BAB I

PENDAHULUAN

A. Latar Belakang

Pada era informasi ini, *helpdesk* memiliki peranan yang sangat penting dalam menjamin ketersediaan dan kualitas layanan teknologi informasi (TI) pada suatu organisasi. *Helpdesk* adalah bagian TI yang pertama kali dihubungi *user* saat mempunyai pertanyaan atau masalah yang berhubungan dengan layanan TI. *Helpdesk* merupakan sebuah organisasi yang membantu menangani kebutuhan pelanggan atau *user* terkait dengan pertanyaan, pelayanan, dukungan teknis, atau keluhan terhadap produk dan jasa tertentu dengan memanfaatkan sistem penomoran (*request ticket*) untuk memudahkan penelusuran terhadap tindakan penyelesaian yang dikoordinasi oleh suatu tim. *Helpdesk* yang baik juga berfungsi untuk mencatat dan mengklasifikasikan permasalahan yang terjadi serta solusinya, sehingga dapat dipergunakan dan menjadi *asset knowledge* bagi instansi yang bersangkutan. *Helpdesk* sebaiknya dibantu oleh perangkat lunak tertentu untuk memfasilitasi rekap data, monitoring kegiatan, dan pelaporan.

Universitas Muhammadiyah Parepare merupakan salah satu perguruan tinggi yang memiliki fasilitas layanan *helpdesk*. Penulis melakukan observasi secara langsung terhadap *helpdesk* yang ada di Universitas Muhammadiyah Parepare untuk mengetahui pengetahuan mahasiswa terkait layanan *helpdesk* di lingkungan universitas. Penulis menyebar kuesioner secara *online* kepada mahasiswa

Universitas Muhammadiyah Parepare yang telah mengetahui sistem layanan *Helpdesk*.

Mereka lebih sering melakukan pengaduan secara langsung ke teknisi atau divisi terkait. Hal tersebut terjadi dikarenakan responden tidak puas dengan layanan *helpdesk* yang telah ada saat ini dengan alasan prosedur terlalu rumit dan fitur – fitur layanan *helpdesk* yang tersedia kurang memadai sehingga responden tidak tertarik untuk menggunakannya. Terbukti dari kuesioner yang penulis sebar, bahwa memang pengetahuan mahasiswa mengenai layanan *helpdesk* sudah cukup tinggi, namun minat dan penggunaannya masih rendah. Salah satu solusi untuk meningkatkan minat dan penggunaan *helpdesk* ialah dengan menerapkan *gamification*.

Gamification adalah konsep penerapan elemen *game* terhadap elemen *non-game*. Tujuan *gamification* ialah menciptakan interaksi dengan pengguna melalui suatu mekanisme yang menyerupai *game*. Pengguna akan merasakan pengalaman yang serupa dengan pengalaman yang muncul saat pengguna memainkan suatu *game*, yaitu adanya interaksi yang menyenangkan. Saat memainkan suatu *game*, pemain dituntut untuk mencapai *level* tertentu, melalui berbagai rintangan untuk mengakhiri suatu *level*, dan memperoleh *reward* saat berhasil mengakhiri suatu *level*. Elemen dan karakteristik *game* tersebut secara umum diterapkan pada *gamification* melalui *badge*, *point*, dan *leaderboard*. Ketika pengguna berhasil menyelesaikan suatu aktivitas tertentu, maka mereka akan memperoleh *badge* dan *point* sebagai *reward*. Perolehan *badge* dan *point* tersebut dapat dibandingkan dengan *badge* dan *point* yang diperoleh oleh pengguna lainnya *(leaderboard).*

(Hunter, R. (2011), *The Gamification Handbook Everything You Need ToKnow About Gamification*, Tebbo).

Berdasarkan hasil penelitian yang dilakukan oleh (Farozi, Suyanto, & Lutfi, 2015) didapatkan beberapa faktor menurunnya kinerja sumber daya manusia dalam sebuah perusahaan salah satunya adalah faktor kebosanan dalam bekerja. Penyebabnya bisa bermacam – macam, salah satunya adalah rutinitas atau pekerjaan yang dirasakan monoton karena selalu harus dikerjakan setiap hari dalam bentuk yang sama. Dengan dibentuknya sistem menggunakan metode gamifikasi, aplikasi bisnis pada perusahaan menjadi lebih menyenangkan untuk digunakan. Salah satu pendekatan yang dilakukan adalah dengan memberikan penghargaan kepada pengguna baik secara virtual maupun *non-virtual* ketika melakukan kegiatan di dalam sistem dan pengguna lain dapat melihat per*form*a dari masing – masing pengguna. Hal ini berimbas dalam meningkatnya keterikatan dan umpan balik pengguna dengan aplikasi tersebut. Keterikatan ini diharapkan akan memberikan hal positif berupa peningkatan per*form*a dari pengguna, dalam hal ini adalah mengunjungi serta memanfaatkan fungsi *helpdesk* yang telah disediakan.

Berdasarkan pemaparan permasalahan dan latar belakang diatas peneliti merancang sebuah sistem informasi *helpdesk*, dengan menggunakan konsep *gamification* sebagai metode untuk meningkatkan minat dan keaktifan pengguna dalam melaporkan permasalahan layanan TI di lingkungan universitas, yang akan dipaparkan lebih lanjut dalam penulisan skripsi yang berjudul "**Perancangan Aplikasi *Help Desk* Universitas Muhammadiyah Parepare**"

B. Rumusan Masalah

Berdasarkan pemaparan pada bagian latar belakang masalah, maka permasalahan yang akan dikaji pada penelitian ini adalah sebagai berikut :

"Bagaimana merancang Aplikasi *Help Desk* dapat meningkatkan keaktifan mahasiswa Universitas Muhammadiyah Parepare dalam menggunakan *helpdesk* pelayanan TI (Teknologi Informasi)?"

C. Batasan Masalah

Untuk mendapatkan sasaran sistem informasi yang tepat dan tujuan dari penelitian dapat tercapai, penulis membatasi ruang lingkup sistem informasi yang akan dirancang ini adalah sebagai berikut:

1. Implementasi dari aplikasi ini berfokus pada aplikasi layanan *helpdesk* dengan pada mahasiswa Universitas Muhammadiyah Parepare.
2. Lokasi penelitian dilakukan di Universitas Muhammadiyah Parepare

D. Tujuan Penelitian

Adapun Tujuan dari Penelitian ini adalah sebagai berikut:

Adapun tujuan yang ingin dicapai dalam pelaksanaan penelitian ini yaitu merancang sebuah aplikasi *helpdesk* dengan pendekatan *gamification* yang dapat meningkatkan keakfitan mahasiswa Universitas Muhammadiyah Parepare dalam dalam menggunakan *helpdesk* pelayanan TI.

BAB II

TINJAUAN PUSTAKA

A. Tinjauan Penelitian Sebelumnya

Penelitian relevan berisikan penelitian yang terdahulu kemudian dicari bahan penelitian yang ada dan relevan dengan penelitian ini karena sangat berguna untuk mendukung kajian teori yang dikemukakan, sehingga dapat digunakan sebagai landasan pada penyusunan kerangka pikir. Beberapa hasil penelitian yang relevan dengan penelitian ini adalah penelitian yang dilakukan oleh:

1. Andrianto & Wibowo (2021) – *"Pengembangan Aplikasi Pendataan Mahasiswa Menggunakan Visual Basic .NET dan SQL Server"* Penelitian ini merancang sistem aplikasi pendataan mahasiswa berbasis Visual Basic .NET 2019 yang terintegrasi dengan SQL Server. Metode pengembangan menggunakan Waterfall, dimulai dari analisis kebutuhan, desain antarmuka, hingga pengujian. Hasilnya adalah aplikasi yang mampu mengelola data mahasiswa, nilai, dan riwayat akademik secara cepat serta aman.
2. Saputra, Kurniawan, & Hidayat (2022) – *"Implementasi Augmented Reality pada Brosur Pameran Interior Berbasis Markerless di Android"* Aplikasi dibuat dengan Unity 3D dan ARCore untuk menampilkan furnitur dalam bentuk 3D pada brosur digital. Metode Rational *Unified Process* (RUP) digunakan untuk memastikan siklus pengembangan yang sistematis. Uji coba

menunjukkan pengalaman pengguna meningkat karena visualisasi yang lebih interaktif.

3. Lestari & Pratama (2023) – *"Augmented Reality untuk Brosur Promosi Kampus Menggunakan Metode Marker Based Tracking"* Penelitian ini bertujuan meningkatkan minat calon mahasiswa dengan brosur AR yang menampilkan gedung, laboratorium, dan fasilitas kampus dalam bentuk 3D. Aplikasi dibuat menggunakan Vuforia SDK dan diuji di perangkat *Android*. Hasilnya menunjukkan tingkat ketertarikan calon mahasiswa meningkat hingga 35% dibanding brosur biasa.
4. Rahmawati, Yuliana, & Anwar (2024) – *"Pengembangan Brosur Digital Interaktif Menggunakan Augmented Reality untuk Promosi Fakultas Teknik"* Metode yang digunakan adalah Prototype Model, dengan hasil akhir berupa aplikasi yang memuat animasi 3D, narasi suara, dan link interaktif ke website fakultas. Uji coba di 50 responden menunjukkan bahwa media ini lebih efektif dalam menyampaikan informasi dibanding brosur statis.
5. Firmansyah & Dewi (2022) – *"Penerapan Augmented Reality untuk Visualisasi Produk Interior Rumah pada Media Cetak Brosur"* Penelitian ini mengintegrasikan brosur cetak dengan fitur AR yang memungkinkan pengguna melihat produk seperti meja, kursi, dan lemari dalam skala sebenarnya menggunakan kamera smartphone. Pengujian menunjukkan peningkatan keputusan pembelian sebesar 20%.

1. *Helpdesk*

Menurut Rahman dan Nugroho (2021), *Helpdesk* merupakan unit layanan dalam organisasi yang berfungsi sebagai titik pusat bantuan (*centralized support*) untuk menangani masalah teknis, pertanyaan, maupun permintaan layanan dari pengguna, baik secara langsung maupun melalui media digital. Peran ini menjadi vital di era transformasi digital untuk menjaga kontinuitas layanan.

Fauzi, Hidayat, dan Sari (2022) menjelaskan bahwa *Helpdesk* modern berperan sebagai *Single Point of Contact* (SPOC), mengintegrasikan komunikasi antara pengguna dan tim pendukung melalui berbagai saluran seperti email, aplikasi, *chatbot*, dan portal layanan mandiri.

Menurut Setiawan dan Putri (2023), sistem *Helpdesk* berbasis *cloud* saat ini telah dilengkapi fitur *ticketing automation*, integrasi *knowledge base*, dan real-time analytics yang membantu perusahaan, lembaga pendidikan, maupun instansi pemerintah untuk memantau status tiket, menganalisis tren masalah, dan meningkatkan waktu respon.

Halim, Widodo, dan Pratama (2024) menambahkan bahwa *Helpdesk System* yang efektif tidak hanya menyediakan jawaban cepat, tetapi juga memanfaatkan *Artificial Intelligence* (AI) untuk memberikan rekomendasi solusi otomatis dan memprediksi potensi masalah berdasarkan riwayat layanan.

Berbeda dengan *service desk* atau *call center* tradisional, *Helpdesk* masa kini mengutamakan layanan mandiri (*self-service* portal), di mana

pengguna dapat mengakses panduan (FAQ, video tutorial, dan artikel teknis) untuk menemukan solusi secara mandiri, yang menurut Lestari dan Hakim (2025) terbukti dapat menurunkan beban kerja tim pendukung hingga 35% dan meningkatkan kepuasan pengguna.

a. Kelebihan *Helpdesk*

Helpdesk memiliki beberapa kelebihan, antara lain:

1. *Helpdesk* dapat memberikan solusi atas pertanyaan-pertanyaan maupun complain yang masuk dalam waktu yang lebih singkat.
2. *Helpdesk* dapat mengecek status permasalahan yang ada dan mengatur pembagian kerja staf.
3. *Helpdesk* dapat meningkatkan efisiensi perusahaan dalam menangani pertanyaan dan complain dari pelanggan.
4. *Helpdesk* dapat memberikan laporan kerja perkembangan kinerja para staff kepada pemimpin.

b. Tipe *Helpdesk*

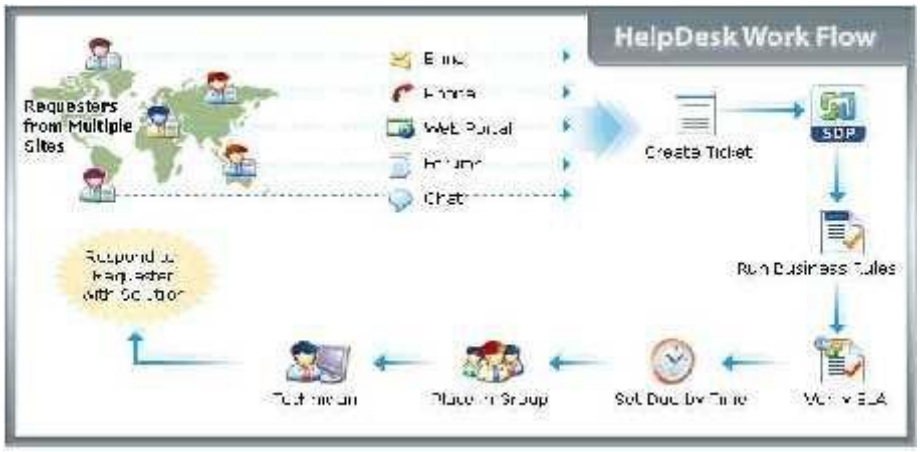
Berdasarkan tipe *customer* yang ditanganinya, *helpdesk* dapat dibagi menjadi dua jenis :

1. Internal *helpdesk* adalah *helpdesk* memberikan pelayanan kepada internal *customer* yaitu para karyawan yang bekerja pada perusahaan tersebut. Tujuan utama daripada dibentuknya *helpdesk* ini adalah membantu karyawan untuk dapat memaksimalkan infrastruktur yang ada untuk mengoptimalkan kinerja mereka.

2. *External helpdesk* adapun *external helpdesk* biasanya terdapat pada perusahaan yang melakukan penjualan produk atau layanan tertentu kepada pihak di luar perusahaan. Sehingga *helpdesk* disini ditekankan pada pelayanan terhadap *customer* pengguna produk/jasa perusahaan sebagai *additional services after* sales terhadap penjualan perusahaan tersebut.

2. Alur *Helpdesk*

Proses *helpdesk system* akan bekerja ketika layanan permintaan *email* dikirim oleh pemohon atau pelanggan maka *Web Helpdesk* otomatis mengkonversi *email* tersebut ke dalam layanan tiket atau layanan surat perintah kerja. Selanjutnya permohonan atau permintaan tersebut akan akan diproses sesuai dengan aturan dalam perusahaan dan diperiksa pada bagian *Service Level Agreement* (SLA) untuk diketahui pada staf bagian mana permintaan ini akan diserahkan setelah itu pelanggan hanya perlu menuggu sampai proses permintaan, permohonan atau keluhan direspon oleh tim *helpdesk* atau *place is group,* setelah solusi telah ada maka akan diserahkan kepada *tenhnician* untuk diteruskan ke pelanggan dan pelanggan akan menerima sebagai hasil respon *system*. Gambar 2.1 menunjukkan alur kerja *helpdesk system* dan Gambar 2.2. menunjukkan contoh aplikasi *helpdesk system.*



Gambar *2. 1 Alur kerja helpdesk*

3. JAVA

a. Pengertian Java

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat suatu program . Java juga mendukung sumber daya internet yang saat ini populer, yaitu *World Wide Web* atau yang sering disebut web. Java juga mendukung aplikasi klien/server, baik dalam jaringan lokal (LAN) maupun jaringan berskala luas (WAN).

Java dikembangkan oleh Sun *Microsystems* pada Agustus 1991, dengan nama Oak. Konon Oak adalah pohon semacam jati yang terlihat dari jendela tempat pembuatnya, James Gosling, bekerja. Ada yang mengatakan bahwa Oak adalah singkatan "*Object Application Kernel*", tetapi ada juga yang menyatakan hal itu muncul setelah nama Oak diberikan pada Januari 1995, karen nama Oak dianggap kurang komersial, maka diganti menjadi Java.

Dalam sejumlah literatur disebutkan bahwa Java merupakan hasil perpaduan sifat dari sejumlah bahasa pemrograman, yaitu C, C++, *Object*-C,*SmallTalk*, dan *Common* LISP. Selain itu Java juga dilengkapi dengan unsur keamanan. Yang tak kalah penting adalah bahwa Java menambahkan unsur paradigma pemrograman yang sederhana.

Dengan kata lain, Java merupakan bahasa pemrograman berorientasi objek (suatu model pengembangan perangkat lunak yang saat ini populer). Sebagai bahasa pemrograman berorientasi objek, Java menggunakan kelas untuk membentuk suatu objek. Sejumlah kelas sudah tersedia dan Anda dapat menggunakannya dengan mudah, dan bahkan dapat mengembangkannya dengan mudah, serta dapat mengembangkannya lebih jauh melalui konsep pewarisan . Pewarisan adalah sifat yang ada pada bahasa pemrograman berorientasi objek, yang memungkinkan sifat-sifat suatu objek diturunkan dengan mudah ke objek lain. Kurniawan dan Santoso (2021).

b. Karateristik java

1) Sederhana

Bahasa pemrograman Java menggunakan sintaks yang mirip dengan bahasa C++ namun sintaks pada Java telah banyak diperbaiki, terutama dengan menghilangkan *pointer* yang rumit dan *multipleinheritance*. Java juga menggunakan *automatic memory allocation* dan *garbage collection*.

2) Berorientasi Objek

Java merupakan bahasa pemrograman berorientasi objek yang memungkinkan program untuk dibuat secara modular dan digunakan kembali.

3) Terdistribusi

Java dibuat untuk memudahkan distribusi aplikasi dengan adanya *networking libraries* yang terintegrasi dalam Java.

4) Interpreted

Program Java dijalankan menggunakan program *Interpreter,* yaitu Java *Virtual Machine* (JVM). Hal ini menyebabkan source code Java yang telah dikompilasi menjadi *bytecodes* dapat dijalankan pada berbagai *platform*.

5) Robust

Java mempunyai *reliabilitas* yang tinggi. *Kompiler* pada Java mempunyai kemampuan mendeteksi *error* yang lebih baik dibandingkan bahasa pemrograman yang lain. Java mempunyai *Runtime Exception Handling* untuk membantu mengatasi *error* pada pemrograman.

6) Secure

Sebagai bahasa pemrograman aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga agar aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

c. Cara Kerja

Program Java bersifat tidak bergantung pada *platform*, artinya Java dapat dijalankan pada sembarang komputer dan bahkan pada sembarang sistem operasi. Ketidakbergantungan terhadap *platform* sering dinyatakan dengan istilah portabilitas. Yang menarik, tingkat portabilitas Java tidak hanya sebatas program sumber *(source code)*, melainkan juga pada tingkat kode biner yang disebut *bytecode*. Kode yang disebut *bytecode* dapat dijalankan pada berbagai sistem operasi karena kode ini berbeda dengan kode mesin . Kode mesin sangat bergantung pada *platform*, sedangkan *bytecode* dapat dimengerti oleh semua *platform* yang telah di lengkapi dengan *interpreter* Java. Mengingat bahwa hasil kompilasi Java dapat dijalankan pada sebarang sistem operasi ataupun prosesor, Java sering dikatakan bersikap netral terhadap arsitektur komputer.

4. *Smartphone*

Smartphone adalah perangkat telekomunikasi serbaguna. *Smartphone* atau yang disebut juga piranti pintar dewasa ini menjadi *trend* yang mewabah di Indonesia, digemarinya *Smartphone* ini juga bukan tanpa alasan, tapi karena feature yang ditawarkan sangat menarik dan mengubah hobi pengguna untuk *browsing*, *chat*ing dan semacam nya yang awalnya dilakukan secara setatis, sekarang dengan *smartphone* bisa dilakukan dengan mobile atau bergerak.

