



Dokumen Pengembangan Produk Lembar Sampul Dokumen

Judul Dokumen

TUGAS AKHIR:

Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*

Jenis Dokumen

PROPOSAL

Catatan: Dokumen ini dikendalikan penyebarannya oleh Dept. Teknik Elektro Undip

Nomor Dokumen

B100-03-TA2223.2.19012

Nomor Revisi

03

Nama File

B100-2-TA2223

Tanggal Penerbitan

30 Januari 2023

Unit Penerbit

Departemen Teknik Elektro Undip

Jumlah Halaman

77

(termasuk lembar sampul ini)

Data Pengusul

Pengusul	Nama NIM	Henric Dhiki Wicaksono 21060119120011	Jabatan Tanda Tangan	Anggota 
	Nama NIM	Novi Dianasari 21060119120039	Jabatan Tanda Tangan	Anggota 
	Nama NIM	Muhammad Khoiril Wafi 21060119140133	Jabatan Tanda Tangan	Anggota 
Pembimbing Utama	Nama NIP	M. Arfan, S.Kom., M.Eng. 198408172015041002	Tanda Tangan	
Pendamping	Nama NIP	Imam Santoso, S.T., M.T. 197012031997021001	Tanda Tangan	

DAFTAR ISI

1 PENDAHULUAN	4
1.1 RINGKASAN ISI DOKUMEN	4
1.2 APLIKASI DOKUMEN	5
1.3 REFERENSI.....	5
1.4 DAFTAR SINGKATAN	7
2 PROPOSAL PENGEMBANGAN PRODUK	9
2.1 PENDAHULUAN	9
2.1.1 Latar Belakang Masalah	9
2.1.2 Rumusan Masalah.....	10
2.1.3 Tujuan	11
2.1.4 Alternatif Desain.....	11
2.2 KONSEP DESAIN	27
2.2.1 Konfigurasi Umum	27
2.2.2 Kemampuan dan Kapasitas Produk	46
2.2.3 Dasar Teori yang Mendukung Proses Pengembangan	49
2.2.4 Teknologi yang Digunakan	60
2.2.5 Batasan – Batasan Sistem.....	61
2.2.6 Standarisasi Produk.....	62
2.2.7 Etika Profesi yang Dijunjung.....	63
2.3 SKENARIO PEMANFAATAN PRODUK	64
2.4 NILAI STRATEGIS	68
2.5 USAHA PENGEMBANGAN.....	69
2.5.1 <i>Man-Month</i>	69
2.5.2 <i>Machine-Month</i>	69
2.5.3 <i>Development Tools</i>	69
2.5.4 <i>Test Equipment</i>	70
2.5.5 Kebutuhan <i>Expert</i>	70
2.5.6 Perkiraan Biaya	71
2.5.7 Peluang Keberhasilan	71
2.5.8 Jadwal dan Waktu Pengembangan	72
3 KESIMPULAN	73
4 BIODATA TIM PENGUSUL	74

Catatan Sejarah Perbaikan Dokumen

VERSI, TGL, OLEH	PERBAIKAN
01, 21 November 2022, oleh Henric Dhiki Wicaksono, Novi Dianasari, dan Muhammad Khoiril Wafi.	<ul style="list-style-type: none"> • Penulisan • Perbandingan alternatif • <i>Constraint</i> kurang • Bisa dicarikan literatur dari Undip atau paper jurnal • Untuk referensi ditulis sesuai format IEEE • Apakah tidak menggunakan protokol yang khusus untuk IoT? • Apakah tidak dimungkinkan dengan 4G sebagai komplemen bila koneksi Wi-Fi sedang <i>off</i>? • Bagaimana <i>hosting server</i>-nya? <i>Di-cloud</i>? • Bagaimana pengenalan wajah oleh kamera di android? Adakah keamanan lainnya sebagai komplemen pengaman pendekripsi wajah? • Data apa saja yang disimpan dalam <i>database server</i>? • Mengapa pakai MySQL? • Mengapa pakai FCM?
02, 21 Desember 2022, oleh Henric Dhiki Wicaksono, Novi Dianasari, dan Muhammad Khoiril Wafi.	<ul style="list-style-type: none"> • Kompleksitas masalahnya kurang • Penulisan referensi tidak sesuai pedoman • Rujukan dalam tulisan malah tidak ada pada referensi • Tidak ada rumus matematika • 1. Sudah banyak TA di luar yang membahas ini dan semua memiliki kesamaan topik yaitu: menggunakan ESP8266, <i>firebase</i>, arduino, mysql, android. Rata-rata dikerjakan hanya satu orang saja cukup. • 2. Kenapa tidak pakai <i>micro</i> ESP32 yang sudah <i>all in one</i> atau <i>raspberry</i> yang lebih <i>advanced</i>? • 3. Tunjukkan kelebihan sistem anda dan kalau mungkin pakai alternatif lain selain ESP8266 (karena semua topik di no 1 selalu pakai arduino dan ESP8266) sebagai pembeda utama. Syukur kalau bisa <i>support</i> IOS.
03, 13 Januari 2023, oleh Henric Dhiki Wicaksono, Novi Dianasari, dan Muhammad Khoiril Wafi.	<ul style="list-style-type: none"> • Penulisan referensi tidak sesuai format. • 1. Pembahasan <i>Basic Science and math</i> minim. • 2. Belum terlihat <i>standard engineering</i> yang dirujuk. • 3. Kerja sama tim dan pembagian kerja tidak terlihat. Pada proposal pertama sudah ada diagram pembagian tugas (Gambar 2.1), yang sekarang malah tidak ada.

PROPOSAL

Rancang Bangun Sistem Keamanan Kunci Pintu Gedung

Berbasis Internet of Things

1 PENDAHULUAN

1.1 RINGKASAN ISI DOKUMEN

Dokumen ini berisikan uraian proposal proyek pengembangan Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*. Dokumen rancang bangun sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) adalah dokumen yang menjelaskan tentang cara membangun sistem keamanan kunci pintu gedung yang menggunakan teknologi IoT. Sistem ini akan menggunakan sensor kunci pintu yang terhubung dengan jaringan internet, sehingga dapat diakses dan dikontrol secara *remote* melalui perangkat yang terhubung dengan internet. Sistem ini akan memiliki beberapa fitur seperti kemampuan untuk membuka dan mengunci pintu secara *remote*, memantau aktivitas pintu secara *real-time*, dan memberikan notifikasi kepada pengguna jika terjadi aktivitas yang tidak diinginkan di pintu. Dokumen ini juga akan menjelaskan tentang komponen-komponen yang dibutuhkan untuk membangun dan konfigurasi sistem ini. Dokumen ini digunakan sebagai acuan dalam pelaksanaan proyek dan penggerjaan produk Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things* yang direncanakan.

Rancangan awal tugas akhir Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things* yang dimuat dalam dokumen B100 ini mencakup empat bagian, yaitu:

- Bagian satu meliputi ringkasan isi dokumen, aplikasi dokumen, referensi, serta daftar singkatan.
- Bagian dua berisi pendahuluan, konsep desain, skenario pemanfaatan produk, nilai strategis, serta usaha pengembangan produk.
- Bagian tiga berisi kesimpulan.
- Bagian empat berisi biodata tim pengusul.

1.2 APLIKASI DOKUMEN

Dokumen ini berlaku untuk pengembangan produk “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*” untuk:

- (1) Sebagai gambaran umum dari segi teknik maupun non-teknis tugas akhir Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things* yang akan dikerjakan.
- (2) Memastikan kelayakan tugas akhir Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*, baik dari segi teknik, waktu, biaya/ekonomis, maupun strategis.
- (3) Pencatatan proses kerja dan revisi yang dilakukan.

Proposal ini diajukan kepada dosen pembimbing dan tim tugas akhir Program Studi Sarjana Teknik Elektro Undip sebagai bahan penilaian tugas akhir.

1.3 REFERENSI

- [1] G. R. G. Wisnu, “RANCANG BANGUN SISTEM KEAMANAN PADA SMART BUILDING DENGAN PENERAPAN IoT (INTERNET OF THINGS),” *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 1, no. 1, hlm. 828–835, 2017, doi: <https://doi.org/10.36040/jati.v1i1.2074>.
- [2] A. T. Sianturi, “RANCANG BANGUN SISTEM KEAMANAN RUANGAN DENGAN SENSOR CAHAYA BERBASIS ARDUINO UNO MENGGUNAKAN SMS GATEWAY,” Universitas Sumatra Utara, Medan, 2019. Diakses: Jan 19, 2023. [Daring]. Available: <https://123dok.com/document/zpn136v4-rancang-keamanan-ruangan-berbasis-arduino-menggunakan-gateway-laporan.html>
- [3] Y. Efendi, “Internet Of Things (Iot) Sistem Pengendalian Lampu Menggunakan Raspberry Pi Berbasis Mobile,” *JURNAL ILMIAH ILMU KOMPUTER*, vol. 4, no. 2, hlm. 21–27, Sep 2018, doi: 10.35329/jiik.v4i2.41.
- [4] M. K. Syabibi dan A. Subari, “RANCANG BANGUN SISTEM MONITORING KEAMANAN RUMAH BERBASIS WEB MENGGUNAKAN RASPBERRY PI B+ SEBAGAI SERVER DAN MEDIA KONTROL,” *GEMA TEKNOLOGI*, vol. 19, no. 1, hlm. 22–29, 2016.

- [5] O. R. Galaxy, B. W. Sanjaya, dan F. T. P. W, “RANCANG BANGUN SISTEM PENGAMAN RUMAH TINGGAL BERBASIS ARDUINO UNO MENGGUNAKAN TELEPON PINTAR / SMARTPHONE ANDROID,” *JURNAL TEKNIK ELEKTRO UNIVERSITAS TANJUNGPURA*, vol. 1, no. 1, 2020, Diakses: Jan 20, 2023. [Daring]. Available: <https://jurnal.untan.ac.id/index.php/jteuntan/article/view/39605>
- [6] D. Widcaksono dan Masyhad, “RANCANG BANGUN SECURED DOOR AUTOMATIC SYSTEM UNTUK KEAMANAN RUMAH MENGGUNAKAN SMS BERBASIS ARDUINO,” *Ejournal Kajian Teknik Elektro*, vol. 3, no. 1, hlm. 52–66, 2018.
- [7] I. P. A. W. Widyatmika, N. P. A. W. Indrawati, I. W. W. A. Prastyo, I. K. Darminta, I. G. N. Sangka, dan A. A. N. G. Sapteka, “Perbandingan Kinerja Arduino Uno dan ESP32 Terhadap Pengukuran Arus dan Tegangan,” *Jurnal Otomasi, Kontrol & Instrumentasi*, vol. 13, no. 1, hlm. 37–45, 2021.
- [8] S. Tjandra dan G. S. Chandra, “Pemanfaatan Flutter dan Electron Framework pada Aplikasi Inventori dan Pengaturan Pengiriman Barang,” *Journal of Information System, Graphics, Hospitality and Technology*, vol. 2, no. 02, hlm. 76–81, Des 2020, doi: 10.37823/insight.v2i02.109.
- [9] P. F. Nahak, N. M. R. Mamulak, dan Y. C. H. Siki, “Sistem Informasi Geografis Untuk Pemetaan Wifi.id Corner Dan Wifi Gratis di Kota Kupang Berbasis Web,” *Jurnal Teknik Informatika UnikaSt. Thomas (JTIUST)*, vol. 5, no. 1, hlm. 71–79, Jun 2020.
- [10] F. Luthfi, “Penggunaan Framework Laravel dalam Rancang Bangun Modul Back-End Artikel Website Bisnisbisnis.ID,” *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 2, no. 1, hlm. 34–41, Agu 2017, doi: 10.14421/jiska.2017.21-05.
- [11] N. S. Hapsari, Y. Fatman, dan Isbandi, “Implementasi Metode One Time Password pada Sistem Pemesanan Online,” *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 4, no. 4, hlm. 930–939, Okt 2020.
- [12] J. Dedy irawan dan E. Adriantantri, “PEMANFAATAN QR-CODE SEBAGAI MEDIA PROMOSI TOKO,” *Jurnal Mnemonic*, vol. 1, no. 2, hlm. 56–61, Des 2019, doi: 10.36040/mnemonic.v1i2.39.

1.4 DAFTAR SINGKATAN

Tabel 1.1 Daftar Singkatan

SINGKATAN	ARTI
IoT	<i>Internet of Things</i>
iOS	<i>iPhone Operating System</i>
CCTV	<i>Closed Circuit Television</i>
RFID	<i>Radio Frequency Identification</i>
QR Code	<i>Quick Response Code</i>
SSO	<i>Single Sign On</i>
ESP	<i>Espressif</i>
GPIO	<i>General Purpose Input/Output</i>
IDE	<i>Integrated Development Environment</i>
Wi-Fi	<i>Wireless Fidelity</i>
BLE	<i>Bluetooth Low Energy</i>
DBMS	<i>Database Management System</i>
Ms. Access	<i>Microsoft Office Access</i>
SQL	<i>Structured Query Language</i>
VBA	<i>Visual Basic for Applications</i>
MVCC	<i>Multi-Version Concurrency Control</i>
JS	<i>Java Script</i>
API	<i>Application Programming Interface</i>
JWT	<i>JSON Web Token</i>
JSON	<i>Java Script Object Notation</i>
URL	<i>Uniform Resource Locator</i>
PWA	<i>Progressive Web Apps</i>
GPS	<i>Global Positioning System</i>
MVC	<i>Model View Controller</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
SSL/TLS	<i>Secure Sockets Layer/Transport Layer Security</i>

SINGKATAN	ARTI
WPA	<i>Wi-Fi Protected Access</i>
AES	<i>Advanced Encryption Standard</i>
LED	<i>Light Emitting Diode</i>
PC817	<i>Photo Coupler 817</i>
FCM	<i>Firebase Cloud Messaging</i>
OTP	<i>One Time Password</i>
SMS	<i>Short Messaging Services</i>
CCTV	<i>Closed Circuit Television</i>
kB	<i>kilo Byte</i>
ROM	<i>Read Only Memory</i>
SRAM	<i>Static Random-Access Memory</i>
RTC	<i>Real-Time Clock</i>
MB	<i>Mega Byte</i>
ADC	<i>Analog to Digital Converter</i>
SPI	<i>Serial Peripheral Interface</i>
SDK	<i>Software Development Kit</i>
PAN	<i>Personal Area Networks</i>
GHz	<i>Giga Hertz</i>
PHP	<i>Hypertext Preprocessor</i>
OOP	<i>Object Oriented Programming</i>
HOTP	<i>HMAC-based OTP</i>
TOTP	<i>Time-based OTP</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
NSPE	<i>National Society of Professional Engineer</i>
PCB	<i>Printed Circuit Board</i>
FTP	<i>File Transfer Protocol</i>
SMTP	<i>Simple Mail Transfer Protocol</i>
IMAP	<i>Instant Messaging Protocol</i>
HP	<i>Handphone</i>

2 PROPOSAL PENGEMBANGAN PRODUK

2.1 PENDAHULUAN

2.1.1 Latar Belakang Masalah

Sistem keamanan menjadi suatu aspek penting untuk menunjang kinerja dari sebuah gedung[1]. Gedung itu memiliki beberapa ruangan dan beberapa pintu untuk masuk. Tidak banyak orang yang meninggalkan ruangan, namun tidak menutup pintu dengan sempurna. Hal ini mungkin terdengar sepele tetapi siapa tahu ruangan tersebut memiliki privasi atau hal-hal penting yang perlu dijaga. Gedung pasti memiliki cara tersendiri dalam menjaga privasi dan keamanan setiap ruangnya.

Salah satu masalah utama dalam sistem keamanan kunci pintu gedung adalah bagaimana cara mengelola akses masuk ke dalam gedung dengan cepat dan efisien. Biasanya, dibutuhkan seorang petugas untuk mengelola akses masuk ke dalam gedung, yang tentunya dapat menyebabkan kesalahan atau kelambatan dalam proses pengelolaan akses. Selain itu, sistem keamanan kunci pintu gedung yang biasa digunakan saat ini juga sering mengalami masalah seperti kunci hilang atau tidak dapat digunakan, yang dapat menyebabkan gangguan dalam aktivitas di dalam gedung. Dengan sistem keamanan yang terintegrasi akan sangat membantu meminimalisir sebuah masalah sistem keamanan dalam gedung/ruangan dari bahaya adanya orang lain yang masuk tanpa seizin pemilik[2].

Sebuah sistem keamanan *Access Control* memungkinkan pemilik bangunan dan properti untuk melakukan lebih dari sekedar mengontrol masuk ke daerah yang diproteksi[2]. Sistem ini juga dapat membuat catatan *history* atau informasi secara elektronik mengenai siapa saja yang masuk ke dalam ruangan yang sudah diproteksi[2]. Dengan adanya catatan informasi tersebut membantu pemilik usaha mengidentifikasi siapa saja yang masuk ke ruangan pada waktu-waktu tertentu[2].

Internet of Thing (IoT) merupakan suatu konsep yang bertujuan untuk memperluas manfaat dari koneksi internet yang tersambung secara terus menerus[3]. *Internet of Things* (IoT) bisa dimanfaatkan pada gedung perkantoran maupun rumah sebagai alat untuk mengendalikan peralatan elektronik dan juga sebagai suatu sistem keamanan yang dapat dioperasikan dari jarak jauh melalui jaringan komputer. Sehingga tidak dapat dipungkiri bahwa teknologi ini harus bisa diterapkan dalam kehidupan sehari-hari.

Dengan menggunakan sistem keamanan kunci pintu gedung berbasis IoT, diharapkan dapat memudahkan pengelolaan akses masuk ke dalam gedung, serta meningkatkan keamanan dengan mengontrol akses masuk hanya kepada orang-orang yang memiliki izin saja. Sistem ini menggunakan teknologi IoT untuk mengontrol akses masuk ke dalam gedung dengan menggunakan perangkat seluler atau kartu akses yang terhubung ke internet. Misalnya, dengan menggunakan aplikasi atau *web interface* yang terhubung ke internet, admin dapat dengan mudah mengelola akses masuk ke gedung, menambah atau menghapus kunci elektronik, dan mengontrol pintu dari jarak jauh. Selain itu, sistem keamanan kunci pintu gedung berbasis IoT juga dapat dilengkapi dengan sensor dan memberikan notifikasi kepada admin jika terdeteksi aktivitas yang tidak diinginkan. Hal ini dapat membantu meningkatkan keamanan gedung dan mengurangi risiko kejahatan seperti pencurian.

Oleh karena itu, berdasarkan uraian latar belakang masalah di atas maka muncul sebuah gagasan yang berjudul “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*” yang diharapkan dapat meningkatkan efisiensi dan keamanan dalam pengelolaan akses masuk ke dalam gedung.

2.1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan di atas, maka permasalahan yang akan dibahas yaitu:

1. Bagaimana perancangan sistem keamanan kunci pintu gedung dengan *Access Control*?
2. Bagaimana perancangan perangkat penguncian yang mendukung sistem keamanan kunci pintu gedung berbasis IoT?
3. Bagaimana perancangan sistem *database* dan *server* untuk mendukung sistem keamanan *Access Control* dan pembuatan catatan *history* atau informasi?
4. Bagaimana perancangan aplikasi perangkat *mobile* yang dapat *support* android dan iOS sebagai piranti akses masuk pintu gedung?
5. Bagaimana perancangan *website* untuk mendukung sistem *monitoring* dan *controlling* jarak jauh?
6. Bagaimana perancangan sistem komunikasi data dua arah untuk mendukung sistem keamanan kunci pintu gedung berbasis IoT?

2.1.3 Tujuan

Produk yang dirancang pada proposal tugas akhir ini adalah sebuah sistem keamanan kunci pintu gedung berbasis *Internet of Things* dengan tujuan pengembangan sebagai berikut:

1. Merancang sistem keamanan kunci pintu gedung dengan *Access Control*.
2. Merancang perangkat penguncian yang mendukung sistem keamanan kunci pintu gedung berbasis IoT.
3. Merancang sistem *database* dan *server* untuk mendukung sistem keamanan *Access Control* dan pembuatan catatan *history* atau informasi.
4. Merancang aplikasi perangkat *mobile* yang dapat *support* android dan iOS sebagai piranti akses masuk pintu gedung.
5. Merancang *website* untuk mendukung sistem *monitoring* dan *controlling* jarak jauh.
6. Merancang sistem komunikasi data dua arah untuk mendukung sistem keamanan kunci pintu gedung berbasis IoT.

2.1.4 Alternatif Desain

a. Perancangan sistem keamanan kunci pintu gedung dengan *Access Control*

Membangun sistem keamanan berbasis teknologi memerlukan perhatian khusus baik dari instansi pemerintah maupun swasta. Bagian dari sistem keamanan adalah kontrol akses, yang tidak hanya menjaga keamanan gedung tetapi juga membatasi akses masuk. Sistem kontrol akses membatasi akses ke sumber daya baik secara fisik maupun virtual. Dengan demikian, hanya pengguna tertentu yang memiliki hak akses untuk menjaga keamanan gedung.

Orang yang menerima hak pakai harus memiliki informasi identifikasi dan dapat diverifikasi dengan identifikasi. Oleh karena itu, sistem keamanan gedung dapat diimplementasikan menggunakan kode atau nilai biometrik, sedangkan proses otentikasi dapat diimplementasikan menggunakan berbagai metode. Proses otentikasi dapat dilakukan melalui penggunaan kata sandi atau PIN, pengukuran biometrik seperti pemindaian retina, sidik jari, wajah, dan lainnya. Faktor lain dalam proses otentikasi dapat diimplementasikan oleh kartu atau kunci pengguna.

Selain menjadikan keamanan gedung lebih terjaga, akses kontrol membantu memantau riwayat akses secara rinci.

Biasanya hotel, perusahaan, rumah dan kantor menerapkan sistem akses kontrol, bahkan ada yang menggunakannya di rumah pribadi. Sistem ini tidak hanya penting dari segi keamanan, tetapi juga dapat membantu meningkatkan kinerja karyawan di perusahaan. Kontrol akses hanya memberikan hak akses terbatas kepada pihak yang memiliki informasi lisensi. Sistem akses ini juga memberikan kemudahan bagi petugas keamanan untuk memantau dengan fitur-fitur yang disediakan oleh sistem.

Setiap akses dapat dikelola dengan baik karena terdapat riwayat penggunaan yang detail. Bayangkan perusahaan tidak memiliki sistem keamanan, seperti kontrol akses yang tidak menyebabkan hak akses terbatas. Akibatnya, ada risiko bagi perusahaan bahwa penyusup dapat memasuki pintu masuk melalui kontrol akses seperti di hotel yaitu dengan kartu. Sistem keamanan juga dilengkapi dengan fasilitas CCTV untuk memantau dengan baik lingkungan hotel demi kenyamanan pelanggan.

Berdasarkan hasil analisis sistem *Access Control* untuk sistem keamanan kunci pintu gedung berbasis *Internet of Things*, maka terdapat beberapa alternatif desain yang dapat digunakan, yaitu:

1) Sistem *Access Control* berbasis kartu

Sistem ini menggunakan kartu yang diberikan kepada pengguna gedung untuk membuka pintu-pintu gedung. Kartu dapat berupa kartu RFID.

Sistem *Access Card* berbasis RFID sangat populer di kalangan perusahaan atau perkantoran. Dengan menggunakan kartu RFID setiap pengguna akan memiliki kunci atau *key* yang tersimpan di dalam kartu tersebut. Setiap kartu RFID mempunyai ruang penyimpanan cukup besar yaitu 2000byte yang dapat digunakan untuk menyimpan data untuk keperluan tertentu.

Adapun kelemahan sistem akses *card* berbasis RFID adalah RFID memiliki tingkat keamanan yang rendah, setiap kartu RFID dapat dengan mudah disalin hanya dengan menggunakan *reader* yang beredar dipasaran, sehingga data di dalamnya dapat disalahgunakan oleh orang lain yang tidak berwenang. Selain

itu, karena berukuran kecil, sebuah kartu RFID juga sering hilang atau tertinggal, sehingga perlu mencetak kartu baru untuk mendapatkan akses lagi.

2) Sistem *Access Control* berbasis biometrik

Sistem ini menggunakan fitur-fitur biometrik seperti sidik jari, iris, atau wajah sebagai cara untuk memverifikasi identitas pengguna gedung.

Sistem biometrik sangat banyak digunakan sebagai metode untuk melakukan autentikasi pada keamanan *Access Control*. Sistem biometrik menggunakan fisik manusia sebagai suatu identitas yang unik sehingga sangat sulit untuk dipalsukan.

Adapun kelemahannya adalah mesin identifikasi biometrik lebih mahal untuk dibeli daripada yang tradisional. Selain itu, beberapa pengguna mungkin menentang biometrik sama sekali, menganggapnya sebagai pelanggaran privasi. Selain itu, perangkat pengenalan biometrik tidak selalu akurat. Misalnya, orang yang kedinginan tidak dapat mengidentifikasi dirinya dengan perangkat pengenalan suara, dan orang yang mengalami kenaikan atau penurunan berat badan tiba-tiba dapat kehilangan akses ke area yang dilindungi oleh sistem yang menganalisis fitur wajah.

3) Sistem *Access Control* berbasis *Single Sign On* (SSO)

Sistem *Access Control* berbasis *Single Sign On* (SSO) adalah salah satu jenis sistem keamanan kunci pintu gedung yang memungkinkan pengguna gedung untuk membuka pintu-pintu gedung dengan menggunakan satu akun yang sama yang digunakan untuk *login* ke sistem manajemen gedung atau aplikasi keamanan lainnya. Hal ini sangat memudahkan pengguna karena tidak perlu banyak akun untuk mengakses beberapa layanan yang ada. Dikerenakan hanya menggunakan satu akun maka *log* akses dapat mencatat semua riwayat tentang siapa saja yang telah mengakses sistem atau aplikasi, sehingga siapa saja yang telah menyalahgunakan akun bisa diketahui melalui *log* akses ini.

Namun, sistem *Access Control* berbasis SSO juga memiliki beberapa kekurangan. Salah satunya adalah bahwa perangkat yang digunakan untuk *login* ke sistem harus terhubung dengan *Internet of Things* agar dapat bekerja dengan baik. Selain itu, ada kemungkinan bahwa akun yang digunakan dapat

dicuri atau disalahgunakan oleh pihak yang tidak bertanggung jawab, sehingga mengurangi keamanan sistem.

4) Sistem *Access Control* berbasis QR *Code*

Sistem *Access Control* berbasis QR *Code* atau kode QR adalah salah satu jenis sistem keamanan kunci pintu gedung yang menggunakan kode QR sebagai cara untuk membuka pintu-pintu gedung. Pengguna gedung dapat memindai kode QR yang terpasang di dekat pintu dengan menggunakan *smartphone* atau perangkat lain yang terhubung *dengan Internet of Things*.

Kelebihan dari sistem *Access Control* berbasis kode QR adalah mudah digunakan dan tidak memerlukan perangkat tambahan seperti kartu atau *keypad*. Sistem ini juga dapat dengan mudah diintegrasikan dengan aplikasi manajemen gedung atau aplikasi keamanan lainnya. Selain itu, kode QR memiliki layar yang lebih kecil daripada *barcode*. Karena kode QR dapat menerima informasi secara horizontal dan vertikal, maka ukuran tampilan gambar kode QR otomatis hanya sepersepuluh dari ukuran *barcode*. Oleh karena itu, meskipun beberapa simbol kode QR kotor atau rusak, informasinya masih dapat disimpan dan dibaca. Kode QR juga dapat dengan mudah diakses dengan cepat sehingga cocok digunakan dengan *smartphone*.

Namun, sistem *Access Control* berbasis kode QR juga memiliki beberapa kekurangan. Sebuah kode QR memerlukan sebuah *scanner* untuk membaca data yang tersimpan di dalamnya dan juga memerlukan sebuah sistem yang dapat menerjemahkan data tersebut sehingga prosesnya cukup kompleks.

b. Perancangan perangkat penguncian yang mendukung sistem keamanan kunci pintu gedung berbasis IoT

Perangkat penguncian merupakan alat yang akan dipasang pada setiap pintu dan digunakan untuk melakukan proses penguncian. Pada perangkat penguncian terdapat 4 bagian yaitu sensor, mikrokontroler, aktuator, dan *power*.

1) Sensor

Pada perangkat penguncian, sensor digunakan untuk melakukan *monitoring* kondisi pintu, sensor yang digunakan harus dapat mengetahui kondisi pintu sedang terbuka atau tertutup.

Berdasarkan kedua kondisi pintu tersebut (terbuka dan tertutup), untuk sistem keamanan kunci pintu gedung berbasis *Internet of Things*, maka terdapat beberapa alternatif desain sensor yang dapat digunakan, yaitu:

- *Limit switch*

Kelebihan menggunakan *limit switch* untuk sistem keamanan kunci pintu gedung berbasis IoT adalah harganya relatif murah, sehingga dapat menjadi pilihan yang ekonomis untuk mengontrol akses ke pintu gedung. Tidak memerlukan banyak perawatan atau pemeliharaan, sehingga dapat mengurangi biaya pemeliharaan. Dapat bekerja dengan baik dalam kondisi yang keras, seperti suhu tinggi atau lingkungan yang berdebu.

Kelemahan menggunakan *limit switch* untuk sistem keamanan kunci pintu gedung berbasis IoT adalah pemasangan dan kalibrasi dapat menjadi rumit, terutama jika digunakan dalam aplikasi yang kompleks. Dapat menjadi tidak akurat jika terkena benturan atau terguncang. Dapat menjadi tidak responsif dalam situasi yang berubah cepat. Dapat terpengaruh oleh lingkungan yang berdebu atau kotor, yang dapat menyebabkan kontak terbuka atau tertutup secara tidak sengaja.

- *Saklar magnetik*

Saklar magnetik menggunakan magnet yang terpasang pada daun pintu untuk memicu saklar sehingga tidak ada kontak langsung dengan daun pintu.

Kelebihan menggunakan saklar magnetik untuk sistem keamanan kunci pintu gedung berbasis IoT adalah dapat mendeteksi objek dengan akurasi yang tinggi, sehingga dapat meningkatkan keamanan gedung dengan membatasi akses hanya untuk orang yang memiliki izin atau otorisasi yang sesuai. Dapat bekerja dengan baik dalam kondisi yang keras, seperti suhu tinggi atau lingkungan yang berdebu. Dapat bekerja dengan baik dalam kondisi yang terlalu cepat atau terlalu lambat.

Kelemahan menggunakan saklar magnetik untuk sistem keamanan kunci pintu gedung berbasis IoT adalah harganya relatif lebih mahal dibandingkan dengan sensor lain yang memiliki fungsi yang sama. Pemasangan dan kalibrasi dapat menjadi rumit, terutama jika digunakan dalam aplikasi yang kompleks. Dapat menjadi tidak responsif dalam situasi yang berubah cepat.

Dapat terpengaruh oleh lingkungan yang berdebu atau kotor, yang dapat menyebabkan kontak terbuka atau tertutup secara tidak sengaja.

2) Mikrokontroler

Untuk membaca sensor, kita memerlukan sebuah mikrokontroler. Dikarenakan sensor yang digunakan hanya memiliki dua kondisi maka kita cukup menggunakan *input* digital, yaitu pin *input* pada mikrokontroler yang digunakan untuk membaca nilai atau data digital (nilai benar atau salah).

Untuk sistem keamanan kunci pintu gedung berbasis *Internet of Things*, maka terdapat beberapa alternatif desain mikrokontroler yang dapat digunakan, yaitu:

- Arduino

Arduino memiliki 14 GPIO dan 6 *input* analog yang dapat kita gunakan dan juga lingkungan pengembangan yang mudah menggunakan ArduinoIDE.

Kelebihan Arduino untuk sistem keamanan kunci pintu gedung berbasis IoT adalah Arduino sangat mudah diprogram dan tidak memerlukan pemrograman yang rumit. Arduino memiliki banyak sensor yang tersedia yang dapat digunakan untuk mendeteksi gerakan, suhu, kelembaban, dan banyak lagi. Arduino sangat terjangkau dan mudah didapat dipasaran. Arduino memiliki banyak dokumentasi dan sumber daya *online* yang tersedia untuk membantu memulai proyek.

Kelemahan Arduino untuk sistem keamanan kunci pintu gedung berbasis IoT adalah Arduino tidak memiliki kemampuan pemrosesan yang kuat, sehingga mungkin tidak cocok untuk proyek yang membutuhkan pemrosesan data yang intensif. Arduino memiliki batasan dalam hal penyimpanan data, sehingga mungkin tidak cocok untuk proyek yang membutuhkan penyimpanan data yang besar. Arduino tidak memiliki kemampuan konektivitas yang kuat, sehingga mungkin tidak cocok untuk proyek yang membutuhkan konektivitas yang kuat atau konektivitas ke internet. Arduino mungkin tidak cocok untuk proyek yang membutuhkan keamanan yang ketat, karena tidak memiliki fitur keamanan yang canggih.

- ESP8266

ESP8266 tidak memiliki banyak GPIO, akan tetapi ESP8266 memiliki modul WiFi yang dapat kita gunakan untuk berkomunikasi dengan pengguna.

Kelebihan ESP8266 untuk sistem keamanan kunci pintu gedung berbasis IoT adalah ESP8266 memiliki kemampuan konektivitas yang kuat, sehingga mudah terhubung ke internet melalui WiFi. ESP8266 memiliki kemampuan pemrosesan yang lebih baik dibandingkan Arduino, sehingga lebih cocok untuk proyek yang membutuhkan pemrosesan data yang intensif. ESP8266 memiliki penyimpanan data yang lebih besar dibandingkan Arduino, sehingga lebih cocok untuk proyek yang membutuhkan penyimpanan data yang besar. ESP8266 memiliki fitur keamanan yang lebih canggih dibandingkan Arduino, sehingga lebih cocok untuk proyek yang membutuhkan keamanan yang ketat.

Kelemahan ESP8266 untuk sistem keamanan kunci pintu gedung berbasis IoT adalah ESP8266 mungkin lebih sulit diprogram dibandingkan Arduino, karena memerlukan pemrograman yang lebih rumit. ESP8266 mungkin lebih mahal dibandingkan Arduino, karena memiliki kemampuan yang lebih baik. ESP8266 mungkin kurang tersedia dipasaran dibandingkan Arduino, sehingga mungkin lebih sulit didapat. Dokumentasi dan sumber daya *online* untuk ESP8266 mungkin lebih terbatas dibandingkan Arduino, sehingga mungkin lebih sulit untuk memulai proyek dengan ESP8266.

- **ESP32**

ESP32 merupakan pengembangan dari ESP8266, ESP32 mempunyai banyak GPIO (seperti Arduino), memiliki modul WiFi dan ditambah dengan modul BLE (*Bluetooth Low Energy*) yang dapat digunakan secara bersamaan.

Kelebihan ESP32 untuk sistem keamanan kunci pintu gedung berbasis IoT adalah ESP32 memiliki kemampuan pemrosesan yang lebih baik dibandingkan ESP8266, sehingga lebih cocok untuk proyek yang membutuhkan pemrosesan data yang intensif. ESP32 memiliki penyimpanan data yang lebih besar dibandingkan ESP8266, sehingga lebih cocok untuk proyek yang membutuhkan penyimpanan data yang besar. ESP32 memiliki fitur keamanan yang lebih canggih dibandingkan ESP8266, sehingga lebih

cocok untuk proyek yang membutuhkan keamanan yang ketat. ESP32 memiliki kemampuan konektivitas yang kuat, sehingga mudah terhubung ke internet melalui WiFi atau *Bluetooth*.

Kelemahan ESP32 untuk sistem keamanan kunci pintu gedung berbasis IoT adalah ESP32 mungkin lebih sulit diprogram dibandingkan ESP8266, karena memerlukan pemrograman yang lebih rumit. ESP32 mungkin lebih mahal dibandingkan ESP8266, karena memiliki kemampuan yang lebih baik. ESP32 mungkin kurang tersedia dipasaran dibandingkan ESP8266, sehingga mungkin lebih sulit didapat. Dokumentasi dan sumber daya *online* untuk ESP32 mungkin lebih terbatas dibandingkan ESP8266, sehingga mungkin lebih sulit untuk memulai proyek dengan ESP32.

3) Aktuator

Aktuator adalah perangkat yang digunakan untuk mengeluarkan sinyal atau tindakan sesuai dengan perintah yang diterima dari sistem keamanan kunci pintu gedung berbasis *Internet of Things*. Berikut adalah beberapa alternatif desain aktuator yang dapat digunakan dalam sistem keamanan kunci pintu gedung berbasis *Internet of Things*:

- Motor elektronik

Motor elektronik dapat digunakan sebagai aktuator untuk membuka dan menutup pintu-pintu gedung.

Kelebihan motor elektronik untuk sistem keamanan kunci pintu gedung berbasis IoT adalah motor elektronik memiliki torsi yang lebih tinggi dibandingkan motor mekanik, sehingga lebih cocok untuk menggerakkan beban yang berat. Motor elektronik lebih efisien dibandingkan motor mekanik, karena memiliki rendahnya kehilangan daya. Motor elektronik memiliki umur pakai yang lebih panjang dibandingkan motor mekanik, karena tidak memiliki bagian-bagian mekanik yang harus diganti. Motor elektronik memiliki kecepatan yang mudah diatur, sehingga lebih mudah untuk mengontrol kecepatan gerakan sistem keamanan kunci pintu.

Kelemahan motor elektronik untuk sistem keamanan kunci pintu gedung berbasis IoT adalah motor elektronik mungkin lebih mahal dibandingkan motor mekanik, karena memiliki kemampuan yang lebih baik. Motor elektronik mungkin membutuhkan lebih banyak peralatan elektronik untuk

mengontrolnya, sehingga mungkin lebih sulit untuk dipasang dan diatur. Motor elektronik mungkin lebih rentan terhadap kerusakan akibat kelebihan arus atau tegangan, sehingga membutuhkan proteksi yang lebih baik. Motor elektronik mungkin tidak cocok untuk aplikasi yang membutuhkan kekuatan yang tinggi, karena tidak dapat menghasilkan torsi yang sama dengan motor mekanik.

- **Solenoid**

Solenoid adalah perangkat yang mengeluarkan medan magnet yang bisa digunakan untuk membuka atau mengunci pintu-pintu gedung.

Kelebihan solenoid untuk sistem keamanan kunci pintu gedung berbasis IoT adalah solenoid mudah diatur dan dioperasikan, karena hanya memerlukan arus listrik untuk bekerja. Solenoid memiliki ukuran yang kecil dan ringan, sehingga mudah dipasang dan dipindahkan. Solenoid memiliki kecepatan yang tinggi, sehingga dapat dengan cepat mengunci atau membuka kunci pintu. Solenoid memiliki umur pakai yang panjang, karena tidak memiliki bagian-bagian mekanik yang harus diganti.

Kelemahan solenoid untuk sistem keamanan kunci pintu gedung berbasis IoT adalah solenoid tidak dapat menghasilkan torsi yang tinggi, sehingga tidak cocok untuk aplikasi yang membutuhkan kekuatan yang tinggi. Solenoid mungkin tidak cocok untuk aplikasi yang membutuhkan gerakan yang lincah atau akurasi yang tinggi, karena tidak dapat menghasilkan gerakan yang halus. Solenoid mungkin tidak cocok untuk aplikasi yang membutuhkan kecepatan yang tinggi, karena tidak dapat bekerja dengan cepat dalam jangka waktu yang lama. Solenoid mungkin tidak cocok untuk aplikasi yang membutuhkan keandalan yang tinggi, karena mungkin rentan terhadap kerusakan akibat kelebihan arus atau tegangan.

- **Relay**

Relay adalah perangkat yang bisa mengeluarkan sinyal atau tindakan sesuai dengan perintah yang diterima dari sistem.

Kelebihan *relay* untuk sistem keamanan kunci pintu gedung berbasis IoT adalah *relay* dapat mengontrol arus yang lebih besar dibandingkan solenoid, sehingga lebih cocok untuk aplikasi yang membutuhkan kekuatan yang

tinggi. *Relay* memiliki umur pakai yang panjang, karena tidak memiliki bagian-bagian mekanik yang harus diganti. *Relay* dapat diatur dan dioperasikan dengan mudah, karena hanya memerlukan arus listrik untuk bekerja. *Relay* memiliki keandalan yang tinggi, karena tidak rentan terhadap kerusakan akibat kelebihan arus atau tegangan.

Kelemahan *relay* untuk sistem keamanan kunci pintu gedung berbasis IoT adalah *relay* mungkin tidak cocok untuk aplikasi yang membutuhkan gerakan yang lincah atau akurasi yang tinggi, karena tidak dapat menghasilkan gerakan yang halus. *Relay* mungkin tidak cocok untuk aplikasi yang membutuhkan kecepatan yang tinggi, karena tidak dapat bekerja dengan cepat dalam jangka waktu yang lama. *Relay* mungkin memiliki ukuran yang lebih besar dan berat dibandingkan solenoid, sehingga mungkin lebih sulit dipasang dan dipindahkan. *Relay* mungkin membutuhkan lebih banyak peralatan elektronik untuk mengontrolnya, sehingga mungkin lebih sulit untuk dipasang dan diatur.

4) *Power*

Berikut adalah beberapa alternatif desain *power* yang dapat digunakan untuk sistem keamanan kunci pintu gedung berbasis *Internet of Things*:

- **Baterai**

Baterai dapat digunakan sebagai *power* untuk sistem keamanan kunci pintu gedung berbasis *Internet of Things*. Baterai memiliki kelebihan dalam hal portabilitas dan tidak tergantung pada sumber listrik, namun memiliki masa pakai yang terbatas dan harus diganti secara teratur.

- **Adaptor listrik**

Adaptor listrik dapat digunakan sebagai *power* untuk sistem keamanan kunci pintu gedung berbasis *Internet of Things*. Adaptor listrik tergantung pada sumber listrik yang tersedia, namun memiliki masa pakai yang lebih panjang dibandingkan baterai.

c. Perancangan sistem *database* dan *server* untuk mendukung sistem keamanan *Access Control* dan pembuatan catatan *history* atau informasi

Database merupakan salah satu komponen teknologi informasi yang mutlak diperlukan bagi setiap organisasi yang ingin memiliki sistem informasi yang

terintegrasi untuk mendukung operasional organisasi dalam mencapai tujuannya. Pada sistem kunci pintu gedung berbasis IoT ini, *database* digunakan untuk menyimpan data-data yang diperlukan untuk menjalankan sistem kunci pintu gedung. Data yang akan disimpan terdiri dari beberapa tabel data seperti data pengguna, data perangkat yang terpasang, data jadwal perangkat, data riwayat akses, dan lain sebagainya. Sebuah *database* membutuhkan sebuah DBMS atau *Database Management System* yang digunakan untuk mengatur kinerja dan pengolahan data di dalam *database*. Pada pengembangan sistem kunci pintu gedung berbasis IoT, terdapat beberapa DBMS yang dapat digunakan, yaitu:

1) Ms. Access

Salah satu keunggulan *Microsoft Access* yaitu kompatibilitasnya dengan bahasa pemrograman *Structured Query Language* (SQL). Pengguna dapat mencampur dan mencocokkan kedua bahasa (VBA dan makro) untuk memprogram bentuk dan logika serta menerapkan konsep berorientasi objek. Namun, *Microsoft Access* kurang baik jika digunakan melalui *web*, sehingga aplikasi yang digunakan banyak pengguna biasanya menggunakan solusi sistem manajemen basis data yang bersifat *client-server*.

2) MySQL

MySQL merupakan salah satu sistem *database* yang banyak digunakan untuk mendukung kinerja dari berbagai aplikasi. MySQL dapat mendukung berbagai macam bahasa pemrograman sehingga dapat digunakan untuk berbagai *platform*. MySQL memberikan kemudahan pengelolaan *database* dengan dukungan dari komunitas yang banyak serta keamanan dan perkembangan *software*-nya yang cepat.

MySQL sangat cocok digunakan untuk mengelola data yang terstruktur akan tetapi untuk data yang berukuran besar MySQL akan mengalami penurunan performa. MySQL juga memiliki keterbatasan kinerja pada *server* ketika data yang disimpan melebihi kapasitas maksimal *server* karena tidak menggunakan konsep teknologi *server cluster*.

3) PostgreSQL

Dengan *PostgreSQL*, tidak ada yang dapat menuntut pelanggaran perjanjian lisensi karena tidak ada (paket) biaya lisensi yang terkait dengan perangkat

lunak. Akibatnya, *PostgreSQL* menawarkan keuntungan tambahan, termasuk bisnis yang lebih menguntungkan dengan penerapan skala besar. Tidak ada cara untuk memeriksa kepatuhan lisensi, fleksibilitas untuk mengimplementasikan konsep penelitian dan penggunaan eksperimental tanpa biaya lisensi tambahan. Postgre juga dapat menyimpan data dengan banyak baris (*multiple rows*) yang dinamakan MVCC sehingga *PostgreSQL* sangat responsif pada *high volume environments*.

Dilihat dari layanan yang diberikan, *PostgreSQL* kurang unggul dalam hal ketersediaan fungsi *built-in* dan replikasi di *PostgreSQL* belum disertakan dalam distribusi standarnya yang terbatas hanya bisa melakukan penambahan kolom, penggantian nama kolom, dan penggantian nama tabel.

4) *Firebase*

Firebase menawarkan fitur pengelolaan data yang cepat dan gratis, *Firebase* juga dapat digunakan pada berbagai *platform IoT*. Akan tetapi, *Firebase* memiliki penyimpanan data tidak terstruktur sehingga sangat sulit untuk mengelola data yang terstruktur.

Selain membutuhkan *database*, sistem kunci pintu gedung berbasis IoT juga membutuhkan sebuah *server*. *Server* digunakan untuk melakukan operasi dan pengolahan data, melakukan autentikasi, serta mencatat riwayat pengguna. Beberapa *server* yang dapat digunakan untuk mendukung kinerja dari sistem kunci pintu berbasis IoT yaitu:

1) *NodeJS*

Node.js merupakan sebuah *environment* lintas *platform* yang dibangun berdasarkan *engine JavaScript V8 Chrome*. Pembuatan aplikasi dengan *NodeJS* dilakukan melalui *virtual private server*. *NodeJS* menawarkan operasi *input/output non-blocking*, serta dibangun dengan arsitektur asinkron dan *event-driven* untuk membantu *developer* membuat berbagai *project* dengan mudah dan efisien. *NodeJS* cocok digunakan untuk aplikasi *realtime* yang membutuhkan waktu respon cepat.

2) *Laravel*

Laravel menyediakan cara mudah untuk membangun *Restful API* yang aman dengan sebuah sistem autentikasi dengan menerapkan sistem autentikasi

dengan JWT. *Restful API* digunakan untuk bertukar sumber daya *web* seperti data dalam *database*, gambar, dan file. Independen dari format sumber daya ini mengirimkannya ke klien dalam format JSON oleh URL. Klien meminta sumber daya melalui metode yang sesuai seperti (GET) permintaan jika klien ingin mendapatkan sumber daya, dan POST permintaan untuk mengirim sumber daya ke sisi *server*, atau permintaan (DELETE) untuk menghapus sumber daya tertentu.

d. Perancangan aplikasi perangkat *mobile* yang dapat *support* android dan iOS sebagai piranti akses masuk pintu gedung

Perancangan aplikasi *mobile* penguncian pintu dapat digunakan untuk mengontrol sistem penguncian pintu melalui perangkat *mobile*. Aplikasi *mobile* dapat membantu meningkatkan keamanan dan kenyamanan dengan memungkinkan pengguna untuk membuka atau mengunci pintu dengan mudah dan cepat. Aplikasi *mobile* penguncian pintu dapat membantu meningkatkan keamanan dan kenyamanan dengan memungkinkan pengguna untuk memantau dan mengontrol akses kepada pintu. Beberapa alternatif desain yang dapat digunakan untuk membuat aplikasi *mobile* yang dapat *support* android dan iOS pada pengembangan sistem keamanan kunci pintu gedung yaitu:

1) *Flutter*

Flutter adalah *framework open source* yang dikembangkan oleh *Google* untuk membuat aplikasi *mobile* yang dapat dijalankan di sistem operasi Android dan iOS. *Flutter* menyediakan fitur *Hot Reload* yang memungkinkan *developer* untuk mengedit, menambahkan, atau menghapus kode tanpa harus memulai ulang aplikasi. Ini mempercepat proses pengembangan dan memudahkan *developer* untuk mencoba ide-ide baru. *Flutter* juga menyediakan *widget* yang responsif sehingga aplikasi yang dibuat akan terlihat baik di berbagai ukuran layar.

