ANALISIS DAN RANCANG SISTEM OTOMATIS *DATABASE SCHEMA MIGRATION* BERBASIS GITOPS PADA PROSES CI/CD PADA PERUSAHAAN *FINTECH*

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat Kelulusan
Program Pendidikan Sarjana

Oleh:
Cristian Stevanus Oloan
2020110021

JURUSAN SISTEM INFORMASI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER-LIKMI
BANDUNG
2025

ANALISIS DAN RANCANG SISTEM OTOMATIS *DATABASE SCHEMA MIGRATION* BERBASIS GITOPS PADA PROSES CI/CD PADA PERUSAHAAN *FINTECH*

Oleh:
Cristian Stevanus Oloan
2020110021

Bandung, 28 Februari 2025
Menyetujui,

Yusup Jauhari Shandi, S.Kom., M.Kom.	Kezia Stefani, S.T., M.Kom.
Pembimbing	Ketua Jurusan

JURUSAN SISTEM INFORMASI
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER-LIKMI
BANDUNG
2025

ABSTRAK

PT. XYZ adalah salah satu bursa aset Kripto di Indonesia yang menyediakan *platform* untuk dapat melakukan investasi terhadap aset Kripto yang sudah terpercaya dan sudah mendapatkan legalitas dari BAPPEBTI. Dalam proses bisnis yang dijalankan oleh PT. XYZ , *engineer* selaku *developer* aplikasi melakukan praktik *database schema migration* secara manual yang dilakukan cukup sering, yang setidaknya dalam kurun waktu satu minggu terdapat satu aktivitas untuk melakukan praktik tersebut. Hal tersebut perlu mendapatkan pertimbangan, dikarenakan PT. XYZ selaku perusahaan *FinTech* yang menyediakan jasa *financial* dalam bentuk teknologi. Terutama, praktik dalam mengelola *database* yang dilakukan secara manual harus dihindari. Dampak dari praktik tersebut tidak adanya *track record* atas *database schema migration* yang telah dilakukan, dan maka daripada itu tidak adanya juga data untuk dijadikan laporan, semisalnya ada audit. Berdasarkan hal tersebut, direncanakan sistem aplikasi yang menerapkan metodologi GitOps, yang menjadikan Git sebagai sumber satu-satunya kebenaran terkait kondisi infrastruktur termasuk *database*, dan melakukan *trigger* secara otomatis menjalankan praktik *database schema migration* disaat adanya perubahan terkait *database schema* yang telah dilampirkan pada GitHub *repository*. Sistem aplikasi yang dirancang berdasarkan pemodelan dari *Unfiled Modeling Language* (UML) yaitu *use case diagram, activity diagram,* dan *sequence diagram* dengan pendekatan berbasis objek. Python dan Vue JS merupakan bahasa pemrograman yang digunakan dalam perancangan sistem ini, dan menghasilkan sistem aplikasi berbasis *website*. Dengan adanya sistem ini, dapat menjawab persoalan mengenai praktik yang perlu dipertimbangkan sebelumnya dengan lebih efisien dan tercatat segala aktivitas *database schema migration*.

Kata kunci: Sistem Informasi, *Database Schema Migration*, *Prototype*, CI/CD, *Python*, *Automation*, *FinTech*, *Vue JS*.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada ke hadirat Tuhan Yang Maha Esa yang telah melimpahkan segala kasih dan karunia-Nya kepada kita, sehingga penulis dapat menyelesaikan skripsi dengan semaksimal mungkin. Alasan penulis membuat skripsi, adalah sebagai syarat untuk kelulusan pendidikan sarjana pada jurusan Manajemen Informatika, Sekolah Tinggi Manajemen Informatika dan Komputer - LIKMI. Judul skripsi yang dibuat yaitu "Analisis dan Rancang Sistem Otomatis *Database Schema Migration Berbasis* GitOps Pada Proses CI/CD Pada Perusahaan *Fintech*".

Dalam penyusunan skripsi, penulis telah melibatkan berbagai pihak yang begitu sangat membantu dalam penyusunan skripsi. Oleh karena itu, disini penulis menyampaikan rasa terimakasih yang begitu banyak kepada:

1. Tuhan Yang Maha Esa, karena berkat bantuan dan kasih sayang-Nya, penulis dapat menyelesaikan skripsi.
2. Bapak Yusup Jauhari Shandi, S.Kom., M.Kom., selaku dosen pembimbing yang telah menyempatkan waktu dan tenaga untuk membimbing saya dalam menyusun skripsi. Selain daripada itu, beliau juga telah memberikan saran dan ide yang dapat membantu menyempurnakan skripsi ini.
3. Kedua orang tua penulis, yang telah memberikan banyak dukungan dan doa dalam penyusunan laporan praktek kerja lapangan.
4. Kepada PT. XYZ sebagai perusahaan tempat saya bekerja saat ini, yang telah memperbolehkan saya untuk melakukan penelitian dan perancangan sistem aplikasi untuk pengerjaan skripsi ini.

Penutup kata dari penulis untuk pembaca, semoga hasil dari analisa dan rancang sistem aplikasi yang telah saya lakukan dapat bermanfaat bagi pembaca, dan juga bagi perusahaan PT. XYZ agar dapat mencapai tujuan yang meningkatkan performa bisnis.

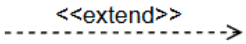
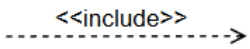
DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

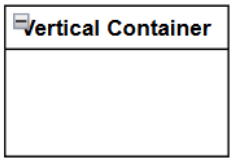
DAFTAR SIMBOL

1. *Use Case Diagram*

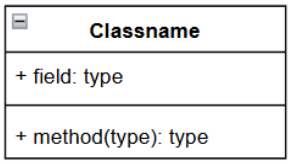
No.	Simbol	Nama	Keterangan
1		*Actor*	Pihak yang berhubungan langsung dengan sistem, dengan memberikan suatu informasi kepada sistem atau bahkan juga mendapatkan informasi dari sistem
2		*Use case*	Penggambaran secara abstrak mengenai proses fungsional yang terdapat pada sistem dan berhubungan langsung dengan *actor*.
3		*Association*	Penghubung antara *actor* dengan *use case* yang menandakan adanya interaksi yang dilakukan oleh *actor* terhadap sistem.
4		*Extend*	Menunjukkan *use case* yang dapat dieksekusi jika *use case* sebelumnya terpenuhi, maka dapat dilanjutkan ke *use case* selanjutnya.
5		*Include*	Menunjukkan hubungan berkelanjutan dari *use case* sebelumnya dan menandakan *use case* selanjutnya merupakan bagian dari *use case* sebelumnya.

2. *Activity Diagram*

No.	Simbol	Nama	Keterangan
1		*Start*	Menentukan titik awal suatu aktivitas

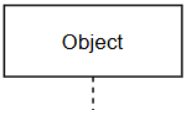
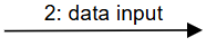
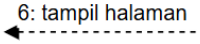
No.	Simbol	Nama	Keterangan
2		*Activity state*	Penggambaran aktivitas yang dilakukan
3		*Decision*	Adanya percabangan kondisional yang melibatkan opsi aktivitas yang lebih dari satu
4		*Contra flow*	Menunjukkan alur aktivitas
5		*End*	Menandakan titik akhir dari suatu aktivitas
6		*Swimlane*	Jalur aktivitas yang dikelompokkan berdasarkan entitas agar dapat memudahkan mengenali aktor yang menjalankan aktivitas

3. *Class Diagram*

No.	Simbol	Nama	Keterangan
1		*Class*	Merupakan blok yang pada umumnya digunakan pada pemrograman berbasis objek. *Class* terdapat komponen *class name*, *attribute* dan juga *method* yang menggambarkan *class* tersebut.
2		*Association*	Hubungan antara 2 *class* dengan mencantumkan multiplisitas, seperti *one-to-many*, *one-to-one* dan lain-lain.
3		*Inheritance*	Menunjukkan adanya pewarisan dari *parent class* dan *child class* sehingga *attribute* dan *method* yang

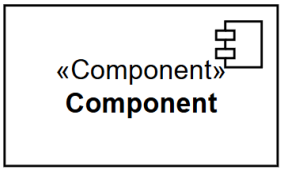
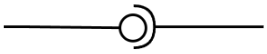
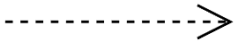
No.	Simbol	Nama	Keterangan
			dimiliki oleh *parent class* maka dimiliki juga oleh *child class*. Anak panah diarahkan ke salah satu *class* menandakan *class* tersebut merupakan *parent class*

4. *Sequence Diagram*

No.	Simbol	Nama	Keterangan
1		*Actor*	Peran yang menggambarkan entitas berinteraksi langsung dengan sistem
2		*Lifeline*	Penggambaran dari aktivitas suatu objek
3		*Object*	Mewakili dari suatu objek pada suatu sistem, dan dapat merepresentasikan juga sebagai suatu *class*
4		*Activation Box*	Merepresentasikan durasi waktu yang dilakukan dalam melakukan suatu operasi dari aktivitas
5		*Message input*	Komunikasi input yang dapat bermula dari *actor* atau *object* ke *object* lainnya.
6		*Message return*	Komunikasi balasan berupa output dalam bentuk *respons* dari *message input*

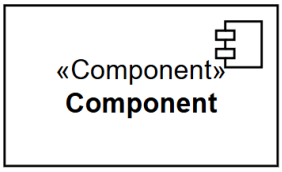
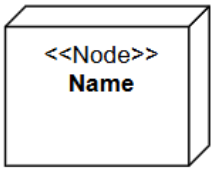
No.	Simbol	Nama	Keterangan
7		*Self message*	Menjalankan aktivitas pada objek dirinya sendiri.
8		*Alternative fragments*	Pengelompokkan dari suatu aktivitas tambahan sebagai alternatif dari aktivitas utama. Untuk dapat menjalankan aktivitas ini perlu memenuhi kondisi tertentu.

5. *Component Diagram*

No.	Simbol	Nama	Keterangan
1		*Component*	Merepresentasikan *block logic* dari salah satu *class*, *modul*, perangkat keras, atau sistem.
2		*Interface*	Simbol ini menunjukkan jembatan antar *component* melalui antarmuka. *Interface* terdapat dua jenis, *required interface* dan *provided interface*. *Required interface* disimbolkan dengan setengah lingkaran yang diartikan untuk menghubungkan ke *component* diperlukan antarmuka, sedangkan *provided interface* disimbolkan dengan lingkaran penuh yang diartikan *component* tersebut menyediakan antarmuka.
3		*Dependency*	Menjelaskan adanya ketergantungan dari *component* terhadap *component* lainnya. Ketergantungan yang dimaksud merupakan *component* memerlukan suatu hal yang berasal dari

No.	Simbol	Nama	Keterangan
			component tersebut
4		*Association*	Menghubungkan antar *component*
5		*Port*	Titik koneksi yang dilewati oleh *component* untuk dapat berkomunikasi dengan *component* tujuan.

6. *Deployment Diagram*

No.	Simbol	Nama	Keterangan
1		*Component*	Merepresentasikan *block logic* dari salah satu *class*, *module* pada suatu sistem. *Component* pada *deployment diagram* menunjukkan keberadaan *component* berada..
2		*Link*	Penghubung relasi antar *node*
3		*Dependency*	Menjelaskan adanya ketergantungan dari suatu *component* atau *node*. Ketergantungan yang dimaksud merupakan *component* atau *node* memerlukan suatu hal yang berasal dari *component* atau *node* tersebut.
4		*Node*	Merepresentasikan suatu sumber daya yang dapat melakukan suatu komputasi, pada umumnya berupa seperti *server*.
5		*Port*	Titik koneksi yang dilewati oleh *node* untuk dapat berkomunikasi dengan *node* tujuan.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pada era teknologi saat ini, tidak sedikit perusahaan yang memanfaatkan teknologi dalam proses bisnis. Salah satu jenis perusahaan yang menggunakan teknologi dalam proses bisnisnya yaitu perusahaan *Startup*. *Fintech* merupakan bagian dari *Startup* dikarenakan menggunakan teknologi dalam proses bisnis mereka, teknologi yang dihasilkan oleh perusahaan tersebut berupa sebuah aplikasi, baik dalam bentuk *web* ataupun *mobile*. Dalam membuat suatu aplikasi, *Startup* pada umumnya menggunakan tahapan SDLC (*Software Development Lifecycle*) agar tercapai produk yang direncanakan.

PT. XYZ merupakan suatu perusahaan *Fintech* yang berfokus dalam aset kriptografi. Jasa yang mereka sediakan yaitu menyediakan *platform* aplikasi dalam bentuk *web* dan *mobile* untuk dapat melakukan investasi, membeli dan menjual aset kriptografi. Dalam proses *development* aplikasi, PT. XYZ menggunakan tipe SDLC yang bernama *Agile*, yang mengutamakan kecepatan dan fleksibilitas. Disisi lain, PT. XYZ menerapkan metodologi GitOps dalam menerapkan infrastruktur berbasis kode (*Infrastructure as Code*) dan menjadikan Git sebagai satu-satunya sumber dalam membuat, mengontrol atau bahkan menghapus suatu *infrastructure environment*. PT. XYZ mempunyai divisi *technology* dalam *development application* seperti contohnya yaitu *Backend Engineer* dan *Site Reliability Engineer*.

Dalam proses *development*, *Backend Engineer* mempunyai kebutuhan untuk merancang suatu *database schema* yang akan digunakan oleh aplikasi dalam proses *development*. Dalam proses eksekusi *database schema migration*, *Backend Engineer* tidak bisa melakukannya seorang diri saja, dikarenakan peran yang mempunyai akses terhadap *database* yaitu *Site Reliability Engineer*. Maka daripada itu, *Backend Engineer* perlu bekerja sama dengan *Site Reliability Engineer* dengan menggunakan *platform Slack*

untuk berkomunikasi dan menyampaikan kebutuhan untuk *database schema migration* dengan cara *Backend Engineer* memberikan *file script* yang berkait dengan SQL DDL/DML untuk dieksekusi secara manual menggunakan *platform* PHPMyAdmin oleh *Site Reliability Engineer*. Bagian dari proses tersebut, akan dilakukan secara berulang-ulang saat adanya kebutuhan mengubah atau memperbaharui *database schema*. *Site Reliability Engineer* mengelola sejumlah *database* dengan total 7 *database* di setiap *environment* (terdapat 2 *environment* yaitu *beta* dan *production*), maka daripada itu jumlah keseluruhannya yaitu 14 database. Frekuensi adanya *database schema migration* dalam 4 bulan (April 2024 - Agustus 2024) yaitu sebanyak 35 kali *database schema migration* dengan rata-rata per bulannya yaitu 7x per bulan.

Berdasarkan data jumlah frekuensi *database schema migration* dan jumlah *database* yang dikelola, tentunya dapat dikatakan tidak efisien dikarenakan proses tersebut dilakukan secara manual, tidak tercatat, dan bertolak belakang dengan prinsip yang telah ditetapkan oleh PT. XYZ, yang mana menerapkan metodologi GitOps dan SDLC *Agile*. Proses yang berlangsung berjalan dengan tidak efisien, dapat mempengaruhi kinerja dari *developer application* (*Backend Engineer*). *Database schema migration* yang dilakukan secara manual, menjadikan suatu batasan bagi *Site Reliability Engineer* untuk melacak *database schema migration* yang sudah terjadi di masa lampau.

Saat ini sudah ada beberapa alat yang tersedia untuk dapat melakukan *database schema migration*, namun ada beberapa hal keterbatasan yang dari alat-alat tersebut untuk memenuhi kebutuhan PT. XYZ, seperti halnya integrasi GitOps, notifikasi melewati Slack, dan juga pencatatan historis dari *database schema migration*. Berdasarkan latar belakang yang sudah dijelaskan sebelumnya, dilakukannya penelitian yang berjudul "**Analisis dan Rancang Sistem Otomatis *Database Schema Migration* Berbasis *GitOps* Pada Proses CI/CD Pada Perusahaan *FinTech***" untuk mencapai dan menyesuaikan metodologi yang diterapkan pada perusahaan.

1.2. Identifikasi Masalah

Berdasarkan latar belakang yang sudah dijelaskan, maka rumusan masalah yang akan dibahas antara lain sebagai berikut:

1. Proses *database schema migration* yang berulang-ulang dilakukan secara manual yang membuat proses yang berlangsung tidak efisien.
2. Tidak adanya detail informasi terkait *database schema migration* yang tercatat dalam Git.
3. Tidak ada catatan historis terkait *database schema migration* pada lingkungan *database*.

1.3. Maksud dan Tujuan

Berdasarkan rumusan masalah yang telah dijelaskan, maka tujuan dari penelitian ini antara lain sebagai berikut:

1. Merancang dan membangun suatu sistem untuk menjalankan proses *database schema migration* secara otomatis pada proses CI/CD.
2. Menerapkan metodologi *GitOps*, pada *database schema migration*.
3. Mencatat histori setiap adanya *database schema migration*, untuk dapat membantu melacak ataupun audit.

1.4. Batasan Masalah

Dalam proses penelitian, adanya batasan yang ditetapkan agar penelitian lebih fokus dalam mencapai tujuan penelitian yang telah ditetapkan, antara lain sebagai berikut:

1. Sistem yang dirancang dapat melakukan *database schema migration* berupa DDL/DML *query* pada *database management system* (DBMS) MySQL v5.7.
2. Sistem yang dirancang terintegrasi dengan *GitHub repository* dan mengimplementasikan CI/CD untuk *trigger database schema migration* secara otomatis.
3. Sistem yang dirancang dapat terintegrasi dengan aplikasi *Slack* untuk memberikan suatu notifikasi setiap adanya *database schema migration*.

4. Sistem yang dirancang dapat mencatat historis pada interval waktu 1 hari, 1 minggu dan 1 bulan terakhir terkait *database schema migration*.

1.5. Metodologi Penelitian

Sebelum melakukan penelitian, adanya beberapa tahapan yang perlu dilakukan untuk mendukung penelitian. Tahapan yang diperlukan antara lain yaitu:

1. Studi Pustaka

   Penelitian dilakukan dengan mencari dan mempelajari materi dan teori yang dibutuhkan dari buku, jurnal, dan sumber lain yang menjadi landasan teori dalam pengembangan sistem.

2. Pengumpulan Data

   Mengumpulkan data dan informasi mengenai tahapan proses yang sedang berlangsung yang untuk diperbaharui untuk mendukung proses yang dapat membantu performa kinerja.

3. *Prototyping*

   Mengembangkan sistem aplikasi untuk membantu performa kinerja karyawan pada perusahaan PT. XYZ, dengan runtutan tahapan sebagai berikut:

   a. Mengidentifikasi kebutuhan proses yang sedang berlangsung.
   b. Membuat *prototyping* sistem aplikasi.
   c. Mengevaluasi *prototyping* sistem aplikasi
   d. Pengkodean sistem aplikasi.
   e. Menguji sistem aplikasi.
   f. Evaluasi sistem aplikasi.

1.6. Lokasi dan Waktu Penelitian

Proses analisa dan perancangan sistem ini beralamat lengkap di Equity Tower *11th Floor*, *Suite* E - SCBD Jakarta, RT.5/RW.3, Senayan, Kecamatan Kebayoran Baru, Kota

Jakarta Selatan, Daerah Khusus Ibukota Jakarta 12190 dengan waktu penelitian dimulai pada bulan Agustus 2023.

1.7. Sistem Penulisan

Sistem penulisan akan membantu untuk menjelaskan hal yang terkait dengan penelitian ini, dan akan diselaraskan menjadi beberapa bab dan subbab. Antara lain sebagai berikut:

BAB I: PENDAHULUAN

Pada bagian ini, menjelaskan terkait aspek mengenai latar belakang yang mendasari penelitian, kemudian adanya beberapa hal identifikasi masalah yang ditemukan, lalu adanya tujuan penelitian, pembatasan masalah yang membuat penelitian ini lebih fokus, metode penelitian yang digunakan, lokasi dan waktu penelitian yang dilakukan, dan sistem penulisan yang digunakan dalam penelitian ini.

BAB II: LANDASAN TEORI

Pada bagian ini, menjelaskan materi, konsep, pengertian dan referensi yang menjadi suatu landasan dari penelitian ini yang menunjang dalam membangun sistem aplikasi untuk mencapai tujuan dari penelitian ini.

BAB III: ANALISIS SISTEM INFORMASI

Pada bagian ini, menjelaskan analisa dan perancangan dari sistem aplikasi yang meliputi analisa proses yang lama dan baru dan perancangan sistem aplikasi baru.

BAB IV: PERANCANGAN SISTEM INFORMASI

Pada bagian ini, menjelaskan implementasi dari sistem aplikasi yang telah direncanakan, lalu diujikan efektivitas dari sistem aplikasi baru.

BAB V: KESIMPULAN DAN SARAN

Pada bagian ini, menjelaskan dari kesimpulan dan saran atas penelitian yang telah dilakukan dalam menyelesaikan permasalahan yang terjadi.

BAB II
LANDASAN TEORI

2.1. Pembangunan Sistem Informasi

2.1.1. Sistem informasi

Pelaksanaan sistem informasi tidak selalu berhubungan dengan komputer, akan tetapi sistem informasi yang menggunakan komputer dinamakan sistem informasi berbasis komputer. Akan tetapi dasar dari sistem informasi yaitu suatu komponen (manusia, komputer, teknologi informasi, dan prosedur kerja) yang melakukan proses perubahan data menjadi informasi agar mencapai tujuan atau sasaran. (Kadir, 2014).

Namun menurut (Soufitri, 2023) dalam bukunya yang berjudul "*Konsep Sistem Informasi*", menyatakan bahwa

"*Sistem informasi adalah kombinasi antara prosedur kerja, informasi manusia dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi*"
(Soufitri, 2023)

Dari kedua referensi tersebut, dapat disimpulkan bahwa sistem informasi akan mengubah data yang dilakukan oleh suatu entitas, dan pada umumnya entitas tersebut antara lain manusia dan komputer yang melakukan suatu proses untuk mengelola data menjadi suatu informasi. Tujuan dari sistem informasi dengan mengumpulkan data, mengelola data dan menyebarkan suatu informasi yang bermanfaat bagi suatu organisasi. (Erkamim, 2023).

2.2. Metode Pengembangan yang digunakan

Prototype adalah metode yang digunakan dalam perancangan sistem. *Prototype* mempunyai pengertian yaitu proses pengembangan sistem yang berfokus dalam pembuatan sistem aplikasi untuk memvisualisasikan ide, menguji rancangan, mengidentifikasi potensi masalah, dan menyelesaikan solusi secara komprehensif. (Fridayanthie, Haryanto, & Tsabitah, 2021). Disebutkan bahwa *prototype* yaitu proses

pengembangan sistem, maka dalam pengembangan sistem terdapat beberapa tahapan yang perlu dilakukan antara lain (Fridayanthie, Haryanto, & Tsabitah, 2021), yaitu:

a. Tahapan Pengumpulan Kebutuhan.

   Pengembang mengumpulkan informasi yang dibutuhkan dalam merancang sistem aplikasi. Tahapan ini juga, pengembang dapat bekerja sama dengan orang (pengguna) yang berinteraksi langsung dengan proses untuk mendapatkan informasi seperti kebutuhan dari pengguna, ataupun kekurangan dari proses yang sedang berlangsung.

b. Tahap *Prototyping*

   Berlandaskan informasi yang telah didapatkan, pengembang dapat membuat suatu rancangan awal sebagai inisiasi untuk nantinya akan digunakan dalam membangun sistem aplikasi, dan pengembang biasanya sebuah sketsa atau gambaran *input* dan *output* dari sistem aplikasi.

c. Tahap Evaluasi *Prototyping*

   Pengguna atau orang yang berinteraksi langsung dengan proses, akan mengevaluasi dari sketsa atau gambaran dari sistem aplikasi. Hal yang ditelusuri antara lain untuk memastikan sistem aplikasi tersebut sesuai dengan kebutuhan yang telah ditentukan.

d. Tahap Pengkodean Sistem

   Sketsa atau gambaran awal yang telah disetujui dan diperbaharui, akan ditransformasikan dan dibuatkan ke dalam sistem aplikasi dengan menggunakan bahasa pemrograman dan alat bantuan lainnya.

e. Tahap Pengujian Sistem

   Sistem aplikasi yang telah dibuat, akan diuji mengenai *input*, proses, *output* dan fitur yang tersedia. Tahapan pengujian ini memiliki cara dengan menggunakan metode pengujian *black box*, *white box* ataupun pengujian arsitektur. Hal ini dilakukan untuk mengupayakan sekecil mungkin kesalahan pada sistem aplikasi.

f. Tahap Evaluasi Sistem

Pengguna atau orang yang langsung berinteraksi dengan sistem aplikasi akan melakukan pengujian secara langsung terhadap sistem aplikasi. Pengujian ini untuk memastikan sesuai dengan skema atau gambaran awal yang telah disepakati. Jika terjadi adanya hal yang tidak sesuai, maka pengembang akan memperbaharuinya.

g. Tahap Menggunakan Sistem.

