TUGAS AKHIR – KS 141501

RANCANG BANGUN APLIKASI MOBILE CLIENT-SIDE SERVICE DESK ITS BERBASIS ANDROID

DESIGN AND DEVELOPMENT OF CLIENT-SIDE ITS SERVICE DESK MOBILE APPLICATION BASED ON ANDROID

GINANJAR PRABOWO
NRP 05211440000115

Dosen Pembimbing
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.
Hanim Maria Astuti, S.Kom., M.Sc.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KS 141501

RANCANG BANGUN APLIKASI MOBILE CLIENT-SIDE SERVICE DESK ITS BERBASIS ANDROID

GINANJAR PRABOWO
NRP 05211440000115

Dosen Pembimbing
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.
Hanim Maria Astuti, S.Kom., M.Sc.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – KS 141501

DESIGN AND DEVELOPMENT OF CLIENT-SIDE ITS SERVICE DESK MOBILE APPLICATION BASED ON ANDROID

GINANJAR PRABOWO
NRP 05211440000115

Supervisor
Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.
Hanim Maria Astuti, S.Kom., M.Sc.

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Information Technology and Communication
Sepuluh Nopember Institute of Technology
Surabaya 2018

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI MOBILE CLIENT-SIDE SERVICE DESK ITS BERBASIS ANDROID

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

GINANJAR PRABOWO
NRP 05211440000115

Surabaya, 12 Juli 2018

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M.Kom
NIP. 196503101991021001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI MOBILE CLIENT-SIDE SERVICE DESK ITS BERBASIS ANDROID

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

GINANJAR PRABOWO
NRP 05211440000115

Disetujui Tim Penguji: Tanggal Ujian: 5 Juli 2018
Periode Wisuda: September 2018

Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom. (Pembimbing 1)

Hanim Maria Astuti, S.Kom., M.Sc. (Pembimbing 2)

Dr. Ir. Aris Tjahyanto, M.Kom. (Penguji 1)

Hatma Suryotrisongko, S.Kom., M.Eng. (Penguji 2)

Halaman ini sengaja dikosongkan

RANCANG BANGUN APLIKASI MOBILE CLIENT-SIDE *SERVICE DESK ITS* BERBASIS ANDROID

Nama Mahasiswa	**:**	**Ginanjar Prabowo**
NRP	**:**	**05211440000115**
Jurusan	**:**	**SISTEM INFORMASI**
Dosen Pembimbing I	**:**	**Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.**
Dosen Pembimbing II	**:**	**Hanim Maria Astuti, S.Kom., M.Sc.**

ABSTRAK

DPTSI atau Direktorat Pengembangan Teknologi dan Sistem Informasi merupakan salah satu direktorat milik ITS (Institut Teknologi Sepuluh Nopember). DPTSI bertugas melakukan pengembangan serta pengelolaan layanan teknologi dan sistem informasi (TSI) terpadu. DPTSI memiliki unit fungsional yang disebut sebagai service desk. Untuk mendukung berjalannya service desk dalam pemenuhan serta pelaporan layanan, DPTSI akhirnya merilis aplikasi service desk. Aplikasi service desk adalah sebuah aplikasi yang digunakan untuk menyampaikan keluhan dan request bagi civitas akademika ITS ataupun pihak luar ITS yang ditujukan kepada unit/departemen/fakultas terkait.

Di lain sisi berkembangnya penggunaan perangkat pintar/smartphone akhirnya memunculkan kebutuhan baru untuk berbagai layanan aplikasi di Indonesia, di ITS, khususnya aplikasi service desk yang telah dirilis DPTSI. Aplikasi service desk dianggap masih memiliki keterbatasan dalam sisi fleksibilitas pengoperasian. User/pembuat tiket menginginkan pengalaman penggunaan aplikasi yang lebih baik tanpa harus membuka browser untuk mengelola tiket yang telah dibuat, begitu pula ticket worker. Kedua user role

menginginkan pengalaman terbaik untuk mengelola tiket pada mobile device yang dimiliki.

Pada penelitian ini, penulis membangun sebuah aplikasi service desk versi mobile, native, berbasis android. Dalam pengembangannya penulis juga mengimplementasikan design pattern MVP, dimana 1 kelas activity akan dibagi menjadi 3 kelas meliputi Model, View, dan Presenter. Hal ini menciptakan separasi yang jelas antara komponen / class yang dibangun. Selain itu pada pengerjaan tugas akhir ini juga dilakukan blackbox testing guna mendukung penciptaan aplikasi yang siap naik produksi / production ready. Penulis juga melakukan deploy web service ke cloud agar aplikasi dapat dipergunakan dan dapat dilakukan pengujian kelayakan oleh pihak internal DPTSI. Adapun web service dibangun secara native dengan menggunakan bahasa pemrograman PHP dengan struktur 3 class.

Kata kunci : DPTSI, Service Desk, Android, Design Pattern, MVP, Blackbox Testing, Web Service.

DESIGN AND DEVELOPMENT OF ANDROID BASED MOBILE CLIENT-SIDE APPLICATION FOR *ITS SERVICE DESK*

Name	**:**	**Ginanjar Prabowo**
NRP	**:**	**05211440000115**
Department	**:**	**INFORMATION SYSTEMS**
First Supervisor	**:**	**Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom.**
Second Supervisor	**:**	**Hanim Maria Astuti, S.Kom., M.Sc.**

ABSTRACT

DPTSI (Technology and Information System Development Directorate) is one of ITS (Sepuluh Nopember Institute of Technology) directorate. DPTSI has a duty for developing and managing integrated technology and information system (TSI) services. DPTSI has a functional unit called service desk. To support the running of the service desk in the fullfillment and reporting of services, DPTSI finally released a service desk application. Service desk application is an app used to submit complaint and request dedicated to academic community of ITS and external community to related unit / department / and faculty.

On the other hand, the growing of use of smart devices / smartphones has finally led to new needs for various application services in Indonesia, in ITS, especially service desk application that have been released by DPTSI. Service desk app is considered to still have limitations in terms of flexibility of operation. User want a better experience while using the app without having to open a browser to manage tickets that have been created, as well as ticket worker. Both user roles want the best experience to manage tickets through their own mobile devices.

In this research, the authors build an android based mobile vesion of service desk app, that built natively. while developing the code, the authors also implements the MVP design pattern, where 1 activity class divided into 3 classes including Model, View, and Presenter. Those thing creates a very clear task separation between the components / classes. The authors also do blackbox testing with some scenario to support the creation of app that ready for production / deployment. The author also deploy web service to cloud so that application can be used and can be test by internal staff of DPTSI. as for web service built natively using PHP programming language with 3 class structure.

Keywords : DPTSI, Service Desk, Android, Design Pattern, MVP, Blackbox Testing, Web Service.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan kekuatan serta hidayah-Nya kepada penulis sehingga penulis dapat menyelesaikan tugas akhir ini yang merupakan salah satu syarat kelulusan di Jurusan Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik berupa materiil maupun moril demi tercapainya tujuan pembuatan tugas akhir ini. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Segenap keluarga besar terutama kedua orang tua dan kakak penulis, Bapak Ir. Djamiat, M.M.T., Ibu Yuliati, dan Reni Fernisa yang senantiasa mendoakan, memberikan motivasi dan semangat, sehingga penulis mampu menyelesaikan Pendidikan S1 ini dengan baik.
2. Ibu Mahendrawati ER, ST, M.Sc, Ph.D sebagai dosen wali penulis selama menempuh pendidikan di Departemen Sistem Informasi.
3. Bapak Dr. Ir. Aris Tjahyanto, M.Kom., selaku Kepala Departemen Sistem Informasi ITS, Bapak Nisfu Asrul Sani, S.Kom., M.Sc selaku Kaprodi S1 Sistem Informasi ITS serta seluruh dosen pengajar beserta staf dan karyawan Departemen Sistem Informasi, FTIK ITS Surabaya selama penulis menjalani kuliah.
4. Bapak Dr. Eng. Febriliyan Samopa, S.Kom., M.Kom. dan Ibu Hanim Maria Astuti, S.Kom., M.Sc. selaku dosen pembimbing yang telah banyak meluangkan waktu untuk membimbing, mengarahkan, dan mendukung dengan memberikan ilmu, petunjuk, dan motivasi dalam penyelesaian Tugas Akhir ini.

5. Bapak Dr. Ir. Aris Tjahyanto, M.Kom. dan Bapak Hatma Suryotrisongko, S.Kom., M.Eng. selaku dosen penguji yang telah memberikan kritik, saran, dan masukan penyempurnaan Tugas Akhir ini.
6. Teman-teman Sistem Informasi Angkatan 2014 (OSIRIS) yang senantiasa menemani dan memberikan motivasi bagi penulis selama perkuliahan hingga dapat menyelesaikan Tugas Akhir ini.
7. Kawan – kawan lab ADDI, RDIB, SE, dan MSI yang telah mempersilakan penulis bernaung dan mencari inspirasi dalam mengerjakan Tugas Akhir ini.
8. Arif Rahman Hakim selaku rekan kerja di PT Pelindo III (Persero) yang telah membantu sedikit banyak pekerjaan kantor dan memaklumi selama penulis mengerjakan Tugas Akhir.
9. Serta seluruh pihak-pihak lain yang tidak dapat disebutkan satu per satu, yang telah banyak membantu penulis selama perkuliahan hingga dapat menyelesaikan Tugas Akhir ini.

Penyusunan laporan tugas akhir ini masih jauh dari kata sempurna sehingga penulis menerima adanya kritik maupun saran yang membangun untuk perbaikan di masa yang akan dating. Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 27 Juni 2018

Penulis,

Ginanjar Prabowo

DAFTAR ISI

ABSTRAK viii
ABSTRACT x
KATA PENGANTAR xii
DAFTAR ISI xiv
DAFTAR GAMBAR xviii
DAFTAR TABEL xxvi
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Batasan Masalah 4
1.4 Tujuan Penelitian 4
1.5 Manfaat Penelitian 4
1.6 Relevansi 5
BAB II TINJAUAN PUSTAKA 7
2.1 Penelitian Sebelumnya 7
2.2 Dasar Teori 9
2.2.1 Service Desk 9
2.2.2 Service Desk DPTSI 12
2.2.3 Unified Modeling Language (UML) 14
2.2.4 Model Rekayasa Perangkat Lunak Waterfall 15
2.2.5 Android 16
2.2.6 Design Pattern 20
2.2.7 Design Pattern MVP 21
2.2.8 Blackbox Testing 23
2.2.9 Web Service 23
BAB III METODE PENELITIAN 27
3.1 Tahap Analisis Kebutuhan 28
3.1.1 Referensi, Penelitian Sebelumnya, Kode Program Aplikasi Existing 28
3.1.2 Wawancara 28
3.1.3 Observasi 28
3.1.4 Tujuan dan Spesifikasi Tugas Akhir 28
3.2 Tahap Desain Sistem 29

3.2.1 Pembuatan Use Case Aplikasi 29
3.2.2 Pembuatan Desain Aplikasi 29
3.2.3 Desain Kebutuhan dan Pemetaan Service 29
3.2.4 Desain Web Service, Use Case Aplikasi, dan Prototipe Aplikasi 29
3.3 Tahap Implementasi 30
3.3.1 Implementasi Desain Aplikasi dan Web Service 30
3.3.2 Web Service dan Aplikasi Mobile Service Desk 30
3.4 Tahap Testing 30
3.5 Tahap Deployment 31
3.5.1 Perbaikan Bug Aplikasi 31
3.5.2 Generate APK 31
3.5.3 Deployment Web Service ke Cloud 31
3.6 Tahap Dokumentasi 31
BAB IV PERANCANGAN 33
4.1 Analisis Kebutuhan 33
4.1.1 Wawancara 33
4.1.2 Observasi 34
4.2 Desain Sistem 43
4.2.1 Pembuatan Use Case Aplikasi 43
4.2.2 Pembuatan Desain Kebutuhan dan Pemetaan Service 70
4.2.3 Pembuatan Desain Aplikasi 84
4.2.4 Perancangan Test Case 126
BAB V IMPLEMENTASI 127
5.1 Lingkungan Implementasi 127
5.2 Implementasi Desain Aplikasi dan Web Service 128
5.2.1 Implementasi Login 128
5.2.2 Implementasi Logout 132
5.2.3 Implementasi Lihat Notifikasi 134
5.2.4 Implementasi Search Knowledge 137
5.2.5 Implementasi Lihat Knowledge 140
5.2.6 Implementasi Lihat Info Buat Tiket 142
5.2.7 Implementasi Lihat Profile 144

5.2.8 Implementasi Edit Profile.....147
5.2.9 Implementasi Edit Profile.....149
5.2.10 Implementasi Lupa Password.....152
5.2.11 Implementasi Register Member 154
5.2.12 Implementasi Buat Tiket 155
5.2.13 Implementasi Lihat List Tiket 158
5.2.14 Implementasi Search Tiket.....161
5.2.15 Implementasi Lihat Detail Tiket.....164
5.2.16 Implementasi Balas Tiket.....166
5.2.17 Implementasi Check Tiket (Guest).....170
5.2.18 Implementasi Hapus Tiket.....174
5.2.19 Implementasi Hapus Tiket Via Detail 177
5.2.20 Implementasi Lihat History Tiket 179
5.2.21 Implementasi Lihat Data Client 183
5.2.22 Implementasi Assign Tiket / MySelf.....186
5.2.23 Implementasi Ubah Status Tiket 189
5.3 Implementasi Test Case 191
5.4 Implementasi Deployment Cloud 194
BAB VI HASIL DAN PEMBAHASAN 195
6.1 Operasional Aplikasi Mobile *Service Desk*.....195
6.1.1 Manajemen Role User 196
6.1.2 Pengelolaan Tiket.....203
6.1.3 Informasi Penggunaan Aplikasi (Knowledge Base).....210
6.1.4 Manajemen Informasi Pengguna.....212
6.2 Arsitektur Aplikasi Mobile *Service Desk*.....214
6.3 Arsitektur Web Service Service Desk.....218
BAB VII KESIMPULAN DAN SARAN.....221
7.1 Kesimpulan 221
7.2 Saran.....222
DAFTAR PUSTAKA 223
BIODATA PENULIS 229
LAMPIRAN A : Perancangan Test Case.....231
A.1 Test Case User 231
A.2 Test Case Ticket Worker 250

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1 Service Desk Flow 12
Gambar 2.2 Arsitektur Aplikasi Service Desk 13
Gambar 2.3 Contoh Query Aplikasi Existing Service Desk .. 14
Gambar 2.4 Siklus Rekayasa Perangkat Lunak Waterfall 15
Gambar 2.5 Arsitektur Sistem Operasi Android 18
Gambar 2.6 Version History Sistem Operasi Android 20
Gambar 2.7 Arsitektur Design Pattern Model-View-Presenter 21
Gambar 2.8 Arsitektur Design Pattern Model-View-Presenter 22
Gambar 2.9 Ilustrasi Web Service 24
Gambar 3.1 Metodologi Pengerjaan Tugas Akhir 27
Gambar 4.1 Diagram Arsitektur Aplikasi *Service Desk* Existing 35
Gambar 4.2 Use Case User 44
Gambar 4.3 Use Case Ticket Worker 56
Gambar 4.4 Format Request Web Service (NoAuth) 71
Gambar 4.5 Format Request Web Service (with Auth) 71
Gambar 4.6 Diagram Arsitektur Aplikasi Service Desk 84
Gambar 4.7 User Interface Login 85
Gambar 4.8 Sequence Diagram Use Case Login 86
Gambar 4.9 User Interface Logout 87
Gambar 4.10 Sequence Diagram Logout 88
Gambar 4.11 User Interface Lihat Notifikasi 89
Gambar 4.12 Sequence Diagram Lihat Notifikasi 90
Gambar 4.13 User Interface Search Knowledge 91
Gambar 4.14 Sequence Diagram Search Knowledge 92
Gambar 4.15 User Interface Lihat Knowledge 93
Gambar 4.16 Sequence Diagram Lihat Knowledge 94
Gambar 4.17 Lihat Info Buat Tiket 95
Gambar 4.18 Sequence Diagram Lihat Info Buat Tiket 95
Gambar 4.19 Sequence Diagram Lihat Profile 96
Gambar 4.20 Sequence Diagram Lihat Profile 97
Gambar 4.21 User Interface Edit Profile 98
Gambar 4.22 Sequence Diagram Edit Profile 99

Gambar 4.23 User Interface Ubah Password 100
Gambar 4.24 Sequence Diagram Ubah Password..... 101
Gambar 4.25 User Interface Lupa Password..... 102
Gambar 4.26 Sequence Diagram Lupa Password 102
Gambar 4.27 User Interface Register Member 103
Gambar 4.28 Sequence Diagram Register Member..... 103
Gambar 4.29 User Interface Buat Tiket 104
Gambar 4.30 Sequence Diagram Buat Tiket..... 105
Gambar 4.31 User Interface Lihat List Tiket 107
Gambar 4.32 Sequence Diagram Lihat List Tiket..... 107
Gambar 4.33 User Interface Search Tiket..... 108
Gambar 4.34 Sequence Diagram Search Tiket 109
Gambar 4.35 User Interface Lihat Detail Tiket..... 110
Gambar 4.36 Sequence Diagram Lihat Detail Tiket 111
Gambar 4.37 User Interface Balas Tiket..... 112
Gambar 4.38 Sequence Diagram Balas Tiket 113
Gambar 4.39 User Interface Check Ticket (Guest)..... 114
Gambar 4.40 Sequence Diagram Check Tiket 115
Gambar 4.41 User Interface Hapus Tiket..... 116
Gambar 4.42 Sequence Diagram Hapus Tiket 117
Gambar 4.43 User Interface Hapus Tiket (Via Detail) 118
Gambar 4.44 Sequence Diagram Hapus Tiket (Via Detail). 119
Gambar 4.45 User Interface Lihat History Tiket 120
Gambar 4.46 Sequence Diagram Lihat History Tiket..... 120
Gambar 4.47 User Interface Lihat Data Client 121
Gambar 4.48 Sequence Diagram Lihat Data Client..... 122
Gambar 4.49 User Interface Assign Tiket / Myself 123
Gambar 4.50 Sequence Diagram Assign Tiket / Myself..... 124
Gambar 4.51 User Interface Ubah Status Tiket 125
Gambar 4.52 Sequence Diagram Ubah Status Tiket..... 126
Gambar 5.1 User Interface Login Aplikasi 129
Gambar 5.2 Potongan Kode Program *View* Fungsi Login ... 129
Gambar 5.3 Potongan Kode Program *Presenter* Fungsi Login 130
Gambar 5.4 Potongan Kode Program Service *login* method *getUserByEmail* 131
Gambar 5.5 User Interface Menu Logout Aplikasi..... 132

Gambar 5.6 Potongan Kode Program *View* Fungsi Logout. 133
Gambar 5.7 User Interface Lihat Notifikasi Aplikasi 134
Gambar 5.8 Potongan Kode Program *View* Fungsi Lihat Notifikasi..... 135
Gambar 5.9 Potongan Kode Program *Presenter* Fungsi Login 135
Gambar 5.10 Potongan Kode Program Service *getNotifications* 136
Gambar 5.11 User Interface Search Knowledge..... 137
Gambar 5.12 Potongan Kode Program *View* Fungsi Search Knowledge 138
Gambar 5.13 Potongan Kode Program *Presenter* Fungsi Login 138
Gambar 5.14 Potongan Kode Program Service *getKnowledgeByCatOrArt* 139
Gambar 5.15 User Interface Lihat Knowledge 140
Gambar 5.16 Potongan Kode Program *View* Fungsi Lihat Knowledge 141
Gambar 5.17 Potongan Kode Program *Presenter* Fungsi Lihat Knowledge 141
Gambar 5.18 Potongan Kode Program Service *getKnowledgeByCatOrArt* 142
Gambar 5.19 User Interface Lihat Info Buat Tiket 143
Gambar 5.20 Potongan Kode Program *View* Lihat Info Buat Tiket 143
Gambar 5.21 User Interface Lihat Profile..... 144
Gambar 5.22 Potongan Kode Program *View* Fungsi Lihat Profile..... 145
Gambar 5.23 Potongan Kode Program onResponse *Presenter* Fungsi Lihat Profile 145
Gambar 5.24 Potongan Kode Program Service *getUsrByIdDB* 146
Gambar 5.25 User Interface Edit Profile 147
Gambar 5.26 Potongan Kode Program *View* Fungsi Edit Profile 148
Gambar 5.27 Potongan Kode Program *Presenter* Fungsi Edit Profile..... 149

Gambar 5.28 Potongan Kode Program Service *updateInfoUser* 149
Gambar 5.29 User Interface Edit Profile 150
Gambar 5.30 Potongan Kode Program *View* Fungsi Edit Profile 151
Gambar 5.31 Potongan Kode Program *Presenter* Fungsi Edit Profile 151
Gambar 5.32 Potongan Kode Program Service *updateInfoUser* 152
Gambar 5.33 User Interface Lupa Password 153
Gambar 5.34 Potongan Kode Program *View* Lupa Password 153
Gambar 5.35 User Interface Lupa Password 154
Gambar 5.36 Potongan Kode Program *View* Lupa Password 154
Gambar 5.37 User Interface Buat Tiket 155
Gambar 5.38 Potongan Kode Program *View* Fungsi Buat Tiket 156
Gambar 5.39 Potongan Kode Program *Presenter* Fungsi Buat Tiket 156
Gambar 5.40 Potongan Kode Program Service *getCategoryID* 157
Gambar 5.41 User Interface Lihat List Tiket 158
Gambar 5.42 Potongan Kode Program *View* Fungsi Lihat List Tiket 158
Gambar 5.43 Potongan Kode Program *Presenter* Fungsi Lihat List Tiket 159
Gambar 5.44 Potongan Kode Program Service *getTicketByYourAll* 160
Gambar 5.45 User Interface Search Tiket 161
Gambar 5.46 Potongan Kode Program *View* Fungsi Search Tiket 161
Gambar 5.47 Potongan Kode Program *Presenter* Fungsi Search Tiket 162
Gambar 5.48 Potongan Kode Program Service *getTicketByKeywordAllMember* 163
Gambar 5.49 User Interface Lihat Detail Tiket 164

Gambar 5.50 Potongan Kode Program *View* Fungsi Lihat Detail Tiket 165
Gambar 5.51 Potongan Kode Program *Presenter* Fungsi Lihat Detail Tiket 165
Gambar 5.52 Potongan Kode Program Service *getTicketDetails* 166
Gambar 5.53 User Interface Balas Tiket..... 167
Gambar 5.54 Potongan Kode Program *View* Fungsi Balas Tiket 167
Gambar 5.55 Potongan Kode Program *Presenter* Fungsi Balas Tiket 168
Gambar 5.56 Potongan Kode Program Service *addTicketReply* 169
Gambar 5.57 User Interface Check Ticket (Guest)..... 170
Gambar 5.58 Potongan Kode Program *View* Fungsi Check Tiket (Guest)..... 171
Gambar 5.59 Potongan Kode Program *Presenter* Fungsi Check Tiket 172
Gambar 5.60 Potongan Kode Program Service *getTicketGuestDetails* 173
Gambar 5.61 User Interface Hapus Ticket..... 174
Gambar 5.62 Potongan Kode Program *View* Fungsi Hapus Tiket 175
Gambar 5.63 Potongan Kode Program *Presenter* Fungsi Hapus Tiket 175
Gambar 5.64 Potongan Kode Program Service *deleteTicketSD* 176
Gambar 5.65 User Interface Hapus Tiket 177
Gambar 5.66 Potongan Kode Program *View* Fungsi Hapus Tiket Via Detail 178
Gambar 5.67 Potongan Kode Program *Presenter* Fungsi Hapus Tiket Via Detail..... 178
Gambar 5.68 Potongan Kode Program Service *deleteTicketSD* 179
Gambar 5.69 User Interface Lihat History Tiket 180
Gambar 5.70 Potongan Kode Program *View* Fungsi Lihat History Tiket 181

Gambar 5.71 Potongan Kode Program *Presenter* Fungsi Lihat History Tiket 181
Gambar 5.72 Potongan Kode Program Service *getTicketHistory* 182
Gambar 5.73 User Interface Lihat Data Client 183
Gambar 5.74 Potongan Kode Program *View* Fungsi Lihat Data Client 184
Gambar 5.75 Potongan Kode Program *Presenter* Fungsi Lihat Data Client..... 184
Gambar 5.76 Potongan Kode Program Service *getTicketDetails* 185
Gambar 5.77 User Interface Assign Tiket..... 186
Gambar 5.78 Potongan Kode Program *View* Fungsi Assign User / Myself 187
Gambar 5.79 Potongan Kode Program *Presenter* Fungsi Assign Tiket / Myself 187
Gambar 5.80 Potongan Kode Program Service *addHistoryAssignTicket* 188
Gambar 5.81 User Interface Ubah Status Tiket 189
Gambar 5.82 Potongan Kode Program *View* Fungsi Change Status Tiket..... 190
Gambar 5.83 Potongan Kode Program *Presenter* Fungsi Ubah Status Tiket..... 190
Gambar 5.84 Potongan Kode Program Service *updateStatusTicket* 191
Gambar 6.1 Tampilan Home Aplikasi 196
Gambar 6.2 Tampilan Login (Terlalu Banyak Attempt)..... 197
Gambar 6.3 Tampilan Login (Akun Belum Aktif) 198
Gambar 6.4 Perbedaan Tampilan Sidebar Login dan Tidak Login 199
Gambar 6.5 Perbedaan Tampilan Home Antar Role..... 200
Gambar 6.6 Perbedaan Tampilan List Tiket Antar Role..... 201
Gambar 6.7 Perbedaan Tampilan Detail Tiket Antar Role .. 202
Gambar 6.8 Perbedaan Tampilan Buat Tiket Antar Role 203
Gambar 6.9 Tampilan Buat Tiket..... 204
Gambar 6.10 Tampilan Konfirmasi Buat Tiket dan Generate Password..... 205

Gambar 6.11 Tampilan Check Tiket dan Balas Tiket (Guest) 206
Gambar 6.12 Tampilan Lihat Notifikasi 207
Gambar 6.13 Tampilan Hapus, Ubah Status, dan Assign Tiket 208
Gambar 6.14 Tampilan Lihat Data Client dan History Tiket 209
Gambar 6.15 Tampilan List Tiket..... 210
Gambar 6.16 Tampilan Utama Menu Knowledge Base 211
Gambar 6.17 Tampilan Search Knowledge 212
Gambar 6.18 Tampilan Menu Profile (Change Password) .. 213
Gambar 6.19 Tampilan Menu Profile (Edit Profile) 214
Gambar 6.20 Struktur Desain Pattern MVP..... 214
Gambar 6.21 Struktur Kode Program Fungsi Login..... 215
Gambar 6.22 Kode Program *Class* LoginContract 216
Gambar 6.23 Inheritance *Class* LoginFragment dan LoginPresenter 217
Gambar 6.24 Pemanggilan Method *Class* LoginFragment dan LoginPresenter 217
Gambar 6.25 Format Request Web Service (JSON)..... 218
Gambar 6.26 Potongan Kode Program Web Service Index.php 218
Gambar 6.27 Potongan Kode Program Web Service Functions.php..... 219
Gambar 6.28 Potongan Kode Program Web Service DBOperations.php..... 220
Gambar 6.29 Potongan Kode Program Web Service upload_image.php 220

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 2.1 Penelitian Sebelumnya 7

Tabel 4.1 Kebutuhan Fungsional Aplikasi 33
Tabel 4.2 Kebutuhan Non Fungsional Aplikasi 34
Tabel 4.3 Arsitektur Aplikasi Existing *Service Desk* 34
Tabel 4.4 Pemetaan Kebutuhan Fungsional dan Fungsi Aplikasi 36
Tabel 4.5 Pemetaan Kode Program dan Fungsi Aplikasi 37
Tabel 4.6 Use Case Description Login 45
Tabel 4.7 Use Case Description Logout 45
Tabel 4.8 Use Case Description Lihat Notifikasi 46
Tabel 4.9 Use Case Description Search Knowledge 47
Tabel 4.10 Use Case Description Lihat Knowledge 47
Tabel 4.11 Use Case Description Lihat Info Buat Tiket 48
Tabel 4.12 Use Case Description Lihat Profile 48
Tabel 4.13 Use Case Description Edit Profile 49
Tabel 4.14 Use Case Description Ubah Password 50
Tabel 4.15 Use Case Description Lupa Password 50
Tabel 4.16 Use Case Description Register Member 51
Tabel 4.17 Use Case Description Membuat Tiket 51
Tabel 4.18 Use Case Description Lihat List Tiket 52
Tabel 4.19 Use Case Description Search Tiket 53
Tabel 4.20 Use Case Description Lihat Detail Tiket 53
Tabel 4.21 Use Case Description Balas Tiket 54
Tabel 4.22 Use Case Description Lihat Detail Tiket (Guest) 54
Tabel 4.23 Use Case Description Balas Tiket (Guest) 55
Tabel 4.24 Use Case Description Login 56
Tabel 4.25 Use Case Description Logout 57
Tabel 4.26 Use Case Description Lihat Notifikasi 58
Tabel 4.27 Use Case Description Search Knowledge 58
Tabel 4.28 Use Case Description Lihat Knowledge 59
Tabel 4.29 Use Case Description Lihat Info Buat Tiket 60
Tabel 4.30 Use Case Description Lihat Profile 60
Tabel 4.31 Use Case Description Edit Profile 61
Tabel 4.32 Use Case Description Ubah Password 61

Tabel 4.33 Use Case Description Lupa Password..... 62
Tabel 4.34 Use Case Description Register Member 63
Tabel 4.35 Use Case Description Membuat Tiket 63
Tabel 4.36 Use Case Description Lihat List Tiket 64
Tabel 4.37 Use Case Description Search Tiket..... 64
Tabel 4.38 Use Case Description Lihat Detail Tiket 65
Tabel 4.39 Use Case Description Balas Tiket..... 65
Tabel 4.40 Use Case Description Hapus Tiket 66
Tabel 4.41 Use Case Description Hapus Tiket (Via Detail) .. 67
Tabel 4.42 Use Case Description Lihat History Tiket 67
Tabel 4.43 Use Case Description Lihat Data Client 68
Tabel 4.44 Use Case Description Assign Tiket..... 68
Tabel 4.45 Use Case Description Assign Myself..... 69
Tabel 4.46 Use Case Description Ubah Status Tiket 69
Tabel 4.47 Desain Pemetaan Service Aplikasi..... 71
Tabel 5.1 Spesifikasi Komputer Pengembang 127
Tabel 5.2 Teknologi Pengembangan yang Digunakan..... 127
Tabel 5.3 Hasil Testing Role User (Member dan Non Member) 192
Tabel 5.4 Hasil Testing Role Ticket Worker 193

BAB I
PENDAHULUAN

Pada bab pendahuluan akan diuraikan proses identifikasi masalah penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir, manfaat kegiatan tugas akhir dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bab ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir dapat dipahami.

1.1 Latar Belakang

DPTSI atau Direktorat Pengembangan Teknologi dan Sistem Informasi merupakan salah satu direktorat yang dimiliki ITS (Institut Teknologi Sepuluh Nopember). DPTSI bertugas melakukan pengembangan serta pengelolaan layanan teknologi dan sistem informasi (TSI) terpadu di kampus ITS. DPTSI menyediakan dan mengelola beberapa layanan TI, antara lain layanan email, penyediaan koneksi wifi, integra, share, mailing list serta berbagai layanan TI Lainnya [1]. Sehingga DPTSI sebagai penyedia layanan harus mampu menyediakan layanan TI yang dibutuhkan oleh pengguna dan melakukan pengelolaan terhadap layanan yang diberikan.

Dalam penyediaan dan pengelolaan terhadap layanan untuk memenuhi harapan pengguna, DPTSI memiliki unit fungsional yang disebut sebagai *service desk*. Service desk DPTSI melayani seluruh pengguna layanan TI di ITS. Layanan service desk bertugas sebagai gardu terdepan yang mewakili untuk melakukan komunikasi dengan pengguna [2]. Melalui *service desk* tersebut, pengguna dapat menyampaikan permasalahan terhadap layanan TI yang digunakan, meminta tambahan layanan, dan permintaan hak akses layanan. *Service desk* memastikan pengguna menerima bantuan yang sesuai dalam jangka waktu yang ditentukan [3].

Untuk mendukung berjalannya *service desk* dalam pemenuhan serta pelaporan layanan, DPTSI akhirnya merilis *aplikasi service desk*. *Aplikasi service desk* adalah sebuah aplikasi yang digunakan untuk menyampaikan keluhan terkait insiden dan request bagi civitas akademika ITS ataupun pihak luar ITS yang ditujukan kepada unit/departemen/fakultas terkait [4], dimana user dapat melakukan akses kedalam aplikasi melalui web browser untuk membuat tiket, dan *ticket worker* dapat membalas tiket serta melakukan assign/mengarahkan tiket aduan ke unit/departemen/fakultas terkait.

Pada tahun 2016, sebanyak 65,2 juta orang di Indonesia telah menggunakan smartphone [5]. Berkembangnya penggunaan perangkat pintar/smartphone akhirnya memunculkan kebutuhan baru untuk berbagai layanan aplikasi di Indonesia, di ITS, khususnya *aplikasi service desk* yang telah dirilis DPTSI. Aplikasi service desk dianggap masih memiliki keterbatasan dalam sisi fleksibilitas pengoperasian. User/pembuat tiket menginginkan pengalaman penggunaan aplikasi yang lebih baik tanpa harus membuka browser untuk mengelola tiket yang telah dibuat, begitu pula ticket worker. Kedua user role menginginkan pengalaman terbaik untuk mengelola tiket pada mobile device yang dimiliki.

Oleh karena itu, penulis mengusulkan untuk membangun sebuah aplikasi *service desk* versi mobile, native, berbasiskan android untuk mendukung peran fungsi *service desk* dalam pemenuhan layanan sesuai harapan pengguna. Penulis memilih pengembangan secara *native*, karena didasarkan pada penelitian sebelumnya yang menunjukkan bahwa sebanyak 69% test case pengujian terkait performa, aplikasi yang dikembangkan secara native lebih unggul [6]. Hal ini menunjukkan bahwa secara empiris, performa yang ditawarkan aplikasi *native* jauh lebih baik ketimbang aplikasi yang dibangun secara *webview*. Aplikasi ini akan dibangun sesuai functional requirement yang ada pada aplikasi existing sesuai role yang telah ditetapkan sebelumnya. Aplikasi ini nantinya akan dibangun dengan

design pattern karena penggunaan design pattern terbukti meningkatkan *maintainability* [9] dan kemudahan dalam menjalankan testing [7]. Hal ini menyebabkan semakin mudahnya komponen aplikasi untuk dilakukan modifikasi, peningkatan fungsi, maintenance, dan perubahan-perubahan pada atributnya [10].

Design pattern yang dipilih adalah design pattern MVP (Model-View-Presenter). Design pattern MVP merupakan pattern yang bertujuan untuk menciptakan pemisahan yang sangat jelas antara Model, View, dan Presenter [11]. Pemilihan design pattern MVP dilandaskan pada struktur pattern yang benar benar fleksibel untuk mengimplementasikan suatu fungsi pada class yang dituju dan memudahkan dalam pembuatan berbagai skenario testing [12]. Alasan kedua karena separasi antar komponen yang ada, design pattern MVP memungkinkan penggunaan mock untuk melakukan test pada suatu unit dengan menjaga unit lain dalam keadaan utuh [13]. Adapun alasan terakhir pemilihan design pattern MVP adalah dukungan Google pada pengembangan codelab testingnya yang telah mengimplementasikan design pattern MVP [14]. Selain itu penulis juga akan melakukan blackbox testing yang bertujuan untuk memastikan seluruh kode program berjalan sesuai flow dan memenuhi seluruh *expected result* dalam skenario testing. Hal ini dilakukan untuk menjamin aplikasi telah *production ready* / siap untuk dinaikkan ke produksi untuk dilakukan deploy dan implementasi.

Pengerjaan tugas akhir ini diharapkan menjadi salah satu aspek pendukung untuk meningkatkan layanan yang diberikan oleh service desk DPTSI dalam pengelolaan tiket keluhan serta permintaan layanan dari sisi user dan ticket worker.

1.2 Rumusan Masalah

Merujuk pada latar belakang yang telah dikemukakan sebelumnya, maka rumusan masalah pada tugas akhir ini adalah sebagai berikut :

1. Bagaimana arsitektur aplikasi mobile layanan *service desk* ITS berbasis android ?
2. Bagaimana merancang dan membangun aplikasi mobile layanan *service desk* ITS berbasis android ?

1.3 Batasan Masalah

Merujuk pada latar belakang yang telah disebutkan diatas, Batasan masalah dalam tugas akhir ini adalah :

1. Fokus tugas akhir lebih kepada *client-side* aplikasi.
2. Aplikasi dikembangkan untuk role *user* dan ticket *worker*.
3. Pengujian hanya dilakukan pada aplikasi mobile (client-side), tidak untuk web service yang telah dibangun.

1.4 Tujuan Penelitian

Berdasarkan hasil perumusan dan batasan masalah yang telah disebutkan sebelumnya, maka tujuan yang dicapai dari tugas akhir ini adalah sebagai berikut :

1. Merancang arsitektur dan dokumen desain aplikasi mobile *service desk* ITS.
2. Mengembangkan aplikasi mobile *service desk* ITS berbasis android.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dapat diperoleh dari tugas akhir ini adalah sebagai berikut :

1. Memberi tambahan wawasan dan menyediakan referensi dalam dunia pengembangan aplikasi mobile berbasiskan android.

2. Pengembangan aplikasi mobile service desk dapat menunjang peningkatan layanan yang diberikan DPTSI dalam pengelolaan tiket keluhan serta permintaan layanan dari sisi user dan ticket worker.

1.6 Relevansi

Penulis pada tugas akhir ini akan menulis kode program aplikasi mobile berbasiskan android beserta pengujiannya. Sehingga relevansi tugas akhir ini terhadap laboratorium Infrastruktur Sistem dan Keamanan Teknologi Informasi (IKTI) ialah berkaitan dengan penerapan mata kuliah bidang keilmuan laboratorium IKTI yaitu Pemrograman Perangkat Bergerak dengan implementasi konsep software quality blacbox testing yang ada pada mata kuliah Analisis Desain Perangkat Lunak dan mata kuliah Konstruksi dan Pengujian Perangkat Lunak

Halaman ini sengaja dikosongkan

BAB II
TINJAUAN PUSTAKA

Pada bagian ini akan dijelaskan mengenai penelitian sebelumnya dan dasar teori yang dijadikan acuan atau landasan dalam pengerjaan tugas akhir ini.

2.1 Penelitian Sebelumnya

Penelitian yang dijadikan acuan dalam pengerjaan tugas akhir ini disajikan pada tabel 2.1 sebagai berikut :

Tabel 2.1 Penelitian Sebelumnya

1. A Tale of Two Fashions: An Empirical Study on the Performance of Native Apps and Web Apps on Android
Penulis, Tahun : Yun Ma., Xuanzhe Liu., Yi Liu., Yunxin Liu., dan Gang Huang; 2017 [6]
Hasil : Sebanyak 61% dari test case yang dijalankan menunjukkan bahwa performa native apps lebih baik ketimbang web apps dari sisi jumlah request HTTP(S), traffic/size file yang diunduh, round-trip time/waktu yang dibutuhkan dari request pertama dikirim sampai respon terakhir diterima, dan komsumsi energi dalam satuan joule.
Keterkaitan : Penelitian yang dilakukan berkaitan dengan data penunjang pengembangan aplikasi mobile service desk berbasiskan android dilakukakan secara native, tidak dengan menggunakan webview.
2. Design and Implementation of Android Based Mobile App For An Institute
Penulis, Tahun :

Golhar, Reetesh V., Vyawahare Prasann A., Borghare Pavan H., Manusmare Ashwini; 2016
Hasil : Peneliti mengimplementasikan dan mendokumentasikan pengembangan aplikasi mobile android dengan menggunakan android studio berbasiskan IntelliJ IDEA.
Keterkaitan : Penelitian yang dilakukan berkaitan dengan data mekanisme penunjang yang menjadi gambaran penulis dalam mengembangkan aplikasi android di environment yang sama berbasiskan IntelliJ IDEA.
3. Evaluating Impact of Design Patterns on Software Maintainability and Performance
Penulis, Tahun : Abdullah., Farooq; 2017
Hasil : Peneliti mendokumentasikan berbagai macam design pattern dalam penelitiannya dan melakukan analisis terhadap maintainability dan efisiensi dari tiap penggunaannya. Namun dalam penelitian ini tidak dimunculkan rekomendasi pemilihan design pattern yang tepat sesuai kebutuhan.
Keterkaitan : Penelitian yang dilakukan berkaitan dengan data penunjang implementasi design pattern dalam aplikasi yang dibuat penulis dalam rangka meningkatkan mai ntainability dari kode program yang akan dibuat oleh penulis.
4. A Journey Through the Land of Model-View-* Design Patterns
Penulis, Tahun : Syromiatnikov, Artem., Weyns Danny; 2014 [12]
Hasil :

Peneliti mendokumentasikan Model-View-* design pattern dan memberikan penjelasan yang jelas tentang kelebihan dan perbedaan di tiap pattern dari kelompok Model-View-*. Peneliti juga memberikan rekomendasi penggunaan tiap pattern dari kelompok Model-View-*.
Keterkaitan : Penelitian yang dilakukan berkaitan dengan data penunjang penulis dalam memilih design pattern MVP(Model-View-Presenter) untuk membangun aplikasi mobile.
5. An Architecture and Implement Model for Model-View-Presenter Pattern
Penulis, Tahun : Yang Zhang., Yanjing Luo; 2010 [11]
Hasil : Peneliti melakukan implementasi design pattern MVP(Model-View-Presenter) dan mendokumentasikan step implementasi ke dalam web application pada platform .NET.
Keterkaitan : Penelitian yang dilakukan berkaitan dengan data penunjang penulis dan menjadi gambaran umum dalam melakukan implementasi MVP(Model-View-Presenter) ke dalam kode program aplikasi mobile berbasis android.

2.2 Dasar Teori

Berikut merupakan teori-teori yang mendukung serta berkaitan dalam pengerjaan tugas akhir.

2.2.1 Service Desk

Service desk (dapat disebut sebagai help desk, support desk atau IT service center) merupakan unit dalam suatu organisasi yang berperan sebagai gerbang komunikasi antara penyedia layanan dengan pengguna (single point of contact) [2]. Service desk merupakan pihak pertama yang melakukan kontak langsung dengan pengguna terkait layanan TI yang disediakan oleh

organisasi. Pengguna dapat menghubungi bagian service desk apabila memiliki permasalahan atau pertanyaan terkait layanan TI yang digunakan. Service desk juga berperan untuk memastikan agar pengguna memperoleh value sebanyak mungkin dari layanan TI yang diberikan dengan berusaha menyelesaikan permasalahan yang dihadapi pengguna tersebut. Sebagai intinya, service desk berperan dalam membantu penyedia layanan TI untuk bertanggung jawab terhadap setiap laporan, masalah dan permintaan pengguna mulai laporan tersebut diterima hingga selesai dan ditutup. Service desk Dalam ITIL, service desk termasuk ke dalam fase service operation.

Terdapat empat pilihan struktur *service desk* dalam organisasi, yaitu [2] :

- ***Local Service desk***
  Service desk yang dibangun sebanyak satu unit untuk menangani setiap unit pelanggan. Seperti contoh adanya service desk di setiap jurusan di ITS untuk menangani permasalahan TI di setiap jurusan

- **Centralised Service desk**
  Merupakan service desk yang dibangun sebanyak satu unit untuk menangani semua unit pelanggan. Seperti contoh adanya service desk di pusat ITS untuk menangani permasalahan TI semua jurusan

- **Virtual Service desk**
  Merupakan service desk yang menyediakan nomor telepon sebagai kontak untuk menghubungi apabila terdapat permalasahan, namu pengguna tidak mengetahui lokasi fisik service desk tersebut. Seperti contoh call center.

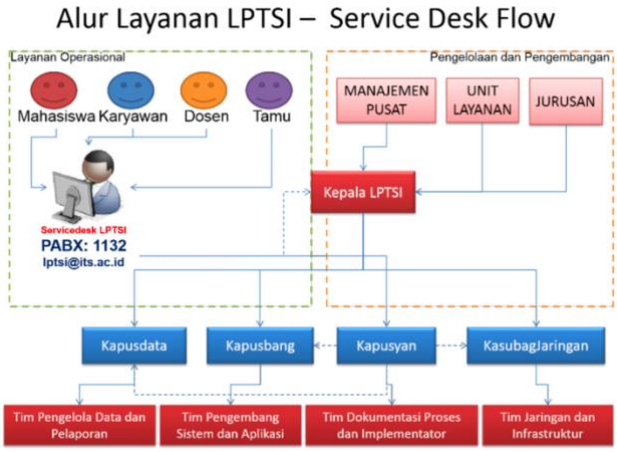
- **Follow-the-Sun Service desk**
  Merupakan service desk yang menyediakan layanan service desk setiap hari (7 hari) selama 24 jam dan memiliki pelanggan di berbagai negara dengan perbedaan waktu di setiap negara. Seperti contoh suatu perusahaan di Inggris yang memiliki service desk di beberapa wilayah negara, yaitu di Inggris dan Amerika Serikat. Apabila terdapat laporan masuk antara pukul 08.00 hingga 21.00 waktu Inggris, maka laporan akan diteruskan ke service desk di Inggris, dan apabila laporan masuk antara pukul 21.00- 08.00, maka laporan akan diteruskan ke service desk di Amerika Serikat.