Flutter memiliki kekurangan yaitu hanya dapat digunakan dengan bahasa pemrograman Dart, yang mungkin tidak *familiar* bagi sebagian *developer*. Meskipun *Flutter* dapat digunakan untuk membuat aplikasi yang dapat dijalankan di Android dan iOS, terkadang ada keterbatasan dalam integrasi

dengan fitur-fitur *native* pada sistem operasi tersebut. Aplikasi yang dibuat dengan *Flutter* cenderung lebih besar dibandingkan dengan aplikasi yang dibuat dengan *framework* lain karena *Flutter* membawa *library* dan komponen-komponen yang dibutuhkan untuk menjalankannya.

2) *Progressive Web Apps (PWA)*

Progressive Web Apps (PWA) adalah aplikasi *web* yang dapat di-*install* di perangkat seperti aplikasi *native*, tetapi masih dibuka dan dijalankan melalui *browser*. PWA dapat di-*install* di perangkat dengan mudah melalui *browser*, tanpa perlu mengunduh dari toko aplikasi seperti *Google Play* atau *App Store*. PWA cenderung memiliki performa yang lebih baik dibandingkan dengan aplikasi *native* karena tidak membutuhkan waktu untuk di-*download* dan di-*install*. PWA dapat dijalankan di berbagai perangkat yang mendukung *browser*, seperti *smartphone*, tablet, maupun desktop.

Dibalik dari kelebihannya, PWA terkadang memiliki keterbatasan dalam mengakses fitur-fitur perangkat seperti kamera atau GPS, tergantung pada *browser* yang digunakan. PWA terkadang memiliki keterbatasan dalam mengirim notifikasi *push* ke perangkat, tergantung pada *browser* yang digunakan.

e. Perancangan *website* untuk mendukung sistem *monitoring* dan *controlling* jarak jauh

Perancangan aplikasi berbasis *website* dimaksudkan untuk lebih memudahkan admin untuk mengendalikan atau mengatur sistem kunci pintu gedung berbasis IoT seperti mengatur penjadwalan, menambahkan pengguna baru, membuat undangan, dan lain sebagainya. Beberapa pilihan *framework* yang dapat digunakan untuk membuat *website* pada pengembangan sistem keamanan kunci pintu gedung yaitu:

1) *ReactJS*

ReactJS merupakan sebuah *library javascript* yang digunakan untuk membangun tampilan pada sebuah aplikasi berbasis *website*. Keunggulan dari *ReactJS* yaitu setiap tampilan yang dibuat menggunakan basis komponen sehingga setiap komponen dapat digunakan secara berulang pada tampilan yang

berbeda. Keunggulan lainnya yaitu *ReactJS* bersifat *Single Page Application* sehingga aplikasi yang dibuat tidak perlu melakukan *reload* halaman secara utuh setiap ada perubahan data.

2) Laravel

Laravel mengikuti pola arsitektur *Model View Controller* (MVC). MVC memisahkan aplikasi berdasarkan komponen aplikasi seperti komputasi, pengontrol, dan antarmuka pengguna. Keuntungan pengembangan aplikasi *web* menggunakan metode ini adalah dalam proses *maintenance* dan *scalability* yang lebih mudah. Dengan menggunakan laravel, sebuah *website* juga dapat bekerja dengan web API secara simultan dalam satu *server*.

f. Perancangan sistem komunikasi data dua arah untuk mendukung sistem keamanan kunci pintu gedung berbasis IoT

Untuk mendukung sistem keamanan kunci pintu gedung berbasis IoT, perlu membuat sistem komunikasi data yang dapat mengirim dan menerima data dari perangkat IoT yang terhubung ke sistem. Ada beberapa hal yang perlu dipertimbangkan dalam perancangan sistem komunikasi ini:

1) Protokol Komunikasi

Protokol komunikasi perlu disesuaikan dengan kemampuan dan operasi yang akan dijalankan dalam sistem tersebut. Beberapa protokol komunikasi standar dapat digunakan dalam pengembangan sistem keamanan kunci pintu gedung berbasis IoT seperti HTTPS dan MQTT.

- **HTTPS**

Hypertext Transfer Protocol Secure (HTTPS) adalah versi aman dari protokol jaringan internet yang paling umum, yaitu *Hypertext Transfer Protocol* (HTTP). HTTPS menggunakan enkripsi untuk mengamankan koneksi jaringan antara klien (seperti peramban *web*) dan *server*. Ini membuat komunikasi antara klien dan server tidak dapat diintip atau dimodifikasi oleh pihak ketiga yang tidak berwenang.

HTTPS biasanya digunakan untuk melakukan komunikasi jaringan yang membutuhkan keamanan tinggi, seperti transaksi finansial atau pertukaran informasi pribadi. HTTPS dapat digunakan pada berbagai jenis protokol

jaringan, termasuk *File Transfer Protocol* (FTP), *Simple Mail Transfer Protocol* (SMTP), dan *Instant Messaging Protocol* (IMAP).

Untuk menggunakan HTTPS, server harus memiliki sertifikat SSL (*Secure Sockets Layer*) atau TLS (*Transport Layer Security*). Sertifikat ini digunakan untuk memverifikasi identitas *server* dan mengamankan koneksi jaringan dengan enkripsi. Klien (seperti peramban *web*) kemudian dapat memverifikasi sertifikat tersebut untuk memastikan bahwa koneksi jaringan dengan *server* aman.

- **MQTT**

MQTT (*Message Queuing Telemetry Transport*) adalah protokol komunikasi yang digunakan untuk mentransfer data di internet terutama untuk sistem IoT. MQTT merupakan protokol yang sangat cepat dan efisien, karena tidak memerlukan banyak data *overhead*. Ini sangat penting untuk sistem keamanan kunci pintu gedung, karena harus responsif dan dapat membuka pintu secara cepat. MQTT memerlukan jumlah data yang sangat kecil untuk mentransfer pesan, sehingga sangat efisien dari segi penggunaan *bandwidth*. Akan tetapi, MQTT tidak menyediakan enkripsi secara *default*, sehingga harus menambahkan enkripsi tambahan untuk menjamin keamanan komunikasi data. Beberapa pilihan yang umum digunakan adalah enkripsi SSL/TLS (*Secure Sockets Layer/Transport Layer Security*) atau penggunaan MQTT-TLS (MQTT over TLS).

2) Modul Komunikasi

Sebuah modul komunikasi digunakan untuk menghubungkan sistem kunci pintu dengan jaringan komunikasi di luar. Penggunaan modul komunikasi disesuaikan dengan kemampuan perangkat serta metode penggunaan yang akan dipakai contohnya seperti WiFi dan *Bluetooth*.

- **Wi-Fi**

Wi-Fi adalah salah satu pilihan komunikasi yang umum digunakan untuk sistem berbasis IoT. Wi-Fi dapat menyediakan kecepatan yang cukup tinggi untuk mentransfer data, tergantung pada jenis dan kekuatan jaringan yang digunakan. Wi-Fi menyediakan enkripsi secara *default* dengan menggunakan WPA2 dan WPA3. Penggunaan jaringan Wi-Fi tidak

memerlukan biaya tambahan, kecuali jika menggunakan jaringan publik yang memerlukan pembayaran. Namun, perlu mempertimbangkan biaya perangkat yang diperlukan untuk terhubung ke jaringan Wi-Fi, seperti *router* atau *access point*.

- *Bluetooth*

Bluetooth merupakan modul komunikasi yang dapat digunakan dalam sistem IoT. Dalam sistem kunci pintu gedung berbasis IoT, diperlukan sebuah koneksi *peer-to-peer* antara perangkat kunci pintu dengan aplikasi pengguna sehingga penggunaan *bluetooth* sangat dimungkinkan dalam sistem kunci pintu ini. *Bluetooth* hanya dapat digunakan untuk mentransfer data dalam jarak yang terbatas. Ini cocok dengan kebutuhan dari sistem keamanan kunci pintu gedung untuk mendukung kinerjanya. *Bluetooth* juga menyediakan enkripsi secara *default* dengan menggunakan AES. Sehingga proses transmisi data dapat dilakukan dengan aman melalui proses enkripsi.

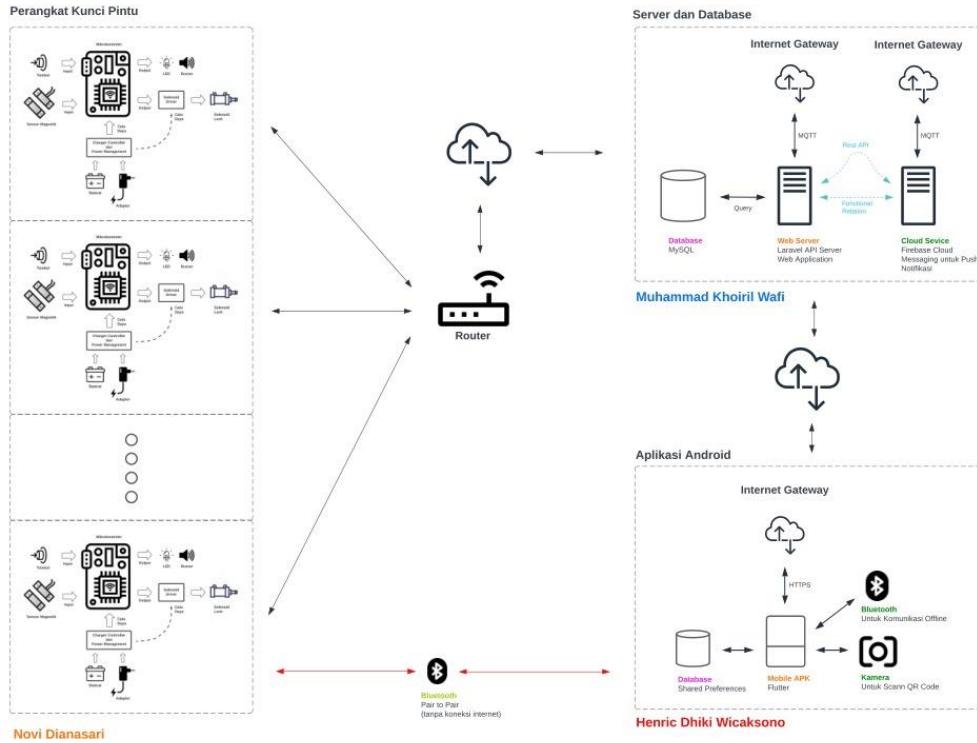
2.2 KONSEP DESAIN

2.2.1 Konfigurasi Umum

a. Bagian-Bagian Sistem Keamanan Kunci Pintu Gedung Berbasis IoT

Konfigurasi umum sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) terdiri dari beberapa komponen utama, yaitu perangkat kunci pintu, *internet gateway*, *server*, aplikasi *mobile* atau *web apps*.

Konfigurasi ini memungkinkan pihak yang berwenang (admin) untuk mengontrol dan mengakses kunci pintu gedung secara *remote* melalui perangkat yang terhubung ke internet. Selain itu, sistem ini juga memungkinkan pihak yang berwenang untuk mencatat dan mengelola data akses ke gedung secara elektronik. Adapun diagram bagian-bagian sistem keamanan kunci pintu gedung berbasis IoT dan pembagian tugas dapat dilihat pada Gambar 2.1.



Gambar 2.1 Diagram Bagian-Bagian Sistem Keamanan Kunci Pintu gedung Berbasis IoT dan Pembagian Tugas

Berdasarkan Gambar 2.1 di atas, sistem keamanan kunci pintu gedung berbasis IoT mempunyai 4 bagian utama yaitu:

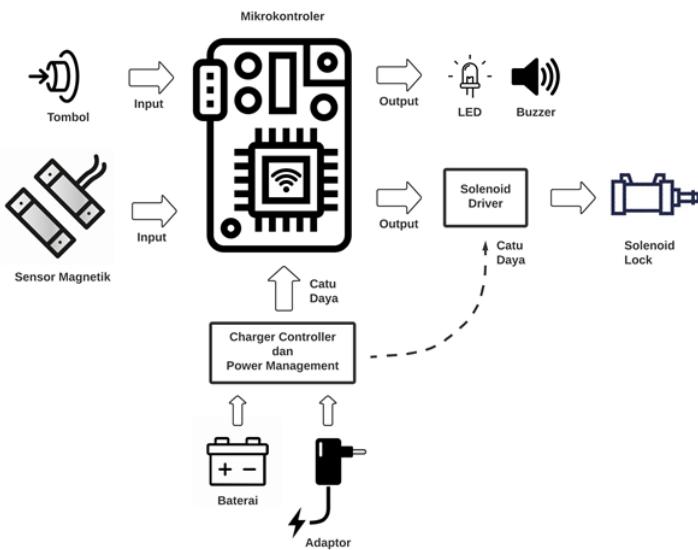
1) Perangkat Kunci Pintu

Kunci pintu yang dilengkapi dengan sensor saklar magnetik dan QR *code* yang dapat diakses melalui *smartphone* atau perangkat lain yang terhubung ke internet untuk *monitoring* dan *controlling* jarak jauh.

Perangkat kunci pintu akan terpasang pada setiap pintu dalam gedung dan akan terhubung ke *server* untuk membangun sebuah konektivitas IoT. Setiap perangkat kunci pintu dilengkapi dengan 2 modul komunikasi yaitu modul WiFi dan *Bluetooth* dengan menggunakan mikrokontroler ESP32.

Modul WiFi digunakan untuk menghubungkan perangkat kunci pintu dengan *server* utama, *server* utama akan mengirimkan perintah dan data ke perangkat kunci pintu seperti data pengguna yang diizinkan untuk membuka akses pintu tersebut, perintah penjadwalan serta perintah kontrol jarak jauh dari admin. Modul WiFi juga digunakan oleh perangkat kunci pintu untuk mengirimkan data riwayat pengguna yang mengakses pintu dan juga mengirimkan peringatan jika pintu terbuka secara tidak normal.

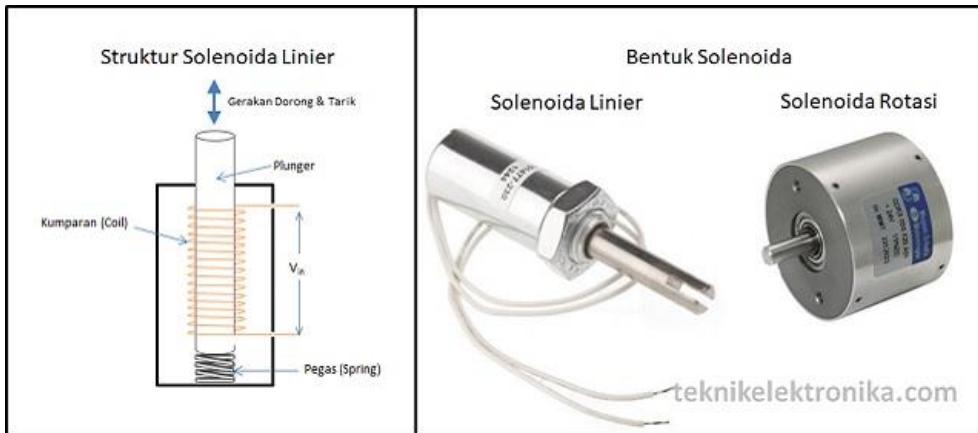
Modul *Bluetooth* digunakan untuk menghubungkan perangkat kunci pintu dengan aplikasi *mobile* pengguna secara *peer-to-peer*. Hal tersebut dimaksudkan untuk memastikan bahwa pengguna terhubung secara langsung dengan perangkat kunci pintu saat melakukan *scanning QR-Code* untuk membuka kunci pintu.



Gambar 2.2 Diagram Perangkat Kunci Pintu

Dapat dilihat pada Gambar 2.2 perangkat kunci pintu menggunakan solenoid. Solenoid adalah salah satu jenis komponen mekanik yang digunakan dalam sistem keamanan kunci pintu gedung berbasis IoT. Solenoid merupakan sebuah alat yang mengubah energi listrik menjadi energi mekanik.

Pada sistem keamanan kunci pintu gedung berbasis IoT, solenoid digunakan untuk mengontrol akses masuk ke dalam gedung dengan cara menarik atau menolak komponen mekanik yang digunakan untuk membuka atau menutup pintu. Solenoid terhubung dengan sensor magnetik yang digunakan untuk mengidentifikasi dan mengautentikasi pengguna. Untuk gambar struktur solenoida linier dapat dilihat pada Gambar 2.3.



Gambar 2.3 Struktur Solenoida Linier

Secara sains, solenoid seperti yang terlihat pada Gambar 2.3 menggunakan prinsip kerja magnet yaitu ketika arus listrik diberikan ke kumparan, kumparan tersebut akan menghasilkan medan magnet, medan magnet tersebut akan menarik *plunger* yang berada di dalam kumparan masuk ke pusat kumparan dan merapatkan atau mengkompreskan pegas yang terdapat di satu ujung *plunger* tersebut. Gaya dan kecepatan *plunger* tergantung pada kekuatan fluks magnetik yang dihasilkan oleh kumparan. Saat arus listrik dimatikan, medan elektromagnetik yang terbentuk sebelumnya menghilang, sehingga energi yang tersimpan di pegas terkompresi mendorong piston kembali ke posisi semula.

Secara matematis, solenoid dapat dihitung dengan menggunakan hukum *Faraday* yaitu:

$$\Phi = B \cdot A \cdot L = N \cdot I \cdot dt \quad (2.1)$$

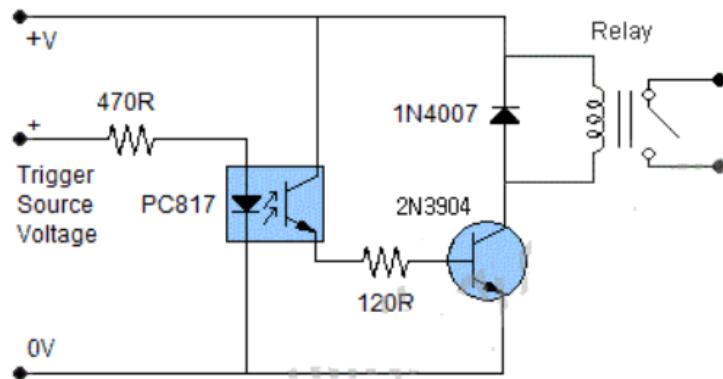
dimana Φ adalah fluks magnet, B adalah kuat medan magnet, A adalah luas penampang solenoid, L adalah panjang solenoid, N adalah jumlah lilitan, I adalah arus listrik, dan dt adalah waktu.

Pada *datasheet* solenoid bekerja pada tegangan 12volt dan mempunyai nilai resistansi 16 ohm, sehingga untuk dapat bekerja solenoid tersebut membutuhkan arus dari mikrokontroler sebesar:

$$I = \frac{V}{R} = \frac{12}{16} = 0.75 \text{ A atau } 750 \text{ mA} \quad (2.2)$$

Sedangkan mikrokontroler hanya dapat memberikan arus sebesar 250 mA, tentunya sangat kurang untuk menggerakkan solenoid. Oleh karena itu, perlu sebuah *driver* untuk menggerakkan solenoid tersebut.

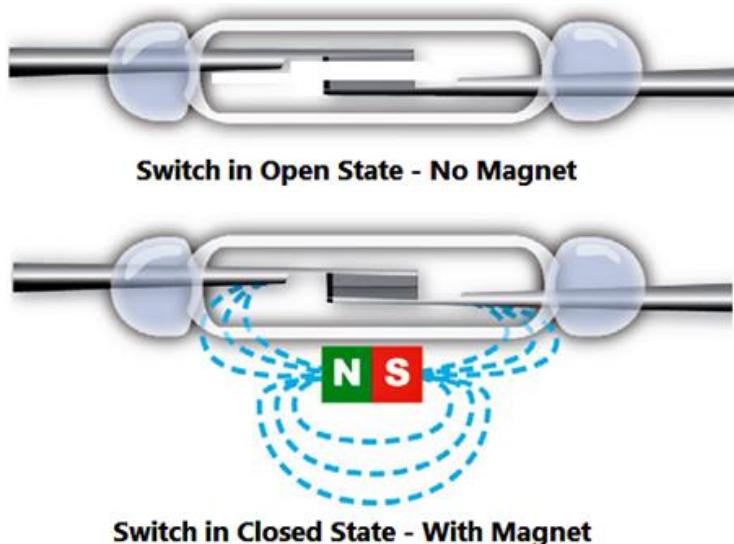
Pada Gambar 2.2 mikrokontroler yang digunakan adalah ESP32 untuk mengontrol sistem keamanan kunci pintu gedung berbasis IoT. Dalam sistem ini, ESP32 akan digunakan untuk mengontrol akses ke pintu dengan mengirimkan perintah ke sistem kunci pintu melalui koneksi internet.



Gambar 2.4 Rangkaian Driver Solenoid

Dengan menggunakan *driver* solenoid seperti yang terlihat pada Gambar 2.4 maka mikrokontroler hanya perlu menyalaikan LED pada *optocoupler* PC817 dengan karakteristik $V_f = 1.2\text{volt}$ dan $I_f = 50 \text{ mA}$. Sehingga arus yang dikeluarkan oleh mikrokontroler senilai $3.3\text{volt} 250\text{mA}$ sudah lebih dari cukup untuk menyalaikan *optocoupler* tersebut.

Pada Gambar 2.2 juga terlihat bahwa perangkat kunci pintu menggunakan sebuah sensor magnetik. Sensor magnetik tersebut digunakan untuk mengetahui kondisi pintu sedang terbuka atau tertutup, sehingga selain untuk keperluan autentikasi juga dapat memberikan informasi kepada admin tentang kondisi pintu secara *realtime*. Untuk gambar prinsip kerja sensor *magnetic* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Prinsip Kerja Sensor Magnetic

Secara sains, sensor *magnetic* ini pada dasarnya cara kerjanya sama dengan *reed switch* menggunakan prinsip kerja magnet. Saat satu bagian sensor *magnetic* terpasang pada pintu dan satu bagian lain pada bingkai pintu. Salah satu dari dua bagian tersebut memiliki magnet sementara yang lainnya memiliki *reed switch* yang menutup saat berada di dekat magnet. Ketika kedua bagian tersebut terpisah, maka medan magnet akan kehilangan kekuatannya, dan rangkaian akan terputus[4]. Sensor *magnetic* ini bersifat *normally open*. Jadi, ketika pintu tertutup, maka arus mengalir melalui *switch*, ketika pintu terbuka, maka arus akan terputus[4].

Secara matematis, sensor magnetik menggunakan hukum gaya magnet yang dijelaskan oleh hukum *Coulomb* yaitu:

$$F = k(Q_1 Q_2)/r^2 \quad (2.3)$$

dimana F adalah gaya magnet, k adalah konstanta gaya magnet, Q1 dan Q2 adalah muatan listrik, dan r adalah jarak antara kedua muatan listrik.

2) Internet Gateway

Internet gateway atau perangkat yang terhubung ke jaringan internet dan ke kunci pintu gedung. *Internet gateway* ini berfungsi sebagai perantara antara kunci pintu gedung dengan jaringan internet. Dalam sistem ini, *internet gateway* digunakan untuk mengirimkan dan menerima perintah dari ESP32 ke sistem kunci pintu melalui koneksi internet.

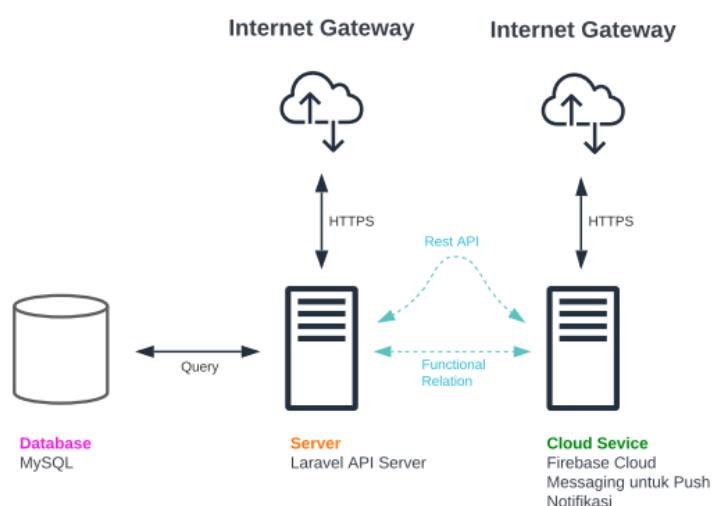
Sains dasar yang digunakan dalam *internet gateway* meliputi komunikasi dan jaringan komputer, yang digunakan untuk mengirimkan dan menerima data

melalui koneksi internet. Matematika dasar yang digunakan meliputi algoritma jaringan, yang digunakan untuk mengatur aliran data dan menjamin kualitas koneksi.

Internet gateway harus dapat menangani trafik data yang tinggi dan memiliki kemampuan enkripsi data yang kuat, untuk menjamin keamanan data yang dikirimkan dan diterima. Selain itu, *internet gateway* harus kompatibel dengan perangkat keras yang digunakan dalam sistem dan dapat diintegrasikan dengan sistem keamanan kunci pintu gedung yang ada.

3) Server dan Database

Server atau perangkat yang digunakan untuk menyimpan dan mengelola data akses ke gedung. *Server* ini terhubung ke jaringan internet dan dapat diakses oleh pihak yang berwenang melalui aplikasi *smartphone* atau perangkat lain yang terhubung ke internet. Setiap perangkat kunci pintu akan terhubung ke sebuah *server*. *Server* akan mengatur kinerja dari perangkat kunci pintu dan mencatat setiap aktivitas pengguna di dalam sebuah *database*. Diagram dari *server* dan *database* yang digunakan pada sistem kunci pintu dapat dilihat pada Gambar 2.6.



Gambar 2.6 Diagram Server dan Database

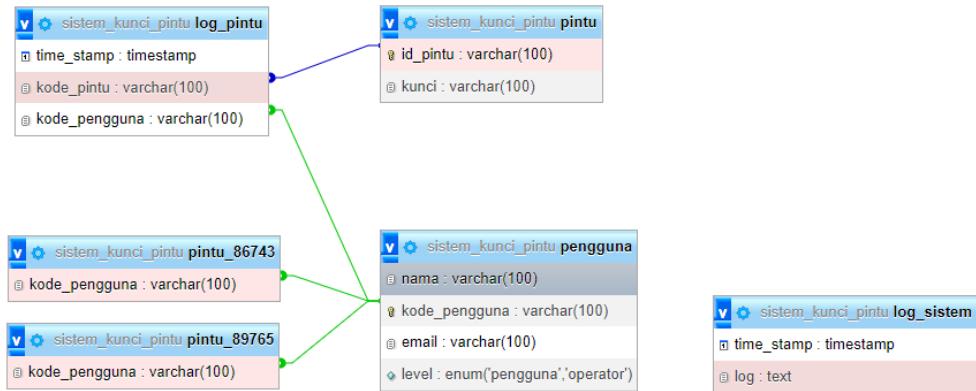
Dengan berbagai macam operasi yang dilakukan pengguna maupun admin seperti *login*, *management* pengguna, *management* perangkat kunci pintu, pencataan riwayat atau *log*, *monitoring* dan lain sebagainya maka diperlukan sebuah *server* yang mendukung operasi tersebut.

Jika menggunakan protokol standar IoT saja yaitu dengan menggunakan metode *publish* dan *subscribe* maka tidak akan mengakomodasi semua operasi tersebut. Oleh karena itu, pada pengembangan sistem keamanan kunci pintu gedung berbasis IoT ini menggunakan sebuah *server API* yang berjalan pada sebuah *cloud hosting*.

Dengan adanya kebutuhan penyimpanan data yang terstruktur baik itu berupa akun pengguna, daftar perangkat kunci pintu dalam satu sistem, daftar pengguna yang memiliki akses pada kunci pintu tertentu, dan lain sebagainya maka diperlukan sebuah sistem *database* yang dapat memudahkan pengolahan data tersebut. Sistem *database* yang paling sesuai yaitu dengan menggunakan MySQL yang dikombinasikan menggunakan Laravel *API server* dengan kelebihan masing-masing sudah dijelaskan pada bagian alternatif desain.

Pada Gambar 2.6 juga terlihat bahwa *server* terhubung dengan *cloud service* yaitu *Firebase Cloud Messaging* atau FCM. FCM digunakan untuk melakukan *push notification* baik ke admin jika terjadi penerobosan di pintu gedung maupun ke pengguna jika ada informasi yang harus dikirimkan oleh *server* ke *client*. Penggunaan FCM memudahkan dan meringankan beban kinerja dari aplikasi dan *server* karena *server* dan *client* tidak harus terhubung secara terus-menerus.

Sebuah *database* digunakan untuk menyimpan data yang diperlukan untuk keseluruhan sistem keamanan kunci pintu gedung berbasis IoT. Skema *database* yang digunakan pada sistem ini dapat dilihat pada Gambar 2.7.



Gambar 2.7 Skema Database Sistem Keamanan Kunci Pintu Gedung Berbasis IoT

Dapat dilihat pada Gambar 2.7, terdapat tabel pengguna yang digunakan untuk menyimpan data *user* baik itu untuk pengguna maupun admin yang berwenang seperti nama, kode_pengguna, email, dan level dengan penjelasan seperti pada tabel 2.1.

Tabel 2.1 Tabel Pengguna

No	Nama	Tipe	Keterangan
1	Nama	varchar(100)	Menyimpan nama pengguna
2	Kode_pengguna	varchar(100)	Menyimpan kode atau ID pengguna
3	Email	varchar(100)	Menyimpan email pengguna untuk verifikasi 2 langkah
4	Level	enum(pengguna,operator)	Menyimpan level pengguna

Pada Gambar 2.7 juga terdapat tabel pintu, tabel pintu digunakan untuk menyimpan daftar pintu yang terdapat dalam satu sistem dengan isi yaitu id_pintu dan kunci dengan penjelasan seperti pada Tabel 2.2.

Tabel 2.2 Tabel Pintu

No	Nama	Tipe	Keterangan
1	Id_pintu	varchar(100)	Menyimpan ID dari

2	Kunci	varchar(100)	pintu Untuk menyimpan kunci yang valid untuk membuka kunci pintu
---	-------	--------------	--

Pada Gambar 2.7 juga terdapat tabel pintu_xxxxx, tabel ini merupakan tabel yang dibuat oleh *server* secara otomatis sesuai dengan pintu yang ada. Tabel ini akan menyimpan daftar pengguna yang memiliki akses terhadap pintu tersebut dengan penjelasan seperti pada Tabel 2.3.

Tabel 2.3 Tabel Pintu_xxxx

No	Nama	Tipe	Keterangan
1	Kode_pengguna	varchar(100)	Kode pengguna yang mempunyai akses ke pintu

Pada Gambar 2.7 juga terdapat tabel log_pintu, tabel ini digunakan untuk menyimpan data riwayat akses pengguna pada setiap pintu dengan penjelasan seperti pada Tabel 2.4.

Tabel 2.4 Tabel Log_pintu

No	Nama	Tipe	Keterangan
1	Time_stamp	Timestamp	Menyimpan waktu akses sistem
2	Kode_pintu	varchar(100)	Menyimpan kode pintu yang diakses
3	Kode_pengguna	varchar(100)	Menyimpan kode pengguna yang mengakses pintu tersebut

Pada Gambar 2.7 juga terdapat tabel log_sistem, tabel log_sistem digunakan untuk menyimpan riwayat aktivitas dari sistem dengan penjelasan seperti pada Tabel 2.5.

Tabel 2.5 Tabel Pintu

No	Nama	Tipe	Keterangan
1	Time_stamp	Timestamp	Menyimpan waktu akses sistem
2	Log	Text	Menyimpan teks riwayat aktivitas sistem

Sains dasar yang digunakan dalam *server* dan *database* meliputi jaringan komputer dan basis data, yang digunakan untuk menyimpan dan mengelola data. Matematika dasar yang digunakan meliputi algoritma pengelolaan data, yang digunakan untuk mengelola dan mengambil keputusan dari data yang tersimpan.

Server dan *database* harus dapat menangani jumlah data yang besar dan memiliki kemampuan enkripsi data yang kuat, untuk menjamin keamanan data yang disimpan. Selain itu, *server* dan *database* harus dapat diakses secara *remote* oleh ESP32 dan *internet gateway* untuk memungkinkan akses ke data dari mana saja. *Server* dan *database* juga harus dapat diintegrasikan dengan sistem keamanan kunci pintu gedung yang ada dan diuji secara berkala untuk memastikan bahwa sistem kerja dengan baik dan stabil.

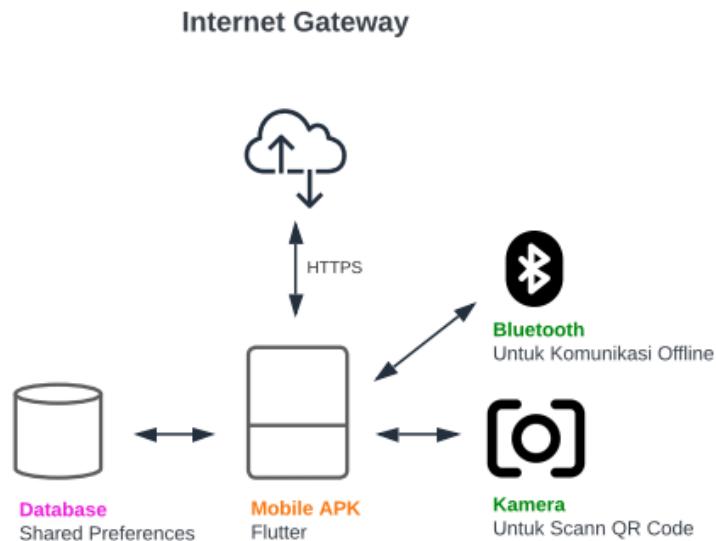
4) Aplikasi *Mobile* atau *Web Apps*

Sebuah aplikasi *mobile* dan aplikasi *web* diperlukan untuk mendukung kinerja dari sistem keamanan kunci pintu gedung yaitu digunakan untuk mengontrol dan mengakses kunci pintu gedung.

Aplikasi *mobile* ini digunakan oleh pengguna untuk membuka kunci pintu dengan cara memindai QR-Code yang ada pada pintu. Di mana aplikasi *mobile* ini dapat digunakan pada *smartphone* Android dan iOS.

Aplikasi *web* digunakan oleh admin untuk mengatur kinerja dari sistem keamanan kunci pintu gedung, seperti menambahkan pengguna baru, membuat undangan pengguna sementara, membuka kunci dari jarak jauh, mengatur

jadwal penguncian, dan lain sebagainya. Diagram dari aplikasi *mobile* dapat dilihat pada Gambar 2.8.

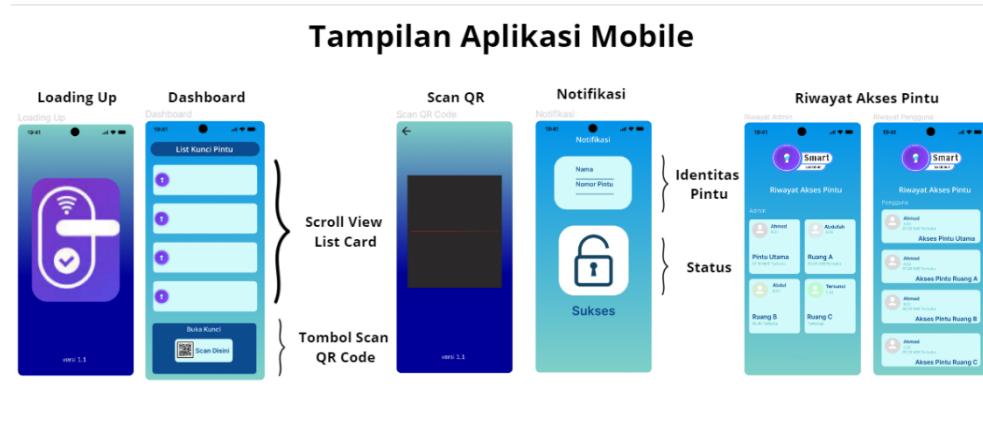


Gambar 2.8 Diagram Aplikasi *Mobile*

Terlihat pada Gambar 2.8 pada aplikasi *mobile* menggunakan kamera untuk memindai QR-Code pada pintu dan mengirimkan kode autentikasi melalui *bluetooth* untuk komunikasi *peer-to-peer* dalam membuka pintu. Alasan penggunaan *bluetooth* supaya lebih cepat atau tidak terkendala dengan kondisi WiFi, serta hanya untuk memastikan antara *device* yang digunakan dengan perangkat kunci pintu dalam kondisi baik. Jadi, *bluetooth* digunakan oleh pengguna untuk komunikasi *peer-to-peer* dengan menggunakan aplikasi *mobile* (Android atau iOS) dalam membuka pintu.

Selain itu, pada sistem ini internet (WiFi) tetap dibutuhkan untuk kebutuhan kontrol jarak jauh, pencatatan *log* aksesnya, juga untuk berkomunikasi secara *real-time* dengan *server* sebagai *command center*. Jadi, WiFi digunakan oleh admin untuk *monitoring* dan *controlling* (pengelolaan akses gedung secara penuh) dengan menggunakan *website* karena tampilannya lebih lengkap.

Adapun tampilan aplikasi *mobile* (Android dan iOS) pengguna dapat dilihat pada Gambar 2.9 berikut.

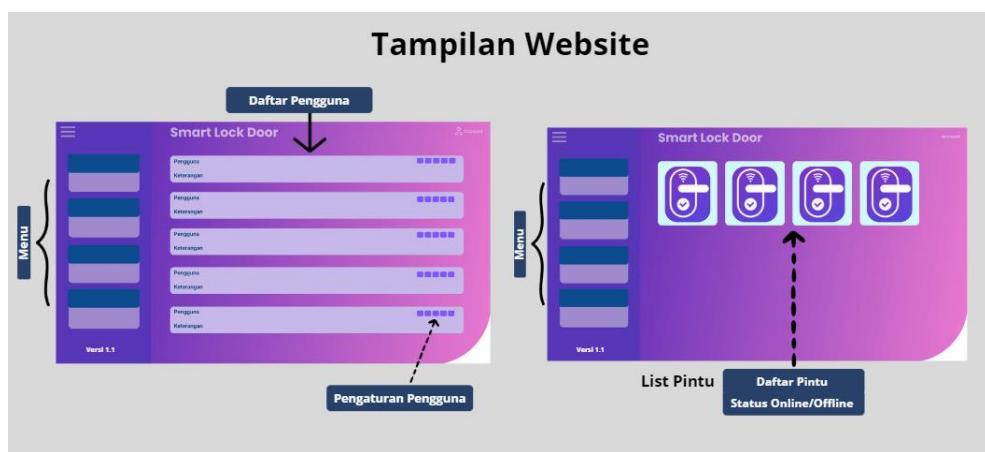


Gambar 2.9 Tampilan Aplikasi *Mobile*

Berdasarkan pada Gambar 2.9 di atas, fitur-fitur yang terdapat pada aplikasi *mobile* sistem keamanan kunci pintu gedung berbasis IoT (*Internet of Things*) ini adalah sebagai berikut:

- Autentikasi pengguna: Fitur ini memungkinkan pengguna untuk masuk ke aplikasi dengan menggunakan kredensial seperti nama pengguna dan kata sandi.
- Tampilan daftar pintu: Fitur ini menampilkan daftar pintu yang terhubung ke sistem keamanan kunci pintu gedung yang dapat diakses oleh pengguna.
- Tombol scan QR *code*: Setiap pintu dalam daftar harus memiliki tombol *scan QR code* yang dapat digunakan oleh pengguna untuk membuka pintu tersebut.
- Riwayat akses: Fitur ini mencatat riwayat akses pintu oleh pengguna dan waktu di mana pintu dibuka atau ditutup.
- Notifikasi: Fitur ini memberi notifikasi kepada pengguna ketika ada aktivitas yang terdeteksi di pintu yang terhubung ke sistem.

Adapun tampilan untuk *website* admin dapat dilihat pada Gambar 2.10 berikut.



Gambar 2.10 Tampilan *Website*

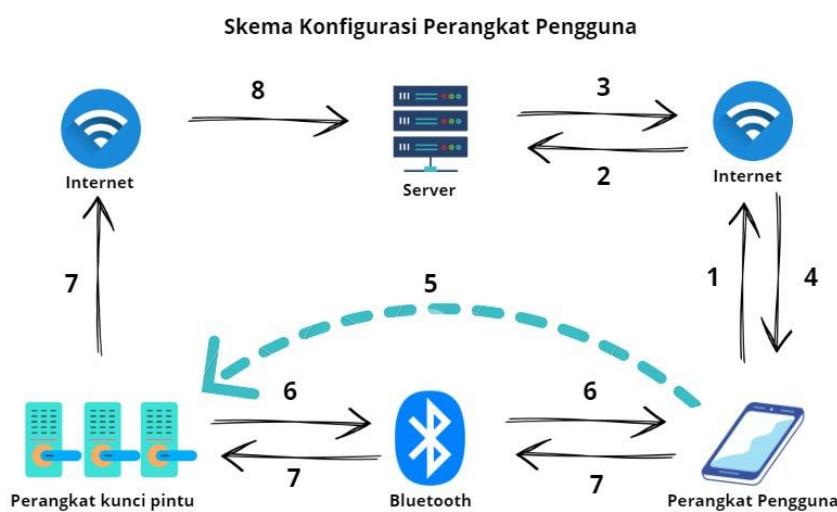
Berdasarkan pada Gambar 2.10 di atas, tampilan *website* admin pada sistem keamanan kunci pintu gedung berbasis IoT (*Internet of Things*) terdiri dari beberapa fitur utama, seperti:

- *Dashboard*: Halaman utama yang menampilkan statistik penting seperti jumlah kunci yang terhubung, jumlah akses masuk/keluar yang tercatat, dan sebagainya.
- Manajemen Kunci: Fitur yang memungkinkan admin menambah, mengedit, atau menghapus kunci yang terhubung ke sistem.
- Manajemen Pengguna: Fitur yang memungkinkan admin menambah, mengedit, atau menghapus pengguna yang terdaftar dalam sistem, serta memberikan akses kepada pengguna terhadap kunci yang terhubung.
- Laporan Akses: Fitur yang menampilkan riwayat akses masuk/keluar untuk setiap kunci yang terhubung, termasuk informasi tanggal, waktu, dan pengguna yang terlibat.
- Pengaturan Sistem: Fitur yang memungkinkan admin mengatur pengaturan sistem seperti koneksi jaringan, notifikasi email, dan sebagainya.

Secara umum, tampilan *website* admin pada sistem keamanan kunci pintu gedung berbasis IoT harus mudah digunakan dan memungkinkan admin untuk mengelola sistem dengan efisien.

b. Skema Konfigurasi Perangkat Pengguna

Adapun skema konfigurasi perangkat pengguna dapat dilihat pada Gambar 2.11.



Gambar 2.11 Skema Konfigurasi Perangkat Pengguna

Berdasarkan pada Gambar 2.11 di atas, skema konfigurasi perangkat pengguna sebagai berikut ini.

- 1) Pengguna membuka aplikasi dan aplikasi melakukan *request* kunci pintu ke *server* melalui internet.
- 2) *Server* menerima *request*.
- 3) *Server* mengirim *list* kunci pintu yang dapat diakses oleh pengguna.
- 4) Aplikasi pengguna menerima *list* kunci pintu.
- 5) Aplikasi memindai QR *Code*.
- 6) Aplikasi membuat koneksi ke perangkat kunci pintu dan mengirimkan kunci pintu ke *Bluetooth* (Data dari *list* kunci pintu yang diambil dari *server*).
- 7) Perangkat kunci pintu memberikan respon autentikasi berhasil/gagal.

c. Penggunaan QR-Code untuk membuka kunci pintu

QR *code* (*Quick Response code*) adalah jenis kode matriks yang digunakan untuk menyimpan dan memindai informasi secara cepat. Penggunaan QR *code* untuk membuka kunci pintu diperlukan sistem yang telah terintegrasi dengan pemindai QR *code* dan dapat mengontrol kunci pintu secara otomatis. Sistem ini dapat dibuat dengan menggunakan perangkat keras dan perangkat lunak yang tersedia

secara komersial atau dibuat sendiri dengan menggunakan perangkat keras dan perangkat lunak yang tersedia secara bebas.

Untuk penggunaannya nantinya pada setiap pintu akan terpasang sebuah QR-*Code* yang menjadi identitas dari pintu. QR-*Code* pada pintu bersifat statis dan hanya menyimpan ID dari pintu. QR-*Code* dibuat oleh *server* secara acak dan unik pada saat perangkat kunci pintu pertama kali didaftarkan ke dalam sistem. Untuk bisa membuka kunci pintu maka pengguna memerlukan kunci (*key*) dan kredensial *bluetooth*. Dengan cara setiap pengguna memindai QR-*Code* menggunakan aplikasi kunci pintu, maka aplikasi akan mengecek data pada penyimpanan lokal aplikasi untuk mendapatkan kunci (*key*) dari pintu beserta kredensial *bluetooth* yang akan menghubungkan aplikasi pengguna dengan perangkat kunci pintu. Data tersebut disimpan di *server* dan akan dikirimkan ke aplikasi pengguna pada saat pengguna melakukan *login* ke aplikasi. Jika informasi yang tersimpan dalam QR *code* cocok dengan yang terdaftar di sistem, maka kunci pintu akan terbuka. Jika informasi yang tersimpan dalam QR *code* tidak cocok dengan yang terdaftar di sistem, maka kunci pintu tidak akan terbuka. Sehingga dari sistem yang sudah dijelaskan, meskipun orang mempunyai QR-*Code* atau memindai QR-*Code* menggunakan aplikasi lain, maka orang tersebut tidak akan bisa membuka kunci pintu dan hanya mendapatkan data acak.

QR *code* dapat digunakan untuk membuka kunci pintu dengan mengikuti beberapa langkah-langkah pada Gambar 2.12 berikut:



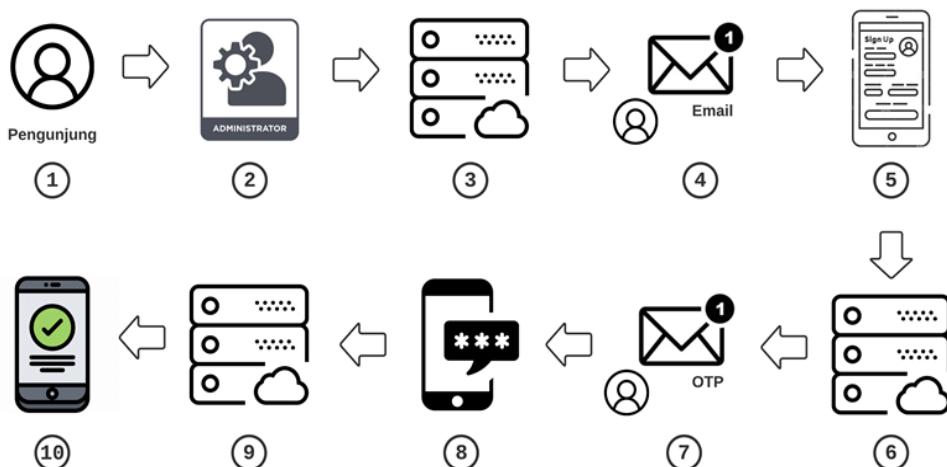
Gambar 2.12 Membuka Kunci Menggunakan QR-*Code*

- 1) Pengguna memindai QR-*Code* pada pintu menggunakan aplikasi *mobile* sehingga mendapatkan ID dari pintu.
- 2) Selanjutnya, aplikasi akan mengecek data pada penyimpanan lokal aplikasi.