Sistem aplikasi yang telah lulus uji kelayakan dan pengujian, tahapan selanjutnya dapat diimplementasikan oleh pengguna.

2.3. Tahapan Analisis dan Perancangan Berorientasi Objek

2.3.1. Desain berorientasi objek

Desain berorientasi objek yaitu suatu metode yang menerapkan objek selaku komponen utama dalam membangun suatu sistem. Objek yang dimaksud memiliki karakteristik antara lain kelas, atribut dan metode yang dapat membuat objek saling berkomunikasi satu dengan yang lainnya. Dengan melihat suatu desain berorientasi objek yang tercantum dalam notasi, dapat mengetahui struktur ataupun karakteristik dari suatu sistem, dikarenakan notasi tersebut merupakan gambaran dari model logis dan fisik sistem. (Mulyani, 2016). Desain berorientasi objek menerapkan istilah *reuse* atau penggunaan kembali untuk membuat sistem lebih efisien dan andal yang melewati proses enkapsulasi. (Anardi, 2019). Karakteristik dari desain berorientasi objek antara lain (Mulyani, 2016):

1. *Encapsulation* (Enkapsulasi)

   Prinsip penting dalam berorientasi objek yaitu membangun sistem yang lebih tertata, aman dan mudah untuk dikelola. Enkapsulasi guna meningkatkan keamanan bagi sistem dengan membuat data atau informasi pada suatu objek tidak dapat diakses secara mudah.

2. *Inheritance* (Turunan)

Suatu objek memiliki data *property* dan metode lalu diwariskan terhadap objek lainnya, merupakan pengertian dari *inheritance*. Hal tersebut membuat objek turunan dapat memiliki dari seluruh *property* dan metode dari objek induk.

3. *Polymorphism*

   Dalam sistem, jika adanya objek yang sama akan memiliki bentuk dan perilaku sama. Walau begitu, objek yang sama dapat menghasilkan *output* data atau respons yang sama namun dengan metode yang digunakan berbeda.

2.3.2. Diagram u*nified modeling language* (UML) yang digunakan

Unified Modeling Language (UML) yaitu bahasa pemodelan mengandung visualisasi dan sistem kerja dari suatu sistem aplikasi yang sangat populer saat ini, dikarenakan UML dapat membuat *output* kode pemrograman yang siap untuk digunakan. (Nistrina & Sahidah, 2022). Menurut (Mulyani, 2016) dalam bukunya yang berjudul "Model Analisis dan Perancangan Sistem", menyatakan bahwa ada 4 kegunaan dari UML, antara lain yaitu:

a. *Visualizing*, yang berarti UML sebagai media perantara dari tim pengembang dalam membentuk sistem aplikasi.
b. *Specifying*, yang berarti UML sebagai alat yang tepat untuk memodelkan sistem aplikasi .
c. *Constructing*, yang berarti UML sebagai bahasa grafis yang menjelaskan *mapping* dari sistem aplikasi untuk dapat diterjemahkan menjadi bahasa pemrograman oleh pengembang sistem aplikasi
d. *Documenting*, yang berarti UML sebagai dokumentasi teknis dari suatu sistem aplikasi.

UML mempunyai beragam diagram yang disediakan, akan tetapi diagram yang digunakan hanya beberapa. Berikut diagram yang digunakan dan beserta penjelasan menurut (Mulyani, 2016) dalam bukunya yang berjudul "Model Analisis dan Perancangan Sistem", antara lain:

1. *Use case diagram*.

   Diagram ini merupakan visualisasi dan dokumentasi yang berupa diagram dan teks yang mengandung proses yang dilakukan oleh seorang aktor atau *user* terhadap sistem.

2. *Activity diagram*

   Diagram ini memiliki kemiripan dengan *flowchart*, namun hal yang membedakan diagram ini dengan *flowchart* yaitu *activity diagram* dapat menjelaskan dari percabangan aktivitas dan membedakan aktivitas dari suatu aktor. Pada dasarnya diagram ini menjelaskan alur aktivitas dari satu proses, dan rangkaian aktivitas tersebut tersusun dengan sesuai urutan.

3. *Class diagram*

   Diagram ini merupakan visualisasi dari kelas, komponen pada suatu kelas dan hubungan antar masing-masing kelas. Diagram ini juga menunjukkan properti dan batasan yang dapat dilakukan pada suatu kelas.

4. *Sequence diagram*

   Diagram ini merupakan visualisasi dari interaksi dari objek, juga menjelaskan dari perilaku yang dimiliki oleh objek pada proses sistem aplikasi.

2.3.3. *Object oriented programming* (OOP)

Pemrograman Berbasis Objek atau dalam bahasa inggris *Object Oriented Programming* (OOP), yaitu paradigma yang menjadikan objek sebagai pusat dalam suatu pemrograman yang memiliki data dan fungsi guna membantu *programmer* dalam membangun hubungan dan mewariskan karakteristik setiap objek (Subiyantoro, 2013). Keuntungan dalam sebagai *programmer* menggunakan OOP antara lain (Rais, 2019):

1. Objek dan kelas yang dapat digunakan kembali, sehingga dapat meningkatkan produktivitas.

2. Sistem yang tertata dengan baik, dapat memudahkan dan mempercepat pengembangan sistem aplikasi dikarenakan tidak memerlukan waktu yang lama dalam menganalisis dan merancang fitur tambahan baru.
3. Konsistensi dari objek, membuat pemeliharaan kode menjadi lebih mudah.

Objek dalam OOP merupakan bagian terkecil dalam pemrograman yang mempunyai data dan fungsi. Sedangkan kelas dalam OOP merupakan wadah yang dijadikan dari suatu objek. Maka daripada itu dalam metode OOP ini, harus membuat kelas terlebih dahulu, kemudian membentuk objek. (Retnoningsih, Shadiq, & Oscar, 2017).

2.4. Perangkat Lunak untuk Analisis dan Perancangan Sistem

2.4.1. Perangkat lunak yang digunakan

Merancang dan membangun sistem, adanya komponen dari perangkat lunak yang digunakan antara lain:

1. MySQL

   MySQL merupakan *database management system* yang menerapkan *multithread*, *multi-user* dan telah digunakan sekitar 6 juta pengguna di seluruh dunia. (Susanto & Meiryani, 2019). *MySQL* adalah salah satu dari *database server* bertipe *open source*, dan itu merupakan alasan banyak penggunanya. (Beaulieu, 2020)

2. Redis

   Redis yaitu singkatan dari *Remote Dictionary Server* yang diciptakan menggunakan pemrograman bahasa C oleh Salvatore Sanfilippo di tahun 2006. Redis merupakan bagian dari salah satu contoh sistem penyimpanan data yang menerapkan *No*-SQL, dengan kata lain menyimpan data penyimpanan pada memori dan sistem pembacaan data yang dilakukan oleh Redis lebih cepat daripada penyimpanan data lainnya. Redis juga menyediakan fitur untuk menerapkan kedaluwarsa kunci, transaksi dan

publish/*subscribe* yang dapat dikonfigurasi. (Da Silva & Tavares, 2015). Redis mempunyai beberapa penyimpanan data (*keys*) yang dipetakan dalam beberapa bentuk tipe data dengan struktur STRING, LIST, SET, HASH, ZSET. (Carlson, 2013)

3. DRF

Django Rest Framework (DRF) merupakan bagian dari *framework* dari Django yang ditujukan untuk membangun API *web*. DRF menyediakan beberapa komponen yang siap digunakan antara lainnya *Class-based* REST *views*, *Viewsets*, *Serializers*. Dalam C*lass-Based Views* (CBV) pada DRF menyediakan *tools* dalam membangun REST API, dapat dilihat pada tabel 2.1 (Gagliardi, 2021).

Tabel 2.1 *Class-Based Views Django*

Generic View	**Kegunaan**
CreateAPIView	Membuat data berdasarkan *request* yang diterima oleh API dalam bentuk metode POST
ListAPIView dan *RetrieveAPIView*	Menampilkan data berdasarkan *request* yang diterima oleh API dalam bentuk metode GET
DestroyAPIView	Menghapus data berdasarkan *request* yang diterima oleh API dalam bentuk metode DELETE
UpdateAPIView	Memperbaharui data berdasarkan *request* yang diterima oleh API dalam bentuk metode PUT dan PATCH

Sumber: (Gagliardi, 2021)

DRF menyediakan metode *viewsets* untuk mengelola operasi *Create Read Update Delete* (CRUD) dalam membangun sebuah API, dijelaskan pada tabel 2.3. (Gagliardi, 2021)

Tabel 2.2 Relasi antara metode *viewsets*, metode HTTP dan operasi CRUD

Metode *Viewset*	**Metode HTTP**	**Operasi CRUD**
create()	POST	Membuat suatu *resource*
list() atau *retrieve*()	GET	Mengambil suatu *resource*

update()	PUT	Memperbaharui suatu *resource*
destroy()	DELETE	Menghapus suatu *resource*
update()	PATCH	Memperbaharui sebagian pada *resource*

Sumber: (Gagliardi, 2021)

Pada D*jango*, terdapat 3 bagian istilah yang digunakan yaitu *models*, *forms*, dan *serializers*. *Models* yaitu penghubung antara pengguna suatu aplikasi dengan *database*, lalu *serializers* merupakan penghubung antara pengguna akhir, REST API dan model pada *Django*. (Gagliardi, 2021)

4. Vue JS

Pada buku yang berjudul "*Vue.js 3 Design Patterns and Best Practices*" karya (Garaguso, 2023), menuliskan pengertian *Vue* JS seperti pada di bawah ini.

"*Vue is a progressive framework for building user interfaces. Being progressive means that it has the architectural benefits of a framework, but also the speed and modular advantages of a library, as features and functionality can be incrementally implemented*."
(Garaguso, 2023)

Pengertian lainnya oleh (Quinten, 2024) pada karya bukunya yang berjudul "*Building Real-World Web Applications with Vue.js 3*", menyatakan seperti pada penggalan di bawah ini.

"*Vue.js is an open source progressive JavaScript frontend web framework used to develop interactive frontend web interfaces. It is a very popular and simplified JavaScript framework that focuses on the view layer of web development. It can be easily integrated into big and enterprise web development projects."*
(Quinten, 2024)

Berdasarkan kedua sumber buku tersebut, disimpulkan bahwa Vue JS yaitu sebuah *framework* pada *JavaScript* yang digunakan dalam membantu dalam membangun *user interface* pada *web*. Vue JS ini banyak diminati dan digunakan dikarenakan mudah untuk dipelajari dan Vue JS ini juga bersifat *open source* yang berarti tidak perlu *license*. Kasus umumnya

pengguna menggunakan Vue JS yaitu untuk membangun *user interface* yang interaktif dan *responsive*.

5. Git

Git menurut (Skoulikari, 2023) memiliki pengertian bahwa Git merupakan suatu teknologi yang berfungsi dalam membantu orang dalam berkolaborasi dalam melacak suatu perubahan pada proyek, maka daripada itu Git dapat dikatakan sebagai *version control system*. Sedangkan pengertian dari (Ponuthorai & Loeliger, 2022), pada bukunya berjudul "*Version Control with Git, 3rd Edition*" menyatakan bahwa:

" Simply put, Git is a content tracker. Given that notion, Git shares common principles of most version control systems. However, the distinct feature that makes Git unique among the variety of tools available today is that it is a distributed version control system. This distinction means Git is fast and scalable, has a rich collection of command sets that provide access to both high-level and low-level operations, and is optimized for local operations."
(Ponuthorai & Loeliger, 2022)

Dari kedua pengertian yang telah didapatkan, dapat disimpulkan bahwa Git yaitu *version control system* yang terdistribusi, dengan dapat melakukan pelacakan konten yang terdapat pada Git. Dengan penerapan sistem terdistribusi pada Git, yang membuat pembeda dengan *version control system* lainnya, mudah dikembangkan dan mengoptimalkan pekerjaan proyek, baik yang dilakukan pada lingkungan lokal maupun secara *cloud*

6. GitHub

GitHub yaitu suatu lingkungan kolaboratif terkait *coding* terhadap *developer* untuk melakukan kolaborasi yang menerapkan *version control system* terdistribusi yang berbasiskan *web*. (Wu, Wang, Kropczynski, & Carroll, 2017)

Pemahaman terkait GitHub bukan hanya terkait *platform* untuk menyimpan suatu kode, tapi GitHub juga dapat digunakan sebagai *platform* untuk melakukan perencanaan suatu proyek dan melacak perubahan yang

terjadi proyek tersebut. Mayoritas *developer* proyek yang berbasis *open source* memilih GitHub, dengan suatu kemudahan dalam melakukan *release* proyek yang mereka kelola dan mudah juga untuk diakses. (Tsitoara, 2020)

7. GitOps

   Dalam buku yang berjudul "*GitOps and Kubernetes*", karya (Ekenstam, Suen, Matyushentsev, & Yuen, 2021), mengatakan bahwa

   "*Git is the most widely used version-control system in the software industry today. GitOps is a set of procedures that uses the power of Git to provide both revision and change control within the Kubernetes platform. A GitOps strategy can play a big part in how quickly and easily teams manage their services' environment creation, promotion, and operation.*"
   (Ekenstam, Suen, Matyushentsev, & Yuen, 2021)

   Sedangkan menurut (Vinto & Bueno, 2023) dalam karya bukunya yang berjudul "*GitOps CookBook*", mempunyai pengertian mengenai *GitOps* seperti pada di bawah ini.

   "*GitOps is a methodology and practice that uses Git repositories as a single source of truth to deliver infrastructure as code. It takes the pillars and approaches from DevOps culture and provides a framework to start realizing the results. The relationship between DevOps and GitOps is close, as GitOps has become the popular choice to implement and enhance DevOps, platform engineering, and SRE*"
   (Vinto & Bueno, 2023)

   Berlandaskan pada kutipan kedua buku tersebut, dapat disimpulkan bahwa GitOps mempunyai pengertian yaitu suatu rangkaian metodologi yang menerapkan Git dalam mendefinisikan dan mengelola infrastruktur maupun suatu *service* yang disimpan pada suatu *repository* Git. Implementasi GitOps ini biasanya digunakan oleh *DevOps Engineer*, *Site Reliability Engineer* dan *Platform Engineering* dalam mengelola infrastruktur *environment* mereka, disisi lain dengan mengimplementasikan GitOps, juga dalam melakukan pemeriksaan *state* dari infrastruktur atau *service* yang sedang berlangsung pada suatu *environment*.

8. API

Application programming interface (API) merupakan suatu teknologi yang menjembatani antara dua atau lebih *service* untuk dapat saling berinteraksi dan berkomunikasi dengan menggunakan protokol dan definisi tertentu. Dalam interaksi antar *service* tersebut, adanya proses *request* dan juga *response*. Proses *request* yang dilakukan disebut dengan *client* dan proses *response* yang dilakukan disebut dengan *server*. (AWS, 2024). Untuk melakukan interaksi terhadap API, ada beberapa fungsi yang terdapat pada API, antara lain sistem *Create Read Update Delete* (CRUD) dengan metode GET, POST, PUT dan DELETE. (Yanti & Rihyanti, 2021).

API memiliki beberapa jenisnya berdasarkan arsitekturnya, dan jenis API yang digunakan yaitu REST (*Representational State Transfer*) API. Tipe ini merupakan jenis dari API yang sederhana, yang mana hanya berfokus pada interaksi pada komunikasi dan mensinkronisasi objek antar *server* dengan *client*. Pada proses komunikasi antara *server* dengan *client*, tidak menyimpan *state* dikarenakan REST APIs bersifat *stateless*. Akan tetapi proses yang menggunakan protokol seperti *Transmission Control Protocol* (TCP) akan ada proses menyimpan informasi mengenai durasi sesi berlangsung. (Shaw, Badhwar, Guest, & Chandra, 2023).

9. Celery

Celery merupakan *library* yang tersedia untuk Python yang bertujuan untuk menjalankan dan mengelola *task* secara *background*. *Task* yang dijalankan dapat berupa secara *real-time* dan ada pula dieksekusi secara terjadwal. *Task* yang dimaksud pada Celery yaitu objek yang dieksekusi yang dapat dilakukan secara bersamaan pada satu atau lebih *worker servers* dengan menerapkan proses *multiprocessing*, *Eventlent*, atau *gevent*. (Banaszek, 2023).

2.4.2. Bahasa pemrograman yang digunakan

Merancang dan membangun sistem, adanya bahasa pemrograman dari perangkat lunak yang digunakan antara lain:

1. Python

   Python yaitu suatu bahasa pemrograman yang memiliki *variable*, *loop*, *conditional*, *function*, dan hal lainnya. Yang membedakan Python dengan bahasa pemrograman lainnya yaitu mudah untuk dipahami dan dibaca, contohnya yang pertama yaitu dalam membuat suatu *script* pada Python itu tidak memerlukan titik koma (;) pada akhir *statement*. Yang kedua yaitu itu tidak menggunakan simbol kurung kurawal, akan tetapi digantikan dengan penggunaan *indentation* dengan spasi. (Barry, 2023). Python pada umumnya digunakan pada kasus untuk data, operasional dan *web development*. Maksud dari kasus operasional itu untuk mempercepat operasional yang dilakukan secara manual dan diterapkan secara otomatis dan juga untuk menjaga infrastruktur berjalan dengan baik. (VanderPlas & Harisson, 2018) .

2. Javascript

   Pemrograman JavaScript (JS) pada umumnya digunakan untuk membangun suatu aplikasi *web*, dan *script*. JS sering dikaitkan dengan HTML untuk mendukung suatu fungsi pada *web*, akan tetapi JS juga dapat digunakan sebagai suatu fungsi logika yang berhubungan langsung dengan *database*. (Svekis, Putten, & Percival, 2021)

   Pada umumnya bahasa pemrograman mempunyai suatu fungsi *input* bahkan *output* dalam menjalankan suatu proses, akan tetapi JS tidak mempunyai seperti itu dan bergantung pada *environment* yang menjadi wadah JS dijalankan dan jika dikaitkan sangat korelasi dengan *web* karena JS hanya akan membaca berdasarkan *action event* yang dilakukan terhadap

web. JS mempunyai berbagai *framework* yang dapat digunakan dalam membangun *web* dengan menyediakan *library* yang terdiri dari *user interface* yang *reusable* untuk mempercepat dalam membangun suatu *web*. (Flanagan, 2020)

3. SQL

Structured Query Language (SQL) *database* yaitu berupa gabungan dari beberapa tabel yang terstruktur, dengan terdapatnya baris sebagai entitas data dan kolom sebagai suatu informasi tertentu. (SolarWinds, 2023). Dalam pengoperasian *database* ada suatu proses *desired* yang dikirimkan ke *database* disebut dengan istilah *query*. *Query* ini digunakan sebagai istilah bahasa yang digunakan dalam *database* dengan sebutan *query language* dan dari bahasa yang ada dalam *database*, SQL menjadi bahasa yang sering digunakan. (Wang & Tan, 2006). SQL dijabarkan menjadi beberapa sub-bahasa (Oracle, 2021), antara lain:

a. *Data Definition Language* (DDL). Merupakan suatu perintah SQL untuk mendeskripsikan skema *database* terkait relasi, menghapus relasi, dan juga mengubah skema. (Amornchewin, 2018)
b. *Data Manipulation Language* (DML). Merupakan suatu perintah SQL untuk melakukan mendapatkan data dan serta memanipulasi data tersebut. (Amornchewin, 2018)

4. HTML

HTML (*Hyper Text Mark Language*) merupakan suatu standar pemrograman *markup* yang ditujukan dalam membangun suatu halaman *web*. Dalam membangun halaman *web*, HTML dapat membuat suatu struktur bagan, paragraf dan *link* tautan dengan adanya elemen pada HTML. (Astari, 2023). Jika dibedah berdasarkan namanya, *HyperText* yaitu suatu metode yang digunakan oleh *user* dalam melakukan navigasi terhadap *web*, lalu

Markup yaitu suatu hal yang sudah didefinisikan pada *tag* HTML terhadap teks akan dimodifikasikan menjadi suatu tampilan teks baru. HTML memiliki istilah *tag* yang berguna untuk memisahkan teks normal yang dicantumkan pada HTML dengan kode HTML. (Hayes, 2022)

5. CSS

Buku yang berjudul "*CSS: The Definitive Guide, 5th Edition*", karya (Meyer & Weyl, 2023), mengatakan bahwa

"*Cascading Style Sheets (CSS), a powerful programming language that transforms the presentation of a document or a collection of documents, has spread to nearly every corner of the web as well as many ostensibly nonweb environments. For example, embedded-device displays often use CSS to style their user interfaces, many RSS clients let you apply CSS to feeds and feed entries, and some instant message clients use CSS to format chat windows. Aspects of CSS can be found in the syntax used by JavaScript (JS) frameworks and even in JS itself. It's everywhere!*"
(Meyer & Weyl, 2023)

Pada buku lainnya karya (Grant, 2018) yang berjudul "*CSS in Depth*" mengatakan bahwa

"*CSS is unlike a lot of things in the world of software development. It's not a programming language, strictly speaking, but it does require abstract thought. It's not purely a design tool, but it does require some creativity. It provides a deceptively simple declarative syntax, but if you've worked with it on any large projects, you know it can grow into unwieldy complexity.*"
(Grant, 2018)

Berdasarkan pada kedua sumber tersebut, dapat ditarik suatu kesimpulan bahwa CSS yaitu bahasa yang digunakan untuk membuat dan mengubah tampilan yang menarik pada halaman *web*. Disisi lain, tidak hanya ditujukan untuk *web*, namun bisa digunakan juga pada aplikasi yang terpasang pada laptop atau *device* lainnya. CSS dapat ditemukan pada bahasa pemrograman *JavaScript* (JS) dan kedua komponen ini saling berkaitan pada suatu *web*.

2.5. *Database Schema Migration*

Database schema migration merupakan suatu metode dalam melakukan pemindahan *database* dalam bentuk skema yang sama dari suatu perangkat ke perangkat lainnya. Disisi lain juga, *database schema migration* cukup membantu dalam membuat atau melakukan perubahan terhadap suatu tabel di *database*. (Giri, I.G.D., & Wibawa, 2022). Peran yang melakukan perubahan skema *database* pada umumnya dilakukan oleh *Database Administrator* atau disingkat DBA. DBA merupakan seseorang yang bertanggung jawab untuk membuat desain, kontrol dan juga mengelola administrasi *database*. (Wang M. , 2003). Namun pada penelitian ini, peran DBA dirangkap oleh *Site Reliability Engineer* atau SRE, yang berarti pekerjaan DBA dilakukan oleh SRE.

Penerapan *database schema migration* sangat bermanfaat, selain bertujuan untuk mengelola perubahan *schema database* atau struktur data dengan cara terprogram, *database schema migration* memiliki manfaat antara lain (Primsa.Io, 2023):

1. Mencegah hilangnya data dalam proses perubahan struktur database.
2. Membantu *application developer* untuk merencanakan, memvalidasi dan menerapkan perubahan *database schema* secara aman.
3. Memungkinkan penyesuaian manual dalam proses, serta audit dan pengujian setiap perubahan.

Dalam penelitian yang dilakukan oleh (Fernandez, 2021) dalam judulnya "*Database Schema Migration in Highly Available Service*", melampirkan bahwa proses *database schema migration* dilakukan di beberapa perusahaan yang menerapkan proses *development* aplikasi yang cepat dan *agile* atau metodologi *DevOps*, untuk mengurangi durasi dan biaya dalam *delivery* aplikasi hingga ke tahap *production*. Terlebih lagi perubahan *database schema* yang dilakukan secara manual dapat menunda proses pada *development cycle* yang dilakukan oleh *software developer* (Fernandez, 2021). Dalam melakukan *database schema migration*, ada beberapa hal penting yang perlu diperhatikan antara lain sebagai berikut (Laszewski & Nauduri, 2012):

1. Memastikan kelengkapan skema
2. Tabel dengan fungsi sistem sebagai klausa dengan nilai *DEFAULT* pada kolom.
3. Membuat akun pengguna beserta *role* pada *database*. Dengan menetapkan suatu akun dengan *role* tertentu dapat membantu dalam mengelola *object-level*.
4. Mempartisi tabel *database*. Dengan menerapkan partisi tabel, akan memudahkan dalam melakukan manajemen dan mengoptimalkan kinerja pada saat melakukan *query* pada *database*.