Service desk merupakan sebuah unit fungsional yang menangani berbagai kegiatan layanan, baik melalui web interface atau panggilan telepon. Service desk dinilai sebagai pendekatan pertama yang tepat dan sesuai untuk menangani permasalahan TI. Berikut beberapa manfaat yang dapat diperoleh organisasi dengan menerapkan service desk [15]:

- Meningkatkan layanan dan kepuasan pelanggan
- Meningkatkan aksesibilitas karena berperan sebagai satu titik kontak, komunikasi dan informasi
- Meningkatkan fokus dan pendekatan yang proaktif terhadap penyediaan layanan kepada pengguna
- Pemenuhan permintaan pengguna dengan kualitas yang lebih baik
- Meningkatkan penggunaan dukungan sumber daya TI dan meningkatkan produktivitas personel bisnis
- Manajemen informasi yang lebih bermanfaat dalam mendukung keputusan

2.2.2 Service Desk DPTSI

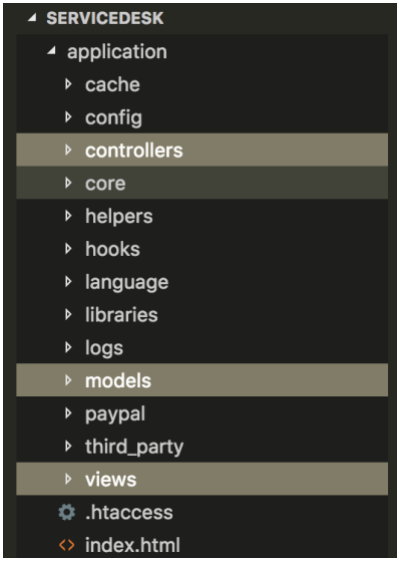
Service desk DPTSI merupakan unit fungsional yang dimiliki oleh DPTSI, tepatnya pada Sub Direktorat Layanan dan Teknologi Informasi DPTSI. Service desk DPTSI menangani berbagai macam keluhan dan permasalahan layanan TI yang terjadi di lingkungan ITS. Permasalahan layanan yang dikelola oleh service desk terkait dengan insiden layanan TI, permintaan layanan TI, serta pengelolaan akses pengguna terhadap layanan TI. Yang termasuk pengguna layanan TI di lingkungan ITS adalah mahasiswa, karyawan, dosen dan tamu. Pengguna layanan ini dapat menyampaikan permasalahan dan keluhan yang dialami ketika menggunakan layanan TI kepada service desk DPTSI. Penyampaian permasalahan dan keluhan dapat dilakukan kepada service desk DPTSI melalui website servicedesk.its.ac.id, atau datang secara langsung ke DPTSI untuk menyampaikan permasalahan [1]. Berikut service desk flow yang digunakan oleh DPTSI saat ini yang ditunjukkan pada gambar 2.1 sebagai berikut



Gambar 2.1 Service Desk Flow

2.2.2.1 Aplikasi Existing Service Desk DPTSI

Hingga saat ini telah terdapat aplikasi existing service desk yang telah berjalan. Aplikasi service desk dibangun dengan design pattern *MVC* menggunakan framework *Code Igniter*. Arsitektur aplikasi terbagi menjadi 3 bagian yakni bagian Model, View, dan Controller yang dapat dilihat pada gambar 2.2 berikut.



Gambar 2.2 Arsitektur Aplikasi Service Desk

Adapun aplikasi existing *service desk* tidak dibangun dengan *web service*. Hal ini berarti aplikasi existing saat menjalankan fungsinya langsung menembakkan transaksi ke database tanpa web service (middleware). Berikut contoh query yang ditembakkan langsung dari model ke database sebagaimana ditunjukkan pada gambar 2.3.

Login_model.php

```php
<?php

class Login_Model extends CI_Model
{

    public function getUser($email, $pass)
    {
        return $this->db->select("ID")
        ->where("email", $email)->where("password", $pass)->get("users");
    }

```

Gambar 2.3 Contoh Query Aplikasi Existing Service Desk

2.2.3 Unified Modeling Language (UML)

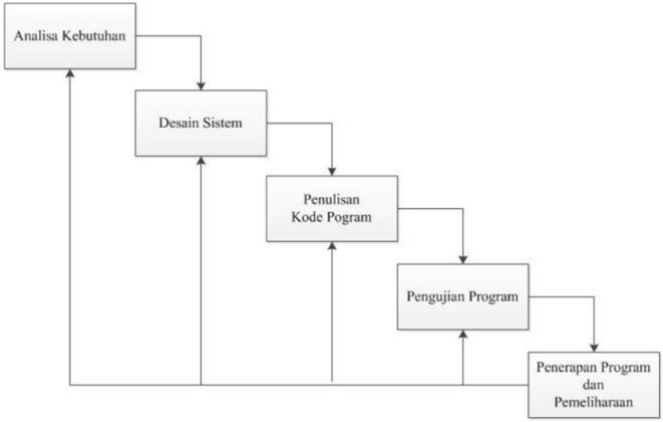
UML adalah sebuah bahasa yang telah menjadi standart dalam industri untuk visualisasi, merancang, dan mendokumentasikan sistem perangkat lunak [25]. Pengembangan sistem perangkat lunak lebih baik dilihat dari beberapa sudut pandang untuk mendapatkan gambaran perangkat lunak secara menyeluruh. Oleh karena itu UML menyediakan 9 jenis diagram atau model untuk menggambarkan beberapa sudut pandang sistem perangkat lunak. Berikut adalah daftar 9 diagram tersebut :

- Class Diagram
- Object Diagram
- Usecase Diagram
- Sequence Diagram
- Collaboration Diagram
- Statechart Diagram
- Activity Diagram
- Component Diagram
- Deployment Diagram

Dalam membuat UML dibutuhkan tools atau perangkat lunak yang mendukung bahasa UML. Salah satu pengembangan perangkat lunak yang menuntut menggunakan UML dalam proses pengerjaannya adalah ketika membangun perangkat lunak berbasis objek atau Object Oriented Programming (OOP). Penggunaan diagram tidak semuanya harus dibuat. Penggunaan diagram disesuaikan dengan keadaan kebutuhan pengembangan.

2.2.4 Model Rekayasa Perangkat Lunak Waterfall

Pengembangan perangkat lunak memerlukan suatu model atau metodologi dalam proses pengembangannya. Hal ini disebut juga sebagai Software Development Life Cycle (SDLC). Model atau metodologi tersebut digunakan sebagai petunjuk untuk mengelola bagaimana seharunya perangkat lunak dikembangkan [26]. Salah satu model dalam SDLC adalah waterfall. Model waterfall digambarkan secara sederhana seperti gambar 2.4 dibawah



Gambar 2.4 Siklus Rekayasa Perangkat Lunak Waterfall

Metode waterfall merupakan metode dengan pendekatan alur hidup perangkat lunak secara sekuensial dan terurut [27]

dimulai dari analisa kebutuhan, desain, tahapan implementasi desain (pengkodean), pengujian, dan tahapan pendukung lain yang dikerjakan secara bertahap. Pada pengembangan aplikasi *service desk* ini akan digunakan metode waterfall.

2.2.5 Android

2.2.5.1 Pengertian dan Sejarah

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel Android pertama mulai dijual pada bulan Oktober 2008.

Android adalah sistem operasi dengan sumber terbuka, dan Google merilis kodenya di bawah Lisensi Apache. Kode dengan sumber terbuka dan lisensi perizinan pada Android memungkinkan perangkat lunak untuk dimodifikasi secara bebas dan didistribusikan oleh para pembuat perangkat, operator nirkabel, dan pengembang aplikasi [16].

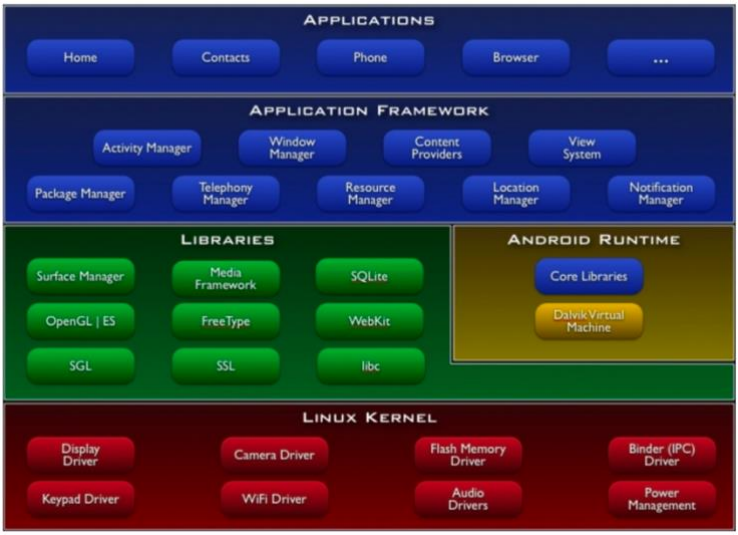
Faktor-faktor di atas telah memberikan kontribusi terhadap perkembangan Android, menjadikannya sebagai sistem operasi telepon pintar yang paling banyak digunakan di dunia, mengalahkan Symbian pada tahun 2010. Android juga menjadi pilihan bagi perusahaan teknologi yang menginginkan sistem operasi berbiaya rendah, bisa dikustomisasi, dan ringan untuk perangkat berteknologi tinggi tanpa harus mengembangkannya dari awal. Sifat Android yang terbuka juga telah mendorong munculnya sejumlah besar komunitas pengembang aplikasi untuk menggunakan kode sumber terbuka sebagai dasar proyek

pembuatan aplikasi, dengan menambahkan fitur-fitur baru bagi pengguna tingkat lanjut atau mengoperasikan Android pada perangkat yang secara resmi dirilis dengan menggunakan sistem operasi lain.

Pada November 2013, Android menguasai pangsa pasar telepon pintar global, yang dipimpin oleh produk-produk Samsung, dengan persentase 64% pada bulan Maret 2013. Pada Juli 2013, terdapat 11.868 perangkat Android berbeda dengan beragam versi. Keberhasilan sistem operasi ini juga menjadikannya sebagai target ligitasi paten "perang telepon pintar" antar perusahaan-perusahaan teknologi. Hingga bulan Mei 2013, total 900 juta perangkat Android telah diaktifkan di seluruh dunia, dan 48 miliar aplikasi telah dipasang dari Google Play Store.

2.2.5.2 Arsitektur

merupakan metode dengan pendekatan alur hidup perangkat lunak secara sekuensial dan terurut [27] dimulai dari analisa kebutuhan, desain, tahapan implementasi desain (pengkodean), pengujian



Gambar 2.5 Arsitektur Sistem Operasi Android

Adapun kernel yang digunakan adalah kernel linux seri 2.6 yang dimodifikasi untuk memenuhi kebutuhan khusus dalam manajemen daya, manajemen memory, dan runtime environment. tepat diatas kernel berjalan suatu daemon atau sebuah proses yang tidak dapat diinterupsi seperti bluez untuk menjalankan bluetooth dan wpa.supplicant untuk enkripsi WIFI. Dikarenakan android berjalan dengan alokasi memori yang cenderung kecil dan dengan CPU yang rendah daya, library untuk CPU dan GPU dalam menjalankan tugas-tugas yang bersifat intensif langsung di compile ke dalam kode native dari device itu sendiri. Library standar seperti libc atau libm telah dikembangkan khusus untuk melakukan pekerjaan dengan alokasi memori yang cenderung kecil. Kemudian di layer native library ini pula terjadi handling untuk akses layar (windows manager). Beranjak ke layer android runtime, di dalam layer ini terdapat sebuah virtual machine bernama Dalvik dan library utama java core. Dalvik virtual machine bertugas untuk

mengimprentasikan kode java ke dalam byte code dalvik. Dalvik sendiri tercompile dari layer native libary milik sistem operasi android. Kemudian berpindah ke framework layer. Framework layer berisikan framework yang dibangun dengan java yang menyediakan abstraksi hingga ke level native library dan dalvik. Sementara layer terluar, layer aplikasi berjalan pada virtual machine dalvik yang terdiri dari beberapa component, antara lain : Activities, services, broadcast receivers, dan content providers. Komponen satu dapat berinteraksi dengan komponen lainnya dalam satu layer application.teknologi [17].

2.2.5.3 Riwayat Versi

Pada November 2013, Android menguasai pangsa pasar telepon pintar global, yang dipimpin oleh produk-produk Samsung, dengan persentase 64% pada bulan Maret 2013. Pada Juli 2013, terdapat 11.868 perangkat Android berbeda dengan beragam versi. Keberhasilan sistem operasi ini juga menjadikannya sebagai target ligitasi paten "perang telepon pintar" antar perusahaan-perusahaan teknologi. Hingga bulan Mei 2013, total 900 juta perangkat Android telah diaktifkan di seluruh dunia, dan 48 miliar aplikasi telah dipasang dari Google Play.

Tabel di bawah ini menampilkan data mengenai persentase jumlah perangkat Android yang mengakses Google Play baru-baru ini, dan menjalankan platform Android versi tertentu hingga tanggal 9 September 2014. Android 4.1/4.2/4.3 *Jelly Bean* adalah versi Android yang paling banyak digunakan, yakni sekitar 53,7% dari keseluruhan perangkat Android di seluruh dunia.

Versi	Nama kode	Tanggal rilis	Level API	Distribusi
7.0	*Nougat*	22 Agustus 2016	24	Kurang dari 0.1%
6.0	Marshmallow	19 Agustus 2015	23	
5.x	*Lollipop*	15 Oktober 2014	21	
4.4.x	*KitKat*[179]	31 Oktober 2013[180]	19	24,5%
4.3.x	*Jelly Bean*	24 Juli 2013	18	8%
4.2.x	*Jelly Bean*	13 November 2012	17	20,7%
4.1.x	*Jelly Bean*	9 Juli 2012	16	25,1%
4.0.3–4.0.4	*Ice Cream Sandwich*	16 Desember 2011	15	9,6%
3.2	*Honeycomb*	15 Juli 2011	13	
3.1	*Honeycomb*	10 Mei 2011	12	
2.3.3–2.3.7	*Gingerbread*	9 Februari 2011	10	11,7%
2.3–2.3.2	*Gingerbread*	6 Desember 2010	9	
2.2	*Froyo*	20 Mei 2010	8	0,7%
2.0–2.1	*Eclair*	26 Oktober 2009	7	
1.6	*Donut*	15 September 2009	4	
1.5	*Cupcake*	30 April 2009	3	

Gambar 2.6 Version History Sistem Operasi Android

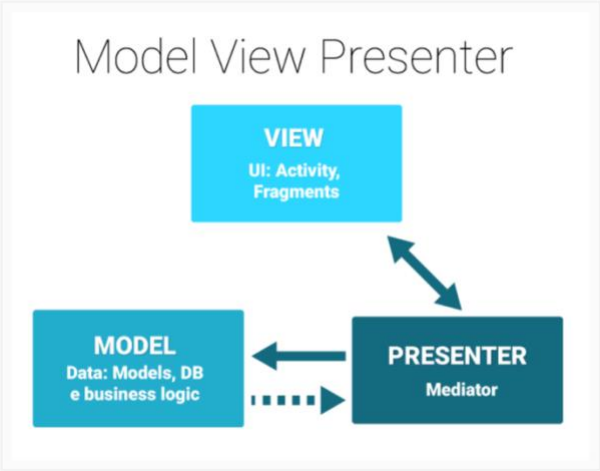
2.2.6 Design Pattern

Design pattern merupakan solusi umum untuk mengatasi permasalahan yang terjadi di dunia software engineering. Desing pattern bukan suatu desain pasti yang akan menyelesaikan berbagai permasalahan programming yang ada, melainkan design pattern akan menyediakan template struktur yang dapat digunakan dan disesuaikan dengan kondisi yang ada [18]. Design pattern setidaknya harus memiliki 4 komponen utama yakni : nama pattern, permasalahan, solusi dan konsekuensi. Nama pattern harus dapat mendeskripsikan problem desain. Nama pattern juga harus memberikan dukungan untuk meningkatnya kemudahan komunikasi dan transfer ilmu antar pengembang aplikasi. Kemudian design pattern juga harus dapat merepresentasikan konteks serta kapan seorang pengembang aplikasi harus menggunakannya [19].

Design pattern dapat dikategorikan menjadi 3, yakni : creational patterns, structural pattern dan behavioural pattern. Creational pattern berkaitan dengan bagaimana membuat objek, structural pattern berkaitan dengan simplifikasi dari realisasi hubungan antar objek/entitas, dan behavioral pattern berkaitan dengan bagaimana memberikan objek suatu tanggung jawab. Karena salah satu karakteristik dari arsitektur yang scalable adalah high cohession dan loose coupling. High cohession berarti komponen dari software harus fokus dalam mengerjakan hanya satu hal, loose coupling berarti komponen software tidak harus tau tentang internal antar komponen satu dengan lainnya [19].

2.2.7 Design Pattern MVP

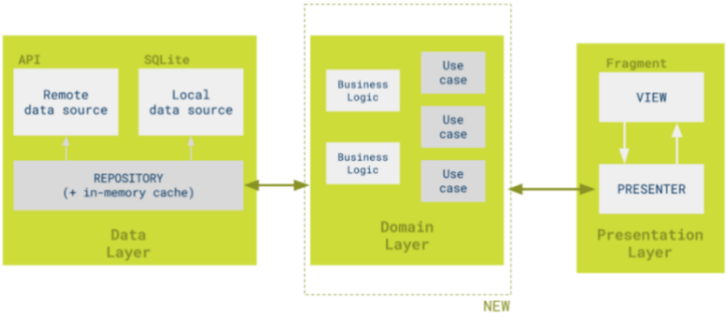
Design pattern MVP merupakan pattern yang memiliki konsep sama dengan MVC, tapi dengan paradigma modern yang akhirnya menghasilkan suatu design pattern yang benar-benar melakukan separasi pada tiap komponennya. Seperti ditampilkan pada gambar 2.7, *view* benar-benar menjadi pasif dalam design pattern ini.



Gambar 2.7 Arsitektur Design Pattern Model-View-Presenter

Ia tidak bisa menerima data apapun dari model, dan tetap benar-benar pasif hingga presenter memberitahu atau memerintahkan view untuk melakukan sesuatu. Presenter benar-benar terikat kuat dengan satu view, dan berperan sebagai mediator antara model dan view. Presenter menerima data dari model, dan memerintahkan view untuk melakukan render terhadap data yang diterima, yang pada akhirnya view akan memproses aksi yang dilakukan oleh user. Dengan ini Komponen view dan model otomatis menjadi *loose coupling* dimana hal ini dapat memudahkan perubahan, penambahan, serta testing fitur [19].

Membiarkan model menyimpan dan mengolah *business logic* membuat aplikasi menjadi lebih mudah untuk dites. Arsitektur yang bersih dapat mengecilkan kompleksitas model dengan memasukkan sebuah layer antara presenter dan data model yang disebut dengan layer domain.



Gambar 2.8 Arsitektur Design Pattern Model-View-Presenter

Sebagaimana ditampilkan pada gambar 2.8, layer domain menghandle semua *application logic*, atau yang dapat disebut dengan *use cases*, bersamaan dengan proses menerima data dari *model*. *Model* dalam arsitektur ini hanya menghandle *business logic*. *Business logic* dalam design pattern MVP benar-benar independen dari segala interaksi dengan user, membuat aplikasi

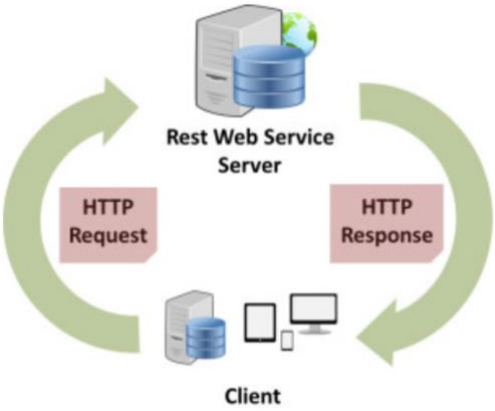
makin fleksibel dan meningkatkan maintainability. Dengan memisahkan proses kedalam layer model, domain, dan data, arsitektur ini berarti telah berhasil memisahkan antara business logic dan alur bagaimana suatu data akan disimpan. Hal ini berarti semakin mudah untuk melakukan *mock* dan melakukan test pada *business logic*. Dengan design pattern ini pengembang aplikasi hanya tinggal menduplikasi 3 layer yang ada untuk menerapkan suatu arsitektur pengembangan aplikasi yang benar-benar bersih dan modular [19].

2.2.8 Blackbox Testing

Salah satu metode pengujian pada pengembangan aplikasi yang dilakukan adalah blackbox testing. Blackbox testing adalah pengujian yang dilakukan untuk memeriksa apakah fungsional aplikasi berjalan dengan baik dan benar tanpa mengetahui proses yang terjadi di dalam aplikasi [31].

2.2.9 Web Service

Web service adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda.



Gambar 2.9 Ilustrasi Web Service

Contoh implementasi dari web service antara lain adalah SOAP dan REST. Web service yang berbasis arsitektur REST kemudian dikenal sebagai RESTful web services. Layanan web ini menggunakan metode HTTP untuk menerapkan konsep arsitektur REST [23].

2.2.9.1 RESTful Web Service

REST (REpresentational State Transfer) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP (Hypertext Transfer Protocol) sebagai protocol untuk komunikasi data. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000 [24].

Pada arsitektur REST, REST server menyediakan resources (sumber daya/data) dan REST client mengakses dan menampilkan resource tersebut untuk penggunaan selanjutnya. Setiap resource diidentifikasi oleh URIs (Universal Resource Identifiers) atau global ID. Resource tersebut direpresentasikan

dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

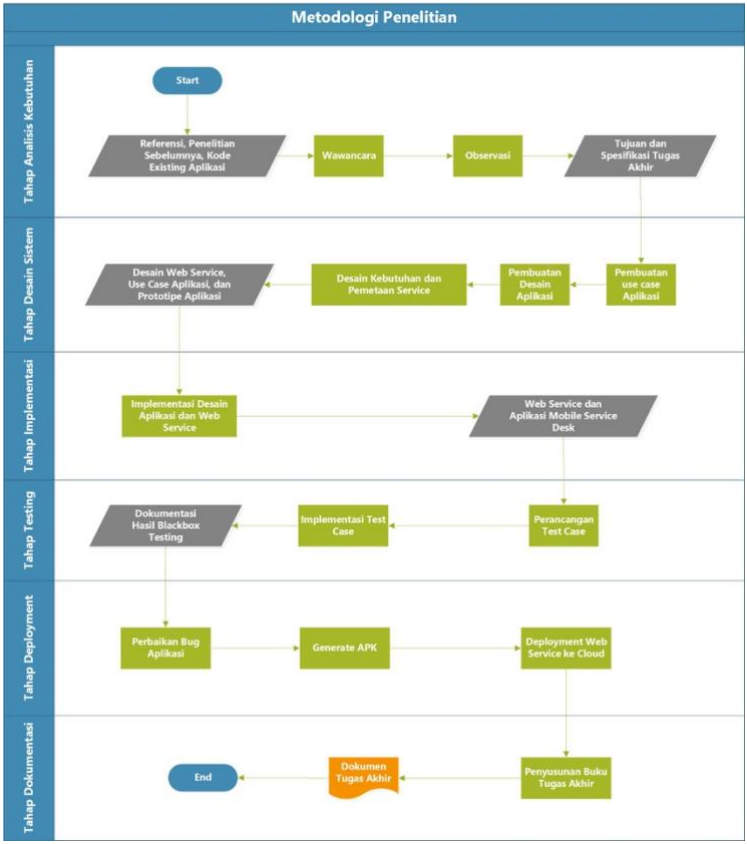
Keuntuangan REST [24] :

- Bahasa dan platform agnostic
- Lebih sederhana/simpel untuk dikembangkan ketimbang SOAP
- Mudah dipelajari, tidak bergantung pada tools
- Ringkas, tidak membutuhkan layer pertukaran pesan (messaging) tambahan
- Secara desain dan filosofi lebih dekat dengan web

Halaman ini sengaja dikosongkan

BAB III
METODE PENELITIAN

Pada bab ini akan dijelaskan mengenai tahapan ysang dilakukan dalam pengerjaan tugas akhir menggunakan SDLC Waterfall yang telah disesuaikan beserta deskripsi dan penjelasannya.



Gambar 3.1 Metodologi Pengerjaan Tugas Akhir

3.1 Tahap Analisis Kebutuhan

Pada tahap ini akan dilakukan penggalian kebutuhan mengenai aplikasi mobile berbasis android yang akan dibuat. Penggalian kebutuhan nantinya akan menjadi data penunjang pada tahapan penelitian berikutnya.

3.1.1 Referensi, Penelitian Sebelumnya, Kode Program Aplikasi Existing

Merupakan input pertama dari dimulainya pengerjaan tugas akhir. Referensi serta penelitian sebelumnya yang dimaksud merupakan pikiran dan hasil penelitian terdahulu yang dihimpun penulis sebagai landasan berjalannya pengerjaan tugas akhir, ditambah kode existing aplikasi service desk ITS yang telah berjalan untuk dimanfaatkan pada kegiatan observasi.

3.1.2 Wawancara

Pada tahapan ini akan dilakukan wawancara terhadap karyawan atau pihak service desk DPTSI untuk mengetahui kondisi arsitektur aplikasi service desk existing yang telah berjalan. Selain itu dilakukannya wawancara bertujuan untuk menggali kebutuhan pengguna yang akan dirumuskan menjadi kebutuhan fungsional dan non-fungsional pengembangan aplikasi.

3.1.3 Observasi

Pada tahapan ini akan dilakukan observasi terhadap struktur kode existing aplikasi service desk, meliputi mekanisme jalannya kode program untuk membuat web service RESTful yang akan dikonsumsi oleh aplikasi mobile yang akan dibuat. Pada tahap observasi juga dilakukan penyesuaian antara kemungkinan pengembangan yang akan dilakukan dan kebutuhan pengguna yang didapat dari hasil wawancara.

3.1.4 Tujuan dan Spesifikasi Tugas Akhir

Tujuan dan spesifikasi tugas akhir merupakan output yang didapat dari tahap analisis kebutuhan. Entitas ini akan mendefinisikan scope pengembangan tugas akhir serta

teknologi yang dipakai dalam pengembangan produk tugas akhir. Tujuan dan spesifikasi tugas akhir akan digunakan sebagai input proses pertama pada tahap desain sistem.

3.2 Tahap Desain Sistem

Pada tahap ini dilakukan perancangan dan desain aplikasi service desk DPTSI. Perancangan dan desain menggunakan Unified Model Language (UML).

3.2.1 Pembuatan Use Case Aplikasi

Pada tahapan ini akan dilakukan pembuatan use case aplikasi mobile service desk berbasis android sesuai kebutuhan pengguna yang didapatkan dari tahapan sebelumnya.

3.2.2 Pembuatan Desain Aplikasi

Penulis akan membuat prototipe aplikasi / mockup desain yang disesuaikan dengan kebutuhan pengguna. Pada tahapan ini juga dibuat sequence diagram guna menggambarkan jalannya kode program yang akan dibuat berdasarkan desain UI yang telah dibuat.

3.2.3 Desain Kebutuhan dan Pemetaan Service

Pada tahapan ini akan dilakukan pencatatan query yang digunakan untuk kemudian dipetakan sesuai use case yang telah dibuat. dibuat sebagai web service. Pemetaan yang dihasilkan akan digunakan sebagai functional requirement dalam pembuatan REST API yang akan dikonsumsi aplikasi mobile. Pembuatan API disesuaikan dengan prototipe aplikasi yang telah dibuat.

3.2.4 Desain Web Service, Use Case Aplikasi, dan Prototipe Aplikasi

Entitas ini merupakan output dari tahap desain sistem yang akan digunakan dalam proses pertama pada tahap implementasi.

3.3 Tahap Implementasi

Implementasi merupakan tahapan pembuatan kode aplikasi dan kode database aplikasi berdasarkan desain yang telah disepakati. Kode aplikasi dibuat dengan menggunakan design pattern MVP.

3.3.1 Implementasi Desain Aplikasi dan Web Service

Pada tahapan ini dilakukan pembuatan aplikasi sesuai use case dan prototipe yang telah dibuat. Kemudian dilakukan implementasi desain web service pula berdasarkan pemetaan query terhadap aplikasi existing *service desk* yang telah dilakukan sebelumnya. Dalam tahap ini juga dilakukan penyesuaian antara aplikasi dan web service berdasarkan kebutuhan data sesuai use case dan prototipe yang telah dibuat.

3.3.2 Web Service dan Aplikasi Mobile Service Desk

Entitas ini merupakan output dari tahap implementasi, dimana *deliverable* tugas akhir dihasilkan sesuai desain sistem yang telah dibuat pada tahap sebelumnya.

3.4 Tahap Testing

Pada tahapan ini, akan dilakukan pengujian berupa blackbox testing.

3.4.1 Perancangan Test Case

Penulis pada tahapan ini akan merancang skenario testing untuk dilakukan pada tahapan berikutnya. Adapun perencanaan skenario berdasarkan functional requirement aplikasi.

3.4.2 Implementasi Test Case

Setelah rancangan skenario testing siap maka penulis akan menjalankan pengujian untuk mengecek apakah seluruh scenario memenuhi expected result.

3.4.3 Dokumentasi Hasil Blackbox Testing

Entitas ini merupakan output dari tahap testing yang berisi hasil pengujian fungsional / blacbox testing. Hasil testing ini akan digunakan sebagai bahan perbaikan bug aplikasi terakhir untuk kemudian digenerate sebagai apk.

3.5 Tahap Deployment

Pada tahapan ini, dilakukan deployment aplikasi berupa generate apk dan melakukan deployment web service yang telah dibuat ke cloud, agar apk yang di generate dapat digunakan sebagai aplikasi beta dan dilakukan pengujian oleh pihak internal DPTSI.

3.5.1 Perbaikan Bug Aplikasi

Penulis pada tahap ini menggunakan hasil pengujian blackbox testing untuk memperbaiki kode program, hingga seluruh scenario test sesuai dengan expected result yang dihasilkan.

3.5.2 Generate APK

Jika telah memenuhi seluruh rancangan test case yang ada, maka pada tahap ini dilakukan generate file apk dari kode program yang telah dibuat.

3.5.3 Deployment Web Service ke Cloud

Setelah apk siap, dilakukan deployment web service / API yang telah dibuat ke cloud untuk kemudian apk dapat digunakan oleh banyak orang guna dilakukan pengujian oleh pihak internal DPTSI.

3.6 Tahap Dokumentasi

Pada tahapan ini, laporan tugas akhir akan dibuat dimana laporan tugas akhir yang dimaksud akan mendokumentasikan setiap langkah yang telah dilakukan, hasil yang dihasilkan setiap langkah, kesimpulan serta saran untuk penelitian kedepannya.

3.6.1 Penyusunan Buku Tugas Akhir

Pada tahapan terakhir ini akan dilakukan pembuatan laporan dalam bentuk buku tugas akhir yang disusun sesuai format yang telah ditentukan. Buku ini berisi dokumentasi langkah-langkah pengerjaan tugas akhir secara rinci. Buku ini diharapkan dapat bermanfaat sebagai referensi untuk pengerjaan penelitian lain, serta sebagai acuan untuk pengembangan lebih lanjut terhadap topik pengembangan yang serupa.

3.6.2 Dokumen Tugas Akhir

Adapun dokumen tugas akhir merupakan output dari keseluruhan proses penelitian yang dilakukan oleh penulis sesuai metodologi.

BAB IV
PERANCANGAN

Bab ini menjelasakan mengenai perancangan dari desain hingga dihasilkannya luaran tugas akhir.

4.1 Analisis Kebutuhan

Tahap awal dari perancangan dimulai dari mengumpulkan informasi mengenai kondisi dan teknologi yang digunakan aplikasi existing *service desk*. Pengambilan informasi dilakukan dengan menggunakan dua buah metode yaitu observasi dan wawancara.

4.1.1 Wawancara

Wawancara dilakukan kepada 2 orang pihak internal DPTSI. Narasumber pertama yakni Hanim Maria Astuti S.Kom., M.Sc. yang memilki peran sebagai Kasubdit Layanan Teknologi dan Sistem Informasi sedangkan narasumber kedua, Rizqi Rinaldi, memiliki peran sebagai Administrator *Service Desk*. Dalam wawancara, hal yang ditanyakan adalah bahasan seputar kebutuhan pengguna beserta kemungkinan pengembangan yang dapat dilakukan. Sehingga dari wawancara yang dilakukan didapat beberapa kebutuhan fungsional dan nonfungsional untuk user (member dan non member) dan ticket worker sebagai berikut :

1. Kebutuhan Fungsional

Tabel 4.1 Kebutuhan Fungsional Aplikasi

Kode	Aktor	Deskripsi
F1	User dan Ticket Worker	Aplikasi dapat digunakan untuk manajamen tiket sesuai role (kecuali edit tiket).

F2	User dan Ticket Worker	Aplikasi menyediakan informasi penggunaan aplikasi.
F3	User dan Ticket Worker	Aplikasi memungkinkan untuk diakses dengan dan tanpa menggunakan akun.
F4	User dan Ticket Worker	Aplikasi menyediakan menu manajemen user untuk mengubah info pengguna.
F5	User dan Ticket Worker	Aplikasi menyediakan menu notifikasi terkait informasi perubahan tiket

2. Kebutuhan Non Fungsional

Tabel 4.2 Kebutuhan Non Fungsional Aplikasi

Kode	Aktor	Deskripsi
NF1	User dan Ticket Worker	Aplikasi memilliki tampilan yang menarik
NF2	User dan Ticket Worker	Aplikasi memiliki performa yang baik

4.1.2 Observasi

Metode ini dilakukan dengan mengamati aplikasi *service desk* existing lengkap beserta kode program yang ada. Dari hasil observasi yang dilakukan didapatkan tumpukan teknologi yang digunakan aplikasi existing *service desk* sebagai berikut.

Tabel 4.3 Arsitektur Aplikasi Existing *Service Desk*

No	Tumpukan Teknologi	Teknologi yang digunakan
1	Bahasa Pemrograman	PHP
2	*Framework*	Code Igniter
3	*Runtime Environment*	XAMPP - Apache

		- Phpymyadmin
4	Basis Data	mysql
5	Daftar *library*	Bootstrap Jquery Facebook Auth (Off) Google Auth (Off) Twitter (Off) Stripe (Off) ReCaptcha (Off)

Selain tumpukan teknologi, didapati pula arsitektur aplikasi service desk yang ada pada saat ini sebagaimana dapat dilihat pada gambar 4.1.

Gambar 4.1 Diagram Arsitektur Aplikasi *Service Desk* Existing

Dari hasil wawancara dan observasi yang dilakukan, dibuat pemetaan fungsi yang ada pada aplikasi berdasarkan kebutuhan

fungsional yang telah didapat dari hasil wawancara. Adapun tabel pemetaan yang dihasilkan dapat dilihat pada tabel 4.4.

Tabel 4.4 Pemetaan Kebutuhan Fungsional dan Fungsi Aplikasi

Kode	Aktor	Fungsi dalam aplikasi
F1	User	Buat Tiket Lihat List Tiket Search Tiket Lihat Detail Tiket Balas Tiket Check Tiket (Non Member) Balas Tiket (Non Member)
	Ticket Worker	Buat Tiket Lihat List Tiket Search Tiket Lihat Detail Tiket Balas Tiket Hapus Tiket Hapus Tiket (Via Detail) Lihat History Tiket Lihat Data Client Assign Tiket Assign Myself Ubah Status Tiket
F2	User	Search Knowledge Lihat Knowledge Lihat Info Buat Tiket
	Ticket Worker	Search Knowledge Lihat Knowledge Lihat Info Buat Tiket
F3	User	Login Logout
	Ticket Worker	Login Logout
F4	User	Lihat Profile Edit Profile Ubah Password Lupa Password Register Member

	Ticket Worker	Lihat Profile Edit Profile Ubah Password Lupa Password Register Member
F5	User	Lihat Notifikasi
	Ticket Worker	Lihat Notifikasi

Kemudian dari hasil pemetaan fungsi yang didapat, dilakukan pemetaan kode program (controller dan database) aplikasi existing berdasarkan kode program yang digunakan sebagai bahan desain aplikasi dan web service yang akan dibuat sebagai berikut.

Tabel 4.5 Pemetaan Kode Program dan Fungsi Aplikasi

Fungsi	Controller (Function)	Model (Query Function)
Login	Controller : Login Function : 1. Pro () 2. Login_protect ()	Pro () : 1. User_model -> check_block_ip 2. Login_model -> get_login_attempts 3. Login_model -> getUserByEmail 4. Login_model -> getUserByUsername 5. Library -> update_online_timestamp Login_protect () : 1. Login_model -> get _login_attempts 2. Login_model -> update_login_attempts 3. Login_model -> add_login_attempts
Logout	Controller : Login	-

	Function : Logout ()	
Lihat Notifikasi	Controller : Client Function : Notifications_page ()	Notifications_page () : 1. User_model -> get_notifications_unread 2. User_model -> get_notifications 3. User_model -> update_notification
Search Knowledge	Controller : Knowledge Function : Datatables	1. Knowledge_model -> get_articles
Lihat Knowledge	Controller : Knowledge Function : 1. Article_page () 2. Categories () 3. Cat_page ()	Article_page () : 1. Knowledge_model -> get_articles Categories () : 1. Knowledge_model -> get_categories Cat_page () : 1. Knowledge_model -> get_categories_dt
Lihat Info Buat Tiket	Index (menu)	-
Lihat Profile	Controller : Profile Function : Index ()	Index () : 1. User_model -> get_user_by_username 2. User_model -> get_user_groups 3. User_model -> get_custom_field_answer
Edit Profile	Controller : User_settings Function :	Pro () : 1. Register_model -> checkEmailIsFree

	Pro ()	2. User_model -> update_user 3. User_model -> update_custom_field
Ubah Password	Controller : Login Function : resetpw_pro ()	Resetpw_pro () : 1. login_model -> updatePassword
Lupa Password	Controller : Login Function : forgotpw_pro ()	Forgotpw_pro () : 1. login_model -> getResetLog 2. login_model -> addToResetLog 3. login_model -> getUserEmail 4. login_model -> resetPW
Register Member	Controller : Register Function : 1. Index () 2. Add_username () 3. Add_username_ pro () 4. Check_username () 5. Activate_account () 6. Send_activation_ code ()	Index () : 1. Register_model -> check_username_is_free 2. Register_model -> check_EmailIsFree 3. Register_model -> add_user 4. User_model -> add_custom_field Add_username_pro () : 1. Register_model -> check_username_is_fre e 2. Register_model -> checkEmailIsFree 3. Register_model -> update_username Check_username () : 1. Register_model -> check_username_is_fre e Activate_account () :

		1. User_model -> get_verify_user 2. User_model -> update_user Send_activation_code () : 1. User_model -> get_user_event 2. User_model -> add_user_event
Buat Tiket	Controller : Tickets Function : Add_pro ()	Add_pro () : 1. User_model -> get_user_by_username 2. Tickets_model -> add_ticket 3. Tickets_model -> add_custom_field_data 4. Tickets_model -> add_attached_files 5. Tickets_model -> add_history
Lihat List Tiket	Controller : Tickets Function : Ticket_page ()	Ticket_page () : 1. Tickets_model -> get_tickets 2. Tickets_model -> get_tickets_assigned 3. Tickets_model -> get_tickets_your
Search Tiket	Controller : Tickets Function : Datatables	Tickets_model -> get_tickets
Lihat Detail Tiket	Controller : Tickets Function : View ()	View () : 1. Tickets_model -> get_ticket 2. Tickets_model -> get_ticket_files

		3. Tickets_model -> get_ticket_replies 4. User_model -> get_custom_field_answers 5. Tickets_model -> Get_custom_fields_for_ticket 6. Tickets_model -> get_ticket_history_limit
Balas Tiket	Controller : Tickets Function : ticket_reply ()	Ticket_reply () : 1. Tickets_model -> add_ticket_reply 2. Tickets_model -> add_attached_files 3. Tickets_model -> update_ticket 4. User_model -> increment_field 5. User_model -> add_notification 6. Tickets_model -> add_history
Check Tiket (Guest)	Controller : Client Function : guest_login_pro ()	Guest_login_pro () : 1. Tickets_model -> get_guest_ticket 2. Tickets_model -> get_ticket_files 3. Tickets_model -> get_ticket_replies 4. User_model -> get_custom_field_answers 5. Tickets_model -> Get_custom_fields_for_ticket 6. Tickets_model -> add_history

Balas Tiket (Guest)	Controller : Tickets Function : ticket_reply ()	Ticket_reply () : 1. Tickets_model -> add_ticket_reply 2. Tickets_model -> add_attached_files 3. Tickets_model -> update_ticket 4. User_model -> increment_field 5. User_model -> add_notification
Hapus Tiket	Controller : Tickets Function : delete_ticket ()	Delete_ticket () : 1. Tickets_model -> delete_ticket
Hapus Tiket (Via Detail)	Controller : Tickets Function : delete_ticket ()	Delete_ticket () : 1. Tickets_model -> delete_ticket
Lihat History Tiket	Controller : Tickets Function : Ticket_history_page ()	Ticket_history_page () : 1. Tickets_model -> get_ticket_history
Lihat Data Client	Controller : Tickets Function : Ticket_page ()	Ticket_page () : 1. Tickets_model -> get_ticket 2. User_model -> get_custom_fields_answers
Assign Tiket	Controller : Tickets Function : Assign_user_pro ()	Assign_user_pro () : 1. User_model -> get_user_by_username 2. Tickets_model -> update_ticket 3. Tickets_model -> add_history

		4. User_model -> increment_field 5. User_model -> add_notification
Assign Myself	Controller : Tickets Function : Assign_user_pro ()	Assign_user_pro () : 1. User_model -> get_user_by_username 2. Tickets_model -> update_ticket 3. Tickets_model -> add_history 4. User_model -> increment_field 5. User_model -> add_notification
Ubah Status Tiket	Controller : Tickets Function : Change_status ()	Change_status () : 1. Tickets_model -> update_ticket 2. Tickets_model -> add_history

4.2 Desain Sistem

Setelah didapatkan kebutuhan fungsional dan nonfungsional, maka dilakukan desain sistem. Setelah kebutuhan fungsional telah dipastikan tidak berubah dan disepakati maka dimulai tahap desain sistem untuk menjadi dasaran pengembangan aplikasi mobile *service desk*. Desain sistem dilakukan dengan beberapa tahapan sebagai berikut :

4.2.1 Pembuatan Use Case Aplikasi

Desain use case dibuat dari hasil kebutuhan fungsional (tabel 4.1) yang diiriskan dengan fungsi yang terdapat pada aplikasi existing (tabel 4.4), beserta kode program yang dapat dilihat pada tabel 4.5. Use case bertujuan untuk menunjukkan interaksi antara aktor (user) dan aplikasi. Selain pembuatan *use case*, juga dilakukan pembuatan deskripsi (Use Case Description) yang berisikan detail behavior dari sebuah use case beserta pola

interaksinya, adapun *use case* beserta *use case description* aplikasi mobile service desk sebagai berikut :

4.2.1.1 Use Case User (Member dan Non Member)

Use case untuk aktor user (Member dan Non Member) dapat dilihat pada gambar 4.2. Adapun use case description untuk use case user (Member dan Non Member) dapat dilihat pada tabel 4.6 sampai 4.23.

