BAB I

PENDAHULUAN

A. Latar Belakang

Pelanggaran lalu lintas adalah masalah yang seringkali dihadapi di banyak negara, termasuk Indonesia. Proses penanganan tilang yang masih menggunakan sistem manual di beberapa daerah di Indonesia cenderung memakan waktu dan tidak selalu efisien. Oleh karena itu, diperlukan inovasi dalam penegakan hukum lalu lintas, terutama dalam hal penanganan surat tilang.

Dalam upaya meningkatkan efisiensi dan akurasi proses penanganan tilang, penggunaan teknologi informasi menjadi suatu keharusan. Android sebagai platform mobile telah menjadi salah satu pilihan utama untuk pengembangan aplikasi berbasis mobile.

Dalam konteks ini, penelitian ini bertujuan untuk merancang dan membangun aplikasi tilang elektronik berbasis Android dengan pemanfaatan teknologi. Aplikasi ini diharapkan dapat memberikan solusi untuk mempermudah dalam proses penindakan pelanggaran lalu lintas terutama dalam pencatatan pelanggaran.

Pemanfaatan teknologi diharapkan dapat mempermudah identifikasi dan pencatatan data pada surat tilang, mengurangi kesalahan input, serta meningkatkan kecepatan dalam penanganan tilang. Selain itu, dengan integrasi platform Android, aplikasi tilang elektronik ini dapat diakses secara mudah dan cepat oleh petugas penegak hukum di lapangan.

Penelitian ini akan memberikan kontribusi signifikan dalam pengembangan sistem penanganan tilang yang lebih modern. Selain itu, diharapkan aplikasi yang dihasilkan dapat menjadi referensi dan model bagi pengembangan sistem serupa di masa depan. Dengan pemanfaatan teknologi informasi, diharapkan penegakan hukum lalu lintas dapat menjadi lebih mudah dan dapat meningkatkan kinerja petugas lalu lintas.

B. Rumusan Masalah

Berdasarkan latar belakang masalah yang dipaparkan, rumusan masalah dalam penelitian ini adalah: Bagaimana merancang dan membangun aplikasi tilang elektronik berbasis andoroid yang dapat mempermudah dalam pencatatan pelanggaran lalu lintas

C. Batasan Masalah

Berdasarkan rumusan masalah di atas, maka batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Lingkup penggunaan aplikasi ini terbatas pada masyarakat yang memiliki kendaraan roda dua.Fitur pada aplikasi Android tidak disediakan untuk pengguna kendaraan roda empat.
2. Aplikasi yang dibuat untuk petugas yang dilapangan di tujukan untuk Android, oleh karena itu bahasa pemrograman yang digunakan dalam pembuatannya adalah Java.
3. Aplikasi yang dibuat untuk petugas yang bertindak sebagai admin ditujukan untuk website, oleh karena itu Bahasa yang digunakan dalam pembuatannya adalah PHP dengan framework CodeIgniter

D. Tujuan Penelitian

Penelitian ini bertujuan untuk merancang dan membangun aplikasi tilang elektronik berbasis Android yang memanfaatkan teknologi. Tujuan utamanya adalah menghasilkan antarmuka pengguna yang mudah digunakan oleh petugas atau pihak yang bertanggung jawab, serta mengintegrasikan teknologi untuk identifikasi pelanggaran lalu lintas secara cepat dan akurat. Selain itu, penelitian ini berfokus untuk mempermudah pencatatan pelanggaran dengan memastikan keamanan dan keakuratan data yang tercatat dalam aplikasi.

E. Manfaat Penelitian

Adapun manfaat yang dapat diberikan dari penelitian ini adalah sebagai berikut:

1. Bagi akademik

   Sebagai bahan referensi bagi penulis lain untuk mengembangkan kemampuan di bidang yang sama.

2. Bagi penulis

   Penulis dapat mengimplementasikan ilmu yang didapat selama proses perkuliahan.

3. Bagi masyarakat

   Masyarakat terkhususnya petugas berwenang dapat melakukan pencatatan pelanggaran lalu lintas dengan lebih mudah

BAB II

TINJAUAN PUSTAKA

A. Kajian Teori

1. Pelanggaran Lalu Lintas

pelanggaran lalu lintas adalah perbuatan atau Tindakan yang bertentangan dengan ketentuan-ketentuan peraturan perundang-undangan mengenai lalu lintas dimana merupakan perbuatan atau tindakan yang melanggar sesuatu yang berhubungan dengan hukum, berarti tidak lain dari pada perbuatan melawan hukum (Naning R, 1983).

Sanksi pelanggaran lalu lintas di jalan raya semakin berat. Dalam undangundang tentang lalu lintas yang terbaru, sanksi denda atau tilang naik sekitar 10 kali lipat dengan kisaran Rp 250 ribu hingga Rp 1 juta. Berdasarkan UndangUndang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan, yang disahkan DPR pada 22 Juni 2009

Adapun Peraturan Pelanggaran Lalu Lintas antara lain:

a. Setiap pengendara kendaraan bermotor yang tidak memiliki SIM dipidana dengan pidana kurungan paling lama 4 bulan atau dendapaling banyak Rp 1 juta (Pasal 281).
b. Setiap pengendara kendaraan bermotor yang memiliki SIM namun tak dapat menunjukkannya saat razia dipidana dengan pidana kurungan paling lama 1 bulan atau denda paling banyak Rp 250 ribu (Pasal 288 ayat 2).

c. Setiap pengendara kendaraan bermotor yang tak dipasangi Tanda Nomor Kendaraan dipidana dengan pidana kurungan paling lama 2 bulan atau denda paling banyak Rp 500 ribu (Pasal 280).
d. Setiap pengendara sepeda motor yang tidak memenuhi persyaratan teknis dan laik jalan seperti spion, lampu utama, lampu rem, klakson, pengukur kecepatan, dan knalpot dipidana dengan pidana kurungan paling lama 1 bulan atau denda paling banyak Rp 250 ribu (Pasal 285 ayat1).
e. Setiap pengendara mobil yang tidak memenuhi persyaratan teknis seperti spion, klakson, lampu utama, lampu mundur, lampu rem, kaca depan, bumper, penghapus kaca dipidana dengan pidana kurungan paling lama 2 bulan atau denda paling banyak Rp 500 ribu (Pasal 285 ayat 2).
f. Setiap pengendara mobil yang tidak dilengkapi dengan perlengkapan berupa ban cadangan, segitiga pengaman, dongkrak, pembuka roda, dan peralatan pertolongan pertama pada kecelakaan dipidana dengan pidana kurungan paling lama 1 bulan atau denda paling banyak Rp 250 ribu (Pasal 278).
g. Setiap pengendara yang melanggar rambu lalu lintas dipidana dengan pidana kurungan paling lama 2 bulan atau denda paling banyak Rp 500 ribu (Pasal 287 ayat 1).

h. Setiap pengendara yang melanggar aturan batas kecepatan paling tinggi atau paling rendahdipidana dengan pidana kurungan paling lama 2 bulan atau denda paling banyak Rp 500 ribu (Pasal 287 ayat 5).

i. Setiap pengendara yang tidak dilengkapi Surat Tanda Nomor Kendaraan Bermotor atau Surat Tanda Coba Kendaraan Bermotor dipidana dengan pidana kurungan paling lama 2 bulan atau denda paling banyak Rp 500 ribu (Pasal 288 ayat 1).

j. Setiap pengemudi atau penumpang yang duduk disamping pengemudi mobil tak mengenakan sabuk keselamatan dipidana dengan pidana kurungan paling lama 1 bulan atau denda paling banyak Rp 250 ribu (Pasal 289).

k. Setiap pengendara atau penumpang sepeda motor yang tak mengenakan helm standar nasionaldipidana dengan pidana kurungan paling lama 1 bulan atau denda paling banyak Rp 250 ribu (Pasal 291 ayat 1).

l. Setiap orang yang mengemudikan Kendaraan Bermotor di Jalan tanpa menyalakan lampu utama pada malam hari dan kondisi tertentu sebagaimana dimaksud dalam Pasal 107 ayat (1) dipidana dengan pidana kurungan paling lama 1 (satu) bulan atau denda paling banyak Rp250.000,00 (dua ratus lima puluh ribu rupiah) (Pasal 293 ayat 1).

m. Setiap orang yang mengemudikan Sepeda Motor di Jalan tanpa menyalakan lampu utama pada siang hari sebagaimana dimaksud dalam Pasal 107 ayat (2) dipidana dengan pidana kurungan paling lama 15

(lima belas) hari atau denda paling banyak Rp100.000,00 (seratus ribu rupiah) (Pasal 293 ayat 2).

n. Setiap pengendara sepeda motor yang akan berbelok atau balik arah tanpa memberi isyarat lampu dipidana kurungan paling lama 1 bulan atau denda paling banyak Rp 250 ribu (Pasal 294).

2. Android

Android adalah sistem operasi untuk perangkat seluler berbasis Linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Android dikembangkan oleh sebuah perusahaan kecil di Silicon Valley bernama Android Inc. Selanjutnya, Google mengadopsi sistem operasi tersebut pada tahun 2005 dan mendeklarasikannya sebagai sistem operasi "*Open Source*". Hasilnya, siapa pun dapat menggunakannya secara gratis, termasuk kode sumber yang digunakan untuk mengkompilasi sistem operasi tersebut. Android sendiri merupakan sistem operasi dengan pengguna terbanyak di dunia, karena sifatnya yang terbuka bagi para *developer* untuk membuatnya lebih fungsional.

Gambar 2. 1 Logo *Android*

Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang dapat digunakan pada berbagai

perangkat penggerak. Smartphone pertama yang menggunakan Android adalah *HTC Dream*, yang dirilis pada tahun 2008 (Junadhi et al., 2021).

Adapun Kelebihan dari android yaitu:

a. Platform lengkap. Menyediakan alat yang berguna untuk membuat aplikasi yang kemudian dapat dikembangkan lebih lanjut oleh pengembang.
b. Platform *Open Source*. Pengembang mudah mengembangkannya karena bersifat terbuka
c. Platform gratis. Pengembang dapat dengan bebas mengembangkan, mendistribusikan, dan memperdagangkan sistem operasi Android tanpa harus membayar royalti untuk mendapatkan lisensi

3. Android Studio

Android Studio merupakan IDE (*Integrated Development Environment*) untuk mengembangkan aplikasi Android. Aplikasi ini diterbitkan oleh Google pada tanggal 16 Mei 2013 dan tersedia secara gratis dengan lisensi *Apache* 2.0. Android Studio menggantikan perangkat lunak pengembangan Android sebelumnya, yaitu *Eclipse*.

IDE (*Integrated Development Environment*) adalah aplikasi untuk pengembang perangkat lunak yang berisi fungsi-fungsi terintegrasi yang diperlukan untuk membangun perangkat lunak, seperti editor kode, debugger, kompiler, dan sebagainya. Android Studio sendiri dikembangkan berdasarkan *IntelliJ IDEA*, mirip dengan *Eclipse*, disertai dengan plugin ADT (*Android Development Tools*).

Android Studio memiliki fitur:

a. Proyek berdasarkan *Gradle Build.*

b. *Refactoring* cepat dan perbaikan bug.

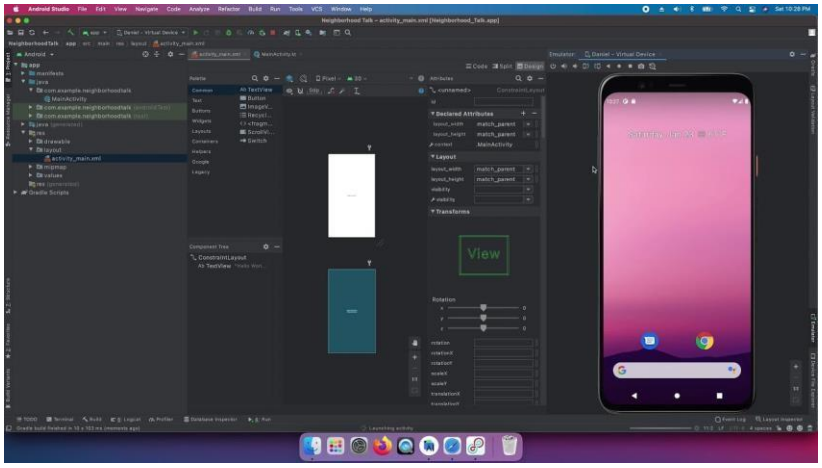
c. Tools baru bernama "*Lint*" mengklaim dapat dengan cepat memantau kecepatan, kegunaan, dan kompatibilitas aplikasi.

d. Mendukung *Proguard* dan penandatanganan aplikasi untuk keamanan.

e. Memiliki GUI aplikasi Android lebih mudah.

f. Didukung oleh Google *Cloud Platform* untuk setiap aplikasi yang dikembangkan.

Android Studio dipilih karena memiliki banyak fitur yang memudahkan para programmer, khususnya programmer tingkat dasar yang ingin mempelajari lebih lanjut tentang Android. Meski menggunakan Android Studio memakan waktu, kelebihan fungsionalitas dan dukungan yang diberikan membuatnya menjadi pilihan yang baik dalam pengembangan aplikasi Android. Perangkat lunak Android Studio menjadi salah satu software yang cukup populer sebab termasuk ke dalam software yang mudah dalam penggunaannya. Selain itu, perangkat lunak ini juga sudah mendukung berbagai macam fitur menarik untuk pembuatan aplikasi (Yanti & Lesmideryati, 2022).



Gambar 2. 2 Tampilan *Android Studio*

Meski menggunakan Android Studio memakan cukup banyak RAM pada perangkat PC, namun Android Studio memiliki sejumlah keunggulan lain, yaitu:

a. *Instant Run*
b. *Intelligent Code Editor*
c. Sistem Versi yang Flexibel
d. Dioptimalkan untuk semua perangkat Android
e. Didesain untuk Tim

4. Java

Java adalah bahasa pemrograman berbasis komputer ke berorientasi objek pemrograman komputer. Java dirancang seperti itu sehingga ukurannya kecil,simple dan portable (dapat dialihkan antara yang berbeda platform dan sistem operasi). Program yang mana dihasilkan dalam bahasa Java dapat berupa applet (aplikasi kecil yang berjalan di atas browser web) atau dalam bentuk aplikasi mandiri yang berjalan dengan program Java Interpreter.

Gambar 2. 3 Logo *JAVA*

Java adalah bahasa pemrograman yang populer, dikembangkan oleh *Sun Microsystems*. Salah satu penggunaan terbesar Java adalah dalam pembuatan aplikasi *native* untuk Android. Bahasa pemrograman ini bersifat *multiplatform*, artinya dapat digunakan di berbagai *platform*, seperti desktop, Android, dan bahkan untuk sistem operasi Linux (Sibarani et al., 2019).

Menurut Moch. Anif, R.A.M. Dyah Ayuningtyas, dan Dwi Agung Nugroho (2020), bahasa pemrograman java memiliki beberapa karakteristik, antara lain:

a. Sederhana Bahasa Java menggunakan syntax yang mirip dengan bahasa pemrograman C++. Akan tetapi syntax di bahasa pemrograman Java banyak yang telah diperbaiki, terutama dengan menghilangkan pointer yang rumit dan multiple inheritance. Java juga menggunakan garbage collection dan automatic memory allocation.
b. Berorientasi Objek Bahasa pemrograman Java merupakan bahasa pemrograman yang berorientasi objek sehingga memungkinkan program untuk dibuat secara modular dan digunakan kembali.

c. Terdistribusi Bahasa pemrograman Java dibuat untuk memudahkan distribusi aplikasi dengan terintegrasinya networking libraries ke dalam Java.

d. Interpreted Program Java dijalankan dengan menggunakan Interpreter, yaitu JVM (Java Virtual Machine). Hal tersebut mengakibatkan source code Java yang telah dikompilasi menjadi bytecodes (sebuah format yang tidak tergantung pada arsitektur tertentu yang didesain untuk mengirimkan kode ke banyak platform hardware dan software secara efisien) dapat dijalankan dalam berbagai platform.

e. Robust Bahasa pemrograman Java mempunyai reliabilitas yang tinggi. Kompiler pada bahasa ini memiliki kemampuan untuk mendeteksi kesalahan yang lebih baik dibandingkan bahasa pemrograman lain. Bahasa Java memiliki Runtime Exception Handling yang berfungsi untuk membantu mengatasi kesalahan dalam pemrograman. f. Secure Java, sebagai bahasa pemrograman aplikasi internet dan terdistribusi, mempunyai beberapa mekanisme keamanan yang berfungsi menjaga agar aplikasi yang dijalankan tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

g. Architecture Neutral Program Java tidak bergantung pada platform dimana program akan dijalankan. Cukup dengan membuat program yang dapat dijalankan di berbagai platform dengan menggunakan Java Virtual Machine.

h. Portable Source code (kode sumber) dan program pada Java dapat dibawa dengan mudah ke berbagai platform yang berbeda tanpa harus dilakukan kompilasi ulang.

i. Multithreaded Bahasa pemrograman Java dapat membuat sebuah program yang mampu melakukan beberapa pekerjaan secara simultan dan bersamaan.

j. Performance Kinerja bahasa pemrograman ini sering kali disebut kurang, tetapi kinerjanya bisa ditingkatkan dengan menggunakan compiler Java lain seperti buatan Microsoft, Symantec, maupun Inprise yang menggunakan JIT (Just In Time) Compilers. k. Dynamic Java dapat didesain untuk mampu dijalankan di lingkungan yang dinamis. Perubahan suatu class dengan menambahkan properties maupun metode mampu dilakukan tanpa menggangu program yang menggunakan class tersebut.

5. *Library* Java

Menurut Yanti, N., dan Lesmideryati, D. (2022), "*Library* Java adalah bagian penting dari bahasa pemrograman Java. Dengan memanfaatkan *library* yang tepat, Anda dapat mengembangkan aplikasi Java yang lebih cepat, berkualitas, dan aman."

Keunggulan *Library* Java:

a. Menghemat waktu dan tenaga: Penggunaan *library* Java dapat mengurangi kebutuhan untuk menulis kode dari awal, sehingga mempercepat proses pengembangan aplikasi.

b. Meningkatkan kualitas dan keamanan: *Library* Java yang dikembangkan dengan baik telah melalui pengujian dan perbaikan sehingga meminimalisir bug dan meningkatkan keamanan aplikasi.
c. Standarisasi dan interoperabilitas: Banyak *library* Java mengikuti standar pemrograman yang umum, sehingga memudahkan kolaborasi dan integrasi dengan komponen lain.
d. Variasi dan kelengkapan: Terdapat banyak *library* Java yang tersedia untuk berbagai kebutuhan, mulai dari pengembangan antarmuka pengguna, *database*, jaringan, hingga *machine learning*.

Faktor Pemilihan *Library*:

a. Kebutuhan fungsional: Pilih *library* yang menyediakan fungsi dan fitur sesuai kebutuhan aplikasi Anda.
b. Dokumentasi dan komunitas: Ketersediaan dokumentasi yang jelas dan komunitas aktif memudahkan proses belajar dan mengatasi masalah.
c. Lisensi dan dukungan: Perhatikan jenis lisensi (*open source* atau komersial) dan ketersediaan dukungan teknis.
d. Performa dan popularitas: Pertimbangkan efisiensi dan kestabilan library, serta tingkat popularitas dan adopsi *developer* lain.

Contoh *Library* Java Populer:

a. *ZXing*: *ZXing ("Zebra Crossing")* adalah API populer untuk pemrosesan kode QR di Java.
b. *Spring Framework*: *Framework web development* dan *enterprise* dengan fitur lengkap dan fleksibel.

c. *Apache Commons Lang*: *Library* berisi utilitas umum *untuk string, date, collection*, dan lainnya.

d. *Apache POI*: *Library* untuk membaca dan menulis file Microsoft Office (Word, Excel, PowerPoint).

e. *JavaFX*: *Framework* untuk pengembangan aplikasi *desktop* dan *mobile* dengan user *interface* yang kaya.

f. *Guava*: *Library* berisi koleksi utilitas tambahan untuk bahasa Java, memperkaya *functionality* standar.

Saat memilih *library* Java, penting untuk mempertimbangkan kebutuhan fungsional, dokumentasi dan komunitas, lisensi dan dukungan, performa dan popularitas. Dengan mempertimbangkan faktor-faktor tersebut, akan dapat menemukan *library* Java yang tepat bagi kebutuhan sebuah aplikasi yang akan dikerjakan (Fatdha, et al., 2021).

6. **JDK (*Java Development Kit)***

JDK (*Java Development Kit*) adalah paket perangkat lunak yang menyediakan berbagai fungsi dan alat yang diperlukan untuk pengembangan aplikasi dengan bahasa pemrograman Java. JDK meliputi komponen-komponen kunci seperti:

a. *Java Runtime Environment* (JRE): Komponen ini memungkinkan pengguna untuk menjalankan aplikasi Java yang telah dikompilasi. JRE berisi JVM, kumpulan kelas-kelas dasar, dan perpustakaan yang diperlukan untuk menjalankan program Java.

b. *Java Virtual Machine* (JVM): JVM adalah mesin virtual yang menjalankan bytecode Java. JVM bertanggung jawab untuk mengeksekusi program Java dengan menerjemahkan bytecode menjadi instruksi yang dapat dijalankan oleh sistem operasi dan perangkat keras target. JVM juga memiliki tugas lain, seperti manajemen memori, garbage collection, dan keamanan.
c. Kompiler Java: Kompiler Java mengonversi kode sumber Java menjadi bytecode, yaitu kode yang berbentuk bahasa mesin virtual yang dapat dipahami oleh JVM.

Menurut Fatdha, T. S. E., Junadhi, Susanti, Yenni, H., dan Zoromi, F. (2021), "JDK adalah paket perangkat lunak yang sangat penting bagi *developer* Java. Dengan adanya JDK, pengembang dapat dengan mudah mengembangkan aplikasi Java yang berkualitas dan aman."