Menurut (Botros & Tinley, 2021) dalam bukunya yang berjudul "*High Performance MySQL, 4th Edition"*, untuk mengelola perubahan skema ada beberapa poin yang perlu dipertimbangkan:

1. Menetapkan rekan untuk meraih kesuksesan. Dalam hal ini yang mempunyai makna bahwa dalam melakukan perubahan skema tidak membuat *developer* bergantung pada seseorang atau tim yang mengelola *database*, akan tetapi mereka sendiri dapat melakukannya sendiri.
2. Mengintegrasikan manajemen skema dengan *continuous integration*. Dengan menetapkan premis bahwa peran yang mengelola skema yaitu *developer*, maka perlu ditentukan alur kerja yang berhubungan dengan mereka agar meningkatkan kinerja *developer*.
3. *Source control* (contohnya: *Git*) sebagai media yang menyimpan informasi terkait skema. Mengelola perubahan skema dalam jangka besar perlunya media yang mendukung dan melacak perubahan skema yang telah dilakukan.
4. Menghindari perubahan skema yang dilakukan pada lingkungan *production* yang menyebabkan *downtime* dengan cara menggunakan *native* DDL *statements* dan menggunakan *external tool* untuk menjalankan perubahan skema.
5. Mengimplementasikan perubahan skema pada alur CI/CD dapat menghilangkan *bottleneck* pada produktivitas *engineer*.

2.5.1. Alat-alat *database schema migration*

Saat ini, sudah ada beberapa alat *database schema migration* yang populer di kalangan perusahaan IT untuk dapat memudahkan dan mengelola *database schema migration*, seperti contohnya yaitu Liquibase dan Flyway. Kedua aplikasi tersebut dibentuk dari perusahaan yang berbeda dan juga mempunyai keunggulannya masing-masing. Perbandingan dari kedua alat tersebut disajikan dalam tabel .

Tabel 2.3 *Tools Database Schema Migration*

	Flyway	**Liquibase**
Perusahaan	RedGate	Liquibase Inc.
Dipublikasikan	2010	2006
Engine	Java + JVM	Java + JVM
Edition	Community, Teams, and Enterprise	Open Source and Pro
***Database* GitOps**	Tidak tersedia (Kalau ingin mengimplementasikan perlu ada cara tambahan diluar dari fitur yang disediakan)	Tidak tersedia (Kalau ingin mengimplementasikan perlu ada cara tambahan diluar dari fitur yang disediakan)
Pencatatan Historis	Tersedia (Hanya terdapat pada *Enterprise Edition* atau diatasnya)	Tidak tersedia
Developer Interface	CLI	CLI
Support Database	Aurora MySQL, Aurora PostgreSQL, Azure Synapse, Clickhouse, CockroachDB, Databricks, DB2, Derby, Firebird, Google BigQuery, Google Cloud Spanner, H2, HSQLDB, Informix, MariaDB, MongoDB, MySQL, Oracle, Percona XtraDB Cluster, PostgreSQL, Redshift, SAP HANA	Amazon DocumentDB, Amazon DynamoDB, Amazon Redshift, AWS Aurora - MySQL, AWS Aurora for PostgreSQL, AWS RDS - MariaDB, AWS RDS - Microsoft SQL Server, AWS RDS - MySQL Server, AWS RDS - Oracle Database, AWS RDS - PostgreSQL, Azure Database for MySQL,

	Flyway	Liquibase
Support Database	(Including SAP HANA Cloud), SingleStoreDB, Snowflake, SQLite, SQL Server, Sybase, ASE, TiDB, TimescaleDB, YugabyteDB	Azure Database - PostgreSQL - Flexible Server, Azure Database - PostgreSQL Single Server, EnterpriseDB, Google BigQuery Pro, Google BigQuery Open Source, Google Cloud SQL - MSSQL, Google Cloud SQL - PostgreSQL, IBM DB2 LUW, MariaDB Server, MariaDB SkySQL, Microsoft Azure SQL Database, Microsoft Azure SQL Managed Instance, Microsoft SQL Server, MongoDB Pro Extension Community and Enterprise Server, MongoDB Atlas Pro Extension, MySQL Server, Oracle Database, Oracle Autonomous Database, PostgreSQL, Snowflake
***Support* MySQL v5.7**	Ya (Limitasi pengujian pada *Community Edition*)	Ya (Akan tetapi sudah tidak berlaku lagi pada MySQL Server dan AWS RDS)
Kemudahan Penggunaan	Ya (Mudah digunakan bagi pemula)	Tidak (Banyaknya fitur yang disediakan sehingga tidak cocok untuk pemula)
Integrasi notifikasi	Tidak	Tidak
SQL *Check Validation*	Tidak	Tidak
Terdapat nama pengeksekusi *database schema migration*	Tidak	Ya

2.6. *Continuous Integration and Continuous Delivery/Deployment* (CI/CD)

Beck memperkenalkan metodologi bernama *eXtreme Programming* dengan menghasilkan praktik dari *software engineering* yang bernama CI/CD. CI/CD merupakan

bagian dari *software development* untuk menunjang otomatis proses dari rangkaian alur seperti kompilasi *software*, pembuatan *software*, pengujian *software*, validasi *software*, rilis *software* dan hal-hal lainnya. (Mazrae, Mens, Golzadeh, & Decan, 2023). CI/CD terbagi menjadi beberapa proses, antara lainnya adalah:

2.6.1. *Continuous Integration* (CI)

CI diperkenalkan pada tahun 2006, akan tetapi tidak terlalu populer oleh kalangan praktisi, akan tetapi setelah hadirnya alat CI seperti Travis CI, implementasi CI meningkat dibanding sebelumnya (Rahman, Agrawal, Krishna, & Sobran, 2018). Tahapan proses ini mengacu pada penggabungan dari *branch feature* ke *branch main* pada suatu *repository*, kemudian dilakukannya berbagai proses uji coba pada *code*. (Loubser, 2021). Manfaat dari c*ontinuous integration* adalah membantu proses dalam menemukan dan memperbaiki *bug*, meningkatkan kualitas *software*, menemukan dan memperbaiki *commit* yang rusak, meningkatkan *development workflow*. (Arachchi, 2018)

2.6.2. *Continuous Delivery* (CD)

Fase berikutnya setelah *continuous integration*, merupakan implementasi *code software* pada lingkungan infrastruktur *staging* dan sekaligus melakukan pengujian fungsi *software* tersebut. (Abiola & Olufemi, 2023). Namun hal lain mengatakan bahwa fase ini berarti perubahan yang terjadi pada *software* oleh *developer* itu otomatis diuji oleh *bug* dan *upload* ke suatu *repository* seperti *GitHub* atau *container registry* (RedHat, 2023).

2.6.3. *Continuous Deployment* (CD)

Fase terakhir yang menerapkan alur proses yang dilakukan secara otomatis dengan menjalankan tahapan rilis *software* bahkan implementasi *software* pada lingkungan infrastruktur *production* (Abiola & Olufemi, 2023).

Proses ini dapat membantu dalam menyelesaikan permasalahan pada saat tim operasional melakukan implementasi *software* atau *delivery software* pada lingkungan infrastruktur *production* dan proses yang dilakukan hanya membutuhkan waktu beberapa menit saja. Akan tetapi walau proses ini dilakukan secara otomatis, proses ini sebelumnya harus melewati fase yang telah teruji dan tervalidasi sebelum *software* dirilis. (RedHat, 2023)

BAB III
ANALISIS SISTEM INFORMASI

3.1. Gambaran Umum Perusahaan

Perusahaan PT. XYZ merupakan suatu perusahaan bursa aset kriptografi asal Indonesia yang didirikan sejak tahun 2018. Perusahaan ini menyediakan *platform* yang berbasis *web* dan *mobile application* yang terpercaya dan unggul bagi masyarakat untuk melakukan investasi, membeli dan menjual aset kriptografi. Perusahaan ini telah terdaftar dalam Badan Perdagangan Berjangka Komoditi (BAPPEBTI), sehingga transaksi yang dengan menggunakan *platform* PT.XYZ aman dan legal. Kriptografi yang ditawarkan oleh perusahaan ini antara lain menyediakan berbagai koin dan *token*, yang telah melewati proses kurasi yang ketat. PT. XYZ melayani pelanggan dalam jangka waktu 24/7, yang berarti siap membantu penggunanya 24 jam dalam 7 hari seminggu dengan *live chat*. PT. XYZ mengklaim bahwa transaksi menggunakan *platform* ini, proses yang dilakukan cepat dan harga yang ditawarkan termurah dari kompetitif lainnya. PT. XYZ memiliki Visi dan Misi sebagai berikut:

1. Visi

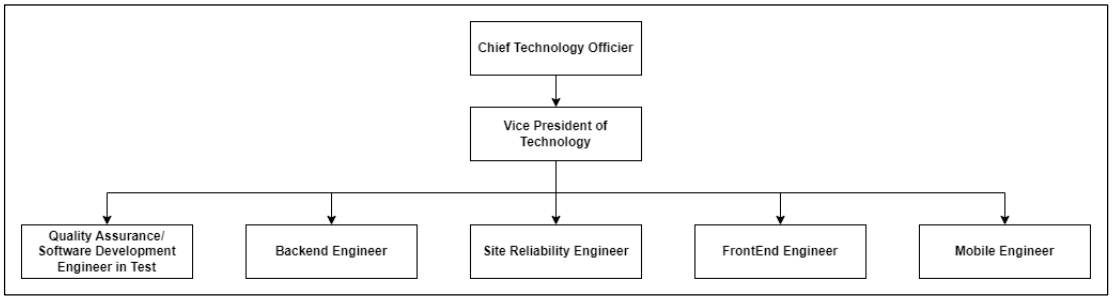
   Menjadi panduan terpercaya untuk semua hal tentang kriptografi bagi seluruh masyarakat Indonesia

2. Misi
   a. Menyediakan *platform* yang aman, mudah digunakan, dan legal untuk berinvestasi aset kriptografi.
   b. Memberikan edukasi dan informasi yang akurat tentang kriptografi kepada masyarakat Indonesia.
   c. Membangun komunitas kriptografi yang kuat dan positif di Indonesia

3.1.1. Struktur organisasi

Struktur organisasi dibentuk untuk mengelola suatu organisasi dalam mencapai tujuan organisasi tersebut. Dengan adanya struktur organisasi, dapat

memberikan dampak positif pada organisasinya, tidak hanya untuk memudahkan dalam mencapai tujuan, antara lain yaitu dapat membagi tugas dan tanggung jawab terhadap individu atau bagian tertentu, lalu memudahkan untuk dalam mendelegasi wewenang. PT. XYZ memiliki struktur organisasi, Gambar 3.1 merupakan bagian struktur organisasi yang lebih berfokus pada divisi *technology*.



Gambar 3.1 Struktur Organisasi PT. XYZ

3.1.2. Uraian tugas

Berdasarkan struktur organisasi PT. XYZ yang telah tertera pada Gambar 3.1, berikut yaitu uraian tugas antara lain:.

1. *Chief Technology Officer*

   Uraian tugas dari posisi ini sebagai berikut:

   a. Merencanakan dan membangun visi dan misi strategi terhadap teknologi berdasarkan visi dan misi perusahaan.
   b. Memimpin tim teknologi secara keseluruhan untuk memastikan strategi teknologi yang diterapkan berjalan dengan efisien.
   c. Mengidentifikasi peluang bisnis yang dapat menguntungkan perusahaan dengan berdasarkan teknologi.

2. *Vice President of Engineering*

   Uraian tugas dari posisi ini sebagai berikut:

   a. Memimpin tim teknologi untuk dapat mencapai visi dan misi dalam strategi teknologi.

b. Melakukan manajemen anggaran terhadap teknologi yang digunakan agar sumber anggaran yang digunakan secara efektif.
c. Mengawasi dan memastikan keamanan dan infrastruktur teknologi aman dari serangan dan *reliable*.
d. Menerapkan dan mengevaluasi proses standar dalam mengembangkan sistem aplikasi.
e. Menyampaikan laporan dari tim teknologi terhadap *stakeholder* seperti CTO atau eksekutif lainnya.

3. *Quality Assurance/Software Development in Test*

   Uraian tugas dari posisi ini sebagai berikut:

   a. Melakukan pengujian terhadap sistem aplikasi, baik secara manual maupun secara otomatis.
   b. Mengidentifikasikan dan melakukan pelaporan terhadap temuan *bug* yang terjadi pada sistem aplikasi terhadap tim *developer* (*Frontend Engineer* atau *Backend Engineer*).

4. *Backend Engineer*

   Uraian tugas dari posisi ini sebagai berikut:

   a. Membangun dan mengembangkan sistem *server-side* dan pula API sebagai perantara komunikasi antara aplikasi *mobile* atau web dengan sistem *backend*.
   b. Memastikan keamanan data pengguna aplikasi *web* dan *mobile* terhadap serangan siber dengan melakukan enkripsi data sensitif.
   c. Merencanakan dan mengembangkan perintah ataupun *query* yang mengakses dan memanipulasi data ke dalam *database*.
   d. Memantau dan mengoptimalkan performa *query* yang mengakses ke *database*, agar tidak mengakibatkan penurunan performa pada *database*.

e. Membuat pengujian dan pemeliharaan kode *backend* dengan menerapkan *unit test* dan pembaharuan *library.*

f. Melakukan *debugging* dan mengoptimalkan sistem *backend*.

5. *Site Reliability Engineer*

   Uraian tugas dari posisi ini sebagai berikut:

   a. Meningkatkan keandalan dari sistem aplikasi dan memastikan sistem aplikasi berjalan secara konsisten.

   b. Menjaga *scalability* dari sistem aplikasi dengan merencanakan dan mengimplementasikan arsitektur sistem yang *scalable*

   c. Bekerja sama dengan anggota *developer* lainnya (*Frontend Engineer* atau *Backend Engineer*) untuk dalam memenuhi kebutuhan dalam membangun sistem aplikasi.

   d. Menerapkan praktik *DevOps* untuk mempercepat dan mengefisiensikan waktu dalam membangun dan *release* sistem aplikasi.

   e. Memantau dan mencegah adanya serangan terhadap sistem aplikasi agar sistem aplikasi tidak mengalami kendala.

   f. Merencanakan dan membangun infrastruktur teknologi berdasarkan kebutuhan dari sistem aplikasi agar dapat menunjang sistem aplikasi yang andal.

   g. Mengelola dan mengoptimalkan sistem aplikasi dan teknologi pendukung seperti *database* agar tidak mengalami *bottleneck*.

6. *Frontend Engineer*

   Uraian dari tugas posisi ini sebagai berikut:

   a. Membangun dan mengembangkan *user interface* yang telah dibuat oleh *product designer* menjadi suatu kode dengan hasil akhir yaitu *web*.

   b. Mengembangkan dan mengimplementasikan interaktif dan fungsionalitas pada *web*.

c. Memastikan *responsivitas web* dapat menyesuaikan terhadap *device* yang digunakan oleh pengguna *web* agar menampilkan tampilan optimal.
d. Mengintegrasikan API yang telah disediakan oleh *Backend Engineer* untuk dapat menampilkan fungsionalitas pada *web*.
e. Menguji dan melakukan *debugging* tampilan dan fungsionalitas web pada berbagai perangkat dan *browser*.

7. *Mobile Engineer*

Uraian dari tugas posisi ini sebagai berikut:

a. Membangun dan mengembangkan *user interface* yang telah dibuat oleh *product designer* menjadi suatu kode dengan hasil akhir yaitu aplikasi *mobile*.
b. Mengembangkan dan mengimplementasikan interaktif dan fungsionalitas pada aplikasi *mobile*.
c. Mengintegrasikan API yang telah disediakan oleh *Backend Engineer* untuk dapat menampilkan fungsionalitas pada aplikasi *mobile*.
d. Menguji dan melakukan *debugging* tampilan dan fungsionalitas aplikasi *mobile* pada berbagai perangkat *mobile*.

3.2. Analisis Prosedur Kerja

Menganalisis sistem prosedur dilakukan untuk menemukan tahapan yang kurang dianggap efisien. Prosedur yang dianggap kurang efisien, akan diidentifikasi kemudian ditelaah untuk perbaharui, ditingkatkan untuk dapat menghasilkan prosedur yang lebih efektif dan berkualitas. Proses analisa prosedur kerja berfokus pada tahapan dalam *database schema migration* yang berada pada ruang lingkup prosedur "Mengimplementasi rancangan aplikasi". Tahapan tersebut bersinggungan dengan 3 entitas, antara lain *Site Reliability Engineer*, *Backend Engineer* dan VP *of Engineering*.

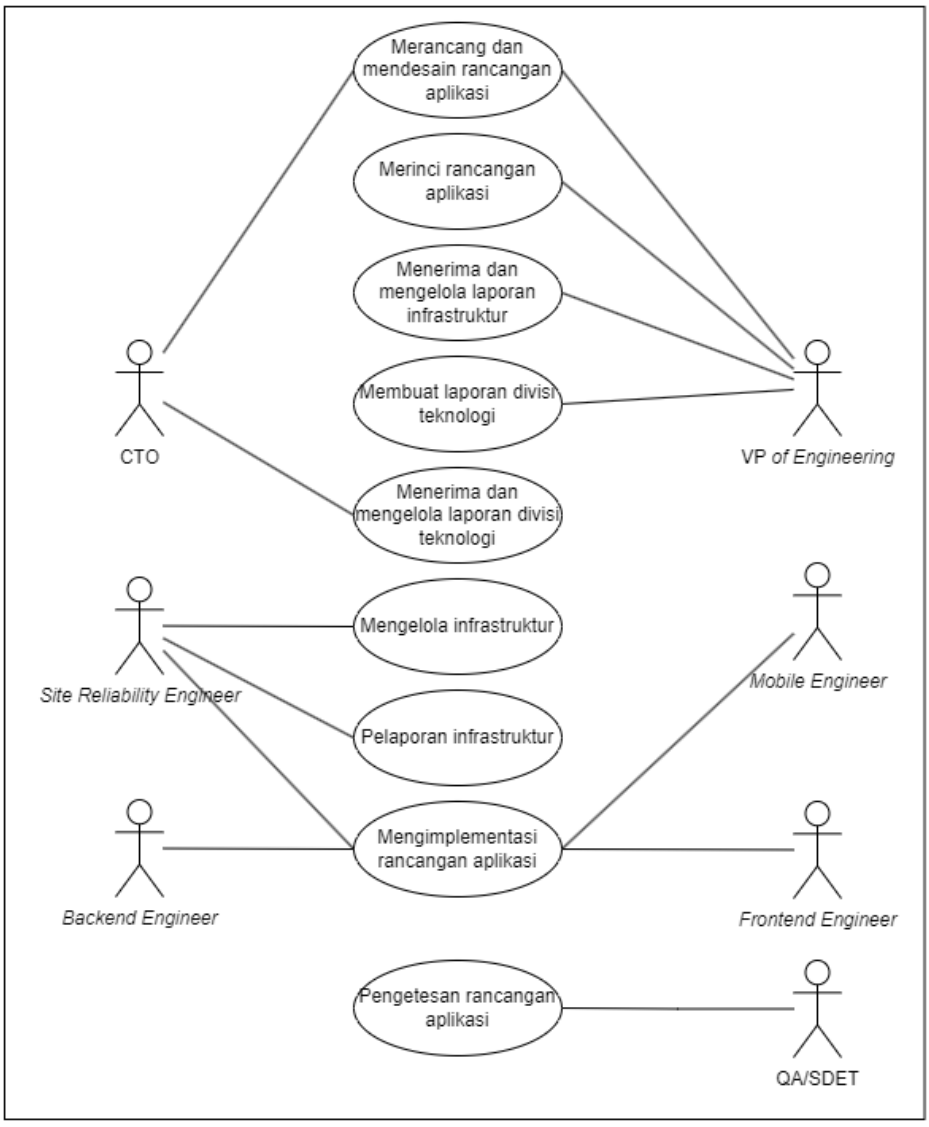
3.2.1. Deskripsi prosedur kerja

Prosedur kerja dari tahapan *database schema migration* dijelaskan secara deskriptif yang meliputi:

1. *Backend Engineer* memerlukan *database schema migration* untuk kebutuhan proses *development application*, lalu melakukan permintaan *database schema migration* kepada *Site Reliability Engineer* melalui *platform Slack* dengan melampirkan file *script* berupa SQL DML/DDL.
2. *Site Reliability Engineer* menerima permintaan dari *Backend Engineer* pada *platform Slack*.
3. Setelah itu, *Site Reliability Engineer* menggunakan *tool PHPMyAdmin* untuk mengeksekusi *database schema migration*.
4. Setelah selesai mengeksekusi *query* SQL DML/DDL tersebut, *Site Reliability Engineer* menginformasikan terhadap *Backend Engineer* melewati *platform Slack* bahwa *file script* SQL DDL/DML sudah dieksekusi.
5. Kemudian dalam kurun waktu satu bulan sekali, *Site Reliability Engineer* akan melakukan pelaporan historis *database schema migration* berdasarkan data permintaan *database schema migration* yang bersumber dari *platform Slack* yang nantinya akan dilaporkan kepada VP *of Engineering*.

3.2.2. *Business use case*

PT. XYZ mempunyai *business use case* yang berfokus pada divisi *technology.* Berikut merupakan bisnis prosedur kerja yang akan dijelaskan dalam menggunakan diagram yang dapat dilihat pada Gambar 3.2.



Gambar 3.2 *Business Use Case*

Seperti yang telah dijelaskan pada sub-bab 3.2, dalam pembahasan analisis prosedur kerja, tahapan *database schema migration* dilakukan pada prosedur "Mengimplementasi rancangan aplikasi", namun untuk pelaporan mengenai tahapan *database schema migration* yang dilaporkan kepada VP *of Engineering* berada pada prosedur "Pelaporan infrastruktur" dan prosedur "Menerima dan mengelola laporan infrastruktur".

3.3. Deskripsi Dokumen

Adanya dokumen pada prosedur berguna untuk mendukung tahapan tersebut, terlebih pula prosedur yang dilakukan secara rutin. Tautan yang terdapat dokumen memuat hal-hal yang dapat membantu dalam melakukan prosedur. Tanpa adanya dokumen dalam suatu tahapan, maka tindakan untuk melakukan pencatatan atau pengarsipan akan sulit untuk dilakukan. Pada prosedur yang telah dijelaskan pada bagian 3.2, PT. XYZ menggunakan juga dokumen dalam mendukung prosedurnya. Berikut merupakan penjelasan terkait dokumen yang digunakan oleh PT. XYZ.

1. Nama dokumen : Menjelaskan nama dari dokumen
2. Fungsi : Menjelaskan fungsi dari dokumen
3. Sumber : Menjelaskan sumber dokumen dibuat
4. Frekuensi : Menjelaskan tanggal pembuatan dokumen
5. Rangkap : Menjelaskan total salinan dokumen
6. Distribusi : Menjelaskan penerima dokumen
7. Isi dokumen : Menjelaskan data yang termuat dalam dokumen

Berdasarkan penjelasan dokumen yang digunakan oleh PT. XYZ, berikut merupakan penjelasan terkait dokumen yang digunakan dalam prosedurnya, antara lain

1. Nama dokumen : Catatan SQL *query* DDL/DML
   Fungsi : Informasi terkait *query* SQL DDL/DML
   Sumber : *Backend Engineer*
   Frekuensi : Ketika *Backend Engineer* membutuhkan untuk mengubah atau memperbaharui *database schema*
   Rangkap : Satu (1)
   Distribusi : *Site Reliability Engineer*
   Isi dokumen : *Command query* SQL DDL/DML

3.4. Analisis dan Evaluasi Sistem

Berdasarkan data dan informasi yang telah didapatkan seperti tahapan dari prosedur dan dokumen terkait yang telah dijelaskan bagian sebelumnya, langkah selanjutnya akan adanya menganalisis dan mengevaluasi prosedur pada proses tersebut, lalu memberikan solusi untuk meningkatkan tahapan prosedur kerja agar lebih efisien.