Gambar 4.2 Use Case User

Tabel 4.6 Use Case Description Login

U-01 *Use Case Name :* Login
Primary actor : User (Member)
Brief Description : Use case ini digunakan user yang telah terdaftar sebagai member untuk masuk ke dalam aplikasi menggunakan username/email dan password
Pre Condition : 1. User telah memiliki akun dan terdaftar sebagai member
Basic Course : 1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password dan menekan tombol login 4. User masuk ke dalam menu home aplikasi sebagai member
Alternate Course : 4.1. Tampil peringatan untuk melengkapi data isian 4.2. Tampil peringatan username / password salah 4.3. Tampil dialog bahwa akun belum aktif 4.4. Tampil dialog terlalu banyak login

Tabel 4.7 Use Case Description Logout

U-02 *Use Case Name :* Logout
Primary actor : User (Member)
Brief Description : Use case ini digunakan user yang telah masuk kedalam aplikasi untuk keluar dari status login yang didapat setelah melakukan login
Pre Condition :

1. User telah melakukan login
Basic Course : 1. User menekan tombol menu 2. User menekan menu logout 3. User masuk ke dalam halaman login
Alternate Course : -

Tabel 4.8 Use Case Description Lihat Notifikasi

U-03 *Use Case Name :* Lihat Notifikasi
Primary actor : User (Member)
Brief Description : Use case ini digunakan user yang telah masuk kedalam aplikasi sebagai member untuk melihat notifikasi/pemberitahuan terkait aktivitas terakhir tiket yang terasosiasi dengan user
Pre Condition : 1. User telah melakukan login
Basic Course : 1. User menekan tombol menu 2. User menekan menu notifikasi 3. User masuk ke halaman notifikasi kemudian menekan salah satu entry notifikasi 4. User masuk ke halaman detail tiket terkait
Alternate Course : -

Tabel 4.9 Use Case Description Search Knowledge

U-04 *Use Case Name :* Search Knowledge
Primary actor : User (Member dan Non Member)
Brief Description : Use case ini digunakan user untuk mencari topik informasi terkait service desk beserta aplikasinya
Pre Condition : 1. -
Basic Course : 1. User menggeser menu utama 2. User menekan menu knowledge base 3. User menekan tombol search pada halaman knowledge base 4. User memasukkan kata kunci pencarian dan menekan tombol pencarian 5. Topik informasi sesuai dengan kata kunci yang dimasukkan muncul pada tampilan
Alternate Course : 5.1. Topik informasi tidak ditemukan

Tabel 4.10 Use Case Description Lihat Knowledge

U-05 *Use Case Name :* Lihat Knowledge
Primary actor : User (Member dan Non Member)
Brief Description : Use case ini digunakan user untuk melihat informasi terkait service desk beserta aplikasinya
Pre Condition : 1. -
Basic Course : 1. User menggeser menu utama

2. User menekan menu knowledge base 3. User menekan salah satu info pada list sesuai pilihan 4. User masuk ke halaman detail informasi
Alternate Course (Via Search Knowledge) : 1. <<<U-04>>>, user menekan salah satu info pada list sesuai pilihan 2. User masuk ke halaman detail informasi

Tabel 4.11 Use Case Description Lihat Info Buat Tiket

U-06 *Use Case Name :* Lihat Info Buat Tiket
Primary actor : User (Member dan Non Member)
Brief Description : Use case ini digunakan user untuk melihat informasi bagaimana cara membuat tiket pada aplikasi
Pre Condition : 1. -
Basic Course : 1. User menggeser menu utama 2. User menekan icon (i) pada menu knowledge base 3. PDF info buat tiket terdownload
Alternate Course : -

Tabel 4.12 Use Case Description Lihat Profile

U-07 *Use Case Name :* Lihat Profile
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk melihat profile terkait informasi diri yang telah terdaftar

Pre Condition : 1. User telah melakukan login
Basic Course : 1. User menekan tombol menu 2. User menekan menu profile 3. User masuk pada halaman profile
Alternate Course : -

Tabel 4.13 Use Case Description Edit Profile

U-08 *Use Case Name :* Edit Profile
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk melakukan modifikasi terkait informasi diri yang telah terdaftar
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<U-07>>>, user menekan tombol edit profile 2. User menekan tombol konfirmasi 3. User melengkapi kembali data informasi terkait 4. User menekan tombol selesai 5. User menekan tombol konfirmasi 6. Dialog sukses edit profile ditampilkan
Alternate Course : 4.1. Peringatan lengkapi isian ditampilkan

Tabel 4.14 Use Case Description Ubah Password

U-09 *Use Case Name :* Ubah Password
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk mengganti password yang digunakan untuk melakukan login
Pre Condition : 1. User telah memiliki akun dan terdaftar sebagai member
Basic Course : 1. <<<U-07>>>, user menekan tombol change password 2. User memasukkan password lama dan isian konfirmasi password baru 3. User menekan tombol lanjutkan untuk mengubah password 4. Dialog sukses ganti password ditampilkan
Alternate Course : -

Tabel 4.15 Use Case Description Lupa Password

U-10 *Use Case Name :* Lupa Password
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk melakukan reset password
Pre Condition : 1. -
Basic Course : 1. User menekan tombol menu 2. User menekan menu login

3. User menekan tombol lupa password, dan redirect ke browser 4. User memasukkan informasi lupa password
Alternate Course : -

Tabel 4.16 Use Case Description Register Member

U-11 *Use Case Name :* Register Member
Primary actor : User (Non Member)
Brief Description : Use case ini digunakan user untuk melakukan registrasi akun
Pre Condition : 1. -
Basic Course : 1. User menekan tombol menu 2. User menekan menu login 3. User menekan tombol belum memiliki akun, dan redirect ke browser 4. User memasukkan informasi registrasi
Alternate Course : -

Tabel 4.17 Use Case Description Membuat Tiket

U-12 *Use Case Name :* Buat Tiket
Primary actor : User (Member dan Non Member)
Brief Description : Use case ini digunakan user untuk membuat tiket baru

Pre Condition : 1. -
Basic Course : 1. User menekan tombol create new ticket 2. User menekan konfirmasi buat tiket 3. User mengisi form yang telah disediakan menekan tombol next 4. User menekan tombol selesai 5. User menekan tombol lanjutkan untuk membuat tiket 6. Dialog berhasil dibuat ditampilkan
Alternate Course : 4.1. Peringatan llengkapi data isian ditampilkan

Tabel 4.18 Use Case Description Lihat List Tiket

U-13 *Use Case Name :* Lihat List Tiket
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk melihat list tiket yang terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. User menekan tombol menu 2. User menekan menu tiket 3. List tiket terasosiasi dengan user muncul pada tampilan
Alternate Course : -

Tabel 4.19 Use Case Description Search Tiket

U-14 *Use Case Name :* Search Tiket
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk mencari tiket berdasarkan kata pencarian yang relevan
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<U-13>>>, user menekan tombol search 2. User memasukkan kata kunci pencarian, klik search 3. Tiket relevan dengan kata kunci ditampilkan
Alternate Course : 3.1. Pemberitahuan tiket tidak ditemukan muncul pada tampilan

Tabel 4.20 Use Case Description Lihat Detail Tiket

U-15 *Use Case Name :* Lihat Detail Tiket
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk melihat informasi detail dari list tiket yang telah ditampilkan
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<U-13>>>, user menekan salah satu tiket dalam list 2. User masuk pada halaman detail tiket
Alternate Course : 1.1 <<<U-14>>>, user menekan salah satu tiket dalam list

Tabel 4.21 Use Case Description Balas Tiket

U-16 *Use Case Name :* Balas Tiket
Primary actor : User (Member)
Brief Description : Use case ini digunakan user untuk membalas tiket yang terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<U-15>>>, user menekan tombol tambah balasan 2. User memasukkan seluruh isian balasan 3. User menekan tombol konfirmasi balas tiket 4. Informasi balasan berhasil dibuat ditampilkan
Alternate Course : -

Tabel 4.22 Use Case Description Lihat Detail Tiket (Guest)

U-17 *Use Case Name :* CheckTiket (Guest)
Primary actor : User (Non Member)
Brief Description : Use case ini digunakan user guest untuk melihat informasi detail dari list tiket yang telah ditampilkan
Pre Condition : 1. -
Basic Course : 1. User menekan menu cek tiket 2. User memasukkan data isian cek tiket 3. Pemberitahuan tiket ditemukan ditampilkan 4. User masuk pada halaman detail tiket terkait

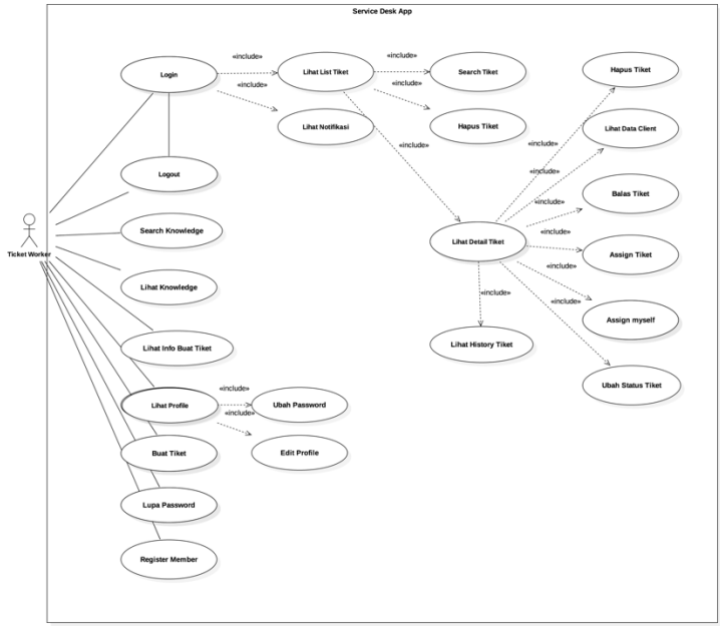
Alternate Course : 3.1. Pemberitahuan tiket tidak ditemukan ditampilkan

Tabel 4.23 Use Case Description Balas Tiket (Guest)

U-18 *Use Case Name :* Balas Tiket (Guest)
Primary actor : User (Non Member)
Brief Description : Use case ini digunakan user guest untuk membalas tiket yang terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<U-17>>>, user menekan tombol tambah balasan 2. User memasukkan seluruh isian balasan 3. User menekan tombol konfirmasi balas tiket 4. Informasi balasan berhasil dibuat ditampilkan
Alternate Course : -

4.2.1.2 Use Case Ticket Worker

Use case untuk aktor ticket worker dapat dilihat pada gambar 4.3 sebagai berikut.



Gambar 4.3 Use Case Ticket Worker

Adapun use case description untuk use case ticket worker dapat dilihat pada tabel 4.24 sampai 4.46.

Tabel 4.24 Use Case Description Login

T-01 *Use Case Name :* Login
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user yang telah terdaftar sebagai ticket worker untuk masuk ke dalam aplikasi menggunakan username/email dan password
Pre Condition :

1. User telah memiliki akun dan terdaftar sebagai member
Basic Course : 1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password dan menekan tombol login 4. User masuk ke dalam menu home aplikasi sebagai member
Alternate Course : 4.1. Tampil peringatan untuk melengkapi data isian 4.2. Tampil peringatan username / password salah 4.3. Tampil dialog bahwa akun belum aktif 4.4. Tampil dialog terlalu banyak login

Tabel 4.25 Use Case Description Logout

T-02 *Use Case Name :* Logout
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user yang telah masuk kedalam aplikasi untuk keluar dari status login yang didapat setelah melakukan login
Pre Condition : 1. User telah melakukan login
Basic Course : 1. User menekan tombol menu 2. User menekan menu logout 3. User masuk ke dalam halaman login
Alternate Course : -

Tabel 4.26 Use Case Description Lihat Notifikasi

T-03	*Use Case Name :* Lihat Notifikasi
Primary actor : Ticket Worker	
Brief Description : Use case ini digunakan user yang telah masuk kedalam aplikasi sebagai ticket worker untuk melihat notifikasi/pemberitahuan terkait aktivitas terakhir tiket yang terasosiasi dengan user	
Pre Condition : 1. User telah melakukan login	
Basic Course : 1. User menekan menu notifikasi 2. User masuk ke halaman notifikasi yang berisikan list aktivitas terkait tiket terasosiasi 3. User menekan list notifikasi yang dipilih 4. User masuk ke halaman detail tiket terkait	
Alternate Course : -	

Tabel 4.27 Use Case Description Search Knowledge

T-04	*Use Case Name :* Search Knowledge
Primary actor : Ticket Worker	
Brief Description : Use case ini digunakan user untuk mencari topik informasi terkait service desk beserta aplikasinya	
Pre Condition : 1. -	
Basic Course : 1. User menggeser menu utama	

2. User menekan menu knowledge base 3. User menekan tombol search pada halaman knowledge base 4. User memasukkan kata kunci pencarian dan menekan tombol pencarian 5. Topik informasi sesuai dengan kata kunci yang dimasukkan muncul pada tampilan
Alternate Course : 5.1. Topik informasi tidak ditemukan

Tabel 4.28 Use Case Description Lihat Knowledge

T-05 *Use Case Name* : Lihat Knowledge
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melihat informasi terkait service desk beserta aplikasinya
Pre Condition : 1. -
Basic Course : 1. User menggeser menu utama 2. User menekan menu knowledge base 3. User menekan salah satu info pada list sesuai pilihan 4. User masuk ke halaman detail informasi
Alternate Course (Via Search Knowledge) : 3.1. <<<T-04>>>, user menekan salah satu info pada list sesuai pilihan 3.2. User masuk ke halaman detail informasi

Tabel 4.29 Use Case Description Lihat Info Buat Tiket

T-06 *Use Case Name :* Lihat Info Buat Tiket
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melihat informasi bagaimana cara membuat tiket pada aplikasi
Pre Condition : 1. -
Basic Course : 1. User menggeser menu utama 2. User menekan icon (i) pada menu knowledge base 3. PDF info buat tiket terdownload
Alternate Course : -

Tabel 4.30 Use Case Description Lihat Profile

T-07 *Use Case Name :* Lihat Profile
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melihat profile terkait informasi diri yang telah terdaftar
Pre Condition : 1. User telah melakukan login
Basic Course : 1. User menekan tombol menu 2. User menekan menu profile 3. User masuk pada halaman profile

Alternate Course : -

Tabel 4.31 Use Case Description Edit Profile

T-08 *Use Case Name :* Edit Profile
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melakukan modifikasi terkait informasi diri yang telah terdaftar
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-07>>>, user menekan tombol edit profile 2. User menekan tombol konfirmasi 3. User melengkapi kembali data informasi terkait 4. User menekan tombol selesai 5. User menekan tombol konfirmasi 6. Dialog sukses edit profile ditampilkan
Alternate Course : 4.2. Peringatan lengkapi isian ditampilkan

Tabel 4.32 Use Case Description Ubah Password

T-09 *Use Case Name :* Ubah Password
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk mengganti password yang digunakan untuk melakukan login
Pre Condition :

1. User telah memiliki akun dan terdaftar sebagai member
Basic Course : 1. <<<T-07>>>, user menekan tombol change password 2. User memasukkan password lama dan isian konfirmasi password baru 3. User menekan tombol lanjutkan untuk mengubah password 4. Dialog sukses ganti password ditampilkan
Alternate Course : -

Tabel 4.33 Use Case Description Lupa Password

T-10 *Use Case Name :* Lupa Password
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melakukan reset password
Pre Condition : 1. -
Basic Course : 1. User menekan tombol menu 2. User menekan menu login 3. User menekan tombol lupa password, dan redirect ke browser 4. User memasukkan informasi lupa password
Alternate Course : -

Tabel 4.34 Use Case Description Register Member

T-11	*Use Case Name :* Register Member
Primary actor : Ticket Worker	
Brief Description : Use case ini digunakan user untuk melakukan registrasi akun	
Pre Condition : 1. -	
Basic Course : 1. User menekan tombol menu 2. User menekan menu login 3. User menekan tombol belum memiliki akun, dan redirect ke browser 4. User memasukkan informasi registrasi	
Alternate Course : -	

Tabel 4.35 Use Case Description Membuat Tiket

T-12	*Use Case Name :* Buat Tiket
Primary actor : Ticket Worker	
Brief Description : Use case ini digunakan user untuk membuat tiket baru	
Pre Condition : 1. -	
Basic Course : 1. User menekan tombol create new ticket 2. User menekan konfirmasi buat tiket 3. User mengisi form yang telah disediakan menekan tombol next	

4. User menekan tombol selesai 5. User menekan tombol lanjutkan untuk membuat tiket 6. Dialog berhasil dibuat ditampilkan
Alternate Course : 4.1. Peringatan llengkapi data isian ditampilkan

Tabel 4.36 Use Case Description Lihat List Tiket

T-13 *Use Case Name :* Lihat List Tiket
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melihat list tiket yang terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. User menekan tombol menu 2. User menekan menu tiket 3. List tiket terasosiasi dengan user muncul pada tampilan
Alternate Course : -

Tabel 4.37 Use Case Description Search Tiket

T-14 *Use Case Name :* Search Tiket
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk mencari tiket berdasarkan kata pencarian yang relevan
Pre Condition :

1. User telah melakukan login
Basic Course : 1. <<<T-13>>>, user menekan tombol search 2. User memasukkan kata kunci pencarian, klik search 3. Tiket relevan dengan kata kunci ditampilkan
Alternate Course : 3.1. Pemberitahuan tiket tidak ditemukan muncul pada tampilan

Tabel 4.38 Use Case Description Lihat Detail Tiket

T-15 *Use Case Name :* Lihat Detail Tiket
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melihat informasi detail dari list tiket yang telah ditampilkan
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-13>>>, user menekan salah satu tiket dalam list 2. User masuk pada halaman detail tiket
Alternate Course : 1.1 <<<T-14>>>, user menekan salah satu tiket dalam list

Tabel 4.39 Use Case Description Balas Tiket

T-16 *Use Case Name :* Balas Tiket
Primary actor : Ticket Worker

Brief Description : Use case ini digunakan user untuk membalas tiket yang terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-15>>>, user menekan tombol tambah balasan 2. User memasukkan seluruh isian balasan 3. User menekan tombol konfirmasi balas tiket 4. Informasi balasan berhasil dibuat ditampilkan
Alternate Course : -

Tabel 4.40 Use Case Description Hapus Tiket

T-17 *Use Case Name :* Hapus Tiket
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk menghapus tiket yang terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-13>>>, user menekan tombol hapus tiket 2. User menekan tombol lanjutkan konfirmasi hapus tiket 3. Informasi tiket berhasil dihapus ditampilkan
Alternate Course : -

Tabel 4.41 Use Case Description Hapus Tiket (Via Detail)

T-18	*Use Case Name :* Hapus Tiket (Via Detail)
Primary actor : Ticket Worker	
Brief Description : Use case ini digunakan user untuk menghapus tiket yang terasosiasi	
Pre Condition : 1. User telah melakukan login	
Basic Course : 1. <<<T-15>>>, user menekan tombol hapus tiket 2. User menekan tombol lanjutkan konfirmasi hapus tiket 3. Informasi tiket berhasil dihapus ditampilkan	
Alternate Course : -	

Tabel 4.42 Use Case Description Lihat History Tiket

T-19	*Use Case Name :* Lihat History Tiket
Primary actor : Ticket Worker	
Brief Description : Use case ini digunakan user untuk melihat history/rekam jejak transaksi tiket	
Pre Condition : 1. User telah melakukan login	
Basic Course : 1. <<<T-15>>>, user menekan icon tombol lihat history tiket 2. History tiket ditampilkan	

Alternate Course : -

Tabel 4.43 Use Case Description Lihat Data Client

T-20 *Use Case Name :* Lihat Data Client
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melihat informasi client pada tiket terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-15>>>, user menekan icon tombol lihat informasi client 2. Data client ditampilkan
Alternate Course : -

Tabel 4.44 Use Case Description Assign Tiket

T-21 *Use Case Name :* Assign Tiket
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melakukan assignment tiket ke username lain pada aplikasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-15>>>, user menekan tombol assign tiket

2. User memasukkan kata kunci pencarian username 3. Username ditemukan, user memilih username yang dituju 4. User menekan tombol lanjutkan konfirmasi assign tiket 5. Informasi assign tiket berhasil dilakukan ditampilkan
Alternate Course : 3.1. Username tidak ditemukan

Tabel 4.45 Use Case Description Assign Myself

T-22 *Use Case Name :* Assign Myself
Primary actor : Ticket Worker
Brief Description : Use case ini digunakan user untuk melakukan assignment tiket ke user tersebut
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-15>>>, user menekan tombol assign tiket 2. User menekan tombol assign myself 3. User menekan tombol lanjutkan konfirmasi assign tiket 4. Informasi assign tiket berhasil dilakukan ditampilkan
Alternate Course : -

Tabel 4.46 Use Case Description Ubah Status Tiket

T-23 *Use Case Name :* Ubah Status Tiket
Primary actor : Ticket Worker

Brief Description : Use case ini digunakan user untuk mengubah status tiket terasosiasi
Pre Condition : 1. User telah melakukan login
Basic Course : 1. <<<T-15>>>, user menekan tombol change status tiket 2. User memilih status tiket yang diinginkan 3. User menekan tombol lanjutkan konfirmasi ubah status tiket 4. Informasi ubah status tiket berhasil dilakukan ditampilkan
Alternate Course : -

4.2.2 Pembuatan Desain Kebutuhan dan Pemetaan Service

Peta aplikasi existing yang didapat dari tahapan observasi dan use case yang telah dibuat menjadi bahan pada tahapan ini untuk kemudian dibuat pemetaan baru terkait service yang akan digunakan aplikasi mobile *service desk*. Pemetaan dilakukan dengan mengelompokkan query berdasarkan use case, lalu dibuat nama service, objek, dan entitas (parameter yang dikirim) untuk tiap servicenya. Sebelum pemetaan service dilakukan, penulis menentukan template JSON yang digunakan saat melakukan request. Template request dapat dilihat pada gambar 4.4 untuk request tanpa otentifikasi dan 4.5 untuk request dengan menggunakan otentifikasi.

Gambar 4.4 Format Request Web Service (NoAuth)

```
{
  "operation":"nameOfOperation",
  "object":{
                    “entity1”:”value”,
                    “entity2”:”value”,
                    “entity3”:”value”
  }
}
```

Gambar 4.5 Format Request Web Service (with Auth)

```
{
  "operation":"nameOfOperation",
  “email”:”emailValue”,
  “password”:”passwordValue”,
  "object":{
                    “entity1”:”value”,
                    “entity2”:”value”,
                    “entity3”:”value”
  }
}
```

Sedangkan tabel pemetaan service dapat dilihat pada tabel 4.47 dibawah.

Tabel 4.47 Desain Pemetaan Service Aplikasi

Use Case	Model (Query Function)	Services
U-01, T-01	Pro () : 1. User_model -> check_block_ip 2. Login_model -> get_login_attempts	Operation : login Object : user Entity : email, password Operation : usrInfo

	3. Logsin_model -> getUserByEmail 4. Login_model -> getUserByUsername 5. Library -> update_online_timestamp Login_protect () : 1. Login_model -> get _login_attempts 2. Login_model -> update_login_attempts 3. Login_model -> add_login_attempts	Object : user Entity : id Operation : usrGroup Object : user Entity : id Operation : updateStampAndT Object : user Entity : id
U-02, T-02	-	-
U-03, T-03	Notifications_page () : 1. User_model -> get_notifications_unread 2. User_model -> get_notifications 3. User_model -> update_notification	Operation : getNotificationsUnread Object : notification Entity : userid Operation : getNotifications Object : notification Entity : userid Operation : updateNotifications Object : notification Entity : id Operation : getNotiCountUser Object : user Entity : id Operation : updateNotiCountUser Object : user

		Entity : id, noti_count
U-04, T-04	1. Knowledge_model -> get_articles	Operation : getKnowledgeByCatOrArt Object : knowledge Entity : keyword
U-05, T-05	Article_page () : 1. Knowledge_model -> get_articles Categories () : 1. Knowledge_model -> get_categories Cat_page () : 1. Knowledge_model -> get_categories_dt	Operation : getKnowledgeCat Object : - Entity : - Operation : getKnowledgeArtic Object : knowledge Entity : catid Operation : getKnowledgeArticDetail Object : knowledge Entity : id
U-06, T-06	-	Redirect PDF
U-07, T-07	Index () : 1. User_model -> get_user_by_username 2. User_model -> get_user_groups 3. User_model -> get_custom_field_answer	Operation : Operation : getUsrById Object : user Entity : id Operation : usrInfo Object : user Entity : id Operation : usrGroup Object : user Entity : id
U-08, T-08	Pro () :	Operation : checkEmailIsFree

	1. Register_model -> checkEmailIsFree 2. User_model -> update_user 3. User_model -> update_custom_field	Object : user Entity : email Operation : updateInfoUser Object : user Entity : id, email, firstname, lastname, aboutme, status, unit, telepon, emailnoti
U-09, T-09	Resetpw_pro () : 1. login_model -> updatePassword	Operation : changePass Object : user Entity : id, old_password, new_password
U-10, T-10	Forgotpw_pro () : 1. login_model -> getResetLog 2. login_model -> addToResetLog 3. login_model -> getUserEmail 4. login_model -> resetPW	Redirect Service Desk Website.
U-11, T-11	Index () : 1. Register_model -> check_username_is_free 2. Register_model -> check_EmailIsFree 3. Register_model -> add_user 4. User_model -> add_custom_field Add_username_pro () : 1. Register_model -> check_username_is_free 2. Register_model -> checkEmailIsFree	Redirect Service Desk Website.

	3. Register_model -> update_username Check_username () : 1. Register_model -> check_username_is_free Activate_account () : 1. User_model -> get_verify_user 2. User_model -> update_user Send_activation_code () : 1. User_model -> get_user_event 2. User_model -> add_user_event	
U-12, T-12	Add_pro () : 1. User_model -> get_user_by_username 2. Tickets_model -> add_ticket 3. Tickets_model -> add_custom_field_data 4. Tickets_model -> add_attached_files 5. Tickets_model -> add_history	Operation : getCategoryID Object : ticket Entity : name Operation : addTicket Object : ticket Entity : title, body, userid, assignedid, categoryid, status, priority, notes, guest_email, guest_password Operation : addCustomFieldTicket Object : ticket Entity : ticketed, value Operation : addHistoryAddTicket

		Object : tickethistory Entity : ticketid, userid, message Operation : getGuestPassword Object : ticket Entity : id Operation : addFilesTicket Object : ticketfiles Entity : ticketid, upload_file_name, file_type, extension, file_size, replyid, userid Operation : uploadFile() Object : - Entity : -
U-13, T-13	Ticket_page () : 1. Tickets_model -> get_tickets 2. Tickets_model -> get_tickets_assigned 3. Tickets_model -> get_tickets_your	Operation : getTicketByYourAll Object : ticket Entity : userid
U-14, T-14	1. Tickets_model -> get_tickets	Operation : getTicketByKeyword AllMember Object : ticket Entity : keyword, userid
U-15, T-15	View () :	Operation : getTicketDetails

	1. Tickets_model -> get_ticket 2. Tickets_model -> get_ticket_files 3. Tickets_model -> get_ticket_replies 4. User_model -> get_custom_field_answers 5. Tickets_model -> Get_custom_fields_for_ticket 6. Tickets_model -> get_ticket_history_limit	Object : ticket Entity : id Operation : getTicketFiles Object : ticket Entity : id Operation : getTicketReplies Object : ticket Entity : id Operation : getReplyFiles Object : reply Entity : replyid Operation : getTicketCustomInfo Object : ticket Entity : id
U-16, T-16	Ticket_reply () : 1. Tickets_model -> add_ticket_reply 2. Tickets_model -> add_attached_files 3. Tickets_model -> update_ticket 4. User_model -> increment_field 5. User_model -> add_notification 6. Tickets_model -> add_history	Operation : addTicketReply Object : reply Entity : ticketid, userid, body, files Operation : updateTicketReplyStamp Object : ticket Entity : ticketid, userid Operation : addHistoryGlobalReply Object : tickethistory

		Entity : ticketid, userid, message Operation : addFilesTicket Object : ticketfiles Entity : ticketid, upload_file_name, file_type, extension, file_size, replyid, userid Operation : uploadFile() Object : - Entity : - Operation : addFilesTicket Object : ticketfiles Entity : ticketid, upload_file_name, file_type, extension, file_size, replyid, userid Operation : addNotiGlobal Object : notification Entity : userid, url, fromid, message Operation : getNotiCountUser Object : user Entity : id Operation : updateNotiCountUser Object : user Entity : id, noti_count

U-17	Guest_login_pro () : 1. Tickets_model -> get_guest_ticket 2. Tickets_model -> get_ticket_files 3. Tickets_model -> get_ticket_replies 4. User_model -> get_custom_field_answers 5. Tickets_model -> Get_custom_fields_for_ticket 6. Tickets_model -> add_history	Operation : getTicketGuestDetails Object : ticket Entity : email, password Operation : getTicketFiles Object : ticket Entity : id Operation : getTicketReplies Object : ticket Entity : id Operation : getReplyFiles Object : reply Entity : replyid Operation : getTicketCustomInfo Object : ticket Entity : id Operation : addHistoryLoginTicket Object : ticket Entity : ticketid
U-18	Ticket_reply () : 1. Tickets_model -> add_ticket_reply 2. Tickets_model -> add_attached_files 3. Tickets_model -> update_ticket	*All U-17 Services* + Operation : addTicketReply Object : reply Entity : ticketid, userid, body, files

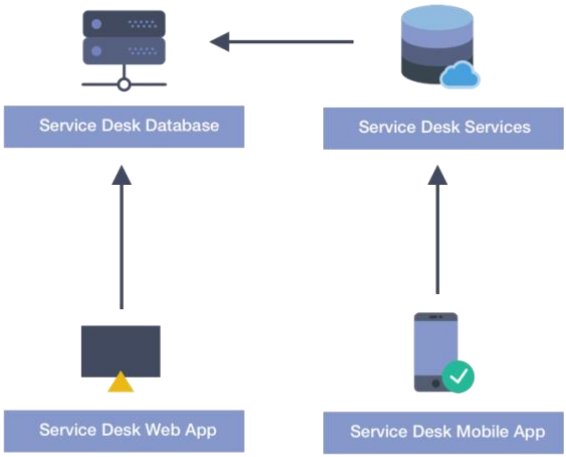
	4. User_model -> increment_field 5. User_model -> add_notification	Operation : updateTicketReplyStamp Object : ticket Entity : ticketid, userid Operation : addHistoryGlobalReply Object : tickethistory Entity : ticketid, userid, message Operation : addFilesTicket Object : ticketfiles Entity : ticketid, upload_file_name, file_type, extension, file_size, replyid, userid Operation : uploadFile() Object : - Entity : - Operation : addFilesTicket Object : ticketfiles Entity : ticketid, upload_file_name, file_type, extension, file_size, replyid, userid Operation : addNotiGlobal Object : notification

		Entity : userid, url, fromid, message Operation : getNotiCountUser Object : user Entity : id Operation : updateNotiCountUser Object : user Entity : id, noti_count
T-17	Delete_ticket () : 1. Tickets_model -> delete_ticket	Operation : deleteTicketSD Object : ticket Entity : id
T-18	Delete_ticket () : 1. Tickets_model -> delete_ticket	Operation : deleteTicketSD Object : ticket Entity : id
T-19	Ticket_history_page () : 1. Tickets_model -> get_ticket_history	Operation : getTicketHistory Object : ticket Entity : ticketid
T-20	Ticket_page () : 1. Tickets_model -> get_ticket 2. User_model -> get_custom_fields_answers	Operation : getTicketDetails Object : ticket Entity : id Operation : usrInfo Object : user Entity : id
T-21	Assign_user_pro () : 1. User_model -> get_user_by_username 2. Tickets_model -> update_ticket	Operation : addHistoryAssignTicket Object : tickethistory

	3. Tickets_model -> add_history 4. User_model -> increment_field User_model -> add_notification	Entity : ticketid, userid Operation : addNotiAssignTicket Object : notification Entity : userid, url, fromid Operation : getNotiCountUser Object : user Entity : id Operation : updateNotifCountUser Object : user Entity : id, noti_count
T-22	Assign_user_pro () : 1. User_model -> get_user_by_username 2. Tickets_model -> update_ticket 3. Tickets_model -> add_history 4. User_model -> increment_field 5. User_model -> add_notification	Operation : addHistoryAssignTicket Object : tickethistory Entity : ticketid, userid Operation : addNotiAssignTicket Object : notification Entity : userid, url, fromid Operation : getNotiCountUser Object : user Entity : id Operation :

		updateNotifCountUser Object : user Entity : id, noti_count
T-23	Change_status () : 1. Tickets_model -> update_ticket 2. Tickets_model -> add_history	Operation : updateStatusTicket Object : ticket Entity : id, status Operation : addHistoryGlobal Object : tickethistory Entity : ticketid, userid, message

Dengan pembuatan web service sesuai desain yang telah dirancang, maka arsitektur aplikasi service desk akan mengalami perubahan sebagaimana dapat dilihat pada gambar 4.6.

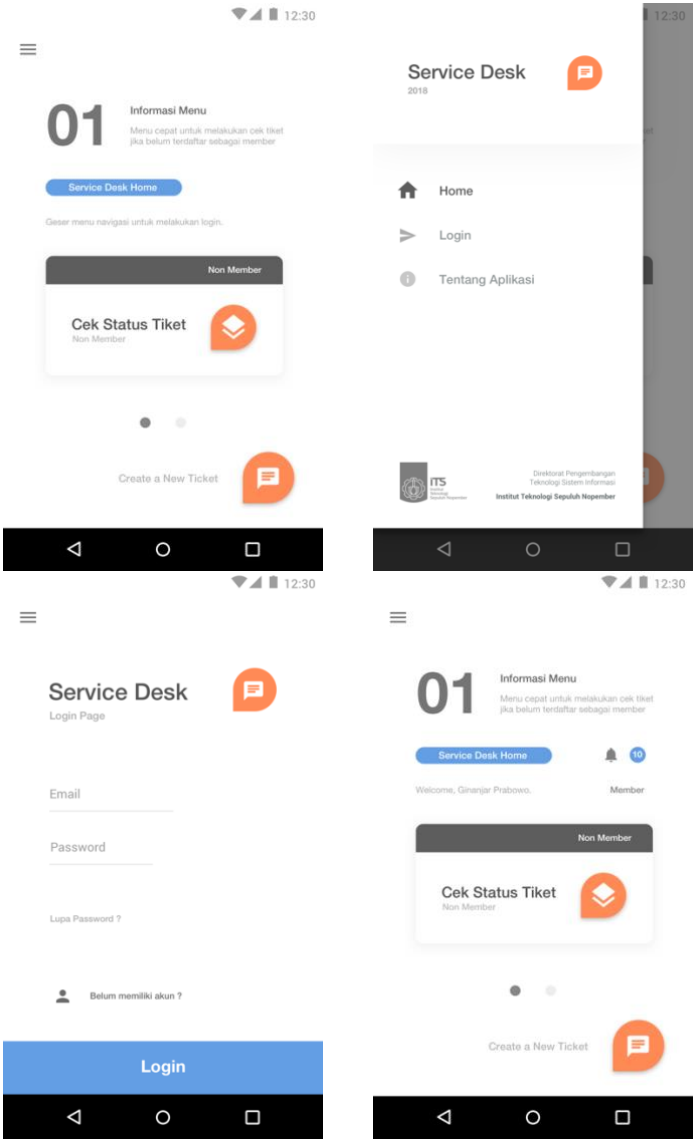


Gambar 4.6 Diagram Arsitektur Aplikasi Service Desk

4.2.3 Pembuatan Desain Aplikasi

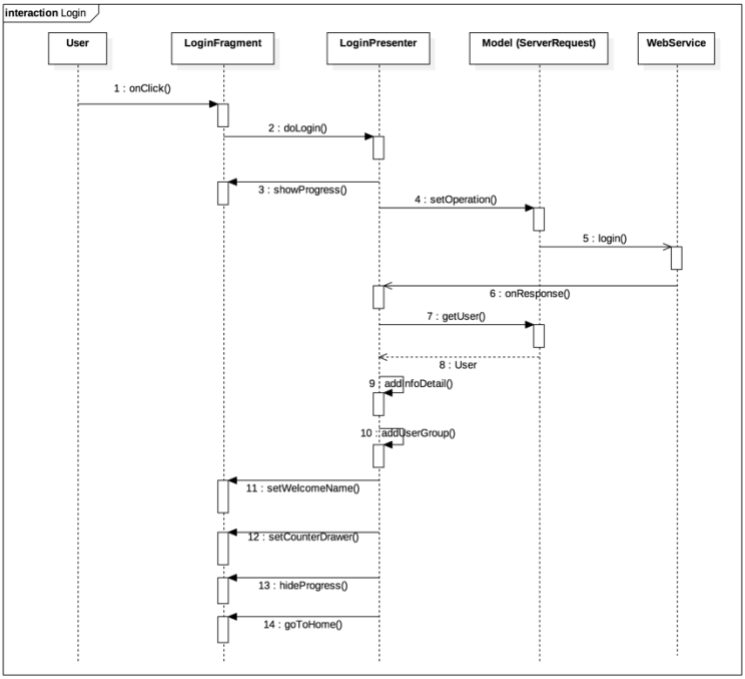
Dari use case, dan peta service yang telah dibuat, maka langkah selanjutnya adalah pembuatan desain user interface aplikasi beserta sequence diagram sebagai berikut.

4.2.3.1 Use Case U-01 dan T-01 (Login)



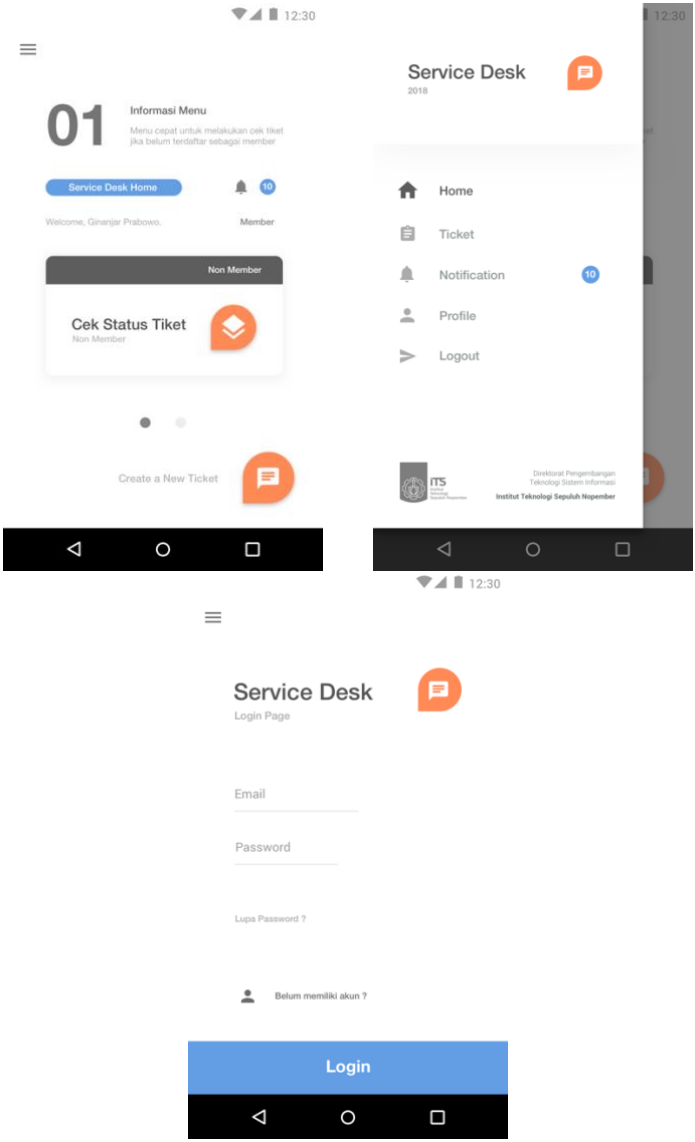
Gambar 4.7 User Interface Login

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.8.



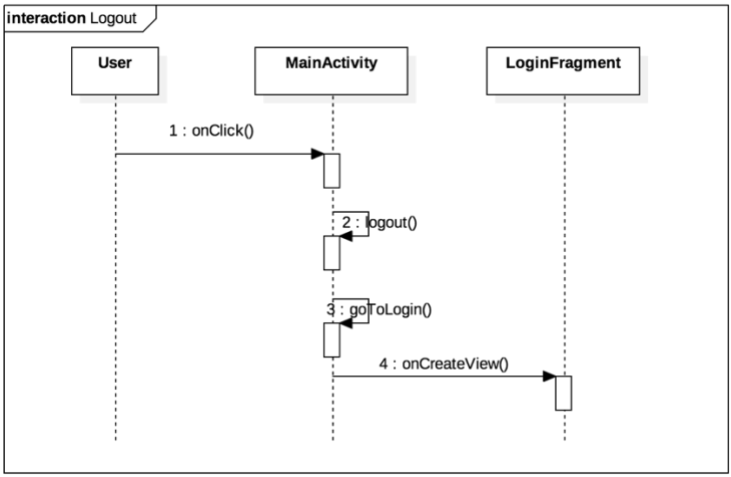
Gambar 4.8 Sequence Diagram Use Case Login

4.2.3.2 Use Case U-02 dan T-02 (Logout)



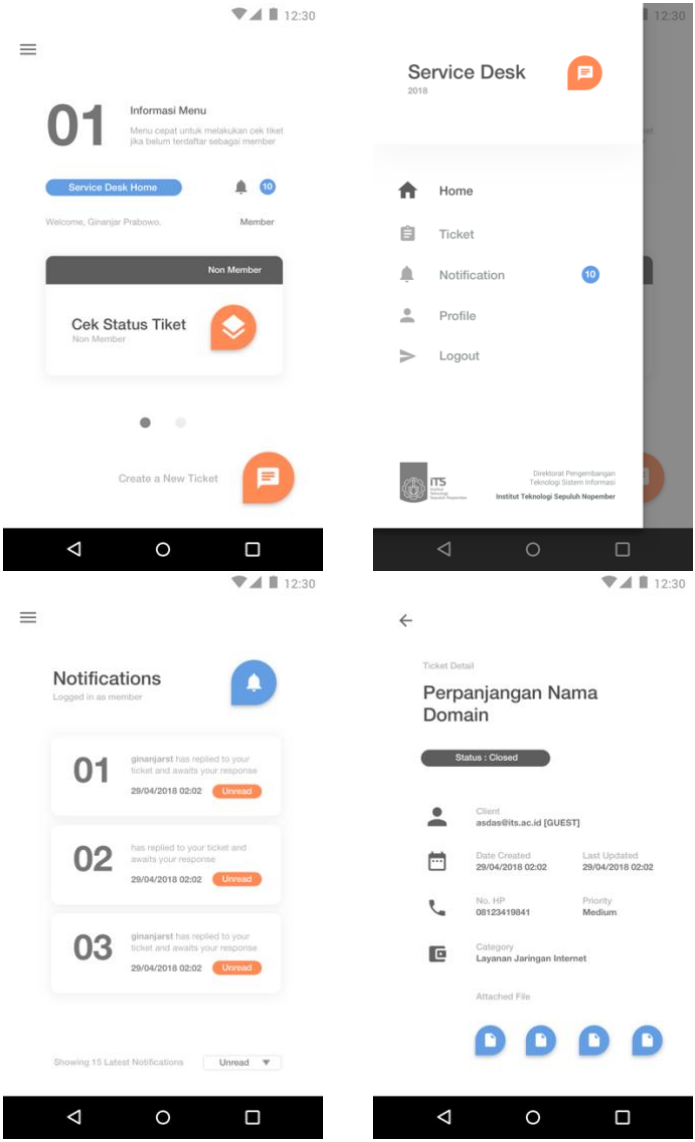
Gambar 4.9 User Interface Logout

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.10.



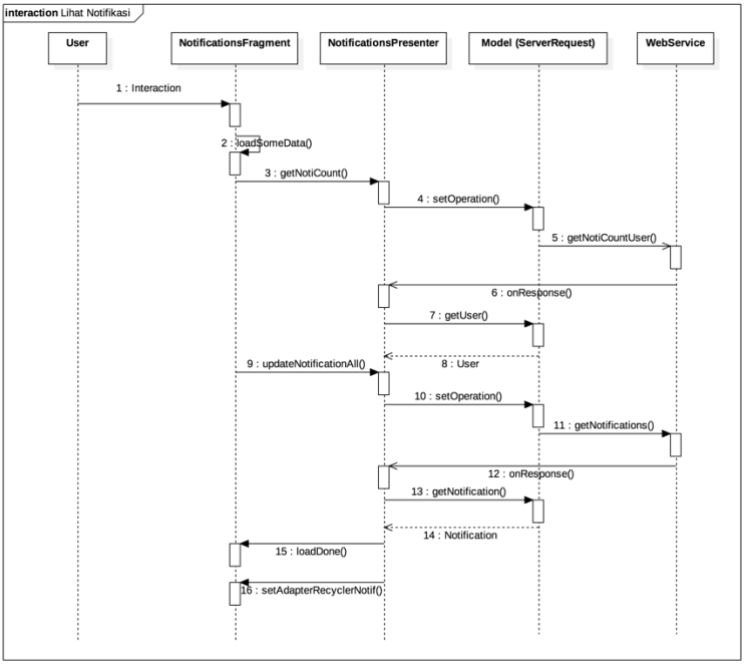
Gambar 4.10 Sequence Diagram Logout

4.2.3.3 Use Case U-03 dan T-03 (Lihat Notifikasi)



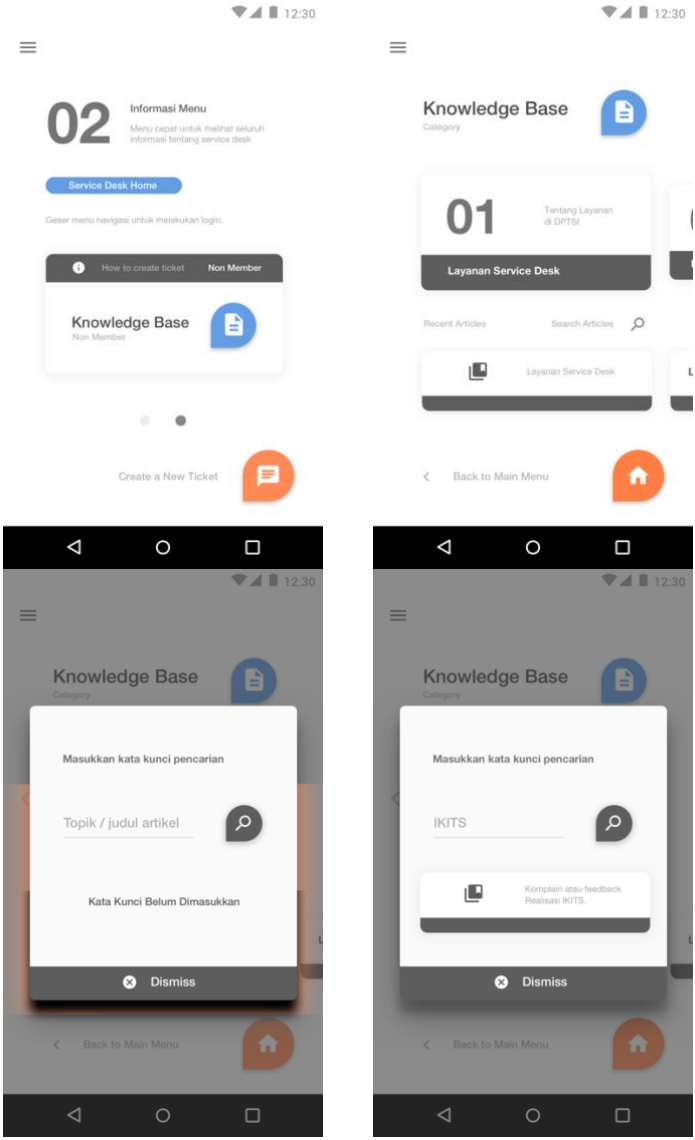
Gambar 4.11 User Interface Lihat Notifikasi

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.12.



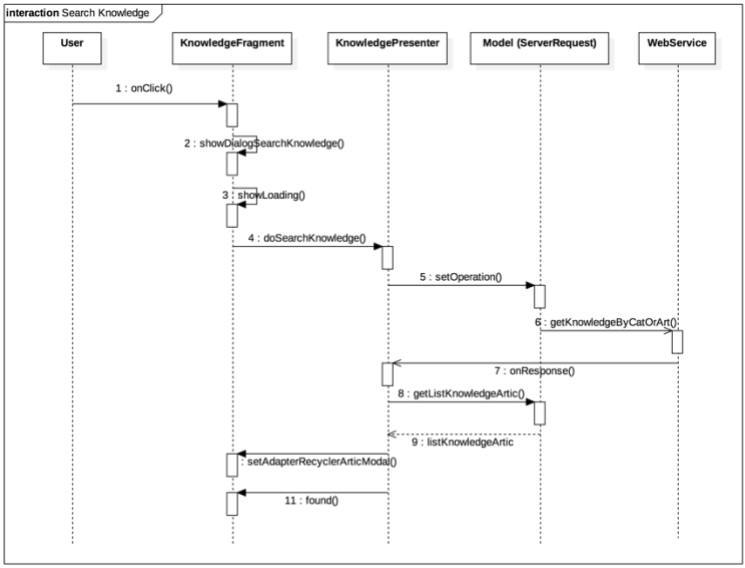
Gambar 4.12 Sequence Diagram Lihat Notifikasi

4.2.3.4 Use Case U-04 dan T-04 (Search Knowledge)



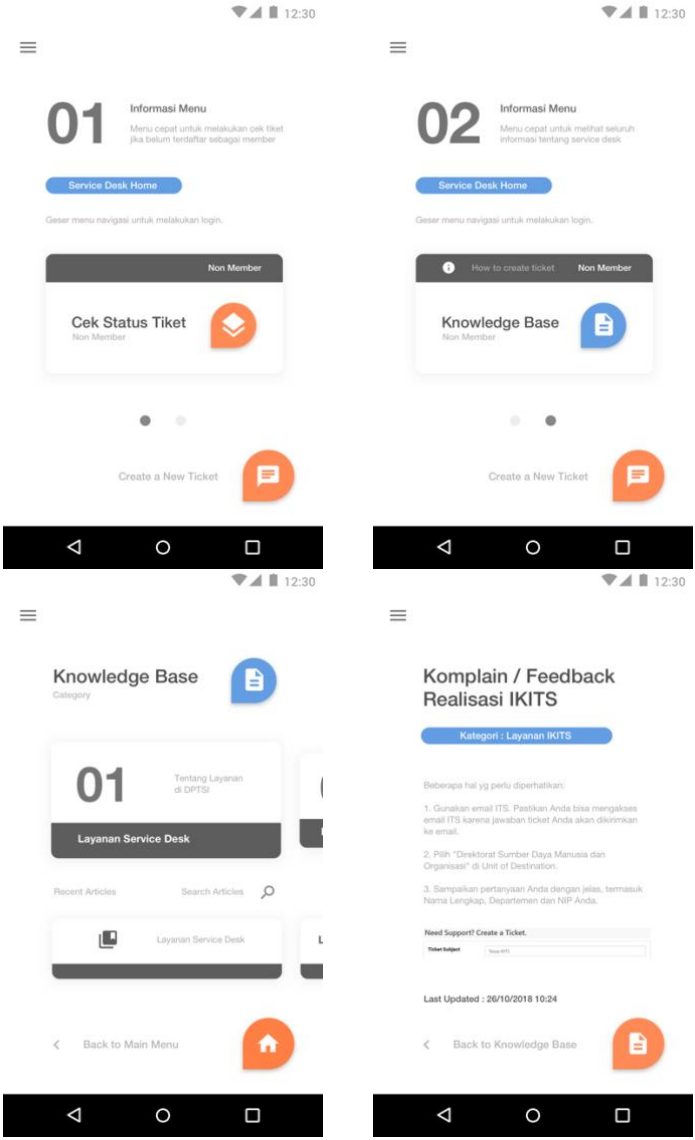
Gambar 4.13 User Interface Search Knowledge

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.14.