Berikut adalah beberapa keuntungan menggunakan JDK:

a. Portabilitas: Aplikasi Java yang dikompilasi dapat dijalankan di berbagai platform dan sistem operasi tanpa perlu kompilasi ulang. Hal ini karena JVM bertanggung jawab untuk menerjemahkan *bytecode* menjadi instruksi yang dapat dijalankan oleh sistem operasi dan perangkat keras target.
b. Keamanan: JVM memiliki *built-in security features* yang membantu melindungi aplikasi Java dari serangan *malware* dan lainnya.
c. Efisiensi: JVM menggunakan teknik-teknik optimasi untuk meningkatkan kinerja aplikasi Java.

7. SDK (*Software Development Kit)*

Dalam pengembangan aplikasi mobile, SDK (*Software Development Kit*) merupakan komponen penting yang menyediakan berbagai fungsi dan API yang diperlukan untuk membangun aplikasi mobile. SDK dapat membantu pengembang untuk membangun aplikasi mobile yang berkualitas dan efisien (Fitri et al., 2022).

Keberadaan SDK mempermudah pengembang untuk mengakses berbagai fitur sistem dan kemampuan khusus dari platform yang dituju, memastikan bahwa aplikasi yang dihasilkan dapat berfungsi secara optimal dan responsif.

Dalam dunia pengembangan perangkat lunak, SDK menjadi komponen krusial yang mendefinisikan bagaimana sebuah aplikasi dapat diintegrasikan dan beroperasi di atas platform tertentu. Dengan berbagai alat dan API yang tersedia, SDK memungkinkan pengembang untuk fokus pada logika dan fungsionalitas aplikasi tanpa harus memikirkan detail teknis dari platform yang digunakan. Hal ini memfasilitasi proses pengembangan yang lebih cepat dan efektif, sekaligus memastikan bahwa aplikasi yang dihasilkan sesuai dengan standar kualitas dan kinerja yang diharapkan.

Dalam konteks Android, SDK menjadi jembatan antara ide dan realisasi aplikasi. Dengan dukungan dari SDK, pengembang dapat mengeksplorasi potensi penuh dari sistem operasi Android dan menghasilkan aplikasi yang memanfaatkan keunikan serta kecanggihan platform tersebut. Sebagai alat yang mendukung, SDK berperan penting dalam menginspirasi inovasi dan kreativitas

pengembang, memungkinkan terciptanya aplikasi yang memenuhi berbagai kebutuhan pengguna dengan beragam fitur dan fungsionalitas.

8. XML

XML merupakan singkatan dari eXtensible Markup Language. Bahasa markup merupakan sekumpulan peraturan yang mendefinisikan suatu sintak yang digunakan untuk mendeskripsikan dan menjelaskan teks atau data pada sebuah dokumen menggunakan penggunaan tag. Bahasa XML dibuat untuk menyimpan sebuah data secara ringkas dan mudah diatur. Intinya, bahasa XML yaitu data yang diolah dapat memberikan informasi.

XML merupakan sebuah bahasa markup yang digunakan untuk mengolah data informasi yang menggambarkan struktur dan maksud tujuan data yang terdapat dalam dokumen XML, namun tidak menggambarkan format tampilan data tersebut. XML merupakan sebuah standar sederhana yang digunakan untuk mendeskripsikan data teks dengan cara deskripsi diri (self-describing).

Namespace XML digunakan untuk nama unik dari elemen dan atribut di dalam sebuah dokumen XML. Pada pengembangan Aplikasi Android, setiap kali kita ingin menerapkan RelativeLayout atau LinearLayout sebagai RootView pada Layout, kita wajib mendefinisikan namespace XML, yang di mana kita menggunakan xmlns:android sebagai atribut, dan nilai atau value "http://schemas.android.com/apk/res/android". Karena ini merupakan unique identifier, sama halnya dengan penggunaan di bahasa pemrograman PHP atau C++.

9. MYSQL

MySQL adalah salah satu jenis database yang terkenal dan termasuk jenis *RDBMS (Relational Database Management System).* Kepopuleran *MySQL* disebabkan karena *MySQL* menggunakan bahasa *SQL* sebagai bahasa dasar untuk query dan bersifat open source di berbagai platform.

MySQL adalah Relational Database Management System (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (General Public License), di mana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat closed source atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Gambar 2. 4 Logo *MYSQL*

Sebagai database server, MySQL dapat dikatakan lebih unggul dibandingkan database server lainnya, terutama dalam kecepatan. Berikut beberapa keistimewaan MySQL, antara lain:

a) Portability

MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac OS X Server, Solaris Amiga, dan masih banyak lagi.

b) Multiuser

MySQL dapat digunakan oleh beberapa pengguna (user) dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

c) Security

MySQL memiliki beberapa lapisan keamanan seperti level subnetmask, nama host, dan izin akses pengguna dengan sistem perizinan yang mendetail serta kata sandi terenkripsi.

d) *Scalability* dan *Limits*

MySQL mampu menangani database dalam skala besar, dengan jumlah rekaman lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu, batas indeks yang dapat ditampung mencapai 32 indeks pada setiap tabelnya (Huda, 2022).

10. GPS (Global Positioning System)

GPS adalah sebuah sistem untuk menentukan posisi di permukaan bumi dengan bantuan sinkronisasi sinyal satelit. Sistem ini menggunakan 24 satelit untuk mengirimkan sebuah sinyal gelombang mikro ke Bumi. Sinyal itu

diterima oleh alat penerima yang ada di permukaan. Sistem ini juga didesain untuk memberikan posisi dan kecepatan tiga dimensi serta informasi mengenai waktu, secara kontinyu di seluruh dunia tidak bergantung pada waktu dan cuaca, bagi banyak orang secara simultan.

Saat ini, GPS sudah banyak digunakan orang di berbagai belahan dunia dalam berbagai bidang yang menuntut informasi tentang posisi, waktu, kecepatan, ataupun percepatan yang teliti. GPS dapat memberikan suatu informasi posisi dengan ketelitian yang bervariasi dari beberapa milimeter (orde nol) sampai ratusan meter.

11. API

API (Application Programming Interface) merupakan sekumpulan perintah, fungsi, serta protokol yang digunakan oleh programmer saat membangun suatu perangkat lunak pada sistem operasi tertentu. API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi, sehingga mempermudah proses pengembangan perangkat lunak. API menyediakan lapisan abstraksi yang mengizinkan aplikasi untuk berkomunikasi dengan sistem operasi atau layanan lainnya tanpa harus memahami detail implementasinya.

API terdiri dari berbagai jenis, termasuk API berbasis web, API sistem operasi, API perpustakaan perangkat lunak, dan API aplikasi. Misalnya, API berbasis web memungkinkan aplikasi untuk berkomunikasi dengan server melalui protokol HTTP atau HTTPS, sedangkan API sistem operasi

memungkinkan perangkat lunak untuk mengakses sumber daya sistem seperti file, perangkat keras, atau proses lainnya.

API juga dapat mendukung berbagai fungsi, mulai dari autentikasi pengguna, pengambilan dan pengiriman data, hingga manajemen sesi dan transaksi. Dengan menggunakan API, pengembang dapat mempercepat waktu pengembangan, meningkatkan efisiensi, dan memastikan interoperabilitas antara berbagai sistem dan aplikasi. API juga mendukung modularitas dalam pengembangan perangkat lunak, yang memungkinkan kode untuk digunakan kembali di berbagai aplikasi dan konteks.

12. Goole Maps

Google Maps merupakan sebuah jasa peta dunia virtual gratis yang disediakan oleh Google dan dapat diakses secara online di http://maps.google.com/. Google Maps menawarkan peta yang dapat digeser dan gambar di seluruh dunia, memungkinkan pengguna untuk menjelajahi berbagai lokasi secara visual. Layanan ini juga menawarkan fitur peta jalan, citra satelit, pemandangan jalan (Street View), kondisi lalu lintas real-time, dan perencanaan rute untuk perjalanan dengan berjalan kaki, mobil, sepeda, atau transportasi umum.

Google Maps juga menyediakan beberapa utilitas untuk memanipulasi peta dan menambahkan konten di dalam peta tersebut melalui berbagai layanan. Salah satu fitur utama adalah API Google Maps, yang memungkinkan pengembang untuk menambahkan peta yang interaktif ke situs web atau aplikasi mereka. API ini mendukung berbagai fungsi, termasuk penandaan

lokasi (markers), poligon dan polylines untuk menandai area dan rute, serta pencarian tempat dan rute.

Selain itu, Google Maps telah berkembang menjadi platform yang kuat untuk integrasi data geografis dan visualisasi. Misalnya, API Google Maps dapat digunakan untuk menampilkan data demografis, statistik kriminalitas, informasi bisnis, dan banyak lagi, dalam konteks geografis yang mudah dipahami. Hal ini memungkinkan pengguna untuk mendapatkan wawasan yang lebih mendalam dari data yang ditampilkan pada peta.

Dengan Google Maps, pengguna juga dapat memanfaatkan fitur geolokasi untuk menentukan posisi mereka secara real-time, yang sangat berguna dalam aplikasi navigasi dan pelacakan. Fitur ini telah banyak diadopsi dalam berbagai aplikasi, mulai dari layanan ride-sharing, aplikasi pengiriman makanan, hingga platform media sosial yang menawarkan geotagging. Google Maps terus berkembang dengan penambahan fitur-fitur baru yang semakin memperkaya pengalaman pengguna dan memfasilitasi berbagai kebutuhan geografis dan navigasi.

13. *HTML*

Gambar 2. 5 *Hypertext Markup Language*

Hypertext Markup Language atau *HTML*, adalah bahasa *markup* standar industri untuk membangun halaman *web*. Ini menawarkan kumpulan

tag dan properti yang menentukan pengorganisasian halaman *web*, konten, dan format halaman *web*. Untuk merepresentasikan elemen pada halaman *web*, *HTML* menggunakan struktur hierarkis yang dikenal sebagai *Document Object Model* (*DOM*). Teks, foto, tautan, multimedia, dan elemen lainnya semuanya dapat dimuat dalam setiap elemen, yang dibatasi oleh *tag* pembuka dan penutup.

Istilah "*HyperText*" mengacu pada kemampuan *HTML* untuk membuat tautan yang memungkinkan pengguna dengan mudah dan cepat berpindah dari satu situs *web* ke situs *web* lainnya. *HyperText Markup Language*, juga dikenal sebagai HTML, adalah bahasa global *World Wide Web*. Seperti yang tersirat dari namanya *HyperText Markup Language*, HTML adalah bahasa *markup* bukan bahasa pemrograman (Hartl & Donahoe, 2019).

Dokumen yang dibuat dengan menggunakan bahasa bahasa markup seperti *HTML* memiliki bagian yang dirender pada hasil akhir serta bagian yang menginstruksikan program rendering cara menginterpretasikan teks yang tersisa. Proses mengubah dokumen teks yang berisi teks *HTML* menjadi, misalnya, representasi visualnya di layar disebut sebagai "*rendering*" (Nematrian, 2020).

Perkembangan *HTML* sangat pesat. Dari permulaan *HTML* 1.0 yang sederhana, yang mencakup elemen dasar seperti judul dan paragraf, hingga penambahan tabel dan gambar pada *HTML* 2.0, dan akhirnya ke *frame* dan dukungan *form* yang disempurnakan pada *HTML* 3.2. Dengan *CSS*, konten dan presentasi sekarang dapat dipisahkan dalam *HTML* 4.01, yang juga

meningkatkan komponennya. Dengan elemen semantiknya, dukungan multimedia, bentuk yang lebih baik, fitur aksesibilitas, dan kemampuan *mobile-friendly*, *HTML5* menandai periode transformasi.

Versi *HTML* terbaru, yang dikenal sebagai *HTML5*, digunakan untuk membuat dan mengatur dokumen di internet. Ini mencakup elemen dan fitur baru yang memungkinkan pembuatan halaman *web* yang lebih dinamis dan menarik, termasuk dukungan untuk audio dan video, kanvas untuk grafik dan animasi, dan peningkatan kompatibilitas dengan perangkat seluler.

Fungsionalitas baru yang diperkenalkan oleh *HTML5* memungkinkan pembuatan halaman *web* yang lebih dinamis dan interaktif, yaitu:

a. Dukungan audio dan video.
b. Kanvas visual dan animasi.
c. Dukungan yang ditingkatkan untuk *mobile*..
d. *Semantic Element* baru termasuk *nav*, *article*, *sections* dan *headers*.
e. Kontrol untuk form, yaitu tanggal, waktu, email, pencarian, dll.
f. Penyimpanan *web* untuk menyimpan informasi secara lokal.
g. Fitur *drag* and *drop*.

14. *CSS*

CSS adalah bahasa yang terus berkembang untuk menggambarkan presentasi konten *web* di layar, printer, sintesis ucapan, pembaca layar, dan jendela obrolan. *CSS* digunakan oleh semua *browser* pada semua ukuran layar di semua jenis perangkat *IoT*, termasuk ponsel, komputer, video game, televisi, jam tangan, dan konsol otomatis (Meyer & Weyl, 2020). Sedangkan menurut

(Duckett, 2022) *CSS* adalah bahasa pemrograman *web* yang digunakan untuk mengendalikan tampilan halaman *web*. (Castro, 2022) menjelaskan bahwa *CSS* memiliki berbagai properti yang dapat digunakan untuk mengatur tampilan halaman *web*. *CSS* terdiri dari dua bagian utama, yaitu deklarasi dan selektor. Deklarasi adalah bagian yang menentukan properti dari suatu halaman *web*. Selektor adalah bagian yang menentukan elemen *web* mana yang akan diatur oleh deklarasi. *CSS* juga memiliki berbagai properti yang dapat digunakan untuk mengatur tampilan halaman *web*. Beberapa properti *CSS* yang umum digunakan antara lain:

a. Ukuran: properti ini digunakan untuk mengatur ukuran halaman *web*, seperti ukuran font, ukuran gambar, dan ukuran jarak antar elemen.
b. Warna: properti ini digunakan untuk mengatur warna halaman *web*, seperti warna font, warna latar belakang, dan warna *border*.
c. Font: properti ini digunakan untuk mengatur font elemen *web*, seperti jenis font, ukuran font, dan gaya font.
d. Tata letak: properti ini digunakan untuk mengatur tata letak elemen *web*, seperti posisi elemen, ukuran elemen, dan jarak antar elemen.

15. *PHP*

Gambar 2. 6 *Hypertext Preprocessor*

PHP adalah bahasa *server-side* yang secara khusus digunakan untuk aplikasi *web*. Bahasa *PHP* berjalan di sisi *server*, artinya ia berjalan pada *server* dan bisa dimasukkan di antara bahasa *HTML* (*HyperText Markup Language*). Oleh karena itu, kode *PHP* tidak akan terlihat lagi dan hasil akhir dalam bentuk *HTML* akan dikirimkan ke *browser* (Sutarman, 2020). Karena *PHP* dijalankan di komputer *server*, itu dikenal sebagai bahasa pemrograman sisi *server*. Ini berbeda dengan bahasa pemrograman sisi klien, seperti JavaScript, yang beroperasi pada klien, yaitu *web browser*.

Pada awalnya, *PHP* adalah singkatan dari *Personal Home Page*. *PHP* digunakan untuk mengembangkan situs *web* pribadi, sesuai dengan namanya. Seiring berjalannya waktu, *PHP* berkembang menjadi bahasa pemrograman *web* yang sangat kuat, tidak hanya digunakan untuk membuat halaman *web* dasar tetapi juga untuk situs-situs populer seperti *Wikipedia*, *WordPress*, *Joomla*, dan lainnya yang dikunjungi oleh jutaan orang.

16. *Javascript*

Gambar 2. 7 *Javascript*

Javascript adalah bahasa pemrograman *open-source* untuk teknologi *web* yang dapat digunakan untuk memodifikasi *HTML* dan *CSS*. Biasanya disingkat menjadi *JS* saja. Sebagian besar *browser web*, termasuk *Internet*

Explorer, *Google Chrome*, dan *Mozilla Firefox*, menggunakan *JavaScript* untuk operasi yang dilakukan di antarmuka penggunanya. Karena kemampuannya untuk menginstruksikan *browser* untuk melakukan tugas, sekarang JavaScript menjadi *client-side scripting language* yang paling populer (Shute, 2020).

JavaScript dapat meningkatkan kegunaan halaman *web* dengan memberikan umpan balik yang cepat. Dengan bantuan bahasa pemrograman *JavaScript,* Anda dapat meningkatkan *HTML* Anda dengan menambahkan animasi, interaksi, dan efek visual yang dinamis untuk pengguna *browser*. (McFarland, 2021). Selain itu, *JavaScript* dapat digunakan pada *platform* apa pun dan merupakan alat yang berguna untuk mengembangkan *program* dan makro sementara. Anda dapat dengan cepat membuat perangkat lunak yang layak dengan menggunakan *text editor* dan *browser web* berkat ketersediaan *browser* yang tersebar luas (Patrick, 2019).

Saat pertama kali keluar, *JavaScript* hanya memiliki sedikit "bagian yang bagus". Saat ini, sebagian besar komponen sangat baik. Meskipun masih ada beberapa fitur lawas, fitur baru yang ditawarkan oleh standar *ES6/7/8* membantu Anda menulis kode yang lebih aman, mudah, dan singkat (Subramaniam, 2019) Kode JavaScript mengalami perubahan yang cukup signifikan saat *ES6* atau *ECMAScript 6* diterbitkan. Sekarang Anda akan dapat dengan cepat membedakan antara kode yang ditulis menggunakan sintaks sebelum *ES6* diterbitkan dan kode yang ditulis menggunakan sintaks *ES6* karena perubahannya sangat signifikan. *Javascript* modern saat ini sangat

menyenangkan untuk ditulis. Saya telah lama menggeluti dunia pemrograman, tetapi saya akui bahwa kadang-kadang saya merasa sintaks sebelumnya canggung dan sulit dijelaskan daripada sintaks Javascript modern ini yang lebih unggul (Morgan, 2019).

17. CodeiIgniter

CodeIgniter merupakan aplikasi open source berupa framework PHP dengan model MVC (Model, View, Controller) untuk membangun aplikasi web dinamis dengan cepat dan mudah. CodeIgniter memiliki desain dan struktur file yang sederhana, didukung dengan dokumentasi yang lengkap sehingga framework ini lebih mudah dipelajari. Framework ini memungkinkan para pengembang untuk menggunakan framework secara parsial atau secara keseluruhan, memberi kebebasan kepada pengembang untuk menulis bagian-bagian kode tertentu di dalam aplikasi menggunakan cara konvensional atau dengan sintaks umum dalam PHP, tanpa harus sepenuhnya mengikuti aturan penulisan kode di CodeIgniter.

Keunggulan utama CodeIgniter terletak pada performanya yang cepat, footprint yang kecil, dan fleksibilitasnya dalam hal konfigurasi. CodeIgniter tidak memerlukan server berbasis command-line, tidak memaksakan penggunaan template engine tertentu, dan memberikan kontrol penuh kepada pengembang atas struktur aplikasi mereka. Ini membuatnya ideal bagi pengembang yang membutuhkan framework yang kuat namun tetap ingin mempertahankan kebebasan dalam metode pengembangan.

18. *PhpMyAdmin*

Phpmyadmin adalah sebuah aplikasi *open source* yang berfungsi untuk memudahkan manajemen *MySQL*. Dengan menggunakan *phpmyadmin*, anda dapat membuat *database*, membuat tabel, meng-*insert*, menghapus dan meng-*update data* dengan *GUI* dan terasa lebih mudah, tanpa perlu mengetikkan perintah *SQL* secara *manual*. Karena berbasis *web*, maka *phpmyadmin* dapat dijalankan di banyak *OS*, selama dapat menjalankan *web server* dan *MySQL*

19. UML (*Unified Modelling Language*)

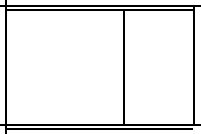
UML (*Unified Modeling Language*) adalah alat desain sistem berorientasi objek. Secara filosofis kemunculan UML terinspirasi dari konsep yang sudah ada yaitu konsep pemodelan *Object Oriented* (OO), karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh objek dan digambarkan atau dicatat dalam simbol-simbol yang cukup spesifik. proses standar dan independen.

UML adalah pemodelan untuk membantu proses perancangan sistem sehingga meminimalisir kesalahan dalam membuat program. Penerapan UML ini menggambarkan struktur aktor yang terlibat, aktifitas setiap aktor, proses danmekanisme dimana memberikan kemudahan perancangan dalam membatu proses pengkodingan menjadi sebuah aplikasi (Voutama, 2022).

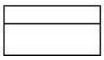
UML memiliki tiga kategori utama yaitu diagram struktur, diagram perilaku, dan diagram interaksi. Dimana setiap kategori mempunyai diagram yang menjelaskan arsitektur sistem dan saling terintegrasi.

