A

Sementara penyebab dari prototipe B memiliki 1 *heuristic violation* dengan *severity rating* 4 adalah *keyboard on-screen* yang tidak sesuai.



Gambar 6-12 Penggunaan keyboard virtual pada prototipe A (a), dan pada prototipe B (b)

Sementara untuk *heuristic violation* dengan *severity rating* 3 pada kedua prototipe kebanyakan disebabkan beberapa fitur UI yang tidak sesuai dengan kaidah HCI.

Sesuai dengan batasan iterasi MFP, dimana prototipe yang memiliki *heuristic violation* lebih dari 0 untuk *severity rating* 4 dan *heuristic violation* lebih dari 5 untuk *severity rating* 3 dianggap gagal, maka dilakukan beberapa perbaikan UI dengan fokus pada bagian UI yang menyebabkan adanya *heuristic violation* dengan *severity rating* 3 dan 4.

6.1.6. Usability Study Medium Fidelity Prototyping Iterasi 1

Pada tahapan ini, ketiga kelompok pengguna dilibatkan pada *usability study*. Berikut ini hasil dari pengujian yang dilakukan:

Tabel 6-9 Hasil Usability Study MFP Iterasi 1

Task	**Prototipe A**			**Prototipe B**		
	Avg. Success Rate	***Avg. Error Per Task***	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Error Per Task***	***Avg. Completion Time***
1. Memeriksa Informasi Terkait Mata Kuliah	80%	5,8	75,0	100%	0,2	27,8
2. Melakukan Presensi	100%	2,4	42,8	100%	0,2	23,2
3. Melihat Batas Maksimal Absensi	100%	1,4	70,0	60%	0,4	24,7
4. Memperbarui Informasi Perkuliahan	100%	0	27	100%	3	51
5. Memastikan Status Kehadiran Mahasiswa	100%	0	17	100%	2	13
6. Mengubah Status Kehadiran Mahasiswa	100%	0	26	100%	0	85
7. Mengatasi Mahasiswa Pindah Kelas	100%	6	179	100%	7	120

Berdasarkan Tabel 6-9 Hasil Usability Study MFP Iterasi 1, bisa dilihat bahwa tidak ada prototipe yang berhasil mencapai *success rate* 100% untuk kelompok pengguna mahasiswa (prototipe A : *success rate* 80% untuk *task* 1, *success rate* 100% untuk *task* 2, *success rate* 100% untuk *task* 3; sementara prototipe B : *success rate* 100% untuk *task* 1, *success rate* 100%

untuk *task* 2, *success rate* 60% untuk *task* 3). Jika dilihat dari *avg. completion time* dan *avg. error per task*, maka prototipe B lebih unggul dibandingkan dengan prototipe A. namun prototipe A lebih unggul dibandingkan prototipe B jika dilihat dari *success rate*.

Sementara untuk kelompok pengguna dosen, meskipun keduanya memiliki *success rate* 100%, namun terlihat bahwa kelompok pengguna dosen lebih cocok dengan prototipe A (*completion time* 27 detik untuk *task* 4, *completion time* 17 detik untuk *task* 5, *completion time* 26 detik untuk *task* 6) dibandingkan prototipe B (*completion time* 51 detik untuk *task* 4, *completion time* 13 detik untuk *task* 5, *completion time* 85 detik untuk *task* 6). Meskipun pada *task* 5 *completion time* yang didapat oleh prototipe B namun prototipe A mendapat nilai *avg. error per task* 0 sementara prototipe B mendapat nilai *avg. error per task* 2 dimana *error* yang terjadi karena *human error*.

Untuk kelompok pengguna TU, kedua prototipe berhasil mencapai *success rate* 100% dengan prototipe B unggul di *completion time* sementara prototipe A unggul di *avg. error per task.* Namun berdasarkan hasil *screen-capturing* yang dilakukan selama *usability study*, mayoritas *error* yang terjadi pada prototipe B karena kesalahan desain (3 *human error*, 4 kesalahan desain), sementara pada prototipe A didominasi oleh *human error* (4 *human error*, 2 kesalahan desain).

Selain itu juga ada hasil kuosioner PSSUQ yang menggambarkan *user satisfaction*. Berikut ini hasil dari proses pengolahan data kuosioner PSSUQ :

Tabel 6-10 Hasil Kuosioner PSSUQ MFP Iterasi 1

	Prototipe A	**Prototipe B**
Avg. Mahasiswa Satisfaction	3,4	2,1
Avg. Dosen Satisfaction	4,6	3,9
Avg. TU Satisfaction	4,2	4,3
Avg. Satisfaction	3,7	2,7

Sementara itu, hasil kuosioner PSSUQ menunjukkan bahwa secara keseluruhan partisipan merasa lebih senang menggunakan prototipe B dibanding prototipe A meskipun partisipan dari kelompok pengguna TU merasa sedikit lebih senang menggunakan prototipe A dibandingkan prototipe B. dari data di atas juga terlihat bahwa meskipun partisipan dari kelompok pengguna TU lebih suka menggunakan prototipe A dibandingkan prototipe B, namun nilai yang didapat di atas 4 jadi bisa dikatakan bahwa kedua prototipe yang diajukan masih belum memenuhi kepuasan partisipan. Sementara itu, partisipan dari kelompok pengguna dosen juga memberikan nilai di atas 4 untuk prototipe A.

6.1.7. Heuristic Evaluation Medium Fidelity Prototyping Iterasi 1

Berikut ini hasil dari *heuristic evaluation* yang dilakukan terhadap prototipe yang ada :

Tabel 6-11 Hasil Heuristic Evaluation MFP Iterasi 1

Severity Rating	*Heuristic Violation* Pada Prototipe A	*Heuristic Violation* Pada Prototipe B
0	3	3
1	5	7
2	34	27
3	9	10
4	0	0
Total	51	47

Seperti yang bisa dilihat pada hasil *heuristic evaluation*, prototipe A memiliki 51 *heuristic violation* dengan pembagian sebagai berikut: 0 *heuristic violation* dengan *severity rating* 4, 9 *heuristic violation* dengan *severity rating* 3, 34 *heuristic violation* dengan *severity rating* 2, 5 *heuristic violation* dengan *severity rating* 1, dan 3 *heuristic violation* dengan *severity rating* 0.

Sementara prototipe A memiliki 47 *heuristic violation* dengan pembagian sebagai berikut: 0 *heuristic violation* dengan *severity rating* 4, 10 *heuristic violation* dengan *severity rating* 3, 27 *heuristic violation* dengan *severity rating* 2, 7 *heuristic*

violation dengan *severity rating* 1, dan 3 *heuristic violation* dengan *severity rating* 0.

Penyebab dari prototipe A memiliki 9 *heuristic violation* dengan *severity rating* 3 adalah tampilan pada halaman *login* mahasiswa yang masih kurang baik, tidak adanya konfirmasi logout, inkonsistensi tampilan untuk konteks yang sama, penggunaan kata - kata yang berlainan untuk fungsi yang sama, alur UI yang kurang sesuai dan merepotkan pengguna, serta *breathing space* yang kurang.

Sementara penyebab dari prototipe B memiliki 10 *heuristic violation* dengan *severity rating* 3 adalah penggunaan kata yang ambigu, pemilihan ikon yang kurang tepat untuk fungsi yang ada pada UI, ukuran elemen UI yang kurang sesuai dengan kaidah HCI, serta *breathing space* yang kurang.

Sesuai dengan batasan iterasi MFP, dimana prototipe yang memiliki *heuristic violation* lebih dari 0 untuk *severity rating* 4 dan *heuristic violation* lebih dari 5 untuk *severity rating* 3 dianggap gagal, maka dilakukan beberapa perbaikan UI dengan fokus pada bagian UI yang menyebabkan *heuristic violation* dengan *severity rating* 3 dan 4.

6.1.8. Usability Study Medium Fidelity Prototyping Iterasi 2

Pada tahapan ini, ketiga kelompok pengguna dilibatkan pada *usability study*. Berikut ini hasil dari pengujian yang dilakukan:

Tabel 6-12 Hasil Usability Study MFP Iterasi 2

Task	**Prototipe A**			**Prototipe B**		
	Avg. Success Rate	***Avg. Error Per Task***	***Avg. Completion Time***	***Avg. Success Rate***	***Avg. Error Per Task***	***Avg. Completion Time***
1. Memeriksa Informasi Terkait Mata Kuliah	100%	2,6	104,6	100%	0,0	42,2
2. Melakukan Presensi	100%	1,4	70,6	100%	1,0	47,2

Task	Prototipe A			Prototipe B		
	Avg. Success Rate	*Avg. Error Per Task*	*Avg. Completion Time*	*Avg. Success Rate*	*Avg. Error Per Task*	*Avg. Completion Time*
3. Melihat Batas Maksimal Absensi	80%	0,8	85,75	100%	0,4	49,2
4. Memastikan Status Kehadiran Mahasiswa	100%	1,0	41,0	100%	1,0	86,0
5. Mengubah Status Kehadiran Mahasiswa	100%	0,0	17,0	100%	3,0	95,0
6. Mengubah Status Kehadiran Mahasiswa	100%	0,0	58,0	100%	8,0	88,0
7. Mengatasi Mahasiswa Pindah Kelas	100%	11,0	99,0	100%	0,0	53,0

Berdasarkan Tabel 6-12 Hasil Usability Study MFP Iterasi 2, bisa disimpulkan bahwa partisipan dari kelompok pengguna dosen merasa lebih cocok menggunakan prototipe A (*avg. completion time* 41,0 detik untuk *task* 5, *avg. completion time* 17,0 detik untuk *task* 6) daripada prototipe B (*avg. completion time* 86,0 detik untuk *task* 5, *avg. completion time* 95,0 detik untuk *task* 6). Sementara itu partisipan dari kelompok pengguna mahasiswa merasa lebih cocok menggunakan prototipe B daripada prototipe A, terbukti dari *success rate*, *avg. error per task*, dan *avg. completion time* prototipe B yang lebih unggul dibanding Prototipe A.

Sementara itu, meskipun kedua prototipe mendapat *success rate* 100%, untuk partisipan dari kelompok pengguna TU ada anomali. Untuk *task* 6, prototipe A lebih unggul dalam hal *avg. error per task* dan *completion time* dibanding prototipe B. Untuk *task* 7, prototipe B lebih unggul dalam hal *avg. error per task* dan *completion time* dibanding prototipe A. Berdasarkan hasil *screen-capture*, kekurangan dari prototipe B adalah fitur ubah status mahasiswa letaknya tersembunyi dalam obyek *expandable-card* yang pada awalnya partisipan menganggap bahwa obyek itu hanya berupa tampilan statis yang tidak bisa diapa – apakan. Sementara itu, kelemahan dari prototipe A

adalah proses pemindahan kelas dilakukan dengan cara tradisional (pilih kelas ➔ tambah mahasiswa ➔ ganti kelas ➔ hapus mahasiswa ➔ selesai) sedangkan prototipe B menggunakan cara yang lebih cepat (nrp mahasiswa ➔ daftar kelas ➔ pilih kelas ➔ selesai).

Gambar 6-13 Fitur ubah status mahasiswa pada prototipe B

Selain itu juga ada hasil kuosioner PSSUQ yang menggambarkan *user satisfaction*. Berikut ini hasil dari proses pengolahan data kuosioner PSSUQ :

Tabel 6-13 Hasil Kuosioner PSSUQ MFP Iterasi 2

	Prototipe A	Prototipe B
Avg. Mahasiswa Satisfaction	2,6	2,7
Avg. Dosen Satisfaction	2,6	2,9
Avg. TU Satisfaction	4,3	3,2
Avg. Satisfaction	2,8	2,8

menunjukkan hasil y *Avg. satisfaction* ang berimbang untuk kedua prototipe (2,8 untuk prototipe A dan 2,8 untuk prototipe B).

Ada sedikit deviasi terhadap nilai *avg. satisfaction* untuk kelompok pengguna mahasiswa. Hasil *usability study* menunjukkan bahwa partisipan lebih menyukai prototipe A dibandingkan prototipe B dengan selisih nilai 0,1. Padahal berdasarkan hasil 3 matrik lainnya menunjukkan bahwa prototipe B lebih unggul dibandingkan prototipe A. Hal ini bisa diterjemahkan bahwa partisipan kurang suka menggunakan aplikasi dengan model *tab layout* yang digunakan pada prototipe B.

Sementara untuk kelompok pengguna dosen, hasil kuosioner PSSUQ mendukung hasil *usability study* yang dilakukan karena keempat matriks yang digunakan menunjukkan bahwa partisipan lebih menyukai prototipe A dibandingkan prototipe B.

Untuk kelompok pengguna TU, meskipun hasil *usability study* terbelah karena prototipe A unggul di *task* 6 namun prototipe B unggul di *task* 7, hasil kuosioner PSSUQ menunjukkan bahwa partisipan lebih menyukai prototipe B.

6.1.9. Heuristic Evaluation Medium Fidelity Prototyping Iterasi 2

Berikut ini hasil dari *heuristic evaluation* yang dilakukan terhadap prototipe yang ada :

Tabel 6-14 Hasil Heuristic Evaluation MFP Iterasi 2

Severity Rating	*Heuristic Violation* Pada Prototipe A	*Heuristic Violation* Pada Prototipe B
0	2	3
1	8	9
2	24	23
3	12	3
4	0	0
Total	46	38

Seperti yang bisa dilihat pada hasil *heuristic evaluation*, prototipe A memiliki 46 *heuristic violation* dengan pembagian sebagai berikut: 0 *heuristic violation* dengan *severity rating* 4, 12 *heuristic violation* dengan *severity rating* 3, 24 *heuristic violation* dengan *severity rating* 2, 8 *heuristic violation* dengan

severity rating 1, dan 2 *heuristic violation* dengan *severity rating* 0.

Sementara prototipe A memiliki 38 *heuristic violation* dengan pembagian sebagai berikut: 0 *heuristic violation* dengan *severity rating* 4, 3 *heuristic violation* dengan *severity rating* 3, 23 *heuristic violation* dengan *severity rating* 2, 9 *heuristic violation* dengan *severity rating* 1, dan 3 *heuristic violation* dengan *severity rating* 0.

Penyebab dari prototipe A memiliki 12 *heuristic violation* dengan *severity rating* 3 adalah tampilan pada halaman *login* mahasiswa yang masih kurang baik, alur UI yang masih bisa disederhanakan lagi, pemilihan kata yang kurang tepat maupun inkonsisten pada beberapa tampilan UI, adanya informasi yang dibutuhkan oleh pengguna namun tidak ditampilkan, serta *breathing space* yang kurang.

Sementara penyebab dari prototipe B memiliki 9 *heuristic violation* dengan *severity rating* 3 adalah adanya elemen navigasi yang tidak membawa pengguna kemanapun, elemen UI yang terlalu berimpit sehingga berpotensi menyebabkan pengguna salah klik, serta pemilihan kata yang kurang tepat.

Sesuai dengan batasan iterasi MFP, dimana prototipe yang memiliki *heuristic violation* lebih dari 0 untuk *severity rating* 4 dan *heuristic violation* lebih dari 5 untuk *severity rating* 3 dianggap gagal, maka pembuatan desain UI dilanjutkan ke tahap selanjutnya yaitu *high fidelity prototyping* dikarenakan prototipe B memiliki nilai yang cukup baik pada *heuristic evaluation*.

6.1.10. Heuristic Evaluation High Fidelity Prototyping Iterasi 0

Berikut ini hasil dari *heuristic evaluation* yang dilakukan terhadap prototipe yang ada :

Tabel 6-15 Hasil Heuristic Evaluation HFP Iterasi 0

Severity Rating	*Heuristic Violation* Pada Prototipe A
0	0
1	21
2	15
3	2
4	0
Total	38

Seperti yang bisa dilihat pada Tabel 6-15 Hasil Heuristic Evaluation HFP Iterasi 0, prototipe UI yang ada memiliki 38 *heuristic violation* dengan pembagian sebagai berikut: 0 *heuristic violation* dengan *severity rating* 4, 2 *heuristic violation* dengan *severity rating* 3, 15 *heuristic violation* dengan *severity rating* 2, 21 *heuristic violation* dengan *severity rating* 1, dan 0 *heuristic violation* dengan *severity rating* 0.

Penyebab dari prototipe UI memiliki 2 *heuristic violation* dengan *severity rating* 3 adalah pemilihan kata dan pemilihan ikon yang kurang tepat untuk fungsi yang ada pada UI.

Sesuai dengan batasan iterasi HFP, dimana prototipe yang memiliki *heuristic violation* lebih dari 0 untuk *severity rating* 4 dan *heuristic violation* lebih dari 0 untuk *severity rating* 3 dianggap gagal, maka dilakukan beberapa perbaikan UI dengan fokus pada bagian UI yang menyebabkan *heuristic violation* dengan *severity rating* 3 dan 4.

6.1.11. Heuristic Evaluation High Fidelity Prototyping Iterasi 1

Berikut ini hasil dari *heuristic evaluation* yang dilakukan terhadap prototipe yang ada :

Tabel 6-16 Hasil Heuristic Evaluation HFP Iterasi 1

Severity Rating	Heuristic Violation Pada Prototipe A
0	0
1	16
2	14
3	0
4	0
Total	30

Seperti yang bisa dilihat pada Tabel 6-16 Hasil Heuristic Evaluation HFP Iterasi 1, prototipe UI yang ada memiliki 30 *heuristic violation* dengan pembagian sebagai berikut: 0 *heuristic violation* dengan *severity rating* 4, 0 *heuristic violation* dengan *severity rating* 3, 14 *heuristic violation* dengan *severity rating* 2, 16 *heuristic violation* dengan *severity rating* 1, dan 0 *heuristic violation* dengan *severity rating* 0.

Sesuai dengan batasan iterasi HFP, dimana prototipe yang memiliki *heuristic violation* lebih dari 0 untuk *severity rating* 4 dan *heuristic violation* lebih dari 0 untuk *severity rating* 3 dianggap gagal, maka tahapan pembuatan desain UI dihentikan dan pengerjaan tugas akhir dilanjutkan ke tahapan pembuatan desain sistem.

6.2. Validasi Desain Sistem

Validasi desain sistem dilakukan dengan menggunakan *requirement traceability matrix* dimana hasil akhir *use case*, prototipe UI, *robustness analysis* dan *sequence diagram* dipetakan ke dalam *functional requirement* / kebutuhan fungsional yang ada untuk memastikan bahwa semua kebutuhan fungsional telah terpenuhi.

6.2.1. Requirement Traceability Matrix

Tabel 6-17 Requirement Traceability Matrix

Kode FR	*Functional Requirement*	Prototipe UI	*Use Case*	*Robustness Analysis*	*Sequence Diagram*
FR-101	Mahasiswa bisa melakukan presensi / pencatatan kehadiran	UI001, UI101, UI102, UI103, UI104, UI105, UI106	UC-101 [Invoke]	RA101 [invoke]	SD101 [Invoke]
			UC-102	RA102	SD102
FR-102		UI001, UI101, UI102,	UC-101 [Invoke]	RA101 [Invoke]	SD101 [Invoke]

Kode FR	*Functional Requirement*	Prototipe UI	*Use Case*	*Robustness Analysis*	*Sequence Diagram*
	Mahasiswa bisa melihat data ketidakhadirannya	UI103, UI104, UI105, UI107, UI108, UI109, UI110, UI111	UC-103	RA103	SD103
FR-103	Mahasiswa bisa melihat daftar mahasiswa yang satu kelas mata kuliah dengan dirinya	UI001, UI101, UI102, UI103, UI104, UI107, UI112, UI113, UI114, UI115	UC-101 [Invoke]	RA101 [Invoke]	SD101 [Invoke]
			UC-104	RA104	SD104
FR-104	Mahasiswa bisa melihat informasi kelas mata kuliah yang sedang diikuti	UI001, UI101, UI102, UI103, UI104, UI105	UC-101	RA101	SD101
FR-201	Dosen bisa melihat daftar mahasiswa pada kelas mata kuliah yang sedang ia ajar	UI001, UI204, UI205, UI206, UI209, UI210, UI211	UC-201	RA201	SD201
FR-202	Dosen bisa mengubah status kehadiran mahasiswa	UI001, UI201, UI202, UI203, UI204, UI205, UI206, UI207, UI208, UI209	UC-201 [Invoke]	RA201 [Invoke]	SD201 [Invoke]
			UC-202	RA202	SD202

Kode FR	*Functional Requirement*	Prototipe UI	*Use Case*	*Robustness Analysis*	*Sequence Diagram*
FR-203	Dosen bisa memilih kelas mana yang akan dilakukan proses pencatatan kehadiran berdasarkan daftar kelas yang berisi mata kuliah yang ia ajar	UI001, UI201, UI202, UI203, UI204, UI213, UI216, UI217, UI218, UI219, UI220, UI221	UC-201 [Invoke]	RA201 [Invoke]	SD201 [Invoke]
			UC-204	RA204	SD204
FR-204	Dosen bisa melihat informasi kelas mata kuliah yang sedang ia ajar	UI001, UI201, UI202, UI203, UI204, UI212, UI213, UI215	UC-201 [Invoke]	RA201 [Invoke]	SD201 [Invoke]
			UC-203	RA203	SD203
FR-301	Pegawai TU bisa memasukkan data mahasiswa ke dalam kelas mata kuliah yang belum diambilnya	UI001, UI301, UI302, UI303, UI304, UI306, UI307, UI308, UI310, UI313, UI322, UI328, UI334, UI335, UI336, UI337, UI338, UI339	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-302 [Invoke]	RA302 [Invoke]	SD302 [Invoke]
			UC-305	RA305	SD305
FR-302	Pegawai TU bisa menghapus data mahasiswa dari dalam kelas mata	UI001, UI301, UI302, UI303, UI304, UI306,	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-302 [Invoke]	RA302 [Invoke]	SD302 [Invoke]

Kode FR	*Functional Requirement*	Prototipe UI	*Use Case*	*Robustness Analysis*	*Sequence Diagram*
	kuliah yang sedang diambilnya	UI307, UI308, UI310, UI313, UI322, UI328, UI329	UC-307	RA307	SD307
FR-303	Pegawai TU bisa memindahkan mahasiswa dari kelas mata kuliah yang diambil oleh mahasiswa ke kelas lain	UI001, UI301, UI302, UI303, UI304, UI306, UI307, UI308, UI310, UI313, UI322, UI328, UI330, UI331, UI332, UI333	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-302 [Invoke]	RA302 [Invoke]	SD302 [Invoke]
			UC-306	RA306	SD306
FR-304	Pegawai TU bisa mengubah status kehadiran mahasiswa	UI001, UI301, UI302, UI303, UI304, UI306, UI307, UI308, UI310, UI313, UI322, UI322, UI323, UI324, UI326, UI327	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-302 [Invoke]	RA302 [Invoke]	SD302 [Invoke]
			UC-303	RA303	SD303
FR-305	Pegawai TU bisa menambahkan data mahasiswa baru	UI001, UI301, UI302, UI303, UI304,	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-308	RA308	SD308

Kode FR	*Functional Requirement*	Prototipe UI	*Use Case*	*Robustness Analysis*	*Sequence Diagram*
		UI306, UI315, UI316, UI317, UI318, UI319, UI320, UI321			
FR-306	Pegawai TU bisa menghapus data mahasiswa	UI001, UI301, UI302, UI303, UI304, UI306, UI307, UI308, UI310, UI313, UI322, UI309, UI325	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-302 [Invoke]	RA302 [Invoke]	SD302 [Invoke]
			UC-304	RA304	SD304
FR-307	Pegawai TU bisa menambahkan kelas mata kuliah baru	UI001, UI301, UI302, UI303, UI304, UI306, UI310, UI314	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-309	RA309	SD309
FR-308	Pegawai TU bisa memilih direktori pada handset android yang digunakan untuk memuat template absensi	UI001, UI301, UI302, UI303, UI304, UI306, UI341, UI342	UC-301 [Invoke]	RA301 [Invoke]	SD301 [Invoke]
			UC-310	RA310	SD310

Seperti yang bisa dilihat pada Tabel 6-17 Requirement Traceability Matrix, ada beberapa kode tampilan prototipe UI yang dimasukkan ke lebih dari 1 kebutuhan fungsional. Hal

tersebut disebabkan karena beberapa UI dibutuhkan sebelum mengakses tampilan yang berhubungan langsung dengan kebutuhan fungsional yang ada, misal : halaman *login* dosen dibutuhkan untuk diakses oleh empat *use case*, karena pengguna harus mengakses halaman tersebut sebelum bisa melakukan kebutuhan fungsional.

Selain itu juga ada beberapa kode tampilan prototipe yang tidak dimasukkan sama sekali ke dalam kebutuhan fungsional yang ada, diantaranya : UI214 dan UI 305. Untuk hal tersebut ada alasan tertentu kenapa tampilan tampilan tersebut tidak dimasukkan ke RTM, yaitu karena mereka tidak memiliki padanan di kebutuhan fungsional. Namun bukan berarti tampilan prototipe tersebut tidak diperlukan. Ambil contoh untuk UI214 & UI305, kedua tampilan itu ada karena kaidah heuristik "*visibility of system status*" subheuristik nomer 1 dan kaidah heuristik "*user control and freedom*" subheuristik nomer 62 membutuhkan adanya kedua tampilan tersebut.

Selain itu juga ada beberapa *functional requirement* / kebutuhan fungsional yang memiliki lebih dari 1 *use case* yang dilampirkan. Untuk *use case* yang diberi tanda tambahan "[Invoke]" berarti *use case* itu dipanggil dalam proses kebutuhan fungsional yang dilakukan oleh pengguna. Atau dengan kata lain, untuk melakukan *use case* yang berkaitan langsung dengan kebutuhan fungsional, pengguna perlu melakukan *use case* yang diberi tanda "[Invoke]" terlebih dahulu.

Untuk *robustness analysis*, berhubung fungsi dari *robustness analysis* adalah untuk menguji apakah *use case* yang ada sudah *robust* atau belum, maka ada lebih dari 1 *robustness analysis* yang dilampirkan untuk beberapa kebutuhan fungsional yang ada karena mendampingi *use case* yang berkorelasi.

Untuk *sequence diagram* alasan kenapa bisa ada lebih dari 1 *sequence diagram* yang dilampirkan untuk 1 kebutuhan fungsional hampir sama dengan *robustness analysis*, yaitu

karena fungsi dari *sequence diagram* adalah menggambarkan proses yang terjadi pada *use case* sehingga *sequence diagram* dilampirkan sesuai dengan *use case* yang ada.

BAB VII
PENUTUP

Bab ini berisi tentang simpulan dari keseluruhan tugas akhir dan saran maupun rekomendasi terhadap penelitian tugas akhir ini untuk penelitian lanjutan.

7.1. Kesimpulan

Dari pelaksanaan tugas akhir ini didapatkan kesimpulan sebagai berikut:

1. Metode *Iterative Parallel Prototyping* bisa digunakan dalam perancangan aplikasi pencatatan kehadiran berbasis android untuk studi kasus di jurusan sistem informasi ITS. hal ini bisa dilihat dari hasil *Usability Study* pada tahapan MFP iterasi ke 0 yang menunjukkan nilai *avg. error per task* yang menurun serta *completion time* yang lebih cepat jika dibandingkan dengan MFP iterasi ke 3 meskipun jumlah kelompok responden yang dilibatkan (3 kelompok responden) dan jumlah task yang dilakukan sama (7 task). Maupun hasil *Heuristic Evaluation* dari HFP 1 yang menunjukan tidak adanya *heuristic violation* dengan *severity rating* 3 ataupun *severity rating* 4 jika dibandingkan dengan MFP 0 yang menunjukkan adanya *heuristic violation* dengan *severity rating* 3 (18 *heuristic violation* pada prototipe A dan 18 *heuristic violation* pada prototipe B) serta *heuristic violation* dengan *severity rating* 4 (5 *heuristic violation* pada prototipe A dan 1 *heuristic violation* pada prototipe B).
2. Dalam perancangan desain sistem, meskipun sedikit terhambat pada tahapan *technical architecture*, penggunaan ICONIX Process cukup membantu untuk menemukan *class* yang digunakan dalam pembuatan baris kode program. Bisa dilihat dari *domain model* yang pada akhirnya berevolusi menjadi *class diagram*

setelah melalui beberapa tahapan hingga akhirnya menjadi baris kode seperti yang dicontohkan pada bab V.

7.2. Saran

Untuk pengembangan tugas akhir ini agar ke depannya lebih baik dan lebih bermanfaat, maka terdapat beberapa saran yang dapat dipertimbangkan, yaitu:

1. Perlu diadakan penelitian lebih lanjut terkait batasan dari sebuah prototipe UI untuk disebut sebagai "*Paper and Pen Prototype*", "*Wireframe*", "*Mock Up*" agar proses pembuatan desain UI lebih valid.
2. Perlu ditambahkan metode untuk penarikan kesimpulan dari hasil PSSUQ sehingga matrik kepuasan pengguna bisa lebih valid.
3. Pengujian *Heuristic Evaluation* sebaiknya melibatkan pakar UI agar hasil yang didapat lebih valid.
4. Diperlukan adanya penelitian terkait pemetaan ICONIX Process terhadap *framework* android, terutama pada bagian *technical architecture*.
5. Lingkup studi kasus bisa diperluas lagi agar dampak dari penggunaan metode yang digunakan bisa lebih representatif.
6. Perlu diadakan penelitian lebih lanjut dengan tahapan ICONIX secara menyeluruh agar hasil perancangan lebih valid.
7. Diperlukan adanya studi silang dengan bahasan keamanan infrastuktur teknologi informasi mengingat data yang diproses cukup penting.

DAFTAR PUSTAKA

[1] R. Fakhruddin and A. Harsoyo, "Implementasi Portable Smart Card Reader Untuk Absensi," in *Prosiding Konferensi Nasional Teknologi Informasi & Komunikasi untuk Indonesia*, Bandung, 2006.

[2] L. Rajasekar and S. Vivek, "Wireless Fingerprint Attendance System using ZigBee Technology," *Wireless Fingerprint Attendance System using,* vol. III, no. 1, pp. 118-121, Jan-Mar 2012.