- 3) Jika data ditemukan, maka aplikasi akan menghubungkan *smartphone* pengguna ke *bluetooth* sesuai dengan data yang ditemukan.
- 4) Kemudian, melalui *bluetooth* aplikasi mengirimkan *username* dan *password* pengguna yang tersimpan di aplikasi untuk membuka kunci pintu.
- 5) Jika berhasil, maka pada aplikasi akan tampil notifikasi berhasil.

d. Pendaftaran pengguna baru

Untuk pengguna yang belum mempunyai akun untuk mengakses kunci pintu, pengguna dapat mengajukan pembuatan akun ke admin dengan mengikuti langkah-langkah sesuai dengan Gambar 2.13 berikut ini.



Gambar 2.13 Pendaftaran Pengguna Baru

- 1) Pengguna baru atau pengunjung menghubungi admin dengan keperluan untuk membuat akun baru dengan mengirimkan identitas yang valid.
- 2) Admin akan memasukkan identitas pengguna ke *server*.
- 3) *Server* akan mengolah data tersebut serta menyesuaikan pengaturan level akses ke perangkat kunci pintu.
- 4) Kemudian, *server* akan menerbitkan undangan yang dikirimkan ke email pengguna.
- 5) Selanjutnya, pengguna melakukan pendaftaran melalui aplikasi yang telah disediakan.

- 6) Kemudian, *server* akan mengecek pendaftaran dan mengirimkan kode OTP untuk autentikasi 2 langkah.
- 7) Pengguna menerima kode OTP melalui email yang sudah terdaftar di *server*.
- 8) Pengguna memasukkan kode OTP ke aplikasi dan mengirimkannya ke *server*.
- 9) *Server* akan mengecek kode OTP.
- 10) Jika berhasil, maka di aplikasi akan tampil notifikasi berhasil.

Langkah-langkah tersebut juga berlaku untuk proses *login* pegawai, tetapi untuk langkah-langkah nomor 1 sampai 4 dilewati dan pada langkah ke 5 pegawai memasukkan *username* dan *password* masing-masing. Pada langkah ke 7, *server* akan membuat sebuah kode OTP yang digunakan untuk verifikasi 2 langkah. Kode OTP sendiri dibuat dengan menggunakan *timestamp* dan kunci rahasia di dalam *server* dengan langkah-langkah sebagai berikut:

- 1) *Server* membuat kunci rahasia, misalnya “ADMIN”.
- 2) Kemudian, *server* mengambil *timestamp*, misalnya “89764765547753”.
- 3) Data tersebut kemudian digabungkan menjadi “ADMIN89764765547753”.
- 4) Dari data di atas, maka dilakukan proses *hashing* menggunakan SHA-256 sehingga menghasilkan kode berikut:
“7b84e61e82892f34c0587d87afb3c9b3f1dbd55e50a30a64b6c64a70a0f44a87”
- 5) Dari *hash* tersebut diambil 5byte pertama sehingga menghasilkan kode berikut:
“7b84e61e82”
- 6) Selanjutnya, *hash* tersebut diubah ke dalam nilai desimal untuk setiap byte-nya menggunakan rumus:

$$\text{desimal} = (d_1 \times 16^1) + (d_0 \times 16^0) \quad (2.4)$$

Dimana:

d: digit

n: jumlah digit

r: posisi digit

sehingga menghasilkan nilai 711481461482

- 7) Selanjutnya, nilai desimal tersebut dipotong sesuai dengan kebutuhan kode OTP dengan menggunakan rumus berikut:

$$kode_otp = D \% (10^n)$$

(2.5)

Dimana:

d: nilai desimal

n: jumlah digit yang diperlukan

8) Misalkan kita menginginkan 6digit kode OTP maka:

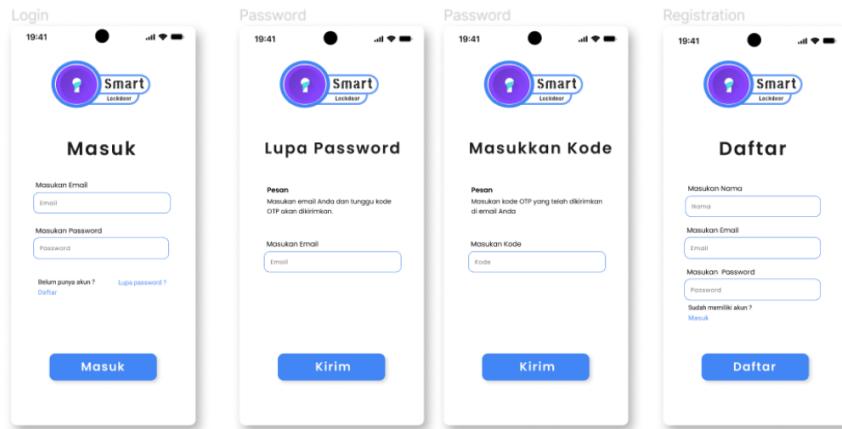
$$kode_otp = 711481461482 \% (10^6)$$

$$kode_otp = 711481461482 \% 1000000$$

$$kode_otp = 461482$$

Adapun untuk tampilan halaman *login* pengguna di aplikasi *mobile* ketika pertama kali *login* dapat dilihat pada Gambar 2.14 di bawah ini.

Halaman Login Pengguna



Gambar 2.14 Halaman *login* pengguna

Berdasarkan pada Gambar 2.14 pada tampilan halaman *login* pengguna pertama kali di aplikasi *mobile* sistem keamanan kunci pintu gedung berbasis IoT, terdapat beberapa elemen yang biasa terdapat pada halaman *login*. Elemen-elemen tersebut diantaranya:

- *Form login/masuk*: merupakan bagian yang digunakan untuk memasukkan informasi *login* seperti *username* dan *password*.
- Tombol *login/masuk*: merupakan tombol yang digunakan untuk memproses *login* dengan mengirimkan informasi *login* yang telah dimasukkan ke *server*.

- *Link lupa password*: merupakan *link* yang mengarah ke halaman untuk mengatur ulang *password* jika pengguna lupa *password*-nya.
- *Link daftar*: merupakan *link* yang mengarah ke halaman daftar jika pengguna belum memiliki akun.
- Tampilan logo aplikasi: merupakan tampilan logo aplikasi yang menunjukkan bahwa ini adalah halaman *login* aplikasi sistem keamanan kunci pintu gedung berbasis IoT.

2.2.2 Kemampuan dan Kapasitas Produk

Sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) memiliki beberapa kemampuan dan kapasitas yang dapat memberikan keamanan tambahan bagi gedung. Berikut ini beberapa kemampuan dan kapasitas sistem keamanan kunci pintu gedung berbasis IoT:

1) Kendali Jarak Jauh

Proses pengaturan kunci pintu (*lock* dan *unlock*) dapat dilakukan dari jarak jauh. Pengendalian dilakukan dengan menggunakan *website*. Kendali jarak jauh hanya bisa dilakukan oleh admin yang memiliki akses penuh atas sistem kunci pintu IoT. Jadi, salah satu keuntungan utama dari desain sistem keamanan kunci pintu gedung berbasis IoT adalah kemampuan untuk mengontrol akses ke gedung secara *remote*. Dengan sistem ini, pihak yang berwenang (admin) dapat mengakses kunci pintu gedung melalui perangkat yang terhubung ke internet, seperti *smartphone* atau komputer, dan mengontrol akses ke gedung dari jarak jauh.

Ini memungkinkan pihak yang berwenang untuk memberikan atau mencabut akses ke gedung tanpa harus berada di lokasi gedung secara fisik. Ini bisa sangat bermanfaat dalam situasi di mana pihak yang berwenang tidak bisa datang ke gedung secara fisik, misalnya karena sedang berada di luar kota atau di luar negeri.

Selain itu, dengan mengontrol akses ke gedung secara *remote*, pihak yang berwenang juga dapat memantau aktivitas akses ke gedung secara *real-time* dan membuat laporan atas aktivitas tersebut. Ini memudahkan pengelolaan akses ke gedung dan membantu menjamin keamanan gedung.

2) *History* akses

Salah satu keuntungan menggunakan desain sistem keamanan kunci pintu gedung berbasis IoT adalah kemampuan untuk mencatat aktivitas akses ke gedung. Dengan sistem ini, setiap kali seseorang masuk atau keluar dari gedung, aktivitas tersebut akan tercatat dalam sistem dan dapat diakses oleh pihak yang berwenang. Ini memungkinkan pihak yang berwenang untuk memantau aktivitas akses ke gedung dan membuat laporan atas aktivitas tersebut.

Informasi yang biasanya dicatat dalam sistem keamanan kunci pintu gedung berbasis IoT termasuk tanggal dan waktu akses, informasi tentang orang yang melakukan akses (seperti nama atau nomor identitas), dan informasi tentang pintu yang diakses. Informasi ini dapat bermanfaat untuk menganalisis pola akses ke gedung atau untuk membantu menyelidiki kejadian yang mungkin terjadi di gedung.

3) Kemudahan pengelolaan akses ke gedung

Desain sistem keamanan kunci pintu gedung berbasis IoT juga dapat memudahkan pengelolaan akses ke gedung. Dengan sistem ini, pihak yang berwenang dapat dengan mudah menambah atau menghapus akses ke gedung untuk individu tertentu. Misalnya, jika seseorang hendak memasuki gedung untuk pertama kalinya, pihak yang berwenang dapat dengan mudah menambahkan akses untuk orang tersebut ke dalam sistem. Begitu juga, jika seseorang tidak lagi memerlukan akses ke gedung, pihak yang berwenang dapat dengan mudah menghapus akses untuk orang tersebut dari sistem. Ini memudahkan pengelolaan akses ke gedung karena pihak yang berwenang tidak perlu mengeluarkan atau mengganti kunci fisik setiap kali ada perubahan dalam akses ke gedung.

4) Pengawasan

Setiap perangkat penguncian pintu akan mengawasi kondisi pintu setiap saat. Jika pintu terbuka secara tidak normal, maka sistem akan memberikan notifikasi peringatan bahwa ada pintu yang terbuka dan dimungkinkan adanya indikasi penerobosan. Jadi, sistem ini dapat memantau aktivitas pintu secara *real-time* serta memberikan notifikasi kepada pengguna jika terjadi aktivitas yang tidak diinginkan di pintu.

5) Penjadwalan

Sistem yang dibuat memungkinkan admin mengatur dan menetapkan jadwal *lock* dan *unlock* pada setiap pintu. Hal tersebut akan memudahkan jika sistem penguncian ini dipasang pada ruangan umum seperti ruang kelas atau gedung perkuliahan yang mempunyai pintu yang harus dibuka dan ditutup secara periodik.

6) Undangan

Admin dapat membuat undangan untuk memberikan akses ke pengguna. Setiap pengguna akan mendaftar pada sistem dengan menggunakan *link* yang sudah dibagikan.

7) Pengaturan Hak Akses

Admin juga dapat mengatur level akses dari setiap pengguna yang terdaftar sehingga hak akses dari setiap pengguna dapat disesuaikan dengan kondisi dan kebutuhan yang ada.

8) Kartu Pengunjung

Selain dapat mengendalikan pintu dari jarak jauh, admin juga dapat memberikan kartu pengunjung (berupa URL dan kode akses) yang dapat digunakan oleh pengunjung untuk mendapatkan akses pada pintu tertentu. Akses ini bersifat sementara dan hanya valid pada periode waktu tertentu.

9) *Unlock* Cepat

Di setiap pintu yang telah terpasang perangkat penguncian akan ditandai dengan QR-code. Setiap pengguna (admin, pengguna, dan pengunjung) dapat memindai QR-code tersebut untuk membuka pintu secara cepat dan aman.

10) Keamanan

Keamanan menjadi fokus utama dalam mengerjakan proyek ini, setiap pengguna memiliki *username* dan *password* serta *email* sebagai identitas pengguna. Pengguna akan diminta melakukan *login* pada saat pertama kali meng-*install* aplikasi serta memasukkan kode autentikasi yang dikirim melalui *email* (autentikasi 2 arah). Setiap aktivitas yang dilakukan oleh pengguna akan tercatat di *database* sebagai *log* aktivitas.

2.2.3 Dasar Teori yang Mendukung Proses Pengembangan

a. Saklar Magnetik/ *Magnetic Switch*

Magnetic Switch/ Door Sensor merupakan saklar yang dapat merespon medan magnet yang berada di sekitarnya. *Magnetic switch* ini seperti sensor *limit switch* dengan tambahan pelat logam yang dapat bereaksi dengan adanya magnet. *Magnetic Switch* biasa digunakan untuk pengamanan pada pintu dan jendela[5]. *Switch* ini di dalamnya terdapat dua buah lempengan logam yang terbuat dari nikel dan besi, dimana secara keadaan elektromagnetik *door sensor* ini adalah *normally open*. Ketika magnet diletakkan di dekat *Electromagnetic door sensor* maka dua lempengan logam akan menempel dan *switch* ini akan tersambung sehingga keadaannya adalah *normally closed*. Ketika magnet dijauhkan dari *switch* ini, maka *reed switch* akan kembali ke posisi semula yaitu *normally open*.

Adapun gambar dari *Magnetic Switch* dapat dilihat pada gambar 2.15 di bawah ini.



Gambar 2.15 *Magnetic Switch/ Door Sensor*

Saklar magnetik dapat digunakan dalam sistem keamanan kunci pintu gedung berbasis *internet of things* (IoT) dengan cara memasang saklar magnetik pada pintu gedung dan menghubungkannya ke sistem keamanan yang terhubung ke internet. Saat pintu dibuka, medan magnet akan terputus dan sistem keamanan akan memberikan notifikasi kepada pemilik gedung atau petugas keamanan

melalui aplikasi. Dengan demikian, saklar magnetik dapat digunakan sebagai tambahan keamanan untuk mencegah akses pintu gedung yang tidak sah.

Selain itu, saklar magnetik juga dapat digunakan untuk memantau keadaan pintu gedung secara *real-time*. Saat pintu terbuka, sistem keamanan dapat memberikan notifikasi kepada pemilik gedung atau petugas keamanan, sehingga dapat segera diambil tindakan jika terjadi keadaan yang tidak diinginkan. Dengan demikian, saklar magnetik dapat meningkatkan tingkat keamanan gedung dan memberikan kemudahan bagi pemilik gedung untuk memantau keadaan gedung secara *real-time*.

b. *Solenoid Door Lock*

Solenoid door lock merupakan alat elektromekanik yang berfungsi sebagai pengunci pintu otomatis[6]. Dalam kondisi normal *solenoid door lock* dalam posisi terkunci jika diberi tegangan maka *solenoid door lock* akan terbuka[6]. Tegangan yang dibutuhkan untuk menjalankan perangkat ini sebesar 12vdc, di dalamnya terdapat *coil* kawat tembaga[6]. Jika kawat tembaga dialiri arus listrik maka akan terjadi medan magnet untuk menghasilkan gaya magnet yang akan menarik inti besi kedalam[6]. *Solenoid door lock* ini dapat dihubungkan ke arduino untuk kunci pintu otomatis[6].

Tampilan *Solenoid door lock* dapat dilihat pada gambar 2.16 di bawah ini.



Gambar 2.16 Solenoid Door Lock

Pada sistem keamanan kunci pintu gedung berbasis *internet of things*, solenoid dapat digunakan untuk mengontrol akses masuk ke dalam gedung. Misalnya, saat seseorang ingin masuk ke dalam gedung, mereka harus memasukkan kode akses. Jika kode akses yang dimasukkan cocok dengan *database* yang tersimpan, maka solenoid akan diaktifkan dan menggerakkan inti ke arah yang tepat untuk membuka kunci pintu.

Selain itu, solenoid juga dapat digunakan untuk mengontrol akses ke ruangan-ruangan tertentu di dalam gedung. Misalnya, solenoid dapat digunakan untuk mengontrol akses masuk ke ruangan *server* atau ruangan penting lainnya dengan cara yang sama seperti yang telah dijelaskan di atas. Dengan menggunakan solenoid, sistem keamanan kunci pintu gedung berbasis *internet of things* dapat lebih efektif dan mudah diintegrasikan dengan sistem keamanan lainnya, seperti sistem pengawasan CCTV atau sistem pemantauan akses.

c. ESP32

ESP32 adalah mikrokontroler yang diperkenalkan oleh *Espressif System* merupakan penerus dari ESP8266[7]. Selain itu, ESP32 juga memiliki keunggulan dibandingkan mikrokontroler lainnya, mulai dari *output* pin yang lebih banyak, pin analog yang lebih banyak, memori yang lebih besar, dan *Bluetooth* 4.0 yang hemat energi. Mikrokontroler ini sudah menyertakan modul WiFi dalam *chip* prosesor *dual-core* yang berjalan pada instruksi Xtensa LX16, sehingga mendukung pembuatan sistem aplikasi IoT dengan sangat baik. Memori ESP32 terdiri atas 448 kB ROM, 520 kB SRAM, dua 8 kB RTC *memory*, dan *flash memory* 4MB[7]. *Chip* ini mempunyai 18 pin ADC (12-bit), empat unit SPI, dan dua unit I2C[7]. Kelebihan utama dari mikrokontroler ini adalah harganya relatif murah, mudah diprogram, dan memiliki jumlah pin I/O yang cukup. Tampilan ESP32 dapat dilihat pada gambar 2.17 di bawah ini.



Gambar 2.17 ESP32

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, ESP32 dapat digunakan sebagai kontroler utama yang bertugas untuk mengontrol seluruh sistem keamanan. Modul ini dapat terhubung ke jaringan internet melalui WiFi atau *Bluetooth*, sehingga dapat terkoneksi dengan perangkat lain yang terhubung ke internet, seperti *smartphone* atau komputer.

Dengan menggunakan ESP32, sistem keamanan kunci pintu gedung dapat diakses dan dikontrol secara *remote* melalui internet. Misalnya, pengguna dapat membuka atau mengunci pintu gedung dengan menggunakan aplikasi *smartphone* yang terhubung ke internet. Selain itu, ESP32 juga dapat digunakan untuk mengirim notifikasi ke pengguna saat ada aktivitas yang tidak diinginkan di area gedung, seperti masuknya orang yang tidak memiliki izin.

Di samping itu, ESP32 juga dapat digunakan untuk mengumpulkan data dari sensor-sensor yang terpasang di gedung. Data tersebut dapat diolah dan dianalisis oleh ESP32 untuk mengambil keputusan yang sesuai, seperti membuka atau mengunci pintu gedung. Dengan demikian, ESP32 merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*.

d. *Flutter* SDK

Flutter adalah SDK untuk pengembangan aplikasi *mobile* dengan kinerja tinggi, aplikasi untuk iOS dan Android, dari satu *codebase* (basis kode) yang dibuat oleh Google dengan lisensi *open source*. Tujuannya adalah memungkinkan pengembang untuk menghadirkan aplikasi berkinerja tinggi[8].

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, *Flutter* SDK dapat digunakan untuk membuat aplikasi yang akan di-*install* pada *smartphone* pengguna. Aplikasi tersebut dapat digunakan oleh pengguna untuk mengakses dan mengontrol sistem keamanan kunci pintu gedung secara *remote* melalui internet.

Flutter memiliki banyak fitur yang memudahkan pengembangan aplikasi *mobile*, seperti *hot reload*, yang memungkinkan *developer* untuk memperbarui kode aplikasi tanpa harus mengompilasi ulang. Selain itu, *Flutter* juga memiliki seperangkat *widget* yang dapat digunakan untuk membangun *interface* aplikasi dengan cepat dan mudah. Dengan demikian, *Flutter* SDK merupakan pilihan yang tepat untuk membangun aplikasi *mobile* yang akan digunakan dalam sistem keamanan kunci pintu gedung berbasis *internet of things*.

e. *MySQL*

MySQL adalah sistem manajemen basis data sumber terbuka. MySQL adalah sistem manajemen basis data relasional. Dengan kata lain, data yang dikelola dalam *database* ditempatkan dalam tabel terpisah, yang secara signifikan mempercepat pemrosesan data. *Database* dari kecil hingga sangat besar dapat dikelola dengan MySQL. Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, MySQL dapat digunakan untuk menyimpan data yang dibutuhkan oleh sistem, seperti data akses pengguna, data kunci pintu, dan data lainnya yang berkaitan dengan sistem keamanan. Dengan menggunakan MySQL, data tersebut dapat dengan mudah diakses dan dimanipulasi oleh sistem keamanan kunci pintu gedung untuk mengambil keputusan yang sesuai.

Dengan demikian, MySQL merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*. MySQL memungkinkan sistem untuk menyimpan dan mengolah data dengan cepat dan efisien, sehingga sistem dapat beroperasi dengan baik dan menjamin keamanan yang optimal bagi gedung yang diproteksi.

f. *Cloud Hosting*

Cloud hosting adalah jenis *hosting web* yang menggunakan banyak *server* untuk menyeimbangkan beban dan memaksimalkan kinerja. Contohnya, *cloud* sebagai *web* dari beberapa komputer berbeda dan semuanya akan saling terhubung.

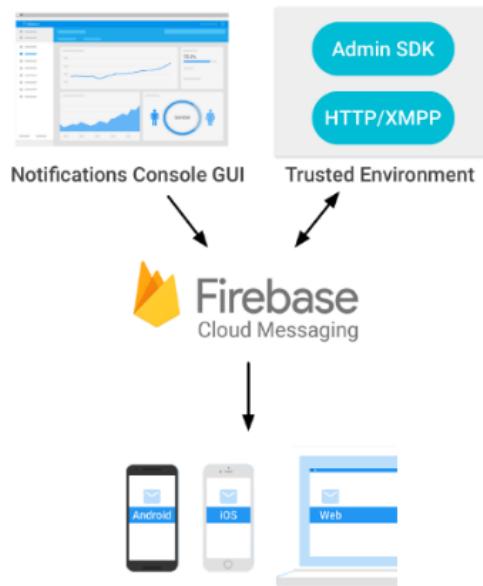
Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, *cloud hosting* dapat digunakan untuk menyimpan dan menjalankan aplikasi yang digunakan untuk mengontrol sistem keamanan. Dengan menggunakan *cloud hosting*, aplikasi tersebut dapat diakses secara *remote* melalui internet, sehingga pengguna dapat mengakses sistem keamanan dari mana saja dengan koneksi internet.

Cloud hosting juga memiliki keuntungan lain seperti skalabilitas yang tinggi, yang memungkinkan sistem keamanan kunci pintu gedung untuk menangani jumlah akses yang tinggi tanpa terganggu. Selain itu, *cloud hosting* juga menyediakan keamanan yang tinggi, sehingga data yang disimpan aman dari serangan *cyber* atau kebocoran data. Dengan demikian, *cloud hosting* merupakan pilihan yang tepat untuk mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*.

g. *Firebase Cloud Messaging (FCM)*

Firebase Cloud Messaging (FCM) adalah layanan perpesanan lintas *platform* gratis dari *Google*. FCM juga menawarkan fungsi untuk membuat *push notification*, yaitu notifikasi yang muncul di bagian atas layar *smartphone* dan dapat ditarik ke bawah. Untuk mengakses seluruh pesan, pengguna cukup menekan pesan yang ditampilkan di notifikasi. Menggunakan fitur *push notification* dengan FCM sangat berguna karena FCM mengirimkan notifikasi secara *real time*.

(Proses pengiriman pesan notifikasi dari aplikasi melalui FCM ke perangkat klien dapat dilihat pada Gambar 2.18.



Gambar 2.18 Proses Pengiriman Notifikasi FCM

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, FCM dapat digunakan untuk mengirim notifikasi ke pengguna saat terjadi aktivitas yang tidak diinginkan di gedung, seperti masuknya orang yang tidak memiliki izin. Dengan menggunakan FCM, pengguna dapat dengan cepat menerima notifikasi dan mengambil tindakan yang diperlukan, seperti memanggil polisi atau mengunci pintu gedung.

Selain itu, FCM juga dapat digunakan untuk mengirim notifikasi ke pengguna saat ada perubahan pada sistem keamanan kunci pintu gedung, seperti perubahan kode akses atau perubahan izin akses pengguna. Dengan demikian, FCM merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, karena memungkinkan sistem untuk memberikan informasi ke pengguna secara cepat dan efisien.

h. JSON

JSON (*JavaScript Object Notation*) adalah format sederhana untuk memasukkan data ke dalam variabel. Mudah bagi manusia untuk memahami dan menerapkan dan mudah bagi komputer untuk menganalisis. JSON adalah bagian dari bahasa pemrograman JavaScript (standar ECMA-262, edisi ke-3 - Desember 1999). JSON merupakan format teks yang sepenuhnya *independent*, tetapi menggunakan konvensi yang *familiar* dengan bahasa pemrograman dari keluarga-C, termasuk C,

C++, C#, Java, JavaScript, Perl, Python, dan sebagainya. Keunggulan ini menjadikan JSON sebagai bahasa komunikasi yang ideal.

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, JSON dapat digunakan untuk menyimpan dan mengirimkan data yang dibutuhkan oleh sistem, seperti data akses pengguna, data kunci pintu, dan data lainnya yang berkaitan dengan sistem keamanan. Dengan menggunakan JSON, data-data tersebut dapat dengan mudah diakses dan dimanipulasi oleh sistem keamanan kunci pintu gedung untuk mengambil keputusan yang sesuai.

Dengan demikian, JSON merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*. JSON memungkinkan sistem untuk menyimpan dan mengirimkan data dengan cepat dan mudah, sehingga sistem dapat beroperasi dengan baik dan menjamin keamanan yang optimal bagi gedung yang diproteksi.

i. WiFi

WiFi adalah teknologi terkenal yang memanfaatkan peralatan elektronik untuk bertukar data secara nirkabel (menggunakan gelombang radio) melalui jaringan komputer, termasuk koneksi internet berkecepatan tinggi[9]. Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, WiFi dapat digunakan sebagai salah satu cara untuk terhubung ke internet. Dengan menggunakan WiFi, sistem keamanan kunci pintu gedung dapat terkoneksi ke internet dan berkomunikasi dengan perangkat lain yang terhubung ke internet, seperti *smartphone* atau komputer.

Dengan menggunakan WiFi, sistem keamanan kunci pintu gedung dapat diakses dan dikontrol secara *remote* melalui internet. Misalnya, pengguna dapat membuka atau mengunci pintu gedung dengan menggunakan aplikasi *smartphone* yang terhubung ke internet. Selain itu, WiFi juga dapat digunakan untuk mengumpulkan data dari sensor-sensor yang terpasang di gedung.

Di samping itu, WiFi juga memiliki keuntungan lain seperti kecepatan yang tinggi dan biaya yang relatif rendah, sehingga menjadi pilihan yang tepat untuk digunakan dalam sistem keamanan kunci pintu gedung berbasis *internet of things*. Dengan demikian, WiFi merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*.

j. Bluetooth

Bluetooth adalah spesifikasi industri untuk *Personal Area Network (PAN)* nirkabel. *Bluetooth* membuat koneksi dan dapat digunakan untuk bertukar data antar perangkat. Spesifikasi perangkat *Bluetooth* ini dikembangkan dan didistribusikan oleh *Bluetooth Special Interest Group*. *Bluetooth* beroperasi di pita frekuensi 2,4 GHz menggunakan frekuensi melompat, mampu menyediakan layanan komunikasi data dan suara *real-time* antara *host Bluetooth* dalam jangkauan terbatas.

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, *Bluetooth* dapat digunakan untuk terhubung dengan perangkat-perangkat lain yang terpasang di gedung, seperti sensor-sensor atau kunci pintu. Dengan menggunakan *Bluetooth*, sistem keamanan kunci pintu gedung dapat menerima *input* dari perangkat-perangkat tersebut dan mengambil tindakan yang diperlukan. Selain itu, *Bluetooth* juga dapat digunakan untuk mengirimkan notifikasi ke pengguna saat terjadi aktivitas yang tidak diinginkan di gedung, seperti masuknya orang yang tidak memiliki izin. Dengan menggunakan *Bluetooth*, pengguna dapat menerima notifikasi secara *real-time* dan mengambil tindakan yang diperlukan, seperti memanggil polisi atau mengunci pintu gedung.

Bluetooth memiliki keuntungan lain seperti biaya yang relatif rendah dan mudah digunakan, sehingga menjadi pilihan yang tepat untuk digunakan dalam sistem keamanan kunci pintu gedung berbasis *internet of things*. Dengan demikian, *Bluetooth* merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*.

k. *Laravel Web Apps*

Laravel adalah kerangka kerja pengembangan *web MVC* yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan dan pemeliharaan serta meningkatkan produktivitas kerja dengan sintaks yang bersih dan fungsional yang dapat mengurangi waktu penerapan. *Laravel* merupakan *framework* dengan versi PHP yang *up-to-date*, karena *Laravel* mensyaratkan PHP versi 5.3 ke atas. *Laravel* adalah *framework* PHP yang menekankan kesederhanaan dan fleksibilitas dalam desainnya. *Laravel* menyediakan alat yang diperbarui untuk berinteraksi dengan *database* yang disebut *Migrasi*. Dengan *Migrasi*, pengembang dapat dengan mudah untuk melakukan modifikasi secara *independen* karena implementasi skema *database*

direpresentasikan dalam sebuah *class*. Migrasi dapat dilakukan dengan beberapa *database* yang didukung oleh Laravel (MySQL, PostgreSQL, MSSQL dan SQLITE) dan implementasi *Active Record* di Laravel disebut *Eloquent*, yang menggunakan standar OOP *modern*. Laravel juga memberikan sebuah *Command Line Interface* disebut dengan artisan dengan artisan, pengembang dapat berinteraksi dengan aplikasi untuk sebuah aksi seperti *migrations*, *testing*, atau membuat *controller* dan model[10]. Selain itu, laravel juga memiliki *Blade template engine* yang memberikan estetika dan kebersihan kode pada *view* secara parsial[10].

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, Laravel dapat digunakan untuk membangun aplikasi *web* yang digunakan untuk mengontrol sistem keamanan. Dengan menggunakan Laravel, aplikasi tersebut dapat dikembangkan dengan cepat dan mudah, sehingga pengguna dapat dengan mudah mengakses sistem keamanan dari mana saja dengan koneksi internet.

Selain itu, Laravel juga dapat digunakan untuk mengembangkan aplikasi *web* yang digunakan untuk menyimpan dan mengolah data yang diperoleh dari sensor-sensor yang terpasang di gedung. Data tersebut dapat diolah dan dianalisis oleh sistem keamanan kunci pintu gedung untuk mengambil tindakan yang sesuai, seperti membuka atau mengunci pintu gedung.

Dengan demikian, Laravel merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*.

1. Laravel API Server

Laravel API *server* adalah sebuah aplikasi yang dikembangkan menggunakan *framework* Laravel yang digunakan untuk membuat API (*Application Programming Interface*). API merupakan sekumpulan fungsi yang tersedia untuk digunakan oleh aplikasi, yang memungkinkan aplikasi tersebut untuk berkomunikasi dengan sistem lain atau mengakses data yang tersimpan di sistem tersebut.

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, Laravel API *server* dapat digunakan untuk membuat API yang digunakan oleh aplikasi yang terhubung dengan sistem keamanan kunci pintu gedung.

Dengan menggunakan Laravel API *server*, aplikasi tersebut dapat dengan mudah

mengakses data yang diperlukan dari sistem keamanan kunci pintu gedung, seperti data akses pengguna atau data kunci pintu.

Selain itu, Laravel API *server* juga dapat digunakan untuk mengirimkan data yang diperoleh dari sensor-sensor yang terpasang di gedung ke aplikasi yang terhubung dengan sistem keamanan kunci pintu gedung. Data tersebut dapat diolah dan dianalisis oleh sistem keamanan kunci pintu gedung untuk mengambil tindakan yang sesuai, seperti membuka atau mengunci pintu gedung.

Dengan demikian, Laravel API server merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*.

m. One Time Password

One Time Password terdiri dari 2 kategori besar, yaitu HOTP (HMAC-based OTP) dan TOTP (*Time-based OTP*)[11]. *One Time Password* (OTP) yang disebut dengan istilah sandi sekali pakai, biasanya digunakan untuk transaksi *online* atau pendaftaran sebuah akun[11]. Kode OTP terdiri dari kombinasi nomor unik dan rahasia yang diperoleh secara acak, di mana kode OTP dimaksudkan untuk keamanan dan OTP dianggap lebih aman karena perubahan *password* secara terus-menerus[11].

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, OTP dapat digunakan untuk meningkatkan keamanan akses pengguna ke sistem. Misalnya, saat pengguna ingin mengakses sistem keamanan kunci pintu gedung, sistem akan mengirimkan OTP ke perangkat *mobile* pengguna. Pengguna harus memasukkan OTP tersebut ke sistem untuk dapat masuk dan mengakses sistem keamanan kunci pintu gedung.

Dengan menggunakan OTP, sistem keamanan kunci pintu gedung dapat menjadi lebih aman karena hanya pengguna yang memiliki OTP yang dapat mengakses sistem. Selain itu, OTP juga dapat digunakan untuk menghindari akses yang tidak sah ke sistem keamanan kunci pintu gedung, seperti masuknya orang yang tidak memiliki izin.

Dengan demikian, OTP merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*. OTP memungkinkan sistem untuk menjadi lebih aman dan menghindari akses yang tidak sah ke sistem keamanan kunci pintu gedung.

n. *QR-Code*

Qr code merupakan teknik yang mengubah data tertulis menjadi kode-kode 2 dimensi yang tercetak ke dalam suatu media yang lebih ringkas. *QR code* adalah *barcode* 2 dimensi yang diperkenalkan pertama kali oleh perusahaan Jepang Denso Wave pada tahun 1994[12]. *Barcode* ini pertama kali digunakan untuk pendataan inventaris produksi suku cadang kendaraan dan sekarang sudah digunakan dalam berbagai bidang[12]. *QR* adalah singkatan dari *Quick Response* karena bertujuan untuk menerjemahkan kontennya dengan cepat. *QR-Code* salah satu tipe dari *barcode* yang dapat dibaca dengan kamera *handphone*[12].

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*, *QR code* dapat digunakan untuk mempermudah akses pengguna ke sistem keamanan kunci pintu gedung. Misalnya, saat pengguna ingin masuk ke gedung, pengguna hanya perlu meng-scann *QR code* yang terpasang di pintu gedung menggunakan aplikasi *QR code reader*. Sistem keamanan kunci pintu gedung akan mengenali *QR code* tersebut dan membuka pintu gedung untuk pengguna.

Selain itu, *QR code* juga dapat digunakan untuk mengirimkan informasi yang diperlukan oleh sistem keamanan kunci pintu gedung, seperti data akses pengguna atau data kunci pintu. Dengan menggunakan *QR code*, data tersebut dapat dengan mudah diakses oleh sistem keamanan kunci pintu gedung untuk mengambil tindakan yang sesuai.

Dengan demikian, *QR code* merupakan komponen penting yang mendukung proses pengembangan sistem keamanan kunci pintu gedung berbasis *internet of things*.

2.2.4 Teknologi yang Digunakan

Teknologi yang digunakan pada sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) tergantung pada jenis sistem yang digunakan. Berikut ini beberapa teknologi yang digunakan pada sistem keamanan kunci pintu gedung berbasis IoT:

- a. Pemindai *QR code*
- b. Kontroler akses
- c. Perangkat kontrol kunci pintu
- d. Sistem operasi

- e. Sensor
- f. Aktuator
- g. Mikrokontroler
- h. Aplikasi *Mobile* pemindai QR-*Code*
- i. *Database*
- j. *Server*
- k. *Cloud Hosting*
- l. *Firebase Cloud Messaging*
- m. JSON
- n. WiFi
- o. *Bluetooth*
- p. *Website*
- q. Laravel API *Server*
- r. One Time Password
- s. QR-*Code*

2.2.5 Batasan – Batasan Sistem

Berikut ini merupakan batasan – batasan dalam perancangan sistem keamanan kunci pintu gedung berbasis *Internet of Things*:

- a. Komunikasi antara perangkat kunci pintu dengan aplikasi *mobile* dilakukan secara *realtime*. Setiap terjadi perubahan kondisi pada pintu, maka perangkat akan langsung mengirimkan notifikasi ke aplikasi pengguna melalui *server* yang tersedia serta mencatat *log* perubahan dan disimpan di dalam *database*.
- b. Untuk proses membuka pintu, perangkat akan membuat sebuah kunci yang disimpan di dalam perangkat dan akan dibagikan oleh *server*. Pada saat pengguna masuk ke aplikasi, *server* akan mengirimkan kunci pintu sesuai dengan level aksesnya. Kunci tersebut digunakan untuk melakukan autentikasi ditambah dengan *username* dan *password* untuk menambah tingkat keamanan penguncian.
- c. Aplikasi pengguna akan menyimpan kunci yang digunakan untuk autentikasi sehingga hanya pengguna tertentu saja yang dapat membuka.
- d. *Web apps* memiliki akses untuk mengatur semua pengguna yang dapat membuka pintu dan dapat membuka pintu dari jarak jauh menggunakan koneksi internet.

- e. Pada aplikasi pengguna terdapat fitur *scann-QR* yang digunakan untuk membuka pintu hanya dengan memindai QR-*Code* pada pintu.
- f. Keamanan: Sistem ini harus memastikan bahwa tidak ada yang dapat mengakses sistem tanpa izin dan tidak ada yang dapat memanipulasi atau membajak sistem.
- g. Integritas data: Sistem keamanan kunci pintu gedung berbasis IoT harus memastikan integritas data yang disimpan dan ditransmisikan, agar tidak ada yang dapat mengubah atau memalsukan data tersebut.
- h. Ketersediaan: Sistem keamanan kunci pintu gedung berbasis IoT harus selalu tersedia dan dapat diakses oleh pengguna yang memiliki izin, karena ini merupakan bagian penting dari keamanan gedung.
- i. Skalabilitas: Sistem keamanan kunci pintu gedung berbasis IoT harus mampu menangani jumlah pengguna yang beragam dan meningkat dengan mudah.
- j. Biaya: Sistem keamanan kunci pintu gedung berbasis IoT mungkin membutuhkan biaya yang lebih tinggi daripada sistem keamanan tradisional, terutama jika harus diimplementasikan di banyak lokasi atau gedung.
- k. Kompatibilitas: Sistem keamanan kunci pintu gedung berbasis IoT harus kompatibel dengan perangkat yang digunakan oleh pengguna, seperti ponsel pintar atau *smartphone*.
- l. Performa: Sistem keamanan kunci pintu gedung berbasis IoT harus mampu memberikan performa yang tinggi dan respons yang cepat terhadap permintaan akses dari pengguna.

2.2.6 Standarisasi Produk

Ada beberapa standar *engineering* yang dapat dirujuk dalam rancangan bangun sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT). Beberapa standar yang paling penting adalah:

- 1) ISO/IEC 27001:2013: Standar ini mengatur tentang manajemen keamanan informasi, yang membantu dalam menentukan kontrol keamanan yang harus diterapkan dalam sistem IoT.
- 2) ISO/IEC 27002:2013: Standar ini menyediakan panduan untuk implementasi kontrol keamanan yang ditentukan dalam ISO/IEC 27001.

- 3) ISO/IEC 15408 (*Common Criteria*): Standar ini mengatur tentang evaluasi keamanan produk, yang dapat digunakan untuk mengevaluasi produk IoT dari segi keamanan.
- 4) IEEE 802.15.4: Standar ini mengatur tentang komunikasi *wireless* yang digunakan dalam sistem IoT yang digunakan dalam sistem kontrol akses pintu.
- 5) IEEE P2413: Standar ini menyediakan panduan untuk arsitektur IoT, yang dapat digunakan dalam mendesain sistem keamanan kunci pintu gedung berbasis IoT.
- 6) NIST SP 800-160: Standar ini menyediakan panduan untuk desain keamanan sistem IoT, yang dapat digunakan dalam menentukan kontrol keamanan yang harus diterapkan dalam sistem keamanan kunci pintu gedung berbasis IoT.
- 7) UL 294: Standar ini mengatur tentang keamanan sistem akses kontrol, yang dapat digunakan dalam menentukan kontrol keamanan yang harus diterapkan dalam sistem keamanan kunci pintu gedung berbasis IoT.
- 8) UL 681: Standar ini mengatur tentang keamanan sistem akses kontrol yang digunakan dalam lingkungan komersial, yang dapat digunakan dalam menentukan kontrol keamanan yang harus diterapkan dalam sistem keamanan kunci pintu gedung berbasis IoT.

Ketentuan-ketentuan dari standar-standar tersebut harus diperhatikan dalam proses pembuatan sistem keamanan kunci pintu gedung berbasis IoT, untuk memastikan sistem tersebut aman dan sesuai dengan standar industri yang ditentukan.

2.2.7 Etika Profesi yang Dijunjung

Dalam melakukan perancangan dan pengerjaan tugas akhir, prinsip – prinsip kode etik insinyur yang diterapkan tercantum dalam *fundamental canons National Society of Professional Engineer* (NSPE), meliputi:

- 1) Mengutamakan keselamatan, kesehatan, dan kesejahteraan publik.

Penerapan etika profesi tersebut dalam pembuatan proyek tugas akhir ini yaitu dalam pengerjaan proyek ini jam kerjanya disesuaikan berdasarkan kemampuan dan kapasitas masing-masing anggota tim tugas akhir sesuai yang ada di bagian *man-month*. Hal tersebut dilakukan demi mengutamakan keselamatan, kesehatan, dan kesejahteraan masing-masing anggota tim tugas akhir.

- 2) Melakukan pelayanan hanya dalam bidang kompetensi masing-masing.

Penerapan etika profesi tersebut dalam pembuatan proyek tugas akhir ini yaitu dalam pembagian tugas (*partitioning*) pengeraan proyek ini disesuaikan dengan bidang kompetensi masing-masing atau konsentrasi yang diambil oleh anggota tim tugas akhir. Untuk pembagian tugasnya dapat dilihat pada Gambar 2.1 dan Tabel 4.1. Hal tersebut dilakukan demi menjunjung etika profesi melakukan pelayanan hanya dalam bidang kompetensi masing-masing.

3) Mengeluarkan pernyataan secara objektif dan jujur.

Penerapan etika profesi tersebut dalam pembuatan proyek tugas akhir ini yaitu dalam penulisan laporan tugas akhir salah satunya dokumen B-100 ini dilakukan secara objektif dan jujur dengan didukung adanya referensi dari berbagai sumber, seperti paper jurnal baik itu dari Undip atau tidak, internet, dan sumber lainnya. Hal tersebut dilakukan demi menjunjung etika profesi mengeluarkan pernyataan secara objektif dan jujur.

4) Bertindak untuk setiap *employer* atau klien sebagai orang kepercayaan.

Penerapan etika profesi tersebut dalam pembuatan proyek tugas akhir ini yaitu mengerjakan tugas akhir sesuai dengan pedoman tugas akhir.

5) Menghindari perbuatan menipu/berbohong.

Penerapan etika profesi tersebut dalam pembuatan proyek tugas akhir ini yaitu tidak melebih-lebihkan tanggung jawab dalam atau untuk masalah pokok penugasan sebelumnya.

6) Berperilaku terhormat, bertanggung jawab, etis, dan sah untuk meningkatkan kehormatan, nama baik, dan kegunaan profesi.

Penerapan etika profesi tersebut dalam pembuatan proyek tugas akhir ini yaitu antara anggota tim yang satu dengan lainnya selalu mengingatkan untuk meningkatkan kehormatan, nama baik, dan kegunaan profesi tim dengan menjalankan atau mengerjakan tugas akhir sesuai dengan jadwal dan waktu pengembangan yang tertera pada Tabel 2.7 disertai etika yang ada.

2.3 Skenario Pemanfaatan Produk

Desain rekayasa perancangan sistem keamanan kunci pintu gedung berbasis *Internet of Things* ini secara umum ditujukan untuk membangun sistem keamanan pada sebuah gedung, seperti hotel, perusahaan, apartemen, maupun instansi untuk meningkatkan keamanan dalam ruangan.

Adapun pihak yang menjadi pengguna sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) adalah:

- 1) Pemilik gedung: Pemilik gedung merupakan pengguna utama sistem keamanan ini, karena ia bertanggung jawab atas keamanan gedung yang dimilikinya.
- 2) *Tenant*/penyewa: *Tenant* atau penyewa gedung juga merupakan pengguna sistem keamanan, karena mereka membutuhkan akses masuk dan keluar dari gedung yang aman.
- 3) Karyawan: Karyawan yang bekerja di gedung tersebut juga merupakan pengguna sistem keamanan, karena mereka membutuhkan akses masuk dan keluar dari gedung yang aman.
- 4) Tamu: Tamu atau pengunjung yang datang ke gedung tersebut juga merupakan pengguna sistem keamanan, karena mereka membutuhkan akses masuk ke gedung yang aman.

Oleh karena itu, sistem keamanan kunci pintu gedung berbasis IoT akan berguna bagi berbagai pihak yang membutuhkan akses masuk dan keluar dari gedung yang aman.

Berikut adalah beberapa karakteristik (para) pengguna sistem keamanan kunci pintu gedung berbasis *Internet of Things*:

- 1) Menggunakan *smartphone*: Pengguna sistem keamanan kunci pintu gedung berbasis IoT biasanya memiliki *smartphone* yang dapat terhubung ke internet, karena aplikasi atau *platform* yang digunakan untuk mengontrol akses masuk dan keluar dari gedung tersebut biasanya tersedia dalam bentuk aplikasi *mobile*.
- 2) Menggunakan jaringan nirkabel: Pengguna sistem keamanan kunci pintu gedung berbasis IoT biasanya terhubung ke jaringan nirkabel, seperti WiFi, karena setiap kunci pintu yang terhubung ke jaringan tersebut harus terkoneksi ke internet.
- 3) Memiliki akses ke aplikasi atau *platform*: Pengguna sistem keamanan kunci pintu gedung berbasis IoT harus memiliki akses ke aplikasi atau *platform* yang digunakan untuk mengontrol akses masuk dan keluar dari gedung, serta melihat riwayat akses dan melakukan pemantauan keamanan.
- 4) Memiliki mekanisme autentikasi: Pengguna sistem keamanan kunci pintu gedung berbasis IoT harus memiliki mekanisme autentikasi yang aman, seperti *password* untuk memastikan hanya pengguna yang sah yang dapat mengakses sistem keamanan.

- 5) Memiliki kebutuhan akses masuk dan keluar yang aman: Pengguna sistem keamanan kunci pintu gedung berbasis IoT memiliki kebutuhan akses masuk dan keluar yang aman, karena sistem ini bertujuan untuk meningkatkan keamanan gedung.