Adapun daftar simbol UML yaitu :

Tabel 2. 1 *Symbol Use Case Diagram*

No.	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *Use Case*.
2		*Dependency*	Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri (*Independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
3		*Generalization*	Hubungan di mana objek anak (*Descendent*) berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk (*Ancestor*).
4		*Include*	Menypesifikasikan bahwa Use Case sumber secara Eksplisit.
5		*Extend*	Menypesifikasikan bahwa *use case* target memperluas perilaku dari *use* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menypesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemenelemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 2 *Symbol Class Diagram*

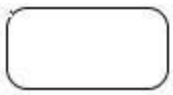
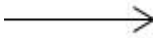
No.	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan di mana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

Tabel 2. 3 *Symbol Sequence Diagram*

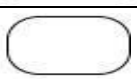
No.	AMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat in*form*asi in*form*asi tentang aktivitas yang terjadi
3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat in*form*asi

			in*form*asi tentang aktivitas yang terjadi

Tabel 2. 4 *Symbol State Chart Diagram*

No.	GAMBAR	NAMA	KETERANGAN
1		*State*	Nilai atribut dan nilai Link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
2		*Initial Pseudo State*	Bagaimana objek dibentuk atau diawali
3		*Final State*	Bagaimana objek dibentuk dan dihancurkan
4		*Transition*	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya
5		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
6		*Node*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2. 5 *Symbol Activity Diagram*

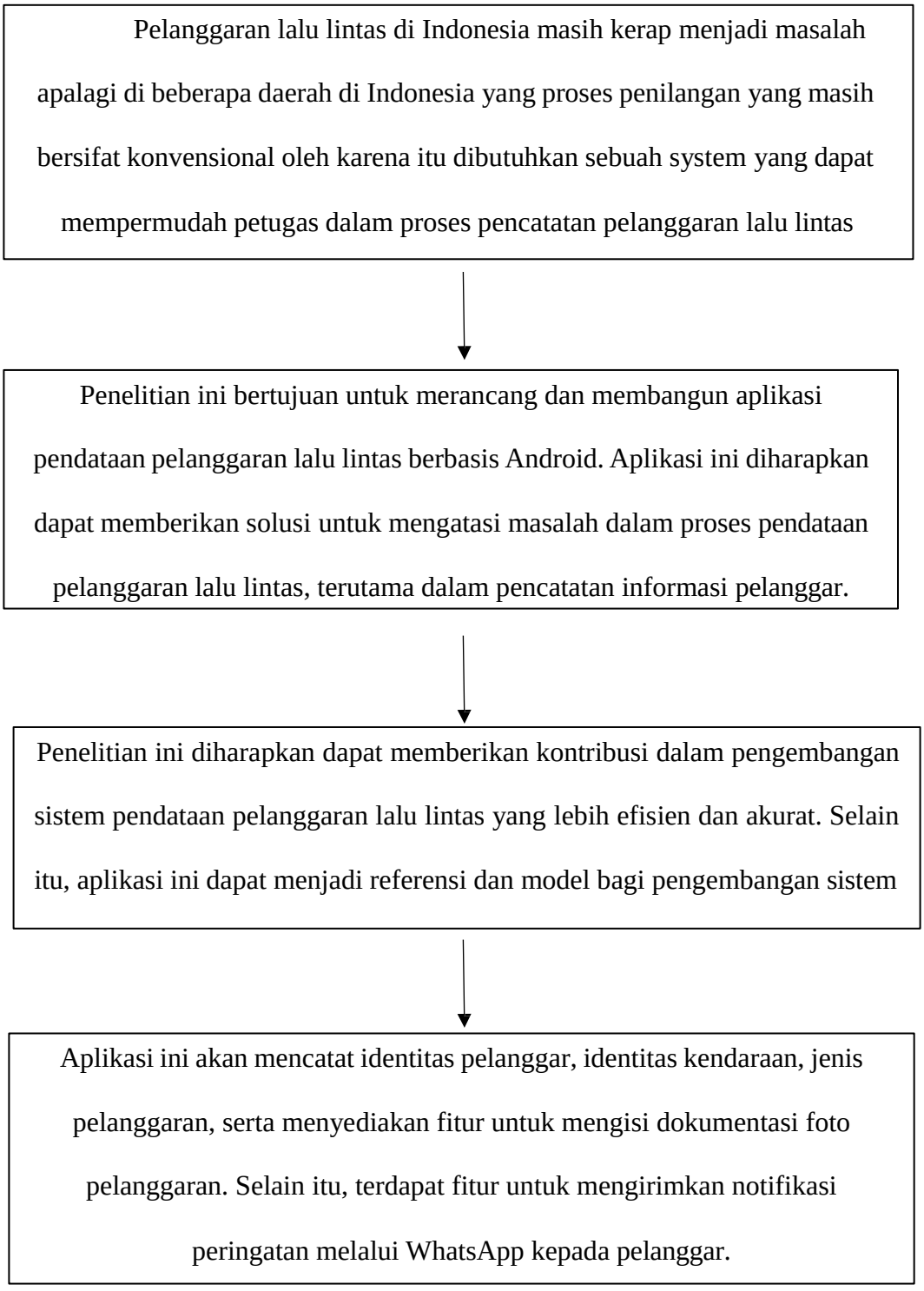
No.	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana interaksi masing-masing antarmuka
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork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

B. Kajian Hasil Penelitian Terdahulu

1. (Nurdiansah, Irmawati, n.d.). “Membangun Aplikasi Tilang Berbasis Database Terdistribusi Pada Dir Satlantas Sul-Sel”. Universitas Dipa Makassar. Tujuan dari penelitian ini adalah untuk membuat aplikasi WEB yang dapat mempermudah proses transaksi data yang dilakukan oleh anggota, khususnya dalam transaksi data tilang bagi pelanggar aturan lalulintas.
2. (Agung Purwantoro et al., 2021). “Aplikasi Tilang Menggunakan Scan Plat Nomor Kendaraan Berbasis Android”. Universitas Adhirajasa Reswara Sanjaya. Tujuan dari penelitian ini adalah untuk membuat sebuah aplikasi tilang mobile yang dapat mempermudah proses penilangan dan berhasil memanfaatkan aplikasi tilang mobile untuk menyampaikan informasi sifat-sifat pada jenis-jenis pelanggaran lalu lintas dengan penerapan metode waterfall.
3. (Dennis & Ekawati, 2020), “Perancangan Aplikasi Absensi Karyawan Dengan Menggunakan Kode Qr Berbasis Android”. Universitas Putera Batam. Tujuan dari penelitian ini adalah membuat sebuah sistem aplikasi absensi QR Code berbasis android yang bisa memudahkan karyawan dalam prosesn absensi.

C. Kerangka Pikir

Berikut ini bentuk kerangka berpikirnya agar bias memahami alur diatas:



BAB III

METODE PENELITIAN

A. Jenis Penelitian

Dalam menyelesaikan penelitian ini, jenis penelitian yang dilakukan adalah sebagai berikut:

1. Penelitian Pustaka (*Library Research*): Penelitian ini dilakukan dengan menggunakan beberapa buku sebagai referensi untuk penulisan.
2. Penelitian Lapangan (*Field Research*): Penelitian ini dilakukan dengan cara mengambil informasi di kantor Satlantas Polres Parepare.

B. Lokasi dan Waktu

Penelitian dilakukan di kantor Satlantas Polres Parepare, Jl. Andi Isa No.3, Ujung Sabbang, Kec. Ujung, Kota Parepare, Sulawesi Selatan 91113 Penelitian ini akan berlangsung selama ± 2 (dua) bulans.

C. Alat dan Bahan

1. Perangkat Keras (*Hardware*)
   a. Laptop Realme Book Prime dengan spesifikasi *hardware*:
      - *Processor* : Intel Core i5 11th i5-11320H @ 3.2GHz
      - *Installed RAM* : 8GB
      - *SSD* : 512GB
      - *Graphic Card* : Intel(R) Iris(R) Xe Graphic

2. Perangkat Lunak (*Software*)
    a. Windows 11 Home Single Language
    b. Android Studio
    c. Java
    d. JDK
    e. SDK

D. Teknik Pengumpulan Data

1. Studi Pustaka

    Studi pustaka dilakukan guna mengatahui perkembangan terkini dari sistem serupa maupun teknologi yang digunakan saat ini.Sumber pustaka yang digunakan ialah cetak dan elektronik.

2. Observasi

    Penelitian dilakukan dengan cara mengumpulkan data-data yang akan dijadikan bahan dasar dalam perancangan sistem.

3. Wawancara

    Penelitian dilakukan dengan proses tanya jawab kepada Pimpinan dan Pegawai yang berkompeten dan mengetahui permasalahan yang akan dibahasRancangan Sistem

E. Metode Pengujian

Dalam penelitian ini, digunakan 2 (dua) metode dalam pengujian datanya yaitu *blackbox testing* dan *whitebox testing*:

1. *Blackbox testing*

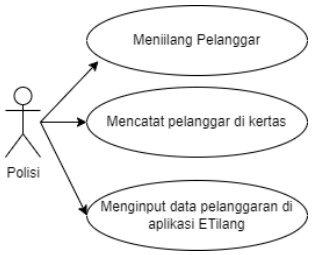
   Blackbox testing terfukus pada fungsional dari program yang ada. Pada *Blackbox testing* diuji dengan cara menjalankan program kemudian diamati apakah program tersebut apakah berhasil atau tidak. *Blackbox testing* menggunakan teknik *equivalence partitions* yang merupakan pengujian berdasarkan masukan setiap menu yang terdapat pada program, setiap menu masukan dilakukan pengujian melalui klasifikasi dan pengelompokan berdasar fungsinya.

2. *Whitebox testing*

   Whitebox testing bertujuan untuk mengetahui apakah struktur pada aplikasi yang dibuat sudah sesuai dengan ketentuan. *whitebox testing* menitikberatkan pada pengujiaan dengan mengecek detail peracangan perangkat lunak. *whitebox testing* dimilai dengan mendefiniskan semua alur dari perangkat lunak, kemudian membangun kasus yang akan digunakan dalam proses pengujian, kemudian menguji kasus tersebut untuk memperoleh hasilnya.

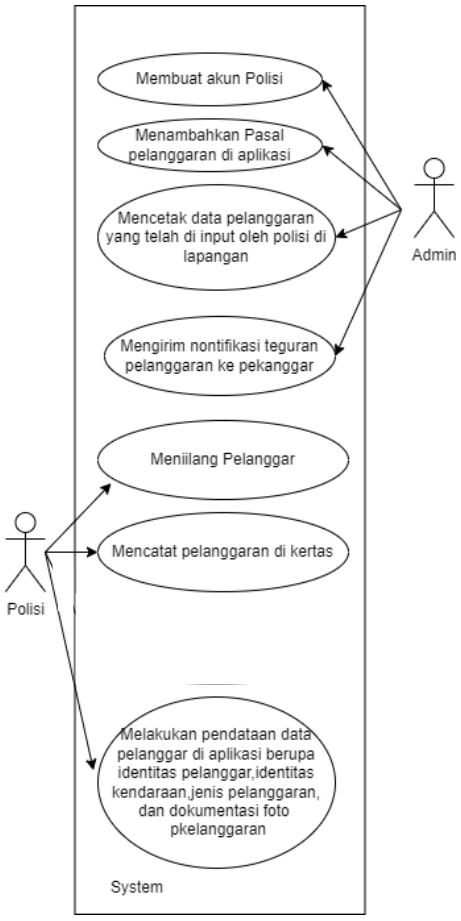
F. Desain Sistem

1. Desain system yang berjalan



Gambar 3. 1 Sistem yang berjalan

2. Desain system yang diusulkan



Gambar 3.2 Sistem yang diusulkan

BAB IV

HASIL DAN PEMBAHASAN

A. Analisis Aliran Data UML

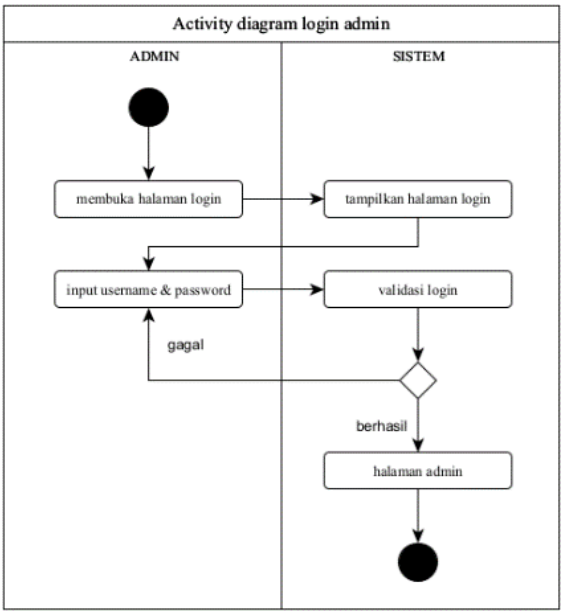
Dalam bagian ini, akan dibahas dua diagram UML yang menggambarkan aliran data dan interaksi dalam aplikasi Ensiklopedia Masakan Nusantara, yaitu *Activity diagram* dan *Sequence diagram*.

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 admin*

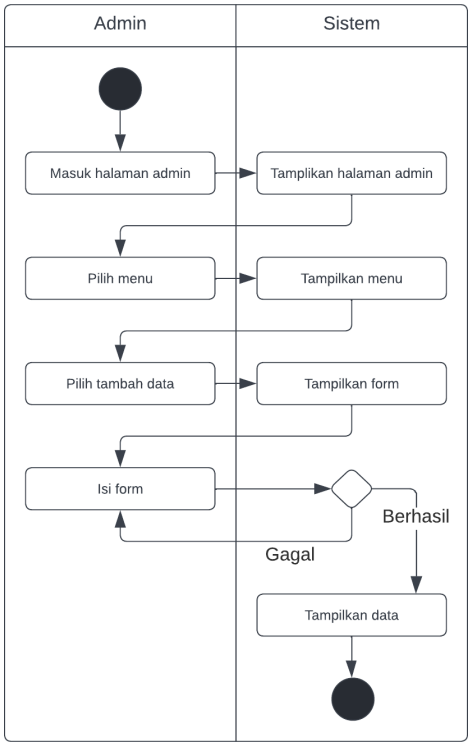
1) *Activity diagram login*



Gambar 4. 1 *Activity diagram login admin*

Pada gambar 4.1 menjelaskan cara masuk sebagai *admin*. *Admin* harus terlebih dahulu mengakses situs web, setelah itu sistem akan menampilkan *form login* dan *admin* harus memasukkan nama pengguna dan kata sandi. Sistem kemudian akan melakukan validasi; jika informasi benar, maka akan masuk ke halaman *admin*; jika tidak, maka akan kembali ke halaman *login*.

2) *Activity diagram* tambah data

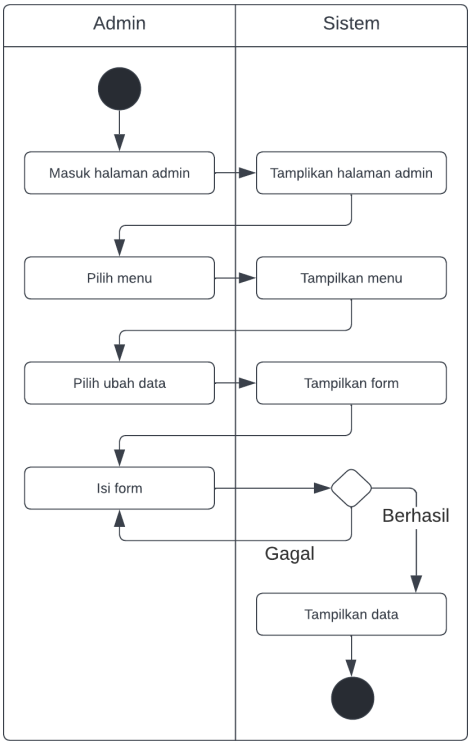


Gambar 4. 2 *Activity diagram* tambah data

Pada gambar 4. 2 menjelaskan langkah-langkah yang digunakan oleh *admin* untuk menambahkan data. Halaman *admin* ditampilkan oleh sistem ketika telah dibuka oleh admin. *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.

3) *Activity diagram* ubah data

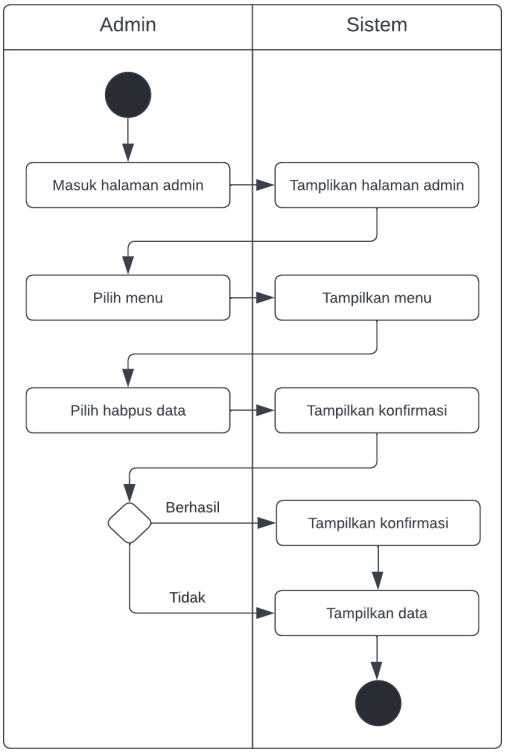


Gambar 4. 3 *Activity diagram* ubah data

Pada gambar 4. 3 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.

4) *Activity diagram* hapus data

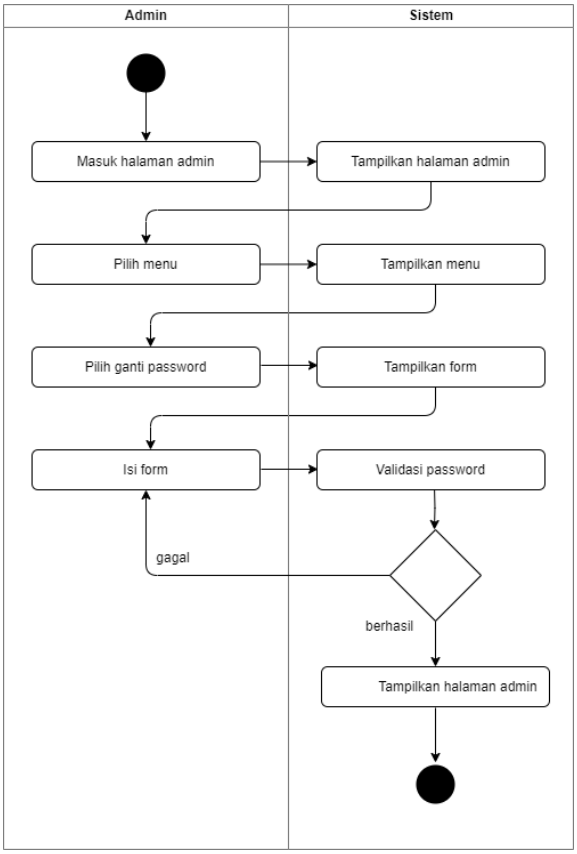


Gambar 4. 4 *Activity diagram* hapus data

Pada gambar 4. 4 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.

5) *Activity diagram* ubah password

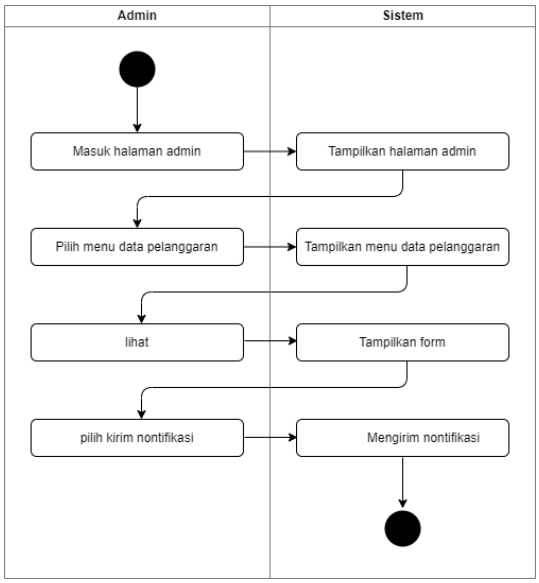


Gambar 4. 5 *Activity diagram ubah password*

Pada gambar 4. 5 menjelaskan prosedur yang digunakan *admin* untuk mengubah password. Halaman *admin* ditampilkan oleh sistem ketika telah dibuka oleh *admin*. *Admin* kemudian memilih menu ganti *password* yang kemudian akan muncul sebuah form dan admin akan mengisi password lama dan password baru nya di sana

dan apabila validasinya berhasil maka akan kembali menampilkan halaman admin

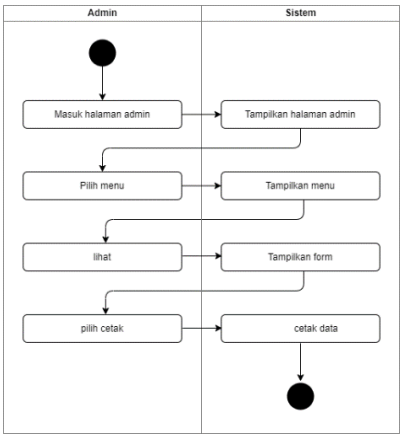
6) *Activity diagram* kirim nontifikasi *whatsapp*



Gambar 4. 6 *Activity diagram* kirim *nontifikasi whatsapp*

Pada gambar 4. 6 menjelaskan prosedur yang digunakan *admin* untuk mengirim nontifikasi *whatsapp* ke pelanggar dimana terlebih dahulu *admin* harus masuk ke halaman *admin* kemudian memilih menu data tilang kemudian memilih aksi lihat dan selanjutnya menekan tombol kirim nontifikasi yang nantinya nontifikasi akan terkirim ke nomor si pelanggar

7) *Activity diagram* cetak data

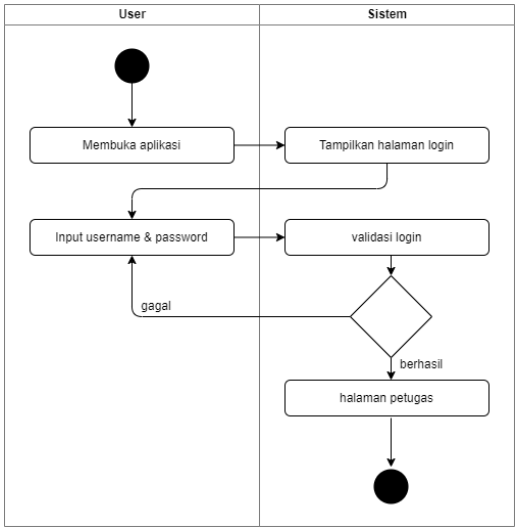


Gambar 4. 7 *Activity diagram* cetak data

Pada gambar 4. 7 menjelaskan prosedur yang digunakan *admin* untuk mencetak data. Halaman *admin* ditampilkan oleh sistem ketika telah dibuka oleh *admin*. *Admin* kemudian memilih menu lihat yang nantinya setelah menu lihat terbuka admin menekan tombol cetak dan selanjutnya system akan mencetak data.

b. Activity diagram user

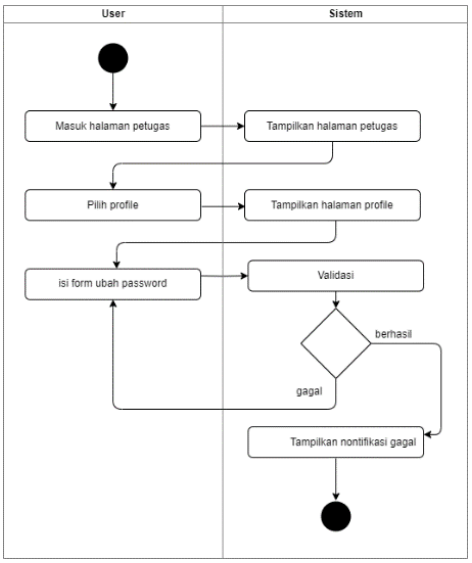
1) *Activity diagram login*



Gambar 4. 8 *Activity diagram login user*

Pada gambar 4.8 menjelaskan cara masuk sebagai *user* / petugas. *user* harus terlebih dahulu membuka aplikasi, setelah itu sistem akan menampilkan *form login* dan *user* harus memasukkan nama pengguna dan kata sandi. Sistem kemudian akan melakukan validasi; jika informasi benar, maka akan masuk ke halaman *user* / petugas dan jika tidak, maka akan kembali ke halaman *login*.