Seiring meningkatnya kecerdasan pengguna dalam memilih perangkat pintar, para vendor berlomba-lomba menawarkan fitur dan antarmuka yang menarik serta mudah digunakan. Sistem operasi (*operating system*) seperti *Android* dan iOS mendominasi pasar global, sementara beberapa vendor mengembangkan ekosistemnya sendiri untuk meningkatkan loyalitas pengguna. Menurut Pratama dan Hidayat (2023), *Android* yang diadopsi oleh merek seperti Samsung, Xiaomi, dan Oppo tetap memimpin pangsa pasar, sedangkan iOS yang digunakan Apple mempertahankan segmen premium.

5. *Android*

Android merupakan sistem operasi berbasis Linux yang dirancang khusus untuk perangkat seluler layar sentuh seperti ponsel pintar dan tablet. Menurut Rahman dan Putri (2022)**,** *Android* saat ini telah berkembang menjadi ekosistem terbuka yang mendukung jutaan aplikasi melalui *Google* Play Store, dengan arsitektur modular yang memudahkan pembaruan keamanan dan kompatibilitas perangkat. Sistem operasi ini dikembangkan oleh *Android* Inc. sebelum diakuisisi oleh *Google* pada tahun 2005, dan sejak itu menjadi *platform mobile* terpopuler di dunia

Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Posel pertama yang menggunakan *Android* sebagai OSnya adalah *HTC Dream* yang dirilis pada 22 Oktober 2008. Dan hingga kini

setidaknya ada sudah ada 8 jenis versi OS *Andoid* yang sudah beredar di dunia telepon seluler. Jenis-jenis *Android* tersebut antara lain:

a. *Android versi 1.1*

Android versi 1.1 di rilis pada 9 Maret 2009 oleh Google. *Android* versi ini dilengkapi disupport oleh *Google Mail Service* dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

b. *Android versi 1.5 Cup Cake*

Android Cup Cake di rilis pada pertengahan Mei 2009, masih oleh Google Inc. Adroid ini dilengkapi *software development kit* dengan berbagai pembaharuan termasuk penambahan beberapa fitur antara lain yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube, upload gambar ke Picasa langsung dari telepon, serta mendapat dukungan Bluetooth A2DP.

c. *Android versi 1.6 Donut*

Android Donut di rilis pada September 2009 menampilkan proses pencarian yang lebih baik dibandingkan versi-versi sebelumnya. Selain itu *Android* Donut memiliki fitur-fitur tambahan seperti galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, camcorder dan galeri yang dintegrasikan; *Text-to-speech engine*; kemampuan dial kontak; teknologi *text to change speech. Android Donut* juga dilengkapi baterai indikator, dan kontrol applet VPN.

d. *Android versi 2.0/2.1 Eclair*

Android Eclair dirilis pada 3 Desember 2009. Perubahan yang ada antara lain adalah pengoptimalan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, *digital Zoom*, dan *Bluetooth* 2.1. *Android Eclair* merupakan Android pertama yang mulai dipakai oleh banyak *smartphone*, fitur utama *Eclair* yaitu perubahan total struktur dan tampilan *user interface*.

e. *Android versi 2.2 Froyo (Frozen Yogurt)*

Android Froyo dirilis pada 20 mei 2012. Android versi ini memiliki kecepatan kinerja dan aplikasi 2 sampai 5 kali dari versi-versi sebelumnya. Selain itu ada penambahan fitur-fitur baru seperti dukungan *Adobe Flash* 10.1, intergrasi V8 *Java Script engine* yang dipakai *Google Chrome* yang mempercepat kemampuan rendering pada *browser*, pemasangan aplikasi dalam SD Card, kemampuan *WiFi Hotspot portabel*, dan kemampuan auto *update* dalam aplikasi *Android Market*.

f. *Android versi 2.3 Gingerbread*

Andoid *Gingerbread* di rilis pada 6 Desember 2010. Perubahan-perubahan umum yang didapat dari *Android* versi ini antara lain peningkatan kemampuan permainan (gaming), peningkatan fungsi *copy paste,* layar antar muka *(User Interface*) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (*reverb, equalization, headphone virtualization,* dan *bass boost*), dukungan kemampuan *Near*

Field Communication (NFC), dan dukungan jumlah kamera yang lebih dari satu.

g. *Android versi 3.0/3.1 Honeycomb*

Android Honeycomb di rilis pada awal 2012. Merupakan versi *Android* yang dirancang khusus untuk *device* dengan layar besar seperti Tablet PC. Fitur baru yang ada pada *Android* Honeycomb antara lain yaitu dukungan terhadap prosessor multicore dan grafis dengan *hardware acceleration*. *User Interface* pada Honeycomb juga berbeda karena sudah didesain untuk tablet. Tablet pertama yang memakai Honeycomb adalah tablet Motorola Xoom yang dirilis bulan Februari 2011. Selain itu sebuah perangkat keras produksi Asus bernama Eee Pad Transformer juga menggunakan OS *Android* honeycomb dan diharapkan akan masuk ke pasaran Indonesia pada Mei 2011.

h. *Android versi 4.0 ICS (Ice Cream Sandwich)*

Android Ice Cream Sandwich diumumkan secara resmi pada 10 Mei 2011 di ajang Google I/O Developer Conference (San Francisco), pihak Google mengklaim *Android* Ice Cream Sandwich akan dapat digunakan baik di *Smartphone* ataupun tablet. *Android* Ice Cream Sandwich membawa fitur Honeycomb untuk *Smartphone* serta ada penambahan fitur baru seperti membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara offline, dan berbagi informasi dengan menggunakan NFC. Ponsel

pertama yang menggunakan sistem operasi ini adalah Samsung Galaxy Nexus.

i. *Android versi 4.1 Jelly Bean*

Android Jelly Bean juga diluncurkan pada acara Google I/O 10 Mei 2011 yang lalu. *Android* versi ini membawa sejumlah keunggulan dan fitur baru, diantaranya peningkatkan input keyboard, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Versi ini juga dilengkapi Google Now yang dapat memberikan informasi yang tepat pada waktu yang tepat pula. Salah satu kemampuannya adalah dapat mengetahui informasi cuaca, lalu-lintas, ataupun hasil pertandingan olahraga. Sistem operasi *Android Jelly Bean 4.1* pertama kali digunakan dalam produk tablet Asus, yakni Google Nexus 7.

j. *Android Versi 4.4 Kitkat*

Sistem operasi *Android* terbaru yang akan melanjutkan sistem operasi *Android* seri 4.1, 4.2 dan 4.3 *Jelly Bean* ini mempunyai nama *KitKat.* Pemberian nama *KitKat* sendiri cukup mengejutkan bagi sejumlah kalangan, pasalnya jauh sebelumnya rumor telah beredar luas bahwa sistem operasi penerus *Jelly Bean* mempunyai nama *Key Lime Pie.*

Pemberian nama *KitKat* sendiri memang mengejutkan, pasalnya *KitKat* merupakan nama sebuah produk bukan nama sebuah makanan umum yang biasa digunakan *Google.* Namun alasan *Google*

menggunakan nama *KitKat* adalah produk ini merupakan coklat batangan kesukaan seorang Kepala Teknik *Android Google Hiroshi Lockheimer*, ungkap juru bicara *Google.*

Yang menarik dari hal ini adalah tidak terjadi adanya pertukaran uang di dalam penamaan *Android KitKat* yang dilakukan oleh pihak *Google* dan *Nestle*. Namun, elemen penting yang terjadi di dalam pemberian nama ini adalah produk *KitKat* adalah adanya produk *KitKat* sebanyak 50 juta yang ada di 19 negara akan memiliki branding *Android* dan menawarkan pembeli untuk mendapatkan kesempatan memenangkan *Google Play Gift Card* dan *Nexus 7*.

k. *Android lollipop (versi 5.0)*

Google kembali meluncurkan versi *Android* terbarunya diberi nama *Android* lollipop atau disingkat dengan *Android* L. Beberapa media terkemuka seperti tech times menuturkan bahsa *Android* lollipop ini akan dirilis, memiliki fitur super canggih dan sudah tertanam dalam tablet nexus 9. Banyak sekali fitur baru dalam *Android* lollipop.

l. *Android marshmallow (versi 6.0)*

Google kembali merilis versi teranyar dari si robot hijau versi *Android* 6.0 ini dinamakan dengan *Android* marshmallow. Pengembangan *Android* telah menanamkan berbagai fitur baru nan canggih dalam versi ini termasuk juga penyempurnaan fitur dari versi sebelumnya.

m. *android nougat (versi 7.0)*

Versi selanjutnya adalah *Android* versi 7.0 atau disebut juga dengan *Android* nougat. Versi nougat ini resmi dirilis pada tanggal 22 agustus 2016 berbagai fitur pada *Android* nougat juga telah disempurnakan termasuk juga penambahan fitur dan fungsi.

6. *Eclipse*

Eclipse adalah sebuah Integrated *Development Environment* (IDE) yang bersifat lintas plat*form* dan mendukung pengembangan berbagai bahasa pemrograman, termasuk Java, C/C++, Python, dan PHP. Menurut Syahputra dan Nugroho (2021), *Eclipse* tidak hanya digunakan untuk pemrograman, tetapi juga untuk manajemen siklus hidup perangkat lunak seperti dokumentasi, pengujian, dan pengembangan web. Keunggulan utama *Eclipse* terletak pada sifatnya yang gratis, *open source*, serta dukungan komunitas global yang besar, sehingga tetap menjadi salah satu IDE favorit di kalangan pengembang.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini.

7. *Android* Developer *Tools*

Android Developer Tools (ADT) adalah plugin untuk *Eclipse* yang menyediakan lingkungan pengembangan terintegrasi bagi pengembang *Android*. Menurut Wijaya dan Pratama (2022), meskipun Google saat ini lebih mendorong penggunaan *Android* Studio, ADT masih digunakan dalam beberapa proyek tertentu karena integrasinya yang baik dengan *Eclipse*,

mendukung proses pengkodean, pengujian, debugging, hingga packaging aplikasi. ADT menawarkan fitur visual dan manajemen proyek yang memudahkan pengembang, terutama pada proyek warisan (*legacy projects*) yang masih menggunakan *Eclipse* sebagai IDE utama.

8. XML

   XML *(Extensible Markup Language)* adalah bahasa markup yang direkomendasikan oleh W3C untuk memfasilitasi pertukaran data antar sistem yang beragam. Menurut Putra dan Rahman (2023), XML memiliki struktur yang fleksibel sehingga memudahkan penyimpanan, pengelolaan, dan pertukaran data secara ringkas. XML banyak digunakan dalam integrasi sistem, web services, dan aplikasi lintas platform karena dapat dibaca oleh manusia maupun mesin, sekaligus bersifat independen terhadap platform.

9. JRE (Java *Runtime Environment*)

   Java Runtime Environment Java *Runtime Environment* (JRE) adalah perangkat lunak yang menyediakan *Runtime* untuk menjalankan aplikasi berbasis Java. Menurut Ali dan Khan (2022), kompatibilitas antara versi JRE dan JDK sangat penting karena perbedaan versi dapat menyebabkan masalah eksekusi, terutama pada fitur bahasa atau pustaka standar yang diperkenalkan pada versi Java terbaru. Disarankan untuk menggunakan JRE dengan versi yang sama atau lebih tinggi dibandingkan JDK saat aplikasi dikompilasi, guna memastikan konsistensi perilaku program.

10. JDK (Java *Development Kit*)

Java Development Kit (JDK) adalah sebuah produk yang dikembangkan oleh Oracle yang ditujukan untuk para *developer java*. Sejak Java, diperkenalkan JDK merupakan *Java Software Development Kit* (SDK) yang paling serig digunakan Pada tanggal 17 November 2006, Sun mengatakan akan merilis JDK sebagai *opensource* menggunakan lisensi GNU *General Public License* (GPL), artinya *software* ini sekarang menjadi *free software*. Pada tanggal 8 Mei 2007, Sun mengontribusikan *source code-nya* ke OpenJDK.

JDK juga berisi paket *Java Runtime Environment* yang komplit, biasaya disebut private run time karena biasanya dipisah dari JRE reguler dan dilengkapi dengan konten tambahan, yaitu terdiri dari Java *Virtual Machine* dan semua *class* library yang ada di *Environment* produk. Selain itu termasuk juga library tambahan yang berguna untuk developer, misalnya library internationalization dan library IDL.

Salinan dari JDK juga berisi beberapa contoh program yang menjelaskan penerapan porsi dari API Java Menurut Patel dan Sharma (2023), Java *Development Kit* (JDK) adalah komponen inti dari *software development kit* (SDK) untuk bahasa Java yang mencakup kompiler (javac), Java *Runtime Environment* (JRE), pustaka API, serta alat bantu seperti debugger dan dokumentasi. JDK sering disertai dengan contoh kode (sample programs) yang menunjukkan penerapan API untuk membantu pengembang memahami penggunaan pustaka standar. Dalam

pengembangan Java SE, EE, maupun ME, JDK menjadi pusat proses compile dan build, sedangkan bagian SDK lain seperti server aplikasi dan alat dokumentasi mendukung proses pengembangan yang lengkap.

11. *Android* SDK (*Software Development Kit*)

Android SDK adalah seperangkat alat dan pustaka API yang disediakan Google untuk memungkinkan pengembang membangun aplikasi *Android*. SDK mencakup compiler, debugger, emulator, serta dokumentasi API yang diperlukan untuk mengembangkan, menguji, dan memublikasikan aplikasi. Meskipun bahasa resmi *Android* modern adalah Kotlin, Java tetap menjadi bahasa utama yang didukung penuh oleh *Android* SDK, sehingga kompatibel dengan proyek-proyek lama maupun baru. Rahman dan Gupta (2023).

12. *Android* studio

Android Studio adalah sebuah IDE untuk *Android Development* yang diperkenalkan *google* pada acara *Google* I/O 2013. *Android* Studio merupakan pengembangkan dari *Eclipse* IDE, dan dibuat berdasarkan IDE Java populer, yaitu *IntelliJ* IDEA. *Android* Studio merupakan IDE resmi untuk pengembangan aplikasi *Android*. Sebagai pengembangan dari *Eclipse*, *Android* Studio mempunyai banyak fitur-fitur baru dibandingkan dengan *Eclipse* IDE. Berbeda dengan *Eclipse* yang menggunakan Ant, *Android* Studio menggunakan Gradle sebagai build *Environment*. Fitur-fitur lainnya adalah sebagai berikut Menggunakan *Gradle-based build system* yang fleksibel.

1. Bisa mem-*build* multiple APK .
2. Template support untuk *Google Services* dan berbagai macam tipe perangkat.
3. *Built-in support* untuk *Google Cloud Platform*, sehingga mudah untuk integrasi dengan *Google Cloud Messaging* dan *App Engine*.
4. Import library langsung dari *Maven repository*

Ketika awal saya berkenalan dengan *Android*, saya menggunakan *Eclipse* sebagai IDE untuk membuat aplikasi *Android*. Jika dibandingkan dengan *Android* Studio memang dari sisi build lebih baik dibandingkan *Eclipse*, karena *Android* Studio menggunakan *Gradle*. Ditambah lagi berbeda dengan *Eclipse*, *kit*a tidak perlu lagi dipusingkan dengan *dependencies package* pada *Android* Studio. Satu hal tambahan lagi yang membuat *Android* Studio unggul adalah dukungan layout xml *edit*or secara visual yang jauh lebih baik daripada *Eclipse*. Walaupun begitu, *Android* Studio saat ini masih dalam tahap beta dan belum mempunyai dukungan untuk NDK/*Native Development Kit*.

a. *Android* Studio Sebagai *Code Edit*or Cerdas

   Inti dari *Android* Studio adalah *edit*or kode cerdas mampu *code completion* dengan cerdas, *refactoring*, dan analisis *code*. *Edit*or yang baik membantu Anda menjadi pengembang aplikasi *Android* lebih produktif.

b. Tersedia template dan bisa integrasi GitHub

New project wizards membuatnya lebih mudah dari sebelumnya untuk memulai sebuah *prjectk* baru, memulai project menggunakan *kode template* untuk pola seperti navigasi laci dan melihat halaman, dan bahkan mengimpor contoh kode *Google* dari *GitHub*.

c. *Multi-screen app development*

Membangun aplikasi untuk ponsel *Android*, *tablet*, *Android Wear*, *Android* TV,*Android Auto dan Google Glass*. Dengan *Project Android View* baru dan modul dukungan di *Android* Studio, lebih mudah untuk mengelola *project* aplikasi dan sumber daya.

d. *Virtual devices for all shapes and sizes*

Android Studio hadir pra-dikonfigurasi dengan emulator gambar yang optimal.Diperbarui dan efisien Virtual *Device* Manager memberikan profil perangkat yang telah ditetapkan untuk perangkat *Android* umum.

e. *Android builds evolved, with Gradle*

Membuat beberapa APK untuk aplikasi *Android* Anda dengan fitur yang berbeda menggunakan *project* yang sama. Mengelola aplikasi dependensi dengan *Maven*.Membangun APK dari *Android* Studio atau *command line.*

13. *System Requirements Android* Studio

a. Windows

1. Microsoft® Windows® 10/8/7/Vista (32 or 64-bit)
2. 2 GB RAM minimum, 4 GB RAM *recommended*

3. 400 MB *hard disk space*
4. At least 1 GB for *Android* SDK, *emulator system images*, and *caches*
5. 1280 x 800 minimum *screen resolution*
6. Java *Development Kit* (JDK) 7
7. *Optional for accelerated emulator*: Intel® processor with *support* for Intel® VT-x, Intel® EM64T (Intel® 64), *and Execute Disable* (XD) Bit *functionality*

b. Mac OS X

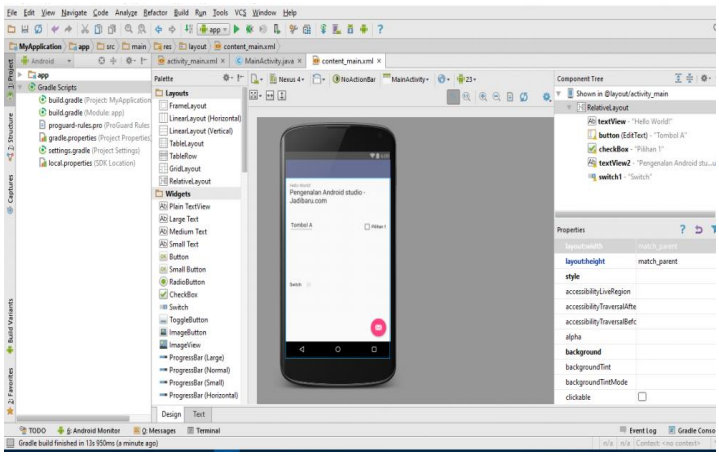
1. Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks)
2. 2 GB RAM minimum, 4 GB RAM *recommended*
3. 400 MB *hard disk space*
4. At least 1 GB for *Android* SDK, *emulator system images, and caches*
5. 1280 x 800 minimum *screen resolution*
6. *Java Runtime Environment* (JRE) 6
7. *Java Development Kit* (JDK) 7
8. *Optional for accelerated emulator*: Intel® *processor with support for Intel*® VT-x, Intel® EM64T (Intel® 64), *and Execute Disable* (XD) Bit *functionality*

   On Mac OS, run Android Studio with Java Runtime Environment (JRE) 6 *for optimized font rendering. You can then*

configure your project to use Java Development Kit (JDK) 6 or JDK 7.

c. Linux

1. GNOME or KDE desktop
2. GNU C Library (glibc) 2.15 or later
3. 2 GB RAM minimum, 4 GB RAM *recommended*
4. 400 MB *hard disk space*
5. At least 1 GB for *Android* SDK, *emulator system images, and caches*
6. 1280 x 800 minimum *screen resolution*
7. Oracle® Java *Development Kit* (JDK) 7



Gambar *2. 2 Android Studio*

14. *Tesseract Engine*

a. Pengertian

Tesseract merupakan sebuah *library open source* untuk mengenali karakter OCR *(Optical Character recognition)* yang pada

awalnya di kembangkan oleh *Hawlet-Packard* antara tahun 1985 dan 1995.