[3] M. J. L. Fernández, J. G. Fernández, S. R. Aguilar, B. S. Selvi and R. G. Crespo, "Control of attendance applied in higher education through mobile NFC technologies," *Expert Systems with Applications,* vol. 40, no. 11, pp. 4478-4489, 1 September 2013.

[4] R. Joshi, V. V. Shete and S. B. Somani, "Android Based Smart Learning and Attendance Management System," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. IV, no. 6, Juni 2016.

[5] A. A. A. Rahni, N. Zainal, M. F. Z. Adna, N. E. Othman and M. F. Bukhori, "Development Of The Online Student Attendance Monitoring System (SAMS) Based On QR-Codes And Mobile Devices," *Journal of Engineering Science and Technology,* no. 2, pp. 28-40, Juni 2015.

[6] N. Saparkhojayev, E. Shakhov and Y. Mailybayev, "Mobile Attendance Checking System on Android Platform for Kazakhstani University," *Journal of Physics: Conference Series 710,* 2016.

[7] Nandakumar, "Why is the Android OS So Popular?," 25 Februari 2016. [Online]. Available: http://opensourceforu.com/2016/02/why-is-the-android-os-so-popular/. [Accessed 14 Oktober 2016].

[8] "5 Reasons why your UI Designer should design for Users," 22 Oktober 2014. [Online]. Available: https://usio.com/blog/5-reasons-why-your-ui-designer-should-design-for-users/. [Accessed 14 Oktober 2016].

[9] J. Raskin, "The Humane Touch: Bad Design Can Be Costly," 28 Mei 2001. [Online]. Available: http://www.forbes.com/asap/2001/0528/016a.html. [Accessed 15 Oktober 2016].

[10] D. Pogue, "5 of the Worst User-Interface Disasters," 1 April 2016. [Online]. Available: https://www.scientificamerican.com/article/pogue-5-of-the-worst-user-interface-disasters/. [Accessed 15 Oktober 2016].

[11] J. Shariat, 9 Oktober 2014. [Online]. Available: https://medium.com/tragic-design/how-bad-ux-killed-jenny-ef915419879e#.ias44ehn6. [Accessed 15 Oktober 2016].

[12] S. P. Dow, K. Heddleston and S. R. Klemmer, "The Efficacy of Prototyping Under Time Constraints," 20 Oktober 2009.

[13] S. P. Dow, A. Glassco, J. Kass, M. Schwarz, D. L. Schwartz and S. R. Klemmer, "Parallel Prototyping Leads to Better Design Results, More Divergence, and Increased Self-Efficacy," *ACM Transactions on Computer-Human Interaction,* vol. Vol. 17, no. No. 4, Desember 2010.

[14] D. Rosenberg, M. Collins-Cope and M. Stephens, Agile Development with ICONIX Process: People, Process, and Pragmatism, 1st ed., J. Sunser, Ed., New York: Springer-Verlag, 2005, p. 41.

[15] [Online]. Available: http://bundymuseum.org/site3/about/the-history/willard-bundy-bio/. [Accessed 18 Oktober 2016].

[16] "Magnetic Stripe Technology," [Online]. Available: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/magnetic/. [Accessed 18 Oktober 2016].

[17] C. Chapman, "The Evolution of Web Design," 28 November 2009. [Online]. Available: http://sixrevisions.com/web_design/the-evolution-of-web-design/. [Accessed 18 Oktober 2016].

[18] S. Mayhew, "History of Biometrics," 14 Januari 2015. [Online]. Available: http://www.biometricupdate.com/201501/history-of-biometrics. [Accessed 18 Oktober 2016].

[19] Open Handset Alliance, "Industry Leaders Announce Open Platform for Mobile Devices | Open Handset Alliance," 5 November 2007. [Online]. Available: http://www.openhandsetalliance.com/press_110507.html. [Accessed 26 Januari 2015].

[20] R. Amadeo, "Google's iron grip on Android: Controlling open source by any means necessary," 21 October 2013. [Online]. Available: http://arstechnica.com/gadgets/2013/10/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/. [Accessed 27 Januari 2015].

[21] "User Interface Design Basic," [Online]. Available: https://www.usability.gov/what-and-why/user-interface-design.html. [Accessed 18 Oktober 016].

[22] [Online]. Available: https://material.google.com/#introduction-principles. [Accessed 18 Oktoboer 2016].

[23] J. Nielsen, "Parallel & Iterative Design + Competitive Testing = High Usability," 18 January 2011. [Online]. Available: https://www.nngroup.com/articles/parallel-and-iterative-design/. [Accessed 10 July 2017].

[24] J. Nielsen, "A/B Testing, Usability Engineering, Radical Innovation: What Pays Best?," 26 Maret 2012. [Online]. Available: http://www.nngroup.com/articles/ab-testing-usability-engineering/. [Accessed 4 Oktober 2016].

[25] J. Nielsen, "Usability Metrics," 21 Januari 2001. [Online]. Available: https://www.nngroup.com/articles/usability-metrics/. [Accessed 4 Oktober 2016].

[26] J. Nielsen, "How-To: Article by Jakob Nielsen," 1 Januari 1995. [Online]. Available: http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/. [Accessed 8 Mei 2015].

[27] A. Evans, "Example Usability Test with a Paper Prototype," 2010. [Online]. Available: http://ed.ted.com/on/JK0cWO9o. [Accessed 11 July 2017].

[28] A. Rautela, "Post-Study System Usability Questionnaire (PSSUQ)," 10 Desember 2010. [Online]. Available: http://www.conetrees.com/2010/12/ux-glossary/post-study-system-usability-questionnaire-pssuq/. [Accessed 4 Oktober 2016].

[29] R. Y. Gómez, D. C. Caballero and J.-L. Sevillano, "Heuristic Evaluation on Mobile Interfaces: A New Checklist," 2014.

[30] E. Wong, "Heuristic Evaluation: How to Conduct a Heuristic Evaluation," Interaction Design FOundation, July 2017. [Online]. Available: https://www.interaction-design.org/literature/article/heuristic-evaluation-how-to-conduct-a-heuristic-evaluation. [Accessed 11 July 2017].

[31] J. Nielsen, "Severity Ratings for Usability Problems," Nielsen Norman Group, 1 Januari 1995. [Online]. Available: https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/. [Accessed 26 Mei 2017].

[32] Developer Mengajar, "Android Module #1: Activity," 2 July 2015. [Online]. Available: https://blog.dicoding.com/android-module-1-activity/. [Accessed 16 May 2017].

[33] "Dialogs," [Online]. Available: https://developer.android.com/guide/topics/ui/dialogs.html. [Accessed 16 May 2017].

[34] "Toast," [Online]. Available: https://developer.android.com/reference/android/widget/Toast.html. [Accessed 16 May 2017].

[35] "Snackbar," [Online]. Available: https://developer.android.com/reference/android/support/design/widget/Snackbar.html. [Accessed 16 May 2017].

[36] "Camera API," [Online]. Available: https://developer.android.com/guide/topics/media/camera.html. [Accessed 16 May 2017].

[37] J. Nielsen, "10 Heuristics for User Interface Design: Article by Jakob Nielsen," 1 Januari 1995. [Online]. Available: http://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 31 Maret 2015].

[38] Payload Media, "An Overview of the Android Architecture - Techotopia," 3 July 2014. [Online]. Available: http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture. [Accessed 17 Februari 2015].

[39] Opera Software ASA, "Over 30 million Indonesians go online with Opera Mini," 9 October 2014. [Online]. Available: http://www.operasoftware.com/press/releases/mobile/2014-10-09. [Accessed 19 Februari 2015].

[40] REPUBLIKA.CO.ID, "Pengguna Smartphone Indonesia Peringkat Kelima Dunia | Republika Online," 02 November 2014. [Online]. Available: http://www.republika.co.id/berita/trendtek/gadget/14/11/02/neehfh-pengguna-smartphone-indonesia-peringkat-kelima-dunia. [Accessed 01 Februari 2015].

[41] Statista, "Smartphones sales by operating system worldwide 2009 - 2014 | Statistic," [Online]. Available:

http://www.statista.com/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/. [Accessed 29 Desember 2014].

[42] H. Gupta, P. Agrawal, A. Jindal and D. Gautan. [Online]. Available: http://www.cse.iitd.ac.in/~cs5090250/harsh/Publication1.pdf. [Accessed 7 April 2015].

[43] A. K. Jain, P. Flynn and A. A. Ross, Handbook of Biometrics, 1 ed., New York: Springer US, 2007.

[44] N. N. Shahade, P. A. Kawade and S. L. Thombare, "Student Attendance Tracker System in Android," *International Journal For Engineering Applications And Technology,* vol. Vol. I, no. Issue 4, pp. 119-124, Februari 2014.

[45] P. Elakiyaselvi, U. R. V. Nandhiny and M. B. Bose, "A Framework for Implementing e-Attendance System Using Near Field Communication in Android OS," *International Journal of Computer Technology and Applications,* vol. Vol. 3, no. Issue 2, pp. 854-858, Maret - April 2012.

[46] P. J. Ebenezer, M. R. Muralidharan, S. Srikanth, E. Ramesh and M. S. Prabhu, "Android Application for Student Activity Register," *IJREAT International Journal of Research in Engineering & Advanced Technology,* vol. Volume 2, no. Issue 2, 2014.

[47] P. Agrawal, A. Jindal, H. Gupta and D. Gautam. [Online]. Available: www.cse.iitd.ernet.in/~cs5090250/harsh/CSP315_presentations.pptx. [Accessed 22 Februari 2015].

[48] S. S. Chawhan, M. P. Girhale and G. Mankar, "Mobile Phone Based Attendance System," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. Volume 10, no. Issue 3, pp. 48-50, 2013.

[49] H. Gupta, P. Agrawal, A. Jindal and D. Gautam, 2011. [Online]. Available: https://sites.google.com/site/csp315attend/. [Accessed 7 April 2015].

[50] D. Cahyadi, "Desain Sistem Absensi PNS Berbasis Teknologi RFID," *Jurnal Informatika Mulawarman,* vol. Vol. 4, no. No. 3, pp. 29 - 36, September 2009.

[51] A. O. T., O. A., F. O.A. and O. O. M., "RFID-Based Students Attendance Management System," *International Journal of Scientific & Engineering Research,* vol. Volume 4, no. Issue 2, Februari 2013.

[52] M. A. P. and W. Yuwono, "Rancang Bangun Sistem Informasi Absensi Perkuliahan PJJ".

[53] S. R. REDDY, D. GOYAL and A. BANSAL, "Mobile Based Attendance Management System".

[54] G. Shanbhag, H. Jivani and S. Shahi, "Mobile Based Attendance Marking System Using Android And

Biometrics," *IJIRST–International Journal for Innovative Research in Science & Technology,* vol. Volume 1, no. Issue 1, pp. 87-90, Juni 2014.

[55] S. Seine Darusatya Jotex, "Pembuatan Aplikasi Sistem Absensi Pendukung Mesin Kombinasi Sidik Jari dan Barcode dari TDX 2500".

[56] X. Ducrohet, "Android 2.3 Platform and Updated SDK Tools | Android Developers Blog," 6 Desember 2010. [Online]. Available: http://android-developers.blogspot.com/2010/12/android-23-platform-and-updated-sdk.html. [Accessed 7 Mei 2015].

[57] C. Z. H.-Y. S. Jingdong Wang, "Face Image Resolution Versus Face Recognition Performance Based on Two Global Methods," *Asian Conference on Computer Vision (ACCV04),* 2004.

[58] J. A. Landay, 2014. [Online]. Available: http://hci.stanford.edu/courses/cs147/2014/au/lectures/12-heuristic-evaluation.pdf. [Accessed 9 Mei 2015].

[59] Z. Musyawamah, Sistem Informasi Absensi dan Penggajian Menggunakan Teknologi Scaner BR-300 Study Kasus di PT. Ganesha E-Commerce Solution, Bandung, 2013.

BIODATA PENULIS

Penulis dilahirkan di Surabaya, 2 November 1993. Penulis telah menempuh pendidikan formal di SD Khadijah 3, SMP Negeri 2 Surabaya, SMA Negeri 9 Surabaya, serta Politeknik Komputer Bina Profesi Indonesia Jurusan Desainer Multimedia. Setelah lulus dari sekolah menengah atas, penulis meneruskan pendidikan di Jurusan Sistem Informasi, Institut Teknologi Sepuluh Nopember, pada tahun 2011 terdaftar dengan NRP 5211100061. Di Jurusan Sistem Informasi penulis mengambil bidang studi Akusisi Data dan Diseminasi Informasi (ADDI). Dalam bidang non-akademik, penulis pernah mengikuti organisasi mahasiswa serta menjadi pengurus inti dari Unit Kegiatan Mahasiswa Ju-Jitsu ITS. Pada pengerjaan Tugas Akhir di Jurusan Sistem Informasi ITS, penulis mengambil topik Perancangan Aplikasi Pencatatan Kehadiran Kuliah Pada Smartphone Android Dengan Metode Pengembangan Iterative Parallel Prototyping dan Iconix Process. Penulis dapat dihubungi melalui e-mail ashrht@yahoo.co.id.

Halaman ini sengaja dikosongkan

LAMPIRAN A
KUOSIONER PSSUQ

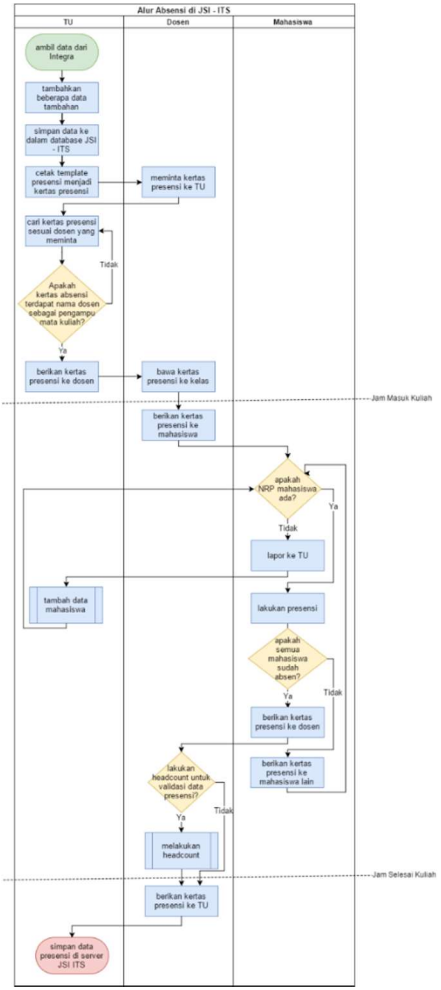
Prototipe : .....

		Sangat Tidak Setuju						Sangat Setuju		
1	Secara keseluruhan, saya puas dengan seberapa mudah untuk menggunakan prototipe ini	1	2	3	4	5	6	7		N\A
2	Prototipe ini simpel untuk digunakan	1	2	3	4	5	6	7		N\A
3	Saya bisa menyelesaikan task dan skenario dengan efektif menggunakan prototipe ini	1	2	3	4	5	6	7		N\A
4	Saya bisa menyelesaikan task dan skenario dengan cepat menggunakan prototipe ini	1	2	3	4	5	6	7		N\A
5	Saya bisa menyelesaikan task dan skenario dengan efisien menggunakan prototipe ini	1	2	3	4	5	6	7		N\A
6	Saya merasa nyaman menggunakan prototipe ini	1	2	3	4	5	6	7		N\A
7	Saya merasa mudah untuk mempelajari cara menggunakan prototipe	1	2	3	4	5	6	7		N\A
8	Saya yakin saya bisa menjadi produktif dengan cepat menggunakan prototipe ini	1	2	3	4	5	6	7		N\A
9	Prototipe ini memberikan 'Error Message' yang menunjukan saya cara untuk memperbaiki kesalahan yang saya lakukan dengan jelas	1	2	3	4	5	6	7		N\A
10	Setiap saya melakukan kesalahan penggunaan prototipe, saya bisa melanjutkan apa yang saya kerjakan sebelumnya dengan mudah dan cepat	1	2	3	4	5	6	7		N\A
11	Informasi (contoh : bantuan online, on-screen message, dan dokumentasi lain) yang disediakan dengan prototipe ini jelas	1	2	3	4	5	6	7		N\A
12	Saya bisa menemukan informasi yang saya butuhkan dengan mudah	1	2	3	4	5	6	7		N\A

13	Informasi yang disediakan di prototipe mudah di pahami	1	2	3	4	5	6	7		N\A
14	Informasi yang disediakan efektif dalam membantu saya menyelesaikan task dan skenario	1	2	3	4	5	6	7		N\A
15	Penataan informasi pada prototipe jelas, mudah dilihat	1	2	3	4	5	6	7		N\A
16	Interface pada prototipe ini menyenangkan	1	2	3	4	5	6	7		N\A
17	Saya suka menggunakan Interface pada prototipe ini	1	2	3	4	5	6	7		N\A
18	Prototipe ini memiliki fungsi dan kapabilitas sesuai dengan harapan saya	1	2	3	4	5	6	7		N\A
19	Secara keseluruhan, saya puas dengan prototipe ini	1	2	3	4	5	6	7		N\A

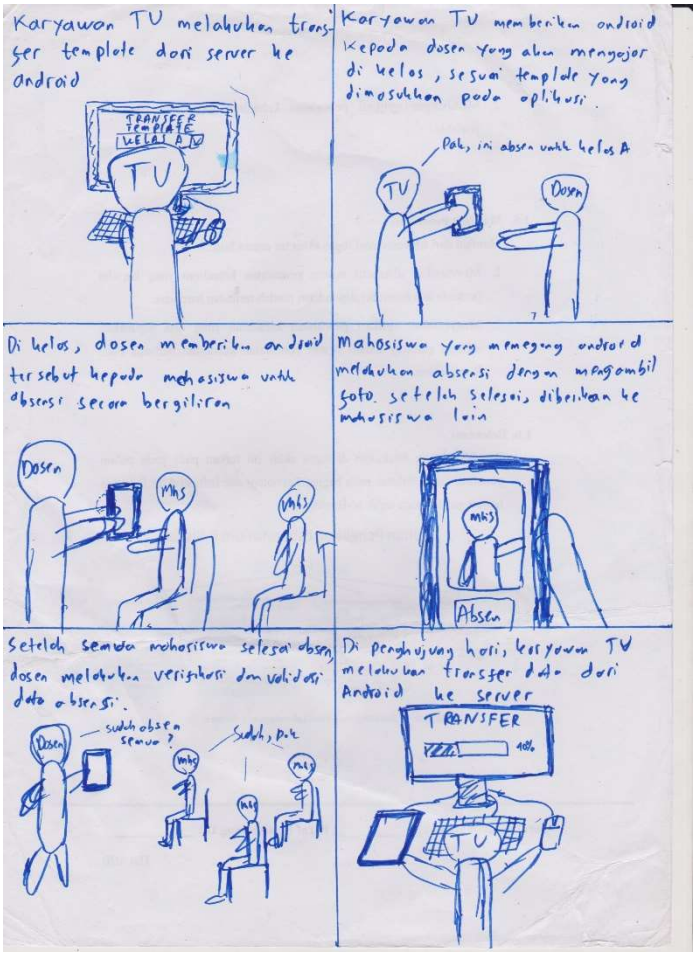
Interface termasuk berbagai hal yang digunakan dalam melakukan interaksi dengan prototipe. contoh beberapa hal yang termasuk interface adalah keyboard, mouse, layar (termasuk penggunaan visual dan bahasa yang ada di dalamnya)

LAMPIRAN B
SWIMLANE DIAGRAM ALUR PENCATATAN KEHADIRAN JSI-ITS

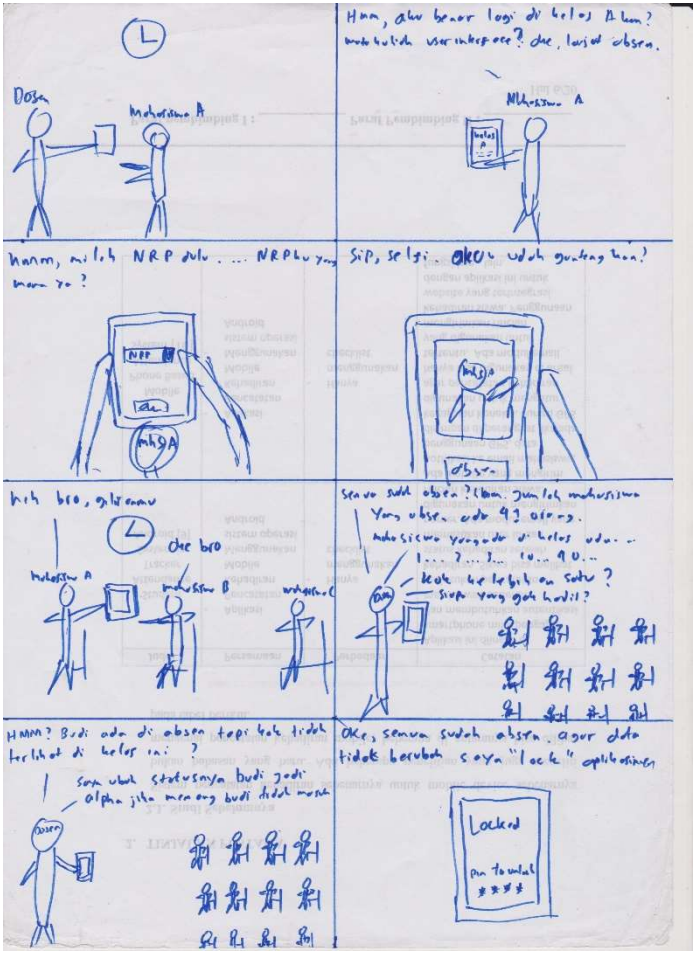


LAMPIRAN C
STORYBOARD

C.1 Storyboard A

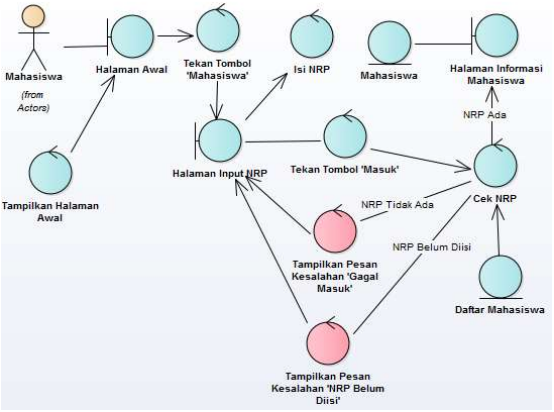


C.2 Storyboard B

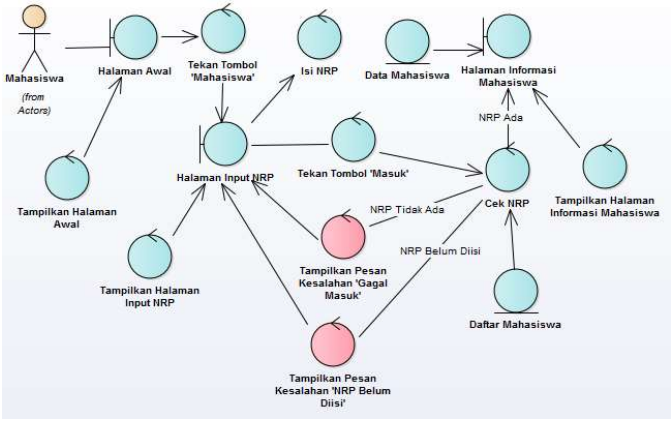


LAMPIRAN D
ROBUSTNESS ANALYSIS

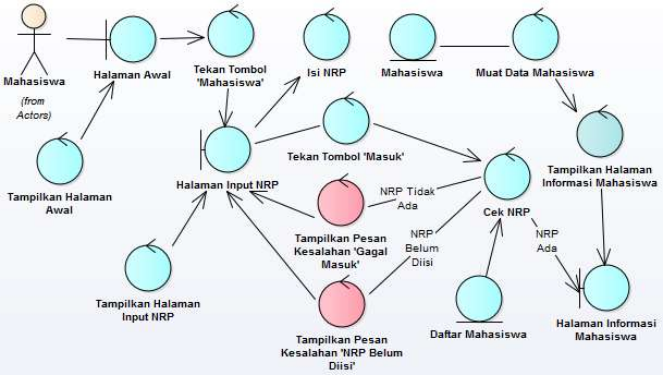
D.1 UC-101 Iterasi 000

Use Case	UC-101 Mahasiswa Membuka Halaman Informasi Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Ketika Mahasiswa membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Mahasiswa menekan tombol 'Mahasiswa' untuk mengakses 'Halaman Input NRP' lalu mengisi NRP dan menekan tombol 'Masuk'. Sistem mengecek NRP yang dimasukkan, jika NRP tersebut ada di dalam 'Daftar Mahasiswa' maka sistem akan menampilkan 'Halaman Informasi Mahasiswa'. **Alternate Scenario :** • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di Halaman Input NRP. • NRP tidak ada di 'Daftar Mahasiswa' : tampilkan pesan kesalahan 'Gagal Masuk' di Halaman Input NRP.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

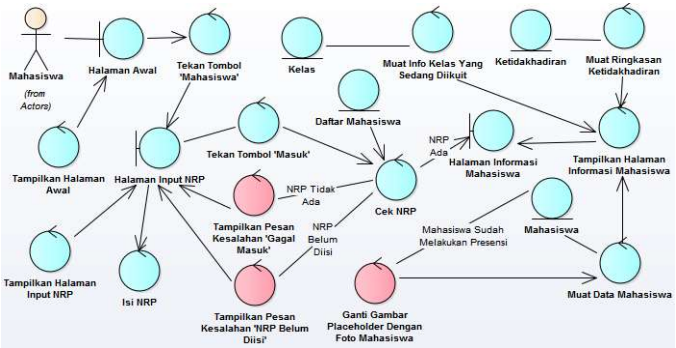
D.2 UC-101 Iterasi 001

Use Case	**UC-101 Mahasiswa Membuka Halaman Informasi Mahasiswa**	**Iterasi**	**001**
Use Case Description :	**Basic Scenario :** Ketika Mahasiswa membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Mahasiswa menekan tombol 'Mahasiswa'. Sistem akan menampilkan 'Halaman Input NRP' dimana Mahasiswa bisa mengisi NRP dan menekan tombol 'Masuk'. Ketika mahasiswa menekan tombol 'Masuk' Sistem mengecek NRP yang dimasukkan, jika NRP tersebut ada di dalam 'Daftar Mahasiswa' maka sistem akan menampilkan 'Halaman Informasi Mahasiswa'. **Alternate Scenario :** • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di Halaman Input NRP. • NRP tidak ada di 'Daftar Mahasiswa' : tampilkan pesan kesalahan 'Gagal Masuk' di Halaman Input NRP.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan controller Tampilkan Halaman Input NRP dan controller Tampilkan Halaman Informasi Mahasiswa. - Merevisi Entity Mahasiswa menjadi Data Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

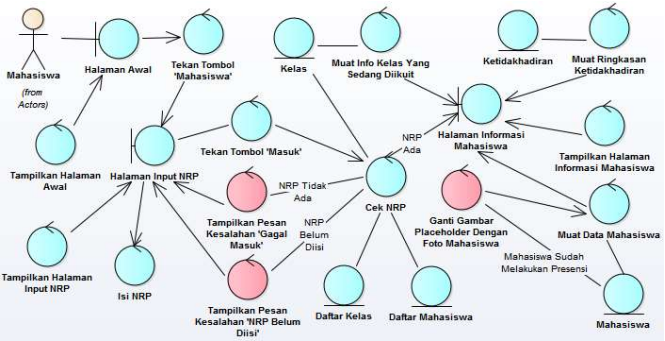
D.3 UC-101 Iterasi 002

Use Case	**UC-101 Mahasiswa Membuka Halaman Informasi Mahasiswa**	**Iterasi**	**002**
Use Case Description :	**Basic Scenario :** Ketika Mahasiswa membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Mahasiswa menekan tombol 'Mahasiswa'. Sistem akan menampilkan 'Halaman Input NRP' dimana Mahasiswa bisa mengisi NRP dan menekan tombol 'Masuk'. Ketika mahasiswa menekan tombol 'Masuk' sistem mengecek NRP yang dimasukkan, jika NRP tersebut ada di dalam 'Daftar Mahasiswa' maka sistem akan memuat data mahasiswa pada 'Halaman Informasi Mahasiswa' dan menampilkan 'Halaman Informasi Mahasiswa'. **Alternate Scenario :** • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di 'Halaman Input NRP'. • NRP tidak ada di 'Daftar Mahasiswa' : tampilkan pesan kesalahan 'Gagal Masuk' di 'Halaman Input NRP'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Merevisi hubungan antara Entity 'Data Mahasiswa' dengan Boundary 'Halaman Informasi Mahasiswa' dari secara langsung menjadi melalui Controller 'Muat Data Mahasiswa' dan Controller 'Tampilkan Informasi Mahasiswa'. - Mengubah Entity 'Data Mahasiswa' menjadi Entity 'Mahasiswa'.		
Luaran Tambahan :	(tidak ada)		

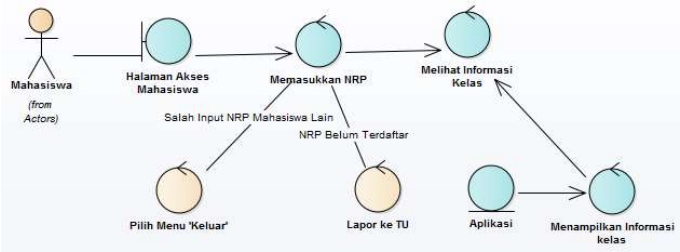
D.4 UC-101 Iterasi 003

Use Case	**UC-101 Mahasiswa Membuka Halaman Informasi Mahasiswa**	**Iterasi**	**003**
Use Case Description :	**Basic Scenario :** Ketika Mahasiswa membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Mahasiswa menekan tombol 'Mahasiswa'. Sistem akan menampilkan 'Halaman Input NRP' dimana Mahasiswa bisa mengisi NRP dan menekan tombol 'Masuk'. Ketika mahasiswa menekan tombol 'Masuk' sistem mengecek NRP yang dimasukkan, jika NRP tersebut ada di dalam 'Daftar Mahasiswa' maka sistem akan memuat data mahasiswa, info kelas yang sedang diikuti, serta ringkasan ketidakhadiran pada 'Halaman Informasi Mahasiswa' dan menampilkan 'Halaman Informasi Mahasiswa'. **Alternate Scenario :** • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di 'Halaman Input NRP'. • NRP tidak ada di 'Daftar Mahasiswa' : tampilkan pesan kesalahan 'Gagal Masuk' di 'Halaman Input NRP'. • Mahasiswa sudah melakukan presensi : ganti gambar placeholder dengan foto mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan Entity Kelas, Entity Ketidakhadiran, dan Controller yang terkait agar sesuai dengan prototipe UI yang sudah dihasilkan. - Menambahkan Alternate Scenario jika mahasiswa sudah melakukan presensi.		
Luaran Tambahan :	(tidak ada)		