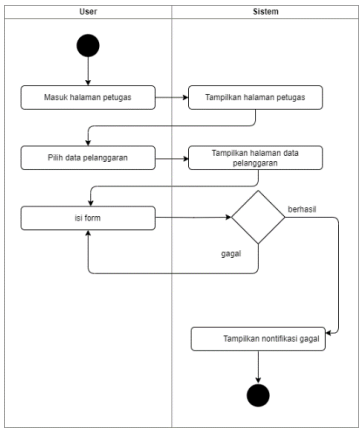
2) *Activity diagram* ubah *password* petugas



Gambar 4. 9 *Activity diagram* ubah *password*

Pada gambar 4.9 menjelaskan cara ubah password *user* / petugas dimana *user* terlebih dahulu memilih menu *profile* kemudian system akan menampilkan halaman profile dan user kemudian mengisi form ubah password yang nantinya akan di validasi oleh system dan akan menampilkan nontifikasi

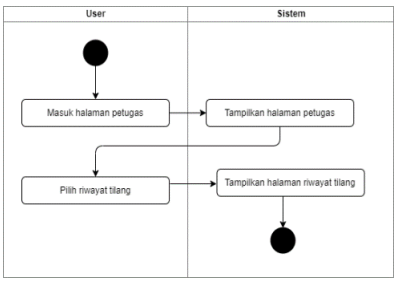
3) *Activity diagram* tambah data tilang



Gambar 4. 10 *Activity diagram* tambah data tilang

Pada gambar 4.10 menjelaskan cara *user* / petugas untuk menambahkan data tilang dimana terlebih dahulupetugas harus masuk terlebih dahulu ke halaman petugas kemudian memilih menu data tilang yang kemudian system akan menampilkan halaman data tilang dan selanjutnya petugas mengisi data tilang dan apabila berhasil maka akan muncul nontifikasi jika berhasil di tambahkan.

4) *Activity diagram* tampilkan riwayat pelanggaran

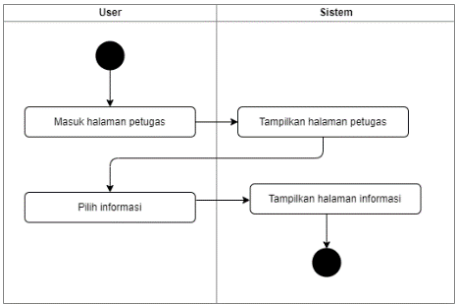


Gambar 4. 11 *Activity diagram* tampilkan riwayat pelanggaran

Pada gambar 4.11 menjelaskan cara *user* / petugas untuk menampilkan halaman riwayat pelanggaran dimana terlebih dahulu

petugas harus masuk ke halaman petugas dan selanjutnya memilih menu riwayat pelanggaran dan selanjutnya aplikasi atau *system* akan menampilkan riwayat pelanggarannya.

5) *Activity diagram* tampilkan informasi



Gambar 4. 12 *Activity diagram* tampilkan informasi

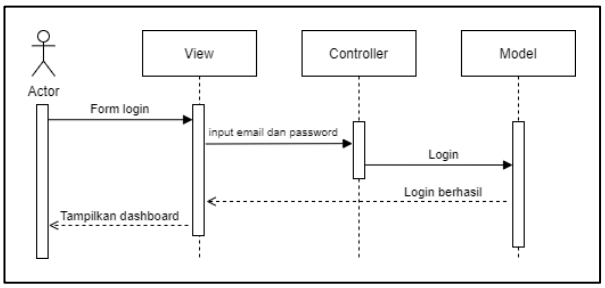
Pada gambar 4.12 menjelaskan cara *user* / petugas untuk menampilkan halaman informasi dimana terlebih dahulu petugas harus masuk ke halaman petugas dan selanjutnya memilih menu informasi dan selanjutnya aplikasi atau *system* akan menampilkan informasi.

2. *Sequence diagram*

Sequence diagram merupakan diagram yang menggambarkan interaksi antara objek-objek dalam sistem secara berurutan berdasarkan waktu. Diagram ini menunjukkan bagaimana objek-objek saling bertukar pesan atau data untuk menyelesaikan suatu tugas atau *use case*.

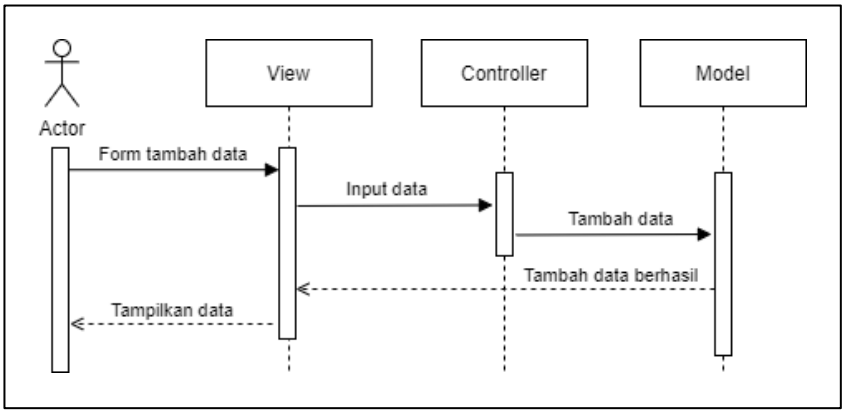
a. *Sequence diagram admin*

1) *Sequence diagram login*



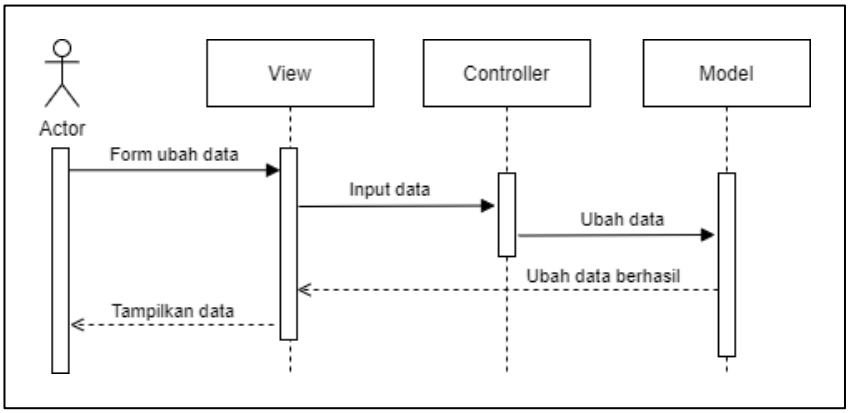
Gambar 4. 13 *Sequence diagram login admin*

2) *Sequence diagram* tambah data



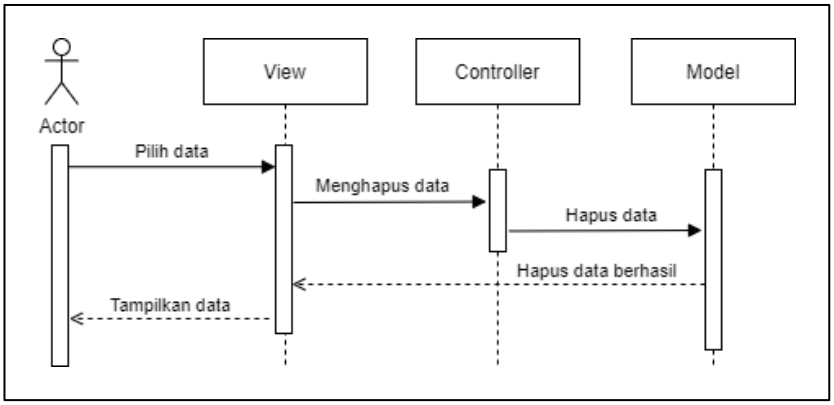
Gambar 4. 14 *Sequence diagram* tambah data

3) *Sequence diagram* ubah data



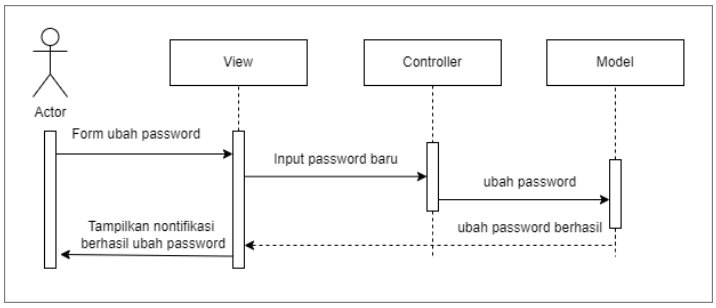
Gambar 4. 15 *Sequence diagram* ubah data

4) *Sequence diagram* hapus data



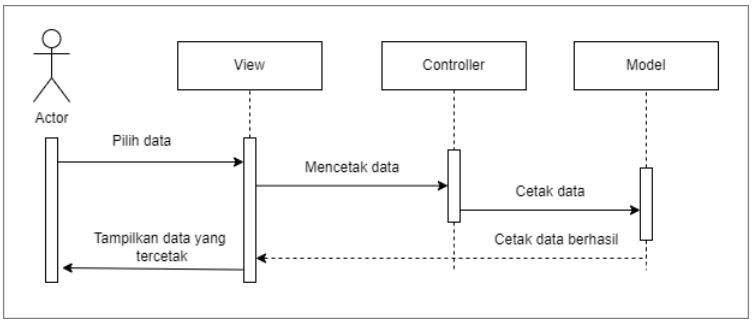
Gambar 4. 16 *Sequence diagram* hapus data

5) *Sequence diagram* ubah *password admin*



Gambar 4. 17 *Sequence diagram* ubah *password admin*

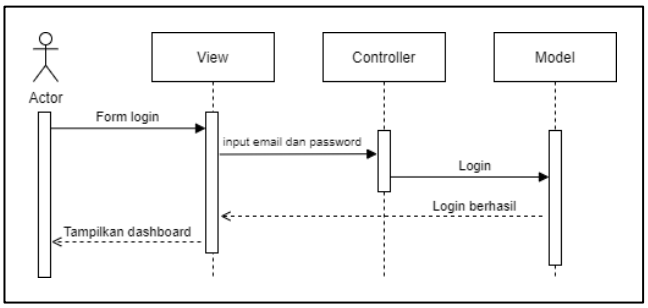
6) *Sequence diagram* cetak data



Gambar 4. 18 *Sequence diagram* cetak data

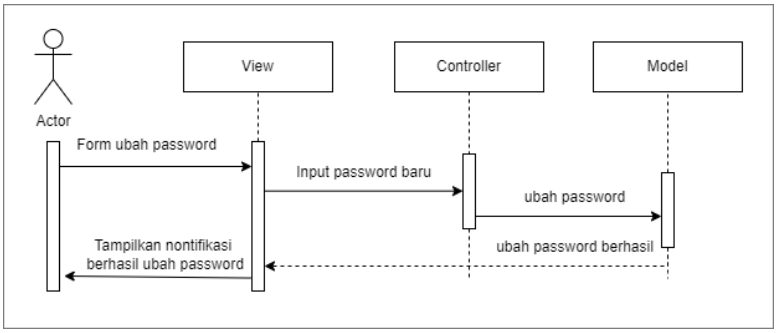
b. *Sequence diagram user*

1) *Sequence diagram* login



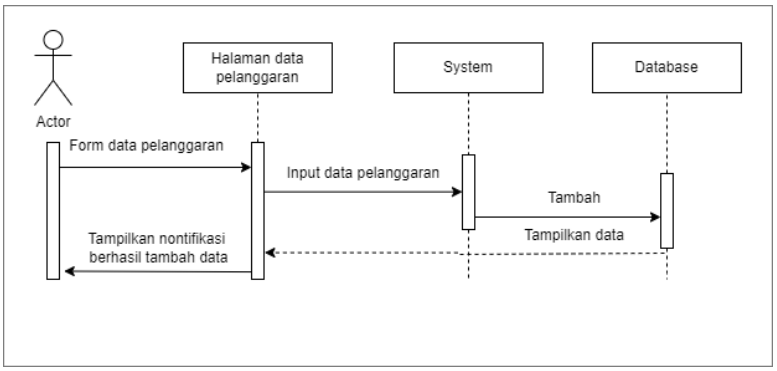
Gambar 4. 19 *Sequence diagram login user*

2) *Sequence diagram* ubah *password* petugas



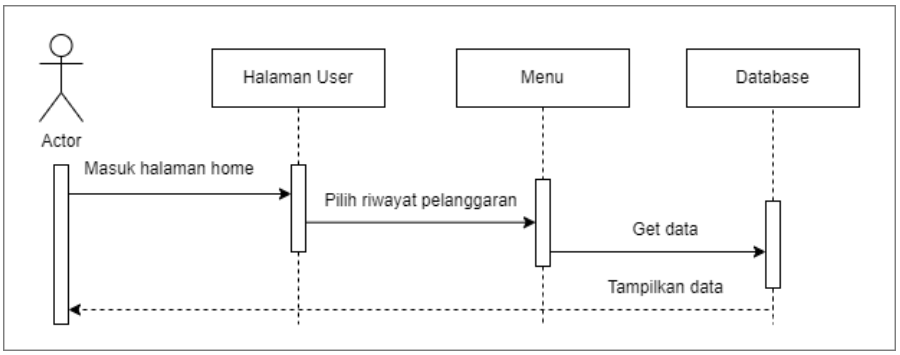
Gambar 4. 20 *Sequence diagram* ubah password petugas

3) *Sequence diagram* tambah data tilang



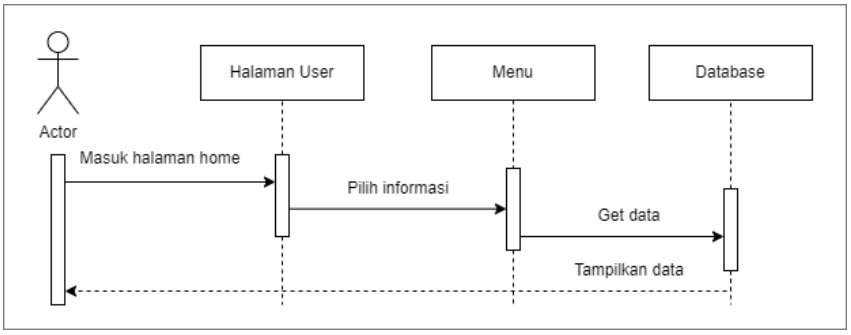
Gambar 4. 21 *Sequence diagram* tambah data tilang

4) *Sequence diagram* tampilkan riwayat pelanggaran



Gambar 4. 22 Sequence diagram tampilkan riwayat pelanggaran

5) *Sequence diagram* tampilkan informasi



Gambar 4. 23 *Sequence diagram* tampilkan informasi

B. Desain Tabel Database

Berikut ini adalah tabel-tabel yang digunakan pada rancang bangun aplikasi tilang elektronik berbasis android beserta strukturnya :

1. Tabel pasal

Tabel pasal adalah table yang digunakan untuk menyimpan data pasal

Tabel 4. 1 Tabel pasal

Field	Type	Key
id_pasal	char(11)	primary key
nama_pasal	varchar(100)	

keterangan	varchar(20)	

2. Tabel tb_bidang

Tabel tb_bidang adalah table yang digunakan untuk menyimpan data slip tilang

Tabel 4. 2 Tabel tb_bidang

Field	Type	Key
kode_kategori	char(11)	primary key
nama_bidang	varchar(50)	

3. Tabel tb_masyarakat

Tabel tb_masyarakat adalah table yang digunakan untuk menyimpan data polisi yang turun ke lapangan untuk melakukan proses penilangan

Tabel 4. 3 Tabel tb_masyarakat

Field	Type	Key
username	char(100)	primary key
nama	varchar(255)	
jenis_kelamin	num('L', 'P')	
tempat_lahir	varchar(50)	
tanggal_lahir	date	
alamat	text	
no_hp	varchar(20)	
password	varchar(255)	
foto_user	varchar(200)	
token	varchar(200)	

4. Tabel tb_pelanggaran

Tabel tb_pelanggaran adalah table yang digunakan untuk menyimpan data id pelanggaran,id laporan, dan id pasal

Tabel 4. 4 Tabel tb_pelanggaran

Field	Type	Key
id_pelanggaran	int(11)	primary key
id_laporan	varchar(100)	
id_pasal	char(11)	

5. Tabel tb_petugas

Tabel tb_petugas adalah table yang digunakan untuk menyimpan data polisi yang menjadi petugas admin

Tabel 4. 5 Tabel tb_petugas

Field	Type	Key
id_petugas	int(11)	primary key
nama_petugas	varchar(100)	
bagian	varchar(200)	
tempat_lahir	varchar(100)	
tanggal_lahir	date	
jenis_kelamin	enum('L', 'P')	
alamat	varchar(100)	
telp	varchar(20)	
username	varchar(20)	
password	varchar(20)	

6. Tabel tb_tilang

Tabel tb_tilang adalah table yang digunakan untuk menyimpan data penilangan yang telah di *input* oleh petugas polisi di lapangan

Tabel 4. 6 Tabel tb_tilang

Field	Type	Key
id_tilang	char(100)	primary key

jam	time	
tgl	date	
username	varchar(100)	
tgl_pelanggaran	date	
lati	varchar(100)	
longi	varchar(100)	
nama_pelanggar	varchar(100)	
alamat	varchar(100)	
no_hp	varchar(100)	
stnk	varchar(100)	
merk	varchar(100)	
plat	varchar(100)	
warna	varchar(100)	
denda	varchar(100)	
kode_kategori	varchar(100)	
keterangan	text	
jadwal	date	
lokasi	varchar(100)	
tujuan	varchar(100)	
gambar	varchar(100)	
status	varchar(100)	
id_tanggapan	char(100)	
signature	varchar(255)	
update_at	datetime	

7. Tabel tujuan_pembayaran

Tabel yang digunakan untuk menyimpan data tujuan pembayaran

Tabel 4. 7 Tabel tujuan_pembayaran

Field	Type	Key
id_tujuan	char(11)	primary key
nama_tujuan	varchar(100)	
keterangan_pembayaran	text	

C. Detail Sistem

1. Admin

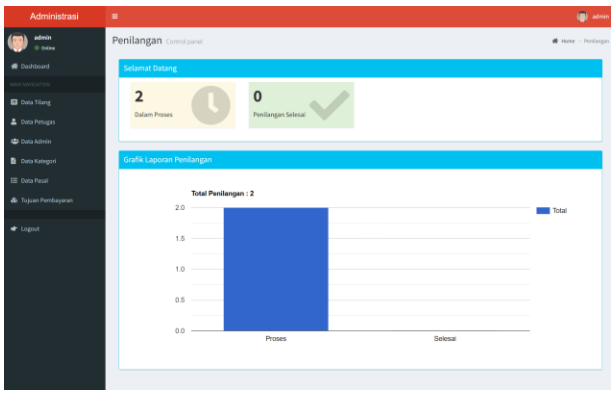
a) Halaman *login admin*

Halaman *login admin* merupakan halaman utama yang digunakan oleh *admin* untuk mendapatkan akses ke halaman *admin*.

Gambar 4. 24 halaman *login admin*

b) Halaman *dashboard admin*

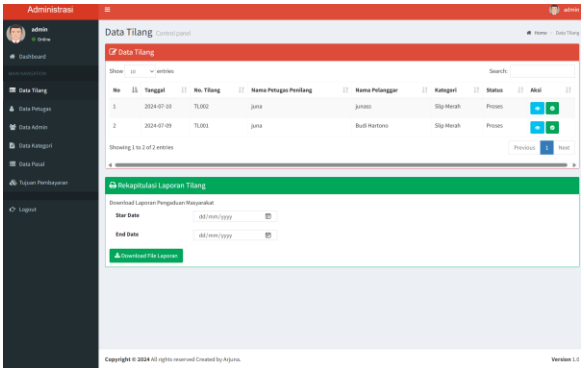
Halaman *dashboard admin* merupakan halaman yang menampilkan informasi mengenai jumlah data tilang baik yang masih berstatus diproses, dan selesai.