Berikut adalah beberapa informasi mengenai waktu, tempat, dan cara penggunaan produk sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) oleh pihak-pihak:

- 1) Waktu penggunaan: Penggunaan produk sistem keamanan kunci pintu gedung berbasis IoT dapat dilakukan kapan saja, tergantung pada kebutuhan pengguna. Misalnya, pemilik gedung dapat mengontrol akses masuk dan keluar dari gedung pada jam kerja saja, atau *tenant* dapat mengakses gedung hanya pada waktu-waktu tertentu saja.
- 2) Tempat penggunaan: Produk sistem keamanan kunci pintu gedung berbasis IoT dapat digunakan di gedung yang telah dipasangi kunci pintu yang terhubung ke jaringan.
- 3) Cara penggunaan: Penggunaan produk sistem keamanan kunci pintu gedung berbasis IoT dilakukan melalui aplikasi atau *platform* yang disediakan. Pengguna dapat mengontrol akses masuk dan keluar dari gedung, melihat riwayat akses, dan melakukan pemantauan keamanan melalui aplikasi atau *platform* tersebut.

Oleh karena itu, produk sistem keamanan kunci pintu gedung berbasis IoT dapat digunakan oleh berbagai pihak sesuai dengan kebutuhan dan di tempat yang telah dipasangi kunci pintu yang terhubung ke jaringan.

Jika memungkinkan diproduksi massal, berikut adalah beberapa skenario pemasaran produk sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT):

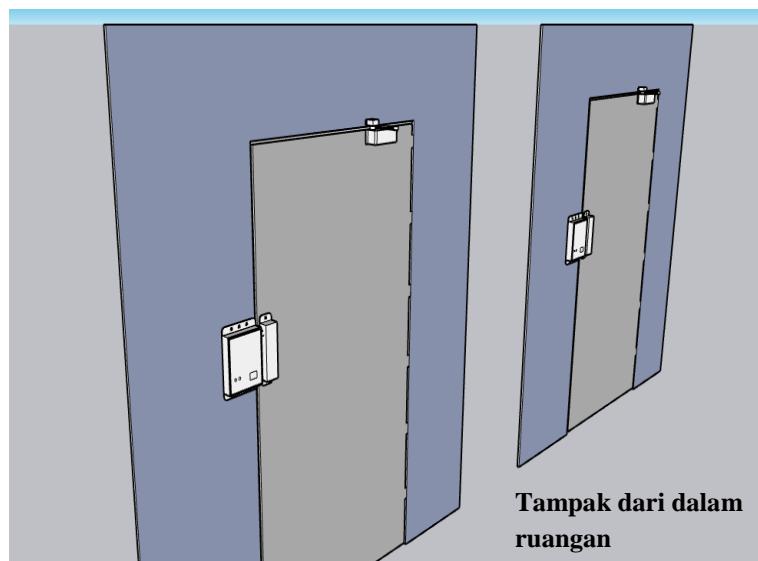
- 1) Menggunakan media *online*: Caranya dengan mempromosikan produk sistem keamanan kunci pintu gedung berbasis IoT melalui media *online*, seperti situs *web*, media sosial, dan forum *online*. Selain itu, juga dapat dipromosikan dengan menggunakan iklan *banner* atau postingan di media sosial yang menjelaskan fitur-fitur dan keunggulan produk yang dipasarkan.
- 2) Menghadiri pameran atau seminar: Pemasaran produk dapat dilakukan dengan menghadiri pameran atau seminar yang terkait dengan teknologi IoT atau keamanan gedung, dan menyajikan produk kepada para pengunjung. Selain itu,

juga dapat dengan cara membagikan brosur atau katalog produk kepada para pengunjung.

- 3) Menggunakan media cetak: Pemasaran produk dapat dilakukan dengan mempromosikan produk sistem keamanan kunci pintu gedung berbasis IoT melalui media cetak, seperti surat kabar, majalah, atau brosur. Selain itu, dapat juga dengan membuat iklan atau artikel yang menjelaskan fitur-fitur dan keunggulan produk.
- 4) Mengadakan demo produk: Pemasaran produk dapat dilakukan dengan mengadakan demo produk sistem keamanan kunci pintu gedung berbasis IoT kepada para calon pembeli atau *investor*. Demo ini dapat dilakukan di gedung yang telah dipasangi produk, atau di tempat lain yang sesuai.
- 5) Menggunakan jaringan koneksi: Pemasaran produk dapat dilakukan dengan memanfaatkan jaringan koneksi yang dimiliki untuk mempromosikan produk sistem keamanan kunci pintu gedung berbasis IoT kepada teman, kolega, atau rekan bisnis yang mungkin berminat.

Dengan demikian, ada beberapa cara yang dapat dilakukan untuk memasarkan produk sistem keamanan kunci pintu gedung berbasis IoT kepada para calon pembeli atau investor.

Gambar 2.19 dan 2.20 diperlihatkan pemanfaatan produk sistem keamanan kunci pintu gedung berbasis IoT.



Gambar 2.19 Pemanfaatan Produk Sistem Keamanan Kunci Pintu Gedung Berbasis IoT Tampak dari Dalam Ruangan



Gambar 2.20 Pemanfaatan Produk Sistem Keamanan Kunci Pintu Gedung Berbasis IoT Tampak dari Luar Ruangan

2.4 Nilai Strategis

Berikut adalah beberapa dampak yang diharapkan dari pengembangan produk sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) terhadap masyarakat secara umum:

- 1) Peningkatan keamanan: Produk sistem keamanan kunci pintu gedung berbasis IoT diharapkan dapat meningkatkan keamanan gedung, sehingga masyarakat yang tinggal atau bekerja di gedung tersebut merasa lebih aman.
- 2) Peningkatan efisiensi: Produk ini diharapkan dapat meningkatkan efisiensi, karena pengguna dapat mengontrol akses masuk dan keluar dari gedung secara mudah melalui aplikasi atau *platform* yang disediakan.
- 3) Peningkatan kenyamanan: Produk ini diharapkan dapat meningkatkan kenyamanan pengguna, karena mereka dapat mengakses gedung dengan lebih mudah dan cepat, serta tidak perlu repot-repot membawa kunci atau kartu akses.
- 4) Peningkatan kinerja: Produk ini diharapkan dapat meningkatkan kinerja pengguna, karena mereka dapat mengontrol akses masuk dan keluar dari gedung secara cepat dan tepat waktu.
- 5) Peningkatan inovasi: Pengembangan produk sistem keamanan kunci pintu gedung berbasis IoT diharapkan dapat mendorong inovasi di bidang keamanan gedung dan teknologi IoT.

Dengan demikian, pengembangan produk sistem keamanan kunci pintu gedung berbasis IoT diharapkan dapat memberikan dampak positif bagi masyarakat secara umum.

2.5 Usaha Pengembangan

Produk akhir yang hendak dikembangkan adalah perangkat *monitoring* dan *controlling* pintu gedung sebagai suatu sistem yang utuh, dengan subsistem berupa modul sensor dan perangkat aktuator yang terpisah. Subsistem tersebut diarahkan untuk menjadi produk-produk tunggal yang mandiri, memiliki kompatibilitas untuk dirakit menjadi sistem terpadu dan dapat dipasarkan secara terpisah sesuai kebutuhan pasarnya masing-masing.

Usaha dalam proses pengembangan didefinisikan sebagai berikut:

2.5.1 *Man-Month*

Proyek tugas akhir ini dikerjakan oleh satu tim tugas akhir Teknik Elektro Universitas Diponegoro yang terdiri dari 3 orang mahasiswa S1 yaitu 2 orang mahasiswa konsentrasi Teknologi Informasi dan 1 orang mahasiswa konsentrasi Telekomunikasi. Selain itu, proyek ini dibimbing oleh 2 orang dosen. Pengerjaan proyek ini berlangsung selama 5 bulan dengan jam kerja masing – masing mahasiswa adalah 20 jam per minggu.

2.5.2 *Machine-Month*

Dalam perancangan sistem ini, memerlukan *software* ataupun *hardware* sebagai berikut:

- a. Komputer/laptop sebanyak 3 buah untuk melakukan berbagai aktivitas, seperti pembuatan prototipe, *programming*, simulasi, administrasi, dan pengujian sistem. Komputer/laptop diperkirakan digunakan selama 300 jam.
- b. Perlengkapan pengembang.

2.5.3 *Development Tools*

Proses pengembangan produk ini menggunakan beberapa perangkat lunak sebagai berikut:

- *Android Studio* untuk pengembangan aplikasi *mobile*.
- *Arduino IDE* merupakan *software* untuk melakukan penulisan program, *compile* serta *upload* program ke ESP32.

- *Visual Code Studio* digunakan sebagai *software code editor*.
 - *Firebase Console* digunakan untuk mempermudah dalam pengaturan FCM.
- Selain itu, ada beberapa perangkat keras yang diperlukan dalam pengembangan produk ini, diantaranya:
- Laptop
 - Perlengkapan pengembang (*toolkit* perangkat keras)

2.5.4 Test Equipment

Untuk keseluruhan proses pengembangan, diperlukan peralatan-peralatan pengujian sebagai berikut:

- Perangkat Android dengan minimal Android 5.0 APIs digunakan sebagai piranti untuk melakukan *monitoring* dan *controlling* keamanan kunci pintu gedung.
- Multimeter digital
- *Postman* digunakan untuk pengujian API.
- *Wireshark*
- *QR-Code reader*
- *Stopwatch*

2.5.5 Kebutuhan Expert

Tim pengembang perancangan sistem ini terdiri dari Dosen, 2 Mahasiswa Konsentrasi Teknologi Informasi, dan 1 Mahasiswa Konsentrasi Telekomunikasi.

Untuk mengembangkan dan mengoperasikan sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT), mungkin diperlukan *expert* dengan keahlian dalam beberapa bidang, di antaranya:

- Keamanan *cyber*: *Expert* ini bertanggung jawab untuk mengelola kerentanan sistem terhadap serangan *cyber* dan memastikan bahwa sistem tidak mudah disusupi oleh pihak yang tidak berwenang.
- Jaringan: *Expert* ini bertanggung jawab untuk mengelola jaringan yang terhubung dengan sistem keamanan dan memastikan bahwa sistem dapat beroperasi dengan baik.
- Pemrograman: *Expert* ini bertanggung jawab untuk menulis kode yang mengendalikan perangkat keras dan mengakses data dari sensor.

- *Hardware*: *Expert* ini bertanggung jawab untuk mengelola perangkat keras yang terhubung dengan sistem keamanan, seperti sensor dan kontroler IoT.
- *Cloud*: *Expert* ini bertanggung jawab untuk mengelola *platform cloud* yang digunakan untuk menyimpan dan mengelola data dari sistem keamanan.

2.5.6 Perkiraan Biaya

Tim proyek tugas akhir menghitung perkiraan biaya yang diperlukan untuk mengembangkan produk ini berdasarkan konsep produk yang diusulkan dan bahan serta peralatan yang akan dibeli atau disewa, serta biaya tenaga kerja eksternal. Tim juga memastikan biaya tersebut terpenuhi, baik dari tim sendiri atau pihak luar. Perkiraan biaya yang dibutuhkan dalam rancang bangun sistem ini dapat dilihat pada Tabel 2.6 di bawah.

Tabel 2.6 Perkiraan Biaya

No.	Pengeluaran	Jumlah	Harga Satuan	Estimasi Harga
			(Rp)	(Rp)
1	Mikrokontroler ESP32	2 buah	70.000	140.000
2	Sensor magnetic	2 buah	10.000	20.000
3	Kabel	5 meter	2.500	12.500
4	Cetak PCB	4 buah	25.000	100.000
5	Baut, mur, <i>spacer</i>	2 buah	15.000	30.000
6	Adaptor 12 Volt	2 buah	20.000	40.000
7	Tombol	2 buah	3000	6.000
8	<i>Hosting</i>	5 buah	50.000	250.000
9	Solenoida	2 buah	40.000	80.000
Jumlah Total				678.500

2.5.7 Peluang Keberhasilan

Dengan mempertimbangkan semua aspek teknik dan non-teknis, termasuk misalnya kerumitan integrasi antar perangkat dan kerumitan dalam *programming*, serta tersedianya referensi dari berbagai sumber yang sesuai dengan sistem terkait, maka dapat diperkirakan proyek tugas akhir ini dapat selesai selama 5 bulan sesuai yang telah direncanakan. Oleh karena itu, tim membuat estimasi peluang keberhasilan menyelesaikan proyek pengembangan ini.

2.5.8 Jadwal dan Waktu Pengembangan

Proyek perancangan sistem keamanan kunci pintu gedung berbasis IoT dirancang untuk rentang 5 bulan, dimulai pada Januari 2023 – Mei 2023. *Time table* proyek ini dapat dilihat pada Tabel 2.7.

Tabel 2.7 Jadwal dan Waktu Pengembangan

Fase	<i>Deliverables</i>	Jadwal	Kebutuhan Sumber Daya
Konsep Produk	Dokumen B100	Januari	Literatur
	Proposal	2023	
Analisis	Dokumen B200	Februari	1. Spek standar
	Spesifikasi Fungsional	2023	2. <i>Engineer</i>
Desain	Dokumen B300	Maret	1. <i>Dvlp. Tools</i>
	Skematik	2023	2. Penguasaan teknologi pendukung
	Rangkaian		3. Literatur
	Rancangan		4. <i>Engineer</i>
(Pedoman standar perencanaan instalasi)	Dokumen B400	April	1. <i>Dvlp. Tools</i>
	Lab Redesain	2023	2. <i>Software Cadsoft Eagle</i>
			3. <i>Software Arduino IDE</i>
			4. <i>Software Android Studio</i>
			5. <i>Engineer</i>
			6. Komponen-komponen hardware
(Berdasarkan standar instalasi)	Dokumen B-500	Mei 2023	1. <i>Dvlp. Tools</i>
	Error Report, redesain		2. <i>Software Cadsoft Eagle</i>
	skematik, ralat		3. <i>Software Arduino IDE</i>
	kode program		4. <i>Engineer</i>

3 KESIMPULAN

Kesimpulan yang dapat diambil dari dokumen ini sebagai berikut:

1. Sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) adalah sistem yang menggunakan teknologi IoT untuk mengelola dan mengontrol akses ke pintu gedung. Sistem ini terdiri dari perangkat keras seperti sensor, kontroler IoT, dan modul komunikasi seluler yang terhubung dengan jaringan internet. Sistem ini juga terdiri dari perangkat lunak yang mengendalikan perangkat keras dan mengakses data dari sensor. Data yang dihasilkan oleh sistem keamanan ini kemudian disimpan dan dikelola menggunakan *platform cloud*. Sistem keamanan ini dapat membantu meningkatkan keamanan gedung dengan cara mengontrol akses ke pintu hanya untuk orang yang berwenang saja.
2. Proyek Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things* yang hendak dibuat dan dikembangkan memiliki kelayakan untuk dijalankan, dari sisi kajian ekonomis, strategis, penguasaan teknologi, maupun kemampuan fabrikasi yang ada. Hal tersebut dikarenakan sistem ini sesuai dengan kebutuhan keamanan gedung. Biaya yang dibutuhkan untuk mengembangkan dan mengoperasikan sistem ini relatif murah. Pendapatan yang akan dihasilkan dari pemasaran sistem ini cukup untuk menutup biaya yang dikeluarkan. Dari sisi ekonomi sistem ini akan memberikan nilai tambah bagi pemilik gedung. Dari sisi strategis sistem ini akan memberikan keuntungan bagi pemilik gedung, seperti meningkatkan reputasi gedung atau meningkatkan loyalitas pelanggan. Pengelolaan dan pengoperasian sistem ini mudah dilakukan. Serta dari segi kemampuan fabrikasi, pemilik gedung memiliki kemampuan untuk memproduksi atau memperoleh perangkat keras yang dibutuhkan untuk pengoperasian sistem ini dengan baik.
3. Beberapa keunggulan sistem keamanan kunci pintu gedung berbasis *Internet of Things* (IoT) adalah keamanan yang tinggi, mudah dioperasikan, dapat terintegrasi dengan sistem keamanan yang ada, tingkat keandalan yang tinggi, dan dapat menghemat biaya.
4. Tujuan dari rancang bangun sistem keamanan kunci pintu gedung berbasis *Internet of Things* ini adalah untuk meningkatkan efisiensi dan keamanan dalam pengelolaan akses masuk ke gedung dengan menggunakan teknologi internet dan sensor.

4 BIODATA TIM PENGUSUL

Tim akan melampirkan biodata yang mencakup informasi demografis dan keahlian/kualifikasi. Kualifikasi meliputi mata kuliah paket yang diselesaikan, pelatihan terkait yang diikuti, kompetisi terkait yang dimenangkan, dan portofolio terkait lainnya (kemampuan menggunakan perangkat lunak/perangkat keras).

4.1 Daftar Nama dan Spesifikasi Pekerjaan

Tabel 4.1 Daftar Nama dan Spesifikasi Pekerjaan

No.	Nama	Keahlian	Pembagian Tugas
1.	Henric Dhiki Wicaksono 21060119120011 Teknologi Informasi	Pemrograman arduino, <i>database</i> , aplikasi <i>mobile</i> sebagai piranti akses masuk pintu gedung dan <i>website</i> untuk mendukung sistem <i>monitoring</i> dan <i>controlling</i> jarak jauh.	Perancangan perangkat
2.	Novi Dianasari 21060119120039 Telekomunikasi	Pemrograman mikrokontroler, matlab, <i>software</i> proteus.	Perancangan sistem komunikasi data dua arah dan perangkat penguncian yang mendukung sistem keamanan kunci pintu gedung berbasis IoT.
3.	Muhammad Khoiril Wafi 21060119140133 Teknologi Informasi	Pemrograman arduino, <i>database</i> , aplikasi android.	Perancangan sistem <i>database & server</i> serta sistem keamanan kunci pintu gedung dengan <i>Access Control</i> .

4.2 Biodata Tim Tugas Akhir

A. Biodata Pengusul 1

Nama Lengkap : Henric Dhiki Wicaksono
Jenis Kelamin : Laki - laki
NIM : 21060119120011
Tempat, Tanggal Lahir : Boyolali, 12 Desember 2000
Alamat : Jl. Baskoro Raya No. 61
Tembalang Semarang
Email : henricwicaksono@gmail.com
Nomor Telepon / HP : 085866844261
Peminatan Konsentrasi : Teknologi Informasi



Kompetensi / Keahlian yang Dimiliki

No	Mata Kuliah Pilihan yang Diambil	SKS
1	Sistem Operasi	2
2	Pengembangan Aplikasi Perangkat Bergerak	3
3	Interaksi Manusia dan Komputer	2
4	<i>Interface dan Peripheral</i>	2
5	Kriptografi	3
6	Struktur Data	3

Pelatihan yang Pernah Diikuti

No	Jenis Pelatihan	Lembaga	SKS
1	HCIA Datacom	Huawei Indonesia	2022
2	HCIA AI	Huawei Indonesia	2022

B. Biodata Pengusul 2

Nama Lengkap : Novi Dianasari
Jenis Kelamin : Perempuan
NIM : 21060119120039
Tempat, Tanggal Lahir : Kab. Semarang, 7 Februari 2002
Alamat : Jl. Baskoro Raya No. 61
Tembalang Semarang
Email : novidianasari722@gmail.com
Nomor Telepon / HP : 085290662689
Peminatan Konsentrasi : Telekomunikasi



Kompetensi / Keahlian yang Dimiliki

No	Mata Kuliah Pilihan yang Diambil	SKS
1	Kecerdasan Buatan	3
2	Pengolahan dan Analisis Sinyal	3
3	Perencanaan Jaringan Telekomunikasi	3
4	Manajemen Jaringan Telekomunikasi	3
5	Perbaikan Kinerja Jaringan	3

Pelatihan yang Pernah Diikuti

No	Jenis Pelatihan	Lembaga	SKS
1	<i>Internet of Things</i>	IoT Telkom	2022

C. Biodata Pengusul 3

Nama Lengkap : Muhammad Khoiril Wafi
Jenis Kelamin : Laki - laki
NIM : 21060119140133
Tempat, Tanggal Lahir : Demak, 4 Maret 2001
Alamat : Jl. Baskoro Raya No.61
Tembalang Semarang
Email : khoirilwafi123@gmail.com
Nomor Telepon / HP : 083116291606
Peminatan Konsentrasi : Teknologi Informasi



Kompetensi / Keahlian yang Dimiliki

No	Mata Kuliah Pilihan yang Diambil	SKS
1	Sistem Operasi	2
2	Pengembangan Aplikasi Perangkat Bergerak	3
3	Interaksi Manusia dan Komputer	2
4	<i>Interface dan Peripheral</i>	2
5	Kriptografi	3
6	Pemrograman Berorientasi Objek	3

Pelatihan yang Pernah Diikuti

No	Jenis Pelatihan	Lembaga	SKS
1	HCIA Datacom	Huawei Indonesia	2022
2	HCIA AI	Huawei Indonesia	2022



Dokumen Pengembangan Produk Lembar Sampul Dokumen

Judul Dokumen

TUGAS AKHIR:
Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*

Jenis Dokumen

SPESIFIKASI

Catatan: Dokumen ini dikendalikan penyebarannya oleh Dept. Teknik Elektro Undip

Nomor Dokumen

B200-01-TA2223.2.19012

Nomor Revisi

01

Nama File

B200-2-TA2223

Tanggal Penerbitan

8 September 2023

Unit Penerbit

Departemen Teknik Elektro Undip

Jumlah Halaman

15 (termasuk lembar sampul ini)

Data Pengusul				
Pengusul	Nama NIM	Henric Dhiki Wicaksono 21060119120011	Jabatan	Anggota
	Nama NIM	Novi Dianasari 21060119120039	Jabatan	Anggota
	Nama NIM	Muhammad Khoiril Wafi 21060119140133	Jabatan	Anggota
Pembimbing Utama	Nama NIP	M. Arfan, S.Kom., M.Eng. 198408172015041002	Tanda Tangan	
Pendamping	Nama NIP	Imam Santoso, S.T., M.T. 197012031997021001	Tanda Tangan	

DAFTAR ISI

1.	PENDAHULUAN	4
1.1	Ringkasan Isi Dokumen	4
1.2	Aplikasi Dokumen.....	4
1.3	Referensi.....	4
1.4	Daftar Singkatan.....	5
2.	GAMBARAN SISTEM.....	5
2.1	Gambaran Sistem Saat Ini	5
2.2	Gambaran Sistem yang Akan Dikembangkan.....	6
2.3	Fungsi	7
2.4	Kebutuhan	9
3.	SPESIFIKASI.....	11
3.1	Target Sistem yang Akan Dikembangkan.....	11
3.2	Aktor.....	11
3.3	Standarisasi.....	12
3.4	Batasan	13
4.	PENGEMBANGAN	13
4.1	Jadwal Pengembangan	13
4.2	Biaya Pengembangan	14
5.	PENUTUP	15

Catatan Sejarah Perbaikan Dokumen

VERSI, TGL, OLEH	PERBAIKAN
01, 8 September 2023, oleh Henric Dhiki Wicaksono, Novi Dianasari, dan Muhammad Khoiril Wafi.	<i>Draft Dokumen B200</i>

1. PENDAHULUAN

1.1 Ringkasan Isi Dokumen

Dokumen ini berisi gambaran serta uraian spesifikasi produk “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*”. Dalam dokumen ini akan dibahas mengenai gambaran dari sistem yang akan dikembangkan, spesifikasi, prosedur pelaksanaan serta waktu pelaksanaan dari proses tersebut. Uraian spesifikasi dari sistem akan memberikan gambaran mengenai kebutuhan yang diperlukan untuk mengimplementasikan produk sesuai dengan spesifikasi yang telah ditentukan. Dalam dokumen ini juga dibahas mengenai fungsionalitas dari produk yang akan dikembangkan serta batasan-batasan dari produk tersebut. Pada akhirnya dokumen ini akan menjadi acuan dalam proses pengembangan produk yang akan dibuat.

1.2 Aplikasi Dokumen

Dokumen ini digunakan dalam proses pengembangan produk “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*” untuk:

- 1) Sebagai gambaran proses pengembangan produk yang akan dilaksanakan serta batasan-batasan dari produk.
- 2) Sebagai gambaran target yang akan dicapai berdasarkan spesifikasi yang telah ditentukan.
- 3) Sebagai dokumentasi dan pencatatan perubahan.

Dokumen B200 ini diajukan kepada dosen pembimbing tugas akhir dan tim tugas akhir Program Studi Sarjana Teknik Elektro Undip sebagai bahan penilaian tugas akhir.

1.3 Referensi

- [1] I. Hermawan, D. Arnaldy, P. Oktivasari, and D. A. Fachrudin, “Development of Intelligent Door Lock System for Room Management Using Multi Factor Authentication,” vol. 16, no. 1, pp. 1–14, 2023.
- [2] K. Y. Sun, Y. Pernando, and M. I. Safari, “Perancangan Sistem IoT pada Smart Door Lock Menggunakan Aplikasi BLYNK,” *JUTSI (Jurnal Teknol. dan Sist. Informasi)*, vol. 1, no. 3, pp. 289–296, 2021, doi: 10.33330/jutsi.v1i3.1360.

1.4 Daftar Singkatan

Tabel 1.1 Daftar singkatan.

SINGKATAN	ARTI
IoT	<i>Internet of Things</i>
WiFi	<i>Wireless Fidelity</i>
JSON	<i>Javascript Object Notation</i>
ACL	<i>Access Control List</i>
IOS	<i>iPhone Operating System</i>
PCB	<i>Printed Circuit Board</i>
SDK	<i>Software Development Kit</i>
ESP	<i>Espressif</i>

2. GAMBARAN SISTEM

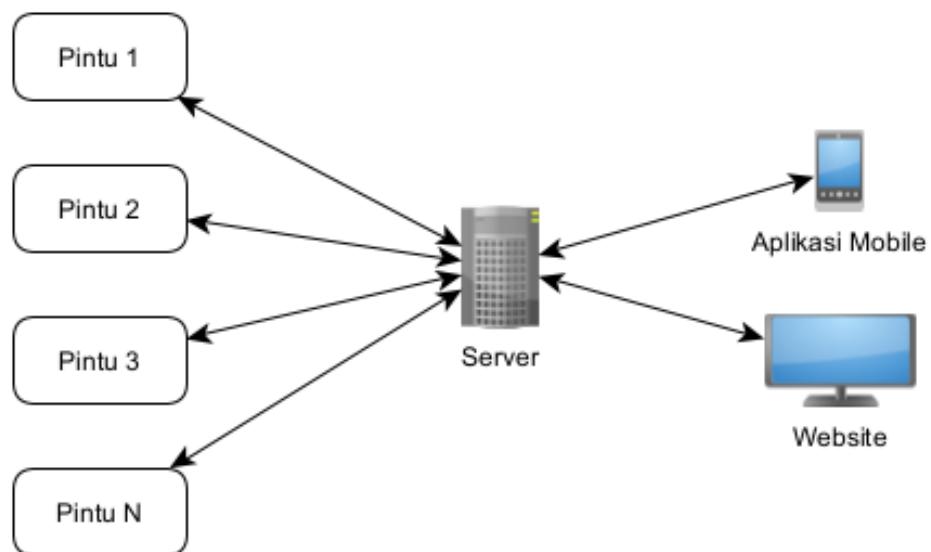
2.1 Gambaran Sistem Saat Ini

Pada umumnya sebuah gedung memiliki banyak ruangan yang menyimpan berbagai barang, contohnya pada sebuah gedung perkuliahan terdapat ruang perkuliahan, laboratorium, ruang dosen, dan lain sebagainya. Semua ruangan tersebut tentunya membutuhkan sebuah kunci untuk mengamankan barang-barang yang ada di dalamnya supaya terhindar dari pencurian, perusakan, dan hal lainnya yang mengakibatkan kerugian. Saat ini, hampir semua gedung masih menggunakan metode penguncian secara tradisional yaitu menggunakan kunci fisik, mungkin jika hanya satu pintu saja metode penguncian tersebut mudah dilaksanakan. Namun, pastinya di sebuah gedung memiliki jumlah ruangan yang banyak sehingga jumlah kunci yang dimiliki juga semakin banyak. Dengan adanya jumlah kunci yang banyak maka pengelolaan kunci akan menjadi kurang optimal seperti harus membuka pintu satu persatu dengan menggunakan kunci yang sekilas terlihat identik sehingga memerlukan waktu untuk menemukan kunci yang tepat[1]. Ada pula kondisi dimana terdapat pinjam-meminjam kunci dengan memberikan kunci fisik secara langsung yang akan memberikan kesempatan kepada orang lain untuk melakukan duplikasi dan kehilangan kunci. Dengan perkembangan teknologi, sistem kunci konvensional sering kali sulit untuk diperbarui atau dikontrol dari jarak jauh. Hal

tersebut menyulitkan pengaturan akses berdasarkan jadwal atau kebutuhan khusus. Oleh sebab itu, diperlukan sebuah mekanisme penguncian yang dapat meningkatkan keamanan serta memudahkan pengelolaan semua kunci di dalam gedung.

2.2 Gambaran Sistem yang Akan Dikembangkan

Dengan adanya kebutuhan mengenai mekanisme penguncian yang dapat meningkatkan keamanan serta memudahkan pengelolaan semua kunci di dalam gedung maka dilaksanakan “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*”. Sistem ini berfokus pada pengelolaan semua kunci pintu pada sebuah gedung dengan menggunakan konsep IoT. Konsep IoT memberikan mekanisme yang menghubungkan sebuah perangkat elektronik ke sebuah jaringan komunikasi sehingga dapat membentuk sebuah jaringan dapat bekerja secara otomatis[2]. Dengan menggunakan konsep IoT, sistem ini diharapkan dapat meningkatkan keamanan dan efisiensi dalam pengelolaan kunci pintu gedung dengan cara membuat sebuah perangkat penguncian yang terpasang di setiap pintu. Semua perangkat penguncian tersebut nantinya akan terhubung ke sebuah *server* untuk mengelola semua kunci yang ada. Adapun pengguna dapat menggunakan antarmuka seperti aplikasi *mobile* dan *website* yang telah disediakan untuk berinteraksi dengan sistem seperti membuka kunci pintu.



Gambar 2.1 Konfigurasi sistem yang akan dibuat.

Gambar di atas menjelaskan konfigurasi sistem yang akan dibuat. Sistem keamanan kunci pintu gedung ini mempunyai 3 bagian utama seperti yang terlihat pada Gambar 2.1 yaitu terdapat perangkat penguncian untuk masing-masing pintu, sebuah *server*, serta aplikasi *mobile* dan *website* sebagai antarmuka. Bagian-bagian tersebut akan bekerja secara bersama-sama untuk meningkatkan keamanan dan efisiensi sistem penguncian.

2.3 Fungsi

Seperti yang sudah dijelaskan pada bagian sebelumnya, sistem yang akan dikembangkan merupakan sebuah sistem yang berfungsi untuk melakukan pengelolaan kunci pintu gedung. Tujuan utamanya adalah untuk meningkatkan keamanan serta efisiensi penggunaan kunci. Fungsi tersebut akan diimplementasikan ke dalam fitur-fitur yang dimiliki oleh sistem seperti pemantauan, pengelolaan hak akses, pencatatan riwayat, penjadwalan, dan lain sebagainya. Dengan adanya fitur-fitur tersebut maka sistem yang akan dikembangkan mampu meningkatkan keamanan dan efisiensi pengelolaan kunci pintu pada gedung. Adapun fungsi-fungsinya secara lebih rinci akan dijelaskan pada Tabel 2.1 di bawah ini.

Tabel 2.1 Kemampuan sistem.

Fitur	Penjelasan
Pemantauan Secara <i>Realtime</i>	Sistem yang akan dibangun memungkinkan adanya pemantauan kondisi pintu secara <i>realtime</i> . Setiap pengelola gedung dapat melihat kondisi pintu setiap saat melalui aplikasi <i>mobile</i> atau <i>website</i> .
Pengaturan Hak Akses	Sistem yang akan dibangun memberikan wewenang akses ke pengelola. Pengelola dapat memberikan akses ke pengguna pada pintu tertentu. Akses yang diberikan dapat berupa akses sementara atau akses tidak terbatas dengan waktu akses harian yang telah ditentukan.

Tabel 2.1 (lanjutan)

Fitur	Penjelasan
Notifikasi Penerobosan	Sistem yang akan dikembangkan memiliki kemampuan untuk mengirimkan notifikasi kepada pengelola jika terjadi penerobosan pada sebuah pintu. Pengelola juga akan mendapatkan notifikasi jika sebuah pintu dalam kondisi yang tidak sesuai misalnya pintu terbuka sehingga tidak bisa dikunci.
Pencatatan Riwayat Aktivitas	Sistem yang akan dikembangkan dapat mencatat riwayat aktivitas dari semua pengguna yang berinteraksi dengan sistem. Dengan adanya catatan ini maka semua tindakan pengguna dapat dipantau.
Penjadwalan	Sistem yang akan dikembangkan mampu untuk membuka pintu secara otomatis pada rentang waktu tertentu. Metode ini memudahkan pengelolaan pintu yang harus dibuka setiap hari dalam periode waktu tertentu misalnya ruang kelas atau sejenisnya. Tentunya pengaturan penjadwalan merupakan wewenang dari pengelola di setiap gedung tersebut.
Kendali Jarak Jauh	Sistem yang akan dikembangkan memiliki kemampuan untuk menerima perintah penguncian secara <i>remote</i> sehingga semua pintu dapat dikendalikan dari jarak jauh melalui koneksi internet.
<i>Multi Offices</i>	Sistem yang akan dikembangkan tidak terbatas hanya untuk satu gedung saja. Sistem ini dapat digunakan untuk banyak gedung tentunya di dalam satu gedung tersebut memiliki satu orang pengelola yang menjadi penanggung jawab.

Dengan berbagai kemampuan yang dimiliki oleh sistem tersebut maka keamanan pada suatu gedung dapat ditingkatkan serta pengelolaan kunci pintu dapat menjadi lebih mudah dimana kunci fisik akan digantikan dengan kunci secara digital.

2.4 Kebutuhan

Pada Gambar 2.1 di atas terlihat bagian-bagian dari sistem yang akan dikembangkan yaitu sebuah perangkat penguncian yang dapat terhubung ke *server*, sebuah *server*, serta aplikasi *mobile* dan *website*. Semua bagian tersebut dilandaskan pada kebutuhan untuk menjalankan semua aktivitas dari pengguna seperti autentikasi, membuka pintu, menambahkan data baru, dan lain sebagainya. Adapun penjelasan kebutuhan sistem secara rinci dapat dilihat pada Tabel 2.2 di bawah ini.

Tabel 2.2 Kebutuhan sistem.

Komponen	Penjelasan
Perangkat Keras Penguncian	Untuk menjalankan mekanisme penguncian tentunya memerlukan perangkat keras yang terpasang di setiap pintu. Perangkat penguncian nantinya akan terhubung ke <i>server</i> sehingga memerlukan modul komunikasi seperti WiFi dan bluetooth. Dengan adanya modul komunikasi tersebut, perangkat penguncian dapat dipantau dan dikendalikan dengan menggunakan metode pengendalian secara digital.
Mekanisme Pengecekan Akses	Sebuah mekanisme diperlukan untuk memeriksa akses setiap pengguna. Setiap pengguna dapat mengakses ruangan jika pengguna tersebut memiliki izin akses yang tersimpan di dalam sistem.

Tabel 2.2 (lanjutan)

Komponen	Penjelasan
<i>Server</i>	Sebuah <i>server</i> digunakan untuk mengatur semua proses dan aktivitas di dalam sistem penguncian. <i>Server</i> akan melakukan autentikasi akses serta melakukan pencatatan aktivitas pengguna. Dengan adanya fitur kendali jarak jauh maka <i>server</i> juga harus memiliki kemampuan untuk mengirimkan perintah ke perangkat penguncian secara langsung.
<i>Database</i>	Untuk dapat bekerja dengan baik tentunya sistem membutuhkan data-data yang terkait dengan penguncian seperti data pengguna, pintu, gedung, dan lain sebagainya. Oleh karena itu, diperlukan sebuah tempat untuk menyimpan data-data tersebut.
<i>Aplikasi Mobile</i>	Aplikasi <i>mobile</i> digunakan oleh pengguna untuk berinteraksi dengan sistem penguncian. Dengan karakteristik perangkat yang ringkas dan dapat dibawa kemana-mana maka penggunaan perangkat <i>mobile</i> dapat meningkatkan kenyamanan pengguna dalam berinteraksi dengan sistem.
<i>Website</i>	Sebuah <i>website</i> dibutuhkan sebagai <i>dashboard</i> pengelola yang digunakan untuk mengatur semua aktivitas di dalam sistem seperti menambahkan pengguna, manambahkan penjadwalan, membuka pintu, dan lain sebagainya.

Daftar kebutuhan tersebut nantinya akan diimplementasikan ke dalam beberapa bagian pengembangan sistem yang terpisah. Dimana pada akhirnya akan membentuk suatu sistem yang utuh.

3. SPESIFIKASI

3.1 Target Sistem yang Akan Dikembangkan

Target yang harus dicapai untuk mengindikasikan bahwa sistem yang akan dikembangkan telah sesuai dengan rancangan awal yaitu dengan tesedianya semua fitur yang telah dijelaskan sebelumnya. Pada Tabel 2.1 telah dijelaskan mengenai fitur-fitur yang dimiliki oleh sistem seperti pengaturan hak akses, pemantauan, dan lain sebagainya. Fitur-fitur tersebut menjadi parameter yang akan dijadikan sebagai poin-poin pengujian yang akan menentukan keberhasilan pengembangan dari sistem tersebut.

3.2 Aktor

Aktor merupakan orang yang berinteraksi dengan sistem penguncian. Berdasarkan uraian fungsi dan fitur yang telah dijelaskan maka terdapat tiga aktor yang berinteraksi dengan sistem ini yang akan ditunjukkan pada Tabel 3.1 di bawah ini yaitu:

Tabel 3.1 Daftar aktor.

Aktor	Wewenang	Penjelasan
Moderator	Mengelola data gedung dan operator	Moderator bertugas untuk mengelola data gedung dan operator. Seorang moderator dapat menambahkan gedung atau operator baru, serta menghapus gedung dan juga operator.
Operator	Mengelola sistem penguncian pada satu gedung	Operator bertugas untuk mengelola sistem penguncian pada satu gedung tertentu dengan fitur yang dimiliki.
Pengguna	Menggunakan hak akses yang dimiliki	Pengguna dapat menggunakan hak akses yang telah dimiliki untuk mengakses ruangan dengan menggunakan aplikasi <i>mobile</i> .

3.3 Standarisasi

Dalam pengembangan sistem keamanan kunci pintu gedung berbasis IoT mengikuti beberapa standar sebagai berikut. Standarisasi yang digunakan dapat dilihat pada Tabel 3.2 di bawah ini.

Tabel 3.2 Standarisasi yang digunakan.

Bagian	Standarisasi	Penjelasan
Protokol Komunikasi	WiFi dan <i>Bluetooth</i>	Protokol komunikasi pada perangkat penguncian menggunakan WiFi untuk berkomunikasi dengan <i>server</i> dan <i>bluetooth</i> untuk berkomunikasi dengan aplikasi <i>mobile</i> .
Format Data	<i>Javascript Object Notation</i> (JSON)	Format data yang dikirimkan oleh <i>server</i> ke perangkat penguncian dan aplikasi <i>mobile</i> mengikuti format dari JSON yaitu berupa <i>javascript object</i> dalam bentuk teks.
Metode Autentikasi	<i>Access Control List</i> (ACL)	Metode autentikasi menggunakan sebuah daftar yang berisi identitas pengguna yang diizinkan untuk mengakses sebuah sumber daya yang dilindungi.
Sistem Operasi	Android dan IOS	Pengembangan aplikasi <i>mobile</i> mengikuti <i>Software Development Kit</i> (SDK) yang disediakan untuk sistem operasi Android dan IOS

3.4 Batasan

Pengembangan sistem keamanan kunci pintu gedung berbasis IoT berfokus untuk menerapkan metode baru dalam proses pengelolaan kunci pintu pada suatu gedung dengan beberapa batasan, yaitu:

1. Pengelolaan data gedung dan operator hanya bisa dilakukan melalui *website* sistem penguncian.
2. Pengguna hanya dapat menggunakan aplikasi *mobile* untuk menggunakan akses yang dimiliki.
3. Modul komunikasi menggunakan WiFi dan *bluetooth* untuk menjalankan mekanisme autentikasi dan kontrol penguncian.

4. PENGEMBANGAN

4.1 Jadwal Pengembangan

Proyek rancang bangun sistem keamanan kunci pintu gedung berbasis IoT dirancang untuk rentang 6 bulan, dimulai pada Maret 2023 – Agustus 2023. *Time table* proyek ini dapat dilihat pada Tabel 4.1.

Tabel 4.1 Jadwal pengembangan.

Fase	<i>Deliverables</i>	Jadwal	Kebutuhan Sumber Daya
Konsep Produk	Dokumen B100	Maret 2023	Literatur
	Proposal		
Analisis	Dokumen B200	April 2023	1. Spek standar
	Spesifikasi Fungsional		2. <i>Engineer</i>
Desain	Dokumen B300	April 2023	1. <i>Dvlp. Tools</i>
	Skematik Rangkaian		2. Penguasaan teknologi pendukung
	Rancangan		3. Literatur
			4. <i>Engineer</i>

Tabel 4.1 (lanjutan)

Fase	Deliverables	Jadwal	Kebutuhan Sumber Daya
Implementasi (Pedoman standar perencanaan)	Dokumen B400 Lab Redesain	Mei - Juni 2023	1. <i>Dvlp. Tools</i> 2. <i>Software Cadsoft Eagle</i> 3. <i>Software Android Studio</i> 4. <i>Engineer</i> 5. Komponen-komponen <i>hardware</i>
Pengujian subsistem	Dokumen B500 <i>Error Report</i> , redesain skematik, ralat kode program	Juli 2023	1. <i>Dvlp. Tools</i> 2. <i>Software Cadsoft Eagle</i> 3. <i>Engineer</i>
Laporan Akhir	Dokumen Laporan	Agustus 2023	1. <i>Dvlp. Tools</i> 2. Mekanisme Pelaporan

4.2 Biaya Pengembangan

Dengan beberapa bagian yang digunakan untuk membangun sistem penguncian ini maka diperlukan sejumlah biaya sebagaimana yang terlihat pada Tabel 4.2.

Tabel 4.2 Perkiraan biaya pengembangan.

No.	Pengeluaran	Jumlah	Harga (Rp)	Total (Rp)
1	Mikrokontroler ESP32	2 buah	70.000	140.000
2	Sensor magnetic	2 buah	10.000	20.000
3	Kabel	5 meter	2.500	12.500
4	Cetak PCB	4 buah	25.000	100.000
5	Baut, mur, <i>spacer</i>	2 buah	15.000	30.000
6	Adaptor 12 Volt	2 buah	20.000	40.000

Tabel 4.2 (lanjutan)

No.	Pengeluaran	Jumlah	Harga (Rp)	Total (Rp)
7	Tombol	2 buah	3.000	6.000
8	<i>Server</i>	5 bulan	120.000	600.000
9	Solenoida	2 buah	40.000	80.000
10	Kusen Pintu	2 buah	300.000	600.000
Jumlah Total				1.628.500

5. PENUTUP

Dokumen B200 memaparkan definisi, fungsi, dan spesifikasi dari produk “Sistem Keamanan Kunci Pintu Gedung Berbasis IoT”. Hasil perancangan spesifikasi pada dokumen B200 akan dijadikan acuan untuk pembuatan desain produk sistem keamanan kunci pintu gedung berbasis IoT, acuan pelaksanaan proyek, dan acuan pengujian fitur dari produk tersebut.



Dokumen Pengembangan Produk Lembar Sampul Dokumen

Judul Dokumen

TUGAS AKHIR:
Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*

Jenis Dokumen

DESAIN

Catatan: Dokumen ini dikendalikan penyebarannya oleh Dept. Teknik Elektro Undip

Nomor Dokumen

B300-01-TA2223.2.19012

Nomor Revisi

01

Nama File

B300-2-TA2223

Tanggal Penerbitan

8 September 2023

Unit Penerbit

Departemen Teknik Elektro Undip

Jumlah Halaman

25 (termasuk lembar sampul ini)

Data Pengusul				
Pengusul	Nama NIM	Henric Dhiki Wicaksono 21060119120011	Jabatan	Anggota
	Nama NIM	Novi Dianasari 21060119120039	Tanda Tangan	
	Nama NIM	Muhammad Khoiril Wafi 21060119140133	Jabatan	Anggota
Pembimbing Utama	Nama NIP	M. Arfan, S.Kom., M.Eng. 198408172015041002	Tanda Tangan	
Pendamping	Nama NIP	Imam Santoso, S.T., M.T. 197012031997021001	Tanda Tangan	

DAFTAR ISI

1.	PENDAHULUAN.....	4
1.1	Ringkasan Isi Dokumen	4
1.2	Aplikasi Dokumen.....	4
1.3	Referensi.....	4
1.4	Daftar Singkatan.....	5
2.	SOLUSI DESAIN	6
2.1	Solusi Desain 1	6
2.2	Solusi Desain 2	6
2.3	Solusi Desain 3	7
3.	PERANCANGAN.....	7
3.1	Metode Verifikasi Akses	7
3.2	Perangkat Penguncian	9
3.3	Komunikasi Data Dua Arah	12
3.4	<i>Database</i>	14
3.5	<i>Backend API</i>	15
3.6	Aplikasi <i>Mobile</i>	17
3.7	Tampilan <i>Website</i>	20
4.	VERIFIKASI	23
4.1	Komunikasi WiFi dan <i>Bluetooth</i>	23
4.2	<i>Backend API Laravel</i>	24
4.3	Kode QR	25
5.	PENUTUP	25

Catatan Sejarah Perbaikan Dokumen

VERSI, TGL, OLEH	PERBAIKAN
01, 8 September 2023, oleh Henric Dhiki Wicaksono, Novi Dianasari, dan Muhammad Khoiril Wafi.	<i>Draft Dokumen B300</i>

1. PENDAHULUAN

1.1 Ringkasan Isi Dokumen

Dokumen ini berisi tentang uraian desain untuk merealisasikan produk berupa “Sistem Keamanan Kunci Pintu Gedung Berbasis IoT”. Uraian desain terdiri dari pemilihan solusi yang digunakan dalam menyelesaikan permasalahan serta gambaran cara kerja dari semua fitur yang telah dijabarkan. Dokumen ini juga akan menjelaskan desain dari subsistem yang ada yaitu perangkat kunci pintu, sistem komunikasi data dua arah, *database*, *server*, *website*, dan aplikasi *mobile*. Desain subsistem meliputi pemilihan kerangka kerja serta algoritma yang akan digunakan dalam mengimplementasikan sistem tersebut.

Dokumen ini selanjutnya akan digunakan sebagai acuan dalam proses implementasi “Sistem Keamanan Kunci Pintu Gedung Berbasis IoT” serta sebagai bahan evaluasi pada proses pengembangan selanjutnya.

1.2 Aplikasi Dokumen

Dokumen ini digunakan dalam proses pengembangan produk “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis IoT” untuk:

1. Menjelaskan pemilihan desain sesuai dengan subsistem yang ada.
2. Gambaran mengenai desain dan cara kerja sistem.
3. Menjelaskan standar-standar yang digunakan.
4. Referensi komponen atau *library* yang digunakan.
5. Menjadi catatan proses penggerjaan dan revisi.

Dokumen B300 ini diajukan kepada dosen pembimbing tugas akhir dan tim tugas akhir Program Studi Sarjana Teknik Elektro Undip sebagai bahan penilaian tugas akhir.