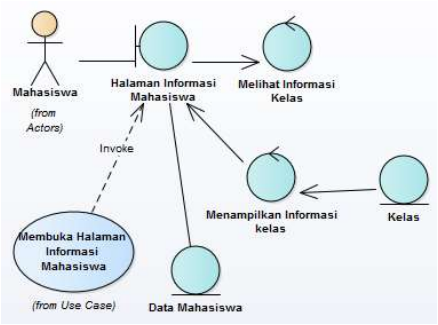
D.5 UC-101 Iterasi 004

Use Case	UC-101 Mahasiswa Membuka Halaman Informasi Mahasiswa	Iterasi	004
Use Case Description :	**Basic Scenario :** Ketika Mahasiswa membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Mahasiswa menekan tombol 'Mahasiswa'. Sistem akan menampilkan 'Halaman Input NRP' dimana Mahasiswa bisa mengisi NRP dan menekan tombol 'Masuk'. Ketika mahasiswa menekan tombol 'Masuk' sistem mengecek NRP yang dimasukkan. jika NRP tersebut ada di dalam 'Daftar Mahasiswa' dari kelas yang ada dalam 'Daftar Kelas', maka sistem akan menampilkan 'Halaman Informasi Mahasiswa' lalu memuat data mahasiswa, info kelas yang sedang diikuti, serta ringkasan ketidakhadiran pada 'Halaman Informasi Mahasiswa'. **Alternate Scenario :** • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di 'Halaman Input NRP'. • NRP tidak ada di 'Daftar Mahasiswa' : tampilkan pesan kesalahan 'Gagal Masuk' di 'Halaman Input NRP'. • Mahasiswa sudah melakukan presensi : ganti gambar placeholder dengan foto mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Mengubah use case description dan merapikan elemen sequence diagram agar sesuai dengan perubahan use case description. - Menambahkan dan menghubungkan beberapa Entity yang terhubung dengan proses cek nrp.		
Luaran Tambahan :	(tidak ada)		

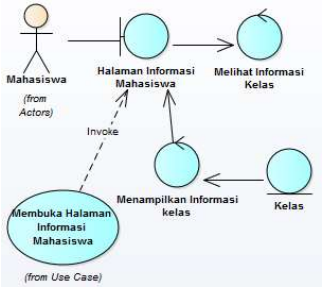
D.6 DUC-101 Iterasi 000

Use Case	DUC-101 Mahasiswa Melihat Informasi Kelas	Iterasi	000
Use Case Description :	**Basic Scenario :** Mahasiswa memilih akses mahasiswa, memasukkan NRP, lalu melihat informasi kelas yang ditampilkan oleh aplikasi. **Alternate Scenario :** • Salah Input NRP mahasiswa lain : pilih menu 'keluar'. • NRP belum terdaftar : lapor ke TU. • Informasi kelas salah : lapor ke TU.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

D.7 DUC-101 Iterasi 001

Use Case	DUC-101 Mahasiswa Melihat Informasi Kelas	Iterasi	001
Use Case Description :	**Basic Scenario :** Mahasiswa Melihat Informasi Kelas yang ditampilkan oleh sistem pada halaman Informasi Mahasiswa. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus beberapa komponen yang sudah menjadi bagian dari use case lain (membuka halaman informasi mahasiswa) dan menghubungkan entity Data Mahasiswa ke Boundary Halaman Informasi Mahasiswa. - Mengubah Entity Aplikasi menjadi Kelas, sesuai dengan domain model yang ada. - Menghapus Alternate Scenario karena tidak fokus ke interaksi antara aplikasi (atau sistem) ke pengguna.		
Luaran Tambahan :	(tidak ada)		

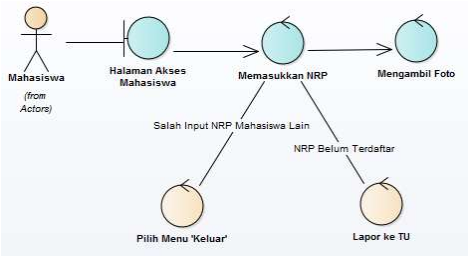
D.8 DUC-101 Iterasi 002

Use Case	DUC-101 Mahasiswa Melihat Informasi Kelas	Iterasi	002
Use Case Description :	**Basic Scenario :** Setelah Mahasiswa membuka 'Halaman Informasi Mahasiswa', mahasiswa bisa melihat informasi kelas yang yang ditampilkan oleh sistem pada 'Halaman Informasi Mahasiswa'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity "Data Mahasiswa".		
Luaran Tambahan :	(tidak ada)		

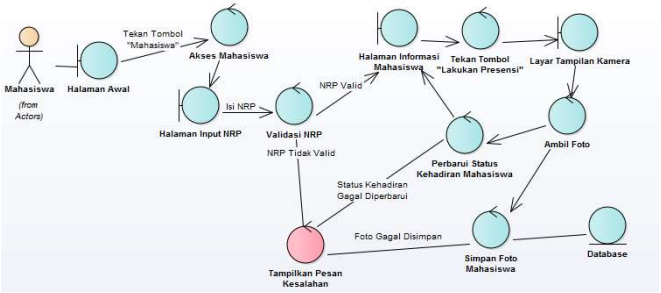
D.9 DUC-101 Iterasi 003

Use Case	DUC-101 Mahasiswa Melihat Informasi Kelas	Iterasi	003
Use Case Description :	**Basic Scenario :** (tidak ada) **Alternate Scenario :** (tidak ada)		
Diagram Robustness:	(tidak ada)		
Pembeda Dengan Iterasi Sebelumnya :	- Use Case digabung dengan Use Case Mhs – Membuka Halaman Informasi Mahasiswa karena 'Halaman Informasi Mahasiswa' tidak hanya menampilkan info kelas tapi juga beberapa info lain.		
Luaran Tambahan :	(tidak ada)		

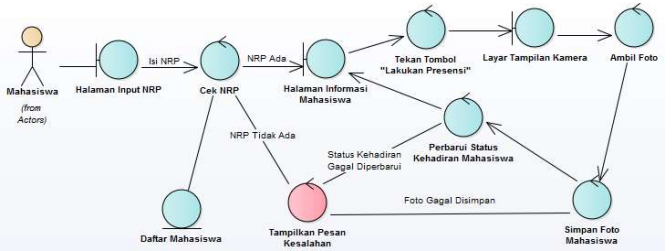
D.10 UC-102 Iterasi 000

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	000
Use Case Description :	**Basic Scenario :** Mahasiswa memilih akses mahasiswa, memasukkan NRP, lalu mengambil foto. **Alternate Scenario :** • Salah Input NRP mahasiswa lain : pilih menu 'keluar'. • NRP belum terdaftar : lapor ke TU.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

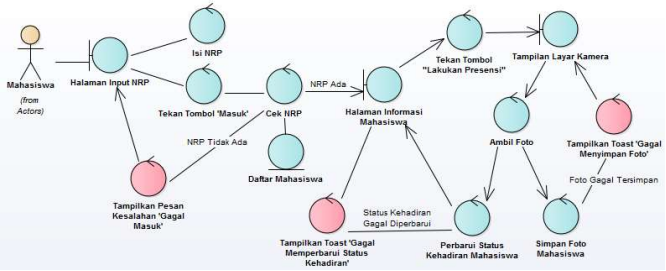
D.11 UC-102 Iterasi 001

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	001
Use Case Description :	**Basic Scenario :** Mahasiswa membuka Halaman Awal lalu menekan tombol 'Mahasiswa' untuk masuk dengan 'Akses Mahasiswa'. Aplikasi memproses akses mahasiswa dan menampilkan 'Halaman Input NRP'. Mahasiswa mengisikan nrp. Sistem melakukan validasi nrp, jika nrp valid maka aplikasi menampilkan 'Halaman Informasi Mahasiswa'. Mahasiswa menekan tombol 'Lakukan Presensi'. Aplikasi membuka 'Layar Tampilan Kamera'. Mahasiswa mengambil foto. Sistem menyimpan foto mahasiswa di database lantas memperbarui status kehadiran mahasiswa. aplikasi membawa mahasiswa kembali ke halaman informasi mahasiswa dengan informasi yang telah diperbarui. **Alternate Scenario :** • NRP tidak valid - aplikasi menampilkan pesan kesalahan. • Foto gagal disimpan – tampilkan pesan kesalahan. • Status kehadiran gagal diperbarui – tampilkan pesan kesalahan.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperjelas Boundary 'Halaman Akses Mahasiswa' menjadi lebih rinci sesuai dengan alur prototipe UI. - Memperbaiki Alternate Scenario menjadi lebih fokus ke interaksi antara aplikasi (atau sistem) ke pengguna tersebut. - Memperjelas Control 'Mengambil Foto' menjadi lebih rinci sesuai dengan alur prototipe UI.		
Luaran Tambahan :	(tidak ada)		

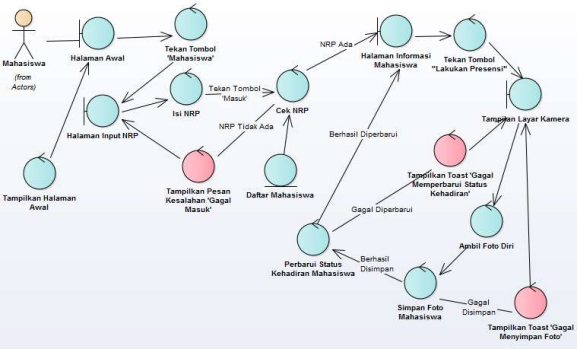
D.12 UC-102 Iterasi 002

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	002
Use Case Description :	**Basic Scenario :** Mahasiswa mengisi NRP di 'Halaman Input NRP'. Sistem akan melakukan cek NRP berdasarkan Daftar Mahasiswa, jika NRP ada maka Mahasiswa dibawa ke 'Halaman Informasi Mahasiswa'. Di halaman 'Informasi Mahasiswa', mahasiswa menekan tombol 'Lakukan Presensi'. Sistem akan membuka tampilan layar kamera. Setelah Mahasiswa mengambil foto diri, sistem akan menyimpan foto, memperbarui status kehadiran, dan membawa mahasiswa kembali ke halaman 'Informasi Mahasiswa' yang telah diperbarui informasinya. **Alternate Scenario :** • NRP tidak ada : tampilkan pesan kesalahan. • Foto gagal disimpan : tampilkan pesan kesalahan. • Status kehadiran gagal diperbarui : tampilkan pesan kesalahan.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Mahasiswa dibawa langsung ke Halaman Input NRP, tidak melalui halaman Awal seperti iterasi sebelumnya. - Menambahkan Entity Daftar Mahasiswa pada Control Cek NRP.		
Luaran Tambahan :	(tidak ada)		

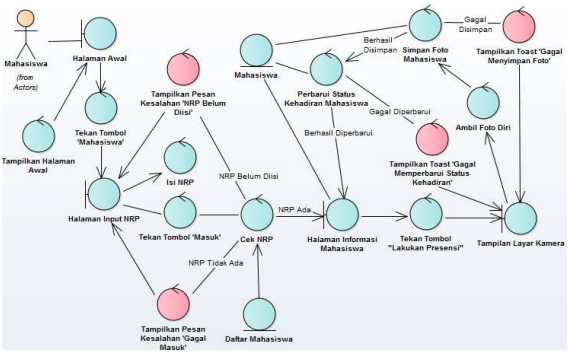
D.13 UC-102 Iterasi 003

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	003
Use Case Description :	**Basic Scenario :** Mahasiswa mengisi NRP di 'Halaman Input NRP'. Ketika mahasiswa menekan tombol 'Masuk', sistem akan melakukan Cek NRP berdasarkan Daftar Mahasiswa. Jika NRP ada maka mahasiswa akan dibawa ke 'Halaman Informasi Mahasiswa'. Di halaman 'Informasi Mahasiswa', mahasiswa menekan tombol 'Lakukan Presensi'. Sistem akan membuka tampilan layar kamera. Setelah Mahasiswa mengambil foto, sistem akan menyimpan foto, memperbarui status kehadiran, dan membawa mahasiswa kembali ke halaman 'Informasi Mahasiswa' yang telah diperbarui informasinya. **Alternate Scenario :** • NRP tidak ada : sistem akan menampilkan pesan 'Gagal Masuk' di halaman Input NRP. • Foto gagal tersimpan : tampilkan toast 'Gagal Menyimpan Foto' di halaman tampilan layar kamera. • Status kehadiran gagal diperbarui : tampilkan toast 'Gagal Memperbarui Status Kehadiran' di halaman informasi mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperjelas Control 'Tampilkan Pesan Kesalahan' untuk alternate case yang ada. - Memperjelas langkah yang dilakukan pada 'Halaman Input NRP'.		
Luaran Tambahan :	(tidak ada)		

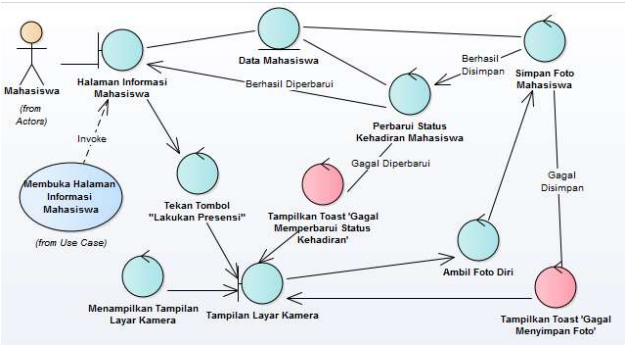
D.14 UC-102 Iterasi 004

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	004
Use Case Description :	**Basic Scenario :** Ketika Mahasiswa membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Mahasiswa menekan tombol 'Mahasiswa'. Sistem menampilkan 'Halaman Input NRP'. Mahasiswa mengisi NRP dan menekan tombol 'Masuk'. Sistem mengecek NRP yang dimasukkan, jika NRP tersebut ada di dalam 'Daftar Mahasiswa' maka sistem akan menampilkan 'Halaman Informasi Mahasiswa'. Untuk melakukan presensi, mahasiswa menekan tombol 'Lakukan Presensi' yang ada di 'Halaman Informasi Mahasiswa'. Sistem menampilkan 'Tampilan Layar Kamera' dan Mahasiswa mengambil foto diri. Selanjutnya sistem menyimpan Foto Mahasiswa, jika foto berhasil disimpan maka sistem akan memperbarui status kehadiran mahasiswa. Setelah itu sistem akan membawa mahasiswa kembali ke 'Halaman Informasi Mahasiswa' dengan status kehadiran yang sudah diperbarui. **Alternate Scenario :** • NRP tidak ada di 'Daftar Mahasiswa' : Tampilkan Pesan Kesalahan Gagal Masuk di Halaman Input NRP. • Sistem gagal menyimpan foto mahasiswa : Tampilkan Toast 'Gagal Menyimpan Foto' dan bawa mahasiswa kembali ke Tampilan Layar Kamera. • Sistem gagal memperbarui status kehadiran mahasiswa : Tampilkan Toast 'Gagal Memperbarui Status Kehadiran Mahasiswa' dan bawa mahasiswa kembali ke Tampilan Layar Kamera.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Halaman Awal dimasukkan kembali ke dalam Use Case + Ada tambahan Control 'Tampilkan Halaman awal'. - Menyederhanakan Control 'Isi NRP' dan 'Tekan Tombol Masuk'. - Memperjelas langkah yang dilakukan pada 'Halaman Input NRP'. - Memperbaiki urutan alur yang terjadi sesudah mahasiswa mengambil foto diri. - Perubahan Alternate Scenario untuk gagal perbarui status kehadiran mahasiswa, Toast ditamplkan pada tampilan layar kamera, bukan Halaman Informasi Mahasiswa seperti iterasi sebelumnya.		
Luaran Tambahan :	(tidak ada)		

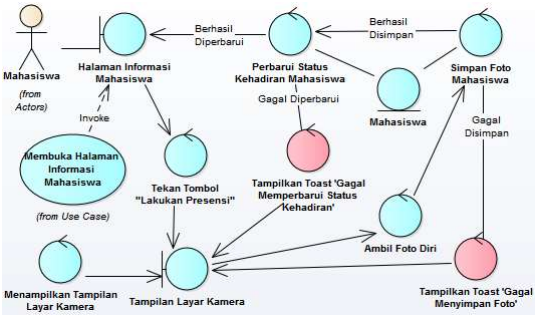
D.15 UC-102 Iterasi 005

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	005
Use Case Description :	**Basic Scenario :** Ketika Mahasiswa membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Mahasiswa menekan tombol 'Mahasiswa' untuk mengakses 'Halaman Input NRP' lalu mengisi NRP dan menekan tombol 'Masuk'. Sistem mengecek NRP yang dimasukkan, jika NRP tersebut ada di dalam 'Daftar Mahasiswa' maka sistem akan menampilkan 'Halaman Informasi Mahasiswa'. Untuk melakukan presensi, mahasiswa menekan tombol 'Lakukan Presensi' yang ada di 'Halaman Informasi Mahasiswa'. Sistem menampilkan 'Tampilan Layar Kamera' dan Mahasiswa mengambil foto diri. Selanjutnya sistem menyimpan Foto Mahasiswa, jika foto berhasil disimpan maka sistem akan memperbarui status kehadiran mahasiswa. Setelah itu sistem akan membawa mahasiswa kembali ke 'Halaman Informasi Mahasiswa' dengan status kehadiran yang sudah diperbarui. **Alternate Scenario :** • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di Halaman Input NRP. • NRP tidak ada di 'Daftar Mahasiswa' : tampilkan pesan kesalahan 'Gagal Masuk' di Halaman Input NRP. • Sistem gagal menyimpan foto mahasiswa : tampilkan toast 'Gagal Menyimpan Foto' dan bawa mahasiswa kembali ke Tampilan Layar Kamera. • Sistem gagal memperbarui status kehadiran mahasiswa : tampilkan toast 'Gagal Memperbarui Status Kehadiran Mahasiswa' dan bawa mahasiswa kembali ke Tampilan Layar Kamera.		
Diagram Robustness:	
Pembeda Dengan Iterasi Sebelumnya :	- Penambahan Entity Mahasiswa yang terhubung dengan controller simpan foto mahasiswa dan controller perbarui status kehadiran mahasiswa maupun Boundary halaman informasi mahasiswa. - Pemisahan controller isi NRP menjadi kembali seperti beberapa iterasi sebelumnya, controller Isi NRP dan controller Tekan Tombol Masuk. - Penambahan alternate scenario jika NRP belum diisi.		
Luaran Tambahan :	(tidak ada)		

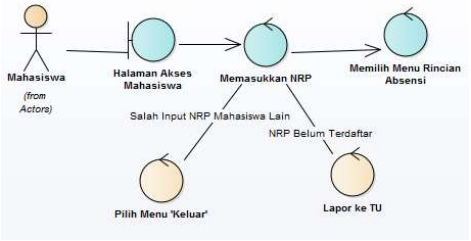
D.16 UC-102 Iterasi 006

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	006
Use Case Description :	**Basic Scenario :** Mahasiswa menekan tombol 'Lakukan Presensi' yang ada Halaman Informasi Mahasiswa. Sistem menampilkan 'Tampilan Layar Kamera' dan Mahasiswa mengambil foto diri. Selanjutnya sistem menyimpan Foto Mahasiswa, jika foto berhasil disimpan maka sistem akan memperbarui status kehadiran mahasiswa. Setelah itu sistem akan membawa mahasiswa kembali ke 'Halaman Informasi Mahasiswa' dengan 'Data Mahasiswa' yang sudah diperbarui. **Alternate Scenario :** • Sistem gagal menyimpan foto mahasiswa : tampilkan toast 'Gagal Menyimpan Foto' dan bawa mahasiswa kembali ke Tampilan Layar Kamera. • Sistem gagal memperbarui status kehadiran mahasiswa : tampilkan toast 'Gagal Memperbarui Status Kehadiran Mahasiswa' dan bawa mahasiswa kembali ke Tampilan Layar Kamera.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memisahkan sebagian besar komponen menjadi use case baru (membuka halaman informasi mahasiswa). - Menambahkan controller Menampilkan Tampilan Layar kamera. - Menambahkan & menghubungkan use case Membuka Halaman Informasi Mahasiswa. - Merevisi Entity Mahasiswa menjadi Data Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

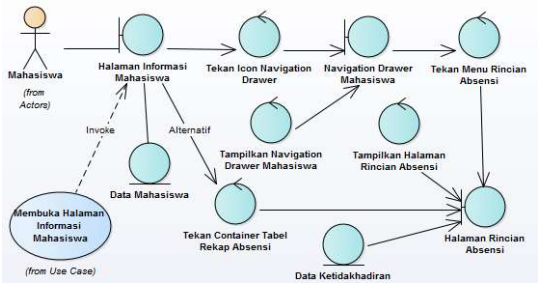
D.17 UC-102 Iterasi 007

Use Case	UC-102 Mahasiswa Melakukan Presensi	Iterasi	007
Use Case Description :	**Basic Scenario :** Setelah Mahasiswa membuka 'Halaman Informasi Mahasiswa', Mahasiswa menekan tombol 'Lakukan Presensi' yang ada di 'Halaman Informasi Mahasiswa'. Sistem menampilkan 'Tampilan Layar Kamera' dan Mahasiswa mengambil foto diri. Selanjutnya sistem menyimpan foto mahasiswa, jika foto berhasil disimpan maka sistem akan memperbarui status kehadiran mahasiswa. Setelah itu sistem akan membawa mahasiswa kembali ke 'Halaman Informasi Mahasiswa'. **Alternate Scenario :** • Sistem gagal menyimpan foto mahasiswa : tampilkan toast 'Gagal Menyimpan Foto' dan bawa mahasiswa kembali ke 'Tampilan Layar Kamera'. • Sistem gagal memperbarui status kehadiran mahasiswa : tampilkan toast 'Gagal Memperbarui Status Kehadiran Mahasiswa' dan bawa mahasiswa kembali ke Tampilan Layar Kamera.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus relasi dari Entity 'Data Mahasiswa' dengan Boundary 'Halaman Informasi Mahasiswa'. - Mengubah Entity 'Data Mahasiswa' menjadi Entity 'Mahasiswa'.		
Luaran Tambahan :	(tidak ada)		

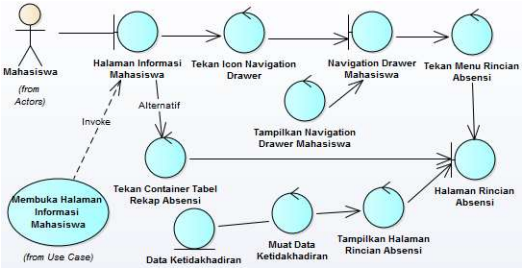
D.18 UC-103 Iterasi 000

Use Case	UC-103 Mahasiswa Melihat Data Ketidakhadiran	Iterasi	000
Use Case Description :	**Basic Scenario :** Mahasiswa memilih akses mahasiswa, memasukkan NRP, lalu memilih menu rincian absensi. **Alternate Scenario :** • Salah Input NRP mahasiswa lain : pilih menu 'keluar'. • NRP belum terdaftar : lapor ke TU.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

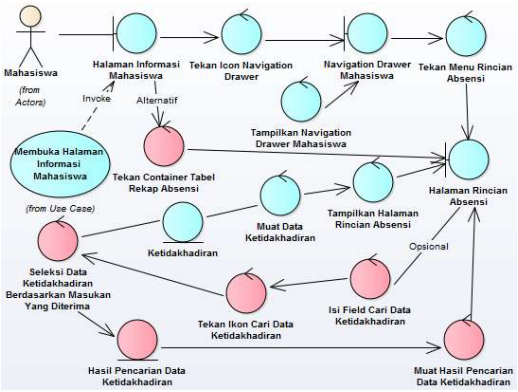
D.19 UC-103 Iterasi 001

Use Case	UC-103 Mahasiswa Melihat Data Ketidakhadiran	Iterasi	001
Use Case Description :	**Basic Scenario :** Pada Halaman Informasi Mahasiswa, mahasiswa menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Mahasiswa dimana mahasiswa menekan Menu Rincian Absensi. Sistem selanjutnya membawa mahasiswa ke Halaman Rincian Absensi. **Alternate Scenario :** • Mahasiswa menekan container Tabel Rekap Absensi : Sistem membawa mahasiswa langsung ke Halaman Rincian Absensi.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus beberapa komponen yang sudah menjadi bagian dari use case lain (membuka halaman informasi mahasiswa) dan menghubungkan entity Data Mahasiswa ke Boundary Halaman Informasi Mahasiswa. - Memperjelas alur yang terjadi dari Halaman Informasi Mahasiswa menuju Halaman Rincian Absensi. - Memperbaiki Alternate Scenario menjadi lebih fokus ke interaksi antara aplikasi (atau sistem) ke pengguna.		
Luaran Tambahan :	RAUI-001		

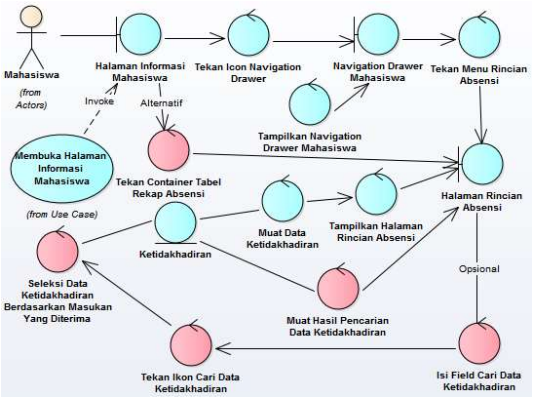
D.20 UC-103 Iterasi 002

Use Case	UC-103 Mahasiswa Melihat Data Ketidakhadiran	Iterasi	002
Use Case Description :	**Basic Scenario :** Pada Halaman Informasi Mahasiswa, mahasiswa menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Mahasiswa dimana mahasiswa menekan Menu Rincian Absensi. Sistem selanjutnya membawa mahasiswa ke Halaman Rincian Absensi. **Alternate Scenario :** • Mahasiswa menekan container Tabel Rekap Absensi : Sistem membawa mahasiswa langsung ke Halaman Rincian Absensi.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity Data Mahasiswa. - Merevisi hubungan antara Entity Data Ketidakhadiran dengan Boundary Halaman Rincian Absensi dari secara langsung menjadi melalui Controller Muat Data Ketidakhadiran dan Controller Tampilkan Halaman Rincian Absensi.		
Luaran Tambahan :	(tidak ada)		

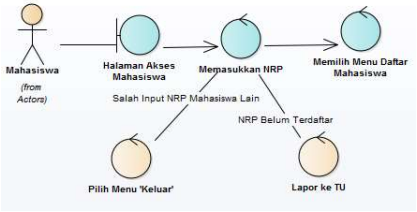
D.21 UC-103 Iterasi 003

Use Case	UC-103 Mahasiswa Melihat Data Ketidakhadiran	Iterasi	003
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman Informasi Mahasiswa', mahasiswa menekan 'Icon Navigation Drawer' yang ada di 'Halaman Informasi Mahasiswa'. Selanjutnya sistem menampilkan 'Navigation Drawer Mahasiswa' dimana mahasiswa menekan 'Menu Rincian Absensi'. Sistem selanjutnya membawa mahasiswa ke 'Halaman Rincian Absensi' dimana mahasiswa bisa melihat data ketidakhadiran yang ditampilkan oleh sistem. **Alternate Scenario :** • Alternatif, Mahasiswa menekan 'Container Tabel Rekap Absensi' : Sistem membawa mahasiswa langsung ke 'Halaman Rincian Absensi'. • Opsional, Mahasiswa mencari data ketidakhadiran : Mahasiswa mengisi 'Field Cari Data Ketidakhadiran' lalu menekan 'Ikon Cari Data Ketidakhadiran'. Sistem memproses perilaku user dengan cara melakukan seleksi data ketidakhadiran berdasarkan masukan yang diterima lalu memuat hasil pencarian ketidakhadiran di 'Halaman Rincian Absensi'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan alernate scenario opsional. - Mengubah Entity 'Data Ketidakhadiran' menjadi Entity 'Ketidakhadiran'.		
Luaran Tambahan :	(tidak ada)		

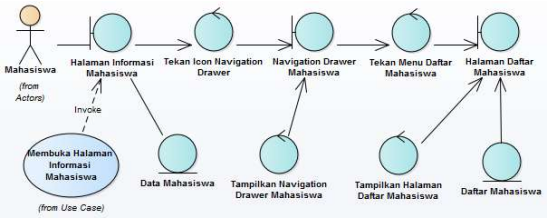
D.22 UC-103 Iterasi 004

Use Case	UC-103 Mahasiswa Melihat Data Ketidakhadiran	Iterasi	004
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman Informasi Mahasiswa', mahasiswa menekan 'Icon Navigation Drawer' yang ada di 'Halaman Informasi Mahasiswa'. Selanjutnya sistem menampilkan 'Navigation Drawer Mahasiswa' dimana mahasiswa menekan 'Menu Rincian Absensi'. Sistem selanjutnya membawa mahasiswa ke 'Halaman Rincian Absensi' dimana mahasiswa bisa melihat data ketidakhadiran yang ditampilkan oleh sistem. **Alternate Scenario :** • Alternatif, Mahasiswa menekan 'Container Tabel Rekap Absensi' : Sistem membawa mahasiswa langsung ke 'Halaman Rincian Absensi'. • Opsional, Mahasiswa mencari data ketidakhadiran : Mahasiswa mengisi 'Field Cari Data Ketidakhadiran' lalu menekan 'Ikon Cari Data Ketidakhadiran'. Sistem memproses perilaku user dengan cara melakukan seleksi data ketidakhadiran berdasarkan masukan yang diterima lalu memuat hasil pencarian ketidakhadiran di 'Halaman Rincian Absensi'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity Hasil Ketidakhadiran karena fungsi search yang diajukan tidak membutuhkan cara yang rumit. - Menyesuaikan Controller Muat Hasil Pencarian Data Ketidakhadiran sehingga terhubung langsung dengan Entity Ketidakhadiran.		
Luaran Tambahan :	(tidak ada)		