Gambar 4. 25 halaman *dashboard admin*

c) Halaman data tilang

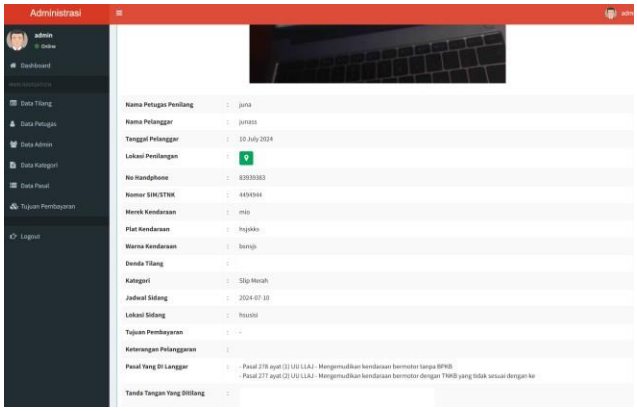
Halaman data tilang merupakan sebuah halaman yang menampilkan semua data tilang yang telah di *input* oleh petugas di lapangan atau *user* yang dimana dihalaman ini juga menampilkan tombol aksi untuk setujui atau di lihat data tilang.Di halaman ini juga ada tombol aksi untuk rekapitulasi data tilang.



Gambar 4. 26 halaman data tilang

d) Halaman lihat data tilang

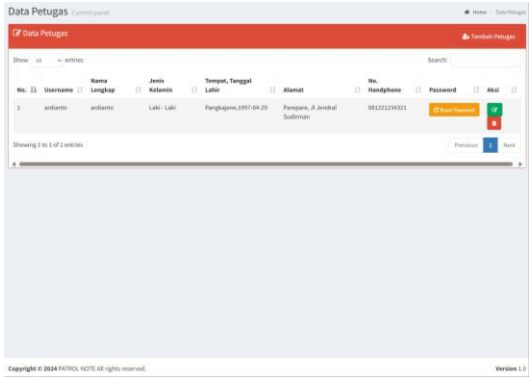
Halaman lihat data tilang ini merupakan halaman yang digunakan bagi admin untuk melihat data tilang yang telah di input oleh petugas. Di halaman ini berisi tombol atau aksi untuk mencetak data tilang yang telah di input oleh petugas. Di halaman ini juga menampilkan tombol aksi untuk mengirimkan nontifikasi ke pelanggar menggunakan *api* whatsapp dimana tujuannya berdasarkan nomor *handphone* dari pelanggar.



Gambar 4. 27 halaman lihat data tilang

e) Halaman data petugas

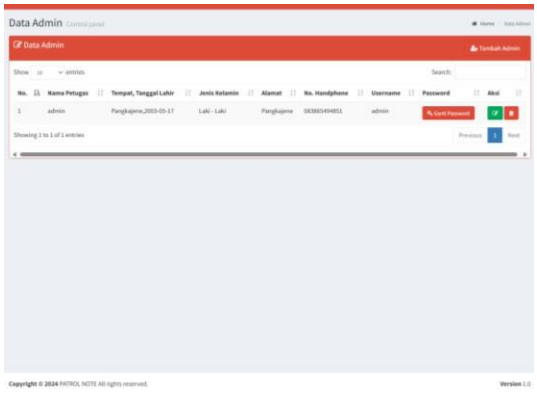
Halaman data petugas ini merupakan halaman bagi admin melihat data petugas serta merupakan halaman bagi admin untuk menambahkan, mengubah, serta menghapus data petugas.



Gambar 4. 28 halaman data petugas

f) Halaman data admin

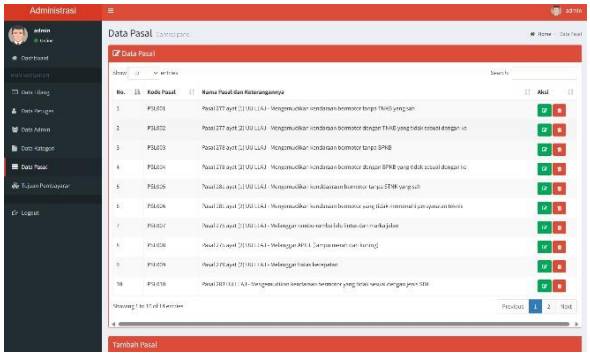
Halaman data *admin* merupakan sebuah halaman bagi admin untuk melihat informasi akun dari admin.Di halaman ini ada sebuah tombol dengan fungsi yang akan memunculkan pop up dengan fitur untuk menambah admin, ganti password, edit data admin, serta menghapus data admin.



Gambar 4. 29 halaman data *admin*

g) Halaman data pasal

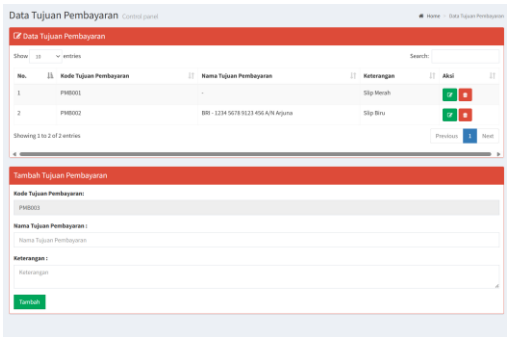
Halaman data pasalini merupakan halaman bagi admin untuk melihat semua nama-nama pasal yang berkaitan dengan lalu lintas beserta keterangannya.Di halaman ini juga admin bisa melakukan penambahan, penngeditan, serta penghapusan pasal pelanggaran lalu lintas.



Gambar 4. 30 halaman data pasal

h) Halaman tujuan pembayaran

Halaman ini merupakan halaman bagi admin untuk melihat tujuan pembayaran beserta keterangannya.Di halaman ini juga admin bisa melakukan penambahan, penngeditan, serta penghapusan tujuan pembayaran.



Gambar 4. 31 halaman tujuan pembayaran

2. User

a) Halaman login *user* / petugas

Halaman *login user* merupakan halaman utama yang digunakan oleh petugas untuk mendapatkan akses ke halaman beranda petugas.

Gambar 4. 32 halaman *login user*

b) Halaman beranda *user* / petugas

Halaman beranda ini muncul ketika *user* memasukkan username dan password dengan benar.Di halaman ini akan ditampilkan menu-menu yang bisa di akses oleh petugas lapangan seperti menu *profile*, pelanggaran, riwayat pelanggaran, dan informasi

Gambar 4. 33 halaman beranda user

c) Halaman *profile*

Halaman *profile* ini adalah halaman yang digunakan petugas untuk mengedit informasi pribadi petugas seperti nama, jenis-kelamin, alamat, dan lain-lain. Di halaman ini juga ada sebuah tombol yang akan mengarahkan kita ke halaman ganti *password* petugas.

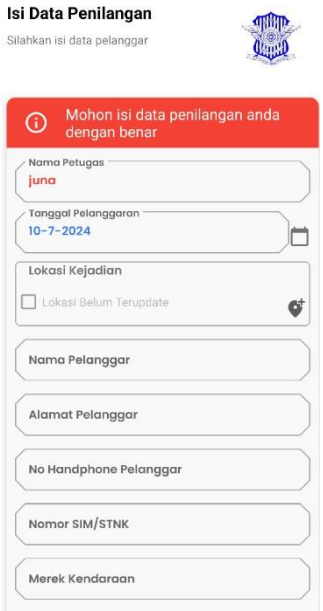
Gambar 4. 34 halaman *profile*

d) Halaman ganti *password user* / petugas Halaman ini merupakan halaman yang digunakan petugas untuk mengganti kata sandinya apabila dia melupa kata sandinya.

Gambar 4. 35 halaman ganti *password user*

e) Halaman tilang

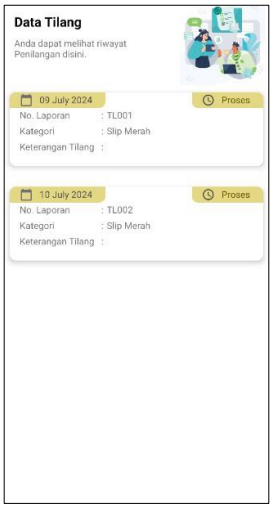
Halaman ini merupakah halaman yang digunakan petugas untuk menambahkan data penilangan seperti identitas pelanggar, identitas kendaraan, jenis pelanggaran, serta dokumentasi pelanggaran yang berupa foto .



Gambar 4. 36 halaman tilang

f) Halaman riwayat tilang

Halaman ini merupakan halaman yang di gunakan petugas yang berada di lapangan untuk melihat riwayat data tilang yang telah di di *input* sebelumnya serta di halaman ini juga petugas bisa melihat tanggapan yang telah di kirimkan oleh admin kepada petugas



Gambar 4. 37 halaman riwayat tilang

g) Halaman informasi

Halaman ini merupakan halaman yang di gunakan petugas melihat informasi dari aplikasi ini.

Gambar 4. 38 halaman informasi

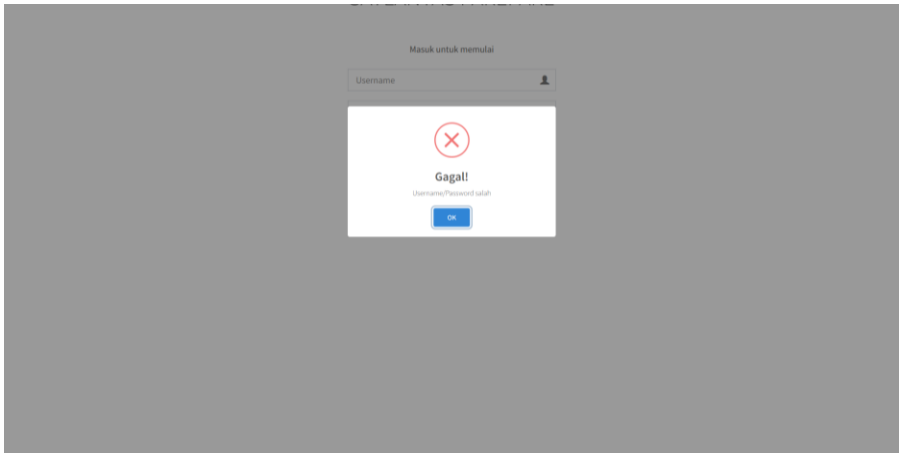
D. Pengujian Sistem

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. Black box

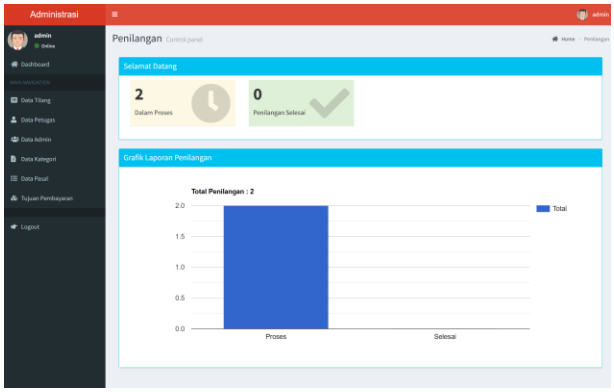
a. *Black box testing* kesalahan *email* dan *password* pada *admin*

Tabel 4. 8 *Black box testing* kesalahan *email* dan *password*

Tes faktor	Hasil	Keterangan
Memasukkan *email* atau *password* yang tidak sesuai pada halaman login *admin*	✓	Berhasil, ketika *email* atau *password* tidak sesuai tampil *login failed*
Screenshot		
		

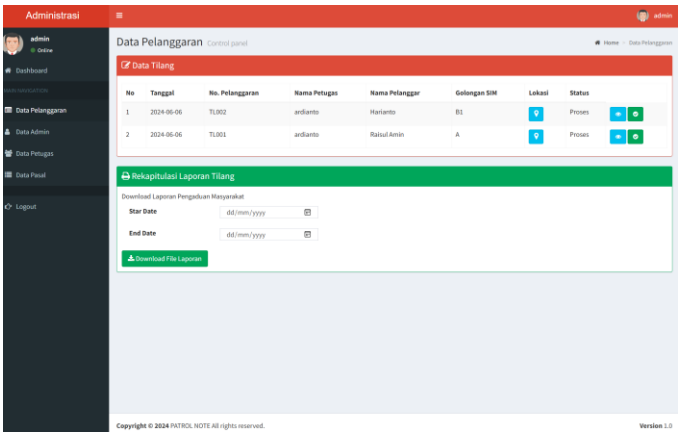
b. *Black box testing login* berhasil pada halaman *admin*

Tabel 4. *9 Black box testing login* berhasil

Tes faktor	Hasil	Keterangan
Memasukkan *email* atau *password* yang benar pada halaman *login admin*	✓	Sistem berhasil menampilkan halaman admin/dashboard.
Screenshot		
		

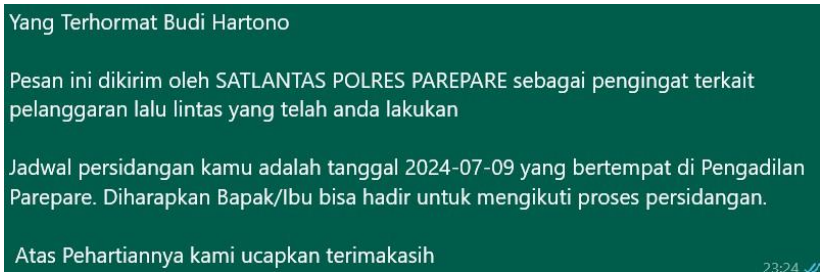
c. *Black box testing* halaman data tilang

Tabel 4. 10 *Black box testing* halaman data tilang

Tes faktor	Hasil	Keterangan
Admin menekan menu data tilang	✓	Sistem berhasil menampilkan halaman data tilang.
Screenshot		
		

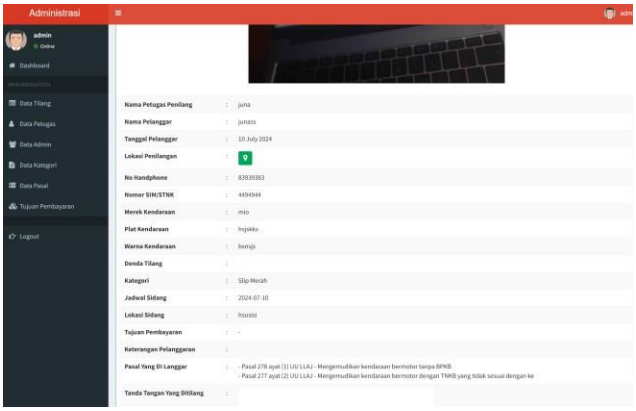
d. *Black box testing* nontifikasi pelanggaran

Tabel 4. 11 *Black box* testing nontifikasi pelanggaran

Tes faktor	Hasil	Keterangan
Admin menekan tombol kirim nontifikasi	✓	Sistem akan mengirimkan nontifikasi berupa pesan whatsapp berdasarkan nomor *handphone* dari data pelanggar
Screenshot		
		

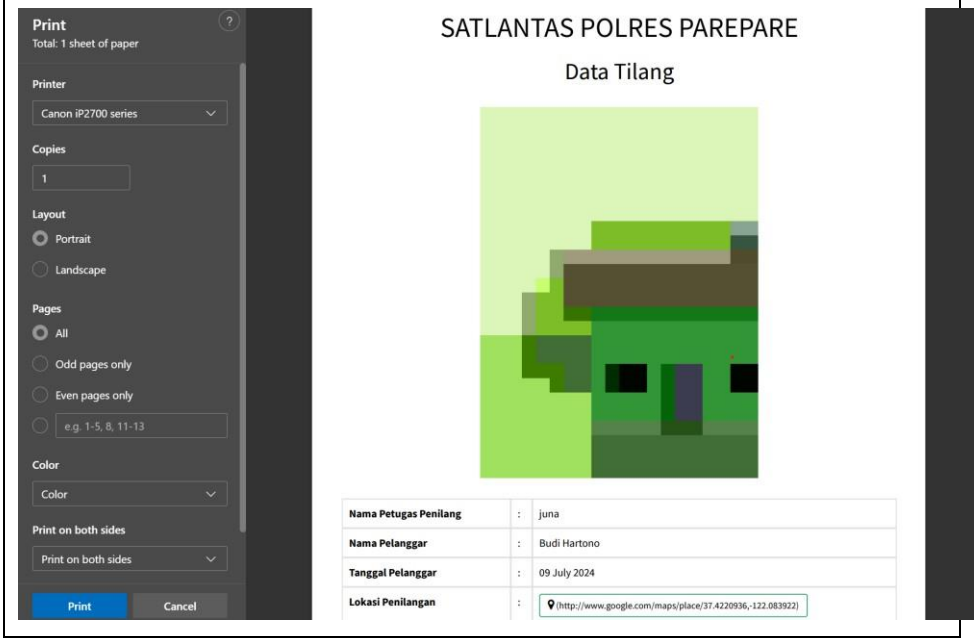
e. *Black box testing* halaman lihat data tilang

Tabel 4. *12 Black box testing* halaman lihat data tilang

Tes faktor	Hasil	Keterangan
Admin menekan tombol lihat data tilang	✓	Sistem berhasil menampilkan halaman lihat data tilang.
Screenshot		
		

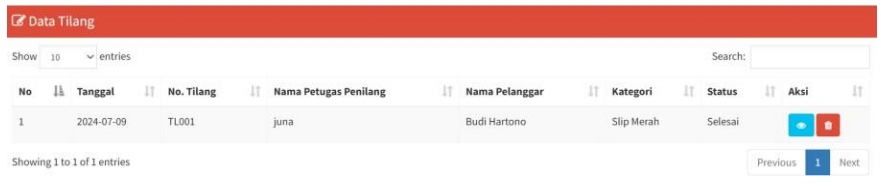
f. *Black box testing* cetak data tilang

Tabel 4. 13 *Black box testing* cetak data tilang

Tes faktor	Hasil	Keterangan
Admin menekan tombol cetak data tilang	✓	Sistem berhasil mencetak data tilang.
Screenshot		
		

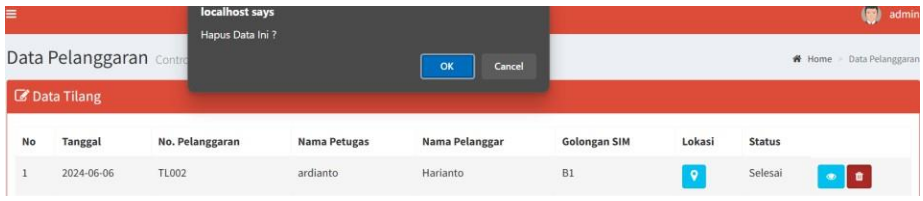
g. *Black box testing* finalisasi data tilang

Tabel 4. 14 *Black box testing* finalisasi data tilang

Tes faktor	Hasil	Keterangan
Admin menekan tombol selesai di halaman data tilang	✓	Sistem mengubah statusnya menjadi selesai serta memunculkan aksi untuk hapus
Screenshot		
		

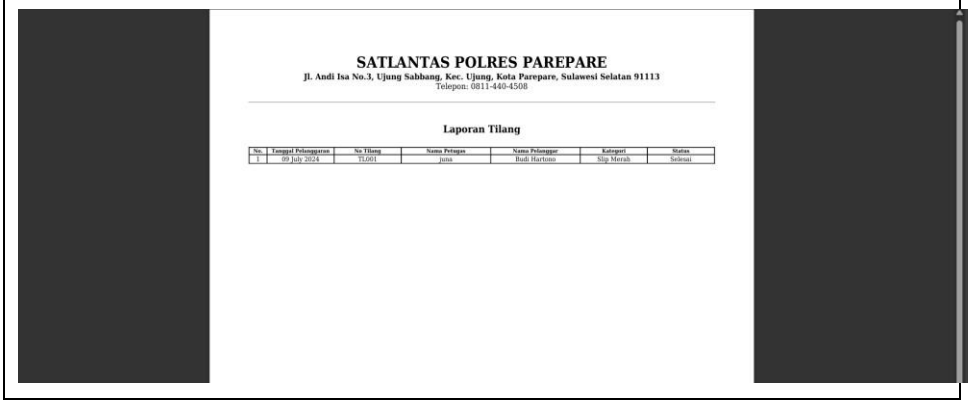
h. *Black box testing* hapus data tilang

Tabel 4. 15 *Black box testing* hapus data tilang

Tes faktor	Hasil	Keterangan
Admin menekan tombol hapus di halaman data tilang	✓	Sistem memunculkan *popup* hapus data dan data tilang yang sebelumnya ada kini telah hilang
Screenshot		
		

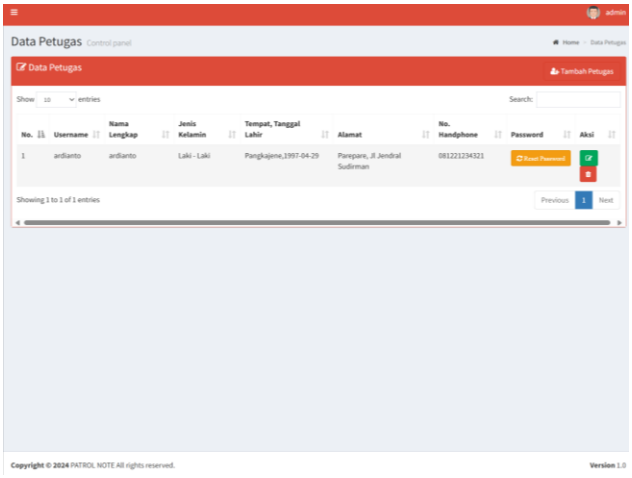
i. *Black box testing* download rekapitulasi data tilang

Tabel 4. 16 *Black box testing download* rekapitulasi data tilang

Tes faktor	Hasil	Keterangan
Admin memilih rentang waktu yang mau di rekap kemudian menekan tombol *download* file laporan	✓	Sistem mengunduh file rekap
Screenshot		
		