1.3 Referensi

- [1] I. Hermawan, D. Arnaldy, P. Oktivasari, and D. A. Fachrudin, “Development of Intelligent Door Lock System for Room Management Using Multi Factor Authentication,” vol. 16, no. 1, pp. 1–14, 2023.
- [2] K. Y. Sun, Y. Pernando, and M. I. Safari, “Perancangan Sistem IoT pada Smart Door Lock Menggunakan Aplikasi BLYNK,” *JUTSI (Jurnal Teknol. dan Sist. Informasi)*, vol. 1, no. 3, pp. 289–296, 2021, doi: 10.33330/jutsi.v1i3.1360.
- [3] C. Asiminidis, G. Kokkonis, and S. Kontogiannis, “Database Systems

Performance Evaluation for IoT Applications,” *SSRN Electron. J.*, no. November 2019, 2019, doi: 10.2139/ssrn.3360886.

- [4] N. A. Amir Hamzah, M. R. Abu Saad, W. Z. Wan Ismai, T. Bhunaeswari, and N. Z. Abd Rahman, “Development of A Prototype of An IoT Based Smart Home with Security System Flutter Mobile,” *J. Eng. Technol. Appl. Phys.*, vol. 1, no. 2, pp. 34–41, 2019, doi: 10.33093/jetap.2019.1.2.70.

1.4 Daftar Singkatan

Tabel 1.1 Daftar singkatan.

SINGKATAN	ARTI
IoT	<i>Internet of Things</i>
WiFi	<i>Wireless Fidelity</i>
JSON	<i>Javascript Object Notation</i>
IOS	<i>iPhone Operating System</i>
QR-Code	<i>Quick Response Code</i>
HTTP	<i>Hypertext Transfer Protocol</i>
API	<i>Application Programming Interface</i>
ERD	<i>Entity Relationship Diagram</i>
MVC	<i>Model-View Controller</i>
PHP	<i>Hypertext Preprocessor</i>
SPA	<i>Single Page Application</i>
AJAX	<i>Asynchronous Javascript and XML</i>
DOM	<i>Document Object Model</i>
OTP	<i>One-Time Password</i>
RFID	<i>Radio Frequency Identification</i>
ESP	<i>Espressif</i>
MySQL	<i>My Structured Structured Query Language</i>

Tabel 1.1 (lanjutan)

SINGKATAN	ARTI
HP	<i>Handphone</i>
IDE	<i>Integrated Development Environment</i>
URL	<i>Uniform Resource Locator</i>

2. SOLUSI DESAIN

Berdasarkan uraian dokumen B200 sebelumnya, dibutuhkan sebuah sistem yang dapat digunakan untuk mengelola kunci pintu gedung secara aman dan efisien dengan berbagai fitur yang dimiliki seperti pemantauan, pengaturan hak akses, kendali jarak jauh, dan lain sebagainya. Oleh sebab itu, diperlukan desain sistem yang dapat menjalankan semua fitur tersebut. Pemilihan desain didasarkan pada kebutuhan serta keamanan dari sistem untuk menjalankan semua fitur yang ada dengan menggunakan ilmu rekayasa yang terkait sehingga menghasilkan desain sistem yang aman dan efisien.

2.1 Solusi Desain 1

Solusi desain yang pertama yaitu menggunakan konsep IoT untuk mengelola semua kunci pintu yang sebelumnya berupa kunci fisik menjadi kunci digital. Pada setiap pintu dalam satu gedung akan terpasang sebuah perangkat kunci digital yang dapat dipantau serta dapat dikendalikan oleh semua pengguna gedung tersebut. Dengan menggunakan perangkat penguncian secara digital memungkinkan untuk melakukan pengelolaan kunci dengan lebih aman dan efektif.

2.2 Solusi Desain 2

Solusi desain yang kedua yaitu membangun sebuah layanan untuk mengendalikan sistem yang sudah dibangun oleh perangkat kunci digital. Sebuah *server* dapat menyediakan layanan untuk melakukan pemantauan serta memberikan perintah ke perangkat kunci pintu secara otomatis. Setiap perangkat kunci pintu akan terhubung ke *server* dengan menggunakan koneksi internet. *Server* juga dapat menentukan siapa saja yang diizinkan untuk mengakses pintu berdasarkan data yang tersimpan di dalamnya.

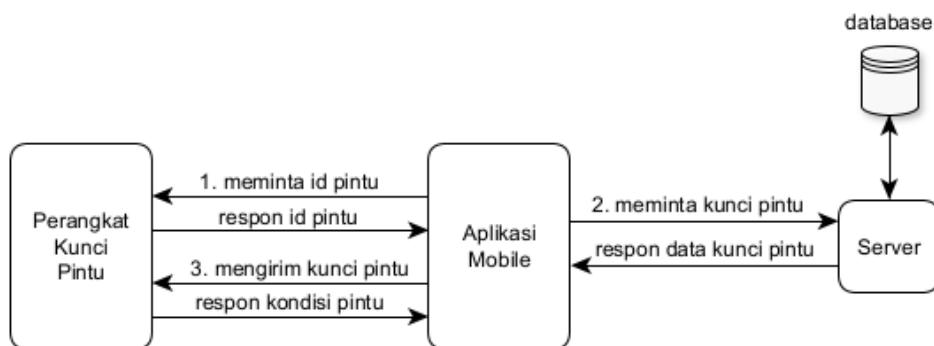
2.3 Solusi Desain 3

Solusi desain yang ketiga yaitu membuat sebuah tampilan sebagai media yang digunakan oleh pengguna gedung untuk berinteraksi dengan perangkat kunci pintu seperti membuka kunci atau melihat kondisi pintu secara jarak jauh. Sebuah aplikasi *mobile* dan *website* dapat digunakan sebagai antarmuka pengguna untuk berinteraksi dengan sistem. Adapun aplikasi *mobile* dapat digunakan untuk membuka atau memantau kondisi pintu dengan karakteristiknya yang ringkas dan mudah, sedangkan *website* digunakan untuk melakukan pengaturan sistem.

3. PERANCANGAN

3.1 Metode Verifikasi Akses

Setiap pintu pada gedung akan dikunci menggunakan perangkat kunci pintu digital sehingga dapat digunakan untuk melakukan verifikasi akses kepada semua pengguna yang ingin mengakses ruangan tersebut. Proses verifikasi akses dapat dilaksanakan dengan menggunakan berbagai macam metode. Penelitian yang dilakukan oleh Indra Hermawan dkk [1] tentang autentikasi pada kunci pintu pintar, RFID dapat digunakan untuk melakukan autentikasi kunci pintu. Namun, metode tersebut tidak menyediakan solusi yang tepat untuk menyelesaikan permasalahan, dikarenakan RFID hanya akan menggantikan tempat dari kunci fisik. Oleh karena itu, pada penelitian ini dikembangkan metode lain untuk melakukan autentikasi akses yaitu dengan metode *Access Control List* (ACL). Metode tersebut digunakan untuk menentukan apakah pengguna diizinkan untuk masuk ke ruangan atau tidak. Diagram proses verifikasi akses yang akan digunakan pada sistem ini dapat dilihat pada Gambar 3.1 di bawah ini.



Gambar 3.1 Diagram proses verifikasi akses.

Pada Gambar 3.1 di atas terlihat bahwa proses verifikasi akses melibatkan tiga bagian yaitu perangkat kunci pintu, aplikasi *mobile*, dan *server* menggunakan verifikasi tiga langkah dengan penjelasan sebagai berikut:

1. Meminta identitas pintu

Untuk membuka kunci pintu tentunya perlu mengetahui pintu mana yang akan dibuka. Identitas pintu akan membedakan pintu satu dengan lainnya sehingga setiap pintu akan memiliki identitas yang unik. Identitas pintu berupa kode QR sehingga memudahkan aplikasi *mobile* untuk mengenali pintu dengan memindai kode QR tersebut.

2. Meminta kunci pintu

Setelah mendapatkan identitas pintu, aplikasi *mobile* akan meminta kode kunci pintu dengan mengirimkan permintaan ke *server* disertai dengan identitas pengguna. Kemudian, *server* akan memeriksa data akses pengguna dengan menggunakan data identitas pintu dan pengguna. Jika aksesnya cocok, maka *server* akan mengembalikan kode kunci pintu yang digunakan untuk membuka pintu.

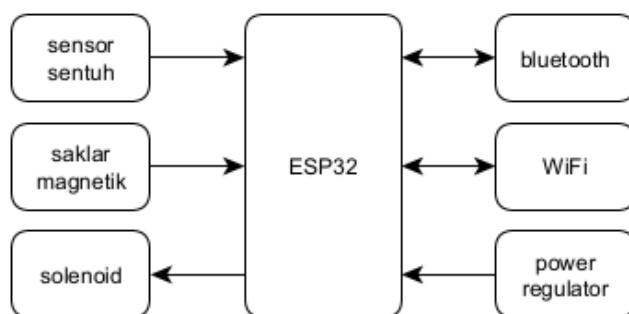
3. Mengirim kunci pintu

Setelah mendapatkan kode kunci pintu, aplikasi *mobile* akan mengirimkan kode kunci tersebut melalui koneksi *bluetooth* untuk membuka kunci pintu.

Berdasarkan penjelasan metode autentikasi di atas, proses untuk membuka kunci pintu yaitu menggunakan koneksi *bluetooth* dengan cara memindai kode QR statis via aplikasi *mobile* yang telah disediakan. Hal tersebut dimungkinkan adanya penduplikasian kode QR. Dengan adanya kekurangan tersebut, penggunaan *bluetooth* dapat dipastikan bahwa pengguna benar-benar memindai kode QR di area atau tepat di depan pintu karena karakteristik *bluetooth* yang memiliki batas jarak koneksi yang dekat. Adapun untuk membuka kunci pintu maka pengguna menggunakan kode kunci pintu dinamis. Kode tersebut akan berubah seiring dengan perubahan kondisi pintu sehingga proses autentikasi benar-benar aman dengan menggunakan kode kunci pintu yang dinamis.

3.2 Perangkat Penguncian

Untuk menjalankan proses penguncian secara digital, pada setiap pintu tentunya membutuhkan sebuah alat yang dapat digunakan untuk mengunci dan memantau kondisi pintu. Pada penilitian yang dilakukan oleh Kaleb Yefune Sun dkk [2] tentang sistem kunci pintu menggunakan BLYNK, ESP32 dapat digunakan untuk mengendalikan kunci pintu serta melakukan pemantauan kondisi pintu. Diagram perangkat penguncian dapat dilihat pada Gambar 3.2 di bawah ini.

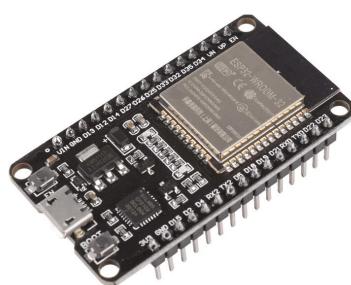


Gambar 3.2 Diagram perangkat penguncian.

Pada Gambar 3.2 di atas terlihat bahwa perangkat penguncian menggunakan ESP32 sebagai komponen utama dan beberapa komponen tambahan sebagai berikut:

1. **ESP32**

ESP32 menjadi komponen utama di dalam perangkat penguncian. ESP32 merupakan sebuah mikrokontroler sehingga dapat digunakan untuk mengatur dan memantau kondisi pintu dengan mengolah data yang diterima dari sensor dan aktuator.

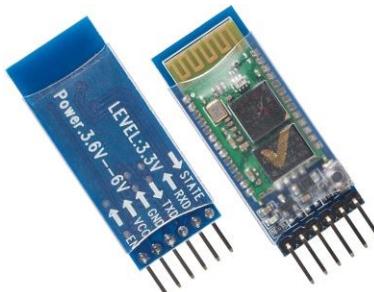


Gambar 3.3 ESP32 *development board*.

Gambar 3.3 merupakan sebuah papan pengembangan ESP32. Pada papan pengembangan tersebut terdapat pin-pin digital yang dapat digunakan untuk membaca sensor atau menggerakkan *actuator*. Pada papan pengembangan tersebut juga terdapat modul *programmer* yang digunakan untuk menjalankan program pada proses pembuatan kode program.

2. *Bluetooth*

Proses autentikasi akses pengguna akan menggunakan koneksi *bluetooth* untuk mengirim kode kunci. Oleh karena itu, perangkat kunci pintu harus mempunyai modul komunikasi *bluetooth* sebagai jalur komunikasi untuk berkomunikasi dengan aplikasi *mobile*. Modul *bluetooth* yang digunakan harus dapat berkomunikasi dengan ESP32 sebagai kontroler utama. Adapun perangkat penguncian ini menggunakan modul HC-05 sebagai modul *bluetooth* dengan komunikasi serial yang didukung oleh ESP32.



Gambar 3.4 Modul *bluetooth* HC-05.

Gambar 3.4 di atas merupakan modul *bluetooth* HC-05. Modul tersebut dapat berkomunikasi dengan perangkat *smartphone* melalui koneksi tanpa kabel (nirkabel). Pada modul *bluetooth* tersebut juga terdapat pin-pin yang digunakan untuk berkomunikasi dengan mikrokontroler melalui koneksi serial.

3. WiFi

Perangkat penguncian perlu terhubung dengan *server* untuk mengirimkan kondisi pintu atau menerima perintah dari jarak jauh. Oleh karena itu, ESP32 membutuhkan modul komunikasi yang dapat terhubung dengan internet. Pada ESP32 sudah tersedia modul WiFi yang dapat digunakan untuk terhubung ke internet sehingga tidak perlu menambah komponen WiFi eksternal.

4. Saklar magnetik

Perangkat penguncian harus dapat memantau kondisi pintu untuk selanjutnya dilaporkan ke *server*. Sebuah sensor diperlukan untuk melakukan tugas pemantauan. Sebuah sensor magnetik dapat digunakan untuk mendeteksi kondisi pintu terbuka atau tidak dengan memanfaatkan medan magnet.



Gambar 3.5 Saklar magnetik.

Gambar 3.5 di atas merupakan sebuah saklar magnetik. Sebuah saklar magnetik terdiri dari dua bagian yaitu bagian magnet dan saklar. Pada saat keduanya berdekatan maka saklar akan tertutup dan saat keduanya terpisah maka saklar akan terbuka. Oleh karena itu, dapat digunakan untuk mendeteksi kondisi pintu sedang terbuka atau tertutup.

5. Solenoid

Sebuah aktuator diperlukan untuk mengunci pintu secara fisik. Solenoid dapat digunakan untuk mengunci pintu fisik dengan kendali berupa arus listrik. Dengan menggunakan solenoid, kendali penguncian dapat dilakukan secara digital dengan mengirimkan arus listrik.



Gambar 3.6 Solenoid.

Gambar 3.6 di atas merupakan sebuah solenoid. Solenoid menggunakan prinsip elektromagnetik untuk menarik pin kunci sehingga dapat digunakan untuk mengunci pintu dengan memberikan arus listrik yang sesuai.

6. Sensor sentuh

Dalam kondisi normal, sebuah solenoid akan selalu mengunci pintu dan solenoid hanya dapat membuka kunci pintu dalam waktu yang relatif singkat atau tidak dapat membuka kunci pintu terus-menerus. Oleh karena itu, diperlukan mekanisme untuk mendeteksi pengguna yang membuka pintu untuk mengaktifkan solenoid. ESP32 menyediakan sebuah sensor sentuh yang dapat digunakan untuk mendeteksi sentuhan pada gagang pintu sehingga informasi tersebut dapat digunakan untuk mengaktifkan solenoid.

7. *Power regulator*

Untuk dapat beroperasi tentunya perangkat penguncian memerlukan arus listrik. Setiap komponen yang digunakan pada perangkat penguncian mungkin memerlukan voltase yang berbeda sehingga diperlukan untuk menyediakan voltase yang sesuai dengan kebutuhan setiap komponen.



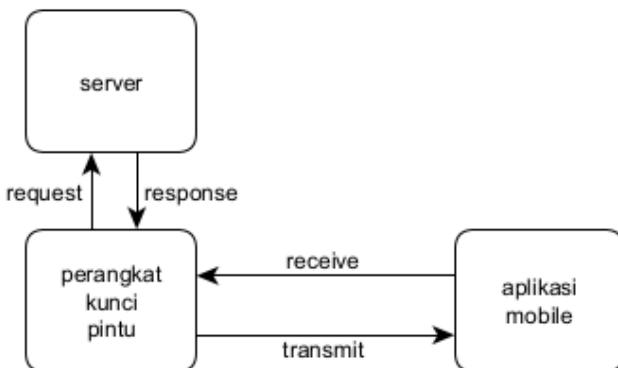
Gambar 3.7 Regulator 5 volt.

Gambar 3.7 di atas merupakan sebuah regulator untuk menurunkan tegangan masukan menjadi 5 volt. Perangkat penguncian menggunakan tegangan masukan sebesar 12 volt. Hal tersebut sesuai dengan voltase kerja dari solenoid, sedangkan ESP32 bekerja pada voltase 5 volt sehingga diperlukan sebuah regulator untuk menurunkan tegangan 12 volt menjadi 5 volt.

3.3 Komunikasi Data Dua Arah

Perangkat penguncian perlu berkomunikasi dengan perangkat lain seperti *smartphone* dan *server*, baik itu untuk menerima perintah atau mengirimkan respon. Oleh

karena itu, perangkat penguncian membutuhkan komunikasi data dua arah untuk mengirim dan menerima data yang akan digunakan pada proses selanjutnya.



Gambar 3.8 Komunikasi data dua arah.

Gambar 3.8 di atas menampilkan diagram komunikasi yang dilakukan oleh perangkat kunci pintu dengan *server* dan aplikasi *mobile*. Komunikasi yang dilakukan oleh perangkat kunci pintu terbagi menjadi dua yaitu komunikasi dengan *server* dan komunikasi dengan aplikasi *mobile* dengan penjelasan sebagai berikut:

1. Komunikasi dengan *server*

Perangkat kunci pintu berkomunikasi dengan *server* untuk menerima perintah secara jarak jauh dan untuk melaporkan kondisi pintu secara *realtime*. Untuk menjalin komunikasi tersebut, perangkat kunci pintu menggunakan protokol HTTP dengan metode API. Perangkat kunci pintu menggunakan API yang telah disediakan oleh *server* untuk melaporkan kondisi pintu.

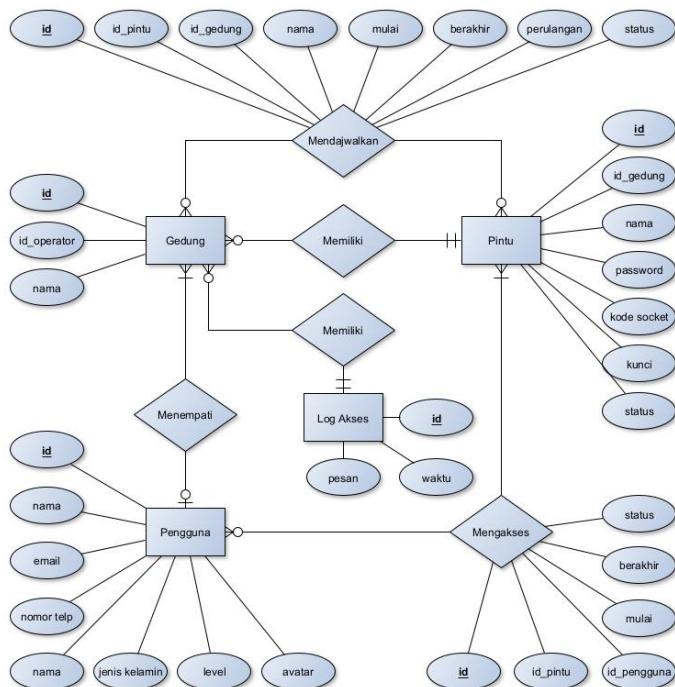
Untuk menerima perintah dari *server*, metode yang digunakan sedikit berbeda dikarenakan perangkat kunci pintu bertindak sebagai *client*. Untuk menerima perintah dari *server*, perangkat kunci pintu menggunakan protokol *websocket*. *WebSocket* digunakan untuk menjalin koneksi dengan *server* dan koneksi tersebut akan terus terjalin. Dengan menggunakan koneksi tersebut, perangkat kunci pintu dapat menerima *event* dari *server* meskipun perangkat kunci pintu tidak melakukan permintaan data. Dengan menggunakan metode tersebut, perintah dapat dikirimkan secara langsung tanpa menunggu *polling* yang dilakukan oleh perangkat kunci pintu.

2. Komunikasi dengan aplikasi *mobile*

Untuk berkomunikasi dengan aplikasi *mobile*, perangkat kunci pintu menggunakan koneksi *bluetooth* yang umumnya disediakan oleh semua perangkat *smartphone*. Dengan menggunakan *bluetooth* maka proses autentikasi hanya bisa dilakukan di dekat perangkat kunci pintu dan juga memberikan keuntungan bahwa perangkat kunci pintu tetap dapat bekerja meskipun tanpa koneksi intenet.

3.4 Database

Database digunakan untuk menyimpan data yang digunakan dalam sistem penguncian pintu gedung seperti data pengguna, data pintu, data akses, dan data lainnya. *Database* yang digunakan harus dapat menunjang kinerja sistem dengan karakteristik respon yang cepat dan pengelolaan data terstruktur. Berdasarkan penelitian [3] yang membandingkan kinerja dari berbagai tipe dan jenis *database*, didapatkan bahwa penggunaan MySQL menunjukkan hasil kinerja yang bagus dalam hal waktu eksekusi permintaan. Dengan sistem penyimpanan data bersifat relasional dan terstruktur maka MySQL dapat diterapkan pada sistem penguncian pintu gedung ini.

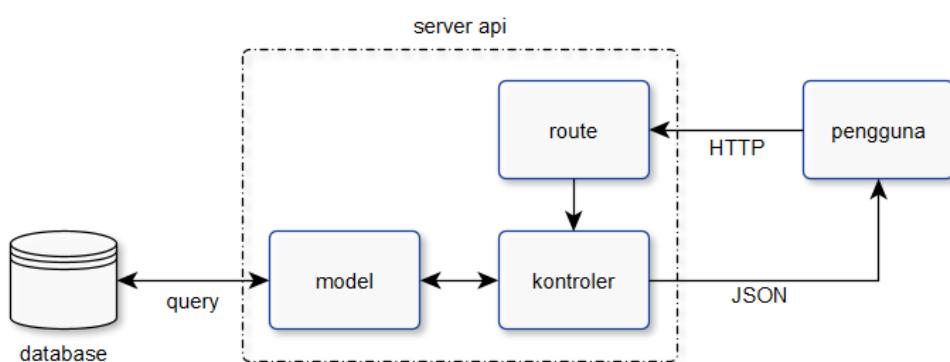


Gambar 3.9 ERD database.

Pada Gambar 3.9 di atas terlihat relasi antar entitas di dalam *database* sistem penguncian pintu gedung. Pada *database* sistem penguncian pintu gedung terdapat beberapa entitas seperti pengguna, pintu, gedung dan *log* akses. Entitas tersebut digunakan untuk meyimpan data yang akan digunakan dalam pengelolaan sistem dengan isi data sesuai dengan atribut dari masing-masing entitas. Di dalam sistem *database* juga terdapat relasi. Relasi menunjukkan hubungan antar entitas seperti menempati, memiliki, menjadwalkan, dan mengakses. Khusus untuk relasi menjadwalkan dan mengakses pada Gambar 3.9 terlihat bahwa relasi tersebut memiliki atribut yang menjadi penghubung antara dua entitas yang berbeda. Pada umumnya, kondisi tersebut menunjukkan kardinalitas *many-to-many* yang bertindak sebagai entitas pivot untuk menghubungkan relasi *many-to-many* dari dua entitas lainnya.

3.5 Backend API

Sebuah API dibutuhkan sebagai layanan komunikasi yang digunakan oleh perangkat kunci pintu dan aplikasi *mobile* untuk berkomunikasi dengan *server*. Selain terdapat *website* yang digunakan sebagai antarmuka utama dalam mengelola sistem ini, juga terdapat aplikasi berbasis *mobile* yang digunakan untuk menunjang kinerja dari sistem terutama pada bagian yang membutuhkan teknologi yang belum disediakan oleh *browser* seperti koneksi *bluetooth* dan pindai kode QR menggunakan kamera.



Gambar 3.10 Diagram *backend API*.

Pada Gambar 3.10 di atas terlihat diagram *backend API*. *Backend API* dibangun menggunakan konsep MVC yang disediakan oleh laravel. Laravel merupakan sebuah kerangka kerja pengembangan *website* dan API dengan bahasa pemrograman PHP yang

menggunakan konsep MVC. Dengan menggunakan Laravel, proses pengembangan API dapat dilakukan dengan cepat serta menggunakan berbagai modul-modul yang telah tersedia seperti autentikasi, koneksi *database*, *query*, token, dan lain sebagainya. API akan menyediakan layanan berupa *endpoint* yang dapat diakses oleh perangkat kunci pintu maupun aplikasi *mobile*. *Endpoint* yang tersedia dapat dilihat pada Tabel 3.1 di bawah.

Tabel 3.1 *Endpoint API*.

Tipe	Endpoint	Auth	Keterangan
POST	/api/login	-	<i>login</i> pengguna dan operator
POST	/api/reset-password	-	<i>reset password</i> menggunakan email
POST	/api/verify-email	<i>Sanctum</i>	verifikasi email
GET	/api/logout	<i>Sanctum</i>	<i>logout</i> pengguna dan operator
POST	/api/update-profile	<i>sanctum, verified</i>	<i>update profile</i> nama, email, dan lainnya
GET	/api/avatar	<i>sanctum, verified</i>	mengambil gambar avatar
POST	update-avatar	<i>sanctum, verified</i>	mengubah gambar avatar
POST	/api/change-password	<i>sanctum, verified</i>	mengubah <i>password</i> pengguna atau operator
GET	/api/my-access	<i>sanctum, verified</i>	mengambil daftar akses
GET	/api/get-doors	<i>sanctum, verified</i>	mengambil daftar pintu
GET	/api/my-history	<i>Sanctum, verified</i>	mengambil daftar riwayat akses
GET	/api/verify-access/{door_id}	<i>sanctum, verified</i>	verifikasi akses dari kode QR pintu
POST	/api/remote-access	<i>sanctum, verified</i>	membuka atau mengunci pintu jarak jauh
POST	/door/login	-	<i>login</i> perangkat kunci pintu
POST	/door/register	-	menambahkan perangkat kunci pintu baru

Tabel 3.1 (lanjutan)

Tipe	Endpoint	Auth	Keterangan
GET	/door/logout	<i>Sanctum</i>	<i>logout</i> perangkat kunci pintu
POST	/door/get-signature	<i>Sanctum</i>	mengambil kode <i>signature channel pusher</i>
POST	/door/update-status	<i>Sanctum</i>	<i>update</i> status pintu
POST	/door/alert	<i>Sanctum</i>	peringatan pada pintu

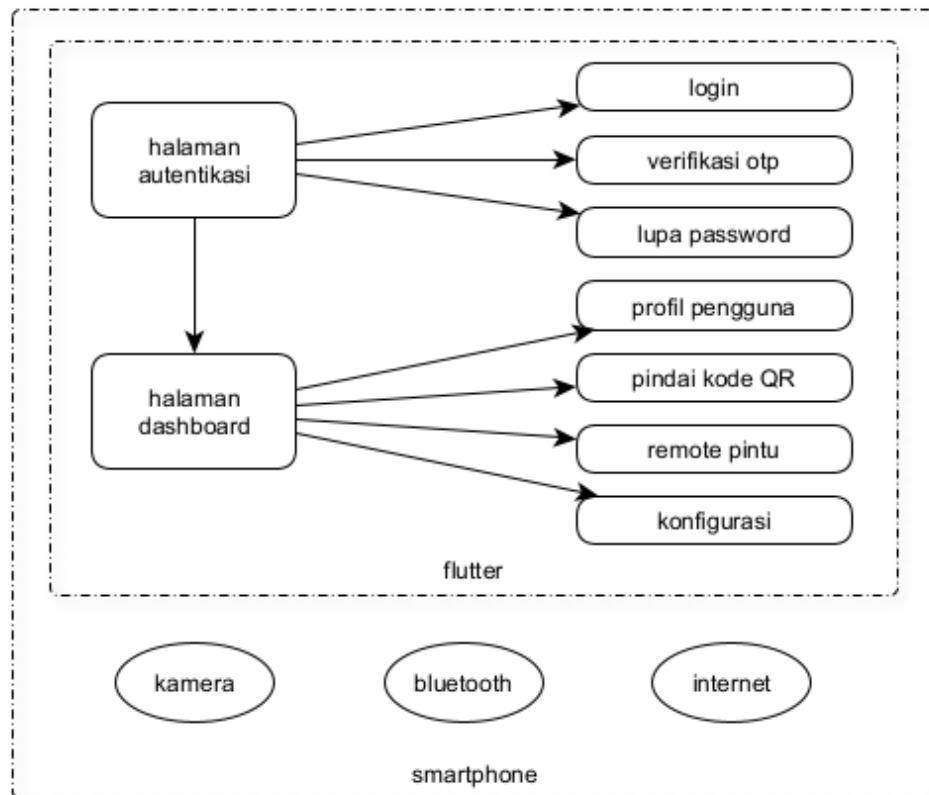
Dapat dilihat pada Tabel 3.1 di atas, API terbagi menjadi 2 bagian yang ditandai dengan awalan (*prefix*) “/api” dan “/door”. Awalan “/api” merupakan *endpoint* API yang khusus ditujukan untuk penggunaan aplikasi *mobile* pengguna dan operator, sedangkan awalan “/door” merupakan API yang ditujukan untuk perangkat kunci IoT. Dengan menggunakan API yang terpisah maka API akan semakin terorganisasi dengan baik.

Pada Tabel 3.1 juga terlihat bahwa beberapa *endpoint* memerlukan autentikasi *sanctum* dan *verified*. Autentikasi *sanctum* merupakan sebuah metode autentikasi berbasis token yang digunakan untuk mengamankan sumber daya API dari *client* dengan hanya mengizinkan pengguna yang sudah terautentikasi yang dapat mengakses API tersebut. Sedangkan, *verified* merupakan autentikasi tambahan yang digunakan untuk memastikan bahwa *client* (pengguna dan operator) sudah melakukan verifikasi email sehingga dapat meningkatkan keamanan API.

3.6 Aplikasi Mobile

Sebuah aplikasi *mobile* diperlukan untuk membuka kunci pintu dengan menggunakan kode QR. Aplikasi *mobile* akan menyediakan fitur koneksi *bluetooth* yang belum tersedia pada sebuah *website* sehingga proses verifikasi akses dapat berjalan dengan baik. Untuk menjalankan proses verifikasi akses dan fitur lainnya, maka aplikasi *mobile* yang dikembangkan harus memiliki beberapa kemampuan yang mendukung kinerja dari sistem penguncian seperti koneksi internet, koneksi *bluetooth*, serta pemindai kode QR. Pada penelitian yang dilakukan oleh Nur Asyiqin Bt. Amir Hamzah dkk [4] tentang perancangan rumah pintar berbasis IoT, kerangka kerja *Flutter* dapat digunakan dalam pengembangan aplikasi *mobile*. *Flutter* merupakan sebuah kerangka kerja untuk membuat aplikasi *mobile*. *Flutter* merupakan kerangka kerja *cross-platform* yang artinya aplikasi yang dihasilkan dapat dijalankan pada sistem operasi yang berbeda misalnya

Android dan IOS dengan menggunakan satu struktur program yang sama. Secara umum, aplikasi yang akan dikembangkan digunakan untuk berinteraksi dengan perangkat IoT misalnya untuk membuka kunci pintu, melihat kondisi pintu, dan melakukan pengaturan.



Gambar 3.11 Struktur aplikasi *mobile*.

Gambar 3.11 memperlihatkan struktur dari aplikasi *mobile* yang akan dikembangkan. Pada aplikasi *mobile* akan terdapat beberapa layar yang dapat digunakan untuk berinteraksi dengan sistem kunci pintu. Penjelasan mengenai setiap layar dapat dilihat pada Tabel 3.2 di bawah ini.

Tabel 3.2 Pembagian layar aplikasi *mobile*.

Nama	Autentikasi	Penjelasan
Login	-	Halaman autentikasi untuk <i>login</i> pengguna dan operator. Proses <i>login</i> menggunakan alamat email dan <i>password</i> yang sudah terdaftar.

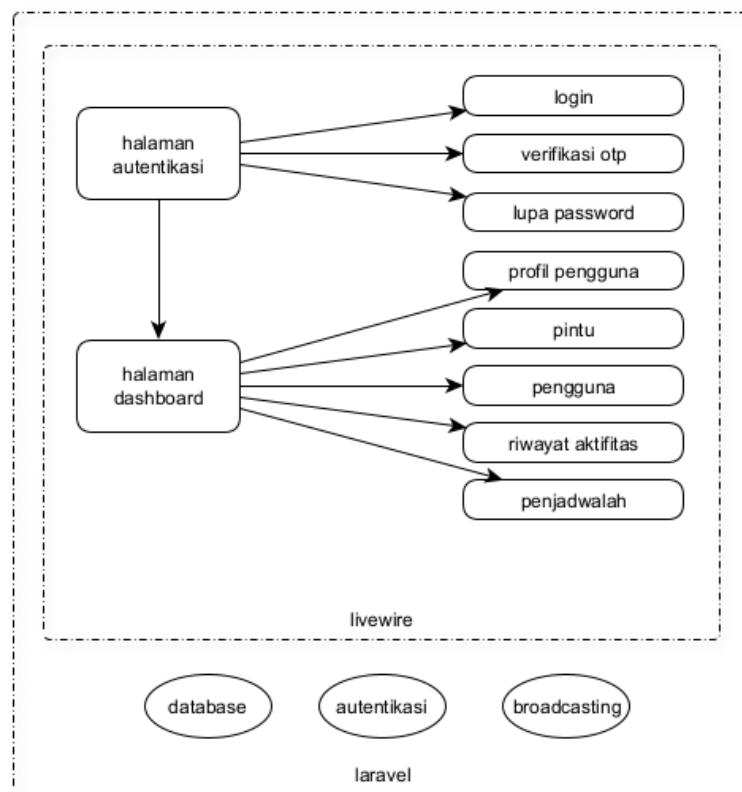
Tabel 3.2 (lanjutan)

Nama	Autentikasi	Penjelasan
Verifikasi OTP	-	Jika email belum terverifikasi maka akan diarahkan ke halaman verifikasi. Verifikasi dengan menggunakan kode OTP yang dikirimkan ke email yang dimasukkan pada saat <i>login</i> .
Lupa password	<i>Login</i>	Pengguna aplikasi dapat memanfaatkan fitur ini jika pengguna kehilangan <i>password</i> , dengan cara memasukkan alamat email yang sesuai untuk menerima petunjuk lebih lanjut.
Profil pengguna	<i>Login</i>	Proses pengaturan pengguna dilakukan pada halaman profil pengguna. Pengguna aplikasi dapat menyesuaikan informasi diri seperti nama, nomor hp, dan foto profil.
Pindai kode QR	<i>Login</i>	Pengguna dapat menggunakan kode QR pada pintu untuk membuka kunci pintu. Pada halaman ini, pengguna memindai kode tersebut dan aplikasi akan meminta akses ke <i>server</i> secara otomatis.
Remote pintu	<i>Login</i> , Operator	Operator dapat membuka atau mengunci pintu secara jarak jauh melalui koneksi internet. Pada halaman ini juga menampilkan semua pintu dan kondisinya.
Konfigurasi	<i>Login</i> , Operator	Halaman ini digunakan untuk melakukan kofigurasi pada perangkat kunci pintu seperti mengatur kredensial WiFi atau mendaftarkan perangkat baru ke <i>server</i> .

Pada Gambar 3.11 juga terlihat bahwa aplikasi membutuhkan dukungan dari kamera, internet, dan *bluetooth*. Pada kerangka kerja *flutter*, penggunaan *peripheral* pada *smartphone* dapat dilakukan dengan menggunakan beberapa modul yang telah disediakan seperti modul *http* untuk menjalankan fungsi yang berkaitan dengan internet, modul *bluetooth_serial* untuk menjalankan komunikasi *bluetooth* sederhana, dan *qr_scanner* untuk menjalankan fungsi pemindaian menggunakan kamera.

3.7 Tampilan Website

Sistem keamanan kunci pintu gedung ini juga memerlukan sebuah *website* yang digunakan untuk melakukan pengaturan pada sistem tersebut. Dengan menggunakan *website* maka proses pengaturan menjadi lebih fleksibel. Proses pengaturan yang dilakukan meliputi pengelolaan pintu, pengguna, akses, penjadwalan, dan riwayat aktivitas. *Website* tersebut hanya bisa diakses oleh pengelola gedung sebagai operator yang menjadi penanggung jawab terhadap sistem keamanan yang dijalankan.



Gambar 3.12 Struktur website.

Pada Gambar 3.12 memperlihatkan struktur dari *website* yang akan dikembangkan. Seperti halnya *backend API*, *website* dikembangkan dengan menggunakan kerangka kerja Laravel dengan memanfaatkan modul-modul yang telah disediakan seperti koneksi *database*, autentikasi, dan *broadcasting*. Seperti pada apliksi *mobile*, *website* memiliki beberapa layar yang digunakan oleh operator untuk mengatur sistem seperti menambahkan pintu baru, membuat jadwal, menambahkan akses, menambahkan pengguna, dan lain sebagainya. Penjelasan fungsi setiap layar pada *website* dapat dilihat pada Tabel 3.3 di bawah ini.

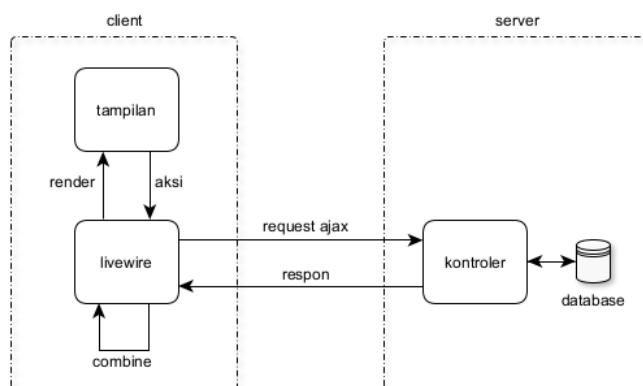
Tabel 3.3 Pembagian layar *website*.

Nama	Autentikasi	Penjelasan
<i>Login</i>	-	Halaman autentikasi untuk <i>login</i> pengguna dan operator. Proses <i>login</i> menggunakan alamat email dan <i>password</i> yang sudah terdaftar.
Verifikasi OTP	-	Jika email belum terverifikasi maka akan diarahkan ke halaman verifikasi. Verifikasi dengan menggunakan kode OTP yang dikirimkan ke email yang dimasukkan pada saat <i>login</i> .
Lupa <i>password</i>	<i>Login</i>	Pengguna aplikasi dapat memanfaatkan fitur ini jika pengguna kehilangan <i>password</i> dengan cara memasukkan alamat email yang sesuai untuk menerima petunjuk lebih lanjut.
Profil pengguna	<i>Login</i> , Operator	Proses pengaturan pengguna dilakukan pada halaman profil pengguna. Pengguna dapat menyesuaikan informasi diri seperti nama, nomor hp, dan foto profil.
Pengguna	<i>Login</i> , Operator	Halaman pengguna digunakan untuk menampilkan semua pengguna yang terdaftar disertai dengan akses yang dimiliki, serta untuk menambah dan menghapus pengguna.

Tabel 3.3 Lanjutan.

Nama	Autentikasi	Penjelasan
Penjadwalan	Login, Operator	Halaman penjadwalan digunakan untuk mengatur jadwal seperti menambahkan, memperbarui, dan menghapus jadwal.
Riwayat akses	Login, Operator	Halaman ini digunakan untuk menampilkan semua aktivitas pengguna pada sistem keamanan kunci pintu gedung.

Untuk meningkatkan kenyamanan dan performa dari *website* maka pembuatan setiap tampilan akan menggunakan *livewire*. *Livewire* merupakan sebuah *library* pada laravel yang digunakan untuk menerapkan konsep *Single Page Application* (SPA) pada Laravel. Laravel sendiri merupakan kerangka kerja yang bersifat *server-side* yaitu semua pengolahan data akan dilakukan di dalam *server* dan hasilnya akan dikirimkan kembali ke *client* berupa tampilan dalam format http. Oleh sebab itu, setiap ada interaksi pengguna maka seluruh halaman akan diperbarui sehingga menjadikan tampilan kurang nyaman dan dapat menurunkan performa *website* karena harus memuat seluruh halaman dari awal.



Gambar 3.13 Cara kerja *livewire*.

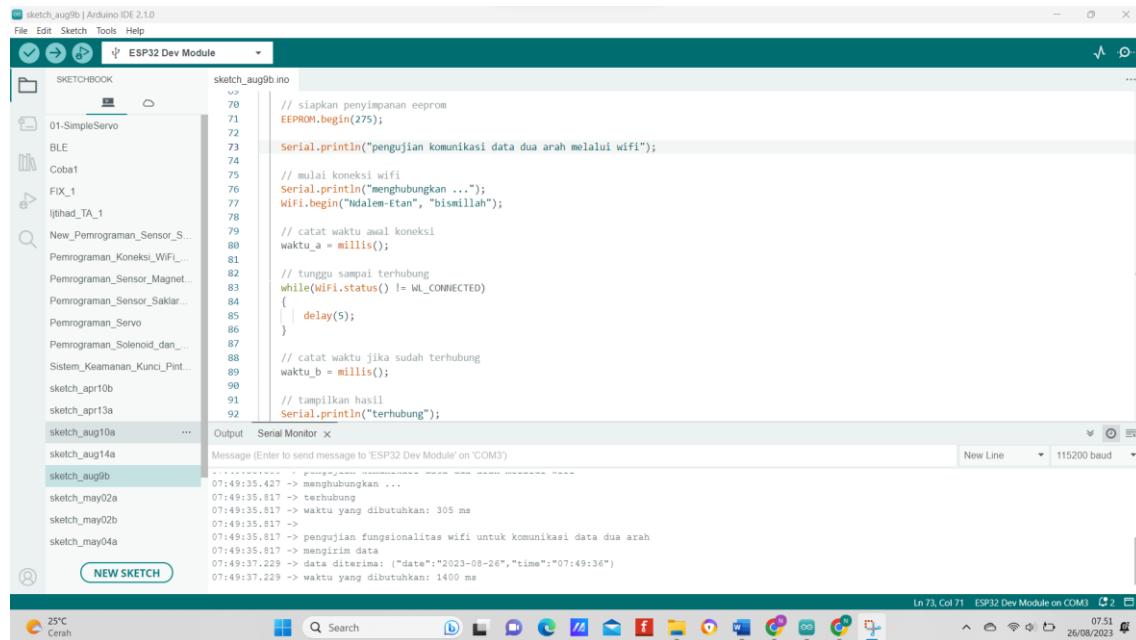
Pada Gambar 3.13 terlihat cara kerja dari *library* *livewire*. *Livewire* bekerja dengan menggunakan sebuah *request* AJAX untuk memanggil sebuah metode pada kontroler. Jika pada tampilan pengguna melakukan sebuah aksi seperti menekan tombol maka *livewire* akan mengirimkan sebuah permintaan ke *server* dengan menggunakan

AJAX. Permintaan dikirimkan ke sebuah *endpoint* API yang dibuat secara otomatis oleh *livewire* untuk menjalankan *method-method* yang ada pada kontroler. Kontroler akan mengirimkan respon berupa DOM hasil dari perubahan yang dilakukan. Setelah diterima oleh *client* maka DOM tersebut akan dibantingkan dan digadungkan dengan DOM yang sedang tampil dan kemudian di-*render*. Dengan menggunakan metode tersebut maka proses pembaruan tampilan hanya dilakukan pada bagian yang berubah dengan menggunakan manipulasi DOM tanpa memuat seluruh tampilan dari awal.

4. VERIFIKASI

4.1 Komunikasi WiFi dan Bluetooth

Dalam pengembangan proyek-proyek elektronik menggunakan *platform* seperti Arduino, komunikasi nirkabel adalah hal yang umum dilakukan. Komunikasi WiFi digunakan untuk mentransfer data atau mengendalikan perangkat dari jarak jauh melalui koneksi internet sedangkan komunikasi *bluetooth* lebih cocok untuk jarak pendek.



```

sketch_aug9b | Arduino IDE 2.1.0
File Edit Sketch Tools Help
ESP32 Dev Module
Sketchbook sketch_aug9b.ino
01-SimpleServo
BLE Coba1
FIX_1
Ijihad_TA_1
New_Pemrograman_Sensor_S...
Pemrograman_Koneksi_WiFi_...
Pemrograman_Sensor_Magnet...
Pemrograman_Sensor_Saklar...
Pemrograman_Servo
Pemrograman_Solenoid_dan...
Sistem_Kemanan_Kunci_Pint...
sketch_apr10b
sketch_apr13a
sketch_apr10a ...
sketch_apr14a
sketch_aug9b
sketch_may02a
sketch_may02b
sketch_may04a
Output Serial Monitor x
Message (Enter to send message to 'ESP32 Dev Module' on 'COM3')
07:49:35.427 -> menghubungkan ...
07:49:35.817 -> terhubung
07:49:35.817 -> waktu yang dibutuhkan: 305 ms
07:49:35.817 -> pengujian fungionalitas wifi untuk komunikasi data dua arah
07:49:35.817 -> mengirim data
07:49:37.229 -> data diterima: {"date":"2023-08-26","time":"07:49:36"}
07:49:37.229 -> waktu yang dibutuhkan: 1400 ms
Ln 73, Col 71 ESP32 Dev Module on COM3 0 2
25°C Cerah 07:51 26/08/2023

```

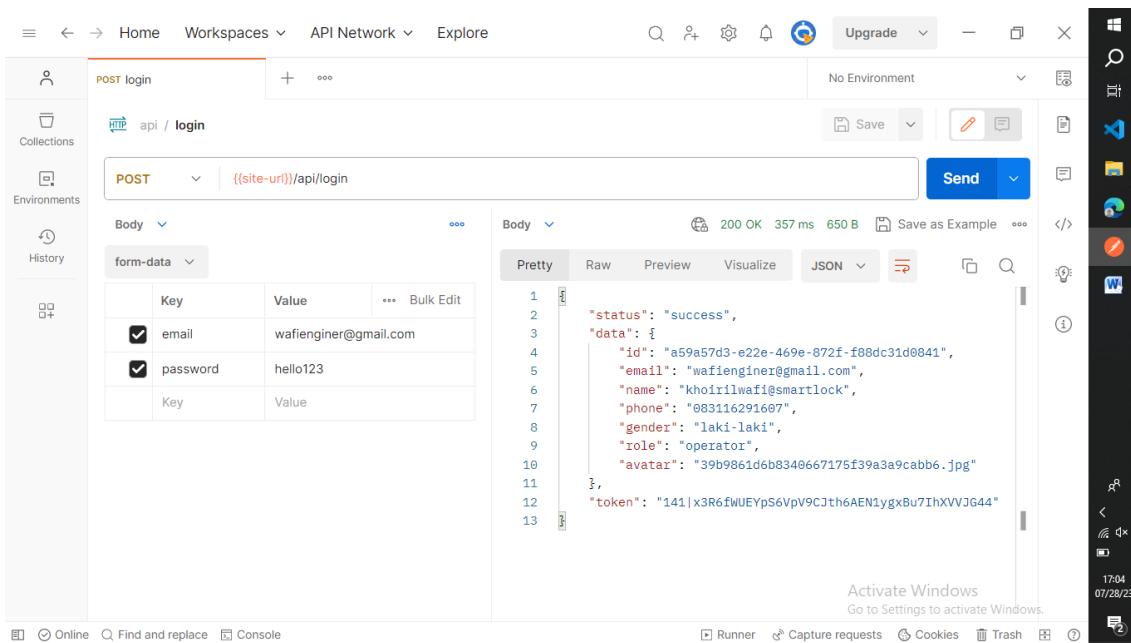
Gambar 4.1 Tampilan serial monitor arduino.