Tesseract tidak pernah dikomersialkan. Keakuratannya diuji di *The Fourt Annual Test of OCR Accurasi* yang diselenggarakan pada tahun 1995 di Universitas *Nevada Las Vegas' Information Science Research Institute*. Namun pada saat itu pengembangan *tesseract* telah berhenti. (Digib Unila,2011)

b. Cara Kerja

Gambaran mengenai cara kerja *Tesseract* secara singkat adalah:

1). Garis-garis dianalisis dan disimpan

2). Garis-garis berkumpul bersama sebagai himpunan

3). Himpunan tersebut akan disusun dalam baris teks

4). Baris teks akan dipecah menjadi kata-kata

5).Langkah pertama proses pengenalan mencoba untuk mengenali tiap kata pada urutannya

6). Kata yang tepat harus melewati proses penyesuaian oleh *adaptive trainer*

7). Pembelajaran oleh adaptive trainer berlangsung pada tahap kedua yang mencoba mengenali kata-kata yang tidak dikenali secara tepat pada tahap pertama

8). Proses fuzzy diselesaikan dan teks diperiksa *small caps*

9). Teks digital dikeluarkan Selama proses tersebut berlangsung,

tesseract menggunakan:

a) Algoritma untuk mendeteksi *skewed pages.*

b) Algoritma untuk mendeteksi kata-kata proporsional dan non proporsional (proporsional merupakan sebuah kata di mana semua huruf yang lebar yang sama).

c) Algoritma untuk mencacah karakter bergabung dan untuk menghubungkan karakter yang rusak.

d) Linguistik analisis untuk mengidentifikasi kata yang paling mungkin dibentuk oleh sekelompok karakter.

e) Dua karakter pengklasifikasi: *class*ifier statis, dan sebuah *class*ifier yang adaptif menggunakan data pelatihan, dan yang lebih baik dalam membedakan antara atas dan huruf kecil.

Tesseract memiliki akurasi yang tinggi jika teks yang diterjemahkan berasal dari bahasa yang *support* untuk bahasa Inggris, 8 komponen yang digunakan:

1. *General Words Wordlist* (tessdata/eng.word-dawg).
2. *Frequent Word Wordlist* (tessdata/eng.freq-dawg).
3. *User Wordlist* (tessdata/eng.*user*-words).
4. *Index for Character Set* (tessdata/eng.inttemp).
5. *Box file for use in locating characters in the training file*
6. *Box file for use in locating characters in the training file* 30
7. *Language's Character Set* (tessdata/eng.unicharset).
8. *Character Cluster Disambiguator for 'm' and 'rn', for instance.*

15. UML(*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berpradigma berorientasi objek". Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

Metode *Unified Modeling Language* (UML) menggunakan tiga bangunan dasar untuk mendeskripsikan sistem atau perangkat lunak yang akan dikembangkan, yaitu:

a. Sesuatu (*things*)

Ada empat *things* dalam *Unified Modeling Language* (UML):

1. *Structural things*, bagian yang relatif statis dapat berupa elemen-elemen yang bersifat fisik maupun konseptual.
2. *Behavorial things,* bagian dinamis biasanya merupakan kata kerja dari model UML yang mencerminkan perilaku sepanjang waktu.
3. *Grouping things,* bagian pengorganisasian dalam UML. Dalam penggambaran model UML yang rumit diperlukan penggambaran paket yang menyederhanakan model. Paket-paket ini kemudian dapat didekomposisi lebih lanjut. Paket berguna bagi pengelompokan sesuatu, misalnya model-model serta subsitem-subsistem.
4. *An notational things,* merupakan bagian yang meperjelas model UML. Dapat berisi komentar yang menjelaskan fungsi serta ciri-ciri tiap element dalam model UML.

b. Relasi (*relationship*)

Ada empat *relationship* (hubungan) dalam *Unified Modeling Language* (UML):

1. Ketergantungan (d*ependency*) adalah hubungan dimana perubahan yang terjadi pada suatu elemn independent akan mempengaruhi elemen yang bergantung padanya.
2. Asosiasi adalah apa dan bagaimana yang mengubungkan antara objek satu dengan yang lainnya. Suatu bentuk asosiasi adalah agregasi yang menampilkan hubungan suatu objek dengan bagian-bagiannya.
3. Generalisasi adalah hubungan dimana objek anak berbagai perilaku dan struktur data dari objek yang ada diatasnya (objek induk). Arah dari objek induk ke objek anak dinamakan spesialisasi sedangkan arah sebaliknya dinamakan generalisasi..
4. Realisasi adalah operasi yang benar-benar dilakukan oleh suatu objek.

c. Diagram, *Unified Modeling Language* (UML) menyediakan Sembilan jenis diagram yang dapat dikelompokkan berdasarkan sifatnya (statis dan dinamis).

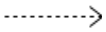
d. diagram *use case*

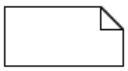
Diagram *use case* menyajikan interaksi antara *use case* dan aktor, dimana aktor dapat berupa orang, peralatan, atau sistem lain yang berinteraksi dengan sistem yang sedang di bangu. *Use case* menggambarkan

fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari pandangan pemakai

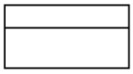
Adapun simbol-simbol *Use Case Diangram* antara lain :

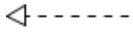
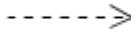
Tabel 2. 1 Simbol Use Case Diagram

GAMBAR	NAMA	KETERANGAN
	*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
	*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
	*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
	*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
	*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.

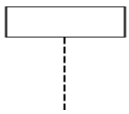
GAMBAR	NAMA	KETERANGAN
	*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
	*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
	*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

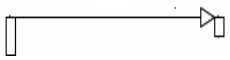
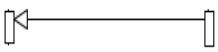
Tabel 2. 2 Simbol Class Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary* *Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.

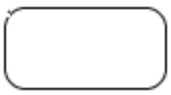
NO	GAMBAR	NAMA	KETERANGAN
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

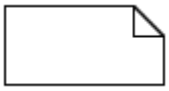
Tabel 2. 3 Simbol Sequence Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.

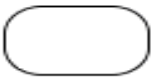
NO	GAMBAR	NAMA	KETERANGAN
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi
3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

Tabel 2. 4 Simbol Statechart Diagram

GAMBAR	NAMA	KETERANGAN
	*State*	Nilai atribut dan nilai link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
	*Initial Pseudo State*	Bagaimana objek dibentuk atau diawali
	*Final State*	Bagaimana objek dibentuk dan dihancurkan
	*Transition*	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya

GAMBAR	NAMA	KETERANGAN
	Associati on	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	*Node*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 5 Simbol Actifity Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

B. Penelitian terdahulu

Penelitian relevan berisikan penelitian yang terdahulu kemudian dicari bahan penelitian yang ada dan relevan dengan penelitian ini karena sangat berguna untuk mendukung kajian teori yang dikemukakan, sehingga dapat digunakan sebagai landasan pada penyusunan kerangka pikir. Beberapa hasil penelitian yang relevan dengan penelitian ini adalah penelitian yang dilakukan oleh:

1. Andrianto & Wibowo (2021) – *"Pengembangan Aplikasi Pendataan Mahasiswa Menggunakan Visual Basic .NET dan SQL Server"* Penelitian ini merancang sistem aplikasi pendataan mahasiswa berbasis Visual Basic .NET 2019 yang terintegrasi dengan SQL Server. Metode pengembangan menggunakan Waterfall, dimulai dari analisis kebutuhan, desain antarmuka, hingga pengujian. Hasilnya adalah aplikasi yang mampu mengelola data mahasiswa, nilai, dan riwayat akademik secara cepat serta aman.
2. Saputra, Kurniawan, & Hidayat (2022) – *"Implementasi Augmented Reality pada Brosur Pameran Interior Berbasis Markerless di Android"* Aplikasi dibuat dengan Unity 3D dan ARCore untuk menampilkan furnitur dalam bentuk 3D pada brosur digital. Metode Rational *Unified* Process (RUP) digunakan untuk memastikan siklus pengembangan yang sistematis. Uji coba menunjukkan pengalaman pengguna meningkat karena visualisasi yang lebih interaktif.
3. Lestari & Pratama (2023) – *"Augmented Reality untuk Brosur Promosi Kampus Menggunakan Metode Marker Based Tracking"* Penelitian ini bertujuan meningkatkan minat calon mahasiswa dengan brosur

AR yang menampilkan gedung, laboratorium, dan fasilitas kampus dalam bentuk 3D. Aplikasi dibuat menggunakan Vuforia SDK dan diuji di perangkat *Android*. Hasilnya menunjukkan tingkat ketertarikan calon mahasiswa meningkat hingga 35% dibanding brosur biasa.

4. Rahmawati, Yuliana, & Anwar (2024) – *"Pengembangan Brosur Digital Interaktif Menggunakan Augmented Reality untuk Promosi Fakultas Teknik"* Metode yang digunakan adalah Prototype Model, dengan hasil akhir berupa aplikasi yang memuat animasi 3D, narasi suara, dan link interaktif ke website fakultas. Uji coba di 50 responden menunjukkan bahwa media ini lebih efektif dalam menyampaikan informasi dibanding brosur statis.
5. Firmansyah & Dewi (2022) – *"Penerapan Augmented Reality untuk Visualisasi Produk Interior Rumah pada Media Cetak Brosur"* Penelitian ini mengintegrasikan brosur cetak dengan fitur AR yang memungkinkan pengguna melihat produk seperti meja, kursi, dan lemari dalam skala sebenarnya menggunakan kamera smartphone. Pengujian menunjukkan peningkatan keputusan pembelian sebesar 20%.

BAB III

METODE PENELITIAN

A. Jenis Penelitian

Jenis penelitian yang digunakan adalah penelitian deskriptif dimana memberikan gambaran mengenai apa yang sesungguhnya terjadi. Dalam pembuatan Skripsi ini digunakan metode deskripsif yang menggambarkan fakta-fakta dan informasi secara sistematis, faktual dan akurat.

Penelitian ini dilakukan melalui internet yang dapat memberikan sumber data dan pengetahuan mengenai sistem yang diteliti, kemudian mencocokkan dengan kemungkinan yang terjadi dalam usaha penyelesaian masalah.

B. Waktu dan Tempat Penelitian

Adapun waktu dan tempat penelitian yang dilaksanakan pada bulan Februari tahun 2023, di Media Center Universitas Muhammadiyah Parepare, dimana proses pengambilan data yang akan dimasukkan dalam aplikasi nantinya

NO	Uraian Kegiatan	Bulan 2023				
		Februari	Maret	Maret	April	April
1	Studi Literatur					
2	Perancangan aplikasi					
3	Pembuatan aplikasi					
4	Pengujian					

C. Metode Pengumpulan Data

Untuk memperoleh data-data yang dibutuhkan dalam rangka melakukan penelitian, maka penulis mengumpulkan data melalui beberapa cara yaitu :

1. Analisis Data

   Menganalisa data-data yang sebelumnya telah dikumpulkan.

2. Perancangan Program

   Sebagai pedoman dalam penulisan program atau kode-kode agar berjalan sesuai rencana.

3. Uji Coba Program

   Pengujian program dilakukan untuk memastikan bahwa program yang dibuat dapat berjalan dengan baik.

4. Evaluasi

   Sistem yang telah selesai dibangun perlu adanya evaluasi untuk menemukan kelemahan yang terdapat pada program yang telah dibangun tadi, yang nantinya bisa digunakan sebagai acuan untuk memperbaiki program sehingga lebih sempurna.

D. Jenis Data

Data yang digunakan dalam penelitian ini berupa data-data yang telah dikumpulkan melalui Penelitian Pustaka (*Library Research*). Adapun jenis data primer dan data sekunder yang relevan dengan masalah yang akan dibahas.

a. Data primer

   Data Primer adalah data yang berasal atau data yang diperoleh langsung dari sumber data dan pengetahuan.

b. Data Sekunder

   Data sekunder adalah data yang diperoleh tidak secara langsung dari objek penelitian. Peneliti mendapatkan data yang sudah jadi dari internet, website dan jurnal.

E. Tahapan Peneletian

Tahapan penelitian yang dimaksud dalam penelitian ini ada beberapa tahapan yaitu persiapan penelitian, pengumpulan data, analisis perancangan, pengujian dan implementasi. Adapun Uraian dari tahapan tersebut adalah sebagai berikut :

1. Persiapan Penelitian

   Pada tahapan ini peneliti melakukan persiapan penelitian. Persiapan penelitian yang dimaksud adalah menyiapkan buku-buku, artikel-artikel tentang topik penelitian serta *software* yang digunakan selama penelitian.

2. Pengumpulan Data

   Pada tahap ini peneliti melakukan observasi dengan peninjauan, pencatatan dan pengamatan langsung di tempat penelitian.

3. Analisis

   Pada tahap analisis, peneliti melakukan analisa terhadap sistem yang di terapkan sekarang berdasarkan kemudian merumuskan masalah yang menjadi pokok penelitian sehingga dapat dibuat alternatif pemecahan masalah.

4. Perancangan

   Peneliti kemudian merancang aplikasi yang ingin dibuat berdasarkan alternatif pemecahan masalah.

5. Pengujian

   Setelah melakukan perancangan, peneliti kemudian menguji hasil perancangan yang telah dibuat. Jika hasil perancangan terdapat kekurangan atau kelemahan maka kembali ke tahap analisis.

6. Implementasi

Setelah pada perancangan tidak terdapat kekurangan maka aplikasi siap untuk di gunakan oleh *user*.

F. Metode Pengujian Aplikasi

Beberapa *test-case* harus dilaksanakan dengan beberapa perbedaan strategi, query, atau jalur navigasi yang mewakili penggunaan sistem yang *tipical,* kritis atau abnormal. Isu kunci pada pengembangan sistem adalah pemilihan sekelompok *test-case* yang cocok, sekecil dan secepat mungkin, untuk meyakinkan perilaku sistem secara detail. Pengujian harus mencakup *unit testing*, yang mengecek validasi dari prosedur dan fungsi-fungsi secara independen dari komponen sistem yang lain. Kemudian modul *testing* harus menyusul dilakukan untuk mengetahui apakah penggabungan beberapa unit dalam satu modul sudah berjalan dengan baik, termasuk eksekusi dari beberapa modul yang saling berelasi, apakah sudah berjalan sesuai karakteristik sistem yang diinginkan.

Jika struktur kendali antar modul sudah terbukti bagus, maka pengujian yang tak kalah pentingnya adalah pengujian unit. Pengujian unit digunakan untuk menguji setiap modul untuk menjamin setiap modul menjalankan fungsinya dengan baik. Ada 2 metode untuk melakukan unit *testing*, yaitu :

1. *White Box Testing*

Uji coba *white box testing* merupakan metode perancangan *testcase* yang mengunakan struktural untuk mendapatkan *testcase*, test ini digunakan untuk meramal cara kerja perangkat lunak secara rinci kepada *logic path* (jalur logika), perangkat lunak di tes dengan kondisi dan perulangan secara fisik.

Contoh pengujian *white box testing* ini merupakan peringatan ketika *user* menginputkan *password user* yang salah, untuk kesalahan semacam ini akan memberikan suatu informasi kepada *user* mengenai kesalahan yang di lakukan.

2. *Black Box*

Berdasarkan hasil uji coba yang dilakukan, seluruh navigasi dan tombol fasilitas program lainnya serta proses yang di jalankan tidak terjadi kesalahan, tetapi aplikasi mempunyai aturan-aturan yang sudah di tetapkan dan harus di ikuti karena apabila di hiraukan maka sistem akan menolak perintah yang tidak sesuai seperti kesalahan ketika *user* belum menginput data yang harusnya di input sesuai ketentuan sistem yang di jalankan dan sistem memberikan informasi kepada *user* karena data yang ingin diproses belum lengkap atau tidak memenuhi ketentuan untuk proses selanjutnya.

G. Metode Pengujian

Metode pengujian yang dilakukan pada penelitian ini menggunakan *Blackbox* yang dipakai untuk mengetahui kinerja aplikasi pada saat melakukan proses *Tour Guide* dan mengetahui berfungsinya semua aplikasi yang telah dibuat, *blackbox* dalam metode pengujian perangkat lunak yang tes fungsionalitas dari aplikasi yang bertentangan dengan struktur internal atau kerja.

H. Desain Sistem

1. Sistem yang berjalan

Sistem yang berjalan saat ini adalah *Helpdesk*. Penulis memberi nama sistem yang berjalan saat ini *Centerdesk*, yaitu sistem infomasi layanan *helpdesk* konvensional. Setelah peneliti melakukan observasi terhadap *user* dalam menjalankan aplikasi pelayanan TI di universitas yang sudah berjalan dan digunakan saat ini oleh seluruh *civitas* perguruan tinggi, peneliti menganilis maksud, tujuan, dan sasaran sistem yang sedang berjalan untuk mengetahui alur aplikasi *helpdesk*.



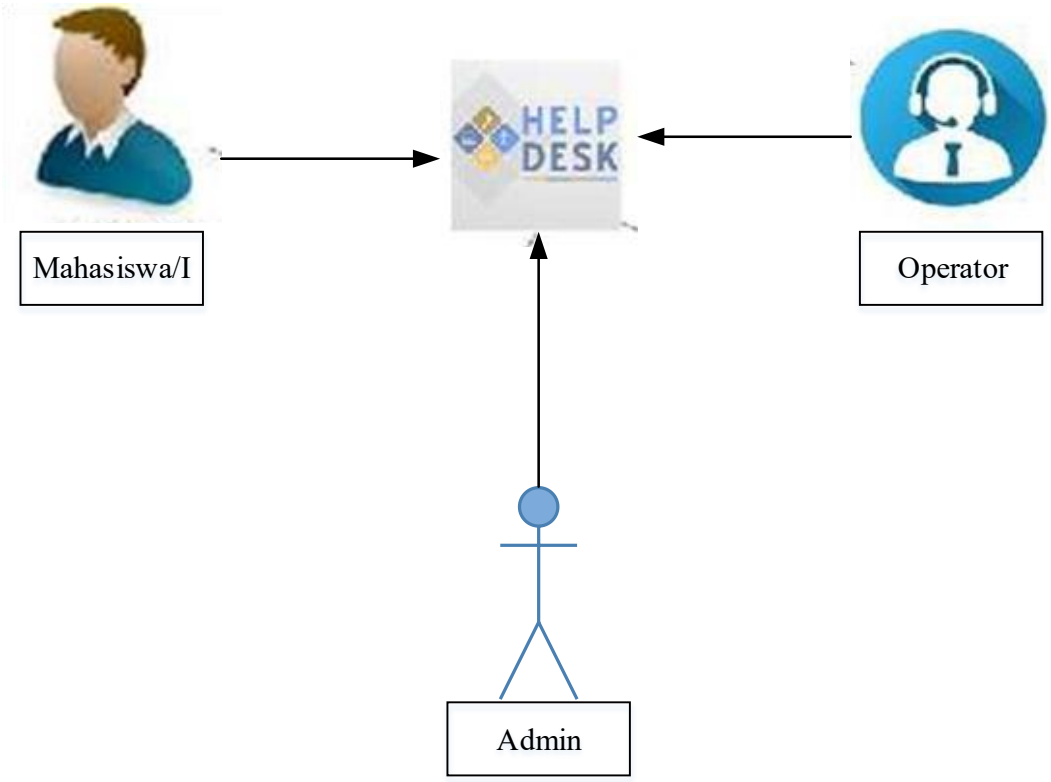
Gambar *3. 1 Sistem saat ini*

2. Sistem yang diusulkan

Setelah menentukan permasalahan maka selanjutnya menentukan apa yang harus diselesaikan dengan menggunakan sistem yang akan dibuat (usulan). Dalam sistem yang diusulkan, penulis memberi nama Sistem Informasi *FunDesk* (*FUN Helpdesk*). Di dalam Sistem Informasi *FunDesk* terdapat konsep *gamification* yang mana elemen – elemen *gamificationnya*

akan menarik minat dan keaktifan *user* agar menggunakan sistem secara optimal.

Dengan pendekatan ini, maka *user* tidak perlu datang ke pusat pelayanan terpadu mahasiswa untuk menyelesaikan permasalahan yang terkait akademik dikampus.