3.4.1. Tinjauan sistem komputer

Setelah ditelusuri mengenai prosedur yang dilakukan oleh PT. XYZ, PT. XYZ menggunakan sistem komputer dalam menjalankan prosedurnya. Untuk lebih lengkapnya perangkat dari sistem yang digunakan dalam menjalankan prosedurnya, dapat dilihat pada Tabel 3.1.

Tabel 3.1 Tinjauan Sistem Komputer PT. XYZ

No.	Komponen	Spesifikasi
1	*Database Management System* (DBMS)	MySQL v5.7
2	*PHPMyAdmin*	Versi 5

3.4.2. Evaluasi sistem

Berdasarkan tahapan prosedur kerja yang telah dijelaskan sebelumnya dan dilakukan analisa, terdapat beberapa hal kekurangan dari tahapan prosedur kerja tersebut. Antara lain yaitu, yang pertama, saat melakukan *database schema migration* secara manual tidak dianggap efisien dikarenakan tahapan prosedur kerja ini akan sering dilakukan dan dikhawatirkan dapat menimbulkan *human error* atau kesalahan personal dalam melakukan tahapan prosedur kerja. Yang kedua yaitu, tidak adanya detail informasi terkait *database schema migration* yang tercatat dalam *Git*. Yang ketiga yaitu, tidak ada pencatatan historis mengenai *database schema migration*.

3.5. Analisis Kebutuhan Sistem

Setelah melakukan evaluasi terhadap sistem prosedur, dan mendapatkan beberapa kekurangan dari prosedur tersebut, maka diperlukan tindakan perubahan untuk

memperbaiki tahapan prosedur yang dianggap kurang efektif dan perlu optimalisasi agar prosedur tersebut menjadi lebih efisien. Tahapan untuk melakukan suatu perubahan tersebut diperlukan analisis terlebih dahulu kebutuhan tahapan prosedur kerja. Maka daripada itu, berdasarkan tahapan prosedur kerja yang berlaku pada PT. XYZ akan dijelaskan beberapa poin-poin untuk dalam analisis kebutuhan sistem pada bagian setelah ini.

3.5.1. Kebutuhan informasi

Setiap tahapan prosedur kerja adanya data yang dapat dijadikan suatu informasi, yang nantinya dapat digunakan antara lain sebagai pencatatan, pelaporan, atau audit. Berdasarkan prosedur pada PT. XYZ, informasi yang dibutuhkan dalam prosedur yang berlaku dapat dilihat pada Tabel 3.2.

Tabel 3.2 Kebutuhan Informasi

No.	Informasi yang Dibutuhkan	Tujuan	Frekuensi
1	*Database Schema Migration*		
	a. Nama *Backend Engineer* yang melakukan *request*	*Site Reliability Engineer*	Setiap ada *request* perubahan *database schema*
	b. Nama *database* dan *table* yang perlu diubah	*Site Reliability Engineer*	Setiap ada *request* perubahan *database schema*
	c. Waktu pengeksekusian *database schema migration* berlangsung	*Site Reliability Engineer*	Setiap ada *request database schema migration*
	d. Status pengeksekusian berlangsung	*Backend Engineer*	Setiap selesai eksekusi *database schema migration*
	e. Laporan historis *database schema migration*	VP *of Engineering*	Setiap bulan

3.5.2. Kebutuhan program aplikasi

Data kebutuhan informasi telah dijelaskan pada Tabel 3.2, akan digunakan untuk acuan dalam membuat suatu program aplikasi. Penjelasan terkait program aplikasi yang akan dibuat untuk mengoptimalkan prosedur yang berlaku pada PT. XYZ, dapat dilihat pada Tabel 3.3.

Tabel 3.3 Kebutuhan Program Aplikasi

No.	Deskripsi Kebutuhan	Keterangan
1	Mengelola data *database schema migration*	Tambah, tampilkan, batal, simpan
2	Mengelola laporan	Tambah dan cetak

3.5.3. Kebutuhan perangkat keras

Perangkat keras yang dibutuhkan dalam menjalankan sistem proses, dapat dilihat pada Tabel 3.4.

Tabel 3.4 Kebutuhan Perangkat Keras

No.	Nama Perangkat	Detail
1	Laptop/PC	Spesifikasi minimum yang memiliki: a. Intel Core i3 atau AMD Ryzen 3 b. RAM 4 GB c. Penyimpanan 140GB d. OS Windows 10/11 atau Ubuntu 22.04

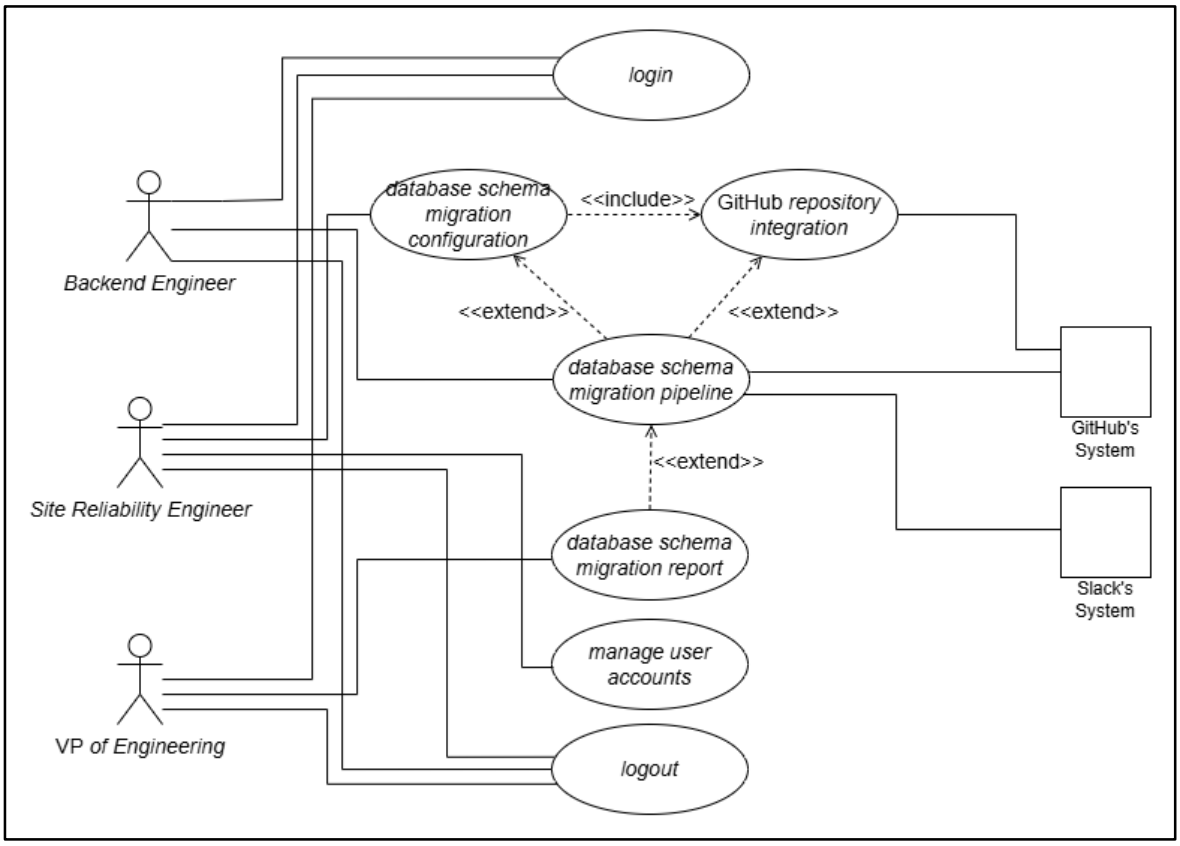
3.6. Pemodelan Berorientasi Objek

Sebelum membuat program aplikasi, akan ada pembahasan terkait pemodelan fungsional yang akan membantu dalam perancangan program aplikasi. Pemodelan fungsional ini akan merencanakan proses pada suatu prosedur yang nantinya akan memanfaatkan program aplikasi dalam menunjang optimalisasi prosedur yang lebih efisiensi. Hal-hal tersebut akan disampaikan pada pembahasan setelah ini.

3.6.1. *System use case*

Pembahasan pertama terkait hal yang akan dirincikan prosedur pada PT. XYZ yaitu diagram konteks. Diagram konteks merupakan penggambaran dari aliran *input* dan *output* dari suatu program aplikasi. Diagram konteks memiliki relasi dengan entitas *external*, yang di mana entitas tersebut berperan untuk berinteraksi langsung dengan suatu sistem.

Berikut diagram *system use case* dari tahapan prosedur kerja *database schema migration* yang dapat dilihat pada Gambar 3.3.



Gambar 3.3 *System Use Case*

3.6.2. *Use case scenario*

Dengan berlandaskan penjelasan sebelumnya dan Gambar 3.3 mengenai *system use case*, maka akan dijabarkan pada *use case scenario*. Pada penjelasan ini, prosedur akan dijelaskan dalam bentuk skenario tindakan setiap prosedur dari aktor atau entitas yang terlibat. Berikut penjelasan dari *use case scenario*:

1. *Login*

   Nama skenario : *Login*

   Aktor : *Backend Engineer*, *Site Reliability Engineer* dan VP *of Engineering*

   Pre-Condition : Aktor sudah mempunyai akun

Tabel 3.5 *Use case login*

Aksi Aktor	Reaksi Sistem
1. Mengakses laman web	
	2. Menampilkan halaman *login*

Aksi Aktor	Reaksi Sistem
3. Input *data username* dan *password*	
4. Klik tombol *"Login"*	
	5. Validasi data input *user*
	6. Menampilkan halaman *home*
Post-Condition	Aktor dapat *login* dan berada halaman *home*
Skenario Alternatif: Data input user tidak valid	
Tahapan 1-5 sama dengan skenario awal	
	6. Menampilkan pesan *error* bahwa data input user tidak valid
Post Condition	Aktor tidak dapat *login*

2. GitHub *repository integration*

Nama skenario : GitHub *repository integration*

Aktor : *Site Reliability Engineer* (SRE) dan GitHub

Pre-Condition : SRE sudah *login* dan berada pada tampilan *home*

Tabel 3.6 *Use case* GitHub *repository integration*

Aksi Aktor	Reaksi Sistem
1. SRE klik tab *"Repository List*"	
	2. Menampilkan laman *repository list*
3. SRE klik tombol *"Add New Repo"*	
	4. Menampilkan laman *new repository integration*
5. SRE menginputkan data *name*, *username*, *token*, dan GitHub *repository* URL	
6. SRE Klik tombol "*Submit*"	
7. GitHub memvalidasi data input *repository integration*	
	8. Mencatat data input ke data *repository integration*
	9. Menampilkan laman *repository list*
Post-Condition	Berhasil menambahkan data *repository integration*. Proses ini dilakukan sekali saja untuk setiap *repository*
Skenario Alternatif 1: Data input *repository integration* tidak valid	
Tahapan 1-7 sama dengan skenario awal	
	8. Menampilkan pesan bahwa data input *repository integration* tidak valid
Post-Condition	Gagal menambahkan data *repository integration*

Aksi Aktor	Reaksi Sistem
Skenario Alternatif 2: Memperbaharui data *repository integration*	
Tahapan 1-2 sama dengan skenario awal	
3. SRE klik *icon* "*edit*" pada *repository* yang akan diperbaharui	
	4. Menampilkan laman *edit repository integration*
5. SRE memperbaharui data *repository integration*	
6. SRE klik tombol "*Update*"	
7. GitHub memvalidasi data *updated repository integration*	
	8. Memperbaharui data *repository integration*
	9. Menampilkan laman *repository list*
Post-Condition	Data *repository integration* telah diperbaharui
Skenario Alternatif 3: Memperbaharui data *repository integration* tapi tidak valid	
Tahapan 1-7 sama dengan skenario alternatif 2	
	8. Menampilkan pesan bahwa data *updated repository integration* tidak valid
Post-Condition	Gagal memperbaharui data *repository integration*
Skenario Alternatif 4: Menghapus data *integration repository*	
Tahapan 1-2 sama dengan skenario awal	
3. SRE klik *icon* "*Trash*" pada data *repository integration* yang akan dihapus	
	4. Menampilkan pesan konfirmasi untuk menghapus data *repository integration*
5. SRE mengkonfirmasi untuk menghapus data *repository integration*	
	6. Menghapus data *repository integration*
	7. Menampilkan laman *repository list*
Post-Condition	Data *repository integration* sudah dihapus

3. *Database schema migration config*

   Nama skenario : *Database schema migration config*

   Aktor : *Site Reliability Engineer*

Pre-Condition : Aktor sudah menambahkan *repository integration* dan berada di tampilan *home*

Tabel 3.7 *Use case database schema migration config*

Aksi Aktor	**Reaksi Sistem**
1. Klik tab "*Migration Config*"	
	2. Menampilkan laman *migration config*
3. Klik tab "*Add New Migration Config*"	
	4. Menampilkan *new migration config*
5. *Input* data *folder location*, *database host*, *database username*, *database name*, *database password*, *repository*	
6. Klik tombol "*Submit*"	
	7. Validasi data input *migration config*
	8. Mencatat data ke input ke data *migration config*
	9. Menampilkan laman *migration config*
Post-Condition	Berhasil menambahkan data *migration config*
Skenario Alternatif 1: Data masukkan *migration config* tidak valid	
Tahapan 1-7 sama dengan skenario awal	
	8. Menampilkan pesan bahwa data input *migration config* tidak valid
Post-Condition	Gagal menambahkan *migration config*
Skenario Alternatif 2: Memperbaharui data *migration config*	
Tahapan 1-2 sama dengan skenario awal	
3. Klik *icon* "*Edit*" pada data *migration config* yang akan diperbaharui	
	4. Menampilkan laman *edit migration config*
5. Perbaharui data *migration config*	
	6. Validasi data *updated migration config*
	7. Memperbaharui data *migration config*
	8. Menampilkan laman *migration config*
Post-Condition	Data *migration config* telah diperbaharui

Aksi Aktor	Reaksi Sistem
Skenario Alternatif 3: Memperbaharui data *migration config* tapi tidak valid	
Tahapan 1-6 sama dengan skenario alternatif 2	
	7. Menampilkan *error* bahwa data input *migration config* tidak valid
Post-Condition	Gagal memperbaharui data *migration config*
Skenario Alternatif 4: Menghapus data *migration config*	
Tahapan 1-2 sama dengan skenario awal	
3. Klik *icon* "*Trash*" pada salah satu data *migration config*	
	4. Menampilkan *pop-up* pesan untuk menghapus data *migration config*
5. Menyetujui menghapus data *migration config*	
	6. Menghapus data *migration config*
	7. Menampilkan laman *migration config*
Post-Condition	Data *migration config* telah terhapus

4. *Database schema migration pipeline*

Nama skenario : *Database schema migration pipeline*

Aktor : *Backend Engineer* (BE), GitHub dan Slack

Pre-Condition : SRE sudah menambahkan *repository integration* dan *migration config*

Tabel 3.8 *Use case database schema migration pipeline*

Aksi Aktor	Reaksi Sistem
1. BE membuat *query* SQL DDL/DML	
2. BE menambah *file script* dari *query* SQL DDL/DML tadi ke GitHub *repository*	
3. GitHub menjalankan proses CI/CD	
4. GitHub CI/CD proses pertama melakukan validasi SQL *query*	
5. GitHub CI/CD proses kedua melakukan *webhook* ke API aplikasi	
	5. Eksekusi *database schema migration*
	6. Mencatat historis eksekusi *database schema migration*

Aksi Aktor	Reaksi Sistem
	7. Notifikasi ke *platform Slack* bahwa eksekusi *database schema migration* telah berhasil dan selesai
8. Slack menerima notifikasi sukses eksekusi *database schema migration*	
Post-Condition	*Database schema migration* berhasil dieksekusi
Skenario Alternatif 1: SQL *query* tidak valid	
Tahapan 1-4 sama dengan skenario awal	
5. GitHub CI/CD mengirim notifikasi ke *platform Slack* bahwa SQL *query* tidak valid	
6. Slack Menerima notifikasi bahwa SQL *query* tidak valid	
Post-Condition	*Database schema migration* tidak dieksekusi
Skenario Alternatif 2: *Database schema migration* tidak berhasil	
Tahapan 1-6 sama dengan skenario awal	
	7. Notifikasi ke *platform Slack* bahwa eksekusi *database schema migration* gagal
8. Slack menerima notifikasi gagal eksekusi *database schema migration*	
Post-Condition	Eksekusi *database schema migration* gagal

5. *Database schema migration report*

Nama skenario : *Database schema migration report*

Aktor : VP *of Engineering* dan *Site Reliability Engineer*

Pre-Condition : *Database schema migration* telah berhasil dilakukan sedang berada di halaman *home*

Tabel 3.9 *Use case database schema migration report*

Aksi Aktor	Reaksi Sistem
1. Klik tab "*Migration History*"	
	2. Menampilkan data historis
Post-Condition	Menampilkan data historis
Skenario Alternatif 1: Eksport data *migration schema* untuk 1 hari lalu	
Tahapan 1-2 sama dengan skenario awal	

Aksi Aktor	Reaksi Sistem
3. Klik tombol "*Export* PDF"	
4. Pilih interval waktu 1 hari	
5. Klik tombol "OK"	
	6. Ekspor data *migration schema* 1 hari lalu dalam bentuk *file* PDF
Post-Condition	Ekspor data *migration schema* dalam kurun 1 hari lalu berupa *file* PDF
Skenario Alternatif 2: Eksport data *migration schema* untuk 1 minggu lalu	
Tahapan 1-2 sama dengan skenario awal	
3. Klik tombol "*Export* PDF"	
4. Pilih interval waktu 1 minggu	
4. Klik tombol "OK"	
	5. Ekspor data *migration schema* 1 hari lalu dalam bentuk *file* PDF
Post-Condition	Ekspor data *migration schema* dalam kurun 1 minggu lalu berupa *file* PDF
Skenario Alternatif 3: Eksport data *migration schema* untuk 1 bulan lalu	
Tahapan 1-2 sama dengan skenario awal	
3. Klik tombol "*Export* PDF"	
4. Pilih interval waktu 1 bulan	
4. Klik tombol "OK"	
	5. Ekspor data *migration schema* 1 bulan lalu dalam bentuk *file* PDF
Post-Condition	Ekspor data *migration schema* dalam kurun 1 bulan lalu berupa *file* PDF

6. *Manage user accounts*

Nama skenario : *Manage user accounts*

Aktor : *Site Reliability Engineer*

Pre-Condition : Sudah *login* dan berada pada tampilan *home*

Tabel 3.10 *Use case manage user accounts*

Aksi Aktor	Reaksi Sistem
1. Klik tab *"Users*"	
	2. Menampilkan lawan *users*
3. Klik tombol *"Add New User"*	
	4. Menampilkan laman "*Add New User*"

Aksi Aktor	Reaksi Sistem
4. *Input* data *form fullname*, *username*, *password*, *role user*	
	5. Mencatat data user baru
	6. Menampilkan laman *users*
Post-Condition	Data *user* baru sudah dibuat
Skenario Alternatif 1: Memperbaharui data *user*	
Tahapan 1-2 sama dengan skenario awal	
3. Klik *icon* "*Edit*" pada salah satu *user*	
	4. Menampilkan laman *edit users*
5. Perbaharui data *user*	
6. Klik tombol "*Update*"	
	7. Mencatat perubahan data akun
	8. Menampilkan laman *users*
Post-Condition	Data *user* berhasil diperbaharui
Skenario Alternatif 2: Menghapus user	
Tahapan 1-2 sama dengan skenario awal	
3. Klik *icon* "*Trash*" pada salah satu akun	
	4. Menampilkan *pop-up* pesan untuk menghapus akun
5. Menyetujui menghapus akun	
	6. Menghapus data akun
	7. Menampilkan laman *users*
Post-Condition	Akun telah terhapus

7. Logout

Nama skenario : *Logout*

Aktor : VP *of Engineering, Backend Engineer* dan *Site Reliability Engineer*

Pre-Condition : Aktor sudah *login* dan berada pada tampilan *home*

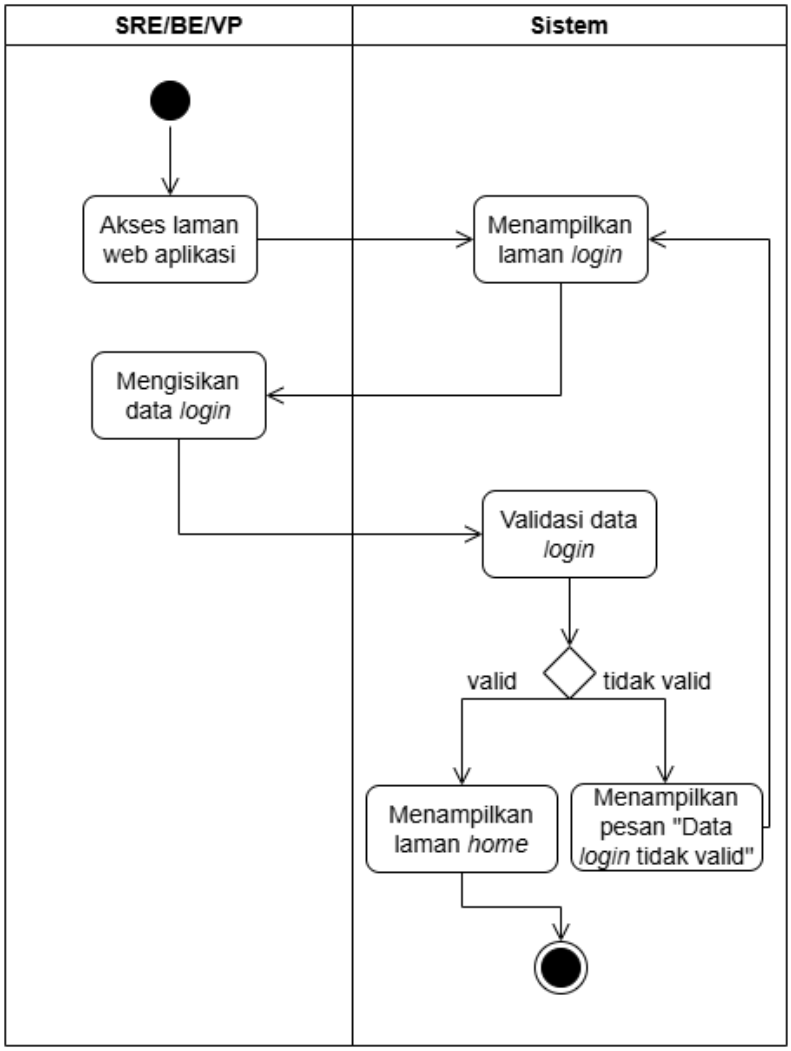
Tabel 3.11 *Use case logout*

Aksi Aktor	Reaksi Sistem
1. Klik tab "*Logout*"	2. Menampilkan *pop-up* pesan untuk keluar dari aplikasi
3. Menyetujui untuk keluar dari aplikasi	
	4. Menampilkan halaman *login*
Post-Condition	Sesi *login* telah berakhir dan akun telah keluar dari aplikasi