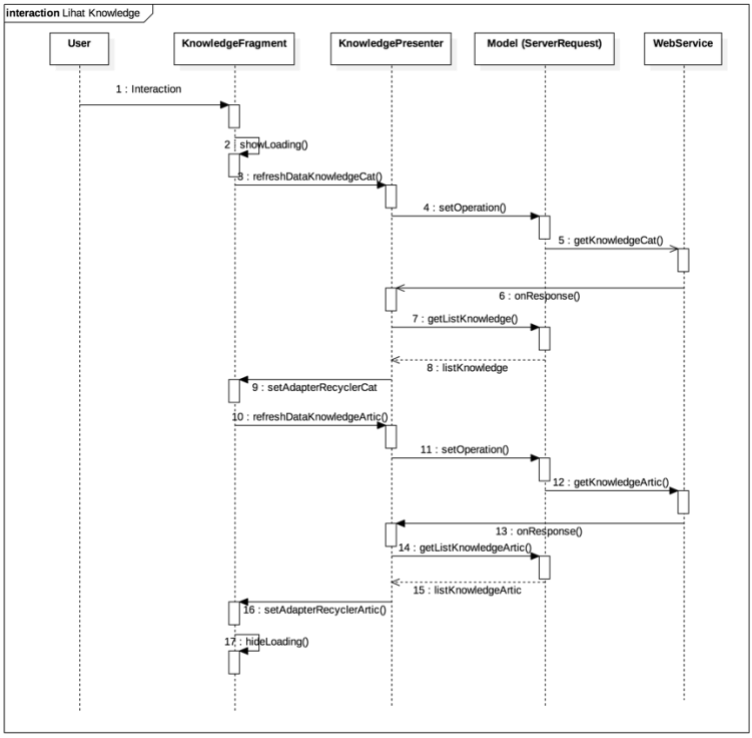
Gambar 4.14 Sequence Diagram Search Knowledge

4.2.3.5 Use Case U-05 dan T-05 (Lihat Knowledge)



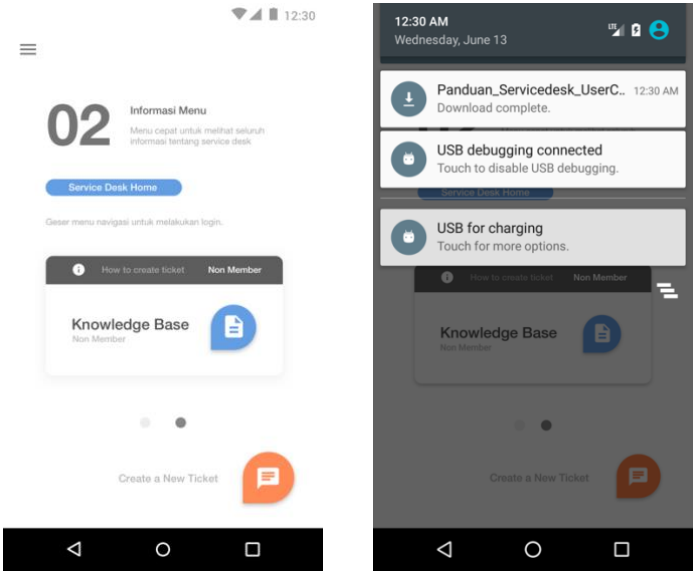
Gambar 4.15 User Interface Lihat Knowledge

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.16.



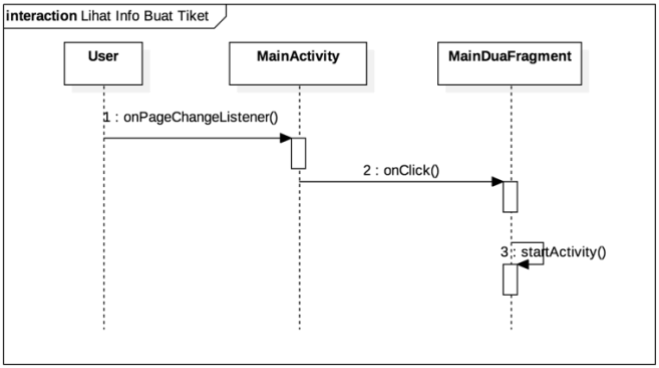
Gambar 4.16 Sequence Diagram Lihat Knowledge

4.2.3.6 Use Case U-06 dan T-06 (Lihat Info Buat Tiket)



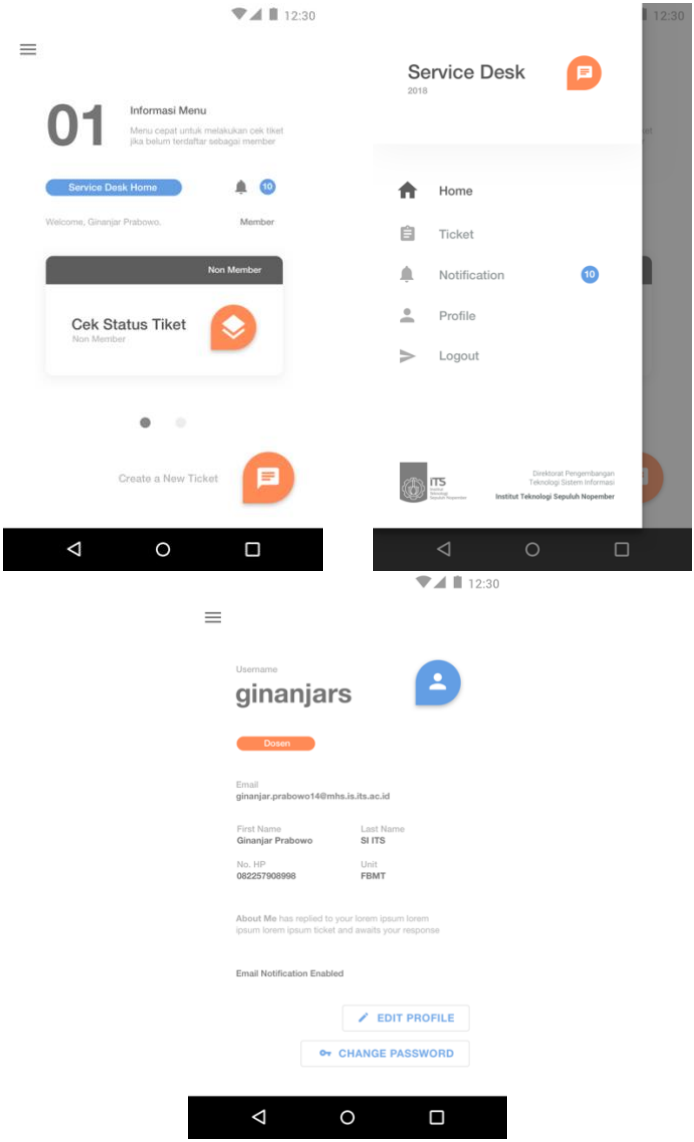
Gambar 4.17 Lihat Info Buat Tiket

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.18.



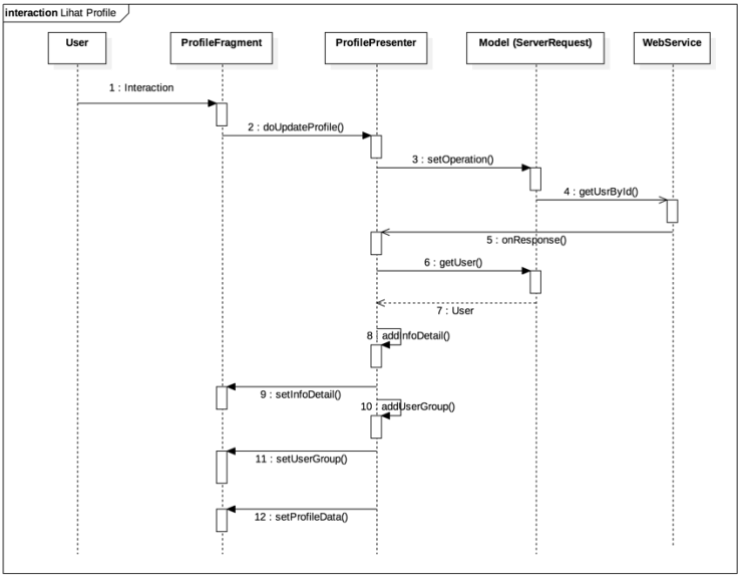
Gambar 4.18 Sequence Diagram Lihat Info Buat Tiket

4.2.3.7 Use Case U-07 dan T-07 (Lihat Profile)



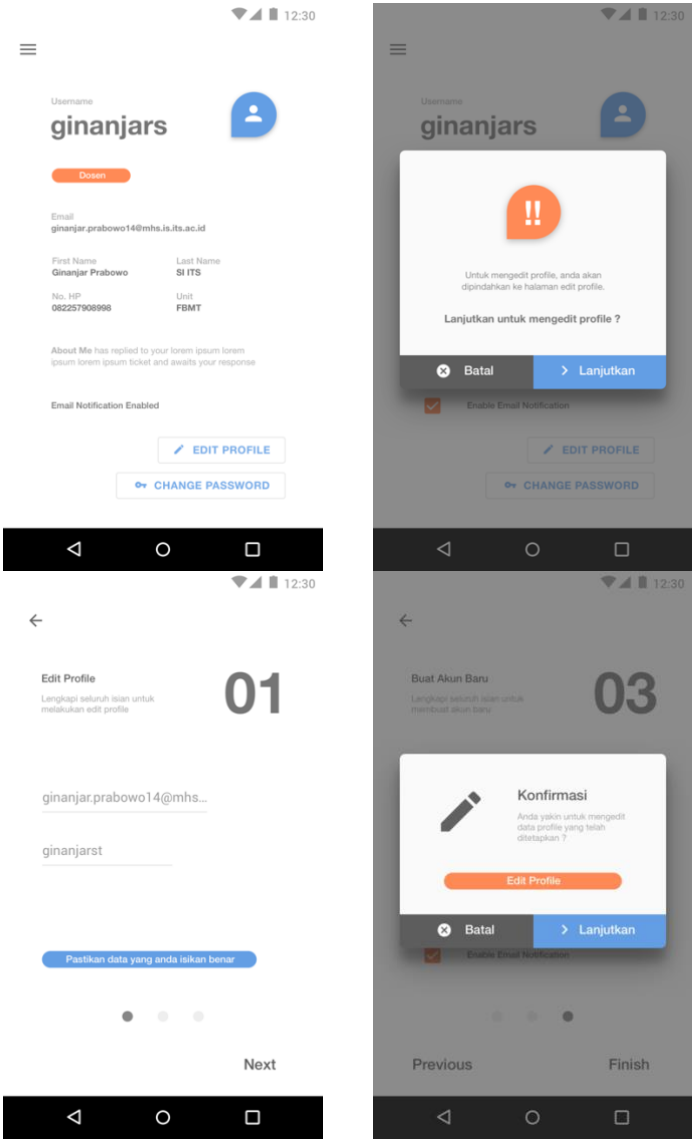
Gambar 4.19 Sequence Diagram Lihat Profile

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.20.



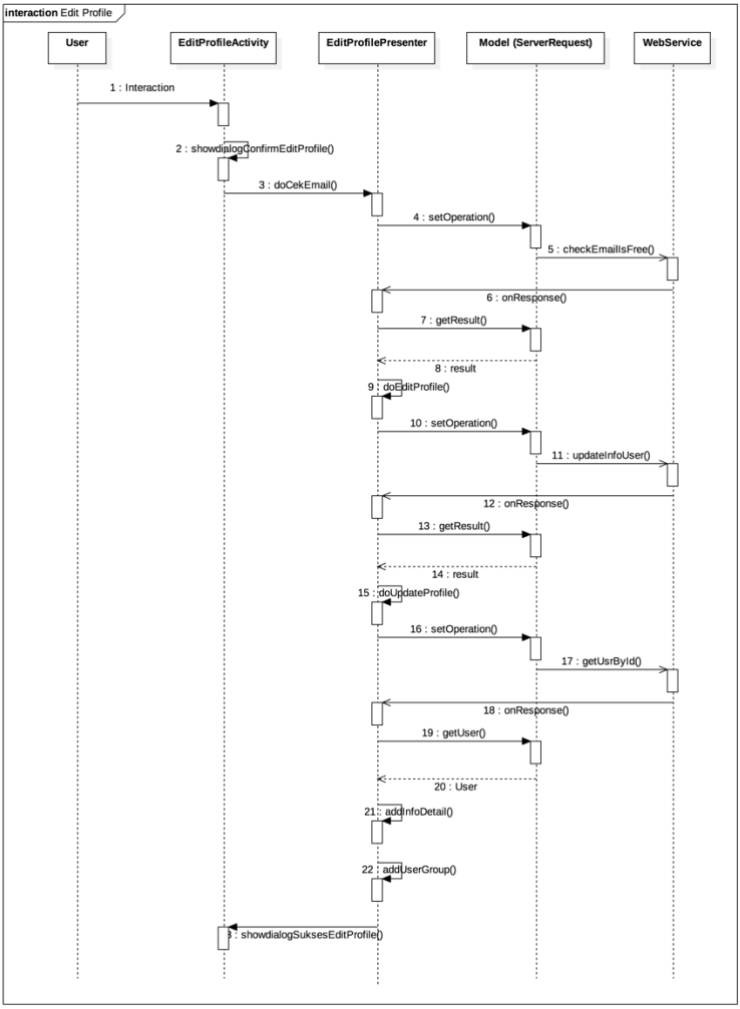
Gambar 4.20 Sequence Diagram Lihat Profile

4.2.3.8 Use Case U-08 dan T-08 (Edit Profile)



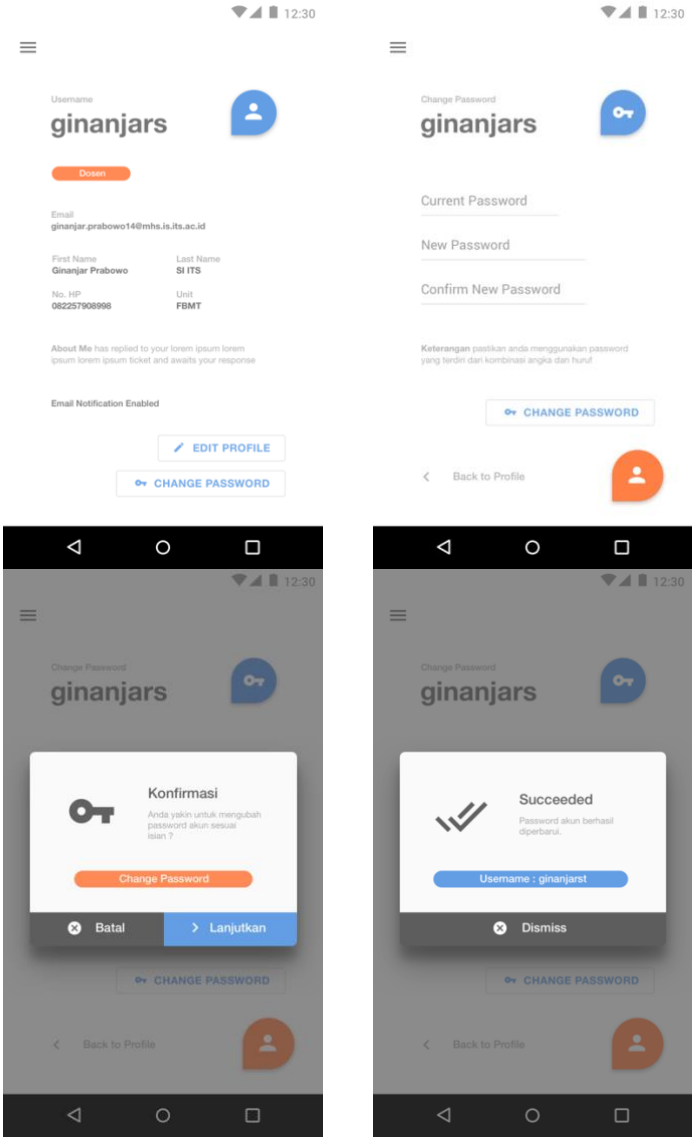
Gambar 4.21 User Interface Edit Profile

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.22.



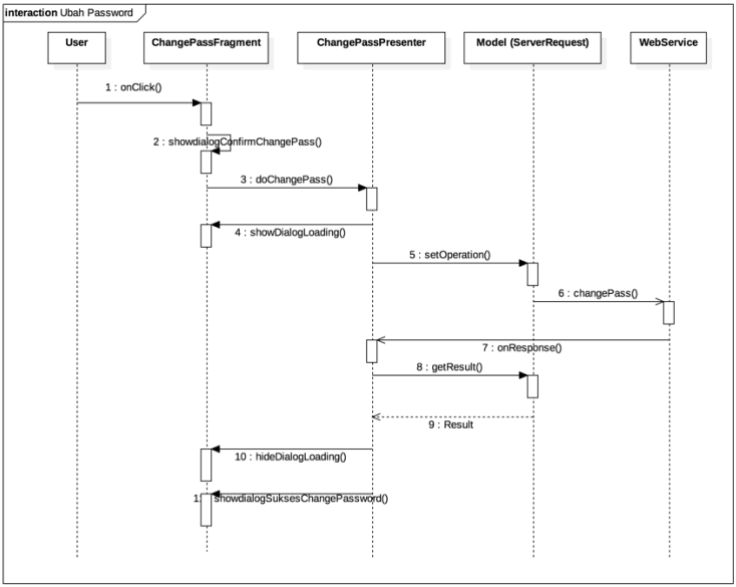
Gambar 4.22 Sequence Diagram Edit Profile

4.2.3.9 Use Case U-09 dan T-09 (Ubah Password)



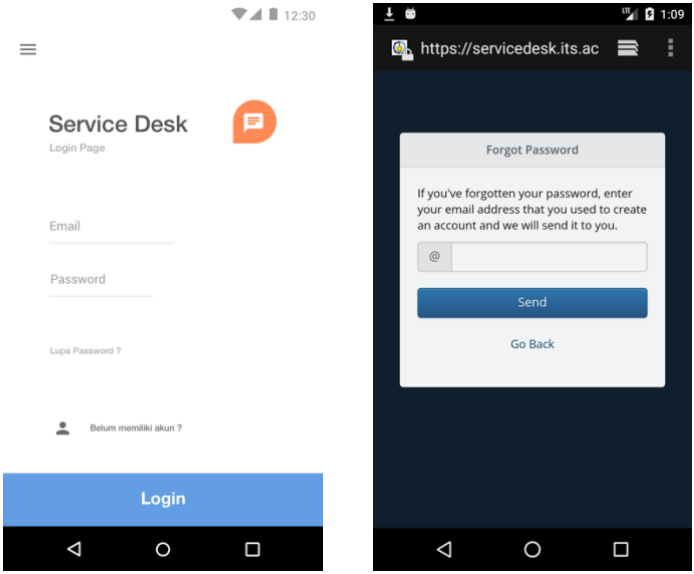
Gambar 4.23 User Interface Ubah Password

Adapun diagram sequence dari user interface diatas dapat dilihat pada gambar 4.24.



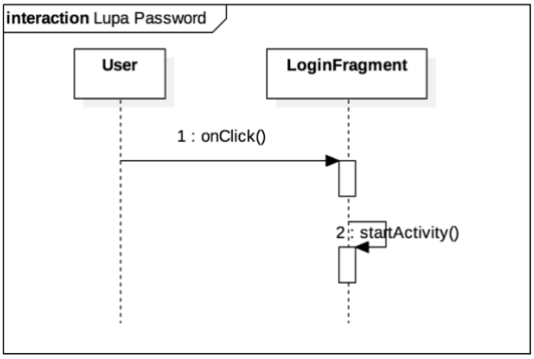
Gambar 4.24 Sequence Diagram Ubah Password

4.2.3.10 Use Case U-10 dan T-10 (Lupa Password)



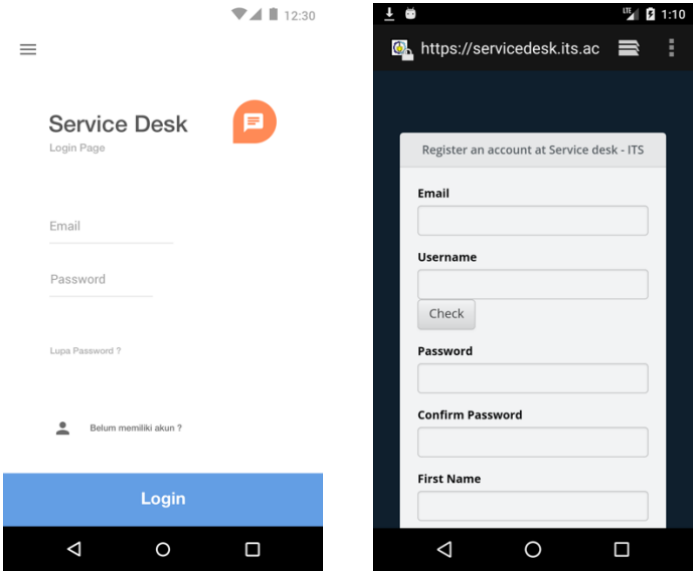
Gambar 4.25 User Interface Lupa Password

Adapun diagram sequence dari user interface diatas dapat dilihat pada gambar 4.26.



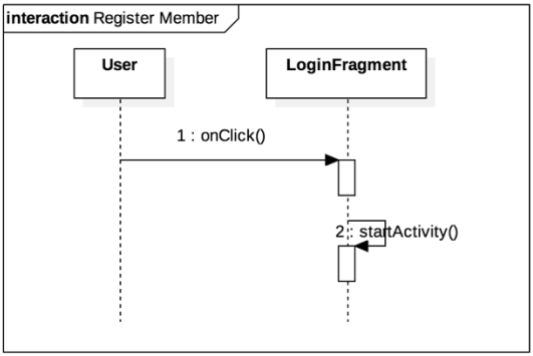
Gambar 4.26 Sequence Diagram Lupa Password

4.2.3.11 Use Case U-11 dan T-11 (Register Member)



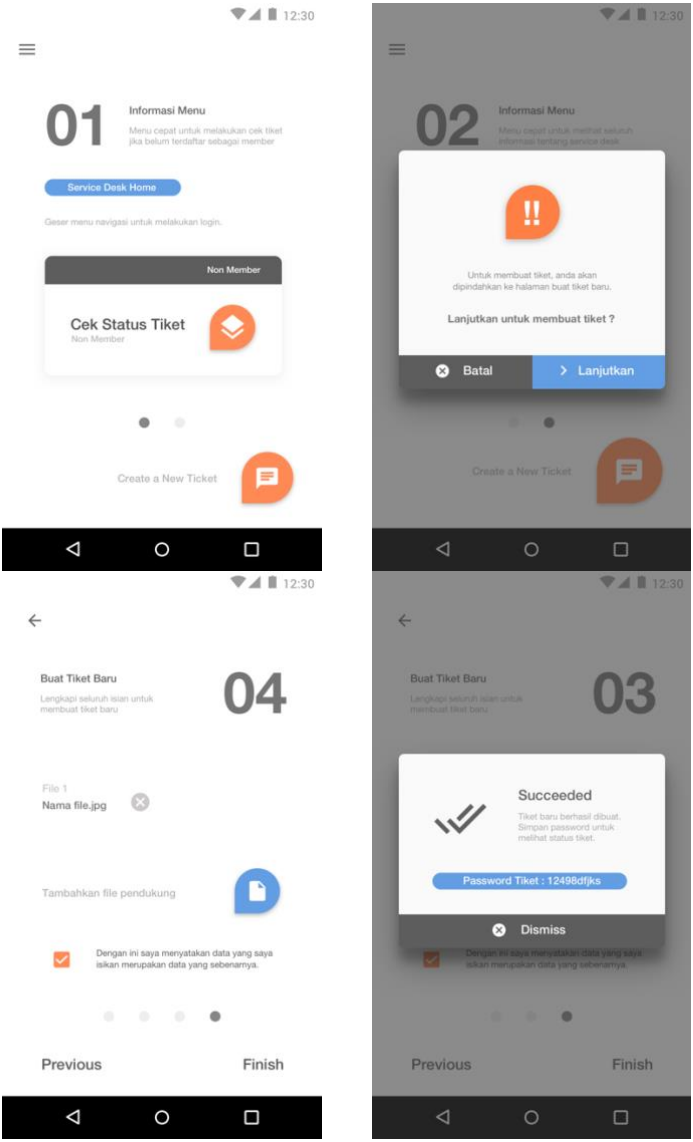
Gambar 4.27 User Interface Register Member

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.28.



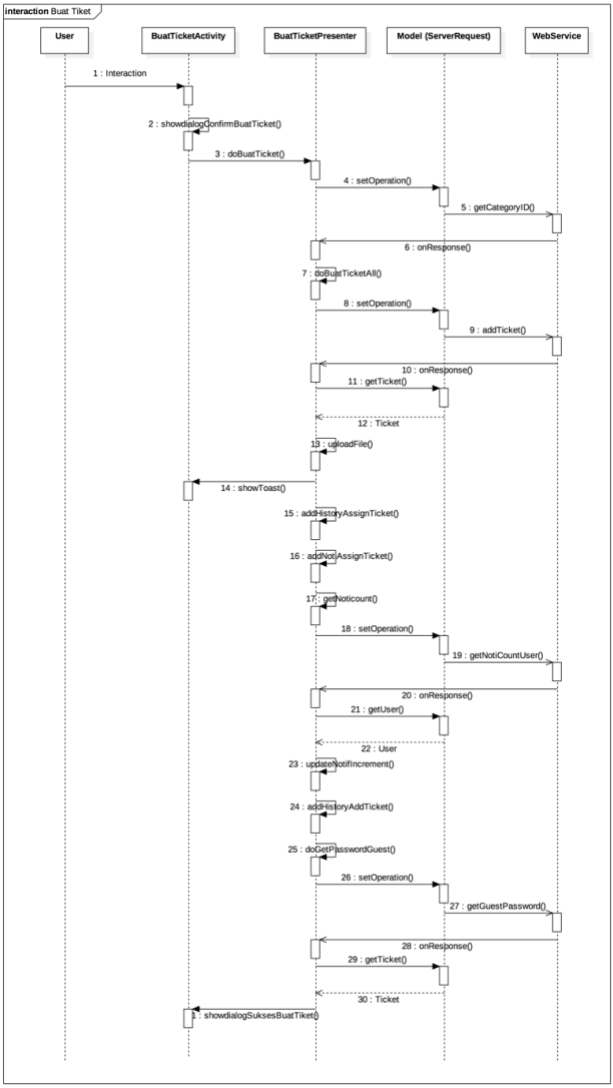
Gambar 4.28 Sequence Diagram Register Member

4.2.3.12 Use Case U-12 dan T-12 (Buat Tiket)



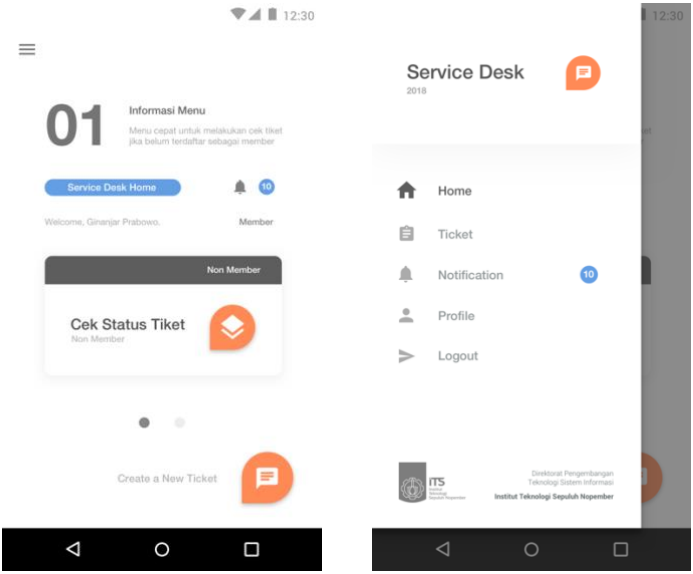
Gambar 4.29 User Interface Buat Tiket

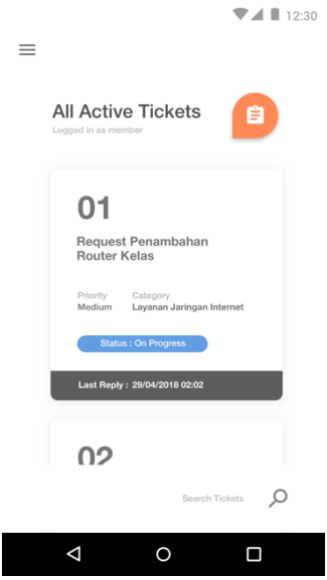
Adapun diagram sequence dari user interface diatas dapat dilihat pada gambar 4.30.



Gambar 4.30 Sequence Diagram Buat Tiket

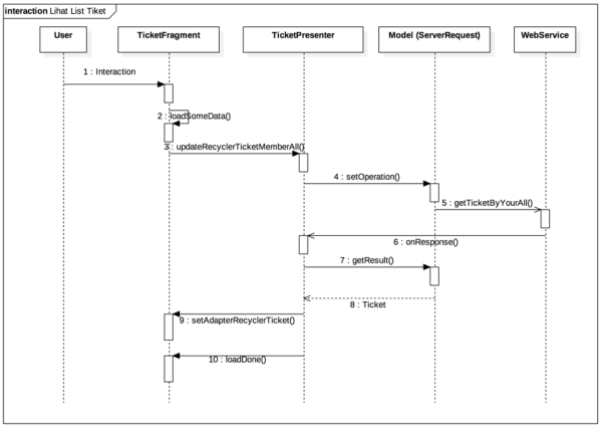
4.2.3.13 Use Case U-13 dan T-13 (Lihat List Tiket)





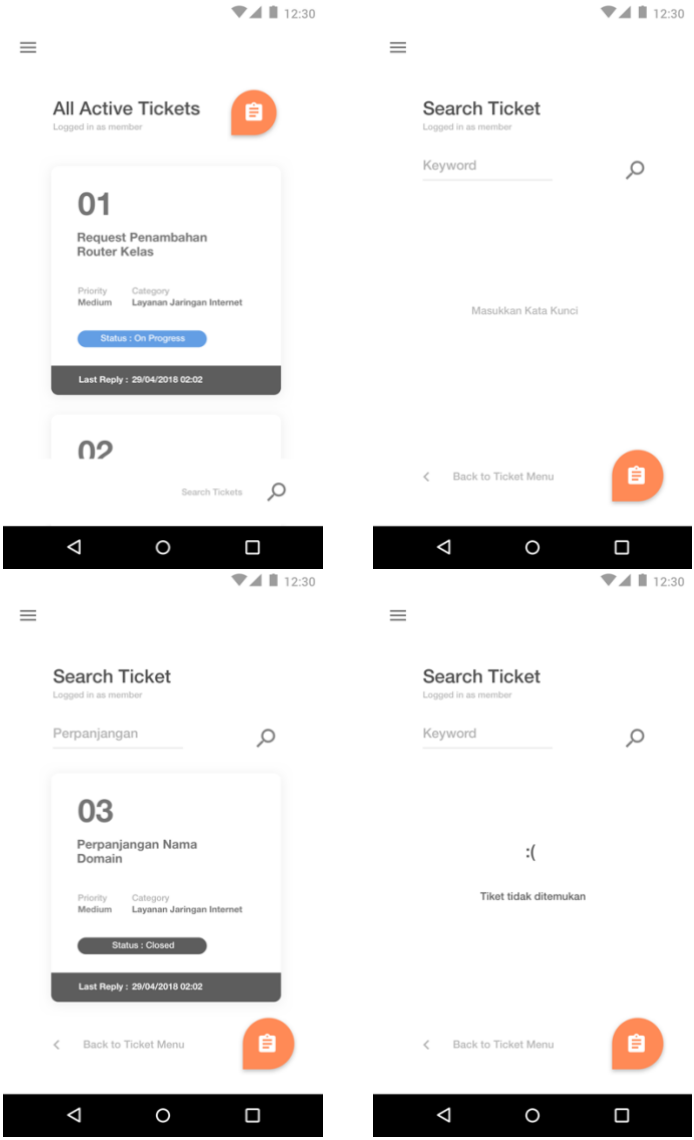
Gambar 4.31 User Interface Lihat List Tiket

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.32.



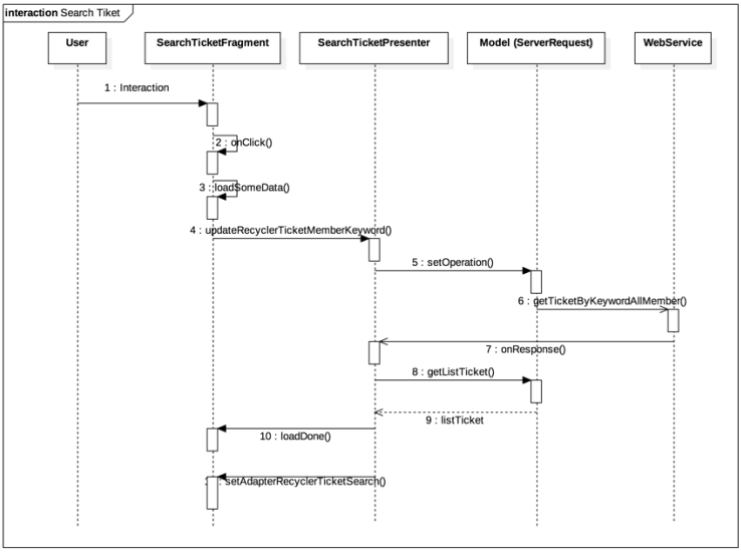
Gambar 4.32 Sequence Diagram Lihat List Tiket

4.2.3.14 Use Case U-14 dan T-14 (Search Tiket)



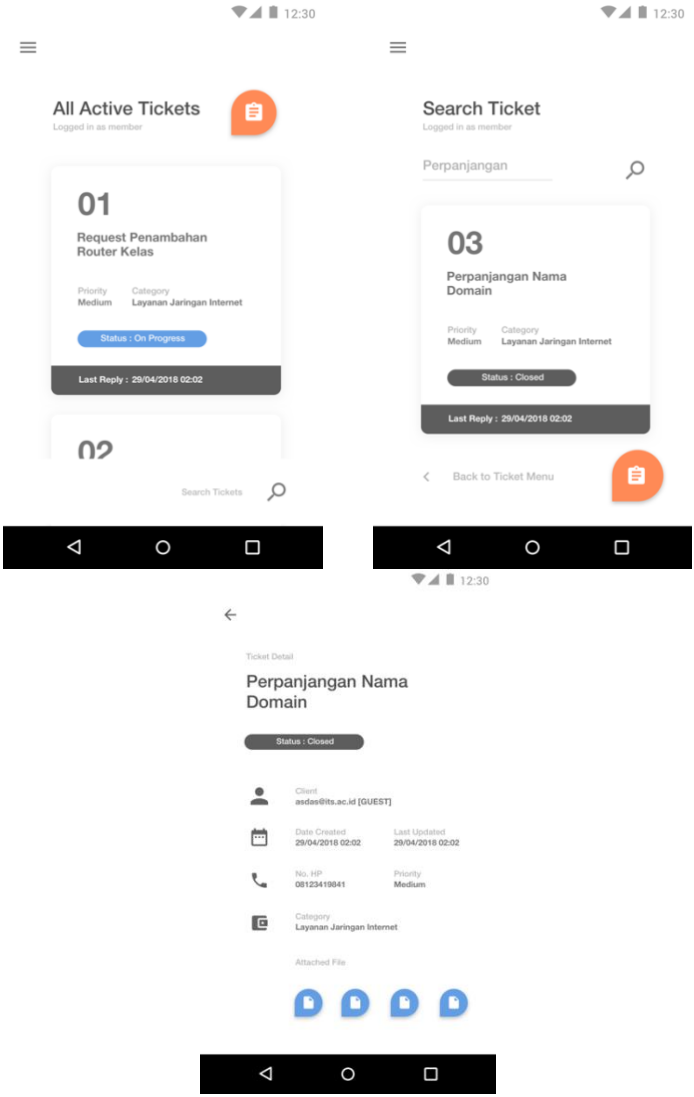
Gambar 4.33 User Interface Search Tiket

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.34.



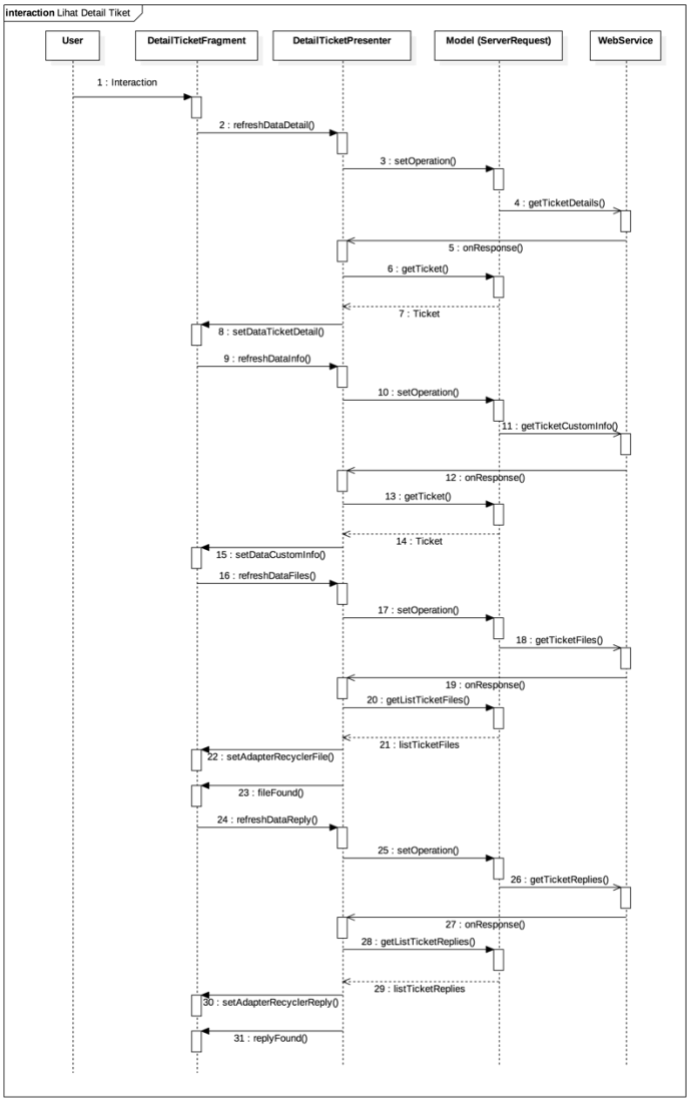
Gambar 4.34 Sequence Diagram Search Tiket

4.2.3.15 Use Case U-15 dan T-15 (Lihat Detail Tiket)



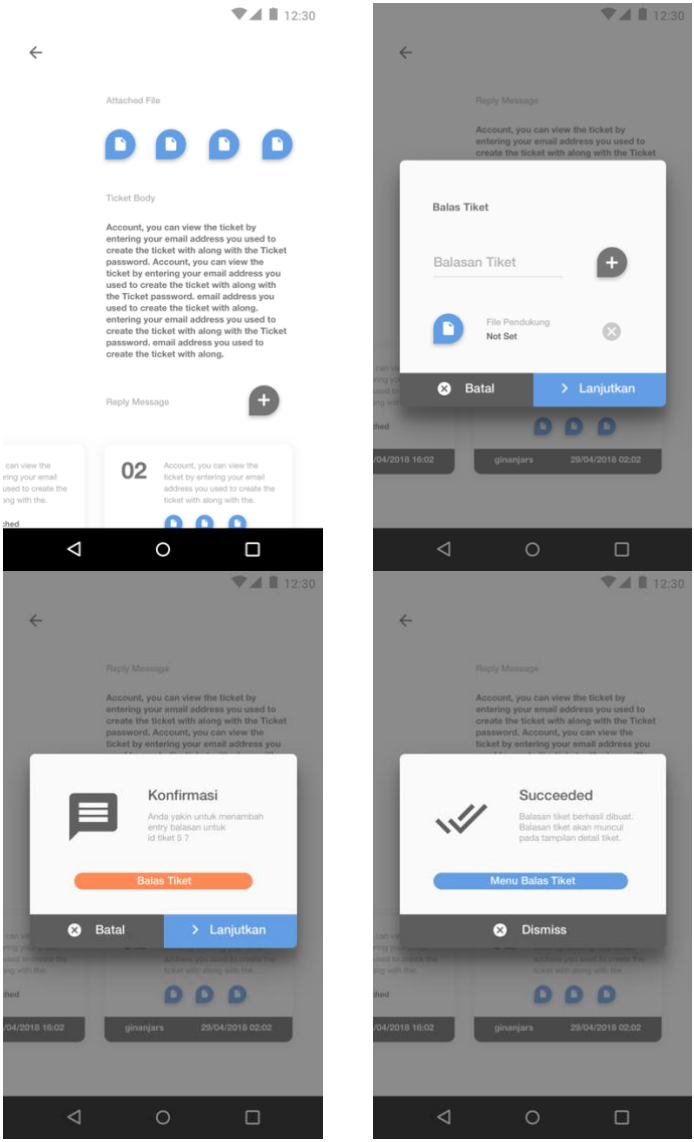
Gambar 4.35 User Interface Lihat Detail Tiket

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.36.



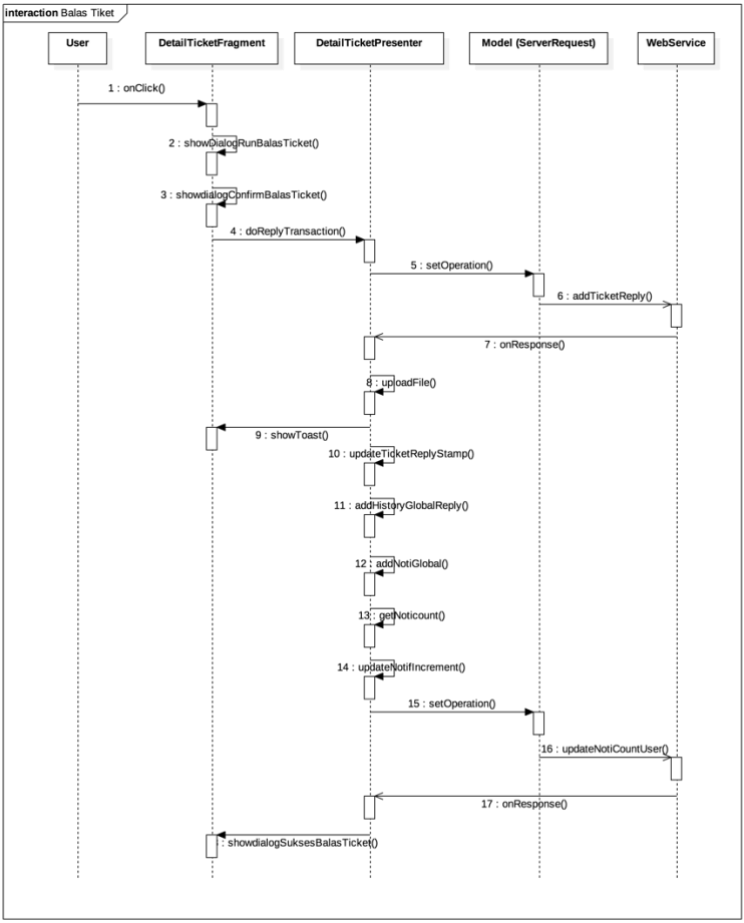
Gambar 4.36 Sequence Diagram Lihat Detail Tiket

4.2.3.16 Use Case U-16, U-18 dan T-16 (Balas Tiket)



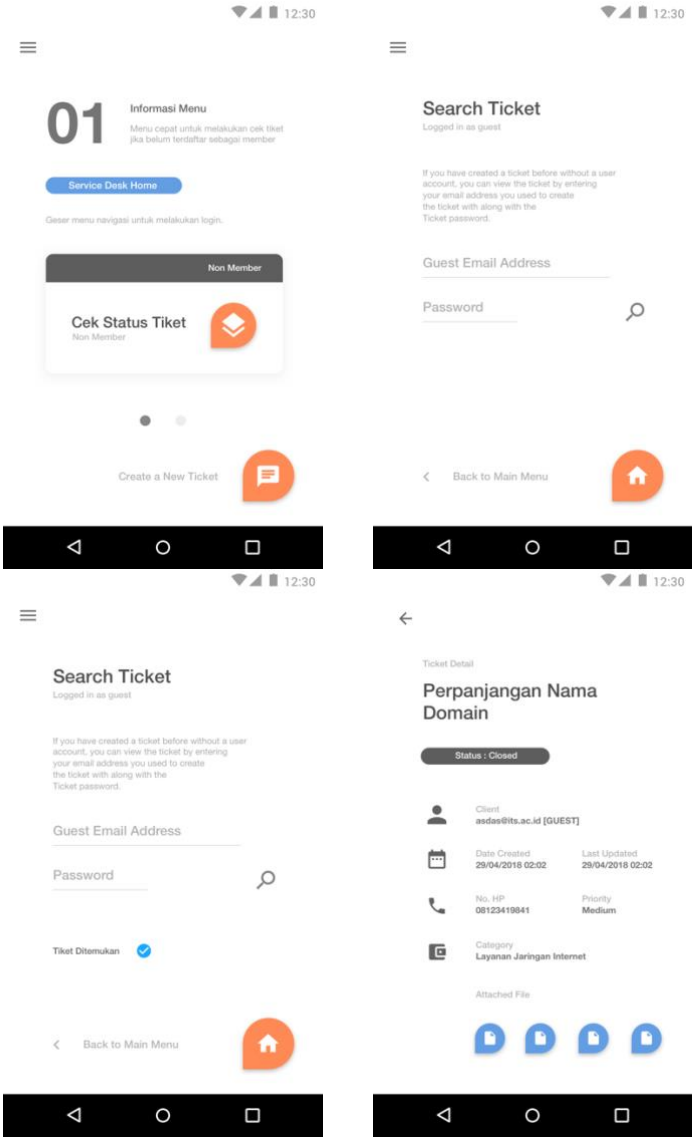
Gambar 4.37 User Interface Balas Tiket

Adapun diagram sequence dari user interface diatas dapat dilihat pada gambar 4.38.