Gambar 4.1 di atas terlihat tampilan serial monitor Arduino. Serial Monitor adalah alat yang digunakan untuk memantau data yang dikirimkan melalui *port serial* antara mikrokontroler dan komputer. Dapat dilihat pada Gambar 4.1 bahwa komputer berhasil terhubung dan melakukan komunikasi data dengan mikrokontroler melalui koneksi WiFi.

Oleh karena itu, penggunaan Arduino IDE untuk pengembangan dan pemrograman perangkat-perangkat yang menggunakan mikrokontroler dapat digunakan pada proses selanjutnya.

4.2 Backend API Laravel

Proses pengembangan *backend* API dilakukan dengan menggunakan kerangka kerja Laravel. Laravel menyediakan modul-modul yang dapat langsung digunakan untuk membuat sebuah API.



Gambar 4.2 API laravel.

Gambar 4.2 di atas memperlihatkan contoh hasil *request* API dengan menggunakan *postman*. *Request* yang dikirimkan ke *endpoint* “/api/login” merupakan contoh proses *login* melalui API yang akan diimplementasikan pada aplikasi *mobile*. Dapat dilihat pada Gambar 4.2, *postman* berhasil melakukan *login* dengan mendapatkan respon kode 200 dan status sukses sehingga penggunaan laravel untuk membuat sebuah API dapat digunakan pada proses selanjutnya.

4.3 Kode QR

Kode QR adalah jenis kode batang yang terdiri dari pola-pola hitam dan putih yang dapat di-*scan* oleh perangkat seperti kamera *smartphone*. Kode QR mengandung informasi tertentu, seperti URL, teks, nomor telepon, atau bahkan informasi kontak. Begitu kode QR di-*scan*, informasi yang terkandung dalam kode dapat dengan mudah diakses oleh perangkat yang melakukan pemindaian.



Gambar 4.3 Pemindaian kode QR melalui aplikasi *mobile*.

Pada Gambar 4.3 di atas memperlihatkan tampilan layar aplikasi *mobile* saat memproses informasi dari kode QR. Penggunaan kode QR untuk pindai akses cepat akan digunakan pada proses selanjutnya.

5. PENUTUP

Dokumen B300 menjelaskan desain perancangan sistem keamanan kunci pintu gedung berbasis IoT yang meliputi perancangan subsistem komunikasi data dua arah, perangkat penguncian, *database*, *server*, metode untuk melakukan verifikasi akses, dan aplikasi *mobile*. Hasil perancangan desain sistem keamanan kunci pintu gedung berbasis IoT akan menjadi acuan dalam proses implementasi.



Dokumen Pengembangan Produk Lembar Sampul Dokumen

Judul Dokumen

TUGAS AKHIR:
Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*

Jenis Dokumen

IMPLEMENTASI

Catatan: Dokumen ini dikendalikan penyebarannya oleh Dept. Teknik Elektro Undip

Nomor Dokumen

B400-01-TA2223.2.19012

Nomor Revisi

01

Nama File

B400-2-TA2223

Tanggal Penerbitan

8 September 2023

Unit Penerbit

Departemen Teknik Elektro Undip

Jumlah Halaman

67 (termasuk lembar sampul ini)

Data Pengusul				
Pengusul	Nama NIM	Henric Dhiki Wicaksono 21060119120011	Jabatan	Anggota
	Nama NIM	Novi Dianasari 21060119120039	Jabatan	Anggota
	Nama NIM	Muhammad Khoiril Wafi 21060119140133	Jabatan	Anggota
Pembimbing Utama	Nama NIP	M. Arfan, S.Kom., M.Eng. 198408172015041002	Tanda Tangan	
Pendamping	Nama NIP	Imam Santoso, S.T., M.T. 197012031997021001	Tanda Tangan	

DAFTAR ISI

1.	PENDAHULUAN	4
1.1	Ringkasan Isi Dokumen.....	4
1.2	Aplikasi Dokumen	4
1.3	Daftar Singkatan	4
2.	IMPLEMENTASI.....	5
2.1	Perangkat Keras Kunci Pintu	5
2.2	Perangkat Lunak Kunci Pintu	10
2.3	<i>Database</i>	21
2.4	<i>Backend API</i>	26
2.5	<i>Server</i>	35
2.6	Tampilan <i>Website</i>	37
2.7	Aplikasi <i>Mobile</i>	52
3.	PENUTUP	67

Catatan Sejarah Perbaikan Dokumen

VERSI, TGL, OLEH	PERBAIKAN
01, 0 September 2023, oleh Henric Dhiki Wicaksono, Novi Dianasari, dan Muhammad Khoiril Wafi.	<i>Draft Dokumen B400</i>

1. PENDAHULUAN

1.1 Ringkasan Isi Dokumen

Dokumen ini berisi penjelasan mengenai proses implementasi yang dilakukan untuk mewujudkan produk tugas akhir “Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*”. Proses implementasi yang dilakukan meliputi pembuatan perangkat kunci pintu berupa *hardware* dan *software*-nya, pembuatan tampilan baik itu berupa *website* maupun aplikasi *mobile*, serta pembuatan *database* dan *backend server*. Proses implementasi yang dilakukan akan mengikuti desain yang telah dibuat dan dijelaskan pada dokumen desain (B300).

1.2 Aplikasi Dokumen

Dokumen ini digunakan dalam proses pengembangan “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*” untuk:

- 1) Sebagai penjelasan proses implementasi dari sistem yang telah dirancang baik dari segi *hardware* maupun *software*.
- 2) Sebagai acuan dalam proses pengujian sistem pada tahap selanjutnya.
- 3) Sebagai dokumentasi dan pencatatan perubahan.

Dokumen B400 ini diajukan kepada dosen pembimbing tugas akhir dan tim tugas akhir Program Studi Sarjana Teknik Elektro Undip sebagai bahan penilaian tugas akhir.

1.3 Daftar Singkatan

Tabel 1.1 Daftar singkatan

SINGKATAN	ARTI
IoT	<i>Internet of Things</i>
WiFi	<i>Wireless Fidelity</i>
JSON	<i>Javascript Object Notation</i>
HTTPS	<i>Hypertext Transfer Protocol Secure</i>
PC	<i>Personal Computer</i>
IDE	<i>Integrated Development Environment</i>
UI	<i>User Interface</i>

Tabel 1.1 (lanjutan)

SINGKATAN	ARTI
QR-Code	<i>Quick Response Code</i>
SSL	<i>Secure Sockets Layer</i>
ESP	<i>Espressif</i>
LED	<i>Light Emitting Diode</i>
SSID	<i>Service Set Identifier</i>
EEPROM	<i>Electrically Erasable Programmable Read-Only Memory</i>
MySQL	<i>My Structured Query Language</i>
HP	<i>Handphone</i>
API	<i>Application Programming Interface</i>
PHP	<i>Hypertext Preprocessor</i>
OTP	<i>One Time Password</i>
SMTP	<i>Simple Mail Transfer Protocol</i>

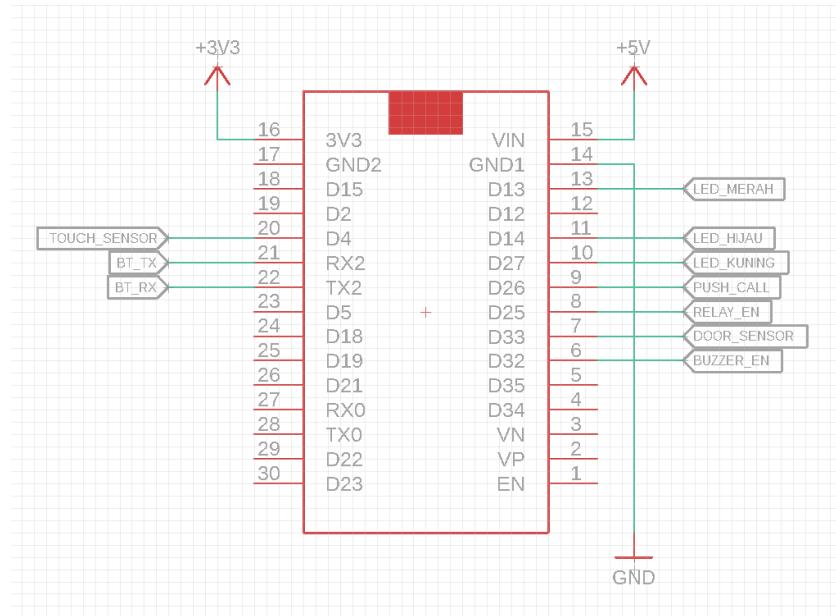
2. IMPLEMENTASI

2.1 Perangkat Keras Kunci Pintu

Proses implementasi perangkat keras kunci pintu dilakukan dengan merangkai komponen-komponen yang telah ditentukan sebelumnya menjadi sebuah alat yang akan digunakan untuk mengunci pintu secara digital. Beberapa komponen yang digunakan adalah sebagai berikut:

1. ESP32

Diperlukan sebuah mikrokontroler yang bertugas untuk mengelola semua aktivitas pada perangkat kunci pintu. Pada proses desain dijelaskan penggunaan ESP32 sebagai kontroler utama dengan memanfaatkan modul komunikasi WiFi yang sudah tersedia di dalamnya.



Gambar 2.1 Konfigurasi ESP32

Gambar 2.1 di atas memperlihatkan penggunaan ESP32 sebagai kontroler utama pada perangkat kunci pintu. ESP32 akan mengatur semua *input* dan *output* pada perangkat kunci pintu dengan penjelasan sesuai dengan Tabel 2.1 di bawah ini.

Tabel 2.1 Konfigurasi pin ESP32

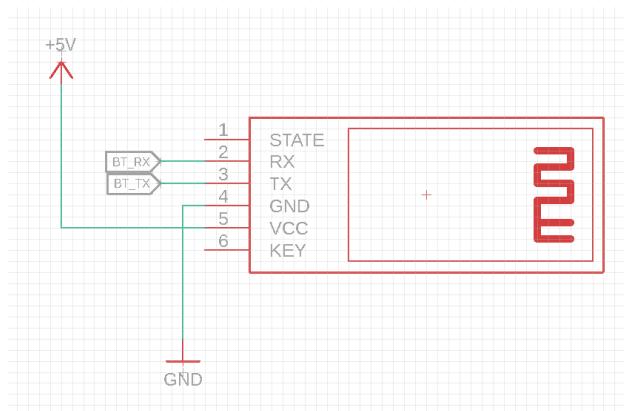
Pin	Mode	Keterangan
D4	<i>Input</i>	Digunakan sebagai sensor sentuh untuk mendeteksi interaksi pengguna.
RX2	<i>Input</i>	Terhubung dengan modul <i>bluetooth</i> untuk menerima data.
TX2	<i>Output</i>	Terhubung dengan modul <i>bluetooth</i> untuk mengirim data.
D13	<i>Output</i>	Terhubung dengan LED indikator status penguncian.
D14	<i>Output</i>	Terhubung dengan LED indikator proses pengiriman data.
D27	<i>Output</i>	Terhubung dengan LED indikator status koneksi WiFi.
D26	<i>Input</i>	Terhubung dengan tombol untuk membuka pintu dari dalam ruangan.
D25	<i>Output</i>	Terhubung dengan modul <i>relay</i> untuk menggerakkan solenoid kunci pintu.

Tabel 2.1 (lanjutan)

Pin	Mode	Keterangan
D33	<i>Input</i>	Terhubung dengan saklar magnetik untuk membaca kondisi pintu.
D32	<i>Output</i>	Terhubung dengan <i>buzzer</i> untuk memberikan suara <i>alarm</i> peringatan.
Vin	<i>Power</i>	Terhubung dengan sumber catu daya.

2. Bluetooth HC-05

Modul HC-05 digunakan untuk melakukan komunikasi dengan aplikasi *mobile* menggunakan koneksi *bluetooth*. ESP32 sudah dilengkapi dengan modul *bluetooth*, tetapi pada implementasinya ESP32 tidak bisa menjalankan modul WiFi dan *bluetooth* secara bersamaan dikarenakan keduanya terhubung ke satu modul radio yang sama dan juga keterbatasan *memory* yang dimiliki.



Gambar 2.2 Konfigurasi *bluetooth* HC-05

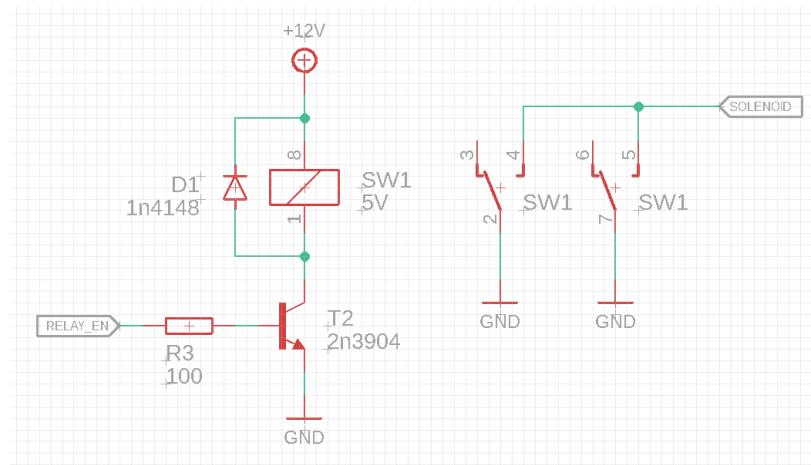
Pada Gambar 2.2 di atas terlihat penggunaan modul *bluetooth* HC-05 pada perangkat kunci pintu. Modul *bluetooth* HC-05 menggunakan komunikasi serial untuk berkomunikasi dengan mikrokontroler dengan *bautrate* yang telah ditentukan. Pada modul *bluetooth* terdapat 4 pin yang digunakan sesuai dengan penjelasan pada Tabel 2.2 di bawah ini.

Tabel 2.2 Konfigurasi pin *bluetooth* HC-05

Pin	Mode	Keterangan
VCC	<i>Power</i>	Terhubung ke sumber catu daya untuk menghidupkan modul <i>bluetooth</i> .
GND	<i>Power</i>	Referensi 0 volt
TX	<i>Output</i>	Terhubung ke mikrokontroler untuk mengirimkan data.
RX	<i>Input</i>	Terhubung ke mikrokontroler untuk menerima data.

3. Solenoid

Untuk melakukan penguncian secara mekanik maka diperlukan sebuah solenoid untuk mengubah arus listrik menjadi gerakan mekanik dengan menggunakan prinsip elektromagnetik.

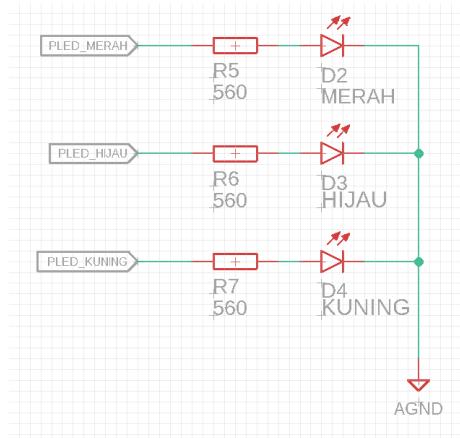


Gambar 2.3 Konfigurasi solenoid

Pada Gambar 2.3 di atas, untuk menggerakkan solenoid maka mikrokontroler memerlukan bantuan dari sebuah modul *relay*. Hal tersebut dikarenakan mikrokontroler hanya mampu mengalirkan arus beberapa mA saja sehingga tidak mampu untuk menggerakkan solenoid secara langsung.

4. LED

LED digunakan untuk memberikan informasi tentang kondisi perangkat kunci pintu.

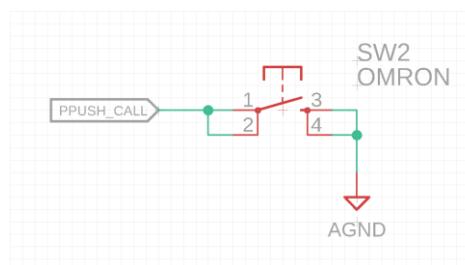


Gambar 2.4 Konfigurasi LED

Pada Gambar 2.4 di atas, perangkat kunci pintu memiliki 3 LED status yaitu LED merah untuk memberikan informasi status penguncian, LED hijau untuk memberikan informasi status pengiriman data, dan LED kuning untuk memberikan status koneksi WiFi.

5. Tombol

Karena solenoid tidak dapat bekerja secara terus-menerus maka solenoid hanya akan aktif jika ada interaksi dari pengguna. Oleh karena itu, kondisi pintu akan selalu terkunci. Untuk membuka pintu, pengguna dapat menggunakan tombol yang disediakan untuk membuka pintu dari dalam ruangan.

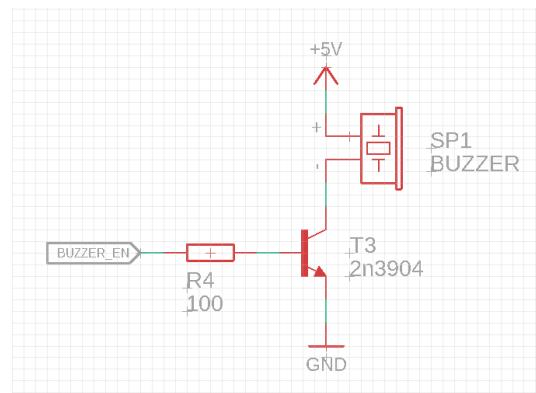


Gambar 2.5 Konfigurasi tombol

Pada Gambar 2.5 di atas terlihat bahwa tombol menggunakan konfigurasi *pull down* dengan memanfaatkan kondisi *pull-up* yang disediakan mikrokontroler. Pada saat ditekan maka tombol akan menghasilkan logika LOW yang dapat dibaca oleh mikrokontroler.

6. Buzzer

Untuk memberikan peringatan suara pada perangkat kunci pintu dapat menggunakan *buzzer*. *Buzzer* akan mengeluarkan suara “beep” pada saat dialiri listrik.



Gambar 2.6 Konfigurasi *buzzer*

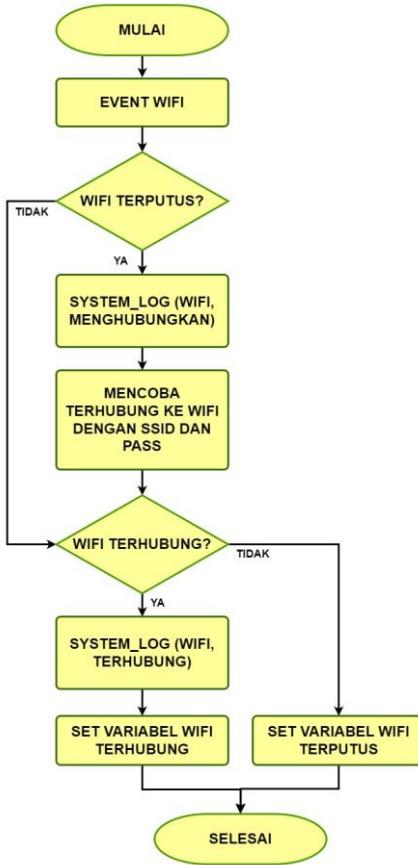
Pada Gambar 2.6 di atas, penggunaan *buzzer* pada perangkat kunci pintu ini dengan menggunakan sebuah transistor untuk membantu mikrokontroler dalam mengaktifkan *buzzer*. Hal tersebut dikarenakan arus yang dikeluarkan mikrokontroler tidak cukup untuk menyalaikan *buzzer* secara maksimal.

2.2 Perangkat Lunak Kunci Pintu

Implementasi perangkat lunak pada perangkat kunci pintu ini dilakukan dengan menggunakan Arduino. Arduino merupakan sebuah lingkungan pengembangan program mikrokontroler dengan menggunakan bahasa C++ yang sudah dipermudah. Beberapa metode yang terdapat pada perangkat kunci pintu ini adalah sebagai berikut:

1. Penanganan koneksi WiFi

Perangkat kunci pintu memerlukan koneksi ke internet. Koneksi tersebut dilakukan melalui modul WiFi yang telah disediakan oleh ESP32. Setiap terjadi perubahan kondisi WiFi, perangkat penguncian mampu beradaptasi seperti melakukan koneksi ulang ke jaringan WiFi saat terjadi masalah.

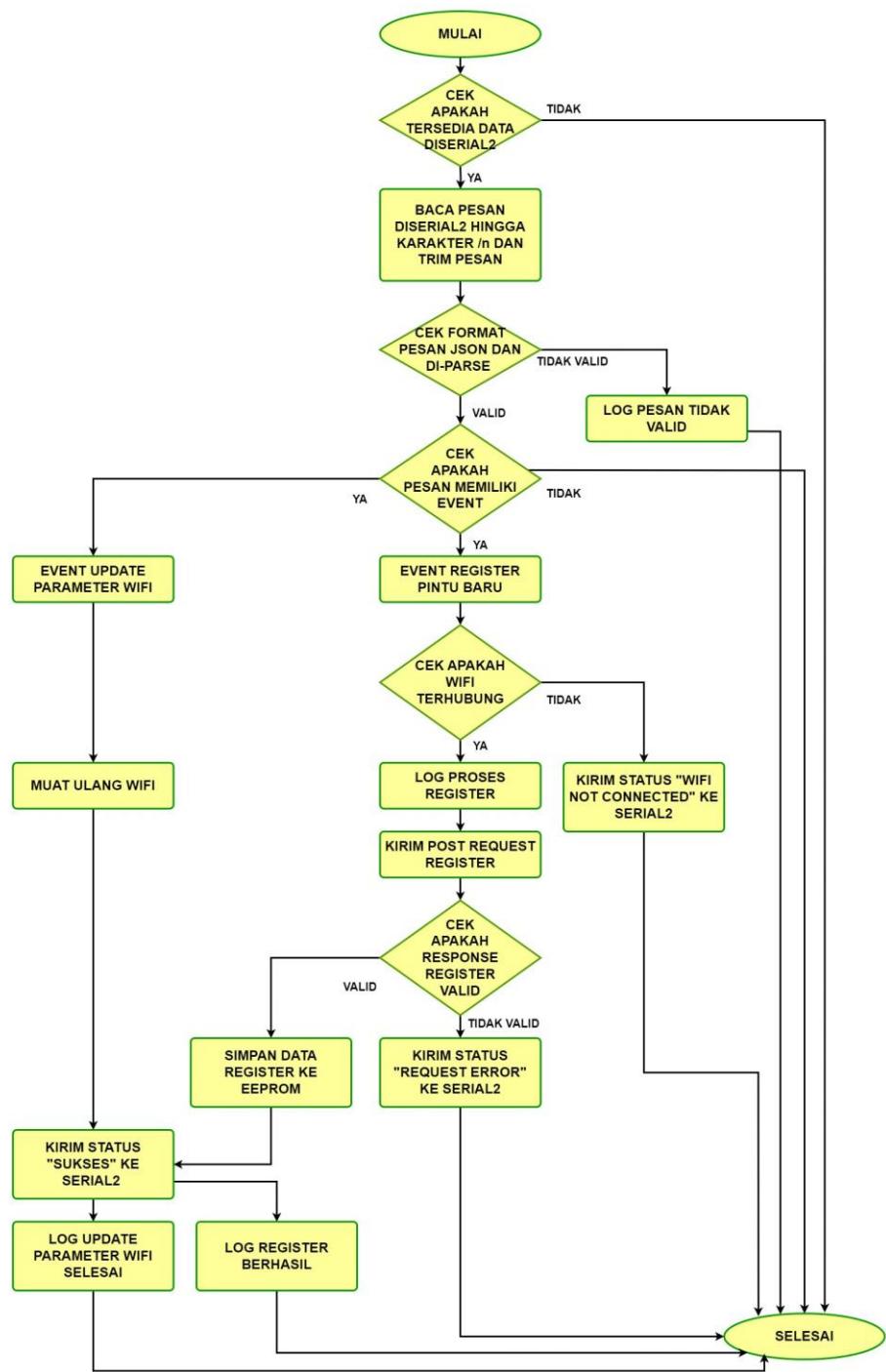


Gambar 2.7 Flowchart koneksi WiFi

Gambar 2.7 di atas memperlihatkan cara kerja program untuk menangani kondisi WiFi. Kondisi WiFi akan dipantau menggunakan sebuah *event*. Setiap terjadi perubahan status maka *event* tersebut juga akan berubah dan memperbarui status koneksi WiFi menggunakan sebuah variabel. Variabel tersebut digunakan untuk menyimpan kondisi WiFi sehingga diketahui secara global. Hal tersebut untuk memastikan bahwa WiFi terhubung sebelum berkomunikasi dengan *server*. Pada saat kondisi WiFi terputus maka variabel status akan diperbarui dan program akan mencoba menghubungkan ulang. Pada saat kondisi WiFi baru saja terhubung maka status koneksi juga akan diperbarui.

2. Pengaturan perangkat kunci pintu

Proses pengaturan kunci pintu seperti mengatur kredensial WiFi serta mendaftarkan perangkat baru ke *server* dilakukan dengan menggunakan aplikasi *mobile* melalui koneksi *bluetooth*.



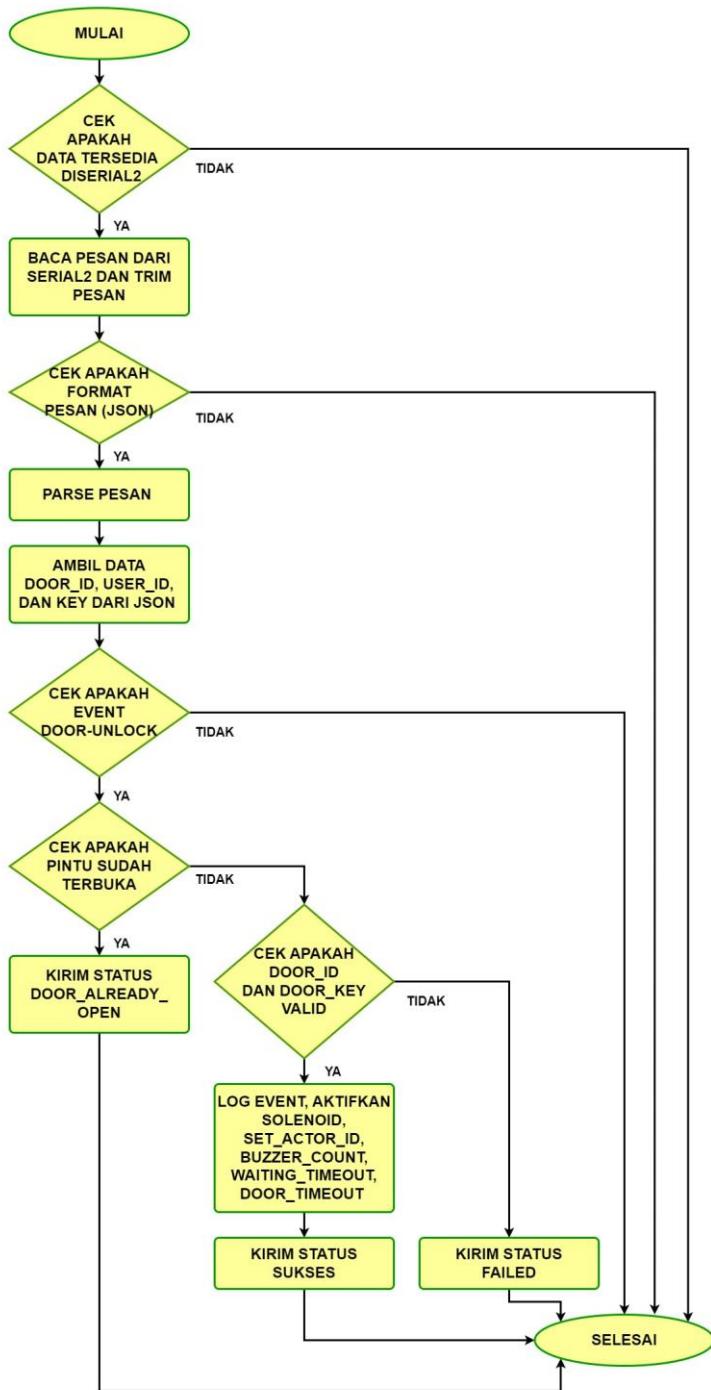
Gambar 2.8 Flowchart fungsi `config_loop()`

Berdasarkan pada Gambar 2.8 di atas, fungsi `config_loop(void)` akan memeriksa apakah ada data yang tersedia di antarmuka `Serial2` yang terhubung ke modul *bluetooth* HC-05. Data yang diterima di `Serial2` dibaca hingga menemukan karakter *newline* ('\n') dan disimpan dalam variabel `bt_message`. Pesan yang

telah diterima kemudian dipangkas untuk menghilangkan karakter yang tidak diinginkan menggunakan fungsi `trim()`. Selanjutnya, kode ini memeriksa apakah pesan yang diterima merupakan objek JSON yang valid dengan memeriksa apakah pesan dimulai dengan '{' dan diakhiri dengan '}' menggunakan `bt_message.startsWith("{")` dan `bt_message.endsWith("}")`. Jika pesan tersebut adalah objek JSON yang valid, selanjutnya dilakukan *parsing* menggunakan fungsi `JSON.parse(bt_message)`. Selanjutnya, fungsi `config_loop(void)` memeriksa properti "event" dari objek JSON yang di-*parsing* untuk mengidentifikasi jenis pembaruan konfigurasi yang diminta menggunakan `bt_package.hasOwnProperty("event")`. Jika properti "event" ada, maka kode akan memeriksa nilai dari properti "event" untuk menentukan tindakan selanjutnya. Jika nilai properti *event* adalah "wifi-update", maka fungsi `config_loop(void)` melakukan *update* parameter WiFi berupa data "ssid" dan "password" dari objek JSON. Selain itu, fungsi ini mengirim status keberhasilan kembali ke aplikasi *mobile* melalui koneksi *bluetooth*. Jika *event* adalah "register", pertama fungsi ini memeriksa apakah WiFi terhubung. Jika tidak, fungsi ini merespons dengan status yang menunjukkan bahwa WiFi tidak terhubung. Jika WiFi terhubung, fungsi ini melanjutkan proses pendaftaran pintu baru. Proses pendaftaran pintu melibatkan pengiriman permintaan POST ke sebuah *endpoint server* ("register") dengan data "door_id" dan "device_name" melalui HTTP. Selanjutnya, tanggapan dari *server* di-*parsing*, dan jika pendaftaran berhasil, data relevan (office ID, door ID, nama pintu, dll) disimpan di dalam EEPROM. Selain itu, status keberhasilan atau kegagalan dikirim kembali ke aplikasi *mobile* melalui koneksi *bluetooth*.

3. Perintah penguncian menggunakan *bluetooth*

Untuk membuka kunci pintu, setiap pengguna akan menggunakan aplikasi *mobile* untuk berkomunikasi dengan perangkat kunci pintu. Pada kondisi tersebut, proses verifikasi pengguna dilakukan dengan memeriksa setiap pesan yang dikirimkan oleh pengguna. Untuk membuka kunci pintu maka pengguna harus mengirimkan pesan sesuai dengan format perintah yang diketahui oleh perangkat kunci pintu. Program yang digunakan untuk melakukan verifikasi tersebut dapat dilihat pada Gambar 2.9 di bawah ini.



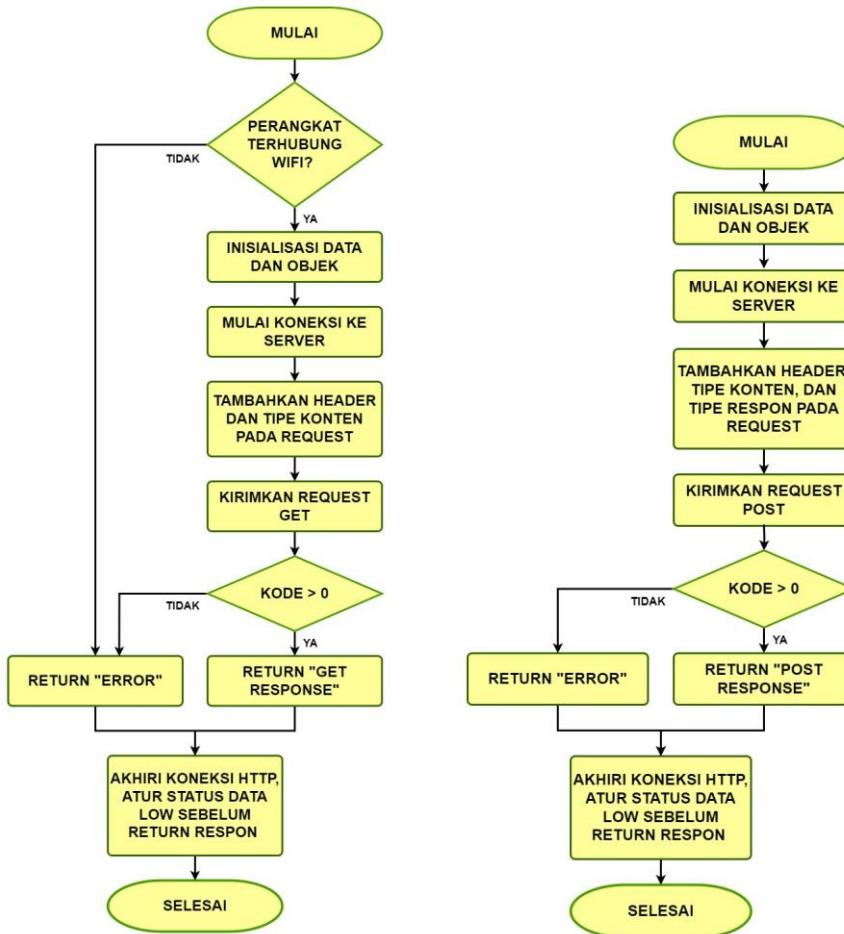
Gambar 2.9 Flowchart fungsi `bt_door_command()`

Berdasarkan pada Gambar 2.9 di atas, fungsi `bt_door_command(void)` memeriksa apakah ada data yang tersedia di antarmuka `Serial2` (*bluetooth*). Data yang diterima dibaca hingga menemukan karakter *newline* ('`\n`') dan disimpan dalam variabel `bt_message`. Pesan yang telah diterima kemudian dipangkas untuk

menghilangkan karakter yang tidak diinginkan menggunakan fungsi `trim()`. Fungsi `bt_door_command(void)` juga memverifikasi apakah pesan yang diterima memiliki format JSON yang valid (dimulai dengan "{" dan diakhiri dengan "}") menggunakan `bt_message.startsWith("{")` dan `bt_message.endsWith("}")`. Jika pesan tersebut adalah objek JSON yang valid, selanjutnya dilakukan *parsing* menggunakan fungsi `JSON.parse(bt_message)`. Kemudian, fungsi `bt_door_command(void)` mengekstrak data "door_id," "user_id," dan "key" dari objek JSON. Jika *event* dalam objek JSON yang di-*parsing* adalah "door-unlock," fungsi `bt_door_command(void)` memeriksa apakah pintu sudah terbuka. Jika sudah, fungsi `bt_door_command(void)` merespons dengan status yang menunjukkan bahwa pintu sudah terbuka. Jika belum, fungsi `bt_door_command(void)` memeriksa apakah "door_id" dan "key" yang diterima sesuai dengan nilai yang disimpan di dalam EEPROM. Jika sesuai, fungsi `bt_door_command(void)` memulai proses membuka kunci pintu. Proses membuka kunci pintu melibatkan mengaktifkan solenoid dan melakukan beberapa tindakan terkait lainnya.

4. Komunikasi dengan *server*

Perangkat kunci pintu akan mengirimkan informasi terkait dengan kondisi pintu dan melakukan autentikasi untuk terhubung ke sistem penguncian. Implementasi komunikasinya menggunakan protokol HTTPS dengan proses enkripsi SSL sehingga setiap data yang dikirimkan akan dienkripsi dengan aman. Protokol HTTPS yang didukung meliputi permintaan GET dan POST sesuai dengan standar komunikasi HTTP.



Gambar 2.10 Flowchart fungsi HTTP GET dan POST

Berdasarkan Gambar 2.10, *flowchart* di atas memperlihatkan diagram alir pemrograman fungsi `post_request()` dan `get_request()` yang berfungsi untuk melakukan permintaan HTTP ke *server*.

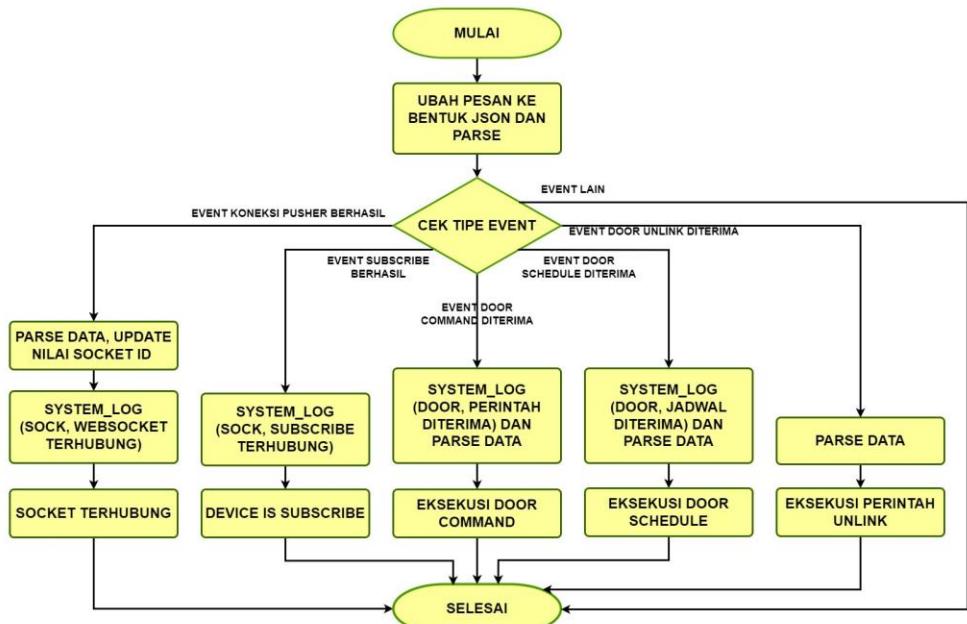
Fungsi `get_request(String endpoint)` digunakan untuk melakukan permintaan HTTP GET ke *endpoint* yang ditentukan pada *server*. Fungsi tersebut memeriksa apakah koneksi WiFi tersedia (`wifi_is_connected`). Jika tidak tersedia, maka fungsi tersebut akan mengembalikan string "error". Fungsi tersebut juga melakukan inisialisasi objek *HTTPClient* (`http_get`) dan variabel *String* (`get_response`) untuk menyimpan respons dari *server*. Fungsi tersebut memulai koneksi dengan menetapkan pin `data_status` ke nilai HIGH dan memulai permintaan HTTP menggunakan `http_get.begin(...)`. Fungsi tersebut menambahkan *header* yang diperlukan untuk permintaan, seperti token otorisasi (`Bearer token`) dan format respons yang diharapkan (`Accept: application/json`). Permintaan GET

dikirim menggunakan `http_get.GET()` yang mengembalikan kode respon HTTP. Jika kode respon lebih besar dari 0, menandakan permintaan berhasil, fungsi tersebut membaca isi respon dari *server* ke dalam variabel `get_response` menggunakan `http_get.getString()`. Namun, jika permintaan gagal (kode respon ≤ 0), variabel `get_response` diatur menjadi "error". Terakhir, koneksi HTTP ditutup dan pin `data_status` diatur kembali ke nilai LOW. Fungsi tersebut mengembalikan string `get_response` yang berisi respon dari *server* atau pesan "error".

Fungsi `post_request(String endpoint, String payload)` digunakan untuk melakukan permintaan HTTP POST ke *endpoint* yang ditentukan pada *server*, dengan mengirimkan data dalam *payload*. Fungsi tersebut mengikuti proses yang mirip dengan fungsi `get_request`. Namun, dengan modifikasi yang diperlukan untuk permintaan POST.

5. Penanganan perintah dari *server*

Server akan mengirimkan perintah ke perangkat penguncian menggunakan sebuah *event websocket*. Oleh karena itu, perangkat kunci pintu harus dapat membaca setiap *event* yang dikirimkan oleh *server*.



Gambar 2.11 Flowchart fungsi `websocket_message()`

Berdasarkan Gambar 2.11 di atas, fungsi `void websocket_message()` dipanggil ketika sebuah pesan *WebSocket* diterima. Di dalam fungsi `websocket_message`, kode pertama kali mengurai pesan yang diterima ke dalam format JSON menggunakan `JSON.parse()`. Objek `JSONVar message` digunakan untuk menyimpan data pesan yang telah di-parse. Selanjutnya, fungsi `websocket_message` mengecek nilai `field "event"` dari pesan JSON yang telah di-parse untuk menentukan tipe `event` yang diterima. Jika `event` adalah `"pusher:connection_established"`, berarti koneksi *WebSocket* telah berhasil dibuat. Kemudian, fungsi `websocket_message` mengurai `field "data"` dari pesan yang merupakan *string* format JSON lainnya menggunakan `JSON.parse()`, mengambil nilai `field "socket_id"` dari `pusher_data` yang telah di-parse, memperbarui nilai variabel `socket_id` dengan `socket_id` yang telah diambil, memanggil fungsi `system_log` dengan parameter `"SOCK"` dan `"websocket terhubung"` yang menandakan `"WebSocket connected"`, dan mengeset variabel `socket_is_connected` menjadi `true` yang menandakan bahwa koneksi *WebSocket* sekarang terhubung.

Jika `event` adalah `"pusher_internal:subscription_succeeded"`, berarti `subscribe`/langganan ke saluran atau topik telah berhasil dibuat, maka fungsi `websocket_message` memanggil fungsi `system_log` dengan parameter `"SOCK"` dan `"subscribe berhasil"` yang menandakan `"subscription succeeded"` dan mengeset variabel `device_is_subscribe` menjadi `true` yang menandakan bahwa perangkat telah berhasil berlangganan/`subscribe` ke saluran.

Jika `event` adalah `"door-command"`, berarti pesan `"door-command"` telah diterima. Fungsi `websocket_message` memanggil fungsi `system_log` dengan parameter `"DOOR"` dan `"perintah diterima"` yang menandakan `"command received"`, mengurai `field "data"` dari pesan yang merupakan *string* format JSON lainnya menggunakan `JSON.parse()`, dan memanggil fungsi `door_command` dengan data `command` yang telah di-parse sebagai argumen.

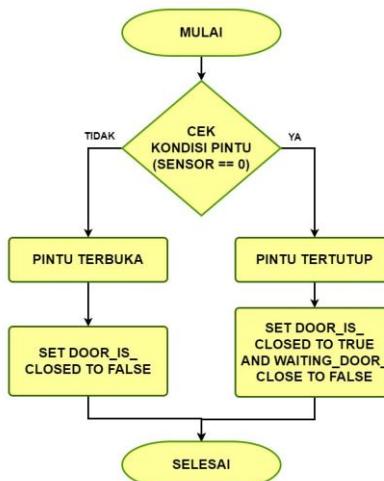
Jika `event` adalah `"door-schedule"`, itu berarti pesan `"door-schedule"` telah diterima. Fungsi `websocket_message` memanggil fungsi `system_log` dengan parameter `"DOOR"` dan `"jadwal diterima"` yang menandakan `"schedule received"`, mengurai/mem-parse `field "data"` dari pesan yang merupakan *string* format JSON

lainnya menggunakan `JSON.parse()`, memanggil fungsi `door_schedule` dengan data `schedule` yang telah di-parse sebagai argumen.

Jika `event` adalah "door-unlink", berarti pesan "door-unlink" telah diterima. Fungsi `websocket_message` mengurai/ mem-parse field "data" dari pesan yang merupakan `string` format JSON lainnya menggunakan `JSON.parse()` dan memanggil fungsi `unlink` dengan data `unlink_data` yang telah di-parse sebagai argumen.

6. Pembacaan kondisi pintu

Kondisi pintu dibaca dengan menggunakan saklar magnetik, saklar magnetik akan menggunakan medan magnet untuk menentukan kondisi pintu sedang terbuka atau tertutup. Setiap terjadi perubahan maka perangkat kunci pintu akan memperbarui status pintu. Cara kerja dari program kondisi pintu dapat dilihat pada Gambar 2.12 di bawah ini.



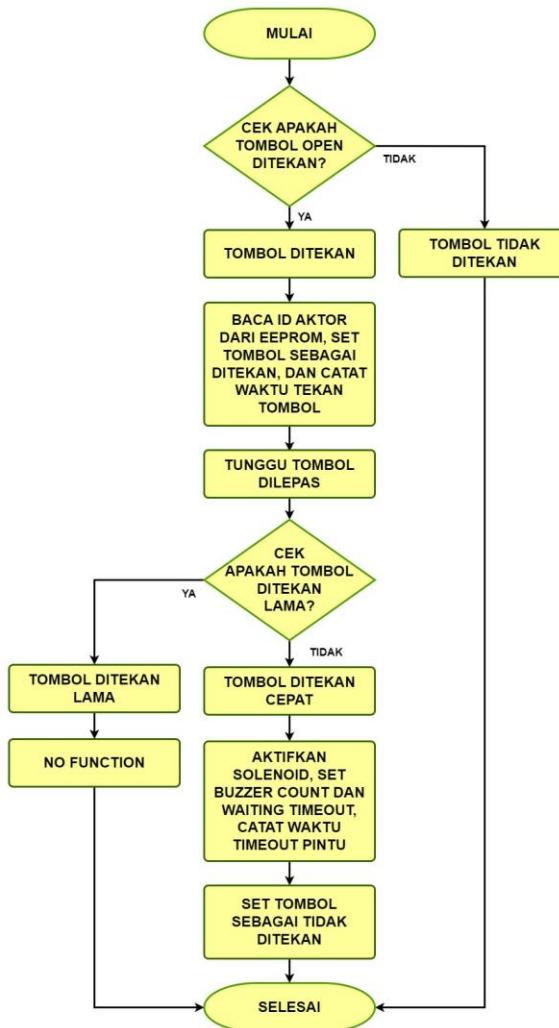
Gambar 2.12 Flowchart pembacaan kondisi pintu

Berdasarkan pada Gambar 2.12 di atas, program dimulai dengan pernyataan pengkondisian menggunakan `if`. Dengan fungsi `digitalRead(sensor)` akan membaca nilai digital dari input "sensor". Nilai yang dikembalikan oleh fungsi tersebut berupa **0** jika pintu dalam keadaan tertutup dan **1** jika pintu dalam keadaan terbuka. Jika hasil bacaan dari sensor adalah **0** (pintu dalam keadaan tertutup), maka blok kode dieksekusi dengan menyetel variabel `door_is_closed` ke `true` untuk menandakan bahwa pintu dalam keadaan tertutup dan variabel

`waiting_door_close` ke `false` untuk digunakan di bagian kode lain yang terkait dengan penanganan pintu dalam keadaan tertutup. Jika hasil bacaan dari sensor bukan 0 (pintu dalam keadaan terbuka), maka blok kode dieksekusi dengan menyetel variabel `door_is_closed` ke `false` yang menandakan bahwa pintu dalam keadaan terbuka.

7. Penanganan interaksi pengguna

Pengguna dapat berinteraksi untuk membuka pintu menggunakan berbagai metode. Metode yang mungkin digunakan oleh pengguna yaitu dengan menggunakan tombol yang telah disediakan atau menggunakan aplikasi *mobile* untuk membuka kunci pintu. Proses penanganan interaksi pengguna menggunakan tombol dapat dilihat pada Gambar 2.13 di bawah ini.



