Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IS184853

RANCANG BANGUN SISTEM PELAPORAN PDDIKTI SEMI-OTOMATIS BERBASIS WEB DAN ANTRIAN DI ITS: MODUL WEB

DESIGN OF WEB-BASED AND QUEUING SEMI-AUTOMATIC PDDIKTI REPORTING SYSTEM AT ITS: WEB MODULE

MUHAMMAD ALWAN WICAKSONO
NRP 05211740000008

Dosen Pembimbing:
Radityo Prasetianto.W, S.Kom, M.Kom

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

Halaman ini sengaja dikosongkan

TUGAS AKHIR – IS184853

RANCANG BANGUN SISTEM PELAPORAN PDDIKTI SEMI-OTOMATIS BERBASIS WEB DAN ANTRIAN DI ITS: MODUL WEB

MUHAMMAD ALWAN WICAKSONO
NRP 05211740000008

Dosen Pembimbing:

Radityo Prasetianto.W, S.Kom, M.Kom

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

Halaman ini sengaja dikosongkan

FINAL PROJECT – IS184853

DESIGN OF WEB-BASED AND QUEUING SEMI-AUTOMATIC PDDIKTI REPORTING SYSTEM AT ITS: WEB MODULE

MUHAMMAD ALWAN WICAKSONO
NRP 05211740000008

Supervisor:

Radityo Prasetianto.W, S.Kom, M.Kom

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2021

Halaman ini sengaja dikosongkan

LP/P/21/095

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM PELAPORAN PDDIKTI SEMI-OTOMATIS BERBASIS WEB DAN ANTRIAN DI ITS: MODUL WEB

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer (S.Kom)
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas (ELECTICS)
Institut Teknologi Sepuluh Nopember

Oleh

Muhammad Alwan Wicaksono
05211740000111

Surabaya, 17 Agustus 2021

Kepala Departemen Sistem Informasi

LP/P/21/095

Dr. Mudjahidin, ST., MT.
NIP. 197010102003121001

LEMBAR PERSETUJUAN

RANCANG BANGUN SISTEM PELAPORAN PDDIKTI SEMI-OTOMATIS BERBASIS WEB DAN ANTRIAN DI ITS: MODUL WEB

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Oleh :

Muhammad Alwan Wicaksono
NRP: 05211740000111

Disetujui Tim Penguji: Tanggal Ujian: 10 August 2021
Periode Wisudas: September 2021

Radityo Prasetianto.W, S.Kom, M.Kom (Pembimbing 1)

Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D (Penguji 1)

Renny Pradina, S.T, M.T (Penguji 2)

RANCANG BANGUN SISTEM PELAPORAN PDDIKTI SEMI-OTOMATIS BERBASIS WEB DAN ANTRIAN DI ITS: MODUL WEB

Nama Mahasiswa : Muhammad Alwan Wicaksono
NRP : 05211740000111
Jurusan : Sistem Informasi FTEIC-ITS
Pembimbing 1 : Radityo Prasetyanto W., S.Kom, M. Kom

ABSTRAK

Pangkalan Data Perguruan Tinggi (PDDikti) merupakan sistem yang mengelola data perguruan tinggi yang terintegrasi secara nasional. PDDikti mengumpulkan data mengenai penyelenggaraan pendidikan tinggi untuk dimanfaatkan sebagai sistem pendukung keputusan untuk menunjang pembangunan pendidikan tinggi di Indonesia. Jika ada perguruan tinggi yang terlambat melakukan pengiriman data berkala kepada PDDikti, data yang dikirimkan salah, atau tidak valid, maka akan dikenai sanksi sesuai dengan ketentuan peraturan perundang-undangan yang berlaku.

*Untuk mengirimkan data yang terbaru dan valid, perguruan tinggi menggunakan sistem pelaporan PDDikti. Institut Teknologi Sepuluh Nopember (*ITS*) Surabaya telah menggunakan sistem pelaporan, namun aplikasi tersebut dibuat oleh vendor luar. Saat ini operator dari sistem tersebut diserahkan kepada setiap departemen di* ITS. *Sebagai operator, banyak sekali aktivitas yang harus dilakukan sebelum bisa mengirim data ke Feeder PDDikti. Di antaranya, melakukan sinkronisasi data dengan database* ITS*, validasi data, membandingkan data di PDDikti dengan Sistem Informasi Manajemen (SIM)* ITS*, lalu melakukan pengiriman data. Dari aktivitas tersebut, risiko kesalahan pengiriman data sangat tinggi.*

Tugas akhir ini menawarkan solusi berupa sistem yang dapat melakukan otomatisasi beberapa aktivitas dari sistem yang

ada, yaitu pengambilan data, sinkronisasi, dan validasi. Solusi yang ditawarkan adalah sebuah aplikasi yang menerapkan sistem antrian sehingga semua aktivitas sinkronisasi dan validasi data dapat diotomatisasi. Solusi selanjutnya adalah pembuatan dashboard sebagai bentuk pemantauan pelaporan data PDDikti. Metode pengembangan perangkat lunak yang digunakan untuk perencanaan dan pengembangan baik aplikasi maupun dashboard, keduanya menggunakan metode RAD. Tahapan yang digunakan dari metode RAD adalah perencanaan kebutuhan pengguna, dilanjutkan dengan perancangan dan pengembangan serta pengujian sistem.

Hasil yang didapat adalah hasil usability testing pada prototipe aplikasi memiliki rata-rata keberhasilan 98% secara keseluruhan. Pada prototipe dashboard, rata-rata keberhasilan hasil usability testing yang didapatkan adalah 87%. Partisipan test juga berpendapat bahwa fitur sistem antrian lebih baik dari aplikasi yang sekarang, sedangkan dashboard menurut partisipan test sangat membantu dalam memantau pelaporan data.

Kata kunci: Feeder PDDikti, Pelaporan, Rancang Bangun, Perguruan Tinggi

DESIGN OF WEB-BASED AND QUEUING SEMI-AUTOMATIC PDDIKTI REPORTING SYSTEM AT ITS: WEB MODULE

Student Name	**: Muhammad Alwan Wicaksono**
NRP	**: 05211740000111**
Department	**: Information Systems FTEIC-ITS**
Supervisor 1	**: Radityo Prasetyanto W., S.Kom, M. Kom**

ABSTRACT

The Higher Education Database (PDDikti) is a system that manages university data that is integrated nationally. PDDikti collects data on the implementation of university education to be used as a decision support system to support the development of higher education in Indonesia. If there are universities that are late in sending periodic data to PDDikti, the data sent is wrong, or invalid, they will be subject to sanctions in accordance with the provisions of the applicable laws and regulations.

To submit the latest and valid data, universities use the PDDikti reporting system. Sepuluh Nopember Institute of Technology (ITS) Surabaya has used a reporting system, but the application was made by an outside vendor. Currently, the operator of the system is handed over to each department at ITS. As an operator, there are a lot of activities before the data can be sent to the PDDikti Feeder. Activities like, synchronizing data with the ITS database, validating data, comparing data in PDDikti with the ITS Management Information System (SIM), then sending data. From these activities, the risk of data transmission errors were very high.

This final project offers a solution in the form of a system that can automate several activities from the existing system, namely data retrieval, synchronization, and validation. The solution offered is an application that implements a queuing system so

that all data synchronization and validation activities can be automated. The next solution is to create a dashboard as a form of monitoring PDDikti data reporting. The software development method used for planning and developing both applications and dashboards, both use the RAD method. The stages used from the RAD method are user requirements planning, followed by design and development and system testing.

The results obtained by usability testing on the application prototype have an average success of 98% overall. In the dashboard prototype, the average success of the usability testing results obtained is 87%. The test participants also think that the queuing system feature is better than the current application, while the dashboard according to the test participants is very helpful in monitoring data reporting.

Keyword: Feeder PDDikti, Reporting, Design, University

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Muhammad Alwan Wicaksono
NRP	: 05211740000111
Tempat/Tanggal Lahir	: Surabaya, 26 Maret 1999
Fakultas/Departemen	: FTEIC / Sistem Informasi
Nomor Telp/Hp/E-mail	: muhammadalwan.52@gmail.com

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul:

RANCANG BANGUN SISTEM PELAPORAN PDDIKTI SEMI-OTOMATIS BERBASIS WEB DAN ANTRIAN DI ITS: MODUL WEB

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain,

Apabila di kemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarism, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, 12 Agustus 2021

Muhammad Alwan W.

NRP: 05211740000111

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT yang dengan rahmat dan kuasanya memberikan kekuatan dan kemampuan kepada penulis untuk dapat meyelesaikan tugas akhir yang berjudul "RANCANG BANGUN SISTEM PELAPORAN PDDIKTI SEMI-OTOMATIS BERBASIS WEB DAN ANTRIAN DI ITS: MODUL WEB" sebagai salah satu syarat kelulusan di Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya.

Dengan sangat tulus kami menyampaikan terima kasih kepada pihak-phak yang telah meluangkan waktu, tenaga, dan pikiran serta memberi dukungan kami baik dukungan moril dan materiil termasuk namun tidak terbatas pada bantuan data, doa, saran, dan motivasi. Tanpa bantuan dan dukungan dari pihak-pihak tersebut, tugas akhir ini tidak akan terwujud. Secara khusus, kami menyampaikan rasa terima kasih kepada:

1. Bapak Adi Sujanto dan Ibu Ulih Sumiratning selaku kedua orang tua dari penulis yang selalu memberikan dukungan penuh dan senantiasa mendoakan keberhasilan kami dalam menyelesaikan tugas akhir ini.
2. Bapak Dr. Mudjahidin, S.T., M.T. selaku Ketua Departemen Sistem Informasi ITS Surabaya.
3. Bapak Radityo Prasetianto Wibowo S.Kom., M.Kom, selaku dosen pembimbing dan sebagai narasumber yang senantiasa meluangkan waktu, membagikan ilmu, dan memberikan saran serta memotivasi untuk kelancaran tugas akhir ini.
4. Ibu Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D. dan Ibu Renny Pradina K., S.T., M.T., SCJP selaku dosen penguji yang telah meluangkan waktu untuk memberikan kritik dan saran perbaikan tugas akhir.
5. Ibu Feby Artwodini Muqtadiroh, S.Kom., M.T. dan Bapak Izzat Aulia Akbar, S.Kom., M.Eng., Ph.D. selaku dosen wali dari penulis.
6. Mas M. Dadang S., Bapak Arief Pramono, dan Ibu Inayati Fajriyah, S.Si selaku evaluator aplikasi dan dashboard dari

pihak DPTSI yang telah membantu kami dengan memberikan saran perbaikan selama proses pengembangan.
7. Seluruh dosen dan karyawan Departemen Sistem Informasi ITS yang telah memberikan ilmu yang bermanfaat kepada penulis.
8. Gita Krismurti Romadhani dan Reynata Tri Damayanti yang telah membantu penulis sebagai partisipan pengujian untuk melakukan pengujian aplikasi dan dashboard dan memberikan kritik serta saran agar pengembangan selanjutnya lebih baik.
9. Rizal Maulana Hadi dan Prasetyo Ramadi selaku anggota tim yang mengerjakan penelitian tugas akhir ini.
10. Mas Ahmad Naufal Rofiif, S.Kom., yang telah meluangkan waktu bagi penulis untuk memberikan saran dan perbaikan pada pengembangan kode web.
11. Berbagai pihak yang tidak bisa disebutkan satu persatu yang membantu penulis dalam menyelesaikan tugas akhir.

Penyusunan tugas akhir yang belum sempurna ini masih membutuhkan kritik dan saran yang membangun untuk pengembangan yang lebih baik ke depannya. Diharapkan tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 12 Agustus 2021

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Bab pendahuluan menjelaskan proses identifikasi masalah yang meliputi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, dan manfaat penelitian. Melalui pendahuluan ini diharapkan dapat memberikan gambaran umum tentang permasalahan yang diangkat dan pemecahan masalahnya.

1.1. Latar Belakang

Pangkalan Data Perguruan Tinggi (PDDikti) merupakan sistem yang mengelola data perguruan tinggi yang terintegrasi secara nasional [1]. Dengan data yang terintegrasi, PDDikti menjadi rujukan data yang valid untuk penjaminan mutu perguruan tinggi [2]. PDDikti mengumpulkan data mengenai penyelenggaraan pendidikan tinggi untuk dimanfaatkan sebagai sistem pendukung keputusan untuk menunjang pembangunan pendidikan tinggi di Indonesia.

Kewenangan dan tanggung jawab PDDikti diatur oleh Undang–undang nomor 12 Tahun 2012 tentang Pendidikan Tinggi. Kemudian dipertegas melalui Permenristek Dikti nomor 61 tahun 2016 tentang Pangkalan Data Pendidikan Tinggi, yang kemudian menjadi pedoman utama dalam pengembangan sistem dan pelaporan dalam Tugas Akhir ini. Disebutkan pada Permenristek Dikti nomor 61 tahun 2016 pasal 10 ayat 7, jika ada perguruan tinggi yang terlambat melakukan pengiriman data berkala kepada PDDikti, maka akan dikenai sanksi sesuai dengan ketentuan peraturan perundang-undangan yang berlaku [1]. Begitu pula dengan data yang salah atau tidak valid.

PDDikti sebagai sistem pendukung keputusan membutuhkan data yang valid dan aktual yang digunakan untuk berbagai hal misalnya Penomoran Ijazah Nasional (PIN). Jika data yang diterima oleh PDDikti tidak valid maka mahasiswa yang bersangkutan tidak bisa mendapatkan PIN [3]. Data yang telah diterima oleh PDDikti tidak dapat dihapus, melainkan hanya dapat diubah dengan ketentuan tertentu [1]. Oleh karena itu, diperlukan data yang diperbarui dengan baik. Untuk mengirimkan data yang terbaru dan valid, perguruan tinggi menggunakan sistem pelaporan PDDikti.

Institut Teknologi Sepuluh Nopember (ITS) Surabaya telah menggunakan sistem pelaporan, namun aplikasi tersebut dibuat oleh vendor luar. Saat ini operator dari sistem tersebut diserahkan kepada setiap departemen di ITS. Direktorat Pengembangan Teknologi dan Sistem Informasi (DPTSI) ITS yang dulunya berperan sebagai penyedia program sekaligus operator saat ini hanya berperan sebagai operator teknis dari sistem. Hal ini meringankan beban DPTSI namun menimbulkan dampak negatif terutama bagi departemen. Sebagai operator, banyak sekali aktivitas yang harus dilakukan sebelum bisa mengirim data ke *Feeder* PDDikti. Di antaranya, melakukan instalasi aplikasi, melakukan sinkronisasi data dengan *database* ITS, validasi data, membandingkan data di PDDikti dengan Sistem Informasi Manajemen (SIM) ITS, lalu melakukan pengiriman data. Dari aktivitas tersebut, risiko kesalahan pengiriman data sangat tinggi [3].

Pada tugas akhir ini, penulis menawarkan solusi berupa otomatisasi beberapa aktivitas dari sistem yang ada sekaligus membangun sistem baru yang tidak bergantung pada vendor luar serta tidak membebani komputer pengguna. Aktivitas yang akan dilakukan otomatisasi adalah pengambilan data, sinkronisasi, dan validasi. Untuk dapat menunjang pengelolaan data di ITS dan pengiriman data ke PDDikti, maka penelitian ini ditujukan untuk membangun sistem yang terintegrasi dan terpadu mulai dari pengiriman dan validasi data melalui sistem hingga pembuatan laporan berupa *dashboard*.

1.2. Rumusan Masalah

Berdasarkan latar belakang tersebut, didapatkan rumusan masalah sebagai berikut:

1. Bagaimana sistem yang dibuat dapat memenuhi kebutuhan pelaporan data ITS ke PDDikti?
2. *Dashboard* seperti apa yang diperlukan untuk memantau pelaporan data ITS ke PDDikti?
3. Bagaimana hasil pengujian sistem terhadap kebutuhan pengguna?

1.3. Batasan Masalah

Adapun dari latar belakang dan rumusan masalah yang telah diuraikan, Tugas Akhir ini memiliki batasan permasalahan sebagai berikut:

1. Data yang digunakan adalah data yang dibutuhkan oleh PDDikti dan berasal dari *database* ITS.
2. Teknologi yang digunakan untuk pengembangan sistem adalah teknologi yang disepakati oleh DPTSI ITS.
3. Pengembangan sistem yang dilakukan adalah sampai tahap pembuatan prototipe.

1.4. Tujuan Penelitian

Berdasarkan rumusan masalah dan Batasan masalah yang telah diuraikan, maka tujuan yang ingin dicapai oleh penelitian ini adalah:

1. Membuat sistem yang dapat melaporkan data PDDikti secara otomatis.
2. Membuat *dashboard* untuk pemantauan yang diperlukan untuk memantau pengiriman data ITS ke PDDikti.
3. Melakukan pengujian sistem terhadap kebutuhan pengguna.

1.5. Manfaat Penelitian

Berikut merupakan manfaat yang diperoleh dengan diselesaikannya tugas akhir ini:

1. Bagi ITS, sistem pelaporan ini dapat digunakan untuk pengiriman data PDDikti yang lebih mudah dan efisien bagi DPTSI maupun departemen dibandingkan sistem yang ada saat ini. Kemudian *dashboard* yang dibuat dapat menjadi pertimbangan dalam pengambilan keputusan terkait peningkatan kualitas pendidikan perguruan tinggi.
2. Bagi penulis, mendapatkan wawasan terkait permasalahan teknologi informasi yang ada di lingkungan ITS. Dengan ini penulis mendapatkan pengetahuan mengenai pengembangan teknologi yang dapat memenuhi kebutuhan pengguna mulai dari penggalian kebutuhan hingga ke pengujian sistem.

Selain itu, tugas akhir ini dapat digunakan sebagai referensi untuk penelitian lebih lanjut terkait implementasi sistem pelaporan PDDikti.

1.6. Relevansi

Topik yang diangkat dalam tugas akhir ini adalah rancang bangun aplikasi dan *monitoring dashboard* sebagai media penyampaian informasi kepada *stakeholder*. Tugas akhir ini relevan dengan topik pada laboratorium Akuisisi Data dan Diseminasi Informasi serta memiliki keterkaitan dengan matakuliah yang penulis pelajari selama perkuliahan yaitu Manajemen Basis Data, Teknologi Basis Data, Visualisasi Informasi, Rekayasa Kebutuhan Perangkat Lunak, dan Rancang Bangun Perangkat Lunak.

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan penelitian sebelumnya yang memiliki keterkaitan dengan topik tugas akhir untuk dijadikan acuan dalam pengerjaan tugas akhir ini. Bab ini juga berisi landasan teori yang menunjang pengerjaan tugas akhir.

2.1. Penelitian Sebelumnya

Terdapat beberapa penelitian yang memiliki topik serupa serta menjadi acuan dalam penelitian ini yang dijelaskan pada Tabel 2.1, Tabel 2.2, Tabel 2.3, dan Tabel 2.4.

Tabel 2.1 Penelitian Sebelumnya

Judul	Implementasi *Web Service* pada Integrasi Data Akademik dengan Replika Pangkalan Data Dikti
Nama Peneliti	Rifki Indra Perwira, Budi Santosa
Tahun	2017
Keterkaitan	Pada penelitian ini pekerjaan yang perlu dilakukan mirip yaitu *mapping*, proses *Read Insert Update Delete*, dan sinkronisasi data. Penelitian ini membahas implementasi dari penelitian sebelumnya yang dilakukan oleh penulis yang sama.
Sumber	[4]

Tabel 2.2 Penelitian Sebelumnya

Judul	Integrasi Data Akademik Dengan Aplikasi *Feeder* PDDIKTI Berbasis *Web service*
Nama Peneliti	Slamet Widodo, Herlambang Brawijaya, Samudi, Endang Retnoningsih
Tahun	2017

Keterkaitan	Pada penelitian ini tahapan analisis kebutuhan mirip dengan Tugas Akhir karena memang menggunakan sistem yang sama yaitu sistem dari PDDikti.
Sumber	[5]

Tabel 2.3 Penelitian Sebelumnya

Judul	Desain dan Implementasi *Services Provider* Berbasis *Web Services Push* Pangkalan Data Perguruan Tinggi pada Sistem Informasi Akademik Politeknik Negeri Lampung
Nama Peneliti	Eko Win Kenali, Halim Fathoni
Tahun	2014
Keterkaitan	Pada penelitian ini salah satu fokusnya adalah untuk dapat melakukan pengiriman data dengan baik. Untuk mencapai tujuan tersebut juga diperlukan desain dari tabel asal dan kueri yang cocok dengan data di PDDikti.
Sumber	[6]

Tabel 2.4 Penelitian Sebelumnya

Judul	Pembuatan Sistem Absensi Berbasis Web dengan Menerapkan Teknologi *Progressive Web Apps* dan Metode *Action Design Research* (Studi Kasus: SMP Al Azhar 13 Surabaya)

Nama Peneliti	Muhammad Khotib
Tahun	2019
Keterkaitan	Pada Tugas Akhir ini menggunakan salah satu metode pengembangan aplikasi yaitu metode *Rapid Application Design* yang cocok dengan *project* yang memiliki *requirement* yang terus berkembang seiring tahapan pengembangan aplikasi tersebut. Metode ini memungkinkan beberapa kali evaluasi dan iterasi pada tahap pengembangannya.
Sumber	[7]

2.2. Landasan Teori

2.2.1. *Feeder* PDDikti

Mengacu pada Permenristek Dikti nomor 61 tahun 2016 tentang Pangkalan Data Pendidikan Tinggi, *Feeder* PDDikti adalah perangkat lunak yang digunakan untuk pelaporan data perguruan tinggi ke PDDikti dan ditempatkan di masing-masing perguruan tinggi. *Feeder* PDDikti akan melakukan sinkronisasi data dari perguruan tinggi kemudian mengirimkan data tersebut ke PDDikti [1]. Data yang dilaporkan oleh perguruan tinggi harus sesuai dengan format pelaporan yang telah disediakan sehingga setiap perguruan tinggi harus menyediakan instrumen yang dapat menyesuaikan data dari sistem informasi akademik perguruan tinggi, kepegawaian, dan sistem informasi lainnya dengan data yang ada di PDDikti. *Feeder* PDDikti menyediakan *Application Programming Interface* (*API*) untuk kepentingan pengembangan aplikasi penghubung antara SIM di perguruan tinggi dengan *Feeder* PDDikti. Dengan bantuan API, aplikasi penghubung dapat mengakses dan merubah data PDDikti melalui *Feeder*. Data dikirimkan atau diubah dalam bentuk *JSON* menggunakan *method*

get untuk mendapatkan data, *delete* untuk menghapus data, *insert* untuk menambahkan data baru, dan *update* untuk modifikasi data yang sudah ada. Setiap *method* tersebut memiliki parameternya masing-masing.

2.2.2. *Dashboard*

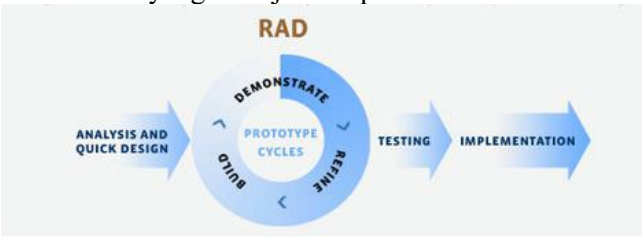
Stephen Few menyebutkan dalam bukunya bahwa *dashboard* adalah sebuah mekanisme yang menyajikan tampilan grafis tentang suatu hal serta berusaha menyediakan wawasan bagi penggunanya mengenai sesuatu yang telah terjadi, sedang terjadi, atau yang mungkin terjadi. *Dashboard* adalah tampilan visual, oleh karena itu semua informasi yang ada pada *dashboard* disajikan dalam bentuk visual dengan kombinasi teks dan grafik. Selain itu, Ada beberapa poin utama tentang ciri-ciri *dashboard* yang diungkapkan oleh Stephen Few, yaitu:

a. *Dashboard* menampilkan informasi yang dibutuhkan untuk mencapai tujuan tertentu.
b. Ukuran *dashboard* dibuat menyesuaikan dengan ukuran layar komputer.
c. *Dashboard* tidak selalu harus ditampilkan dalam web *browser* melainkan dapat disajikan dalam media yang lain.
d. *Dashboard* digunakan untuk mendapatkan wawasan dengan cepat tanpa perlu melihat keseluruhan data yang ada [8].

Dalam penelitiannya, Andrea Janes menyatakan bahwa *dashboard* yang ideal adalah *dashboard* yang mendukung pengguna dalam mencapai tujuannya. Namun kebanyakan *dashboard* justru dibuat dengan untuk menyajikan visualisasi informasi sebanyak mungkin tanpa memperhatikan hal kunci dalam pemenuhan tujuan pengguna. Untuk mendapatkan *dashboard* yang ideal, ada dua aspek yang perlu dipertimbangkan yaitu: memilih data yang benar kemudian memilih teknik visualisasi yang benar [9].

2.2.3. Metode *Rapid Application Development* (*RAD*)

RAD (*Rapid Application Development*) merupakan salah satu model pendekatan *SDLC* (*Software Development Life Cycle*) yang dirancang untuk menghasilkan produk berkualitas tinggi dengan cepat serta menggunakan strategi seperti *protoyping*. *RAD* sangat cocok dengan *project* yang memiliki *requirement* yang terus berkembang seiring pengembangan aplikasi tersebut. Secara umum *RAD* terdiri dari 3 tahapan yaitu tahap perencanaan dan analisa kebutuhan, tahap perancangan atau pengembangan, serta tahap pengujian dan implementasi [10]. Metode ini memungkinkan untuk melakukan iterasi pada tahap pengembangannya. Pada setiap iterasi pengembang akan mendapatkan *feedback* dari *client* secara berulang. Iterasi akan terus berlanjut hingga terjadi kesepakatan bahwa sistem siap untuk diimplementasikan [11]. Berikut merupakan alur dari metode *RAD* yang ditunjukkan pada Gambar 2.1.



Gambar 2.1 Alur Metode *RAD* [12]

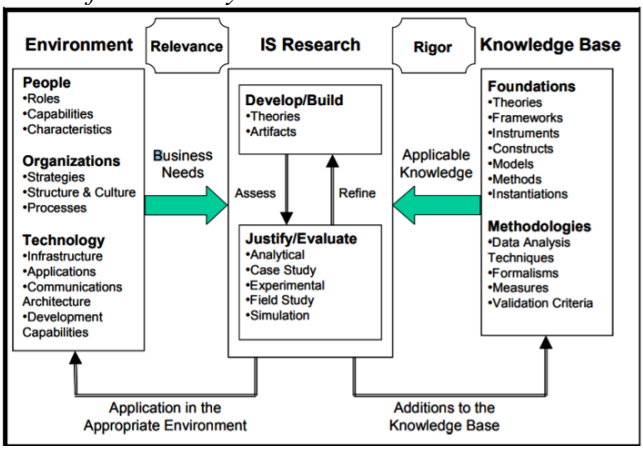
RAD secara umum terdiri dari 3 tahap yaitu:

1. *Requirements Planning*
   Pada tahap ini akan dilakukan identifikasi serta analisa kebutuhan dan tujuan sistem beserta syarat-syarat yang muncul yang muncul dari tujuan tersebut. Pengguna dan analis/pengembang akan bertemu untuk mengidentifikasi tujuan sistem.
2. Perancangan dan Pengembangan

Pada tahap ini pengembang dan pengguna akan bekerjasama untuk merancang dan membangun sistem. Tahap ini memungkinkan dilakukan iterasi untuk terus menyempurnakan prototipe sistem. Pengembang akan mendapatkan *feedback* dan evaluasi dari pengguna pada setiap iterasi hingga sistem siap untuk diimplementasikan.

3. Pengujian dan Implementasi
   Tahap ini merupakan tahap terakhir dari *RAD,* ketika sistem telah disetujui pengguna, selanjutnya akan diuji dan diimplementasikan.

2.2.4. *Information System Research Framework*



Gambar 2.2 *IS Research Framework [13]*

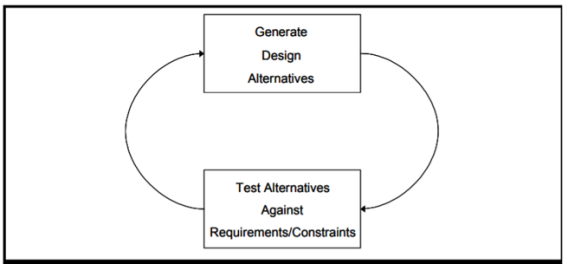
Information System Research Framework merupakan kerangka konseptual untuk memahami, melaksanakan, dan mengevaluasi *Information System Research* yang menggabungkan paradigma *behaviour science* dan *design science* [13]. Seperti yang ditunjukkan pada Gambar 2.2, dalam *Information System Research*, *Environment* mendefinisikan ruang permasalahan yang terdiri dari orang, organisasi (bisnis), serta teknologi yang ada atau direncanakan. Sedangkan *Knowledge Base*

menyediakan bahan mentah agar *Information System Research* dapat dicapai. *Knowledge Base* terdiri dari *Foundations* termasuk teori dasar, kerangka kerja, instrumen, konstruksi, model, metode, dan instantiasi yang digunakan dalam tahap pengembangan, serta *Methodologies* yang merupakan pedoman yang digunakan dalam fase evaluasi. Information *Research* dilakukan dalam dua fase yang saling melengkapi, yaitu *behaviour science* yang membahas penelitian melalui pengembangan dan pembenaran teori yang menjelaskan atau memprediksi fenomena yang terkait dengan kebutuhan bisnis yang diidentifikasi serta *design science* yang membahas penelitian melalui pembangunan dan evaluasi artefak yang dirancang untuk memenuhi kebutuhan bisnis. Terdapat 7 *guideline* dalam perspektif *design science* pada *Information System Research* yaitu:

1. *Guideline 1*: *Design as an Artifact*
   Design-science research harus menghasilkan artefak yang layak dalam bentuk konstruk, model, metode, atau instantiasi. Artefak dibangun untuk mengatasi masalah yang belum terpecahkan. Artefak TI dibuat untuk mengatasi masalah organisasi.
2. *Guideline 2*: *Problem Relevance*
   Tujuan dari *design-science research* adalah untuk mengembangkan solusi berbasis teknologi untuk masalah bisnis yang penting, relevan, dan belum terpecahkan. Secara formal, masalah dapat didefinisikan sebagai perbedaan antara keadaan tujuan dan keadaan sistem saat ini.
3. *Guideline 3*: *Design Evaluation*
   Utilitas, kualitas, dan kemanjuran desain artefak harus ditunjukkan secara ketat melalui metode evaluasi yang dijalankan dengan baik. Evaluasi adalah komponen penting dari proses penelitian. Artefak TI dapat dievaluasi dalam hal fungsionalitas, kelengkapan, konsistensi, akurasi, kinerja, keandalan, kegunaan, kesesuaian dengan

organisasi, dan atribut kualitas relevan lainnya. Karena desain secara inheren merupakan aktivitas berulang dan bertahap, fase evaluasi memberikan umpan balik penting untuk fase konstruksi mengenai kualitas proses desain dan produk desain yang sedang dikembangkan.

4. *Guideline 4*: *Research Contributions*
   Design-science research yang efektif harus memberikan kontribusi yang jelas dan dapat diverifikasi di bidang desain artefak, desain fondasi, dan/atau desain metodologi.
5. *Guideline 5*: *Research Rigor*
   Design-science research bergantung pada penerapan metode yang ketat baik dalam konstruksi dan evaluasi artefak desain.
6. *Guideline 6: Design as a Search Process*
   Pencarian artefak yang efektif membutuhkan pemanfaatan sarana yang tersedia untuk mencapai tujuan yang diinginkan sambil memenuhi hukum di lingkungan masalah. Ilmu desain pada dasarnya bersifat iteratif. Pencarian untuk desain terbaik, atau optimal, sering kali sulit dipecahkan untuk masalah sistem informasi yang realistis. Strategi pencarian heuristik menghasilkan desain yang layak dan bagus yang dapat diimplementasikan dalam lingkungan bisnis. Simon (1996) menggambarkan sifat proses desain sebagai siklus pada Gambar 2.3.

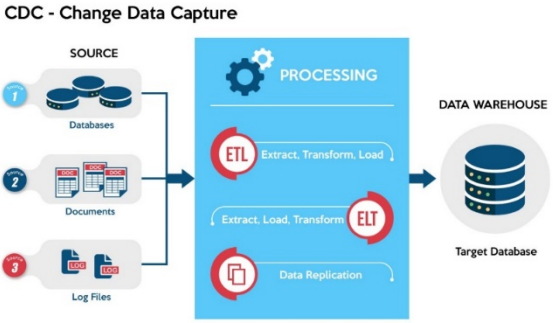


Gambar 2.3 *Design as a Search Process* [13]

7. *Guideline 7*: *Communication of Research*
   Design-science research harus disajikan secara efektif baik untuk audiens yang berorientasi teknologi maupun yang berorientasi manajemen.

2.2.5. *Change Data Capture* (*CDC*)

Change Data Capture (*CDC*) adalah sebuah pendekatan untuk merekam data dalam sebuah database dan hanya data yang berubah saja yang direkam [14]. *CDC* merekam perubahan data seperti *insert*, *update*, dan *delete* yang terjadi di dalam *database relational* [15]. Data yang berubah tersebut kemudian dapat direplikasi ke *database* lain sesuai kebutuhan. Secara visual, cara kerja *CDC* ditunjukkan dengan Gambar 2.4.



Gambar 2.4 Cara Kerja *CDC* [14]

2.2.6. 5 *Why's Analysis*

5 *Why's Analysis* merupakan salah satu teknik dalam melakukan *Root Cause Analysis* yang digunakan untuk menyelesaikan suatu permasalahan dengan mencari akar dari permasalahan tersebut. Teknik ini dilakukan dengan menanyakan "mengapa" sebanyak setidaknya lima kali [16] . Namun pada penerapannya tidak selalu dilakukan dengan bertanya lima kali, bisa kurang atau lebih sampai mendapatkan akar dari permasalahan yang dicari. Seperti yang disampaikan oleh Robert B. Pojasek, ada empat tahap dalam menggunakan teknik ini, yaitu:

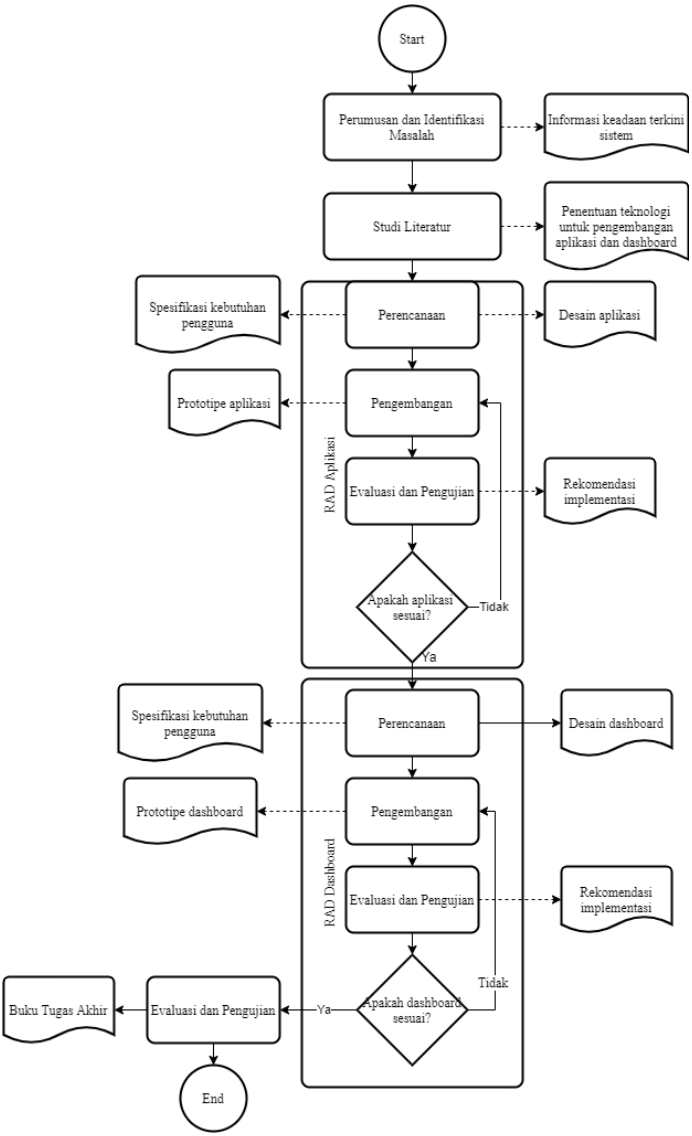
1. Tahap pertama adalah mendiskusikan permasalahan yang akan diselesaikan dengan anggota tim. Dari permasalahan tersebut kemudian ditentukan apakah membutuhkan bantuan pihak lain untuk memahaminya atau mengambil lebih banyak waktu untuk melakukan observasi.
2. Menanyakan "mengapa" pertama untuk mengetahui mengapa permasalahan tersebut bisa atau harus terjadi.
3. Menanyakan lebih banyak pertanyaan "mengapa" untuk setiap jawaban yang didapat.
4. Memilah jawaban-jawaban dari semua pertanyaan "mengapa" tersebut yang memiliki kemungkinan menjadi akar dari permasalahan yang ingin diselesaikan dan mendiskusikannya dengan anggota tim [16].

BAB III
METODE TUGAS AKHIR

Pada bab ini akan dibahas metodologi yang akan digunakan dan jadwal kegiatan dalam pengerjaan Tugas Akhir. Selain itu, dalam bab ini juga dibahas arsitektur sistem baik yang saat ini maupun arsitektur sistem baru yang diusulkan.

3.1. Tahap Pelaksanaan Tugas Akhir

Diagram metodologi untuk pengerjaan Tugas Akhir ini ditunjukkan dengan Gambar 3.1.



Gambar 3.1 Diagram Metodologi

Berikut adalah uraian setiap tahap pengerjaan tugas akhir.

3.1.1. Perumusan dan identifikasi permasalahan

Tahap ini merupakan tahap pertama yang dilakukan terkait dengan mengidentifikasi kondisi eksisting dari sistem yang ada dan masalah apa yang ada pada sistem saat ini. Tahap ini dilakukan dengan cara menggali informasi menggunakan metode wawancara kepada pihak DPTSI selaku operator teknis sistem *Feeder* saat ini. Selain itu juga untuk mencari tahu cara kerja dari sistem *Feeder* PDDikti berdasarkan dokumen – dokumen yang ada. Hasil dari tahapan ini adalah informasi mengenai keadaan terkini dari pelaporan PDDikti oleh ITS. Tahap ini juga mengacu pada *guideline 2 Problem Relevance* pada *design science IS Reseaech Framework* yang membuat peneliti menggali permasalahan yang ada pada kondisi eksisting sebagai acuan untuk pembuatan artefak yang dapat menjawab permasalahan.

3.1.2. Studi literatur

Studi literatur dilakukan untuk mencari alternatif solusi untuk permasalahan yang ada. Termasuk penggunaan teknologi yang diperlukan dalam pembangunan sistem. Studi literatur ini dimulai dengan mengidentifikasi alternatif teknologi yang mungkin dapat digunakan, dilanjutkan dengan mengumpulkan dan mempelajari literatur yang didapat. Langkah akhir dalam tahap ini adalah menentukan teknologi yang digunakan untuk pengembangan aplikasi dan *dashboard*.

3.1.3. Perencanaan aplikasi dan *dashboard*

Tahap ini dibagi menjadi dua yaitu perencanaan aplikasi meliputi web dan sistem antrian serta perencanaan *dashboard,* namun dilakukan dalam waktu yang hampir bersamaan. Meskipun dibagi dua, namun aktivitas yang dilakukan sama yaitu penggalian kebutuhan dan pembuatan desain, baik untuk pengembangan aplikasi maupun untuk pengembangan *dashboard*. Hasil dari tahapan ini adalah desain sistem untuk aplikasi dan desain *dashboard*.

3.1.4. Pengembangan web dan sistem antrian

Tahap pengembangan untuk aplikasi (web dan sistem antrian) serta *dashboard* yang mengadopsi metode *RAD (Rapid Application Development)* serta mengacu pada *guideline 6 Design as a Search Process* pada *design science IS Reseaech Framework* mulai dipisah, pada tahap ini karena pembuatan *dashboard* bisa dilakukan setelah aplikasi selesai. Dalam tahap ini, desain sistem yang telah dibuat sebelumnya akan diimplementasikan ke dalam baris kode dan pada setiap pengembangan akan dilakukan evaluasi terhadap prototipe aplikasi yang telah dibuat dimana hal ini mengacu pada *guideline 5 Research Rigor* pada *design science IS Reseaech Framework* yaitu penerapan metode yang baik dalam konstruksi dan evaluasi pada desain artefak. Setiap iterasi memiliki target capaian pengembangan seperti berikut:

a. Iterasi pertama berkaitan dengan *UI (User Interface)* atau tampilan muka dari web *Feeder Bridge*. Aplikasi harus dapat menampilkan data apa saja yang akan dikirim sebelum pengguna dapat melakukan persetujuan data. Aspek keberhasilan iterasi ini adalah keberhasilan pengguna dalam menggunakan aplikasi.
b. Iterasi kedua adalah aplikasi dapat melakukan sinkronisasi. Aspek keberhasilan pada iterasi ini adalah aplikasi diharapkan dapat melakukan pengambilan data dari sumber data SIAKAD dan PDDikti yang kemudian disimpan dalam *database Feeder Bridge*.
c. Iterasi ketiga aplikasi dapat melakukan pelakukan pengiriman data.

   Tujuan utama dari pembangunan aplikasi adalah untuk bisa mengirimkan data ke *Feeder* PDDikti, sehingga fitur ini menjadi salah satu fitur penting dalam aplikasi. Aspek keberhasilan iterasi ini adalah aplikasi dapat mengirimkan data ke *Feeder* PDDikti tanpa kendala dan dapat menyimpan respon dari pengiriman data tersebut.

d. Iterasi terakhir adalah terkait dengan efektifitas aplikasi dalam mempermudah proses pengiriman data oleh Departemen di ITS ke *Feeder*. Aspek keberhasilan dari iterasi ini adalah perbandingan antara proses saat ini dengan proses yang baru menjadi lebih baik dan lebih mudah. Aspek lain adalah kepuasan pengguna dalam penggunaan aplikasi yang ditunjukkan dengan hasil dari *testing*.

Hasil dari tahap ini adalah prototipe aplikasi yang siap untuk dilakukan pengujian.

3.1.5. Evaluasi web dan sistem antrian

Tahap evaluasi ini adalah tahap untuk menyesuaikan antara kebutuhan pengguna dengan proses pengembangan web dan sistem antrian. Hasil dari tahapan ini adalah *feedback* atau umpan balik dari pengguna terkait pengembangan aplikasi. Tahap ini berjalan secara paralel dengan tahap pengembangan dimana dilakukan pada setiap akhir proses pengembangan. Pada tahap ini dilakukan pengamatan apakah prototipe sistem aplikasi yang telah dikembangkan sebelumnya sesuai dengan pola proses bisnis dan kebutuhan dari pengguna.

3.1.6. Pengembangan *dashboard*

Sama halnya dengan pengembangan web dan sistem antrian, pada tahap pengembangan *dashboard* akan ada perulangan pada tahap pengembangan prototipe dan evaluasi. Berikut rencana iterasi dalam pengembangan prototipe *dashboard* beserta target capaian dalam setiap iterasinya, antara lain:

a. Iterasi pertama adalah melakukan konfigurasi terhadap teknologi yang digunakan.
b. Iterasi kedua menghasilkan *dashboard* yang menampilkan kelompok data beserta departemen yang sudah mengirimkan dan yang belum.
c. Iterasi selanjutnya menghasilkan *dashboard* yang menampilkan jumlah kelompok data yang sudah

dikirimkan dan yang belum untuk masing-masing departemen.

Hasil dari tahap pengembangan ini adalah prototipe *dashboard* yang siap untuk dilakukan pengujian.

3.1.7. Evaluasi *dashboard*

Tahap evaluasi ini adalah tahap untuk menyesuaikan antara kebutuhan pengguna dengan proses pengembangan *dashboard*. Hasil dari tahapan ini adalah *feedback* dari pengguna terkait pengembangan *dashboard*. Tahap ini berjalan secara paralel dengan tahap pengembangan *dashboard* dimana dilakukan pada setiap akhir proses pengembangan. Pada tahap ini dilakukan pengamatan apakah prototipe *dashboard* yang telah dikembangkan sebelumnya sudah sesuai serta mampu menjawab kebutuhan pengguna.

3.1.8. Pengujian

Pengujian yang dilakukan dibagi menjadi dua untuk aplikasi (web dan sistem antrian) serta *dashboard*. Aktivitas pengujian yang dilakukan adalah pengujian *Usability Testing*, pengujian fungsional, dan pengujian performa baik dari sisi *client* maupun dari sisi *server*. Hasil dari tahap ini adalah rekomendasi – rekomendasi terkait implementasi web dan *dashboard* yang dapat digunakan untuk penelitian selanjutnya. Selain itu juga dilakukan pengujian performa aplikasi dari sisi beban dan waktu aplikasi menyelesaikan tugas.

3.1.9. Penulisan tugas akhir

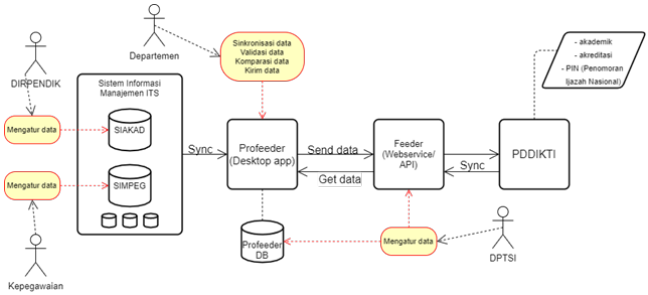
Tahap akhir dari penelitian ini adalah penulisan buku Tugas Akhir. Buku ini berisi hasil pengembangan sistem dan laporan hasil kinerja sistem yang telah dibuat.