3.6.3. *Activity diagram*

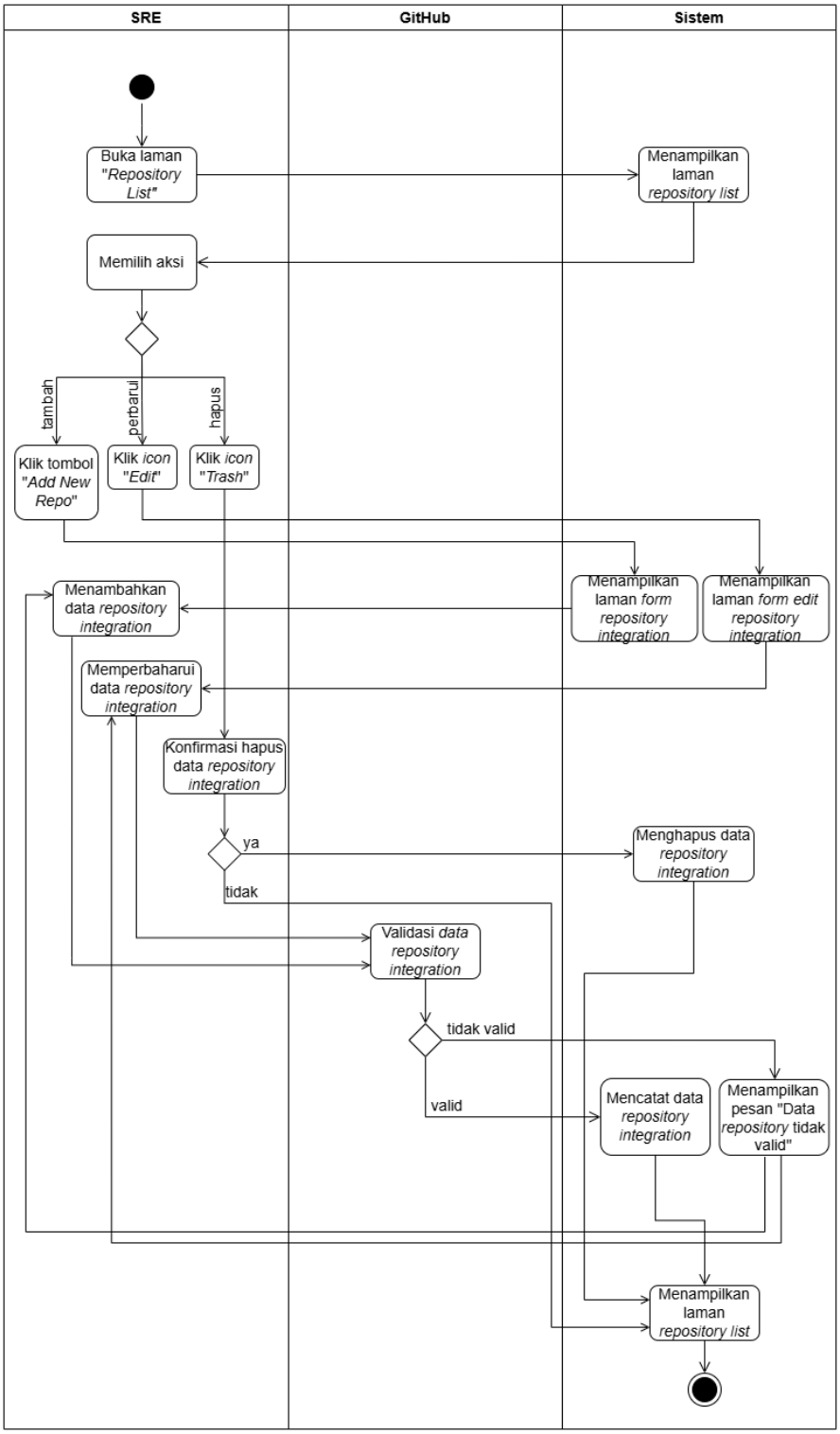
Masih berlandaskan pada *system use case* pada Gambar 3.3, pada bagian ini akan membahas prosedur dalam bentuk *activity diagram*. *Activity diagram* yang memvisualisasikan aliran kerja setiap prosedur. Berikut penjelasan dari *activity diagram*:

1. *Login*



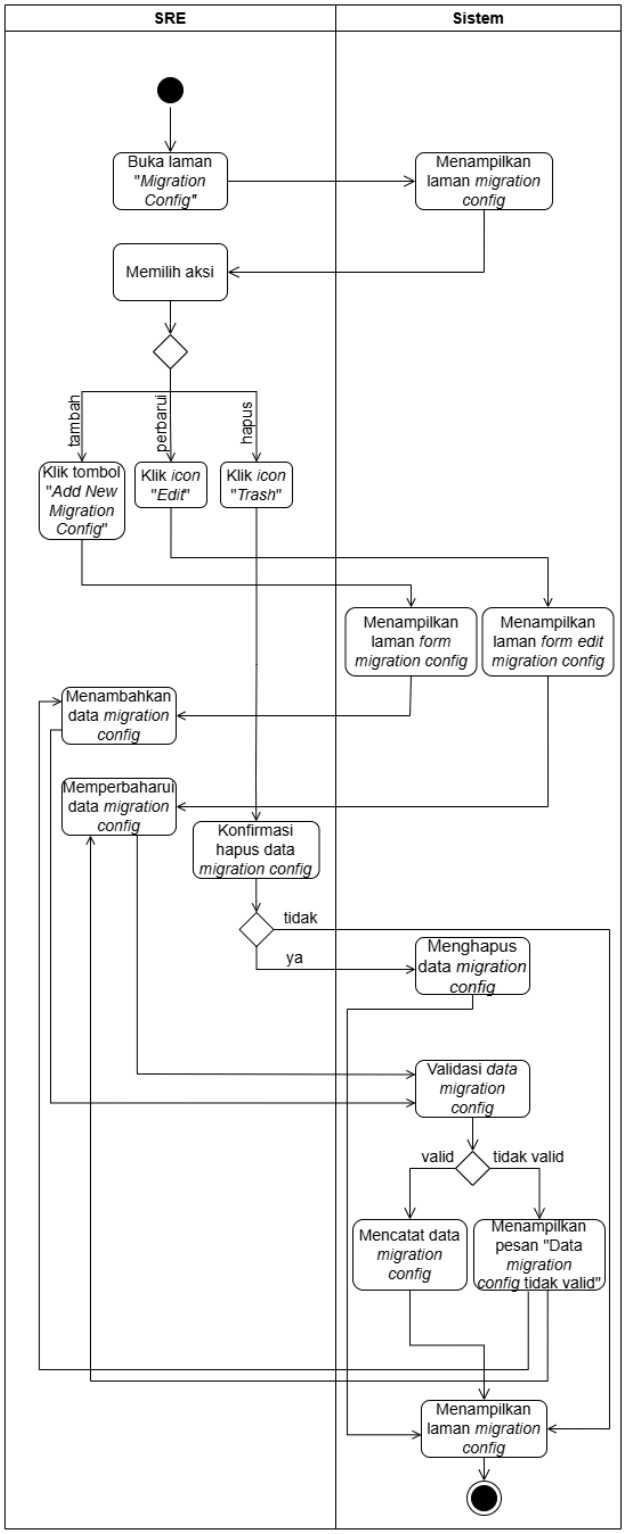
Gambar 3.4 *Activity diagram login*

2. GitHub *repository integration*



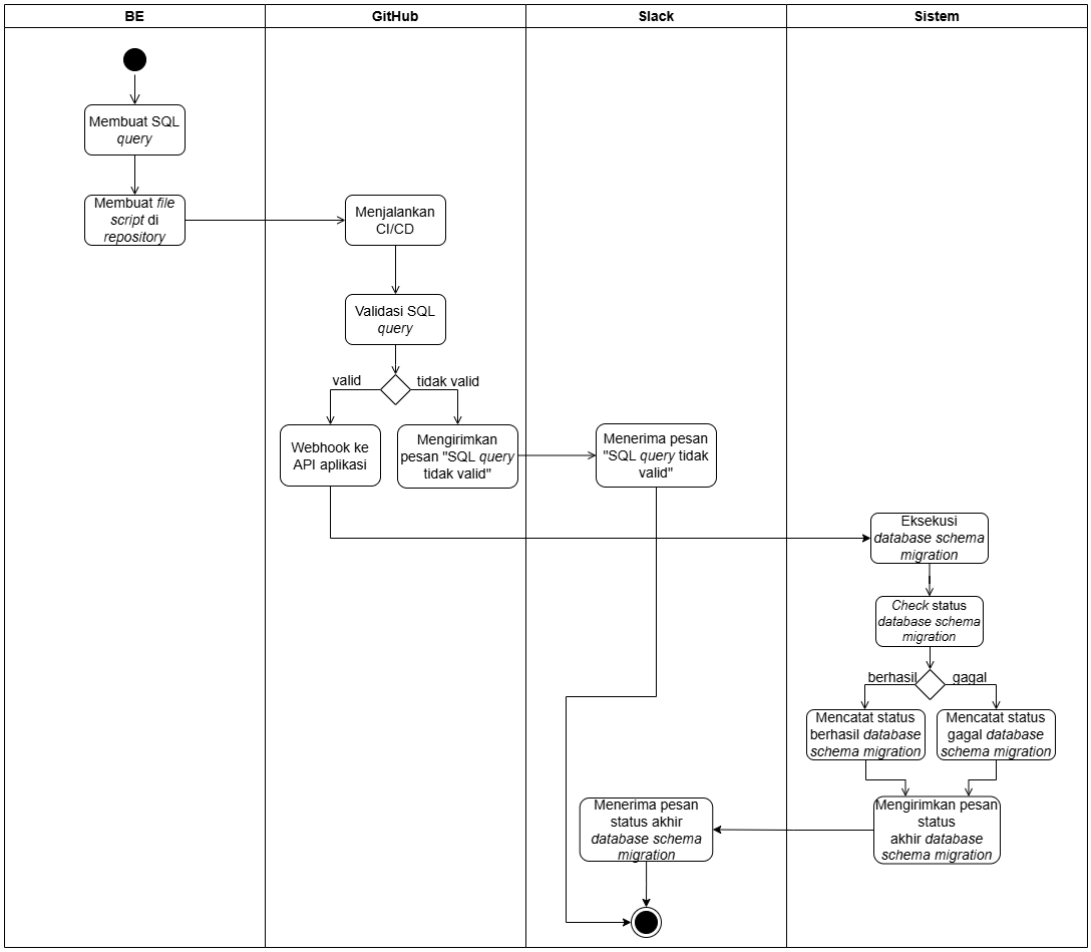
Gambar *3.5 Activity diagram* GitHub *repository integration*

3. *Database schema migration configuration*



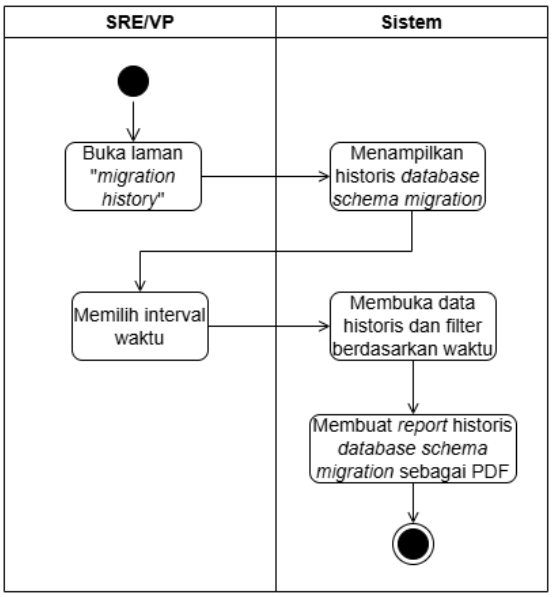
Gambar 3.6 Activity diagram database schema migration configuration

4. *Database schema migration pipeline*



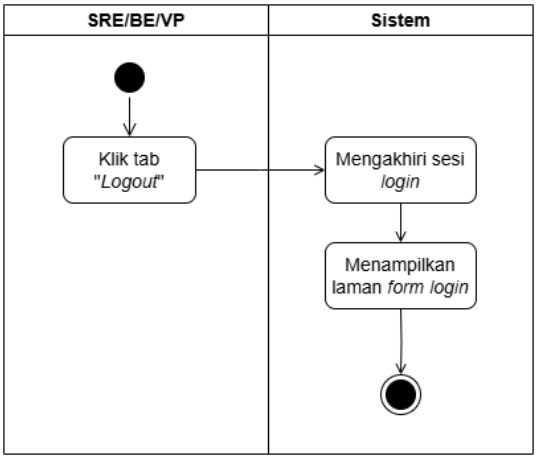
Gambar 3.7 *Activity diagram database schema migration pipeline*

5. *Database schema migration report*



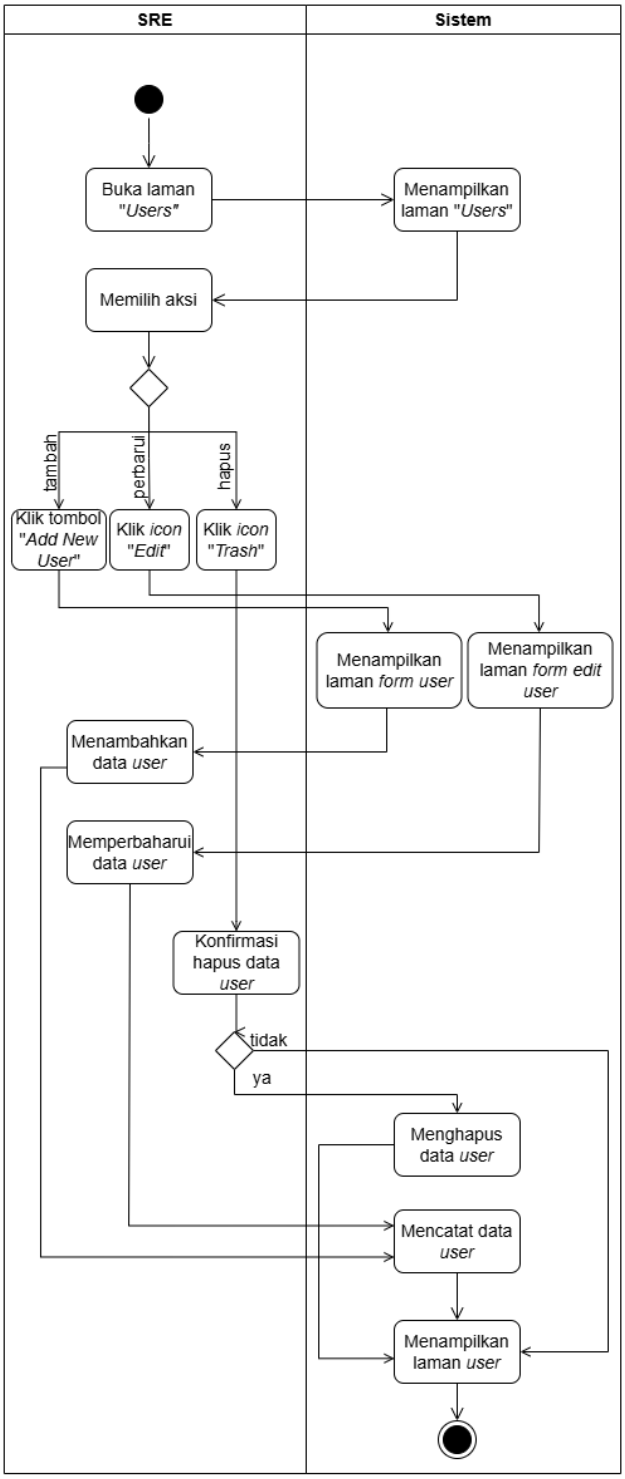
Gambar 3.8 *Activity diagram database schema migration report*

6. *Logout*



Gambar 3.9 *Activity diagram logout*

7. *Manage user accounts*



Gambar 3.10 *Activity diagram manage user accounts*

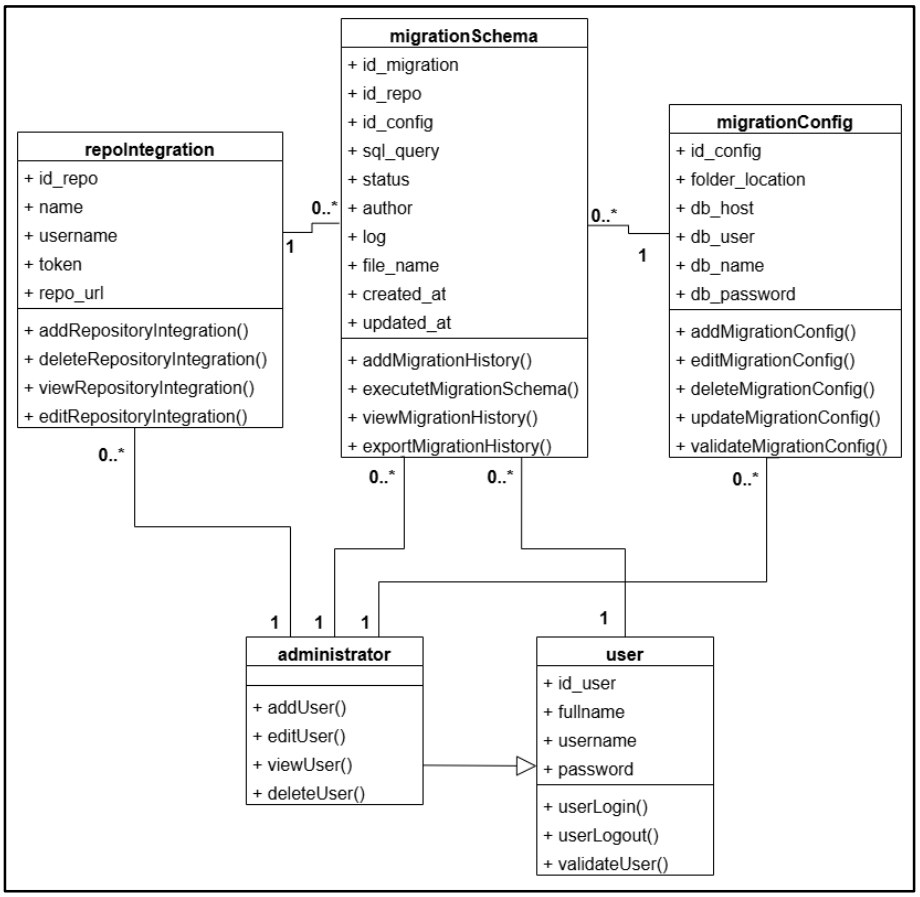
BAB IV
PERANCANGAN SISTEM INFORMASI

4.1. Perancangan Berorientasi Objek

Sebelum pengembangan mengembangkan aplikasi lama menjadi aplikasi baru, diperlukannya suatu rancangan agar dapat mencapai kebutuhan dan tujuan dari pengembangan aplikasi baru. Dengan merujuk pengembangan aplikasi berbasiskan perancangan berorientasi objek, akan dibutuhkan beberapa rancangan diagram agar menunjang pengembangan aplikasi tersebut. Antara lain yaitu *class diagram*, *sequence diagram* dan sistem pengkodean. Dari ketiga diagram tersebut memiliki fungsi dalam pengembangan aplikasi ini, antara lain *class diagram* sebagai *blueprint* dari aplikasi, *sequence diagram* sebagai skenario aktivitas yang terdapat pada aplikasi, dan sistem pengkodean sebagai pengkodean berdasarkan kedua diagram sebelumnya. Rancangan diagram yang sudah dibentuk untuk pengembangan aplikasi baru ini, sebagai berikut:

4.1.1. *Class diagram*

Class diagram menunjukkan rancangan keterhubungan antar *class*, dan setiap *class* tersebut juga dijelaskan kemampuan atau metode yang bisa dilakukan. Rancangan *class diagram* tersebut dapat dilihat pada gambar 4.1.



Gambar 4.1 *Class Diagram*

Berdasarkan pada Gambar 4.1, terdapat mempunyai 5 *class*, antara lain:

a. *Class* "user" digunakan untuk menyimpan data user yang tidak memiliki *role* sebagai administrator.

b. *Class* "administrator" merupakan turunan dari *parent class* "user". Yang membedakan *class* "administrator" dan "user" yaitu *class* "administrator" dapat melakukan fungsi seperti menambahkan, mengubah, dan menghapus *user*.

c. *Class* "repoIntegration" berfungsi untuk mengelola data *repository* dan membangun integrasi dengan GitHub *repository*.

d. *Class* "migrationConfig" berfungsi untuk mengelola *migration configuration* sebagai landasan untuk menjalankan *database schema*

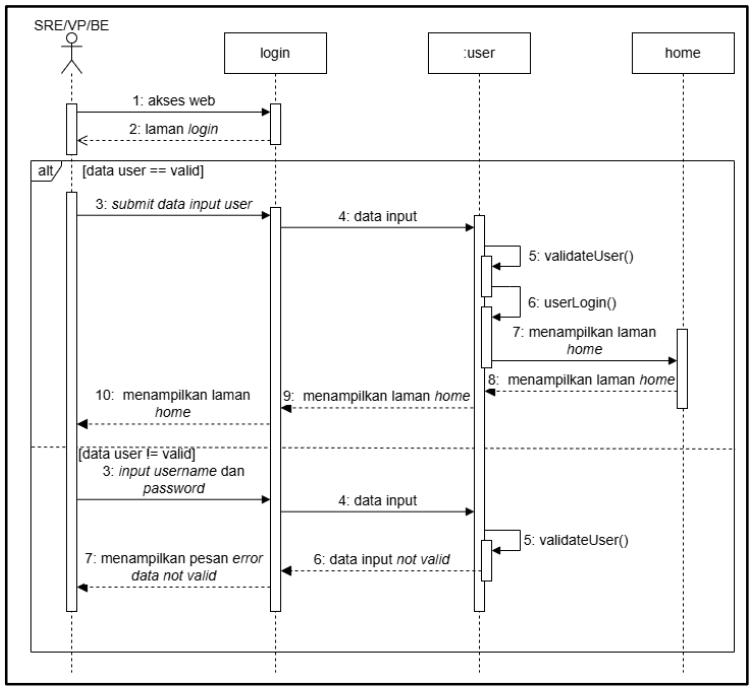
migration, dan dalam *class* ini disertai relasi data terhadap *class* “repoIntegration” yang membutuhkan data nama *repository*.

e. *Class* “migrationSchema” berfungsi untuk eksekusi *database schema migration* yang berlandaskan dari *state* yang telah ditambahkan pada *class* “repoIntegration” dan *class* “migrationConfig”, dan menampung data historis *database schema migration*. .