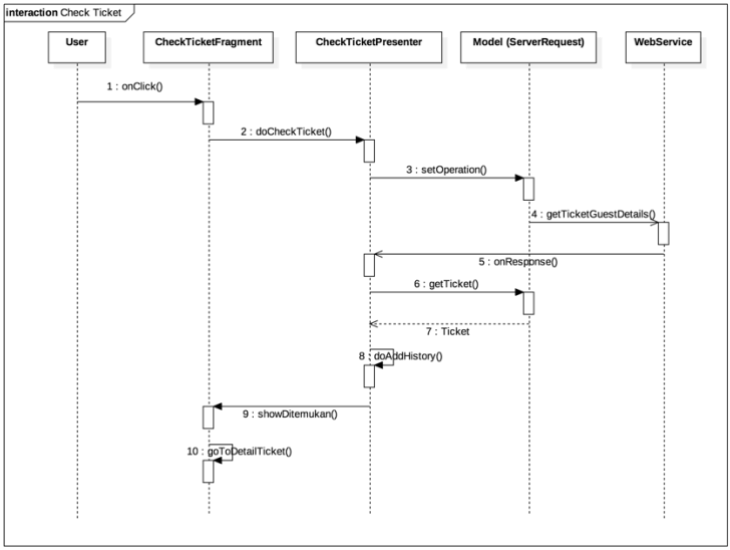
Gambar 4.38 Sequence Diagram Balas Tiket

4.2.3.17 Use Case U-17 (Check Tiket Guest)



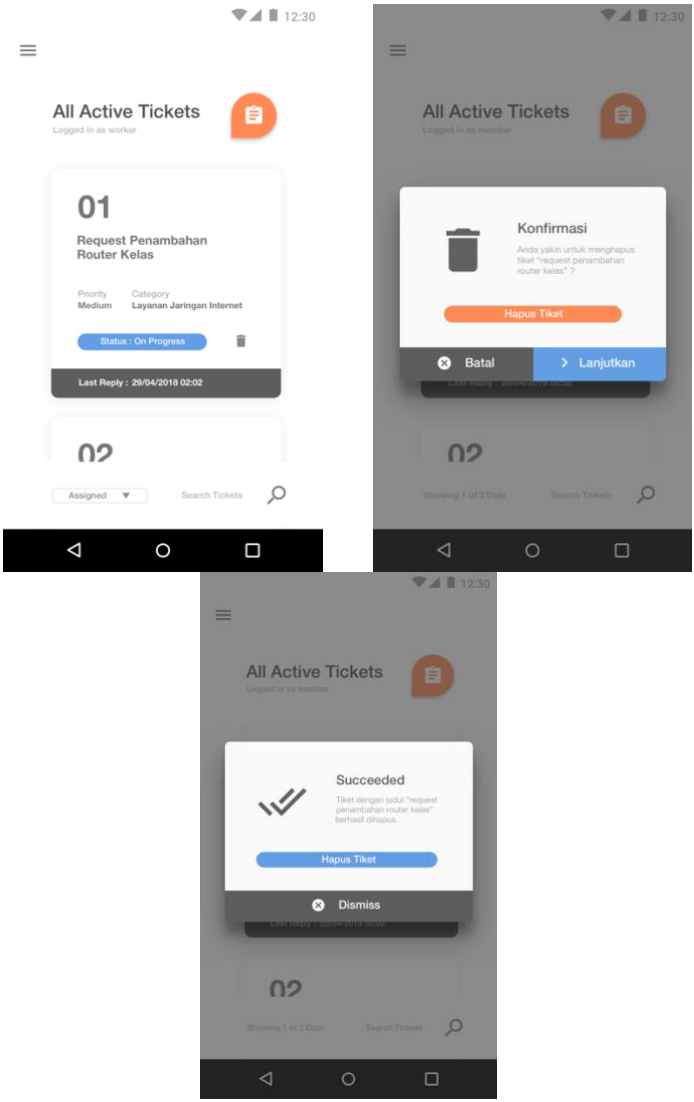
Gambar 4.39 User Interface Check Ticket (Guest)

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.40.



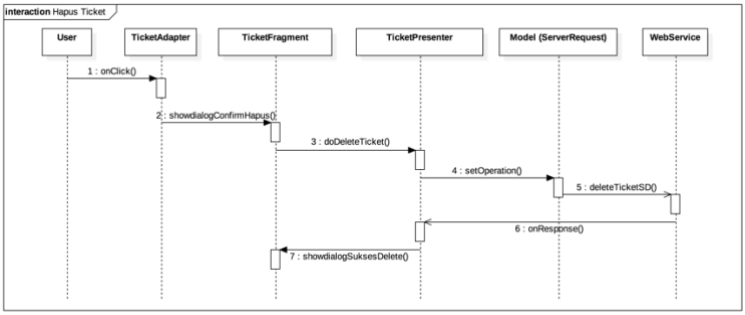
Gambar 4.40 Sequence Diagram Check Tiket

4.2.3.18 Use Case T-17 (Hapus Tiket)



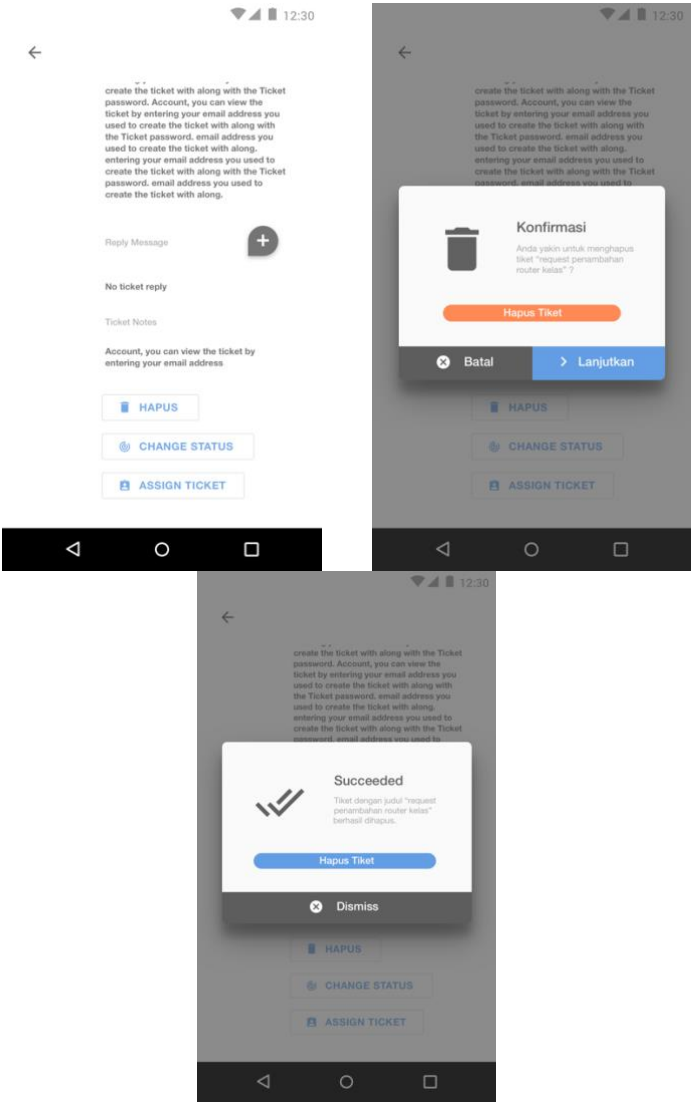
Gambar 4.41 User Interface Hapus Tiket

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.42.



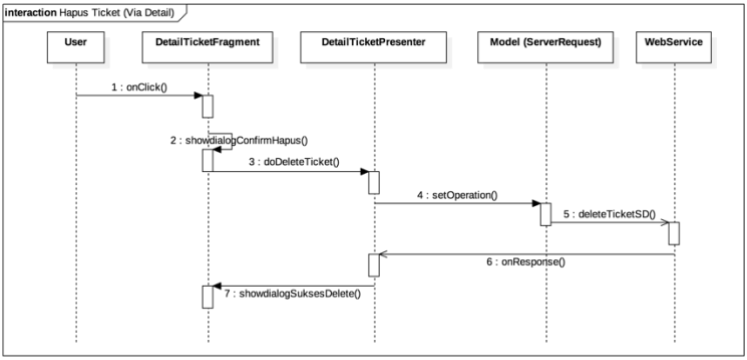
Gambar 4.42 Sequence Diagram Hapus Tiket

4.2.3.19 Use Case T-18 (Hapus Tiket Via Detail)



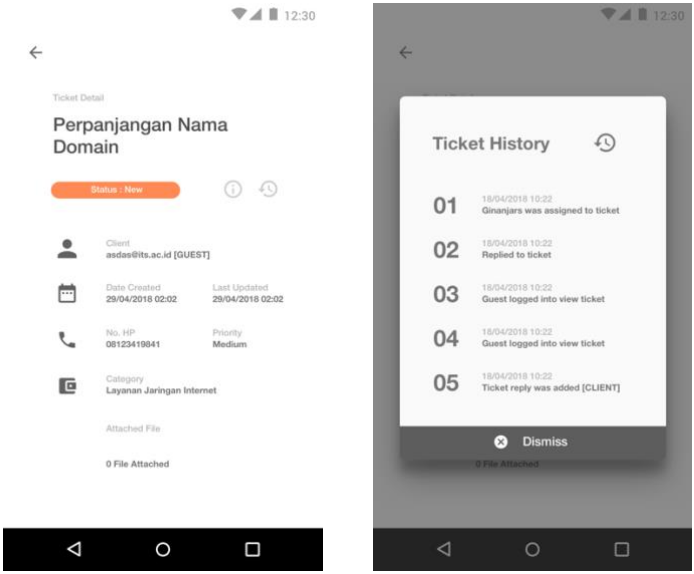
Gambar 4.43 User Interface Hapus Tiket (Via Detail)

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.44.



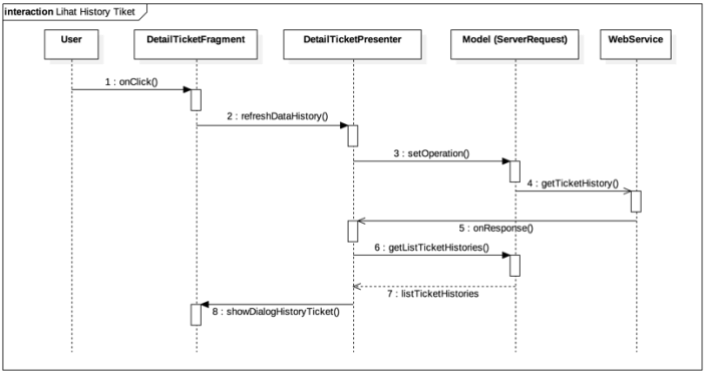
Gambar 4.44 Sequence Diagram Hapus Tiket (Via Detail)

4.2.3.20 Use Case T-19 (Lihat History Tiket)



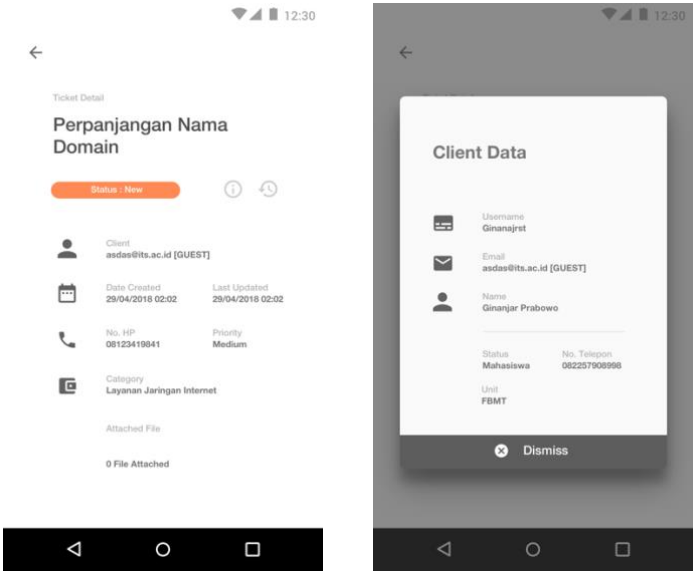
Gambar 4.45 User Interface Lihat History Tiket

Adapun diagram sequence dari desian user interface diatas dapat dilihat pada gambar 4.46.



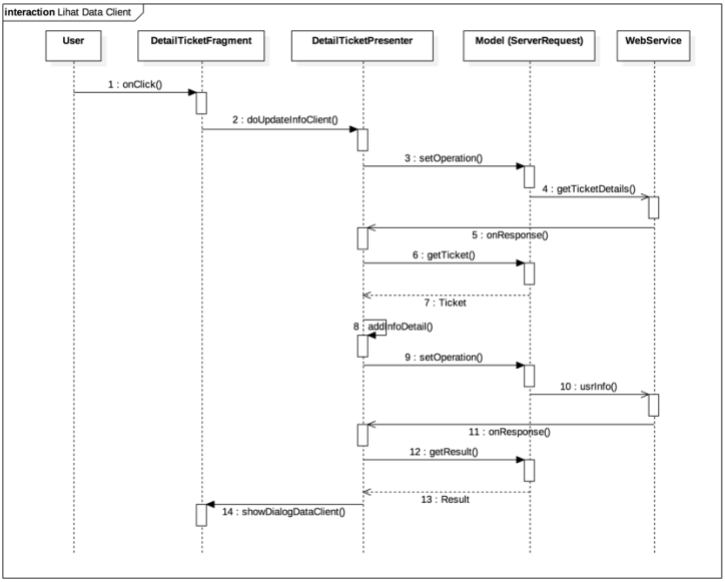
Gambar 4.46 Sequence Diagram Lihat History Tiket

4.2.3.21 Use Case T-20 (Lihat Data Client)



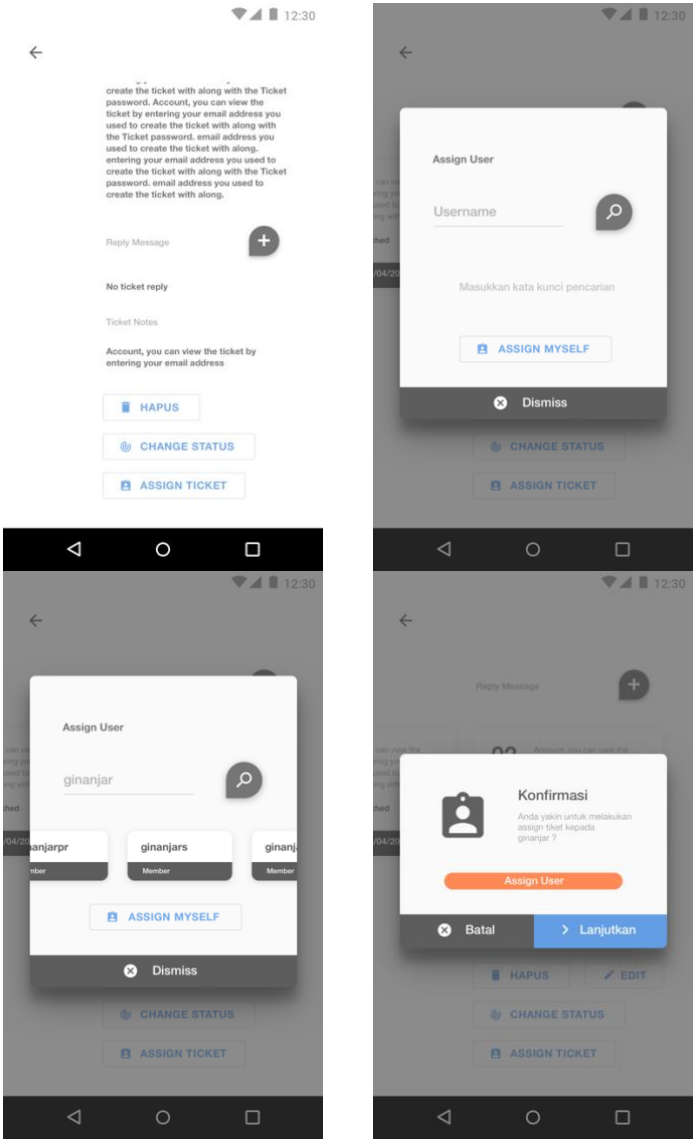
Gambar 4.47 User Interface Lihat Data Client

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.48.



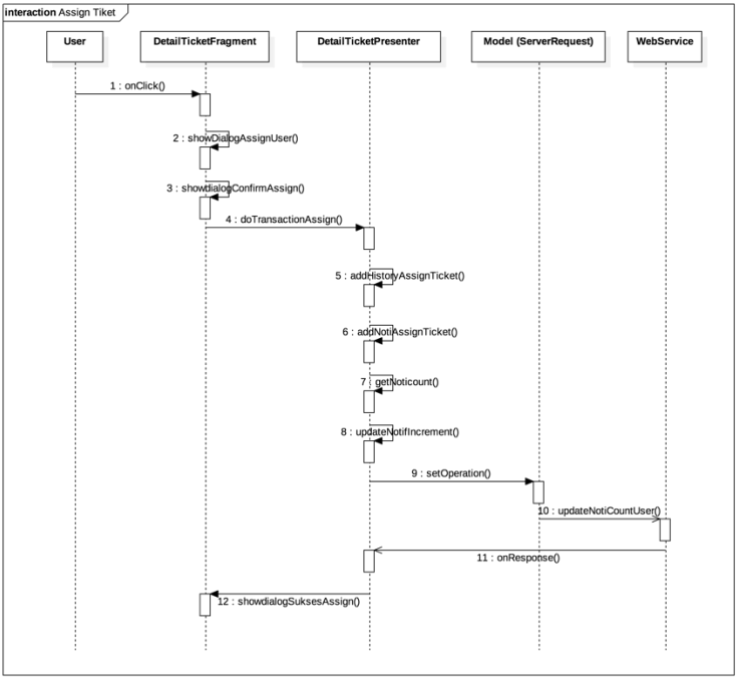
Gambar 4.48 Sequence Diagram Lihat Data Client

4.2.3.22 Use Case T-21 dan T-22 (Assign Tiket / Myself)



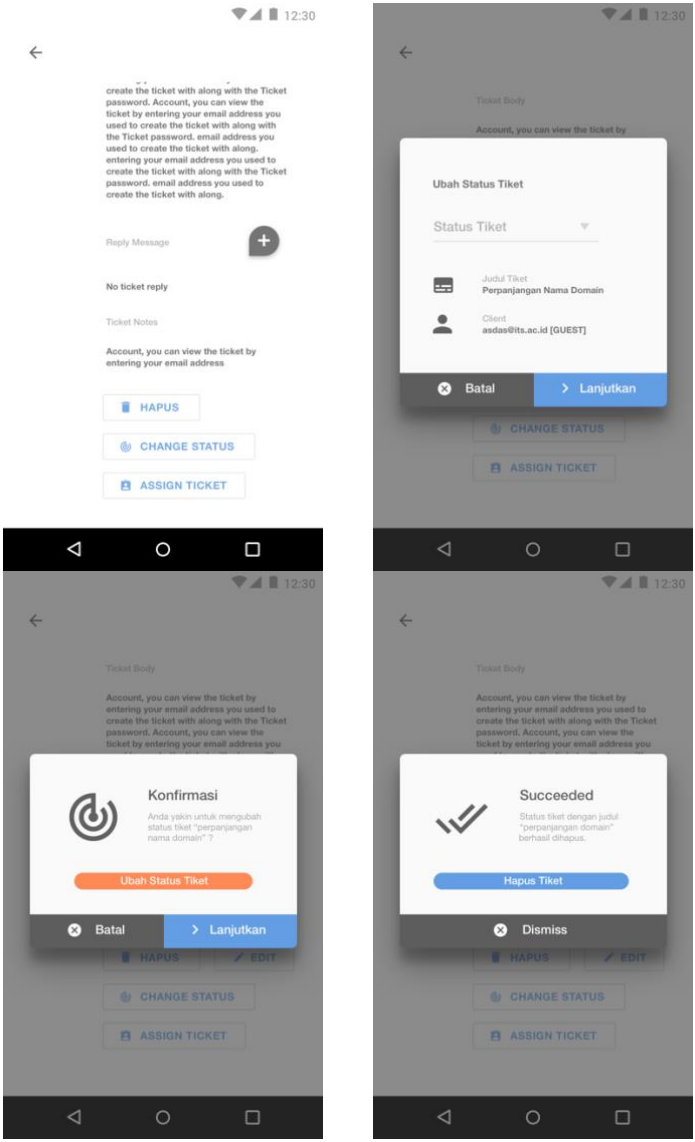
Gambar 4.49 User Interface Assign Tiket / Myself

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.50.



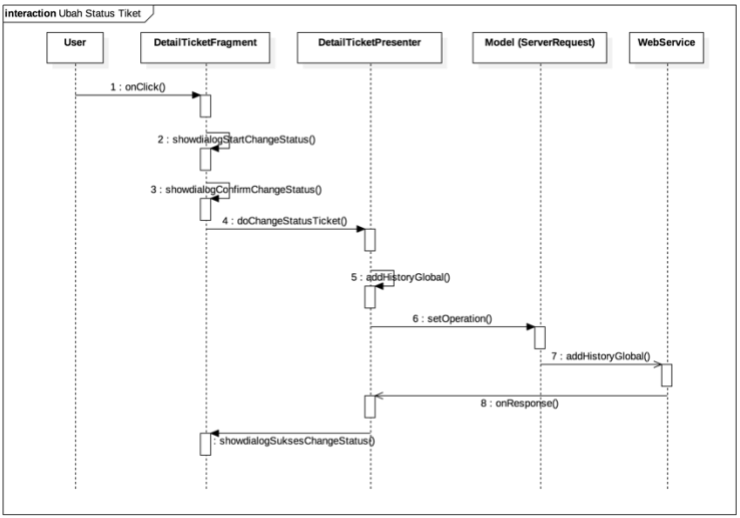
Gambar 4.50 Sequence Diagram Assign Tiket / Myself

4.2.3.23 Use Case T-23 (Ubah Status Tiket)



Gambar 4.51 User Interface Ubah Status Tiket

Adapun diagram sequence dari desain user interface diatas dapat dilihat pada gambar 4.52.



Gambar 4.52 Sequence Diagram Ubah Status Tiket

4.2.4 Perancangan Test Case

Tahap ini merupakan tahapan dimana skenario untuk melakukan pengujian pada setiap use case yang ada dibuat. Tujuan dari pembuatan test case adalah untuk menghasilkan berbagai kemungkinan interaksi guna dilakukan pengujian apakah sistem telah berjalan sesuai dengan desain yang telah dibuat. Test case dibuat berdasarkan metode pengujian black-box testing. Setiap test case mempunyai nomor pengujian dan scenario pengujian tertentu. Perancangan test case terdapat pada lampiran A.

BAB V
IMPLEMENTASI

Dalam bab ini akan dijelaskan implementasi dari perancangan yang telah dilakukan sesuai dengan metode pengembangan yang dibuat.

5.1 Lingkungan Implementasi

Pengembangan aplikasi mobile *service desk* beserta web servicenya menggunakan komputer dengan spesifikasi yang tertera pada tabel 5.1.

Tabel 5.1 Spesifikasi Komputer Pengembang

Processor	2.7 GHz Intel Core i5
Memory	8 GB RAM LPDDR3
Sistem Operasi	*MacOS High Sierra*
Graphic Card	*Intel Iris Graphics 550 1536 MB*

Aplikasi dan web service dikembangkan dengan menggunakan beberapa teknologi seperti editor, database, server, bahasa pemrograman, dan *library* yang terdapat pada tabel 5.2.

Tabel 5.2 Teknologi Pengembangan yang Digunakan

Webserver	Apache
Bahasa Pemrograman	Java, PHP
Database	MySQL

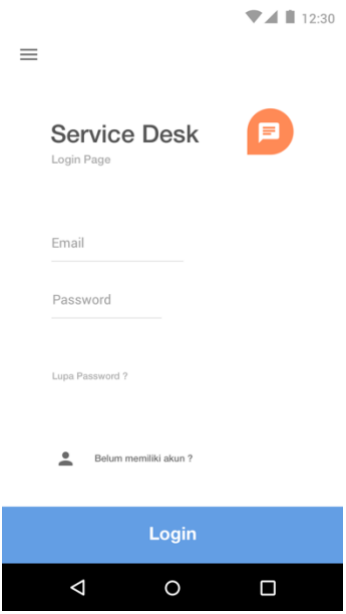
Editor (IDE)	Android Studio, Visual Code Editor
Library	• MySQL • Retrofit 2 • Converter-gson • Topsnackbar • Pageindicatorview • Calligraphy • Lottie

5.2 Implementasi Desain Aplikasi dan Web Service

Aplikasi mobile *service desk* dibangun dengan mengkonsumsi web service yang juga dibuat untuk menjalankan transaksi data. Pada babgian ini akan dijelaskan mengenai implementasi kode program aplikasi mobile *service desk* beserta web servicenya.

5.2.1 Implementasi Login

Fungsi pertama yang diimplementasikan sesuai desain adalah login dengan tampilan seperti pada gambar 5.1 dibawah ini.



Gambar 5.1 User Interface Login Aplikasi

Adapun potongan kode *view* method hideProgress() dalam *class* LoginFragment adalah sebagai berikut.

```
public void hideProgress(){

    progress.animate()
            .alpha(0f)
            .translationX(100f)
            .translationY(0f)
            .scaleX(1f)
            .scaleY(1f)
            .setStartDelay(0)
            .setInterpolator(new FastOutSlowInInterpolator())
            .start();

}
```

Gambar 5.2 Potongan Kode Program *View* Fungsi Login

Fungsi login akan membuka akses kepada user yang telah terdaftar berdasarkan user role yang didapat dari database. Fungsi login berjalan dengan mekanisme single login dengan memanfaatkan *sharedpreferences* untuk menyimpan data user yang telah dikembalikan oleh *Web Service*. Berikut potongan kode program *presenter* dengan nama method doLogin() dalam *class* dengan nama *LoginPresenter* pada gambar 5.3.

```
@Override
public void doLogin(String email, String password, final Context context) {

    mView.showProgress();
    pref = PreferenceManager.getDefaultSharedPreferences(context);

    RequestInterface requestInterface = ApiClient.getClient()
            .create(RequestInterface.class);

    User user = new User();
    user.setEmail(email);
    user.setPassword(password);
    ServerRequest request = new ServerRequest();
    request.setOperation(Constants.LOGIN_OPERATION);
    request.setUser(user);
    Call<ServerResponse> response = requestInterface.operation(request);

    response.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call,
                               final retrofit2.Response<ServerResponse> response) {

            final Handler handler = new Handler();
            handler.postDelayed(() -> {

                    ServerResponse resp = response.body();
                    if (resp.getMessage().equalsIgnoreCase(
                            anotherString: "Invalid Login Credentials")) {

                        mView.hideProgress();
                        mView.loginWrongCredentials();
```

Gambar 5.3 Potongan Kode Program *Presenter* Fungsi Login

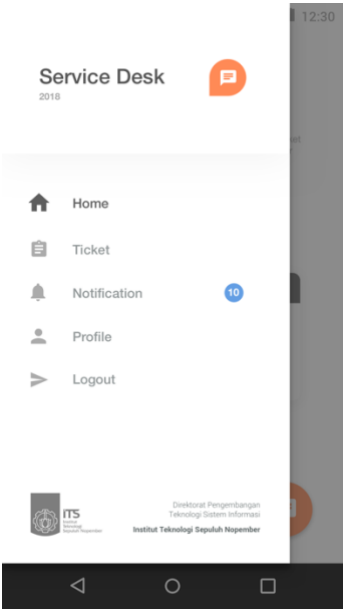
Dan dalam fungsi login, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi data. Adapun potongan kode program pada salah satu service yaitu *login* yang dikonsumsi client adalah sebagai berikut :

```php
public function getUserByEmail($email)
{
    $sql = 'SELECT COUNT(*) FROM users WHERE email = :email';
    $query = $this -> conn -> prepare($sql);
    $query -> execute(array(':email' => $email));

    $row_count = $query -> fetchColumn();

    if ($row_count == 0) {
        return false;
    } else {
        return true;
    }
}
```

Gambar 5.4 Potongan Kode Program Service *login* method *getUserByEmail*

5.2.2 Implementasi Logout

Fungsi kedua yang diimplementasikan sesuai desain adalah logout dengan tampilan seperti pada gambar 5.5 dibawah ini.



Gambar 5.5 User Interface Menu Logout Aplikasi

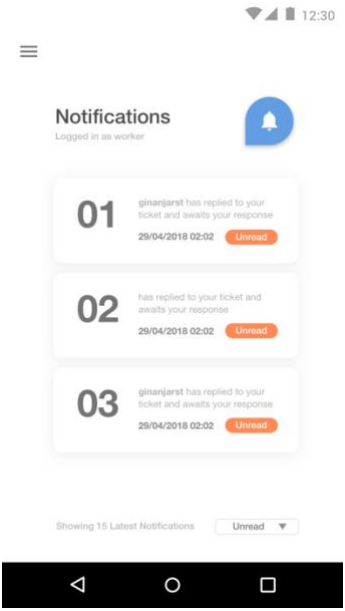
Karena mekanisme logout yang digunakan dalam aplikasi mobile service desk hanya memanfaatkan *sharedpreferences* (single login) tanpa *session* dan *token* maka tidak ada transaksi antara client dan database saat dilakukan proses logout, sehingga fungsi logout hanya memanfaatkan 1 class *view* dengan nama MainActivity untuk mereset kembali *sharedpreferences* yang telah dibuat saat melakukan login. Adapun potongan kode program yang berfungsi untuk melakukan logout dapat dilihat pada gambar 5.6.

```
private void logout() {
    SharedPreferences.Editor editor = pref.edit();
    editor.putBoolean(Constants.IS_LOGGED_IN,  b: false);
    editor.putString(Constants.ID,  s1: "");
    editor.putString(Constants.EMAIL,  s1: "");
    editor.putString(Constants.USERNAME,  s1: "");
    editor.putString(Constants.USER_ROLE,  s1: "");
    editor.putString(Constants.NOTI_COUNT,  s1: "");
    editor.putString(Constants.FIRST_NAME,  s1: "");
    editor.putString(Constants.LAST_NAME,  s1: "");
    editor.putString(Constants.EMAIL_NOTI,  s1: "");
    editor.putString(Constants.USER_GROUP,  s1: "");
    editor.putString(Constants.ABOUT,  s1: "");
    editor.apply();
```

Gambar 5.6 Potongan Kode Program *View* Fungsi Logout

5.2.3 Implementasi Lihat Notifikasi

Fungsi ketiga yang diimplementasikan sesuai desain adalah login dengan tampilan seperti pada gambar 5.8 dibawah ini.



Gambar 5.7 User Interface Lihat Notifikasi Aplikasi

Adapun potongan kode *view* method onCreateView() di dalam *class* NotificationsFragment adalah sebagai berikut.

```
loadSomeData();
mPresenter.getNoticount(view.getContext());
mPresenter.updateNotificationAll(idpref);

if(Constants.connect == 0){

    networkUnavailable();

}

return view;
```

Gambar 5.8 Potongan Kode Program *View* Fungsi Lihat Notifikasi

Pada halaman awal fungsi lihat notifikasi, ditampilkan list notifikasi yang juga dimungkinkan untuk melihat detail apabila salah satu listnya ditekan. Berikut potongan kode program *presenter* dalam method doLogin() yang ada di dalam *class* LoginPresenter pada gambar 5.10 dibawah.

```
public void updateNotificationAll(String userid) {

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Notification notification = new Notification();
    notification.setUserid(userid);
    ServerRequest request = new ServerRequest();
    request.setOperation("getNotifications");
    request.setNotification(notification);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            final Handler handler = new Handler();
            handler.postDelayed(() -> {

                    if (response.body() != null) {
                        notificationList = response.body().getListNotifications();

                        mAdapterNotification = new NotificationAdapter(notificationList);
                        mView.loadDone();
                        mView.setAdapterRecyclerNotif(mAdapterNotification);

                    } else {

                        notificationList = Collections.emptyList();
                        mAdapterNotification = new NotificationAdapter(notificationList);
                        mView.loadDone();
                        mView.setAdapterRecyclerNotif(mAdapterNotification);
                        mView.showNotFound();
```

Gambar 5.9 Potongan Kode Program *Presenter* Fungsi Login

Dan dalam fungsi lihat notifikasi, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *getNotifications* adalah sebagai berikut :

```
public function getNotifications($userid)
{
    $sql = "SELECT users.ID as userid, users.username, users.avatar,
user_notifications.timestamp, user_notifications.message,
user_notifications.url, user_notifications.ID,
user_notifications.status

FROM user_notifications
LEFT OUTER JOIN users ON users.ID = user_notifications.fromid
WHERE user_notifications.userid = ".$userid."
ORDER BY user_notifications.ID DESC
LIMIT 10";

    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    // query products
    $stmt = $query;
    $num = $stmt->rowCount();
```

Gambar 5.10 Potongan Kode Program Service *getNotifications*

5.2.4 Implementasi Search Knowledge

Fungsi keempat yang diimplementasikan sesuai desain adalah search knowledge dengan tampilan seperti pada gambar 5.11 dibawah ini.

Gambar 5.11 User Interface Search Knowledge

Fungsi search knowledge merupakan fungsi yang dijalankan dengan syarat harus melewati trigger onClick() pada button search yang terdapat pada tampilan list knowledge. Potongan kode *view* method showDialogSearchKnowledge() di dalam *class* Knowledge Fragment adalah sebagai berikut.

```
if (!tv_username.getText().toString().isEmpty()) {

    Constants.recyclerviewKnowledgeArticModal.setAlpha(0f);
    Constants.recyclerviewKnowledgeArticModal.setTranslationX(-50f);
    Constants.recyclerviewKnowledgeArticModal.setVisibility(View.VISIBLE);

    Constants.mPresenterModal.doSearchKnowledge(tv_username.getText().toString());
    loadingModal();

} else {

    CSnackbar.createSnackbarBottom(view, Color.WHITE,  background: "#E18585",  isi: "M

}
```

Gambar 5.12 Potongan Kode Program *View* Fungsi Search Knowledge

Berikut potongan kode program *presenter* dalam method doSearchKnowledge() yang ada di dalam *class* KnowledgePresenter pada gambar 5.13 dibawah.

```
public void updateNotificationAll(String userid) {

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Notification notification = new Notification();
    notification.setUserid(userid);
    ServerRequest request = new ServerRequest();
    request.setOperation("getNotifications");
    request.setNotification(notification);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            final Handler handler = new Handler();
            handler.postDelayed(() -> {

                if (response.body() != null) {
                    notificationList = response.body().getListNotifications();

                    mAdapterNotification = new NotificationAdapter(notificationList);
                    mView.loadDone();
                    mView.setAdapterRecyclerNotif(mAdapterNotification);

                } else {

                    notificationList = Collections.emptyList();
                    mAdapterNotification = new NotificationAdapter(notificationList);
                    mView.loadDone();
                    mView.setAdapterRecyclerNotif(mAdapterNotification);
                    mView.showNotFound();
```

Gambar 5.13 Potongan Kode Program *Presenter* Fungsi Login

Dan dalam fungsi search knowledge, aplikasi melakukan request terhadap satu service dalam melakukan transaksi data. Adapun potongan kode program service bernama

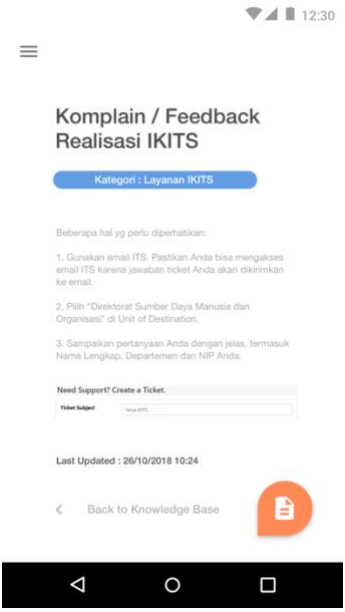
getKnowledgeByCatOrArt yang dikonsumsi client adalah sebagai berikut :

```
public function getKnowledgeByCatOrArt($keyword)
{

    $sql = "SELECT * FROM knowledge_articles
            WHERE (`title` LIKE '%".$keyword."%') OR
            (`body` LIKE '%".$keyword."%')";

    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    // query products
    $stmt = $query;
    $num = $stmt->rowCount();
```

Gambar 5.14 Potongan Kode Program Service *getKnowledgeByCatOrArt*

5.2.5 Implementasi Lihat Knowledge

Fungsi kelima yang diimplementasikan sesuai desain adalah lihat knowledge dengan tampilan seperti pada gambar 5.15 dibawah ini.



Gambar 5.15 User Interface Lihat Knowledge

Potongan kode *view* method onCreateView() di dalam *class* KnowledgeDetailFragment adalah sebagai berikut.

```
String valueId = getArguments().getString( key: "id");
String valueCat = getArguments().getString( key: "cat");

kategoriKnowledge.setText(valueCat);
mPresenter.refreshDataDetail(valueId);

Constants.TOGGLE.setDrawerIndicatorEnabled(false);
Constants.TOGGLE.syncState();

if(Constants.connect == 0){

    networkUnavailable();

}
```

Gambar 5.16 Potongan Kode Program *View* Fungsi Lihat Knowledge

Berikut potongan kode program *presenter* dalam method refreshDataDetail() yang ada di dalam *class* KnowledgeDetailPresenter pada gambar 5.13 dibawah.

```
public void refreshDataDetail(String id) {

    RequestInterface requestInterface = ApiClient.getClient().create(RequestInterface.class);

    Knowledge knowledge = new Knowledge();
    knowledge.setId(id);
    ServerRequest request = new ServerRequest();
    request.setOperation("getKnowledgeArticDetail");
    request.setKnowledge(knowledge);
    Call<ServerResponse> response = requestInterface.operation(request);

    response.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final retrofit2.Response<ServerResponse> response) {

            ServerResponse resp = response.body();

            String result = Html.fromHtml(resp.getKnowledge().getBody()).toString();

            long unixSeconds = Long.parseLong(resp.getKnowledge().getLast_updated_timestamp());
            Date date = new java.util.Date(unixSeconds * 1000L);
            SimpleDateFormat sdf = new java.text.SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss z");
            sdf.setTimeZone(java.util.TimeZone.getDefault());
            String formattedDate = sdf.format(date);

            mView.setInfoKnowledge(resp.getKnowledge().getTitle(),
                    result, formattedDate);
```

Gambar 5.17 Potongan Kode Program *Presenter* Fungsi Lihat Knowledge

Dan dalam fungsi lihat knowledge, aplikasi melakukan request terhadap satu service dalam melakukan transaksi data. Adapun potongan kode program service bernama

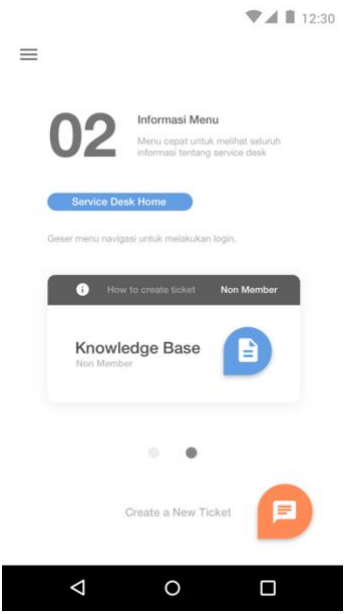
getKnowledgeArticDetail yang dikonsumsi client adalah sebagai berikut :

```php
public function getKnowledgeArticDetail($ID)
{
    $sql = 'SELECT * FROM knowledge_articles WHERE ID = :ID';
    $query = $this -> conn -> prepare($sql);
    $query -> execute(array(':ID' => $ID));
    $data = $query -> fetchObject();

    if ($query) {
        $knowledge["id"] = $data -> ID;
        $knowledge["title"] = $data -> title;
        $knowledge["catid"] = $data -> catid;
        $knowledge["body"] = $data -> body;
        $knowledge["last_updated_timestamp"] = $data -> last_updated_timestamp;
        return $knowledge;
    } else {
        return false;
    }
}
```

Gambar 5.18 Potongan Kode Program Service *getKnowledgeByCatOrArt*

5.2.6 Implementasi Lihat Info Buat Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah lihat info buat tiket dengan tampilan seperti pada gambar 5.19 dibawah ini.



Gambar 5.19 User Interface Lihat Info Buat Tiket

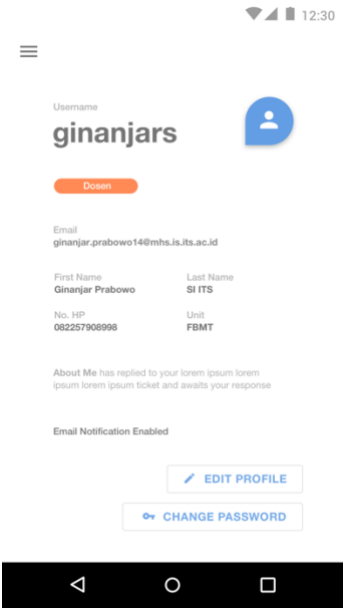
Fungsi lihat info buat tiket hanya berfungsi sebagai pengirim *intent* untuk membuka link yang berisikan info pembuatan tiket. Oleh karena itu fungsi ini hanya memanfaatkan 1 class *view* dan 1 method intent yang dapat dilihat pada gambar 5.20 dibawah.

```
@Override
public void onClick(View v) {

    switch (v.getId()) {

        case R.id.btnInfoBuatTiket:
            Intent viewIntent =
                    new Intent( action: "android.intent.action.VIEW",
                            Uri.parse("https://servicedesk.its.ac.id/
            startActivity(viewIntent);
            break;

    }
```

Gambar 5.20 Potongan Kode Program *View* Lihat Info Buat Tiket

5.2.7 Implementasi Lihat Profile

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah lihat profile dengan tampilan seperti pada gambar 5.21 dibawah ini.



Gambar 5.21 User Interface Lihat Profile

Potongan kode *view* method onCreateView() di dalam *class* ProfileFragment adalah sebagai berikut.

```
pref = getDefaultSharedPreferences(view.getContext());

btnChangePass.setOnClickListener(this);
btnEditProfile.setOnClickListener(this);

mPresenter = new ProfilePresenter( view: this);
mPresenter.doUpdateProfile(pref.getString(Constants.ID
        , s1: "")
        , getActivity());

if(Constants.connect == 0){

    networkUnavailable();

}
```

Gambar 5.22 Potongan Kode Program *View* Fungsi Lihat Profile

Berikut potongan kode program *presenter* dalam method doUpdateProfile() yang ada di dalam *class* ProfilePresenter pada gambar 5.23 dibawah.

```
response.enqueue(new Callback<ServerResponse>() {
    @Override
    public void onResponse(Call<ServerResponse> call, final retrofit2.Response<ServerResponse>

        ServerResponse resp = response.body();

        if (resp.getResult().equals(Constants.SUCCESS)) {
            SharedPreferences.Editor editor = pref.edit();
            editor.putBoolean(Constants.IS_LOGGED_IN,  b: true);

            String id = resp.getUser().getId();
            String email = resp.getUser().getEmail();
            String username = resp.getUser().getUsername();
            String user_role = resp.getUser().getUser_role();
            String noti_count = resp.getUser().getNoti_count();
            String first_name = resp.getUser().getFirst_name();
            String last_name = resp.getUser().getLast_name();
            String email_noti = resp.getUser().getEmail_notification();
            String aboutme = resp.getUser().getAboutme();


            addInfoDetail(id, context);
            addUserGroup(id, context);

            mView.setProfileData(email, username, first_name, last_name, email_noti, aboutme);
```

Gambar 5.23 Potongan Kode Program onResponse *Presenter* Fungsi Lihat Profile

Dan dalam fungsi lihat profile, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi data. Adapun potongan kode program salah satu service dengan nama *getKnowledgeArticDetail* yang dikonsumsi client adalah sebagai berikut :

```php
public function getUsrByIdDB($id)
{
    $sql = 'SELECT * FROM users WHERE ID = :id';
    $query = $this -> conn -> prepare($sql);
    $query -> execute(array(':id' => $id));
    $data = $query -> fetchObject();
    $db_encrypted_password = $data -> password;

    if ($query) {
        $user["id"] = $data -> ID;
        $user["email"] = $data -> email;
        $user["username"] = $data -> username;
        $user["user_role"] = $data -> user_role;
        $user["noti_count"] = $data -> noti_count;
        $user["first_name"] = $data -> first_name;
        $user["last_name"] = $data -> last_name;
        $user["aboutme"] = $data -> aboutme;
        $user["email_notification"] = $data -> email_notification;
        return $user;
    } else {
        return false;
    }
}
```

Gambar 5.24 Potongan Kode Program Service *getUsrByIdDB*

5.2.8 Implementasi Edit Profile

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah edit profile dengan tampilan seperti pada gambar 5.25 dibawah ini.