Gambar 2.13 Flowchart penanganan interaksi pengguna menggunakan tombol

Berdasarkan pada Gambar 2.13 di atas, pertama program memeriksa apakah tombol "open" ditekan menggunakan kode (`digitalRead(button) == LOW`) dengan beberapa kondisi harus terpenuhi yaitu `button_is_pressed` bernilai `false`, `waiting_door_close` bernilai `false`, dan `door_is_closed` bernilai `true`. Jika semua kondisi tersebut terpenuhi, kode akan melanjutkan untuk menjalankan tindakan-tindakan untuk membaca nilai `door_id_addr` dari EEPROM dan menyimpannya dalam variabel `actor_id`, menetapkan `button_is_pressed` menjadi `true`, merekam waktu saat tombol ditekan menggunakan fungsi `millis()` dan menyimpannya dalam variabel `button_pressed_time`. Selanjutnya, memeriksa apakah tombol "open" dilepaskan dengan menggunakan kode (`digitalRead(button) == HIGH`) dan jika `button_is_pressed` bernilai `true`. Jika ya, kode akan memeriksa berapa lama tombol ditekan. Jika tombol ditekan lebih dari 700 milidetik (`(millis() - button_pressed_time) > 700`), maka tidak ada tindakan yang dilakukan (`// no function comment`) yang berarti bahwa jika tombol ditekan dalam waktu yang lebih lama, beberapa tindakan khusus dapat ditambahkan, tetapi dalam kode tersebut tidak ada tindakan khusus yang dijalankan untuk kasus tersebut. Jika tombol dilepaskan dengan cepat (ditekan selama kurang dari atau sama dengan 700 milidetik), tindakan-tindakan berikut dilakukan yaitu menetapkan solenoid menjadi aktif, menetapkan `buzzer_count` menjadi 2 (yang digunakan untuk mengatur bunyi *buzzer*), menetapkan `waiting_timeout` menjadi `true`, dan merekam waktu saat ini menggunakan `millis()`, serta menyimpannya dalam variabel `door_timeout`. Terakhir, kode menetapkan `button_is_pressed` menjadi `false` yang berarti kode siap untuk mendeteksi tekanan tombol kembali.

2.3 Database

Implementasi dilakukan dengan membuat sebuah *database* menggunakan MySQL sesuai dengan diagram yang telah ditentukan. *Database* akan berisi tabel tambahan yang dibutuhkan oleh kerangka kerja laravel. Pembuatan skema *database* dilakukan dengan menggunakan mekanisme migrasi yang disediakan oleh laravel sehingga struktur dari *database* dapat dengan mudah diubah dengan cara mengubah kode migrasi. Struktur dari setiap tabel dapat dilihat pada Tabel 2.3 sampai Tabel 2.9 di bawah ini.

Tabel 2.3 Struktur tabel *users*

No	Nama Kolom	Tipe Data	Keterangan
1	<i>Id</i>	<i>char(36)</i>	kunci primer tabel
2	<i>added_by</i>	<i>char(36)</i>	referensi <i>actor</i>
3	<i>Email</i>	<i>varchar(255)</i>	alamat email pengguna
4	<i>Password</i>	<i>varchar(255)</i>	<i>password</i> pengguna
5	<i>remember_token</i>	<i>varchar(255)</i>	token untuk fitur "ingat saya"
6	<i>Name</i>	<i>varchar(255)</i>	nama pengguna
7	<i>Phone</i>	<i>varchar(255)</i>	nomor hp pengguna
8	<i>Gender</i>	<i>Enum</i>	jenis kelamin pengguna
9	<i>Role</i>	<i>Enum</i>	jabatan pengguna
10	<i>Foto</i>	<i>varchar(255)</i>	foto profil pengguna
11	<i>email_verified_at</i>	<i>Timestamp</i>	waktu verifikasi email
12	<i>created_at</i>	<i>Timestamp</i>	waktu data dibuat
13	<i>updated_at</i>	<i>Timestamp</i>	waktu data diedit

Tabel 2.4 Struktur tabel *offices*

No	Nama Kolom	Tipe Data	Keterangan
1	<i>Id</i>	<i>char(36)</i>	kunci primer tabel
2	<i>user_id</i>	<i>char(36)</i>	referensi operator gedung
3	<i>Name</i>	<i>varchar(255)</i>	nama gedung
4	<i>created_at</i>	<i>Timestamp</i>	waktu data dibuat
5	<i>updated_at</i>	<i>Timestamp</i>	waktu data diedit

Tabel 2.5 Struktur tabel *doors*

No	Nama Kolom	Tipe Data	Keterangan
1	<i>Id</i>	<i>char(36)</i>	kunci primer tabel
2	<i>office_id</i>	<i>char(36)</i>	referensi gedung
3	<i>socket_id</i>	<i>char(36)</i>	identitas <i>websocket</i>
4	<i>Name</i>	<i>varchar(36)</i>	nama pintu
5	<i>device_name</i>	<i>varchar(36)</i>	<i>username</i> pintu
6	<i>device_pass</i>	<i>varchar(36)</i>	<i>password</i> pintu
7	<i>Key</i>	<i>varchar(36)</i>	kunci pintu
8	<i>is_lock</i>	<i>tinyint(1)</i>	status penguncian
9	<i>created_at</i>	<i>Timestamp</i>	waktu data dibuat
10	<i>updated_at</i>	<i>Timestamp</i>	waktu data diedit

Tabel 2.6 Struktur tabel *access*

No	Nama Kolom	Tipe Data	Keterangan
1	<i>Id</i>	<i>char(36)</i>	kunci primer tabel
2	<i>user_id</i>	<i>char(36)</i>	referensi ke pengguna
3	<i>door_id</i>	<i>char(36)</i>	referensi ke pintu
4	<i>time_begin</i>	<i>Time</i>	waktu mulai
5	<i>time_end</i>	<i>Time</i>	waktu berakhir
6	<i>date_begin</i>	<i>Date</i>	tanggal mulai
7	<i>date_end</i>	<i>Date</i>	tanggal berakhir
8	<i>is_temporary</i>	<i>tinyint(1)</i>	status akses sementara
9	<i>is_running</i>	<i>tinyint(1)</i>	status akses sedang berjalan
10	<i>created_at</i>	<i>Timestamp</i>	waktu data dibuat

Tabel 2.6 (lanjutan)

No	Nama Kolom	Tipe Data	Keterangan
11	<i>updated_at</i>	<i>Timestamp</i>	waktu data diedit

Tabel 2.7 Struktur tabel *schedules*

No	Nama Kolom	Tipe Data	Keterangan
1	<i>Id</i>	<i>char(36)</i>	kunci primer tabel
2	<i>office_id</i>	<i>char(36)</i>	referensi gedung
3	<i>Name</i>	<i>varchar(255)</i>	nama jadwal
4	<i>date_begin</i>	<i>Date</i>	tanggal mulai
5	<i>date_end</i>	<i>Date</i>	tanggal berakhir
6	<i>time_begin</i>	<i>Time</i>	waktu mulai
7	<i>time_end</i>	<i>Time</i>	waktu berakhir
8	<i>is_repeating</i>	<i>tinyint(1)</i>	status perulangan
9	<i>day_repeating</i>	<i>varchar(255)</i>	perulangan hari
10	<i>Status</i>	<i>Enum</i>	status pelaksanaan jadwal
11	<i>created_at</i>	<i>Timestamp</i>	waktu data dibuat
12	<i>updated_at</i>	<i>Timestamp</i>	waktu data diedit

Tabel 2.8 Struktur Tabel *Door Schedule*

No	Nama Kolom	Tipe Data	Keterangan
1	<i>Id</i>	<i>char(36)</i>	kunci primer tabel
2	<i>schedule_id</i>	<i>char(36)</i>	referensi ke jadwal
3	<i>door_id</i>	<i>char(36)</i>	referensi ke pintu
4	<i>created_at</i>	<i>Timestamp</i>	waktu data dibuat
5	<i>updated_at</i>	<i>Timestamp</i>	waktu data diedit

Tabel 2.9 Struktur tabel riwayat akses

No	Nama Kolom	Tipe Data	Keterangan
1	<i>Id</i>	<i>char(36)</i>	kunci primer tabel
2	<i>user_id</i>	<i>char(36)</i>	referensi ke pengguna
3	<i>office_id</i>	<i>char(36)</i>	referensi ke gedung
4	<i>door_id</i>	<i>char(36)</i>	referensi ke pintu
5	<i>Log</i>	<i>Text</i>	pesan log
6	<i>created_at</i>	<i>Timestamp</i>	waktu data dibuat
7	<i>updated_at</i>	<i>Timestamp</i>	waktu data diedit

Dengan menggunakan *relational database* tentunya akan terdapat hubungan atau relasi antara dua tabel atau lebih. Relasi memberikan informasi mengenai hubungan antara dua tabel atau lebih yang saling berkaitan beserta dengan perilaku terhadap perubahan data pada tabel induknya. Relasi pada setiap tabel dapat dilihat pada Tabel 2.10 di bawah ini.

Tabel 2.10 Relasi tabel

No	Tabel Induk	Anak		Jenis Hubungan	On Update	On Delete
		Tabel	Kolom			
1	<i>Users</i>	<i>Users</i>	<i>added_by</i>	<i>one to many</i>	<i>cascade</i>	<i>restrict</i>
2	<i>Users</i>	<i>Access</i>	<i>user_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>
3	<i>Doors</i>	<i>Access</i>	<i>door_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>
4	<i>Users</i>	<i>access_logs</i>	<i>user_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>
5	<i>Doors</i>	<i>access_logs</i>	<i>door_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>
6	<i>Offices</i>	<i>access_logs</i>	<i>office_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>
7	<i>Offices</i>	<i>Doors</i>	<i>office_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>
8	<i>Schedules</i>	<i>door_schedule</i>	<i>schedule_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>

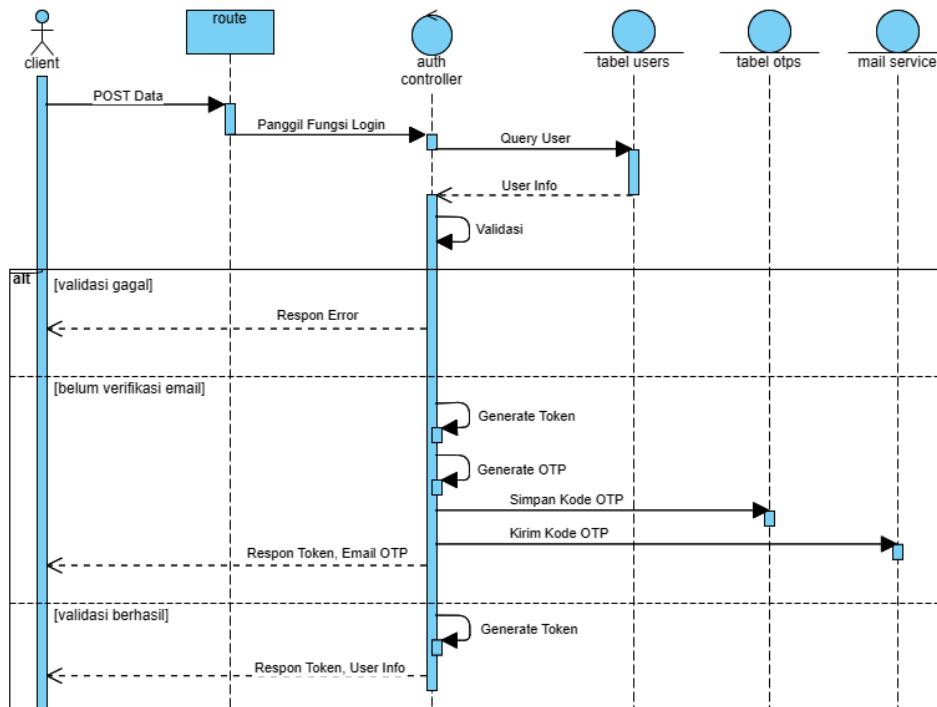
Tabel 2.10 (lanjutan)

No	Tabel Induk	Anak		Jenis Hubungan	On Update	On Delete
		Tabel	Kolom			
9	<i>Doors</i>	<i>door_schedule</i>	<i>door_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>
10	<i>Users</i>	<i>Offices</i>	<i>user_id</i>	<i>one to one</i>	<i>cascade</i>	<i>restrict</i>
11	<i>Users</i>	<i>Otps</i>	<i>user_id</i>	<i>one to one</i>	<i>cascade</i>	<i>cascade</i>
12	<i>Offices</i>	<i>Schedules</i>	<i>office_id</i>	<i>one to many</i>	<i>cascade</i>	<i>cascade</i>

2.4 Backend API

Implementasi *backend API* dilakukan menggunakan kerangka kerja laravel dengan bahasa pemrograman PHP. Beberapa metode API yang diimplementasikan untuk mendukung kinerja dari sistem keamanan kunci pintu ini adalah sebagai berikut:

1. Login

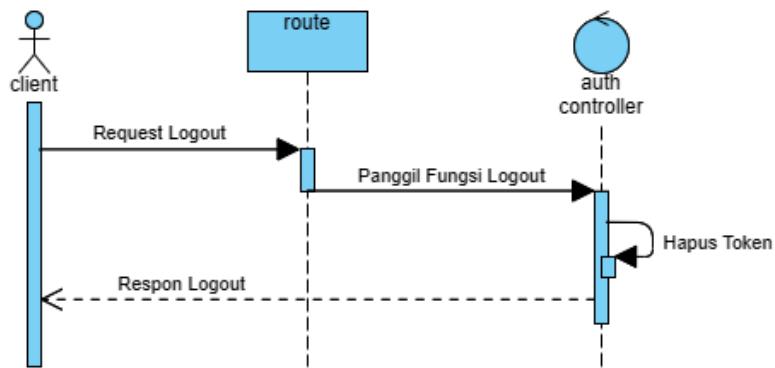


Gambar 2.14 Sequence diagram API login

Pada Gambar 2.14 di atas API *login* dimulai oleh *client*, *client* akan mengirimkan *username* dan *password* melalui *endpoint* “/api/login”. Kemudian, *route*

memanggil fungsi *login* di dalam kontroler. Kontroler akan memeriksa *username* dan *password* dengan melakukan *query* ke tabel *users*. Jika cocok maka kontroler akan membuat token menggunakan modul *sanctum*. *Sanctum* adalah sebuah paket autentikasi dan autorisasi yang disediakan oleh laravel yang dirancang untuk memudahkan implementasi autentikasi API yang sederhana tetapi aman pada aplikasi laravel. Setelah mendapatkan token, kontroler akan mengembalikan token tersebut disertai dengan data *client* seperti nama, email, nomor hp, dan lain sebagainya. Jika *client* terdeteksi belum melakukan verifikasi email maka kontroler membuat kode OTP atau *One Time Password* berupa 6 digit angka acak dan mengirimkan kode tersebut ke email *client*. Jika autentikasi yang dilakukan gagal maka kontroler akan mengembalikan respons *error* ke *client*.

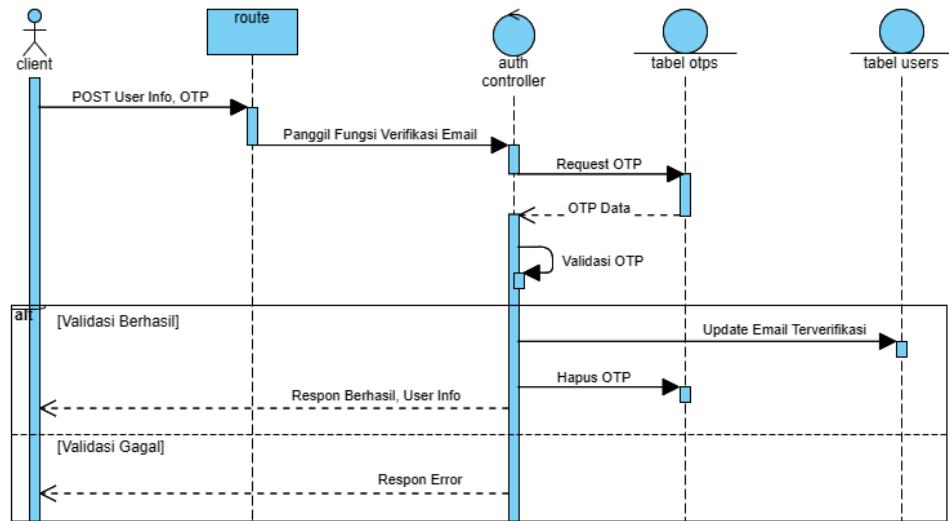
2. Logout



Gambar 2.15 Sequence diagram API *logout*

Pada Gambar 2.15 di atas, sebuah metode *logout* di dalam kontroler akan dipanggil oleh *route* jika ada *client* yang melakukan *request* ke *endpoint* “*/api/logout*”. Kemudian, kontroler akan menghapus token dari *client* sesuai dengan token yang dilampirkan di dalam *header* pada saat *request* diterima. Proses tersebut sangat penting untuk menjaga keamanan sistem dan melindungi privasi pengguna. Dengan menghapus token saat klien keluar atau *logout*, sistem dapat memastikan bahwa akses yang tidak sah, tidak dapat digunakan untuk mengakses pintu.

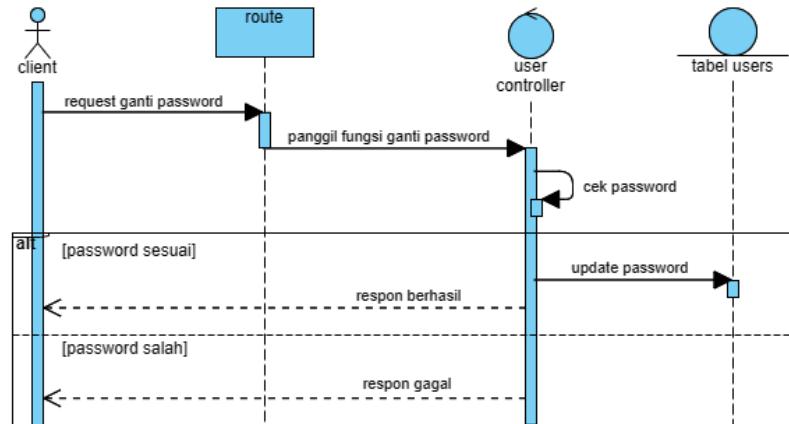
3. Verifikasi email



Gambar 2.16 Sequence diagram API verifikasi email

Pada Gambar 2.16 di atas dapat dilihat bahwa pengguna melakukan verifikasi email dengan mengirimkan kode OTP yang sudah diterima via email disertai dengan detail *client* seperti id, nama, dan email ke *endpoint* “*/api/verify-email*”. Kemudian, kontroler akan memeriksa kode yang diterima dengan kode yang tersimpan pada tabel otps. Jika cocok dan masih aktif maka kontroler akan memperbarui status *client* menjadi terverifikasi, menghapus kode otp yang lama, dan mengembalikan respons berhasil. Jika kode salah atau sudah kedaluwarsa maka kontroler akan mengembalikan respons *error*.

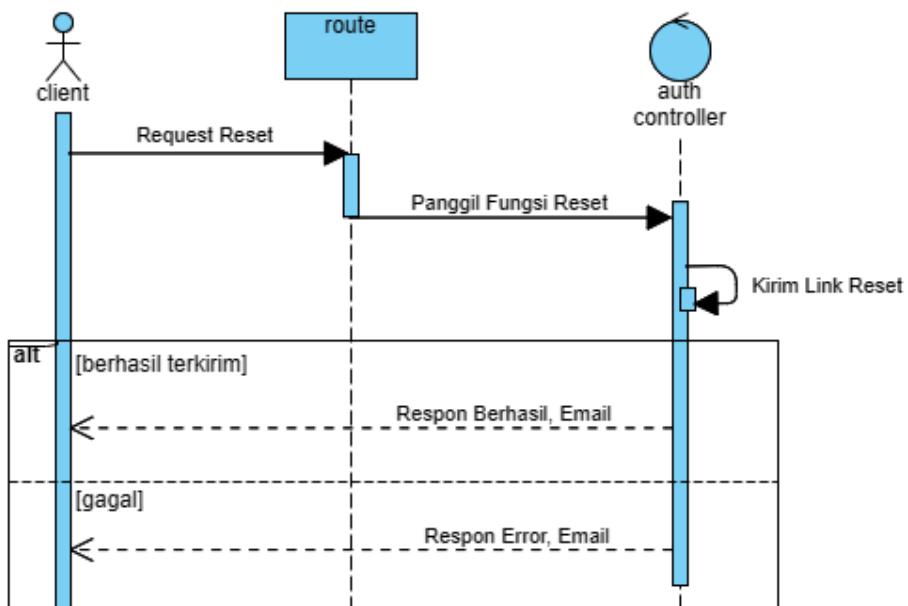
4. Ganti password



Gambar 2. 17 Sequence diagram API ganti password

Dapat dilihat pada Gambar 2.17 di atas, untuk mengganti *password* langkah pertama *client* adalah mengirimkan permintaan ganti *password* ke *endpoint* “*/api/change-password*” dengan mengirimkan *password* lama, *password* baru, dan konfirmasi *password* baru. Kemudian, di dalam kontroler *password* lama yang dikirimkan akan dicocokkan dengan *password* *client* sekarang dengan menggunakan fungsi *Hash*. Fungsi *Hash* merupakan sebuah fungsi yang disediakan oleh laravel yang digunakan untuk pengolahan data yang berkaitan dengan enkripsi. Jika kedua *password* cocok maka kontroler akan memperbarui *password* pada tabel *users* dan mengembalikan respons berhasil. Jika *password* tidak sesuai maka kontroler akan mengembalikan respons *error*.

5. Reset password

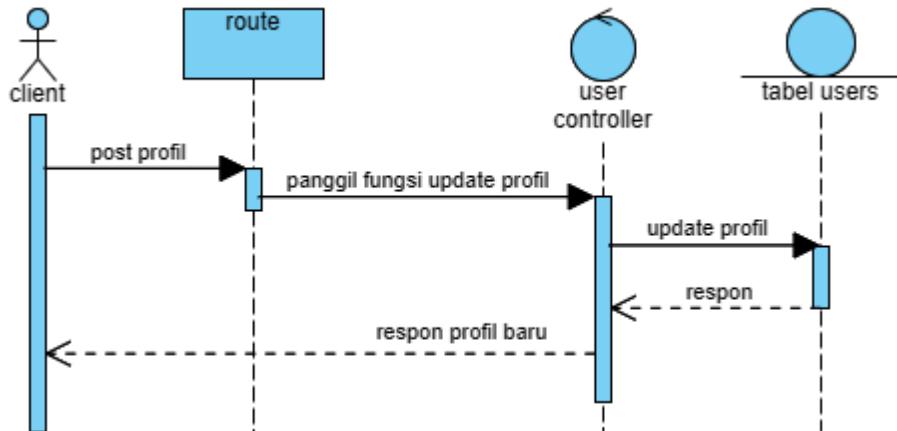


Gambar 2.18 Sequence diagram API reset password

Pada Gambar 2.18 di atas, proses *reset password* menggunakan metode yang terstruktur dan efisien. *Client* memulai proses ini dengan melakukan permintaan (*request*) ke *endpoint* “*/api/reset-password*” pada *server*. Permintaan tersebut berisi data email yang sudah terdaftar sebagai parameter supaya *server* dapat mengidentifikasi akun yang akan di-*reset password*-nya. Setelah permintaan

tersebut diterima maka kontroler pada sisi *server* akan mengeksekusi sebuah fungsi khusus yang telah disediakan oleh laravel. Fungsi tersebut bertugas untuk mengirimkan *link reset password* ke alamat email yang diberikan oleh *client*. Proses pengiriman email tersebut menggunakan layanan email eksternal seperti SMTP yang telah dikonfigurasi pada *server*. Jika email berhasil terkirim dengan sukses maka kontroler akan memberikan respons ke *client* berupa pesan berhasil. Pesan tersebut memberitahukan bahwa email *reset password* telah berhasil dikirim dan pengguna dapat segera memeriksa kotak masuk email-nya untuk melanjutkan proses selanjutnya. Namun, dalam beberapa situasi, pengiriman email mungkin mengalami kegagalan. Misalnya, alamat email yang diberikan tidak valid, terjadi gangguan jaringan, atau layanan email eksternal mengalami masalah. Jika hal tersebut terjadi, kontroler akan memberikan respons *error* ke *client*. Pesan *error* tersebut berisi informasi yang relevan tentang masalah yang terjadi, sehingga pengguna atau operator dapat mengetahui penyebabnya.

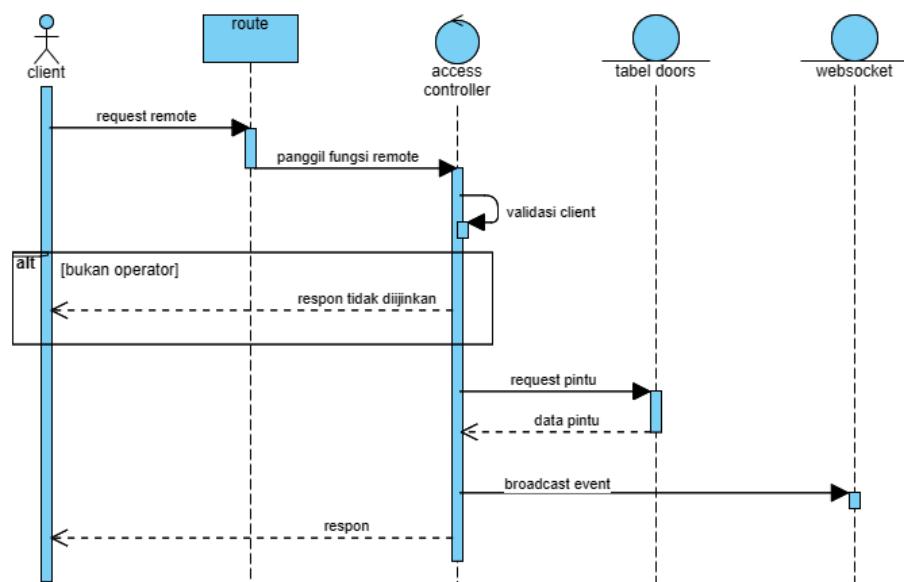
6. Ganti profil



Gambar 2.19 Sequence diagram API ganti profil

Dapat dilihat pada Gambar 2.19 di atas, untuk mengganti profil langkah pertama adalah pengguna atau operator mengirimkan data profil ke *endpoint* “*/api/update-profile*”. Kemudian, di dalam kontroler data yang telah diterima akan dimasukkan ke dalam tabel *users* untuk memperbarui profil dan terakhir kontroler mengembalikan respons bahwa profil berhasil diubah.

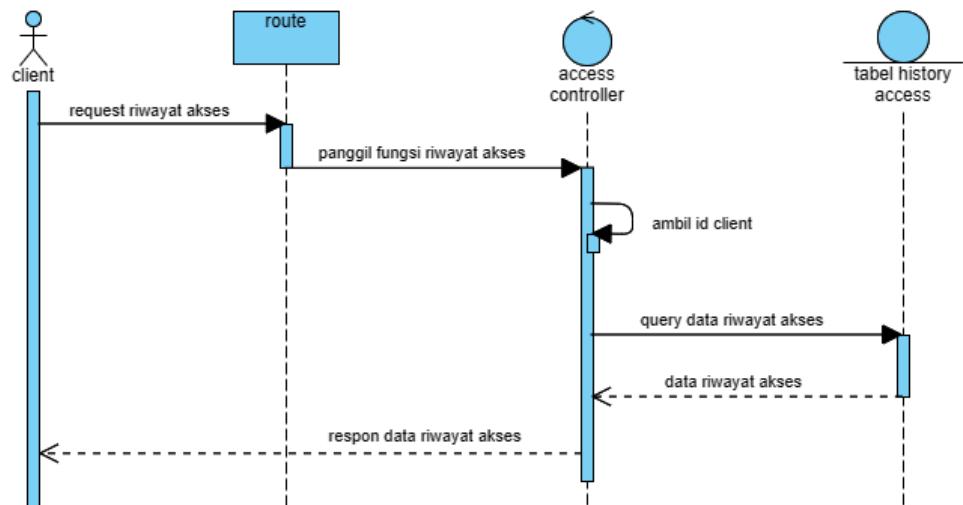
7. Remote pintu



Gambar 2.20 Sequence diagram API remote pintu

Dapat dilihat pada Gambar 2.21 di atas, untuk melakukan *remote* pintu langkah pertama adalah operator melakukan *request* ke *endpoint* “/api/remote-access” dengan mengirimkan identitas pintu yang akan dikendalikan. Kemudian, kontroler memeriksa *client* untuk memastikan permintaan berasal dari operator. Jika berasal dari pengguna biasa maka kontroler akan mengembalikan respons tidak diizinkan. Selanjutnya, kontroler akan mengambil data pintu untuk melengkapi informasi yang dibutuhkan seperti kode kunci, kode gedung, token, dan lain sebagainya. Kemudian, data dikirim ke perangkat kunci pintu melalui koneksi *websocket* yang sudah terhubung. Terakhir, kontroler mengembalikan respons *remote* pintu telah dilaksanakan.

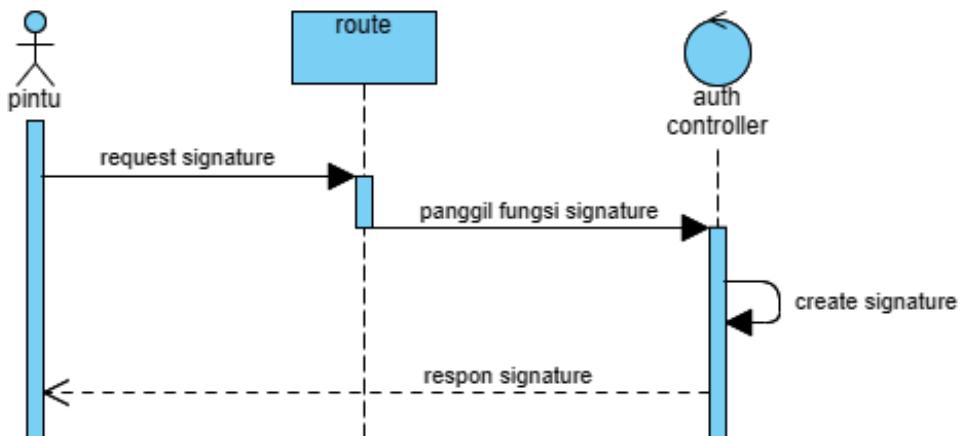
8. Riwayat akses



Gambar 2.21 Sequence diagram API riwayat akses

Dapat dilihat pada Gambar 2.22 untuk mendapatkan data riwayat akses maka pengguna melakukan permintaan ke *endpoint* “/api/my-history”. Kemudian, kontroler mengambil data riwayat akses pengguna dengan melakukan *query* ke *database* dengan menyertakan identitas pengguna sebagai parameter. Selanjutnya, data yang sudah diperoleh seperti nama pintu, nama gedung, waktu aktivitas, dan jenis aktivitas akan dikirimkan kembali ke pengguna sebagai respons untuk diolah pada tampilan aplikasi *mobile*.

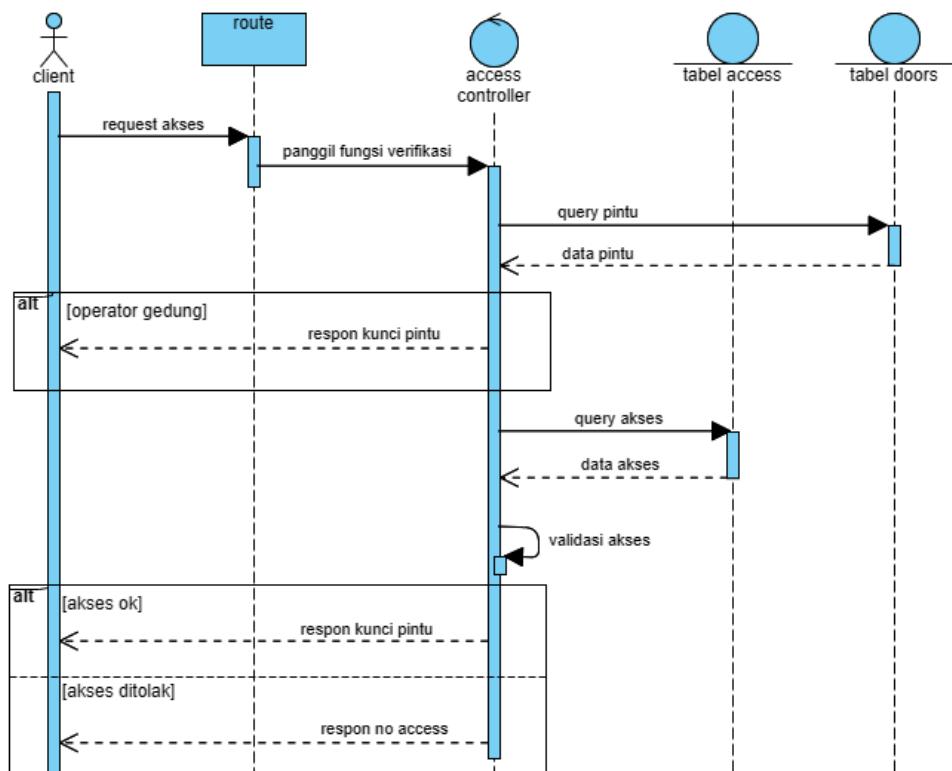
9. Signature



Gambar 2.22 Sequence diagram API door signature

Dapat dilihat pada Gambar 2.23 di atas, untuk mendapatkan kode *signature pusher* langkah pertama adalah perangkat kunci pintu melakukan permintaan ke *endpoint* “/door/get-signature” dengan mengirimkan data-data seperti *socket-id*, *office-id*, dan *channel-data*. Dari data tersebut, kontroler membuat kode *signature* menggunakan metode yang ada pada protokol *pusher*. Setelah mendapatkan nilai *signature*, kontroler mengembalikan respons kode *signature* ke perangkat kunci pintu.

10. Verifikasi akses

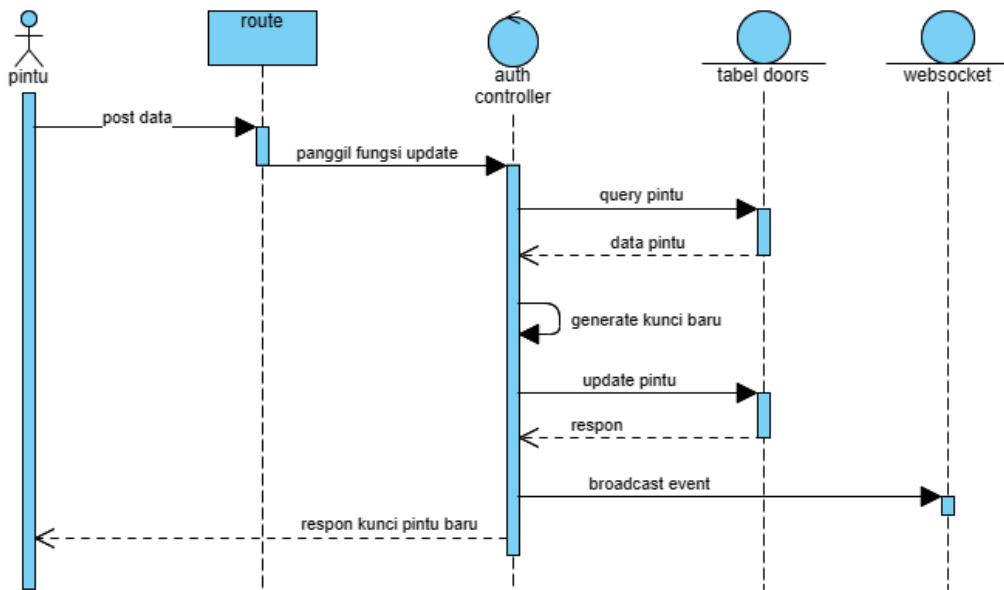


Gambar 2.23 Sequence diagram API verifikasi akses

Pada Gambar 2.24 di atas, untuk mendapatkan akses ke pintu, pengguna memindai kode QR pada pintu untuk mendapatkan informasi terkait pintu tersebut. Kemudian, data tersebut dikirimkan ke *server* melalui *endpoint* “/api/verify-access/{door-id}”. Kemudian, kontroler mengambil data pintu pada tabel *doors*. Jika permintaan berasal dari operator gedung dimana pintu tersebut berada maka kontroler akan mengizinkan dengan mengembalikan respons berupa kode kunci pintu. Jika

permintaan akses dilakukan oleh pengguna biasa maka kontroler akan memeriksa daftar akses di dalam tabel *access*. Jika pengguna memiliki akses dan masih berlaku maka kontroler akan mengembalikan respons akses diizinkan dan mengirimkan kode kunci untuk membuka pintu.

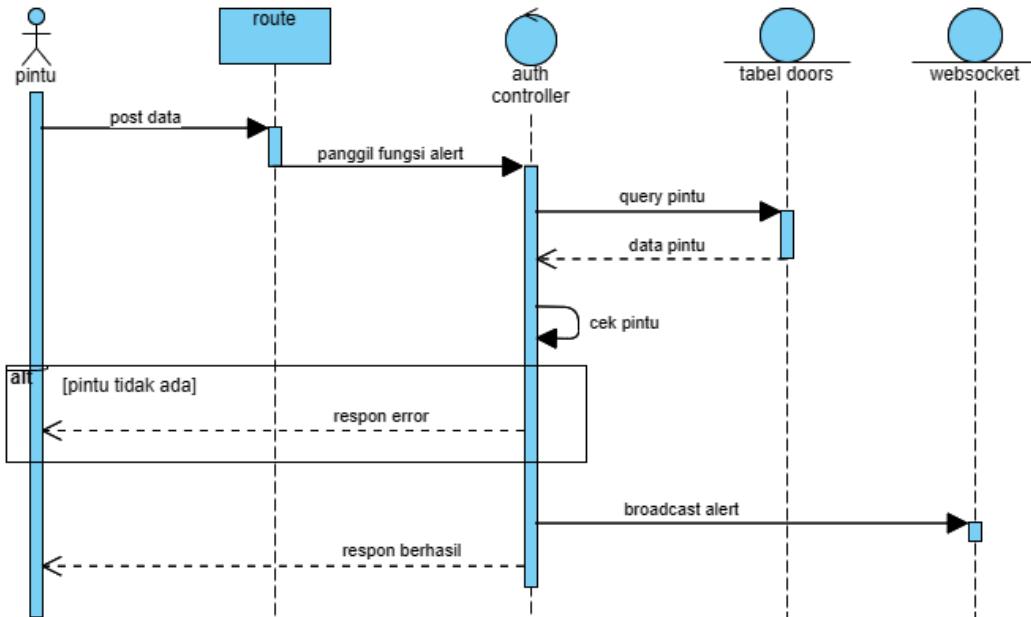
11. Update status pintu



Gambar 2.24 Sequence diagram API door update status

Terlihat pada Gambar 2.25 di atas, untuk melakukan *update* status langkah pertama adalah perangkat kunci pintu mengirimkan data-data seperti status penguncian dan *id socket* melalui endpoint “/door/update-status”. Kemudian, kontroler mengambil data pintu yang terkait untuk diperbarui menggunakan data status dan kode kunci yang baru. Lalu, kontroler melakukan *broadcasting* untuk menyiarkan bahwa status pintu berubah sehingga setiap informasi perubahan dapat tersampaikan secara langsung. Terakhir, kontroler mengembalikan respons *update* status telah dilaksanakan.

12. Peringatan pintu

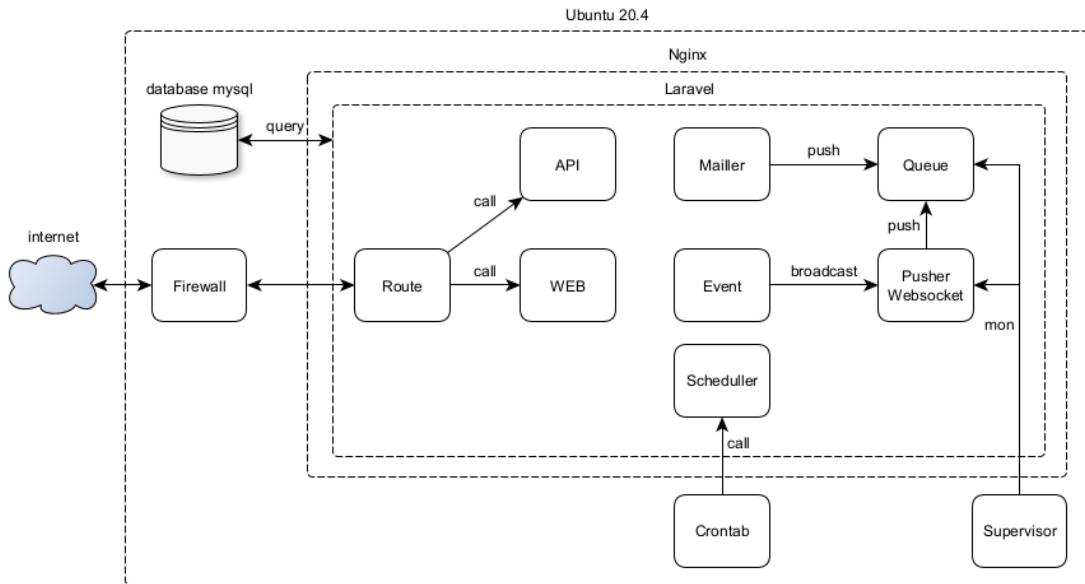


Gambar 2.25 Sequence diagram API door alert

Dapat dilihat pada Gambar 2.26 di atas, untuk memberikan peringatan langkah pertama adalah perangkat kunci pintu mengirimkan data-data seperti id-pintu, id-office, dan status peringatan melalui endpoint “/door/alert”. Kemudian, kontroler memeriksa pada tabel pintu untuk memastika bahwa pintu valid. Jika pintu tidak ditemukan maka kontroler mengembalikan respons *error*. Selanjutnya, kontroler menyiarkan peringatan melalui *websocket*. Terakhir, kontroler mengembalikan respons peringatan sudah dilaksanakan.

2.5 Server

Dengan menggunakan laravel sebagai *backend* yang mengatur kinerja dari perangkat kunci pintu tentunya diperlukan sebuah *server*. *Server* bertindak sebagai pusat pengolahan data dan berfungsi untuk menerima permintaan dari perangkat kunci pintu, mengatur akses, memproses logika bisnis, dan berkomunikasi dengan *database*. Diagram dari *backend server* dapat dilihat pada Gambar 2.27 di bawah ini.



Gambar 2.26 Konfigurasi server

Dapat dilihat pada Gambar 2.27 di atas, *server* dibangun menggunakan sistem operasi ubuntu 20.04. Ubuntu merupakan bagian dari sistem operasi linux yang biasa digunakan baik untuk perangkat dekstop maupun *server* karena *open source* dan ringan. Dengan menggunakan ubuntu 20.04 sebagai sistem operasi *server*, pengguna dapat memanfaatkan kestabilan, keamanan, dan dukungan jangka panjang untuk menjalankan aplikasi laravel dengan aman dan efisien.

Di dalam sistem operasi ubuntu 20.04, dipasang nginx yang digunakan sebagai *web server* untuk menjalankan aplikasi laravel. Dengan menggunakan nginx maka aplikasi laravel dapat dijalankan dengan efisien dikarenakan nginx memiliki karakteristik ringan dan cepat. Dengan menggunakan nginx pengguna juga bisa membagi *server* menjadi beberapa blok yang dapat digunakan untuk menjalankan API dan *websocket* secara bersamaan. Pada nginx juga dipasang sertifikat SSL yang digunakan untuk mengenkripsi semua komunikasi dengan menggunakan HTTPS sehingga keamanan data akan terjamin dengan adanya jalur komunikasi yang terenkripsi.

Di dalam ubuntu juga dipasang MySQL sebagai pusat penyimpanan data yang terhubung ke laravel. Dengan menggunakan *database* yang berjalan pada *server* yang sama maka kecepatan transfer data akan sangat cepat dengan menghilangkan latensi jaringan. Hal tersebut sesuai dengan karakteristik sistem yang dibangun yaitu sistem cepat dan efisien.

Dengan adanya fitur penjadwalan otomatis di dalam sistem kunci pintu maka pengguna juga harus memasang crontab untuk menjalankan penjadwalan yang ada pada Laravel. Penjadwalan akan dipanggil setiap 1 menit sekali untuk memeriksa apakah ada jadwal yang harus dilaksanakan atau tidak.

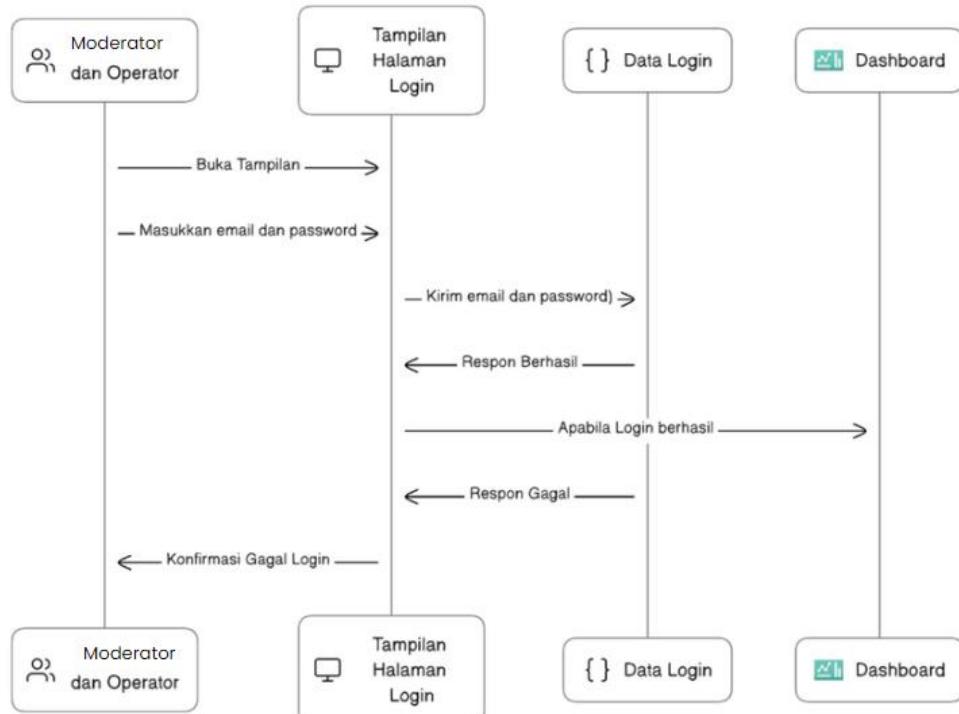
Pada laravel juga mengimplementasikan fitur antrian yang bertujuan untuk meningkatkan kinerja dari sistem sehingga fungsi antrian harus dijaga agar selalu berada dalam kondisi berjalan. Oleh sebab itu, *server* juga dipasang *supervisor* yang digunakan untuk memantau kinerja dari antrian. *Supervisor* merupakan sebuah sistem pengawas atau *process control system* yang digunakan untuk mengontrol dan mengelola proses-proses yang berjalan di dalam sistem operasi. *Supervisor* akan memastikan bahwa proses antrian pada laravel tetap berjalan secara terus-menerus dan akan memulai ulang proses jika terjadi kegagalan serta mengelola jumlah pekerja antrian atau *queue workers* yang berjalan secara paralel untuk meningkatkan efisiensi dan kinerja.