3.2. Arsitektur Sistem

Pada bagian ini akan dijelaskan mengenai arsitektur saat ini dan arsitektur baru sebagai usulan dari sistem pelaporan PDDikti yang ada di ITS.

3.2.1. Arsitektur saat ini

Berikut ini adalah arsitektur sistem yang ada saat ini yang ditunjukkan pada Gambar 3.2.

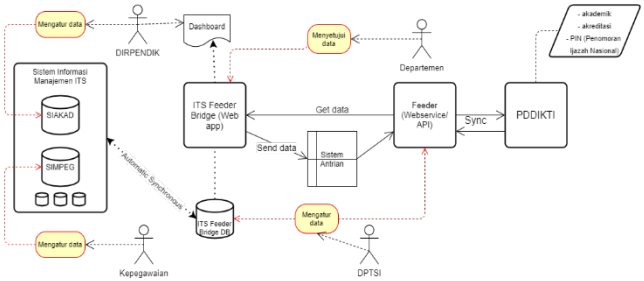


Gambar 3.2 Arsitektur Sistem Saat Ini

Pada arsitektur ini data diambil dari Sistem Informasi Manajemen yang ada di ITS termasuk SIAKAD (Sistem Informasi Akademik) yang dikelola oleh DIRPENDIK ITS (Direktorat Pendidikan) dan SIMPEG (Sistem Informasi Manajemen Kepegawaian) yang dikelola oleh Kepegawaian ITS. Kemudian aplikasi yang digunakan adalah *ProFeeder* yang dikembangkan berbasis *desktop*. Aktivitas yang dilakukan departemen dengan menggunakan aplikasi *ProFeeder* antara lain sinkronisasi data, validasi data, komparasi data, dan kirim data. Setelah itu data dapat dikirim ke *Feeder* PDDikti yang berbasis *web service*/*API* (*Application Programming Interface*) untuk bisa disinkronkan dengan data yang ada pada PDDikti oleh DPTSI selaku operator teknis dan tempat *database* disimpan. Beberapa contoh data yang dimaksud adalah biodata mahasiswa, data konversi nilai, data status keaktifan mahasiswa, data matakuliah dan kurikulum termasuk pemetaan kelas mahasiswa, data mengajar dosen, data bimbingan dan pengujian TA, data perhitungan transkrip, data aktivitas kuliah, data yudisium dan kelulusan mahasiswa, data mahasiswa keluar, dan data pembimbing akademik.

3.2.2. Arsitektur baru

Berikut ini adalah arsitektur baru yang akan dibuat sesuai pengerjaan Tugas Akhir ini yang ditunjukkan pada Gambar 3.3.

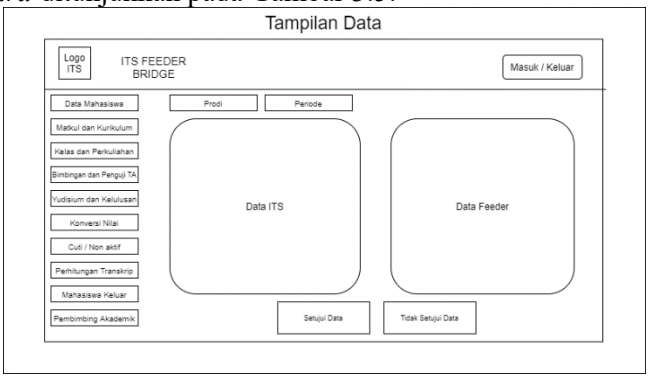


Gambar 3.3 Arsitektur Sistem Baru

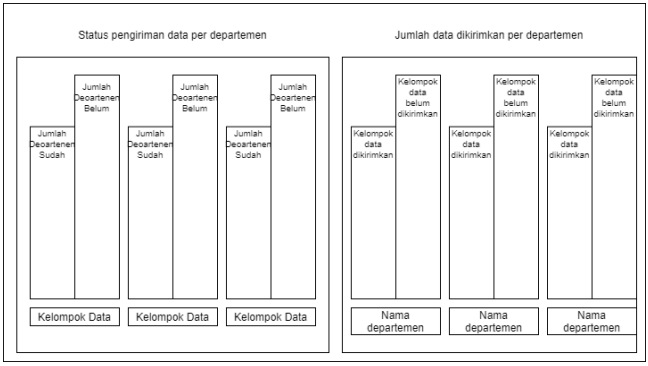
Pada arsitektur baru ini aktivitas yang dilakukan departemen menjadi hanya 1 yaitu menyetujui data. Aktivitas lain seperti validasi data, komparasi data, dan kirim data dilakukan secara otomatis oleh aplikasi baru ITS *Feeder Bridge* yang berbasis aplikasi web. Sinkroniasi data dengan *database* SIM ITS menggunakan konsep *Change Data Capture* (*CDC*) yaitu dengan hanya menangkap data yang berubah. Hal ini berarti jika ada perubahan data pada *database* SIM ITS akan direkam oleh sistem dan disinkronisasikan dengan data dari *Feeder*. Untuk mendukung proses sinkronisasi data tersebut, database yang digunakan ITS *Feeder Bridge* akan menggunakan kombinasi model *database relational* untuk menyesuaikan data dengan *database* di SIM ITS dan *document-based database* untuk menyesuaikan data dari *Feeder* PDDikti. Setelah data disetujui oleh departemen selaku operator, ITS *Feeder Bridge* akan mengirimkan data ke tersebut ke PDDikti menggunakan sistem antrian. Data yang telah siap dikirimkan dari masing-masing departemen tidak langsung dikirimkan namun ditampung terlebih dahulu di dalam sistem antrian untuk dijadwalkan pengiriman datanya dengan algoritma *First In First Out* (*FIFO*) yaitu berdasarkan data mana dulu yang telah siap. Selain itu, dibuat *dashboard* untuk memantau pelaporan data yang dikirim oleh DIRPENDIK, sehingga DIRPENDIK dapat mengawasi secara langsung pelaporan data tanpa perlu menjadi operator.

3.3. Rencana Desain Web dan *Dashboard*

Berdasarkan contoh kelompok data yang telah didefinisikan sebelumnya, dapat dibuat rencana awal untuk desain tampilan aplikasi ditunjukkan. Untuk rencana desain web *Feeder Bridge* ditunjukkan pada Gambar 3.4 dan untuk rencana desain *dashboard* ditunjukkan pada Gambar 3.5.



Gambar 3.4 Gambaran Desain Awal Web



Gambar 3.5 Gambaran Desain Awal *Dashboard*

3.4. Jadwal Kegiatan

Jadwal kegiatan dalam pengerjaan tugas akhir ditunjukkan dengan Tabel 3.1.

Tabel 3.1 Jadwal Pengerjaan Tugas Akhir

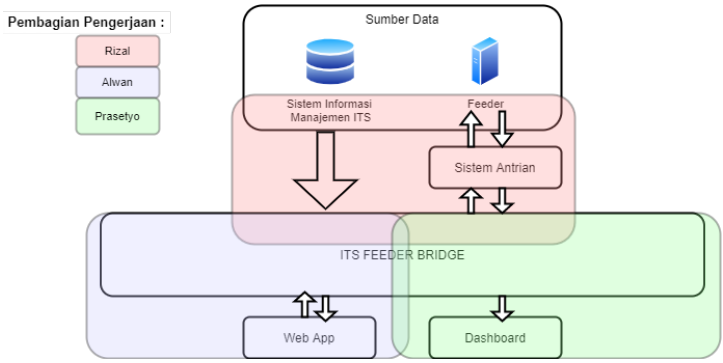
		O	November				Desember				Januari				Februari			
WBS	TASK	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4

1	Perumusan dan Identifikasi Masalah																
2	Studi Literatur																
3	Perencanaan Sistem																
4	Pengembangan Sistem																
5	Evaluasi																
6	Pengujian																
7	Penulisan Tugas Akhir																

		Maret				April				Mei			
WBS	TASK	1	2	3	4	1	2	3	4	1	2	3	4
1	Perumusan dan Identifikasi Masalah												
2	Studi Literatur												
3	Perencanaan Sistem												
4	Pengembangan Sistem												
5	Evaluasi												
6	Pengujian												
7	Penulisan Tugas Akhir												

3.5. Pembagian Pengerjaan

Berdasarkan pada arsitektur baru yang telah dipaparkan, maka cakupan pengerjaan sistem dapat dibagi seperti yang ditunjukkan pada Gambar 3.6.



Gambar 3.6 Diagram Pembagian Cakupan Pekerjaan

Pengerjaan yang terkait pada sumber data sistem seperti *database* SIM ITS dan juga *Feeder* akan dikerjakan oleh Rizal Maulana beserta dengan pembangunan sistem antrian. Untuk pembangunan aplikasi berbasis web pada ITS *Feeder Bridge* akan dikerjakan oleh Muhammad Alwan mulai dari *interface* sistem hingga pengembangan setiap fitur. Kemudian yang terakhir adalah pembangunan *dashboard* untuk *monitoring* pelaporan data PDDikti yang akan dikerjakan oleh Prasetyo Ramadi. Setiap dari penulis akan turut serta dalam pembuatan ITS *Feeder Bridge*.

BAB IV
PERENCANAAN

Pada bab ini akan dijelaskan proses perencanaan dan perancangan web, sistem antrian dan *dashboard* meliputi penggalian kebutuhan dan pembuatan desain.

4.1. Perumusan dan Identifikasi Permasalahan

Tahap awal dalam pengerjaan adalah dengan merumuskan dan menggali permasalahan yang ada. Berikut penjelasan untuk pengumpulan data dan hasilnya.

4.1.1. Pengumpulan data

Pengumpulan data dilakukan dengan metode dokumen dan wawancara kepada pihak DPTSI.

a. Dokumen

Metode dokumen yang dilakukan untuk mempelajari penggunaan aplikasi saat ini oleh departemen. Dokumen yang digunakan adalah *User Guide Web Service* Versi 2.2 yang disediakan oleh Kemenristekdikti. *Web service* tersebut merupakan sistem yang disediakan untuk mempermudah hubungan dari sistem yang ada di perguruan tinggi dengan *Feeder* PDDikti [17]. Adapun *user guide* tersebut berisi petunjuk serta seluruh method yang digunakan untuk mengakses *Feeder* PDDikti baik untuk mendapatkan, menambahkan, mengubah, maupun menghapus data yang ada di *Feeder* PDDikti. Daftar *method* yang disediakan oleh *web service* tersebut dicantumkan dalam **LAMPIRAN B ANALISIS HALAMAN *DASHBOARD***. Selanjutnya penulis juga mendapat informasi tentang penggunaan aplikasi saat ini yaitu *ProFeeder* melalui video tutorial yang diberikan oleh pihak DPTSI. Adapun tutorial yang dimaksud adalah tutorial instalasi *ProFeeder*, melakukan konfigurasi *Virtual Private Network (VPN)*, dan tutorial mengirimkan data.

b. Wawancara
   Wawancara dilakukan dengan memberikan pertanyaan berupa *open-ended question* kepada pihak DPTSI sebagai narasumber.

4.1.2. Rangkuman temuan

Rangkuman temuan berisi temuan-temuan yang didapat dari tahap pengumpulan data baik melalui dokumen maupun wawancara dengan narasumber. Berikut rangkuman yang didapat dari dokumen.

a. *ProFeeder* saat ini dapat melakukan sinkronisasi, validasi, dan pengiriman data dengan baik. Namun belum dapat memberikan detail data ketika terjadi *error* dalam pengiriman.
b. Untuk masing-masing proses yaitu sinkronisasi, validasi, dan pengiriman membutuhkan waktu yang lama dan admin harus menunggu sampai proses selesai untuk dapat lanjut ke proses berikutnya.

Selanjutnya untuk rangkuman temuan dari hasil wawancara dapat dilihat pada Tabel 4.1.

Tabel 4.1 Temuan Hasil Wawancara dan Pengumpulan Data

Nama	M. Dadang S.
Jabatan	Staff Pengelolaan Teknologi Big Data
Keluhan	Perlu perhatian khusus pada data kelas, dosen, dan mahasiswa, karena sering terjadi *error* data pada data – data tersebut
	Penting mengetahui data detail tentang mahasiswa, termasuk sudah berapa semester mahasiswa tersebut ada di ITS
	Masih sering *error* dalam pengiriman data dan tidak ada peringatan penyebab *error*

4.2. Studi Literatur

Pada tahap ini dilakukan studi literatur yang bertujuan untuk menemukan solusi terkait permasalahan yang telah diidentifikasi. *Brainstorming* dilakukan bersama pihak DPTSI untuk menentukan langkah solusi serta penentuan teknologi yang akan digunakan untuk pengembangan aplikasi dan

dashboard. Berikut informasi tentang narasumber seperti yang ditunjukkan pada Tabel 4.2 dan Tabel 4.3.

Tabel 4.2 *Brainstorming* Solusi Narasumber 1

Nama	Radityo Prasetianto Wibowo S.Kom., M.Kom
Jabatan	Kepala Sub Direktorat Pengelolaan Teknologi Big Data
Bidang keahlian	Manajemen Basis Data
Topik yang dibahas saat *brainstroming*	Penggunaan Teknologi sebagai solusi permasalahan

Tabel 4.3 *Brainstorming* Solusi Narasumber 2

Nama	M. Dadang S.
Jabatan	Staff Pengelolaan Teknologi Big Data
Topik yang dibahas saat *brainstroming*	Penggalian permasalahan dan teknis pengimplementasi solusi

Setelah melakukan *brainstorming*, selanjutnya penulis menetukan rekomendasi solusi sesuai dengan hasil *brainstorming*.

4.2.1. Penentuan rekomendasi

Pada langkah ini dilakukan penentuan rekomendasi atas permasalahan yang telah ditemukan saat penggalian permasalahan. Rekomendasi solusi dapat berupa penerapan aplikasi solusi dan penentuan teknologi yang dibutuhkan untuk pengembangan aplikasi. Berikut ini adalah Tabel 4.4 hasil *brainstorming* atas solusi.

Tabel 4.4 Rangkuman Hasil *Brainstorming*

Permasalahan	Solusi
Perhatian khusus pada data kelas, dosen, dan mahasiswa karena	Data kelas, dosen, dan mahasiswa dapat dibuat dan ditampilkan dengan detail. Contohnya data kelas juga

sering terjadi *error* pengiriman data	harus memuat dosen pengampu dan anggota kelas, data mahasiswa juga memuat kelas yang diambil dalam satu semester, dan data dosen juga berisi aktivitas mengajar kelas.
Penting mengetahui detail data mahasiswa termasuk berapa semester yang telah ditempuh	Dapat dibuat fitur tambahan dimana mahasiswa dapat memeriksa datanya sendiri dan melaporkan kesalahan data
Penting mengetahui *error* apa yang terjadi saat pengiriman data	Sistem antrian juga harus dapat menampilkan *error* yang dialami saat pengiriman, baik kesalahan data maupun kesalahan aplikasi.
Pilihan teknologi	Teknologi yang digunakan adalah bahasa pemrograman *Python* untuk membuat *script* antrian dan *ETL*, *CodeIgniter 4* untuk membuat halaman web, template *Dashforge* dari DPTSI untuk desain halaman web, linux *server* dengan *MySQL* untuk *database*, dan *Microsoft Power BI* untuk membuat tampilan *dashboard*

4.3. Perencanaan Web, Sistem Antrian dan *Dashboard*

Pada tahap perencanaan web, sistem antrian dan *dashboard*, penulis mengadopsi standar panduan pembuatan dokumen *SRS* oleh *IEEE* [18]. Selain itu penulis juga merujuk pada langkah perancangan yang dilakukan pada penelitian tugas akhir Muhammad Khotib [7] untuk perancangan aplikasi web dan penelitian tugas akhir Ferdian Dwi Cahyo [19] untuk perancangan *dashboard*, kemudian disesuaikan dengan metode *RAD* yang digunakan oleh penulis.

4.3.1. Analisis kebutuhan

Dari hasil wawancara dan observasi seperti yang dijelaskan pada poin 4.1, penulis dapat mengumpulkan dan menganalisa beberapa fungsi yang dibutuhkan aplikasi.

a. Fungsi utama

Dari hasil wawancara dan observasi yang telah dilakukan, penulis menyimpulkan beberapa fungsi utama yang dapat dilakukan aplikasi. Fungsi utama aplikasi ditunjukkan pada Tabel 4.5.

Tabel 4.5 Fungsi Utama Aplikasi

No.	Fungsi Aplikasi
1	Menyajikan detail dan perbandingan data SIAKAD dan PDDikti sehingga memungkinkan pengguna departemen untuk melihat perbandingan data secara langsung.
2	Memungkinkan departemen dan DPTSI selaku pengguna untuk melakukan sinkronisasi serta pengiriman data SIAKAD ke *Feeder* PDDikti.

b. Kebutuhan fungsional

Berdasarkan poin 4.3.1.1 mengenai fungsi utama aplikasi, maka dapat dibuat kebutuhan fungsional yang dapat mendukung fungsi utama aplikasi. Kebutuhan fungsional dari aplikasi ini adalah:

1. Aplikasi dapat melakukan *login* dan *logout.*
2. Aplikasi ini dapat menampilkan data dari SIAKAD dan *Feeder* PDDikti untuk perbandingan.
3. Aplikasi dapat melakukan sinkronisasi melalui sistem antrian data dimana sistem antrian akan mengambil data dari sumber data SIAKAD dan *Feeder* PDDikti berdasarkan permintaan pengguna.
4. Aplikasi dapat melakukan pengiriman data kepada PDDikti melalui *web service API*

Feeder berdasarkan permintaan pengguna menggunakan sistem antrian.

5. Aplikasi dapat menampilkan *dashboard*.
6. Administrator dapat membuat dan menghapus akun pengguna.
7. Aplikasi ini dapat melakukan pengaturan hak akses data untuk akun pengguna.
8. Aplikasi dapat menampilkan status proses antrian.
9. Aplikasi dapat menampilkan status *error* pengiriman data.
10. Aplikasi dapat menampilkan monitor sistem antrian untuk administrator.
11. Administrator dapat menghentikan proses sistem antrian dan dapat menjalankan kembali.

c. Kebutuhan non fungsional

Kebutuhan non fungsional merupakan kebutuhan aplikasi agar dapat dijalankan dengan lancar serta mendukung kebutuhan fungsional. Berikut merupakan kebutuhan non fungsional dari aplikasi ini.

1. Aplikasi dapat dijalankan pada berbagai jenis sistem operasi komputer.
2. Aplikasi dapat dijalankan pada berbagai jenis spesifikasi komputer.
3. Aplikasi tidak memberikan beban memori besar pada komputer pengguna.
4. Aplikasi dapat dijalankan pada segala jenis *browser* yang mendukung *JavaScript* dan *HTML5*.
5. Aplikasi dan sistem antrian dapat berjalan terus menerus selama 24 jam.

4.3.2. Perencanaan web dan sistem antrian

Aplikasi yang akan dibuat merupakan aplikasi berbasis web yang dibangun menggunakan bahasa pemrograman *PHP* serta menggunakan *framework CodeIgniter 4*. Pemilihan bahasa pemrograman *PHP* dan *framework CodeIgniter 4* dikarenakan *framework* tersebut

merupakan salah satu *framework* yang sangat umum digunakan dalam pengembangan web serta akrab digunakan dalam lingkungan pengembangan proyek DPTSI. Sesuai dengan yang dijelaskan pada poin sebelumnya bahwa aplikasi yang akan dikembangkan memiliki fungsi utama yaitu sinkronisasi data, komparasi data serta pengiriman data. Web akan mengirim *event* pada sistem antrian sebagai *trigger* dalam pengambilan serta pengiriman data.

Pada bagian ini akan dijelaskan bagaimana penulis membuat rencana aplikasi yang dimulai dengan membuat *Use Case Diagram, Use Case Scenario, Sequence Diagram, Entity Relationship Diagram*, perencanaan sistem antrian, dan membuat instrumen pengujian aplikasi untuk pengguna.

a. *Use case diagram* dan *scenario*

   Use case diagram digunakan untuk mendefinisikan aktor yang terlibat dalam penggunaan aplkasi. *Use case diagram* dari aplikasi ini ditampilkan pada Gambar 4.1. Sedangkan beberapa contoh skenario dari aplikasi ditunjukkan pada Tabel 4.6, Tabel 4.7, dan Tabel 4.9. Untuk skenario selengkapnya ditunjukkan pada **LAMPIRAN E DOKUMEN *SRS FEEDER BRIDGE***.

Gambar 4.1 *Use Case Diagram*

Tabel 4.6 *Use Case Scenario* Melihat Data

Nama Use Case	[UC02] Melihat Data	
Aktor	Admin DPTSI dan pengguna departemen	
Deskripsi	Pengguna melihat data yang ada di dalam ITS *Feeder Bridge* sesuai dengan departemen masing-masing	
Tujuan	Untuk melihat data yang ada di ITS *Feeder Bridge*	
Skenario Utama	Pengguna	Sistem
	1. Memilih menu data sesuai kebutuhan (mahasiswa, dosen, kelas, dll)	2. Menampilkan halaman dengan tabel kosong

	3. Memilih prodi dan periode semester sebagai *filter* data	4. Menampilkan data yang diminta ke dalam tabel sesuai prodi dan periode semester
Skenario Alternatif	4.1. Jika data yang diminta tidak ada di *database*, maka yang ditampilkan adalah data kosong	

Tabel 4.7 *Use Case Scenario* Sinkronisasi Data

Nama Use Case	[UC03] Sinkronisasi Data	
Aktor	Admin DPTSI dan pengguna departemen	
Deskripsi	Pengguna melakukan sinkronisasi data yang ada di dalam ITS *Feeder Bridge* sesuai dengan departemen masing-masing	
Tujuan	Untuk memperbarui data yang ada di ITS *Feeder Bridge* sesuai dengan data dari Siakad dan *Feeder* PDDikti	
Skenario Utama	Pengguna	Sistem
	1. Memilih prodi dan periode semester sebagai *filter* data yang akan disinkronisasi	
	2. Klik tombol "Sinkronisasi"	3. Menampilkan *popup* bahwa permintaan berhasil dikirim

		4. Mengirimkan permintaan sinkronisasi ke antrian dan menampilkan notifikasi sinkronisasi serta status permintaan pada menu proses data antrian
		5. Memproses permintaan sinkronisasi hingga selesai
Skenario Alternatif	3.1. Jika pengguna belum memilih *filter* data yang ingin disinkronisasi maka akan menampilkan popup bahwa permintaan ditolak/gagal	
	5.1. Jika permintaan gagal diproses pada sistem antrian maka sistem akan mengubah status menjadi gagal serta menampilkannya pada menu proses data antrian	

Tabel 4.8 *Use Case Scenario* Kirim Data

Nama Use Case	[UC05] Kirim Data
Aktor	Admin DPTSI dan pengguna departemen
Deskripsi	Pengguna melakukan pengiriman data dari ITS *Feeder Bridge* ke *Feeder* PDDikti sesuai dengan departemen masing-masing

Tujuan	Untuk mengirimkan data terbaru yang ada di ITS *Feeder Bridge* ke *Feeder* PDDikti	
Skenario Utama	Pengguna	Sistem
	1. Memilih prodi dan periode semester sebagai *filter* data yang akan dikirim	
	2. Klik tombol "Kirim"	3. Menampilkan popup bahwa permintaan berhasil dikirim
		4. Mengirimkan permintaan kirim ke antrian dan menampilkan notifikasi kirim serta status permintaan pada menu proses data antrian
		5. Memproses permintaan kirim data hingga selesai
Skenario Alternatif	3.1. Jika pengguna belum memilih *filter* data yang ingin disinkronisasi	

	maka akan menampilkan popup bahwa permintaan ditolak/gagal
	5.1. Jika permintaan gagal diproses pada sistem antrian maka sistem akan mengubah status menjadi gagal serta menampilkannya pada menu proses data antrian

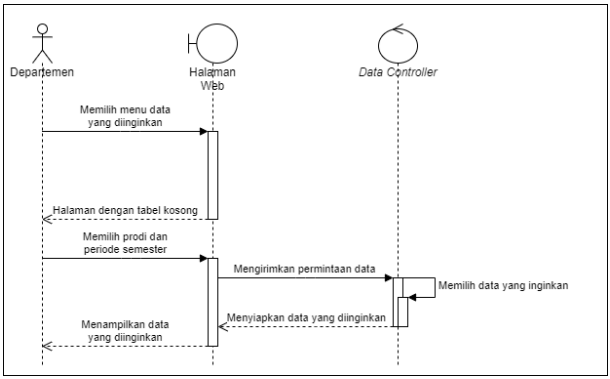
Tabel 4.9 *Use Case Scenario* Melihat *Dashboard*

Nama Use Case	[UC06] Melihat *dashboard*	
Aktor	Admin DPTSI dan pengguna *dashboard*/Dirpendik	
Deskripsi	Pengguna membuka *dashboard* pelaporan yang menunjukkan keadaan terkini pelaporan data ke PDDikti	
Tujuan	Untuk membuka *dashboard* dan menunjukkan keadaan terkini pelaporan data dari ITS *Feeder Bridge* ke *Feeder* PDDikti	
Skenario Utama	Pengguna	Sistem
	1. *Login* sebagai Dirpendik/pengguna *dashboard*	2. Validasi *login* credentials
		3. Menampilkan menu utama
	4. Memilih menu *dashboard*	5. Menghubungkan *Feeder Bridge* dengan sistem *dashboard*

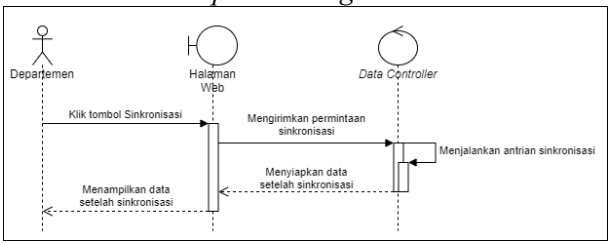
		6. Menampilkan *dashboard*
	7. Memilih halaman *dashboard* yang diinginkan	8. Menampilkan halaman sesuai pilihan pengguna
Skenario Alternatif	2.1. Jika *login* credentials tidak sesuai maka pengguna tidak dapat masuk ke menu utama	
	4.1. Jika tidak *login* sebagai dirpendik, maka tidak dapat memilih menu *dashboard*	
	5.1. Jika koneksi *Feeder Bridge* dengan sistem *dashboard* tidak dapat dilakukan, maka *dashboard* tidak dapat ditampilkan	

b. *Sequence diagram*

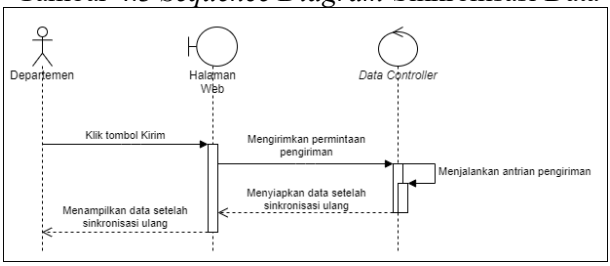
Sequence Diagram digunakan untuk menunjukkan alur setiap *use case* dari aplikasi. Beberapa *Sequence Diagram* aplikasi ditunjukkan pada Gambar 4.2, Gambar 4.3, Gambar 4.4, dan Gambar 4.5. Untuk *Sequence Diagram* selengkapnya ditunjukkan pada **LAMPIRAN E DOKUMEN *SRS FEEDER BRIDGE***.



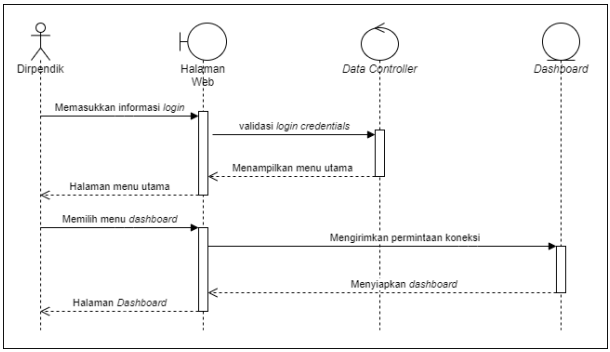
Gambar 4.2 *Sequence Diagram* Melihat Data



Gambar 4.3 *Sequence Diagram* Sinkronisasi Data



Gambar 4.4 *Sequence Diagram* Kirim Data



Gambar 4.5 *Sequence Diagram* Melihat *Dashboard*

c. Model data

Model penyimpanan data yang dibuat menerapkan campuran antara *relational database* dengan *document-based database*. Tabel data masih menggunakan kolom tetapi juga menggunakan *key* dari *document database*. Penggunaan bentuk *document-based database* dalam bentuk *JSON* bertujuan untuk menyederhanakan tabel serta meminimalkan relasi yang ada pada tabel dikarenakan banyaknya jenis data yang digunakan. Bentuk *document-based* atau *JSON* juga memiliki tingkat fleksibilitas yang lebih tinggi ketika terjadi perubahan data. Kolom yang ada pada entitas data digunakan untuk memudahkan filter data, contohnya entitas mahasiswa masih akan memiliki kolom periode masuk dan kode prodi.

d. *Entity Relationship Diagram*

Setelah proses *brainstorming* dan penggalian informasi, ditemukan *entity relationship diagram* seperti yang dicantumkan pada dokumen *SRS* yaitu *ERD* versi pertama yang ditunjukkan pada **LAMPIRAN E DOKUMEN *SRS FEEDER BRIDGE***. Adapun entitas-entitas data dari *entity relationship diagram* versi pertama ditunjukkan pada Tabel 4.10.

Tabel 4.10 Entitas *ERD* Versi Pertama

Nama Entitas	**Keterangan**
Perguruan Tinggi	Entitas perguruan tinggi berisi data terkait perguruan tinggi tersebut seperti jumlah program studi, jumlah mahasiswa, jumlah dosen, alamat, dll.
Prodi (Program Studi)	Entitas prodi berisi data terkait program studi di perguruan tinggi seperti jenjang prodi, kode prodi, dll.
Kurikulum	Entitas kurikulum berisi data kurikulum yang sedang berlaku maupun kurikulum lama yang digunakan oleh prodi.
Mata kuliah	Entitas mata kuliah berisi mata kuliah dalam suatu prodi dengan kurikulum yang digunakan oleh prodi tersebut.
Kelas	Entitias kelas berisi data perkuliahan yang dilakukan dalam satu periode semester. Data kelas berisi data pengajar kelas, anggota kelas, mata kuliah, nama kelas, dll.
KRS	Entitas KRS berisi data aktivitas KRS mahasiswa baik rencana KRS maupun KRS semester sebelumnya.

Mahasiswa	Entitas mahasiswa berisi data terkait mahasiswa seperti biodata mahasiswa, aktivitas kuliah, dosen wali, dll.
Biodata mahasiswa	Entitas biodata mahasiswa berisi data biodata mahasiswa termasuk alamat, data orang tua/wali, tempat/tanggal lahir, jenis kelamin, dll.
Dosen	Entitas dosen berisi data terkait dosen seperti biodata dosoen, aktivitas dosen, dll.
Aktivitas Mengajar Dosen	Entitas aktivitas mengajar dosen berisi data kelas yang diajarkan dosen dalam satu periode/semester.
Nilai	Entitas nilai berisi data kelas dan data mahasiswa sebagai anggota kelas.
Aktivitas	Entitas mahasiswa berisi aktivitas mahasiswa diluar aktivitas kelas perkuliahan. Contohnya seperti aktivitas KKN, seminar proposal Tugas Akhir, sidang Tugas Akhir, bimbingan Tugas Akhir, dll.
Anggota aktivitas	Entitas anggota aktivitas berisi anggota mahasiswa yang menjalankan aktivitas.

Jenis aktivitas	Entitas jenis aktivitas berisi tipe – tipe aktivitas yang dapat dilakukan oleh mahasiswa diluar aktivitas perkuliahan.

Kemudian dilakukan penyederhanaan dan penerapan *document-based database* atau bentuk *JSON* kepada tabel data. Sehingga didapatkan *ERD* versi kedua seperti dicantumkan pada dokumen *SRS ERD* versi kedua pada yang ditunjukkan pada **LAMPIRAN E DOKUMEN *SRS FEEDER BRIDGE***. Adapun entitas-entitas data dari *entity relationship diagram* versi kedua ditunjukkan pada Tabel 4.11.

Tabel 4.11 Entitas *ERD* Versi Kedua

Nama Entitas	**Keterangan**
Mahasiswa	Entitas mahasiswa menjadi gabungan seluruh entitas yang terkait dengan data mahasiswa seperti KRS, nilai, aktivitas kuliah, dan biodata.
Dosen	Entitas dosen menjadi gabungan entitas yang terkait dengan data dosen seperti biodata dosen, aktivitas mengajar dosen, dan aktivitas kuliah.
Kelas	Entitas kelas menjadi gabungan entitas yang terkait dengan kelas seperti dosen pengajar, dan anggota kelas
Prodi	Entitas prodi menjadi hanya berisi program studi yang ada di ITS sehingga tidak menggunakan entitas perguruan tinggi.
Mata kuliah	Entitas mata kuliah menjadi satu dengan entitas kurikulum dengan

	menjadikan kurikulum sebagai salah satu kolom data.
Aktivitas	Entitas aktivitas mejadi gabungan antara entitas yang terkait dengan aktivitas mahasiswa seperti jenis aktivitas, anggota aktivitas, dll.

Penggunaan bentuk *document-based database* atau *JSON* bertujuan untuk menyederhanakan tabel serta meminimalkan relasi yang ada pada tabel dikarenakan banyaknya jenis data yang digunakan. Bentuk *document-based* atau *JSON* juga memiliki tingkat fleksibilitas yang lebih tinggi ketika terjadi perubahan data.

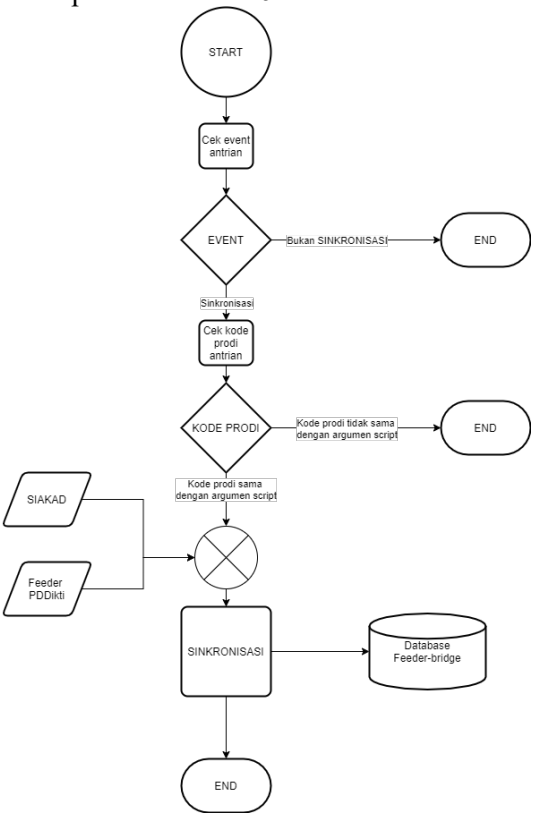
e. Perencanaan pembangunan sistem antrian

Sistem antrian pada aplikasi akan dibangun dengan bahasa pemrograman *Python*. Bahasa pemrograman tersebut dipilih karena kemudahan dalam menuliskan baris kode sehingga pembuatan aplikasi akan berjalan lebih cepat. *Library* yang didukung untuk *Python* juga memadai untuk penggunaan sistem antrian. Sistem antrian memiliki dua fungsi utama untuk mendukung aplikasi. Fungsi pertama adalah pengumpulan data (*data gathering)*, yaitu proses mengambil data sekaligus men-sinkronisasi data yang ada pada SIAKAD (ITS) dan *feeder* (PDDikti). Data yang ada di siakad dan *feeder* kemudian akan dipetakan sesuai dengan *ERD* yang dibuat sehingga dapat digunakan oleh aplikasi. Fungsi sistem antrian yang kedua adalah pengiriman data termasuk *insert, update,* dan *delete* data *Feeder* PDDikti. Pengiriman ini dilakukan berdasarkan data yang sebelumnya sudah dikumpulkan dan di-sinkronisasi oleh sistem antrian. Fungsi ini kemudian disebut dengan '*event'* yaitu *event* sinkronisasi dan *event* kirim. Setiap *event* membutuhkan detail data apa saja yang akan di-

sinkronisasi atau dikirimkan. Sehingga, setiap *event* dibagi menjadi beberapa modul.

1. Sinkronisasi

   Aktivitas pengambilan data dari sumber data SIAKAD dan PDDikti yang kemudian disimpan dalam *database Feeder Bridge*. Aktivitas ini melakukan *query* pada SIAKAD dan menggunakan *method GET* pada PDDikti. *Flowchart* lengkap proses sinkronisasi dapat dilihat pada Gambar 4.6 berikut.



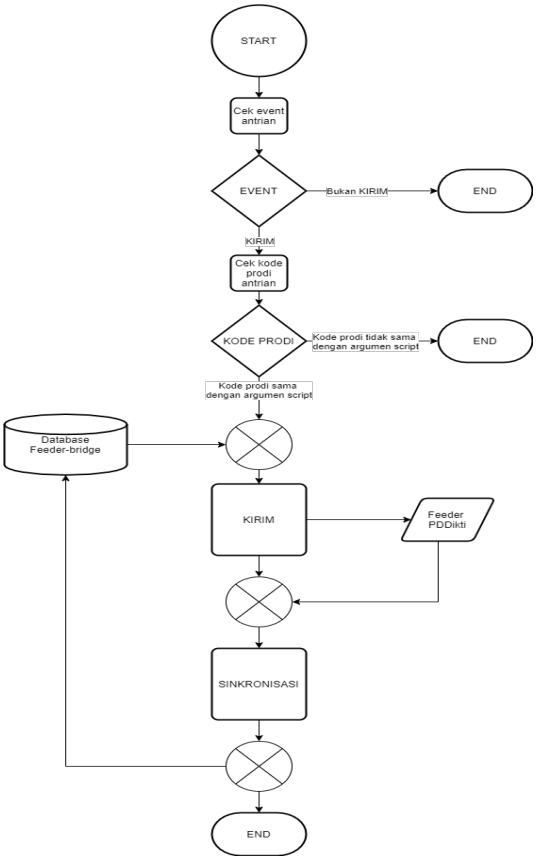
Gambar 4.6 *Flowchart* Proses Sinkronisasi

Event sinkronisasi dibagi berdasarkan data apa yang akan di-sinkronisasi. Pembagian tersebut antara lain:

a. sinkronisasi_datamahasiswa
   Data mahasiswa terdiri dari biodata mahasiswa, aktivitas perkuliahan mahasiswa, kelas mahasiswa, dan aktivitas mahasiswa.
b. sinkronisasi_datakelas
   Data kelas terdiri dari data kurikulum, periode kelas, nama kelas, dosen pengampu, dan anggota kelas.
c. sinkronisasi_datadosen
   Data dosen terdiri dari data biodata dosen.
d. sinkronisasi_dataprodi
   Data prodi terdiri dari data program studi di ITS.
e. sinkronisasi_datamatkul
   Data mata kuliah terdiri dari data mata kuliah, kurikulum, dan sks.
f. sinkronisasi_aktivitas
   Data aktivitas terdiri dari aktivitas perkuliahan seperti KKN, magang, Kampus Merdeka, dan sidang/Tugas Akhir.

Pembagian tersebut juga didasarkan atas penentuan cakupan sistem antrian saat ini yaitu data terkait perkuliahan, mahasiswa, dan dosen.

2. Kirim
   Aktivitas mengirimkan data dari *database Feeder Bridge* ke PDDikti sesuai data yang telah di-sinkronisasi. Aktivitas ini menggunakan *method update, insert,* dan *delete* yang ada pada *API* PDDikti. *Flowchart* lengkap proses kirim dapat dilihat pada Gambar 4.7 berikut.



Gambar 4.7 *Flowchart* Proses Kirim Data

Event kirim juga dibagi berdasarkan data apa yang dikirimkan kepada PDDikti. Pembagian tersebut antara lain:

a. kirim_datakelas
   Data kelas yang dikirim adalah data kelas, dosen pengampu, dan anggota kelas.
b. kirim_datamahasiswa
   Data mahasiswa yang dikirim adalah biodata mahasiswa.
c. kirim_aktivitaskuliahmahasiswa

Data yang dikirim adalah data perkuliahan mahasiswa seperti kelas yang diambil, jumlah sks, ips, dan ipk.

d. kirim_aktivitasmahasiswa
   Data yang dikirim adalah data terkait kegiatan mahasiswa seperti KKN, magang, Kampus Merdeka, Tugas Akhir, dll.
e. kirim_lulusmahasiswa
   Data yang dikirm adalah data ketika ada mahasiswa yang telah melaksanakan sidang tugas akhir dan dinyatakan lulus.

Pembagian tersebut juga didasarkan atas penentuan cakupan sistem antrian yaitu data terkait perkuliahan, mahasiswa, dan dosen.

Selain *event*, sistem antrian juga membagi pekerjaan menjadi setiap program studi yang ada di ITS. Hal ini dilakukan atas beberapa alasan seperti:

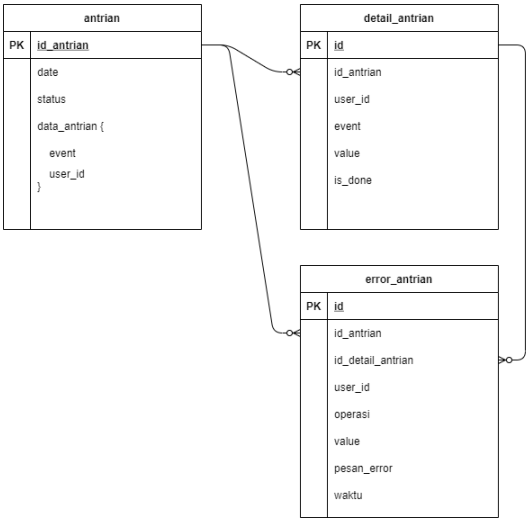
a. Mempercepat pekerjaan dengan tidak menumpuk antrian menjadi keseluruhan ITS.
b. Menghindari antrian yang diproses 2 kali oleh sistem/*script* antrian yang berbeda.

Maka dari itu untuk bisa membagi tugas sistem antrian menjadi setiap program studi di ITS, maka diperlukan agumen yang berisi kode program studi yang akan dikerjakan oleh sistem antrian tersebut. Apabila kode program studi yang ada di antrian sama dengan yang dijalankan suatu *script* antrian maka proses akan berjalan, sebaliknya jika tidak sama maka antrian tersebut tidak akan diproses oleh antrian tersebut melainkan oleh *script* antrian lain yang memiliki argument yang sesuai. Setelah penerapan pembagian sistem antrian menjadi setiap program studi di ITS, untuk dapat lebih melakukan pekerjaan secara parallel sehingga

lebih cepat, penulis membuat sistem lain yaitu detail antrian. Detail antrian digunakan untuk membagi pekerjaan – pekerjaan terutama proses sinkronisasi menjadi beberapa tugas. Contoh dalam penerapan antara lain dalam sinkronisasi mahasiswa. Dalam sinkronisasi mahasiswa ada banyak data yang harus diambil dari SIAKAD dan PDDikti yang terkait dengan mahasiswa tersebut. Contoh data yang dimaksud dapat dikategorikan menjadi 3, yaitu:

a. Biodata mahasiswa
b. Aktivitas kuliah mahasiswa
c. Kelas mahasiswa

Maka penerapan detail antrian dapat digunakan pada sinkronisasi kelas mahasiswa, sehingga sistem antrian dapat melakukan sinkronisasi biodata mahasiswa dan aktivitas kuliah mahasiswa bersamaan dengan sinkronisasi data kelas mahasiswa.Setelah mengetahui kebutuhan sistem antrian, dirumuskan *ERD* yang sesuai. *ERD* sistem antrian dapat dilihat pada Gambar 4.8.



Gambar 4.8 *Entity Relationship Diagram* Sistem Antrian

Tabel antrian digunakan untuk mendapatkan antrian yang berasal dari aplikasi web. Sistem antrian kemudian akan memeriksa *event* dan kode program studi dalam antrian. Tergantung dari proses yang akan dijalankan, sistem antrian terkadang akan menggunakan detail antrian. Tabel *error* antrian berisi *error* yang terjadi ketika sistem antrian melakukan proses antrian. Tabel tersebut dibuat untuk dapat memonitor pekerjaan dari sistem antrian. Setelah perencanaan *ERD* dan cara kerja sistem antrian, perencanaan yang juga diperlukan adalah perencaan dalam *deploy* sistem. Sistem antrian membutuhkan argument kode program studi sehingga sistem antrian dapat mengerjakan proses secara paralel, terpisah, dan bersamaan. Antrian yang berasal dari satu program studi tidak akan dikerjakan oleh sistem antrian program studi yang lain. Contohnya seorang pengguna dari program studi S1 Sistem Informasi melakukan sinkronisasi data mahasiswa, maka *event* tersebut tidak akan

dikerjakan oleh sistem antrian dengan argument kode program studi milik program studi lain begitu pula sebaliknya. Maka dari itu sistem antrian perlu mencegah *script* antrian berjalan tanpa argument.

f. Perencanaan implementasi sistem antrian

Perencanaan implementasi mencakup bagaimana sistem antrian akan berjalan dari sisi *server* dan bagaimana sistem antrian mengatasi antrian yang masuk.