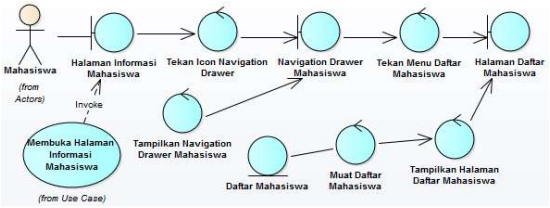
D.23 UC-104 Iterasi 000

Use Case	UC-104 Mahasiswa Melihat Daftar Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Mahasiswa memilih akses mahasiswa, memasukkan NRP, lalu memilih menu daftar mahasiswa. **Alternate Scenario :** • Salah Input NRP mahasiswa lain : pilih menu 'keluar'. • NRP belum terdaftar : lapor ke TU.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity Hasil Ketidakhadiran karena fungsi search yang diajukan tidak membutuhkan cara yang rumit. - Menyesuaikan Controller Muat Hasil Pencarian Data Ketidakhadiran sehingga terhubung langsung dengan Entity Ketidakhadiran.		
Luaran Tambahan :	(tidak ada)		

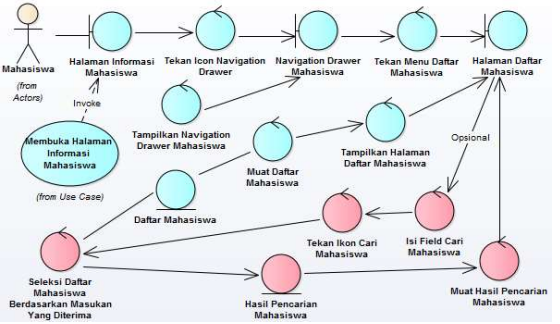
D.24 UC-104 Iterasi 001

Use Case	UC-104 Mahasiswa Melihat Daftar Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Pada Halaman Informasi Mahasiswa, mahasiswa menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Mahasiswa dimana mahasiswa menekan Menu Daftar Mahasiswa. Sistem selanjutnya membawa mahasiswa ke Halaman Daftar Mahasiswa. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus beberapa komponen yang sudah menjadi bagian dari use case lain (membuka halaman informasi mahasiswa) dan menghubungkan entity Data Mahasiswa ke Boundary Halaman Informasi Mahasiswa. - Memperjelas alur yang terjadi dari Halaman Informasi Mahasiswa menuju Halaman Daftar Mahasiswa. - Menghapus Alternate Scenario karena tidak fokus ke interaksi antara aplikasi (atau sistem) ke pengguna.		
Luaran Tambahan :	RAUI-001		

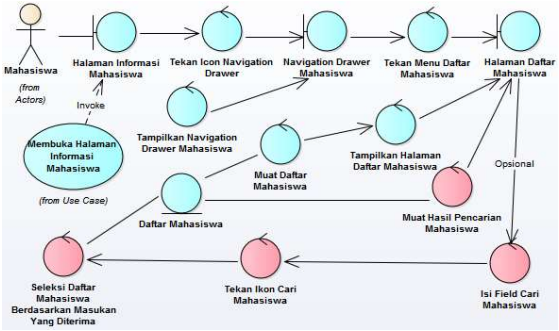
D.25 UC-104 Iterasi 002

Use Case	UC-104 Mahasiswa Melihat Daftar Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** Pada Halaman Informasi Mahasiswa, mahasiswa menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Mahasiswa dimana mahasiswa menekan Menu Daftar Mahasiswa. Sistem selanjutnya membawa mahasiswa ke Halaman Daftar Mahasiswa. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity Data Mahasiswa. - Merevisi hubungan antara Entity Daftar Mahasiswa dengan Boundary Halaman Daftar Mahasiswa dari secara langsung menjadi melalui Controller Muat Daftar Mahasiswa dan Controller Tampilkan Halaman Daftar Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

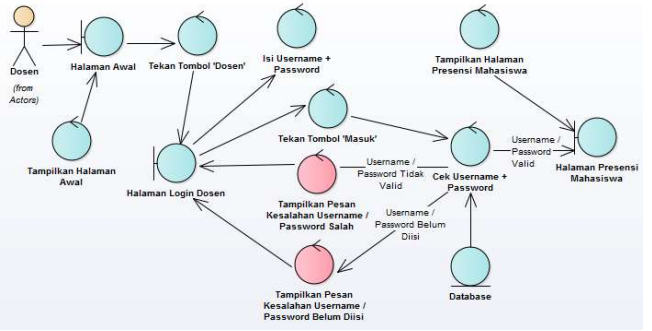
D.26 UC-104 Iterasi 003

Use Case	UC-104 Mahasiswa Melihat Daftar Mahasiswa	Iterasi	003
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman Informasi Mahasiswa', Mahasiswa menekan 'Icon Navigation Drawer' yang ada di 'Halaman Informasi Mahasiswa'. Selanjutnya sistem menampilkan 'Navigation Drawer Mahasiswa' dimana Mahasiswa menekan 'Menu Daftar Mahasiswa'. Sistem selanjutnya membawa Mahasiswa ke 'Halaman Daftar Mahasiswa' dimana Mahasiswa bisa melihat daftar mahasiswa yang ditampilkan oleh sistem. **Alternate Scenario :** • Opsional, Mahasiswa mencari mahasiswa : Mahasiswa mengisi 'Field Cari Mahasiswa' lalu menekan 'Ikon Cari Mahasiswa'. Sistem memproses perilaku user dengan cara melakukan seleksi daftar mahasiswa berdasarkan masukan yang diterima lalu memuat hasil pencarian mahasiswa di 'Halaman Daftar Mahasiswa'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan alernate scenario opsional.		
Luaran Tambahan :	(tidak ada)		

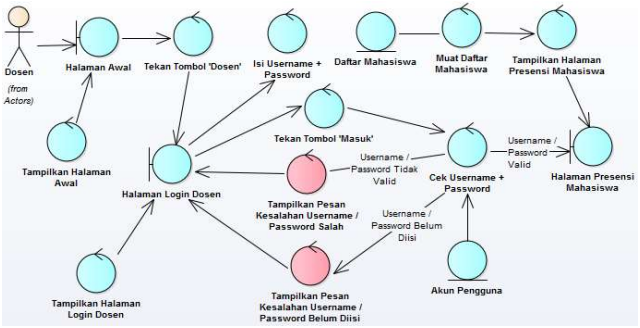
D.27 UC-104 Iterasi 004

Use Case	UC-104 Mahasiswa Melihat Daftar Mahasiswa	Iterasi	004
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman Informasi Mahasiswa', Mahasiswa menekan 'Icon Navigation Drawer' yang ada di 'Halaman Informasi Mahasiswa'. Selanjutnya sistem menampilkan 'Navigation Drawer Mahasiswa' dimana Mahasiswa menekan 'Menu Daftar Mahasiswa'. Sistem selanjutnya membawa Mahasiswa ke 'Halaman Daftar Mahasiswa' dimana Mahasiswa bisa melihat daftar mahasiswa yang ditampilkan oleh sistem. **Alternate Scenario :** • Opsional, Mahasiswa mencari mahasiswa : Mahasiswa mengisi 'Field Cari Mahasiswa' lalu menekan 'Ikon Cari Mahasiswa'. Sistem memproses perilaku user dengan cara melakukan seleksi daftar mahasiswa berdasarkan masukan yang diterima lalu memuat hasil pencarian mahasiswa di 'Halaman Daftar Mahasiswa'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity Hasil Pencarian Mahasiswa karena fungsi search yang diajukan tidak membutuhkan cara yang rumit. - Menyesuaikan Controller Muat Hasil Pencarian Mahasiswa sehingga terhubung langsung dengan Entity Daftar Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

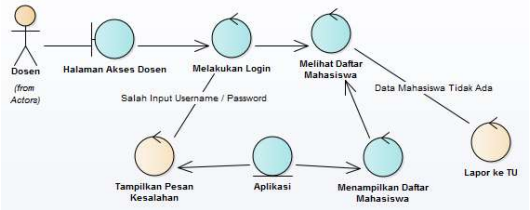
D.28 UC-201 Iterasi 000

Use Case	UC-201 Dosen Membuka Halaman Presensi Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Ketika dosen membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Dosen selanjutnya menekan tombol 'Dosen' dan sistem akan menampilkan 'Halaman Login Dosen', dimana dosen bisa mengisi Username + Password dan menekan tombol 'Masuk'. Ketika dosen menekan tombol 'Masuk' Sistem mengecek Username + Password yang dimasukkan berdasar database yang ada, jika valid maka sistem akan menampilkan 'Halaman Presensi Mahasiswa'. **Alternate Scenario :** • Username / Password tidak valid : tampilkan pesan kesalahan 'Username / Password Salah' di Halaman Login Dosen. • Username / Password belum diisi : tampilkan pesan kesalahan 'Username / Password Belum Diisi' di Halaman Login Dosen.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

D.29 UC-201 Iterasi 001

Use Case	UC-201 Dosen Membuka Halaman Presensi Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Ketika Dosen membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Dosen selanjutnya menekan tombol 'Dosen' dan sistem akan menampilkan 'Halaman Login Dosen', dimana Dosen bisa mengisi Username + Password dan menekan tombol 'Masuk'. Ketika Dosen menekan tombol 'Masuk' sistem mengecek Username + Password yang dimasukkan berdasarkan akun pengguna yang ada, jika valid maka sistem akan memuat daftar mahasiswa pada 'Halaman Presensi Mahasiswa' lalu menampilkan 'Halaman Presensi Mahasiswa'. **Alternate Scenario :** • Username / Password tidak valid : tampilkan pesan kesalahan 'Username / Password Salah' di 'Halaman Login Dosen'. • Username / Password belum diisi : tampilkan pesan kesalahan 'Username / Password Belum Diisi' di 'Halaman Login Dosen'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan Control 'Tampilkan Halaman Login Dosen' yang terhubung dengan Boundary 'Halaman Login Dosen'. - Menambahkan Entity 'Daftar Mahasiswa' beserta Controller 'Muat Daftar Mahasiswa' yang terhubung ke Boundary 'Halaman Presensi Mahasiswa' melalui Controller 'Tampilkan Halaman Presensi Mahasiswa'. - Mengubah Entity 'Database' menjadi Entity 'Akun Pengguna'.		
Luaran Tambahan :	(tidak ada)		

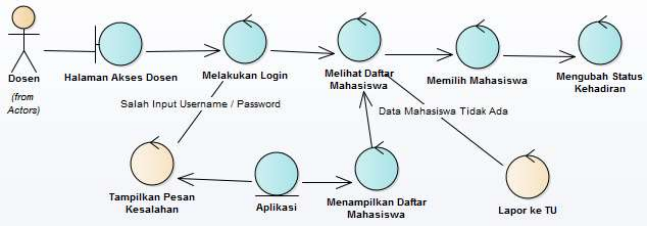
D.30 DUC-201 Iterasi 000

Use Case	DUC-201 Dosen Melihat Daftar Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Dosen memilih akses dosen, melakukan login, lalu melihat daftar mahasiswa yang ditampilkan oleh aplikasi. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan. • Data mahasiswa tidak ada : lapor ke TU.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

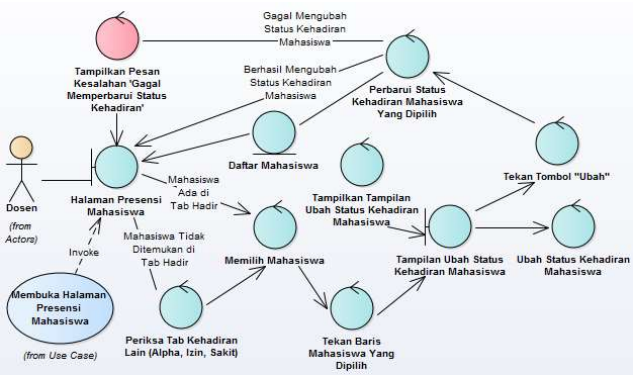
D.31 DUC-201 Iterasi 001

Use Case	DUC-201 Dosen Melihat Daftar Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** (tidak ada) **Alternate Scenario :** (tidak ada)		
Diagram Robustness:	(tidak ada)		
Pembeda Dengan Iterasi Sebelumnya :	- Use Case digabung dengan Use Case Dsn – Membuka Halaman Presensi Mahasiswa karena secara konsep mirip.		
Luaran Tambahan :	(tidak ada)		

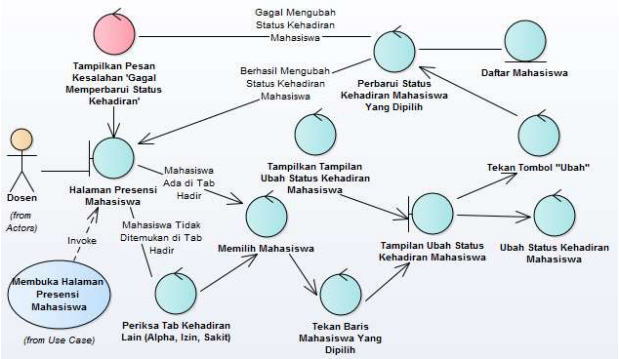
D.32 UC-202 Iterasi 000

Use Case	UC-202 Dosen Mengubah Status Kehadiran Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Dosen memilih akses dosen, melakukan login, melihat daftar mahasiswa yang ditampilkan oleh aplikasi, memilih mahasiswa mana yang akan diubah status kehadirannya, lalu mengubah status kehadiran mahasiswa tersebut. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan. • Data mahasiswa tidak ada : lapor ke TU.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

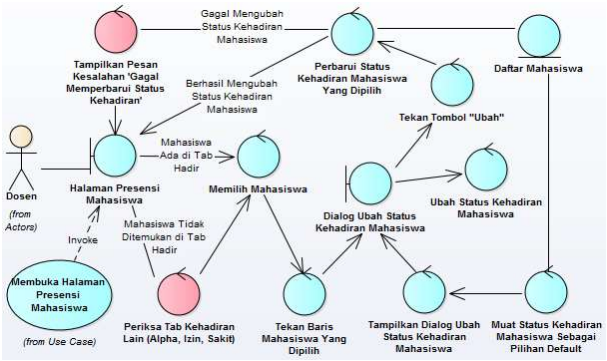
D.33 UC-202 Iterasi 001

Use Case	UC-202 Dosen Mengubah Status Kehadiran Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen mencari Mahasiswa mana yang ingin diubah status kehadirannya. Jika sudah didapat, dosen menekan baris mahasiswa yang dipilih yang akan menyebabkan sistem menampilkan Tampilan Ubah Status Kehadiran Mahasiswa yang dipilih. Dosen selanjutnya bisa mengubah Status Kehadiran Mahasiswa dan menekan Tombol "Ubah" untuk memperbarui status kehadiran mahasiswa yang dipilih. Sistem selanjutnya akan membawa dosen kembali ke Halaman Presensi Mahasiswa. **Alternate Scenario :** • Mahasiswa Tidak Ditemukan Di Tab Hadir : Periksa Tab Kehadiran Lain (Alpha, Izin, Sakit). • Gagal Memperbarui Halaman Status Kehadiran Mahasiswa : tampilkan pesan kesalahan 'Gagal Memperbarui Status Kehadiran Mahasiswa' di Halaman Presensi Mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus beberapa komponen yang sudah menjadi bagian dari use case lain (membuka halaman presensi mahasiswa). - Memperjelas alur yang terjadi dari Halaman Presensi Mahasiswa menuju Tampilan Ubah Status Kehadiran Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

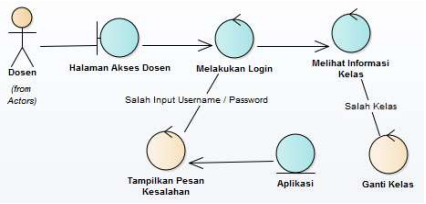
D.34 UC-202 Iterasi 002

Use Case	UC-202 Dosen Mengubah Status Kehadiran Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen mencari Mahasiswa mana yang ingin diubah status kehadirannya. Jika sudah didapat, dosen menekan baris mahasiswa yang dipilih yang akan menyebabkan sistem menampilkan Tampilan Ubah Status Kehadiran Mahasiswa yang dipilih. Dosen selanjutnya bisa mengubah Status Kehadiran Mahasiswa dan menekan Tombol "Ubah" untuk memperbarui status kehadiran mahasiswa yang dipilih. Sistem selanjutnya akan membawa dosen kembali ke Halaman Presensi Mahasiswa. **Alternate Scenario :** • Mahasiswa Tidak Ditemukan Di Tab Hadir : Periksa Tab Kehadiran Lain (Alpha, Izin, Sakit). • Gagal Memperbarui Halaman Status Kehadiran Mahasiswa : tampilkan pesan kesalahan 'Gagal Memperbarui Status Kehadiran Mahasiswa' di Halaman Presensi Mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus hubungan secara langsung antar Entity Daftar Mahasiswa dengan Boundary Halaman Presensi Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

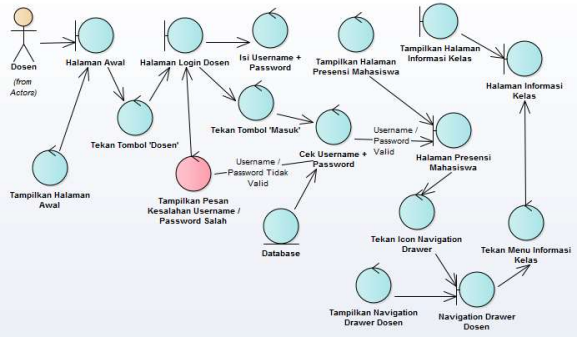
D.35 UC-202 Iterasi 003

Use Case	UC-202 Dosen Mengubah Status Kehadiran Mahasiswa	Iterasi	003
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman Presensi Mahasiswa', Dosen memilih mahasiswa mana yang ingin diubah status kehadirannya di 'Tab Hadir'. Jika sudah didapat, Dosen menekan 'Baris Mahasiswa' yang dipilih. Sistem menampilkan 'Tampilan Ubah Status Kehadiran Mahasiswa' berdasarkan 'Baris Mahasiswa' yang dipilih. Sistem menampilkan 'Dialog Ubah Status Kehadiran' dan memuat status kehadiran mahasiswa sebagai pilihan default pada 'Dialog Ubah Status Kehadiran'. Dosen selanjutnya bisa mengubah status kehadiran mahasiswa dan menekan Tombol "Ubah". Sistem merespon dengan cara memperbarui status kehadiran mahasiswa yang dipilih lalu membawa Dosen kembali ke 'Halaman Presensi Mahasiswa'. **Alternate Scenario :** • Mahasiswa tidak ditemukan di 'Tab Hadir' : Dosen memeriksa 'Tab Kehadiran Lain' (Alpha, Izin, Sakit) lalu memilih mahasiswa dari sana. • Gagal mengubah status kehadiran mahasiswa : tampilkan pesan kesalahan 'Gagal Memperbarui Status Kehadiran Mahasiswa' di 'Halaman Presensi Mahasiswa'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan Controller 'Muat Status Kehadiran Mahasiswa Sebagai Pilihan Default' dan menghubungkannya dengan Controller 'Tampilkan Dialog Ubah Status Kehadiran Mahasiswa'.		
Luaran Tambahan :	(tidak ada)		

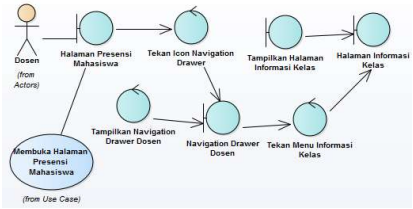
D.36 UC-203 Iterasi 000

Use Case	UC-203 Dosen Melihat Informasi Kelas	Iterasi	000
Use Case Description :	**Basic Scenario :** Dosen memilih akses dosen, melakukan login, lalu memilh menu Informasi Kelas **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan • Salah Kelas : ganti kelas.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

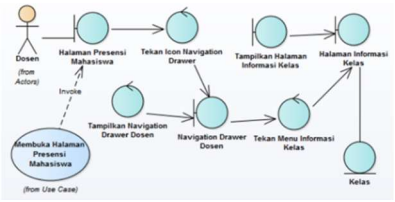
D.37 UC-203 Iterasi 001

Use Case	UC-203 Dosen Melihat Informasi Kelas	Iterasi	001
Use Case Description :	**Basic Scenario :** Ketika dosen membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. Dosen selanjutnya menekan tombol 'Dosen' dan sistem akan menampilkan 'Halaman Login Dosen', dimana dosen bisa mengisi Username + Password dan menekan tombol 'Masuk'. Ketika dosen menekan tombol 'Masuk' Sistem mengecek Username + Password yang dimasukkan berdasar database yang ada, jika valid maka sistem akan menampilkan 'Halaman Presensi Mahasiswa'. Dosen selanjutnya menekan Icon Navigation Drawer yang berakibat sistem menampilkan Navigation Drawer Dosen. Di navigation drawer tersebut, dosen bisa memilih menu Informasi Kelas dengan cara menekan menu tersebut. Sistem selanjutnya akan menampilkan Halaman Informasi Kelas dimana dosen bisa melihat Informasi terkait kelas yang dibutuhkannya. **Alternate Scenario :** • Username / Password tidak valid : tampilkan pesan kesalahan 'Username / Password Salah' di Halaman Login Dosen.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperjelas alur yang terjadi dari Halaman Awal menuju Informasi Dosen sesuai dengan alur yang ada di prototipe. - Memperbaiki Alternate Scenario menjadi lebih fokus ke interaksi antara aplikasi (atau sistem) ke pengguna.		
Luaran Tambahan :	RAUI-002		

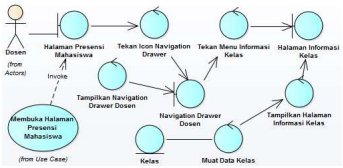
D.38 UC-203 Iterasi 002

Use Case	UC-203 Dosen Melihat Informasi Kelas	Iterasi	002
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Dosen dimana dosen menekan Menu Informasi Kelas. Sistem selanjutnya membawa dosen ke Halaman Informasi Kelas. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memisahkan sebagian besar komponen menjadi use case baru (membuka halaman informasi mahasiswa).		
Luaran Tambahan :	(tidak ada)		

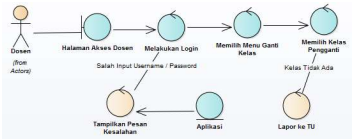
D.39 UC-203 Iterasi 003

Use Case	UC-203 Dosen Melihat Informasi Kelas	Iterasi	003
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Dosen dimana dosen menekan Menu Informasi Kelas. Sistem selanjutnya membawa dosen ke Halaman Informasi Kelas. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan Entity Kelas yang terhubung dengan Halaman Informasi Kelas. - Merevisi relationship Use Case Membuka Halaman Presensi Mahasiswa menjadi invoke ke Boundary Halaman Presensi Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

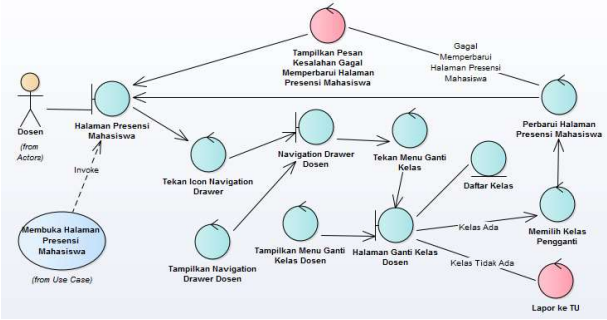
D.40 UC-203 Iterasi 004

Use Case	UC-204 Dosen Melihat Informasi Kelas	Iterasi	004
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman Presensi Mahasiswa', Dosen menekan 'Icon Navigation Drawer' yang ada di 'Halaman Presensi Mahasiswa'. Selanjutnya Sistem menampilkan 'Navigation Drawer Dosen' dimana Dosen menekan 'Menu Informasi Kelas'. Sistem selanjutnya memuat data kelas pada 'Halaman Informasi Kelas' dan menampilkan 'Halaman Informasi Kelas' untuk Dosen. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Merevisi hubungan antara Entity 'Daftar Mahasiswa' dengan Boundary 'Halaman Daftar Mahasiswa' dari secara langsung menjadi melalui Controller 'Muat Data Kelas' dan Controller 'Tampilkan Halaman Daftar Mahasiswa'.		
Luaran Tambahan :	(tidak ada)		

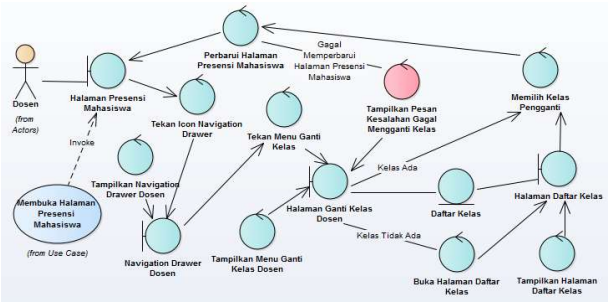
D.41 UC-204 Iterasi 000

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	000
Use Case Description :	**Basic Scenario :** Dosen memilih akses dosen, melakukan login, memilh menu ganti kelas dan memilih kelas pengganti. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan. • Kelas tidak ada : lapor ke TU.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

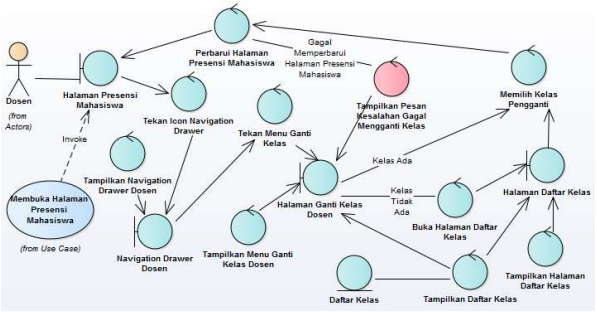
D.42 UC-204 Iterasi 001

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	001
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Dosen dimana dosen menekan Menu Ganti Kelas. Sistem selanjutnya menampilkan Halaman Ganti Kelas Dosen yang menampilkan daftar kelas. Dosen memilih kelas pengganti dari daftar kelas yang ada. Setelah kelas dipilih, sistem akan memperbarui Halaman Presensi Mahasiswa dan selanjutnya membawa dosen kembali ke Halaman Presensi Mahasiswa. **Alternate Scenario :** • Kelas Tidak Ada : Lapor ke TU. • Gagal Memperbarui Halaman Presensi Mahasiswa : tampilkan pesan kesalahan 'Gagal Memperbarui Halaman Presensi Mahasiswa' di Halaman Presensi Mahasiswa .		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus beberapa komponen yang sudah menjadi bagian dari use case lain (membuka halaman presensi mahasiswa). - Memperjelas alur yang terjadi dari Halaman Presensi Mahasiswa menuju Halaman Ganti Kelas Dosen.		
Luaran Tambahan :	RAUI-002		

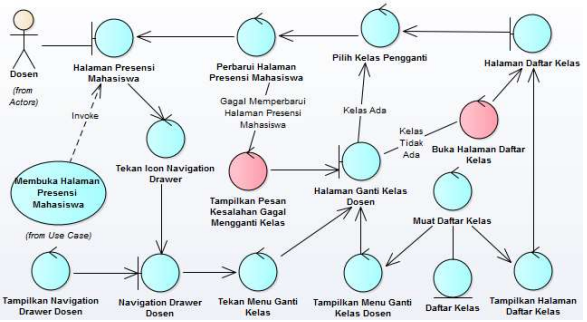
D.43 UC-204 Iterasi 002

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	002
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Dosen dimana dosen menekan Menu Ganti Kelas. Sistem selanjutnya menampilkan Halaman Ganti Kelas Dosen yang menampilkan daftar kelas. Dosen memilih kelas pengganti dari daftar kelas yang ada. Setelah kelas dipilih, sistem akan memperbarui Halaman Presensi Mahasiswa dan selanjutnya membawa dosen kembali ke Halaman Presensi Mahasiswa. **Alternate Scenario :** • Kelas Tidak Ada : Dosen membuka Halaman Daftar Kelas dan memilih kelas pengganti dari daftar kelas yang ada. • Gagal Memperbarui Halaman Presensi Mahasiswa : tampilkan pesan kesalahan 'Gagal Mengganti Kelas' di Halaman Halaman Ganti Kelas Dosen.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki Alternate Case jika kelas tidak ada di Halaman Ganti Kelas Dosen. - Merevisi Controller Tampilan Pesan Kesalahan Gagal Memperbarui Halaman Presensi Mahasiswa.		
Luaran Tambahan :	RAUI-005		

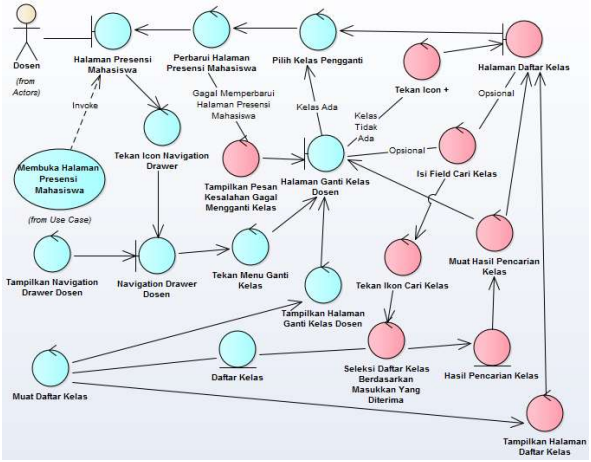
D.44 UC-204 Iterasi 003

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	003
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Dosen dimana dosen menekan Menu Ganti Kelas. Sistem selanjutnya menampilkan Halaman Ganti Kelas Dosen yang menampilkan daftar kelas. Dosen memilih kelas pengganti dari daftar kelas yang ada. Setelah kelas dipilih, sistem akan memperbarui Halaman Presensi Mahasiswa dan selanjutnya membawa dosen kembali ke Halaman Presensi Mahasiswa. **Alternate Scenario :** • Kelas Tidak Ada : Dosen membuka Halaman Daftar Kelas dan memilih kelas pengganti dari daftar kelas yang ada. • Gagal Memperbarui Halaman Presensi Mahasiswa : tampilkan pesan kesalahan 'Gagal Mengganti Kelas' di Halaman Halaman Ganti Kelas Dosen.		
Diagram Robustness:	
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki hubungan antara Entity Daftar Kelas dengan Boundary Halaman Ganti Kelas dan Boundary Halaman Daftar Kelas melalui Controller Tampilkan Daftar Kelas.		
Luaran Tambahan :	(tidak ada)		