Gambar 5.25 User Interface Edit Profile

Pada fungsi edit profile, karena pemakaian viewpager pada tampilannya, maka class view dibagi menjadi 3 kelas turunan yang berfungsi untuk menginflate tiap fragment yang ada. Kode program *view* method showDialogConfirmEditProfile() di dalam *class* EditProfileActivity adalah sebagai berikut.

```
dialogConfirmEditProfile.show();

LinearLayout btnLanjut = (LinearLayout) dialogConfirmEditProfile.findViewById(R.id.btnKirimDialog);
btnLanjut.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        dialogConfirmEditProfile.dismiss();
        if(pref.getString(Constants.EMAIL, s1: "").equalsIgnoreCase(Constants.EMAILVP)){
            mPresenter.doEditProfile(pref.getString(Constants.ID, s1: ""), username, email, firstName,
                    lastName, nomor, unit, aboutMe, status, emailNoti, context);

        } else {
            Toast.makeText(context, Constants.EMAILVP, Toast.LENGTH_SHORT).show();
            mPresenter.doCekEmail(pref.getString(Constants.ID, s1: ""), username, email, firstName,
                    lastName, nomor, unit, aboutMe, status, emailNoti, context);
        }

    }
});
```

Gambar 5.26 Potongan Kode Program *View* Fungsi Edit Profile

Berikut potongan kode program *presenter* dalam method doEditProfile() yang ada di dalam *class* EditProfilePresenter pada gambar 5.27 dibawah.

```
@Override
public void doEditProfile(final String id, final String username, final String email,
                          final String firstName, final String lastName, final String nomor,
                          final String unit, final String aboutMe, final String status,
                          final String emailNoti, final Context context) {

    pref = PreferenceManager.getDefaultSharedPreferences(context);
    mView.showDialogLoading();

    RequestInterface requestInterface = ApiClient.getClient().create(RequestInterface.class);

    User user = new User();
    user.setId(id);
    user.setEmail(email);
    user.setFirst_name(firstName);
    user.setLast_name(lastName);
    user.setAboutme(aboutMe);
    user.setStatus(status);
    user.setUnit(unit);
    user.setTelepon(nomor);
    user.setEmail_notification(emailNoti);
    ServerRequest request = new ServerRequest();
    request.setOperation("updateInfoUser");
    request.setUser(user);
    Call<ServerResponse> response = requestInterface.operation(request);

    response.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final retrofit2.Response<ServerRespo


            final Handler handler = new Handler();
            handler.postDelayed(() -> {

                    mView.hideDialogLoading();
                    ServerResponse resp = response.body();

                    if (resp.getResult().equals(Constants.SUCCESS)) {

                        doUpdateProfile(id, context);
                        mView.showdialogSuksesEditProfile();
```

Gambar 5.27 Potongan Kode Program *Presenter* Fungsi Edit Profile

Dan dalam fungsi edit profile, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi data. Adapun potongan kode program salah satu service dengan nama *updateInfoUser* yang dikonsumsi client adalah sebagai berikut :

```
public function updateInfoUserDB($ID, $email, $first_name, $last_name, $aboutme,
$status, $unit, $telepon, $email_notification)
{
    $sql = 'UPDATE users SET email = :email, first_name
    = :first_name, last_name = :last_name, aboutme
    = :aboutme, email_notification = :email_notification WHERE ID = :ID';
    $query = $this -> conn -> prepare($sql);
    $query -> execute(array(':ID' => $ID, ':email' => $email, ':first_name' => $f

    if ($query) {
        $this -> doUpdateStatus($ID, $status);
        $this -> doUpdateUnit($ID, $unit);
        $this -> doUpdateTelepon($ID, $telepon);
        return true;
    } else {
        return false;
    }
}
```

Gambar 5.28 Potongan Kode Program Service *updateInfoUser*

5.2.9 Implementasi Edit Profile

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah edit profile dengan tampilan seperti pada gambar 5.29 dibawah ini.

Gambar 5.29 User Interface Edit Profile

Pada fungsi edit profile, karena pemakaian viewpager pada tampilannya, maka class view dibagi menjadi 3 kelas turunan yang berfungsi untuk menginflate tiap fragment yang ada. Kode program *view* method showDialogConfirmEditProfile() di dalam *class* EditProfileActivity adalah sebagai berikut.

```
dialogConfirmEditProfile.show();

LinearLayout btnLanjut = (LinearLayout) dialogConfirmEditProfile.findViewById(R.id.btnKirimDialog);
btnLanjut.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        dialogConfirmEditProfile.dismiss();
        if(pref.getString(Constants.EMAIL, s1: "").equalsIgnoreCase(Constants.EMAILVP)){
            mPresenter.doEditProfile(pref.getString(Constants.ID, s1: ""), username, email, firstName,
                    lastName, nomor, unit, aboutMe, status, emailNoti, context);

        } else {
            Toast.makeText(context, Constants.EMAILVP, Toast.LENGTH_SHORT).show();
            mPresenter.doCekEmail(pref.getString(Constants.ID, s1: ""), username, email, firstName,
                    lastName, nomor, unit, aboutMe, status, emailNoti, context);
        }

    }
});
```

Gambar 5.30 Potongan Kode Program *View* Fungsi Edit Profile

Berikut potongan kode program *presenter* dalam method doEditProfile() yang ada di dalam *class* EditProfilePresenter pada gambar 5.31 dibawah.

```
@Override
public void doEditProfile(final String id, final String username, final String email,
                          final String firstName, final String lastName, final String nomor,
                          final String unit, final String aboutMe, final String status,
                          final String emailNoti, final Context context) {

    pref = PreferenceManager.getDefaultSharedPreferences(context);
    mView.showDialogLoading();

    RequestInterface requestInterface = ApiClient.getClient().create(RequestInterface.class);

    User user = new User();
    user.setId(id);
    user.setEmail(email);
    user.setFirst_name(firstName);
    user.setLast_name(lastName);
    user.setAboutme(aboutMe);
    user.setStatus(status);
    user.setUnit(unit);
    user.setTelepon(nomor);
    user.setEmail_notification(emailNoti);
    ServerRequest request = new ServerRequest();
    request.setOperation("updateInfoUser");
    request.setUser(user);
    Call<ServerResponse> response = requestInterface.operation(request);

    response.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final retrofit2.Response<ServerResp


            final Handler handler = new Handler();
            handler.postDelayed(() -> {

                    mView.hideDialogLoading();
                    ServerResponse resp = response.body();

                    if (resp.getResult().equals(Constants.SUCCESS)) {

                        doUpdateProfile(id, context);
                        mView.showdialogSuksesEditProfile();
```

Gambar 5.31 Potongan Kode Program *Presenter* Fungsi Edit Profile

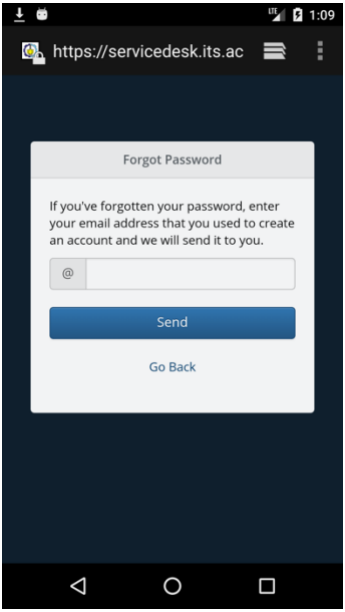
Dan dalam fungsi edit profile, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi data. Adapun potongan kode program salah satu service dengan nama *updateInfoUser* yang dikonsumsi client adalah sebagai berikut :

```
public function updateInfoUserDB($ID, $email, $first_name, $last_name, $aboutme,
$status, $unit, $telepon, $email_notification)
{
    $sql = 'UPDATE users SET email = :email, first_name
    = :first_name, last_name = :last_name, aboutme
    = :aboutme, email_notification = :email_notification WHERE ID = :ID';
    $query = $this -> conn -> prepare($sql);
    $query -> execute(array(':ID' => $ID, ':email' => $email, ':first_name' => $

    if ($query) {
        $this -> doUpdateStatus($ID, $status);
        $this -> doUpdateUnit($ID, $unit);
        $this -> doUpdateTelepon($ID, $telepon);
        return true;
    } else {
        return false;
    }
}
```

Gambar 5.32 Potongan Kode Program Service *updateInfoUser*

5.2.10 Implementasi Lupa Password

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah lupa password dengan tampilan seperti pada gambar 5.33 dibawah ini.



Gambar 5.33 User Interface Lupa Password

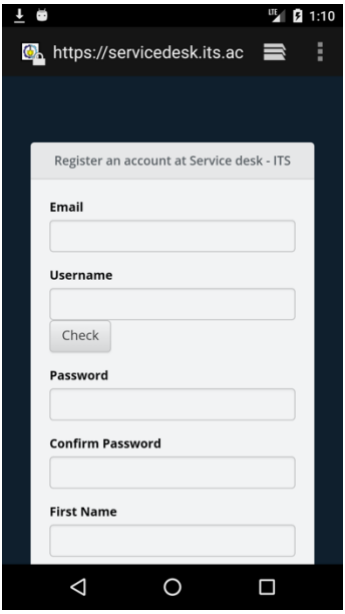
Fungsi lupa password hanya berfungsi sebagai pengirim *intent* untuk membuka link yang mengarah pada halaman lupa password *servicedesk.its.ac.id*. Oleh karena itu fungsi ini hanya memanfaatkan 1 class *view* dan 1 method intent yang dapat dilihat pada gambar 5.34 dibawah.

```
@Override
public void onClick(View v) {

    switch (v.getId()){

        case R.id.lupa_password:
            Intent viewIntent =
                    new Intent( action: "android.intent.action.VIEW",
                            Uri.parse("https://servicedesk.its.ac.id/index.php/login/forgotpw"));
            startActivity(viewIntent);
            break;
```

Gambar 5.34 Potongan Kode Program *View* Lupa Password

5.2.11 Implementasi Register Member

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah register member dengan tampilan seperti pada gambar 5.33 dibawah ini.



Gambar 5.35 User Interface Lupa Password

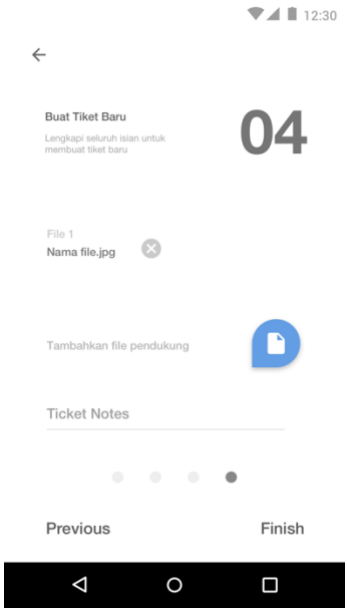
Sama seperti lupa password, fungsi register member hanya berfungsi sebagai pengirim *intent* untuk membuka link yang mengarah pada halaman lupa password *servicedesk.its.ac.id*. Oleh karena itu fungsi ini hanya memanfaatkan 1 class *view* dan 1 method intent yang dapat dilihat pada gambar 5.34 dibawah.

```
case R.id.tv_register:
    Intent viewIntents =
            new Intent( action: "android.intent.action.VIEW",
                    Uri.parse("https://servicedesk.its.ac.id/index.php/register"));
    startActivity(viewIntents);
    break;
```

Gambar 5.36 Potongan Kode Program *View* Lupa Password

5.2.12 Implementasi Buat Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah buat tiket dengan tampilan seperti pada gambar 5.35 dibawah ini.



Gambar 5.37 User Interface Buat Tiket

Pada fungsi buat tiket, karena pemakaian viewpager pada tampilannya, maka class view dibagi menjadi 4 kelas turunan yang berfungsi untuk menginflate tiap fragment yang ada. Kode program *view* method showDialogConfirmBuatTicket() di dalam *class* BuatTicketActivity adalah sebagai berikut.

```
if (Constants.valueUsernameStepSatu.contains("@")) {

    guest_email = Constants.valueUsernameStepSatu;
    guest_password = "1";

    if (status.equalsIgnoreCase( anotherString: "-1")) {
        status = "0";
    }

    mPresenter.doBuatTicket(view.getContext(), message, namafiles, subject,
            status, priority, notes, guest_email, guest_password, telepon, u

} else {

    if (status.equalsIgnoreCase( anotherString: "-1")) {
        status = "0";
    }

    mPresenter.doBuatTicket(view.getContext(), message, namafiles, subject,
            status, priority, notes, guest_email, guest_password, telepon, u

}
```

Gambar 5.38 Potongan Kode Program *View* Fungsi Buat Tiket

Berikut potongan kode program *presenter* dalam method doBuatTiket() yang ada di dalam *class* BuatTicketPresenter pada gambar 5.39 dibawah.

```
public void doBuatTicket(final Context context, final String message, final String namafiles,
                         final String subject, final String body, final String userid, final String

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Ticket ticket = new Ticket();
    ticket.setName(categories);
    ServerRequest request = new ServerRequest();
    request.setOperation("getCategoryID");
    request.setTicket(ticket);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            ServerResponse resp = response.body();
            doBuatTicketAll(context, message, namafiles, subject, body, userid, assignedid,
                    resp.getTicket().getId(), status, priority, notes, guest_email, guest_password,

        }

        @Override
        public void onFailure(Call<ServerResponse> call, Throwable t) {
            Log.e( tag: "Retrofit Get", t.toString());
        }

    });

}
```

Gambar 5.39 Potongan Kode Program *Presenter* Fungsi Buat Tiket

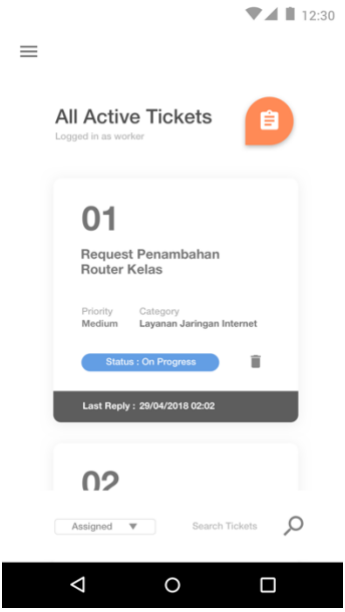
Dan dalam fungsi buat tiket, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi data. Adapun potongan kode program salah satu service dengan nama *getCategoryID* yang dikonsumsi client adalah sebagai berikut :

```php
public function getCategoryID($name)
{
    $sql = "SELECT * FROM ticket_categories WHERE name = '".$name."'";
    $query = $this -> conn -> prepare($sql);
    $query -> execute();
    $data = $query -> fetchObject();

    if ($query) {
        $user["id"] = $data -> ID;
        return $user;
    } else {
        return false;
    }
}
```

Gambar 5.40 Potongan Kode Program Service *getCategoryID*

5.2.13 Implementasi Lihat List Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah lihat list tiket dengan tampilan seperti pada gambar 5.41 dibawah ini.



Gambar 5.41 User Interface Lihat List Tiket

Adapun potongan kode *view* method onCreateView() di dalam *class* TicketFragment adalah sebagai berikut.

```
if (rolespref.equalsIgnoreCase( anotherString: "10")) {

    locationSpinTicket.setVisibility(GONE);
    labelLogin.setText("Logged in as member");
    loadSomeData();

    mPresenter.updateRecyclerTicketMemberAll(idpref);
```

Gambar 5.42 Potongan Kode Program *View* Fungsi Lihat List Tiket

Berikut potongan kode program *presenter* dalam method updateRecyclerTicketMemberAll() yang ada di dalam *class* TicketPresenter pada gambar 5.43 dibawah.

```
public void updateRecyclerTicketMemberAll(String userid) {

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Ticket ticket = new Ticket();
    ticket.setUserid(userid);
    ServerRequest request = new ServerRequest();
    request.setOperation("getTicketByYourAll");
    request.setTicket(ticket);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            final Handler handler = new Handler();
            handler.postDelayed(() -> {

                    if (response.body() != null) {
                        ticketList = response.body().getListTicket();

                        mAdapterTicket = new TicketAdapter(ticketList);
                        mView.setAdapterRecyclerTicket(mAdapterTicket);
                        mView.loadDone();
                    } else {

                        ticketList = Collections.emptyList();
                        mAdapterTicket = new TicketAdapter(ticketList);
                        mView.setAdapterRecyclerTicket(mAdapterTicket);
                        mView.loadDone();
                        mView.showNotFound();
```

Gambar 5.43 Potongan Kode Program *Presenter* Fungsi Lihat List Tiket

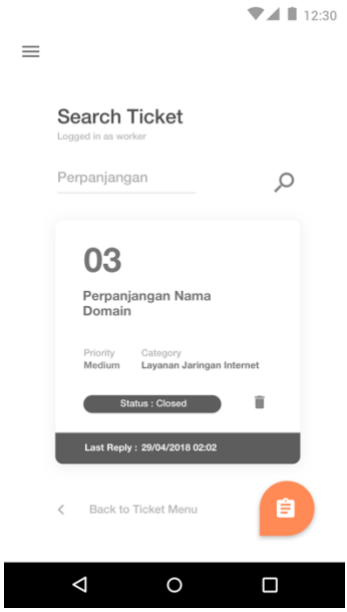
Dan dalam fungsi lihat list tiket, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *getTicketByYourAll* adalah sebagai berikut :

```
public function getTicketByYourAll($userid)
{
    $sql = "SELECT tickets.ID, tickets.title, tickets.userid, tickets.assigne
tickets.timestamp, tickets.categoryid, tickets.status,
tickets.priority, tickets.last_reply_timestamp,
tickets.last_reply_userid, tickets.message_id_hash,
tickets.guest_email,
users.username as client_username, users.avatar as client_avatar,
users.online_timestamp as client_online_timestamp,
users.first_name as client_first_name,
users.last_name as client_last_name,
users.email as client_email,
u2.username, u2.avatar, u2.online_timestamp, users.first_name,
users.last_name,
u3.username as lr_username, u3.avatar as lr_avatar,
u3.online_timestamp as lr_online_timestamp,
ticket_categories.name as cat_name, ticket_categories.ID as cat_id, ticket_ca

FROM tickets
LEFT OUTER JOIN users ON users.ID = tickets.userid
LEFT OUTER JOIN users as u2 ON u2.ID = tickets.assignedid
LEFT OUTER JOIN users as u3 ON u3.ID = tickets.last_reply_userid
INNER JOIN ticket_categories ON ticket_categories.ID = tickets.categoryid

WHERE tickets.userid = ".$userid;

    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    // query products
    $stmt = $query;
    $num = $stmt->rowCount();
```

Gambar 5.44 Potongan Kode Program Service *getTicketByYourAll*

5.2.14 Implementasi Search Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah search tiket dengan tampilan seperti pada gambar 5.41 dibawah ini.



Gambar 5.45 User Interface Search Tiket

Adapun potongan kode *view* method onClick() di dalam *class* SearchTicketFragment adalah sebagai berikut.

```
if (!et_keyword.getText().toString().isEmpty() && rolespref.equalsIgnoreCase( anotherString: "10")) {

    loadSomeData();
    mPresenter.updateRecyclerTicketMemberKeyword(idpref, et_keyword.getText().toString());

} else if (!et_keyword.getText().toString().isEmpty() && (rolespref.equalsIgnoreCase( anotherString: "8"))) {

    mPresenter.updateRecyclerTicketWorkerKeyword(catidpref, et_keyword.getText().toString());
    loadSomeData();

} else {

    dataNotComplete();

}
```

Gambar 5.46 Potongan Kode Program *View* Fungsi Search Tiket

Berikut potongan kode program *presenter* dalam method updateRecyclerTicketMemberKeyword() yang ada di dalam *class* SearchTicketPresenter pada gambar 5.47 dibawah.

```
public void updateRecyclerTicketMemberKeyword(String userid, String keyword) {

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Ticket ticket = new Ticket();
    ticket.setUserid(userid);
    ticket.setKeyword(keyword);
    ServerRequest request = new ServerRequest();
    request.setOperation("getTicketByKeywordAllMember");
    request.setTicket(ticket);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            final Handler handler = new Handler();
            handler.postDelayed(() -> {

                    if (response.body() != null) {
                        ticketList = response.body().getListTicket();

                        mAdapterTicket = new TicketAdapter(ticketList);
                        mView.loadDone();

                        mView.setAdapterRecyclerTicketSearch(mAdapterTicket);
                    } else {

                        ticketList = Collections.emptyList();
                        mAdapterTicket = new TicketAdapter(ticketList);
                        mView.loadDone();
                        mView.setAdapterRecyclerTicketSearch(mAdapterTicket);
                        mView.showNotFound();

                    }
```

Gambar 5.47 Potongan Kode Program *Presenter* Fungsi Search Tiket

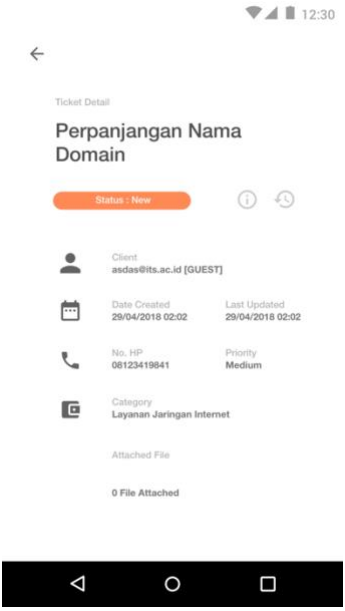
Dan dalam fungsi search tiket, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *getTicketByKeywordAllMember* adalah sebagai berikut :

```
public function getTicketByKeywordAllMember($keyword, $userid)
{
    $sql = "SELECT * FROM
(SELECT tickets.ID, tickets.title, tickets.userid, tickets.assignedid,
tickets.timestamp, tickets.categoryid, tickets.status,
tickets.priority, tickets.last_reply_timestamp,
tickets.last_reply_userid, tickets.message_id_hash,
tickets.guest_email,
users.username as client_username, users.avatar as client_avatar,
users.online_timestamp as client_online_timestamp,
users.first_name as client_first_name,
users.last_name as client_last_name,
users.email as client_email,
u2.username, u2.avatar, u2.online_timestamp, users.first_name,
users.last_name,
u3.username as lr_username, u3.avatar as lr_avatar,
u3.online_timestamp as lr_online_timestamp,
ticket_categories.name as cat_name, ticket_categories.ID as cat_id, ticket_ca

FROM tickets
LEFT OUTER JOIN users ON users.ID = tickets.userid
LEFT OUTER JOIN users as u2 ON u2.ID = tickets.assignedid
LEFT OUTER JOIN users as u3 ON u3.ID = tickets.last_reply_userid
INNER JOIN ticket_categories ON ticket_categories.ID = tickets.categoryid

WHERE tickets.title LIKE "."'%".$keyword."%'"."OR tickets.body LIKE"."'%".$ke
AS t
WHERE t.userid =".$userid;

    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    // query products
    $stmt = $query;
    $num = $stmt->rowCount();
```

Gambar 5.48 Potongan Kode Program Service *getTicketByKeywordAllMember*

5.2.15 Implementasi Lihat Detail Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah lihat detail tiket dengan tampilan seperti pada gambar 5.49 dibawah ini.



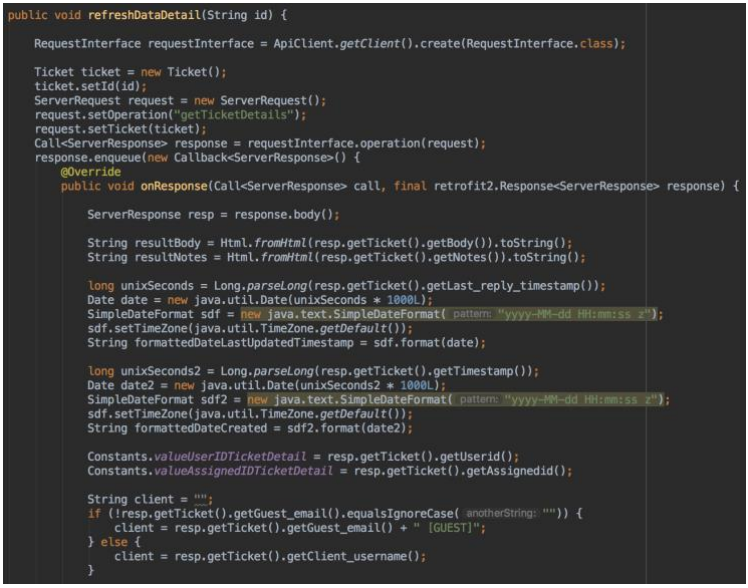
Gambar 5.49 User Interface Lihat Detail Tiket

Fungsi lihat detail tiket merupakan fungsi yang dapat diakses dengan melalui fungsi lihat list ticket ataupun check tiket terlebih dahulu. Adapun potongan kode *view* method onCreateView() di dalam *class* DetailTicketFragment adalah sebagai berikut.

```
String valueId = getArguments().getString( key: "id");
mPresenter.refreshDataDetail(valueId);
mPresenter.refreshDataInfo(valueId);
mPresenter.refreshDataFiles(valueId);
mPresenter.refreshDataReply(valueId);
```

Gambar 5.50 Potongan Kode Program *View* Fungsi Lihat Detail Tiket

Berikut potongan kode program *presenter* dalam method updateRecyclerTicketMemberKeyword() yang ada di dalam *class* DetailTicketPresenter pada gambar 5.51 dibawah.



```
public void refreshDataDetail(String id) {

    RequestInterface requestInterface = ApiClient.getClient().create(RequestInterface.class);

    Ticket ticket = new Ticket();
    ticket.setId(id);
    ServerRequest request = new ServerRequest();
    request.setOperation("getTicketDetails");
    request.setTicket(ticket);
    Call<ServerResponse> response = requestInterface.operation(request);
    response.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final retrofit2.Response<ServerResponse> response) {

            ServerResponse resp = response.body();

            String resultBody = Html.fromHtml(resp.getTicket().getBody()).toString();
            String resultNotes = Html.fromHtml(resp.getTicket().getNotes()).toString();

            long unixSeconds = Long.parseLong(resp.getTicket().getLast_reply_timestamp());
            Date date = new java.util.Date(unixSeconds * 1000L);
            SimpleDateFormat sdf = new java.text.SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss z");
            sdf.setTimeZone(java.util.TimeZone.getDefault());
            String formattedDateLastUpdatedTimestamp = sdf.format(date);

            long unixSeconds2 = Long.parseLong(resp.getTicket().getTimestamp());
            Date date2 = new java.util.Date(unixSeconds2 * 1000L);
            SimpleDateFormat sdf2 = new java.text.SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss z");
            sdf.setTimeZone(java.util.TimeZone.getDefault());
            String formattedDateCreated = sdf2.format(date2);

            Constants.valueUserIDTicketDetail = resp.getTicket().getUserid();
            Constants.valueAssignedIDTicketDetail = resp.getTicket().getAssignedid();

            String client = "";
            if (!resp.getTicket().getGuest_email().equalsIgnoreCase( anotherString: "")) {
                client = resp.getTicket().getGuest_email() + " [GUEST]";
            } else {
                client = resp.getTicket().getClient_username();
            }
```

Gambar 5.51 Potongan Kode Program *Presenter* Fungsi Lihat Detail Tiket

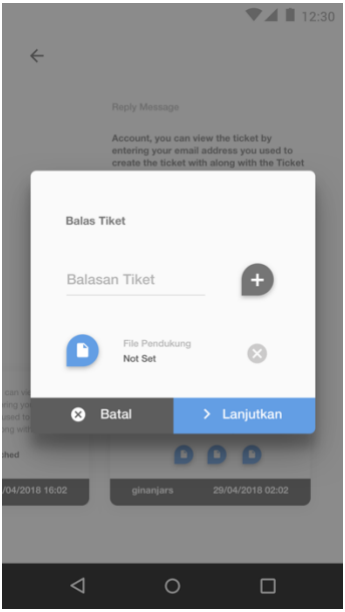
Dan dalam fungsi lihat detail tiket, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *getTicketDetails* adalah sebagai berikut :

```
public function getTicketDetails($ID)
{
    $sql = "SELECT tickets.ID, tickets.title, tickets.userid, tickets
            tickets.timestamp, tickets.categoryid, tickets.status,
            tickets.priority, tickets.last_reply_timestamp,
            tickets.last_reply_userid, tickets.message_id_hash,
            tickets.guest_email, tickets.guest_password,
            tickets.notes, tickets.body, tickets.rating,
            users.username as client_username, users.avatar as client
            users.online_timestamp as client_online_timestamp, users.
            client_email, users.email_notification as client_email_no
            users.first_name, users.last_name, users.id,
            u2.username, u2.avatar, u2.online_timestamp, u2.email,
            u2.email_notification,
            ticket_categories.name as cat_name, ticket_categories.cat

            FROM tickets
            LEFT OUTER JOIN users ON users.ID = tickets.userid
            LEFT OUTER JOIN users as u2 ON u2.ID = tickets.assignedid
            INNER JOIN ticket_categories ON ticket_categories.ID = ti
            WHERE tickets.ID = ".$ID;
    $query = $this -> conn -> prepare($sql);
    $query -> execute();
    $data = $query -> fetchObject();
```

Gambar 5.52 Potongan Kode Program Service *getTicketDetails*

5.2.16 Implementasi Balas Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah balas tiket dengan tampilan seperti pada gambar 5.53 dibawah ini.



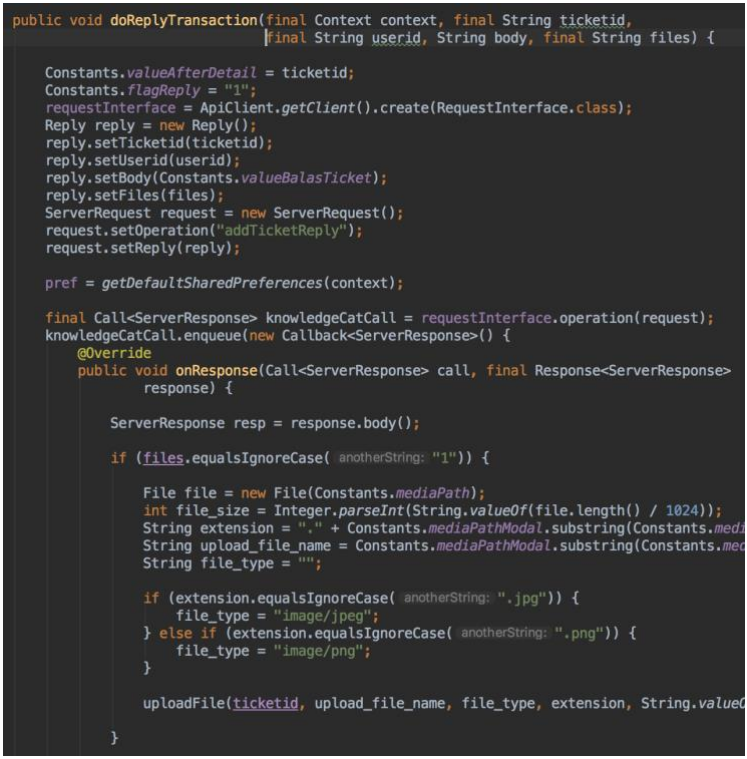
Gambar 5.53 User Interface Balas Tiket

Fungsi balas tiket merupakan fungsi yang dapat diakses dengan melalui fungsi lihat detail tiket terlebih dahulu. Adapun potongan kode *view* method showDialogConfirmBalasTicket() di dalam *class* DetailTicketFragment adalah sebagai berikut.

```
String filesNumber = "0";
if (Constants.mediaPathModal.equalsIgnoreCase( anotherString: "Not Set")) {
    filesNumber = "0";
} else {
    filesNumber = "1";
}

dialogConfirmBalasTicket.dismiss();

String useridFix = "0";
if (pref.getBoolean(Constants.IS_LOGGED_IN,  b: false)) {
    useridFix = pref.getString(Constants.ID,  s1: "");
}

mPresenter.doReplyTransaction(view.getContext(), ticketid, useridFix, Cor
```

Gambar 5.54 Potongan Kode Program *View* Fungsi Balas Tiket

Berikut potongan kode program *presenter* dalam method doReplyTransaction() yang ada di dalam *class* DetailTicketPresenter pada gambar 5.55 dibawah.



```
public void doReplyTransaction(final Context context, final String ticketid,
                               final String userid, String body, final String files) {

    Constants.valueAfterDetail = ticketid;
    Constants.flagReply = "1";
    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Reply reply = new Reply();
    reply.setTicketid(ticketid);
    reply.setUserid(userid);
    reply.setBody(Constants.valueBalasTicket);
    reply.setFiles(files);
    ServerRequest request = new ServerRequest();
    request.setOperation("addTicketReply");
    request.setReply(reply);

    pref = getDefaultSharedPreferences(context);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            ServerResponse resp = response.body();

            if (files.equalsIgnoreCase( anotherString: "1")) {

                File file = new File(Constants.mediaPath);
                int file_size = Integer.parseInt(String.valueOf(file.length() / 1024));
                String extension = "." + Constants.mediaPathModal.substring(Constants.medi
                String upload_file_name = Constants.mediaPathModal.substring(Constants.med
                String file_type = "";

                if (extension.equalsIgnoreCase( anotherString: ".jpg")) {
                    file_type = "image/jpeg";
                } else if (extension.equalsIgnoreCase( anotherString: ".png")) {
                    file_type = "image/png";
                }

                uploadFile(ticketid, upload_file_name, file_type, extension, String.valueO

            }
```

Gambar 5.55 Potongan Kode Program *Presenter* Fungsi Balas Tiket

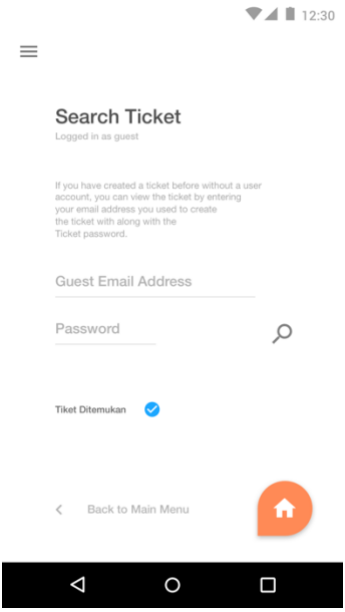
Dan dalam fungsi balas tiket, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *addTicketReply* adalah sebagai berikut :

```
public function addTicketReply($ticketid, $userid, $body, $files)
{
    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "db_etiket";

    $message_id_hash = md5(rand(1, 100000) . "fhhfh" . time());

    $coba = new Common();

    try {
        $conns = new PDO("mysql:host=$servername;dbname=$dbname",
        // set the PDO error mode to exception
        $conns->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEP
        $sql = "INSERT INTO ticket_replies SET ticketid = '".$tick
        // use exec() because no results are returned
        $conns->exec($sql);
        $last_insert_id = $conns->lastInsertId();
        $reply["last_insert_id"] = $last_insert_id;
        return $reply;
    } catch (PDOException $e) {
        echo $sql . "<br>" . $e->getMessage();
    }
}
```

Gambar 5.56 Potongan Kode Program Service *addTicketReply*

5.2.17 Implementasi Check Tiket (Guest)

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah check tiket (guest) dengan tampilan seperti pada gambar 5.57 dibawah ini.



Gambar 5.57 User Interface Check Ticket (Guest)

Potongan kode *view* method onClick() di dalam *class* CheckTicketFragment adalah sebagai berikut.

```
case R.id.btnSearchTicketSearch:
    if(Constants.connect == 0){

        networkUnavailable();

    } else {

        if (!et_passCheckTicket.getText().toString().isEmpty()
                && !et_guestEmailCheckTicket.getText().toString().isEmpty()) {

            if (counterClick == 0) {

                showLoading();

                final Handler handler = new Handler();
                handler.postDelayed(() -> {

                        mPresenter.doCheckTicket(et_guestEmailCheckTicket.getText().toString(),
                                et_passCheckTicket.getText().toString());

                }, delayMillis: 1250);
            }


        } else {

            isianNotCompleted();

        }

    }
    break;
```

Gambar 5.58 Potongan Kode Program *View* Fungsi Check Tiket (Guest)

Berikut potongan kode program *presenter* dalam method doCheckTicket() yang ada di dalam *class* SearchTicketPresenter pada gambar 5.59.

```
public void doCheckTicket(String email, String password) {


    RequestInterface requestInterface = ApiClient.getClient().create(RequestInterface.class);

    Ticket ticket = new Ticket();
    ticket.setEmail(email);
    ticket.setPassword(password);
    ServerRequest request = new ServerRequest();
    request.setOperation("getTicketGuestDetails");
    request.setTicket(ticket);
    Call<ServerResponse> response = requestInterface.operation(request);

    response.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final retrofit2.Response<ServerResponse>

            ServerResponse resp = response.body();

            mView.hideLoading();
            if (resp.getResult().toString().equalsIgnoreCase(Constants.SUCCESS)) {

                final Bundle args = new Bundle();
                args.putString("id", resp.getTicket().getId());

                doAddHistory(resp.getTicket().getId());
                mView.showDitemukan(args);

            } else {

                mView.showTidakDitemukan();

            }

        }
```

Gambar 5.59 Potongan Kode Program *Presenter* Fungsi Check Tiket

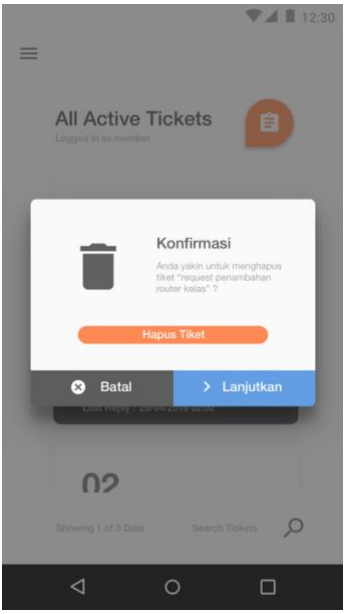
Dan dalam fungsi check tiket, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *getTicketGuestDetails* adalah sebagai berikut :

```
public function getTicketGuestDetails($guest_email, $guest_password)
{
    $sql = "SELECT tickets.ID, tickets.title, tickets.userid, ticket
tickets.timestamp, tickets.categoryid, tickets.status,
tickets.priority, tickets.last_reply_timestamp,
tickets.last_reply_userid, tickets.message_id_hash,
tickets.guest_email, tickets.notes, tickets.body,
users.username as client_username, users.avatar as client_avatar,
users.online_timestamp as client_online_timestamp, users.email as
client_email, users.first_name, users.last_name,
u2.username, u2.avatar, u2.online_timestamp,
ticket_categories.name as cat_name, ticket_categories.cat_parent

FROM tickets
LEFT OUTER JOIN users ON users.ID = tickets.userid
LEFT OUTER JOIN users as u2 ON u2.ID = tickets.assignedid
INNER JOIN ticket_categories ON ticket_categories.ID = tickets.categ

WHERE tickets.guest_email = '".$guest_email."' AND tickets.guest_pas
    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    $stmt = $query;
    $num = $stmt->rowCount();
```

Gambar 5.60 Potongan Kode Program Service *getTicketGuestDetails*

5.2.18 Implementasi Hapus Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah hapus tiket dengan tampilan seperti pada gambar 5.61 dibawah ini.



Gambar 5.61 User Interface Hapus Ticket

Potongan kode *view* method showDialogConfirmHapus() di dalam *class* TicketFragment adalah sebagai berikut.

```
LinearLayout btnLanjut = (LinearLayout) dialogConfirmBuatTiket.f
btnLanjut.setOnClickListener((view) → {

        mPresenter.doDeleteTicket(ticketid);
        dialogConfirmBuatTiket.dismiss();

});
LinearLayout btnCancel = (LinearLayout) dialogConfirmBuatTiket.f
btnCancel.setOnClickListener((view) → {
        dialogConfirmBuatTiket.dismiss();

});
```

Gambar 5.62 Potongan Kode Program *View* Fungsi Hapus Tiket

Berikut potongan kode program *presenter* dalam method doDeleteTicket() yang ada di dalam *class* TicketPresenter pada gambar 5.63 dibawah.

```
public void doDeleteTicket(String ticketid){

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Ticket ticket = new Ticket();
    ticket.setId(ticketid);
    ServerRequest request = new ServerRequest();
    request.setOperation("deleteTicketSD");
    request.setTicket(ticket);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            mView.showdialogSuksesDelete();
            Log.d( tag: "Retrofit Get History",  msg: "Delete ticket successful");
        }

        @Override
        public void onFailure(Call<ServerResponse> call, Throwable t) {
            Log.e( tag: "Retrofit Get History", t.toString());
        }

    });

}
```

Gambar 5.63 Potongan Kode Program *Presenter* Fungsi Hapus Tiket

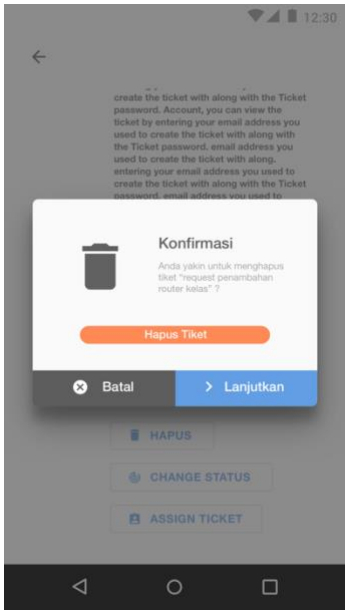
Dan dalam fungsi hapus tiket, aplikasi melakukan request terhadap satu service dalam melakukan transaksi. Potongan kode program service *deleteTicketSD* adalah sebagai berikut :

```php
public function deleteTicketSD($id)
{
    $sql = "DELETE FROM tickets WHERE ID = '".$id."'";
    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    if ($query) {
        return true;
    } else {
        return false;
    }
}
```

Gambar 5.64 Potongan Kode Program Service *deleteTicketSD*

5.2.19 Implementasi Hapus Tiket Via Detail

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah hapus tiket via detail dengan tampilan seperti pada gambar 5.64 dibawah ini.



Gambar 5.65 User Interface Hapus Tiket

Potongan kode *view* method showDialogConfirmHapus() di dalam *class* DetailTicketFragment adalah sebagai berikut.

```
LinearLayout btnLanjut = (LinearLayout) dialogConfirmBuatTiket.
btnLanjut.setOnClickListener((view) → {

        mPresenter.doDeleteTicket(ticketid);
        dialogConfirmBuatTiket.dismiss();

});
LinearLayout btnCancel = (LinearLayout) dialogConfirmBuatTiket.
btnCancel.setOnClickListener((view) → {
        dialogConfirmBuatTiket.dismiss();

});
```

Gambar 5.66 Potongan Kode Program *View* Fungsi Hapus Tiket Via Detail

Berikut potongan kode program *presenter* dalam method doDeleteTicket() yang ada di dalam *class* DetailTicketPresenter pada gambar 5.67 dibawah.

```
public void doDeleteTicket(String ticketid){

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Ticket ticket = new Ticket();
    ticket.setId(ticketid);
    ServerRequest request = new ServerRequest();
    request.setOperation("deleteTicketSD");
    request.setTicket(ticket);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            mView.showdialogSuksesDelete();
            Log.d( tag: "Retrofit Get History",  msg: "Delete ticket successful");
        }

        @Override
        public void onFailure(Call<ServerResponse> call, Throwable t) {
            Log.e( tag: "Retrofit Get History", t.toString());
        }

    });
}
```

Gambar 5.67 Potongan Kode Program *Presenter* Fungsi Hapus Tiket Via Detail

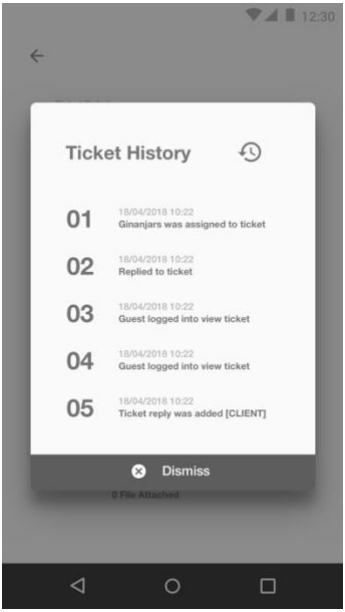
Dan dalam fungsi hapus tiket, aplikasi melakukan request terhadap satu service dalam melakukan transaksi. Potongan kode program service *deleteTicketSD* adalah sebagai berikut :

```
public function deleteTicketSD($id)
{
    $sql = "DELETE FROM tickets WHERE ID = '".$id."'";
    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    if ($query) {
        return true;
    } else {
        return false;
    }
}
```

Gambar 5.68 Potongan Kode Program Service *deleteTicketSD*

5.2.20 Implementasi Lihat History Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah lihat history tiket dengan tampilan seperti pada gambar 5.69 dibawah ini.