Pada bagian ini perlu diperjelas beberapa istilah sehingga tidak menimbulkan pemahaman berbeda. Istilah – istilah tersebut seperti:

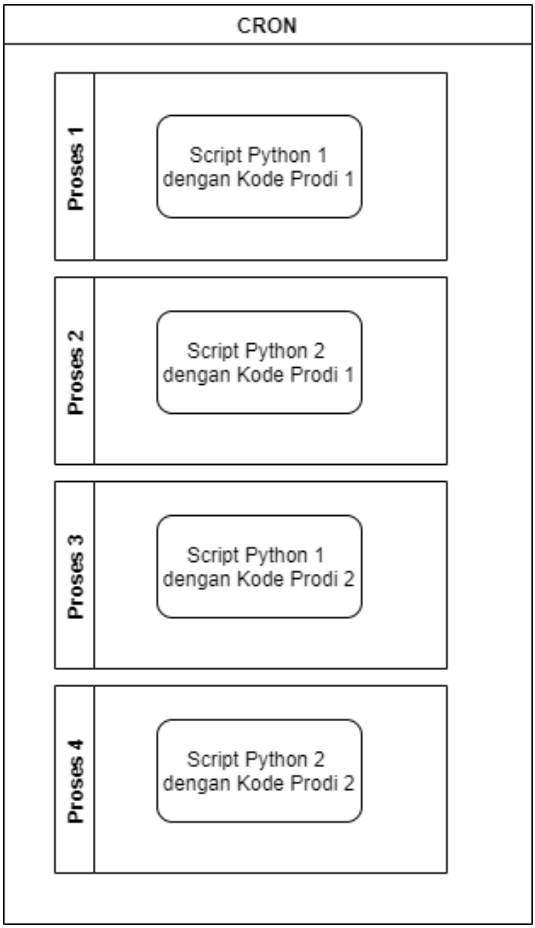
- Fungsi antrian: Kumpulan perintah untuk melakukan proses antrian seperti sinkronisasi dan kirim data.
- *Event* antrian: Perintah yang dikirim oleh pengguna untuk dapat melakukan fungsi antrian dengan data tertentu seperti data mahasiswa atau data kelas.
- *Script Python*: Sebuah program *Python* yang berisi perintah – perintah dengan fungsi tertentu, contohnya sinkronisasi_datamahasiswa.py
- Proses: Kumpulan satu atau lebih *script.*
- *Task:* Sebuah proses yang akan dijalankan oleh CRON.

Maka perencanaan implementasi sistem antrian adalah sebagai berikut.

1. Membuat sistem antrian berjalan di *server*

   Untuk dapat bekerja dengan baik sistem antrian diharapkan dapat terus berjalan 24 jam setiap hari, sehingga kapanpun pengguna menggunakan aplikasi sistem antrian dapat memproses antrian. Hal tersebut dapat dicapai dengan membuat *script Python* sistem antrian yang berjalan terus menerus dengan kondisi spesifik untuk berhenti. Kondisi spesifik untuk berhenti tersebut harus berasal dari suatu input admin dan tidak berhenti meskipun terdapat

error, karena setiap *error* yang muncul harus dapat ditampilkan melalui tabel *error*_antrian. Untuk mencapai hal tersebut, sistem antrian akan dijalankan menggunakan pekerjaan CRON pada *server*. CRON adalah penjadwalan sistem berbasis UNIX dan LINUX. Tujuannya adalah menjalankan perintah, kumpulan perintah, atau *script* pada waktu yang ditentukan [20]. Perintah – perintah CRON tersebut akan penulis sebut sebagai sebuah *task* dan program yang menjalankan kumpulan *script* untuk tugas tertentu akan disebut sebagai proses. Teknik pengimplementasian sistem antrian yang akan dilakukan adalah membagi setiap *script* berdasarkan fungsinya dan berdasarkan kode prodi yang akan dikerjakan oleh *script* tersebut. Sehingga setiap *script* fungsi antrian dan prodi kemudian dijalankan sebagai satu proses. Sehingga jika digambarkan akan menjadi seperti pada Gambar 4.9 berikut.



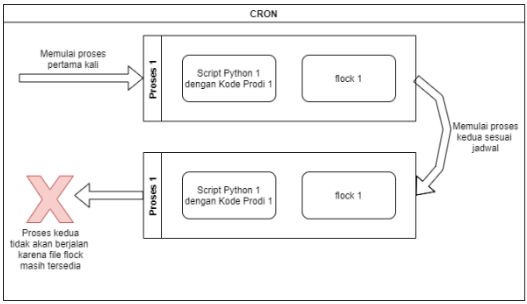
Gambar 4.9 Proses CRON

Setiap proses berisi *script Python* dan kode program studi yang berbeda sehingga proses antara satu dan lain tidak saling berpengaruh. Terutama proses diantara program studi satu dengan lain tidak bisa saling menghalangi dan mempengaruhi. Proses – proses ini kemudian dijalankan sebagai *task* pada CRON. Berdasarkan pembagian ini dapat diperkirakan berapa jumlah *task* yang harus dijalankan oleh

CRON. Berdasarkan sub-bab sebelumnya terdapat 6 tipe sinkronisasi data dan 5 tipe kirim data. Maka total *script* adalah 11 *script* dengan 1 *script* tidak membutuhkan kode prodi yang spesifik yaitu sinkronisasi_dataprodi. Sementara untuk jumlah program studi di ITS diperkirakan sekitar 200 program studi (S1, S2, S3, dan Vokasi). Maka perkiraan proses yang dibutuhkan untuk mencakup keseluruhan ITS adalah 2000 proses (10 *script* dan 200 prodi). Untuk memudahkan pemasangan proses – proses tersebut, dapat dibuat *text file* yang berisi setiap proses. CRON dapat membaca *text file* yang dibuat dan menambahkan semua proses dalam *text file* tersebut.

2. Cara sistem antrian mengatasi antrian yang masuk

   Salah satu hal yang perlu diperhatikan dalam implementasi sistem antrian adalah tidak boleh ada 2 *script* yang sama berjalan bersamaan. Hal tersebut dapat menyebabkan *error* dan kesalahan keakuratan data. Dan salah satu kekurangan yang ditemukan dalam proses CRON adalah CRON tidak memperdulikan apakah proses yang sama sedang berjalan atau tidak. Namun, hal tersebut dapat dicegah dengan sebuah *utility* yang tersedia dalam sistem operasi LINUX bernama *flock*. *Flock* membuat sebuah file dengan eksetensi .lock yang jika tersedia, maka CRON tidak akan menjalankan *script* tersebut kembali dan mencegah 2 *script* yang sama berjalan bersamaan. Dengan demikian maka setiap antrian yang masuk akan diproses oleh satu proses saja. Secara logika penggunaan *flock* dapat dipahami dengan Gambar 4.10 berikut.



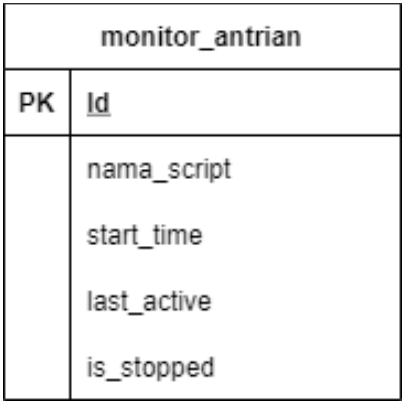
Gambar 4.10 Proses Cara Kerja CRON

Jika dalam proses yang sedang berjalan terdapat file *flock* maka setiap kali CRON berupaya menjalankan proses yang sama sesuai jadwal, proses tersebut akan tertolak dan tidak berjalan sedangkan proses pertama yang sedang berjalan akan tetap berjalan seperti biasa. Tujuan penerapan *flock* adalah untuk menghentikan proses yang sama berjalan bersamaan, maka setiap proses harus menggunakan *lock* unik yang berbeda antara satu dengan lainnya. Penerapan *flock* bukan berarti akan membuat sistem antrian tidak berjalan terus menerus. Untuk dapat mencapai sistem antrian yang terus berjalan, ada 2 cara yang diterapkan penulis untuk memastikan hal tersebut. Cara tersebut antara lain:

1. Membuat setiap *script Python* berjalan terus menerus hingga ada perintah berhenti.
2. CRON menjalankan proses secara rutin, misalnya setiap jam sehingga jika proses berhenti tanpa ada perintah berhenti misalnya karena *error* yang tidak ditangani dengan baik, proses dapat berjalan kembali dan tidak terlalu mempengaruhi sistem antrian.

3. Monitor dan menghentikan sistem antrian

Monitor *script* antrian diperlukan karena sistem antrian adalah yang memproses fungsi utama aplikasi yaitu kirim dan sinkronisasi data. Maka dari itu perlu tabel lain yang berisi informasi tentang *script* yang berjalan seperti nama *script*, waktu aktivasi, waktu terakhir aktif, dan perintah mematikan *script*. Sehingga rencana tabel monitor_antrian dapat dilihat pada Gambar 4.11.



Gambar 4.11 Tabel Monitor Antrian

Tabel tersebut memiliki *primary key* berupa Id, nama_*script* dengan tipe data varchar, start_time dan last_active dengan tipe data *datetime*, serta is_stopped dengan tipe data *Boolean* 1 dan 0. Berdasarkan tabel tersebut admin dapat memantau, menghentikan, dan memulai proses antrian.

g. Instrumen pengujian aplikasi web dan sistem antrian

Instrumen pengujian aplikasi yang digunakan adalah *usability testing* dengan menyiapkan skenario pengujian aplikasi dan rencana pengujian . Pengujian juga dilakukan dengan melakukan pengamatan dari sisi performa melalui beban *CPU*, memori (*RAM*) baik dari sisi *client* maupun sisi *server* serta beban internet yang digunakan oleh aplikasi aplikasi. Partisipan dari *usability testing*

serta pengujian performa aplikasi merupakan mahasiswa ITS yang merupakan tenaga magang DPTSI dan telah berpengalaman mengoperasikan aplikasi *ProFeeder* selama 1,5 tahun untuk partisipan 1 dan 1 tahun untuk partisipan 2. Sebelum pengujian partisipan diminta untuk mempersiapkan laptop masing-masing serta jaringan internet yang memadai karena pengujian dilakukan secara *online* serta partisipan akan diminta untuk menjalankan web *Feeder Bridge* serta aplikasi *ProFeeder*. Saat pengujian partisipan akan diberikan akses web serta akun yang sudah dibuat. Pada saat *usablity testing* pengguna akan diminta untuk melakukan skenario aktivitas yang berhubungan dengan fitur aplikasi yang sebelumnya telah dibuat oleh penulis yang berperan sebagai penguji dan pengamat. Setelah itu pengguna akan diberikan *open ended question* seputar aplikasi untuk menggali lebih dalam pendapat serta pengalaman partisipan setelah mencoba aplikasi. Selanjutnya dilakukan pengujian performa dari aplikasi. Partisipan akan diminta untuk membuka *Task Manager* sebagai alat untuk memantau beban memori, *CPU*, serta internet yang diperlukan oleh aplikasi. Setelah itu partisipan juga diminta untuk menjalankan aplikasi web *Feeder Bridge* serta aplikasi *ProFeeder* untuk membandingkan performa dari kedua aplikasi.

1. Skenario pengujian pengguna
   Berikut merupakan skenario pengujian pengguna untuk aplikasi yang ditunjukkan pada Tabel 4.12, Tabel 4.13, Tabel 4.14, Tabel 4.15, Tabel 4.16, Tabel 4.17, Tabel 4.18, dan Tabel 4.19.

Tabel 4.12 Skenario Pengujian Data Program Studi

ID Skenario	Skenario	ID Aktivitas	Aktivitas

SPRD01	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan sinkronisasi data prodi	TPRD01	Pengguna memilih data prodi pada menu
		TPRD02	Pengguna memilih *filter* prodi data prodi yang ingin disinkronisasi
		TPRD03	Pengguna mengklik tombol sinkronisasi
SPRD02	Anda sebagaipengguna departemen/admin DPTSI, Anda ingin melihat data prodi	TPRD04	Pengguna mengklik tombol 'Tampilkan'
SPRD03	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melihat detail perbandingan data prodi	TPRD05	Pengguna mengklik tombol 'Detail' pada baris data prodi yang ingin dilihat detailnya
SPRD04	Anda sebagaipengguna departemen/admin DPTSI, Anda ingin melakukan kirim data prodi	TPRD06	Pengguna mengklik tombol 'Kirim'

Tabel 4.13 Skenario Pengujian Data Mahasiswa

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SMHS01	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan sinkronisasi data mahasiswa	TMHS01	Pengguna memilih data mahasiswa pada menu
		TMHS02	Pengguna memilih *filter* prodi dan periode data mahasiswa yang ingin disinkronisasi
		TMHS03	Pengguna mengklik tombol sinkronisasi
SMHS02	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melihat data mahasiswa	TMHS04	Pengguna mengklik tombol 'Tampilkan'
SMHS03	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melihat detail perbandingan data mahasiswa	TMHS05	Pengguna mengklik tombol 'Detail' pada baris data mahasiswa yang ingin dilihat detailnya

SMHS04	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan kirim data mahasiswa	TMHS06	Pengguna mengklik tombol 'Kirim'

Tabel 4.14 Skenario Pengujian Data Dosen

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SDSN01	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan sinkronisasi data dosen	TDSN01	Pengguna memilih data mahasiswa pada menu
		TDSN02	Pengguna mengklik tombol sinkronisasi
SDSN02	Anda sebagai opengguna departemen/admin DPTSI, Anda ingin melihat data dosen	TDSN03	Pengguna memilih data mahasiswa pada menu
SDSN03	Anda sebagai opengguna departemen/admin DPTSI, Anda ingin melihat detail perbandingan data dosen	TDSN04	Pengguna mengklik tombol 'Detail' pada baris data dosen yang ingin dilihat detailnya

SDSN05	Anda sebagai operator TU departemen, Anda ingin melakukan kirim data dosen	TDSN05	Pengguna mengklik tombol 'Kirim'

Tabel 4.15 Skenario Pengujian Data Matakuliah

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SMKL01	Anda sebagaipengguna departemen/admin DPTSI, Anda ingin melakukan sinkronisasi data matakuliah	TMKL01	Pengguna memilih data matakuliah pada menu
		TMKL02	Pengguna memilih *filter* prodi dan kurikulum data matakuliah yang ingin disinkronisasi
		TMKL03	Pengguna mengklik tombol sinkronisasi
SMKL02	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melihat data matakuliah	TMKL04	Pengguna mengklik tombol 'Tampilkan'
SMKL03	Anda sebagai pengguna departemen/admin DPTSI, Anda	TMKL05	Pengguna mengklik tombol 'Detail' pada baris data

	ingin melihat detail perbandingan data matakuliah		matakuliah yang ingin dilihat detailnya
SMKL04	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan kirim data matakuliah	TMKL06	Pengguna mengklik tombol 'Kirim'

Tabel 4.16 Skenario Pengujian Data Kelas

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SKLS01	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan sinkronisasi data kelas	TKLS01	Pengguna memilih data kelas pada menu
		TKLS02	Pengguna memilih *filter* prodi dan periode data kelas yang ingin disinkronisasi
		TKLS03	Pengguna mengklik tombol sinkronisasi
SKLS02	Anda sebagai pengguna departemen/admin DPTSI, Anda	TKLS04	Pengguna mengklik tombol 'Tampilkan'

	ingin melihat data kelas		
SKLS03	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melihat detail perbandingan data mahasiswa	TKLS05	Pengguna mengklik tombol 'Detail' pada baris data kelas yang ingin dilihat detailnya
SKLS04	Anda sebagaipengguna departemen/admin DPTSI, Anda ingin melakukan kirim data mahasiswa	TKLS06	Pengguna mengklik tombol 'Kirim'
		TKLS07	Pengguna mengisi tanggal awal dan akhir pada pop up
		TKLS08	Pengguna mengklik tombol 'Kirim' pada pop up

Tabel 4.17 Skenario Pengujian Data Aktivitas

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SAKT01	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan sinkronisasi data aktivitas	TAKT01	Pengguna memilih data aktivitas pada menu

		TAKT02	Pengguna memilih *filter* prodi dan periode data prodi yang ingin disinkronisasi
		TAKT03	Pengguna mengklik tombol sinkronisasi
SAKT02	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melihat data aktivitas	TAKT04	Pengguna mengklik tombol 'Tampilkan'
SAKT03	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melihat detail perbandingan data prodi	TAKT05	Pengguna mengklik tombol 'Detail' pada baris data aktivitas yang ingin dilihat detailnya
SAKT04	Anda sebagai pengguna departemen/admin DPTSI, Anda ingin melakukan kirim data aktivitas	TPRD06	Pengguna mengklik tombol 'Kirim'

Tabel 4.18 Skenario Pengujian Menu Proses Data Antrian

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SATR01	Anda sebagai pengguna	TATR01	Pengguna memilih menu

	departemen dan admin DPTSI anda ingin melihat status permintaan sinkronisasi atau kirim data pada antrian		proses data antrian
SATR02	Anda sebagai pengguna departemen dan admin DPTSI anda ingin melihat pesan kegagalan dari pengiriman data	TATR01	Pengguna memilih menu eror pengiriman data

Tabel 4.19 Skenario Pengujian Menu Manajemen Pengguna

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SADM01	Anda sebagai admin DPTSI, Anda ingin melihat daftar pengguna	TADM01	Pengguna memilih menu manajemen pengguna
SADM02	Anda sebagai admin DPTSI, Anda ingin mendaftarkan atau membuat akun pengguna baru	TADM02	Pengguna mengklik tombol 'Tambahkan Pengguna'
		TADM03	Pengguna mengisi form *user id, role*,

			dan *password*
		TADM04	Pengguna mengklik daftarkan akun
SADM03	Anda sebagai admin DPTSI, Anda ingin menghapus akun pengguna	TADM05	Pengguna mengklik tombol 'Hapus' pada baris data pengguna yang ingin dihapus
SADM04	Anda sebagai admin DPTSI, Anda ingin melihat akses prodi yang dapat diakses pengguna	TADM06	Pengguna mengklik tombol 'Ubah Akses' pada baris data pengguna yang ingin dilihat aksesnya
SADM05	Anda sebagai admin DPTSI, Anda ingin menambah akses prodi yang dapat diakses pengguna	TADM07	Pengguna mengklik checkbox menjadi centang pada akses prodi yang ingin ditambah
SADM06	Anda sebagai admin DPTSI, Anda ingin	TADM08	Pengguna mengklik

	membatalkan/mengurangi akses prodi yang dapat diakses pengguna		checkbox menjadi uncheck pada akses prodi yang ingin ingin dikurangi

2. *Usability Test Plan*

Berikut merupakan *usability test plan* aplikasi yang ditunjukkan pada Tabel 4.20.

Tabel 4.20 *Usability Test Plan* Aplikasi

Tujuan Pengujian	Untuk mengetahui apakah pengguna dapat menggunakan prototipe dengan skenario tertentu
Partisipan Pengujian	Pihak operator dari departemen untuk uji coba aplikasi
Durasi Pengujian	60 menit dengan rincian sebagai berikut: 1. Pembukaan (5 menit) 2. Uji coba skenario (50 menit) 3. Wawancara (5 menit)
Peran & Tanggung Jawab	Penulis akan betindak sebagai moderator, pengamat, dan pencatat dengan rincian tugas sebagai berikut. - Moderator bertugas untuk menyapa dan menyampaikan instruksi kepada partisipan sekaligus menjadi teknisi - Pengamat bertugas mengamati partisipan saat melakukan uji coba termasuk emosi partisipan melalui raut wajah - Pencatat bertugas untuk mencatat semua kejadian dan

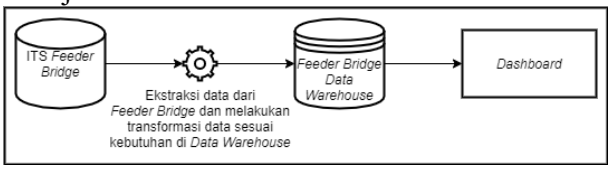
	komentar yang dikeluarkan oleh partisipan

4.3.3. Perencanaan *dashboard*

Bagian ini menjelaskan bagaimana penulis membuat rancangan *dashboard* yang dimulai dengan menentukan arsitektur *dashboard*, mengidentifikasi halaman *dashboard*, mengidentifikasi sumber data, membuat rancangan awal tata letak *dashboard*, membuat desain *data warehouse*, membuat rencana proses *ETL*, dan membuat instrumen pengujian *dashboard* untuk pengguna.

a. Arsitektur *dashboard*

Arsitektur *dashboard* menggambarkan bagaimana *dashboard* memperoleh dan mengolah data. Hal ini bertujuan untuk mempermudah pengulis dalam mengembangkan *dashboard* terutama dalam identifikasi sumber data dan pembuatan desain *data warehouse*. Adapun arsitektur *dashboard* ditunjukkan oleh Gambar 4.12.



Gambar 4.12 Arsitektur *Dashboard*

dashboard mengambil data dari *Feeder Bridge Data Warehouse* setelah dilakukan proses *Extract*, *Transform*, dan *Load* (*ETL*) dari *database* ITS *Feeder Bridge*.

b. Identifikasi halaman *dashboard*

Identifikasi halaman *dashboard* adalah proses yang dilakukan untuk menentukan halaman-halaman *dashboard* yang dapat memenuhi kebutuhan monitoring pelaporan data sesuai dengan standar pengelolaan PDDikti. Tahap ini dilakukan dengan mempelajari standar pengelolaan PDDikti untuk mengetahui indikator yang berkaitan dengan pelaporan data PDDikti. Dalam standar pengelolaan PDDikti disebutkan bahwa indikator pelaporan PDDikti ada tiga, yaitu kelengkapan data, kevalidan data, dan ketaatan pelaporan [21]. Setelah mengetahu indikator, selanjutnya mengidentifikasi halaman *dashboard* yang merepresentasikan indikator tersebut seperti yang ditunjukkan oleh Tabel 4.21.

Tabel 4.21 Identifikasi Halaman

No	Indikator Pengiriman PDDikti	*dashboard*	Referensi
1	Kelengkapan Pelaporan	Pengiriman AKM	Keputusan Sekertaris Jenderal Kemenristekdikti Nomor 85/A/KPT/2018 Tentang Standar Pengelolaan PDDikti
		Pengiriman Kelas	
2	Kevalidan Data	Penyelesaian Komplain	Keputusan Sekertaris Jenderal Kemenristekdikti Nomor 85/A/KPT/2018 Tentang Standar

			Pengelolaan PDDikti
3	Ketaatan Pelaporan	Ketepatan Waktu Pengiriman Data	Keputusan Sekertaris Jenderal Kemenristekdikti Nomor 85/A/KPT/2018 Tentang Standar Pengelolaan PDDikti

c. Identifikasi sumber data

Identifikasi sumber data merupakan proses yang dilakukan untuk menentukan data-data yang digunakan dalam pembuatan *dashboard* sesuai dengan kebutuhan tiap halaman *dashboard*-nya. Setelah halaman *dashboard* telah diidentifikasi, selanjutnya dilakukan identifikasi sumber data untuk masing-masing halaman *dashboard* yang semuanya berasal dari *database* ITS *Feeder Bridge*. Hal ini bertujuan untuk mempermudah penulis dalam membuat desain *data warehouse* dan proses *ETL*-nya. Berikut penjelasan sumber data yang digunakan.

1. Data Mahasiswa

   Data mahasiswa digunakan sebagai sumber data untuk halaman *dashboard* pengiriman akm. Data yang digunakan adalah id mahasiswa atau nrp baru, nrp lama, nama mahasiswa, kode prodi, dan periode, serta ketersediaan data di Siakad dan *Feeder* PDDikti.

2. Data Kelas

   Data kelas digunakan sebagai sumber data untuk halaman *dashboard* pengiriman data kelas yang di dalamnya juga terdapat data

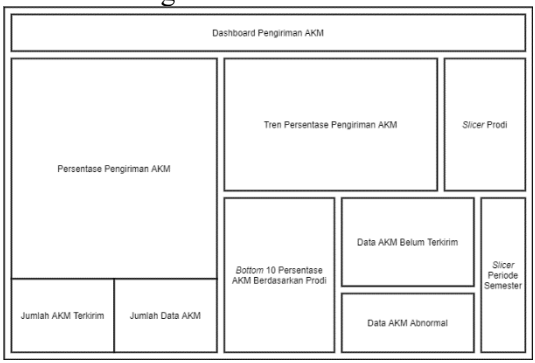
aktivitas mengajar dosen. Data yang digunakan dari kelas adalah id kelas, kode prodi, periode, kode matakuliah, nama matakuliah, dan nama kelas. Selain itu, ketersediaan data di Siakad dan *Feeder* PDDikti juga diperlukan.

3. Data Prodi
   Data prodi digunakan untuk *filter* di setiap halaman *dashboard* dan hanya memerlukan data kode prodi dan nama prodi.
4. Data Dosen
   Data dosen diperlukan sebagai sumber data pendukung untuk halaman *dashboard* penyelesaian komplain dan ketepatan waktu. Data yang dibutuhkan adalah id dosen dan nama dosen yang bersangkutan.
5. Data Aktivitas Kuliah
   Data aktivitas kuliah juga diperlukan sebagai sumber data pendukung untuk halaman *dashboard* penyelesaian komplain dan ketepatan waktu. Data yang dibutuhkan adalah id aktivitas kuliah dan nama aktivitas kuliah.
6. Data Komplain
   Data komplain digunakan sebagai sumber data untuk halaman *dashboard* penyelesaian komplain. Data yang digunakan adalah id komplain, nama data yang dikeluhkan, kode prodi, periode, status penyelesaian komplain, tanggal komplain masuk, dan tanggal komplain selesai jika status komplain telah selesai.
7. Data Antrian
   Data Antrian digunakan sebagai sumber data untuk halaman *dashboard* ketepatan waktu pengiriman data. Data yang dibutuhkan dari antrian adalah nama data yang dikirimkan beserta kode prodi dan periodenya, status antrian, dan catatan waktu pada saat antrian selesai atau data sudah dikirimkan.

d. Rancangan tata letak *dashboard*

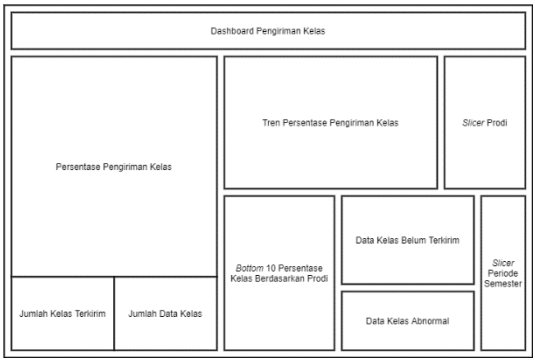
Perancangan tata letak *dashboard* merupakan proses yang dilakukan untuk menggambarkan tata letak *dashboard* yang akan dibuat. Tahap ini bertujuan untuk mempermudah penulis dalam membuat halaman *dashboard* di *Microsoft Power BI*. Rancangan awal tata letak *dashboard* dibuat berdasarkan halaman *dashboard* yang telah diidentifikasi pada tahpa sebelumnya. Setelah itu dilakukan analisis halaman *dashboard* seperti yang ditunjukkan pada **LAMPIRAN B**. Berikut rancangan tata letak *dashboard* yang telah dibuat ditunjukkan pada gambar 4.13, gambar 4.14, gambar 4.15, dan gambar 4.16.

1. Halaman Pengiriman AKM



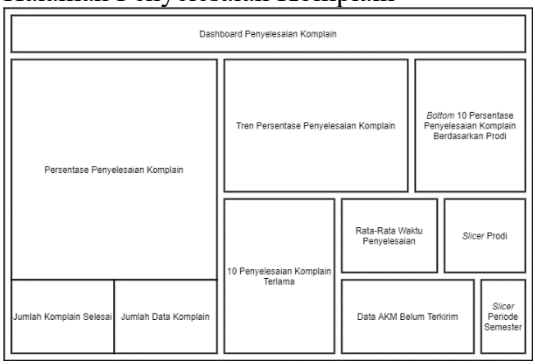
Gambar 4.13 Rancangan Halaman Pengiriman AKM

2. Halaman Pengiriman Kelas



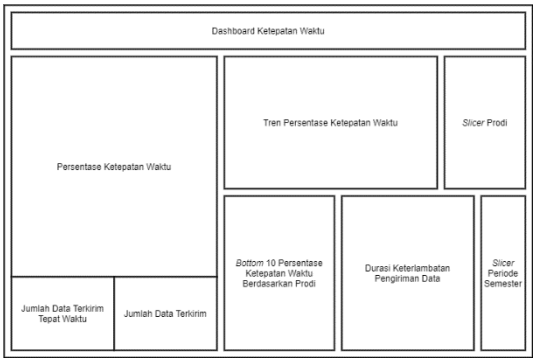
Gambar 4.14 Rancangan Halaman Pengiriman Kelas

3. Halaman Penyelesaian Komplain



Gambar 4.15 Rancangan Halaman Penyelesaian Komplain

4. Halaman Ketepatan Waktu



Gambar 4.16 Rancangan Halaman Ketepatan Waktu

e. Desain *data warehouse*

Tahap ini dilakukan untuk membuat desain *data warehouse* menggunakan model *star schema* dengan mengadopsi teknik pembuatan model data warehouse oleh Ralph Kimball [22]. Desain ini dibuat berdasarkan analisis halaman *dashboard* dan disesuaikan dengan sumber data yang tersedia. Hal ini bertujuan untuk mempermudah penulis dalam membuat skema *data warehouse* yang menunjukkan hubungan antara tabel fakta dan dimensi yang digunakan. *Data warehouse* digunakan sebagai sumber data bagi *dashboard* dan menampung data setelah dilakukan *ETL* dari *database* ITS *Feeder Bridge*. Berikut desain *data warehouse* yang telah penulis buat dalam bentuk tabel fakta dan dimensi seperti pada Tabel 4.22, Tabel 4.23, Tabel 4.24, dan Tabel 4.25.

Tabel 4.22 Tabel Fakta AKM

Nama Tabel	Jenis Tabel	Nama Kolom
fact_akm	Fakta	id_mahasiswa, id_prodi, id_periode, jumlah_data_akm, jumlah_data_akm_pd dikti

dim_prodi	Dimensi	id_prodi, nama_prodi
dim_periode	Dimensi	id_periode, nama_periode
dim_mahasiswa	Dimensi	id_mahasiswa, nim, nama_mahasiswa

Tabel 4.23 Tabel Fakta Kelas

Nama Tabel	Jenis Tabel	Nama Kolom
fact_pengiriman_kelas	Fakta	id_kelas, id_prodi, id_periode, jumlah_data_kelas, jumlah_data_kelas_pddikti
dim_prodi	Dimensi	id_prodi, nama_prodi
dim_periode	Dimensi	id_periode, nama_periode
dim_kelas	Dimensi	id_kelas, nama_kelas
dim_matkul	Dimensi	id_matkul, nama_matkul

Tabel 4.24 Tabel Fakta Penyelesaian Komplain

Nama Tabel	Jenis Tabel	Nama Kolom
fact_komplain	Fakta	id_komplain, id_prodi, id_periode, jumlah_komplain, jumlah_komplain_selesai, selisih_waktu
dim_prodi	Dimensi	id_prodi, nama_prodi
dim_periode	Dimensi	id_periode, nama_periode
dim_komplain	Dimensi	id_komplain, nama_komplain

Tabel 4.25 Tabel Fakta Ketepatan Waktu

Nama Tabel	Jenis Tabel	Nama Kolom
fact_ketepatan_waktu	Fakta	id_data, id_prodi, id_periode, jumlah_data_terkirim, jumlah_terkirim_tepat_waktu, delta_waktu
dim_prodi	Dimensi	id_prodi, nama_prodi
dim_periode	Dimensi	id_periode, nama_periode
dim_data_waktu	Dimensi	id_data, nama_data

Dari tabel-tabel tersebut dapat diketahui bahwa ada beberapa tabel dimensi yang sama antara satu fakta dengan lainnya, oleh karena itu dapat dijadikan satu seperti yang ditunjukkan dengan skema *database* pada **LAMPIRAN E DOKUMEN *SRS FEEDER BRIDGE***.

f. Perencanaan proses *ETL*

Proses *ETL* adalah proses yang menangani masalah pembersihan dan pemuatan data dari sumber data ke *data warehouse* [23]. Proses tersebut mengambil data yang dibutuhkan sesuai dengan desain *data warehouse* yang telah dibuat. Data tersebut diambil dari sumber data terkait, disesuaikan dengan *data warehouse*, dan dimuat ke dalam *data warehouse*. Proses *ETL* dibuat menggunakan bahasa pemrograman *Python* dalam bentuk *script*. *Script* tersebut berisi perintah untuk mengambil data yang dibutuhkan dari *database* ITS *Feeder Bridge* dan disesuaikan dengan *data warehouse*. Untuk menjalankan proses *ETL*, *script* harus dijalankan

dengan dua argumen yaitu jenis data *dashboard* yang ingin dimuat dan kode prodi. Adapun argumen jenis data *dashboard* yang dimaksud adalah "akm", "kelas", "komplain", dan "waktu". Secara sederhana, aliran proses *ETL* dicantumkan dalam **LAMPIRAN C DIAGRAM ALUR PROSES *ETL*** Berikut penjelasan proses *ETL* untuk setiap halaman *dashboard*.

1. Pengiriman akm
   Id mahasiswa, nim lama, nama mahasiswa, data *JSON* siakad dan *feeder* yang ada di tabel mahasiswa ITS *Feeder Bridge* diambil dengan kueri *MySQL*. Selanjutya id mahasiswa, nim lama, dan nama mahasiswa di-*insert* kedalam tabel dimensi mahasiswa dengan *filter* prodi. Selanjutnya mengisi dimensi prodi dan periode dengan mengambil data dari tabel prodi untuk nama prodi dan *JSON* siakad di tabel mahasiswa untuk periode. Setelah dimensi mahasiswa, prodi, dan periode terisi, selanjutnya adalah memasukkan data id mahasiswa, nama mahasiswa, prodi, dan periode ke tabel fakta akm sesuai kolomnya. Tahap selanjutnya adalah melakukan pembaharuan pada jumlah data akm dan jumlah data akm PDDikti. Jumlah data akm diketahui dengan menghitung setiap periode di dalam data *JSON* siakad mahasiswa yang bersangkutan, sedangkan jumlah data akm PDDikti diketahui dari periode di dalam data *JSON feeder*.
2. Pengiriman Kelas
   Id kelas, prodi, data *JSON* matkul kelas periode, data *JSON* siakad dan *feeder* yang ada di tabel kelas ITS *Feeder Bridge* diambil dengan kueri *MySQL*. Selanjutya id kelas dan nama kelas dari data *JSON* siakad di-*insert*

kedalam tabel dimensi kelas dengan *filter* prodi. Dimensi matakuliah diisi dengan mengambil data kode matakuliah dan nama matakuliah dari data *JSON* matkul kelas periode dan *JSON* siakad kemudian di-*insert* ke dalam tabel dimensi matakuliah. Selanjutnya mengisi dimensi prodi dan periode dengan mengambil data dari tabel prodi untuk nama prodi dan data *JSON* matkul kelas periode di tabel kelas untuk periode. Setelah dimensi kelas, matakuliah, prodi, dan periode terisi, selanjutnya adalah memasukkan data id kelas, prodi, periode, dan kode matakuliah ke tabel fakta kelas sesuai kolomnya. Tahap selanjutnya adalah melakukan permbaharuan pada jumlah data kelas dan jumlah data kelas PDDikti. Jumlah data kelas diketahui dengan menghitung data *JSON* SIAKAD dari data kelas yang bersangkutan, sedangkan jumlah data kelas PDDikti diketahui dengan menghitung data *JSON feeder*.

3. Penyelesaian Komplain
   Tahap ini diawali dengan mengambil id komplain, *JSON* data komplain, tanggal komplain masuk, status penyelesaian, dan tanggal komplain selesai dari tabel komplain dengan kueri *MySQL*. Selanjutnya adalah melihat *event* yang ada di dalam *JSON* data komplain. Jika *event* komplain adalah komplain data mahasiswa maka mengambil data yang dibutuhkan dari tabel mahasiswa, begitu juga dengan *event* komplain yang lain yaitu komplain data kelas, komplain data dosen, komplain data prodi, komplain data matakuliah, dan komplain data aktivitas kuliah. Adapun data yang dibutuhkan dari masing-masing tersebut adalah id data dan nama data

yang selanjutnya dimasukkan ke dalam tabel dimensi data komplain. Selanjutnya mengisi dimensi prodi dan periode dengan mengambil data dari tabel prodi untuk nama prodi dan data *JSON* data komplain di tabel komplain untuk periode. Setelah dimensi data komplain, prodi, dan periode terisi, selanjutnya adalah memasukkan data id komplain, prodi, dan periode ke dalam tabel fakta komplain. Tahap selanjutnya adalah melakukan permbaharuan pada jumlah komplain dan jumlah komplain selesai. Jumlah data komplain diketahui dengan menghitung seluruh data komplain yang bersangkutan di tabel komplain, sedangkan jumlah data komplain selesai diketahui dengan menghitung data komplain yang statusnya selesai. Selain itu pembaharuan juga dilakukan pada selisih waktu yang didapat dengan mengurangkan data tanggal komplain masuk dengan data komplain selesai yang ada di tabel komplain.

4. Ketepatan Waktu

   Tahap ini diawali dengan mengambil data id antrian, *JSON* data antrian, tanggal antrian, dan status antrian dari tabel antrian dengan kueri *MySQL*. Selanjutnya adalah melihat *event* yang ada di dalam *JSON* data anrian. Jika *event* antrian adalah kirim, kemudian melihat data yang dikirimkan. Jika data yang dikirimkan adalah data mahasiswa maka mengambil data yang dibutuhkan dari tabel mahasiswa, begitu juga dengan data kirim yang lain yaitu data kelas, data dosen, data prodi, data matakuliah, dan data aktivitas kuliah. Adapun data yang dibutuhkan dari masing-masing tersebut adalah id data dan nama data yang selanjutnya dimasukkan ke dalam tabel dimensi data

waktu. Selanjutnya mengisi dimensi prodi dan periode dengan mengambil data dari tabel prodi untuk nama prodi dan *JSON* data antrian di tabel antrian untuk periode. Setelah dimensi data waktu, prodi, dan periode terisi, selanjutnya adalah memasukkan id data, prodi, dan periode ke dalam tabel fakta ketepatan waktu. Tahap selanjutnya adalah melakukan permbaharuan pada jumlah data terkirim dan jumlah terkirim tepat waktu. Jumlah data terkirim diketahui dengan menghitung seluruh jumlah data antrian dengan *event* kirim dan status selesai. Untuk data terkirim tepat waktu, diketahui dengan membandingkan waktu data terkirim dengan jadwal sesuai dengan periode pelaporan PDDikti kemudian menghitung data yang terkirim tepat waktu.

g. Instrumen *usability testing dashboard*

Tahap ini dilakukan untuk membuat instrumen pengujian *dashboard* dengan cara membuat skenario pengujian untuk masing-masing halaman *dashboard*. Pengujian yang digunakan adalah *Usability Testing* untuk mengetahui tingkat keberhasilan skenario penggunaan *dashboard* terhadap pengguna. Selain skenario pengujian, penulis juga menyiapkan rencana untuk pengujian.

1. Skenario pengujian pengguna

   Skenario pengujian terdiri dari beberapa skenario dengan satu aktivitas atau lebih. Tujuan dari skenario pengujian ini adalah untuk melihat apakah rancangan *dashboard* yang telah penulis buat dapat digunakan oleh pengguna atau tidak. Berikut merupakan skenario yang telah dibuat oleh penulis dan akan dilakukan oleh pengguna pada saat pengujian yang ditunjukkan pada Tabel 4.26, Tabel 4.27, Tabel 4.28, dan Tabel 4.29.

Tabel 4.26 Skenario Pengujian Pengiriman AKM

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SPA01	Anda ingin mengetahui persentase pengiriman data akm dan angka data yang sudah terkirim	APA01	Melihat halaman pengiriman akm yang menampilkan persentase pengiriman akm
		APA02	Melihat halaman pengiriman akm yang menampilkan jumlah data terkirim
SPA02	Anda ingin melihat tren persentase akm dari satu periode ke periode yang lain	APA03	Melihat halaman pengiriman akm yang menampilkan tren persentase pengiriman akm
SPA03	Anda ingin melihat prodi dengan persentase pengiriman terendah	APA04	Melihat halaman pengiriman akm yang menampilkan 10 prodi dengan persentase akm terendah
SPA04	Dari persentase itu, anda diminta untuk melihat	APA05	Pengguna memilih periode semester 20181 dari *filter*

	data pengiriman akm hanya pada periode semester 20181 dari prodi S1 Sistem Informasi saja		
		APA06	Pengguna memilih prodi S1 Sistem Informasi dari *filter*
SPA05	Anda ingin melihat data akm yang belum terkirim ke *Feeder* PDDikti	APA07	Melihat halaman pengiriman akm yang menampilkan data akm yang belum terkirim
SPA06	Anda ingin mengetahui apakah ada data yang tidak normal	APA08	Melihat halaman pengiriman akm yang menampilkan data akm abnormal

Tabel 4.27 Skenario Pengujian Pengiriman Kelas

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SPK01	Anda ingin mengetahui persentase pengiriman data kelas dan angka data yang sudah terkirim	APK01	Melihat halaman pengiriman kelas yang menampilkan persentase pengiriman kelas
		APK02	Melihat halaman pengiriman kelas yang menampilkan jumlah data terkirim
SPK02	Anda ingin melihat tren persentase kelas dari satu periode ke periode yang lain	APK03	Melihat halaman pengiriman kelas yang menampilkan tren persentase pengiriman kelas
SPK03	Anda ingin melihat prodi dengan persentase pengiriman terendah	APK04	Melihat halaman pengiriman kelas yang menampilkan 10 prodi dengan persentase kelas terendah
SPK04	Dari angka persentase itu, anda diminta	APK05	Pengguna memilih periode

	untuk melihat data pengiriman kelas hanya pada periode semester 20181 dari prodi S1 Sistem Informasi saja		semester 20181 dari *filter*
		APK06	Pengguna memilih prodi S1 Sistem Informasi dari *filter*
SPK05	Anda ingin melihat data kelas yang belum terkirim ke *Feeder* PDDikti	APK07	Melihat halaman pengiriman kelas yang menampilkan data kelas yang belum terkirim
SPK06	Anda ingin mengetahui apakah ada data yang tidak normal	APK08	Melihat halaman pengiriman kelas yang menampilkan data kelas abnormal

Tabel 4.28 Skenario Pengujian Komplain Data

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SKD01	Anda ingin mengetahui persentase penyelesaian	AKD01	Melihat halaman penyelesaian komplain yang

	komplain dan angka komplain yang sudah diselesaikan		menampilkan persentase penyelesaian komplain
		AKD02	Melihat halaman penyelesaian komplain yang menampilkan komplain selesai
SKD02	Anda ingin melihat tren persentase penyelesaian komplain dari satu periode ke periode yang lain	AKD03	Melihat halaman penyelesaian komplain yang menampilkan tren persentase penyelesaian komplain
SKD03	Anda ingin melihat prodi dengan persentase penyelesaian komplain terendah	AKD04	Melihat halaman penyelesaian komplain yang menampilkan 10 prodi dengan persentase penyelesaian komplain terendah
SKD04	Dari persentase itu, anda diminta untuk melihat data penyelesaian komplain hanya pada	AKD05	Pengguna memilih periode semester 20181 dari *filter*

	periode semester 20181 dari prodi S1 Sistem Informasi saja		
		AKD06	Pengguna memilih prodi S1 Sistem Informasi dari *filter*
SKD05	Anda ingin melihat durasi penyelesaian komplain	AKD07	Melihat halaman penyelesaian komplain yang menampilkan 10 komplain dengan durasi penyelesaian paling lama.
SKD06	Anda ingin melihat data komplain yang belum diselesaikan	AKD08	Melihat halaman penyelesaian komplain yang menampilkan data komplain yang belum diselesaikan

Tabel 4.29 Skenario Pengujian Ketepatan Waktu

ID Skenario	Skenario	ID Aktivitas	Aktivitas
SKW01	Anda ingin mengetahui persentase ketepatan waktu dan	AKW01	Melihat halaman ketepatan waktu yang menampilkan

	angka data yang sudah terkirim tepat waktu		persentase ketepatan waktu
		AKW02	Melihat halaman ketepatan waktu yang menampilkan jumlah data terkirim tepat waktu
SKW02	Anda ingin melihat tren persentase ketepatan waktu dari satu periode ke periode yang lain	AKW03	Melihat halaman ketepatan waktu yang menampilkan tren persentase ketepatan waktu
SKW03	Anda ingin melihat prodi dengan persentase terendah	AKW04	Melihat halaman ketepatan waktu yang menampilkan 10 prodi dengan persentase ketepatan waktu terendah
SKW04	Dari angka persentase itu, anda diminta untuk melihat data ketepatan waktu hanya pada periode semester	AKW05	Pengguna memilih periode semester 20181 dari *filter*

	20181 dari prodi S1 Sistem Informasi saja		
		AKW06	Pengguna memilih prodi S1 Sistem Informasi dari *filter*
SKW05	Anda ingin melihat selisih waktu pengiriman data dengan jadwal	AKW07	Melihat halaman ketepatan waktu yang menampilkan selisih waktu

2. *Usability Test Plan*
   Pengguna menyiapkan *Usability Test Plan* yang berisi rencana pengujian sebagai panduan bagi penulis dalam melakukan pengujian. Berikut *Usability Test Plan dashboard* yang ditujukkan pada Tabel 4.30.

Tabel 4.30 *Usability Test Plan Dashboard*

Tujuan Pengujian	Untuk mengetahui apakah pengguna dapat menggunakan prototipe dengan skenario tertentu
Partisipan Pengujian	Pihak dari Dirpendik untuk uji coba *dashboard* dan operator dari departemen untuk uji coba aplikasi
Durasi Pengujian	60 menit dengan rincian sebagai berikut: 1. Pembukaan (5 menit) 2. Uji coba skenario (50 menit) 3. Wawancara (5 menit)

Peran & Tanggung Jawab	Penulis akan betindak sebagai moderator, pengamat, dan pencatat dengan rincian tugas sebagai berikut. - Moderator bertugas untuk menyapa dan menyampaikan instruksi kepada partisipan sekaligus menjadi teknisi - Pengamat bertugas mengamati partisipan saat melakukan uji coba termasuk emosi partisipan melalui raut wajah - Pencatat bertugas untuk mencatat semua kejadian dan komentar yang dikeluarkan oleh partisipan

4.3.4. Perhitungan hasil *usability testing*

Hasil *Usability Testing* baik untuk aplikasi maupun *dashboard* diberikan skor dengan tiga kategori, antara lain:

a. 1 = Tidak berhasil menjalankan aktivitas yang diminta.
b. 2 = Berhasil menjalankan aktivitas yang diminta namun mengalami kendala atau hanya berhasil menyelsaikan sebagian skenario.
c. 3 = Berhasil menjalankan aktivitas tanpa kendala.