4.1.2. *Sequence diagram*

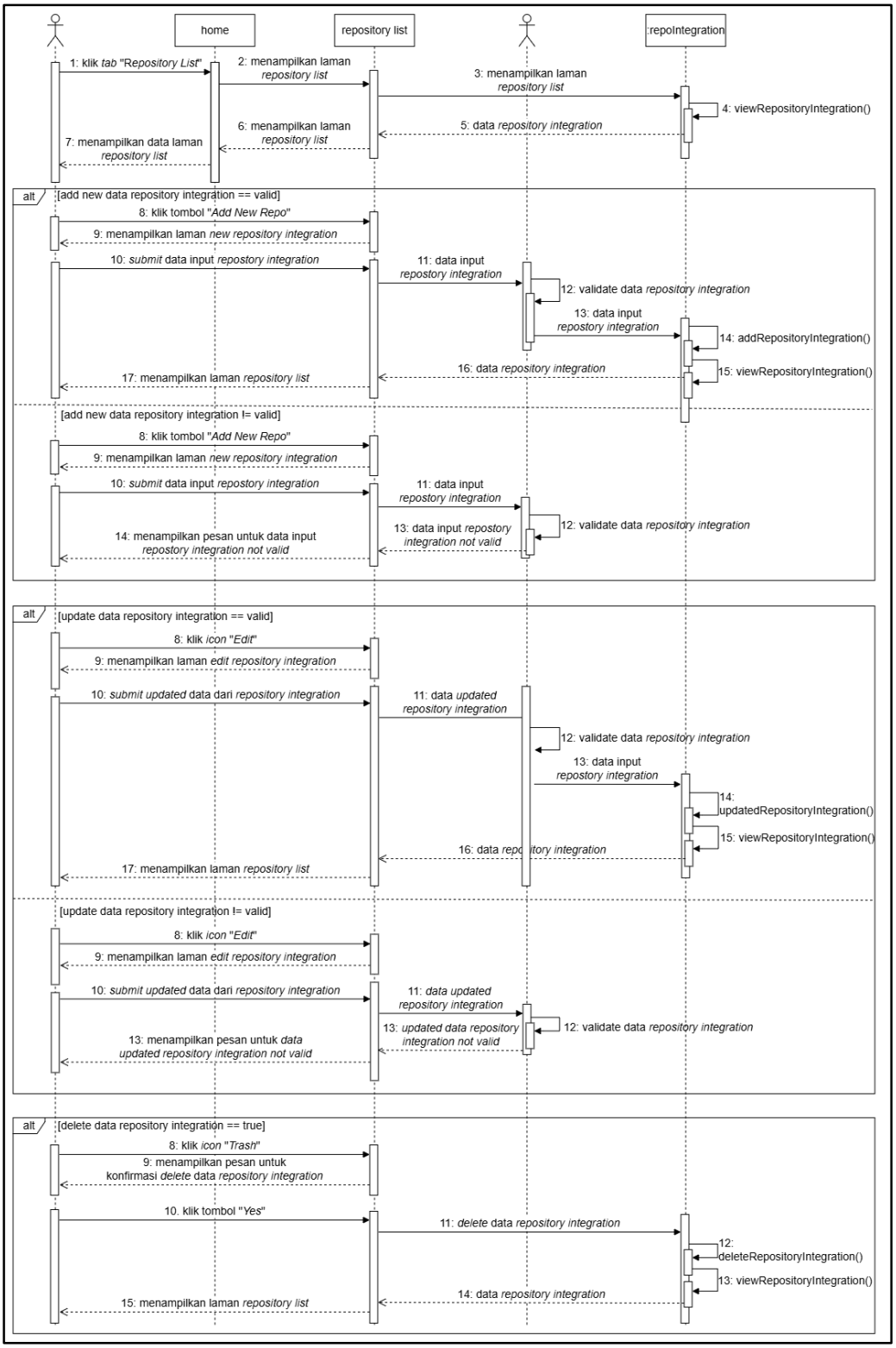
Sequence diagram akan menjelaskan aktivitas secara berurutan yang mengacu pada data dari *system use case*. Pada *system use case* yang sudah dibuat pada bab 3, terdapat 7 *case* dan *sequence diagram* akan menjelaskan 7 *sequence diagram*. Untuk lebih lanjut berikut *sequence diagram* pada aplikasi ini:

1. Login



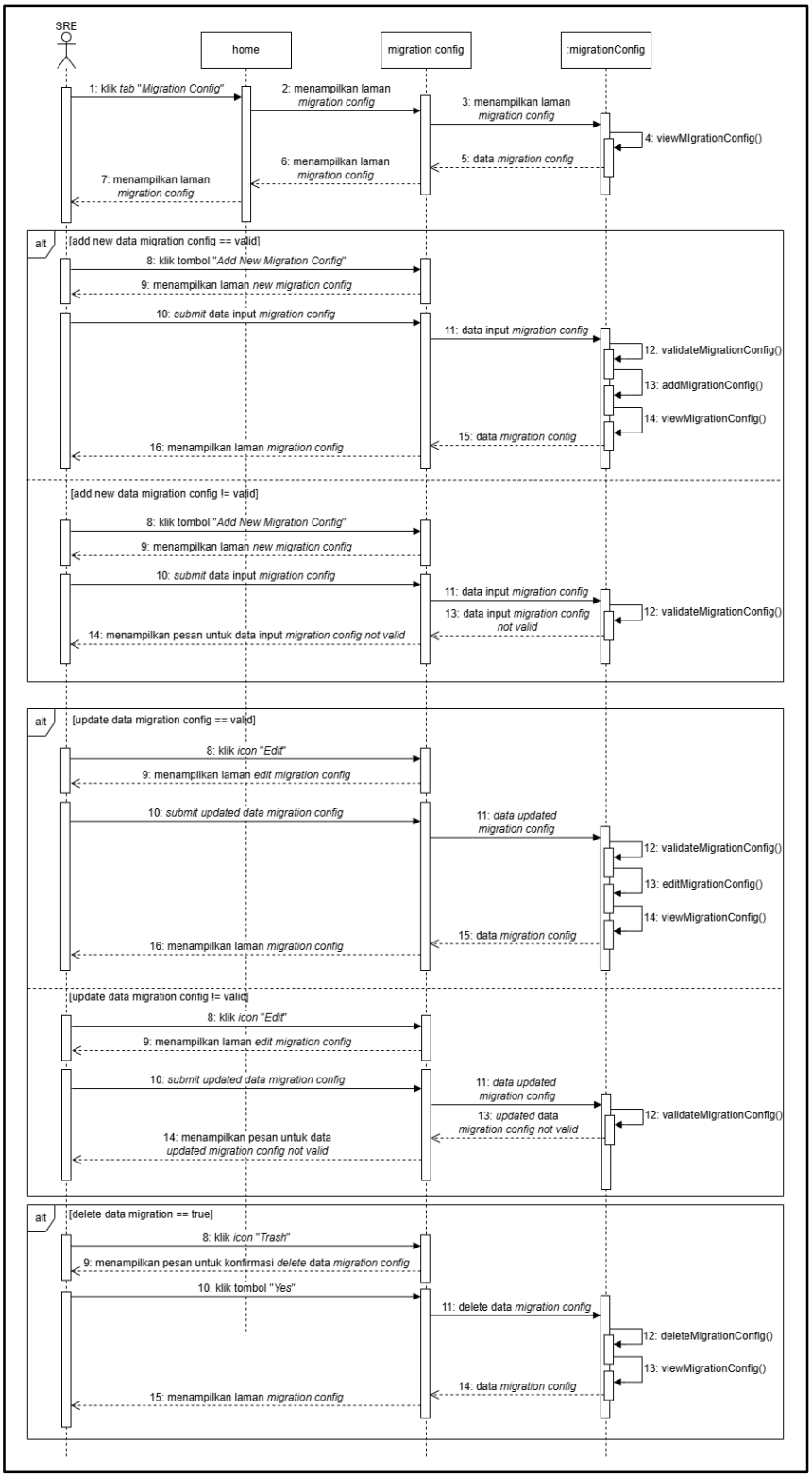
Gambar 4.2 *Login Sequence Diagram*

2. GitHub *repository integration*



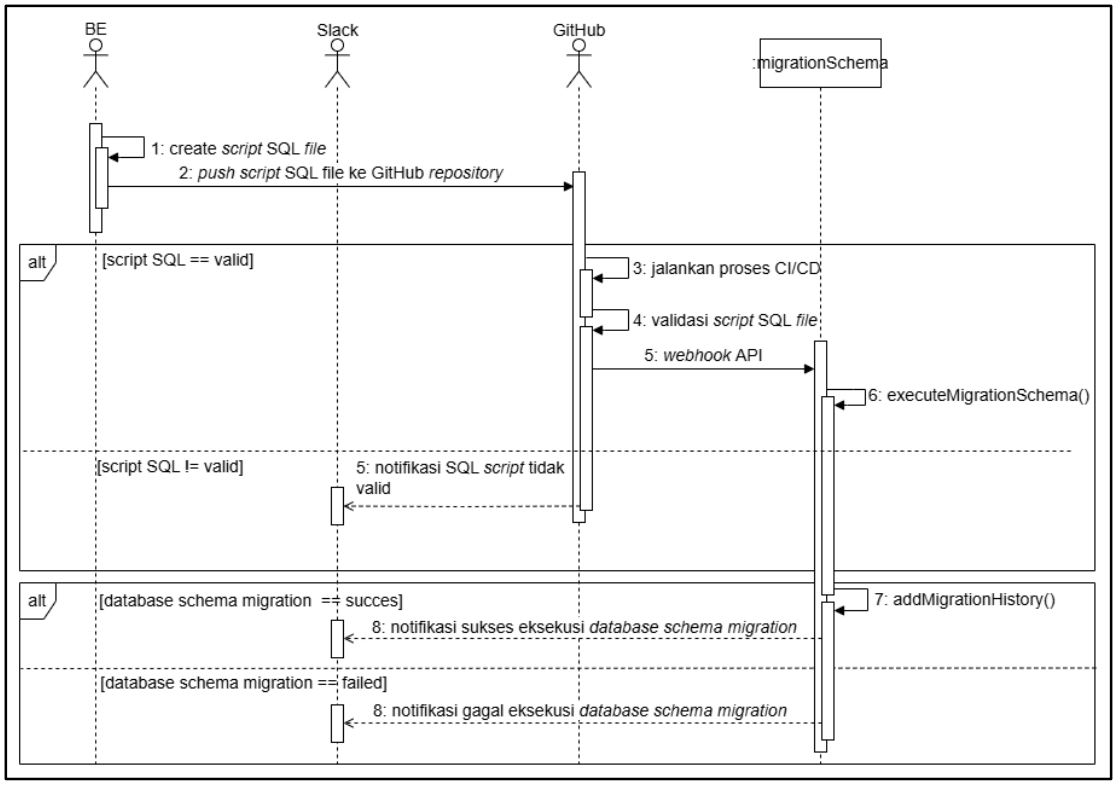
Gambar 4.3 *GitHub repository integration sequence diagram*

3. *Database schema migration configuration*



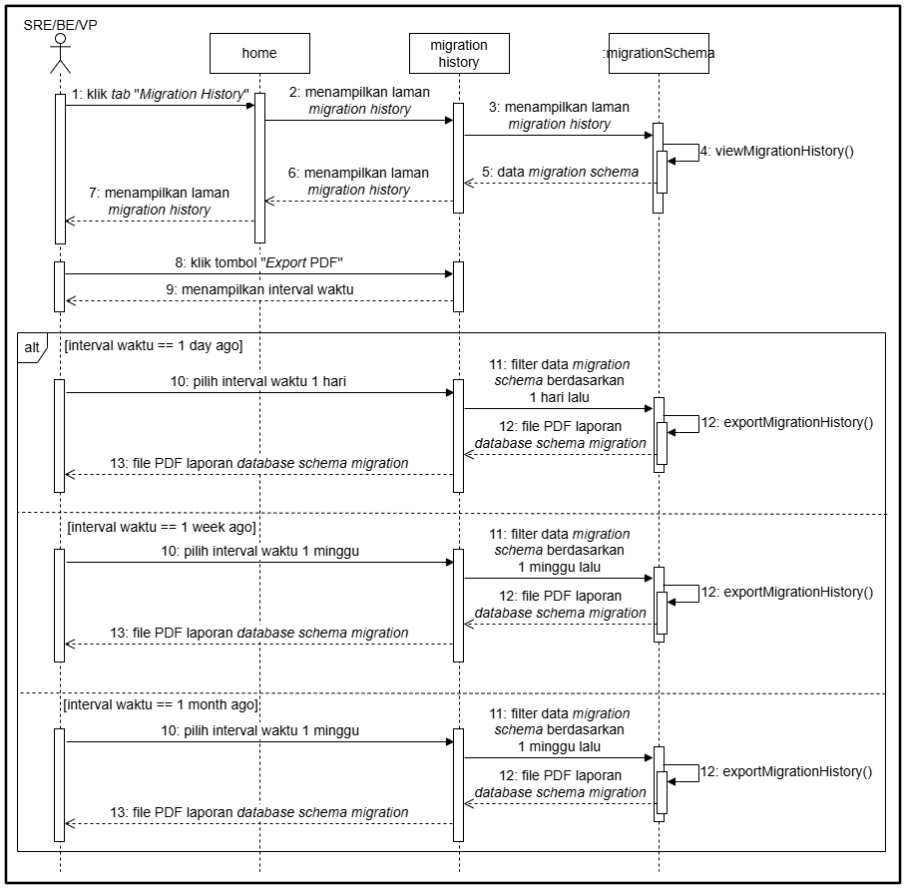
Gambar 4.4 *Database Schema Migration Configuration Sequence Diagram*

4. *Database schema migration pipeline*



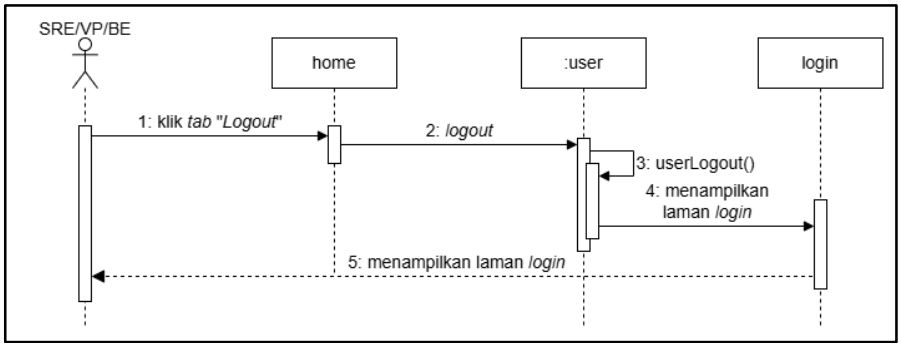
Gambar 4.5 *Database Schema Migration Pipeline Sequence Diagram*

5. *Database schema migration report*



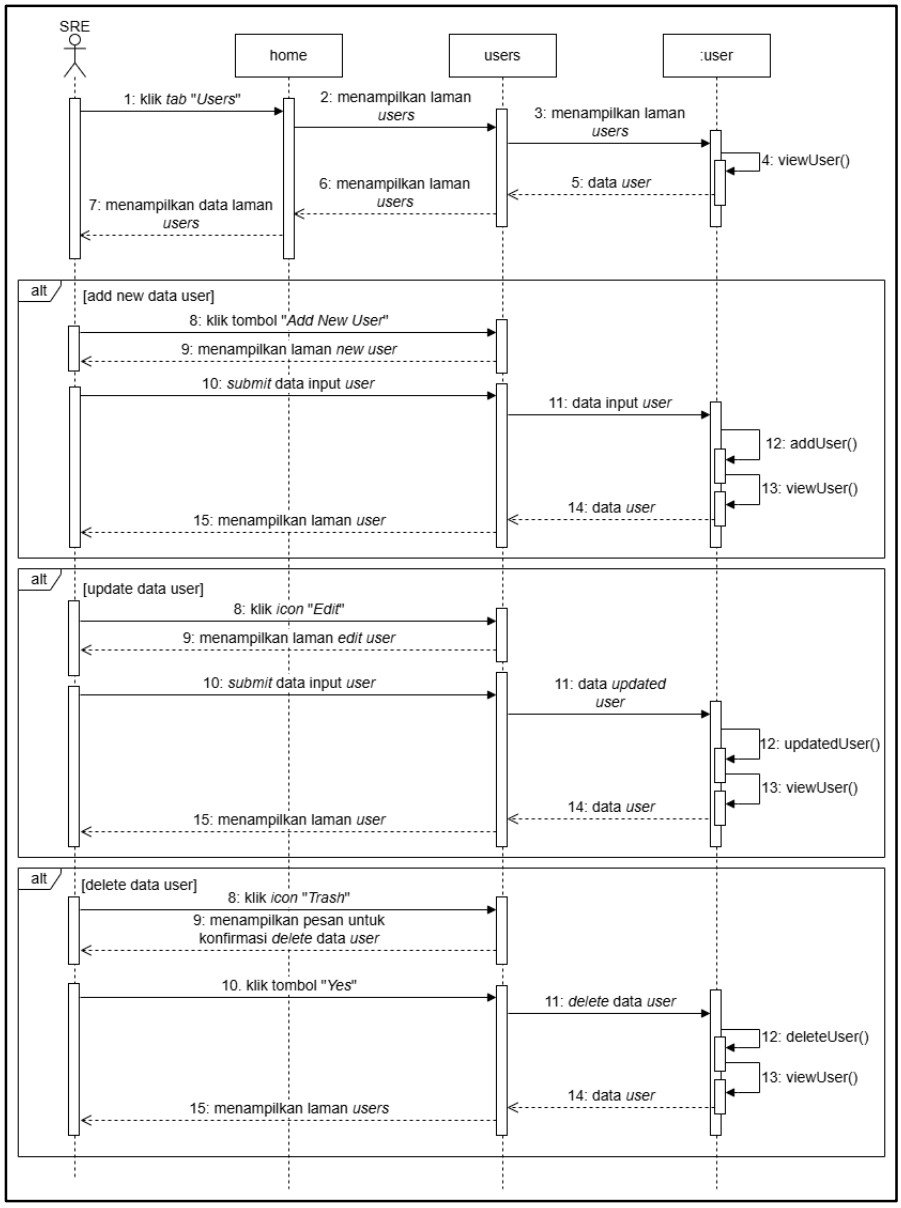
Gambar 4.6 *Database Schema Migration Report Sequence Diagram*

6. *Logout*



Gambar 4.7 *Logout Sequence Diagram*

7. *Manage user accounts*



Gambar 4.8 *Manage User Account Sequence Diagram*

4.1.3. Sistem pengkodean

Perancangan *database* pada sistem aplikasi, diperlukannya pembuatan kode pada setiap *class* atau *tabel* agar data yang tersimpan pada *database* terstruktur, mudah dikelola dan dapat diakses secara efisien. Kode yang dibentuk pada aplikasi ini disesuaikan dengan *class* yang sudah ditetapkan yang merujuk data informasi yang relevan dengan *class*. Berikut sistem pengkodean yang telah ditetapkan:

1. id_user.

   Kode ini merepresentasikan identitas unik untuk *user* yang terdapat pada *class* user".

   Contoh : 20250422-01

   2025 : Tahun *user* dibuat

   04 : Bulan *user* dibuat

   22 : Tanggal *user* dibuat

   01 : Nomor urut

2. id_repo

   Kode ini merepresentasikan identitas unik untuk *repository* yang terdapat pada *class* "repoIntegration"

   Contoh : RP001

   RP : Singkatan dari *repository*

   001 : Nomor urut

3. id_config

   Kode ini merepresentasikan identitas unik untuk *migration config* yang terdapat pada *class* "migrationConfig"

   Contoh: CFG001

   CFG : Singkatan dari *config*

   001 : Nomor urut

4. id_migration

   Kode ini merepresentasikan identitas unik untuk *migration schema* yang terdapat pada *class* “migrationSchema”

   Contoh : RK001

   RK : Inisial nama perusahaan

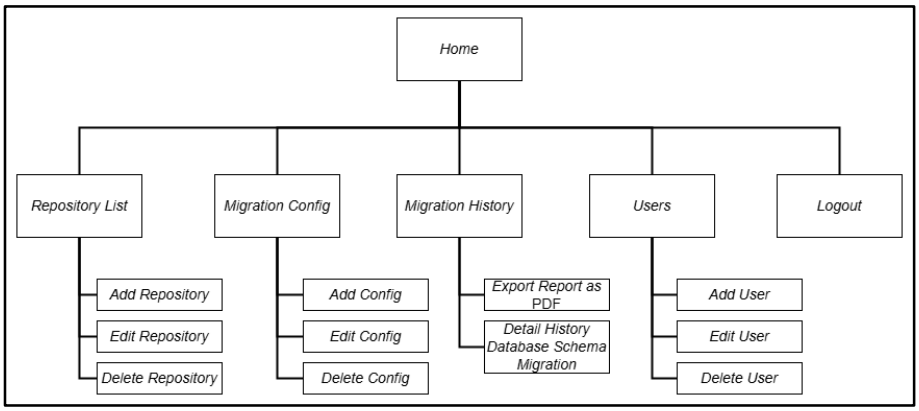
   001 : Nomor urut

4.2. Perancangan Antar Muka

Sistem aplikasi tentunya berinteraksi langsung dengan *user*, oleh karena itu sistem aplikasi memerlukan tampilan dan interaksi yang memudahkan *user* atau pengguna sistem aplikasi dalam mengoperasikannya. Tahapan yang diperlukan yaitu perancangan antar muka, dan hal itu dianggap penting dapat berdampak positif pada pengalaman dan juga produktivitas dari *user* atau pengguna aplikasi.

4.2.1. Struktur menu

Perancangan antar muka melibatkan adanya struktur menu, yang bertujuan untuk pemetaan pilihan menu yang tersedia pada sistem aplikasi. Struktur menu dapat diartikan sebagai kerangka tampilan sistem aplikasi, seperti layaknya navigasi terhadap perancangan sistem aplikasi. Struktur menu yang dirancang ini, terdapat beberapa menu yang berkaitan dengan *case* pada *system use case* yang berjumlahkan ada lima sub-menu pada tampilan utama atau halaman *home*. Penggambaran pada struktur menu tersebut dapat dilihat pada Gambar 4.9.



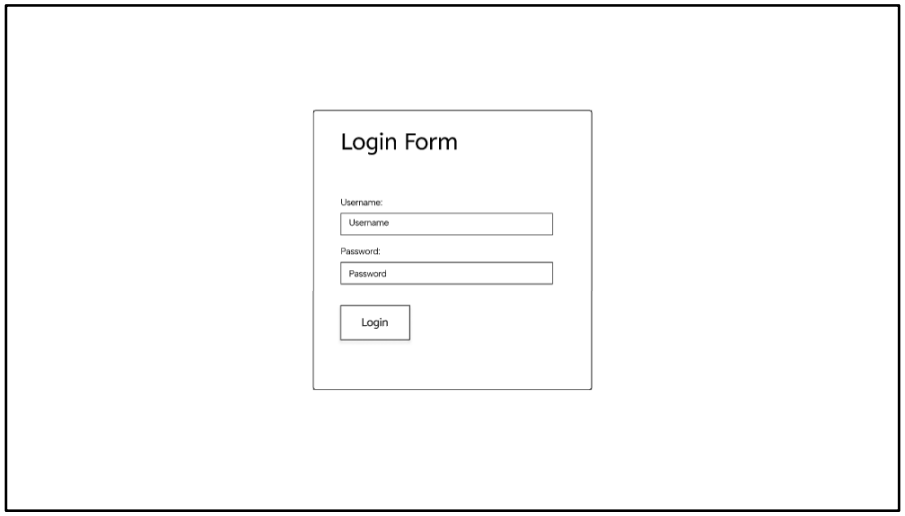
Gambar 4.9 Struktur Menu

4.2.2. Tata letak layar

Tata letak layak merupakan penggambaran yang asli daripada rancangan struktur menu pada Gambar 4.9. Tata letak layar ini dirancang menggunakan *platform* Figma dalam membantu merancang tampilan layar. Berikut keseluruhan tata letak layar pada sistem aplikasi:

1. *Login*

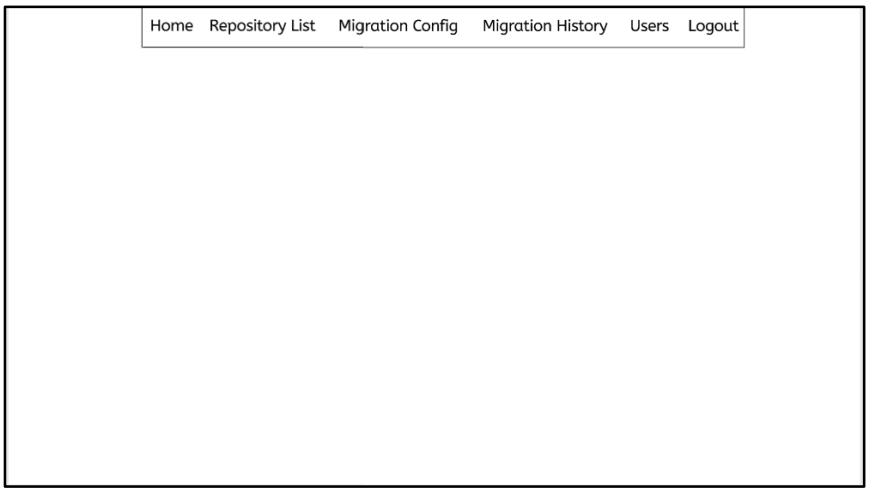
   Tampilan merupakan halaman untuk dapat memasuki web aplikasi dengan menginputkan *username* dan *password*.



Gambar 4.10 Tata Letak *Login*

2. *Home*

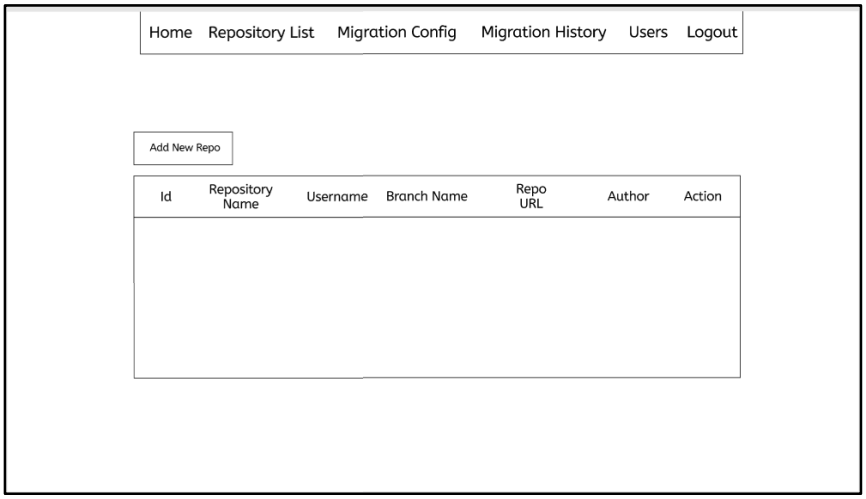
Tampilan merupakan halaman awal setelah berhasil *login*. Pada halaman ini terdapat menu pada bagian atas yang terdiri dari "*Repository List*", "*Migration Config*", "*Migration History*", "*Users*", dan "*Logout*".