Gambar *3. 2 Sistem yang di usulkan*

BAB IV

PEMBAHASAN DAN HASIL

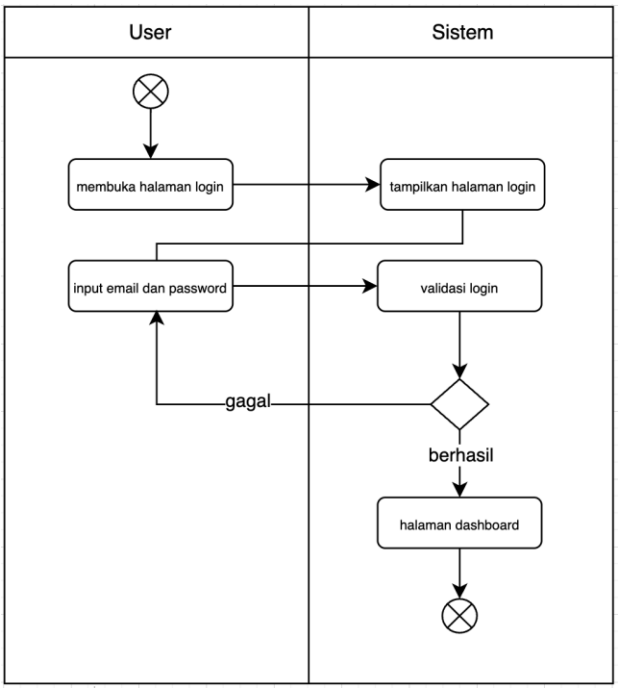
A. Analisis Aliran Data UML

Dalam bagian ini, akan dibahas dua diagram UML yang menggambarkan aliran data dan interaksi dalam aplikasi ini, yaitu *Activity diagram* dan *Sequence diagram*.

1. *Activity diagram*

   Activity diagram digunakan untuk memodelkan aliran kerja atau aliran kontrol dari sebuah proses bisnis atau *use case*. Diagram ini menunjukkan aktivitas-aktivitas yang terjadi dalam sistem dan urutan eksekusinya.

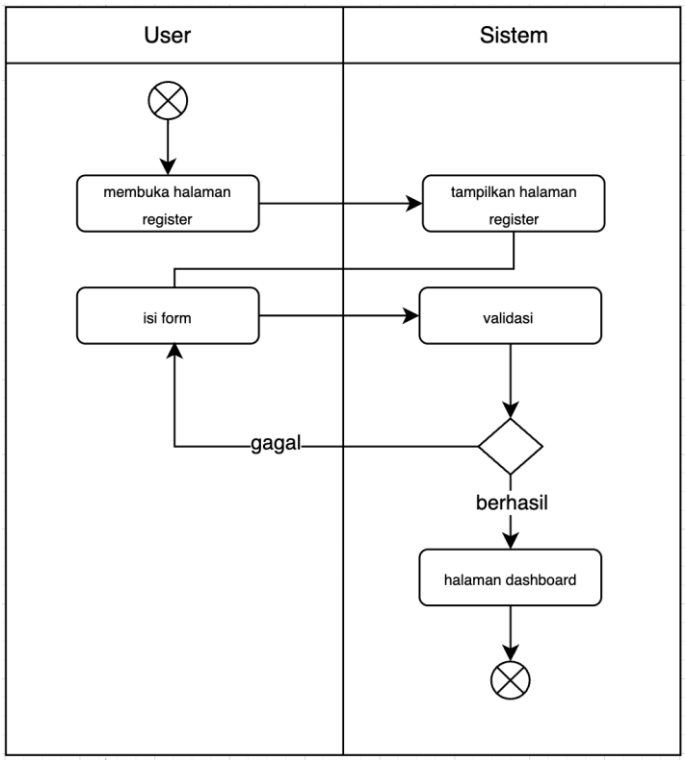
   a. *Activity diagram login*



Gambar *4. 1 Activity diagram login*

Pada gambar 4. 1 menjelaskan cara *login*. *user* harus terlebih dahulu mengakses situs web, setelah itu sistem akan menampilkan *form login* dan *user* harus memasukkan email dan kata sandi. Sistem kemudian akan melakukan validasi; jika informasi benar, maka akan masuk ke halaman *dashboard*; jika tidak, maka akan kembali ke halaman *login*.

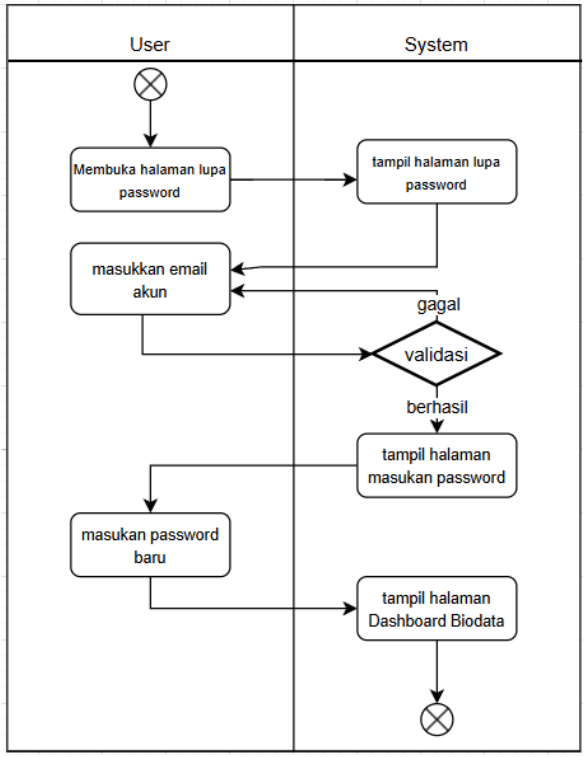
b. *Activity diagram register*



Gambar *4. 2 Activity diagram register*

Pada gambar 4. 2 menjelaskan cara *register*. *user* harus terlebih dahulu mengakses situs web, setelah itu sistem akan menampilkan *form register* dan *user* harus mengisi *form* nya untuk memasuki halaman *dashboard*.

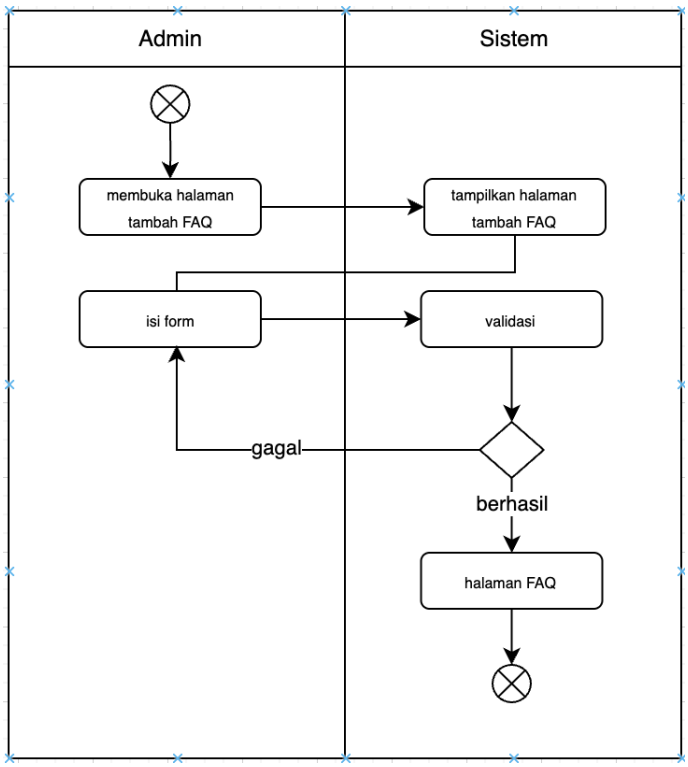
c. *Activity diagram* lupa *password*



Gambar *4. 3 Activity diagram lupa password*

Activity diagram ini menggambarkan proses lupa *password*: pengguna membuka halaman, memasukkan email, divalidasi oleh sistem, lalu mengatur *password* baru dan diarahkan ke halaman *dashboard* biodata.

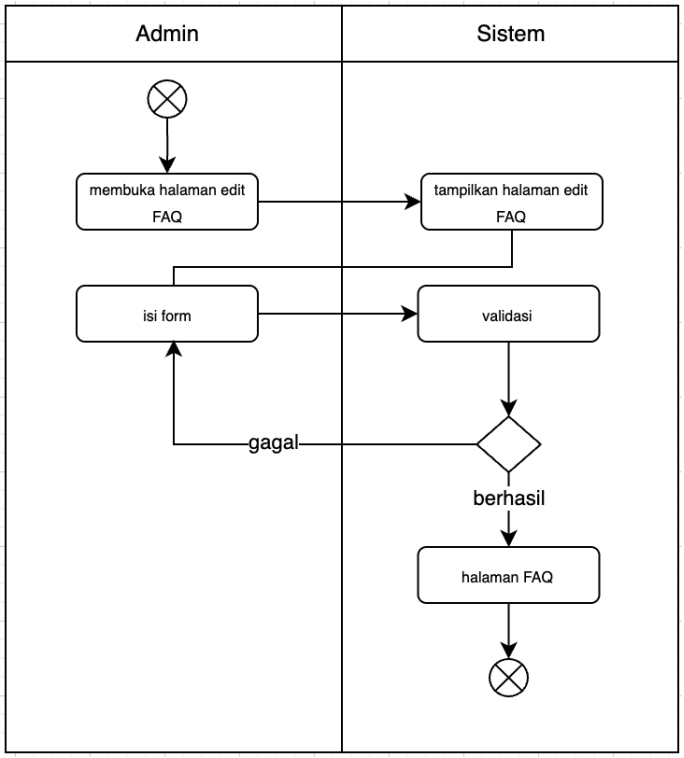
d. *Activity diagram* tambah FAQ



Gambar *4. 4 Activity diagram tambah FAQ*

Pada gambar 4. 4 menjelaskan langkah-langkah yang digunakan oleh *Admin* untuk menambahkan data. Halaman *Admin* ditampilkan oleh sistem ketika telah dibuka oleh *User*. *Admin* kemudian memilih item menu tambah data, setelah itu sistem menampilkan halaman menu yang dipilih admin dan admin memilih tambah data. *Form* tambah data kemudian akan ditampilkan oleh sistem. *Admin* selanjutnya diminta untuk melengkapi *form* tambah data. Setelah selesai, sistem akan mengecek data dan jika berhasil, sistem akan menampilkan halaman menu untuk data yang ditambahkan. Jika tidak berhasil, administrator harus mengisi formulir penambahan data dengan benar sekali lagi.

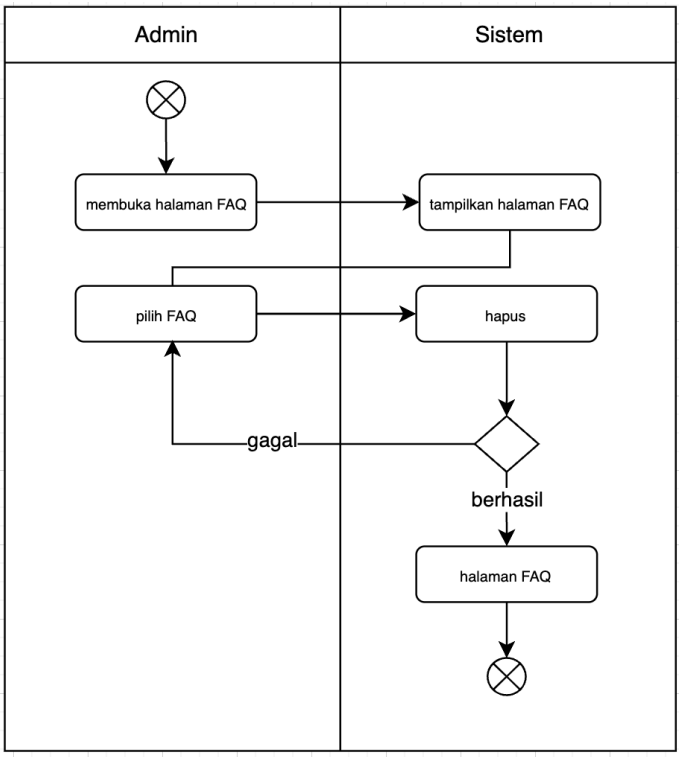
e. *Activity diagram Edit* FAQ



Gambar *4. 5 Activity diagram edit FAQ*

Pada gambar 4. 5 menjelaskan langkah-langkah yang harus dilakukan oleh *admin* untuk memodifikasi data. Hal pertama yang dilakukan *admin* adalah membuat folder *admin*. Kemudian, sistem akan menampilkan menu *admin*. Selanjutnya, *admin* memilih menu yang akan diubah datanya; sistem akan menampilkan judul menu yang dipilih *admin*; terakhir, *admin* mengubah data. Sistem akan menampilkan data dalam bentuk tabel. Setelah *admin* mengisi *form* data, sistem akan memvalidasi informasi tersebut. Jika validasi berhasil, sistem akan menampilkan item menu dengan data yang dimasukkan, dan jika tidak, administrator disarankan untuk mengisi *form* data kembali dengan akurat.

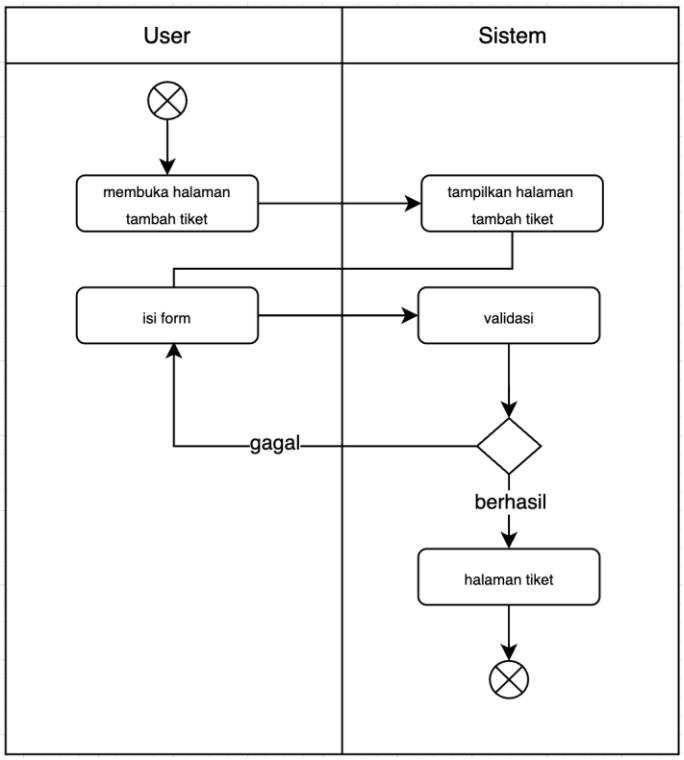
f. *Activity diagram* Hapus FAQ



Gambar *4. 6 Activity diagram hapus FAQ*

Pada gambar 4. 6 menjelaskan prosedur yang digunakan *admin* untuk menghapus data. Halaman *admin* ditampilkan oleh sistem ketika telah dibuka oleh *admin*. *Admin* kemudian memilih menu yang datanya akan dimusnahkan, sistem menampilkan halaman menu yang telah dipilih oleh *admin*, dan *admin* mengklik pilihan hapus data. Sistem kemudian akan menampilkan konfirmasi penghapusan. *Admin* kemudian melakukan konfirmasi. Jika admin konfirmasi, data akan terhapus, jika tidak maka data batal dihapus.

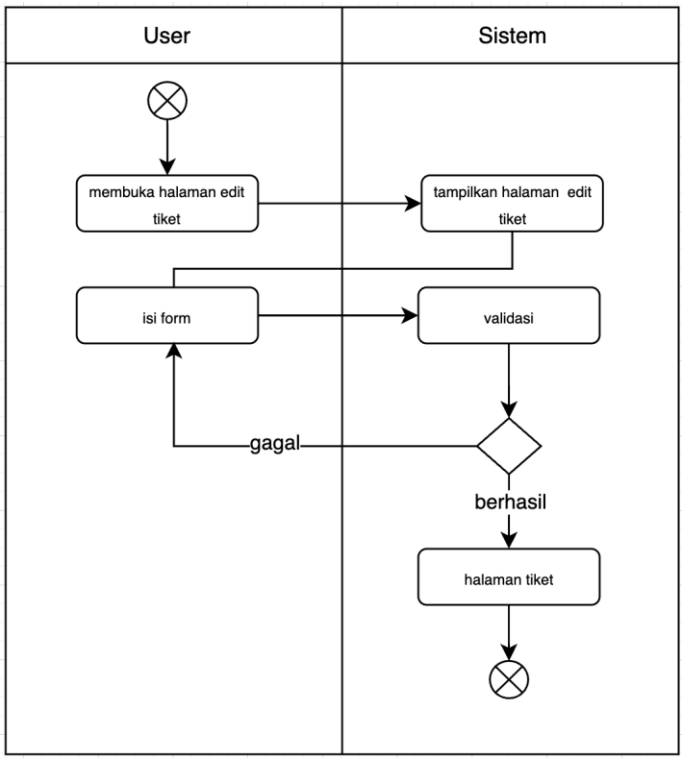
g. *Activity diagram* tambah tiket



Gambar *4. 7 Activity diagram tambah tiket*

Pada gambar 4. 7 menjelaskan langkah-langkah yang digunakan oleh *User* untuk menambahkan data. Halaman *User* ditampilkan oleh sistem ketika telah dibuka oleh *User*. *User* kemudian memilih item menu tambah data, setelah itu sistem menampilkan halaman menu yang dipilih *User* dan *User* memilih tambah data. *Form* tambah data kemudian akan ditampilkan oleh sistem. *User* selanjutnya diminta untuk melengkapi *form* tambah data. Setelah selesai, sistem akan mengecek data dan jika berhasil, sistem akan menampilkan halaman menu untuk data yang ditambahkan. Jika tidak berhasil, Useristrator harus mengisi formulir penambahan data dengan benar sekali lagi.

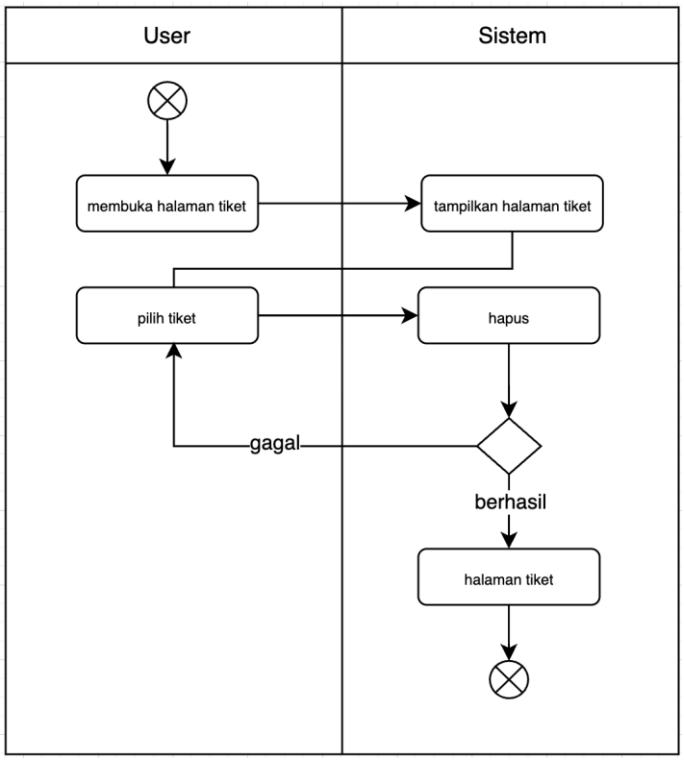
h. *Activity diagram edit* tiket



Gambar *4. 8 Activity diagram edit tiket*

Pada gambar 4. 8 menjelaskan langkah-langkah yang harus dilakukan oleh *User* untuk memodifikasi data. Hal pertama yang dilakukan *User* adalah membuat folder *User*. Kemudian, sistem akan menampilkan menu *User*. Selanjutnya, *User* memilih menu yang akan diubah datanya; sistem akan menampilkan judul menu yang dipilih *User*; terakhir, *User* mengubah data. Sistem akan menampilkan data dalam bentuk tabel. Setelah *User* mengisi *form* data, sistem akan memvalidasi informasi tersebut. Jika validasi berhasil, sistem akan menampilkan item menu dengan data yang dimasukkan, dan jika tidak, *User*istrator disarankan untuk mengisi *form* data kembali dengan akurat.