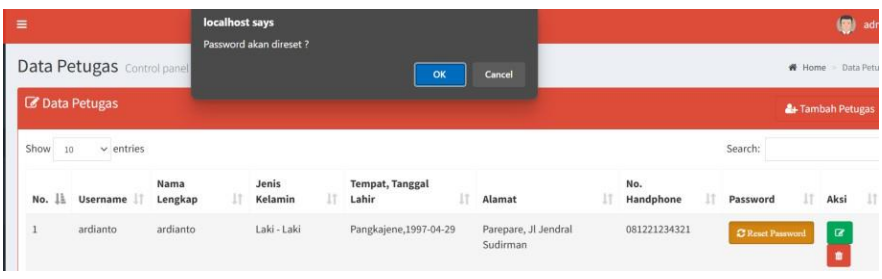
j. *Black box testing* halaman data petugas

Tabel 4. 17 Black box testing halaman data petugas

Tes faktor	Hasil	Keterangan
Admin menekan menu data petugas	✓	Sistem berhasil menampilkan halaman data petugas.
Screenshot		
		

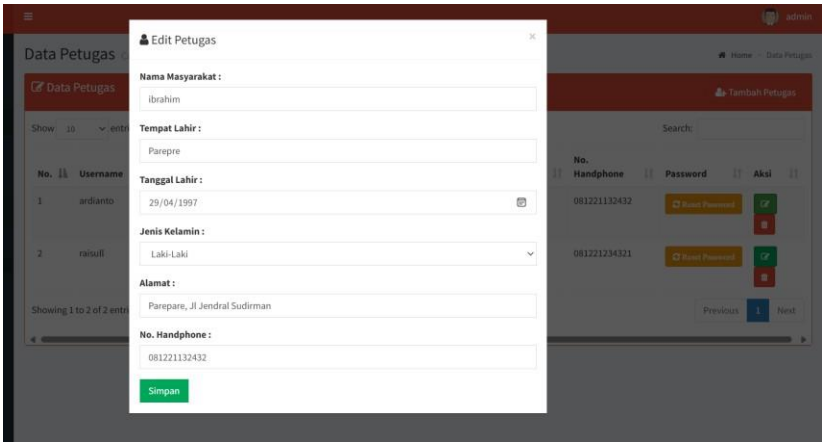
k. *Black box testing* reset *password* petugas

Tabel 4. 18 *Black box testing reset* password petugas

Tes faktor	Hasil	Keterangan
Admin menekan tombol *reset password* di halaman data petugas	✓	Sistem akan memunculkan *pop up* untuk *mereset password* dimana password default untuk petugas adalah 12345
Screenshot		
		

l. *Black box testing* menampilkan popup ubah data petugas

Tabel 4. 19 *Black box testing* menampilkan *popup* ubah data petugas

Tes faktor	Hasil	Keterangan
Admin menekan tombol ubah data di halaman data petugas	✓	Sistem memunculkan *popup form* untuk melakukan pengeditan data petugas
Screenshot		
		

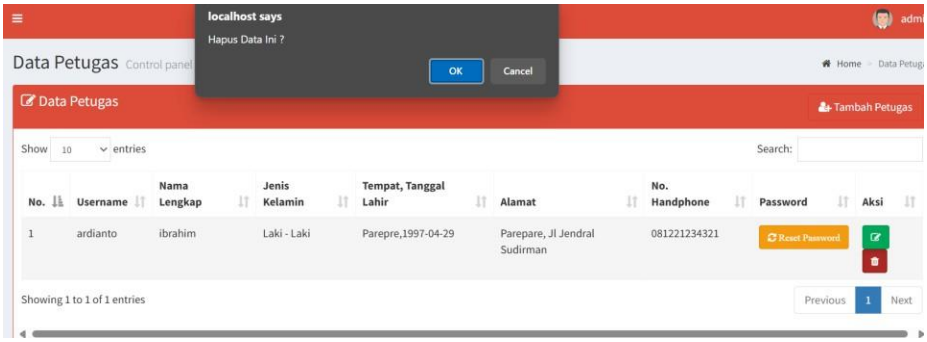
m. *Black box testing* mengubah data petugas

Tabel 4. 20 *Black box testing* mengubah data petugas

Tes faktor	Hasil	Keterangan
Admin mengisi perubahan data petugas di *form* ubah data petugas kemudian menekan tombol simpan	✓	Sistem berhasil mengubah data petugas
Screenshot		
		

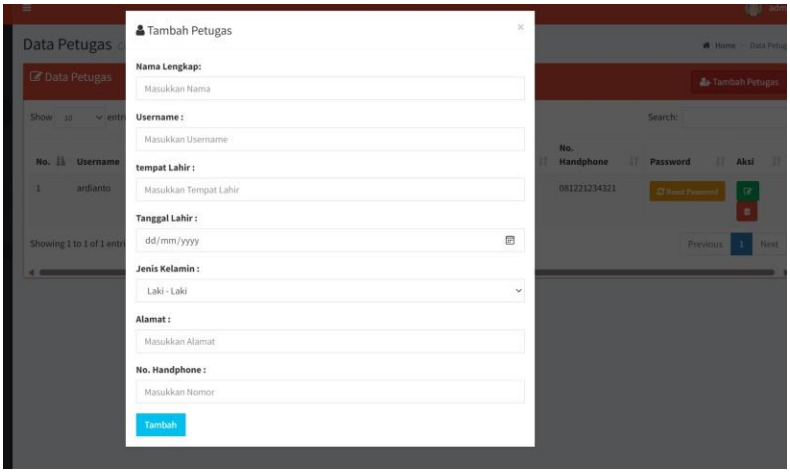
n. *Black box testing* menghapus data petugas

Tabel 4. 21 *Black box testing* menghapus data petugas

Tes faktor	Hasil	Keterangan
Admin menekan tombol hapus data di halaman data petugas	✓	Sistem akan memunculkan *pop up* untuk menghapus data petugas
Screenshot		
		

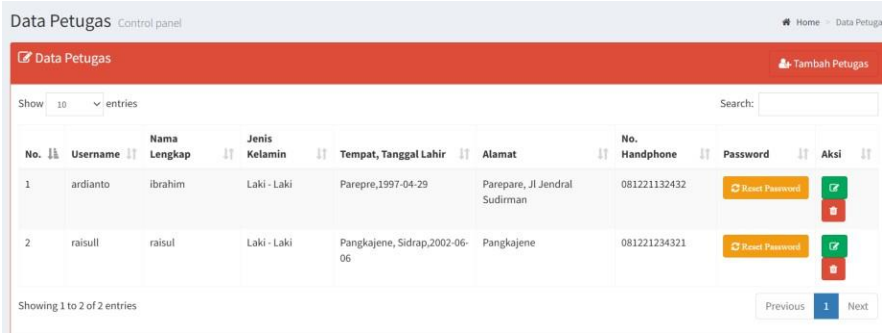
o. *Black box testing* memunculkan *popup* tambah petugas

Tabel 4. 22 *Black box testing* memunculkan *popup* tambah petugas

Tes faktor	Hasil	Keterangan
Admin menekan tombol tambah petugas di halaman data petugas	✓	Sistem memunculkan *popup* tambah data petugas
Screenshot		
		

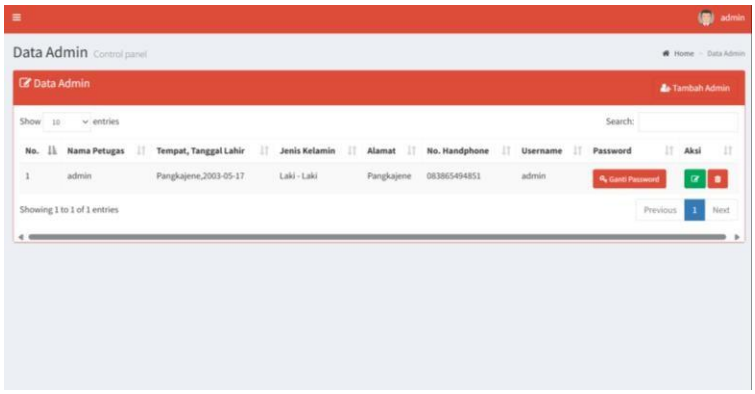
p. *Black box testing* tambah petugas

Tabel 4. 23 *Black box testing* tambah petugas

`Tes faktor	Hasil	Keterangan
Admin mengisi data petugas di *form* tambah petugas kemudian menekan tombol tambah	✓	Sistem berhasil menambah data petugas dan tersimpan di database
Screenshot		
		

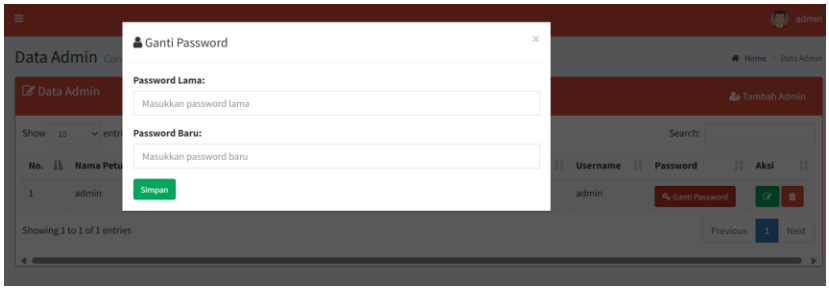
q. *Black box testing* halaman data admin

Tabel 4. 24 *Black box testing* halaman data admin

Tes faktor	Hasil	Keterangan
Admin menekan menu data *admin*	✓	Sistem berhasil menampilkan halaman data *admin.*
Screenshot		
		

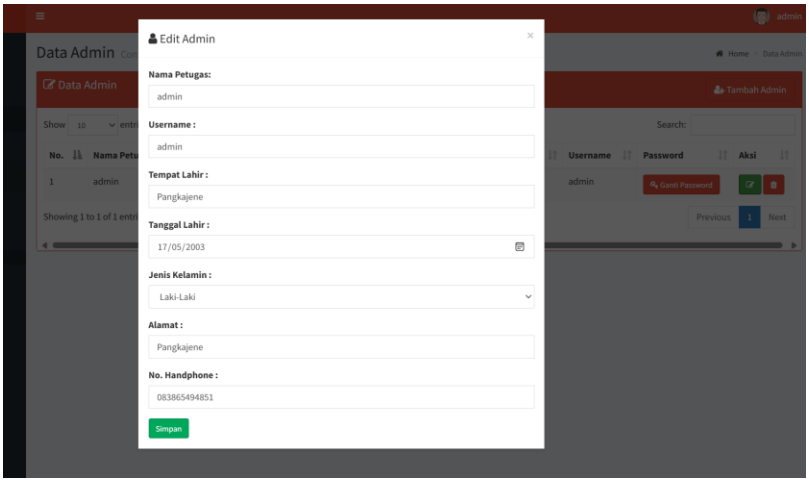
r. *Black box testing* ganti *password admin*

Tabel 4. 25 *Black box testing* ganti *password* petugas

Tes faktor	Hasil	Keterangan
Admin menekan tombol ganti *password* di halaman data *admin* kemudian memasukkan *password* lama dan *password* barunya	✓	Sistem berhasil mengganti *password* petugas *admin* di *database* dan memunculkan pop up berhasil
Screenshot		
		

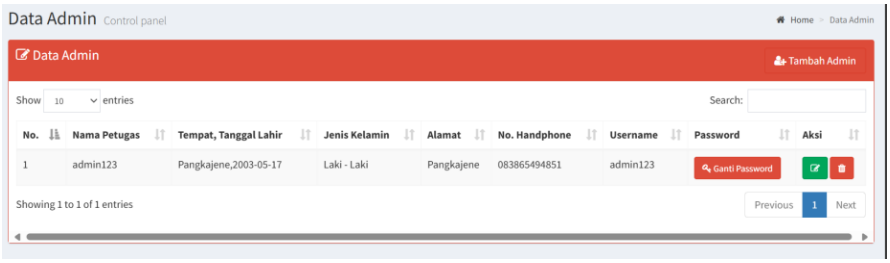
s. *Black box testing* menampilkan popup edit data *admin*

Tabel 4. 26 *Black box testing* menampilkan *popup* edit data *admin*

Tes faktor	Hasil	Keterangan
Admin menekan tombol ubah data di halaman data admin	✓	Sistem memunculkan *popup form* untuk melakukan pengeditan data admin
Screenshot		
		

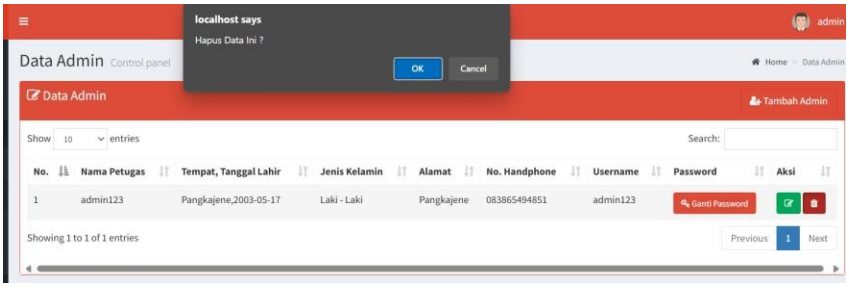
t. *Black box testing* mengubah data *admin*

Tabel 4. 27 *Black box testing* mengubah data *admin*

Tes faktor	Hasil	Keterangan
Admin mengisi perubahan data di *form* ubah data *admin* kemudian menekan tombol simpan	✓	Sistem berhasil mengubah data *admin*
Screenshot		
		

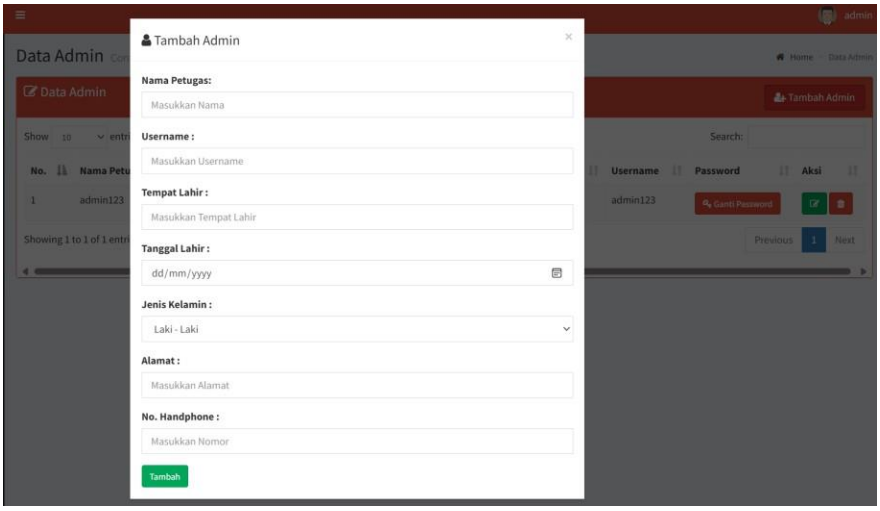
u. *Black box testing* menghapus data admin

Tabel 4. 28 *Black box testing* menghapus data *admin*

Tes faktor	Hasil	Keterangan
Admin menekan tombol hapus data di halaman data *admin*	✓	Sistem memunculkan *pop up* untuk menghapus data *admin* dan apabila di pilih Ok maka akan berhasil menghapus data *admin* di *database*
Screenshot		
		

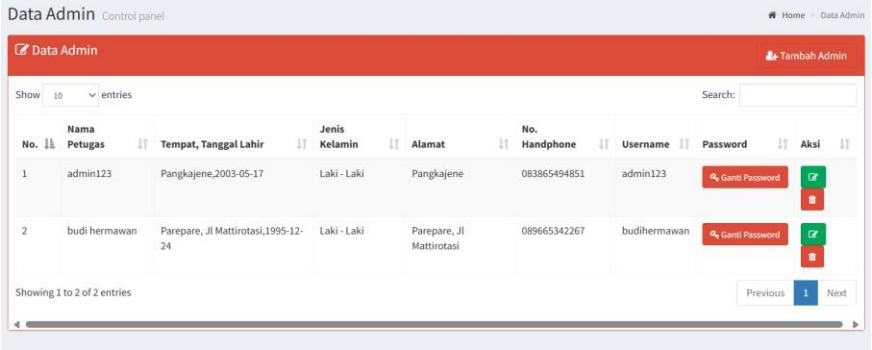
v. *Black box testing memunculkan popup* tambah admin

Tabel 4. 29 *Black box testing* memunculkan *popup* tambah *admin*

Tes faktor	Hasil	Keterangan
Admin menekan tombol tambah admin di halaman data *admin*	✓	Sistem memunculkan *popup* tambah data *admin*
Screenshot		
		

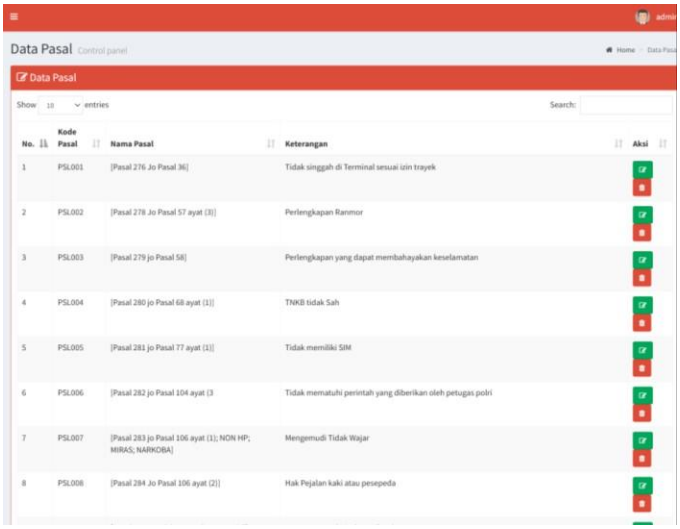
w. *Black box testing* tambah *admin*

Tabel 4. 30 *Black box testing* tambah *admin*

Tes faktor	Hasil	Keterangan
Admin mengisi data *admin* di *form* tambah *admin* kemudian menekan tombol tambah	✓	Sistem berhasil menambahkan data *admin* di *database*
Screenshot		
		

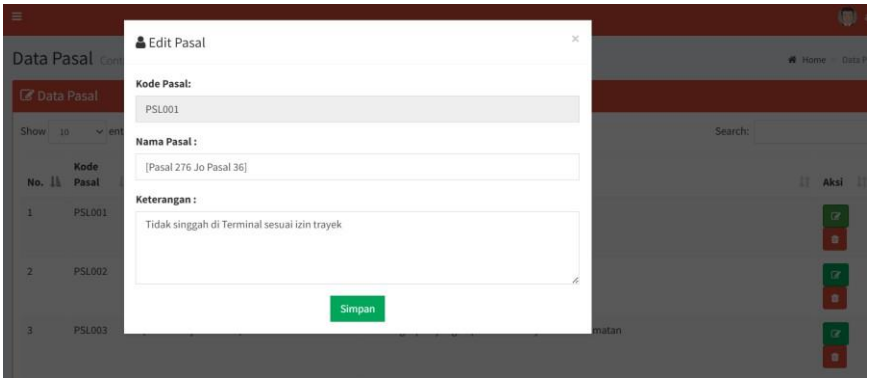
x. *Black box testing halaman* pasal

Tabel 4. 31 *Black box testing* halaman pasal

Tes faktor	Hasil	Keterangan
Admin menekan menu data pasal	✓	Sistem berhasil menampilkan halaman pasal.
Screenshot		
		

y. *Black box testing* menampilkan popup edit pasal

Tabel 4. 32 *Black box testing* menampilkan *popup* edit pasal

Tes faktor	Hasil	Keterangan
Admin menekan tombol ubah data di halaman data pasal	✓	Sistem memunculkan *popup form* untuk melakukan pengeditan pasal
Screenshot		
		

z. *Black box testing* mengubah pasal

Tabel 4. 33 *Black box testing* mengubah pasal

Tes faktor	Hasil	Keterangan
Admin mengisi perubahan data di *form* ubah pasal kemudian menekan tombol simpan	✓	Sistem berhasil mengubah data pasal
Screenshot		
		

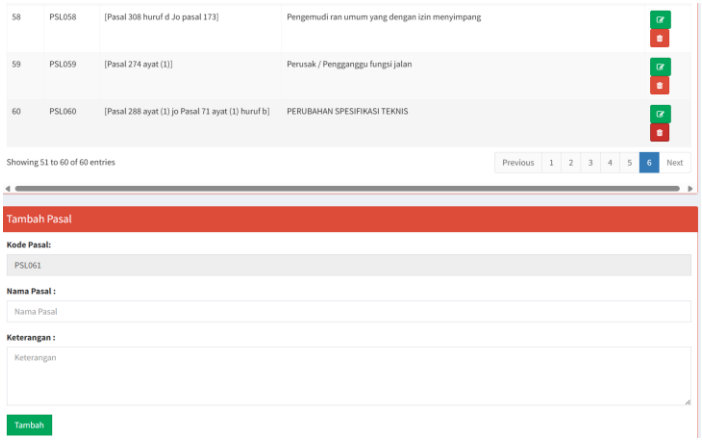
aa. *Black box testing menghapus* pasal

Tabel 4. 34 *Black box testing* menghapus pasal

Tes faktor	Hasil	Keterangan
Admin menekan tombol hapus di halaman data pasal	✓	Sistem akan memunculkan *pop up* untuk menghapus pasal dan apabila di klik OK maka berhasil menghapus data pasal di database
Screenshot		
		

bb. *Black box testing tambah pasal*

Tabel 4. 35 *Black box testing* tambah pasal

Tes faktor	Hasil	Keterangan
Admin mengisi data pasal di *form* tambah pasal kemudian menekan tombol tambah	✓	Sistem berhasil menambah data
Screenshot		
		