Gambar 4.11 Tata Letak *Home*

3. *Repository list*

Tampilan merupakan halaman yang menampilkan data dari *repository* yang disajikan dalam bentuk tabel. Pada halaman ini dapat melakukan menambah, memperbaharui dan menghapus data *repository integration*.



Gambar 4.12 Tata Letak *Repository List*

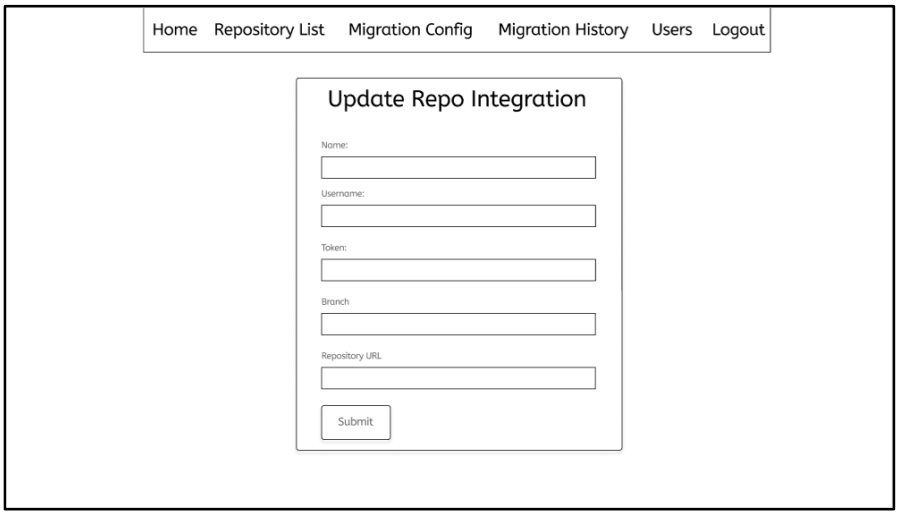
4. *Add data repository list*

Tampilan merupakan halaman untuk menambahkan data *repository integration* dengan menginputkan data nama, *username*, token, *branch* dan *repository* URL yang ingin ditambahkan.

Gambar 4.13 Tata Letak *Add Data Repository List*

5. *Edit data repository*

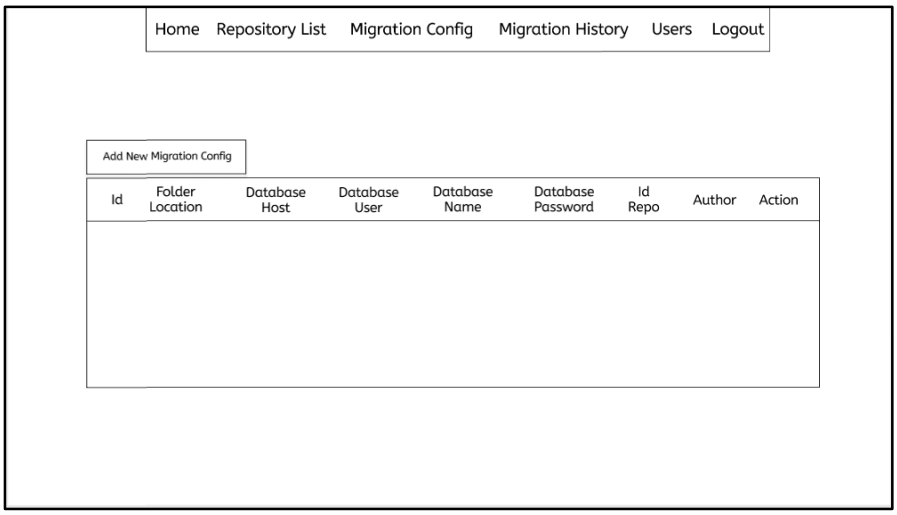
Tampilan merupakan halaman untuk memperbaharui data *repository integration* yang sebelumnya sudah ditambahkan.



Gambar 4.14 Tata Letak *Edit Data Repository*

6. *Migration config*

Tampilan merupakan halaman yang menampilkan data dari *migration config* yang disajikan dalam bentuk tabel. Pada halaman ini dapat melakukan menambah, memperbaharui dan menghapus data *migration config*.



Gambar 4.15 Tata Letak *Migration Config*

7. *Add data migration config*

Tampilan ini merupakan halaman untuk menambahkan data *migration config* dengan menginputkan data *folder location*, *database host*, *database username*, *database name*, *database password* dan *repository* yang ingin ditambahkan.

Gambar 4.16 Tata Letak *Add Data Migration Config*

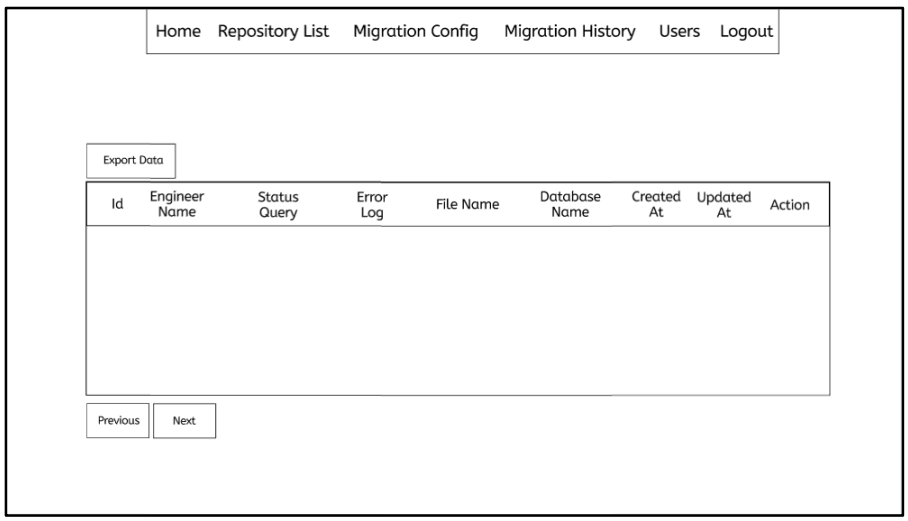
8. *Edit data migration config*

Tampilan merupakan halaman untuk memperbaharui data *migration config* yang sebelumnya sudah ditambahkan.

Gambar 4.17 Tata Letak *Edit Data Migration Config*

9. *Migration history*

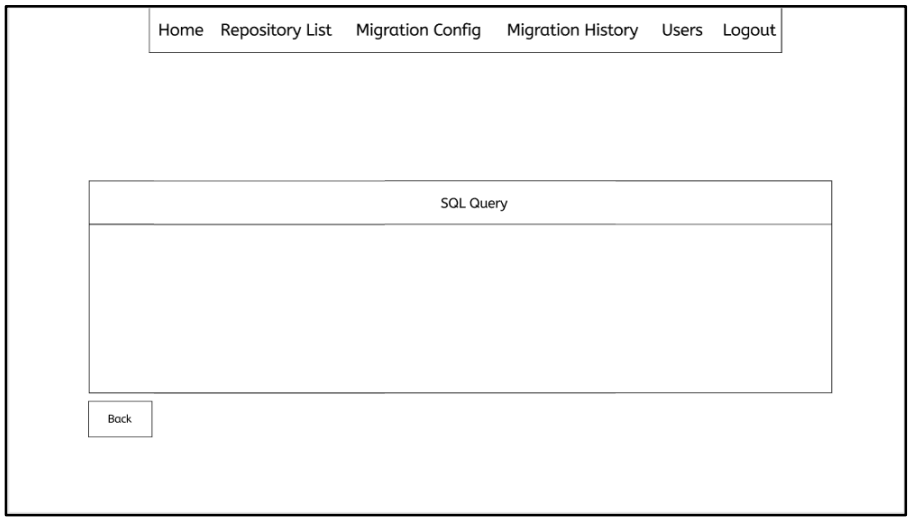
Tampilan merupakan halaman yang menampilkan data dari historis dalam bentuk tabel. Pada tampilan ini juga dapat melihat detail dari SQL *query* yang telah dilakukan dan ekspor data historis ini dalam bentuk PDF.



Gambar 4.18 Tata Letak *Migration History*

10. *Detail* SQL *query*

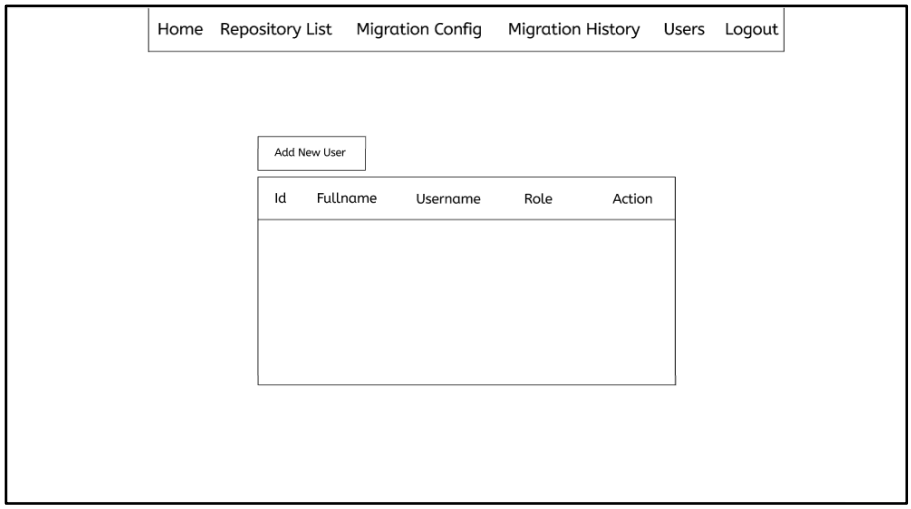
Tampilan ini merupakan halaman yang menampilkan data detail dari SQL *query* yang sudah dijalankan. Adanya halaman ini biasanya SQL *query* yang dijalankan terlalu panjang sehingga terpotong pada tampilan *migration history*.



Gambar 4.19 Tata Letak *Detail SQL Query*

11. *Users*

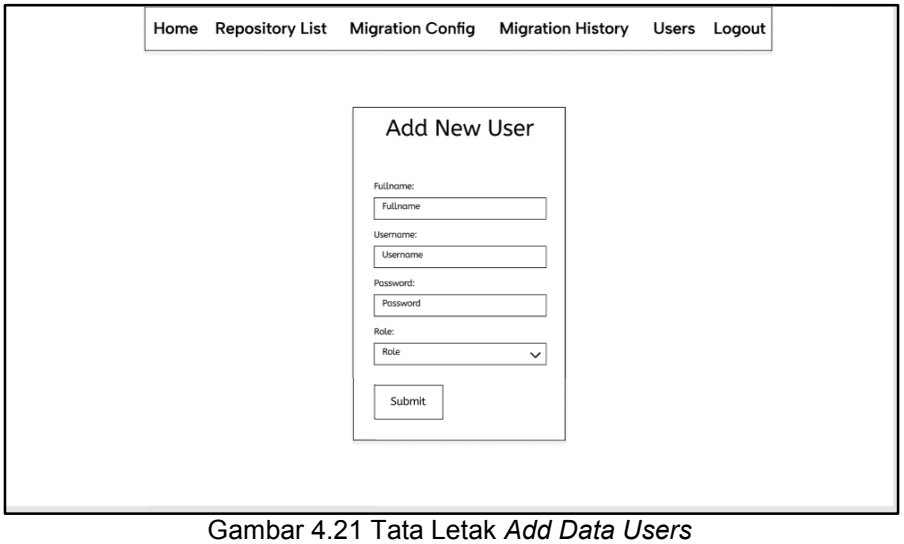
Tampilan merupakan halaman yang menampilkan data *user* dalam bentuk tabel. Pada halaman ini dapat melakukan menambah, memperbaharui dan menghapus data *user*.



Gambar 4.20 Tata Letak *Users*

12. *Add data users*

    Tampilan ini merupakan halaman untuk menambahkan data *user* dengan menginputkan data *fullname*, *username*, *password*, *role* yang ingin ditambahkan.



Gambar 4.21 Tata Letak *Add Data Users*

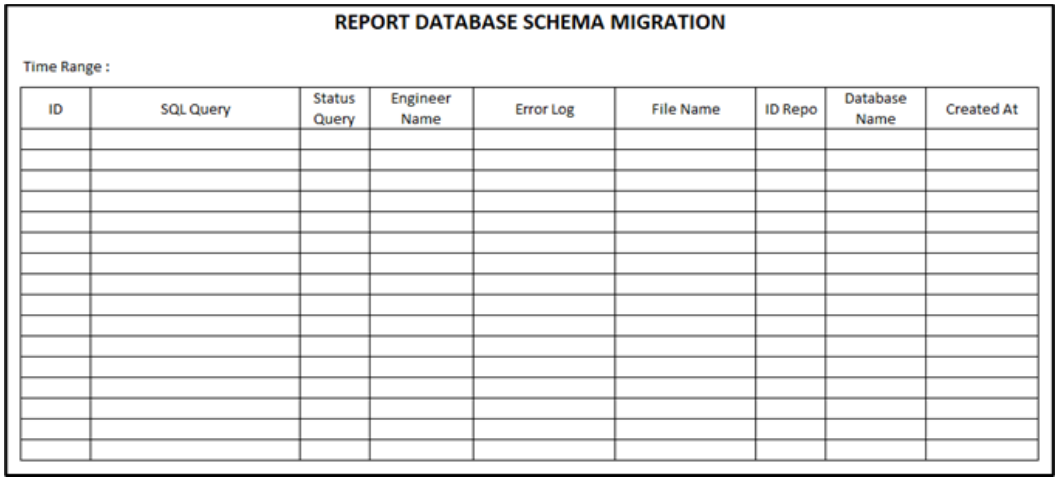
13. *Edit data users*

    Tampilan merupakan halaman untuk memperbaharui data *user* yang sebelumnya sudah ditambahkan.

Gambar 4.22 Tata Letak E*dit Data Users*

4.3. Perancangan Dokumen

Sistem aplikasi mempunyai rekapan data aktivitas yang berlangsung, biasanya rekapan data tersebut sangat berguna dan dijadikan suatu laporan. Rancangan dokumen merupakan gambaran atau pola yang menggambarkan laporan, dan pada aplikasi ini telah merancang dokumen untuk pelaporan tersebut. Rancangan dokumen pada aplikasi ini hanya tersedia untuk data historis *database schema migration*. Tampilan rancangan dokumen dapat dilihat pada Gambar 4.23.



REPORT DATABASE SCHEMA MIGRATION

Time Range :

ID	SQL Query	Status Query	Engineer Name	Error Log	File Name	ID Repo	Database Name	Created At

Gambar 4.23 Perancangan Dokumen *Report Database Schema Migration*

4.4. Implementasi Sistem

Pembahasan sebelumnya mencakup perancangan berorientasi objek, perancangan antarmuka dan perancangan dokumen. Pada bagian ini akan mengenai pembahasan yang mencakup pemilihan perangkat lunak, *component diagram*, persiapan penerapan sistem, *deployment diagram*, perkiraan kebutuhan biaya dan pengujian sistem. Topik-topik itu tidak berkaitan dengan fungsi ataupun alur kerja dari sistem aplikasi, namun lebih condong membahas bagaimana sistem aplikasi diterapkan.

4.4.1. Pemilihan perangkat lunak

Menjalankan sistem aplikasi, perlu *software* (perangkat lunak) yang mendukung berjalannya sistem aplikasi. *Software* yang digunakan pada sistem aplikasi dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Perangkat Lunak

No.	Software	Versi
1	OS Ubuntu Server	20.04
2	MySQL	8.0
3	Redis	1.89.0
4	GitHub	-
5	Slack	4.38.115
6	Docker	20.10.21
7	Docker compose	2.13.0
8	Caddy	2.6.4
9	NPM	9.5.1

Python dan Node JS yaitu bahasa pemrograman yang melandasi sistem aplikasi ini. Python digunakan yaitu versi 3.10.12 sebagai *backend service* yang melakukan proses di belakang layar dan *Node* JS versi 18.16.0 digunakan sebagai *frontend service* yang menampilkan data yang telah diproses oleh *backend service*. Dari kedua bahasa pemrograman tersebut, menggunakan *framework* dari setiap bahasa pemrograman tersebut agar membantu dalam proses membangun aplikasi, untuk penjelasan lebih jelas *framework* yang digunakan dapat dilihat pada tabel 4.2.

Tabel 4.2 Daftar Bahasa Pemrograman

No.	**Bahasa Pemrograman**	***Framework***	**Versi**
1	Python	Django	3.14.0
		DjangoRestFramework	5.0.2
2	Node JS	Vue	3.4.21

4.4.2. Persiapan penerapan sistem

Sistem aplikasi yang telah dibangun, tidak semerta-merta dapat digunakan oleh pengguna. Namun ada aspek yang perlu dilakukan sebelum sistem aplikasi dapat digunakan semestinya. Aspek tersebut biasanya diperlukan dikarenakan sistem aplikasi mempunyai ketergantungan dengan komponen atau sistem lainnya. Hal ini cukup relevan dengan diagram komponen yang telah dijelaskan pada poin sebelumnya. Persiapan yang perlu dilakukan, antara lain yaitu:

1. *Virtual machine* dan OS (*Operating System*).

   Persiapan pertama yaitu menyiapkan *virtual machine* yang menjadi wadah sistem aplikasi dijalankan. *Virtual machine* yang digunakan yaitu menggunakan produk *compute engine* dari *Google Cloud Platform* (GCP). Tipe e2-medium merupakan spesifikasi yang digunakan oleh *compute engine* dengan 2vCPU core dan 4 GB RAM. Kemudian OS (*Operating System*) yang berjalan pada *virtual machine* tersebut yaitu Ubuntu Server 20.04.

2. Docker dan docker-compose.

   Setelah *virtual machine* selesai disiapkan, selanjutnya *install* docker dan docker-compose untuk sebagai *engine* yang menjalankan keseluruhan sistem aplikasi. Namun sebelum itu, perlu membuat *file* dari dalam bentuk format docker-compose untuk dapat menjalankan *multi-container* pada waktu yang bersamaan. Hal yang didefinisikan pada docker-compose adalah *service* dari sistem aplikasi, antara lain yaitu Redis, MySQL Server, Nginx, *frontend service*, dan *backend service*.

3. DNS

   Konfigurasi domain untuk sistem aplikasi, dengan menggunakan layanan DNS Cloudflare.

4. Integrasi notifikasi

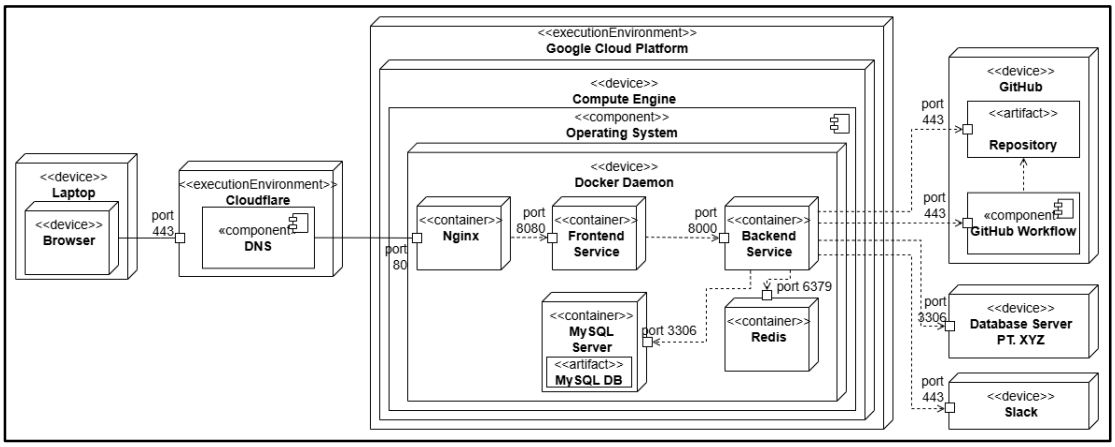
   Persiapan selanjutnya, integrasi sistem aplikasi dengan platform kolaborasi Slack. Dengan membuat aplikasi pada Slack, aplikasi pada Slack akan memberikan notifikasi pesan dari hasil operasi *database schema migration* berdasarkan hasil aktifitas *database schema migration*.

5. GitHub *Workflow*

   Membuat *script* pada GitHub Workflow pada suatu *repository* untuk membuat *pipeline* operasi validasi *script* SQL *file* dan *trigger webhook* operasi *database schema migration*.

4.4.3. *Deployment diagram*

Deployment diagram merupakan representasi dari aspek fisik yang berkaitan dengan sistem aplikasi. *Deployment diagram* pada sistem aplikasi di PT. XYZ melibatkan seperti Cloudflare, GitHub, *Database Server* pada PT. XYZ, dan Slack. Sistem aplikasi yang dijalankan pada *Google Compute Engine*. Untuk lebih lengkapnya dapat Gambar 4.24 .



Gambar 4.24 *Deployment Diagram*

4.4.4. Perkiraan kebutuhan biaya

Sistem aplikasi tidak hanya membahas mengenai alur kerja, ataupun fitur yang dimiliki, namun perlu juga membahas seperti biaya yang dibutuhkan saat menjalankan sistem aplikasi tersebut, seperti *server*, *domain* dan lain-lain. Pada sistem aplikasi yang telah dibuat, telah dirincikan lebih jelas pada Tabel 4.3 , mengenai biaya yang diperlukan. Biaya ini, merupakan pengeluaran yang dilakukan setiap bulan atas penggunaan layanan dari pihak ketiga atau semacamnya.

Tabel 4.3 Perkiraan Kebutuhan Biaya

No.	Keterangan	Biaya
1	*Compute Engine from Google Cloud Platform* (Termasuk dengan IP Publik dan *storage*)	Rp. 695.902
2.	DNS + SSL *from* Cloudflare	Free
	Total	**Rp. 659.902**

4.4.5. Pengujian sistem

Aplikasi yang telah dirancang dan sudah dibangun perlu melewati tahapan pengujian, agar aplikasi yang telah dibangun sesuai dengan perancangan di awal. Pengujian ini dilakukan secara manual dengan memastikan bahwa aplikasi berfungsi dengan spesifikasi yang telah ditentukan. Hasil pengujian tersebut dapat dilihat pada Tabel 4.4.