Gambar 5.69 User Interface Lihat History Tiket

Potongan kode *view* method showDialogHistoryTicket() di dalam *class* DetailTicketFragment adalah sebagai berikut.

```
public void showDialogHistoryTicket(RecyclerView.Adapter adapters) {


    int widthinPixels = (int) 150dp;
    int heightinPixels = (int) 210dp;

    final Dialog dialogHistoryTicket = CDialog.createDialog(getActivity(), R.lay

    recyclerHistory = (RecyclerView) dialogHistoryTicket.findViewById(R.id.recyc
    recyclerHistory.setHasFixedSize(true);
    RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(dialogHi
    recyclerHistory.setLayoutManager(mLayoutManager);

    recyclerHistory.setAdapter(adapters);

    dialogHistoryTicket.show();

    LinearLayout btnDismiss = (LinearLayout) dialogHistoryTicket.findViewById(R.
    btnDismiss.setOnClickListener((view) → { dialogHistoryTicket.dismiss(); });

}
```

Gambar 5.70 Potongan Kode Program *View* Fungsi Lihat History Tiket

Berikut potongan kode program *presenter* dalam method refreshDataHistory() yang ada di dalam *class* DetailTicketPresenter pada gambar 5.71 dibawah.

```
public void refreshDataHistory(String id) {

    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Ticket ticket = new Ticket();
    ticket.setTicketid(id);
    ServerRequest request = new ServerRequest();
    request.setOperation("getTicketHistory");
    request.setTicket(ticket);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, Response<ServerResponse>
                response) {

            if (response.body() != null) {
                ticketHistoryList = response.body().getListTicketHistories();

                mAdapterHistoryList = new TicketHistoryAdapter(ticketHistoryList);
                mView.showDialogHistoryTicket(mAdapterHistoryList);

            } else {

                ticketHistoryList = Collections.emptyList();

                mAdapterHistoryList = new TicketHistoryAdapter(ticketHistoryList);
                mView.showDialogHistoryTicket(mAdapterHistoryList);

            }

        }
```

Gambar 5.71 Potongan Kode Program *Presenter* Fungsi Lihat History Tiket

Dan dalam fungsi lihat history tiket, aplikasi melakukan request terhadap satu service dalam melakukan transaksi. Potongan kode program service *getTicketHistory* adalah sebagai berikut :

```
public function getTicketHistory($id)
{
    $sql = "SELECT ticket_history.ID, ticket_history.message,
ticket_history.userid, ticket_history.ticketid,
ticket_history.timestamp,
users.avatar, users.username, users.online_timestamp

FROM ticket_history
LEFT OUTER JOIN users ON users.ID = ticket_history.userid
WHERE ticket_history.ticketid = ".$id."
LIMIT 5";

    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    // query products
    $stmt = $query;
    $num = $stmt->rowCount();
```

Gambar 5.72 Potongan Kode Program Service *getTicketHistory*

5.2.21 Implementasi Lihat Data Client

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah lihat data client dengan tampilan seperti pada gambar 5.73 dibawah ini

Gambar 5.73 User Interface Lihat Data Client

Fungsi lihat data client memiliki mekanisme apabila email guest terdeteksi pada alamat email tiket, maka sisa data client yang lain tidak akan diambil dan langsung menjalankan method showDialogDataClient() tanpa melewati presenter. Potongan kode *view* method showDialogDataClient() di dalam *class* DetailTicketFragment adalah sebagai berikut.

```
public void showDialogDataClient(String username, String email, String nama,

    int widthinPixels = (int) 150dp;
    int heightinPixels = (int) 210dp;

    final Dialog dialogDataClient = CDialog.createDialog(getActivity(), R.lay

    TextView txtUsernameDataClient, txtEmailDataClient, txtNamaDataClient, tx
            txtNomorTeleponDataClient, txtUnitDataClient;

    txtUsernameDataClient = (TextView) dialogDataClient.findViewById(R.id.txt
    txtEmailDataClient = (TextView) dialogDataClient.findViewById(R.id.txtEma
    txtNamaDataClient = (TextView) dialogDataClient.findViewById(R.id.txtNama
    txtStatusDataClient = (TextView) dialogDataClient.findViewById(R.id.txtSt
    txtNomorTeleponDataClient = (TextView) dialogDataClient.findViewById(R.id
    txtUnitDataClient = (TextView) dialogDataClient.findViewById(R.id.txtUnit

    txtUsernameDataClient.setText(username);
    txtEmailDataClient.setText(email);
    txtNamaDataClient.setText(nama);
    txtStatusDataClient.setText(status);
    txtNomorTeleponDataClient.setText(nomor);
    txtUnitDataClient.setText(unit);

    dialogDataClient.show();
```

Gambar 5.74 Potongan Kode Program *View* Fungsi Lihat Data Client

Berikut potongan kode program *presenter* dalam method doUpdateInfoClient() yang ada di dalam *class* DetailTicketPresenter pada gambar 5.75 dibawah.

```
public void doUpdateInfoClient(String id) {

    RequestInterface requestInterface = ApiClient.getClient().create(RequestInterface.class);

    Ticket ticket = new Ticket();
    ticket.setId(id);
    ServerRequest request = new ServerRequest();
    request.setOperation("getTicketDetails");
    request.setTicket(ticket);
    Call<ServerResponse> response = requestInterface.operation(request);

    response.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final retrofit2.Response<ServerResp

            ServerResponse resp = response.body();

            String username = resp.getTicket().getClient_username();
            String name = resp.getTicket().getFirst_name() + " " + resp.getTicket().getLast_n
            String email = resp.getTicket().getClient_email();

            addInfoDetail(resp.getTicket().getUserid(), username, name, email);

        }
```

Gambar 5.75 Potongan Kode Program *Presenter* Fungsi Lihat Data Client

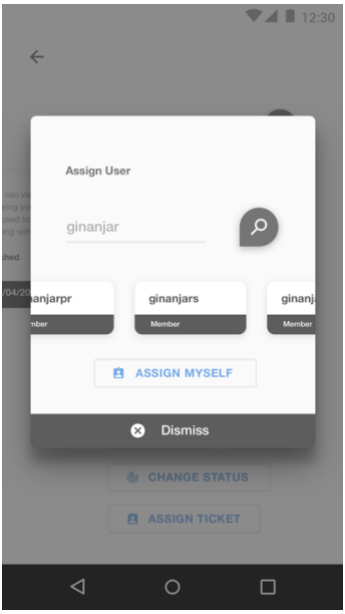
Dan dalam fungsi lihat data client, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *getTicketDetails* adalah sebagai berikut :

```
public function getTicketDetails($ID)
{
    $sql = "SELECT tickets.ID, tickets.title, tickets.userid, t
            tickets.timestamp, tickets.categoryid, tickets.stat
            tickets.priority, tickets.last_reply_timestamp,
            tickets.last_reply_userid, tickets.message_id_hash,
            tickets.guest_email, tickets.guest_password,
            tickets.notes, tickets.body, tickets.rating,
            users.username as client_username, users.avatar as
            users.online_timestamp as client_online_timestamp,
            client_email, users.email_notification as client_em
            users.first_name, users.last_name, users.id,
            u2.username, u2.avatar, u2.online_timestamp, u2.ema
            u2.email_notification,
            ticket_categories.name as cat_name, ticket_categori

            FROM tickets
            LEFT OUTER JOIN users ON users.ID = tickets.userid
            LEFT OUTER JOIN users as u2 ON u2.ID = tickets.assi
            INNER JOIN ticket_categories ON ticket_categories.I
            WHERE tickets.ID = ".$ID;
    $query = $this -> conn -> prepare($sql);
    $query -> execute();
    $data = $query -> fetchObject();
```

Gambar 5.76 Potongan Kode Program Service *getTicketDetails*

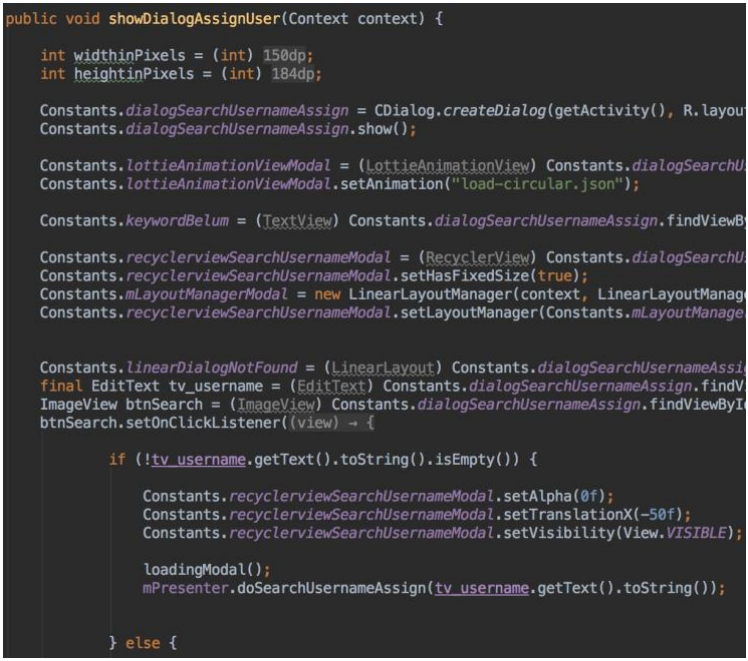
5.2.22 Implementasi Assign Tiket / MySelf

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah assign tiket / myself dengan tampilan seperti pada gambar 5.77 dibawah ini



Gambar 5.77 User Interface Assign Tiket

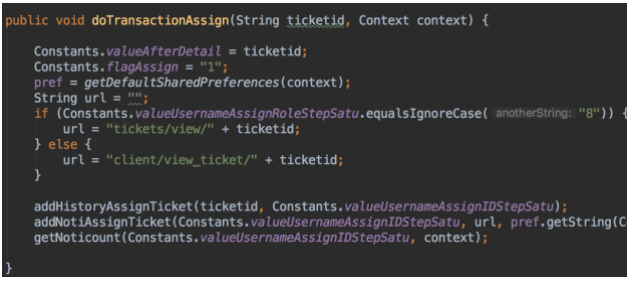
Pada fungsi assign tiket, user diharuskan memilih untuk melakukan assign pada user lain atau assign pada akunnya sendiri. Terdapat search username pula pada fungsi assign tiket ini. Potongan kode *view* method showDialogAssignUser() di dalam *class* DetailTicketFragment adalah sebagai berikut.



```
public void showDialogAssignUser(Context context) {

    int widthinPixels = (int) 150dp;
    int heightinPixels = (int) 184dp;

    Constants.dialogSearchUsernameAssign = CDialog.createDialog(getActivity(), R.layou
    Constants.dialogSearchUsernameAssign.show();

    Constants.lottieAnimationViewModal = (LottieAnimationView) Constants.dialogSearchU
    Constants.lottieAnimationViewModal.setAnimation("load-circular.json");

    Constants.keywordBelum = (TextView) Constants.dialogSearchUsernameAssign.findViewB

    Constants.recyclerviewSearchUsernameModal = (RecyclerView) Constants.dialogSearchU
    Constants.recyclerviewSearchUsernameModal.setHasFixedSize(true);
    Constants.mLayoutManagerModal = new LinearLayoutManager(context, LinearLayoutManag
    Constants.recyclerviewSearchUsernameModal.setLayoutManager(Constants.mLayoutManage


    Constants.linearDialogNotFound = (LinearLayout) Constants.dialogSearchUsernameAssi
    final EditText tv_username = (EditText) Constants.dialogSearchUsernameAssign.findV
    ImageView btnSearch = (ImageView) Constants.dialogSearchUsernameAssign.findViewById
    btnSearch.setOnClickListener((view) → {

        if (!tv_username.getText().toString().isEmpty()) {

            Constants.recyclerviewSearchUsernameModal.setAlpha(0f);
            Constants.recyclerviewSearchUsernameModal.setTranslationX(-50f);
            Constants.recyclerviewSearchUsernameModal.setVisibility(View.VISIBLE);

            loadingModal();
            mPresenter.doSearchUsernameAssign(tv_username.getText().toString());


        } else {
```

Gambar 5.78 Potongan Kode Program *View* Fungsi Assign User / Myself

Berikut potongan kode program *presenter* dalam method doTransactionAssign() yang ada di dalam *class* DetailTicketPresenter pada gambar 5.79 dibawah.



```
public void doTransactionAssign(String ticketid, Context context) {

    Constants.valueAfterDetail = ticketid;
    Constants.flagAssign = "1";
    pref = getDefaultSharedPreferences(context);
    String url = "";
    if (Constants.valueUsernameAssignRoleStepSatu.equalsIgnoreCase( anotherString: "8")) {
        url = "tickets/view/" + ticketid;
    } else {
        url = "client/view_ticket/" + ticketid;
    }

    addHistoryAssignTicket(ticketid, Constants.valueUsernameAssignIDStepSatu);
    addNotiAssignTicket(Constants.valueUsernameAssignIDStepSatu, url, pref.getString(C
    getNoticount(Constants.valueUsernameAssignIDStepSatu, context);

}
```

Gambar 5.79 Potongan Kode Program *Presenter* Fungsi Assign Tiket / Myself

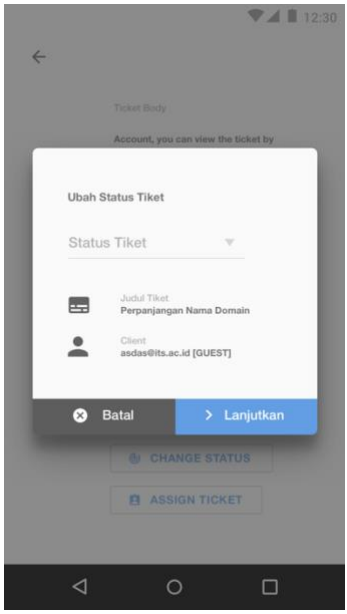
Dan dalam fungsi lihat data client, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *addHistoryAssignTicket* adalah sebagai berikut :

```
public function addHistoryAssignTicket($ticketid, $userid)
{
    $sql = "INSERT INTO ticket_history SET
ticketid ='".$ticketid."', userid = '".$userid."',  message =
    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    if ($query) {
        return true;
    } else {
        return false;
    }
}
```

Gambar 5.80 Potongan Kode Program Service *addHistoryAssignTicket*

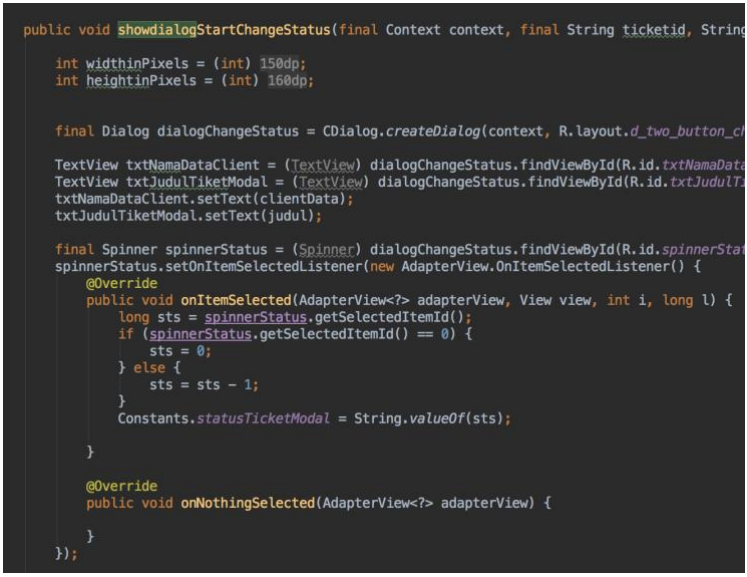
5.2.23 Implementasi Ubah Status Tiket

Fungsi selanjutnya yang diimplementasikan sesuai desain adalah ubah status tiket dengan tampilan seperti pada gambar 5.81 dibawah ini



Gambar 5.81 User Interface Ubah Status Tiket

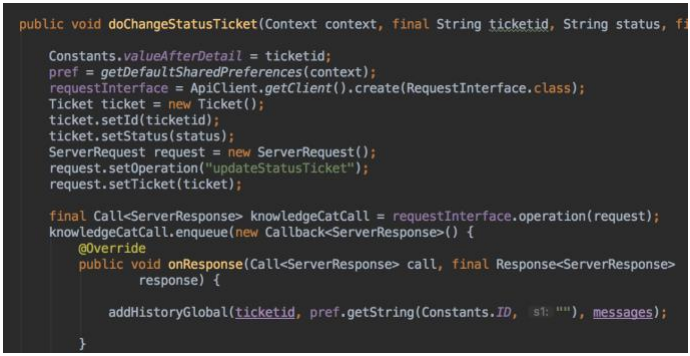
Potongan kode *view* method showDialogStartChangeStatus() di dalam *class* DetailTicketFragment adalah sebagai berikut.



```
public void showdialogStartChangeStatus(final Context context, final String ticketid, String

    int widthinPixels = (int) 150dp;
    int heightinPixels = (int) 160dp;


    final Dialog dialogChangeStatus = CDialog.createDialog(context, R.layout.d_two_button_c

    TextView txtNamaDataClient = (TextView) dialogChangeStatus.findViewById(R.id.txtNamaData
    TextView txtJudulTiketModal = (TextView) dialogChangeStatus.findViewById(R.id.txtJudulT
    txtNamaDataClient.setText(clientData);
    txtJudulTiketModal.setText(judul);

    final Spinner spinnerStatus = (Spinner) dialogChangeStatus.findViewById(R.id.spinnerSta
    spinnerStatus.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
            long sts = spinnerStatus.getSelectedItemId();
            if (spinnerStatus.getSelectedItemId() == 0) {
                sts = 0;
            } else {
                sts = sts - 1;
            }
            Constants.statusTicketModal = String.valueOf(sts);

        }

        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {

        }
    });
```

Gambar 5.82 Potongan Kode Program *View* Fungsi Change Status Tiket

Berikut potongan kode program *presenter* dalam method doChangeStatusTicket() yang ada di dalam *class* DetailTicketPresenter pada gambar 5.83 dibawah.



```
public void doChangeStatusTicket(Context context, final String ticketid, String status, f

    Constants.valueAfterDetail = ticketid;
    pref = getDefaultSharedPreferences(context);
    requestInterface = ApiClient.getClient().create(RequestInterface.class);
    Ticket ticket = new Ticket();
    ticket.setId(ticketid);
    ticket.setStatus(status);
    ServerRequest request = new ServerRequest();
    request.setOperation("updateStatusTicket");
    request.setTicket(ticket);

    final Call<ServerResponse> knowledgeCatCall = requestInterface.operation(request);
    knowledgeCatCall.enqueue(new Callback<ServerResponse>() {
        @Override
        public void onResponse(Call<ServerResponse> call, final Response<ServerResponse>
                response) {

            addHistoryGlobal(ticketid, pref.getString(Constants.ID,  s1: ""), messages);

        }
```

Gambar 5.83 Potongan Kode Program *Presenter* Fungsi Ubah Status Tiket

Dan dalam fungsi lihat data client, aplikasi melakukan request terhadap beberapa service dalam melakukan transaksi. Potongan kode program salah satu service yaitu *updateStatusTicket* adalah sebagai berikut :

```
public function updateStatusTicket($id, $status)
{
    $date = "";
    if ($status == 2) {
        $date = date("d-n-Y");
    }
    $sql = "UPDATE tickets SET close_ticket_date = '"
    .$date."', status = '".$status."' WHERE ID = ".$id;
    $query = $this -> conn -> prepare($sql);
    $query -> execute();

    if ($query) {
        return true;
    } else {
        return false;
    }
}
```

Gambar 5.84 Potongan Kode Program Service *updateStatusTicket*

5.3 Implementasi Test Case

Metode yang dilakukan pada pengujian aplikasi mobile *service desk* adalah metode *blacbox testing*, yang artinya *testing* akan berfokus pada kesesuaian jalannya interaksi aplikasi yang telah dikembangkan dengan use case story yang telah dibuat. Dalam pelaksanaan *testing*, *testcase* yang dijalankan akan disesuaikan dengan use case dari 2 aktor yakni user (member dan non member) dan ticket worker. Testing dilakukan ketika aplikasi dan web service telah selesai dikembangkan sesuai kebutuhan fungsional. Dengan dilakukannya blackbox testing, dapat dipastikan seluruh fungsional aplikasi berjalan dengan baik sesuai dengan kebutuhan fungsional yang telah disepakati. Berikut merupakan hasil *testing* role user (member dan non

member) yang dilakukan pada aplikasi mobile *service desk* yang ditampilkan pada tabel 5.3.

Tabel 5.3 Hasil Testing Role User (Member dan Non Member)

Use Case	Identifikasi Pengujian	Skenario	Hasil
U-01	UCT001	Normal	Passed
	UCT002	Alternatif 1	Passed
	UCT003	Alternatif 2	Passed
	UCT004	Alternatif 3	Passed
	UCT005	Alternatif 4	Passed
U-02	UCT006	Normal	Passed
U-03	UCT007	Normal	Passed
U-04	UCT008	Normal	Passed
	UCT009	Alternatif 1	Passed
U-05	UCT010	Normal	Passed
	UCT011	Alternatif 1	Passed
U-06	UCT012	Normal	Passed
U-07	UCT013	Normal	Passed
U-08	UCT014	Normal	Passed
	UCT015	Alternatif 1	Passed
U-09	UCT016	Normal	Passed
U-10	UCT017	Normal	Passed
U-11	UCT018	Normal	Passed
U-12	UCT019	Normal	Passed
	UCT020	Alternatif 1	Passed
U-13	UCT021	Normal	Passed
U-14	UCT022	Normal	Passed
	UCT023	Alternatif 1	Passed
U-15	UCT024	Normal	Passed
	UCT025	Alternatif 1	Passed
U-16	UCT026	Normal	Passed
U-17	UCT027	Normal	Passed
	UCT028	Alternatif 1	Passed
U-18	UCT029	Normal	Passed

Sedangkan berikut merupakan hasil testing role ticket worker yang dilakukan pada aplikasi mobile service desk yang dapat dilihat pada tabel 5.4 dibawah ini.

Tabel 5.4 Hasil Testing Role Ticket Worker

Use Case	Identifikasi Pengujian	Skenario	Hasil
T-01	TCT001	Normal	Passed
	TCT002	Alternatif 1	Passed
	TCT003	Alternatif 2	Passed
	TCT004	Alternatif 3	Passed
	TCT005	Alternatif 4	Passed
T-02	TCT006	Normal	Passed
T-03	TCT007	Normal	Passed
T-04	TCT008	Normal	Passed
	TCT009	Alternatif 1	Passed
T-05	TCT010	Normal	Passed
	TCT011	Alternatif 1	Passed
T-06	TCT012	Normal	Passed
T-07	TCT013	Normal	Passed
T-08	TCT014	Normal	Passed
	TCT015	Alternatif 1	Passed
T-09	TCT016	Normal	Passed
T-10	TCT017	Normal	Passed
T-11	TCT018	Normal	Passed
T-12	TCT019	Normal	Passed
	TCT020	Alternatif 1	Passed
T-13	TCT021	Normal	Passed
T-14	TCT022	Normal	Passed
	TCT023	Alternatif 1	Passed
T-15	TCT024	Normal	Passed
	TCT025	Alternatif 1	Passed
T-16	TCT026	Normal	Passed
T-17	TCT027	Normal	Passed

T-18	TCT028	Normal	Passed
T-19	TCT029	Normal	Passed
T-20	TCT030	Normal	Passed
T-21	TCT031	Normal	Passed
	TCT032	Alternatif 1	Passed
T-22	TCT033	Normal	Passed
T-23	TCT034	Normal	Passed

5.4 Implementasi Deployment Cloud

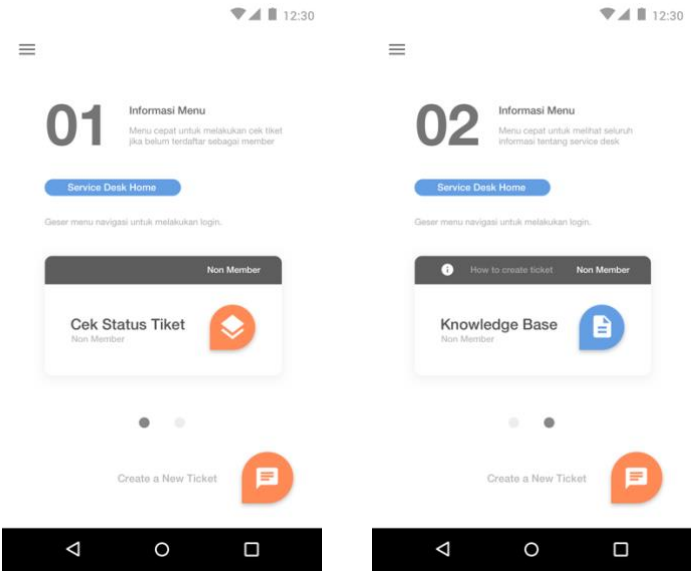
Tahapan ini merupakan tahapan akhir dari pengembangan aplikasi. Setelah *testing* dilakukan dan memenuhi seluruh output ekpektasi, maka dilakukan pemindahan web service dari lingkungan pengembangan *local* ke dalam *cloud*. Web service di deploy dalam bentuk folder dengan nama *sdservice* yang nantinya juga menjadi alamat akses uri yang di set pada aplikasi. Dalam proses deploy web service juga dilakukan beberapa konfigurasi pada server sehingga aplikasi dapat menjalankan fungsinya secara penuh. Konfigurasi yang dilakukan antara lain seperti instalasi web server apache, MySQL, dan setting owner folder untuk menyediakan akses write file yang dilakukan client. Adapun untuk file apk yang telah digenerate akan dibagikan ke pihak internal DPTSI dalam bentuk *beta production* untuk dilakukan pengujian kelayakan lebih lanjut sebelum aplikasi mobile *service desk* go-live / dipublish ke playstore.

BAB VI
HASIL DAN PEMBAHASAN

Bab ini berisikan hasil dan pembahasan setelah dilkukannya pengembangan aplikasi mobile *service desk* beserta web servicenya.

6.1 Operasional Aplikasi Mobile *Service Desk*

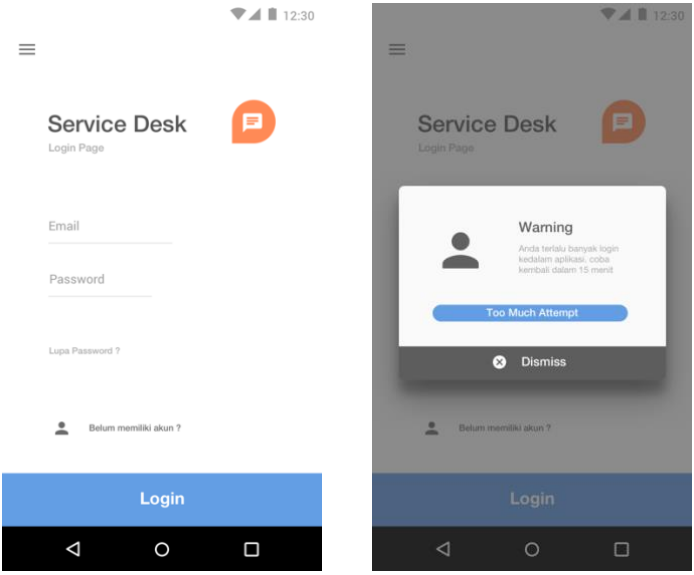
Proses *reengineering* yang dilakukan dalam pengembangan aplikasi mobile *service desk* membuat aplikasi mobile *service desk* memiliki flow operasional yang sama dengan aplikasi existing *service desk* yang beralamat di servicedesk.its.ac.id. Hal ini berarti dalam pengembangan yang dilakukan, semua fungsional aplikasi mobile telah disesuaikan dengan semua fungsional pada aplikasi existing tanpa adanya penggalian kebutuhan baru di luar aplikasi existing. Seperti halnya aplikasi *service desk* existing, user dapat mengakses aplikasi tanpa atau dengan menggunakan akun. Menu utama aplikasi mobile *service desk* adalah check ticket dan knowledge base, dimana check ticket memungkinkan pengguna non member untuk mengecek tiket yang telah dibuat tanpa menggunakan akun dengan memasukkan nomor tiket dan password tiket yang telah digenerate oleh sistem. Sedangkan menu knowledge base memungkinkan pengguna member dan non member untuk mencari tahu seluruh informasi mengenai *service desk* ataupun layanan yang berkaitan dengan IT dan DPTSI. Adapun tampilan utama yang berisikan menu check tiket dan menu knowledge base dapat dilihat pada gambar 6.1.



Gambar 6.1 Tampilan Home Aplikasi

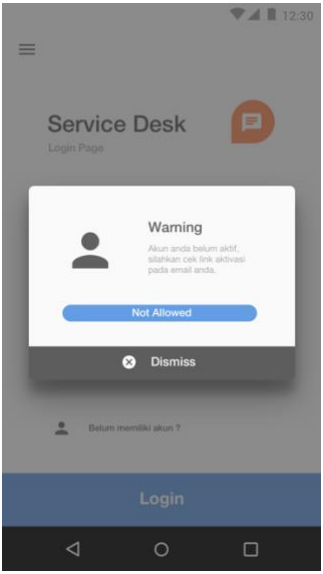
6.1.1 Manajemen Role User

Aplikasi mobile *service desk* mendukung pengaturan hak akses aplikasi sesuai role yang melekat ke masing-masing user. Saat melakukan login, dilakukan beberapa pengecekan, yang pertama adalah pengecekan jumlah login yang dilakukan. Sama seperti aplikasi *service desk* existing, jika terdapat kesalahan login lebih dari 5x dalam jangka waktu 15 menit, web service akan mengirimkan pesan yang menandakan pengguna harus menunggu 15 menit untuk melakukan login kembali seperti pada gambar 6.2 berikut.



Gambar 6.2 Tampilan Login (Terlalu Banyak Attempt)

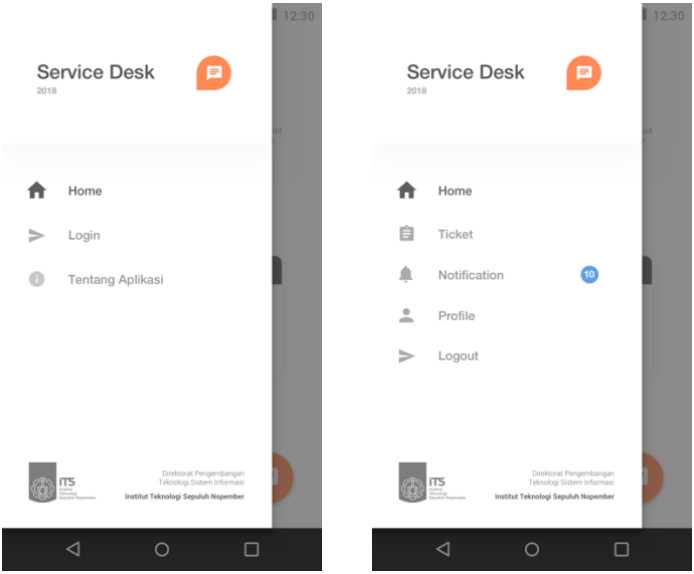
Menu login dalam aplikasi mobile service desk juga memungkinkan pengecekan apakah akun yang hendak digunakan sebagai *credential* login telah aktif atau belum aktif. Jika akun yang dimasukkan terdeteksi tidak aktif, maka web service akan mengirimkan pesan yang menandakan pengguna harus mengaktifkan akun melalui link aktivasi akun terlebih dahulu yang dikirim ke email yang dimasukkan saat melakukan registrasi akun seperti gambar 6.3 berikut.



Gambar 6.3 Tampilan Login (Akun Belum Aktif)

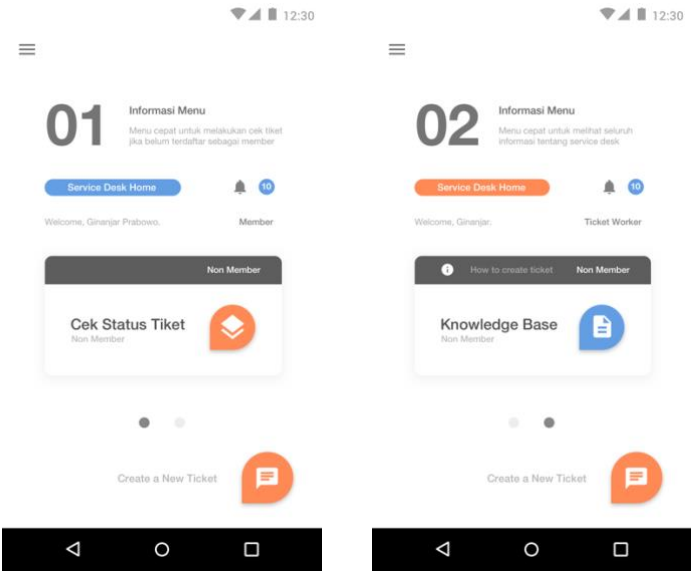
Jika berhasil melakukan login, akan muncul ucapan selamat datang dan user akan diarahkan ke menu home sesuai dengan role yang melekat pada akun. Data pengguna akan dimasukkan ke dalam *sharedpreferences* dan digunakan sebagai parameter untuk aplikasi dalam menjalankan fungsinya sesuai hak akses pengguna. Dimana terdapat perbedaan tampilan antara role member dan ticketworker, antara lain :

- Menu Sidebar Aplikasi
  Jika user berstatus login maka sidebar akan menampilkan seluruh menu yang ada di dalam aplikasi, jika tidak berstatus login, maka hanya 3 menu yang ditampilkan pada sidebar (Home, Login, dan Tentang Aplikasi). Perbedaan tampilan dapat dilihat pada gambar 6.4 berikut.



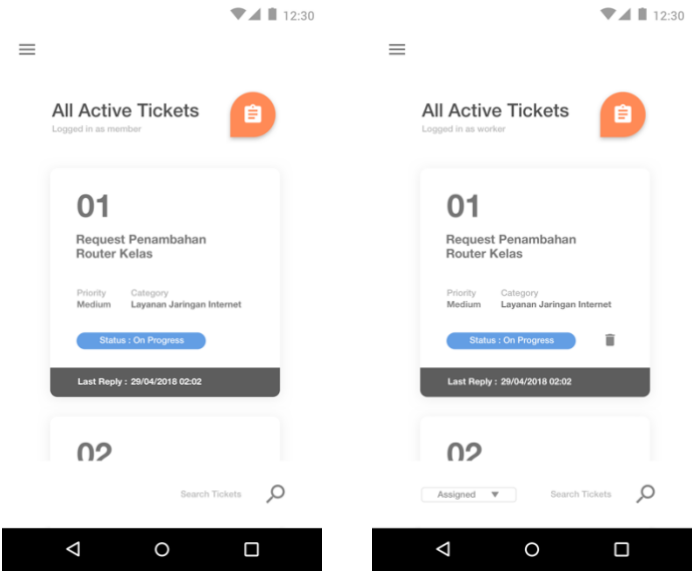
Gambar 6.4 Perbedaan Tampilan Sidebar Login dan Tidak Login

- Tampilan Home Aplikasi
  Jika terdeteksi sebagai member maka tulisan yang muncul pada halaman home adalah member, dan sebaliknya untuk ticketworker dengan warna aksen orange. Perbedaan tampilan dapat dilihat pada gambar 6.4 dibawah.



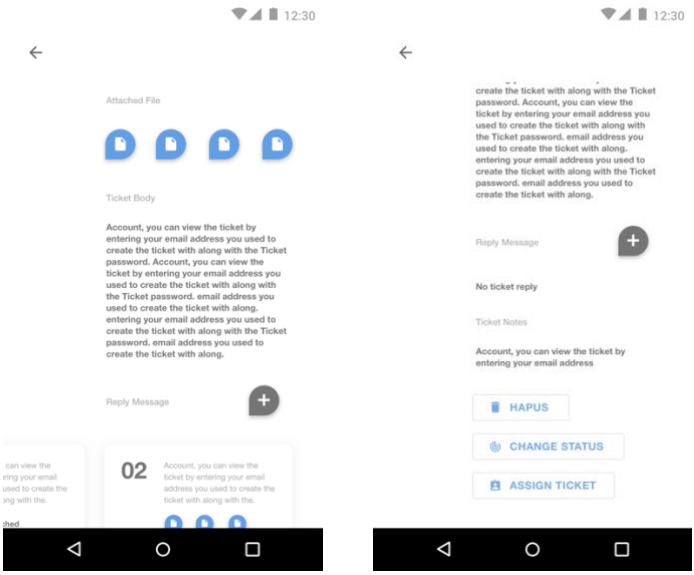
Gambar 6.5 Perbedaan Tampilan Home Antar Role

- Tampilan List Tiket Aplikasi
  Jika terdeteksi sebagai ticketworker, maka aplikasi akan memunculkan logo tong sampah di tiap listnya, sebaliknya untuk member. Adapun tampilan perbedaan dapat dilihat pada gambar 6.6 berikut.



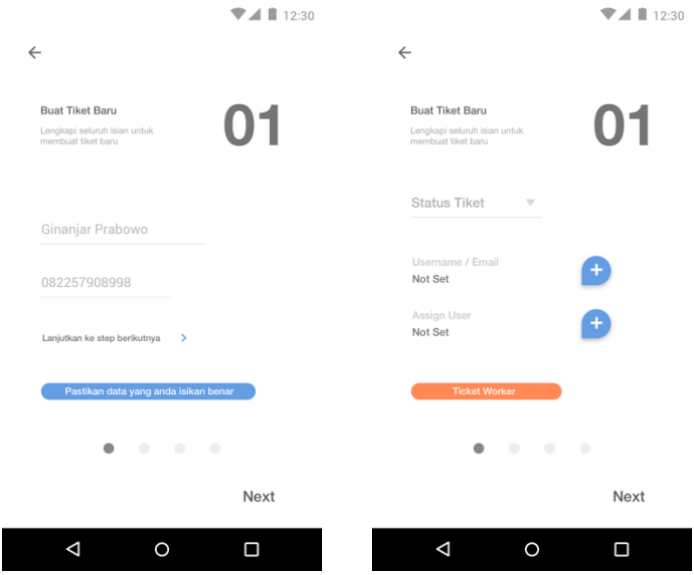
Gambar 6.6 Perbedaan Tampilan List Tiket Antar Role

- Tampilan Detail Tiket Aplikasi
  Jika terdeteksi sebagai ticketworker, notes dan button untuk melakukan transaksi hapus, assign, dan change status akan dimunculkan oleh aplikasi. Dan sebaliknya untuk member. Perbedaan tampilan dapat dilihat pada gambar 6.7 dibawah.



Gambar 6.7 Perbedaan Tampilan Detail Tiket Antar Role

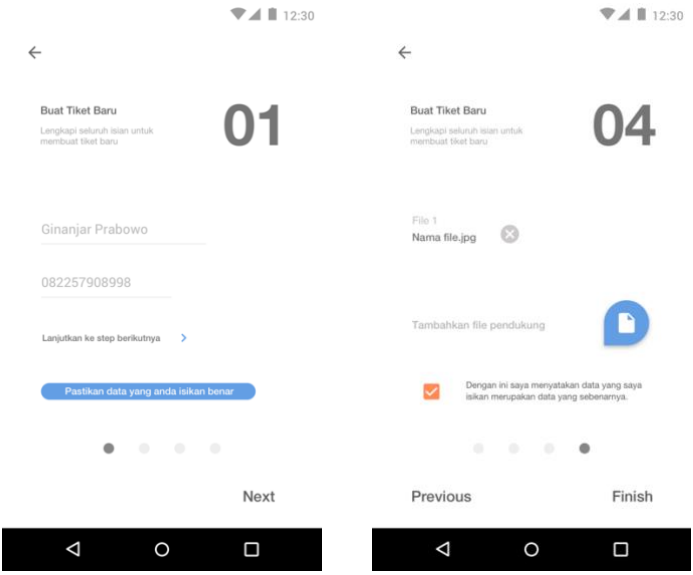
- Tampilan Buat Tiket Aplikasi.
  Jika terdeteksi sebagai ticketworker, step pertama yang ditampilkan adalah isian untuk melakukan assign dan menentukan username kepemilikan tiket. Jika terdeteksi sebagai member, maka step pertama yang ditampilkan merupakan data email dan username sesuai data yang disimpan dalam *sharedpreferencfes*. Adapun perbedaan tampilan dapat dilihat pada gambar 6.8 berikut.



Gambar 6.8 Perbedaan Tampilan Buat Tiket Antar Role

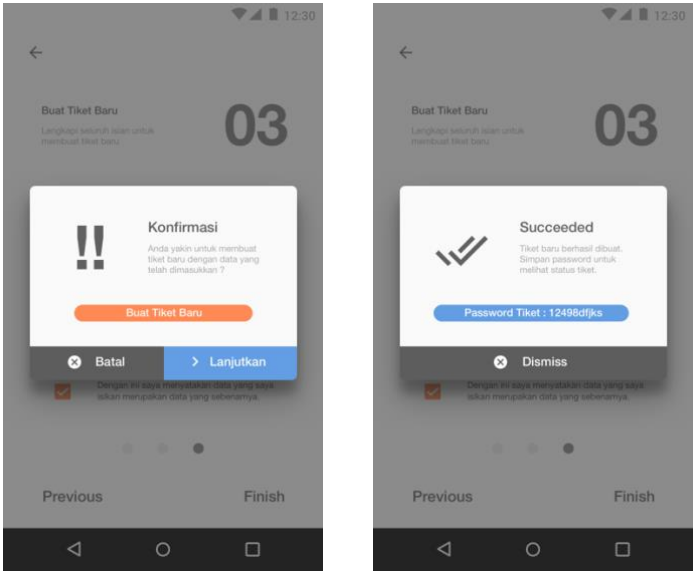
6.1.2 Pengelolaan Tiket

Tiket baru dapat dibuat dengan menekan tombol create a new ticket sampai halaman membuat tiket muncul dalam bentuk 4 step isian seperti pada gambar 6.9 dibawah.



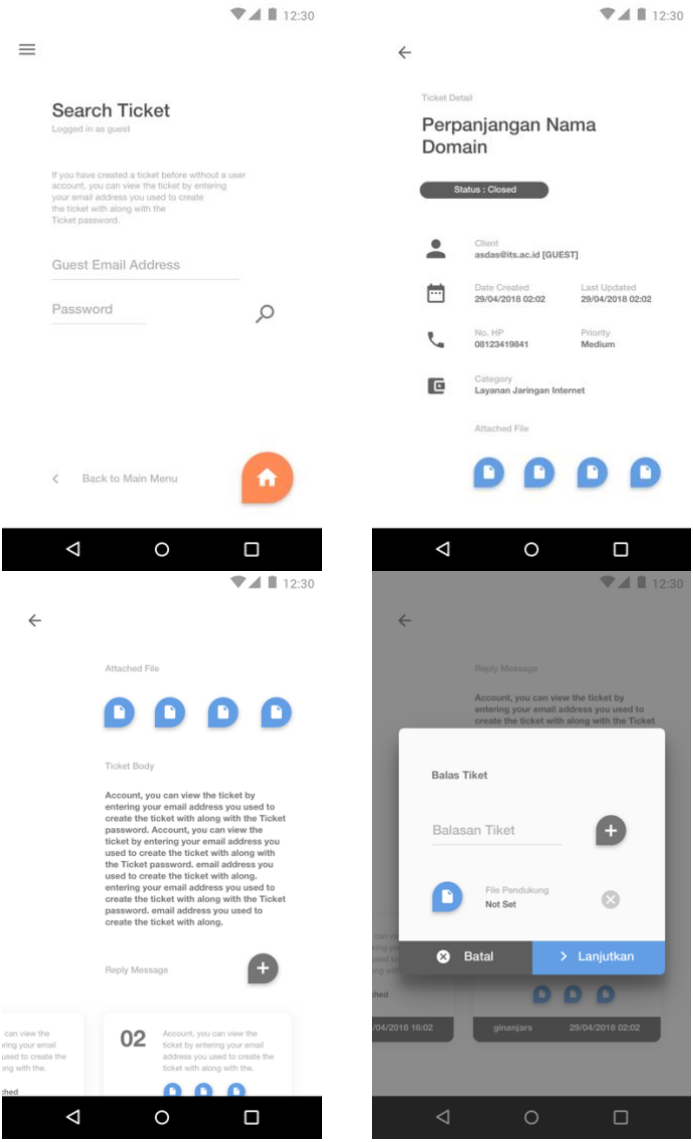
Gambar 6.9 Tampilan Buat Tiket

Setelah user mengisikan seluruh isian yang ada dan tiket berhasil dibuat, akan muncul password tiket yang dapat digunakan oleh guest (user non member) untuk melakukan cek tiket seperti pada gambar 6.10 dibawah ini.



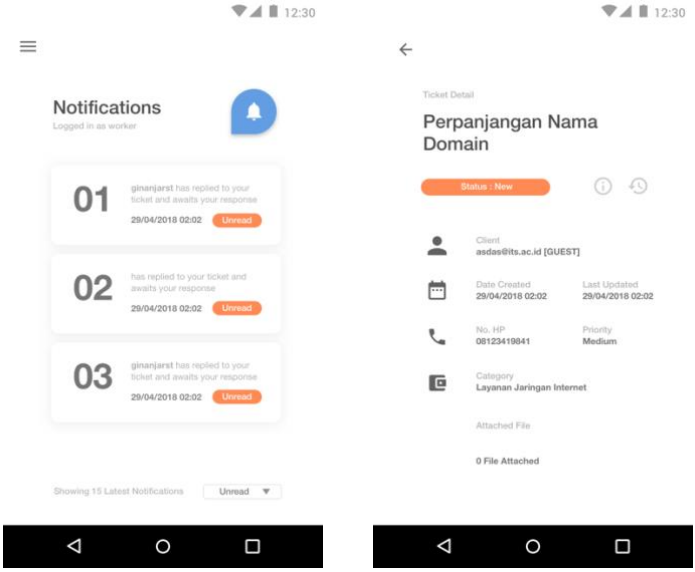
Gambar 6.10 Tampilan Konfirmasi Buat Tiket dan Generate Password

Setelah password didapatkan maka guest dapat melakukan cek info detail tiket melalui menu check tiket di halaman home. Dimana guest akan memasukkan alamat email yang digunakan beserta password yang didapatkan saat membuat tiket, kemudian setelah memasukkan email dan password jika terdeteksi benar, maka guest akan diarahkan langsung ke halaman detail tiket. Pada halaman detail tiket pun guest dapat membalas tiket apabila memang diperlukan. Adapun tampilan interaksi dapat dilihat pada gambar 6.11 dibawah.



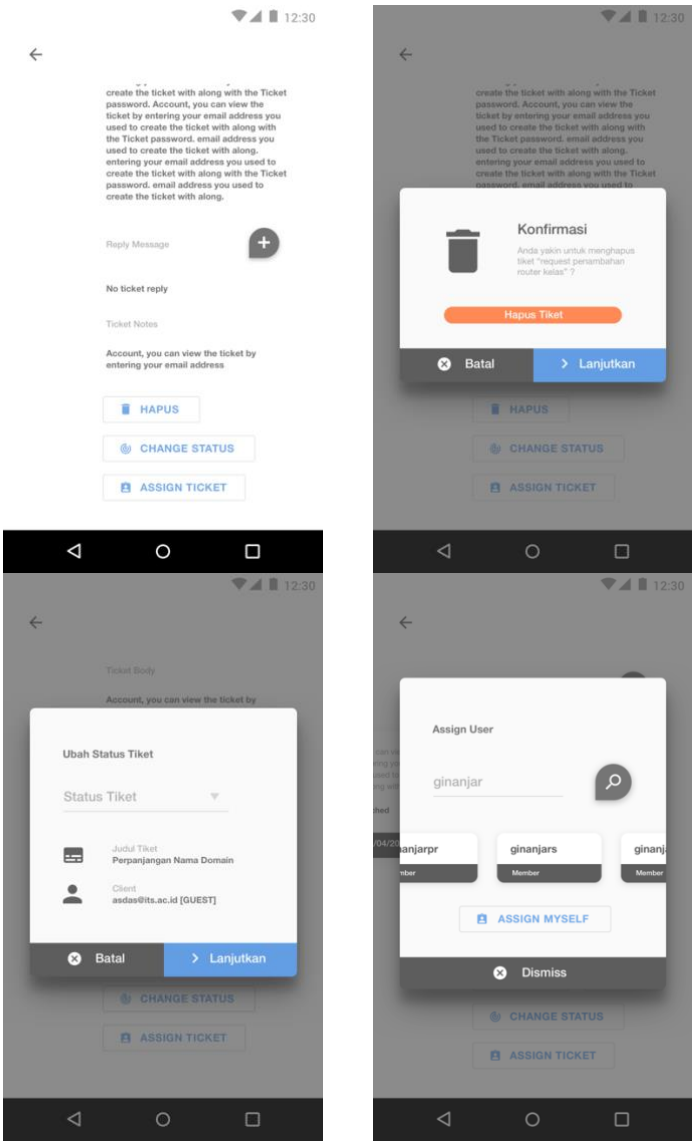
Gambar 6.11 Tampilan Check Tiket dan Balas Tiket (Guest)

Ketika ticket yang telah dibuat oleh guest telah di assign ke salah satu ticketworker oleh user lain, maka akan muncul notifikasi pada menu notifikasi yang dimiliki ticketworker, kemudian jika ticketworker menekan list notifikasi tersebut, maka ticketworker akan diarahkan ke halaman detail tiket terkait seperti yang ditampilkan pada gambar 6.12 dibawah.