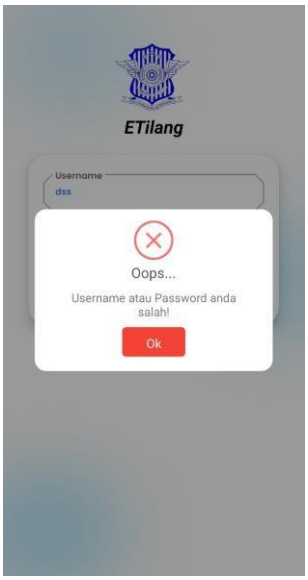
cc. *Black box testing logout*

Tabel 4. 36 *Black box testing* ***l****ogout*

Tes faktor	Hasil	Keterangan
Admin menekan tombol logout	✓	Sistem mengeluarkan *admin* dari sesi serta melakukan *redirect* ke halaman *login admin*
Screenshot		
		

dd. *Black box testing* kesalahan *username* dan *password* pada *user*

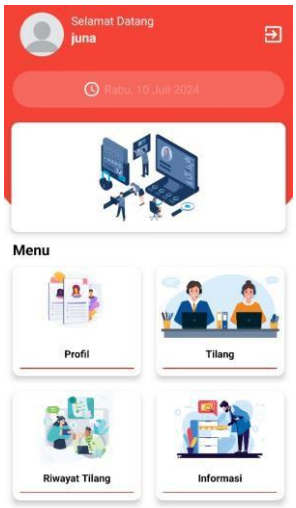
Tabel 4. 37 *Black box testing* kesalahan *username* dan *password* pada *user*

Tes faktor	Hasil	Keterangan
Memasukkan *username* atau *password* yang tidak sesuai pada halaman login *user*	✓	Berhasil, ketika *username* atau *password* tidak sesuai tampil *login failed*
Screenshot		
		

ee. *Black box testing login* berhasil pada halaman *user*

Tabel 4. 38 *Black box testing login* berhasil pada halaman *user*

Tes faktor	Hasil	Keterangan
Memasukkan *username* atau *password* yang benar pada halaman *login user*	✓	Sistem berhasil menampilkan halaman *user/dashboard.*
Screenshot		



ff. *Black box testing* halaman *profile*

Tabel 4. 39 *Black box testing* halaman *profile*

Tes faktor	Hasil	Keterangan
User menekan menu *profile*	✓	Sistem berhasil menampilkan halaman *profile*.
Screenshot		

gg. *Black box testing* memunculkan halaman ganti *password user*

Tabel 4. 40 *Black box testing* memunculkan halaman ganti *password user*

Tes faktor	Hasil	Keterangan
User menekan tulisan ganti user di halaman *profile*	✓	Sistem berhasil menampilkan halaman ganti *password*.
Screenshot		
		

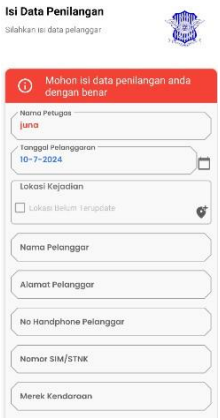
hh. *Black box testing* ganti *password user*

Tabel 4. 41 *Black box testing* ganti *password user*

Tes faktor	Hasil	Keterangan
User mengisi *password* baru di halaman ganti password dan klik simpan	✓	Sistem berhasil menyimpan password baru dan memunculkan *pop up* berhasil
Screenshot		
		

ii. *Black box testing* memunculkan halaman pelanggaran

Tabel 4. 42 *Black box testing* memunculkan halaman pelanggaran

Tes faktor	Hasil	Keterangan
User menekan menu pelanggaran	✓	Sistem berhasil menampilkan halaman pelanggaran.
Screenshot		
		

jj. *Black box testing* menambahkan data tilang

Tabel 4. 43 *Black box testing* menambagkan data tilang

Tes faktor	Hasil	Keterangan
User mengisi form data tilang dan menekan simpan	✓	Sistem berhasil menyimpan data tilang yang akan bisa dilihhat oleh *admin*.
Screenshot		
		

mm. *Black box testing* memunculkan halaman riwayat pelanggaran

Tabel 4. 44 *Black box testing* memunculkan halaman riwayat pelanggaran

Tes faktor	Hasil	Keterangan
User menekan menu riwayat pelanggaran	✓	Sistem berhasil menampilkan halaman pelanggaran.
Screenshot		
		

nn. *Black box testing* memunculkan halaman informasi

Tabel 4. 45 *Black box testing* memunculkan halaman informasi

Tes faktor	Hasil	Keterangan
User menekan menu informasi	✓	Sistem berhasil menampilkan halaman informasi.
Screenshot		
		

oo. Black box testing logout* pada *user

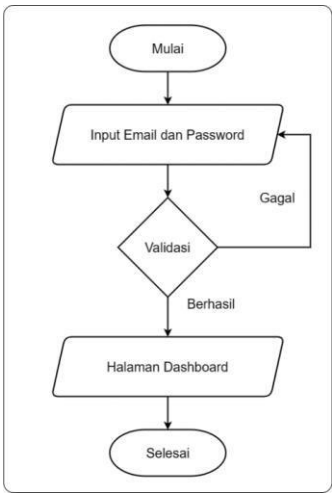
Tabel 4. 46 *Black box testing logout* pada *user*

Tes faktor	Hasil	Keterangan
User menekan tombol logout	✓	Sistem mengeluarkan *user* dari sesi serta melakukan *redirect* ke halaman *login user*
Screenshot		
		

2. White box

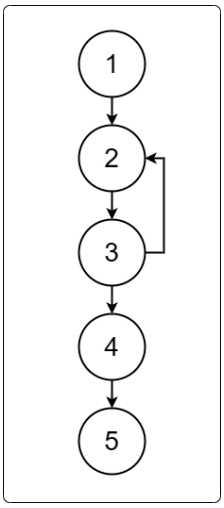
a. *White box testing* kesalahan *email* dan *password admin*

(1) *Flowchart*



Gambar 4. 39 *Flowchart* kesalahan *email* dan *password admin*

(2) *Flowgraph*



Gambar 4. 40 *Flowgraph* kesalahan *email* dan *password admin*

Berdasarkan gambar 4. 69 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

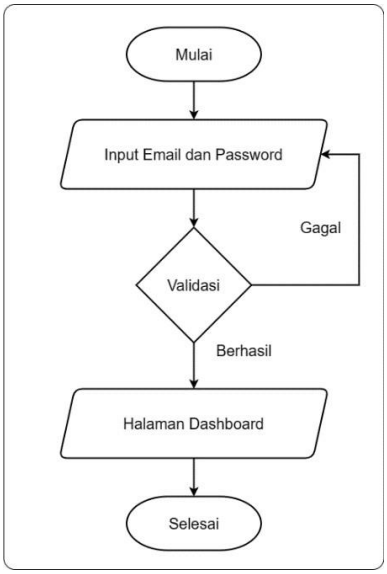
(4) Grafik matriks kesalahan *email* dan *password admin*

Tabel 4. 47 Grafik matriks kesalahan *email* dan *password admin*

	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

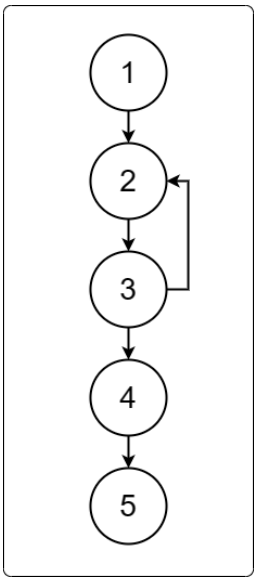
b. *White box testing* login berhasil *admin*

1) *Flowchart*



Gambar 4. 41 *Flowchart login* berhasil *admin*

2) *Flowgraph*



Gambar 4. 42 *Flowgraph login* berhasil *admin*

Berdasarkan gambar 4. 71 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

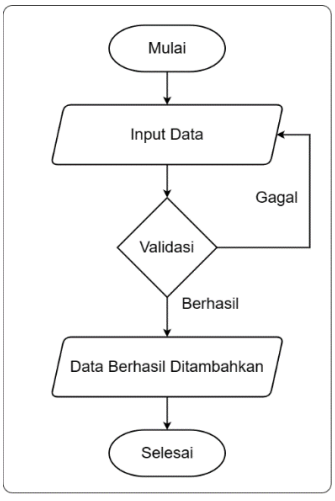
(4) Grafik matriks *login* berhasil *admin*

Tabel 4. 48 Grafik matriks *login* berhasil *admin*

	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

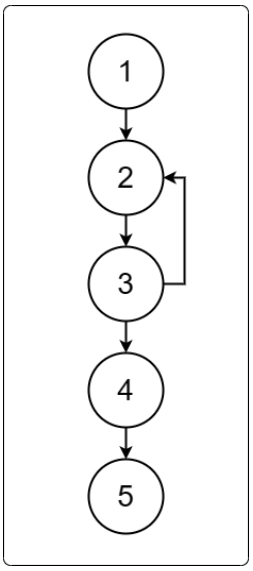
c. *White box testing* tambah data

1) *Flowchart*



Gambar 4. 43 *Flowchart* tambah data

2) *Flowgraph*



Gambar 4. 44 *Flowgraph* tambah data

Berdasarkan gambar 4. 73 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

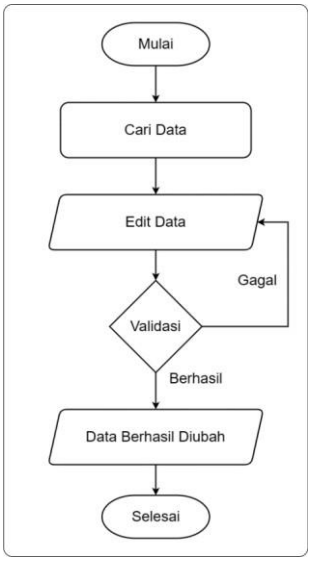
(4) Grafik matriks tambah data

Tabel 4. 49 Grafik matriks tambah data

	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

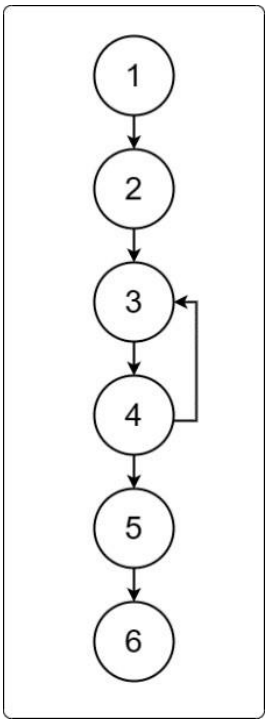
d. *White box testing* ubah data

1) *Flowchart*



Gambar 4. 45 *Flowchart* ubah data

2) *Flowgraph*



Gambar 4. 46 *Flowgraph* ubah data

Berdasarkan gambar 4. 75 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 6

N (*node*) = 6

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 6 – 6 + 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 – 4 – 3

Path 2 = 1 – 2 – 3 – 4 – 5 – 6

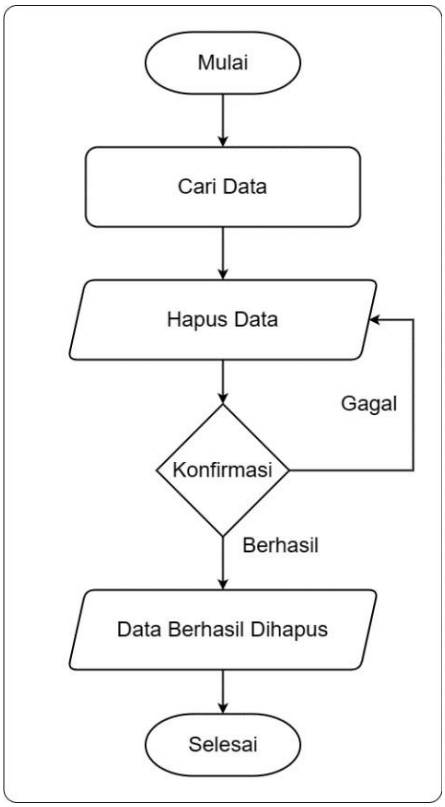
(4) Grafik matriks ubah data

Tabel 4. 50 Grafik matriks ubah data

	1	2	3	4	5	6	E-1
1		1					1 - 1 = 0
2			1				1 – 1 = 0
3				1			1 – 1 = 0
4			1		1		2 – 1 = 1
5						1	1 – 1 = 0
6							0
	SUM (E + 1)						1 + 1 = 2

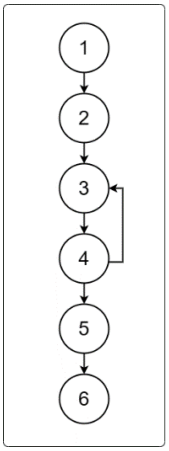
e. *White box testing* hapus data

1) *Flowchart*



Gambar 4. 47 *Flowchart* hapus data

2) *Flowgraph*



Gambar 4. 48 *Flowgraph* hapus data

Berdasarkan gambar 4. 77 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 6

N (*node*) = 6

P (Predikat *node*) = 1

Penyelesaian :

V(G) = E – N + 2

= 6 – 6 + 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 – 4 – 3

Path 2 = 1 – 2 – 3 – 4 – 5 – 6

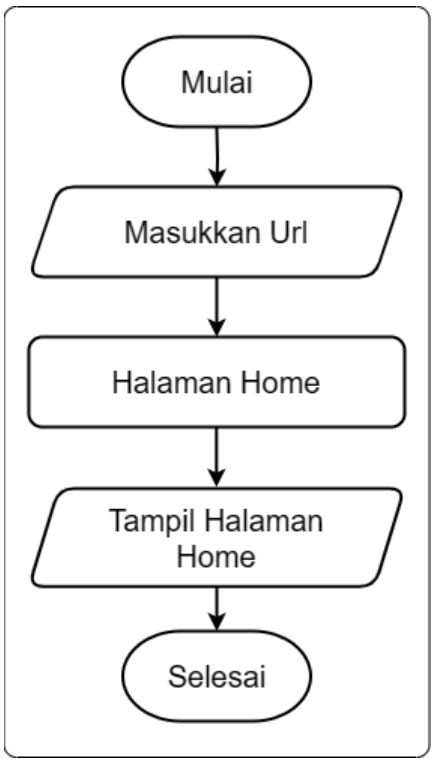
(4) Grafik matriks hapus data

Tabel 4. 51 Grafik matriks hapus data

	1	2	3	4	5	6	E-1
1		1					1 - 1 = 0
2			1				1 – 1 = 0
3				1			1 – 1 = 0
4			1		1		2 – 1 = 1
5						1	1 – 1 = 0
6							0
	SUM (E + 1)						1 + 1 = 2

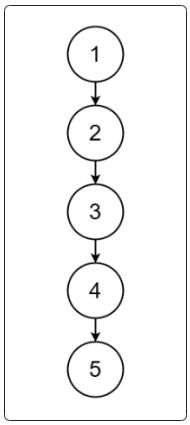
f. *White box testing* halaman beranda/*home*

1) *Flowchart*



Gambar 4. 49 *Flowchart* halaman beranda/*home*

2) *Flowgraph*



Gambar 4. 50 *Flowgraph* halaman beranda/*home*

Berdasarkan gambar 4. 79 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 1*

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

Path 1 = 1 – 2 – 3 – 4 – 5

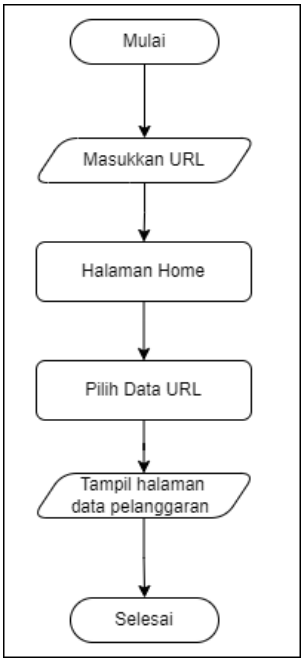
(4) Grafik matriks halaman beranda/*home*

Tabel 4. 52 Grafik matriks halaman beranda/*home*

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

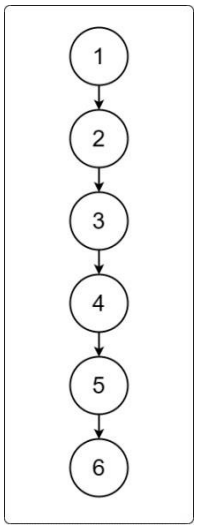
g. *White box testing* halaman data tilang

1) *Flowchart*



Gambar 4. 51 *Flowchart* halaman data tilang

2) *Flowgraph*



Gambar 4. 52 *Flowgraph* halaman data tilang

Berdasarkan gambar 4. 81 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 1*

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

Path 1 = 1 – 2 – 3 – 4 – 5

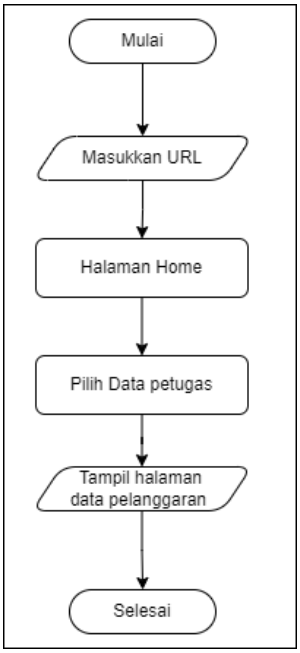
(4) Grafik matriks halaman data tilang

Tabel 4. 53 Grafik matriks halaman data tilang

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

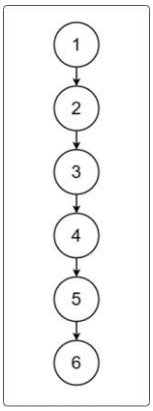
h. *White box testing* halaman data petugas

1) *Flowchart*



Gambar 4. 53 *Flowchart* halaman data petugas

2) *Flowgraph*



Gambar 4. 54 *Flowgraph* halaman data petugas

Berdasarkan gambar 4. 81 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 1*

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

Path 1 = 1 – 2 – 3 – 4 – 5

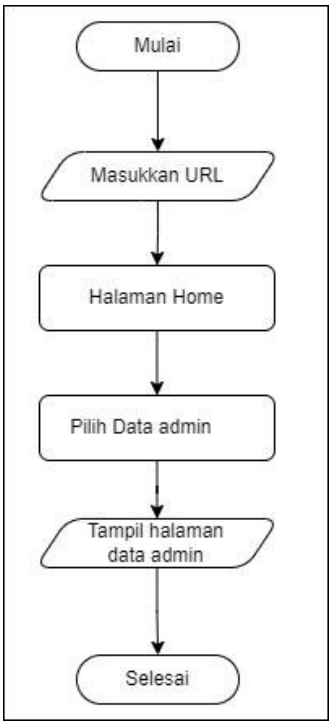
(4) Grafik matriks halaman data petugas

Tabel 4. 54 Grafik matriks halaman data petugas

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

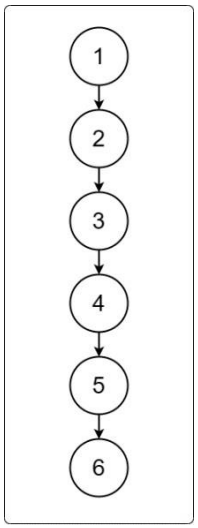
i. *White box testing* halaman data *admin*

1) *Flowchart*



Gambar 4. 55 *Flowchart* halaman data *admin*

2) *Flowgraph*



Gambar 4. 56 *Flowgraph* halaman data *admin*

Berdasarkan gambar 4. 81 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 1*

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

Path 1 = 1 – 2 – 3 – 4 – 5

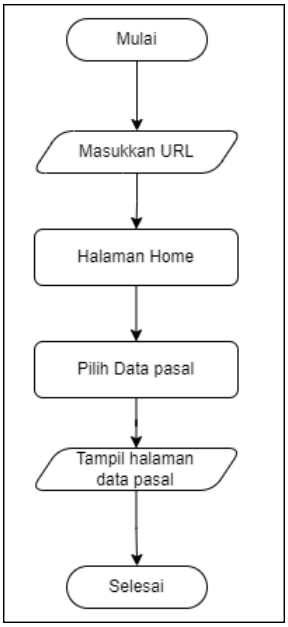
(4) Grafik matriks halaman data *admin*

Tabel 4. 55 Grafik matriks halaman data *admin*

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

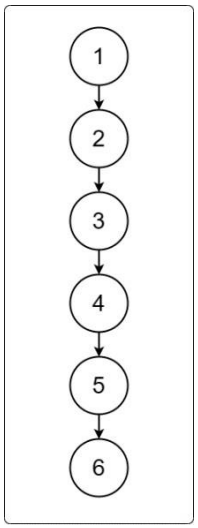
j. *White box testing* halaman data pasal

1) *Flowchart*



Gambar 4. 57 *Flowchart* halaman data pasal

2) *Flowgraph*



Gambar 4. 58 *Flowgraph* halaman data pasal

Berdasarkan gambar 4. 81 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 1*

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

Path 1 = 1 – 2 – 3 – 4 – 5

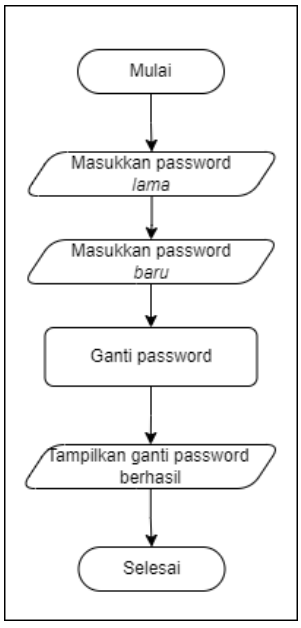
(4) Grafik matriks halaman data pasal

Tabel 4. 56 Grafik matriks halaman data pasal

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

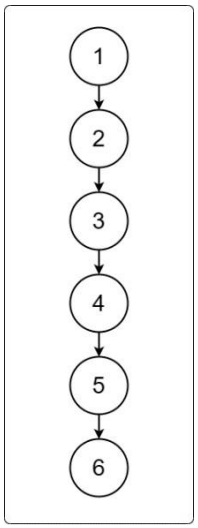
k. *White box testing* ganti *password admin*