Tabel 4.4 Pengujian Sistem

No.	Item Uji	Kasus Uji	Hasil yang diharapkan	Hasil Uji
1.	Login	Input *username* dan *password* yang valid	Berhasil *login* dan menampilkan halaman *home*	Sesuai
		Salah satu dari *username* dan *password* tidak valid	Menampilkan pesan “Gagal *login*, silahkan coba lagi”	Sesuai
		Salah satu dari *username* atau *password* tidak diisi	Menampilkan pesan “Gagal *login*, silahkan coba lagi”	Sesuai
2	*Repository Integration*	Input *name, username, token, branch dan repository* URL yang valid	Berhasil menambahkan *repository integration*	Sesuai
		Salah satu dari *name, username, token, branch dan repository* URL tidak diisi	Menampilkan pesan “Gagal menambahkan *repository*	Sesuai

No.	Item Uji	Kasus Uji	Hasil yang diharapkan	Hasil Uji
			integration, silahkan coba lagi”	
		Token tidak sesuai	Menampilkan pesan “Token GitHub tidak valid”	Sesuai
		Nama *branch* tidak sesuai	Menampilkan pesan “*Branch* tidak tersedia”	Sesuai
		Repository URL tidak sesuai	Menampilkan pesan “*Repository* tidak tersedia”	Sesuai
		Username tidak sesuai	Menampilkan pesan “GitHub *username* tidak tersedia”	Sesuai
		Nama *repository integration* duplikat	Menampilkan pesan “Nama *repository integration* sudah digunakan”	Sesuai
3	*Migration Config*	Input *folder location*, *database host*, *database username*, *database name*, *database password* dan repo yang sesuai	Menampilkan pesan “Berhasil menambahkan *migration config*”	Sesuai
		Salah satu dari *folder location*, *database host*, *database username*, *database name*, *database password* dan repo tidak diisi	Menampilkan pesan “Gagal menambahkan *migration config,* silahkan coba lagi”	Sesuai
		Folder location tidak sesuai	Menampilkan pesan “*Folder*	Sesuai

No.	Item Uji	Kasus Uji	Hasil yang diharapkan	Hasil Uji
			location tidak tersedia pada *repository*”	
		Database host tidak dapat dijangkau oleh aplikasi atau tidak valid	Menampilkan pesan “Gagal koneksi ke *database*”	Sesuai
		Database username tidak sesuai atau tidak ada	Menampilkan pesan “Gagal koneksi ke *database*”	Sesuai
		Database name tidak sesuai atau tidak tersedia	Menampilkan pesan “*Database name* tidak tersedia”	Sesuai
		Database password tidak sesuai	Menampilkan pesan “Gagal koneksi ke *database*”	Sesuai
4	*Migration*	*File* SQL yang ditambahkan ke *repository* tidak valid	Mengirimkan pesan “SQL script tidak valid” ke *channel* Slack	Sesuai
		Gagal melakukan *database schema migration*	Mengirimkan pesan berupa detail terkait *database schema migration* ke *channel* Slack	Sesuai
		Berhasil melakukan *database schema migration*	Mengirimkan pesan berupa detail terkait *database schema migration* ke *channel* Slack	Sesuai
		Ekspor historis *database schema migration* berdasarkan interval yang dibutuhkan	Mengunduh *file report* historis berupa PDF	
5	*Users*	Input *fullname*, *username*, *password* dan *role* yang valid	Berhasil menambahkan *user*	Sesuai

No.	Item Uji	Kasus Uji	Hasil yang diharapkan	Hasil Uji
		Salah satu *fullname*, *username*, *password* tidak diisi	Menampilkan pesan "Gagal menambahkan *user,* silahkan coba lagi"	Sesuai
		Tidak memilih satu satu *role* antara *admin* dan *viewer*.	Menampilkan pesan "Belum memilih *role*"	Sesuai

Dari hasil uji yang telah disajikan pada Tabel 4.4,dilanjutkan membandingkan sistem aplikasi yang telah dibuat dengan aplikasi *database schema migration* yang tersedia seperti Flyway dan Liquibase.

Tabel 4.5 Perbandingan *Tools Database Schema Migration*

	Flyway	Liquibase	Sistem Aplikasi PT. XYZ
Engine	Java + JVM	Java + JVM	Python
Edition	Community, Teams, and Enterprise	Open Source and Pro	-
***Database* GitOps**	Tidak tersedia (Kalau ingin mengimplementasikan perlu ada cara tambahan diluar dari fitur yang disediakan)	Tidak tersedia (Kalau ingin mengimplementasikan perlu ada cara tambahan diluar dari fitur yang disediakan)	Tersedia
Pencatatan Historis	Tersedia (Hanya terdapat pada *Enterprise Edition* atau diatasnya)	Tidak tersedia	Tersedia
Developer Interface	CLI	CLI	GUI
***Support Database* MySQL**	Ya	Ya	Ya
***Support* MySQL v5.7**	Ya (Tetapi ada limitasi pengujian pada *Community Edition*)	Ya (Akan tetapi sudah tidak berlaku lagi pada MySQL Server	Ya

	Flyway	Liquibase	Sistem Aplikasi PT. XYZ
		dan AWS RDS)	
Kemudahan Penggunaan	Ya (Mudah digunakan bagi pemula)	Tidak (Banyaknya fitur yang disediakan sehingga tidak cocok untuk pemula)	Ya
Integrasi notifikasi Slack	Tidak	Tidak	Ya
SQL *Check Validation*	Tidak	Tidak	Ya
Terdapat nama pengeksekusi *database schema migration*	Tidak	Ya	Ya

Sistem aplikasi *database schema migration* yang dibentuk jika dibandingkan dengan aplikasi lain secara keandalan dan fitur yang disediakan akan kalah. Faktor yang dipengaruhi dikarenakan kedua aplikasi dari Flyway dan Liquibase dibentuk sudah sejak tahun awal tahun 2000. Dapat dikatakan kedua aplikasi tersebut sudah mempunyai usia yang lebih matang jika dibandingkan dengan sistem aplikasi dari PT. XYZ. Selain daripada itu, kedua aplikasi tersebut tidak hanya kompatibel dengan *database server* MySQL, namun sudah banyak *database server* yang kompatibel dengan kedua aplikasi tersebut.

Namun kebutuhan utama dari PT. XYZ untuk menjalankan *database schema migration* secara otomatis, integrasi dengan platform untuk mengirimkan notifikasi, menerapkan GitOps, dan mempunyai data historis dari aktifitas *database schema migration*. Hal itu semuanya dapat dipenuhi oleh sistem aplikasi yang telah dibuat pada PT. XYZ. Adapun dari kedua aplikasi yang sudah ada, hanya dapat memenuhi beberapa kebutuhan dari PT. XYZ dan itu pun perlu dengan tambahan biaya.

Membandingkan ketiga aplikasi tersebut dalam konteks fitur yang dibutuhkan, sistem aplikasi yang telah dibuat pada PT. XYZ dapat disesuaikan dengan kebutuhan yang diperlukan suatu saat nanti atau lebih kompleks dari kebutuhan saat ini, sedangkan dengan kedua sistem aplikasi yang sudah ada tidak dapat dikembangkan secara *internal* dan perlu menyesuaikan dengan fitur yang kedua aplikasi tersebut sediakan.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Analisa dan perancangan sistem aplikasi *database schema migration* pada PT. XYZ berlandaskan adanya proses kerja dalam melakukan *database schema migration* yang dilakukan secara manual sehingga tidak adanya pencatatan historis terkait *database schema migration*, dan SQL DDL/DML untuk menjalankan *database schema migration* tidak dicantumkan pada suatu GitHub *repository*. Oleh karena itu direncanakan dan dibentuknya sistem aplikasi yang dijalankan secara otomatis untuk menjawab persoalan tersebut.

Kesimpulan atas hasil analisis ini antara lain yaitu perancangan sistem aplikasi *database schema migration* yang telah dibangun dapat mengotomatiskan proses *database schema migration* melalui integrasi dengan alur CI/CD, serta menyediakan fitur pelaporan historis yang detail. Selain daripada itu, sistem aplikasi ini menerapkan GitOps dalam konteks *database schema migration* dapat meningkatkan efisiensi, reliabilitas, dan transparansi dalam pengelolaan *database*. Kemudian sistem aplikasi ini juga membuat laporan historis dari proses *database schema migration* yang telah terjadi dan dilaporkan secara berkala kepada *Vice President of Technology*.

5.2. Saran

Berdasarkan hasil analisa dan perancangan sistem aplikasi yang telah dibentuk dalam menjalankan *database schema migration*, adanya saran yang ditujukan mengenai pengembangan sistem aplikasi agar sistem aplikasi ini sangat berdampak dan berguna di masa yang akan datang. Hal-hal tersebut antara lain yaitu:

1. Tahapan validasi SQL *query* saat ini masih menggunakan pihak ketiga dari penyedia layanan GitHub *marketplace*, dan disarankan hal tersebut termasuk dalam fitur yang disediakan oleh sistem aplikasi.

2. Sistem aplikasi hanya *support* 1 *database server* yaitu MySQL, disarankan dapat dikembangkan menjadi lebih banyak lagi, seperti contohnya dapat kompatibel dengan PostgreSQL, MariaDB dan MongoDB. Hal tersebut dikarenakan *database server* tersebut merupakan bagian dari *database server* yang populer saat ini.
3. Menambahkan fitur *rollback* setelah melakukan *database schema migration*, untuk membantu pengguna sistem aplikasi untuk mengembalikan skema sebelumnya.
4. Sistem aplikasi saat ini belum memenuhi aspek dalam mencegah adanya serangan *cybersecurity,* terlebih lagi aplikasi ini berhubungan dengan *database* yang menyimpan data yang cukup krusial.

DAFTAR PUSTAKA

Abiola, O. B., & Olufemi, O. G. (2023). An Enhanced CICD Pipeline: A DevSecOps Approach. *International Journal of Computer Applications*, 6.

Alami, A., Krancher, O., & Paasivaara, M. (2022). The journey to technical excellence in agile software development. *Information and Software Technology*, 14.

Amornchewin, R. (2018). The Development of SQL Language Skills in Data Definition and Data Manipulation Languages Using Exercises with Quizizz for Students' Learning Engagement. *Indonesian Journal of Informatics Education*, 85-89.

Amundsen, M. (2020). *Design and Build Great Web APIs.* New York: O'Reilly Media.

Anardi, S. (2019). *Perancangan Sistem Berorientasi Objek Dengan Pemodelan UML (Unified Modeling Language) Tools.* Madiun: UNIPMA Press.

Arachchi, I. P. (2018). Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management. 6.

Astari. (2023, August 25). *What Is HTML? Hypertext Markup Language Basics Explained*. Dipetik March 2024, 2024, dari Hostinger: https://www.hostinger.com/tutorials/what-is-html

AWS. (2024, March 20). *Apa itu API (Antarmuka Pemrograman Aplikasi)?* Dipetik March 23, 2024, dari AWS: https://aws.amazon.com/id/what-is/api

Banaszek, W. (2023, November 14). *Orchestrating Celery Python Background Jobs*. Dipetik March 26, 2024, dari Toptal: https://www.toptal.com/python/orchestrating-celery-python-background-jobs

Barry, P. (2023). *Head First Python, 3rd Edition.* New York: O'Reilly Media.

Beaulieu, A. (2020). *Learning SQL.* New York: O'Reilly Media.

Botros, S., & Tinley, J. (2021). *High Performance MySQL: Proven Strategies for Operating at Scale 4th Edition.* New York: O'Reilly Media .

Carlson, J. (2013). *Redis in Action.* New York: O'Reilly Media.

Chandra, M., Suroso, A. I., & Hermadi, I. (2015). Evaluasi Cobit dan Perancangan IT Balanced Scorecard untuk Perbaikan Penerapan System Development. *Jurnal Manajemen Teknologi*, 15.

Coursera. (2023, November 30). *Database Schema: Definition, Types, and Benefits*. Dipetik March 10, 2024, dari Coursera: https://www.coursera.org/articles/database-schema

Da Silva, M. D., & Tavares, H. L. (2015). *Redis Essentials.* New York: O'Reilly Media.

Ekenstam, T., Suen, J., Matyushentsev, A., & Yuen, B. (2021). *GitOps and Kubernetes.* New York: O'Reilly Media.

Erkamim, M. (2023). *Buku Ajar Pengantar Sistem Informasi.* Ponorogo: PT Prime Identity Home.

Flanagan, D. (2020). *JavaScript: The Definitive Guide, 7th Edition.* New York: O'Reilly Media.

Fridayanthie, E. W., Haryanto, & Tsabitah, T. (2021). Penerapan Metode Prototype Pada Perancangan Sistem Informasi Penggajian Karyawan (Persis Gawan) Berbasis Web. *Jurnal Paradigma, Jurnal Komputer dan Informatika* , Jurnal Paradigma, Jurnal .

Gagliardi, V. (2021). *Decoupled Django: Understand and Build Decoupled Django Architectures for JavaScript Front-ends.* New York: O'Reilly Media.

Garaguso, P. D. (2023). *Vue.js 3 Design Patterns and Best Practices.* New York: O'Reilly Media.

GeeksForGeeks. (2023, November 28). *Database Schemas*. Dipetik March 10, 2024, dari GeeksForGeeks: https://www.geeksforgeeks.org/database-schemas/

Ghumatkar, R. S. (2023). Software Development Life Cycle (SDLC). 6.

Giri, G., I.G.D., D., & Wibawa, I. (2022). Rancang Skema Database dan Implementasi Database Migration Pada Aplikasi Peminjaman Ruangan. 10.

Grant, K. (2018). *CSS in Depth.* New York: O'Reilly Media.

Hartono, N., Utami, E., & Amborowati, A. (2016). Migrasi dan Optimalisasi Database Sistem Informasi Manajemen Universitas Cokroaminoto Palopo. 10.

Hayes, A. (2022, November 24). *HyperText Markup Language (HTML): What It Is, How It Works*. Dipetik March 26, 2024, dari Investopedia: https://www.investopedia.com/terms/h/html.as

IBM. (t.thn.). *What is a database schema?* Diambil kembali dari IBM: https://www.ibm.com/topics/database-schema

Kadir, A. (2014). *Pengenalan Sistem Informasi Edisi Revisi.* Yogyakarta: CV. ANDI OFFSET.

Kaur, G. K., Singla, S., & Khawas, S. (2023). Database Management System: A Study of Increasing Impact of NoSQL Databases. *International Conference on Advanced Computing & Communication Technologies (ICACCTech)*, 5.

Laszewski, T., & Nauduri, P. (2012). *Migration to The Cloud Oracle Client/Server Modernization.* Waltham: Elsevier Inc.

Loubser, N. (2021). *Software Engineering for Absolute Beginners.* London: Apress.

Mazrae, P. R., Mens, T., Golzadeh, M., & Decan, A. (2023). On the usage, co-usage and migration of CI/CD tools. *Empirical Software Engineering*, 45. Dipetik March 6, 2024, dari https://link.springer.com/content/pdf/10.1007/s10664-022-10285-5.pdf

Meyer, E., & Weyl, E. (2023). *CSS: The Definitive Guide, 5th Edition.* New York: O'Reilly Media.

Mulyani, S. (2016). *Metode Analisis dan Perancangan Sistem.* Bandung: Abdi Sistematika.

Nistrina, K., & Sahidah, L. (2022). UNIFIED MODELLING LANGUAGE (UML) UNTUK PERANCANGAN SISTEM INFORMASI PENERIMAAN SISWA BARU DI SMK MARGA INSAN KAMIL. *Jurnal Sistem Informasi*.

Oracle. (2021, October 27). *What Is a Database?* Dipetik March 12, 2024, dari OCI: https://www.oracle.com/id/database/what-is-database/

Pandey, R. K., Singh, K. Y., & Milan, R. (2017). Advantage of Database Management System in Cloud Infrastructure. *Journal of Computer Science and Engineering*, 8.

Petrov, A. (2019). *Database Internals.* Gravenstein Highway North: O'Reilly Media.

Ponuthorai, P. K., & Loeliger, J. (2022). *Version Control with Git, 3rd Edition.* New York: O'Reilly Media.

Quinten, J. (2024). *Building Real-World Web Applications with Vue.js 3.* New York: O'Reilly Media.

Rahman, A., Agrawal, A., Krishna, R., & Sobran, A. (2018). Characterizing The Influence of Continuous Integration. Empirical Results from 250+ Open Source and Proprietary Projects. 7.

Rais, M. (2019). Penerapan Konsep Object Oriented Programming Untuk Aplikasi Pembuat Surat. *Jurnal PROtek*.

RedHat. (2023, December 12). *RedHat*. Dipetik March 8, 2024, dari What is CI/CD?: https://www.redhat.com/en/topics/devops/what-is-ci-cd

Retnoningsih, E., Shadiq, J., & Oscar, D. (2017). Pembelajaran Pemrograman Berorientasi Objek (Object Oriented Programming) Berbasis Project Based Learning. *INFORMATICS FOR EDUCATORS AND PROFESSIONALS*, 95-104.

Richardson, L., Amundsen, M., & Ruby, S. (2013). *RestFul Web APIs.* New York: O'Reilly Media.

Rouse, M. (2024, January 12). *DBMS (Database Management System)*. Dipetik March 12, 2024, dari Technopedia: https://www.techopedia.com/definition/24361/database-management-systems-dbms

Shaw, B., Badhwar, S., Guest, C., & Chandra, B. (2023). *Web Development with Django - Second Edition.* New York: O'Reilly Media.

Skoulikari, A. (2023). *Learning Git.* New York: O'Reilly Media.

SolarWinds. (2023, June 16). *What is SQL Database?* Dipetik March 12, 2024, dari SolarWinds: https://www.solarwinds.com/resources/it-glossary/sql-database

Soufitri, F. (2023). *Konsep Sistem Informasi.* Padangsidimpuan : PT Inovasi Pratama Internasional.

Subiyantoro, E. (2013). *Pemrograman Berbasis Objek.* Malang: Kementerian Pendidikan & Kebudayaan.

Susanto, A., & Meiryani. (2019). Database Management System. *International Journal of Scientific* , 4.

Svekis, L. L., Putten, M. v., & Percival, R. (2021). *JavaScript from Beginner to Professional.* New York: O'Reilly Media.

Tsitoara, M. (2020). *Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, Teamwork for the New Developer.* New York: O'Reilly Media.

VanderPlas, J., & Harisson, M. (2018). *What is Python.* New York: O'Reilly Media.

Vinto, N., & Bueno, A. S. (2023). *GitOps Cookbook.* New York: O'Reilly Media.

Wahyudin, Y., & Rahayu, D. N. (2020). Analisis Metode Pengembangan Sistem Informasi Berbasis Website: A Literature Review. *Jurnal Publikasi Ilmiah Bidang Teknologi Informasi dan Komunikasi*, 15.

Wang, L., & Tan, K. C. (2006). *Modern Industrial Automation Software Design.* Wiley.

Wiesche, M. (2021). Interruptions in Agile Software Developments Teams. *Project Management Journal*, 13.

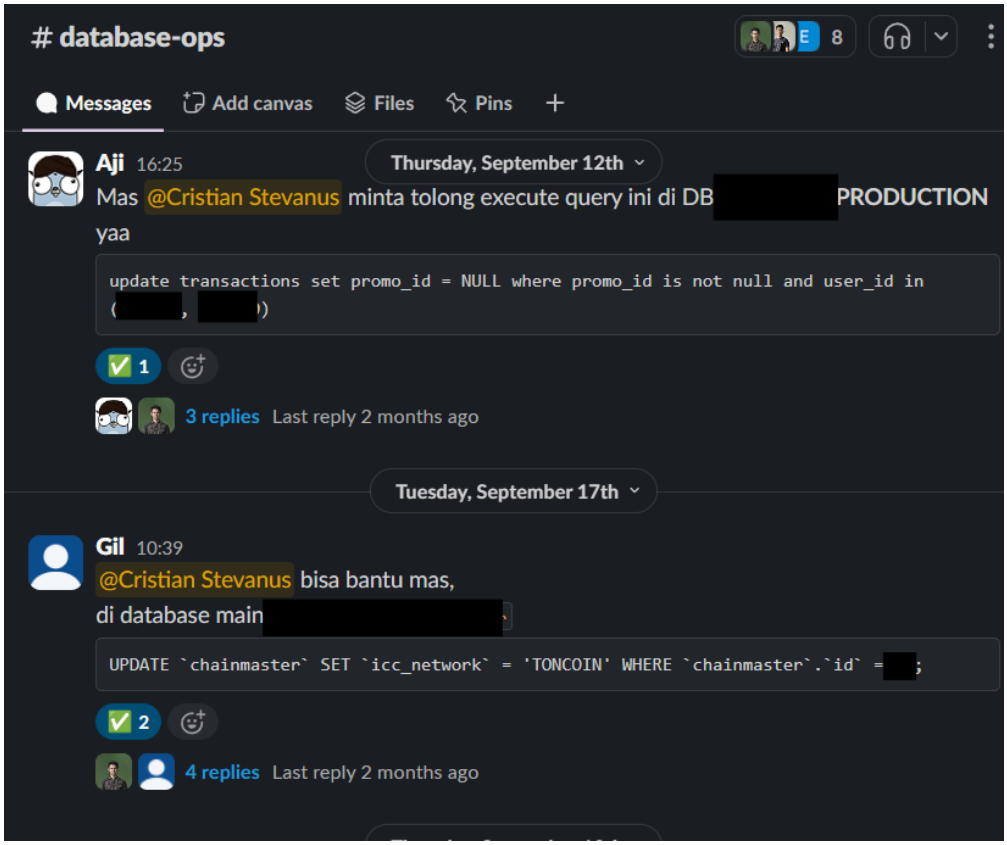
Wu, Y., Wang, N., Kropczynski, J., & Carroll, J. M. (2017). The appropriation of GitHub for curation. *PeerJ Computer Science*, 20.

Yanti, S. N., & Rihyanti, E. (2021). Penerapan Rest API untuk Sistem Informasi Film Secara Daring. *Jurnal Informatika Universitas Pamulang*, 195-201.

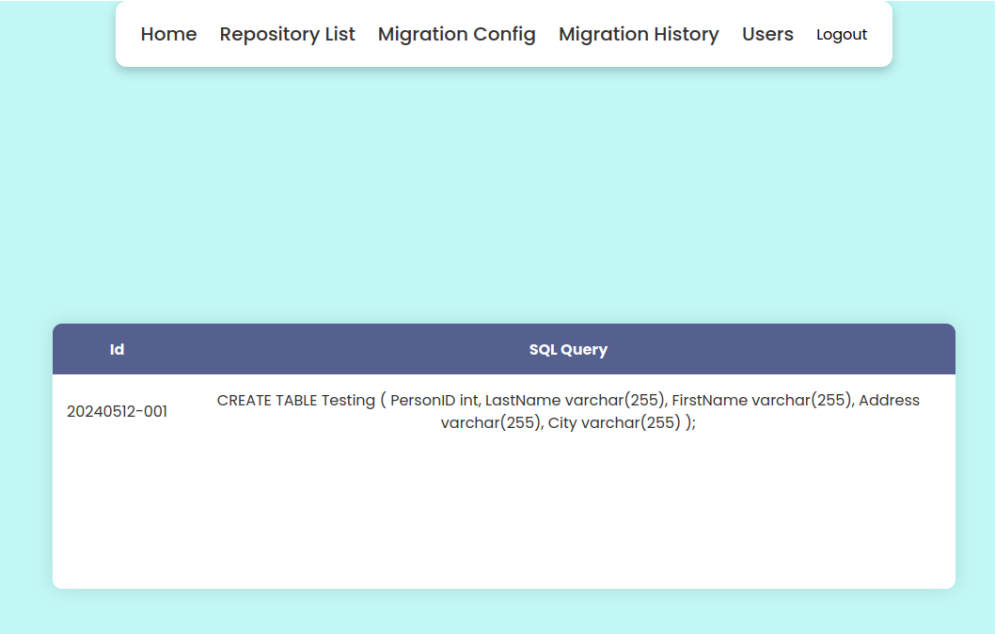
Primsa.Io. (2023, December 15). *What are database migrations?* (N. Burk, Editor) Dipetik March 23, 2025, dari Prima's Data Guide: https://www.prisma.io/dataguide/types/relational/what-are-database-migrations#:~:text=Migrations%20are%20helpful%20because%20they,schema%20changes%20to%20their%20environments.

DAFTAR LAMPIRAN

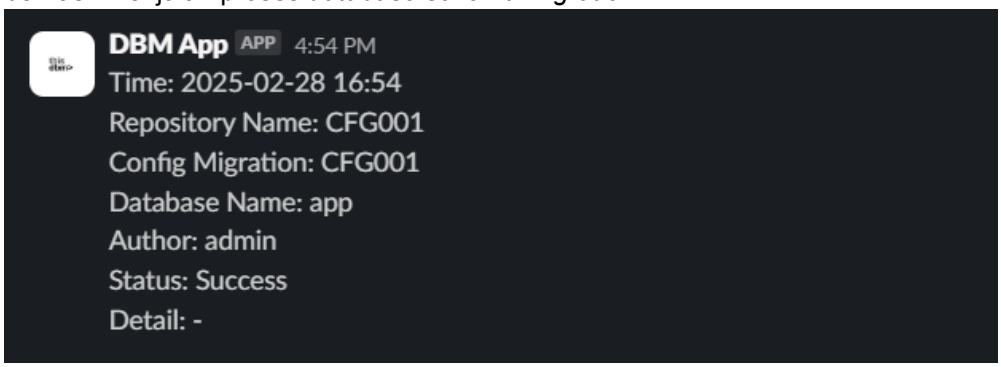
1. Screenshot *backend engineer* melakukan *request database schema migration* secara manual pada *channel* “database-ops” di *platform* Slack.



2. Tampilan aplikasi yang menunjukkan laman "*Migration History*".



3. Tampilan notifikasi bot aplikasi Slack ke *channel* #database-ops dengan status berhasil menjalan proses *database schema migration*.



4. Tampilan notifikasi bot aplikasi Slack ke *channel* #database-ops dengan status gagal menjalan proses *database schema migration* dengan informasi "SQL *command not valid*".