Pengukuran hasil pengujian dilakukan menggunakan rumus yang dijabarkan oleh Jakob Nielsen [24] untuk mengetahui *success rate*. Berikut rumusnya.

$$Success\ rate(\%) = \frac{S + (P \times 0.5)}{N}$$

Keterangan:

S = Jumlah aktivitas yang berhasil dijalankan

P = Jumlah aktivitas yang berhasil dijalankan sebagian atau mengalami kendala

N = Jumlah seluruh aktivitas

4.3.5. Rancangan *open-ended question*

Pada tahap ini penulis membuat rancangan pertanyaan terbuka untuk ditanyakan kepada partisipan pengujian setelah dilakukan *usability testing*. Daftar pertanyaan yang telah penulis buat ditunjukkan dengan Tabel 4.31.

Tabel 4.31 *Open-ended Question*

No	*Open-ended Question*
1	Bagaimana pendapat anda mengenai web ini sebelum dan setelah mencoba web?
2	Bagaimana menurut amda tentang fitur sinkronisasi web kami? Apakah ada kesulitan saat melakukan sinkronisasi data? Jika iya di bagian mana dan mengapa? Bagaimana ekspektasi anda sebelum dan sesudah mencoba fitur tersebut?
3	Bagaimana menurut anda tentang tampilan data dari web kami? Apakah ada kesulitan saat ingin menampilkan data? Jika iya di bagian mana dan mengapa? Bagaimana ekspektasi anda sebelum dan sesudah mencoba menampilkan data?
4	Bagaimana menurut anda tentang fitur kirim data web kami? Apakah ada kesulitan saat ingin mengirimkan data? Jika iya di bagian mana dan mengapa? Bagaimana ekspektasi anda sebelum dan sesudah mencoba fitur tersebut?
5	Bagaimana menurut anda tentang proses antrian dari web kami? Apakah ada kesulitan saat ingin melihat halaman proses antrian dan eror pengiriman data? Jika iya di bagian mana dan mengapa? Bagaimana ekspektasi anda sebelum dan sesudah mencoba fitur tersebut?
6	Bagaimana menurut anda tentang fitur manajemen pengguna dari web kami? Apakah ada kesulitan saat ingin membuat atau mengubah akses data dari akun pengguna? Jika iya di bagian mana dan mengapa? Bagaimana ekspektasi anda sebelum dan sesudah mencoba fitur tersebut?

7	Bagaimana menurut anda tentang fitur *dashboard* dari web kami? Apakah ada kesulitan saat ingin membuka halaman *dashboard* yang diinginkan? Jika iya di bagian mana dan mengapa? Bagaimana ekspektasi anda sebelum dan sesudah mencoba fitur tersebut?
8	Selain itu apakah ada kesulitan lain yang kamu alami? Jika ada, di bagian mana dan mengapa?
9	Jika web kami dibandingkan dengan *ProFeeder* saat ini, bagaimana pendapat anda?
10	Bagaimana saran anda untuk pengembangan web ke depannya?

Pertanyaan-pertanyaan tersebut nantinya dapat dikembangkan lagi untuk mendapatkan hasil wawancara yang lebih mendalam menggunakan teknik 5 *Why's Analysis* tergantung dari jawaban partisipan pengujian.

4.3.6. Rencana pengujian performa dan beban aplikasi

Pengujian performa dilakukan dari sisi *server Feeder Bridge* untuk mengetahui beban *server* untuk menjalankan aplikasi. Kemudian dilakukan pengujian performa dari sisi pengguna untuk membandingkan beban *Feeder Bridge* dan *ProFeeder* pada masing-masing pengguna. Penulis sebagai penguji dan pengamat akan meminta partisipan untuk menjalankan aplikasi *Feeder Bridge* secara langung menggunakan komputer milik partisipan. Selanjutnya penulis akan mengamati beban *CPU* dan memori (*RAM*) dari aplikasi melalui fitur *Task Manager* pada komputer partisipan saat aplikasi sedang berjalan. Kemudian penulis akan meminta pengguna untuk menjalankan aplikasi *ProFeeder*. Penulis juga akan mengamati beban *CPU* dan memori (*RAM*) dari aplikasi *ProFeeder* untuk dilakukan perbandingan. Pada tahap ini penulis juga meminta partisipan untuk melakukan aktivitas sinkronisasi data

menggunakan data yang serupa pada kedua aplikasi. Selanjutnya penulis akan mengamati waktu kedua aplikasi dalam menyelesaikan proses sinkronisasi untuk dilakukan perbandingan. Sementara dari sisi *server*, penulis akan mengamati penggunaan *CPU* dan *RAM server* menggunakan *tools* bernama TOP.

BAB V
PENGEMBANGAN

Bab ini berisi hasil dari tahap pengembangan aplikasi meliputi web dan sistem antrian serta *dashboard* yang telah penulis lakukan. Pada tahap ini dilakukan implementasi desain sistem yang telah dibuat sebelumnya ke dalam baris kode. Tahap ini dilakukan selama kurang lebih 2 bulan dengan 7 kali iterasi untuk web dan sistem antrian serta 6 kali iterasi untuk *dashboard.* Setiap iterasi membutuhkan waktu kurang lebih 1 minggu pengerjaan yang pada setiap akhir minggu dilakukan evaluasi terhadap hasil pengerjaan 1 kali iterasi pada minggu tersebut, serta perencanaan target pengerjaan pada iterasi berikutnya. Hasil dari setiap iterasi adalah pengembangan fitur aplikasi atau baris kode hasil perencanaan dan evaluasi pada iterasi sebelumnya. Adapun target dari setiap iterasi dari web dan sistem antrian adalah sebagai berikut:

a. Target dari iterasi pertama adalah halaman tampilan data yang yang menampilkan dan membandingkan data SIAKAD dan PDDikti. Pada tahap ini tampilan data yang pertamakali dibuat adalah data mahasiswa.
b. Pada iterasi kedua dilakukan penyempurnaan tampilan data hasil evaluasi dari iterasi pertama.
c. Target dari iterasi ketiga adalah aplikasi dapat melakukan sinkronisasi dan pengiriman data.
d. Target dari iterasi keempat adalah tampilan data untuk data yang lain seperti data kelas, dosen, mata kuliah, dan program studi. Pada tahap ini juga dilakukan pembuatan fitur sinkronisasi dan kirim data untuk data-data tersebut.
e. Pada iterasi kelima dilakukan pembuatan fitur *login*, registrasi akun pengguna, serta hapus akun pengguna.
f. Target dari iterasi keenam adalah fitur manajemen akses pengguna.
g. Target dari iterasi ketujuh adalah halaman untuk menampilkan proses data antrian serta *error log* pengiriman data.

Sedangkan untuk target dari setiap iterasi *dashboard* adalah sebagai berikut:

a. Pada iterasi pertama dilakukan pembuatan halaman *dashboard* pengiriman data.
b. Target dari iterasi kedua adalah *dashboard* dapat menampilkan visualisasi pengiriman data kelas. Pada iterasi ini juga dibuat visualisasi yang menampilkan data yang tidak sesuai yang merupakan hasil evaluasi dari iterasi sebelumnya.
c. Pada iterasi ketiga dilakukan pembuatan *filter* prodi, halaman visualisasi penyelesaian komplain, serta halaman visualisasi ketepatan waktu.
d. Pada iterasi keempat dilakukan penyesuaian tata letak *dashboard* serta penyesuaian visualisasi data yang terlambat pada halaman ketepatan waktu hasil dari evaluasi pada iterasi sebelumnya. Pada iterasi ini juga dilakukan pembuatan visualisasi durasi penyelesaian komplain dan rata-rata durasinya pada halaman penyelesaian komplain.
e. Pada iterasi kelima dilakukan penyesuaian *filter* hasil dari evaluasi iterasi sebelumnya serta membagi halaman ketepatan waktu sesuai gelombang pelaporan.

Adapun saat evaluasi aplikasi dan *dashboard*, penulis juga meminta evaluasi dari anggota Sub Direktorat Pengelolaan Teknologi *Big Data* DPTSI yang lain. Berikut informasi narasumber yang ditunjukkan pada Tabel 5.1 dan Tabel 5.2.

Tabel 5.1 Narasumber 1

Nama	Arief Pramono
Jabatan	Staff Pengelolaan Teknologi Big Data

Tabel 5.2 Narasumber 2

Nama	Inayati Fajriyah, S.Si
Jabatan	Staff Pengelolaan Teknologi Big Data

Dikarenakan pandemi COVID-19 yang menyebabkan kegiatan perkuliahan luring ditiadakan dan akses ke kampus dibatasi, penulis mengalami kendala dalam proses evaluasi karena tidak dapat setiap saat bertemu dengan pihak DPTSI untuk meminta *feedback*. Setelah dilakukan iterasi pengembangan aplikasi dan *dashboard*, selanjutnya dilakukan tahap pengujian. Pengujian dilakukan dengan menggunakan *usability test.* Penulis akan

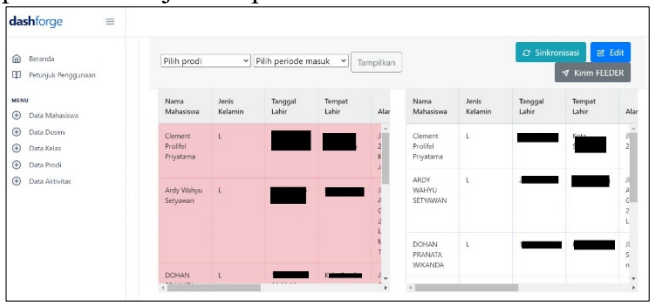
melakukan pengamatan dan pencatatan terhadap bagaimana perilaku pengguna saat menggunakan aplikasi. Hasil pengujian akan digunakan sebagai bahan perbaikan atau modifikasi kekurangan dari aplikasi.

5.1. Pengembangan Web dan Sistem Antrian

Tahap pengembangan web dan sistem antrian dilakukan dengan menggunakan metode *RAD* dimana terdapat beberapa iterasi pada tahap pengembangannya. Tahap ini dilakukan setelah perencanaan dan pengumpulan *requirement* aplikasi.

5.1.1. Iterasi pertama

Pada iterasi pertama pengembangan aplikasi, dilakukan pembuatan tampilan aplikasi yang sudah direncanakan serta implementasi dari *ERD*. Pertama kali dilakukan pembuatan *view* untuk menampilkan dan membandingkan data SIAKAD dan PDDikti. Hasil tampilan aplikasi tahap pertama merupakan hasil dari perencanaan awal desain aplikasi yang merujuk pada tampilan sistem saat ini seperti yang ditunjukkan pada Gambar 3.4 dengan menampilkan perbandingan data SIAKAD dan PDDikti melalui 2 tabel. Pada tahap pertama dilakukan pembuatan tampilan untuk data mahasiswa terlebih dahulu. Tampilan aplikasi tahap pertama ditunjukkan pada Gambar 5.1.



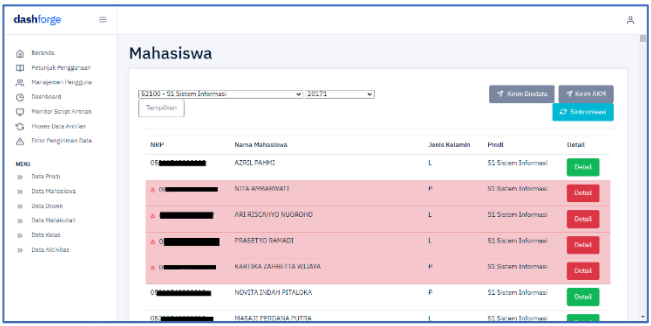
Gambar 5.1 Halaman Tampilan Data Tahap Pertama

Sementara untuk implementasi *ERD*, akan dibuat *database* sesuai dengan entitas-entitas *ERD* versi kedua pada Tabel 4.11 Namun pembuatan *ERD* perlu mendapat tanggapan karena hal ini terkait dengan proses pengambilan data dari SIAKAD dan PDDikti. Apabila

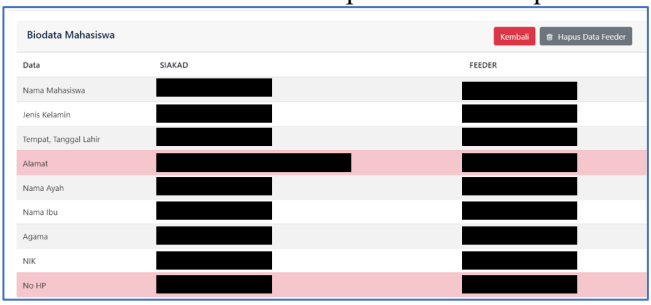
penulis salah dalam menentukan data maka data yang disimpan di aplikasi juga bisa menjadi salah.

5.1.2. Iterasi kedua

Dari hasil tampilan aplikasi pada iterasi pertama, penulis mendapatkan umpan balik bahwa dengan tampilan tahap pertama, pengguna mengalami kesulitan dalam membandingkan data tiap kolom secara detail antara data SIAKAD dan *Feeder* PDDikti ketika data tersebut memiliki banyak kolom, sehingga penulis mengubah tata letak tampilan data seperti yang ditunjukkan pada Gambar 5.2 dan Gambar 5.3.



Gambar 5.2 Halaman Tampilan Data Tahap Kedua



Gambar 5.3 Halaman Detail Data

Gambar 5.3 merupakan tampilan awal data mahasiswa tahap kedua. Pada tampilan tahap kedua pengguna akan disajikan tampilan data secara umum yang telah disinkronisasi. Baris berwarna merah menunjukkan bahwa terdapat perbedaan data antara SIAKAD dan PDDikti. Selanjutnya pengguna dapat melihat data lebih

detail dengan memilih tombol detail sehingga sistem akan menampilkan detail perbandingan data SIAKAD dan PDDikti seperti yang ditunjukkan pada Gambar 5.4. Pada tampilan tersebut pengguna lebih mudah dalam menemukan data yang berbeda. Pada contoh Gambar 5.4 data yang berbeda ditunjukkan dengan baris berwarna merah yaitu alamat dan nomor *HP*. Untuk pengecekan data yang berbeda dilakukan pencocokan karakter *string* pada setiap data. Berikut merupakan potongan kode dari pencocokan data yang ditunjukkan pada Kode 5.1.

```
if
(strtolower($jsonSiakad['biodata']['nama_maha
siswa']) !==
strtolower($json*Feeder*['biodata']['nama_mahas
iswa'])) {
  echo 'table-danger';
}
```

Kode 5.1 Potongan Kode Perbandingan Data

Pada iterasi ini *ERD* juga mendapat respon yang kemudian mengubah *ERD* seperti yang ditujukkan pada **LAMPIRAN E DOKUMEN *SRS FEEDER BRIDGE***. Adapun entitas yang dimiliki oleh *ERD* ditunjukkan pada Tabel 5.3.

Tabel 5.3 Entitas *ERD* Terbaru

Nama Entitas	**Keterangan**
Mahasiswa	Entitas mahasiswa berisi semua data terkait mahasiswa mulai dari kelas, biodata mahasiswa, hingga aktivitas perkuliahan mahasiswa.
Mahasiswa Kelas	Entitas mahasiswa kelas merupakan tabel turunan dari entitas mahasiswa dan entitas kelas.
Kelas	Entitas kelas berisi semua data terkait kelas seperti anggota kelas, mata kuliah, dan dosen pengajar.
Dosen	Entitas dosen berisi data biodata dosen.
Mata Kuliah	Entitas mata kuliah berisi data mata kuliah dalam kurikulum.
Prodi	Entitas prodi berisi data tentang program studi di ITS.

ERD terbaru menerapkan *document-based database* sehingga data lebih mudah dibaca pada tingkat aplikasi dan mengurangi *join table*.

5.1.3. Iterasi ketiga

Pada iterasi ketiga dilakukan pembuatan fitur sinkronisasi data. Fitur sinkronisasi data bertujuan untuk mengambil data dari *database* SIAKAD serta data dari web *API Feeder* PDDikti yang kemudian dimasukkan kedalam *database Feeder Bridge* untuk dilakukan komparasi data. Serta fitur pengiriman data SIAKAD ke *Feeder* PDDikti. Sinkronisasi dan pengiriman data dilakukan untuk setiap jenis data seperti data program studi, mahasiswa, dosen, matakuliah, kelas, serta aktivitas. Pada iterasi ketiga dilakukan pembuatan fitur sinkronisasi dan pengiriman untuk data mahasiswa terlebih dahulu. Pembuatan fitur sinkronisasi dilakukan melalui tiga komponen arsitektur *CodeIgniter* pada aplikasi web yaitu *controller, view*, dan *model*. Berikut merupakan komponen dari fitur sinkronisasi.

a. *Controller Function Index* Sinkronisasi

Controller merupakan pusat logika dari aplikasi. Algoritma dan logika dari aplikasi ditulis pada bagian *controller* termasuk memuat *view* dan *model* dari aplikasi. Dalam hal ini *controller* juga berperan sebagai penghubung antara *model* dan *view* sehingga data yang didapat dari model dapat diproses dan ditampilkan pada halaman view. Kode *controller* sinkronisasi *function index* untuk menampilkan halaman sinkronisasi yang juga termasuk ke dalam halaman untuk menampilkan data yang ditunjukkan pada Kode 5.2.

```
 public function index()
{

        $prodi = $this->request-
        >getVar('prodi');

         $periode = $this->request-
        >getVar('periode');

        $mahasiswa = $this->mahasiswaModel-
        >mhsFilter($prodi, $periode);
        $data = [
          'test' => 'Mahasiswa',
          'mahasiswa' => $mahasiswa,
          'prodi' => $get_prodi

           'user_id' => session()-
           >get('user_id'),

    ];

    return view('mahasiswa/index', $data);
}
```

Kode 5.2 *Controller Function Index* untuk Sinkronisasi

Pada *function index* dilakukan *load view* halaman yang menampilkan tombol sinkronisasi serta *filter* data yang akan disinkronisasi berdasarkan prodi dan periode.

b. *View* Sinkonisasi

Halaman view sinkronisasi tergabung dalam halaman untuk menampilkan data yang ditunjukkan pada Gambar 5.2 Pada halaman sinkronisasi terdapat tombol sinkronisasi untuk melakukan sinkronisasi, serta *filter dropdown* prodi dan periode selain untuk menampilkan data, *filter* tersebut juga digunakan untuk memilih data apa saja yang akan disinkronisasi.

c. *Function JavaScript* Sinkronisasi

Dalam proses sinkronisasi juga melibatkan *function JavaScript* yang didalamnya terdapat *AJAX JavaScript* dalam bentuk *JQuery*. Potongan kode *JavaScript* untuk sinkronisasi ditunjukkan pada Kode 5.3.

```
    $('body').on('click',
    '#sinkronisasiBiodata', function(e) {

   e.preventDefault();

   var prodi = $('#filter-prodi').val();

   var periode = $('#filter-
periode').val();

   $.ajax({

    method: 'GET',

    url: '/mahasiswa/sinkronisasi',

    data: {

     prodi: prodi,

     periode: periode

    },

    dataType: 'json'

    }

   });

  });
```

Kode 5.3 *Function JavaScript* untuk Sinkronisasi

Kode 5.3 menerima *event* klik dari tombol sinkronisasi dimana setelah tombol diklik, maka *function JavaScript* akan menerima *value* dari *filter dropdown* prodi dan periode. Kemudian *value* prodi dan periode akan dikirim ke *function* sinkronisasi pada *controller* yang akan dijelaskan pada poin selanjutnya.

d. *Controller Function* Sinkronisasi

Function sinkronisasi pada *controller* dibuat untuk memasukkan *event* sinkronisasi serta informasi data yang akan disinkronisasi kedalam tabel antrian

untuk dijadikan *trigger* oleh sistem antrian dalam memasukkan data SIAKAD dan *Feeder* PDDikti kedalam *database* aplikasi *Feeder Bridge*. Berikut merupakan potongan kode *function* sinkronisasi ditunjukkan pada Kode 5.4.

```
  public function sinkronisasi()

  {

    $prodi = $this->request-
>getVar('prodi');

    $periode = $this->request-
>getVar('periode');

    $antrianModel = new AntrianModel();

    $datamahasiswa = json_encode([

      'user_id' => session()-
>get('user_id'),

      'prodi' => $prodi,

      'periode' => $periode,

      'event' =>
'sinkronisasi_datamahasiswa'

    ]);
```

```
    $data_m = [
      'id_antrian' => $this-
>create_uuid(),
      'data_antrian' => $datamahasiswa,
      'status' => 'ANTRI',
      'date' => date('Y-m-d h:i:s')
    ];
    if ($prodi == "" or $periode == "") {
      $status = 1;
    } else {
      $status = $antrianModel-
>insert($data_m);
    }
  }
```

Kode 5.4 *Controller Function* Sinkronisasi

Function sinkronisasi pada *controller* menerima *value* prodi dan periode yang dikirim melalui *AJAX* seperti yang dijelaskan pada poin sebelumnya. Data prodi dan periode selanjutnya akan dimasukkan kedalam tabel antrian bersama dengan data *event* 'sinkronisasi' serta status 'ANTRI'. Kemudian data tersebut digunakan oleh sistem antrian sebagai *trigger* serta informasi mengenai data apa saja yang harus disinkronisasi.

e. *Model* Antrian

Model merupakan kode untuk melakukan eksekusi kueri pada *database*. Pada *model* antrian dilakukan kueri *insert* data yang telah diterima oleh *function* sinkronisasi pada *controller*. Kueri *insert* digunakan untuk memasukkan data yang diterima *function* sinkronisasi kedalam tabel antrian melalui *model*

antrian. Berikut merupakan potongan kode *mode*l antrian yang ditujukkan pada Kode 5.5.

```
$antrianModel = new AntrianModel();
$status = $antrianModel->insert($data_m);
```

Kode 5.5 *Model* Antrian

Sementara pada sistem aplikasi, sinkronisasi adalah '*event*' yang mengharuskan sistem antrian membaca data SIAKAD dan PDDikti kemudian memetakan data tersebut ke *database* aplikasi.

f. *Function* Sinkronisasi Sistem Antrian

Ketika sistem membaca antrian baru dengan *event* sinkronisasi, maka sistem akan mulai mengambil data dari SIAKAD dan PDDikti sesuai dengan data *event.* Kemudiandata tersebut akan di-*insert* ke *database* jika belum pernah ada, atau akan di-*update* jika sudah ada. Contoh kode sinkronisasi dapat dilihat pada contoh sinkronisasi data mahasiswa pada Kode 5.6.

```
def SinkDataMahasiswa(nrp_baru):

mhs_siakad = BioMhsSiakad(nrp_baru)

bridge = connectBridge()

with bridge.cursor() as cursor:

try:

cursor.execute("SELECT id_mahasiswa,
id_registrasi FROM mahasiswa WHERE
id_mahasiswa = %s",
(mhs_siakad['nrp_lama']))

mhs = cursor.fetchone()
```

```
except (pymysql.Error, pymysql.Warning) as
e:

print(f'error! select mahasiswa {e}')

if mhs == None: # insert baru

with bridge.cursor() as cursor:

try:

SQL = """ INSERT INTO mahasiswa
(id_mahasiswa, nama_mahasiswa,
       nrp_baru, kode_prodi,
periode_masuk)

VALUES (%s, %s, %s, %s, %s, %s) """

cursor.execute(SQL,
(mhs_siakad['nrp_lama'],
       mhs_siakad['biodata']['nama_mahasis
wa'],mhs_siakad['nrp'],
       mhs_siakad['kode_prodi'],
mhs_siakad['periode_masuk']))

bridge.commit()

UpdateJSONDataMahasiswa(mhs_siakad)

except (pymysql.Error, pymysql.Warning) as
e:

print(f'error! insert baru mahasiswa {e}')
```

Kode 5.6 Function Sinkronisasi Sistem Antrian

Setelah pembuatan fitur sinkronisasi dilakukan pembuatan fitur pengiriman data SIAKAD. Pembuatan fitur pengiriman data dilakukan dengan tahapan yang tidak jauh berbeda dengan sinkronisasi data. Pengiriman data dilakukan dengan memilih *filter* data yang akan dikirim terlebih dahulu kemudian klik tombol kirim yang juga terdapat pada halaman awal tampilan data bersama dengan tombol sinkronisasi seperti yang ditunjukkan pada Gambar

5.3. Berikut merupakan komponen fitur pengiriman data:

1. *Function JavaScript*
   Setelah tombol kirim diklik, maka *event* klik akan dibaca oleh kode *JavaScript* seperti pada saat proses sinkronisasi data. Kode tersebut ditunjukkan pada Kode 5.7.

```
    $('body').on('click',
    '#kirim_Feeder_Biodata',
    function(e) {

   e.prevent_Default();

   var prodi = $('#filter-
prodi').val();

   var periode = $('#filter-
periode').val();

   $.ajax({

    method: 'GET',

    url: '/mahasiswa/kirim',

    data: {

     prodi: prodi,

     periode: periode

    },

    dataType: 'json'

    }

   });

  });
```

Kode 5.7 *Function JavaScript* Pengiriman Data

Setelah kode *function JavaScript* menerima *event* klik dari tombol kirim, selanjutnya

function tersebut akan menerima *value* dari *filter* prodi dan periode yang nantinya akan dikirim ke *function* kirim*Feeder* pada *controller* melalui *AJAX.*

2. *Controller Function* Kirim
   Seperti *function* sinkronisasi pada *controller*, *function* kirim dibuat untuk memasukkan *event* kirim serta informasi data yang akan dikirim kedalam tabel antrian untuk dijadikan *trigger* oleh sistem antrian dalam mengirim data SIAKAD ke *Feeder* PDDikti melalui API yang telah disediakan. Potongan kode *function* kirim ditunjukkan pada Kode 5.8.

```
  public function kirimFeeder()
  {    {
    $prodi = $this->request->getVar('prodi');
    $periode = $this->request->getVar('periode');
    $antrianModel = new AntrianModel();
    $datamahasiswa = json_encode([
      'user_id' => session()->get('user_id'),
      'prodi' => $prodi,
      'periode' => $periode,
      'event' => 'kirim_datamahasiswa'
    ]);
    $data_m = [
      'id_antrian' => $this->create_uuid(),
```

```
      'data_antrian' =>
$datamahasiswa,

      'status' => 'ANTRI',

      'date' => date('Y-m-d h:i:s')

    ];

    if ($prodi == "" or $periode ==
"") {

      $status = 1;

    } else {

      $status = $antrianModel-
>insert($data_m);

    }

  }
```

Kode 5.8 *Controller Function* Kirim

3. *Function* Kirim Sistem Antrian

Pada pengiriman data sistem antrian baris kode *Python* yang digunakan ditunjukkan pada Kode 5.9.

```
try:

  if kelas_feeder == None or
kelas_feeder['id_feeder_kelas'] ==
None:

       respon =
libs.InsertKelasKuliah(libs.GetToken(
), dict_kls_kuliah)

  else:
```

```
        respon =
libs.UpdateKelasKuliah(libs.GetToken(
), kelas_feeder['id_feeder_kelas'],
dict_kls_kuliah)

        filter_dosen_pengajar =
respon

except Exception as e:

  sql_antrian = "SELECT data_antrian
FROM antrian WHERE id_antrian = %s"

  SQL = """INSERT INTO error_antrian
(id, id_antrian, id_detail_antrian,
user_id, operasi, value, pesan_error)
VALUES (%s, %s, %s, %s, %s, %s,
%s)"""

  with bridge.cursor() as cursor:

        cursor.execute(sql_antrian,
antrian['id_antrian'])

        data_antri =
cursor.fetchone()

        user_id =
json.loads(data_antri['data_antrian']
)['user_id']

  with bridge.cursor() as cursor:

        cursor.execute(SQL,
(uuid.uuid4(), antrian['id_antrian'],
antrian['id'], user_id,
antrian['event'],

        json.dumps(dict_kls_kuliah),
e))
```

Kode 5.9 Kode Pengiriman Sistem Antrian

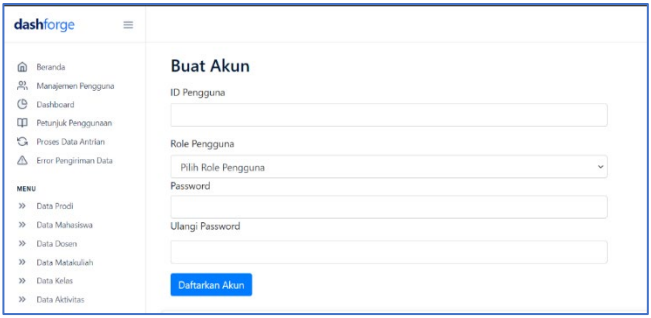
5.1.4. Iterasi keempat

Pada iterasi keempat dilakukan pembuatan tampilan untuk data kelas, dosen, matakuliah, program studi, serta

aktivitas pada aplikasi web. Pada sistem antrian dilakukan pengembangan untuk data lain seperti data dosen, matkul, program studi, dll. Pembuatan tampilan dilakukan dengan tahapan yang sama dengan data mahasiswa yang telah dilakukan pada iterasi sebelumnya. Kode yang digunakan pada data mahasiswa digunakan kembali untuk data-data lainnya dengan beberapa penyesuaian.

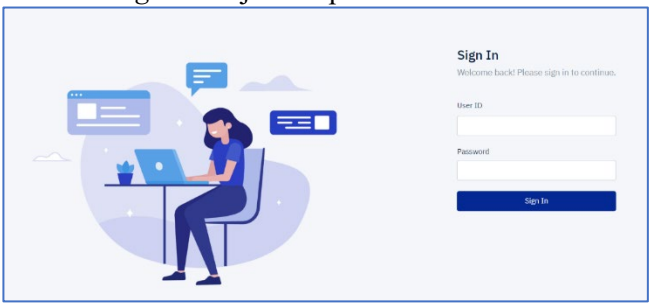
5.1.5. Iterasi kelima

Pada iterasi kelima dilakukan pembuatan fitur regitrasi akun dan *login* pada aplikasi web dan pembuatan sistem antrian yang dapat membagi tugas dalam antrian. Halaman registrasi akun hanya dapat digunakan oleh pengguna dengan role administrator. Halaman registrasi akun atau tambah pengguna ditunjukkan oleh Gambar 5.4.



Gambar 5.4 Halaman Registrasi Akun

Halaman *login* ditunjukkan pada Gambar 5.5.



Gambar 5.5 Halaman *Login*

Selanjutnya dilakukan pembuatan validasi data pengguna pada *controller*. Pertamakali dilakukan pembuatan validasi untuk registrasi akun. Data yang dimasukkan pada form registrasi seperti *user id, role*, dan *password* akan divalidasi pada *controller function registered* sebelum. Potongan kode *controller function registered* ditunjukkan pada Kode 5.10.

```
public function registered()
 {
     $userModel = new UserModel();
     if (!$this->validate([
        'user_id' => [
          'rules' =>
'required|is_unique[user.user_id]|trim',
          'errors' => [
            'required' => 'User ID is
required.',
            'is_unique' => 'This user has
already registered!'
          ]
        ],
        'role' => [
          'rules' => 'required',
          'errors' => [
            'required' => 'Rolse is
```

```
required.'
          ]
        ],
        'password1' => [
          'rules' =>
'required|trim|min_length[3]|matches[passw
ord2]',
          'errors' => [
           'required' => 'Password is
required.',
           'matches' => 'password dont
match!',
           'min_length' => 'Password too
short!'
          ]
        ], 'password2' => [
           'rules' =>
'matches[password1]',
         ]
    ]))
$validation =
\Config\Services::validation()
      } else {
        $data = [

       'user_id' => $this->request-
       >getVar('user_id'),
        'password' => password_hash($this-
       >request-  >getVar('password1'),
       PASSWORD_DEFAULT),
        'role_id' => $this->request-
       >getVar('role'),
        'date_created' => time()
       ];
       $userModel->insert($data)
       }
       }
```

Kode 5.10 *Controller Function Registered*

Selanjutnya dilakukan pembuatan validasi pada fitur *login* menggunakan *controller function login.* Potingan

kode *controller function login* ditunjukkan pada Kode 5.11.

```
public function login() {

   if (!$this->validate([
       'user_id' => [
        'rules' => 'required|trim',
        'errors' => [
          'required' => 'User ID is
required.'
         ]
       ],
       'password' => [
         'rules' => 'required|trim',
         'errors' => [
            'required' => 'Password is
required.',
          ]
         ]
     ]))
    $validation =
\Config\Services::validation();

     return redirect()->to('/login-
     feederbridge')->withInput()-
     >with('validation', $validation);

   }

 }
```

Kode 5.11 *Controller Function Login*

Sementara dalam pengembangan sistem antrian, kode yang dibuat dibagi menjadi beberapa fungsi sinkronisasi dan kirim. Kemudian sistem tersebut dirancang agar bisa berjalan 24 jam dan dapat membagi pekerjaannya sesuai *event*, prodi, dan periode. Contoh pembagian pekerjaan tersebut dapat dilihat pada Kode 5.12.

```
if operation == 'sinkronisasi':

  if data == 'datakelas':

        if 'prodi' in data_antrian and
data_antrian['prodi'] == prodi_args:


        UpdateStatusAntrian(antrian['id_antria
n'], 'PROSES')


        SinkDataKelasPerProdi(data_antrian['pr
odi'], data_antrian['periode'],
antrian['id antrian'])
```

Kode 5.12 Potongan Kode Pembagian Pekerjaan Sistem Antrian

5.1.6. Iterasi keenam

Pada iterasi keenam dilakukan pembuatan ditur manajemen pengguna. Berikut merupakan komponen fitur manajemen pengguna:

a. *View* menu manajemen pengguna

   Menu manajemen pengguna digunakan oleh admin DPTSI untuk membuat akun pengguna dengan mengakses halaman registrasi, mengubah hak akses pengguna, serta menghapus akun pengguna yang telah dibuat. Berikut merupakan tampilan halaman manajemen pengguna yang ditunjukkan pada Gambar 5.6.

Gambar 5.6 Halaman Manajemen Pengguna

Setelah itu dilakukan pembuatan fitur ubah akses pengguna. Fitur ubah akses pengguna dapat diakses pada menu manajemen pengguna. admin dapat mengatur akses data program studi yang dapat diakses oleh setiap pengguna dengan memilih tombol 'Ubah Akses'. Setelah itu sistem akan menampilkan halaman yang menampilkan daftar program studi yang dapat diakses oleh pengguna seperti yang ditunjukkan pada Gambar 5.7.

Gambar 5.7 Halaman Akses Pengguna

Untuk menambahkan akses program studi pada pengguna, admin dapat mencentang *check box* program studi yang ingin diakses oleh pengguna ataupun sebaliknya.

b. *Function JavaScript* manajemen pengguna

Setiap *input check box* program studi memiliki *value user id* dan id prodi. *Function JavaScript* akan

membaca *event* klik dari *check box*, sehingga setelah *check box* diklik akan mengirimkan *value user id* dan id prodi. Potongan kode *JavaScript* untuk manajemen pengguna ditunjukkan pada Kode 5.13.

```
   $('.form-check-
input').change(function() {

   var userId = $(this).data('user');

   var prodiId = $(this).data('prodi');

   $.ajax({

    type: 'post',

    url: "<?=
base_url('user/changeaccess'); ?>",

    data: {

     userId: userId,

     prodiId: prodiId,

    }

   });

  });
```

Kode 5.13 *Function JavaScript* Akses Pengguna

Setelah menerima *value user id* dan id prodi dari *check box*, kemudian *value* akan dikirim oleh *AJAX* ke *controller function changeAccess*.

c. *Controller function changeAcces* manajemen pengguna

Controller function changeAccess berguna untuk menambahkan pengaturan akses data program studi dari pengguna. Potongan kode *controller function changeAcces* ditunjukkan pada Kode 5.14.

```
public function changeAccess(){
   $user_id = $this->request-
>getVar('userId');
   $prodi_id = $this->request-
>getVar('prodiId');
   $prodiAccessModel = new
ProdiAccessModel();
   $id = $prodiAccessModel-
>getId($user_id, $prodi_id);
   $data = [
     'user_id' => $user_id,
     'id_prodi' => $prodi_id
   ];

     $result = $prodiAccessModel-
     >getNumRow($user_id, $prodi_id);

    if ($result < 1) {
      $prodiAccessModel->insert($data);
    } else {
      $prodiAccessModel->delete(['id' =>
$id->id]);
    }

}
```

Kode 5.14 *Controller Function changeAccess*

Setelah *controller function changeAccess* menerima *value user id* dan id prodi dari *AJAX JavaScript*, selanjutnya *value* tersebut akan dijadikan argumen untuk mengambil data dari *prodiAccessModel*. Dari *prodiAccessModel* akan didapatkan data *user id* dengan id prodi yang dapat diakses oleh *user* tersebut. Setelah itu didalam *controller function changeAccess* akan dicek apakah sudah terdapat *user id* dan id prodi pada data. Jika *user id* dan id prodi sudah tersedia maka data akan dihapus, namun jika belum tersedia ada maka *value user id* dan id prodi akan dimasukkan ke *database*.

d. *Model* Akses Prodi Pengguna

Model prodiAccessModel bergungsi untuk menghubungkan aplikasi dengan tabel yang berisi pengaturan akses pengguna. Tabel tersebut berisi pasangan *user id* dan id prodi yang dapat diakses pengguna. Data yang didapatkan dari *prodiAccessModel* akan digunakan sebagai kondisi untuk pengecekan apakah *user id* dan id prodi sudah tersedia didalam tabel akses prodi. Selain itu model ini juga digunakan untuk memasukkan *user id* dan id prodi ke dalam tabel. Contoh potongan kode *prodiAccessModel* ditunjukkan pada Kode 5.15.

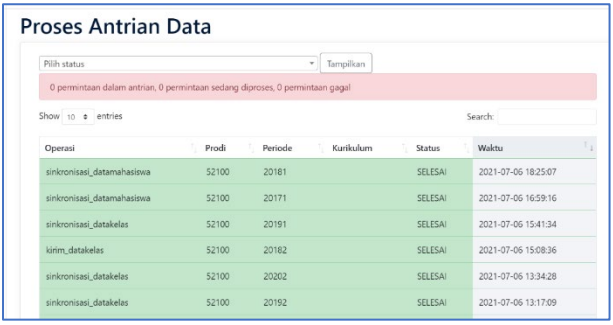
```
$id = $prodiAccessModel->getId($user_id,
$prodi_id);

$result = $prodiAccessModel-
>getNumRow($user_id, $prodi_id)
```

Kode 5.15 Potongan Kode *Model* Akses Prodi

5.1.7. Iterasi ketujuh

Pada iterasi ketujuh dilakukan pembuatan tampilan halaman proses data antrian. Halaman proses data antrian digunakan untuk menampilkan status permintaan sinkronisasi atau kirim data yang terdapat pada tabel antrian. Tampilan proses antrian data ditunjukkan pada Gambar 5.8.



Gambar 5.8 Halaman Proses Data Antrian

Selanjutnya dilakukan pembuatan tampilan halaman *error* pengiriman data. Halaman *error*

pengiriman data digunakan untuk menampilkan pesan *error* pada saat terjadi kegagalan pengiriman data. Halaman *error* pengiriman data ditunjukkan pada Gambar 5.9.

Gambar 5.9 Halaman *Error* Pengiriman Data

5.2. Pengembangan *Dashboard*

Pada tahap pengembangan *dashboard* dilakukan implementasi rencana *dashboard* yang telah dibuat hingga menjadi prototype *dashboard* yang siap dilakukan pengujian. Tahap ini diawali dengan konfigurasi teknologi yang digunakan dan dilanjutkan dengan implementasi rancangan *dashboard*.

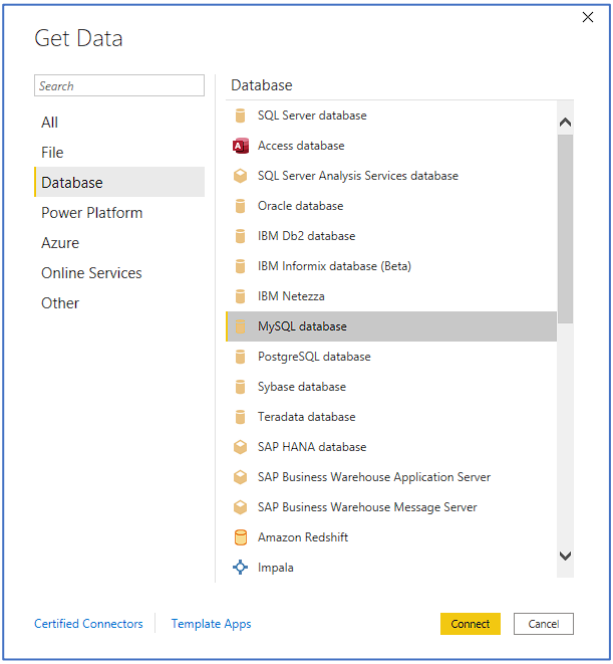
5.2.1. Konfigurasi awal *Power BI*

Konfigurasi *Power BI* dilakukan untuk menyesuaikan ketersediaan data dengan teknologi yang digunakan, yaitu *Microsoft Power BI*. Berikut tahapan yang dilakukan saat konfigurasi.

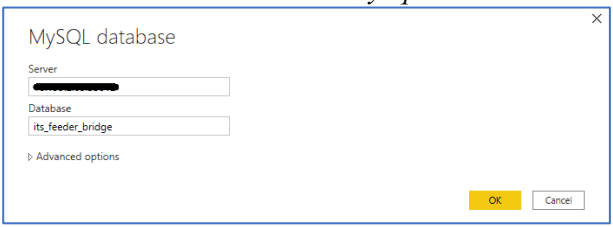
a. Memuat data

Pada tahap ini penulis memuat data dari *data warehouse* ke *Microsoft Power BI* untuk selanjutnya dilakukan konfigurasi lebih lanjut. Data akm, data kelas, data komplain, data ketepatan, dan data lain yang berkaitan yang berasal dari *data warehouse* akan digunakan dan dimuat dalam *Power BI*. Data dimuat dengan menggunakan fitur get data pada *Power BI* dan memilih opsi *Database* kemudian *MySQL Database* seperti ditunjukkan pada Gambar 5.10. Selanjutnya memasukkan alamat sumber data seperti pada Gambar 5.11, kemudian memilih

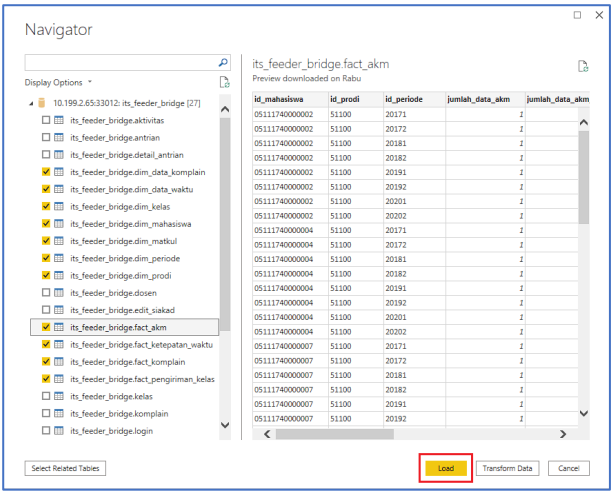
sumber data yang akan dimuat seperti pada Gambar 5.12.



Gambar 5.10 *Get Data MySql Database*



Gambar 5.11 Pengaturan Alamat Sumber Data



Gambar 5.12 Daftar Tabel Sumber Data

Seperti pada Gambar 5.12, data yang dipilih untuk dimuat adalah data dari tabel fact_akm, fact_pengiriman_kelas, fact_komplain, fact_ketepatan_waktu, dim_mahasiswa, dim_kelas, dim_matkul, dim_data_komplain, dim_data_waktu, dim_prodi, dan dim_periode. Data yang dimuat sudah siap pakai karena telah dilakukan proses *ETL* pada saat data masuk ke *data warehouse*.

b. Membuat perhitungan *dashboard*

Pada tahap ini, penulis membuat perhitungan tambahan dengan di *Microsoft Power BI*. Adapun perhitungan yang dibuat adalah persentase perngiriman akm, persentase pengiriman kelas, persentase penyelesaian komplain, dan persentase ketepatan waktu.