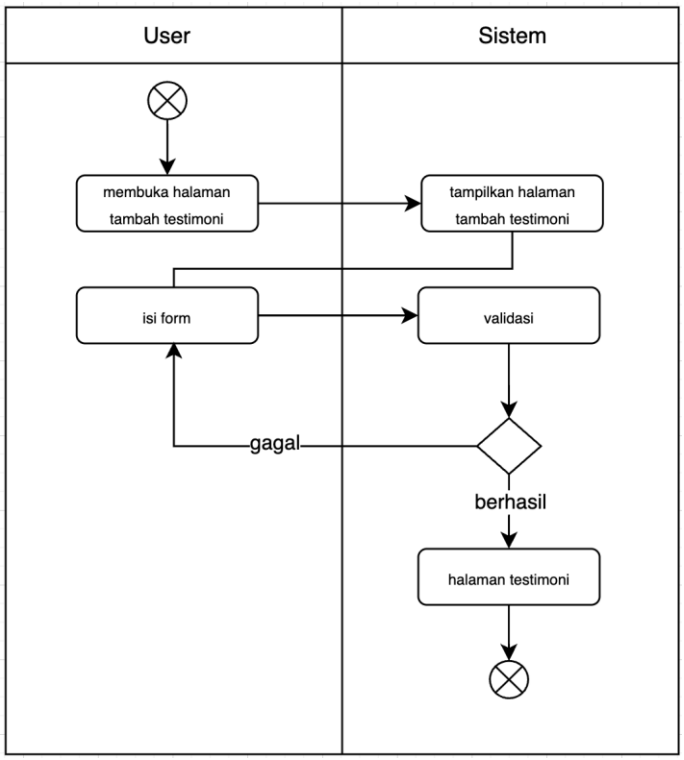
i. *Activity diagram* hapus tiket



Gambar *4. 9 Activity diagram hapus tiket*

Pada gambar 4. 9 menjelaskan prosedur yang digunakan *User* untuk menghapus data. Halaman *User* ditampilkan oleh sistem ketika telah dibuka oleh *User*. *User* kemudian memilih menu yang datanya akan dimusnahkan, sistem menampilkan halaman menu yang telah dipilih oleh *User*, dan *User* mengklik pilihan hapus data. Sistem kemudian akan menampilkan konfirmasi penghapusan. *User* kemudian melakukan konfirmasi. Jika *User* konfirmasi, data akan terhapus, jika tidak maka data batal dihapus.

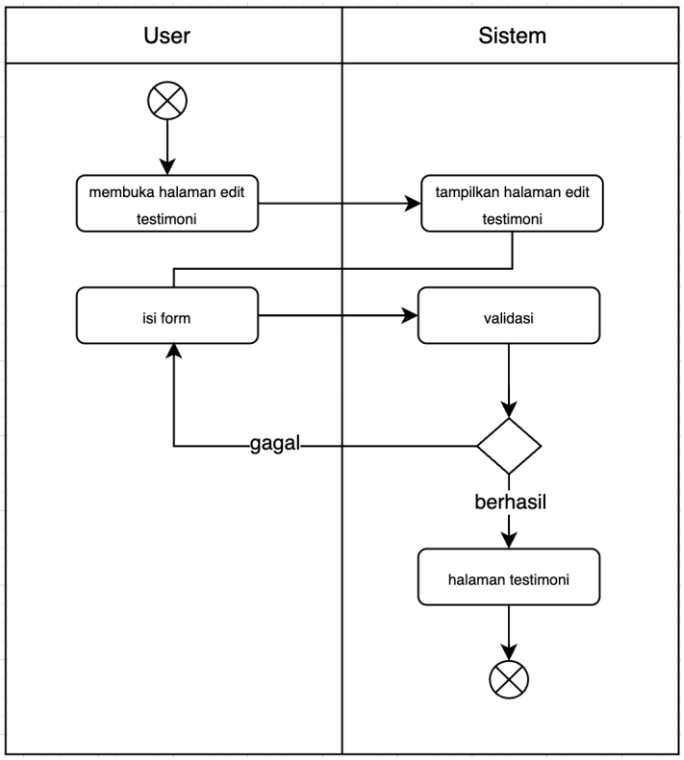
j. *Activity diagram* tambah testimoni



Gambar *4. 10 tambah testimoni*

Pada gambar 4. 10 menjelaskan langkah-langkah yang digunakan oleh *User* untuk menambahkan data. Halaman *User* ditampilkan oleh sistem ketika telah dibuka oleh *User*. *User* kemudian memilih item menu tambah data, setelah itu sistem menampilkan halaman menu yang dipilih *User* dan *User* memilih tambah data. *Form* tambah data kemudian akan ditampilkan oleh sistem. *User* selanjutnya diminta untuk melengkapi *form* tambah data. Setelah selesai, sistem akan mengecek data dan jika berhasil, sistem akan menampilkan halaman menu untuk data yang ditambahkan. Jika tidak berhasil, Useristrator harus mengisi formulir penambahan data dengan benar sekali lagi.

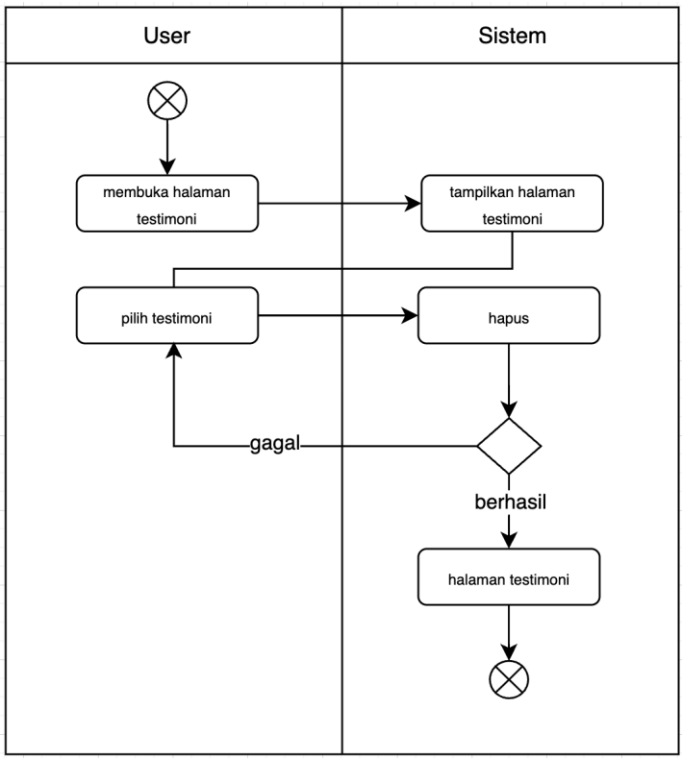
k. *Activity diagram edit* testimoni



Gambar *4. 11 edit testimoni*

Pada gambar 4. 11 menjelaskan langkah-langkah yang harus dilakukan oleh *User* untuk memodifikasi data. Hal pertama yang dilakukan *User* adalah membuat folder *User*. Kemudian, sistem akan menampilkan menu *User*. Selanjutnya, *User* memilih menu yang akan diubah datanya; sistem akan menampilkan judul menu yang dipilih *User*; terakhir, *User* mengubah data. Sistem akan menampilkan data dalam bentuk tabel. Setelah *User* mengisi *form* data, sistem akan memvalidasi informasi tersebut. Jika validasi berhasil, sistem akan menampilkan item menu dengan data yang dimasukkan, dan jika tidak, Useristrator disarankan untuk mengisi *form* data kembali dengan akurat.

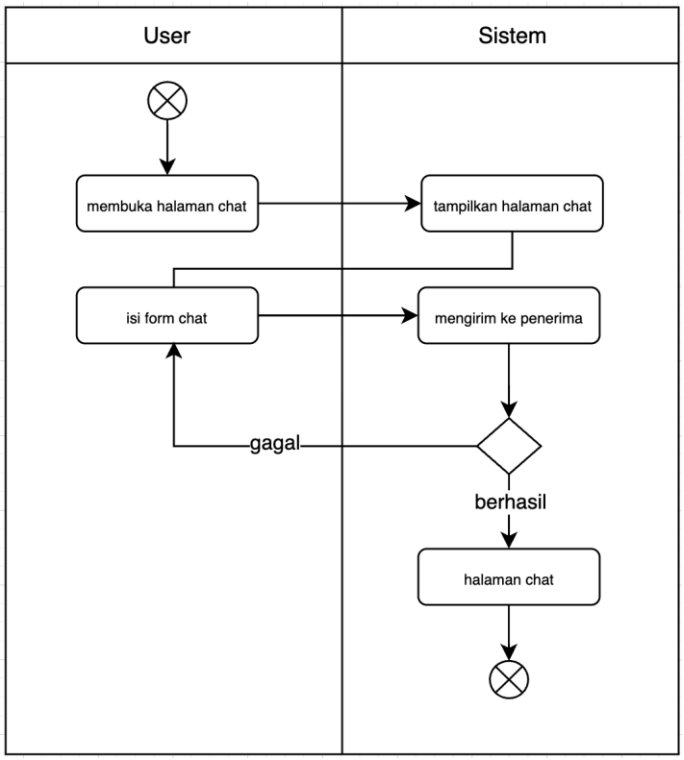
l. *Activity diagram* hapus testimoni



Gambar *4. 12 Activity diagram hapus testimoni*

Pada gambar 4. 12 menjelaskan prosedur yang digunakan *User* untuk menghapus data. Halaman *User* ditampilkan oleh sistem ketika telah dibuka oleh *User*. *User* kemudian memilih menu yang datanya akan dimusnahkan, sistem menampilkan halaman menu yang telah dipilih oleh *User*, dan *User* mengklik pilihan hapus data. Sistem kemudian akan menampilkan konfirmasi penghapusan. *User* kemudian melakukan konfirmasi. Jika *User* konfirmasi, data akan terhapus, jika tidak maka data batal dihapus.

m. *Activity diagram chat*

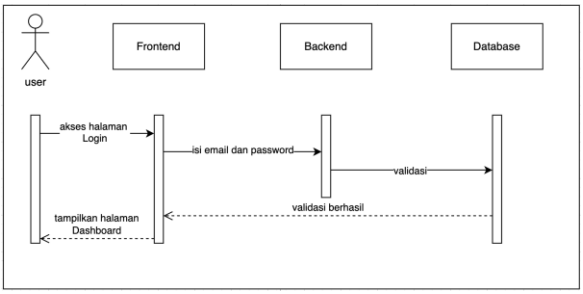


Gambar *4. 13 Activity diagram chat*

Pada gambar 4. 13 menjelaskan prosedur yang digunakan *User* untuk menge*chat* operator atau dalam hal ini konsultan.cara kerjanya sama seperti *social* media yaitu *user* dan operator saling mengirim *chat*.

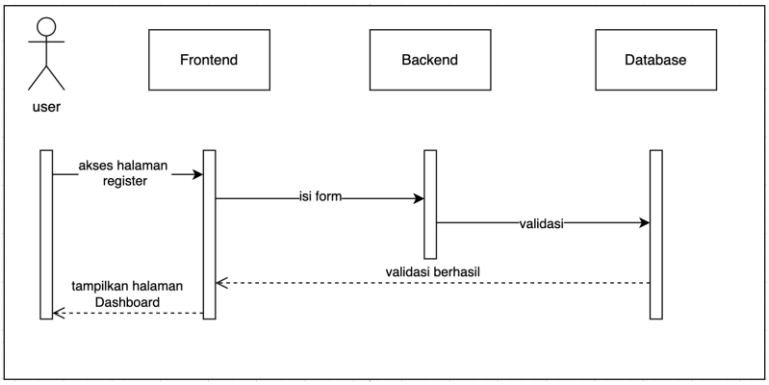
2. *Sequence diagram*

a. *Sequence diagram login*



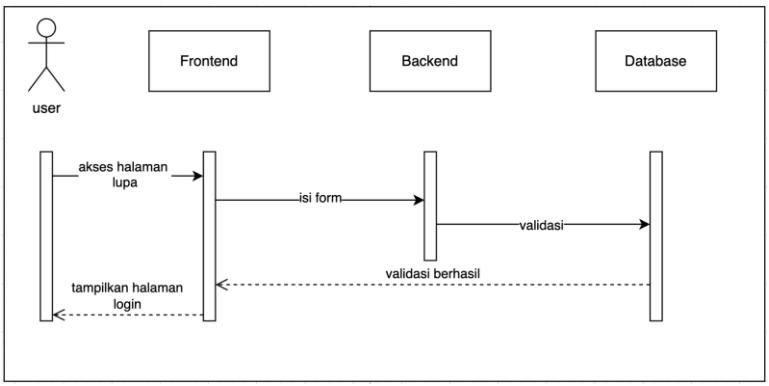
Gambar *4. 14 Sequence diagram login*

b. *Sequence diagram register*



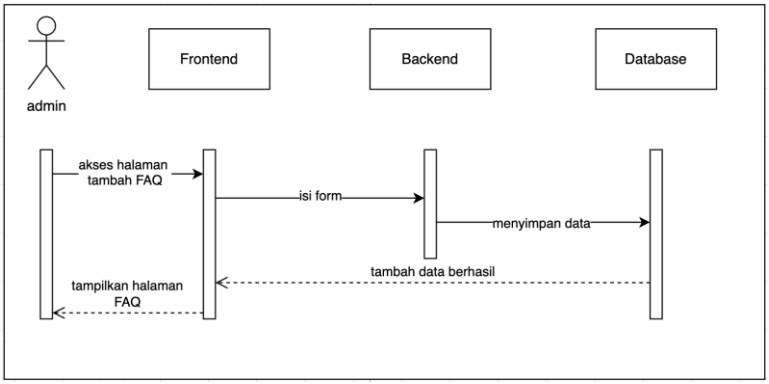
Gambar *4. 15 Sequence diagram register*

c. *Sequence diagram* lupa *password*



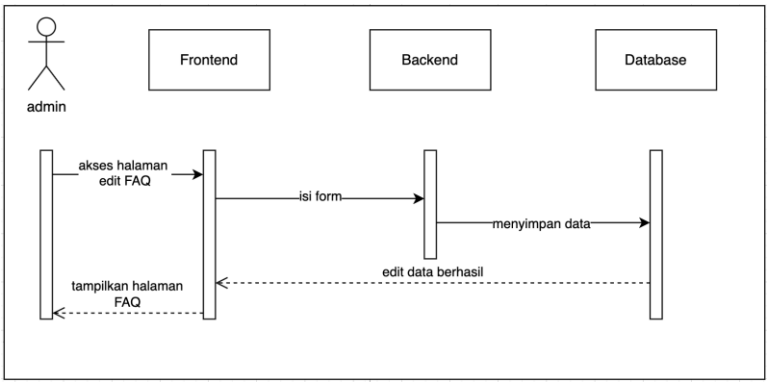
Gambar *4. 16 Sequence diagram lupa password*

d. *Sequence diagram* tambah FAQ



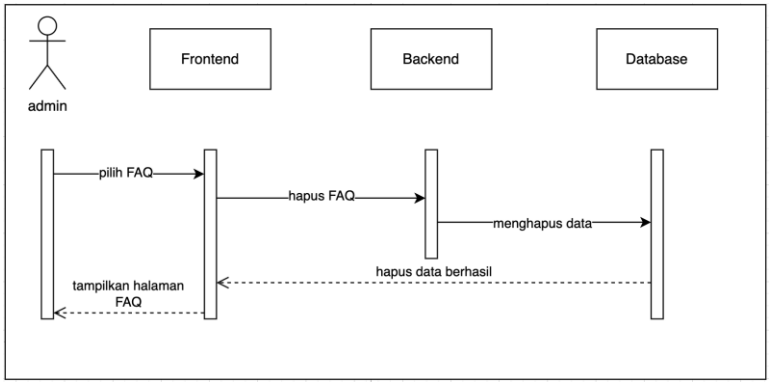
Gambar *4. 17 Sequence diagram tambah FAQ*

e. *Sequence diagram edit* FAQ



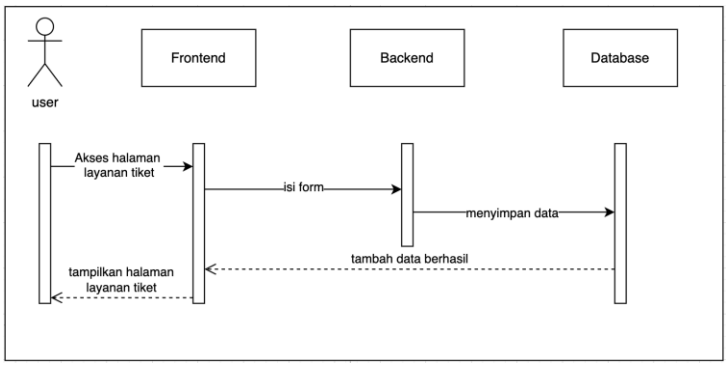
Gambar *4. 18 Sequence diagram edit FAQ*

f. *Sequence diagram* Hapus FAQ



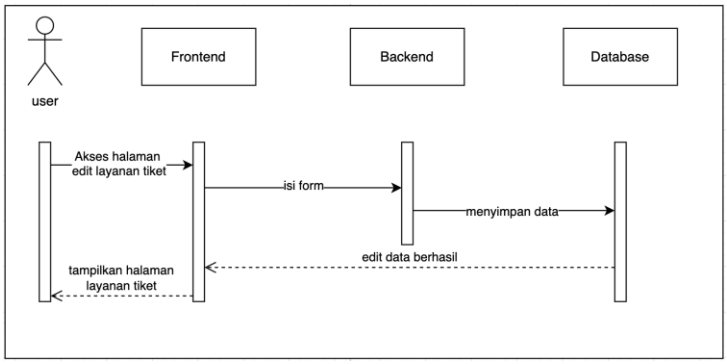
Gambar *4. 19 Sequence diagram hapus FAQ*

g. *Sequence diagram* tambah tiket



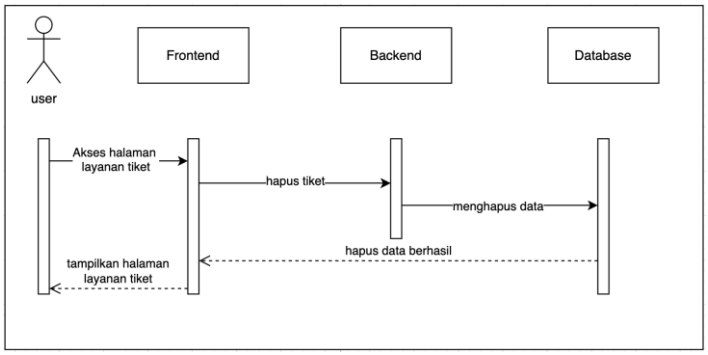
Gambar *4. 20 Sequence diagram tambah tiket*

h. *Sequence diagram edit* tiket



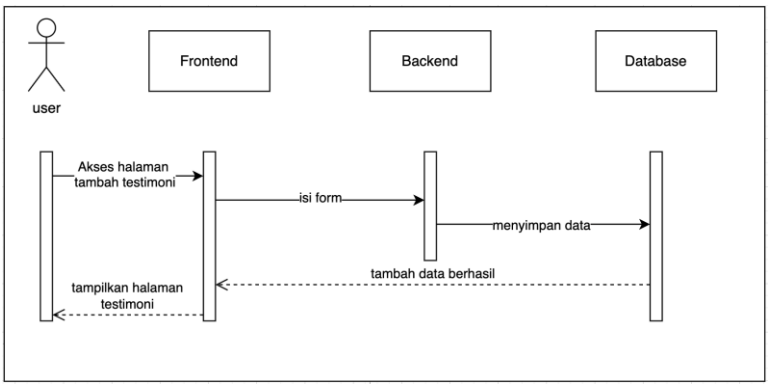
Gambar *4. 21 Sequence diagram edit tiket*