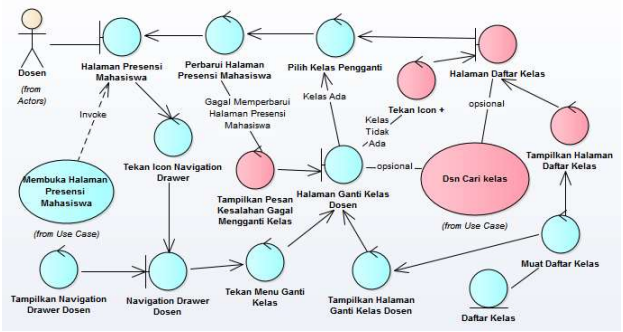
D.45 UC-204 Iterasi 004

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	004
Use Case Description :	**Basic Scenario :** Pada Halaman Presensi Mahasiswa, dosen menekan icon Navigation Drawer. Selanjutnya Sistem menampilkan Navigation Drawer Dosen dimana dosen menekan Menu Ganti Kelas. Sistem selanjutnya menampilkan Halaman Ganti Kelas Dosen dan memilih kelas pengganti dari daftar kelas yang ada. Setelah kelas dipilih, sistem akan memperbarui Halaman Presensi Mahasiswa dan selanjutnya membawa dosen kembali ke Halaman Presensi Mahasiswa. **Alternate Scenario :** • Kelas Tidak Ada : Dosen membuka Halaman Daftar Kelas dan memilih kelas pengganti dari daftar kelas yang ada. • Gagal Memperbarui Halaman Presensi Mahasiswa : tampilkan pesan kesalahan 'Gagal Mengganti Kelas' di Halaman Halaman Ganti Kelas Dosen.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Merevisi Hubungan antara Entity Daftar Kelas dengan Boundary Halaman Ganti Kelas Dosen dan Boundary Halaman Daftar Kelas. - Perbaikan kata - kata minor.		
Luaran Tambahan :	(tidak ada)		

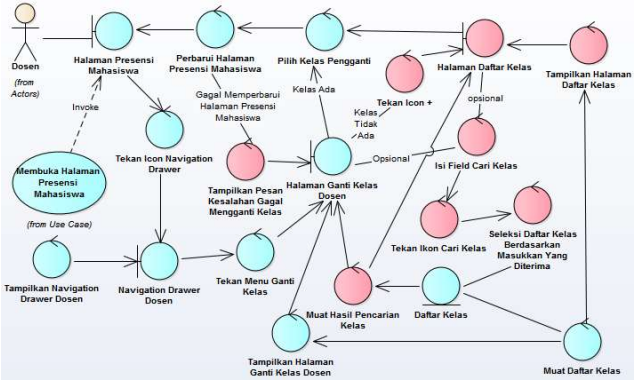
D.46 UC-204 Iterasi 005

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	005
Use Case Description :	**Basic Scenario :** Setelah Dosen membuka 'Halaman Presensi Mahasiswa', Dosen menekan 'Icon Navigation Drawer' yang ada di 'Halaman Presensi Mahasiswa'. Selanjutnya sistem menampilkan 'Navigation Drawer Dosen' dimana Dosen menekan 'Menu Ganti Kelas'. Sistem selanjutnya memuat daftar kelas pada 'Halaman Ganti Kelas Dosen' lalu menampilkan 'Halaman Ganti Kelas Dosen' dimana Dosen bisa memilih kelas pengganti dari daftar kelas yang ada. Setelah kelas dipilih, sistem akan memperbarui 'Halaman Presensi Mahasiswa' dan selanjutnya membawa dosen kembali ke 'Halaman Presensi Mahasiswa'. **Alternate Scenario :** • Kelas tidak ada : Dosen menekan 'Icon +', sistem memuat daftar kelas untuk 'Halaman Daftar Kelas' dan menampilkan 'Halaman Daftar Kelas'. Dosen memilih kelas pengganti pada 'Halaman Daftar Kelas'. • Gagal memperbarui 'Halaman Presensi Mahasiswa' : tampilkan pesan kesalahan 'Gagal Mengganti Kelas' di 'Halaman Ganti Kelas Dosen'. • Opsional, Dosen mencari kelas pada 'Halaman Ganti Kelas Dosen' dan 'Halaman Daftar Kelas' : Dosen mengisi 'Field Cari Kelas' lalu menekan 'Ikon Cari Kelas'. Sistem memproses perilaku user dengan cara melakukan seleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas di 'Halaman Ganti Kelas Dosen' atau 'Halaman Daftar Kelas'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Penambahan alternate scenario opsional. - Memperbaiki alternate scenario kelas tidak ada.		
Luaran Tambahan :	(tidak ada)		

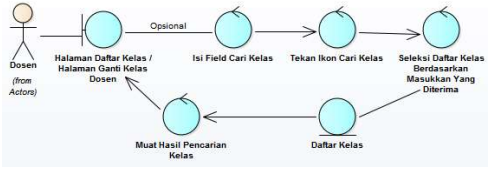
D.47 UC-204 Iterasi 006

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	006
Use Case Description :	**Basic Scenario :** Setelah Dosen membuka 'Halaman Presensi Mahasiswa', Dosen menekan 'Icon Navigation Drawer' yang ada di 'Halaman Presensi Mahasiswa'. Selanjutnya sistem menampilkan 'Navigation Drawer Dosen' dimana Dosen menekan 'Menu Ganti Kelas'. Sistem selanjutnya memuat daftar kelas pada 'Halaman Ganti Kelas Dosen' lalu menampilkan 'Halaman Ganti Kelas Dosen' dimana Dosen bisa memilih kelas pengganti dari daftar kelas yang ada. Setelah kelas dipilih, sistem akan memperbarui 'Halaman Presensi Mahasiswa' dan selanjutnya membawa dosen kembali ke 'Halaman Presensi Mahasiswa'. **Alternate Scenario :** • Kelas tidak ada : Dosen menekan 'Icon +', sistem memuat daftar kelas untuk 'Halaman Daftar Kelas' dan menampilkan 'Halaman Daftar Kelas'. Dosen memilih kelas pengganti pada 'Halaman Daftar Kelas'. • Gagal memperbarui 'Halaman Presensi Mahasiswa' : tampilkan pesan kesalahan 'Gagal Mengganti Kelas' di 'Halaman Ganti Kelas Dosen'. • Opsional, Dosen mencari kelas pada 'Halaman Ganti Kelas Dosen' dan 'Halaman Daftar Kelas' : Use Case Dsn – Cari Kelas.		
Diagram Robustness:	
Pembeda Dengan Iterasi Sebelumnya :	- Memisahkan beberapa elemen diagram menjadi use case baru (Dsn Cari Kelas) dan menghubungkan use case ke dalam diagram ini.		
Luaran Tambahan :	(tidak ada)		

D.48 UC-204 Iterasi 007

Use Case	UC-204 Dosen Mengganti Kelas	Iterasi	007
Use Case Description :	**Basic Scenario :** Setelah Dosen membuka 'Halaman Presensi Mahasiswa', Dosen menekan 'Icon Navigation Drawer' yang ada di 'Halaman Presensi Mahasiswa'. Selanjutnya sistem menampilkan 'Navigation Drawer Dosen' dimana Dosen menekan 'Menu Ganti Kelas'. Sistem selanjutnya memuat daftar kelas pada 'Halaman Ganti Kelas Dosen' lalu menampilkan 'Halaman Ganti Kelas Dosen' dimana Dosen bisa memilih kelas pengganti dari daftar kelas yang ada. Setelah kelas dipilih, sistem akan memperbarui 'Halaman Presensi Mahasiswa' dan selanjutnya membawa dosen kembali ke 'Halaman Presensi Mahasiswa'. **Alternate Scenario :** • Kelas tidak ada : Dosen menekan 'Icon +', sistem memuat daftar kelas untuk 'Halaman Daftar Kelas' dan menampilkan 'Halaman Daftar Kelas'. Dosen memilih kelas pengganti pada 'Halaman Daftar Kelas'. • Gagal memperbarui 'Halaman Presensi Mahasiswa' : tampilkan pesan kesalahan 'Gagal Mengganti Kelas' di 'Halaman Ganti Kelas Dosen'. • Opsional, Dosen mencari kelas pada 'Halaman Ganti Kelas Dosen' dan 'Halaman Daftar Kelas' : Dosen mengisi 'Field Cari Kelas' lalu menekan 'Ikon Cari Kelas'. Sistem memproses perilaku user dengan cara melakukan seleksi data kelas berdasar masukan yang diterima lalu memuat hasil pencarian kelas di 'Halaman Ganti Kelas Dosen' / 'Halaman Daftar kelas'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus use case Dsn – Cari Kelas karena fungsi search yang diajukan tidak membutuhkan cara yang rumit dan menambahkan beberapa elemen untuk mendukung fungsi cari kelas.		
Luaran Tambahan :	(tidak ada)		

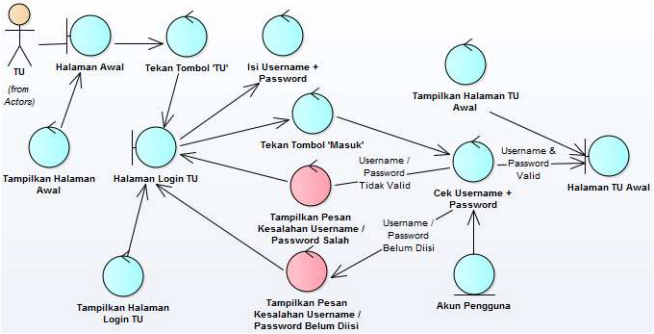
D.49 DUC-202 Iterasi 000

Use Case	DUC-202 Dosen Mencari Kelas	Iterasi	000
Use Case Description :	**Basic Scenario :** Pada 'Halaman Daftar Kelas' / 'Halaman Ganti Kelas', dosen mengisi field cari kelas lalu menekan 'Ikon Cari Kelas'. Sistem menyeleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas pada 'Halaman Daftar Kelas'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

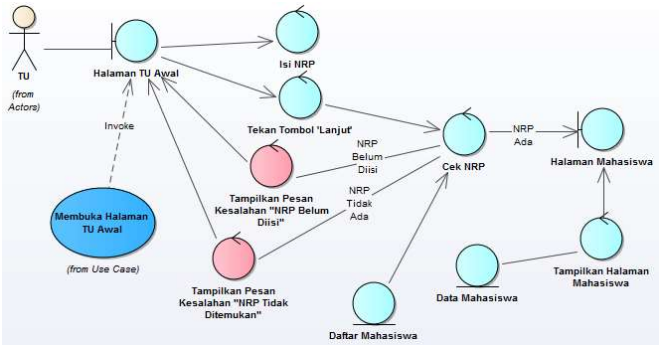
D.50 DUC-201 Iterasi 001

Use Case	DUC-202 Dosen Mencari Kelas	Iterasi	001
Use Case Description :	**Basic Scenario :** (tidak ada) **Alternate Scenario :** (tidak ada)		
Diagram Robustness:	(tidak ada)		
Pembeda Dengan Iterasi Sebelumnya :	- Use Case dihapus dan digabungkan secara langsung dengan Use Case yang meng-extend Use Case ini. Pertimbangannya karena penggunaan fitur search sederhana sehingga tidak perlu dipecah menjadi Use Case baru.		
Luaran Tambahan :	(tidak ada)		

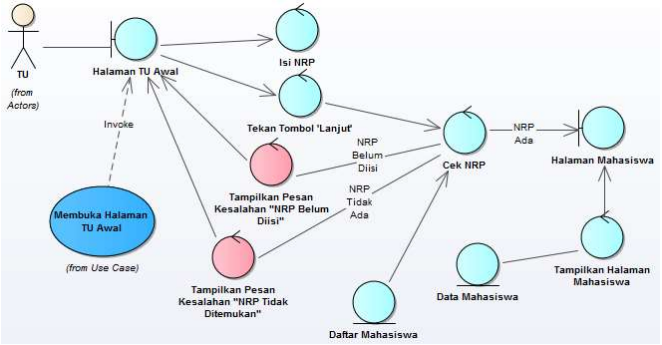
D.51 UC-301 Iterasi 000

Use Case	UC-301 TU Membuka Halaman Awal	Iterasi	000
Use Case Description :	**Basic Scenario :** Ketika TU membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. TU menekan tombol 'TU' untuk mengakses 'Halaman Login TU' lalu mengisi Username + Password dan menekan tombol 'Masuk'. Sistem mengecek Username + Password berdasarkan akun pengguna yang ada, jika valid maka sistem akan menampilkan 'Halaman TU Awal'. **Alternate Scenario :** • Username / Password tidak valid : tampilkan pesan kesalahan 'Username / Password Salah' di 'Halaman Login TU'. • Username / Password belum diisi : tampilkan pesan kesalahan 'Username / Password Belum Diisi' di 'Halaman Login TU'.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

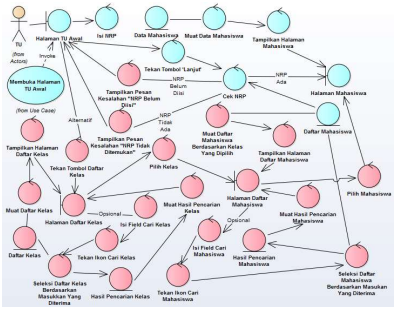
D.52 UC-302 Iterasi 000

Use Case	UC-302 TU Membuka Halaman Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Pada halaman TU Awal, TU mengisi NRP mahasiswa yang akan dilihat / diubah informasinya. Ketika TU menekan tombol 'Lanjut', sistem akan mengecek NRP. Jika NRP ada dalam Daftar Mahasiswa, maka sistem akan menampilkan Halaman Mahasiswa. **Alternate Scenario :** • NRP tidak ada di dalam Daftar Mahasiswa : tampilkan pesan kesalahan 'NRP Tidak Ditemukan' di Halaman TU Awal. • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di Halaman TU Awal.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

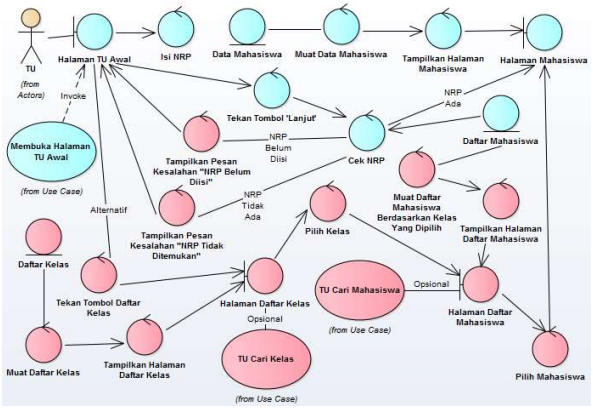
D.53 UC-302 Iterasi 001

Use Case	UC-302 TU Membuka Halaman Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Pada halaman TU Awal, TU mengisi NRP mahasiswa yang akan dilihat / diubah informasinya. Ketika TU menekan tombol 'Lanjut', sistem akan mengecek NRP. Jika NRP ada dalam Daftar Mahasiswa, maka sistem akan menampilkan Halaman Mahasiswa. **Alternate Scenario :** • NRP tidak ada di dalam Daftar Mahasiswa : tampilkan pesan kesalahan 'NRP Tidak Ditemukan' di Halaman TU Awal. • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di Halaman TU Awal.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan Entity Data Mahasiswa dan menghubungkannya ke Boundary Halaman Mahasiswa Melalui Bebeberapa Controller.		
Luaran Tambahan :	(tidak ada)		

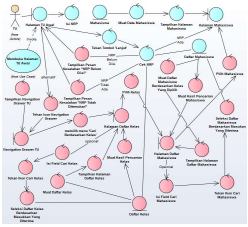
D.54 UC-302 Iterasi 002

Use Case	UC-302 TU Membuka Halaman Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** Pada halaman TU Awal, TU mengisi NRP mahasiswa yang akan dilihat / diubah informasinya. Ketika TU menekan tombol 'Lanjut', sistem akan mengecek NRP. Jika NRP ada dalam Daftar Mahasiswa, maka sistem akan menampilkan Halaman Mahasiswa. **Alternate Scenario :** • NRP tidak ada di dalam Daftar Mahasiswa : tampilkan pesan kesalahan 'NRP Tidak Ditemukan' di Halaman TU Awal. • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di Halaman TU Awal. • Alternatif, melalui daftar kelas : TU menekan tombol 'Daftar Kelas'. sistem memuat daftar kelas dan menampilkannya pada Halaman Daftar Kelas serta membawa TU ke Halaman Daftar Kelas. TU memilih kelas dengan cara menekan baris kelas yang dipilihnya. Sistem merespon dengan cara memuat daftar mahasiswa berdasarkan kelas yang dipilih oleh TU pada Halaman Daftar Mahasiswa serta menampilkan Halaman Daftar Mahasiswa. TU memilih mahasiswa pada Halaman Daftar Mahasiswa dengan cara menekan baris mahasiswa yang dipilihnya dan sistem akan memuat data mahasiswa dan menampilkan halaman mahasiswa untuk TU. • Opsional, TU Mencari Kelas pada Halaman Daftar Kelas : TU mengisi Field Cari Kelas lalu menekan Ikon Cari Kelas. Sistem memproses perilaku user dengan cara melakukan seleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas di halaman Daftar Kelas. • Opsional, TU Mencari Mahasiswa Pada Halaman Daftar Mahasiswa : TU mengisi Field Cari Mahasiswa lalu menekan Ikon Cari Mahasiswa. Sistem memproses perilaku user dengan cara melakukan seleksi daftar mahasiswa berdasarkan masukan yang diterima lalu memuat hasil pencarian mahasiswa di Halaman Daftar Mahasiswa.		
Diagram Robustness:	
Pembeda Dengan Iterasi Sebelumnya :	- Penambahan Alternate Scenario ketika pengguna memilih mencari data mahasiswa melalui daftar kelas serta ketika pengguna menggunakan fungsi search.		
Luaran Tambahan :	(tidak ada)		

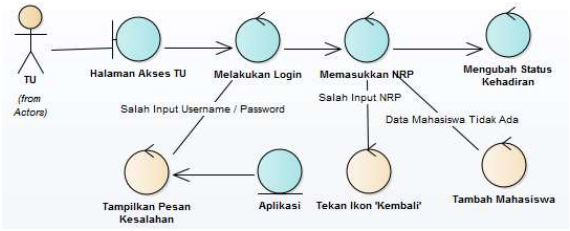
D.55 UC-302 Iterasi 003

Use Case	UC-302 TU Membuka Halaman Mahasiswa	Iterasi	003
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman TU Awal', TU mengisi NRP mahasiswa yang akan dilihat / diubah informasinya pada 'Halaman TU Awal'. Ketika TU menekan tombol 'Lanjut', sistem akan mengecek NRP. Jika NRP ada dalam daftar mahasiswa, memuat data mahasiswa untuk 'Halaman Mahasiswa' lalu menampilkan 'Halaman Mahasiswa'. **Alternate Scenario :** • NRP tidak ada di dalam daftar mahasiswa : tampilkan pesan kesalahan 'NRP Tidak Ditemukan' di 'Halaman TU Awal'. • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di 'Halaman TU Awal'. • Alternatif, melalui 'Halaman Daftar Kelas' : TU menekan tombol 'Daftar Kelas'. sistem memuat daftar kelas dan menampilkannya pada 'Halaman Daftar Kelas' serta membawa TU ke 'Halaman Daftar Kelas'. TU memilih kelas dengan cara menekan baris kelas yang dipilihnya. Sistem merespon dengan cara memuat daftar mahasiswa berdasarkan kelas yang dipilih oleh TU pada 'Halaman Daftar Mahasiswa' serta menampilkan 'Halaman Daftar Mahasiswa'. TU memilih mahasiswa pada 'Halaman Daftar Mahasiswa' dengan cara menekan baris mahasiswa yang dipilihnya dan sistem akan memuat data mahasiswa dan menampilkan 'Halaman Mahasiswa' untuk TU. • Opsional, TU mencari kelas menggunakan search box : Use Case TU - Cari Kelas. • Opsional, TU mencari mahasiswa menggunakan search box : Use Case TU - Cari Mahasiswa.		
Diagram Robustness:	
Pembeda Dengan Iterasi Sebelumnya :	- Memisahkan beberapa elemen diagram menjadi use case baru (TU Cari Kelas dan TU Cari Mahasiswa) dan menghubungkan use case ke dalam diagram ini.		
Luaran Tambahan :	(tidak ada)		

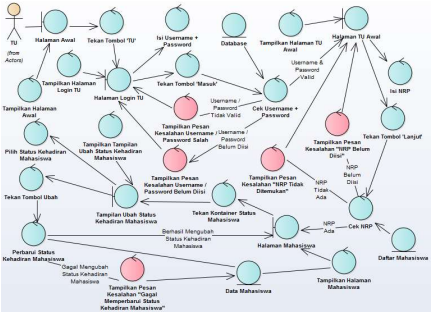
D.56 UC-302 Iterasi 004

Use Case	UC-302 TU Membuka Halaman Mahasiswa	Iterasi	004
Use Case Description :	**Basic Scenario :** Setelah membuka 'Halaman TU Awal', TU mengisi NRP mahasiswa yang akan dilihat / diubah informasinya pada 'Halaman TU Awal'. Ketika TU menekan tombol 'Lanjut', sistem akan mengecek NRP. Jika NRP ada dalam daftar mahasiswa, memuat data mahasiswa untuk 'Halaman Mahasiswa' lalu menampilkan 'Halaman Mahasiswa'. **Alternate Scenario :** • NRP tidak ada di dalam daftar mahasiswa : tampilkan pesan kesalahan 'NRP Tidak Ditemukan' di 'Halaman TU Awal'. • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di 'Halaman TU Awal'. • Alternatif, melalui 'Halaman Daftar Kelas' : TU menekan Icon Navigation Drawer. Sistem selanjutnya menampilkan Navigation Drawer TU. TU memilih menu 'Cari Berdasarkan Kelas'. sistem memuat daftar kelas dan menampilkannya pada 'Halaman Daftar Kelas' serta membawa TU ke 'Halaman Daftar Kelas'. TU memilih kelas dengan cara menekan baris kelas yang dipilihnya. Sistem merespon dengan cara memuat daftar mahasiswa berdasarkan kelas yang dipilih oleh TU pada 'Halaman Daftar Mahasiswa' serta menampilkan 'Halaman Daftar Mahasiswa'. TU memilih mahasiswa pada 'Halaman Daftar Mahasiswa' dengan cara menekan baris mahasiswa yang dipilihnya dan sistem akan memuat data mahasiswa dan menampilkan 'Halaman Mahasiswa' untuk TU. • Opsional, TU Mencari Kelas pada Halaman Daftar Kelas : TU mengisi Field Cari Kelas lalu menekan Ikon Cari Kelas. Sistem memproses perilaku user dengan cara melakukan seleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas di halaman Daftar Kelas. • Opsional, TU Mencari Mahasiswa Pada Halaman Daftar Mahasiswa : TU mengisi Field Cari Mahasiswa lalu menekan Ikon Cari Mahasiswa. Sistem memproses perilaku user dengan cara melakukan seleksi daftar mahasiswa berdasarkan masukan yang diterima lalu memuat hasil pencarian mahasiswa di Halaman Daftar Mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus use case TU – Cari Mahasiswa dan TU – Cari Kelas karena fungsi search yang diajukan tidak membutuhkan cara yang rumit dan menambahkan beberapa elemen untuk mendukung fungsi cari mahasiswa dan cari kelas.		
Luaran Tambahan :	Dokumen HFP 2.1		

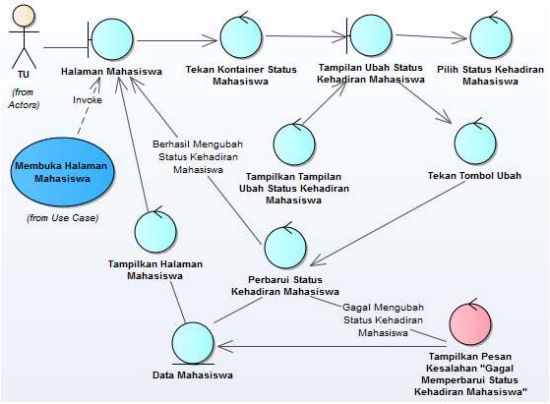
D.57 UC-303 Iterasi 000

Use Case	UC-303 TU Mengubah Status Kehadiran Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** TU memilih akses TU, melakukan login, memasukkan NRP mahasiswa yang bersangkutan, lalu mengubah status kehadiran mahasiswa tersebut. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan. • Salah Input NRP mahasiswa lain : tekan ikon 'kembali' / tombol 'kembali' di gadget . • Data mahasiswa tidak ada : tambah mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

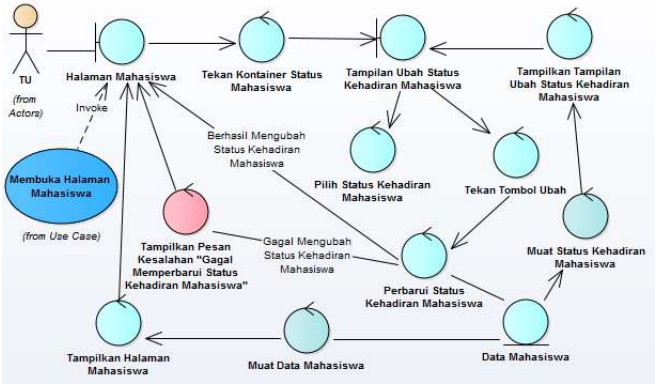
D.58 UC-303 Iterasi 001

Use Case	UC-303 TU Mengubah Status Kehadiran Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Ketika TU membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. TU menekan tombol 'TU' untuk mengakses 'Halaman Login TU' lalu mengisi Username + Password dan menekan tombol 'Masuk'. Sistem mengecek Username + Password yang dimasukkan, jika sesuai dengan yang berada di dalam 'Database' maka sistem akan menampilkan 'Halaman Masukkan NRP TU'. Pada Halaman TU Awal, TU mengisi NRP Mahasiswa yang ingin diubah status kehadirannya dan menekan tombol 'Lanjut'. Ketika TU menekan tombol 'Lanjut' Sistem mengecek NRP yang dimasukkan, jika NRP tersebut ada di dalam 'Daftar Mahasiswa' maka sistem akan menampilkan 'Halaman Mahasiswa. Pada Halaman Mahasiswa, TU menekan Kontainer Status Mahasiswa yang berakibat pada munculnya Tampilan Ubah Status Kehadiran Mahasiswa. TU memilih status kehadiran mahasiswa sesuai dengan status mahasiswa pada saat itu dan menekan tombol 'Ubah'. Sistem selanjutnya akan memperbarui Status Kehadiran Mahasiswa yang terdapat dalam Data Mahasiswa dan membawa TU kembali ke Hamalan Mahasiswa.. **Alternate Scenario :** • Username / Password tidak valid : tampilkan pesan kesalahan 'Username / Password Salah' di Halaman Login TU. • Username / Password belum diisi : tampilkan pesan kesalahan 'Username / Password Belum Diisi' di Halaman Login TU. • NRP belum diisi : tampilkan pesan kesalahan 'NRP Belum Diisi' di Halaman Masukkan NRP TU. • NRP tidak ada di 'Daftar Mahasiswa' : tampilkan pesan kesalahan 'Gagal Masuk' di Halaman Masukkan NRP TU. • Gagal Mengubah Status Kehadiran Mahasiswa : tampilkan pesan kesalahan "Gagal Memperbarui Status Kehardiran Mahasiswa".		
Diagram Robustness:	
Pembeda Dengan Iterasi Sebelumnya :	- Memperjelas alur yang terjadi dari Halaman Presensi Mahasiswa menuju Tampilan Ubah Status Kehadiran Mahasiswa. - Memperbaiki Alternate Scenario menjadi lebih fokus ke interaksi antara aplikasi (atau sistem) ke pengguna.		
Luaran Tambahan :	(tidak ada)		

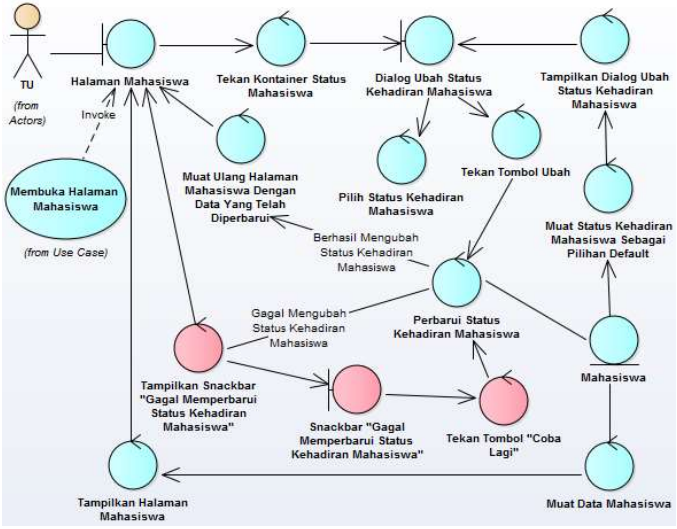
D.59 UC-303 Iterasi 002

Use Case	UC-303 TU Mengubah Status Kehadiran Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** Pada Halaman Mahasiswa, TU menekan Kontainer Status Mahasiswa yang berakibat pada munculnya Tampilan Ubah Status Kehadiran Mahasiswa. TU memilih status kehadiran mahasiswa sesuai dengan status mahasiswa pada saat itu dan menekan tombol 'Ubah'. Sistem selanjutnya akan memperbarui Status Kehadiran Mahasiswa yang terdapat dalam Data Mahasiswa dan membawa TU kembali ke Hamalan Mahasiswa. **Alternate Scenario :** • Gagal Mengubah Status Kehadiran Mahasiswa : tampilkan pesan kesalahan "Gagal Memperbarui Status Kehardiran Mahasiswa".		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memisahkan sebagian besar komponen menjadi use case baru (membuka halaman halaman TU awal dan membuka halaman mahasiswa). - Menghubungkan use case Membuka Halaman Mahasiswa dengan cara Invoke ke Boundary Halaman Mahasiswa (untuk use case membuka halaman TU awal dihubungkan dengan cara invoke di use case Membuka Halaman Mahasiswa).		
Luaran Tambahan :	(tidak ada)		