1. Persentase pengiriman AKM

   Persentase pengiriman akm diperoleh dengan membagi jumlah data akm terkirim dengan jumlah data akm keseluruhan dan hasilnya dibuat dalam bentuk persentase. Kode 5.16 merupakan kueri yang digunakan untuk membuat perhitungan tersebut.

```
persentase_akm =
(SUM('its_feeder_bridge
fact_akm'[jumlah_data_akm_pddikti])/S
UM('its_feeder_bridge
fact_akm'[jumlah_data_akm]))
```

Kode 5.16 Kueri Persentase Pengiriman AKM

2. Persentase pengiriman kelas

Persentase pengiriman kelas diperoleh dengan membagi jumlah data kelas terkirim dengan jumlah data kelas keseluruhan dan hasilnya dibuat dalam bentuk persentase. Kode 5.17 *Kueri Persentase Pengiriman Kelas* merupakan kueri yang digunakna untuk membuat perhitungan tersebut.

```
persentase_ketepatan_waktu =
SUM('its_feeder_bridge
fact_ketepatan_waktu'[jumlah_terkirim
_tepat_waktu])/SUM('its_feeder_bridge
fact_ketepatan_waktu'[jumlah_data_ter
kirim])
```

Kode 5.17 Kueri Persentase Pengiriman Kelas

3. Persentase penyelesaian komplain

Persentase penyelesaian komplain diperoleh dengan membagi jumlah komplain selesai dengan jumlah komplain keseluruhan dan hasilnya dibuat dalam bentuk persentase. Kode 5.18 merupakan kueri yang digunakan untuk membuat perhitungan tersebut

```
persentase_komplain_selesai =
SUM('its_feeder_bridge
fact_komplain'[jumlah_komplain_selesa
i])/SUM('its_feeder_bridge
fact_komplain'[jumlah_komplain])
```

Kode 5.18 Kueri Persentase Penyelesaian Komplain

4. Persentase ketepatan waktu

Persentase ketepatan waktu diperoleh dengan membagi jumlah data terkirim tepat waktu dengan jumlah data terkirim dan hasilnya dibuat dalam bentuk persentase. Kode 5.19 merupakan kueri yang digunakan untuk membuat perhitungan tersebut.

```
persentase_pengiriman_kelas =
(SUM('its_feeder_bridge
fact_pengiriman_kelas'[jumlah_data_ke
las_pddikti])/SUM('its_feeder_bridge
fact_pengiriman_kelas'[jumlah_data_ke
las]))
```

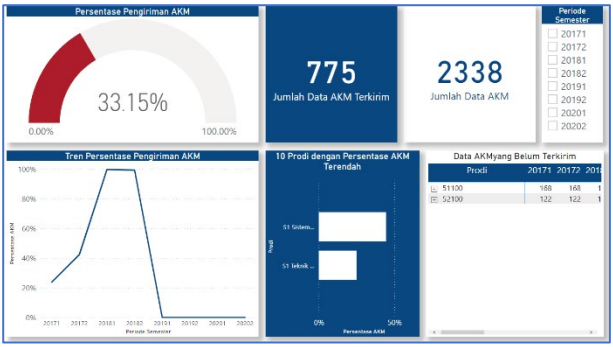
Kode 5.19 Kueri Persentase Ketepatan Waktu

5.2.2. Implementasi rancangan *dashboard*

Tahap ini dilakukan untuk mewujudkan rancangan *dashboard* yang telah dibuat menjadi prototipe *dashboard* yang siap uji dan digunakan oleh pengguna. Tahap ini dilakukan dengan melibatkan pihak DPTSI ITS sebagai pemberi feedback pada tahap evaluasi sebelum dilakukan pengujian.

a. Iterasi pertama

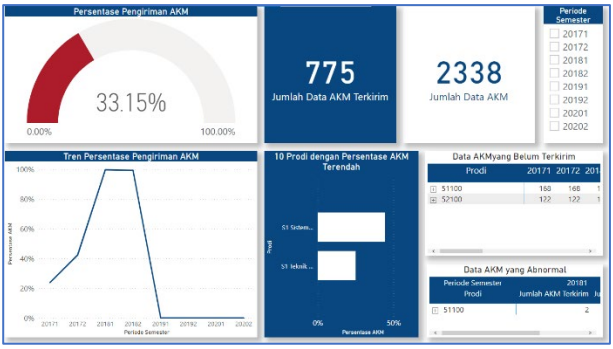
Pada iterasi pertama setelah konfigurasi *Power BI*, penulis membuat halaman *dashboard* yang dapat menampilkan visualisasi yang diperlukan pada rancangan tata letak *dashboard*. Pada iterasi ini, telah dibuat halaman pengiriman akm seperti yang ditunjukkan pada Gambar 5.13.



Gambar 5.13 Halaman Pengiriman AKM Iterasi Pertama

b. Iterasi kedua

Dari iterasi pertama, penulis mendapat umpan balik dari pihak DPTSI bahwa sebaiknya *dashboard* dapat menampilkan data yang tidak sesuai bila memang ada. Oleh karena itu, penulis tambahkan visual untuk hal tersebut seperti yang ditampilkan pada Gambar 5.14 untuk halaman pengiriman akm. Pada iterasi ini juga penulis membuat halaman pengiriman kelas seperti yang ditunjukkan pada Gambar 5.15.

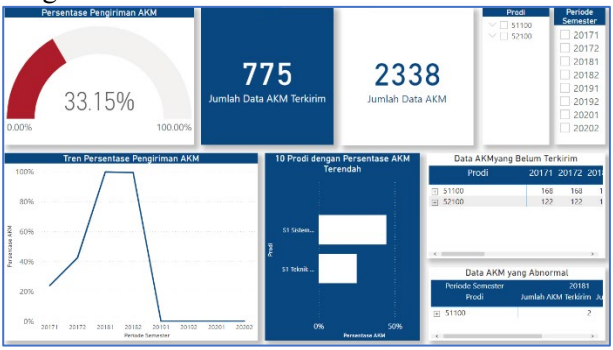


Gambar 5.14 Halaman Pengiriman AKM Iterasi Kedua

Gambar 5.15 Halaman Pengiriman Kelas

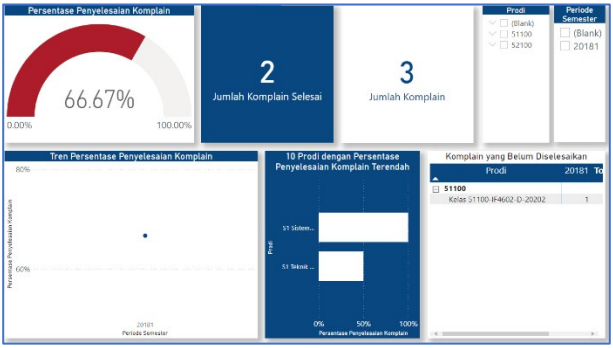
c. Iterasi ketiga

Dari iterasi 2, pihak DPTSI memberikan masukan untuk menambahkan *filter* berdasarkan prodi untuk setiap halaman *dashboard*, bukan hanya *filter* periode semester. Oleh karena itu penulis menambahkan sesuai masukan dari pihak DPTSI seperti pada Gambar 5.16 untuk halaman pengiriman akm dan Gambar 5.17 untuk halaman pengiriman kelas. Selain itu, pada iterasi ini penulis membuat halaman penyelesaian komplain dan ketepatan waktu yang masing-masing ditunjukkan dengan Gambar 5.18 dan Gambar 5.19.

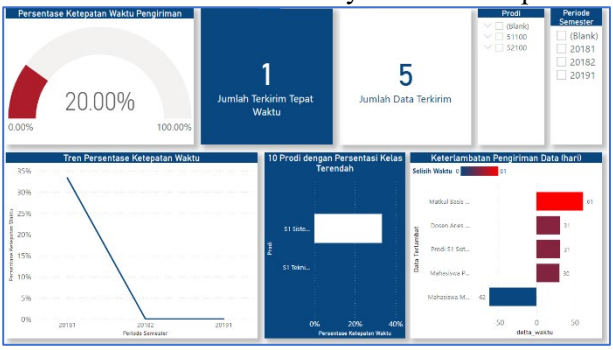


Gambar 5.16 Halaman Pengiriman AKM Iterasi Ketiga

Gambar 5.17 Halaman Pengiriman Kelas Iterasi Ketiga



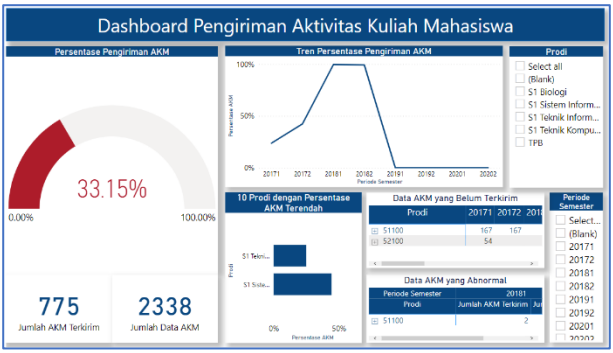
Gambar 5.18 Halaman Penyelesaian Komplain



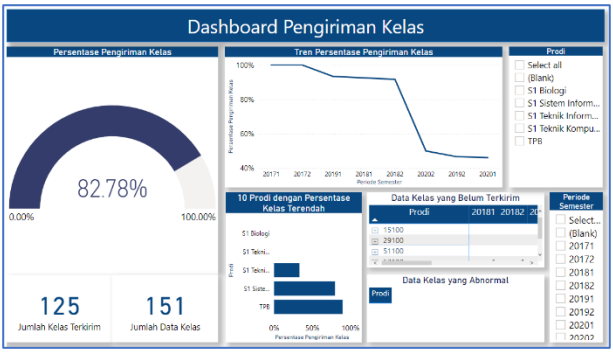
Gambar 5.19 Halaman Ketepatan Waktu

d. Iterasi keempat

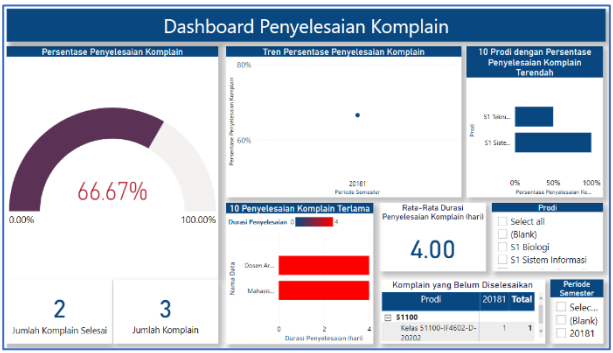
Dari iterasi 3, penulis mendapat evaluasi dari pihak DPTSI. Tampilan *dashboard* yang dibuat masih agak membingungkan terutama halaman ketepatan waktu. Diharapkan halaman ketepatan waktu sebaiknya menampilkan data yang terlambat saja sesuai dengan tujuan halaman *dashboard* ketepatan waktu, yaitu meningkatkan persentase ketepatan waktu sehingga perlu berfokus untuk memperbaiki pengiriman data yang masih terlambat. Oleh karena itu penulis menambahkan sesuai tanggapan yang diberikan sekaligus merubah tata letak agar lebih jelas fokus utama di masing-masing halaman *dashboard*. Berikut menunjukkan halaman pengiriman akm, halaman pengiriman kelas, halaman penyelesaian komplain, dan halaman ketepatan waktu yang masing-masing ditunjukkan dengan Gambar 5.20, Gambar 5.21, Gambar 5.22, dan Gambar 5.23.



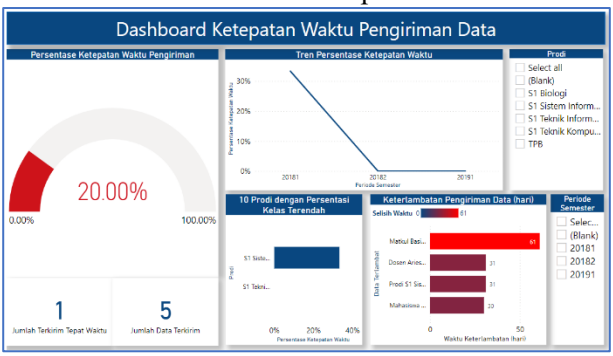
Gambar 5.20 Halaman Pengiriman AKM Iterasi Keempat



Gambar 5.21 Halaman Pengiriman Kelas Iterasi Keempat



Gambar 5.22 Halaman Penyelesaian Komplain Iterasi Keempat



Gambar 5.23 Halaman Ketepatan Waktu Iterasi Keempat

e. Iterasi kelima

Pada iterasi keempat, penulis mendapat evaluasi dari pihak DPTSI. Adapun evaluasi yang diberikan oleh pihak DPTSI ada tiga, yaitu urutan dari pilihan *filter* sebaiknya diurutkan berdasarkan kode prodi dan yang ditampilkan adalah nama prodi untuk *filter* prodi dan periode semester terbaru untuk *filter* periode semester. Evaluasi kedua terkait *filter* periode paling baru yang sebaiknya langsung diterapkan ketika *dashboard* pertama kali dimuat. Kedua evaluasi tersebut diberikan dengan alasan kemudahan pengguna dalam mengoperasikan *dashboard*. Untuk mengakomodasi evaluasi pertama, penulis membuat kolom baru dengan baris kode seperti yang ditunjukkan oleh Kode 5.20.

```
max = MAX('its_feeder_bridge
dim_periode'[id_periode])
```

Kode 5.20 Mencari Periode Terbaru di dim_periode

Baris kode ini digunakan untuk membuat kolom baru yang menemukan periode paling akhir di antara semua periode. Digunakan untuk halaman *dashboard* pengiriman akm dan pengiriman kelas. Begitu pula dengan periode untuk halaman *dashboard* penyelesaian komplain dan ketepatan waktu seperti yang ditunjukkan dengan Kode 5.21dan Kode 5.22.

```
max = MAX('its_feeder_bridge
fact_komplain'[id_periode])
```

Kode 5.21 Mencari Periode Terbaru di fact_komplain

```
max = MAX('its_feeder_bridge
fact_ketepatan_waktu'[id_periode])
```

Kode 5.22 Mencari Periode Terbaru di fact_ketepatan_waktu

Setelah membuat kolom untuk mencari periode maksimal, selanjutnya membuat kolom baru untuk *filter* yang memuat periode terbaru tersebut seperti yang ditunjukkan dengan *Kode 5.23*, *Kode 5.24*, *Kode 5.25*.

```
periode_slicer = IF('its_feeder_bridge
dim_periode'[id_periode]='its_feeder_bridg
e dim_periode'[max], CONCATENATE("Terbaru
","its_feeder_bridge dim_periode'[max]),
'its_feeder_bridge
dim_periode'[id_periode])
```

Kode 5.23 Membuat *Filter* Periode Baru di dim_periode

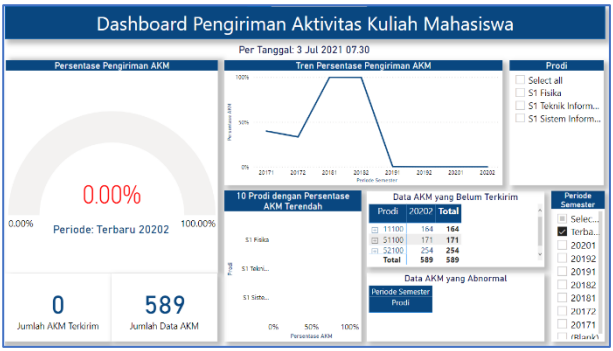
```
periode_slicer = IF('its_feeder_bridge
fact_komplain'[id_periode]='its_feeder_bri
dge fact_komplain'[max],
CONCATENATE("Terbaru ","its_feeder_bridge
fact_komplain'[max]), 'its_feeder_bridge
fact_komplain'[id_periode])
```

Kode 5.24 Membuat *Filter* Periode Baru di fact_komplain

```
periode_slicer = IF('its_feeder_bridge
fact_ketepatan_waktu'[id_periode]='its_fee
der_bridge fact_ketepatan_waktu'[max],
CONCATENATE("Terbaru ",'its_feeder_bridge
fact_ketepatan_waktu'[max]),
'its_feeder_bridge
fact_ketepatan_waktu'[id_periode])
```

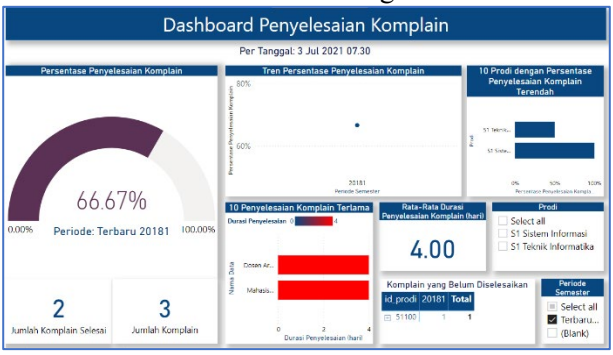
Kode 5.25 Membuat *Filter* Periode Baru di fact_ketepatan_waktu

Evaluasi ketiga yang diberikan terkait dengan halaman *dashboard* ketepatan waktu. Dikatakan bahwa dari halaman *dashboard* saat ini, pengguna masih belum dapat membedakan ketepatan waktu pengiriman berdasarkan gelombang pelaporan PDDikti. Untuk memenuhi kebutuhan tersebut, penulis perlu menambahkan sumber data baru, yaitu data jadwal pengiriman, data waktu terakhir kali proses *ETL* dilakukan, dan data jenis pengiriman beserta gelombang pelaporannya. Selanjutnya penulis merombak *data warehouse* dengan menambahkan tabel dimensi gelombang yang terkait dengan tabel fakta ketepatan waktu. Penulis kemudian merevisi tampilan *dashboard* untuk halaman pengiriman akm, pengiriman kelas, dan penyelesaian komplain yang masing-masing ditunjukkan dengan Gambar 5.24, Gambar 5.25, dan Gambar 5.26.



Gambar 5.24 Halaman Pengiriman AKM

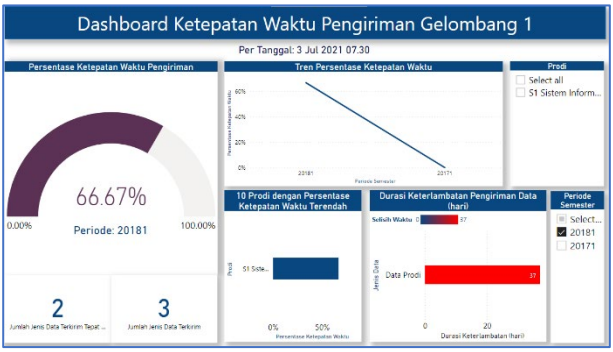
Gambar 5.25 Halaman Pengiriman Kelas



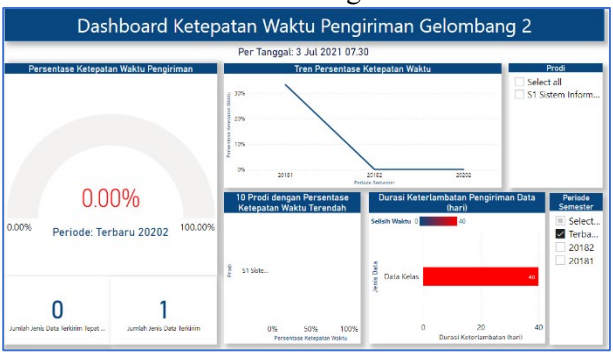
Gambar 5.26 Halaman Penyelesaian Komplain

Halaman *dashboard* ketepatan waktu dibagi menjadi beberapa halaman sesuai dengan gelombang pelaporannya. Dalam hal ini penulis membagi ke dalam tiga halaman gelombang

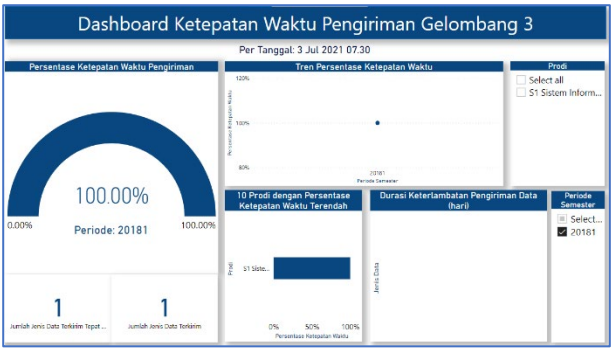
pertama, kedua, dan ketiga yang masing-masing ditunjukkan dengan Gambar 5.27, Gambar 5.28, dan Gambar 5.29.



Gambar 5.27 Halaman Ketepatan Waktu Gelombang 1



Gambar 5.28 Halaman Ketepatan Waktu Gelombang 2



Gambar 5.29 Halaman Ketepatan Waktu Gelombang 3

BAB VI
PENGUJIAN DAN PEMBAHASAN

Pada tahap ini penulis melakukan pengujian prototipe aplikasi dan *dashboard* yang telah dibuat berdasarkan skenario pengujian. Pengujian dilakukan kepada peserta magang DPTSI yang berkaitan dengan pelaporan PDDikti. Adapun peserta magang yang dimaksud adalah Gita Krismurti Romadhani sebagai partisipan 1 dan Reynata Tri Damayanti sebagai partisipan 2. Data dari partisipan ditunjukkan pada Tabel 6.1 dan Tabel 6.2.

Tabel 6.1 Data Partisipan 1

Nama	Gita Krismurti Romadhani
Umur	22 tahun
Pekerjaan	Mahasiswa S1 Sistem Informasi ITS
Peran	Tenaga Magang DPTSI
Tugas	-Mengoperasikan aplikasi *ProFeeder* -Melakukan pengiriman data ke PDDikti menggunakan *ProFeeder* -Melakukan pengawasan terhadap aktivitas pengiriman data yang dilakukan admin prodi
Keterangan	Sekitar 1,5 tahun menjadi tenaga magang DPTSI dan menggunakan aplikasi *ProFeeder*

Tabel 6.2 Data Partisipan 2

Nama	Reynata Tri Damayanti
Umur	22 tahun
Pekerjaan	Mahasiswa S1 Sistem Informasi ITS
Peran	Tenaga Magang DPTSI
Tugas	-Mengoperasikan aplikasi *ProFeeder* -Melakukan pengiriman data ke PDDikti menggunakan *ProFeeder* -Melakukan pengawasan terhadap aktivitas pengiriman data yang dilakukan admin prodi
Keterangan	Sekitar 1 tahun menjadi tenaga magang DPTSI dan menggunakan aplikasi *ProFeeder*

Hasil pengujian digunakan sebagai pertimbangan untuk perbaikan aplikasi dan *dashboard* ke depannya. Berikut hasil uraian hasil pengujian yang telah penulis lakukan.

6.1. Hasil *Usability Testing* Aplikasi

Bagian ini menjelaskan hasil pengujian aplikasi pada setiap bagian data, cara mengatur *user*, dan memantau status antrian.

6.1.1. Data mahasiswa

Tabel 6.3 Hasil Pengujian Skenario Data Mahasiswa

Skenario	Aktivitas	Partisipan	
		1	2
SMHS01	TMHS01	3	3
	TMHS02	3	3
	TMHS03	3	3
SMHS02	TMHS04	3	3
SMHS03	TMHS05	3	3
SMHS04	TMHS06	3	3

Hasil pengujian seperti pada Tabel 6.3 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{10 + (0 \times 0.5)}{10}$$

$$Success\ rate = 100\%$$

Berdasarkan perhitungan tersebut, skenario data mahasiswa dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 100%.

6.1.2. Data kelas

Tabel 6.4 Hasil Pengujian Skenario Data Kelas

Skenario	Aktivitas	Partisipan	
		1	2
SKLS01	TKLS01	3	3
	TKLS02	3	3
	TKLS03	3	3
SKLS02	TKLS04	3	3
SKLS03	TKLS05	3	3
SKLS04	TKLS06	3	3
	TKLS07	3	3
	TKLS08	3	3
	TKLS01	3	3

Hasil pengujian seperti pada Tabel 6.4 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{18 + (0 \times 0.5)}{18}$$

$$Success\ rate = 100\%$$

Berdasarkan perhitungan tersebut, skenario data kelas dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 100%.

6.1.3. Data dosen

Tabel 6.5 Hasil Pengujian Skenario Data Dosen

Skenario	Aktivitas	Partisipan	
		1	2
SDSN01	TDSN01	3	3
	TDSN02	3	3
SDSN02	TDSN03	3	3

SDSN03	TDSN04	3	3
SDSN05	TDSN05	3	3

Hasil pengujian seperti pada Tabel 6.5 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{10 + (0 \times 0.5)}{10}$$

$$Success\ rate = 100\%$$

Berdasarkan perhitungan tersebut, skenario data dosen dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 100%.

6.1.4. Data mata kuliah

Tabel 6.6 Hasil Pengujian Skenario Data Kelas

Skenario	Aktivitas	Partisipan	
		1	2
SMKL01	TMKL01	3	3
	TMKL02	3	3
	TMKL03	3	3
SMKL02	TMKL04	3	3
SMKL03	TMKL05	2	3
SMKL04	TMKL06	2	3

Hasil pengujian seperti pada Tabel 6.6 tersebut kemudian dimasukkan ke dalam rumus *success rate* dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{10 + (2 \times 0.5)}{12}$$

$$Success\ rate = 92\%$$

Berdasarkan perhitungan tersebut, skenario data mata kuliah dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 92%.

6.1.5. Data program studi

Tabel 6.7 Hasil Pengujian Skenario Data Prodi

Skenario	Aktivitas	Partisipan	
		1	2

SPRD01	TPRD01	3	3
	TPRD02	3	3
	TPRD03	3	3
SPRD02	TPRD04	3	3
SPRD03	TPRD05	3	3
SPRD04	TPRD06	3	3

Hasil pengujian seperti pada Tabel 6.7 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{12 + (0 \times 0.5)}{12}$$

$$Success\ rate = 100\%$$

Berdasarkan perhitungan tersebut, skenario data program studi dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 100%.

6.1.6. Data aktivitas

Tabel 6.8 Hasil Pengujian Skenario Data Aktivitas

Skenario	Aktivitas	Partisipan	
		1	2
SAKT01	TAKT01	3	3
	TAKT02	2	3
	TAKT03	3	3
SAKT02	TAKT04	3	3
SAKT03	TAKT05	3	3
SAKT04	TPRD06	2	3

Hasil pengujian seperti pada Tabel 6.8 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{10 + (2 \times 0.5)}{12}$$

$$Success\ rate = 92\%$$

Berdasarkan perhitungan tersebut, skenario data aktivitas dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 92%.

6.1.7. Manajemen pengguna

Tabel 6.9 Hasil Pengujian Skenario Manajemen Pengguna

Skenario	Aktivitas	Partisipan	
		1	2
SADM01	TADM01	3	3
SADM02	TADM02	3	3
	TADM03	3	3
	TADM04	3	3
SADM03	TADM05	3	3
SADM04	TADM06	3	3
SADM05	TADM07	3	3
SADM06	TADM08	3	3

Hasil pengujian seperti pada Tabel 6.9 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{16 + (0 \times 0.5)}{16}$$

$$Success\ rate = 100\%$$

Berdasarkan perhitungan tersebut, skenario mengatur *user* dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 100%.

6.1.8. Status antrian

Tabel 6.10 Hasil Pengujian Skenario Status Antrian

Skenario	Aktivitas	Partisipan	
		1	2
SATR01	TATR01	3	3
SATR02	TATR01	3	3

Hasil pengujian seperti pada Tabel 6.10 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{4 + (0 \times 0.5)}{4}$$

$$Success\ rate = 100\%$$

Berdasarkan perhitungan tersebut, skenario melihat status data antrian dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 100%.

6.2. Hasil *Usability Testing Dashboard*

Bagian ini menjelaskan hasil pengujian prototipe *dashboard* untuk masing-masing halaman *dashboard*.

6.2.1. Halaman pengiriman AKM

Tabel 6.11 Hasil Pengujian Skenario Halaman Pengiriman AKM

Skenario	Aktivitas	Partisipan	
		1	2
SPA01	APA01	3	3
	APA02	2	3
SPA02	APA03	2	3
SPA03	APA04	2	3
SPA04	APA05	3	2
	APA06	3	2
SPA05	APA07	2	2
SPA06	APA08	2	2

Hasil pengujian seperti pada Tabel 6.11 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{7 + (9 \times 0.5)}{16}$$

$$Success\ rate = 72\%$$

Berdasarkan perhitungan tersebut, halaman *dashboard* pengiriman akm dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 72%.

6.2.2. Halaman pengiriman kelas

Tabel 6.12 Hasil Pengujian Skenario Halaman Pengiriman Kelas

Skenario	Aktivitas	Partisipan	
		1	2
SPK01	APK01	3	3
	APK02	3	3
SPK02	APK03	3	3
SPK03	APK04	3	3
SPK04	APK05	3	3
	APK06	3	3
SPK05	APK07	2	2
SPK06	APK08	3	3

Hasil pengujian seperti pada Tabel 6.12 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{14 + (2 \times 0.5)}{16}$$

$$Success\ rate = 94\%$$

Berdasarkan perhitungan tersebut, halaman *dashboard* pengiriman kelas dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 94%.

6.2.3. Halaman penyelesaian komplain

Tabel 6.13 Hasil Pengujian Skenario Halaman Penyelesaian Komplain

Skenario	Aktivitas	Partisipan	
		1	2
SKD01	AKD01	3	3
	AKD02	3	3
SKD02	AKD03	3	3
SKD03	AKD04	3	3
SKD04	AKD05	3	3

	AKD06	3	3
SKD05	AKD07	2	2
SKD06	AKD08	2	3

Hasil pengujian seperti pada Tabel 6.13 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{13 + (3 \times 0.5)}{16}$$

$$Success\ rate = 91\%$$

Berdasarkan perhitungan tersebut, halaman *dashboard* penyelesaian komplain dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 91%.

6.2.4. Halaman ketepatan waktu

Tabel 6.14 Hasil Pengujian Skenario Halaman Ketepatan Waktu

Skenario	Aktivitas	Partisipan	
		1	2
SKW01	AKW01	3	3
	AKW02	3	3
SKW02	AKW03	3	3
SKW03	AKW04	3	3
SKW04	AKW05	3	3
	AKW06	3	3
SKW05	AKW07	2	2

Hasil pengujian seperti pada Tabel 6.14 tersebut kemudian dimasukkan ke dalam rumus success rate dan akan mendapatkan hasil sebagai berikut:

$$Success\ rate(\%) = \frac{12 + (2 \times 0.5)}{14}$$

$$Success\ rate = 93\%$$

Berdasarkan perhitungan tersebut, halaman *dashboard* penyelesaian komplain dapat diselesaikan menggunakan skenario pengujian dengan tingkat keberhasilan 93%.

6.3. Hasil *Open-Ended Question*s

Bagian ini berisi hasil temuan yang penulis dapatkan setelah mengajukan beberapa pertanyaan yang telah disiapkan sebelumnya kepada partisipan 1 dan partisipan 2. Hasil temuan tersebut kemudian dikelompokkan menjadi dua kategori yaitu pengalaman pengguna dan fitur. Berikut rincian hasil temuan dari masing-masing partisipan:

6.3.1. Partisipan 1

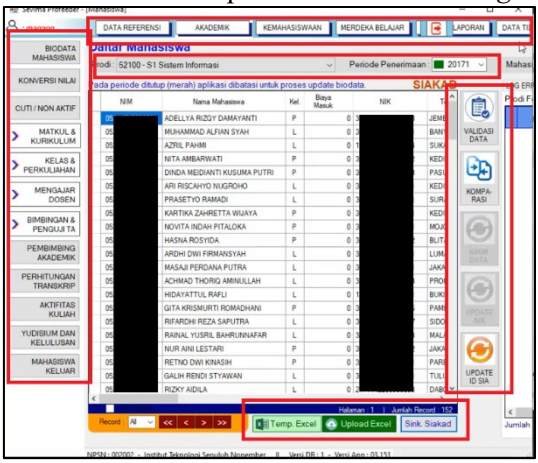
Berikut hasil temuan dari partisipan satu untuk masing-masing kategori:

a. Pengalaman pengguna

Kategori penglaman pengguna memuat hasil temuan dari partisipan 1 yang berkaitan dengan tampilan, kemudahan, kenyamanan, dan kecepatan pemahaman pengguna pada saat menggunakan web *Feeder Bridge*.

1. Sebelum mencoba sistem yang telah penulis buat, partisipan memiliki ekspektasi bahwa *Feeder Bridge* ini adalah replika dari *ProFeeder*, baik itu secara tampilan maupun pengalaman pengguna sehingga pengguna tidak perlu belajar lagi cara mengoperasikan *Feeder Bridge.*
2. Tampilan *Feeder Bridge* berbeda dengan *ProFeeder*, terutama dalam tata letak dan pengelompokkan menu. Perbedaan yang dimaksud ditunjukkan dengan Gambar 6.1 yang menunjukkan tampilan *Feeder Bridge* dan Gambar 6.2 yang menunjukkan tampilan *ProFeeder*. Bagian tombol dan menu diperjelas dengan kotak merah.

Gambar 6.1 Tampilan Web *Feeder Bridge*

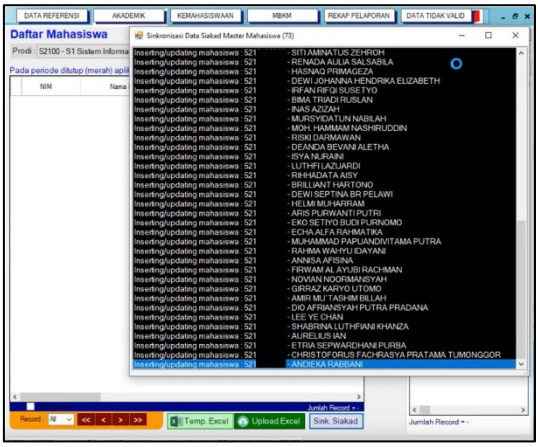


Gambar 6.2 Tampilan Aplikasi *ProFeeder*

3. Pada saat belajar *ProFeeder* dulu cukup rumit karena memerlukan banyak persiapan, misalnya harus menonton video tutorial satu per satu. Lalu dengan *Feeder Bridge* yang tampilan dan pengelompokkan menu yang cukup berbeda, harus belajar lagi. Bagi sebagian orang, hal ini mungkin cukup rumit juga dan mungkin malah menjadi beban.
4. Untuk mempelajari *Feeder Bridge* dirasa cukup mudah karena dari segi konten mirip dengan *ProFeeder*, mungkin membutuhkan waktu dua sampai tiga hari untuk memahami alur sinkronisasi dan kirim data. Jika belum pernah mengguanakan *ProFeeder* mungkin

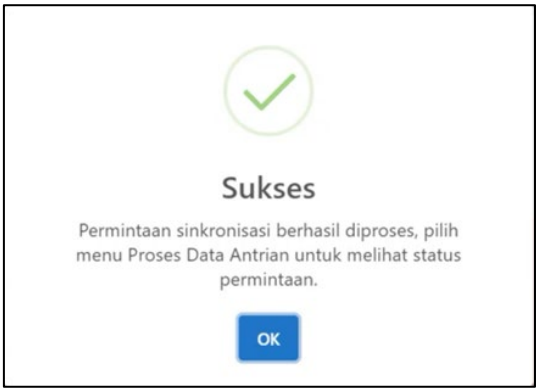
akan bingung apakah harus sinkronisasi dulu, kirim dulu, atau sama saja.

5. Dari sisi bahasa dan istilah yang digunakan aplikasi, partisipan sudah cukup familiar dan dirasa akan mudah dimengerti juga oleh pengguna *ProFeeder* lainnya.
6. Untuk mempelajari *Feeder Bridge* tidak bisa lebih cepat karena selain tampilan yang berbeda, pengelompokkan menu juga berbeda seperti yang ditunjukkan pada Gambar 6.1 dan Gambar 6.2 sebelumnya. Di *ProFeeder* ada dua menu tetapi di sini dijadikan satu menu, itu yang membutuhkan waktu lebih banyak untuk membiasakan diri.
7. Untuk fitur sinkronisasi, ada kelebihan dan kekurangannya dibandingkan dengan *ProFeeder*. Sinkronisasi dari *Feeder Bridge* tidak perlu menunggu hingga proses selesai untuk dapat melakukan proses yang lainnya karena sudah masuk ke antrian, namun tidak selesai saat itu juga yang menyebabkan pengguna bingung kapan proses ini akan selesai. Partisipan lebih suka menyelesaikan satu proses terlebih dahulu baru lanjut ke proses lainnya. Jika menggunakan *Feeder Bridge*, partisipan takut mengalami kebingungan proses apa saja yang sudah dilakukan. Adapun sinkronisasi yang dimaksud oleh partisipan ditunjukkan dengan Gambar 6.3 yang menunjukkan tampilan proses sinkronisasi di *ProFeeder*.



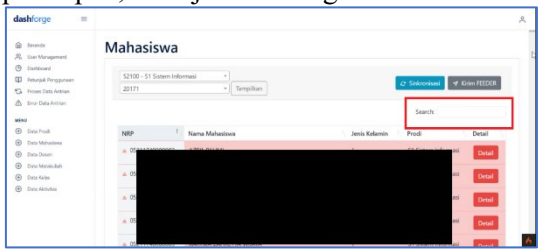
Gambar 6.3 Tampilan Proses Sinkronisasi *ProFeeder*

8. Jika ada *error* saat melakukan proses sinkronisasi, partisipan berpikir harus balik lagi ke prodi mana yang mengalami *error* tadi, sedangkan di *ProFeeder error* sudah dapat diketahui pada saat menjalankan proses tersebut dan belum berpindah ke proses lain sehingga dapat diselesaikan saat itu juga. Jika ada *error* pada saat proses sinkronisasi pada *ProFeeder*, dapat langsung dilihat pada halaman seperti yang ditunjukkan oleh Gambar 6.3 sebelumnya.
9. Saat menggunakan fitur sinkronisasi untuk pertama kali, partisipan terkejut dan senang karena berpikir proses sinkronisasi langsung selesai. Namun setelah mengetahui bahwa proses tersebut masuk ke antrian, itu adalah hal yang wajar. Yang membuat partisipan merasa demikian adalah munculnya notifikasi sukses sesaat setelah pengguna menekan tombol sinkronisasi seperti yang ditunjukkan pada Gambar 6.4.



Gambar 6.4 Notifikasi Sukses

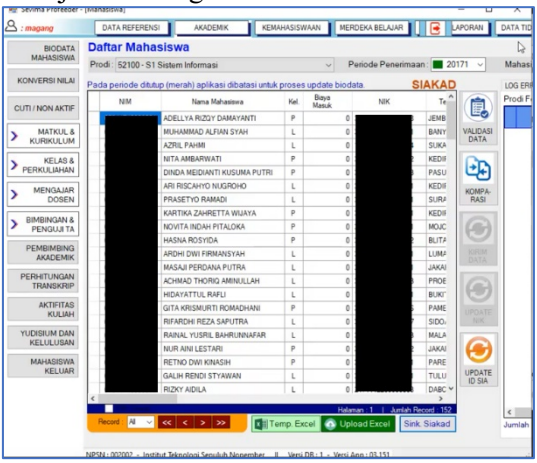
10. Dalam hal fitur sinkronisasi, partisipan lebih suka yang dari *ProFeeder* karena sesuai dengan cara kerjanya.
11. Tampilan data *Feeder Bridge* lebih *user-friendly* namun memakan banyak tempat, sehingga kesulitan ketika akan mencari data yang ada di tengah. Namun fitur *search* sangat membantu pengguna untuk menyelesaikan hal tersebut. Adapun yang tampilan yang dimaksud partisipan, ditunjukkan dengan Gambar 6.5.



Gambar 6.5 Fitur *Search Data*

12. Dalam hal tampilan data, partisipan lebih suka dengan tampilan di *ProFeeder* karena meskipun baris datanya kecil-kecil, namun lengkap sehingga dapat mengetahui data mana yang kemungkinan mengalami *error* dengan sekilas. Oleh karena itu, pengguna dapat lebih cepat melaporkan jika ada yang tidak sesuai

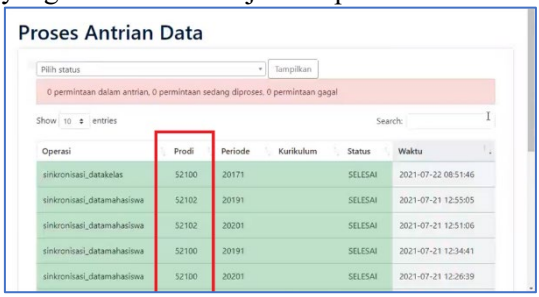
dan semakin cepat pula data tersebut dibenahi. Adapun tampilan yang dimaksud partisipan adalah tampilan data *ProFeeder* seperti yang ditunjukkan dengan Gambar 6.6.



Gambar 6.6 Tampilan Data *ProFeeder*

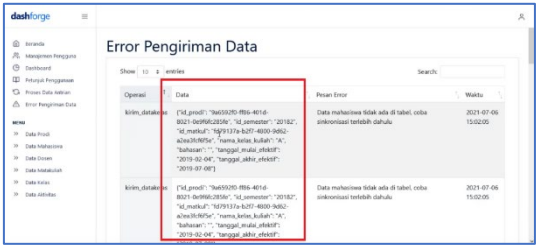
13. Untuk data yang pertama kali dilihat pada *ProFeeder* saat akan melakukan pengiriman biasanya data yang sering muncul *error* seperti NIK, ini yang sebaiknya ada di depan sehingga lebih cepat diketahui jika ada *error*. Hal ini difasililtasi oleh *ProFeeder* dengan tampilan datanya seperti yang ditunjukkan pada Gambar 6.6, sedangkan *Feeder Bridge* menyediakan tampilan yang berbeda.
14. Seperti saat menggunakan fitur sinkronisasi *Feeder Bridge*, awalnya partisipan mengalami kebingungan saat menggunakan fitur kirim karena selesai dengan cepat, namun ternyata sama dengan sinkronisasi yaitu masuk ke antrian.
15. Untuk proses antrian dianggap membingungkan karena tidak dapat melihat estimasi waktu kapan satu proses akan selesai.

16. Partisipan tidak mengalami kesulitan berarti dalam melihat antrian dan mengerti maksud dari halaman tersebut, namun sedikit bingung terkait kode prodi karena tidak semua admin hafal kode prodi. Adapun tampilan kode prodi yang dimaksud ditunjukkan pada Gambar 6.7.



Gambar 6.7 Halaman Proses Antrian *Feeder Bridge*

17. Halaman *error* pengiriman data kurang *user-friendly* terutama pada kolom 'Data' yang sulit dibaca secara sekilas dikarenakan ada UUID yang dirasa tidak perlu ditampilkan karena justru malah membuat rumit pandangan, lebih baik dibuat rapi dan sederhana. Adapun tampilan data yang membuat kesulitan pembaca tersebut ditunjukkan dengan Gambar 6.8.



Gambar 6.8 Halaman *Error* Data *Feeder Bridge*

18. *Feeder Bridge* lebih lengkap dan *user-friendly* dibandingkan dengan *ProFeeder*, namun bagi sebagian orang seperti partisipan lebih nyaman

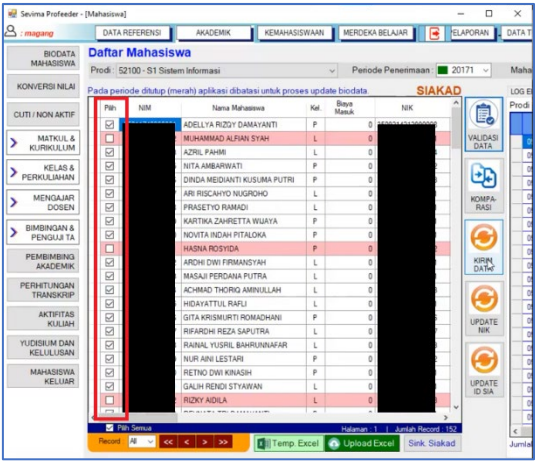
menggunakan *ProFeeder*. Jika suatu saat benar-benar beralih ke *Feeder Bridge*, sistem ini akan mudah diterima.

19. Secara keseluruhan saat ini, *Feeder Bridge* masih belum dapat dibilang memudahkan pengguna karena kelebihan yang dimiliki *Feeder Bridge* belum dapat menutupi kekurangannya.

b. Fitur

Kategori fitur ini memuat hasil temuan dari partisipan 1 yang berkaitan dengan kelengkapan fitur, baik temuan positif maupun negatif.

1. Pada fitur kirim data, partisipan tidak dapat melakukan pengiriman hanya satu data saja dan harus semua data data dikirimkan. Untuk saat ini, fitur tersebut disediakan oleh *ProFeeder* namun *Feeder Bridge* belum menyediakannya. Fitur yang dimaksud oleh partisipan ditunjukkan dengan kotak merah pada Gambar 6.9.



Gambar 6.9 Fitur Pilih Data *ProFeeder*

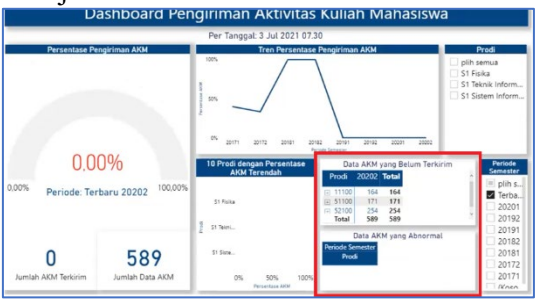
2. Partisipan memiliki ekspektasi pengiriman dapat dilakukan lebih cepat dari *ProFeeder* karena pengiriman di *ProFeeder* sangat

bergantung pada kecepatan internet. Hal ini dibuktikan dengan pengujian performa yang dilakukan pada partisipan 1 dan partisipan 2 yang menggunakan penyedia layanan internet berbeda.

3. Estimasi waktu diperlukan agar dapat memperkirakan kapan proses akan selesai agar tidak terus menerus memantau halaman proses antrian dan dapat memperkirakan kapan harus melakukan pengecekan data *error*.
4. Mengetahui kapan proses akan selesai sangat diperlukan karena biasanya jadwal pengiriman data mepet sehingga harus memperkirakan sendiri kapan harus menjalankan proses apa, belum lagi jika nanti ada data yang *error*.
5. Partisipan membutuhkan setidaknya waktu antrian dimulai dan waktu antrian selesai untuk memperkirakakan waktu proses selanjutnya, bukan hanya waktu perubahan status.
6. Halaman manajemen pengguna cukup mudah dimengerti, namun masih kurang lengkap. Jika dibandingkan dengan *ProFeeder*, belum ada yang fitur untuk mengatur batasan fitur hapus. Fitur hapus perlu dibatasi untuk menjaga data agar tidak sembarangan dihapus dan dapat diketahui siapa saja yang menghapus data. Namun, sayangnya partisipan tidak dapat menunjukkan fitur yang dimaksud karena keterbatasan hak akses, partisipan menyebutkan bahwa hanya admin DPTSI yang memiliki akses untuk fitur manajemen pengguna *ProFeeder*.
7. Fitur *dashboard* yang ada di *Feeder Bridge* cukup baik, tetapi sangat disayangkan karena akses *dashboard* ini tidak dimiliki oleh admin prodi. Pada kenyataannya, admin prodi sering menanyakan progres pengiriman dan pihak top level harus membagikan progres keseluruhan

prodi tersebut ke setiap departemen. Hal ini cukup memakan waktu, sehingga sebaiknya admin prodi juga diberikan akses ke *dashboard* untuk prodi masing-masing karena diperlukan juga untuk kepentingan lain seperti akreditasi.