i. *Sequence diagram* hapus tiket



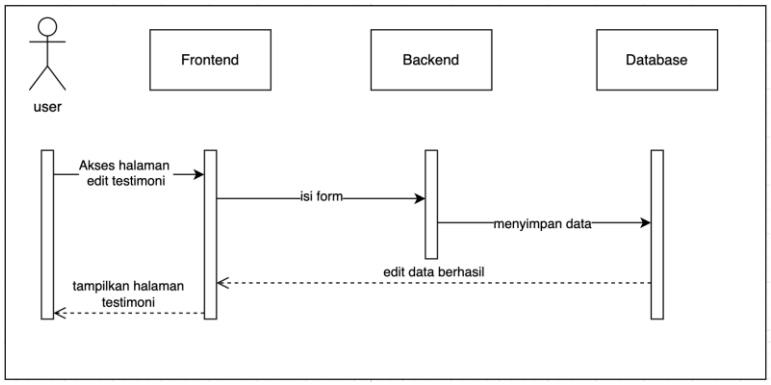
Gambar *4. 22 Sequence diagram hapus tiket*

j. *Sequence diagram* Tambah testimoni



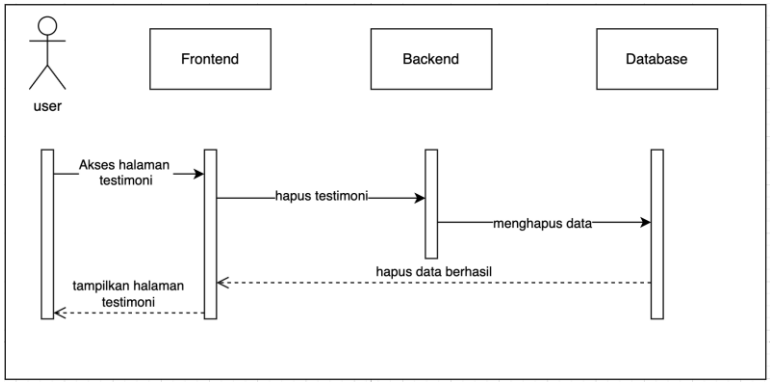
Gambar *4. 23 Sequence diagram tambah testimoni*

k. *Sequence diagram edit* tesminoni



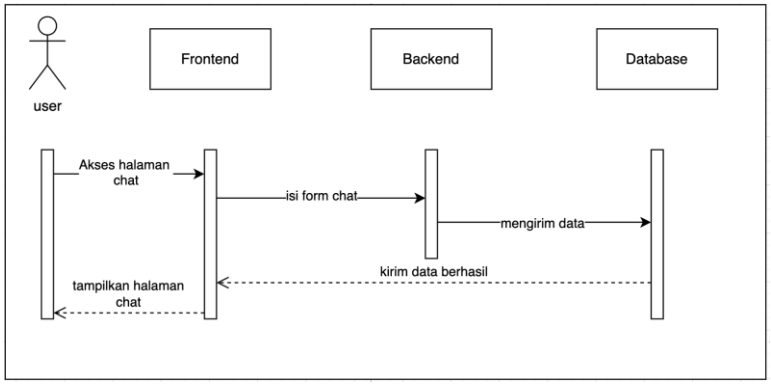
Gambar *4. 24 Sequence diagram edit testimoni*

l. *Sequence diagram* hapus testimoni



Gambar *4. 25 Sequence diagram hapus testimoni*

m. *Sequence diagram chat*



Gambar *4. 26 Sequence diagram chat*

B. Detail sistem

1. Halaman Awal

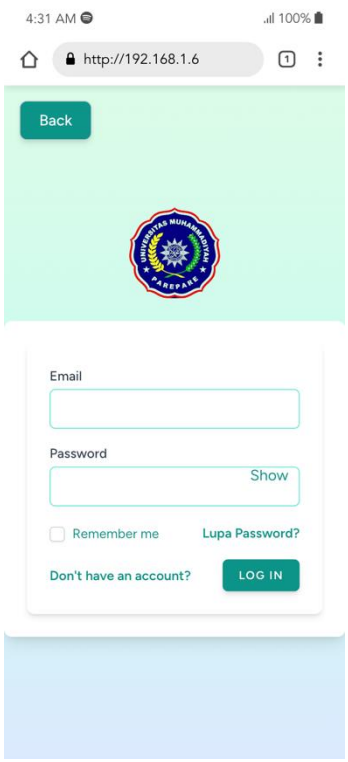
Halaman Awal merupakan halaman yang tampil saat pertama kali membuka aplikasi.



Gambar *4. 27 Halaman awal*

2. Halaman *Login*

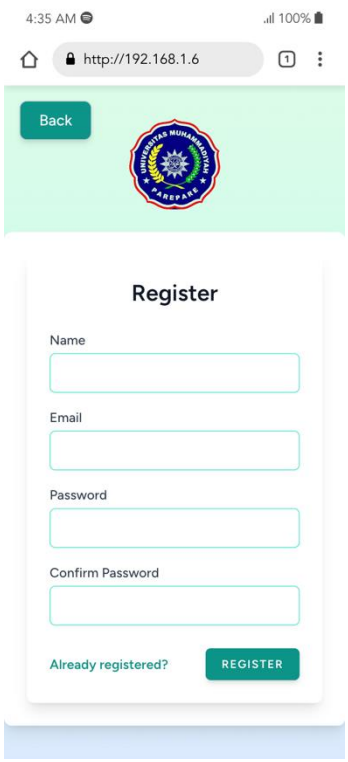
Halaman *login* merupakan halaman utama yang digunakan oleh *user,operator dan admin* untuk mendapatkan akses ke halaman *dashboard*.



Gambar *4. 28 Halaman login*

3. Halaman *register*

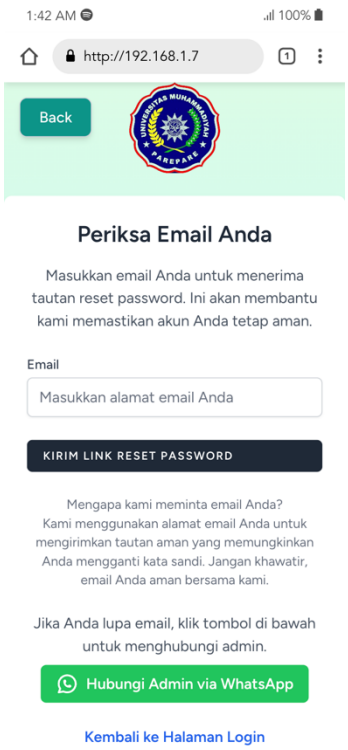
Halaman *register* merupakan halaman utama yang digunakan oleh *user dan operator* untuk membuat akun baru



Gambar *4. 29 Halaman register*

4. Halaman lupa *password*

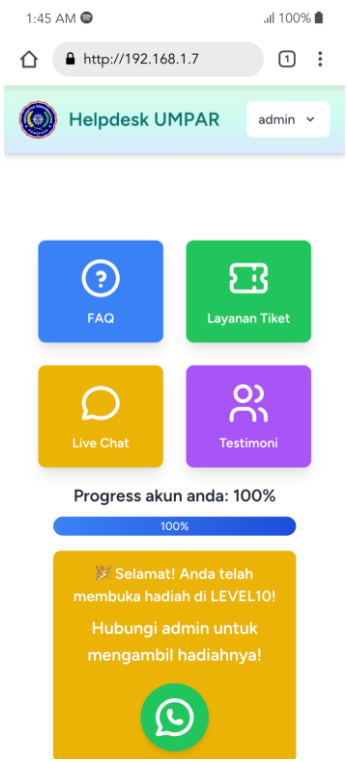
Halaman lupa *password* adalah halaman yg di gunakan untuk *user* mengganti *password*nya apabila *user* lupa dengan *password*nya saat *login*.



Gambar *4. 30 Halaman Lupa password*

5. Halaman *dashboard*

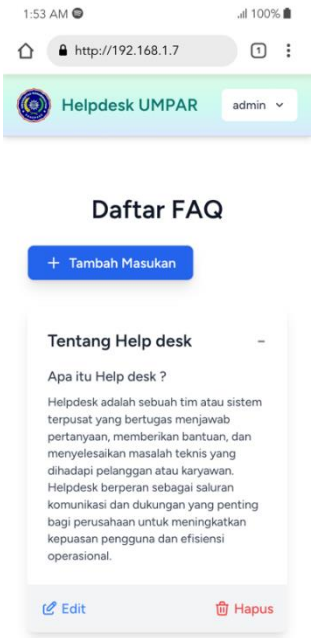
Halaman *Dashboard* merupakan halaman utama yang di akses setelah *login* berhasil.di halaman ini *kit*a bisa mengakses semua fitur sesuai dengan role yang *user* punya.



Gambar *4. 31 Halaman Dashboard*

6. Halaman FAQ

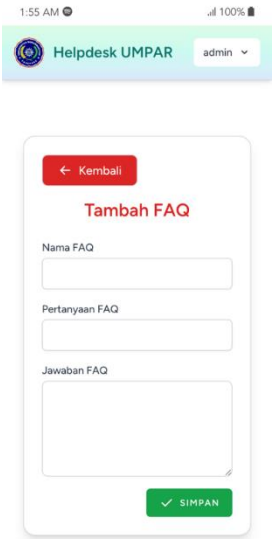
Halaman FAQ merupakan halaman yang berisi pertanyaan umum dan pertanyaan yang sering di tanyakan. Di halaman ini terdapat berbagai pertanyaan umum beserta dengan jawabannya.



Gambar *4. 32 halaman FAQ*

7. Halaman Tambah FAQ

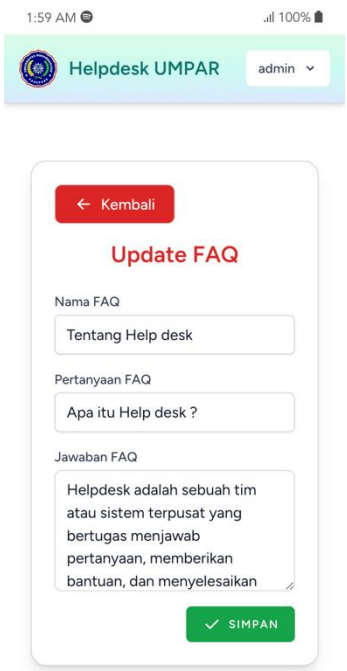
Halaman ini merupakan halaman untuk admin menambahkan pertanyaan umum .



Gambar *4. 33 Halaman Tambah FAQ*

8. Halaman *Edit* FAQ

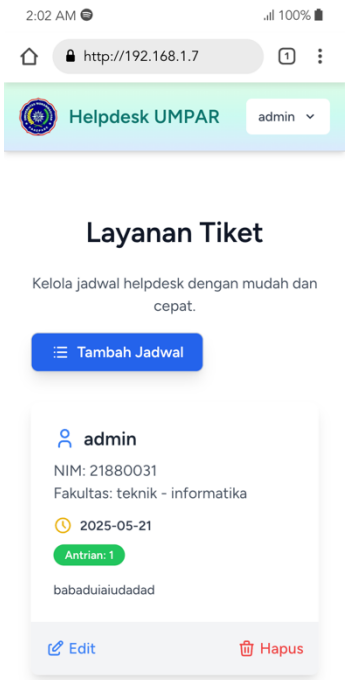
Halaman ini merupakan halaman untuk admin mengedit pertanyaan umum yang terdapat kesalahan penulisan.



Gambar *4. 34 Halaman Edit FAQ*

9. Halaman Layanan tiket

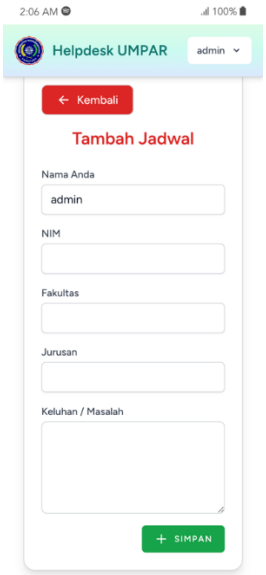
Halaman layanan tiket merupakan halaman untuk *user* mengambil nomor antrian jika *user* ingin melakukan konsultasi secara langsung.



Gambar *4. 35 Halaman Layanan tiket*

10. Halaman tambah antrian

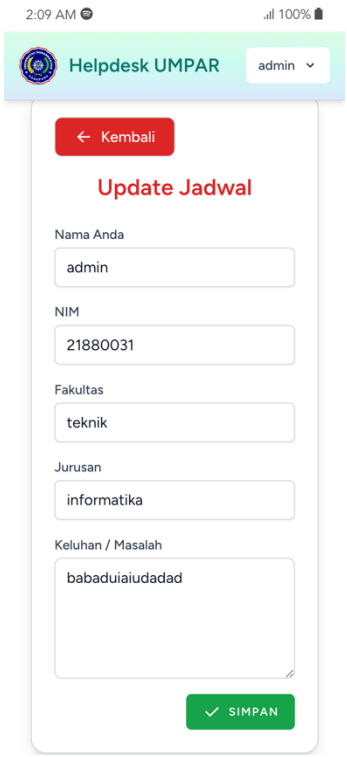
Halaman ini digunakan untuk *user* mengambil nomor antrian.



Gambar *4. 36 Halaman tambah tiket*

11. Halaman *edit* antrian

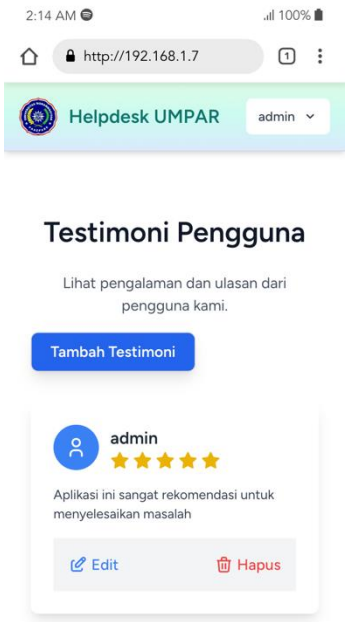
Halaman ini digunakan untuk meng*edit* data nomor antrian apabila terdapat kesalahan penulisan.



Gambar *4. 37 Halaman edit tiket*

12. Halaman testimoni

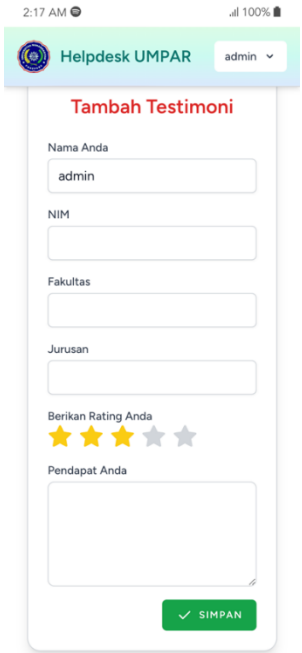
Halaman ini merupakan halaman ini untuk *user* memberikan rating beserta dengan komentarnya terhadap pelayanan dari *Helpdesk* ini.



Gambar *4. 38 Halaman testimoni*

13. Halaman tambah testimoni

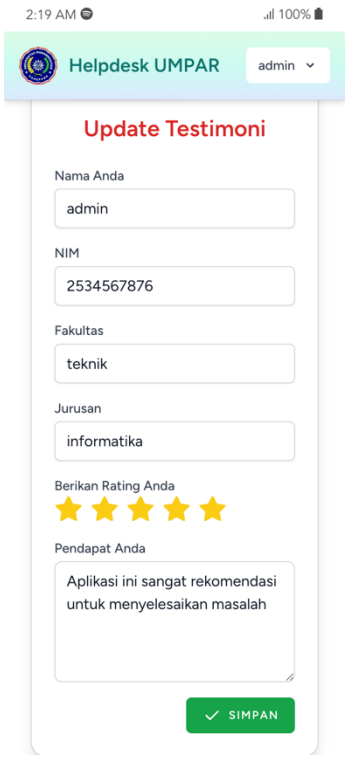
Halaman ini berfungsi untuk menambahkan komentar serta rating dari pelayanan *Helpdesk*.



Gambar *4. 39 Halaman tambah testimoni*

14. Halaman *edit* testimoni

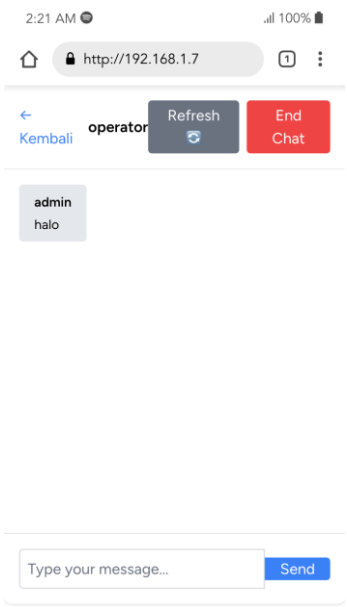
Halaman ini digunakan untuk meng*edit* data testimoni apabila terdapat kesalahan penulisan.



Gambar *4. 40 Halaman edit testimoni*

15. Halaman *Chatting*

Halaman ini berfungsi untuk *user* melakukan konsultasi secara *online*



Gambar *4. 41 Halaman chatting*

16. Halaman admin

Halaman ini berfungsi untuk admin memanajemen *user*nya.



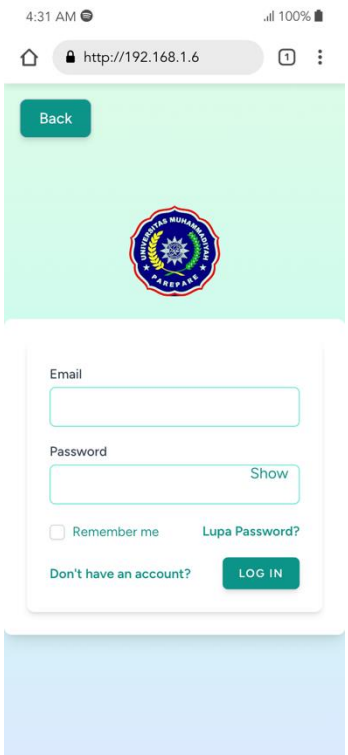
Gambar *4. 42 Halaman admin*

C. Pengujian sistem

Pengujian sistem bertujuan untuk mengidentifikasi kesalahan, kesenjangan, atau kekurangan dalam sistem sebelum digunakan oleh pengguna akhir. Ada dua pendekatan utama dalam pengujian sistem ini, yaitu pengujian *Black box* dan pengujian *white box*.