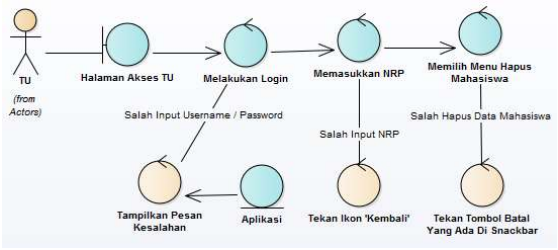
D.60 UC-303 Iterasi 003

Use Case	UC-303 TU Mengubah Status Kehadiran Mahasiswa	Iterasi	003
Use Case Description :	**Basic Scenario :** Pada Halaman Mahasiswa, TU menekan Kontainer Status Mahasiswa. Sistem memproses perilaku TU dengan memuat status Kehadiran Mahasiswa yang ditampilkan di Tampilan Ubah Status Kehadiran Mahasiswa. Pada Tampilan Ubah Status Kehadiran Mahasiswa, TU bisa memilih status kehadiran mahasiswa lalu menekan tombol ubah. Sistem akan memperbarui status kehadiran mahasiswa setelah TU menekan tombol ubah lalu membawa TU kembali ke Halaman Mahasiswa dengan data mahasiswa, terutama status kehadiran mahasiswa, yang telah diperbarui. **Alternate Scenario :** • Gagal Mengubah Status Kehadiran Mahasiswa : tampilkan pesan kesalahan "Gagal Memperbarui Status Kehardiran Mahasiswa" di Halaman Mahasiswa.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki Alternate Scenario. - Memperbaiki hubungan antara Data Mahasiswa dengan menambahkan beberapa controller.		
Luaran Tambahan :	(tidak ada)		

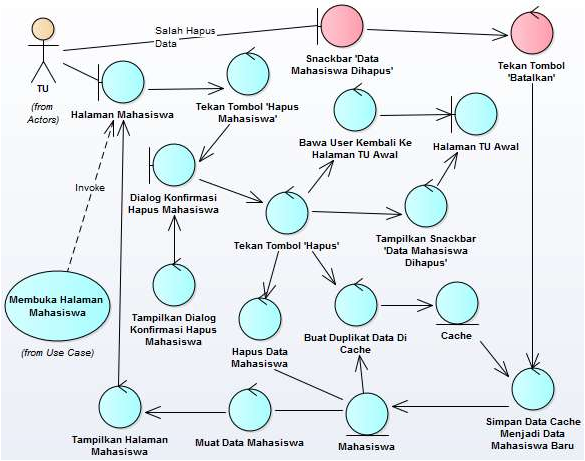
D.61 UC-303 Iterasi 004

Use Case	UC-303 TU Mengubah Status Kehadiran Mahasiswa	Iterasi	004
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman Mahasiswa', TU menekan 'Kontainer Status Mahasiswa'. Sistem merespon dengan cara memproses memuat status kehadiran mahasiswa sebagai pilihan default pada 'Dialog Ubah Status Kehadiran Mahasiswa' yang ditampilkan. Selanjutnya TU bisa memilih status kehadiran mahasiswa lalu menekan tombol 'Ubah' yang akan menyebabkan sistem memperbarui status kehadiran mahasiswa lalu memuat ulang 'Halaman Mahasiswa' dengan data yang telah diperbarui. **Alternate Scenario :** • Gagal mengubah status kehadiran mahasiswa : sistem menampilkan snackbar 'Gagal Memperbarui Status Kehadiran Mahasiswa' di 'Halaman Mahasiswa' yang juga berisi tombol 'Coba Lagi' yang jika ditekan oleh pengguna akan menyebabkan system mencoba memperbarui status kehadiran mahasiswa sekali lagi.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki Alternate Scenario. - Memperbaiki beberapa kata – kata yang ada sehingga lebih sesuai dengan android. - Mengubah Entity 'Data Mahasiswa' menjadi Entity 'Mahasiswa'.		
Luaran Tambahan :	RAUI-006		

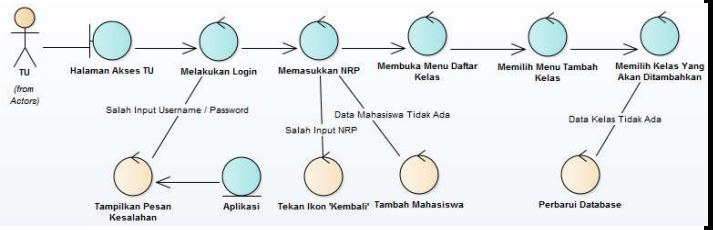
D.62 UC-304 Iterasi 000

Use Case	UC-304 TU Menghapus Data Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** TU memilih akses TU, melakukan login, memasukkan NRP mahasiswa yang bersangkutan, lalu memilih menu hapus mahasiswa. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan. • Salah Input NRP mahasiswa lain : tekan ikon 'kembali' / tombol 'kembali' di gadget . • Data mahasiswa tidak ada : -. • Salah Hapus Data Mahasiswa : gunakan tombol batal yang ada di snackbar.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

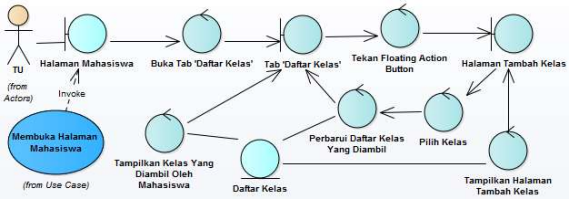
D.63 UC-304 Iterasi 001

Use Case	UC-304 TU Menghapus Data Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman Mahasiswa' yang ingin dihapus, TU lalu menekan tombol 'Hapus Mahasiswa' yang ada di 'Halaman Mahasiswa' tersebut. Sistem menampilkan 'Dialog Konfirmasi Hapus Mahasiswa' untuk memastikan bahwa pengguna TU benar benar ingin menghapus data mahasiswa tersebut. TU menekan tombol 'Hapus' yang akan menyebabkan sistem merespon dengan melakukan beberapa hal, yaitu : membuat duplikat data mahasiswa di cache, menghapus data mahasiswa, membawa pengguna kembali ke 'Halaman TU Awal', dan menampilkan Snackbar 'Data Mahasiswa Dihapus' di 'Halaman TU Awal'. **Alternate Scenario :** • Salah Hapus Data : pada Snackbar 'Data Mahasiswa Dihapus' yang ditampilkan setelah TU menghapus data mahasiswa, Ada tombol 'Batalkan' yang jika ditekan akan menyimpan data cache menjadi data mahasiswa baru.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki alur yang terjadi. - Memperbaiki alternate scenario. - Mengubah Entity 'Data Mahasiswa' menjadi Entity 'Mahasiswa'.		
Luaran Tambahan :	(tidak ada)		

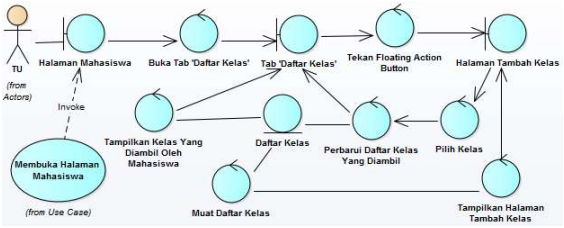
D.64 UC-305 Iterasi 000

Use Case	UC-305 TU Menambah Kelas Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** TU memilih akses TU, melakukan login, memasukkan NRP mahasiswa yang bersangkutan, membuka menu daftar kelas, memilih menu tambah kelas, lalu memilih kelas yang akan ditambahkan. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan. • Salah Input NRP mahasiswa lain : tekan ikon 'kembali' / tombol 'kembali' di gadget. • Data mahasiswa tidak ada : tambah mahasiswa. • Data Kelas tidak ada : perbarui database.		
Diagram Robustness :			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

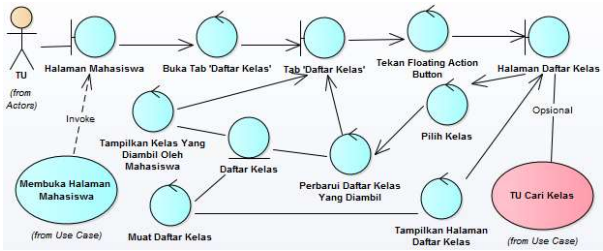
D.65 UC-305 Iterasi 001

Use Case	UC-305 TU Menambah Kelas Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka Halaman Mahasiswa, TU membuka Tab 'Daftar Kelas' dimana sistem menampilkan kelas mana saja yang diambil oleh mahasiswa. TU selanjutnya menekan Floating Action Button dan membawa TU ke Halaman Tambah Kelas yang berisi semua daftar kelas yang ada. Tu selanjutnya memilih kelas dan sistem akan memperbarui daftar kelas yang diambil dan membawa TU kembali ke Tab ' Daftar Kelas'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki alur yang terjadi. - Menghapus Alternate Scenario.		
Luaran Tambahan :	(tidak ada)		

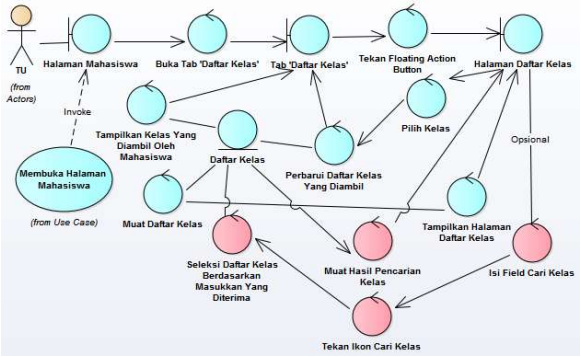
D.66 UC-305 Iterasi 002

Use Case	UC-305 TU Menambah Kelas Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka Halaman Mahasiswa, TU membuka Tab 'Daftar Kelas' dimana sistem menampilkan kelas mana saja yang diambil oleh mahasiswa. TU selanjutnya menekan Floating Action Button dan membawa TU ke Halaman Tambah Kelas yang berisi semua daftar kelas yang ada. TU selanjutnya memilih kelas dan sistem akan memperbarui daftar kelas yang diambil dan membawa TU kembali ke Tab ' Daftar Kelas'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan Controller Muat Daftar Kelas diantara Entity Daftar Kelas dan Controller Tampilkan Halaman Tambah Kelas.		
Luaran Tambahan :	(tidak ada)		

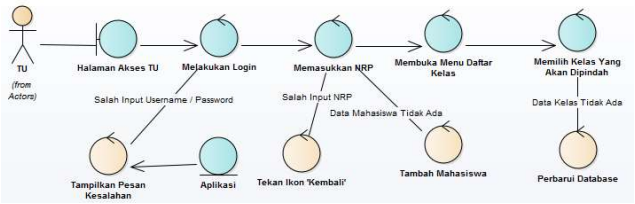
D.67 UC-305 Iterasi 003

Use Case	UC-305 TU Menambah Kelas Mahasiswa	Iterasi	003
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka ‘Halaman Mahasiswa’, TU membuka Tab ‘Daftar Kelas’ dimana sistem menampilkan kelas mana saja yang diambil oleh mahasiswa. TU selanjutnya menekan ‘Floating Action Button’ dan membawa TU ke ‘Halaman Daftar Kelas’ yang berisi semua daftar kelas yang ada. TU selanjutnya memilih kelas dan sistem akan memperbarui daftar kelas yang diambil dan membawa TU kembali ke Tab ‘ Daftar Kelas’. **Alternate Scenario :** • Opsional, TU mencari kelas menggunakan search box : Use Case TU - Cari Kelas.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan alternate scenario opsional. - Mengubah Boundary ‘Halaman Tambah Kelas’ menjadi Boundary ‘Halaman Daftar Kelas’.		
Luaran Tambahan :	(tidak ada)		

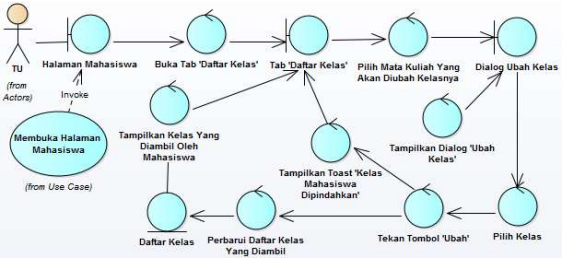
D.68 UC-305 Iterasi 004

Use Case	UC-305 TU Menambah Kelas Mahasiswa	Iterasi	004
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman Mahasiswa', TU membuka Tab 'Daftar Kelas' dimana sistem menampilkan kelas mana saja yang diambil oleh mahasiswa. TU selanjutnya menekan 'Floating Action Button' dan membawa TU ke 'Halaman Daftar Kelas' yang berisi semua daftar kelas yang ada. TU selanjutnya memilih kelas dan sistem akan memperbarui daftar kelas yang diambil dan membawa TU kembali ke Tab ' Daftar Kelas'. **Alternate Scenario :** • Opsional, TU Mencari Kelas pada Halaman Daftar Kelas : TU mengisi Field Cari Kelas lalu menekan Ikon Cari Kelas. Sistem memproses perilaku user dengan cara melakukan seleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas di halaman Daftar Kelas.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus use case TU – Cari Kelas karena fungsi search yang diajukan tidak membutuhkan cara yang rumit dan menambahkan beberapa elemen untuk mendukung fungsi cari kelas.		
Luaran Tambahan :	(tidak ada)		

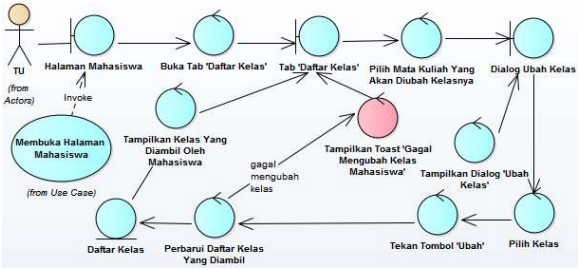
D.69 UC-306 Iterasi 000

Use Case	UC-306 TU Memindah Kelas Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** TU memilih akses TU, melakukan login, memasukkan NRP mahasiswa yang bersangkutan, membuka menu daftar kelas, memilih kelas yang akan dipindah. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan. • Salah Input NRP mahasiswa lain : tekan ikon 'kembali' / tombol 'kembali' di gadget . • Data mahasiswa tidak ada : tambah mahasiswa. • Data Kelas tidak ada : perbarui database.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

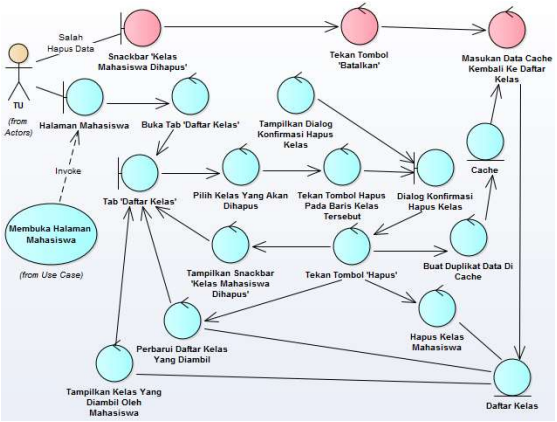
D.70 UC-306 Iterasi 001

Use Case	UC-306 TU Memindah Kelas Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman Mahasiswa', TU membuka Tab 'Daftar Kelas' dimana sistem menampilkan kelas mana saja yang diambil oleh mahasiswa. TU selanjutnya memilih mata kuliah mana yang akan diubah kelasnya. Sistem menampilkan 'Dialog Ubah Kelas' dan TU memilih kelas lalu menekan tombol 'Ubah'. Sistem selanjutnya merespon dengan cara memperbarui daftar kelas yang diambil, membawa TU kembali ke Tab 'Daftar Kelas' dan menampilkan Toast 'Kelas Mahasiswa Dipindah'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki alur yang terjadi. - Menghapus alternate scenario.		
Luaran Tambahan :	(tidak ada)		

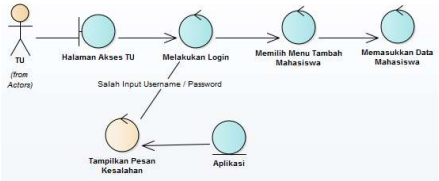
D.71 UC-306 Iterasi 002

Use Case	UC-306 TU Memindah Kelas Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman Mahasiswa', TU membuka Tab 'Daftar Kelas' dimana sistem menampilkan kelas mana saja yang diambil oleh mahasiswa. TU selanjutnya memilih mata kuliah mana yang akan diubah kelasnya. Sistem menampilkan 'Dialog Ubah Kelas' dan TU memilih kelas lalu menekan tombol 'Ubah'. Sistem selanjutnya merespon dengan cara memperbarui daftar kelas yang diambil, membawa TU kembali ke Tab 'Daftar Kelas' dan menampilkan Toast 'Kelas Mahasiswa Dipindah'. **Alternate Scenario :** • Gagal Mengubah Kelas : sistem menampilkan Toast 'Gagal Mengubah Kelas Mahasiswa' di Tab 'Daftar Kelas.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus toast yang dimunculkan setelah memperbarui daftar kelas. - Menambahkan alternate scenario gagal mengubah kelas.		
Luaran Tambahan :	(tidak ada)		

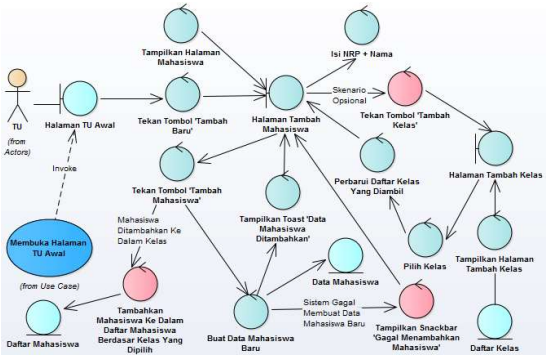
D.72 UC-307 Iterasi 000

Use Case	UC-307 TU Menghapus Kelas Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka Halaman Mahasiswa, TU membuka Tab 'Daftar Kelas' dimana sistem menampilkan kelas mana saja yang diambil oleh mahasiswa. TU selanjutnya memilih mata kelas mana yang akan dihapus dan menekan tombol hapus pada baris kelas tersebut. Sistem menampilkan Dialog Konfirmasi Hapus Kelas untuk memastikan bahwa pengguna TU benar benar ingin menghapus kelas mahasiswa tersebut. TU menekan tombol 'Hapus' yang akan menyebabkan sistem merespon dengan melakukan beberapa hal, yaitu : membuat duplikat data kelas mahasiswa di cache, menghapus kelas mahasiswa, memperbarui datar kelas yang diambil, dan menampilkan Snackbar 'Data Mahasiswa Dihapus' di halaman TU Awal. **Alternate Scenario :** • Salah Hapus Data : pada Snackbar 'Kelas Mahasiswa Dihapus' yang ditampilkan setelah TU menghapus data mahasiswa, Ada tombol 'Batalkan' yang jika ditekan akan memasukkan data kelas mahasiswa yang disimpan di cache kembali ke daftar kelas.		
Diagram Robustness:	
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	RAUI-007		

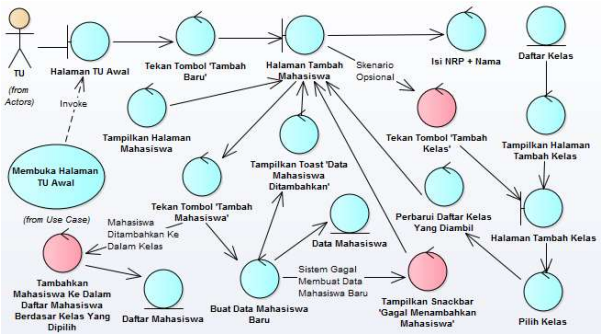
D.73 UC-308 Iterasi 000

Use Case	UC-308 TU Menambah Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** TU memilih akses TU, melakukan login, memilih menu tambah mahasiswa, lalu memasukkan data mahasiswa yang bersangkutan. **Alternate Scenario :** • Salah Input Username / Password : aplikasi menampilkan pesan kesalahan.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

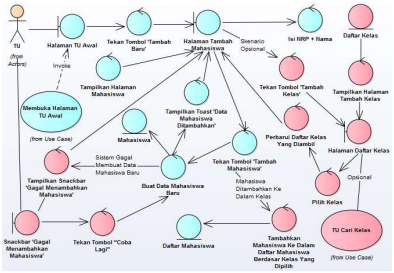
D.74 UC-308 Iterasi 001

Use Case	UC-308 TU Menambah Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka Halaman TU Awal, TU menekan tombol 'Tambah Baru'. Sistem selanjutnya menampilkan Halaman Tambah Mahasiswa dimana TU bisa mengisi NRP + Nama Mahasiswa yang ingin ditambahkan dan menekan tombol 'Tambah Mahasiswa'. Sistem akan memproses perilaku pengguna dengan cara membuat data mahasiswa baru berdasarkan data yang sebelumnya dimasukkan oleh pengguna TU. Setelah itu, system akan membawa TU kembali ke Halaman Tambah Mahasiswa dan menampilkan Toast 'Data Mahasiswa Ditambahkan'. **Alternate Scenario :** • Skenario Opsional : TU memasukkan mahasiswa yang akan ditambahkan ke dalam kelas dengan cara menekan tombol 'Tambah Kelas'. Sistem akan menampilkan Halaman Tambah Kelas yang berisi Daftar Kelas dimana TU bisa memilih kelas untuk Mahasiswa yang bersangkutan. Selanjutnya TU dibawa kembali ke Halaman Tambah Mahasiswa yang daftar kelasnya telah diperbarui. • Sistem Gagal Membuat Data Mahasiswa Baru : tampilkan snackbar "Gagal Menambahkan Mahasiswa" di Halaman Mahasiswa yang juga berisi tombol "Coba Lagi" yang jika ditekan oleh pengguna akan menyebabkan sistem mencoba membuat data mahasiswa baru sekali lagi. • Mahasiswa Ditambahkan Ke Dalam Kelas : Jika pada saat menekan tombol 'Tambah Mahasiswa' daftar kelas yang di ambil ada isinya (yang kemungkinan besar terjadi karena user melakukan Skenario Opsional), maka untuk setiap kelas yang ada di dalam daftar kelas, sistem akan menambahkan mahasiswa ke dalam daftar mahasiswa berdasar kelas yang dipilih..		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Memperbaiki alur yang terjadi sejak TU membuka halaman awal aplikasi. - Memperbaiki dan menambahkan Alternate Scenario.		
Luaran Tambahan :	RAUI-008		

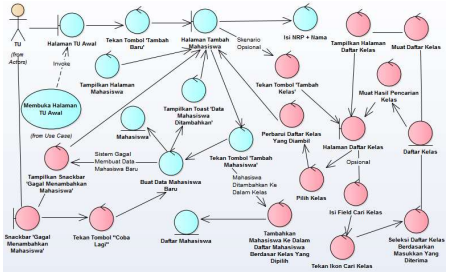
D.75 UC-308 Iterasi 002

Use Case	UC-308 TU Menambah Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka Halaman TU Awal, TU menekan tombol 'Tambah Baru'. Sistem selanjutnya menampilkan Halaman Tambah Mahasiswa dimana TU bisa mengisi NRP + Nama Mahasiswa yang ingin ditambahkan dan menekan tombol 'Tambah Mahasiswa'. Sistem akan memproses perilaku pengguna dengan cara membuat data mahasiswa baru berdasarkan data yang sebelumnya dimasukkan oleh pengguna TU. Setelah itu, system akan membawa TU kembali ke Halaman Tambah Mahasiswa dan menampilkan Toast 'Data Mahasiswa Ditambahkan'. **Alternate Scenario :** • Skenario Opsional : TU memasukkan mahasiswa yang akan ditambahkan ke dalam kelas dengan cara menekan tombol 'Tambah Kelas'. Sistem akan menampilkan Halaman Tambah Kelas yang berisi Daftar Kelas dimana TU bisa memilih kelas untuk Mahasiswa yang bersangkutan. Selanjutnya TU dibawa kembali ke Halaman Tambah Mahasiswa yang daftar kelasnya telah diperbarui. • Sistem Gagal Membuat Data Mahasiswa Baru : tampilkan snackbar "Gagal Memperbarui Status Kehardiran Mahasiswa" di Halaman Mahasiswa yang juga berisi tombol "Coba Lagi" yang jika ditekan oleh pengguna akan menyebabkan sistem mencoba membuat data mahasiswa baru sekali lagi. • Mahasiswa Ditambahkan Ke Dalam Kelas : Jika pada saat menekan tombol 'Tambah Mahasiswa' daftar kelas yang di ambil ada isinya (yang kemungkinan besar terjadi karena user melakukan Skenario Opsional), maka untuk setiap kelas yang ada di dalam daftar kelas, sistem akan menambahkan mahasiswa ke dalam daftar mahasiswa berdasar kelas yang dipilih.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan Entity Daftar Kelas dan menghubungkannya ke Boundary Halaman. Tambah Kelas Melalui Controller Tampilkan Halaman Tambah Kelas.		
Luaran Tambahan :	(tidak ada)		

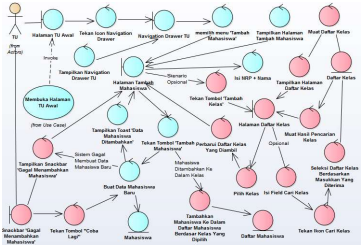
D.76 UC-308 Iterasi 003

Use Case	UC-308 TU Menambah Mahasiswa	Iterasi	003
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman TU Awal', TU menekan tombol 'Tambah Baru'. Sistem selanjutnya menampilkan 'Halaman Tambah Mahasiswa' dimana TU bisa mengisi NRP + Nama Mahasiswa yang ingin ditambahkan dan menekan tombol 'Tambah Mahasiswa'. Sistem akan memproses perilaku pengguna dengan cara membuat data mahasiswa baru berdasarkan data yang sebelumnya dimasukkan oleh pengguna TU. Setelah itu, sistem akan membawa TU kembali ke 'Halaman Tambah Mahasiswa' dan menampilkan Toast 'Data Mahasiswa Ditambahkan'. **Alternate Scenario :** • Skenario Opsional, menambah daftar kelas untuk mahasiswa yang bersangkutan : TU memasukkan mahasiswa yang akan ditambahkan ke dalam kelas dengan cara menekan tombol 'Tambah Kelas'. Sistem akan menampilkan 'Halaman Tambah Kelas' yang berisi Daftar Kelas dimana TU bisa memilih kelas untuk Mahasiswa yang bersangkutan. Selanjutnya TU dibawa kembali ke 'Halaman Tambah Mahasiswa' yang daftar kelasnya telah diperbarui. • Sistem gagal membuat data mahasiswa baru : tampilkan snackbar "Gagal Memperbarui Status Kehardiran Mahasiswa" di Halaman Mahasiswa yang juga berisi tombol "Coba Lagi" yang jika ditekan oleh pengguna akan menyebabkan sistem mencoba membuat data mahasiswa baru sekali lagi. • Mahasiswa ditambahkan ke dalam kelas : Jika pada saat menekan tombol 'Tambah Mahasiswa' daftar kelas yang di ambil ada isinya (yang kemungkinan besar terjadi karena user melakukan Skenario Opsional), maka untuk setiap kelas yang ada di dalam daftar kelas, sistem akan menambahkan mahasiswa ke dalam daftar mahasiswa berdasar kelas yang dipilih. • Opsional, TU mencari kelas menggunakan search box : Use Case TU - Cari Kelas.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menambahkan alternate scenario opsional. - Menambahkan Snackbar 'Gagal Menambahkan Mahasiswa' dan beberapa elemen lainnya untuk mendukung alternate scenario 'Gagal Membuat Data Mahasiswa baru'. - Mengubah Entity 'Data Mahasiswa' menjadi Entity 'Mahasiswa'.		
Luaran Tambahan :	(tidak ada)		