8. Fitur *dashboard* cukup membantu bagi pengguna karena dari *dashboard* dapat langsung mengetahui data apa yang belum terkirim atau bermasalah sehingga dapat lebih cepat menyelesaikan masalah pengiriman dan meningkatkan persentase pengiriman. Adapun visualisasi yang dimaksud partisipan ditunjukan dalam Gambar 6.10.



Gambar 6.10 Visualisasi Data Belum Terkirim

9. Partisipan berharap dapat membuka menu petunjuk penggunaan namun ternyata masih belum tersedia.
10. Kekurangan *Feeder Bridge* tidak banyak, namun mengenai proses inti yaitu sinkronisasi dan kirim. Namun hal ini tergantung pribadi masing-masing karena partisipan lebih suka satu proses langsung selesai dan dapat mengirimkan beberapa data saja tanpa harus mengirimkan semuanya.
11. Partisipan akan lebih memilih menggunakan *Feeder Bridge* jika pada halaman proses antrian terdapat estimasi waktu kapan proses akan selesai karena masalah *ProFeeder* saat ini

selain proses yang lama juga instalasi dan konfigurasi koneksi yang rumit.

6.3.2. Partisipan 2

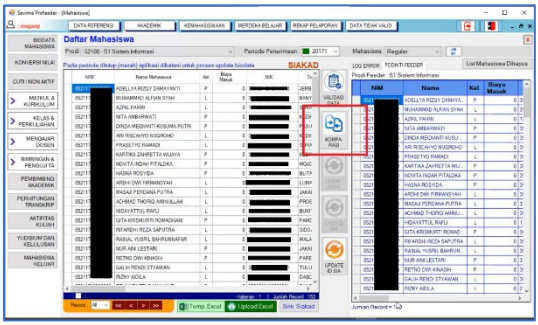
Berikut hasil temuan dari partisipan satu untuk masing-masing kategori:

a. Pengalaman pengguna

Kategori penglaman pengguna ini memuat hasil temuan dari partisipan 2 yang berkaitan dengan tamilan, kemudahan, kenyamanan, dan kecepatan pemahaman pengguna pada saat menggunakan *Feeder Bridge*.

1. Dibandingkan dengan *ProFeeder* yang memiliki banyak tombol yang jarang digunakan, tombol-tombol yang ada di *Feeder Bridge* cukup berbeda, lebih sederhana, dan jelas fungsinya. Perbedaan tombol ini sebelumnya juga disampaikan oleh partisipan 1 seperti yang ditunjukkan pada gambar 6.1 dan gambar 6.2.
2. Dari sisi admin prodi, masih tetap harus belajar lagi untuk dapat membiasakan diri dengan *Feeder Bridge* namun lebih terarah sehingga tidak memakan banyak waktu seperti saat belajar menggunakan *ProFeeder*. Selain itu istilah yang digunakan di *Feeder Bridge* juga mudah dipahami.
3. Untuk mempelajari *ProFeeder*, partisipan membutuhkan waktu dua minggu dan masih belum cukup terbiasa. Untuk proses sinkronisasi dan cukup sudah terbiasa, namun dalam hal menangani data yang *error* memakan waktu lebih lama untuk adaptasi. Sedangkan untuk mempelajari *Feeder Bridge* ini mungkin cukup memakan waktu satu minggu untuk proses sinkronisasi dan kirim, untuk melihat *error* yang membutuhkan waktu lebih lama untuk dipelajari.

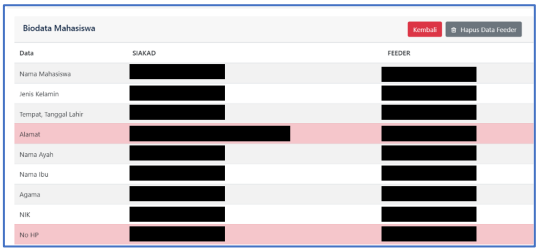
4. Ekspektasi awal pasti berpikir *Feeder Bridge* mirip dengan *ProFeeder*, namun partisipan terkejut saat tidak menemukan fitur untuk komparasi data yang ternyata harus melihat detail data untuk membandingkan data yang *error*, itupun semuanya ditampilkan dan bukan hanya data yang tidak sesuai saja. Partisipan sendiri masih belum menentukan lebih suka yang mana. Adapun fitur komparasi yang dimaksud partisipan di *ProFeeder* ditunjukkan dalam Gambar 6.11.



Gambar 6.11 Fitur Komparasi *ProFeeder*

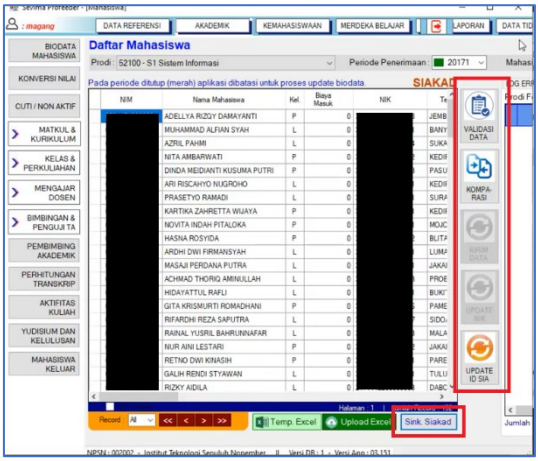
5. Saat ini partisipan masih menggunakan *ProFeeder* sehingga masih nyaman dengan *ProFeeder* karena belum terbiasa dengan *Feeder Bridge*.
6. Untuk fitur sinkronisasi, lebih suka menggunakan *Feeder Bridge* karena bisa ditinggal mengerjakan yang lain. Namun kekurangannya tidak dapat langsung mengetahui jika ada *error* di proses tersebut, harus memantau halaman proses antrian dan *error*. Partisipan mengira proses sinkronisasi akan seperti *ProFeeder* yang ketika diklik akan muncul tampilan sedang menjalankan proses, tetapi ternyata bisa dilihat ketika membuka halaman proses antrian.

7. Partisipan terkejut karena pada saat melakukan sinkron seperti tidak melakukan apa-apa dan sangat mungkin untuk klik ulang tombol sinkronisasi untuk yakin bahwa proses sinkronisasi sudah dijalankan.
8. Awalnya partisipan mengalami kebingungan karena dengan tampilan data *Feeder Bridge* saat ini yang semua baris datanya pada halaman data mahasiswa menunjukkan warna merah. Partisipan mengira bahwa itu memang warna tema yang digunakan, namun setelah itu baru menyadari bahwa yang merah itu adalah data yang tidak sesuai antara Siakad dengan Feeder PDDikti.
9. Secara tampilan data antara *ProFeeder* dengan *Feeder Bridge* cukup berbeda. *ProFeeder* menampilkan data dan semua kolomnya dalam satu baris ke kanan sehingga sangat memungkinkan untuk salah melihat data ketika scroll ke kanan. Sedangkan *Feeder Bridge* menampilkan detail data ke bawah untuk satu data tertentu sehingga perlu untuk adaptasi lagi jika ingin menggunakan *Feeder Bridge*. Partisipan sendiri lebih suka dengan tampilan ke bawah seperti ini hanya saja untuk saat ini belum terbiasa. Untuk halaman tampilan data di *ProFeeder* yang menampilkan keseluruhan kolom ditunjukkan dengan Gambar 6.6 yang sudah dibahas sebelumnya. Untuk halaman detail data di *Feeder Bridge* yang dimaksud partisipan ditunjukkan dengan Gambar 6.12.



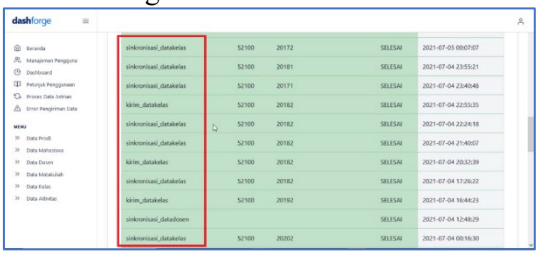
Gambar 6.12 Halaman Detail Data *Feeder Bridge*

10. Data mahasiswa lebih mudah dibaca dengan tampilan awalnya dan tampilan ke bawah untuk detail datanya, namun untuk data dengan detail yang sedikit sangat memakan waktu jika harus melihat detail data satu per satu.
11. Untuk fitur sinkronisasi dan kirim cukup mirip dan memudahkan namun yang kurang adalah tahap setelah klik tombol sinkron atau kirim yang tidak dapat langsung mengetahui *error* pada proses tersebut.
12. Dari segi langkah pengiriman, partisipan cukup terkejut dengan *Feeder Bridge* karena proses sinkronisasi dan kirim dapat dilakukan hampir bersamaan dan belum jelas urutan prosesnya sehingga bisa saja proses sinkronisasi terlewatkan dan langsung kirim. Sedangkan *ProFeeder* memiliki lebih banyak langkah yang harus dilewati mulai dari sinkronisasi, validasi, komparasi, sampai kirim data yang masing-masing harus diselesaikan sebelum beralih ke proses selanjutnya. Hal ini ditandai dengan dimatikannya tombol ketika proses sebelumnya belum dilakukan seperti yang ditunjukkan pada Gambar 6.13.



Gambar 6.13 Tampilan Menu *ProFeeder*

13. Halaman proses antrian kurang rapi dan sedikit membingungkan pengguna dalam membaca datanya, sekilas proses satu dengan lainnya terlihat sama namun ternyata berbeda. Secara keseluruhan cukup bagus, namun kekurangannya pengguna harus bolak-balik ke halaman ini. Bagian yang dianggap membingungkan pengguna dari halaman antrian ditunjukkan oleh Gambar 6.14 dan ditandai dengan kotak merah.



Gambar 6.14 Tampilan Data *Event* Antrian *Feeder Bridge*

14. Pada awalnya, partisipan tidak mengetahui di mana jika ingin mengetahui saat ada *error* dalam pengiriman hingga akhirnya membuka halaman antrian.

15. Sama halnya dengan partisipan 1, partisipan juga mengalami kebingungan saat membuka halaman *error* pengiriman. Halaman *error* pengiriman lebih susah dibaca, terutama pada kolom data yang menyertakan *UUID* sehingga sulit mengetahui data apa yang *error* dengan cepat seperti yang ditunjukkan pada gambar 6.8.
16. Semakin lama mengetahui data yang *error*, semakin lama permasalahan diselesaikan.
17. Halaman manajemen pengguna sudah cukup jelas dan tidak membuat bingung.
18. Fitur *dashboard* yang ada di *Feeder Bridge* memakan waktu yang cukup lama untuk memuat *dashboard* sepenuhnya, mungkin dikarenakan datanya cukup banyak. Namun, salah satu faktor yang menyebabkan *dashboard* lama dimuat adalah koneksi internet. Hal ini juga dibuktikan dengan pengujian performa yang dilakukan pada partisipan 1 dan partisipan 2 yang menggunakan penyedia layanan internet berbeda.
19. Saat mencoba *dashboard* pertama kali, masih mengalami kebingunan di beberapa visualisasi karena tampilannya kecil dan belum mengerti bahwa setiap visualisasi di *dashboard* bisa didetailkan atau diperbesar. Meskipun demikian, setidaknya beberapa visualisasi terutama tabel data belum terkirim dibuat lebih besar karena tidak semua pengguna mengerti bahwa visualisasi bisa didetailkan. Visualisasi yang dimaksud adalah visualisasi data yang belum dikirim dan data abnormal, visualisasi yang sama dengan yang disampaikan oleh partisipan 1. Visualisasi tersebut ditunjukkan dengan gambar 6.10.
20. Secara keseluruhan, *Feeder Bridge* lebih mudah dipakai oleh admin prodi jika

dibandingkan dengan *ProFeeder*, hanya saja masih terbiasa di *ProFeeder* dan perlu beradaptasi lagi karena cukup berbeda dari segi menu dan fitur. Untuk proses sinkronisasi dan kirim perlu diperbaiki lagi terutama untuk memperlihatkan *error*, selain itu sudah bagus.

b. Fitur

Kategori fitur ini memuat hasil temuan dari partisipan 2 yang berkaitan dengan kelengkapan fitur, baik temuan positif maupun negatif.

1. Untuk fitur kirim, di *ProFeeder* dapat dilihat langsung mana data yang gagal, cancel, atau data yang bermasalah sehingga dapat segera dilaporkan untuk diperbaiki. Sedangkan di *Feeder Bridge* harus melihat halaman proses antrian dan *error* dulu untuk mengetahuinya. Selain itu, fitur kirim belum bisa melakukan pengiriman beberapa data saja, padahal di beberapa kasus partisipan butuh hanya mengirimkan beberapa data saja, bukan semuanya.
2. Halaman proses antrian hanya ada di *Feeder Bridge* dan tidak ada di *ProFeeder* sehingga partisipan tidak memiliki ekspektasi apa pun untuk halaman ini tetapi menganggap cukup bagus karena memperlihatkan proses apa saja yang belum dikerjakan, sedang dikerjakan, atau sudah selesai.
3. Dibandingkan dengan *ProFeeder* masih ada yang kurang dari fitur manajemen pengguna yaitu pembatasan hak untuk menghapus data karena untuk hapus itu hanya orang tertentu yang dapat melakukannya sehingga perlu dibatasi.
4. Dari perspektif Dirpendik, tidak diperlukan untuk melihat persentase pengiriman untuk setiap jenis data, tetapi lebih dibutuhkan persentase secara keseluruhan. Tetapi untuk

DPTSI, *dashboard* per jenis data ini berguna. Saat ini, persentase pengiriman secara keseluruhan hanya bisa dilihat oleh Dirpendik yang kemudian dibagikan ke setiap prodi dalam bentuk tabel.

5. Selain persentase pengiriman secara keseluruhan yang dibutuhkan, halaman ketepatan waktu juga berguna untuk Dirpendik.

6.4. Hasil Uji Fungsional

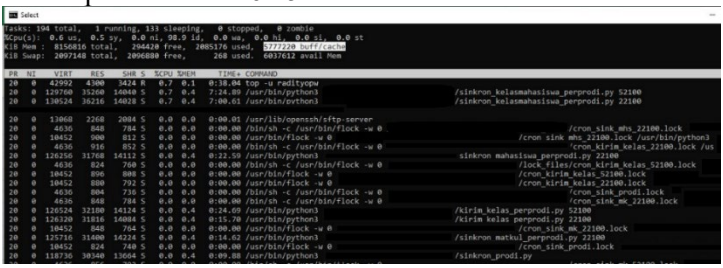
Pada tahap ini dilakukan pengujian dari segi fungsional aplikasi web dan sistem antrian. Fungsional yang diuji adalah manajemen pengguna, sinkronisasi data, dan kirim data. Hasil dari pengujian dapat dilihat pada Tabel 6.15 berikut.
Dokumentasi bukti pengujian dapat dilihat pada **LAMPIRAN F DOKUMENTASI PENGUJIAN FUNGSIONAL**.
tabelxx

Tabel 6.15 Hasil Pengujian Fungsional

Fitur		Berhasil / Tidak
Manajemen pengguna	Tambah pengguna	v
	Edit hak akses	v
	Hapus pengguna	v
Data Prodi	Sinkronisasi	v
	Kirim	
Data Mahasiswa	Sinkronisasi	v
	Kirim	
Data Dosen	Sinkronisasi	v
	Kirim	
Data Mata Kuliah	Sinkronisasi	v
	Kirim	
Data Kelas	Sinkronisasi	v
	Kirim	v
Data Aktivitas	Sinkronisasi	
	Kirim	
Data Aktivitas Kuliah Mahasiswa	Sinkronisasi	v
	Kirim	v

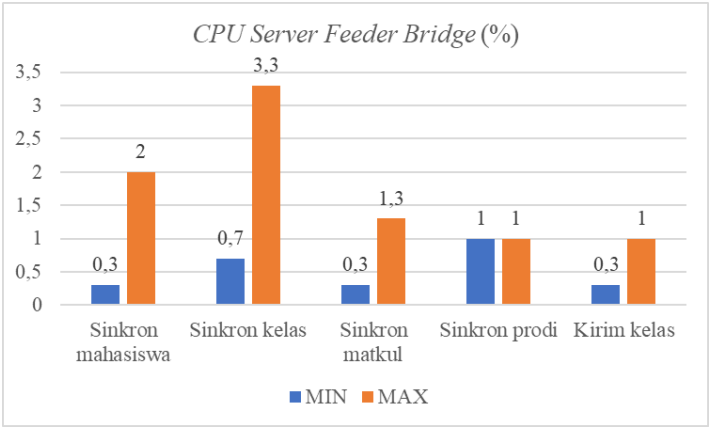
6.5. Hasil Uji Beban *Server Feeder Bridge*

Pada tahap ini dilakukan pemantauan terhadap *server* ketika sistem antrian melakukan proses antrian. Metode pemantauan adalah dengan mencatat penggunaan *CPU* dan *Memory server* menggunakan perintah TOP pada *server.* Data yang dicatat adalah saat *server* melakukan proses antrian yang masuk ke sistem antrian. Hal ini dilakukan karena sistem antrian tidak menggunakan sumber daya *server* ketika tidak sedang melakukan proses antrian. Tampilan dari pemantauan dapat dilihat pada Gambar 6.15 berikut.



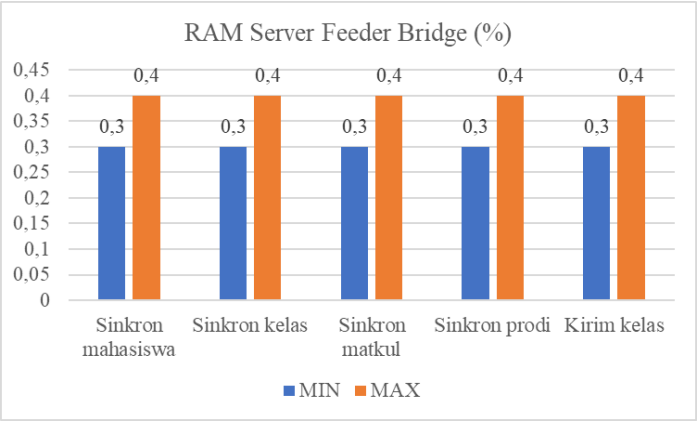
Gambar 6.15 Tampilan Beban *Server* dengan *Tools* TOP

Setelah itu dilakukan pencatatan penggunaan minimal dan maksimal dari *CPU* dan *Memory* dalam satuan persen. Hasil penggunaan *CPU server* dapat dilihat pada Gambar 6.16 berikut.



Gambar 6.16 Hasil Pemantauan Beban *CPU Server*

Sementara hasil penggunaan *Memory server* dapat dilihat pada Gambar 6.17 berikut.



Gambar 6.17 Hasil Pemantauan Beban *RAM Server*

Bedasarkan grafik tersebut dapat terlihat bahwa proses antrian tidak menggunakan banyak sumber daya *server*. Dari segi penggunaan *CPU*, proses hanya mengambil sekitar 1 persen dari keseluruhan *CPU*. Sedangkan penggunaan tertinggi hanya muncul sebesar 3,3 persen pada data kelas dari seluruh pengujian yang dilakukan. Dari segi *memory*, proses antrian sangat stabil dalam penggunaan sumber daya *memory*. Hal ini terlihat dari angka yang sama pada semua data yaitu antara 0,3 sampai 0,4 persen dari keseluruhan *memory*.

6.6. Hasil Perbandingan Performa *Feeder Bridge* dengan *ProFeeder* dari Sisi *Client*

Pada tahap ini kedua partisipan menjalankan web *Feeder Bridge* dan aplikasi *ProFeeder* di komputer masing-masing partisipan. Partisipan melakukan aktivitas seperti menjalankan proses sinkronisasi dan kirim data pada masing-masing sistem. Berdasarkan aktivitas tersebut, ada dua hal yang penulis amati yaitu penggunaan sumber daya untuk masing-masing sistem dan durasi proses dijalankan. Adapaun sumber daya yang penulis adalah sumber daya yang digunakan oleh *Feeder Bridge* dan *ProFeeder* dari sisi *RAM*, dan *CPU* yang dilihat dari fitur *Task Manager* pada komputer partisipan. Untuk durasi proses

yang penulis amati adalah berapa lama waktu yang dibutuhkan untuk menyelesaikan satu proses sinkronisasi dan kirim data. Adapun data yang digunakan dalam proses tersebut adalah data biodata mahasiswa, aktivitas kuliah mahasiswa, kelas, dan peserta kelas di *ProFeeder*. Untuk *Feeder Bridge* menggunakan data biodata mahasiswa, akm, kelas, dan mata kuliah di *Feeder Bridge*. Prodi yang digunakan adalah S1 Sistem Informasi dan S1 Teknik Elektro dengan periode semester 20171 dan 20181.

6.6.1. Hasil uji beban aplikasi

Pengujian ini diawali dengan pemeriksaan spesifikasi komputer masing-masing partisipan. Spesifikasi yang digunakan ditunjukkan pada Tabel 6.16.

Tabel 6.16 Sumber Daya Partisipan

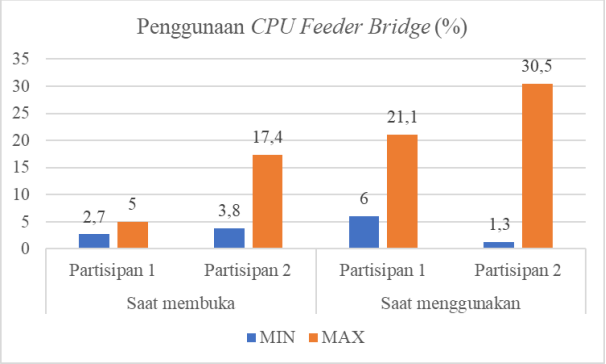
Partisipan	1	2
Penyedia Layanan Internet	wifi.id	*Three*
Kecepatan Akses Internet	20 *MB/s*	13 *MB/s*-15 *MB/s*
OS	*Windows* 10 *Education*	*Windows* 10 *Pro*
RAM	8 *GB*	4 *GB*
Processor	AMD A8	AMD A8

Berdasarkan Tabel 6.15, dapat diketahui bahwa spesifikasi komputer dan layanan internet yang digunakan oleh kedua partisipan cukup berbeda. Melihat perbedaan spesifikasi tersebut, secara umum dapat dikatakan bahwa perangkat yang digunakan oleh partisipan 1 memiliki spesifikasi yang lebih baik dibandingkan partisipan 2. Setelah melakukan pemeriksaan spesifikasi, selanjutnya adalah mengamati penggunaan sumber daya komputer pada saat membuka dan saat menggunakan aplikasi baik *Feeder Bridge* maupun *ProFeeder*. Pengamatan ini dilakukan selama 15 detik untuk pengamatan saat membuka aplikasi dan 1 menit pada saat menggunakan apliksasi. Pada saat menggunakan, hal yang penulis amati adalah pada saat partisipan melakukan proses sinkronisasi atau kirim data

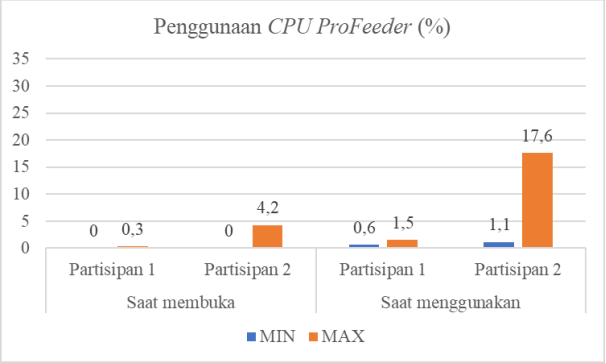
peserta kelas karena memiliki jumlah data yang lebih banyak dari jenis data yang lain sehingga diasumsikan bahwa penggunaan paling berat adalah pada saat proses tersebut. Berikut hasil pengamatan penulis dari partisipan 1 dan partisipan 2:

a. Penggunaan *CPU*

Penggunaan *CPU* untuk aplikasi *Feeder Bridge* dan *ProFeeder* ditunjukkan dengan Gambar 6.18 dan Gambar 6.19.



Gambar 6.18 Hasil Pemantauan Beban *CPU* Web *Feeder Bridge*

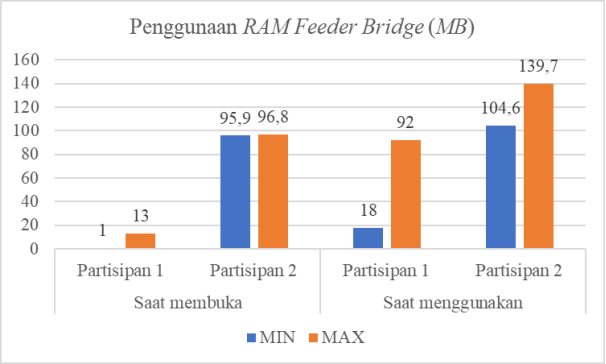


Gambar 6.19 Hasil Pemantauan Beban CPU *ProFeeder*

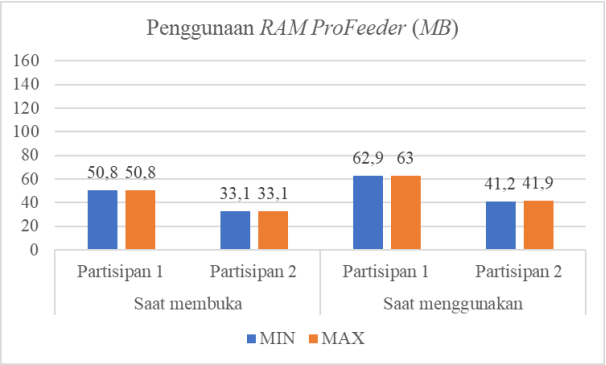
Grafik tersebut dan berikutnya menunjukkan perbedaan pengguaan pada saat membuka dan menggunakan aplikasi *Feeder Bridge* dan *ProFeeder* oleh partisipan 1 dan partisipan 2 beserta penggunaan minimal dan maksimalnya. Angka yang ditunjukkan oleh grafik penggunaan sumber daya *Feeder Bridge* merupakan angka setelah dikurangi dengan penggunaan oleh aplikasi *browser* yang digunakan. Berdasarkan penggunaan *CPU* seperti yang ditunjukkan pada Gambar 6.18 dan Gambar 6.19 dapat diketahui bahwa rentang penggunaan *CPU* minimum dan maksimum *Feeder Bridge* relatif lebih besar dari *ProFeeder*. Selain itu, penggunaan *CPU* partisipan 2 untuk *Feeder Bridge* dan *ProFeeder* relatif lebih besar dari partisipan 1.

b. Penggunaan *RAM*

Penggunaan *RAM* untuk aplikasi *Feeder Bridge* dan *ProFeeder* ditunjukkan dengan Gambar 6.20 dan Gambar 6.21.



Gambar 6.20 Hasil Pemantauan Beban *RAM Feeder Bridge*

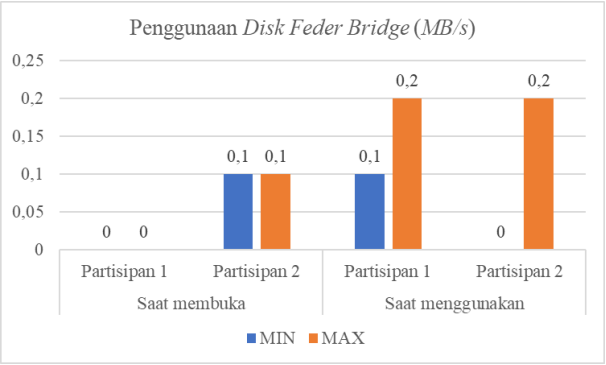


Gambar 6.21 Hasil Pemantauan Beban *RAM ProFeeder*

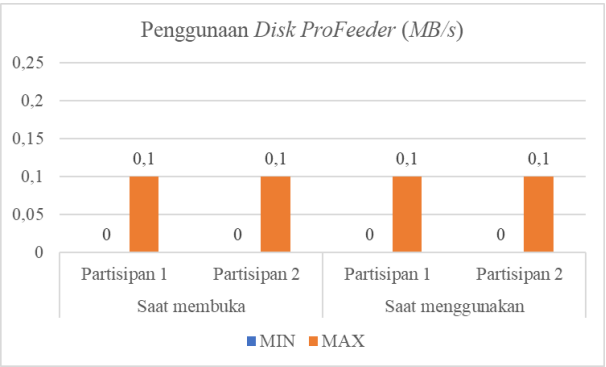
Berdasarkan penggunaan *RAM* seperti yang ditunjukkan pada Gambar 6.20 dan Gambar 6.21 dapat diketahui bahwa selisih penggunaan *RAM* minimum dan maksimum *Feeder Bridge* partisipan 1 relatif lebih besar dari partisipan 2 baik pada saat membuka maupun saat menggunakan. Di sisi lain, penggunaan *RAM* partisipan 2 untuk *Feeder Bridge* lebih besar dari partisipan 1, tetapi penggunaan *RAM* untuk *ProFeeder* relatif lebih kecil.

c. Penggunaan *disk*

Penggunaan *Disk* untuk aplikasi *Feeder Bridge* dan *ProFeeder* ditunjukkan dengan Gambar 6.22 dan Gambar 6.23.



Gambar 6.22 Hasil Pemantauan Beban *Disk Feeder Bridge*



Gambar 6.23 Hasil Pemantauan Beban *Disk ProFeeder*

Berdasarkan penggunaan *disk* seperti yang ditunjukkan pada Gambar 6.22 dan Gambar 6.23 dapat diketahui bahwa penggunaan disk *Feeder Bridge* memiliki penggunaan yang relatif lebih tinggi dari *ProFeeder* baik dari partisipan 1 maupun partisipan 2. Untuk penggunaan *disk ProFeeder* pada partisipan 1 dan partisipan 2 memiliki nilai yang sama yaitu 0 *MB*/*s* sampai 0,1 *MB*/*s* dan stabil di angka tersebut. Namun 0 MB/s penggunaan baik bukan berarti tidak menggunakan sama sekali

karena 1 *MB* = 1024 *KB* sehingga memungkinkan penggunaan kurang dari 0,1 *MB*/*s* namun tidak ditampilkan karena satuan terkecil yang ditampilkan adalah *MB*/*s*. Hal tersebut masih belum penulis teliti lebih dalam.

6.6.2. Hasil uji waktu

Tahap yang penulis lakukan setelah pengamatan penggunaan sumber daya adalah pengamatan terhadap durasi penyelesaian aktivitas sinkronisasi dan kirim. Pada tahap ini partisipan akan melakukan aktivitas sinkronisasi data menggunakan data yang serupa pada kedua aplikasi. Selanjutnya penulis akan mengamati lamanya waktu sinkronisasi dan kirim dari kedua aplikasi. Berikut merupakan hasil pengamatan yang telah penulis dapatkan yang ditunjukkan pada Tabel 6.17, Tabel 6.18, Tabel 6.19, dan Tabel 6.20.

Tabel 6.17 Hasil Pengujian Waktu Sinkronisasi Biodata Mahasiswa & AKM dalam Satuan Detik

Prodi	Periode	*ProFeeder*		*Feeder Bridge*	
		1	2	1	2
52100	20171	257	713	270	321
	20181	188	611	292	346
22100	20171	345	680	354	335
	20181	452	786	586	586

Tabel 6.18 Hasil Pengujian Waktu Sinkronisasi Data Kelas & Peserta Kelas dalam Satuan Detik

Prodi	Periode	*ProFeeder*		*Feeder Bridge*	
		1	2	1	2
52100	20171	529	2807	788	768
	20181	391	3222	873	892
22100	20171			862	834
	20181			1101	1023

Tabel 6.19 Hasil Pengujian Waktu Kirim Data Kelas & Peserta Kelas dalam Satuan Detik

Prodi	Periode	*ProFeeder*		*Feeder Bridge*	
		1	2	1	2
52100	20181	1115	6162	1190	1175

Tabel 6.20 Hasil Pengujian Waktu Sinkronisasi Mata Kuliah Web *Feeder Bridge*

Matkul	Partisipan 1	Partisipan 2
SI kurikulum 18	64 detik	64 detik
Elektro kurikulum 18	159 detik	101 detik

Setelah melakukan pengamatan durasi penyelesaian aktivitas sinkronisasi saat menggunakan web *Feeder Bridge* dengan aplikasi *ProFeeder* kemudian membandingkan hasil keduanya antara partisipan 1 dan partisipan 2, penulis menemukan beberapa temuan seperti berikut:

1. Waktu yang diperlukan partisipan 1 dan 2 untuk menyelesaikan proses sinkronisasi dan kirim pada aplikasi *ProFeeder* memiliki perbedaan dimana partisipan 2 selalu memiliki waktu yang lebih lama dalam menyelesaikan proses.
2. Waktu yang diperlukan partisipan 1 dan partisipan 2 untuk menyelesaikan proses sinkronisasi dan kirim pada web *Feeder Bridge* tidak memiliki perbedaan waktu yang signifikan.
3. Perbedaan *device*, spesifikasi laptop, serta kualitas jaringan internet partisipan tidak mempengaruhi kecepatan sinkronisasi dan pengiriman data pada web *Feeder Bridge*, dikarenakan pada web *Feeder Bridge* proses sinkronisasi dan pengiriman data dilakukan sepenuhnya oleh sistem antrian yang berjalan dari sisi *server*.
4. Web *Feeder Bridge* belum bisa mencapai waktu tercepat aplikasi *ProFeeder* dalam melakukan proses sinkronisasi dan kirim data.

5. Dalam kondisi tertentu, seperti kualitas jaringan internet pengguna yang buruk, aplikasi *ProFeeder* dapat memiliki waktu yang sangat lama dibanding web *Feeder Bridge* dalam menyelesaikan proses sinkronisasi dan pengiriman data.

BAB VII
KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari semua proses yang penulis lakukan serta saran dari penulis untuk pengembangan lebih lanjut yang lebih baik.

7.1. Kesimpulan

Berikut kesimpulan yang diperoleh penulis dari pengerjaan tugas akhir:

1. Berdasarkan penelitian yang telah dilakukan, penulis menemukan beberapa permasalahan yang diperoleh dari studi dokumen dan wawancara, lebih tepatnya dua poin permasalahan dari studi dokumen dan tiga poin permasalahan dari wawancara. Sesuai dengan topik penelitian yaitu sistem pelaporan PDDikti, yang menjadi permasalahan utama untuk diselesaikan penulis dalam penelitian ini adalah kemampuan sistem untuk dapat melakukan sinkronisasi data dan mengirimkan data dari SIAKAD ke *Feeder* PDDikti.
2. Dengan prototipe aplikasi dan *dashboard* yang telah dibuat, penulis telah memenuhi tujuan kedua dan ketiga dari penelitian ini, yaitu membuat dashboard untuk pemantauan yang diperlukan untuk memantau pengiriman data ITS ke PDDikti dan melakukan pengujian sistem terhadap kebutuhan pengguna dengan melakukan *usability testing* untuk mengetahui tingkat keberhasilan dalam menggunakan fitur dalam sistem kepada dua responden dan pengujian performa untuk membandingkan performa *Feeder Bridge* dengan *ProFeeder*. Namun untuk tujuan pertama yaitu membuat sistem yang dapat mengirim data dari SIAKAD ke *Feeder* PDDikti masih belum dapat dipenuhi karena belum semua pengiriman data berhasil dilakukan dan diujikan.
3. Berdasarkan hasil evaluasi yang didapat, penulis mendapat kesimpulan bahwa sistem yang dibuat cukup berbeda dengan *ProFeeder* dari segi tampilan data sehingga admin perlu beradaptasi kembali. Namun dari segi fitur, sistem yang telah dibuat berhasil melakukan aktivitas utama

seperti melakukan sinkronisasi serta pengiriman data, hanya saja kelengkapan datan yang masih kurang. Kendati demikian, penulis terus berupaya memperbaiki sistem tersebut sampai dapat digunakan secara maksimal dan dibuat semirip mungkin dengan *ProFeeder* untuk memudahkan adaptasi pengguna.

4. Dari hasil proses pengujian baik *usability testing*, interview setelah pengujian, dan pengujian performa, diperoleh beberapa poin kesimpulan seperti berikut:
    a. Hasil *usability testing* pada prototipe aplikasi memiliki rata-rata keberhasilan 98% secara keseluruhan. Pada prototipe *dashboard*, rata-rata keberhasilan hasil *usability testing* yang didapatkan adalah 87%. Berdasarkan hasil tersebut, penulis menyimpulkan bahwa secara keseluruhan penggunaan *Feeder Bridge* memiliki tingkat keberhasilan penggunaan yang tinggi.
    b. Tampilan dan pengelompokkan data *Feeder Bridge* cukup berbeda dengan *ProFeeder*. Partisipan 1 cenderung memilih menggunakan *Feeder Bridge* jika terdapat estimasi waktu kapan satu proses akan selesai. Berbeda dengan partisipan 1, partisipan 2 lebih menyukai *Feeder Bridge* daripada *ProFeeder* dan akan lebih suka jika dapat menampilkan *error* pada data yang berkaitan atau setidaknya dapat membedakan dengan cepat *error* data yang ada karena saat ini masih tercampur. Saat ini kedua partisipan masih nyaman menggunakan *ProFeeder* dan belum terbiasa dengan *Feeder Bridge*.
    c. Terlepas dari kekurangan yang ada dalam fitur *dashboard* yang disediakan oleh *Feeder Bridge*, kedua partisipan memiliki pendapat yang sama bahwa fitur *dashboard* ini akan sangat membantu pihak Dirpendik dan DPTSI dalam memantau proses pengiriman data PDDikti.
    d. Berdasarkan hasil pengujian performa terkait dengan penggunaan sumber daya komputer, penulis mendapat kesimpulan bahwa *Feeder Bridge*

menggunakan lebih banyak sumber daya daripada *ProFeeder* dipengaruhi oleh penggunaan aplikasi *browser* yang memiliki perbedaan penggunaan sumber daya untuk setiap *tab* yang dibuka. Sedangkan performa dari sisi waktu yang dibutuhkan untuk menyelesaikan satu proses dapat disimpulkan bahwa waktu yang dibutuhkan oleh *Feeder Bridge* bisa lebih cepat atau lebih lambat dibandingkan *ProFeeder* tergantung dari kecepatan internet saat menggunakan *ProFeeder*. Sedangkan durasi proses *Feeder Bridge* tidak dipengaruhi oleh kecepatan internet yang digunakan oleh *client* melainkan dari kecepatan *server*. Selain itu, semua proses sinkronisasi dan kirim dilakukan oleh *server* dan bukan komputer *client* sehingga hal ini sangat memudahkan admin prodi yang memiliki spesifikasi komputer dan kecepatan internet yang kurang memadai.

5. Berkaitan dengan analisis kebutuhan pengguna, penelitian yang penulis lakukan telah memenuhi kebutuhan untuk fungsi utama, kebutuhan fungsional, dan kebutuhan non fungsional. Pada fungsi utama dibutuhkan fungsi untuk sinkronisasi, melihat perbedaan data, dan mengirim data telah difasilitasi oleh *Feeder Bridge*. Untuk kebutuhan fungsional lain seperti *login*, logout, menampilkan *dashboard*, mengatur akses, menampilkan status proses antrian, menampilkan status *error* pengiriman data, memantau sistem antrian, dan menjalankan serta menghentikan sistem antrian juga telah disediakan oleh *Feeder Bridge*. Selanjutnya kebutuhan non fungsional yang membutuhkan sistem untuk dapat dijalankan di berbagai platform karena web, dapat dijalankan di berbagai komputer dengan spesifikasi berbeda tanpa ada kendala berarti, tidak menggunakan banyak sumber daya komputer dalam penggunaannya, dapat dijalankan pada segala jenis *browser* yang mendukung *JavaScript* dan *HTML5*, dan dapat berjalan terus selama 24 jam juga terpenuhi. Meskipun kebutuhan secara dasar dapat dipenuhi dengan sistem ini, namun ada beberapa bagian

yang perlu diperbaiki untuk dapat digunakan secara maksimal

6. Terlepas dari kekurangan yang masih terdapat pada aplikasi seperti data yang masih belum lengkap, penulis telah berhasil membuat aplikasi berbasis web yang mengubah proses sinkronisasi dan pengiriman data dari yang sebelumnya masih melibatkan sisi *client*, namun pada aplikasi ini proses sinkonisasi dan pengiriman data sepenuhnya dilakukan oleh sistem antrian yang berjalan pada server aplikasi sehingga tidak membebani memori dan internet pengguna.
7. *Change Data Capture (CDC)* masih belum dapat diterapkan karena sumber data SIAKAD yang tidak sesuai dengan *CDC*. Sumber data SIAKAD hanya dapat dilihat berdasarkan tabel *view* yang sudah dibuat admin database SIAKAD.
8. Kondisi pandemi Covid-19 menyebabkan penulis kesulitan dalam melakukan *testing* aplikasi terutama dalam melakukan *User Acceptance Test* sehingga tidak dapat dilakukan.

7.2. Saran

Bab ini berisi saran untuk pengembangan dan penelitian lebih lanjut terkait sistem pelaporan PDDIkti. Penulis menemukan beberapa saran yang dapat diterapkan ketika melanjutkan pengembangan sistem. Berikut ini saran penulis

1. Pengembangan aplikasi tidak mencakup beberapa data seperti data yang terkait kebijakan Kampus Merdeka Belajar dari Kemendikbud. Sehingga disarankan penambahan cakupan data untuk kebijakan Kampus Merdeka Belajar.
2. Pengembangan aplikasi tidak mencakup beberapa perubahan data seperti misalnya perubahan kurikulum dan perubahan struktur program studi dan fakultas.
3. Pengembangan fitur – fitur yang menunjang proses pelaporan data PDDikti seperti fitur estimasi selesai antrian, fitur notifikasi hasil sinkronisasi dan kirim data pada setiap halaman tampilan data, fitur pemilihan data tertentu untuk dikirim, serta kemampuan aplikasi secara

otomatis untuk memilih data bermasalah saja yang akan ditampilkan pada tampilan data.

4. Fitur *dashboard* masih belum memuat keseluruhan jenis data, tidak memiliki akses untuk admin prodi, dan tata letaknya masih membingungkan saat pertama kali digunakan. Untuk pengembangan ke depannya diharapkan dapat mencakup semua data terutama pengiriman biodata mahasiswa, bukan hanya pengiriman akm. Selain itu juga diharapkan admin prodi dapat diberikan akses untuk memantau progres pelaporan masing-masing prodinya. Dari sisi tata letak visualisasi dapat diperbaiki lagi untuk membuat visualisasi lebih jelas dan tidak membuat bingung pengguna.