1. *Blackbox*

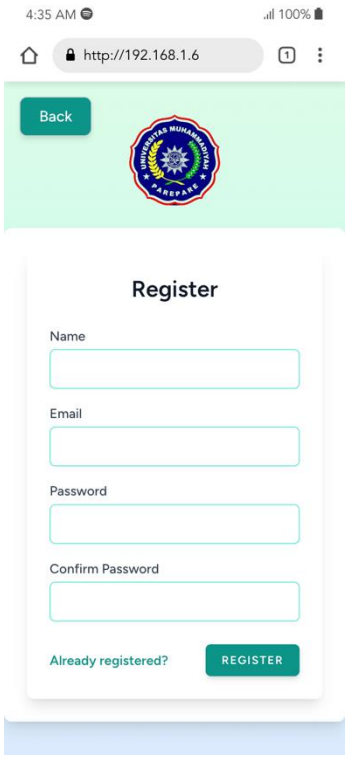
   a. *Blackbox testing login*

Tes faktor	Hasil	Keterangan
Memasukkan *email* atau *password* yang benar	✔	Sistem berhasil menampilkan halaman *dashboard*.
Screenshot		
		

Tabel 4. 1 *Blackbox testing login*

b. *Blackbox testing register*

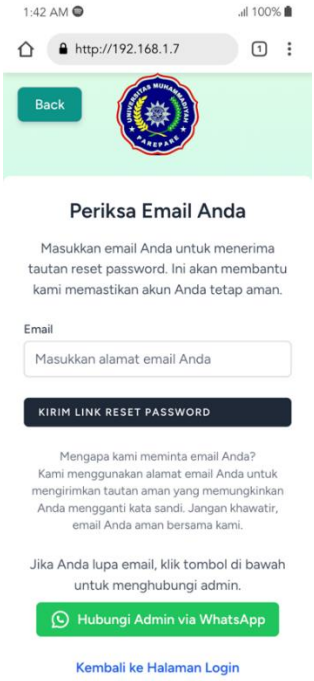
Tes faktor	Hasil	Keterangan
Memasukkan *form* yang benar	✔	Sistem berhasil menampilkan halaman *dashboard*.
Screenshot		



Tabel 4. 2 *Blackbox testing register*

c. *Blackbox testing* lupa *password*

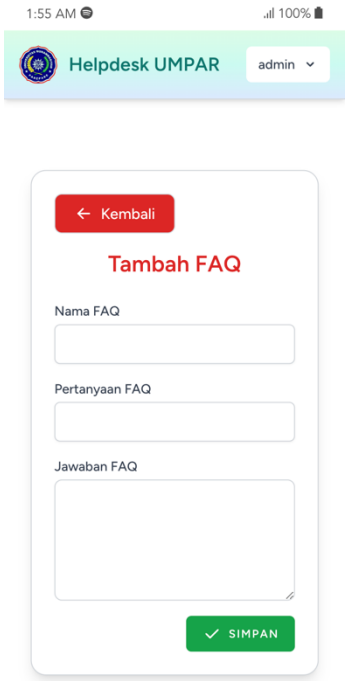
Tes faktor	Hasil	Keterangan
Mengisi *form.*	✔	Sistem berhasil mengganti *password*.
Screenshot		



Tabel 4. 3 *Blackbox testing* lupa *password*

d. *Blackbox testing* tambah FAQ

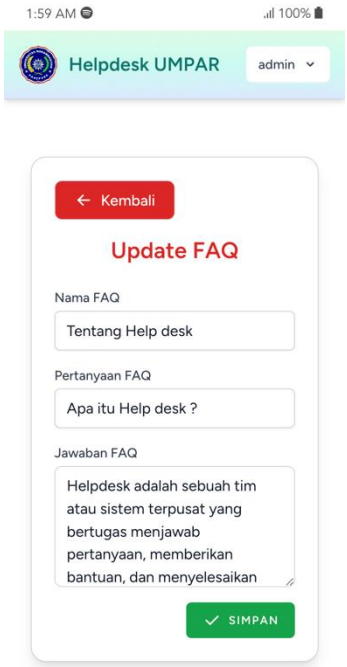
Tes faktor	Hasil	Keterangan
Mengisi *form.*	✔	Sistem berhasil menambah FAQ.
Screenshot		



Tabel 4. 4 *Blackbox testing* tambah FAQ

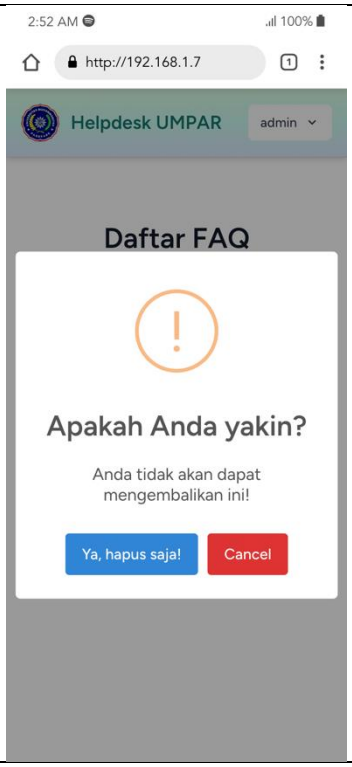
e. *Blackbox testing edit* FAQ

Tes faktor	Hasil	Keterangan
Mengisi *form.*	✔	Sistem berhasil meng*edit* FAQ.
Screenshot		



Tabel 4. 5 *Blackbox testing edit* FAQ

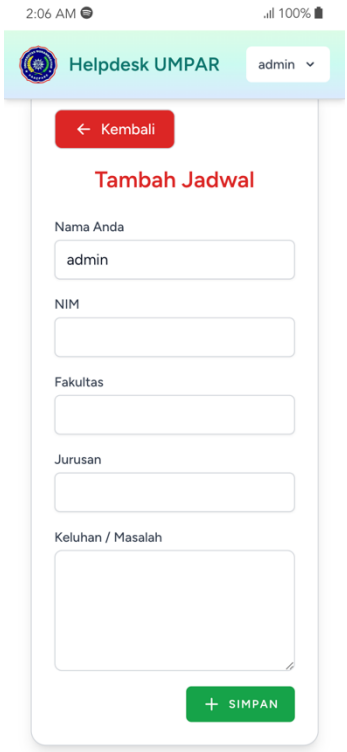
f. *Blackbox testing* hapus FAQ

Tes faktor	Hasil	Keterangan
Memilih data yang ingin dihapus.	✔	Sistem berhasil menhapus FAQ.
Screenshot		
		

Tabel 4. 6 Blackbox *testing* hapus FAQ

g. *Blackbox testing* tambah tiket

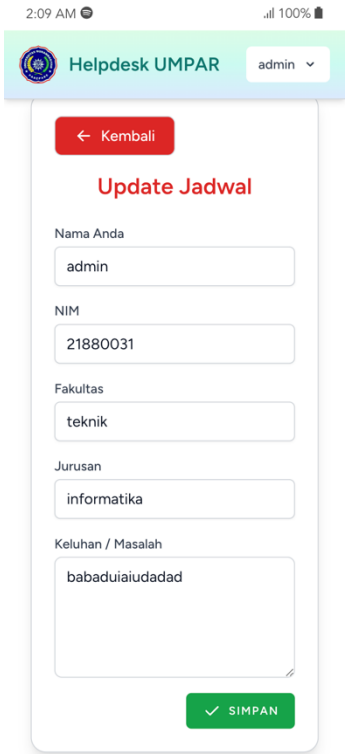
Tes faktor	Hasil	Keterangan
Mengisi *form.*	✔	Sistem berhasil menambah tiket.
Screenshot		



Tabel 4. 7 *Blackbox testing* tambah tiket

h. *Blackbox testing edit* tiket

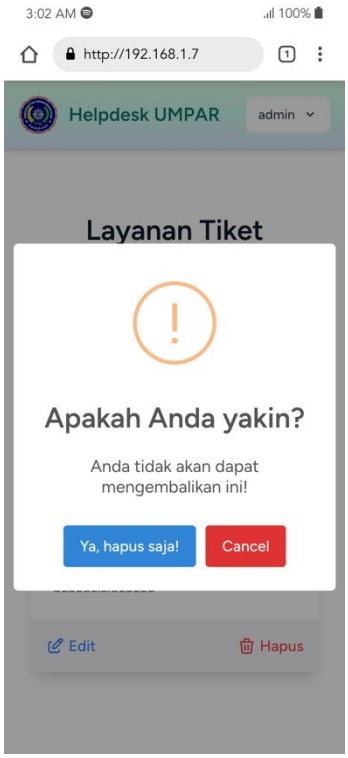
Tes faktor	Hasil	Keterangan
Mengisi *form.*	✔	Sistem berhasil meng*edit* tiket.
Screenshot		



Tabel 4. 8 *Blackbox testing edit* tiket

i. *Blackbox testing* hapus tiket

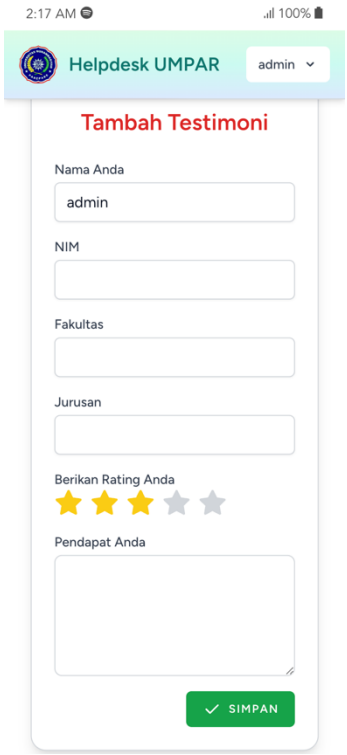
Tes faktor	Hasil	Keterangan
Memilih data yang ingin di hapus.	✔	Sistem berhasil menghapus tiket.
Screenshot		



Tabel 4. 9 *Blackbox testing* Hapus tiket

j. *Blackbox testing* tambah testimoni

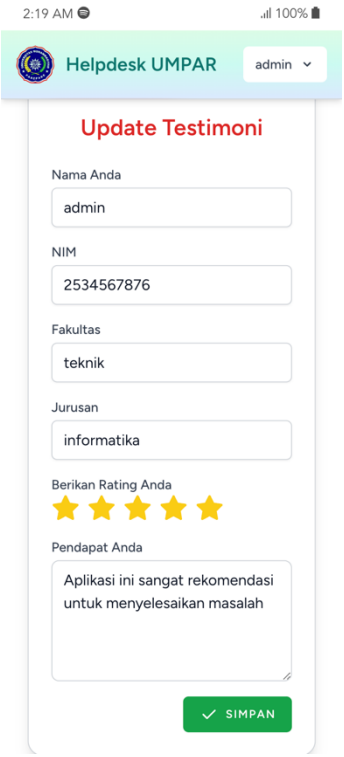
Tes faktor	Hasil	Keterangan
Mengisi *form.*	✔	Sistem berhasil menambah testimoni.
Screenshot		



Tabel 4. 10 *Blackbox testing* tambah testimoni

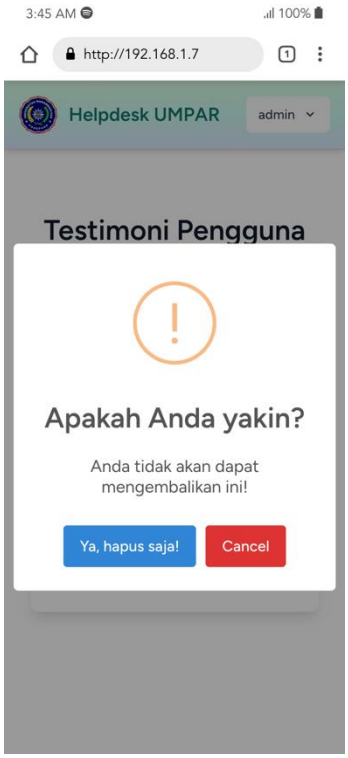
k. *Blackbox testing*

Tes faktor	Hasil	Keterangan
Mengisi *form.*	✔	Sistem berhasil meng*edit* testimoni.
Screenshot		



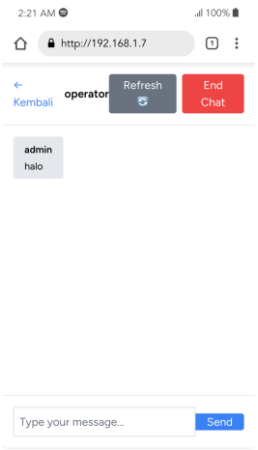
Tabel 4. 11 *Blackbox testing edit* testimoni

l. *Blackbox testing* hapus testimoni

Tes faktor	Hasil	Keterangan
Memilih data yang ingin di hapus.	✔	Sistem berhasil menghapus testimoni.
Screenshot		
		

Tabel 4. 12 *Blackbox testing* hapus testimoni

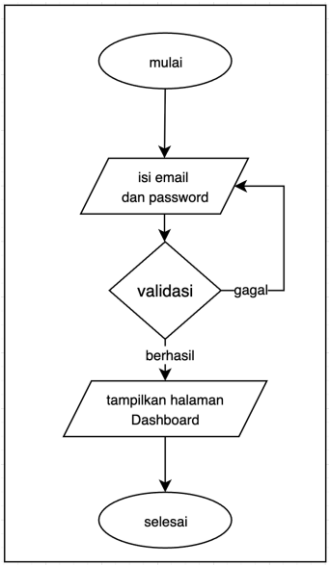
m. *Blackbox testing chat*

Tes faktor	Hasil	Keterangan
Mengisi dan mengirim pesan.	✔	Sistem berhasil mengirim pesan.
Screenshot		
		

2. *Whitebox*

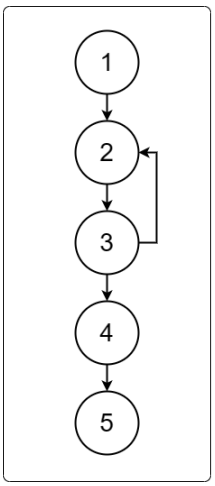
a. *Whitebox testing login*

1) *Flowchart*



Gambar 4. 43 *Flowchart login*

2) *Flowgraph*



Gambar 4. 44 *Flowgraph login*

Berdasarkan gambar 4. 44 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

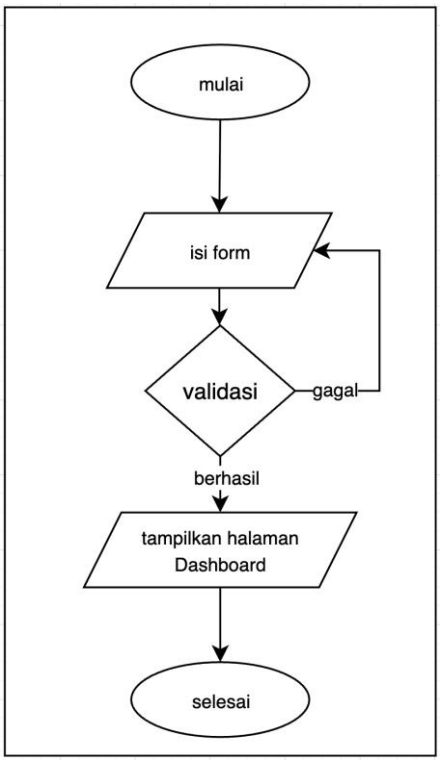
(4) Grafik *matriks login*.

Tabel 4. 13 Grafik *matriks login*

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

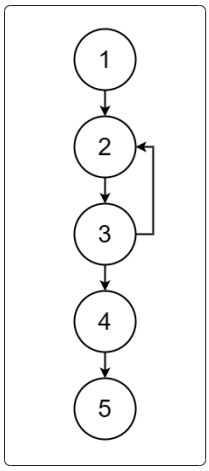
b. *Whitebox testing register*

1) *Flowchart*



Gambar 4. 45 *Flowchart register*

2) *Flowgraph*



Gambar 4. 46 *Flowgraph register*

Berdasarkan gambar 4. 46 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

(4) Grafik *matriks register* .

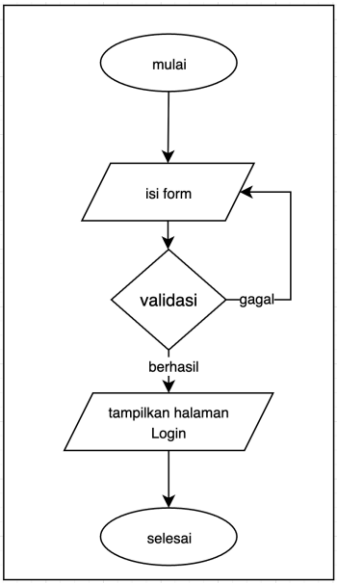
Tabel 4. 14 Grafik *matriks register*

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1

4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

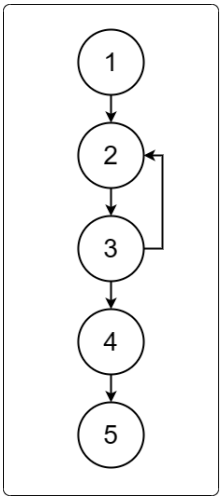
c. *Whitebox testing* lupa *password*

1) *Flowchart*



Gambar 4. 47 *Flowchart* lupa *password*

2) *Flowgraph*



Gambar 4. 48 *Flowgraph* lupa *password*

Berdasarkan gambar 4. 48 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

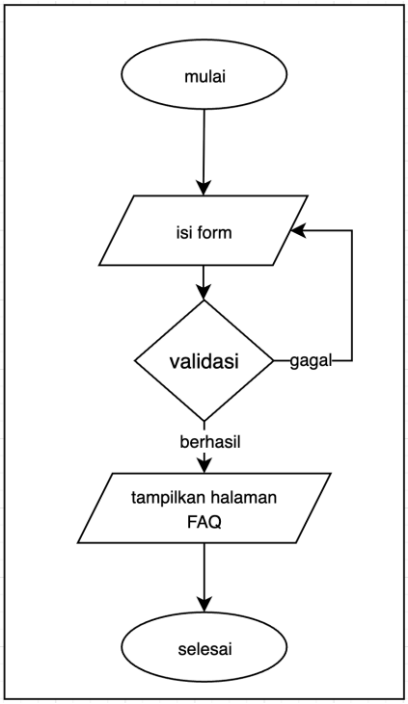
(4) Grafik *matriks* lupa *password*.

Tabel 4. 15 Grafik *matriks* lupa *password*

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

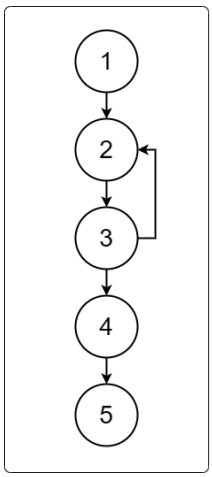
d. *Whitebox testing* tambah FAQ

1) *Flowchart*



Gambar 4. 49 *Flowchart* tambah FAQ

2) *Flowgraph*



Gambar 4. 50 *Flowgraph* tambah FAQ

Berdasarkan gambar 4. 50 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

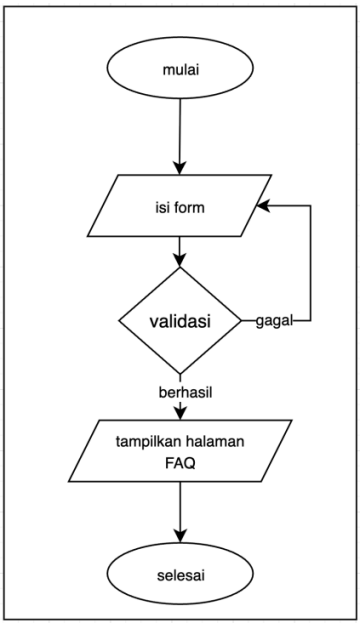
(4) Grafik *matriks tambah FAQ*

Tabel 4. 16 Grafik *matriks* tambah FAQ

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

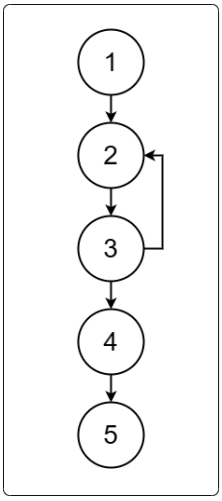
e. *Whitebox testing edit* FAQ

1) *Flowchart*



Gambar 4. 51 *Flowchart edit* FAQ

2) *Flowgraph*



Gambar 4. 52 *Flowgraph edit* FAQ

Berdasarkan gambar 4. 52 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

(4) Grafik *matriks edit FAQ*

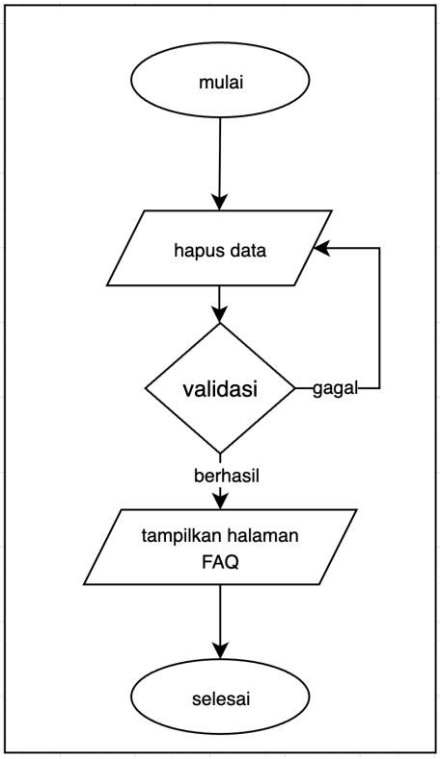
Tabel 4. 17 Grafik *matriks* t *edit* FAQ