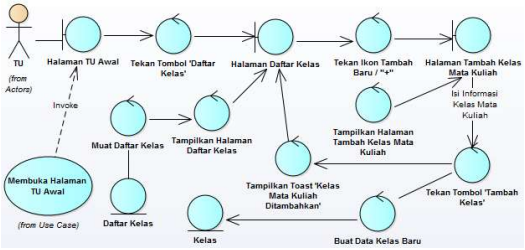
D.77 UC-308 Iterasi 004

Use Case	UC-308 TU Menambah Mahasiswa	Iterasi	004
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman TU Awal', TU menekan tombol 'Tambah Baru'. Sistem selanjutnya menampilkan 'Halaman Tambah Mahasiswa' dimana TU bisa mengisi NRP + Nama Mahasiswa yang ingin ditambahkan dan menekan tombol 'Tambah Mahasiswa'. Sistem akan memproses perilaku pengguna dengan cara membuat data mahasiswa baru berdasarkan data yang sebelumnya dimasukkan oleh pengguna TU. Setelah itu, sistem akan membawa TU kembali ke 'Halaman Tambah Mahasiswa' dan menampilkan Toast 'Data Mahasiswa Ditambahkan'. **Alternate Scenario :** • Skenario Opsional, menambah daftar kelas untuk mahasiswa yang bersangkutan : TU memasukkan mahasiswa yang akan ditambahkan ke dalam kelas dengan cara menekan tombol 'Tambah Kelas'. Sistem akan menampilkan 'Halaman Tambah Kelas' yang berisi Daftar Kelas dimana TU bisa memilih kelas untuk Mahasiswa yang bersangkutan. Selanjutnya TU dibawa kembali ke 'Halaman Tambah Mahasiswa' yang daftar kelasnya telah diperbarui. • Sistem gagal membuat data mahasiswa baru : tampilkan snackbar "Gagal Memperbarui Status Kehardiran Mahasiswa" di Halaman Mahasiswa yang juga berisi tombol "Coba Lagi" yang jika ditekan oleh pengguna akan menyebabkan sistem mencoba membuat data mahasiswa baru sekali lagi. • Mahasiswa ditambahkan ke dalam kelas : Jika pada saat menekan tombol 'Tambah Mahasiswa' daftar kelas yang di ambil ada isinya (yang kemungkinan besar terjadi karena user melakukan Skenario Opsional), maka untuk setiap kelas yang ada di dalam daftar kelas, sistem akan menambahkan mahasiswa ke dalam daftar mahasiswa berdasar kelas yang dipilih. • Opsional, TU Mencari Kelas pada Halaman Daftar Kelas : TU mengisi Field Cari Kelas lalu menekan Ikon Cari Kelas. Sistem memproses perilaku user dengan cara melakukan seleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas di halaman Daftar Kelas.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus use case TU – Cari Kelas karena fungsi search yang diajukan tidak membutuhkan cara yang rumit dan menambahkan beberapa elemen untuk mendukung fungsi cari kelas.		
Luaran Tambahan :	(tidak ada)		

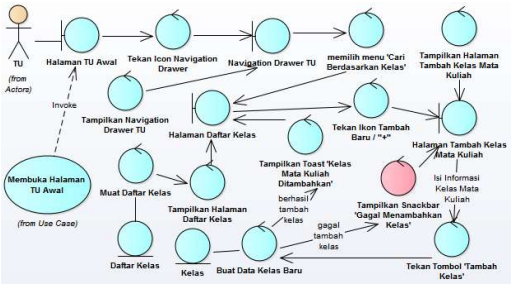
D.78 UC-308 Iterasi 005

Use Case	UC-308 TU Menambah Mahasiswa	Iterasi	005
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman TU Awal', TU menekan Icon Navigation Drawer. Sistem selanjutnya menampilkan Navigation Drawer TU. TU memilih menu 'Tambah Mahasiswa', sistem menampilkan 'Halaman Tambah Mahasiswa' dimana TU bisa mengisi NRP + Nama Mahasiswa yang ingin ditambahkan dan menekan tombol 'Tambah Mahasiswa'. Sistem akan memproses perilaku pengguna dengan cara membuat data mahasiswa baru berdasarkan data yang sebelumnya dimasukkan oleh pengguna TU. Setelah itu, sistem akan membawa TU kembali ke 'Halaman Tambah Mahasiswa' dan menampilkan Toast 'Data Mahasiswa Ditambahkan'. **Alternate Scenario :** • Skenario Opsional, menambah daftar kelas untuk mahasiswa yang bersangkutan : TU memasukkan mahasiswa yang akan ditambahkan ke dalam kelas dengan cara menekan tombol 'Tambah Kelas'. Sistem akan menampilkan 'Halaman Tambah Kelas' yang berisi Daftar Kelas dimana TU bisa memilih kelas untuk Mahasiswa yang bersangkutan. Selanjutnya TU dibawa kembali ke 'Halaman Tambah Mahasiswa' yang daftar kelasnya telah diperbarui. • Sistem gagal membuat data mahasiswa baru : tampilkan snackbar "Gagal Menambahkan Mahasiswa" di Halaman Mahasiswa yang juga berisi tombol "Coba Lagi" yang jika ditekan oleh pengguna akan menyebabkan sistem mencoba membuat data mahasiswa baru sekali lagi. • Mahasiswa ditambahkan ke dalam kelas : Jika pada saat menekan tombol 'Tambah Mahasiswa' daftar kelas yang di ambil ada isinya (yang kemungkinan besar terjadi karena user melakukan Skenario Opsional), maka untuk setiap kelas yang ada di dalam daftar kelas, sistem akan menambahkan mahasiswa ke dalam daftar mahasiswa berdasar kelas yang dipilih. • Opsional, TU Mencari Kelas pada Halaman Daftar Kelas : TU mengisi Field Cari Kelas lalu menekan Ikon Cari Kelas. Sistem memproses perilaku user dengan cara melakukan seleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas di halaman Daftar Kelas.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Perbaikan alur karena adanya revisi prototipe user interface.		
Luaran Tambahan :	Dokumen HFP 2.1		

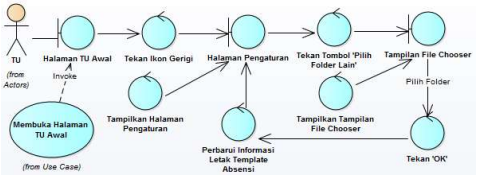
D.79 UC-309 Iterasi 000

Use Case	UC-309 TU Menambah Kelas Mata Kuliah Baru	Iterasi	000
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman TU Awal', TU menekan tombol 'Daftar Kelas'. Sistem merespon dengan cara memuat daftar kelas untuk 'Halaman daftar Kelas' dan menampilkan 'Halaman Daftar Kelas' untuk TU. TU menekan 'Ikon Tambah Baru / +' yang menyebabkan sistem menampilkan 'Halaman Tambah Kelas Mata Kuliah'. TU selanjutnya mengisi informasi kelas mata kuliah dan menekan tombol 'Tambah Kelas' yang akan menyebabkan sistem merespon dengan melakukan beberapa hal, yaitu : membuat data kelas baru di database, menampilkan Toast 'Kelas Mata Kuliah Ditambahkan' di 'Halaman Daftar Kelas' dan membawa TU kembali ke 'Halaman Daftar Kelas'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	RAUI-009		

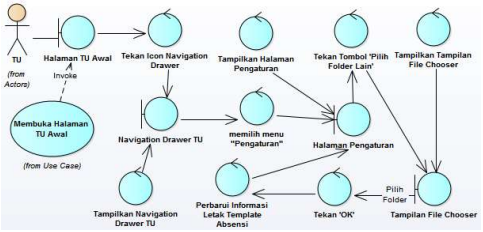
D.80 UC-309 Iterasi 001

Use Case	UC-309 TU Menambah Kelas Mata Kuliah Baru	Iterasi	001
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman TU Awal', TU menekan Icon Navigation Drawer. Sistem selanjutnya menampilkan Navigation Drawer TU. TU memilih menu 'Cari Berdasarkan Kelas'. sistem memuat daftar kelas dan menampilkannya pada 'Halaman Daftar Kelas' serta membawa TU ke 'Halaman Daftar Kelas'. TU menekan 'Ikon Tambah Baru / +' yang menyebabkan sistem menampilkan 'Halaman Tambah Kelas Mata Kuliah'. TU selanjutnya mengisi informasi kelas mata kuliah dan menekan tombol 'Tambah Kelas' yang akan menyebabkan sistem merespon dengan melakukan beberapa hal, yaitu : membuat data kelas baru di database, menampilkan Toast 'Kelas Mata Kuliah Ditambahkan' di 'Halaman Daftar Kelas' dan membawa TU kembali ke 'Halaman Daftar Kelas'. **Alternate Scenario :** • Sistem gagal membuat data mahasiswa baru : tampilkan snackbar "Gagal Menambahkan Kelas" di Halaman Mahasiswa yang juga berisi tombol "Coba Lagi" yang jika ditekan oleh pengguna akan menyebabkan sistem mencoba membuat data kelas mata kuliah baru sekali lagi.		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Perbaikan alur karena adanya revisi prototipe user interface.		
Luaran Tambahan :	Dokumen HFP 2.1		

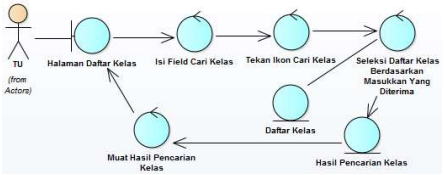
D.81 UC-310 Iterasi 000

Use Case	UC-310 TU Memilih Lokasi Template Presensi	Iterasi	000
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman TU Awal', TU Menekan 'Ikon Gerigi' yang ada di ujung kiri atas tampilan. Sistem menampilkan 'Halaman Pengaturan' dimana TU menekan tombol 'Pilih Folder Lain' yang menamnpilkan 'Tampilan File Chooser' dimana TU bisa memilih folder dan menekan 'OK'. Sistem selanjutnya memperbarui informasi letak template absensi dan membawa TU kembali ke 'Halaman Pengaturan'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	RAUI-010		

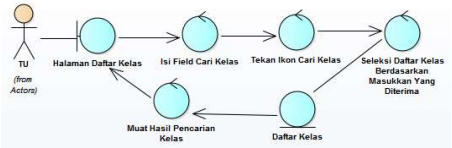
D.82 UC-310 Iterasi 001

Use Case	UC-310 TU Memilih Lokasi Template Presensi	Iterasi	001
Use Case Description :	**Basic Scenario :** Setelah berhasil membuka 'Halaman TU Awal', TU menekan Icon Navigation Drawer. Sistem selanjutnya menampilkan Navigation Drawer TU. TU memilih menu 'Pengaturan'. Sistem menampilkan 'Halaman Pengaturan' dimana TU menekan tombol 'Pilih Folder Lain' yang menamnpilkan 'Tampilan File Chooser' dimana TU bisa memilih folder dan menekan 'OK'. Sistem selanjutnya memperbarui informasi letak template absensi dan membawa TU kembali ke 'Halaman Pengaturan'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Perbaikan alur karena adanya revisi prototipe user interface.		
Luaran Tambahan :	Dokumen HFP 2.1		

D.83 DUC-301 Iterasi 000

Use Case	DUC-301 TU Menari Kelas	Iterasi	000
Use Case Description :	**Basic Scenario :** Pada 'Halaman Daftar Kelas', TU mengisi field cari kelas lalu menekan 'Ikon Cari Kelas'. Sistem menyeleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas pada 'Halaman Daftar Kelas'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

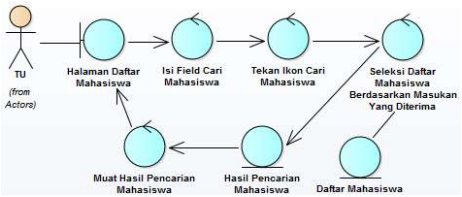
D.84 DUC-301 Iterasi 001

Use Case	DUC-301 TU Mencari Kelas	Iterasi	001
Use Case Description :	**Basic Scenario :** Pada 'Halaman Daftar Kelas', TU mengisi field cari kelas lalu menekan 'Ikon Cari Kelas'. Sistem menyeleksi daftar kelas berdasarkan masukan yang diterima lalu memuat hasil pencarian kelas pada 'Halaman Daftar Kelas'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity Hasil Pencarian Kelas.		
Luaran Tambahan :	(tidak ada)		

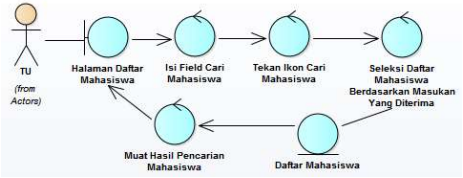
D.85 DUC-301 Iterasi 002

Use Case	DUC-301 TU Mencari Kelas	Iterasi	002
Use Case Description :	**Basic Scenario :** (tidak ada) **Alternate Scenario :** (tidak ada)		
Diagram Robustness:	(tidak ada)		
Pembeda Dengan Iterasi Sebelumnya :	- Use Case dihapus dan digabungkan secara langsung dengan Use Case yang meng-extend Use Case ini. Pertimbangannya karena penggunaan fitur search sederhana sehingga tidak perlu dipecah menjadi Use Case baru.		
Luaran Tambahan :	(tidak ada)		

D.86 DUC-302 Iterasi 000

Use Case	DUC-301 TU Mencari Mahasiswa	Iterasi	000
Use Case Description :	**Basic Scenario :** Pada 'Halaman Daftar Mahasiswa', TU mengisi 'Field Cari Mahasiswa' lalu menekan 'Ikon Cari Mahasiswa'. Sistem menyeleksi daftar mahasiswa berdasarkan masukan yang diterima lalu memuat hasil pencarian mahasiswa pada 'Halaman Daftar Mahasiswa'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	(tidak ada)		
Luaran Tambahan :	(tidak ada)		

D.87 DUC-302 Iterasi 001

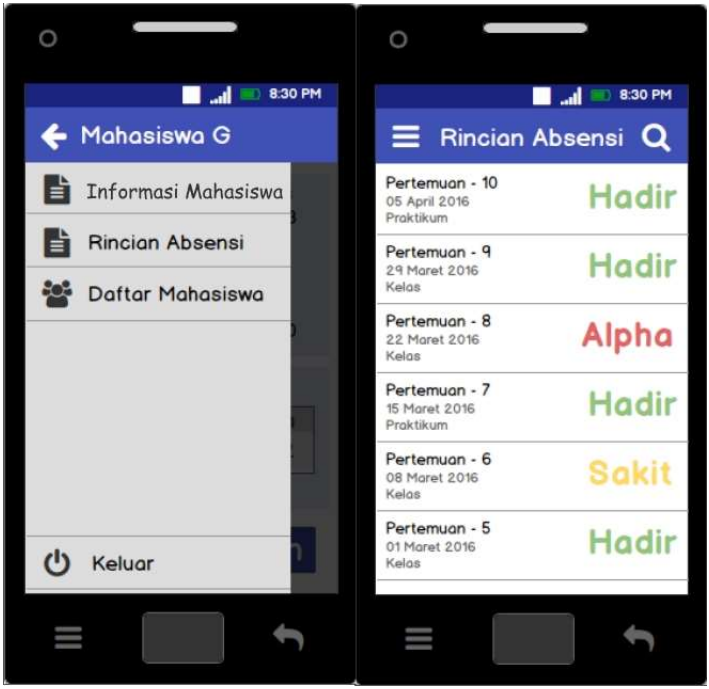
Use Case	DUC-301 TU Mencari Mahasiswa	Iterasi	001
Use Case Description :	**Basic Scenario :** Pada 'Halaman Daftar Mahasiswa', TU mengisi 'Field Cari Mahasiswa' lalu menekan 'Ikon Cari Mahasiswa'. Sistem menyeleksi daftar mahasiswa berdasarkan masukan yang diterima lalu memuat hasil pencarian mahasiswa pada 'Halaman Daftar Mahasiswa'. **Alternate Scenario :** (tidak ada)		
Diagram Robustness:			
Pembeda Dengan Iterasi Sebelumnya :	- Menghapus Entity Hasil Pencarian Mahasiswa.		
Luaran Tambahan :	(tidak ada)		

D.88 DUC-302 Iterasi 002

Use Case	DUC-302 TU Mencari Mahasiswa	Iterasi	002
Use Case Description :	**Basic Scenario :** (tidak ada) **Alternate Scenario :** (tidak ada)		
Diagram Robustness:	(tidak ada)		
Pembeda Dengan Iterasi Sebelumnya :	- Use Case dihapus dan digabungkan secara langsung dengan Use Case yang meng-extend Use Case ini. Pertimbangannya karena penggunaan fitur search sederhana sehingga tidak perlu dipecah menjadi Use Case baru.		
Luaran Tambahan :	(tidak ada)		

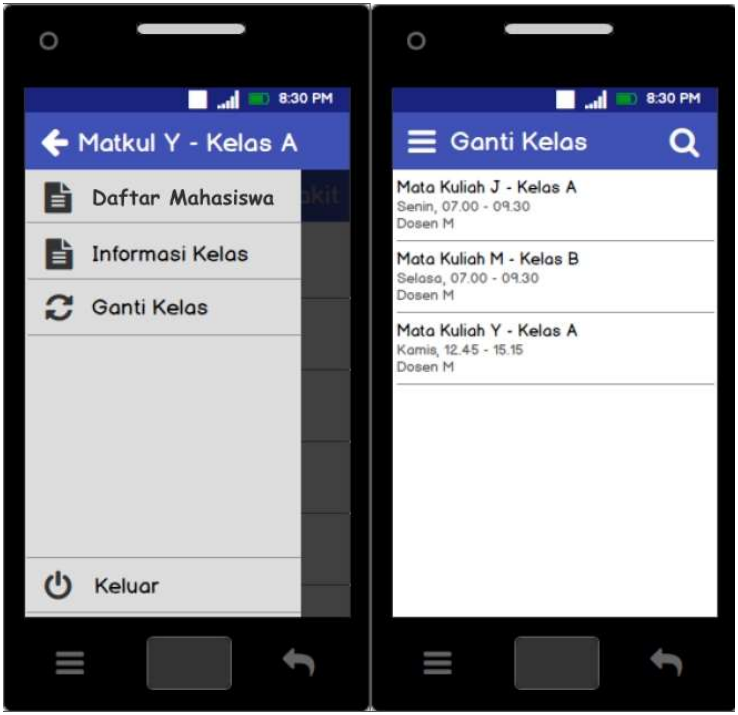
D.89 RAUI-001

Rough Draft UI, perbaikan navigation drawer untuk mahasiswa. Ditambahkan pilihan menu informasi mahasiswa.



D.90 RAUI-002

Rough Draft UI, perbaikan navigation drawer untuk dosen. Ditambahkan pilihan menu daftar mahasiswa.



D.91 RAUI-003

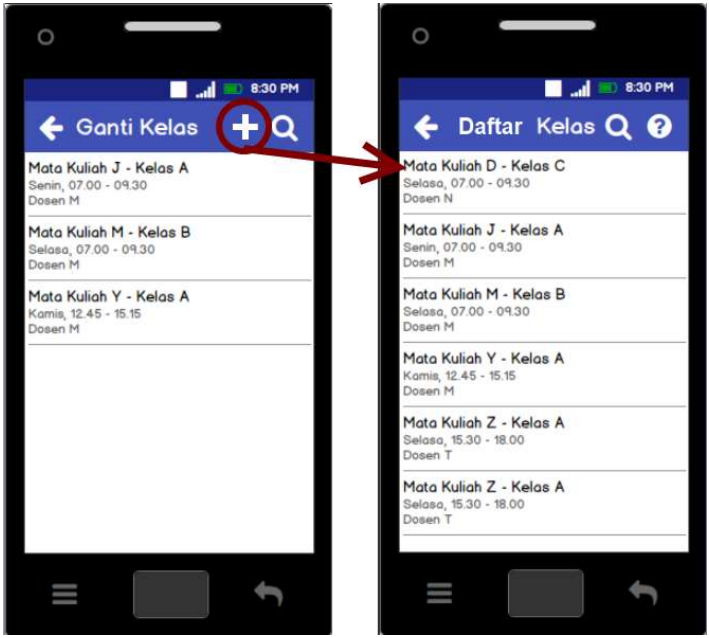
Rough Draft UI, perbaikan tampilan ubah status mahasiswa untuk dosen. Ditambahkan tombol batal sehingga dosen punya pilihan untuk membatalkan mengubah status mahasiswa. Selain itu juga dilakukan sedikit penyesuaian warna.

D.92 RAUI-004

Rough Draft UI, perbaikan tampilan ubah kelas mahasiswa untuk TU. Ditambahkan tombol batal sehingga TU punya pilihan untuk membatalkan mengubah kelas mahasiswa. Selain itu juga dilakukan sedikit penyesuaian warna.

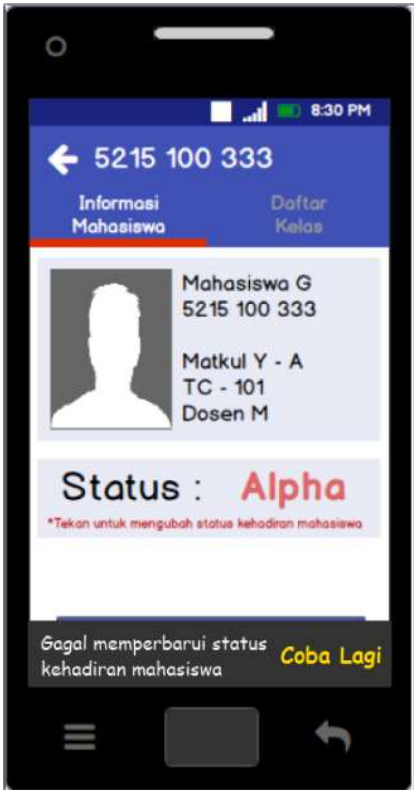
D.93 RAUI-005

Rough Draft UI, ditambahkan icon + pada halaman ganti kelas dosen dan ditambahkan UI baru bernama halaman daftar kelas yang menampilkan daftar semua kelas yang ada di database.



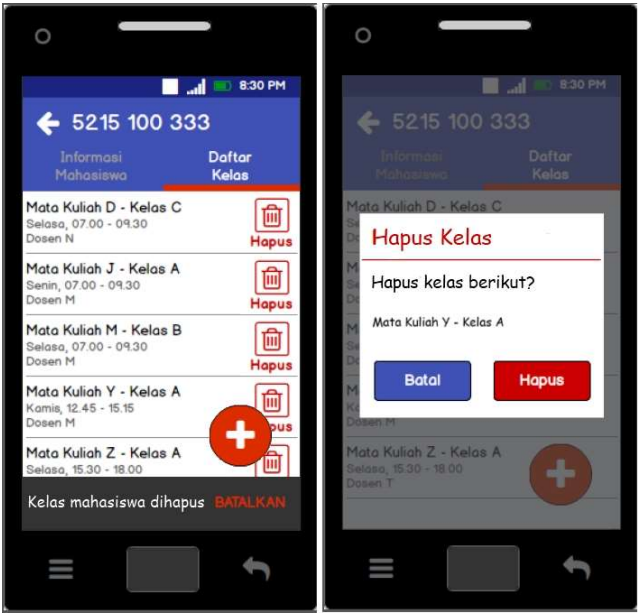
D.94 RAUI-006

Rough Draft UI, ditambahkan snackbar pada halaman mahasiswa ketika system gagal memperbarui status kehadiran mahasiswa



D.95 RAUI-007

Rough Draft UI, ditambahkan ikon / tombol hapus pada setiap kelas yang ada di daftar kelas. Ada dialog yang muncul sebagai konfirmasi ketika pengguna menekan ikon / tombol hapus. Ada Snackbar pada tab daftar kelas ketika sistem menghapus kelas mahasiswa



D.96 RAUI-008

Rough Draft UI, ditambahkan snackbar pada halaman tambah mahasiswa ketika sistem gagal menambahkan data mahasiswa baru. Penambahan kontainer ‘kelas yang diambil’ dan tombol ‘tambah kelas’ seperti yang bisa dilihat pada tampilan pertama.

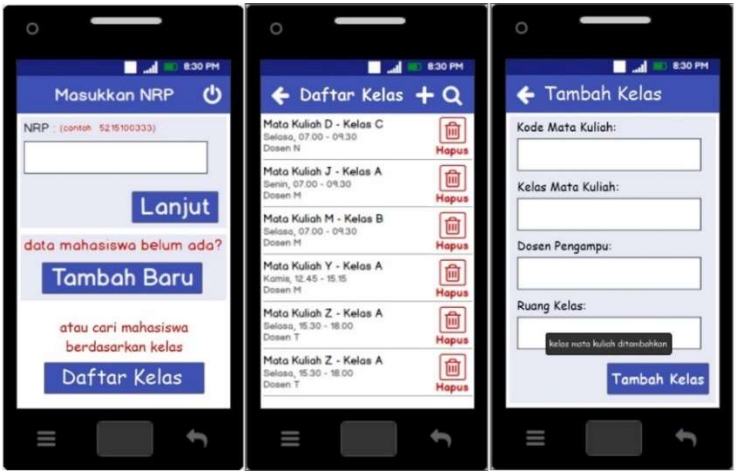
Ketika pengguna menekan tambah kelas, maka pengguna akan dibawa ke tampilan keempat, dan setelah pengguna memilih kelas, maka tampilan halaman tambah mahasiswa akan diperbarui menjadi seperti nomer 2 dan 3, dimana halaman bisa discroll dan panjang halaman menyesuaikan banyak kelas yang diambil oleh mahasiswa.

D.97 RAUI-009

Rough Draft UI, ditambahkan tombol ‘Daftar Kelas’ dan ada penyesuaian beberapa elemen Interface di halaman TU Awal.

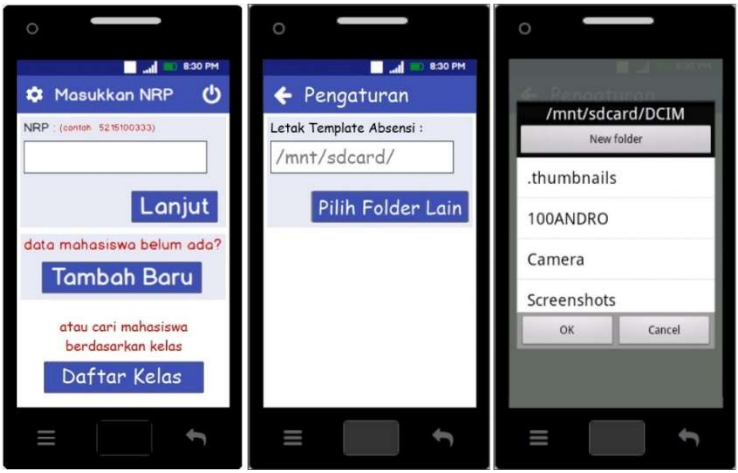
Ketika Pengguna menekan Tombol Daftar Kelas maka akan dibawa ke halaman ‘Daftar Kelas’ dimana fungsionalitasnya hampir serupa dengan halaman Tambah Kelas yang bisa diakses oleh Pengguna TU dengan adanya tombol hapus dan tombol tambah baru serta dihapusnya tombol ‘help’.

Jika pengguna menekan tombol tambah baru (ikon dengan simbol tanda plus / “+”) maka pengguna akan dibawa ke Halaman Tambah Kelas Mata Kuliah dimana setelah pengguna mengisi form yang ada dan menekan tombol ‘Tambah Kelas’ akan muncul Toast jika berhasil.

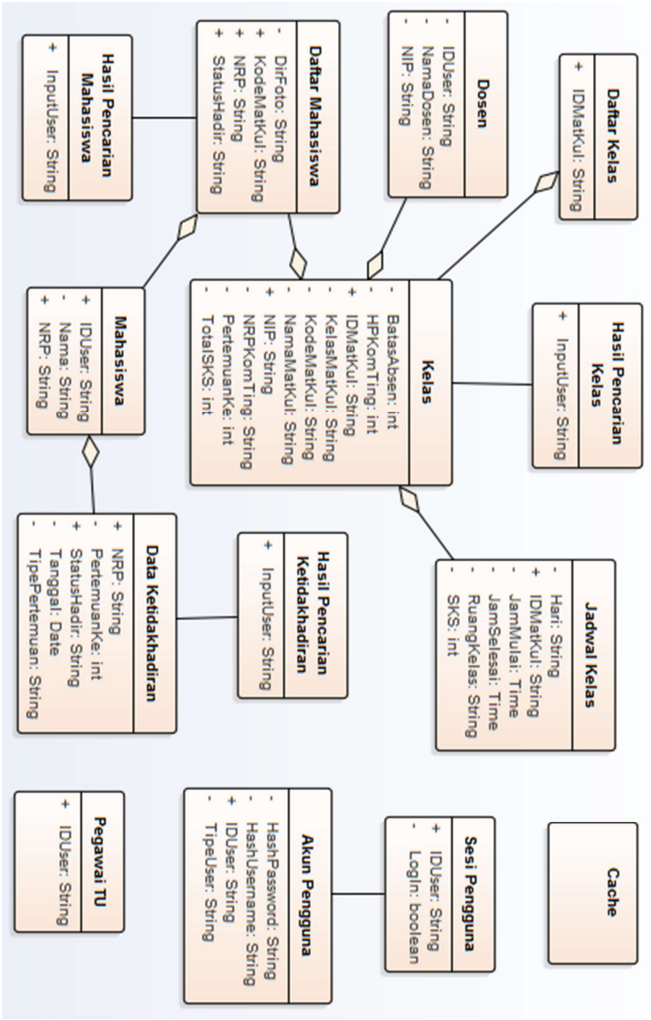


D.98 RAUI-010

Rough Draft UI, ditambahkan ikon gerigi pada halaman TU Awal yang jika ditekan akan membawa pengguna ke halaman pengaturan. Di halaman pengaturan, pengguna bisa memilih lokasi template absensi dengan menggunakan file chooser yang muncul ketika tombol ‘Pilih Folder Lain’ ditekan.