LAMPIRAN A
DAFTAR *METHOD WEB SERVICE FEEDER* PDDIKTI

Nomor	Nama *Method*
1	GetAgama
2	GetAktivitasKuliahMahasiswa
3	GetAktivitasMengajarDosen
4	GetAlatTransportasi
5	GetAllProdi
6	GetAllPT
7	GetBentukPendidikan
8	GetBiodataMahasiswa
9	GetCountAktivitasMahasiswa
10	GetCountAktivitasMengajarDosen
11	GetCountBiodataMahasiswa
12	GetCountDosen
13	GetCountDosenPembimbing
14	GetCountDosenPengajarKelasKuliah
15	GetCountKelasKuliah
16	GetCountKonversiKampusMerdeka
17	GetCountKurikulum
18	GetCountMahasiswa
19	GetCountMahasiswaBimbinganDosen
20	GetCountMahasiswaLulusDO
21	GetCountMataKuliah
22	GetCountMatkulKurikulum
23	GetCountNilaiPerkuliahanKelas
24	GetCountNilaiTransferPendidikanMahasiswa
25	GetCountPenugasanSemuaDosen
26	GetCountPerguruanTinggi
27	GetCountPeriodePerkuliahan
28	GetCountPerkuliahanMahasiswa
29	GetCountPesertaKelasKuliah
30	GetCountPrestasiMahasiswa
31	GetCountProdi
32	GetCountRencanaEvaluasi

33	GetCountRencanaPembelajaran
34	GetCountRiwayatNilaiMahasiswa
35	GetCountRiwayatPendidikanMahasiswa
36	GetCountSkalaNilaiProdi
37	GetCountSubstansiKuliah
38	GetDataLengkapMahasiswaProdi
39	GetDataTerhapus
40	GetDetailKelasKuliah
41	GetDetailKurikulum
42	GetDetailMahasiswaLulusDO
43	GetDetailMataKuliah
44	GetDetailNilaiPerkuliahanKelas
45	GetDetailPenugasanDosen
46	GetDetailPeriodePerkuliahan
47	GetDetailPerkuliahanMahasiswa
48	GetDetailSkalaNilaiProdi
49	GetDictionary
50	GetDosenPembimbing
51	GetDosenPengajarKelasKuliah
52	GetFakultas
53	GetIkatanKerjaSdm
54	GetJabfung
55	GetJalurMasuk
56	GetJenisAktivitasMahasiswa
57	GetJenisEvaluasi
58	GetJenisKeluar
59	GetJenisPendaftaran
60	GetJenisPrestasi
61	GetJenisSertifikasi
62	GetJenisSMS
63	GetJenisSubstansi
64	GetJenisTinggal
65	GetJenjangPendidikan
66	GetKategoriKegiatan
67	GetKebutuhanKhusus
68	GetKRSMahasiswa

69	G*ETL*embagaPengangkat
70	G*ETL*evelWilayah
71	G*ETL*istAktivitasMahasiswa
72	G*ETL*istAnggotaAktivitasMahasiswa
73	G*ETL*istBidangMinat
74	G*ETL*istBimbingMahasiswa
75	G*ETL*istDosen
76	G*ETL*istKelasKuliah
77	G*ETL*istKonversiKampusMerdeka
78	G*ETL*istKurikulum
79	G*ETL*istMahasiswa
80	G*ETL*istMahasiswaLulusDO
81	G*ETL*istMataKuliah
82	G*ETL*istNilaiPerkuliahanKelas
83	G*ETL*istPenugasanDosen
84	G*ETL*istPenugasanSemuaDosen
85	G*ETL*istPeriodePerkuliahan
86	G*ETL*istPerkuliahanMahasiswa
87	G*ETL*istPerubahanRiwayatPendidikan
88	G*ETL*istPrestasiMahasiswa
89	G*ETL*istRencanaEvaluasi
90	G*ETL*istRencanaPembelajaran
91	G*ETL*istRiwayatPendidikanMahasiswa
92	G*ETL*istSkalaNilaiProdi
93	G*ETL*istSubstansiKuliah
94	G*ETL*istUjiMahasiswa
95	GetMahasiswaBimbinganDosen
96	GetMatkulKurikulum
97	GetNegara
98	GetNilaiTransferPendidikanMahasiswa
99	GetPangkatGolongan
100	GetPekerjaan
101	GetPembiayaan
102	GetPenghasilan
103	GetPerhitunganSKS
104	GetPeriode

105	GetPeriodeLampau
106	GetPesertaKelasKuliah
107	GetProdi
108	GetProfilPT
109	GetRekapIPSMahasiswa
110	GetRekapJumlahDosen
111	GetRekapJumlahMahasiswa
112	GetRekapKHSMahasiswa
113	GetRekapKRSMahasiswa
114	GetRekapLaporan
115	GetRiwayatFungsionalDosen
116	GetRiwayatNilaiMahasiswa
117	GetRiwayatPangkatDosen
118	GetRiwayatPendidikanDosen
119	GetRiwayatPenelitianDosen
120	GetRiwayatSertifikasiDosen
121	GetSemester
122	GetStatusKeaktifanPegawai
123	GetStatusKepegawaian
124	GetStatusMahasiswa
125	GetTahunAjaran
126	GetTingkatPrestasi
127	GetToken
128	GetTranskripMahasiswa
129	GetWilayah
130	DeleteAktivitasMahasiswa
131	DeleteAnggotaAktivitasMahasiswa
132	DeleteBimbingMahasiswa
133	DeleteBiodataMahasiswa
134	DeleteDosenPembimbing
135	DeleteDosenPengajarKelasKuliah
136	DeleteKelasKuliah
137	DeleteKonversiKampusMerdeka
138	DeleteKurikulum
139	DeleteMahasiswaLulusDO
140	DeleteMataKuliah

141	DeleteMatkulKurikulum
142	DeleteNilaiTransferPendidikanMahasiswa
143	DeletePeriodePerkuliahan
144	DeletePerkuliahanMahasiswa
145	DeletePesertaKelasKuliah
146	DeletePrestasiMahasiswa
147	DeleteRencanaEvaluasi
148	DeleteRencanaPembelajaran
149	DeleteRiwayatPendidikanMahasiswa
150	DeleteSkalaNilaiProdi
151	DeleteSubstansiKuliah
152	DeleteTranskripMahasiswa
153	DeleteUjiMahasiswa
154	DetailBiodataDosen
155	ExportDataAktivitasKuliah
156	ExportDataKelasPerkuliahan
157	ExportDataMahasiswa
158	ExportDataMahasiswaKRS
159	ExportDataMahasiswaLulus
160	ExportDataMatkulProdi
161	ExportDataMengajarDosen
162	ExportDataNilaiTransfer
163	ExportDataPenugasanDosenProdi
164	HitungTranskripAngkatan
165	InsertAktivitasMahasiswa
166	InsertAnggotaAktivitasMahasiswa
167	InsertBimbingMahasiswa
168	InsertBiodataMahasiswa
169	InsertDosenPembimbing
170	InsertDosenPengajarKelasKuliah
171	InsertKelasKuliah
172	InsertKonversiKampusMerdeka
173	InsertKurikulum
174	InsertMahasiswaLulusDO
175	InsertMataKuliah
176	InsertMatkulKurikulum

177	InsertNilaiTransferPendidikanMahasiswa
178	InsertPeriodePerkuliahan
179	InsertPerkuliahanMahasiswa
180	InsertPerubahanRiwayatPendidikan
181	InsertPesertaKelasKuliah
182	InsertPrestasiMahasiswa
183	InsertRencanaEvaluasi
184	InsertRencanaPembelajaran
185	InsertRiwayatPendidikanMahasiswa
186	InsertSkalaNilaiProdi
187	InsertSubstansiKuliah
188	InsertTranskripMahasiswa
189	InsertUjiMahasiswa
190	UpdateAktivitasMahasiswa
191	UpdateBiodataMahasiswa
192	UpdateDosenPengajarKelasKuliah
193	UpdateKelasKuliah
194	UpdateKonversiKampusMerdeka
195	UpdateKurikulum
196	UpdateMahasiswaLulusDO
197	UpdateMataKuliah
198	UpdateNilaiPerkuliahanKelas
199	UpdateNilaiTransferPendidikanMahasiswa
200	UpdatePeriodePerkuliahan
201	UpdatePerkuliahanMahasiswa
202	UpdatePerubahanRiwayatPendidikan
203	UpdatePrestasiMahasiswa
204	UpdateRencanaEvaluasi
205	UpdateRencanaPembelajaran
206	UpdateRiwayatPendidikanMahasiswa
207	UpdateSkalaNilaiProdi
208	UpdateSubstansiKuliah

LAMPIRAN B
ANALISIS HALAMAN *DASHBOARD*

1. Analisis halaman *dashboard*

No	*dashboard*	Indikator	Target	Rumus
1	Pengiriman akm	Persentase Pengiriman akm	100%	Jumlah Data akm Terkirim/Jumlah data akm*100%
2	Pengiriman Kelas	Persentase Perngiriman Kelas	100%	Jumlah Data Kelas Terkirim/Jumlah data Kelas*100%
3	Ketepatan Waktu Pengiriman Data	Persentase Ketepatan Waktu Pengiriman Data	100%	Jumlah Data Terkirim Tepat Waktu/Jumlah Data Terkirim*100%
4	Penyelesaian Komplain	Persentase Penyelesaian Komplain	100%	Jumlah Data Komplain Selesai/Jumlah Data Komplain*100%

2. Detail halaman pengiriman akm

dashboard	Pengiriman akm					
Tujuan	Meningkatkan persentase pengiriman akm					
No	Bagian	Nama Visual	Deskripsi	Bentuk Visual	Target	keterangan
1	Indikator	Persentase Pengiriman AKM	Memperlihatkan persentase pengiriman akm secara keseluruhan	Gauge	100%	
2	Variabel	Jumlah Pengiriman AKM	Memperlihatkan jumlah data akm yang sudah dikirimkan	Angka		Didapatkan dari jumlah data akm yang ada di *Feeder* PDDikti sesuai dengan mahasiswa, prodi, dan periode semesternya
3	Variabel	Jumlah Data AKM	Memperlihatkan jumlah data akm yang ada	Angka		Didapatkan dari jumlah data akm yang ada di Siakad sesuai dengan mahasiswa, prodi,

						dan periode semesternya
4	Trend	Tren Persentase Pengiriman AKM	Memperlihatkan tren persentase pengiriman akm setiap periode semester	Trendline		
5	Detail	10 Departemen Dengan Persentase Pengiriman AKM Terendah	Memperlihatkan 10 departemen dengan persentase pengiriman akm paling rendah	Bar		
6	Detail	Data AKM yang Belum Terkirim	Memperlihatkan data akm yang belum dikirimkan	Tabel		
7	Detail	Data AKM Abnormal	Memperlihatkan data akm yang tidak sesuai dengan harapan	Tabel		Didapat dari jumlah data akm yang lebih dari satu atau data akm terkirim yang lebih

						dari satu untuk setiap periodenya

3. Detail halaman pengiriman kelas

dashboard	Pengiriman Kelas					
Tujuan	Meningkatkan persentase pengiriman kelas					
No	Bagian	Nama Visual	Deskripsi	Bentuk Visual	Target	keterangan
1	Indikator	Persentase Pengiriman Kelas	Memperlihatkan persentase pengiriman kelas secara keseluruhan	Gauge	100%	
2	Variabel	Jumlah Pengiriman Kelas	Memperlihatkan jumlah data kelas yang sudah dikirimkan	Angka		Didapatkan dari jumlah data kelas yang ada di *Feeder* PDDikti sesuai dengan mahasiswa, prodi, dan periode semesternya
3	Variabel	Jumlah Data Kelas	Memperlihatkan jumlah data kelas yang ada	Angka		Didapatkan dari jumlah data kelas yang ada di Siakad

						sesuai dengan mahasiswa, prodi, dan periode semesternya
4	Trend	Tren Persentase Pengiriman Kelas	Memperlihatkan tren persentase pengiriman kelas setiap periode semester	Trendline		
5	Detail	10 Departemen Dengan Persentase Pengiriman Kelas Terendah	Memperlihatkan 10 departemen dengan persentase pengiriman kelas paling rendah	Bar		
6	Detail	Data Kelas yang Belum Terkirim	Memperlihatkan data kelas yang belum dikirimkan	Tabel		
7	Detail	Data Kelas Abnormal	Memperlihatkan data kelas yang tidak sesuai dengan harapan	Tabel		Didapat dari jumlah data kelas yang lebih dari satu atau data kelas terkirim

						yang lebih dari satu untuk setiap periodenya

4. Detail halaman penyelesaian komplain

dashboard	Penyelesaian Komplain					
Tujuan	Meningkatkan persentase dan kecepatan penyelesaian komplain					
No	Bagian	Nama Visual	Deskripsi	Bentuk Visual	Target	keterangan
1	Indikator	Persentase Penyelesaian Komplain	Memperlihatkan persentase penyelesaian komplain secara keseluruhan	Gauge	100%	
2	Variabel	Jumlah Penyelesaian Komplain	Memperlihatkan jumlah data komplain yang sudah diselesaikan	Angka		Didapatkan dari jumlah data yang ada di antrian komplain dengan status selesai
3	Variabel	Jumlah Data Komplain	Memperlihatkan jumlah data komplain yang ada	Angka		Didapatkan dari jumlah data yang

						ada di antrian komplain
4	Trend	Tren Persentase Penyelesaian Komplain	Memperlihatkan tren persentase penyelesaian komplain setiap periode semester	Trendline		
5	Detail	10 Departemen Dengan Persentase Penyelesaian Komplain Terendah	Memperlihatkan 10 departemen dengan persentase penyelesaian komplain paling rendah	Bar		
6	Detail	10 Penyelesaian Komplain Terlama	Memperlihatkan lama waktu penyelesaian komplain	Bar		Didapat dari selisih antara waktu komplain masuk dengan waktu komplain selesai
7	Detail	Rata-Rata Durasi	Memperlihatkan durasi penyelesaian	Angka		Didapat dari rata-rata selisih antara waktu komplain

		Penyelesaian Komplain	komplain dalam satuan hari			masuk dengan waktu komplain selesai
8	Detail	Data Komplain yang Belum Diselesaikan	Memperlihatkan data komplain yang belum diselesaikan	Tabel		

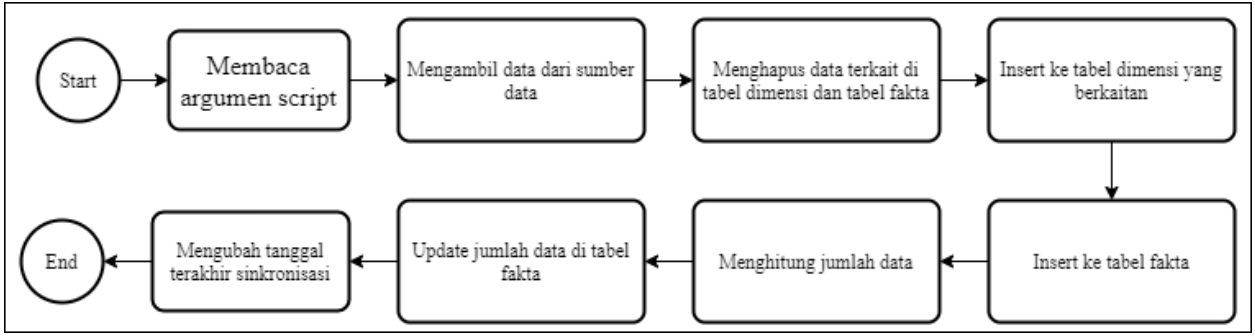
5. Detail ketepatan waktu

dashboard	Ketepatan Waktu					
Tujuan	Meningkatkan persentase ketepatan waktu					
No	Bagian	Nama Visual	Deskripsi	Bentuk Visual	Target	keterangan
1	Indikator	Persentase Ketepatan Waktu	Memperlihatkan persentase ketepatan waktu secara keseluruhan	Gauge	100%	
2	Variabel	Jumlah Data Terkirim Tepat Waktu	Memperlihatkan jumlah data yang sudah dikirimkan tepat waktu	Angka		Didapatkan dari jumlah data yang ada di antrian pengiriman dengan

						status selesai dan waktu pengirimannya tidak lebih dari jadwal
3	Variabel	Jumlah Data Terkirim	Memperlihatkan jumlah data yang sudah dikirimkan	Angka		Didapatkan dari jumlah data yang ada di antrian pengiriman dengan status selesai
4	Trend	Tren Persentase Ketepatan Waktu	Memperlihatkan tren persentase ketepatan waktu setiap periode semester	Trendline		
5	Detail	10 Departemen Dengan Persentase Ketepatan Waktu Terendah	Memperlihatkan 10 departemen dengan persentase ketepatan waktu paling rendah	Bar		

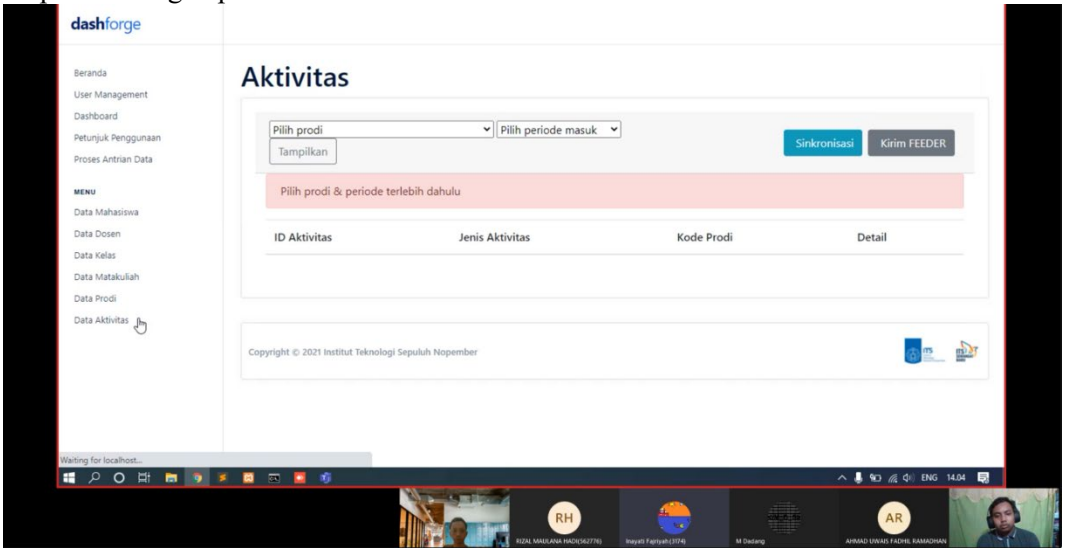
6	Detail	Keterlambatan Pengiriman Data	Memperlihatkan selisih waktu selesainya pengiriman data yang terlambat dikirimkan dengan jadwal yang ditentukan untuk masing-masing jenis data dalam satuan hari	Bar		Didapatkan dari waktu pengiriman dikurangi jadwal yang ditentukan

LAMPIRAN C
DIAGRAM ALUR PROSES *ETL*

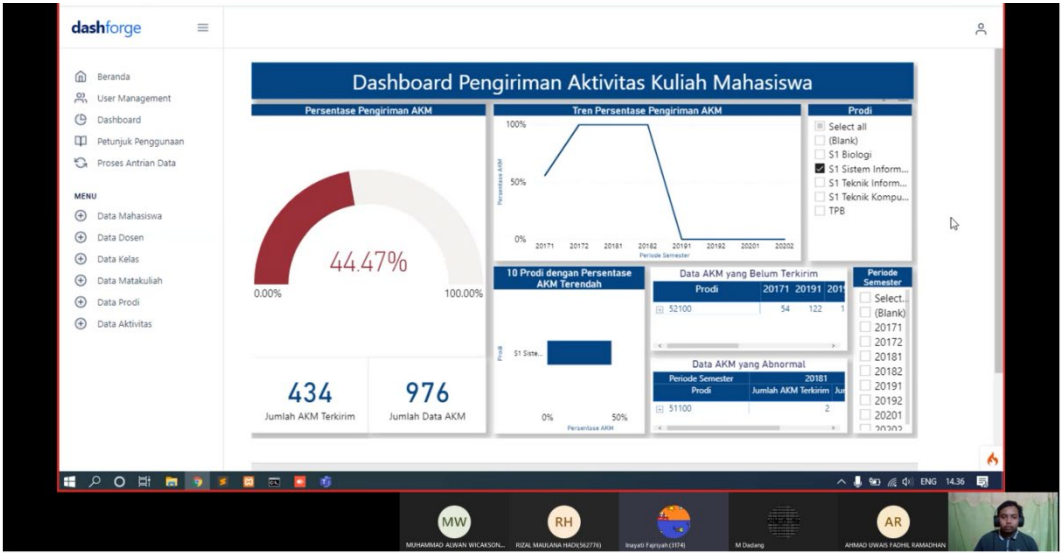


LAMPIRAN D
DOKUMENTASI KEGIATAN EVALUASI DAN USABILITY TESTING

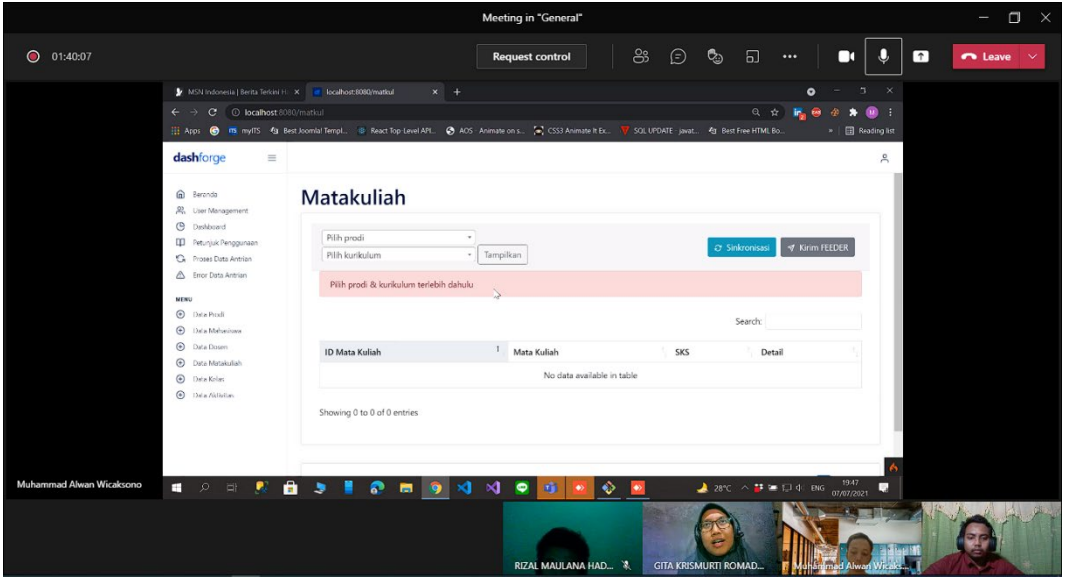
1. Evaluasi aplikasi dengan pihak DPTSI



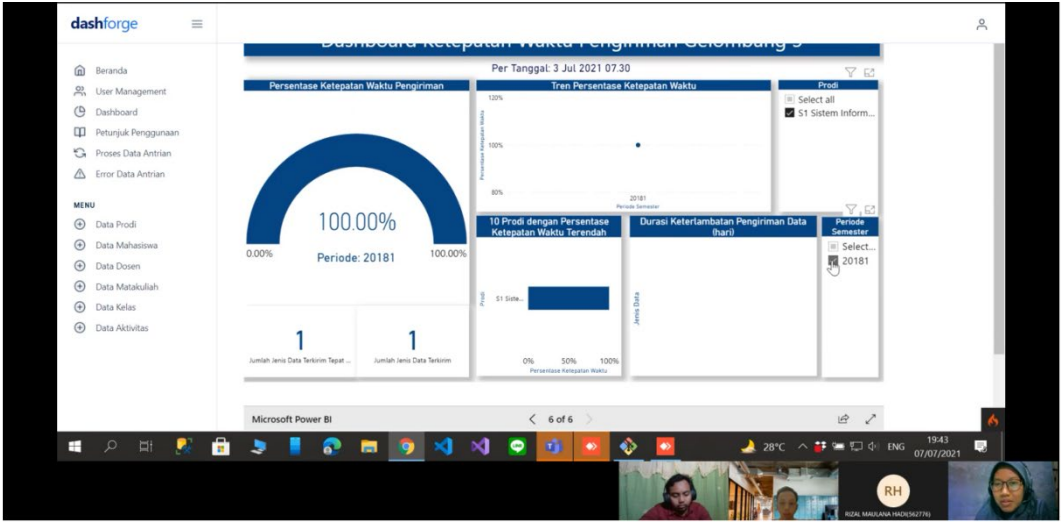
2. Evaluasi *dashboard* dengan pihak DPTSI



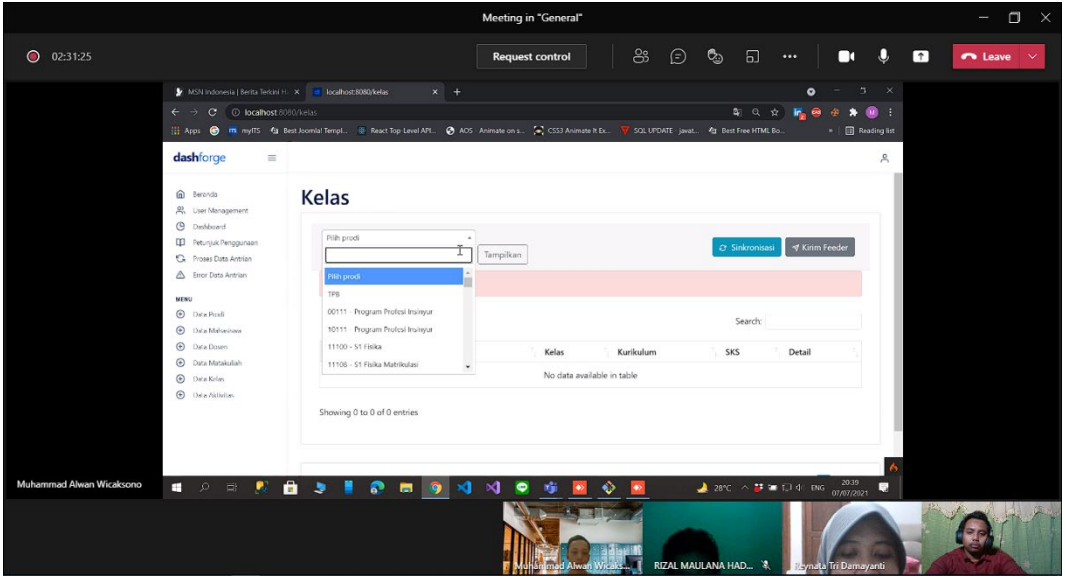
3. *Usability testing* aplikasi dengan partisipan 1



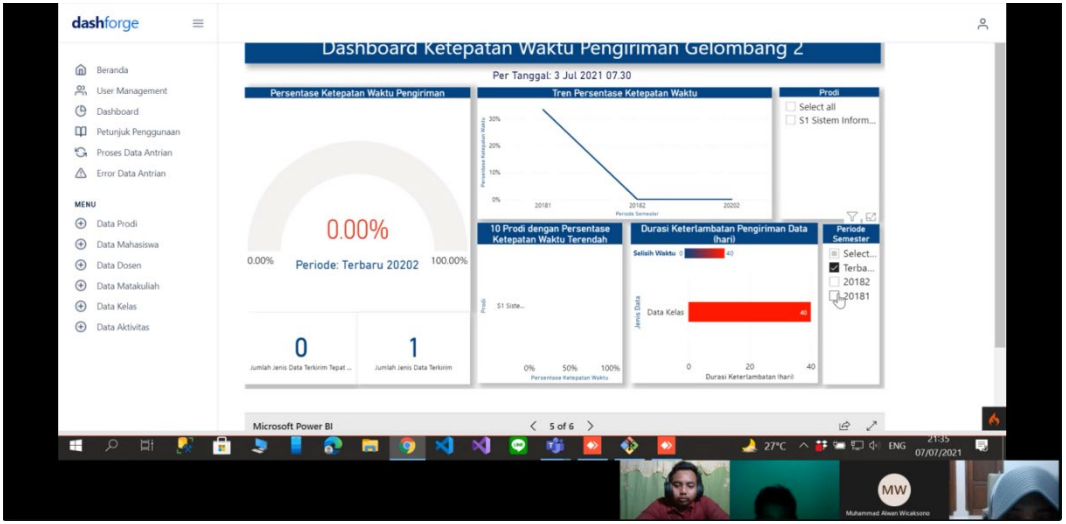
4. *Usability testing dashboard* dengan partisipan 1



5. *Usability testing* aplikasi dengan partisipan 2



6. *Usability testing dashboard* dengan partisipan 2


dashforge
Beranda
User Management
Dashboard
Petunjuk Penggunaan
Proses Data Antrian
Error Data Antrian
MENU
Data Prodi
Data Mahasiswa
Data Dosen
Data Matakuliah
Data Kelas
Data Aktivitas
Dashboard Ketepatan Waktu Pengiriman Gelombang 2
Per Tanggal: 3 Jul 2021 07.30
Persentase Ketepatan Waktu Pengiriman
0.00%
0.00%
Periode: Terbaru 20202
100.00%
0
1
Tren Persentase Ketepatan Waktu
Prodi
Select all
S1 Sistem Inform...
10 Prodi dengan Persentase Ketepatan Waktu Terendah
Durasi Keterlambatan Pengiriman Data (hari)
Selisih Waktu
Data Kelas
Periode Semester
Select...
Terba...
20182
20181
Microsoft Power BI
5 of 6
27°C
ENG
21:35
07/07/2021
MW
Muhammad Alwan Wicaksono

LAMPIRAN E
DOKUMEN *SRS FEEDER BRIDGE*

Daftar Isi

1. Pendahuluan

1.1 Tujuan Penulisan Dokumen

Dokumen *Software Requirement Spesification* (*SRS*) merupakan dokumen spesifikasi perangkat lunak untuk membangun sebuah aplikasi berbasis web "*Feeder Bridge*" yang merupakan aplikasi pelaporan data PDDikti. Dokumen ini dibuat untuk membantu memudahkan dalam pembuatan aplikasi tersebut, dokumen ini menjadi acuan teknis dalam pembuatan aplikasi tersebut.

1.2 Audien yang Dituju dan Pembaca yang Disarankan

Dokumen ini ditujukan kepada tim pengembang aplikasi ini yaitu tenaga magang Subdit *Big Data* DPTSI bagian pengembangan aplikasi *monitoring* dan pengiriman data forlap PDDikti.

1.3 Lingkup Masalah

Setiap perguruan tinggi di Indonseia diwajibkan untuk melakukan pengiriman data penyelenggaraan pendidikan tinggi kepada PDDikti sistem yang mengelola data perguruan tinggi yang terintegrasi secara nasional untuk dimanfaatkan sebagai sistem pendukung keputusan. Institut Teknologi Sepuluh Nopember (ITS) Surabaya telah menggunakan sistem pelaporan dari vendor luar yaitu *ProFeeder* dari SEVIMA. Aplikasi saat ini merupakan aplikasi berbasis destop yang tentunya memakan memori dan *RAM* dari komputer. Proses pengiriman data aplikasi tersebut juga sangat bergantung pada internet *client*. Dengan aplikasi *Feeder Bridge* proses sinkronisasi dan pengiriman tidak memerlukan pemrosesan data dari sisi *client* melainkan dilakukan oleh sistem antrian yang berjalan di server aplikasi sehingga tidak membebani internet pengguna. Aplikasi *Feeder Bridge* juga dibuat berbasis web sehingga tidak memakan memori besar pada komputer pengguna. Aplikasi ini juga dapat dijalankan pada segala sistem operasi serta tidak memerlukan proses intalasi.

1.4 Definisi dan Istilah

- *SRS* : *Software Requirements Specification*, atau Spesifikasi Kebutuhan Perangkat Lunak (SKPL)

- PDDikti : Pangkalan Data Pendidikan Tinggi, merupakan sistem yang mengelola data perguruan tinggi yang terintegrasi secara nasional
- *ProFeeder*: Aplikasi yang saat ini digunakan ITS untuk melakukan pengiriman data.
- *Feeder Bridge* : Aplikasi pengirman data yang sedang dikembangkan.
- SIAKAD : Sistem Informasi Akademik
- DPTSI ITS : Direktorat Pengembangan Teknologi dan Sistem Informasi ITS
- *Server* : Sebuah jaringan komputer yang melayani khusus permintaan *HTTP* dan *HTTPS.* Web *server* menerima kode sedemikian rupa dari browser, lalu mengirimnya kembali dalam bentuk laman web.
- *Database* : *Database* adalah sekumpulan data yang terorganisir dalam bentuk skema, tabel, query, laporan, view, dan objek lainnya, sehingga dapat diperiksa oleh suatu program komputer.
- *HTTP* : (*Hypertext Transfer Protocol*) yaitu sebuah protokol lapisan aplikasi untuk mentransmisi dokumen *hypermedia* seperti *html*.

1.5 Referensi

- *IEEE 830-1998 Recommended Practice for Software Requirements Specifications*

2. Deskripsi Keseluruhan

2.1 Deskripsi Produk

Feeder Bridge merupakan aplikasi berbasis website yang digunakan untuk melakukan pengiriman data dari SIAKAD ITS ke PDDikti. Aplikasi ini dibuat dengan tujuan dapat mempermudah proses pengiriman data menjadi lebih efektif dan efisien. Berbeda dengan aplikasi sebelumnya yang yang memerlukan pemrosesan data dari sisi *client* pada saat melakukan sinkronisasi dan pengiriman data, proses sinkonisasi dan pengiriman data aplikasi ini dilakukan oleh sistem antrian yang berjalan di server aplikasi sehingga tidak membebani internet pengguna.

2.2 Fungsi Produk

1. Membuat & menghapus akun pengguna
2. *Login*
3. *Logout*
4. Melakukan sinkronisasi data SIAKAD dan PDDikti
5. Melakukan pengiriman data dari SIAKAD ke PDDikti
6. Menampilkan perbandingan data SIAKAD dan *Feeder* PDDikti
7. Melakukan pengaturan hak akses data untuk setiap pengguna
8. Aplikasi dapat menampilkan *dashboard*

2.3 Penggolongan Karakterik Pengguna

Pengguna aplikasi ini adalah admin DPTSI ITS selaku administrator. Serta admin departemen selaku pengguna biasa. Aplikasi ini juga dapat digunakan oleh Dirpendik sebagai pengguna *dashboard* untuk memantau statistik pengiriman dan permasalahan pada data.

Tabel 1 Karakteristik Pengguna

Kategori Pengguna	**Tugas**	**Hak Akses ke aplikasi**
Administrator	Mengoperasikan dan mengontrol aplikasi secara menyeluruh	1, 2, 3, 4, 5, 6, 7, 8
User	Melihat tampilan data, melakukan sinkronisasi data, serta melakukan pengiriman data	2, 3, 4, 5, 6
User dashboard	Memantau *dashboard*	2, 3, 8

2.4 Lingkungan Operasi

Aplikasi yang dikembangkan berbasis web sehinga dapat dijalankan pada berbagai macam sistem operasi komputer. Pengguna hanya membutuhkan *browser* yang mendukung *JavaScript* dan *HTML5*

2.5 Batasan Desain dan Implementasi

Aplikasi *Feeder Bridge* dikembangkan dengan menggunakan bahasa pemrograman web *PHP famework CodeIgniter 4* serta *JavaScript*. Pemilihan bahasa pemrograman *PHP* dan *framework CodeIgniter 4* dikarenakan *framework* tersebut merupakan salah satu *framework* yang sangat umum digunakan dalam pengembangan web serta akrab digunakan dalam lingkungan pengembangan proyek DPTSI. Sedangkan pada sistem antrian menggunakan bahasa pemrograma Python dikarenakan *script Python* cocok digunakan *long time process* dikarenakan sistem antrian yang akan berjalan selama 24 jam. Selain itu *library* yang didukung untuk *Python* juga memadai untuk penggunaan sistem antrian. Untuk *database* aplikasi ini menggunakan *MySQL*. Aplikasi ini hanya dapat digunakan oleh admn DPTSI, admin/pengguna departemen, dirpendik sebagai

pengguna *dashboard*. Untuk mengaksesnya hanya diperlukan *browser* yang mendukung *JavaScript* dan *HTML5* serta *VPN* kusus untuk ITS. Pada aplikasi ini juga disediakan *dashboard* sebagai pemantauan untuk pengiriman data. Adapun teknologi yang digunakan untuk pembuatan *dashboard* menggunakan Microsoft Power BI

2.6 Dokumentasi Pengguna

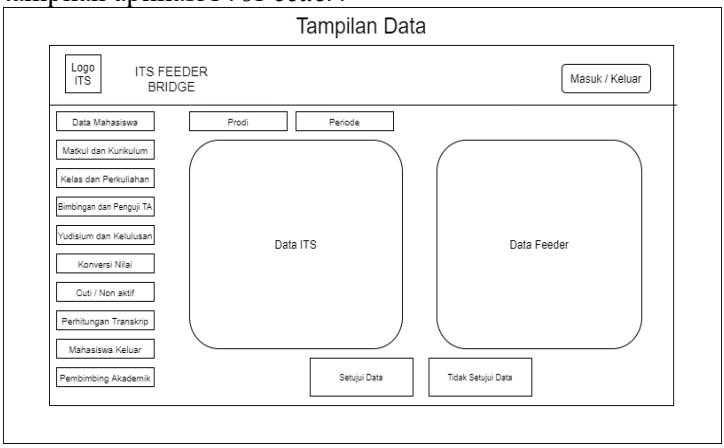
- *Guide-line* yang ada pada web

3. Kebutuhan Antarmuka Eksternal

3.1 *User Interface*

User Interface akan dikembangkan menggunakan desain grafis dan berbasis web. *User* dapat berinteraksi dengan aplikasi menggunakan web browser yang terkoneksi dengan internet. Aplikasi dapat menerima masukan dari user melalui pernitah dari keyboard, mouse. Keluaran dari aplikasi ini dapat *user* lihat secara langsung dari layar monitor secara langsung.

Adapun desain awal dari aplikasi ini dibuat menyerupai tampilan aplikasi *ProFeeder*.



3.2 *Hardware Interface*

Kebutuhan minimum perangkat keras untuk meggunakan aplikasi ini adalah berupa:

1. *PC standard*
2. *Keyboard*
3. *Mouse*

3.3 *Software Interface*

Untuk mengakses website *Feeder Bridge* tidak diperlukan browser khusus , semua jenis browser dapat digunakan untuk mengakses dan menggunakan web*site* selama mendukung *JavaScript* dan *HTML5*.

3.4 *Communication Interface*

Menggunakan standard komunikasi *HTTP (Hypertext Transfer Protocol)* yaitu sebuah protokol lapisan aplikasi untuk mentransmisi dokumen hypermedia seperti *html*. *HTTP*

mengikuti model *client* server diawali dengan *client* membuka sebuah koneksi membuat sebuah permintaan dan menunggu hingga mendapatkan respon.

4. *Functional Requirement*

ID	Kebutuhan Fungsional	Penjelasan
1	Memiliki fitur *login* & *logout*	Pengguna dan admin dapat melakukan *login* ke akun pengguna/admin dan keluar dari akunnya
2	Dapat menampilkan data SIAKAD dan *Feeder* PDDikti	Admin dan pengguna departemen melihat data SIAKAD dan PDDikti
3	Dapat melakukan sinkronisasi data SIAKAD dan PDDikti	Admin dan pengguna departemen dapat melakukan sinkronisasi data SIAKAD dan PDDikti
4	Dapat melakukan pengiriman data SIAKAD ke PDDikti	Admin dan pengguna departemen dapat melakukan pengiriman data dari SIAKAD ke PDDiti
5	Aplikasi dapat menampilkan *dashboard*	Admin dan pengguna *dashboard*/dirpendik dapat melihat *dashboard*
6	Dapat membuat akun pengguna	Admin dapat membuat akun pengguna
7	Dapat menghapus akun pengguna	Admin dapat mebghapus akun pengguna
8	Dapat mengubah akses data akun	Admin dapat mengubah akses akun pengguna

	pengguna	
9	Dapat menampilkan status proses antrian	Admin dan pengguna dapat melihat status antrian dari permintaan yang sudah dikirimkan
10	Dapat menampilkan status *error* pengiriman data	Admin dan pengguna dapat melihat status *error* pengiriman data

4.1 *Use Case Diagram*

4.2 Skenario *Use Case*

4.2.1. *Use Case Scenario: Login*

Nama *Use Case*	[UC01] *Login*	
Aktor	Admin DPTSI, pengguna departemen, pengguna *dashboard*	
Deskripsi	Pengguna *login* ke sistem dengan akun departemen	
Tujuan	Untuk masuk ke halaman utama sistem sebagai departemen	
Skenario Utama	Pengguna	Sistem
	1. Memasukkan *login credentials* di halaman *login*	
	2. Klik tombol *Login*	3. Validasi *login credentials*
		4. Menampilkan menu utama
Skenario Alternatif	2.1. Jika *login* credentials tidak sesuai maka pengguna tidak dapat masuk ke menu utama	

4.2.2. *Use Case Scenario*: Melihat Data

Nama *Use Case*	[UC02] Melihat Data	
Aktor	Admin DPTSI dan pengguna departemen	
Deskripsi	Pengguna melihat data yang ada di dalam ITS *Feeder Bridge* sesuai dengan departemen masing-masing	
Tujuan	Untuk melihat data yang ada di ITS *Feeder Bridge*	
Skenario Utama	Pengguna	Sistem
	1. Memilih menu data sesuai kebutuhan (mahasiswa, dosen, kelas, dll)	2. Menampilkan halaman dengan tabel kosong
	3. Memilih prodi dan periode semester sebagai filter data	4. Menampilkan data yang diminta ke dalam

		tabel sesuai prodi dan periode semester
Skenario Alternatif	4.1. Jika data yang diminta tidak ada di *database*, maka yang ditampilkan adalah data kosong	

4.2.3. *Use Case Scenario*: Sinkronisasi Data

Nama *Use Case*	[UC03] Sinkronisasi Data	
Aktor	Admin DPTSI dan pengguna departemen	
Deskripsi	Pengguna melakukan sinkronisasi data yang ada di dalam ITS *Feeder Bridge* sesuai dengan departemen masing-masing	
Tujuan	Untuk memperbarui data yang ada di ITS *Feeder Bridge* sesuai dengan data dari Siakad dan *Feeder* PDDikti	
Skenario Utama	Pengguna	Sistem
	1. Memilih prodi dan periode semester sebagai filter data yang akan disinkronisasi	
	2. Klik tombil "Sinkronisasi"	3. Menampilkan popup bahwa permintaan berhasil dikirim
		4. Mengirimkan permintaan sinkronisasi ke antrian dan menampilkan notifikasi sinkronisasi serta status permintaan pada menu proses data antrian

		5. Memproses permintaan sinkronisasi hingga selesai
Skenario Alternatif	3.1. Jika pengguna belum memilih filter data yang ingin disinkronisasi maka akan menampilkan popup bahwa permintaan ditolak/gagal	
	5.1. Jika permintaan gagal diproses pada sistem antrian maka sistem akan mengubah status menjadi gagal serta menampilkannya pada menu proses data antrian	

4.2.4. *Use Case Scenario*: Melihat Detail Data

Nama *Use Case*	[UC04] Melihat Detail Data	
Aktor	Admin DPTSI dan pengguna Departemen	
Deskripsi	Pengguna melihat detail data yang ada di dalam ITS *Feeder Bridge* sesuai dengan departemen masing-masing	
Tujuan	Untuk membandingkan kebenaran data yang ada di *Feeder* PDDikti dengan data dari Siakad melalui ITS *Feeder Bridge*	
Skenario Utama	Pengguna	Sistem
	1. Klik tombol "Detail" yang ada di setiap baris data	2. Menampilkan detail data yang dipilih
Skenario Alternatif	2.1. Jika data tidak ada di *database*, maka ditampilkan data kosong	

4.2.5. *Use Case Scenario*: Kirim Data

Nama *Use Case*	[UC05] Kirim Data
Aktor	Admin DPTSI dan pengguna departemen
Deskripsi	Pengguna melakukan pengiriman data dari ITS *Feeder Bridge* ke *Feeder* PDDikti sesuai dengan departemen masing-masing
Tujuan	Untuk mengirimkan data terbaru yang ada di ITS *Feeder Bridge* ke *Feeder* PDDikti

Skenario Utama	Pengguna	Sistem
	1. Memilih prodi dan periode semester sebagai filter data yang akan dikirim	
	2. Klik tombol "Kirim"	3. Menampilkan popup bahwa permintaan berhasil dikirim
		4. Mengirimkan permintaan kirim ke antrian dan menampilkan notifikasi kirim serta status permintaan pada menu proses data antrian
		5. Memproses permintaan kirim data hingga selesai
Skenario Alternatif	3.1. Jika pengguna belum memilih filter data yang ingin disinkronisasi maka akan menampilkan popup bahwa permintaan ditolak/gagal	
	5.1. Jika permintaan gagal diproses pada sistem antrian maka sistem akan mengubah status menjadi gagal serta menampilkannya pada menu proses data antrian	

4.2.6. *Use Case Scenario*: Melihat *dashboard*

Nama *Use Case*	[UC06] Melihat *dashboard*
Aktor	Admin DPTSI dan pengguna *dashboard*/Dirpendik

Deskripsi	Pengguna membuka *dashboard* pelaporan yang menunjukkan keadaan terkini pelaporan data ke PDDikti	
Tujuan	Untuk membuka *dashboard* dan menunjukkan keadaan terkini pelaporan data dari ITS *Feeder Bridge* ke *Feeder* PDDikti	
Skenario Utama	Pengguna	Sistem
	1. *Login* sebagai Dirpendik/pengguna *dashboard*	2. Validasi *login* credentials
		3. Menampilkan menu utama
	4. Memilih menu *dashboard*	5. Menghubungkan *Feeder Bridge* dengan sistem *dashboard*
		6. Menampilkan *dashboard*
	7. Memilih halaman *dashboard* yang diinginkan	8. Menampilkan halaman sesuai pilihan pengguna
Skenario Alternatif	2.1. Jika *login* credentials tidak sesuai maka pengguna tidak dapat masuk ke menu utama	
	4.1. Jika tidak *login* sebagai dirpendik, maka tidak dapat memilih menu *dashboard*	
	5.1. Jika koneksi *Feeder Bridge* dengan sistem *dashboard* tidak dapat dilakukan, maka *dashboard* tidak dapat ditampilkan	

4.2.7. *Use Case Scenario*: Membuat Akun Pengguna

Nama *Use Case*	[UC07] Membuat akun pengguna baru
Aktor	Admin DPTSI
Deskripsi	Admin DPTSI membuat akun pengguna baru
Tujuan	Untuk menambahkan akun pengguna yang akan digunakan oleh pengguna departemen atau admin lainnya

Skenario Utama	Admin DPTSI	Sistem
	1. Pilih menu ‘Manajemen Pengguna’	2. Menampilkan halaman manajemen pengguna dan list pengguna yang tersedia
	3. Pilih tombol ‘Tambah Pengguna’	4. Menampilkan halaman form registrasi pengguna
	5. Isi *user id, role*, dan *password* pada form	
	6. Klik ‘Daftarkan Akun’	7. Melakukan validasi data pengguna yang diinputkan
		8. Data pengguna disimpan didalam *database*
		9. Menampilkan halaman manajemen pengguna dengan tambahan pengguna baru pada daftar pengguna
Skenario Alternatif	8.1. Jika data pengguna yang diisi tidak lengkap atau sudah tersedia maka akan menampilkan pesan kesalahan pada halaman form registrasi	

4.2.8. *Use Case Scenario*: Menghapus Akun Pengguna

Nama *Use Case*	[UC08] Menghapus akun pengguna

Aktor	Admin DPTSI	
Deskripsi	Admin DPTSI menghapus pengguna	
Tujuan	Untuk menghapus pengguna yang telah dibuat	
Skenario Utama	Admin DPTSI	Sistem
	1. Pilih menu ‘Manajemen Pengguna’	2. Menampilkan halaman manajemen pengguna dan tabel list pengguna yang tersedia
	3. Pilih tombol ‘Hapus’ pada pengguna yang ingin dihapus	4. Menghapus data pengguna pada *database* b*erd*asarkan pengguna yang dipilih
		5. Menampilkan halaman manajemen pengguna/daftar pengguna
Skenario Alternatif		

4.2.9. *Use Case Scenario*: Mengubah Akses Akun Pengguna

Nama *Use Case*	[UC09] Mengubah akses prodi pengguna	
Aktor	Admin DPTSI	
Deskripsi	Admin DPTSI mengubah akses prodi pengguna	
Tujuan	Untuk menambahkan atau membatalkan akses prodi yang dimiliki oleh pengguna	
Skenario Utama	Admin DPTSI	Sistem
	1. Pilih menu ‘Manajemen Pengguna’	2. Menampilkan halaman manajemen pengguna dan

		list pengguna yang tersedia
	3. Pilih tombol ‘Ubah Akses’ pada baris data pengguna yang ingin diubah aksesnya	4. Menampilkan halaman ubah akses prodi yang menampilkan list program studi
	5. Klik check box menjadi centang pada prodi yang ingin ditambahkan atau sebaliknya	6. Menyimpan pengaturan akses prodi
Skenario Alternatif	6.1. Jika pengaturan gagal disimpan akan menampilkan alert gagal	