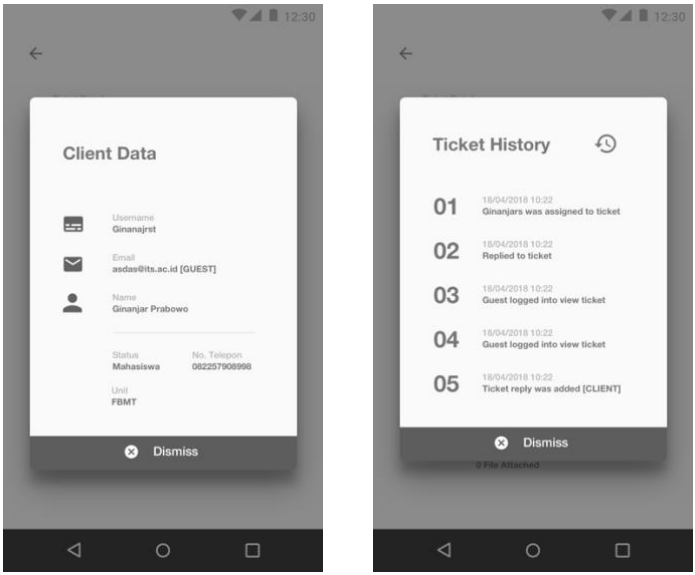
Gambar 6.12 Tampilan Lihat Notifikasi

Selain membalas tiket, ticketworker juga dapat menjalankan berbagai transaksi lain seperti hapus tiket, ubah status tiket, dan assign tiket. Hapus tiket berarti ticketworker akan menghapus seluruh entry tiket terkait, ubah status tiket berarti ticketworker telah mengambil tindakan sehingga status tiket dapat diubah, sedangkan assign tiket berarti ticketworker memindah tangankan assignment tiket terkait ke user lain karena alasan tertentu. Dimana tampilan terkait fungsi tersebut terdapat pada gambar 6.13 dibawah ini.



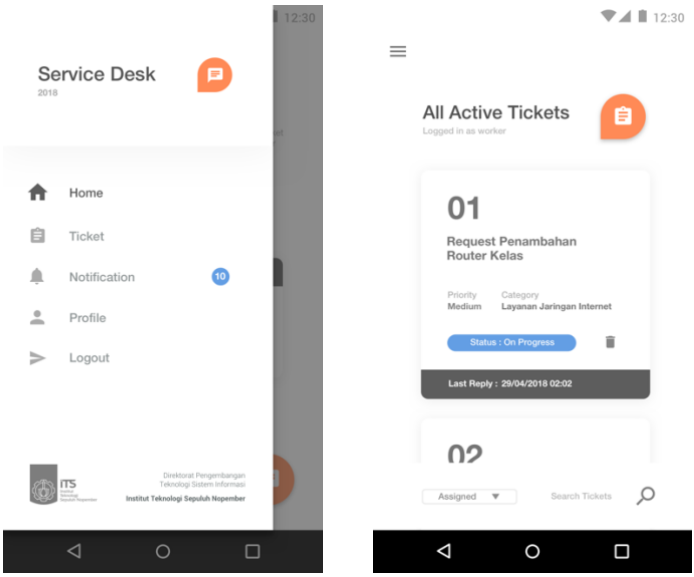
Gambar 6.13 Tampilan Hapus, Ubah Status, dan Assign Tiket

Untuk melakukan tracking terkait informasi client dan tiket, ticketworker juga dapat melihat data client dan melihat history tiket terkait dengan menekan button info client dan button info history pada halaman detail tiket seperti pada gambar 6.14 dibawah ini.



Gambar 6.14 Tampilan Lihat Data Client dan History Tiket

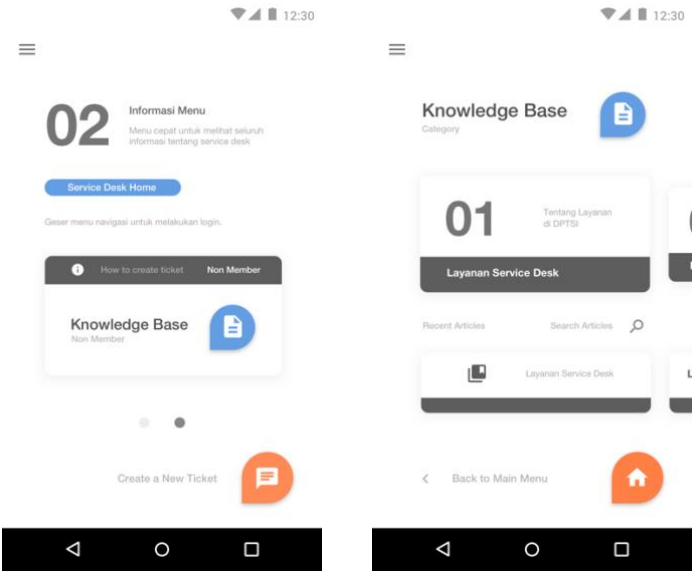
Jika aplikasi mendeteksi status login, maka tlist iket juga dapat diakses melalui menu sidebar. Ticketworker juga memungkinkan melakukan hapus tiket langsung pada list tiket dengan menekan button tong sampah pada tiap listnya. Tampilan menu tiket dapat dilihat pada gambar 6.15 dibawah.



Gambar 6.15 Tampilan List Tiket

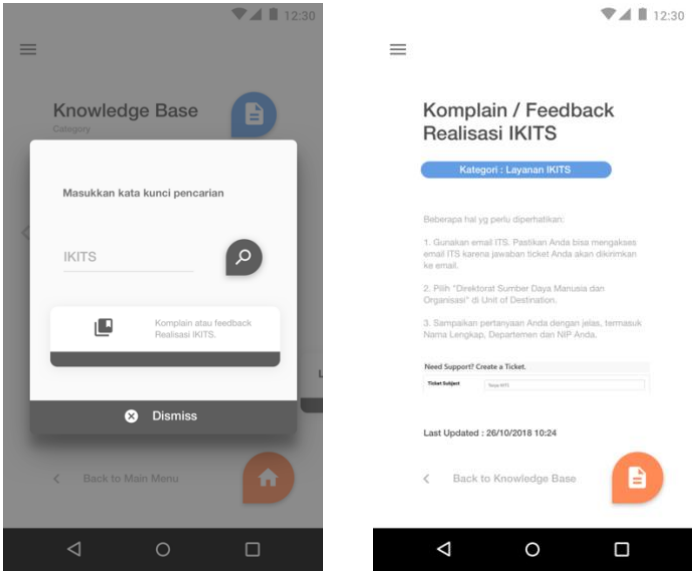
6.1.3 Informasi Penggunaan Aplikasi (Knowledge Base)

Sama seperti aplikasi *service desk* existing, aplikasi mobile *service desk* juga memilki fitur knowledge base yang berisikan seluruh informasi terkait penggunaan aplikasi dan layanan service desk lainnya. Untuk mengakses menu knowledge base, user tinggal menggeser menu pada halaman home dan menekan menu knowledge base seperti pada gambar 6.16 dibawah ini.



Gambar 6.16 Tampilan Utama Menu Knowledge Base

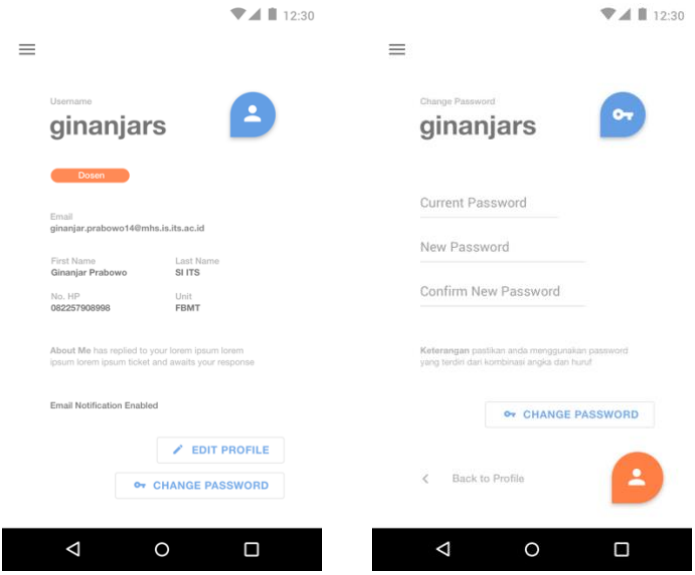
Dalam menu knowledge base ini terdapat beberapa opsi bagi user dalam mengakses informasi terkait. Pertama melalui halaman utama knowledge base dengan memilih kategori terlebih dahulu, dilanjutkan dengan menekan salah satu artikel, kemudian aplikasi akan menampilkan detail artikel terkait. Yang kedua, detail artikel dapat diakses melalui fitur search knowledge dimana user memasukkan terlebih dahulu kata kunci yang dituju, kemudian aplikasi akan menampilkan hasil temuan artikel yang sesuai. Ketika telah menemukan artikel yang sesuai, dengan menekan artikel yang dituju maka user akan diarahkan ke halaman detail artikel seperti gambar 6.17 dibawah.



Gambar 6.17 Tampilan Search Knowledge

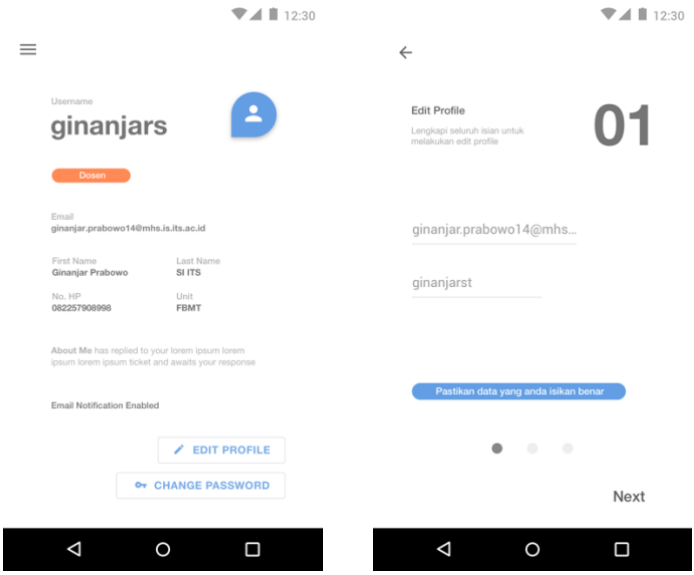
6.1.4 Manajemen Informasi Pengguna

Aplikasi mobile *service desk* juga memungkinkan user untuk melakukan mengubah informasi terkait data pribadi yang telah tersimpan di dalam database *service desk*. Akses menu profile dapat dimulai dengan menekan menu profile pada sidebar, kemudian dilanjutkan dengan user untuk memilih apakah ingin mengubah password atau mengubah data profile. Dengan menyentuh tombol change password pada tampilan, user akan diarahkan ke halaman change password seperti pada gambar 6.18 dibawah ini.



Gambar 6.18 Tampilan Menu Profile (Change Password)

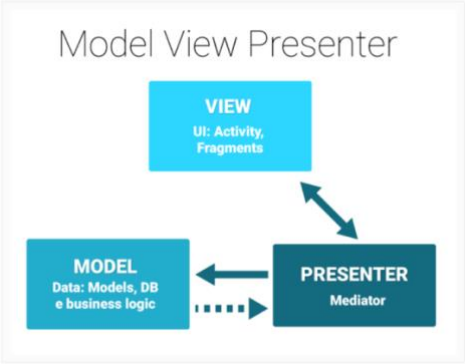
Selain change password, user juga dapat mengubah data profilenya dengan menekan tombol edit profile kemudian aplikasi akan mengarahkan user ke halaman edit profile dengan 3 step isian seperti pada gambar 6.19 berikut.



Gambar 6.19 Tampilan Menu Profile (Edit Profile)

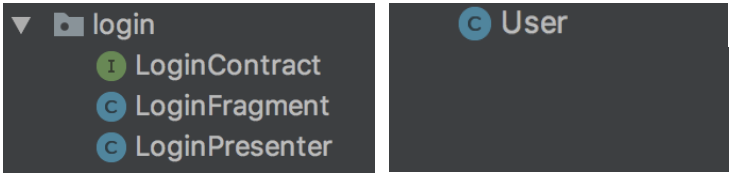
6.2 Arsitektur Aplikasi Mobile *Service Desk*

Aplikasi mobile *service desk* dibangun dengan menggunakan design pattern MVP (Model-View-Presenter).



Gambar 6.20 Struktur Desain Pattern MVP

Sehingga secara general, tiap modul/fungsi yang dikembangkan dalam aplikasi paling tidak memiliki 3 *class* ditambah 1 *class interface* yang berfungsi sebagai jembatan hubungan *inheritance* is-a. Sebagai contoh, terdapat 3 class yang terdapat pada fungsi login, yakni *class* LoginFragment yang berfungsi sebagai *class* View, LoginPresenter yang berfungsi sebagai class Presenter, dan User yang berfungsi sebagai *class* Model. Namun, karena penggunaan web service maka Model pada pengembangan aplikasi mobile service desk ini hanya berfungsi sebagai *entity class* (perantara komunikasi ke web service) dan hanya berisikan method *setter* dan *getter*. Selain itu juga terdapat *class interface* LoginContract yang berfungsi untuk menghubungkan *class* LoginFragment dan LoginPresenter. *Class interface* LoginContract dibuat agar coupling yang terbangun antara LoginFragment dan LoginPresenter seakan tereduksi dan coupling akhirnya tidak terbentuk antara LoginFragment dan LoginPresenter secara langsung, melainkan antara LoginContract dan LoginFragmentnya serta LoginContract dan LoginPresenternya. Memang tidak ada signifikansi dalam pembuatan *class interface* LoginContract, namun penggunaan *class interface* LoginContract lebih mengarah kepada gaya penulisan kode program yang biasa dilakukan oleh penulis untuk meningkatkan tingkat keterbacaan kode program. Struktur kode program login dapat dilihat pada gambar 6.21 dibawah ini.



Gambar 6.21 Struktur Kode Program Fungsi Login

Class interface LoginContract berisikan kumpulan method final yang digunakan oleh *class* LoginFragment maupun

LoginPresenter saat melakukan eksekusi method antar *class* yang dapat dilihat pada gambar 6.22 berikut.

```
/**
 * Created by ginanjarpr on 20/03/18.
 */

public interface LoginContract {

    //Presenter ------> View
    interface View {

        void showProgress();

        void hideProgress();

        void loginWrongCredentials();

        void loginErrorConnection();

        void setWelcomeName(String name);

        void goToHome();

        void setCounterDrawer(String counter);

        void showBelumAktif();

        void showNotAllowed();

        void tooMuchLogin();

    }

    //View ------> Presenter
    interface Presenter {

        void doLogin(String email, String password, Context context);

    }
```

Gambar 6.22 Kode Program *Class* LoginContract

Adapun hubungan inheritance dibangun dengan menerapkan atau mengimplementasikan *(implements)* LoginContract.View pada class LoginFragment dan LoginContract.Presenter pada class LoginPresenter seperti pada gambar 6.23 dibawah ini.

```
public class LoginFragment extends Fragment
        implements View.OnClickListener, LoginContract.Vi

    private LoginPresenter mPresenter;

public class LoginPresenter implements LoginContract.Presenter {

    private LoginContract.View mView;
    public SharedPreferences pref;

    public LoginPresenter(LoginContract.View view) {

        mView = view;

    }
```

Gambar 6.23 Inheritance *Class* LoginFragment dan LoginPresenter

Sehingga ketika *class* LoginFragment akan menggunakan method yang berada di *class* LoginPresenter bentuk kode yang ditulis adalah mPresenter.namaMethod() dan mView.namaMethod() untuk *class* LoginPresenter seperti pada gambar 6.24 berikut.

```
if(!email.isEmpty() && !password.isEmpty()) {

    mPresenter.doLogin(email, password, getActivity());

} else {

final Handler handler = new Handler();
handler.postDelayed(() → {

        mView.hideProgress();
        mView.goToHome();

},  delayMillis: 3000);
```

Gambar 6.24 Pemanggilan Method *Class* LoginFragment dan LoginPresenter

6.3 Arsitektur Web Service Service Desk

Untuk mendukung template request web service yang telah ditentukan pada tahap perencanaan maka ditentukanlah penggunaan method POST dalam tiap melakukan request. Pemilihan method POST dilakukan dengan pertimbangan keperluan untuk mengirimkan pesan pada body, tidak melalui uri. Adapun template request yang dikirim adalah sebagai berikut pada gambar 6.25.

```
{
  "operation":"nameOfOperation",
  "object":{
                    “entity1”:”value”,
                    “entity2”:”value”,
                    “entity3”:”value”
  }
}
```

Gambar 6.25 Format Request Web Service (JSON)

Web service dibangun tanpa menggunakan framework dengan menggunakan bahasa PHP dan terdiri dari 3 class. Yang pertama merupakan *class* dengan nama index.php yang berfungsi untuk mendeteksi *operation* request yang ada pada body request. Potongan kode program *class* index.php dapat dilihat pada gambar 6.26 berikut.

```
elseif ($operation == 'login') {
  if (isset($data -> user) && !empty($data -> user)
  && isset($data -> user -> email) && isset($data -> user -> password)) {
      $user = $data -> user;
      $email = $user -> email;
      $password = $user -> password;

      echo $fun -> loginUser($email, $password);
  } else {
      echo $fun -> getMsgInvalidParam();
  }
```

Gambar 6.26 Potongan Kode Program Web Service Index.php

Dari deteksi *operation* request, *class* index.php akan mengarahkan input kedalam *class* Functions.php. class ini berfungsi untuk mengarahkan lagi inputan dari body request kedalam transaksi database pada class DBOperations.php dan menambah response tambahan jika diperlukan. Adapun potongan kode program class Functions.php terdapat pada gambar 6.27 dibawah.

```
public function loginUser($email, $password)
{
    $db = $this -> db;

    if (!empty($email) && !empty($password)) {
        if ($db -> check_block_ip()) {

  //IP terblock
            return $this -> getMsgBlockedIP();
        } else {

  //IP tidak terblock
            if ($db -> get_login_attemptsG($_SERVER['REMOTE_ADDR'], $email, (15*60))) {

    //Login Banyak Terdeteksi
                return $this -> getMsgLoginTooMuch();
            } else {
```

Gambar 6.27 Potongan Kode Program Web Service Functions.php

Kemudian class terakhir yang berfungsi untuk menjalankan query dan mengembalikan response langsung dari hasil/output query adalah class DBOperations.php seperti pada gambar 6.28 dibawah.

```php
public function check_block_ip()
{
    $sql = 'SELECT COUNT(*) FROM ip_block WHERE IP = :IP';
    $query = $this -> conn -> prepare($sql);
    $query -> execute(array(':IP' => $_SERVER['REMOTE_ADDR']));

    $row_count = $query -> fetchColumn();

    if ($row_count == 0) {
        return false;
    } else {
        return true;
    }
}
```

Gambar 6.28 Potongan Kode Program Web Service DBOperations.php

Juga terdapat *class* tambahan dalam web service yang dibuat bernama upload_image.php yang berfungsi untuk menghandle penulisan file ke web server dimana potongan kode program class tersebut terdapat pada gambar 6.29 dibawah.

```php
<?php
$target_dir = "uploads/";
$target_file_name = $target_dir .basename($_FILES["file"]["name"]);
$response = array();

// Check if image file is a actual image or fake image
if (isset($_FILES["file"])) {
    if (move_uploaded_file($_FILES["file"]["tmp_name"], $target_file_name)) {
        $success = true;
        $message = "Successfully Uploaded";
    } else {
        $success = false;
        $message = "Error while uploading";
    }
} else {
    $success = false;
    $message = "Required Field Missing";
}
$response["success"] = $success;
$response["messages"] = $message;
echo json_encode($response);
```

Gambar 6.29 Potongan Kode Program Web Service upload_image.php

BAB VII
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpuln dari hasil penelitian dan juga saran perbaikan untuk penelitian kedepannya.

7.1 Kesimpulan

Berdasarkan seluruh proses yang telah dilakukan dalam pengerjaan tugas akhir terdapat beberapa kesimpulan yang dapat diambil, diantaranya sebagai berikut :

1. Aplikasi *service desk* versi mobile merupakan aplikasi client-side berbasis android yang dikembangkan dengan design pattern MVP guna meningkatkan maintainability kode program yang telah dibuat. Penggunaan design pattern MVP menciptakan separasi yang jelas terkait role tiap 3 kelas turunan yang dibuat, class model hanya bertugas untuk memodifikasi data (dalam kasus aplikasi *service desk*, penggunaan web service menyebabkan kelas model hanya menjadi entity class yang berfungsi sebagai perantara komunikasi antara client dan web service). Dalam aplikasi, kelas presenter juga hanya bertugas sebagai komunikator view dan model (web service dalam konteks aplikasi service desk) untuk melakukan perubahan data pada model dan modifikasi tampilan pada view, sedangkan kelas view pun juga hanya bertugas untuk memodifikasi tampilan yang ditampilkan ke user tanpa menangani satupun transaksi data dari client ke web service dan sebaliknya.
2. Untuk mengurangi coupling antar entitas MVP, dilakukan pembuatan class interface yang berfungsi sebagai jembatan (is-a) antara view dan presenter. Hal ini menyebabkan coupling tidak dimunculkan langsung dari hubungan class view dan class presenter, namun dengan hal ini coupling terbentuk dari hubungan class view dan class interface, begitupula untuk class presenter.

3. Aplikasi service desk versi mobile memanfaatkan REST API yang dibuat sesuai dengan kebutuhan pengembangan. REST API dibangun dengan bahasa PHP native tanpa menggunakan framework dengan struktur 3 class yang terdiri dari 1 class untuk melakukan filter operasi yang ada pada body request, 1 class untuk memanggil fungsi dan mengembalikan response, dan 1 class yang berisikan query. REST API yang dibuat kemudian di deploy ke cloud.

7.2 Saran

Adapun saran yang diberikan untuk pengembangan selanjutnya adalah sebagai berikut :

1. Aplikasi mobile service desk dalam pengembangan awal hanya berfungsi untuk role ticket worker dan user, kedepan perlu diadakannya pengembangan aplikasi agar dapat berfungsi untuk role user lainnya.
2. Pengembangan pertama aplikasi mobile service desk masih berfokus pada pengembangan di sisi client sehingga agak mengesampingkan kecanggihan web service yang dibuat, hal ini perlu diperhatikan lebih lanjut dalam pengembangan selanjutnya.
3. Pengembangan pertama aplikasi mobile service desk masih belum menggunakan sistem caching dan konsep reactive programming yang dapat mengehemat resource dalam transaksi data, baiknya pengembangan berikutnya memperhatikan hal ini.
4. Ada baiknya pada pengembangan tahap selanjutnya juga dilakukan whitebox testing dan UI test guna meningkatkan kualitas kode program.

DAFTAR PUSTAKA

[1] DPTSI, 2016. [Online]. Available: http://dptsi.its.ac.id. [Diakses 25 Februari 2018].

[2] Susanto, T. D., Manajemen Layanan Teknologi Informasi, Surabaya.

[3] Rouse, Margaret., "Service Desk Definition," Oktober 2010. [Online]. Available: http://searchwindowsserver.techtarget.com/definition/service-desk. [Diakses 25 Februari 2018].

[4] "Sosialisasi Servicedesk dan Refresh Layanan TSI di DPTSI, " Nopember 2017. Available: https://dptsi.its.ac.id/?p=1670. [Diakses 25 Februari 2018].

[5] "Penetrasi Smartphone Indonesia Kalahkan India dan Filipina," Oktober 2016. [Online]. Available: https://databoks.katadata.co.id/datapublish/2016/10/03/penetrasi-smartphone-indonesia-kalahkan-india. [Diakses 25 Februari 2018].

[6] Yun Ma., Xuanzhe Liu., Yi Liu., and Gang Huang., "A Tale of Two Fashions: An Empirical Study on the Performance of Native Apps and Web Apps on Android," *IEEE Transactions on Mobile Computing*, pp. 99, 2017.

[7] Emre, K., Canturk, M., and Bastan, M., "Performance Analysis of a Software Developed with and withous Design Patterns: A Case Study," *Electronics Computer and Computation* (*ICECCO)*, 2015.

[8] Pithva, A. Keval, Vaghela, K. Ravirajsinh., "Software Design Pattern approach to develop Login Framework," *Computing for Sustainable Global Development (INDIACom)*, 2016.

[9] Panca, Billy S., Mardiyanto S., and Hendradjaya, B., "Evaluation of Software Design Pattern on Mobile Application Based Service Development Related to the Value of Maintainability and Modularity," *Data and Software Engineering (ICoDSE)*, 2016.

[10] IEEE, IEEE Standard Glossary of Software Engineering Terminology, report IEEE Std 610.12-2010, *IEEE*, 1990.

[11] Yang Zang, Luo Yanjing., "An Architecture and Implement Model for Model-View-Presenter Pattern," *ICCSIT*, 2010.

[12] Syromiatnikov, Artem., Weyns, Danny., "A Journey Through the Land of Model-View-* Design Patterns,"

Software Architecture (SICSA) IEEE/IFIP Conference, 2014.

[13] "Advantage of MVP in Android," July 2017. [Online]. Available: https://stackoverflow.com/questions/40766185/advantage-of-mvp-in-android. [Diakses 25 Maret 2018].

[14] "Android Testing Codelab," Oktober 2016. [Online]. Available: https://codelabs.developers.google.com/codelabs/android-testing/index.html. [Diakses 25 Maret 2018].

[15] ITIL, ITIL Service Operation Best Management Practice 2011 Edition, Ireland: The Stationery Office (TSO), 2011.

[16] "Android (Sistem Operasi)," Februari 2018. [Online]. Available: http://id.wikipedia.org/wiki/Android_(sistem_operasi). [Diakses 28 Februari 2018].

[17] Brahler, Stefan., "Analysis of the Android Architecture," *Universitat des Landes Baden-Wurttemberg und nationales Forschungszentrum*, 2010.

[18] "Design Pattern," Oktober 2018. [Online]. Available: https://sourcemaking.com/design_patterns. [Diakses 28 Februari 2018].

[19] Duy, Tung Bui., "Reactive Programming and Clean Architecture in Android Development," *Helsinki Metropolia University of Applied Sciences*, 2017.

[20] Priandana, Asep Bagja., "Berkenalan Dengan Unit Testing," Februari 2017. [Online]. Available: https://blog.framework.id/berkenalan-dengan-unit-testing-47688fe0fa4. [Diakses 1 Maret 2018].

[21] Suhartono, Joni., "Unit Testing Cycle," Desember 2016. [Online]. Available: https://sis.binus.ac.id/2016/12/16/unit-testing-cycle. [Diakses 1 Maret 2018].

[22] "8 Testing tools dan Framework Bagi Developer Java," Januari 2017. [Online]. Available: https://jelas.in/8-testing-tools-dan-framework-bagi-developer-java. [Diakses 1 Maret 2018].

[23] Mukhlasin, Hafid., "Mengenal Restful Web Service?," Desember 2015. [Online]. Available: https://hafidmukhlasin.com/2015/12/08/mengenal-

restful-web-service/. [Diakses 1 Maret 2018].

[24] Feridi., “Mengenal Restful Web Services,” Februari 2016. [Online]. Available: https://codepolitan.com/mengenal-restful-web-services. [Diakses 1 Maret 2018].

[25] Susilowati, Susi., "Pengembangan Sistem Informasi Manajemen Zakat, Infaq, Shadaqoh, Waqaf dan Hibah Menggunakan Metode Waterfall," Paradigma, 2017.

[26] Hayder Saad et al., “E-LEARNING DEVELOPMENT " E-LEARNING DEVELOPMENT AND ASSESSMENT REPORT: CASE STUDY I- FOLIO," vol. 85, 2016.

[27] Bukhary Ikhwan Ismail et al., " Evaluation of Docker as Edge Computing Platform," Malaysia, 2015.

[28] Nurzhafar, Eko., “JUnit,” Juni 2012. [Online]. Available: https:// ekonurzhafar.wordpress.com/2012/06/04/junit/. [Diakses 25 Maret 2018].

[29] “User Interface Testing,” Juni 2012. [Online]. Available: https://www.tutorialspoint.com/software_testing_dictionary/use_interface_testing.htm. [Diakses 25 Maret

2018].

[30] "The Top 5 Android UI Frameworks for Automated Testing," September 2017. [Online]. Available: https://saucelabs.com/blog/the-top-5-android-ui-frameworks-for-automated-testing. [Diakses 25 Maret 2018].

[31] Manish Kumar, Santos Kumar Singh, and Dwivedi, "A Comparative Study of Black Box Testing and White Box Testing Techniques," International Journal of Advance Research in Computer Science and Management Studies, 2015.

BIODATA PENULIS

Penulis bernama lengkap Ginanjar Prabowo, lahir di Surabaya, 20 Mei 1996, merupakan anak kedua dari dua bersaudara. Penulis telah menempuh beberapa jenjang pendidikan formal di sekolah yang sama yaitu: SD, SMP, dan SMA Al Hikmah Surabaya. Penulis meneruskan pendidikan di Jurusan Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada tahun 2014 pasca lulus dari pendidikan SMA dan terdaftar sebagai mahasiswa dengan NRP 5214100115. Selama menjadi mahasiswa, penulis aktif dan selalu tertarik mengikuti organisasi kemahasiswaan seperti menjadi staff di Departemen Kewirausahaan (KWU) Himpunan Mahasiswa Sistem Informasi (HMSI) pada tahun 2015, Staff kreatif Information Systems Expo 2015 yang kemudian dilanjutkan dengan menjadi konseptor branding di tahun berikutnya. Berbagai kegiatan lain juga pernah diikuti, salah satunya adalah kegiatan inkubasi startup Start Surabaya Batch 4 dan berhasil mendapat penghargaan "The Most Outstanding Startup" pada kegiatan yang berlangsung selama kurang lebih 6 bulan tersebut.

Pada tahun keempat perkuliahan, di akhir pelaksanaan magang, penulis direkrut oleh PT Pelindo III (Persero) sebagai UI/UX Developer di perusahaan tersebut, dimana selain mobile programming, UI/UX juga menjadi peminatan utama penulis hingga saat ini. Penulis dapat dihubungi melalui email ginanjar.prabowo@pelindo.co.id.

Halaman ini sengaja dikosongkan

LAMPIRAN A : Perancangan Test Case

Pengujian yang digunakan pada penelitian ini adalah pengujian dengan menggunakan metode *blackbox*. Berikut merupakan rancangan testcase pengujian yang dibuat berdasarkan use case yang telah disepakati berdasarkan kebutuhan fungsional

A.1 Test Case User

Use Case	Identifikasi Pengujian	Skenario	Prosedur Pengujian	Masukan	Hasil Ekspektasi
U-01	UCT001	Normal	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password dan	Input karakter username dan password (Member)	Muncul ucapan selamat datang dan berhasil masuk ke halaman home sebagai member

			menekan tombol login		
	UCT002	Alternatif 1	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password dengan tidak lengkap / kosong salah satu dan menekan tombol login	Input karakter username dan password (Tidak lengkap / kosong salah satu isian)	Muncul peringatan untuk melengkapi data isian
	UCT003	Alternatif 2	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan	Input karakter username dan password (Tidak terdapat dalam database)	Muncul peringatan username / password salah

			email/username, password yang salah dan menekan tombol login		
	UCT004	Alternatif 3	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password akun yang belum aktif dan menekan tombol login	Input karakter username dan password (Akun yang belum aktif)	Muncul peringatan dialog akun belum aktif
	UCT005	Alternatif 4	1. User menekan tombol menu 2. User menekan menu login	Input karakter username dan password (Tidak	Muncul peringatan dialog terlalu banyak login

			3. User memasukkan email/username, password yang salah kemudian menekan tombol login secara terus menerus lebih dari 5x dalam jangka waktu 15 menit	terdapat dalam database)	
U-02	UCT006	Normal	1. User menekan tombol menu 2. User menekan menu logout	-	User masuk kedalam halaman login

U-03	UCT007	Normal	1. User menekan tombol menu 2. User menekan menu notifikasi 3. User masuk ke halaman notifikasi kemudian menekan salah satu entry notifikasi	-	User masuk ke halaman detail tiket terkait
U-04	UCT008	Normal	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User memasukkan kata kunci pencarian dan menekan tombol pencarian	Input karakter keyword pencarian (Terdapat dalam Database)	Topik informasi sesuai dengan kata kunci yang dimasukkan muncul

	UCT009	Alternatif 1	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User memasukkan kata kunci pencarian dan menekan tombol pencarian	Input karakter keyword pencarian (Tidak terdapat dalam Database)	Muncul tampilan artikel tidak ditemukan
U-05	UCT010	Normal	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User menekan salah satu info pada list sesuai pilihan	-	User masuk ke halaman detail artikel / informasi

	UCT011	Alternatif 1	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User memasukkan kata kunci pencarian dan menekan tombol pencarian 4. User menekan salah satu info pada list sesuai pilihan	-	User masuk ke halaman detail artikel / informasi
U-06	UCT012	Normal	1. User menggeser menu utama 2. User menekan icon (i) pada menu knowledge base	Input karakter keyword pencarian (Terdapat dalam Database)	PDF info buat tiket terdownload

U-07	UCT013	Normal	1. User menekan tombol menu 2. User menekan menu profile	-	User masuk pada halaman profile

U-08	UCT014	Normal	1. User menekan tombol menu 2. User menekan menu profile 3. User menekan tombol edit profile 4. User menekan tombol konfirmasi 5. User melengkapi kembali data informasi terkait 6. User menekan tombol selesai 7. User menekan tombol konfirmasi	Input karakter informasi profile (lengkap)	Dialog sukses edit profile ditampilkan

	UCT015	Alternatif 1	1. User menekan tombol menu 2. User menekan menu profile 3. User menekan tombol edit profile 4. User menekan tombol konfirmasi 5. User melengkapi kembali data informasi terkait 6. User menekan tombol selesai	Input karakter informasi profile (tidak lengkap)	Peringatan lengkapi data isian ditampilkan

U-09	UCT016	Normal	1. User menekan tombol menu 2. User menekan menu profile 3. User menekan tombol change password 4. User memasukkan password lama dan isian konfirmasi password baru 5. User menekan tombol lanjutkan untuk mengubah password	Input karakter isian password lama dan baru	Dialog sukses ganti password ditampilkan

U-10	UCT017	Normal	1. User menekan tombol menu 2. User menekan tombol login 3. User menekan tombol lupa password	-	Redirect ke browser halaman lupa password service desk
U-11	UCT018	Normal	1. User menekan tombol menu 2. User menekan menu login 3. User menekan tombol belum memiliki akun	-	Redirect ke browser halaman registrasi akun service desk

U-12	UCT019	Normal	1. User menekan tombol create new ticket 2. User menekan konfirmasi buat tiket 3. User mengisi form yang telah disediakan kemudian menekan tombol next 4. User menekan tombol selesai 5. User menekan tombol lanjutkan untuk membuat tiket	Input karakter isian data buat tiket (lengkap)	Dialog berhasil buat tiket ditampilkan

	UCT020	Alternatif 1	1. User menekan tombol create new ticket 2. User menekan konfirmasi buat tiket 3. User mengisi form yang telah disediakan kemudian menekan tombol next 4. User menekan tombol selesai	Input karakter isian data buat tiket (tidak lengkap)	Peringatan lengkapi data isian ditampilkan
U-13	UCT021	Normal	1. User menekan tombol menu 2. User menekan menu tiket	-	List tiket terasosiasi dengan user muncul pada tampilan

U-14	UCT022	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan tombol search 4. User memasukkan kata kunci pencarian, klik search	Inpu karakter keyword pencarian (terdapat dalam database)	Tiket relevan dengan kata kunci ditampilkan
	UCT023	Alternatif 1	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan tombol search 4. User memasukkan kata kunci pencarian, klik search	Input karakter keyword pencarian (tidak terdapat dalam database)	Muncul pemberitahuan tiket tidak ditemukan

U-15	UCT024	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list	-	User masuk ke tampilan detail tiket
	UCT025	Alternatif 1	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan tombol search 4. User memasukkan kata kunci pencarian, klik search 5. User menekan salah satu tiket dalam list	-	User masuk ke tampilan detail tiket

U-16	UCT026	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan tombol tambah balasan 5. User memasukkan seluruh isian balasan 6. User menekan tombol konfirmasi balas tiket	Input karakter dan file balasan tiket	Informasi balasan berhasil dibuat ditampilkan

U-17	UCT027	Normal	1. User menekan menu check tiket 2. User memasukkan data isian cek tiket 3. Pemberitahun tiket ditemukan ditampilkan	Input email dan password tiket (terdapat dalam database)	User masuk pada halaman detail tiket terkait
	UCT028	Alternatif 1	1. User menekan menu check tiket 2. User memasukkan data isian cek tiket	Input email dan password tiket (tidak terdapat dalam database)	Pemberitahuan tiket tidak ditemukan ditampilkan

U-18	UCT029	Normal	1. User menekan menu check tiket 2. User memasukkan data isian cek tiket 3. Pemberitahun tiket ditemukan ditampilkan 4. User menekan tombol tambah balasan 5. User memasukkan seluruh isian balasan 6. User menekan tombol konfirmasi balas tiket	Input karakter dan file balasan tiket	Informasi balasan berhasil dibuat ditampilkan

A.2 Test Case Ticket Worker

Use Case	Identifikasi Pengujian	Skenario	Prosedur Pengujian	Masukan	Hasil Ekspektasi
T-01	TCT001	Normal	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password dan menekan tombol login	Input karakter username dan password (Ticket worker)	Muncul ucapan selamat dating dan berhasil masuk ke halaman home sebagai ticket worker
	TCT002	Alternatif 1	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan	Input karakter username dan password (Tidak lengkap / kosong salah satu isian)	Muncul peringatan untuk melengkapi data isian

			email/username, password dengan tidak lengkap / kosong salah satu dan menekan tombol login		
	TCT003	Alternatif 2	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password yang salah dan menekan tombol login	Input karakter username dan password (Tidak terdapat dalam database)	Muncul peringatan username / password salah
	TCT004	Alternatif 3	1. User menekan tombol menu	Input karakter username dan password	Muncul peringatan dialog akun belum aktif

			2. User menekan menu login 3. User memasukkan email/username, password akun yang belum aktif dan menekan tombol login	(Akun yang belum aktif)	
	TCT005	Alternatif 4	1. User menekan tombol menu 2. User menekan menu login 3. User memasukkan email/username, password yang salah kemudian menekan tombol login secara terus menerus lebih dari	Input karakter username dan password (Tidak terdapat dalam database)	Muncul peringatan dialog terlalu banyak login

			5x dalam jangka waktu 15 menit		
T-02	TCT006	Normal	1. User menekan tombol menu 2. User menekan menu logout	-	User masuk kedalam halaman login
T-03	TCT007	Normal	1. User menekan tombol menu 2. User menekan menu notifikasi 3. User masuk ke halaman notifikasi kemudian menekan salah satu entry notifikasi	-	User masuk ke halaman detail tiket terkait

T-04	TCT008	Normal	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User memasukkan kata kunci pencarian dan menekan tombol pencarian	Input karakter keyword pencarian (Terdapat dalam Database)	Topik informasi sesuai dengan kata kunci yang dimasukkan muncul
	TCT009	Alternatif 1	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User memasukkan kata kunci pencarian dan menekan tombol pencarian	Input karakter keyword pencarian (Tidak terdapat dalam Database)	Muncul tampilan artikel tidak ditemukan

T-05	TCT010	Normal	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User menekan salah satu info pada list sesuai pilihan	-	User masuk ke halaman detail artikel / informasi
	TCT011	Alternatif 1	1. User menggeser menu utama 2. User menekan menu knowledge base 3. User memasukkan kata kunci pencarian dan menekan tombol pencarian 4. User menekan salah satu info	-	User masuk ke halaman detail artikel / informasi

			pada list sesuai pilihan		
T-06	TCT012	Normal	1. User menggeser menu utama 2. User menekan icon (i) pada menu knowledge base	Input karakter keyword pencarian (Terdapat dalam Database)	PDF info buat tiket terdownload
T-07	TCT013	Normal	1. User menekan tombol menu 2. User menekan menu profile	-	User masuk pada halaman profile

T-08	TCT014	Normal	1. User menekan tombol menu 2. User menekan menu profile 3. User menekan tombol edit profile 4. User menekan tombol konfirmasi 5. User melengkapi kembali data informasi terkait 6. User menekan tombol selesai 7. User menekan tombol konfirmasi	Input karakter informasi profile (lengkap)	Dialog sukses edit profile ditampilkan

	TCT015	Alternatif 1	1. User menekan tombol menu 2. User menekan menu profile 3. User menekan tombol edit profile 4. User menekan tombol konfirmasi 5. User melengkapi kembali data informasi terkait 6. User menekan tombol selesai	Input karakter informasi profile (tidak lengkap)	Peringatan lengkapi data isian ditampilkan

T-09	TCT016	Normal	1. User menekan tombol menu 2. User menekan menu profile 3. User menekan tombol change password 4. User memasukkan password lama dan isian konfirmasi password baru 5. User menekan tombol lanjutkan untuk mengubah password	Input karakter isian password lama dan baru	Dialog sukses ganti password ditampilkan

T-10	TCT017	Normal	1. User menekan tombol menu 2. User menekan tombol login 3. User menekan tombol lupa password	-	Redirect ke browser halaman lupa password service desk
T-11	TCT018	Normal	1. User menekan tombol menu 2. User menekan menu login 3. User menekan tombol belum memiliki akun	-	Redirect ke browser halaman registrasi akun service desk

T-12	TCT019	Normal	1. User menekan tombol create new ticket 2. User menekan konfirmasi buat tiket 3. User mengisi form yang telah disediakan kemudian menekan tombol next 4. User menekan tombol selesai 5. User menekan tombol lanjutkan untuk membuat tiket	Input karakter isian data buat tiket (lengkap)	Dialog berhasil buat tiket ditampilkan

	TCT020	Alternatif 1	1. User menekan tombol create new ticket 2. User menekan konfirmasi buat tiket 3. User mengisi form yang telah disediakan kemudian menekan tombol next 4. User menekan tombol selesai	Input karakter isian data buat tiket (tidak lengkap)	Peringatan lengkapi data isian ditampilkan
T-13	TCT021	Normal	1. User menekan tombol menu 2. User menekan menu tiket	-	List tiket terasosiasi dengan user muncul pada tampilan

T-14	TCT022	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan tombol search 4. User memasukkan kata kunci pencarian, klik search	Inpu karakter keyword pencarian (terdapat dalam database)	Tiket relevan dengan kata kunci ditampilkan
	TCT023	Alternatif 1	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan tombol search 4. User memasukkan kata kunci pencarian, klik search	Input karakter keyword pencarian (tidak terdapat dalam database)	Muncul pemberitahuan tiket tidak ditemukan

T-15	TCT024	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list	-	User masuk ke tampilan detail tiket
	TCT025	Alternatif 1	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan tombol search 4. User memasukkan kata kunci pencarian, klik search 5. User menekan salah satu tiket dalam list	-	User masuk ke tampilan detail tiket

T-16	TCT026	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan tombol tambah balasan 5. User memasukkan seluruh isian balasan 6. User menekan tombol konfirmasi balas tiket	Input karakter dan file balasan tiket	Informasi balasan berhasil dibuat ditampilkan

T-17	TCT027	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan tombol hapus tiket 4. User menekan tombol lanjutkan konfirmasi hapus tiket	-	Informasi tiket berhasil dihapus ditampilkan

T-18	TCT028	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan tombol hapus tiket 5. User menekan tombol lanjutkan konfirmasi hapus tiket	Input karakter dan file balasan tiket	Informasi tiket berhasil dihapus ditampilkan

T-19	TCT029	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan tombol lihat history tiket	-	History tiket ditampilkan
T-20	TCT030	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan icon tombol lihat informasi client	-	Data client ditampilkan

T-21	TCT031	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan tombol assign tiket 5. User memasukkan kata kunci pencarian username dan menekan tombol cari 6. Username ditemukan, user memilih	Inputan karakter keyword pencarian username (terdapat dalam database)	Informasi assign tiket berhasil dilakukan ditampilkan

			username yang dituju 7. User menekan tombol lanjutkan konfirmasi assign tiket		
	TCT032	Alternatif 1	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan tombol assign tiket 5. User memasukkan kata kunci pencarian username dan	Inputan karakter keyword pencarian username (tidak terdapat dalam database)	Username tidak ditemukan

			menekan tombol cari		
T-22	TCT033	Normal	6. User menekan tombol menu 7. User menekan menu tiket 8. User menekan salah satu tiket dalam list 9. User menekan tombol assign tiket 10. User menekan tombol assign myself 11. User menekan tombol lanjutkan konfirmasi assign tiket	-	Informasi assign tiket berhasil dilakukan ditampilkan

T-23	TCT034	Normal	1. User menekan tombol menu 2. User menekan menu tiket 3. User menekan salah satu tiket dalam list 4. User menekan tombol change status tiket 5. User memilih status tiket yang diinginkan 6. User menekan tombol lanjutkan konfirmasi ubah status tiket	-	Informasi ubah status tiket berhasil dilakukan ditampilkan