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1

4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

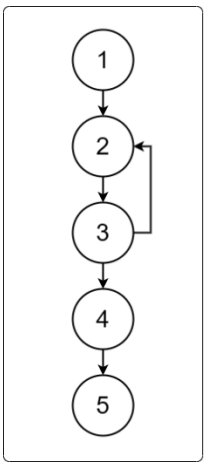
f. *Whitebox testing* hapus FAQ

1) *Flowchart*



Gambar 4. 53 *Flowchart* hapus FAQ

2) *Flowgraph*



Gambar 4. 54 *Flowgraph* hapus FAQ

Berdasarkan gambar 4. 54 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

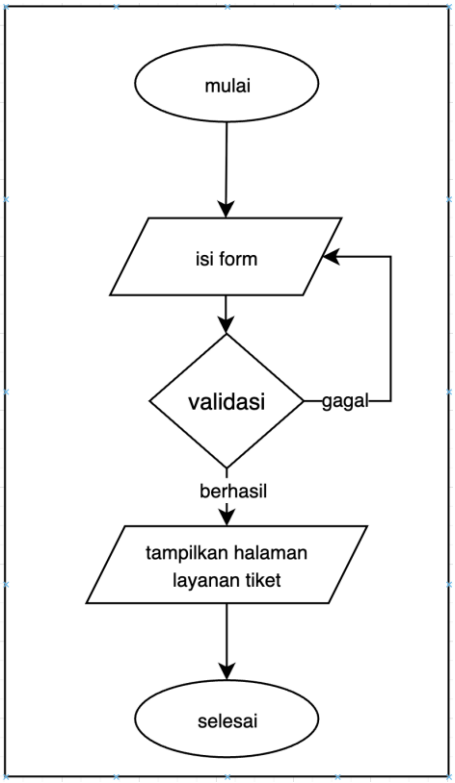
(4) Grafik *matriks hapus FAQ*

Tabel 4. 18 Grafik *matriks* hapus FAQ

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

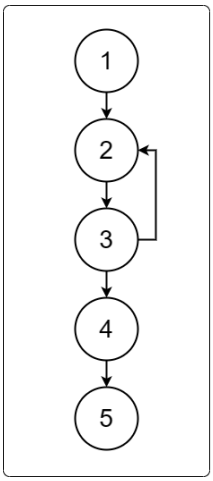
g. *Whitebox testing* tambah tiket

1) *Flowchart*



Gambar 4. 55 *Flowchart* tambah tiket

2) *Flowgraph*



Gambar 4. 56 *Flowgraph* tambah tiket

Berdasarkan gambar 4. 56 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

(4) Grafik *matriks tambah tiket*

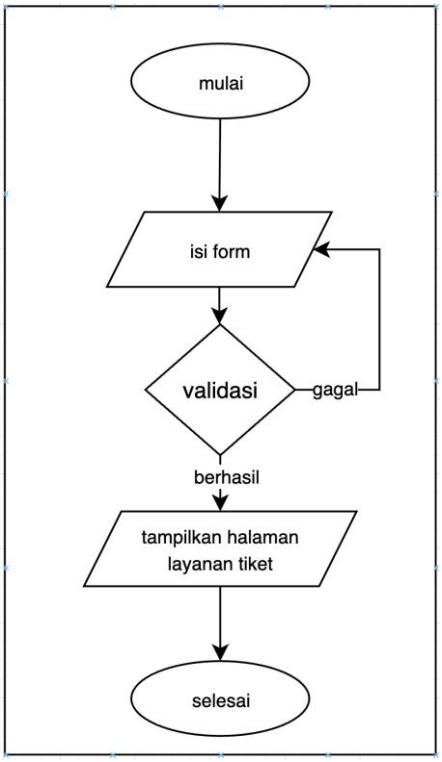
Tabel 4. 19 Grafik *matriks* tambah tiket

	1	2	3	4	5	E-1
1		1				1 - 1 = 0

2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

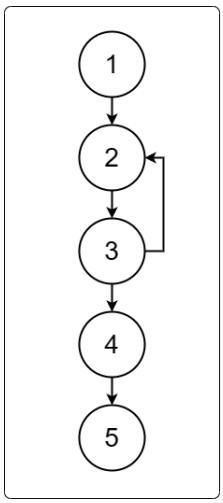
h. *Whitebox testing edit* tiket

1) *Flowchart*



Gambar 4. 57 *Flowchart edit* tiket

2) *Flowgraph*



Gambar 4. 58 *Flowgraph edit* tiket

Berdasarkan gambar 4. 58 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

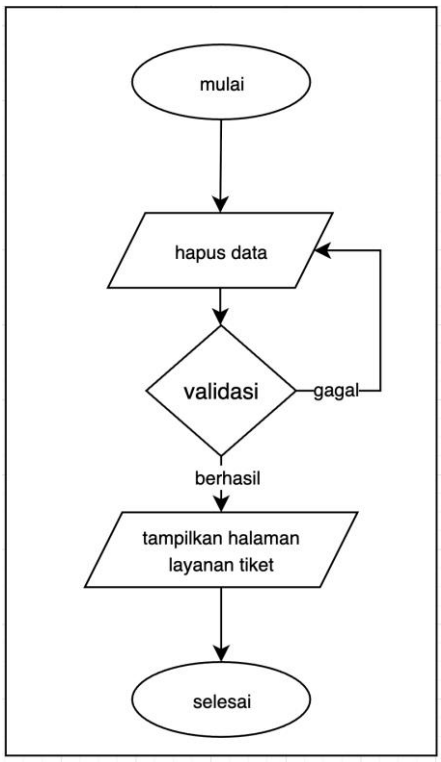
(4) Grafik *matriks tambah tiket*

Tabel 4. 20 Grafik *matriks edit* tiket

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

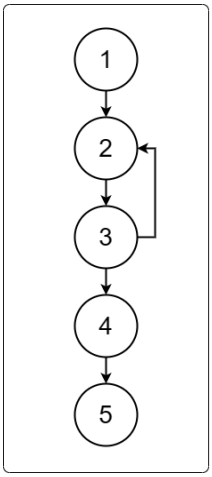
i. *Whitebox testing* hapus tiket

1) *Flowchart*



Gambar 4. 59 *Flowchart* hapus tiket

2) *Flowgraph*



Gambar 4. 60 *Flowgraph* hapus tiket

Berdasarkan gambar 4. 60 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

(4) Grafik *matriks hapus tiket*

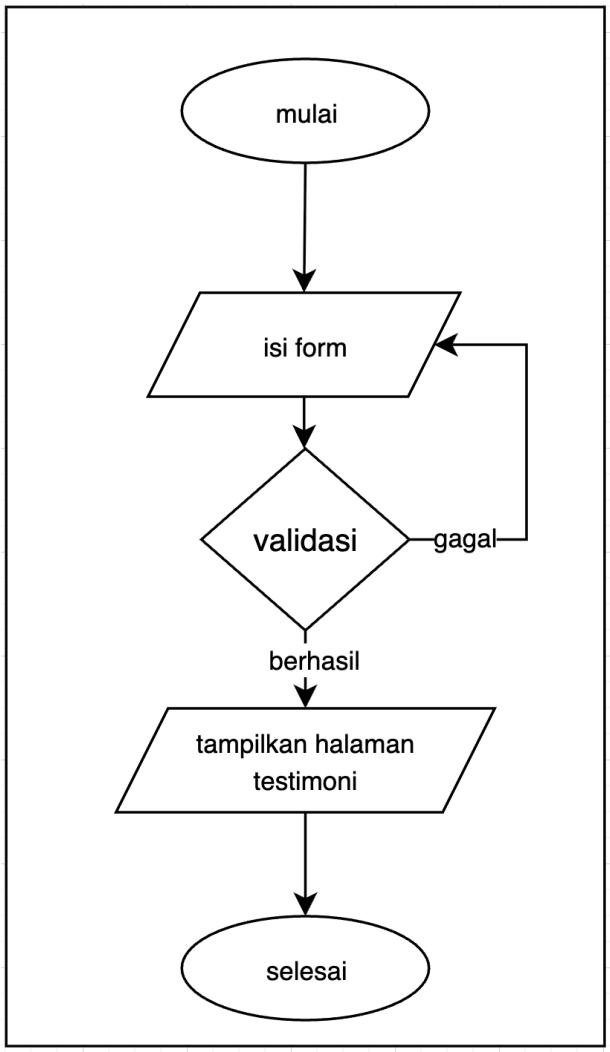
Tabel 4. 21 Grafik *matriks* hapus tiket

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0

3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

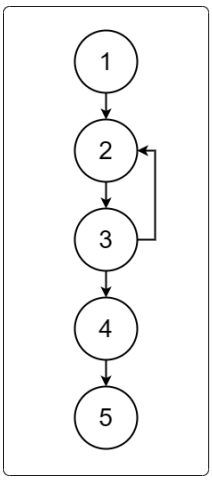
j. *Whitebox testing* tambah testimoni

1) *Flowchart*



Gambar 4. 61 *Flowchart* tambah testimoni

2) *Flowgraph*



Gambar 4. 62 *Flowgraph* tambah testimoni

Berdasarkan gambar 4. 62 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

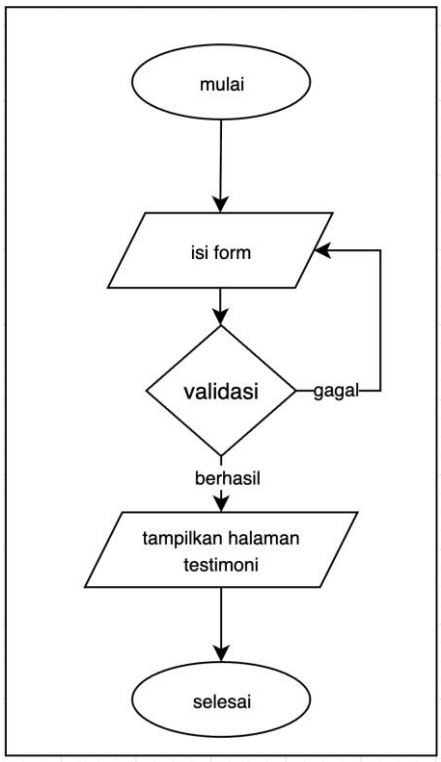
(4) Grafik *matriks tambah testimoni*

Tabel 4. 22 Grafik *matriks* tambah testimoni

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

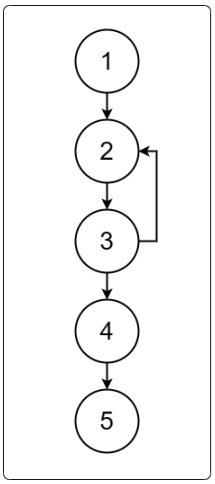
k. *Whitebox testing edit* testimoni

1) *Flowchart*



Gambar 4. 63 *Flowchart edit* testimoni

2) *Flowgraph*



Gambar 4. 64 *Flowgraph edit* testimoni

Berdasarkan gambar 4. 64 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

(4) Grafik *matriks edit testimoni*

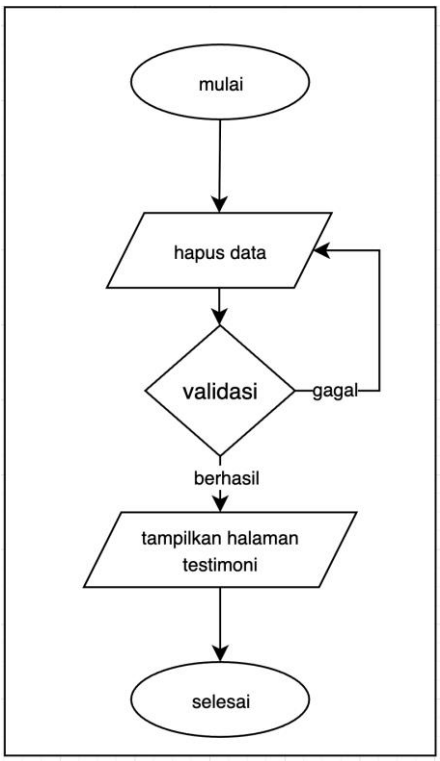
Tabel 4. 23 Grafik *matriks edit* testimoni

	1	2	3	4	5	E-1
1		1				1 - 1 = 0

2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

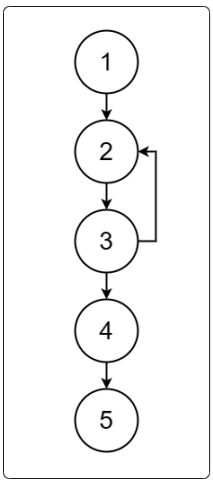
l. *Whitebox testing* hapus testimoni

1) *Flowchart*



Gambar 4. 65 *Flowchart* hapus testimoni

2) *Flowgraph*



Gambar 4. 66 *Flowgraph* hapus testimoni

Berdasarkan gambar 4. 66 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

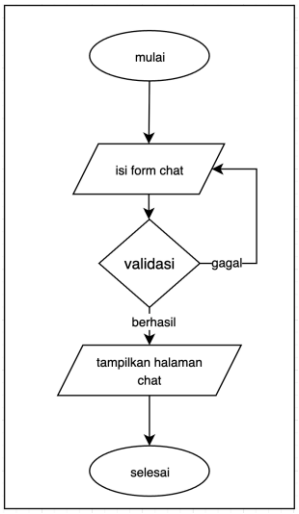
(4) Grafik *matriks hapus testimoni*

Tabel 4. 24 Grafik *matriks* hapus testimoni

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

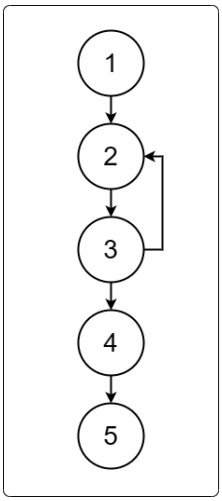
m. *Whitebox testing chat*

1) *Flowchart*



Gambar 4. 67 *Flowchart chat*

2) *Flowgraph*



Gambar 4. 68 *Flowgraph chat*

Berdasarkan gambar 4. 68 diatas, dilakukan perhitungan sebagai berikut:

(1) Menghitung *cyclomatic complexcity* V(G) pada *egde* dan *node*

Pada rumus : V(G) = E – N + 2

E (*edge*) = 5

N (*node*) = 5

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Predikat (P) = P + 1

= 1 + 1

= 2

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 2*

(3) *Independent path* pada *flowgraph* yaitu:

Path 1 = 1 – 2 – 3 – 2

Path 2 = 1 – 2 – 3 – 4 – 5

(4) Grafik *matriks chat*

Tabel 4. 25 Grafik *matriks chat*

	1	2	3	4	5	E-1
1		1				1 - 1 = 0
2			1			1 – 1 = 0
3		1		1		2 – 1 = 1
4					1	1 – 1 = 0
5						0
	SUM (E + 1)					1 + 1 = 2

BAB V

PENUTUP

A. Kesimpulan

Helpdesk berperan penting dalam mendukung layanan TI di sebuah institusi, namun di Universitas Muhammadiyah Parepare, tingkat penggunaan *Helpdesk* oleh mahasiswa masih rendah meskipun pengetahuannya cukup tinggi. Hal ini disebabkan oleh sistem yang dirasa rumit dan fitur yang kurang menarik. Untuk meningkatkan minat dan partisipasi mahasiswa, diperlukan pendekatan baru, salah satunya adalah dengan menerapkan *gamification*. *Gamification* diyakini mampu menciptakan pengalaman pengguna yang lebih menyenangkan dan memotivasi, melalui elemen seperti poin, *badge*, dan *leaderboard*. Oleh karena itu, dirancanglah aplikasi *Helpdesk* berbasis *gamification* guna meningkatkan keaktifan mahasiswa dalam pelaporan permasalahan layanan TI.

B. Saran

Penelitian ini memiliki beberapa keterbatasan yang dapat menjadi bahan pengembangan di masa mendatang. Pertama, penerapan *gamification* dalam sistem *Helpdesk* masih terbatas pada elemen-elemen dasar seperti poin, *badge*, dan *leaderboard*, tanpa mengeksplorasi lebih dalam elemen interaktif lainnya seperti misi, tantangan harian, atau sistem level pengguna. Kedua, penelitian ini belum mengukur secara kuantitatif dampak penerapan *gamification* terhadap peningkatan keaktifan pengguna dalam jangka panjang. Ketiga, penelitian hanya berfokus pada

lingkup Universitas Muhammadiyah Parepare, sehingga belum dapat digeneralisasi untuk institusi lain. Penelitian selanjutnya diharapkan dapat mencakup uji efektivitas sistem secara menyeluruh, pengembangan fitur *gamification* yang lebih variatif, serta penerapan pada skala yang lebih luas.

DAFTAR PUSTAKA

Ali, M., & Khan, S. (2022). Java eco*system* evolution: Managing JDK and JRE version compatibility in modern applications. International Journal of *Software Engine*ering and Technology, 14(3), 55–64.

Andrianto, & Wibowo. (2021). Pengembangan aplikasi pendataan mahasiswa menggunakan Visual Basic .NET dan SQL Server. Jurnal Sistem Informasi dan Komputer, 9(2), 112–120.

Fauzi, R., Hidayat, M., & Sari, L. (2022). Implementasi single point of contact pada help desk berbasis multi-channel communication. Jurnal Teknologi Informasi dan Ilmu Komputer, 9(3), 233–242.

Firmansyah, & Dewi. (2022). Penerapan augmented reality untuk visualisasi produk interior rumah pada media cetak brosur. Jurnal Sistem dan Teknologi Informasi, 6(2), 145–153.

Halim, R., Widodo, P., & Pratama, Y. (2024). Pemanfaatan kecerdasan buatan pada sistem help desk untuk prediksi masalah layanan TI. Jurnal Teknologi Informasi dan Komunikasi, 14(2), 102–110.

Kurniawan, A., & Santoso, B. (2021). Implementasi konsep inheritance pada bahasa pemrograman Java untuk pengembangan aplikasi modular. Jurnal Teknologi Informasi, 15(2), 112–120.

Lestari, & Pratama. (2023). Augmented reality untuk brosur promosi kampus menggunakan metode marker based tracking. Jurnal Ilmu Komputer dan Multimedia, 7(1), 44–52.

Lestari, S., & Hakim, R. (2025). Pengaruh penerapan self-service portal pada help desk terhadap kepuasan pengguna. Jurnal Ilmu Komputer Terapan, 15(1), 25–33.

Patel, R., & Sharma, V. (2023). Modern Java *Development Kit*: Architecture, API integration, and SDK components. Journal of *Software Engine*ering Advances, 18(2), 112–124.

Pratama, R., & Hidayat, F. (2023). Tren dan dominasi sistem operasi mobile di pasar global. Jurnal Teknologi Informasi dan Komunikasi, 18(1), 45–53.

Putra, D. A., & Rahman, F. (2023). Implementasi XML dalam integrasi data lintas plat*form*. Jurnal Teknologi Informasi dan Komputer, 11(2), 101–110.

Rahman, A., & Nugroho, B. (2021). Help desk *system* sebagai pusat layanan terpadu di era trans*form*asi digital. Jurnal Sistem Informasi, 17(2), 145–154.

Rahman, M. A., & Putri, D. S. (2022). Perkembangan dan ekosistem *Android* sebagai sistem operasi berbasis Linux. Jurnal Teknologi dan Sistem Informasi, 10(4), 215–224.

Rahman, T., & Gupta, S. (2023). Advances in *Android* SDK: API toolchain and cross-*language* support. International Journal of Mobile Computing and Application *Development*, 15(1), 45–58.

Rahmawati, Yuliana, & Anwar. (2024). Pengembangan brosur digital interaktif menggunakan augmented reality untuk promosi Fakultas Teknik. Jurnal Teknologi Informasi, 10(3), 201–

Saputra, Kurniawan, & Hidayat. (2022). Implementasi augmented reality pada brosur pameran interior berbasis markerless di *Android*. Jurnal Teknologi Informasi dan Komputer Pendidikan, 6(2), 85–93.

Setiawan, D., & Putri, M. (2023). Cloud-based help desk *system* dengan integrasi knowledge base dan real-time analytics. Jurnal Sistem dan Teknologi Informasi, 11(1), 55–64.

Syahputra, R., & Nugroho, A. (2021). Pemanfaatan *Eclipse* IDE dalam pengembangan perangkat lunak berbasis multiplat*form*. Jurnal Sistem Informasi dan Teknologi, 9(3), 178–186.

Wijaya, F., & Pratama, R. (2022). Pemanfaatan *Android* Developer *Tools* (ADT) pada *Eclipse* untuk pengembangan aplikasi *Android* berbasis Java. Jurnal Teknologi Informasi dan Aplikasi, 14(1), 45–53.