4.2.10. *Use Case Scenario*: Melihat Status Antrian

Nama *Use Case*	[UC10] Melihat daftar dan status antrian permintaan	
Aktor	Admin DPTSI dan pengguna departemen	
Deskripsi	Pengguna melihat daftar antrian dan status antrian permintaan seperti permintaan sinkronisasi dan kirim	
Tujuan	Untuk mengetahui status permintaan apakah belum, gagal, atau berhasil diproses	
Skenario Utama	Pengguna	Sistem
	1. Pilih menu ‘Proses Data Antrian’	2. Menampilkan halaman list status antrian permintaan
Skenario Alternatif		

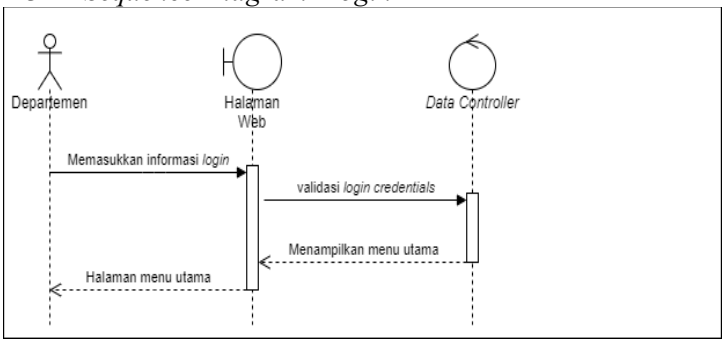
4.2.11. *Use Case Scenario*: Melihat Daftar *Error* Pengiriman Data

Nama *Use Case*	[UC11] Melihat daftar *error* pengiriman data yang gagal
Aktor	Pengguna departemen & admin DPTSI
Deskripsi	Pengguna melihat daftar *error*/pengiriman data yang gagal

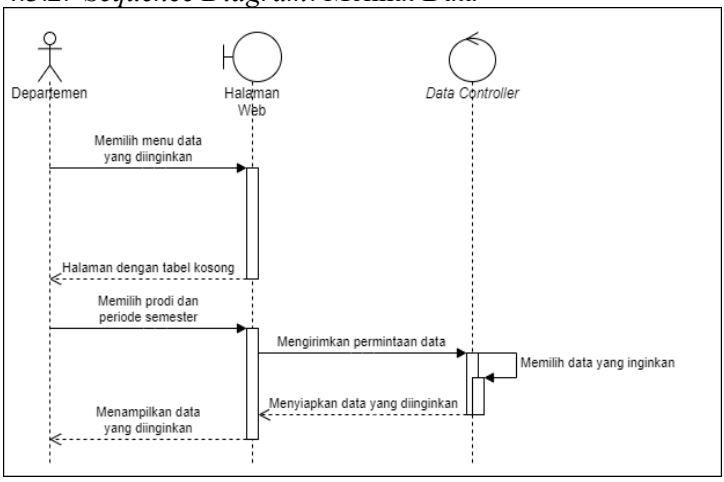
Tujuan	Untuk status/pesan *error* dari pengiriman yang gagal	
Skenario Utama	Pengguna	Sistem
	1. Pilih menu '*Error* Pengiriman Data'	2. Menampilkan halaman list *error* pengiriman datas
Skenario Alternatif		

4.3 *Sequence Diagram*

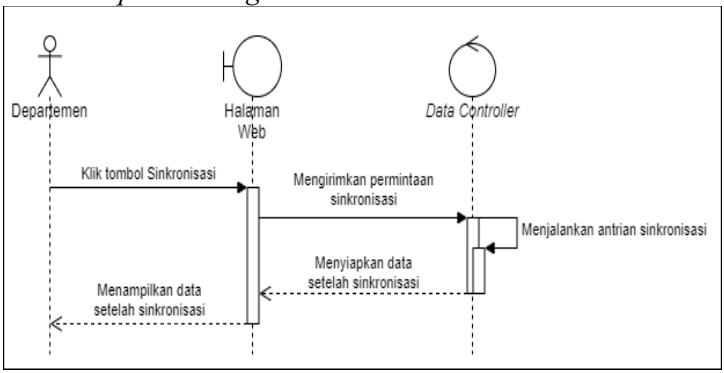
4.3.1. *Sequence Diagram*: *Login*



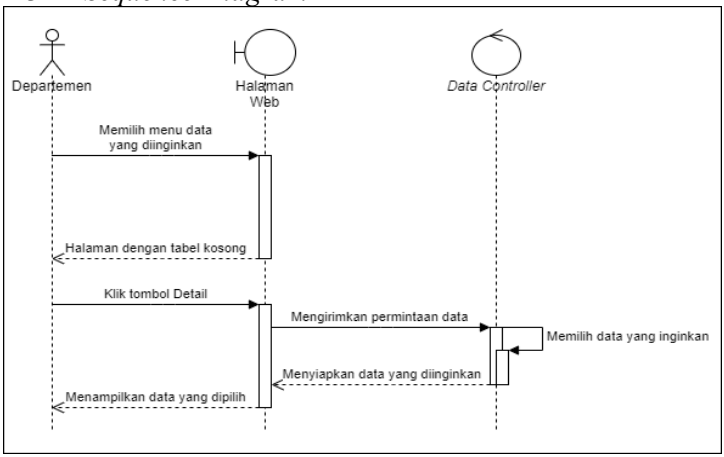
4.3.2. *Sequence Diagram*: Melihat Data



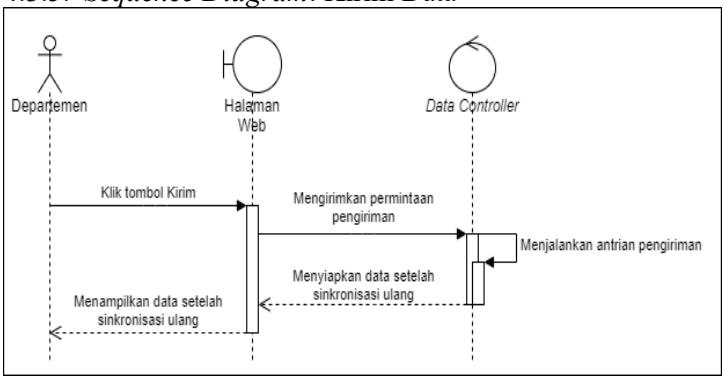
4.3.3. *Sequence Diagram*: Sinkronisasi Data



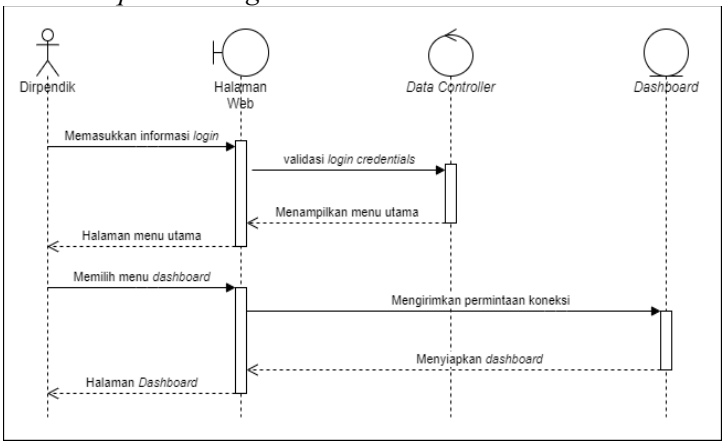
4.3.4. *Sequence Diagram*: Melihat Detail Data



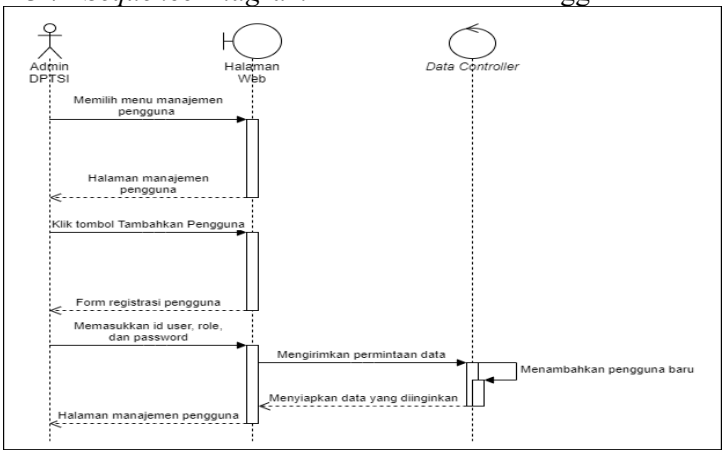
4.3.5. *Sequence Diagram*: Kirim Data



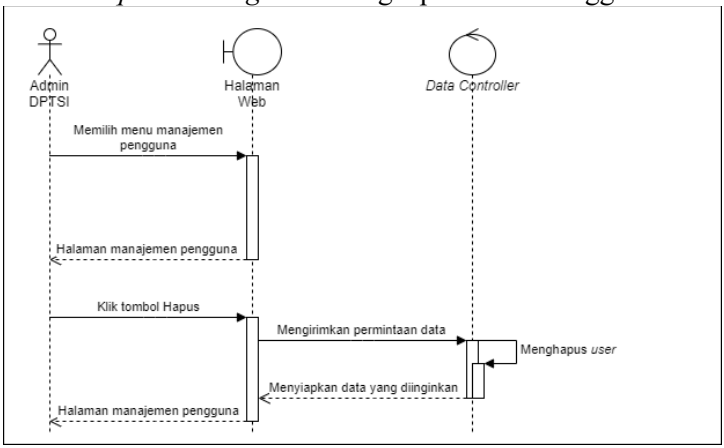
4.3.6. *Sequence Diagram*: Melihat *dashboard*



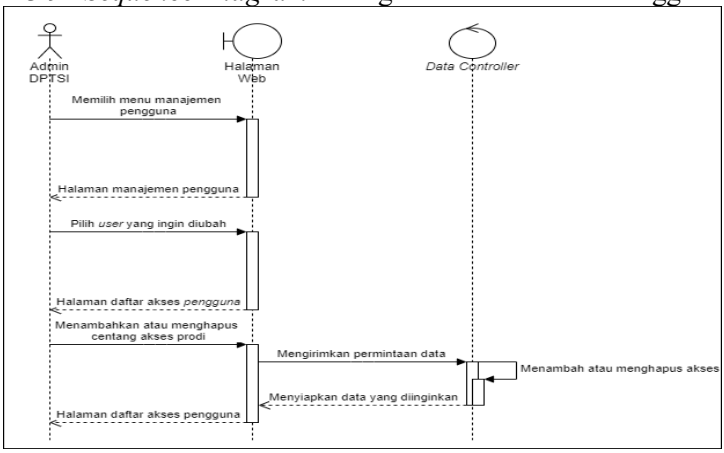
4.3.7. *Sequence Diagram*: Membuat Akun Pengguna



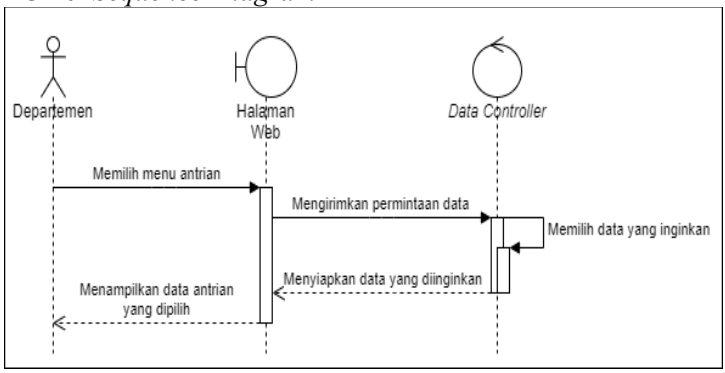
4.3.8. *Sequence Diagram*: Menghapus Akun Pengguna



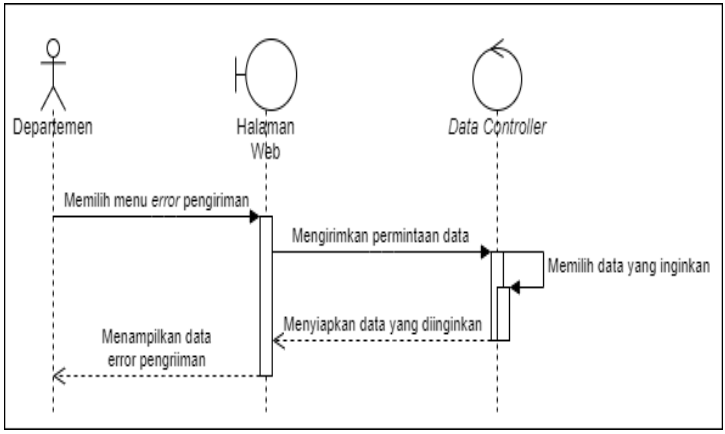
4.3.9. *Sequence Diagram*: Mengubah Akses Akun Pengguna



4.3.10. *Sequence Diagram*: Meluhat Status Antrian



4.3.11. *Sequence Diagram*: Melihat Daftar *Error* Pengiriman Data

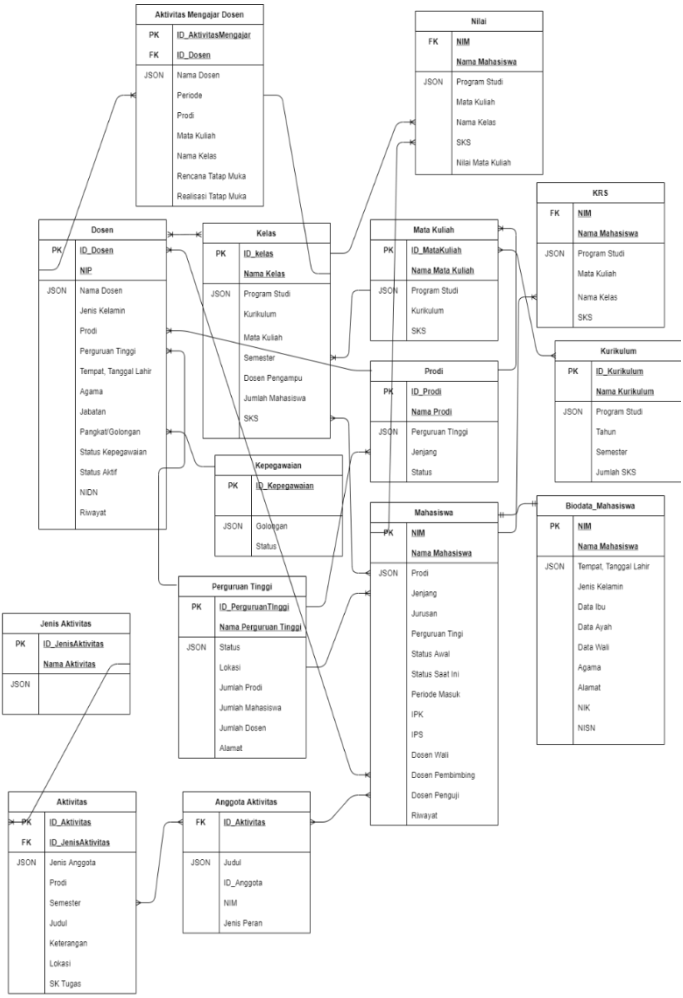


4.4 *Entity Relationship Diagram & Database Schema*

Dalam pengembangan *Entity – Relationship Diagram* (*ERD*) ada beberapa perubahan dalam pengembangannya. T*erd*apat 3 versi *ERD* sebelum akhirnya mejadi versi final dan yang digunakan dalam aplikasi.

4.3.1. *ERD* Versi Pertama

Pengajuan *ERD* pertama sebelum mendapat respon pengembangan dapat dilihat pada gambar xx berikut.



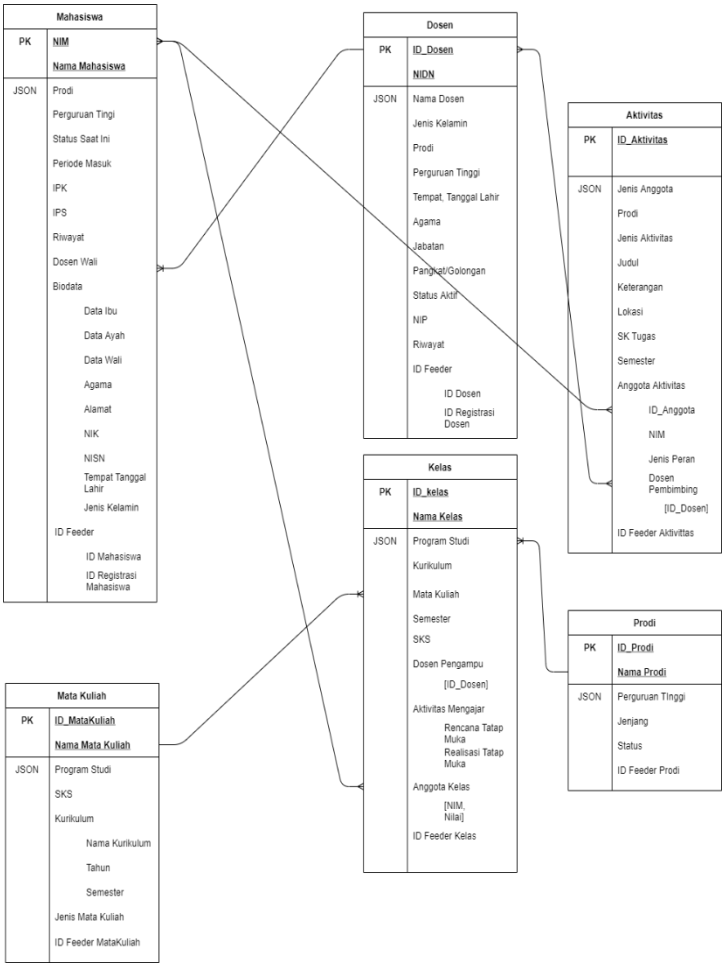
Entitas dari *ERD* pertama adalah sebagai berikut :

1. Perguruan Tinggi
2. Prodi (Program Studi)
3. Kurikulum
4. Mata kuliah
5. Kelas
6. KRS
7. Mahasiswa

8. Biodata Mahasiswa
9. Dosen
10. Aktivitas mengajar dosen
11. Nilai
12. Aktivitas
13. Anggota aktivitas
14. Jenis aktivitas

4.3.2. *ERD* Versi Kedua

Setelah mendapat respon pengembangan dan dilakukan pencocokan data antara *ERD* dan sumber data asli (SIAKAD dan *Feeder*), dibuat *ERD* versi kedua yang menerapkan *document-based database*. Hasil *ERD* versi kedua dapat dilihat pada gambar berikut.



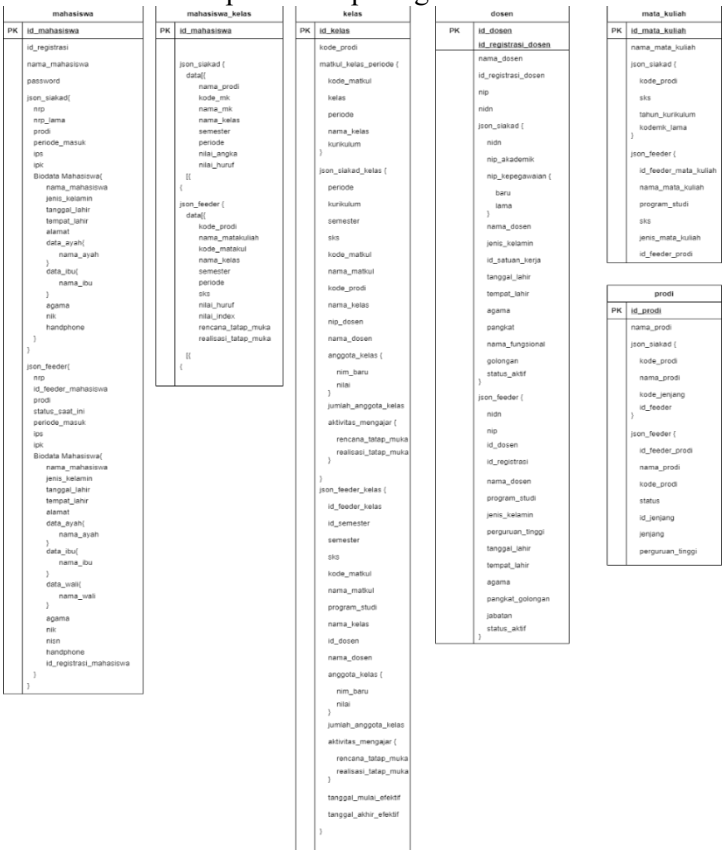
Entitas dari *ERD* versi kedua berkurang menjadi hanya 6 entitas yaitu:

1. Mahasiswa
2. Dosen
3. Kelas
4. Prodi
5. Mata kuliah
6. Aktivitas

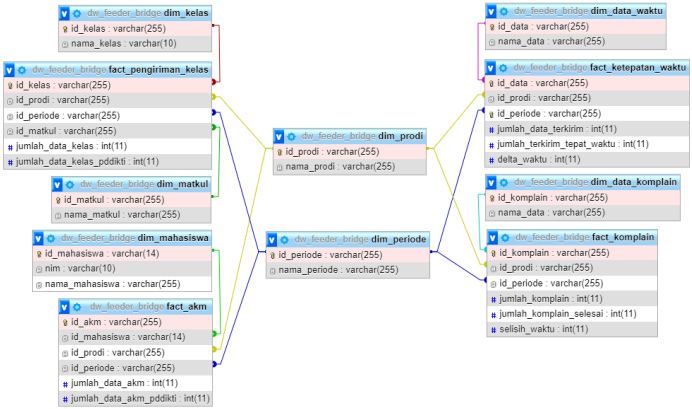
4.3.3. *ERD* Versi Final

Setelah mengembangkan aplikasi b*erd*asarkan *ERD* versi kedua, terdapat beberapa perubahan *ERD* karena perbedaan antara sumber data (SIAKAD) dan *Feeder*.

Hasil *ERD* final dapat dilihat pada gambar berikut.



4.3.4. Skema *Feeder Bridge Data Warehouse*



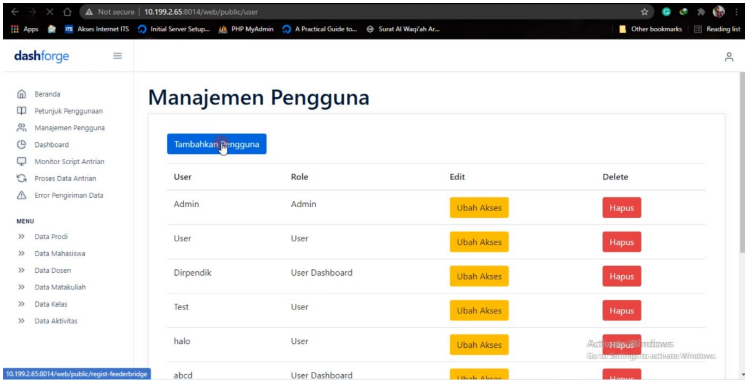
5. *Non Functional Requirements*

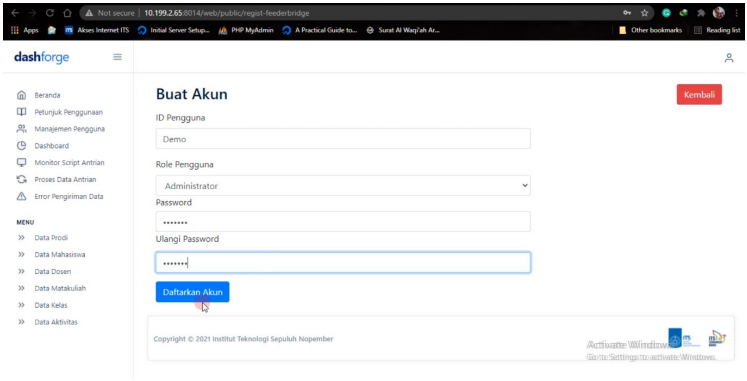
ID	Parameter	Kebutuhan
	Availability	Aplikasi dapat diakses setiap hari selama 24 jam Sistem antrian dapat berjalan selama 24 jam
	Reliability	
	Ergonomy	
	Portability	Aplikasi dapat dijalankan pada berbagai jenis sistem operasi komputer. Aplikasi dapat dijalankan pada berbagai jenis spesifikasi komputer. Aplikasi dapat dijalankan pada segala jenis *browser* yang mendukung *JavaScript* dan *HTML5*.
	Memory	Aplikasi tidak memberikan beban

		memori besar pada komputer pengguna
	Response time	
	Safety	
	Security	Memiliki keamanan akun berupa kata sandi/password Penggunaan *VPN* ITS untuk mengakses aplikasi

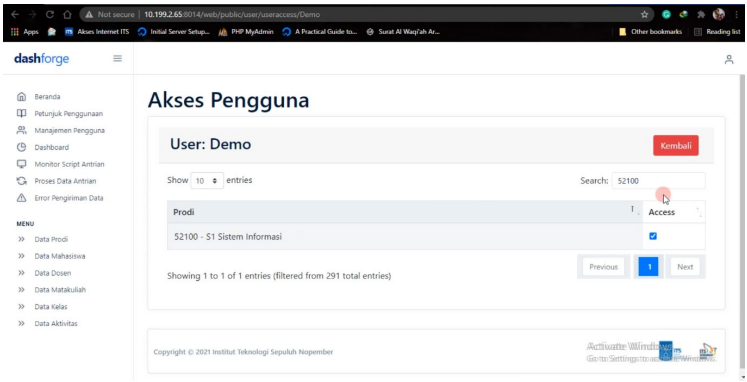
LAMPIRAN F
DOKUMENTASI PENGUJIAN FUNGSIONAL

Manajemen Pengguna

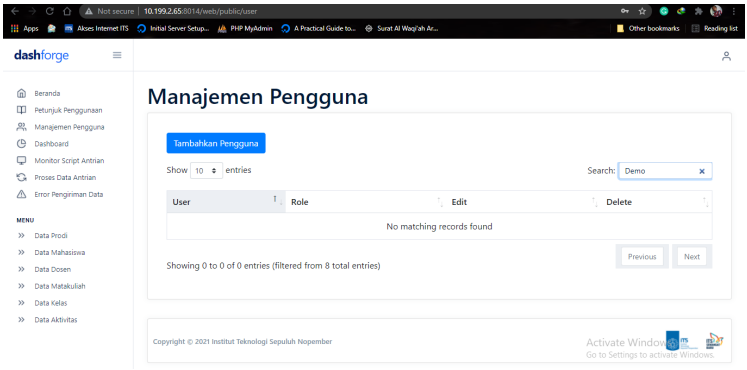




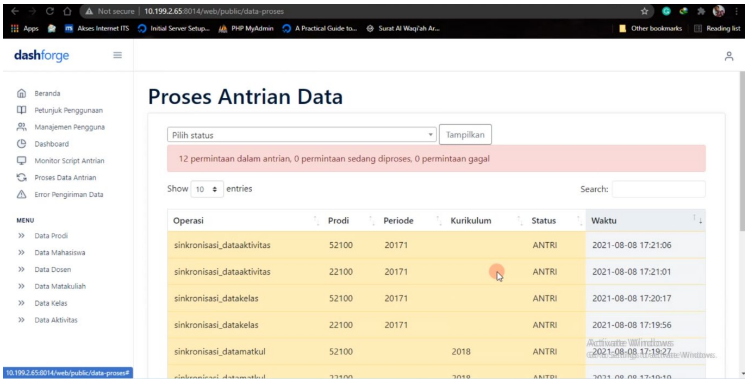
Tambah akun



Edit hak akses

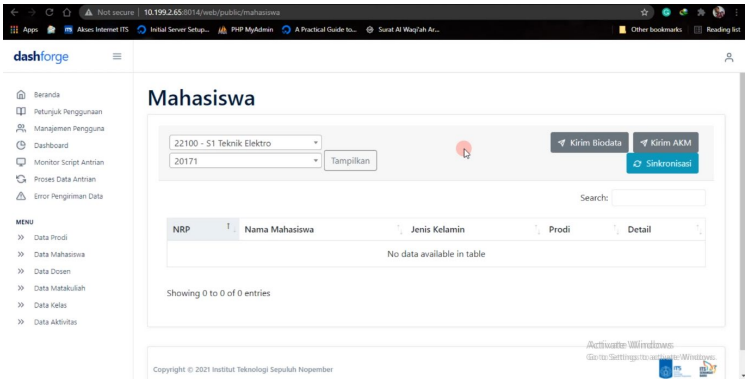


Hapus akun

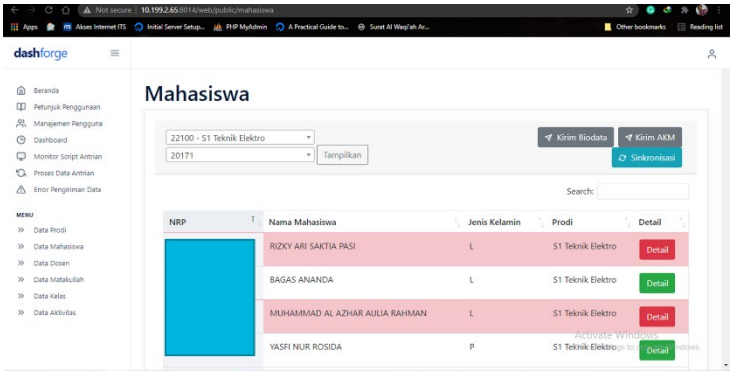


Membuat *event* antrian

Data Mahasiswa

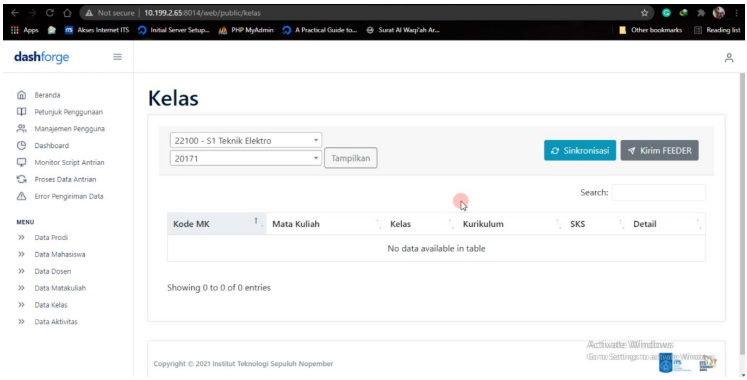


Sebelum sinkronisasi data mahasiswa

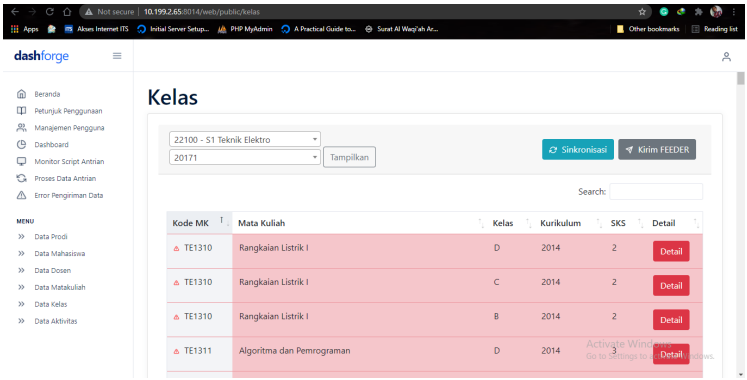


Sesudah sinkronisasi data mahasiswa

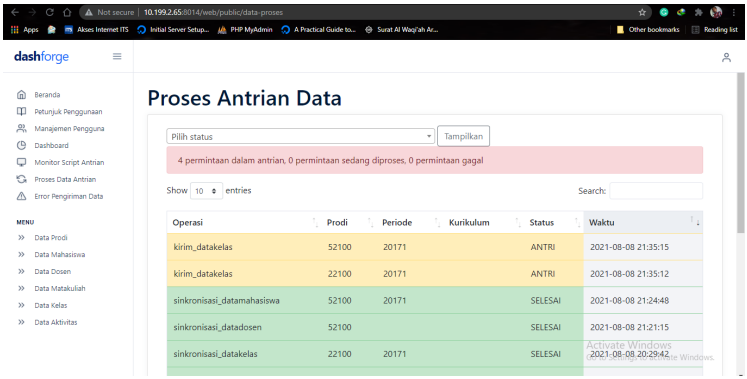
Data Kelas



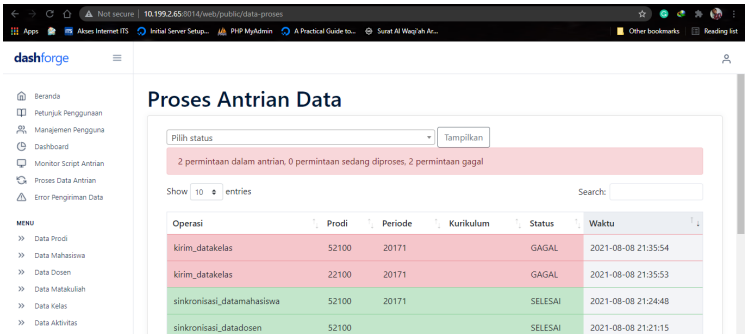
Sebelum sinkronisasi data kelas



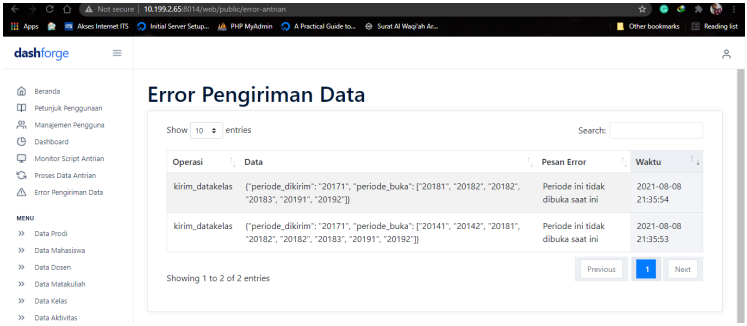
Sesudah sinkronisasi data kelas



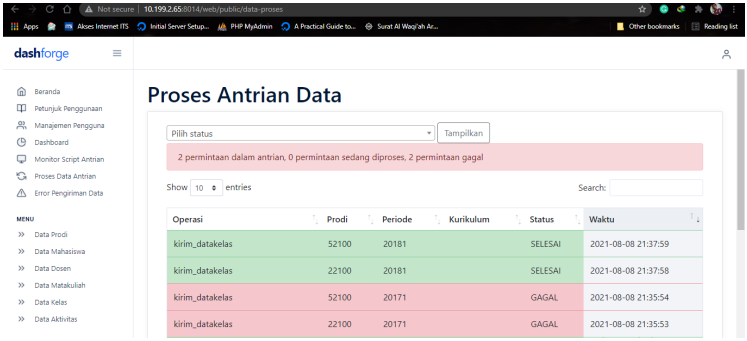
Kirim data kelas



Gagal kirim data kelas



Tampilan *error* kirim data kelas



Selesai kirim data kelas

DAFTAR PUSTAKA

[1] T. D. P. T. R. I. MENTERI RISET, PERATURAN MENTERI RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI REPUBLIK INDONESIA NOMOR 61 TAHUN 2016 TENTANG PANGKALAN DATA PENDIDIKAN TINGGI, Jakarta, 2016.

[2] "Pangkalan Data Pendidikan Tinggi," Kemendikbud, [Online]. Available: https://pddikti.kemdikbud.go.id/roadmap_pd_dikti. [Accessed 29 10 2020].

[3] R. P. Wibowo, Interviewee, *Urgensi Feeder-Brige untuk ITS.* [Interview]. Oktober 2020.

[4] R. I. Perwira and B. Santosa, "IMPLEMENTASI WEB SERVICE PADA INTEGRASI DATA AKADEMIK DENGAN REPLIKA PANGKALAN DATA DIKTI," *TELEMATIKA: Jurnal Informatika dan Teknologi Informasi,* vol. 14, no. 1, 2017.

[5] S. Widodo, H. Brawijaya, Samudi and E. Retnoningsih, "Integrasi Data Akademik Dengan Aplikasi Feeder PDDIKTI Berbasis Web service," *BINA INSANI ICT Journal,* vol. 5, 2018.

[6] E. W. Kenali and H. Fathoni, "DESAIN DAN IMPLEMENTASI SERVICES PROVIDER BERBASIS WEB SERVICES PUSH PANGKALAN DATA PERGURUAN TINGGI PADA SISTEM INFORMASI AKADEMIK POLITEKNIK NEGERI LAMPUNG," 2014.

[7] M. Khotib, "Pembuatan Sistem Absensi Berbasis Web dengan Menerapkan Teknologi Progressive Web Apps dan Metode Action Design Research (Studi Kasus : SMP Al Azhar 13 Surabaya)," *Institut Teknologi Sepuluh Nopember, Surabaya,* 2019.

[8] S. Few, Information Dashboard Design, O'Reilly, 2006.

[9] A. Janes, A. Silliti and G. Succi, "Effective Dashboard Design," *CUTTER IT JOURNAL,* vol. 26, no. 1, 2013.

[10] K. E. Kendall and J. E. Kendall, SYSTEMS ANALYSIS AND DESIGN, Camden, New Jersey: Pearson, 2020.

[11] N. B. Ruparelia, "Software Development Lifecycle," *ACM SIGSOFT Software Engineering Notes,* vol. 35, no. 3, pp. 8-13, 2010.

[12] S. S. Nathan, S. S. Mohan, A. R. Harudas and N. Kashif, "BERKELEYINTERNET NAME DOMAIN(BIND)," *International Journal on Cybernetics & Informatics ,* vol. 1, no. 1, 2012.

[13] A. R. Hevnet, S. T. March and J. Park, "DESIGN SCIENCE IN INFORMATION," *MIS Quarterly,* vol. 28, no. 1, pp. 75-105, 2004.

[14] "CDC - Change Data Capture," Berca Hardayaperkasa, [Online]. Available: https://www.berca.co.id/cdc-change-data-capture. [Accessed 14 01 2021].

[15] "About Change Data Capture (SQL Server)," Microsoft, 14 01 2019. [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/track-changes/about-change-data-capture-sql-server?view=sql-server-ver15. [Accessed 14 01 2021].

[16] R. B. Pojasek, "Asking "Why?" Five Times," *Environmental Quality Management,* p. 79, 2000.

[17] K. R. T. d. P. Tinggi, User Guide Web Service Versi 2.2, Kementrian Riset Teknologi dan Pendidikan Tinggi.

[18] I. C. Society, "IEEE Recommended Practice for Software Requirements Specifications," The Institute of Electrical and Electronics Engineers, New York, USA, 1998.

[19] F. D. Cahyo, "PEMBUATAN DASHBOARDPEMANTAUAN ABSENSI DAN JURNAL SEKOLAH DENGAN METODE ACTION RESEARCH(STUDI KASUS: SMPIA 13

SURABAYA)," *Institut Teknologi Sepuluh Nopember, Surabaya,* 2020.

[20] R. Peters, "cron," in *Expert Shell Scripting*, Berkeley, CA, Apress, 2009, p. 81.

[21] T. d. P. T. Sekretaris Jendral Kementrian Riset, "Keputusan Sekretaris Sekretaris Jendral Kementrian Riset, Teknologi, dan Pendidikan Tinggi Tentang Standar Pengelolaan Pangkalan Data Pendidikan Tinggi," 2018.

[22] R. Kimbal, Kimball Dimensional Modeling Techniques, Kimbal Group.

[23] P. Vassiliadis, A. Simitsis and S. Skiadopoulos, "Conceptual modeling for ETL processes," in *DOLAP '02: Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, New York, NY, United States, 2002.

[24] J. Nielsen and R. Budiu, "Nielsen Norman Group: Success Rate: The Simplest Usability Metric," Nielsen Norman Group, 17 February 2001. [Online]. Available: https://www.nngroup.com/articles/success-rate-the-simplest-usability-metric/. [Accessed 12 June 2021].

[25] S. Balaji and D. M. S. Murugaiyan, "WATERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC," *International Journal of Information Technology and Business Management,* vol. 2, no. 1, 2012.

BIODATA PENULIS

Penulis dari penelitian dalam tugas akhir ini terdiri dari tiga orang. Berikut biodata masing-masing penulis.

Prasetyo Ramadi

Penulis ini merupakan seorang anak tunggal yang lahir di Surabaya pada tanggal 1 Januari. Penulis menempuh pendidikan formal di SDN 1 Sumberrejo, SMPN 1 Bojonegoro, dan terakhir di SMAN 1 Bojonegoro hingga lulus pada tahun 2017.

Selanjutya penulis menempuh pendidikan tinggi di Institut Teknologi Sepuluh Nopember (ITS) Surabaya di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC). Selama menjadi mahasiswa, penulis mengikuti Latihan Keterampilan Manajemen Mahasiswa (LKMM) dari jenjang Pra-Tingkat Dasar hingga aktif menjadi Pemandu LKMM ITS. Dalam kegiatan ormawa, penulis menjadi staf di Departemen *Social Development* HMSI periode 2018, menjadi staf *Roadshow ISE* 2018, dan staf Badan Koordinasi Pemandu BEM FTIK periode 2019. Selain menjadi pengurus ormawa, penulis juga mengikuti perlombaan di bidang kewirausahaan dan pernah menjadi juara dua di tingkat institut dengan mengusung ide jasa servis gadget *online*. Selain itu, penulis juga mengikuti Kompetisi Bisnis Mahasiswa Indonesia (KBMI) dan sempat mendapatkan pendanaan unutk merealisasikan ide bisnis di bidang agrikultur dengan sistem penyiraman otomatis.

Perjalanan penulis selama menjadi mahasiswa dihiasi dengan pengalaman menjadi mahasiswa magang di Direktorat Pengembangan Teknologi dan Sistem Informasi ITS dan mengerjakan sistem pelaporan PDDikti bersama Rizal Maulana Hadi dan Muhammad Alwan Wicaksono yang diharapkan dapat

digunakan oleh ITS. Ketertarikan penulis terhadap penggalian kebutuhan pengguna, pengubahan data menjadi informasi yang kemudian disampaikan dengan dashboard, telah membawa penulis untuk mengambil pilihan laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI). Untuk lebih jelasnya, penulis dapat dihubungi melalui email di prasetyoramadi@gmail.com.

Rizal Maulana Hadi

Penulis lahir di Surabaya, Jawa Timur pada tanggal 8 Desember 1998. Merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu; SDN Klampis Ngasem 1 Surabaya, SMP Negeri 19 Surabaya, dan SMA Negeri 2 Surabaya.

Pada Tahun 2017 pasca kelulusan SMA, penulis melanjutkan pendidikan di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211740000054. Selama menjadi mahasiswa, penulis aktif mengikuti berbagai kegiatan organisasi seperti staf dan staf ahli Departemen *Student Welfare* HMSI FTIK 2018 dan 2019. Di kegiatan kepanitiaan, penulis pernah menjadi staf *Roadshow ISE* 2018 dan staf ahli *Roadshow ISE* 2019. Selain di dalam kampus, penulis juga pernah mengikuti magang di Direktorat Pengembangan Teknologi Informasi dan Komunikasi ITS dan Direktorat Kemitraan Global ITS.

Pada tahun keempat, karena penulis memiliki ketertarikan di bidang pengembangan dan implementasi database, maka penulis mengambil bidang minat Akuisisi Data dan Diseminasi

Informasi (ADDI). Penulis dapat dihubungi melalui email di rizalmaulana6@gmail.com.

Muhammad Alwan Wicaksono

Penulis lahir di Surabaya, Jawa Timur pada tanggal 26 Maret 1999. Merupakan anak pertama dari 2 bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu; SDI Al-Azhar Kelapa Gading Surabaya, SMP Negeri 19 Surabaya, dan SMA Negeri 1 Surabaya.

Pada Tahun 2017 pasca kelulusan SMA, penulis melanjutkan pendidikan di Departemen Sistem Informasi Fakultas Teknologi Elektro dan Informatika Cerdas - Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 05211740000111. Selama menjadi mahasiswa, penulis aktif mengikuti berbagai kegiatan organisasi seperti menjadi staf Departemen Riset dan Teknologi HMSI FTIK 2018 pada tahun kedua sebagai mahasiswa, serta menjadi ketua Departemen Riset dan Teknologi HMSI FTIK 2019 pada tahun ketiga sebagai mahasiswa. Penulis juga aktif menjadi pengurus Lembaga Dakwah KISI sebagai staf Dana Kreatif pada tahun 2018 atau tahun kedua sebagai mahasiswa. Pada kegiatan kepanitiaan, penulis pernah menjadi staf *IT Development ISE* 2018 pada tahun kedua sebagai mahasiswa, serta menjadi staf ahli *IT Development ISE* 2019 pada tahun ketiga menjadi mahasiswa. Selain itu penulis juga aktif dalam perlombaan keilmiahan dimana penulis pernah menjadi finalis dalam ajang GEMASTIK 2019 divisi Karya Tulis Ilmiah TIK dengan topik Privasi Data Pengguna pada Aplikasi Mobile.

Penulis memiliki ketertarikan pada bidang pengembangan web dan telah terlibat dalam beberapa proyek pengembangan web

seperti pengembangan web *event ISE* 2018 dan 2019. Selain itu penulis juga mengikuti magang di Direktorat Pengembangan Teknologi Informasi dan Komunikasi ITS untuk mengerjakan proyek sistem pelaporan PDDikti berbasi web yang juga menjadi produk dalam Tugas Akhir ini.

Pada tahun keempat, karena penulis memiliki ketertarikan di bidang pengolahan data, penulis mengambil bidang minat laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis dapat dihubungi melalui *email* di muhammadalwan.52@gmail.com.