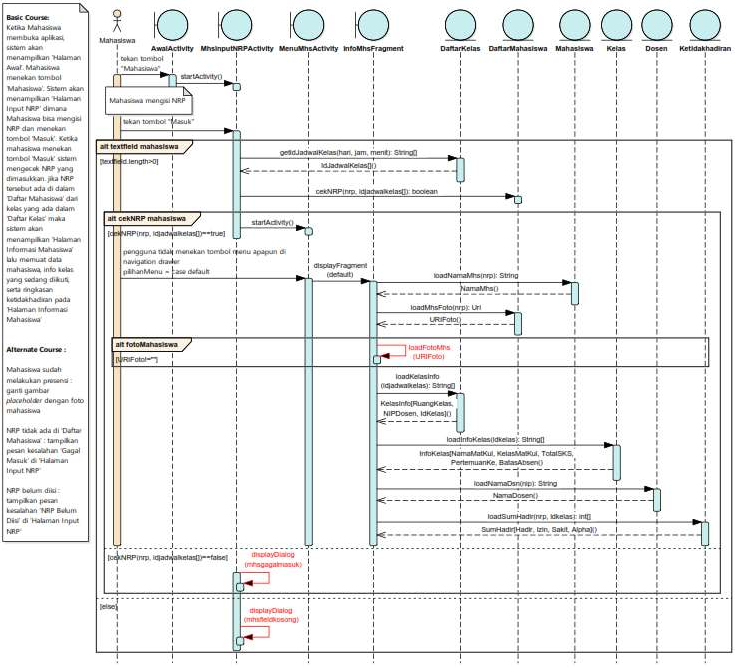


LAMPIRAN E
UPDATED DOMAIN MODEL

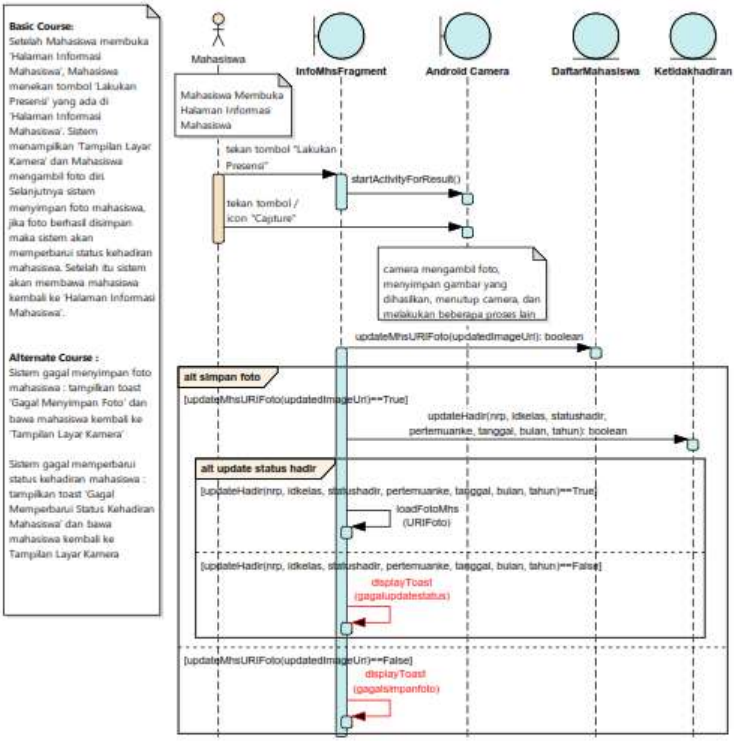


LAMPIRAN F
SEQUENCE DIAGRAMMING

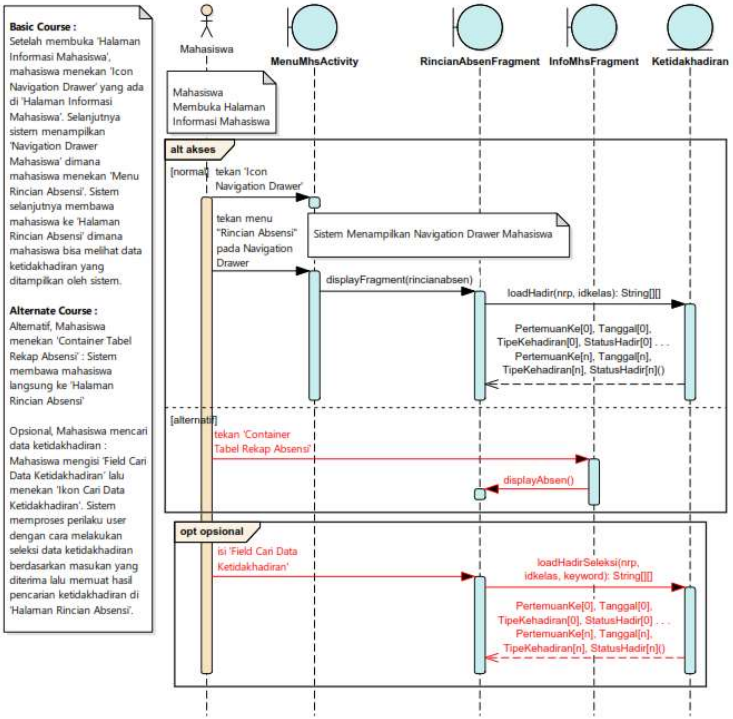
F.1 SD-101 Mahasiswa Membuka Halaman Informasi Mahasiswa



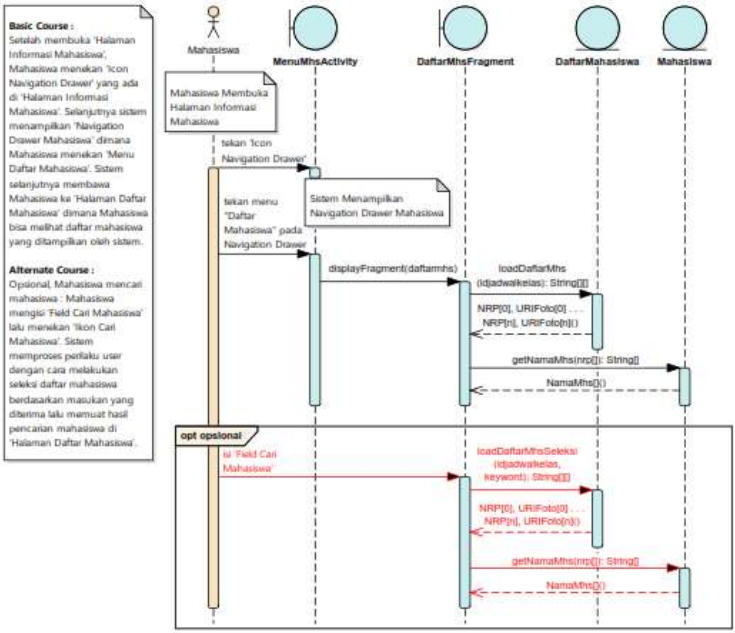
F.2 SD-102 Mahasiswa Melakukan Presensi



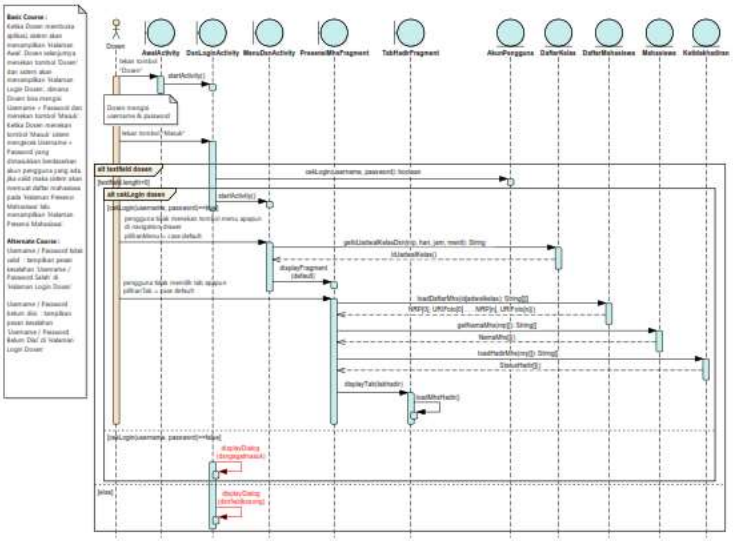
F.3 SD-103 Mahasiswa Melihat Data Ketidakhadiran



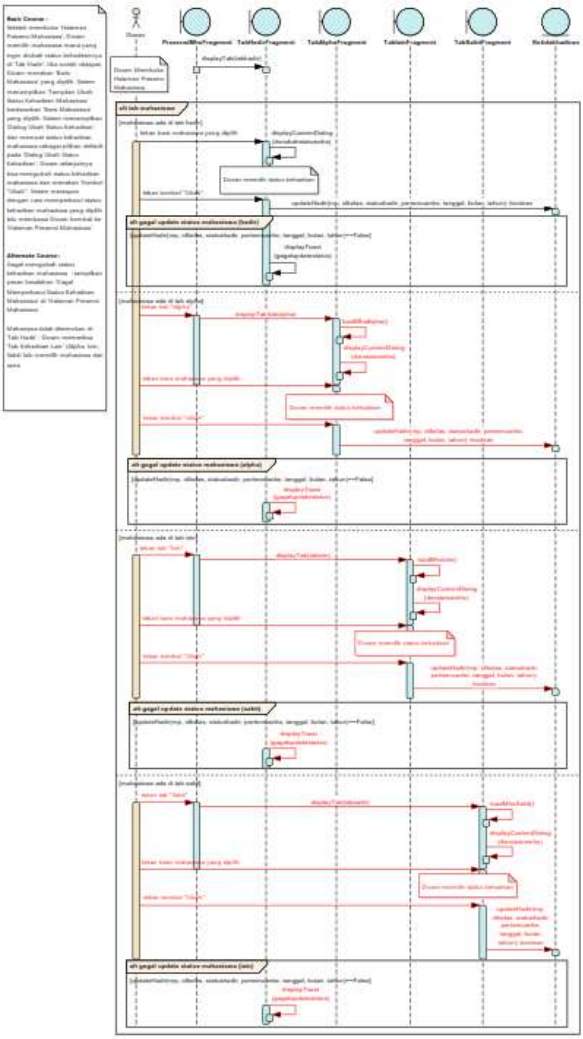
F.4 SD-104 Mahasiswa Melihat Daftar Mahasiswa



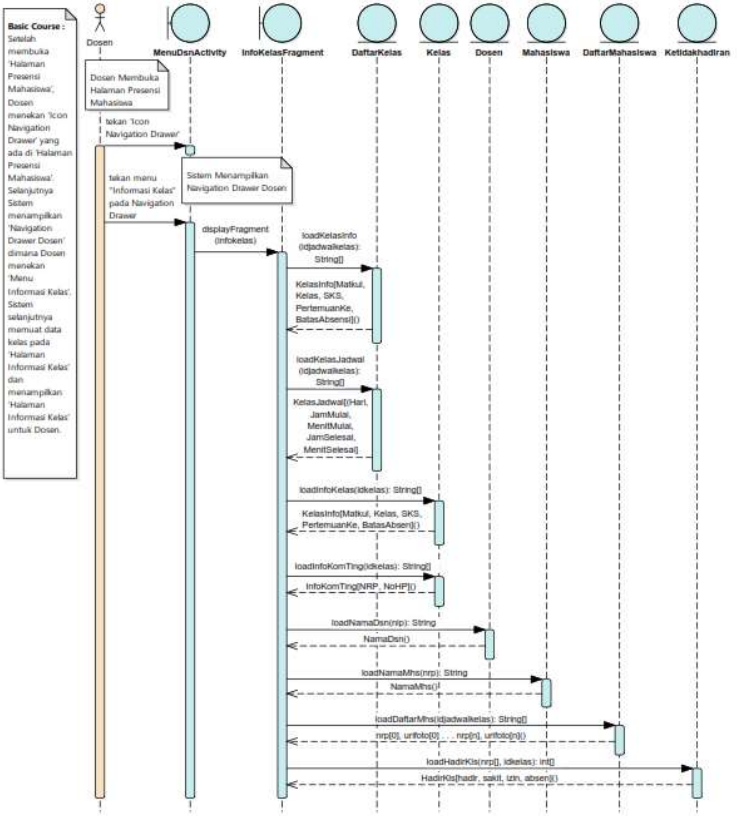
F.5 SD-201 Dosen Membuka Halaman Presensi Mahasiswa



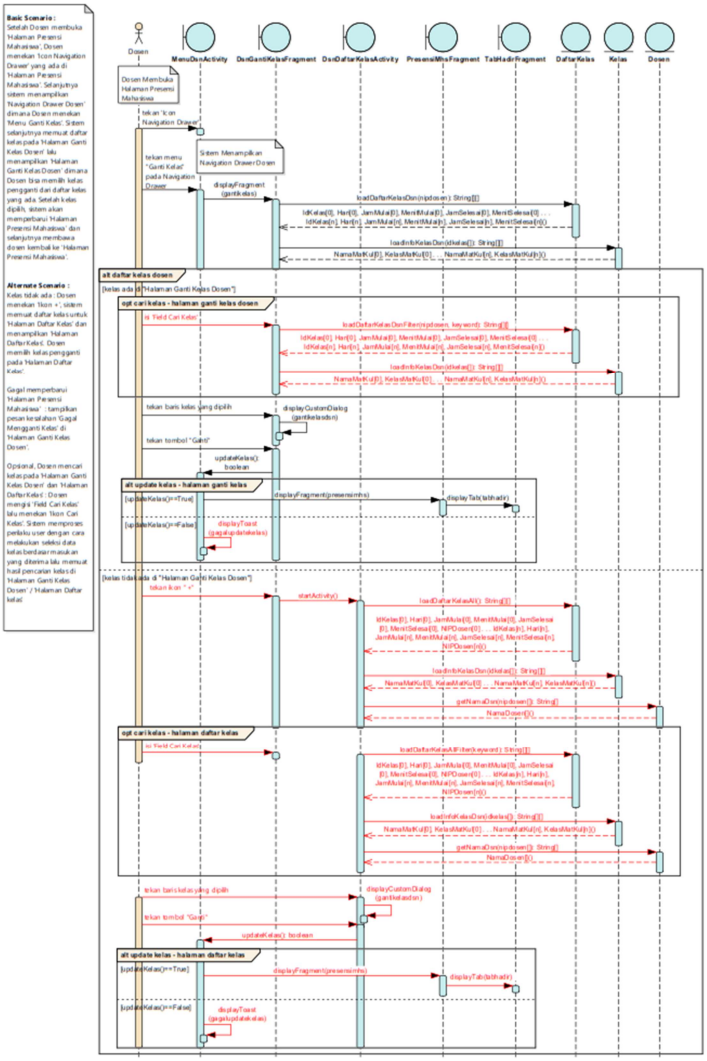
F.6 SD-202 Dosen Mengubah Status Kehadiran Mahasiswa



F.7 SD-203 Dosen Melihat Informasi Kelas



F.8 SD-204 Dosen Mengganti Kelas

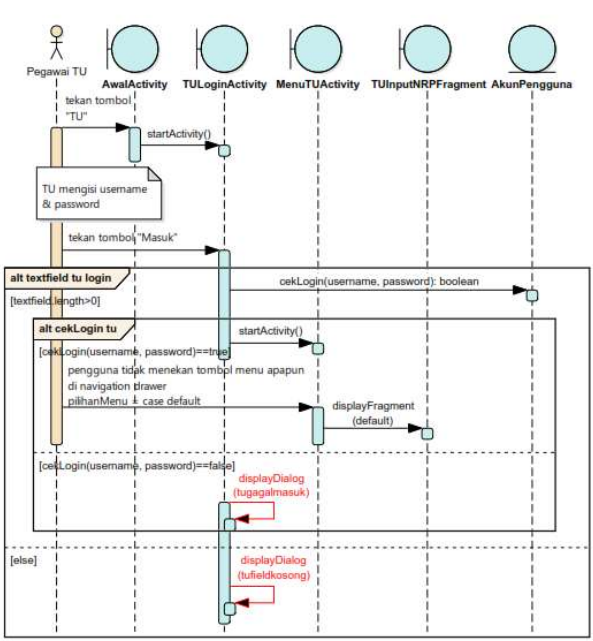


F.9 SD-301 TU Membuka Halaman TU Awal

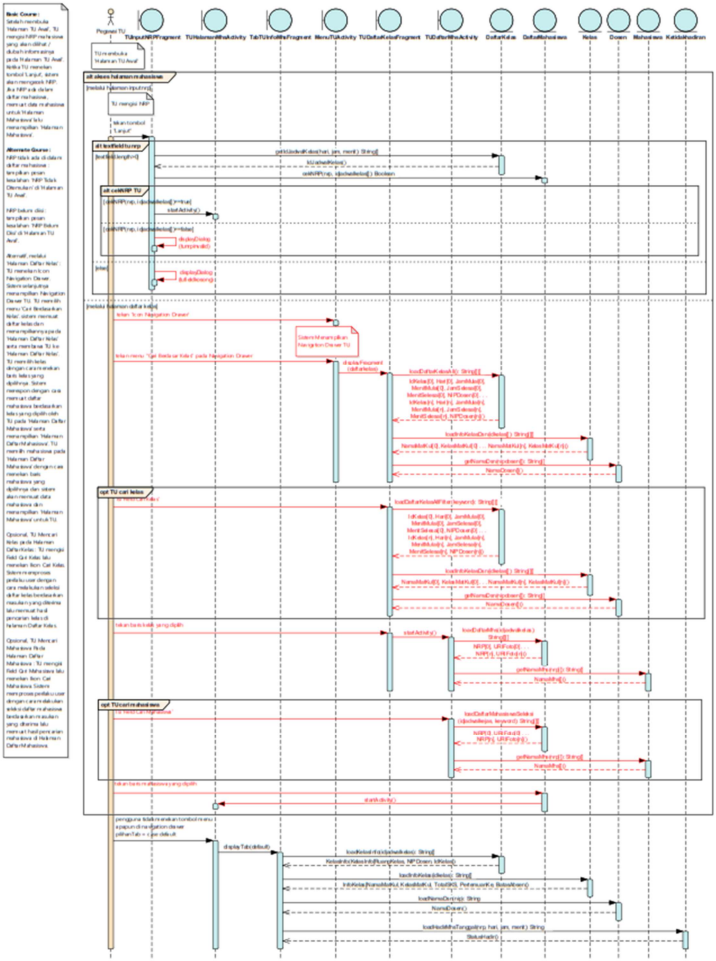
Basic Course :
Ketika TU membuka aplikasi, sistem akan menampilkan 'Halaman Awal'. TU menekan tombol 'TU' untuk mengakses 'Halaman Login TU' lalu mengisi Username + Password dan menekan tombol 'Masuk'. Sistem mengecek Username + Password berdasarkan akun pengguna yang ada, jika valid maka sistem akan menampilkan 'Halaman Masukkan NRP TU'.

Alternate Course :
Username / Password tidak valid : tampilkan pesan kesalahan 'Username / Password Salah' di 'Halaman Login TU'.

Username / Password belum diisi : tampilkan pesan kesalahan 'Username / Password Belum Diisi' di 'Halaman Login TU'.



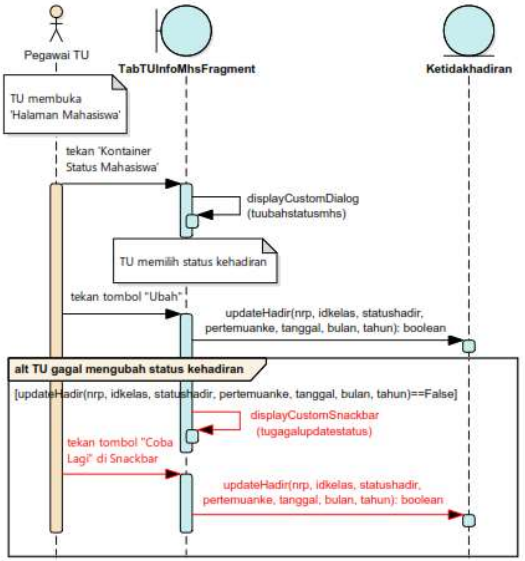
F.10 SD-302 TU Membuka Halaman Mahasiswa



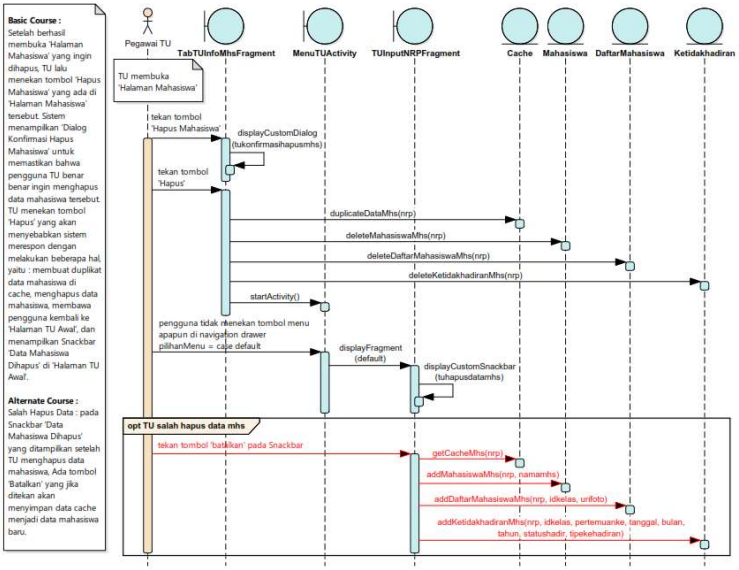
F.11 SD-303 TU Mengubah Status Kehadiran Mahasiswa

Basic Course :
Setelah berhasil membuka 'Halaman Mahasiswa', TU menekan 'Kontainer Status Mahasiswa'. Sistem merespon dengan cara memproses memuat status kehadiran mahasiswa sebagai pilihan default pada 'Dialog Ubah Status Kehadiran Mahasiswa' yang ditampilkan. Selanjutnya TU bisa memilih status kehadiran mahasiswa lalu menekan tombol 'Ubah' yang akan menyebabkan sistem memperbarui status kehadiran mahasiswa lalu memuat ulang 'Halaman Mahasiswa' dengan data yang telah diperbarui.

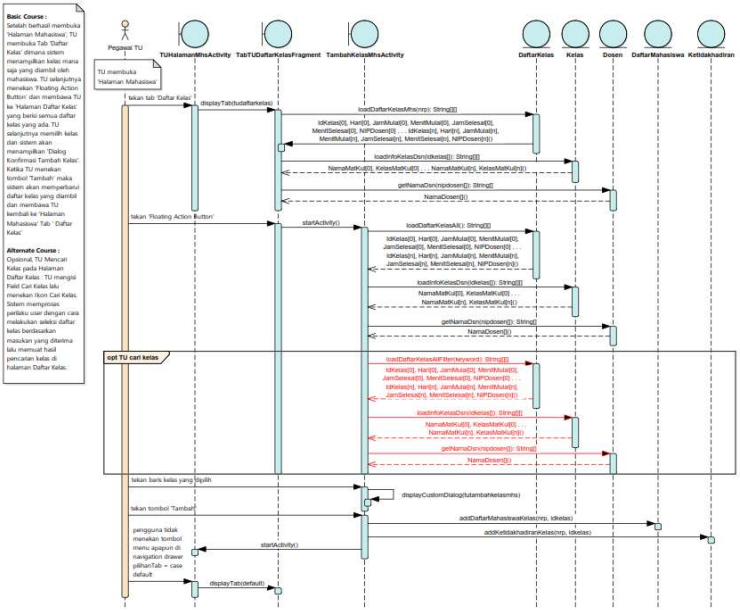
Alternate Course :
Gagal mengubah status kehadiran mahasiswa : sistem menampilkan snackbar 'Gagal Memperbarui Status Kehadiran Mahasiswa' di 'Halaman Mahasiswa' yang juga berisi tombol 'Coba Lagi' yang jika ditekan oleh pengguna akan menyebabkan system mencoba memperbarui status kehadiran mahasiswa sekali lagi.



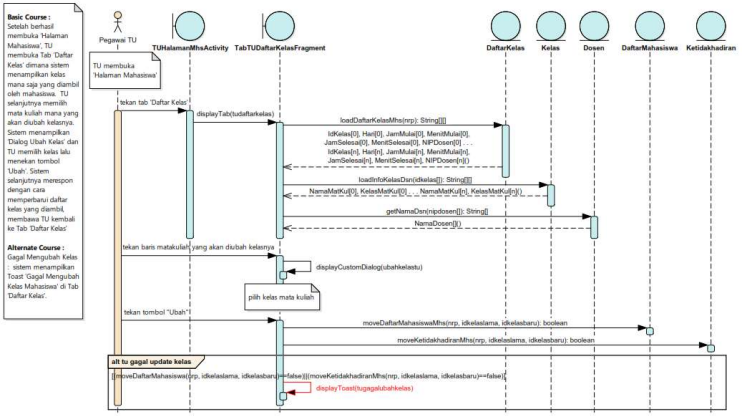
F.12 SD-304 TU Menghapus Data Mahasiswa



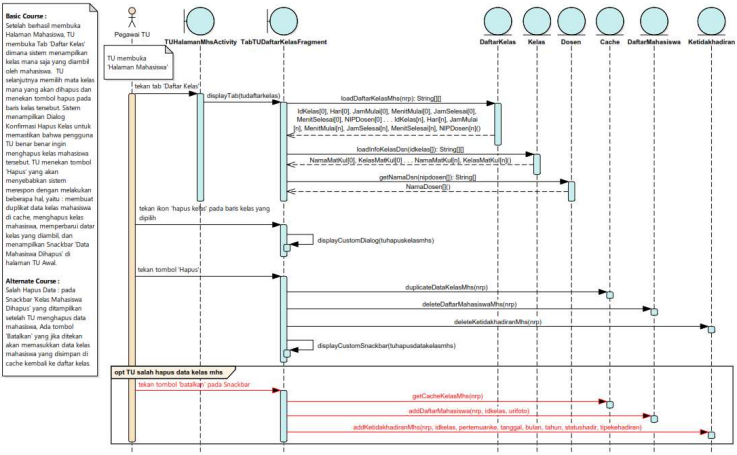
F.13 SD-305 TU Menambah Kelas Mahasiswa



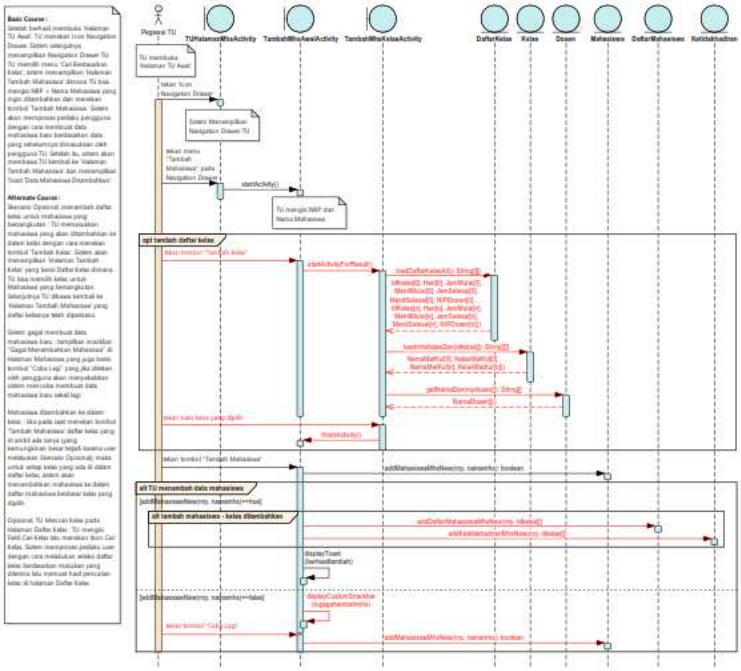
F.14 SD-306 TU Memindah Kelas Mahasiswa



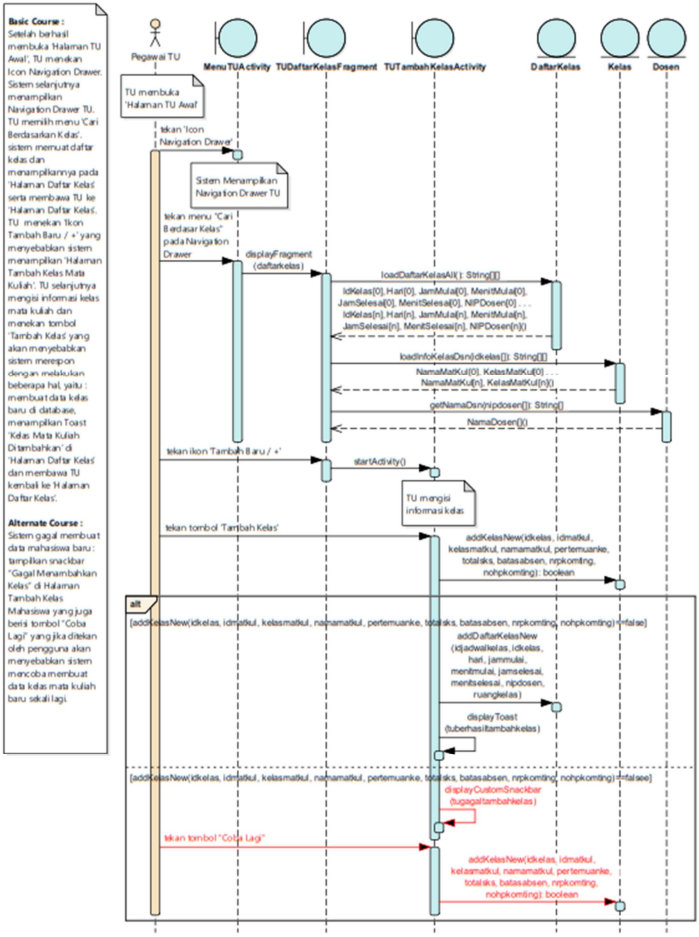
F.15 SD-307 TU Menghapus Kelas Mahasiswa



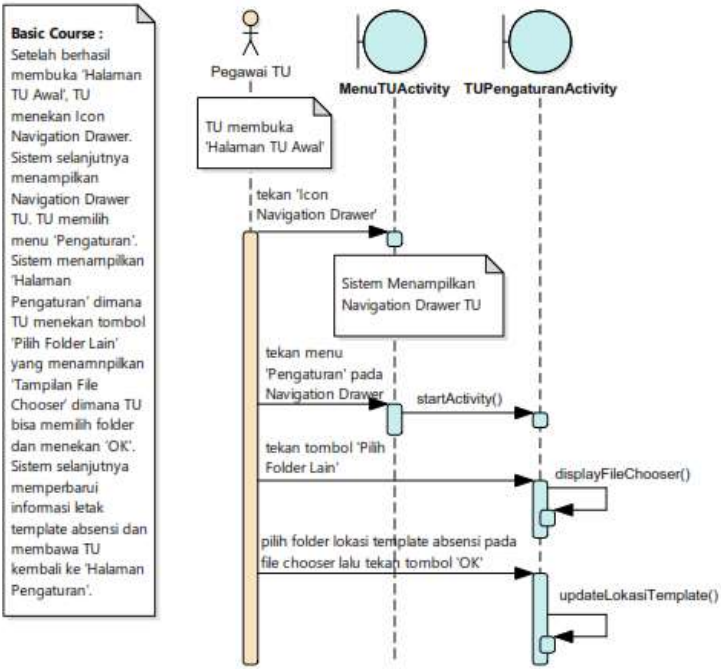
F.16 SD-308 TU Menambah Mahasiswa



F.17 SD-309 TU Menambah Kelas Mata Kuliah Baru



F.18 SD-310 TU Memilih Lokasi Template Presensi



LAMPIRAN G
CLASS DIAGRAM