1) *Flowchart*



Gambar 4. 59 *Flowchart* halaman ganti *password admin*

2) *Flowgraph*



Gambar 4. 60 *Flowgraph* halaman ganti *password admin*

Berdasarkan gambar 4. 81 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 1*

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

Path 1 = 1 – 2 – 3 – 4 – 5

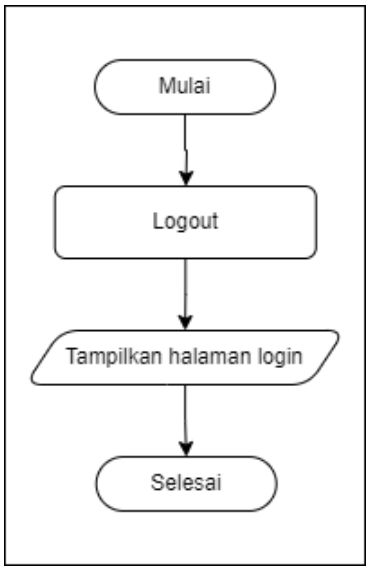
(4) Grafik matriks halaman ganti *password admin*

Tabel 4. 57 Grafik matriks halaman ganti *password admin*

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

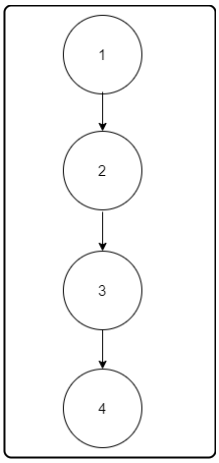
l. *White box testing* logout *admin*

1) *Flowchart*



Gambar 4. 61 *Flowchart* halaman *logout admin*

2) Flowgraph



Gambar 4. 62 *Flowgraph* Halaman *Logout admin*

Berdasarkan gambar 4. 36 diatas dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 3

N (*node*) = 4

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 3 – 4 + 2

= 2

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *Cyclomatic Complexcity* dari *flowgraph* di atas memiliki *Region* = 1

(3) *Independent path* pada *flowgraph* tersebut yakni:

Path 1 = 1 – 2 – 3 – 4

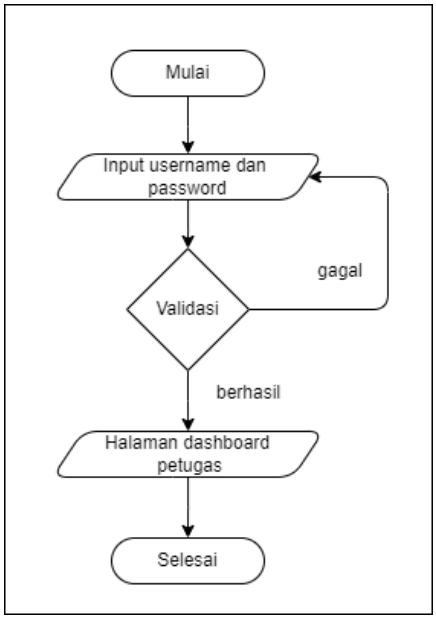
(4) Grafik matriks Halaman *logout admin*

Tabel 4. 58 Grafik Matriks halaman *logout admin*

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

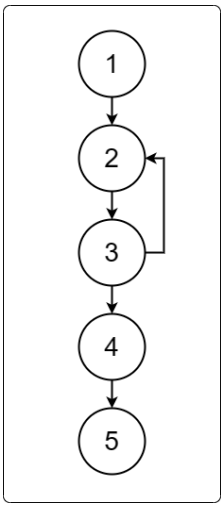
m. *White box testing* kesalahan login *user* / petugas

1) *Flowchart*



Gambar 4. 63 *Flowchart* kesalahan *login user*

(2) *Flowgraph*



Gambar 4. 64 *Flowgraph* kesalahan *login users*

Berdasarkan gambar 4. 69 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

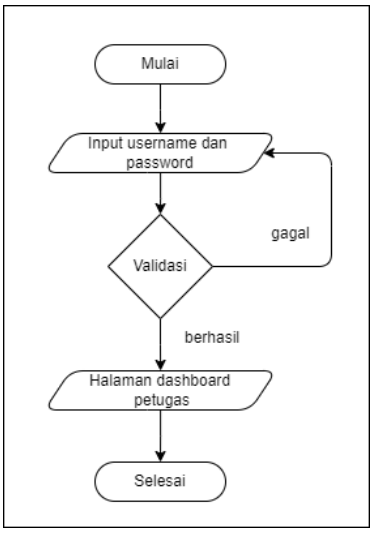
(4) Grafik matriks kesalahan *login user*

Tabel 4. 59 Grafik matriks kesalahan *login user*

	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

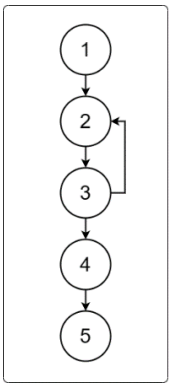
n. *White box testing login* berhasil pada *user* / petugas

1) *Flowchart*



Gambar 4. 65 *Flowchart login* berhasil pada *user*

2) *Flowgraph*



Gambar 4. 66 *Flowgraph login* berhasil pada *user*

Berdasarkan gambar 4. 71 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

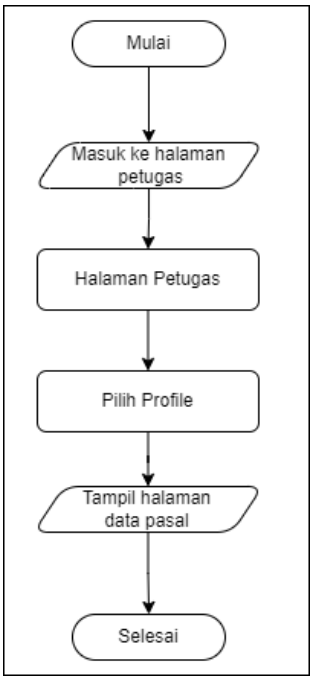
(4) Grafik matriks *login* berhasil pada *user*

Tabel 4. 60 Grafik matriks *login* berhasil pada *user*

	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

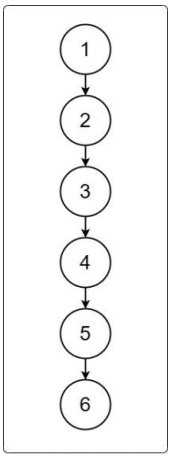
o. *White box testing* halaman *profile*

1) *Flowchart*



Gambar 4. 67 *Flowchart* halaman *profile*

2) *Flowgraph*



Gambar 4. 68 *Flowgraph* halaman *profile*

Berdasarkan gambar 4. 85 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

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

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

Path 1 = 1 – 2 – 3 – 4 – 5

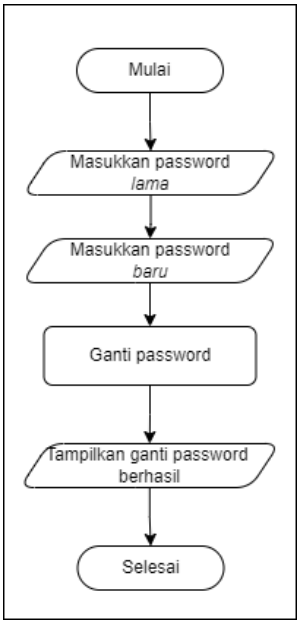
(4) Grafik matriks halaman *profile*

Tabel 4. 61 Grafik matriks halaman *profile*

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

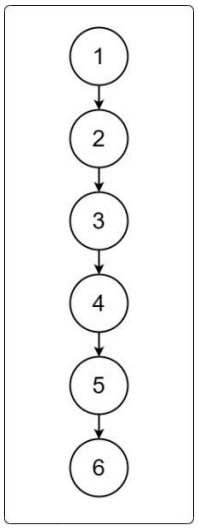
p. *White box testing* ganti *password user* / petugas

1) *Flowchart*



Gambar 4. 69 *Flowchart* halaman ganti *password user*

2) *Flowgraph*



Gambar 4. 70 *Flowgraph* halaman ganti *password user*

Berdasarkan gambar 4. 81 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *cyclomatic complexcity* dari *flowgraph* diatas memiliki *region = 1*

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

Path 1 = 1 – 2 – 3 – 4 – 5

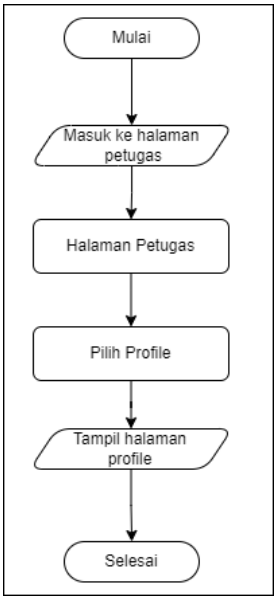
(4) Grafik matriks halaman ganti *password user*

Tabel 4. 62 Grafik matriks halaman ganti *password user*

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

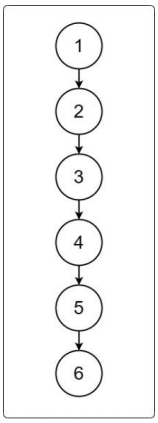
q. *White box testing* halaman data tilang

1) *Flowchart*



Gambar 4. 71 *Flowchart* halaman data tilang

2) *Flowgraph*



Gambar 4. 72 *Flowgraph* halaman data tilang

Berdasarkan gambar 4. 85 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

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

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

Path 1 = 1 – 2 – 3 – 4 – 5

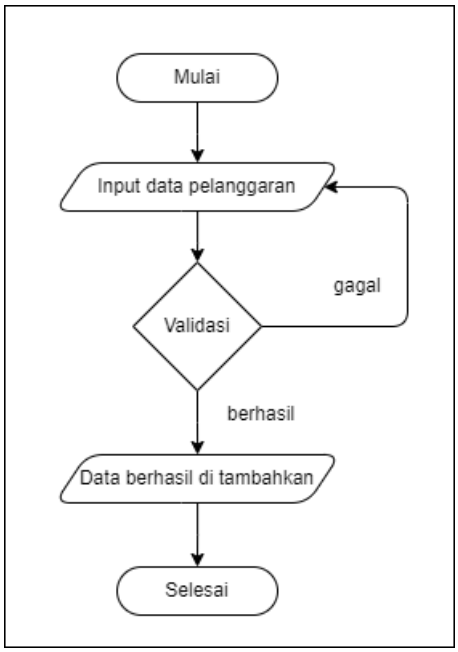
(4) Grafik matriks halaman data tilang

Tabel 4. 63 Grafik matriks halaman data tilang

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

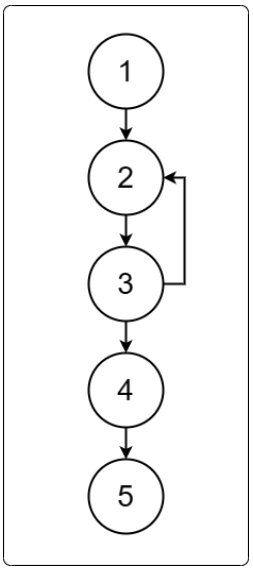
r. *White box testing* tambah data tilang

1) *Flowchart*



Gambar 4. 73 *Flowchart* tambah data tilang

2) *Flowgraph*



Gambar 4. 74 *Flowgraph* tambah data tilang

Berdasarkan gambar 4. 73 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

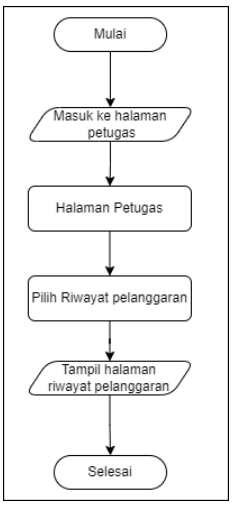
(4) Grafik matriks tambah data tilang

Tabel 4. 64 Grafik matriks tambah data tilang

	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

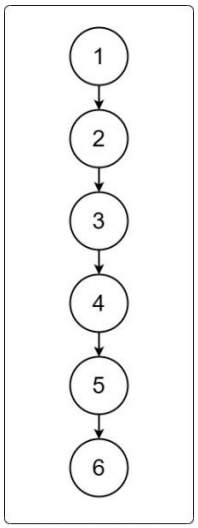
s. *White box testing* halaman riwayat pelanggaran

1) *Flowchart*



Gambar 4. 75 *Flowchart* halaman riwayat pelanggaran

2) *Flowgraph*



Gambar 4. 76 *Flowgraph* halaman riwayat pelanggaran

Berdasarkan gambar 4. 85 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

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

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

Path 1 = 1 – 2 – 3 – 4 – 5

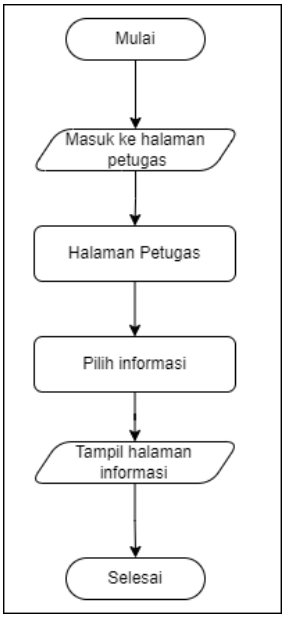
(4) Grafik matriks halaman riwayat pelanggaran

Tabel 4. 65 Grafik matriks halaman riwayat pelanggaran

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

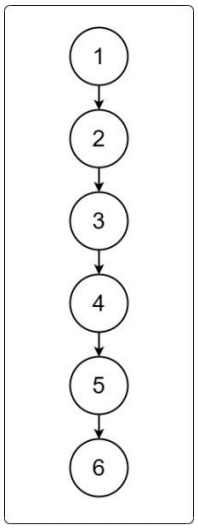
t. *White box testing* halaman informasi

1) *Flowchart*



Gambar 4. 77 *Flowchart* halaman informasi

2) *Flowgraph*



Gambar 4. 78 *Flowgraph* halaman informasi

Berdasarkan gambar 4. 85 diatas, dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 4

N (*node*) = 5

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 4 – 5 + 2

= 1

Predikat (P) = P + 1

= 0 + 1

= 1

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

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

Path 1 = 1 – 2 – 3 – 4 – 5

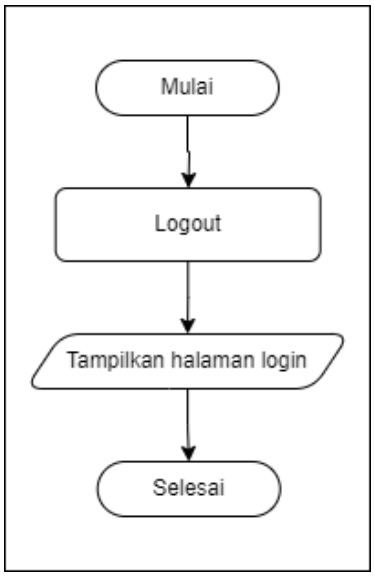
(8) Grafik matriks halaman informasi

Tabel 4. 66 Grafik matriks halaman informasi

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

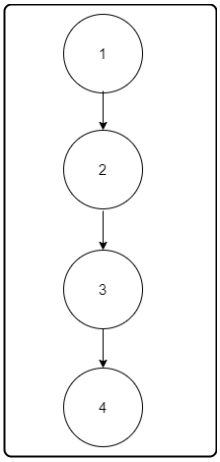
u. *White box testing* logout *user* / petugas

1) *Flowchart*



Gambar 4. 79 *Flowchart logout user*

2) Flowgraph



Gambar 4. 80 *Flowgraph logout user*

Berdasarkan gambar 4. 36 diatas dilakukan perhitungan sebagai berikut:

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

Pada rumus : V(G) = E – N + 2

E (*edge*) = 3

N (*node*) = 4

P (Predikat *node*) = 0

Penyelesaian :

V(G) = E – N + 2

= 3 – 4 + 2

= 2

Predikat (P) = P + 1

= 0 + 1

= 1

(2) Berdasarkan perhitungan *Cyclomatic Complexcity* dari *flowgraph* di atas memiliki *Region* = 1

(3) *Independent path* pada *flowgraph* tersebut yakni:

Path 1 = 1 – 2 – 3 – 4

(4) Grafik matriks *logout user*

Tabel 4. 67 Grafik Matriks *logout user*

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

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, penulis menyatahkan bahwa kesimpulan yang dapat ditarik adalah sebagai berikut:

1. Dalam penelitian yang telah dilakukan, penulis berhasil merancang dan membangun sebuah aplikasi yang petugas kepolisian Satlantas Polres Parepare terkhususnya yang bekerja di lapangan untuk melakukan pendataan pelanggaran lalu lintas serta memudahkan pihak petugas yang bertindak sebagai admin untuk mengirim nontifikasi peringatan ke pelanggar serta melakukan rekapitulasi data tilang.
2. Aplikasi *Android* ini dibuat menggunakan bahasa *Java* dan dihubungkan dengan *website* melalui teknologi *API*, sehingga memungkinkan akses data yang cepat dan efisien.
3. Sistem ini dibangun dengan menggunakan framework Codeignither untuk *backend*. Visual Studio Code digunakan sebagai *text editor* dan MySQL sebagai sistem manajemen *database*.

B. Saran

Pada penelitian ini penulis menyadari bahwa masih ada beberapa kekurangan yang perlu diperbaiki dan dikembangkan di penelitian selanjutnya. Oleh karena itu, penulis memiliki beberapa saran untuk pengembangan selanjutnya, sebagai berikut:

1. Aplikasi ini dapat dikembangkan kembali dengan desain antarmuka yang lebih menarik dan user-friendly.
2. Menambahkan fitur notifikasi otomatis untuk mengingatkan petugas akan pembaruan data atau informasi penting.
3. Mengimplementasikan sistem keamanan yang lebih kuat untuk melindungi data sensitif dari akses yang tidak sah.

DAFTAR PUSTAKA

Anif, M., Ayuningtyas, R.A.M. Dyah, & Nugroho, D.A. (2020). Karakteristik Bahasa Java yang Menjadi Faktor Utama Popularitasnya. Jurnal Sistem Informasi Universitas Negeri Semarang, 17(1), 1–10.

Comasie. (2020). Perancangan Aplikasi Absensi Karyawan dengan Menggunakan Kode QR Berbasis Android. Comasie, 3(3), 21–30.

Dennis, H., & Ekawati, N. (2020). Perancangan Aplikasi Absensi Karyawan Dengan Menggunakan Kode QR Berbasis Android. Comasie, 3(3), 21–30.

Fatdha, T. S. E., Junadhi, Susanti, Yenni, H., & Zoromi, F. (2021). Pelatihan Pengenalan Pemrograman Android untuk Pemula pada Siswa-Siswi SMK N 2 Pekanbaru. J-PEMAS STMIK Amik Riau, 2(1), 25–32.

Fitri, R., & Astuti, P. (2022). Pengembangan Aplikasi Mobile Berbasis Android Menggunakan Metode Waterfall. Jurnal Sistem Informasi dan Teknologi Informasi, 16(2), 109-116.

Hartl, M., & Donahoe, L. (2019). Learn Enough HTML to Be Dangerous. Learn Enough Society.

Irmawati, H., & Nurdiansah, I. (2022). Membangun Aplikasi Tilang Berbasis Database. X, 1, 196–206.

Lestari, D., & dkk. (2022). Firebase: Platform Pengembangan Aplikasi Seluler Inovatif. Jurnal Teknologi Informasi dan Komunikasi, 11(2), 112-123.

Munawar, G., Sibarani, N. S., & Wisnuadhi, B. (2018). Analisis Performa Aplikasi Android pada Bahasa Pemrograman Java dan Kotlin. Dalam Prosiding Industrial Research Workshop and National Seminar. Industrial Research Workshop and National Seminar.

Nematrian, N. (2020). HTML, CSS and JavaScript. Nematrian.

Patrick, J. (2019). Javascript: A Beginner’s Guide to Learning the Basics of JavaScript Programming. CreateSpace Independent Publishing Platform.

Purwantoro, C. A., Sanjaya, A. R., & Sanjaya, R. (2021). Aplikasi Tilang Menggunakan Scan Plat Nomor Kendaraan Berbasis Android. EProsiding Sistem Informasi (POTENSI), 2(1), 244–252.

Shute, Z. (2020). Advanced JavaScript: Speed up Web Development with the Powerful Features and Benefits of JavaScript. Packt Publishing Ltd.

Sidak, D. R. (2020). Pengaruh Kualitas Pelayanan STNK Terhadap Kepuasan Konsumen di Kantor SAMSAT Manado. Corporate Governance (Bingley), 10(1), 54–75.

Sutarman. (2020). Membangun aplikasi Web dengan PHP & MySQL. Andi Publisher.

Voutama, A. (2022). Sistem Antrian Cucian Mobil Berbasis Website Menggunakan Konsep CRM dan Penerapan UML. Komputika: Jurnal Sistem Komputer, 11(1), 102–111. https://doi.org/10.34010/komputika.v11i1.4677.

Yanti, N., & Lesmideryati, D. (2022). Pelatihan Pembuatan Aplikasi Android Berbasis Android Studio untuk Guru dan Siswa SMAIT Al Auliya Balikpapan. Prosiding Seminar Nasional Terapan Riset Inovatif (SENTRINOV), 8(3), 114–123.