Supervisor juga digunakan untuk menjaga *websocket* agar berjalan secara terus-menerus. *Websocket* memegang peranan penting di dalam sistem ini karena *websocket* menjadi jalur komunikasi yang menghubungkan perangkat kunci pintu dengan *server* pusat sehingga jika terjadi gangguan pada kinerja *websocket* maka seluruh kinerja dari perangkat kunci pintu juga akan terganggu. Oleh karena itu, diperlukan pengawasan menggunakan *supervisor* untuk menjaga kinerja dari *websocket*.

2.6 Tampilan Website

Implementasi tampilan *website* menggunakan *library livewire* sehingga memungkinkan tampilan yang interaktif. Beberapa tampilan yang tersedia pada *website* adalah sebagai berikut:

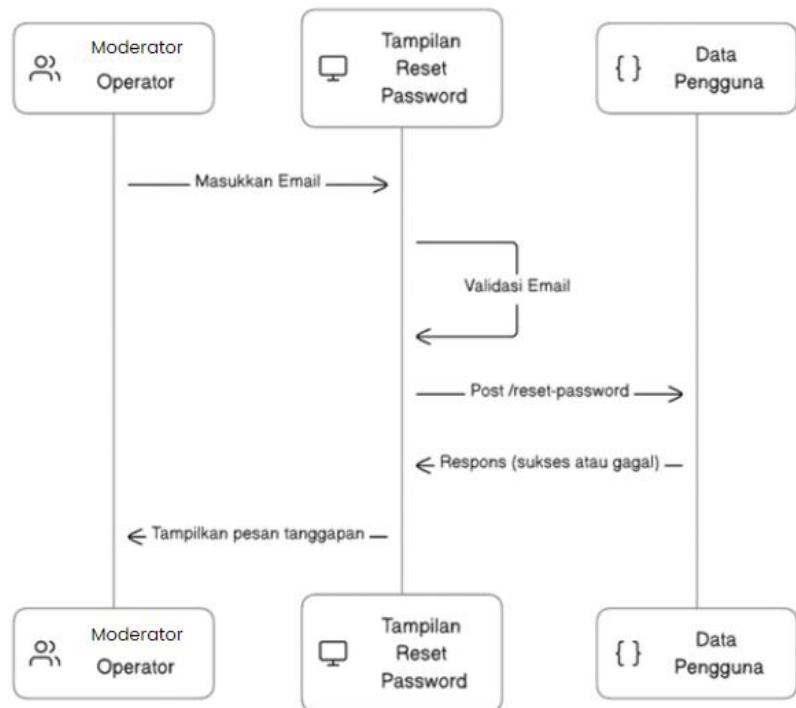
1. Halaman *login*



Gambar 2.28 Diagram *sequence* tampilan halaman *login website*

Gambar 2.28 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah yang terjadi ketika moderator atau operator membuka tampilan halaman *login* dan mencoba melakukan *login* ke dalam sistem. Pertama, moderator atau operator membuka tampilan halaman *login* dengan mengakses antarmuka yang ada pada monitor. Kemudian, moderator atau operator memasukkan email dan *password*. Selanjutnya, email dan *password* tadi dikirim ke data *login*. Jika data *login* yang dikirimkan benar yaitu email dan *password* sesuai dengan yang terdaftar di sistem maka data *login* akan memberikan respons berhasil ke tampilan halaman *login*. Apabila *login* berhasil moderator atau operator diarahkan ke *dashboard*. Namun, jika *login* gagal maka data *login* akan memberikan respons gagal ke tampilan halaman *login*, dilanjutkan mengonfirmasi gagal *login* ke moderator atau operator.

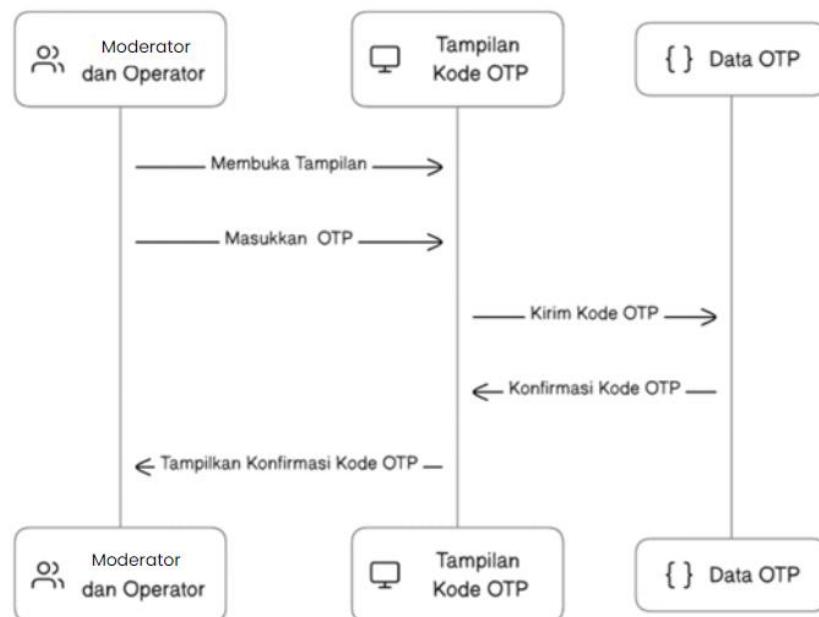
2. Reset Password



Gambar 2.29 Diagram *sequence* tampilan halaman *reset password*

Gambar 2.29 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah saat moderator atau operator menggunakan tampilan *reset password* untuk melakukan *reset* kata sandi dalam sistem. Pertama, moderator atau operator membuka tampilan *reset password* dan memasukkan email. Setelah memasukkan email, tampilan *reset password* melakukan validasi untuk memastikan bahwa email yang dimasukkan valid dan sesuai format. Setelah validasi berhasil, tampilan *reset password* akan mengirim permintaan *reset password* ke data pengguna melalui *endpoint* */reset-password*. Data pengguna memproses permintaan tersebut dan memberikan respons (sukses atau gagal) ke tampilan *reset password*. Kemudian, tampilan *reset password* menampilkan pesan tanggapan ke pengguna atau operator.

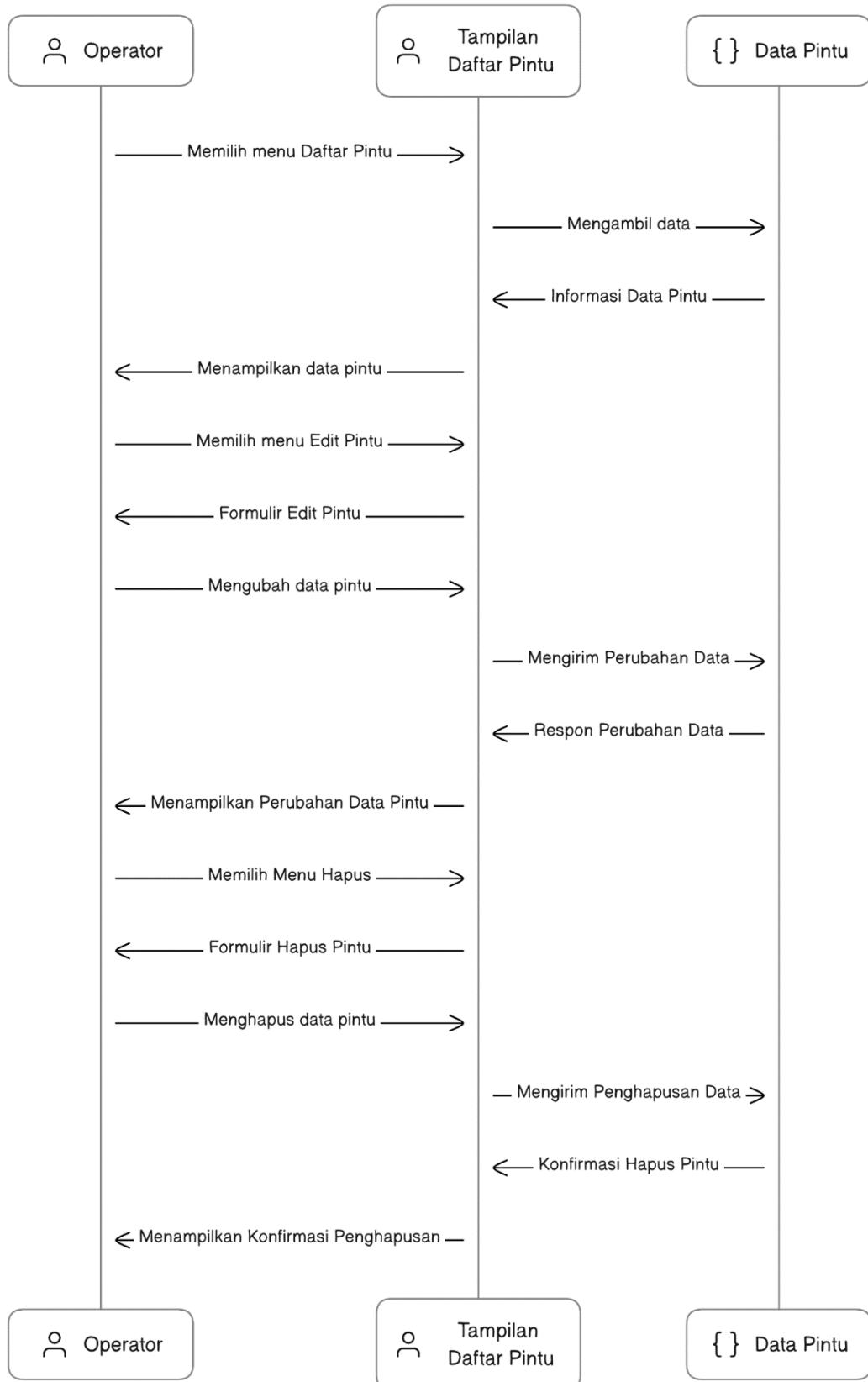
3. Verifikasi email



Gambar 2.30 Diagram *sequence* tampilan kode OTP

Gambar 2.30 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah saat moderator atau operator menggunakan tampilan kode OTP untuk memasukkan OTP (*One-Time Password*) dalam sistem. Pertama, moderator atau operator membuka tampilan kode OTP melalui antarmuka yang ada. Kemudian, pengguna atau operator memasukkan OTP yang telah diterima. Setelah memasukkan OTP, tampilan kode OTP akan mengirimkan kode OTP ke data OTP untuk memvalidasi apakah kode yang dimasukkan benar atau tidak. Data OTP akan memproses kode OTP yang diterima dan memberikan konfirmasi apakah kode OTP tersebut valid atau tidak. Jika kode OTP yang dimasukkan valid, tampilan kode OTP akan menampilkan konfirmasi ke moderator atau operator bahwa kode OTP telah berhasil diverifikasi. Namun, jika kode OTP yang dimasukkan tidak valid, tampilan kode OTP akan menampilkan konfirmasi bahwa kode OTP tidak valid dan moderator atau operator perlu memasukkan kode OTP yang benar untuk dapat melanjutkan proses.

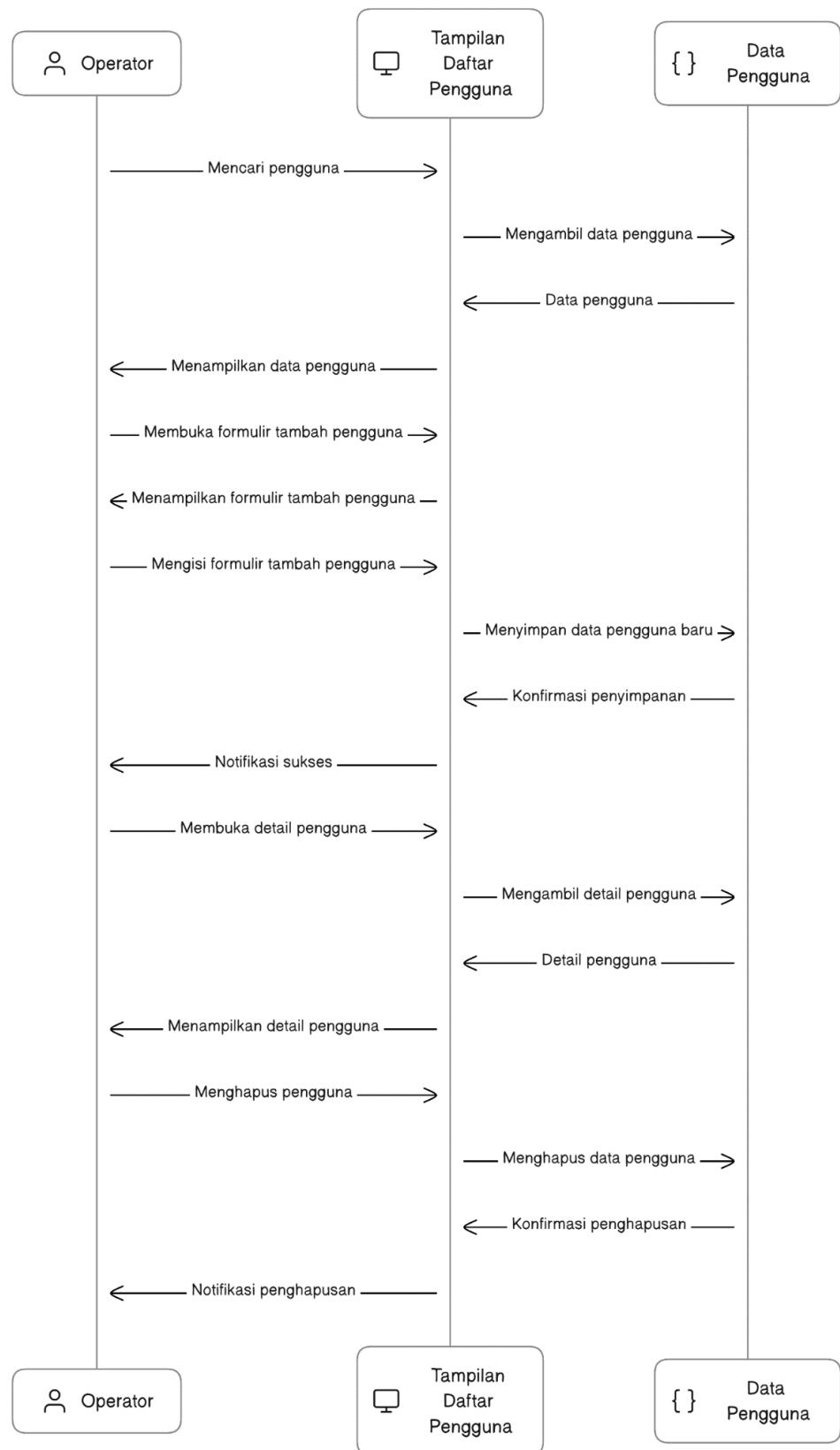
4. Menu Daftar Pintu



Gambar 2.31 Diagram *sequence* tampilan daftar pintu

Gambar 2.31 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah ketika operator menggunakan tampilan daftar pintu untuk mengelola daftar pintu dalam sistem. Pertama, operator memilih menu daftar pintu pada antarmuka yang ada. Kemudian, tampilan daftar pintu mengambil data di data pintu. Data pintu mengirimkan informasi data pintu ke tampilan daftar pintu sehingga operator dapat melihat daftar pintu yang ditampilkan. Selanjutnya, operator dapat memilih menu edit pintu untuk mengubah data pintu tertentu. Tampilan daftar pintu menampilkan formulir edit pintu yang memungkinkan operator untuk mengubah data pintu sesuai kebutuhan. Operator mengisi formulir edit pintu dengan perubahan data yang diinginkan dan menyimpannya. Tampilan daftar pintu mengirim perubahan data ke data pintu. Data pintu mengolah perubahan data yang diterima dan memberikan respons perubahan data. Tampilan daftar pintu menampilkan konfirmasi perubahan data pintu ke operator. Selain itu, operator juga memiliki opsi untuk memilih menu hapus untuk menghapus pintu tertentu. Tampilan daftar pintu menampilkan formulir hapus pintu yang memungkinkan operator untuk melakukan penghapusan data pintu. Operator mengisi formulir hapus pintu untuk mengonfirmasi penghapusan data pintu. Setelah itu, tampilan daftar pintu mengirim permintaan penghapusan data ke data pintu. Data pintu memproses permintaan penghapusan pintu dan memberikan konfirmasi hapus pintu. Tampilan daftar pintu menampilkan konfirmasi penghapusan ke operator.

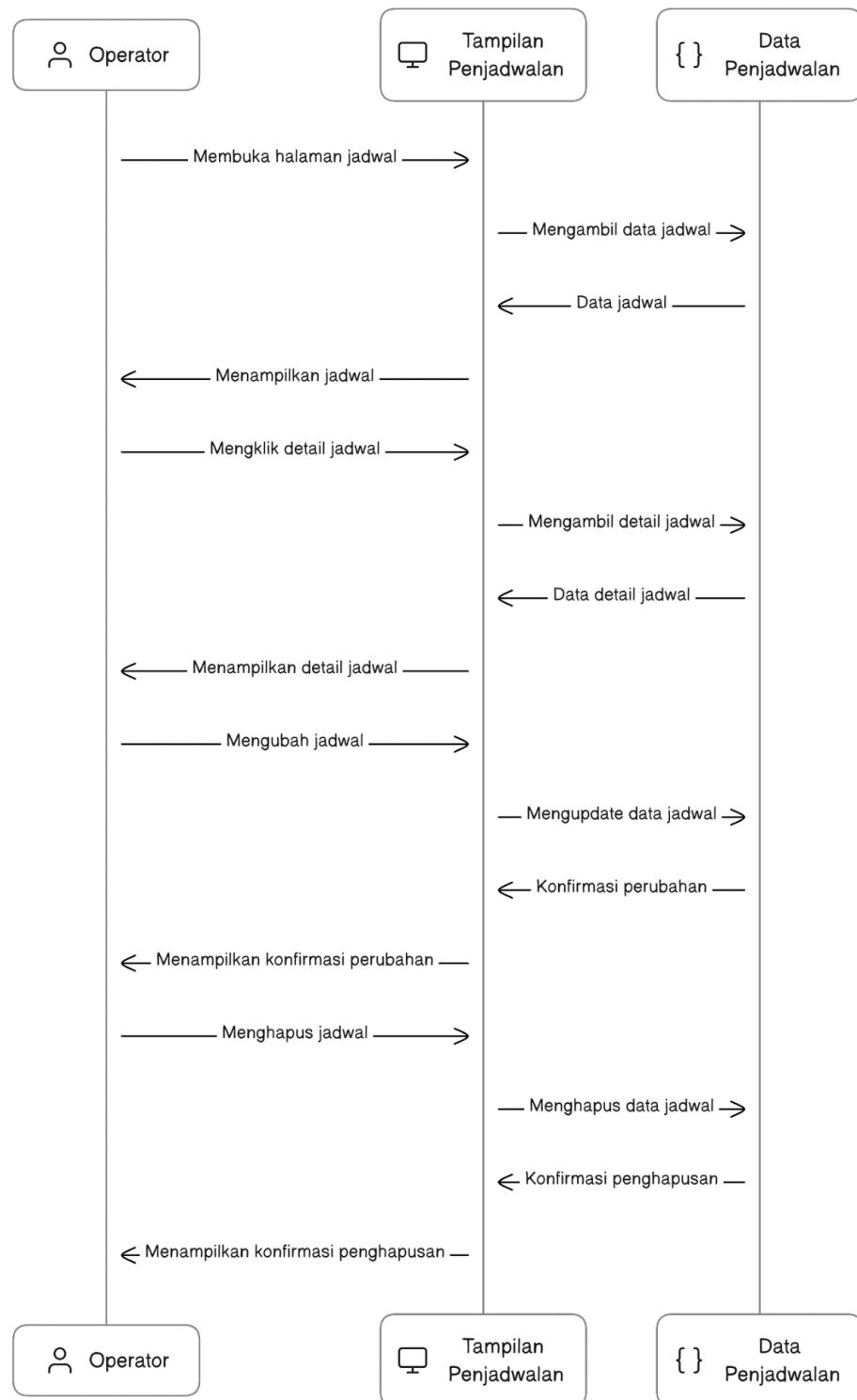
5. Menu Daftar Pengguna



Gambar 2.32 Diagram *sequence* tampilan daftar pengguna

Gambar 2.32 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah ketika operator menggunakan tampilan daftar pengguna untuk mengelola data pengguna dalam sistem. Pertama, operator mencari pengguna dengan membuka tampilan daftar pengguna. Kemudian, tampilan daftar pengguna mengambil data pengguna dari komponen data pengguna untuk menampilkan informasi mengenai daftar pengguna yang ada. Komponen data pengguna mengirimkan data pengguna ke tampilan daftar pengguna sehingga operator dapat melihat daftar pengguna yang ada. Selanjutnya, operator dapat membuka formulir tambah pengguna dengan memilih opsi untuk menambahkan pengguna baru. Tampilan daftar pengguna menampilkan formulir tambah pengguna yang memungkinkan operator untuk mengisi data pengguna baru. Operator mengisi formulir tambah pengguna dengan informasi yang diperlukan dan menyimpannya. Tampilan daftar pengguna menyimpan data pengguna baru ke komponen data pengguna. Komponen data pengguna mengelola data pengguna baru yang diterima dan memberikan konfirmasi bahwa penyimpanan data pengguna telah berhasil dilakukan. Tampilan daftar pengguna menampilkan notifikasi sukses ke operator. Selain itu, operator juga dapat membuka detail pengguna dengan memilih pengguna tertentu dari daftar pengguna. Tampilan daftar pengguna mengambil detail pengguna dari data pengguna untuk menampilkan informasi lebih rinci mengenai pengguna yang dipilih. Data pengguna mengirimkan detail pengguna ke tampilan daftar pengguna sehingga operator dapat melihat informasi detail pengguna tersebut. Operator juga memiliki opsi untuk menghapus pengguna tertentu. Jika memilih untuk menghapus data pengguna, tampilan daftar pengguna mengirim permintaan penghapusan ke data pengguna. Komponen data pengguna mengelola proses penghapusan data pengguna dan memberikan konfirmasi bahwa penghapusan telah berhasil dilakukan. Tampilan daftar pengguna menampilkan notifikasi penghapusan kepada operator.

6. Penjadwalan

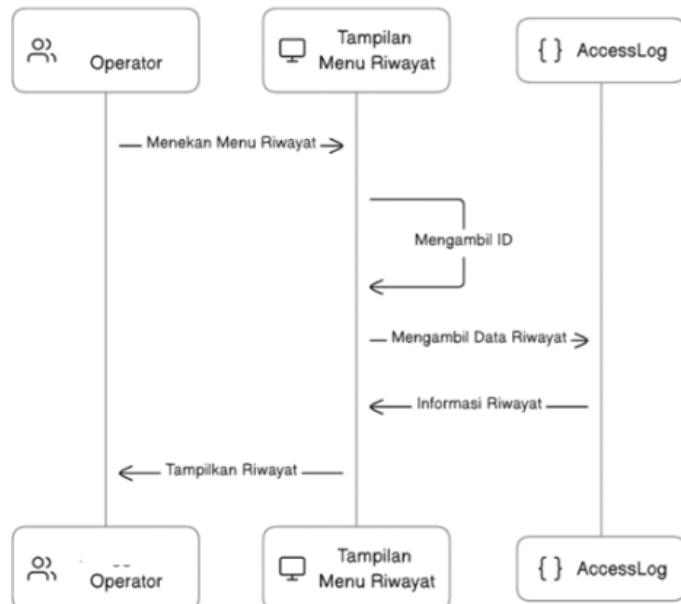


Gambar 2.33 Diagram *sequence* tampilan penjadwalan

Gambar 2.33 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah yang terjadi ketika operator membuka dan mengelola jadwal melalui

tampilan penjadwalan pada *website*. Pertama, operator membuka halaman jadwal yang menampilkan daftar jadwal yang tersedia. Tampilan penjadwalan menghubungi data penjadwalan untuk mengambil data jadwal. Data penjadwalan mengirim data jadwal yang diperlukan ke tampilan penjadwalan. Setelah itu, tampilan penjadwalan menampilkan jadwal kepada operator. Selanjutnya, operator dapat mengklik detail jadwal pada tampilan penjadwalan. Jika operator mengkliknya, tampilan penjadwalan mengambil detail jadwal di data penjadwalan. Data penjadwalan mengirim data detail jadwal ke tampilan penjadwalan. Tampilan penjadwalan menampilkan detail jadwal ke operator. Operator juga memiliki opsi untuk mengubah jadwal. Jika pengguna memilih untuk mengubah jadwal, tampilan penjadwalan menghubungi data penjadwalan untuk meng-*update* data jadwal. Data penjadwalan melakukan konfirmasi perubahan ke tampilan penjadwalan. Tampilan penjadwalan menampilkan konfirmasi perubahan ke operator. Operator juga memiliki opsi untuk menghapus jadwal. Jika operator memilih untuk menghapus jadwal, tampilan penjadwalan menghubungi data penjadwalan untuk menghapus data jadwal. Data penjadwalan melakukan konfirmasi penghapusan ke tampilan penjadwalan. Tampilan penjadwalan menampilkan konfirmasi penghapusan jadwal ke operator.

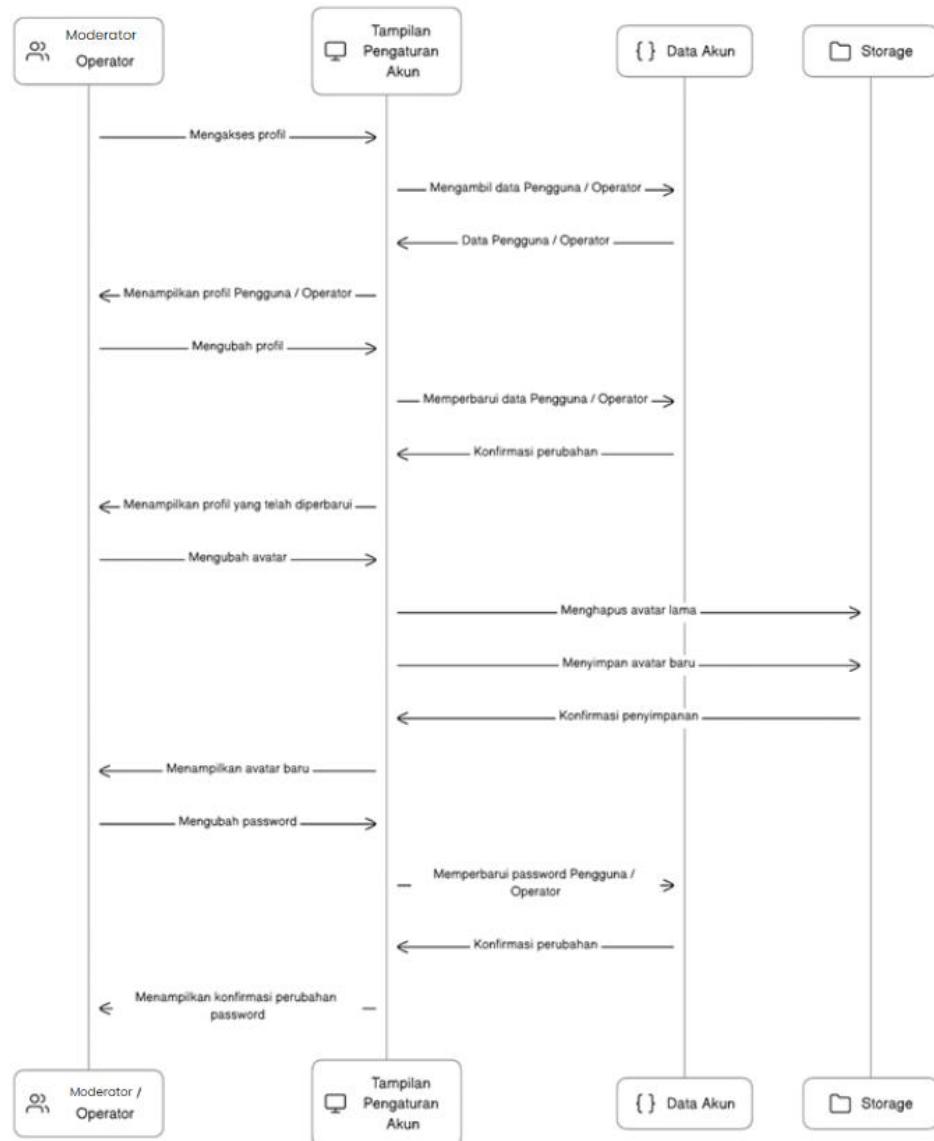
7. Menu Riwayat Akses



Gambar 2.34 Diagram *sequence* tampilan menu riwayat

Gambar 2.34 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah ketika operator menekan menu riwayat pada *website* untuk melihat riwayat akses. Pertama, operator menekan menu riwayat pada tampilan menu riwayat di *website*. Tampilan menu riwayat mengambil ID untuk digunakan dalam pengambilan data riwayat akses di *AccessLog*. *AccessLog* mengirim informasi riwayat ke tampilan menu riwayat. Setelah itu, tampilan menu riwayat menampilkan riwayat akses ke operator.

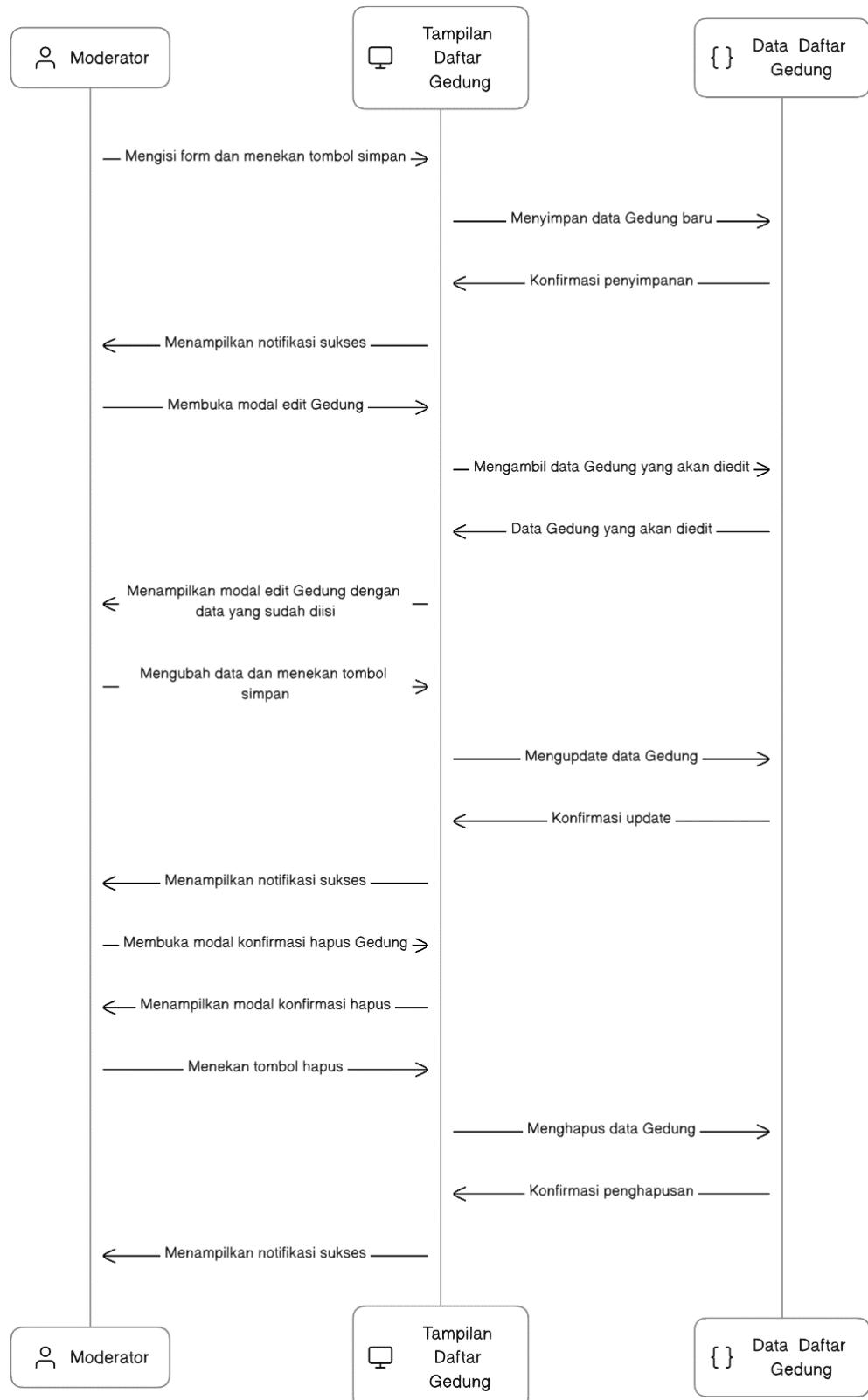
8. Pengaturan Akun



Gambar 2.35 Diagram *sequence* tampilan pengaturan akun

Gambar 2.35 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah ketika moderator atau operator mengakses dan mengelola profil melalui halaman pengaturan akun pada *website*. Pertama, moderator atau operator mengakses profil pada tampilan pengaturan akun. Tampilan pengaturan akun menghubungi data akun untuk mengambil data moderator atau operator. Data akun mengirimkan data moderator atau operator ke tampilan pengaturan akun. Setelah itu, tampilan pengaturan akun menampilkan profil ke moderator atau operator. Selanjutnya, moderator atau operator memiliki opsi untuk mengubah profilnya. Jika pengguna memilih untuk mengubah profil, tampilan pengaturan akun menghubungi data akun untuk memperbarui data moderator atau operator. Data akun mengonfirmasi perubahan ke tampilan pengaturan akun. Tampilan pengaturan akun menampilkan profil yang telah diperbarui ke moderator atau operator. Selain itu, moderator atau operator juga dapat mengubah foto profil. Jika memilih untuk mengubah foto, tampilan pengaturan akun menghubungi *storage* untuk menghapus foto lama dan menyimpan foto baru. *Storage* memberikan konfirmasi penyimpanan foto baru ke tampilan pengaturan akun. Tampilan pengaturan akun menampilkan foto baru ke moderator atau operator. Terakhir, moderator atau operator dapat mengubah *password*. Jika moderator atau operator memilih untuk mengubah *password*, tampilan pengaturan akun menghubungi data akun untuk memperbarui *password* moderator atau operator. Data akun memberikan konfirmasi perubahan ke tampilan pengaturan akun. Tampilan pengaturan akun menampilkan konfirmasi perubahan *password* ke moderator atau operator.

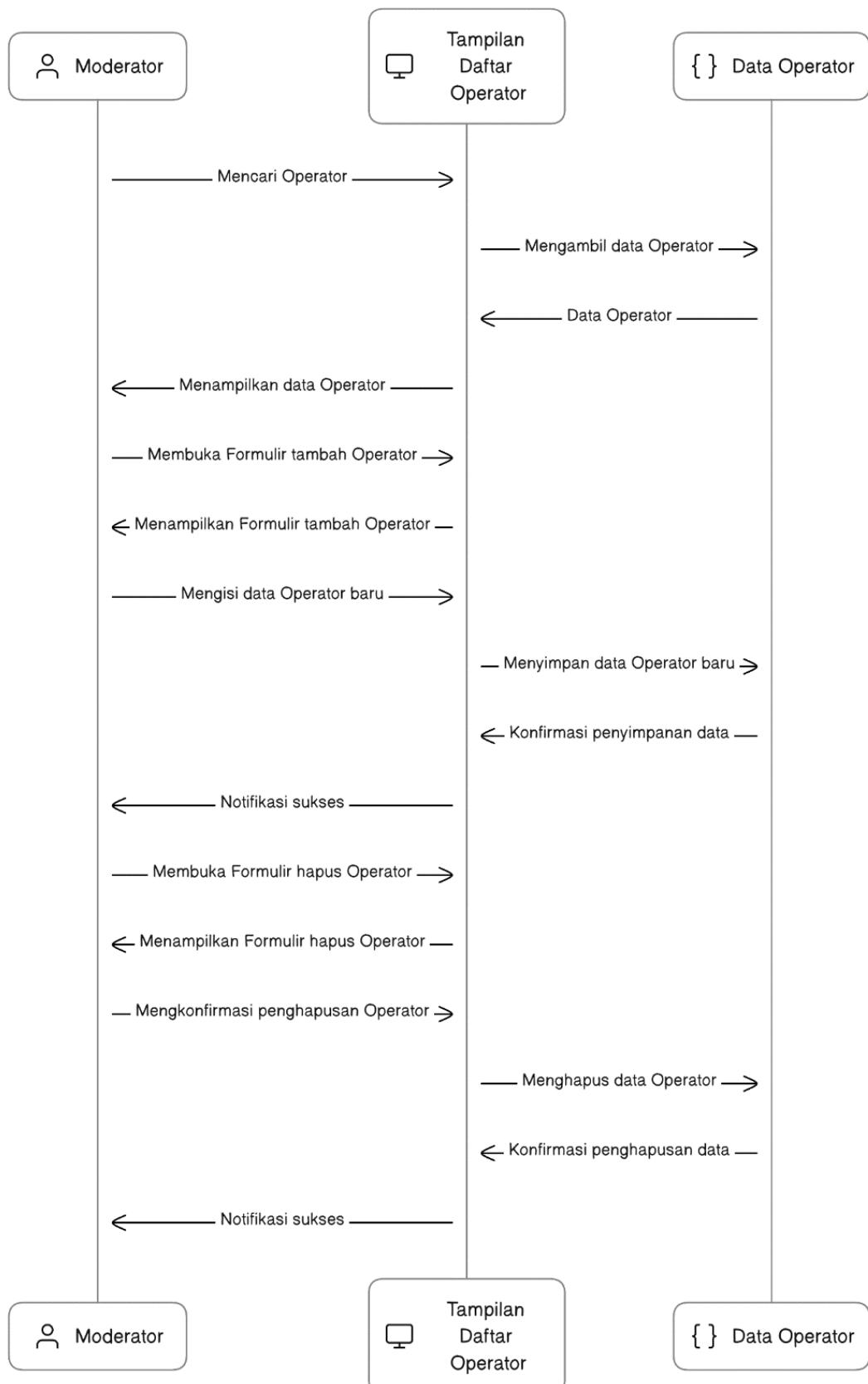
9. Daftar Gedung



Gambar 2.36 Diagram *sequence* tampilan daftar gedung

Gambar 2.36 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah moderator menggunakan halaman daftar gedung untuk menambah, mengubah, dan menghapus data gedung dalam sistem. Pertama, moderator membuka tampilan daftar gedung, mengisi form, dan menekan tombol simpan pada tampilan daftar gedung. Tampilan daftar gedung menyimpan data gedung baru ke dalam data daftar gedung. Data daftar gedung memberikan konfirmasi penyimpanan ke tampilan daftar gedung. Kemudian, tampilan daftar gedung menampilkan notifikasi sukses ke moderator. Selanjutnya, moderator memiliki opsi untuk mengedit gedung dengan membuka modal edit gedung. Tampilan daftar gedung mengambil data gedung yang akan diedit pada data daftar gedung. Data daftar gedung memberikan data gedung yang akan diedit ke tampilan daftar gedung. Tampilan daftar gedung menampilkan modal edit gedung dengan data yang sudah diisi ke moderator. Moderator mengubah data dan menekan tombol simpan pada tampilan daftar gedung. Tampilan daftar gedung meng-*update* data gedung pada data daftar gedung. Data daftar gedung memberikan konfirmasi *update* ke tampilan daftar gedung. Moderator menerima notifikasi sukses dari tampilan daftar gedung. Terakhir, moderator memiliki opsi untuk menghapus gedung dengan membuka modal konfirmasi hapus gedung. Tampilan daftar gedung menampilkan modal konfirmasi hapus gedung ke moderator. Moderator menekan tombol hapus pada tampilan daftar gedung. Tampilan daftar gedung menghubungi data daftar gedung untuk menghapus data gedung yang telah dipilih oleh moderator. Data daftar gedung memberikan konfirmasi penghapusan ke tampilan daftar gedung. Moderator menerima notifikasi sukses dari tampilan daftar gedung setelah data gedung berhasil dihapus.

10. Daftar Operator



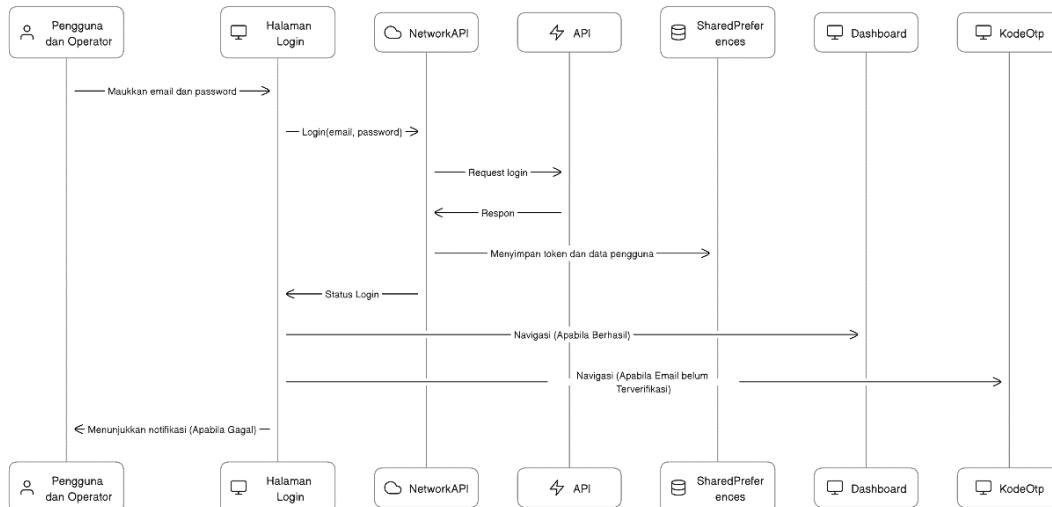
Gambar 2.37 Diagram *sequence* tampilan daftar operator

Gambar 2.37 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah moderator saat menggunakan tampilan daftar operator untuk melakukan pencarian, penambahan, dan penghapusan data operator dalam sistem. Langkah pertama adalah moderator membuka tampilan daftar operator untuk melakukan pencarian operator. Tampilan tersebut akan menghubungi data operator untuk mengambil data operator yang sesuai dengan kriteria pencarian yang dimasukkan oleh moderator. Setelah itu, komponen data operator memberikan data operator yang sesuai ke tampilan daftar operator. Kemudian, tampilan daftar operator menampilkan data operator ke moderator. Langkah berikutnya adalah ketika moderator ingin menambahkan operator baru melalui formulir tambah operator. Moderator membuka formulir tambah operator pada tampilan daftar operator. Tampilan daftar operator menampilkan formulir tersebut ke moderator. Moderator mengisi data operator baru. Setelah selesai mengisi, tampilan daftar operator menyimpan data operator baru di data operator baru. Data operator memberikan konfirmasi penyimpanan ke tampilan daftar operator. Moderator menerima notifikasi sukses dari tampilan daftar operator setelah data operator berhasil ditambahkan. Selain penambahan, moderator juga dapat melakukan penghapusan operator. Moderator membuka formulir hapus operator pada tampilan daftar operator. Tampilan daftar operator menampilkan formulir tersebut ke moderator. Moderator mengonfirmasi penghapusan operator pada tampilan daftar operator. Setelah itu, tampilan daftar operator menghapus data operator di komponen data operator. Data operator memberikan konfirmasi penghapusan data ke tampilan daftar operator. Moderator menerima notifikasi sukses dari tampilan daftar operator setelah data operator berhasil dihapus.

2.7 Aplikasi *Mobile*

Implementasi aplikasi *mobile* ini menggunakan kerangka kerja *Flutter* dengan bahasa *Dart*. Beberapa metode yang diimplementasikan pada aplikasi *mobile* ini adalah sebagai berikut:

1. Tampilan Halaman Login

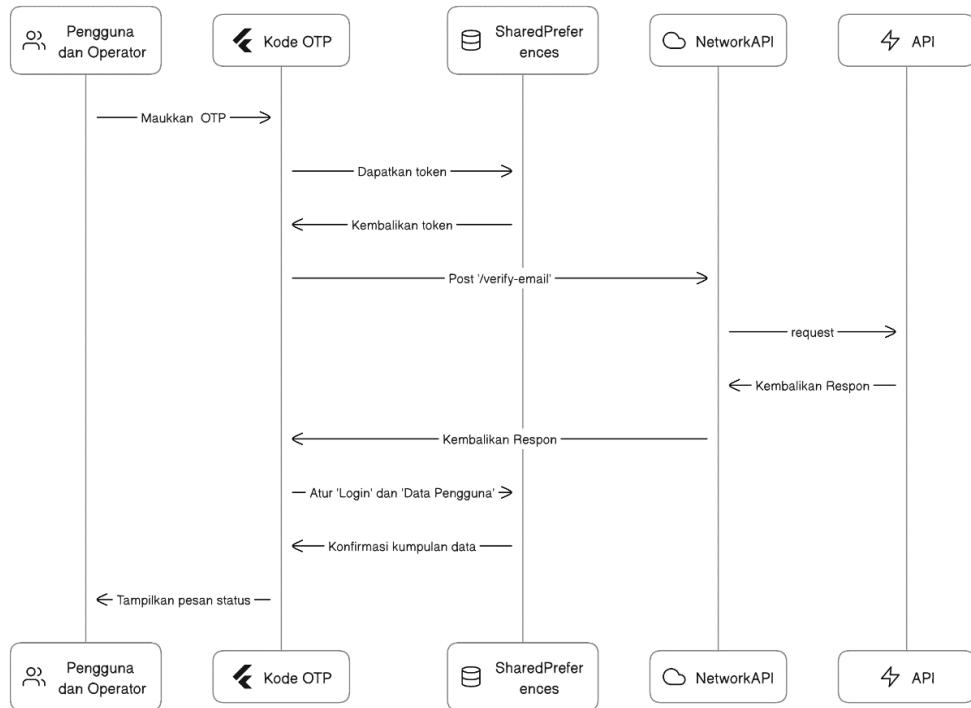


Gambar 2.38 Diagram *sequence* tampilan halaman *login*

Gambar 2.38 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah saat pengguna atau operator melakukan *login* di halaman *login* sistem. Pertama, pengguna atau operator memasukkan email dan *password* melalui antarmuka halaman *login*. Kemudian, antarmuka tersebut menghubungi *NetworkAPI* untuk memproses *login* dengan mengirimkan email dan *password*. Setelah menerima permintaan *login*, *NetworkAPI* berkomunikasi dengan *API* untuk *request login*. *API* memberikan respons ke *NetworkAPI* mengenai status *login* (berhasil atau gagal). Jika *login* berhasil, *NetworkAPI* menyimpan token dan data pengguna yang diperlukan di *SharedPreferences*. Token tersebut akan digunakan untuk mengidentifikasi dan mengautentikasi pengguna atau operator di masa mendatang saat sesi *login*. Setelah menerima respons dari *API*, *NetworkAPI* memberikan status *login* ke halaman *login*. Jika *login* berhasil, halaman *login* akan berlanjut ke halaman *dashboard*. Namun, jika email belum terverifikasi, halaman *login* akan berlanjut ke halaman *kodeOtp* untuk memverifikasi email. Jika *login* gagal, halaman *login* akan menampilkan notifikasi kegagalan *login* ke pengguna atau operator. Dengan demikian, diagram tersebut memberikan gambaran yang jelas tentang proses *login* dalam sistem serta bagaimana antarmuka dan komponen

lainnya berinteraksi untuk mencapai autentikasi dan navigasi ke halaman yang sesuai berdasarkan status *login*.

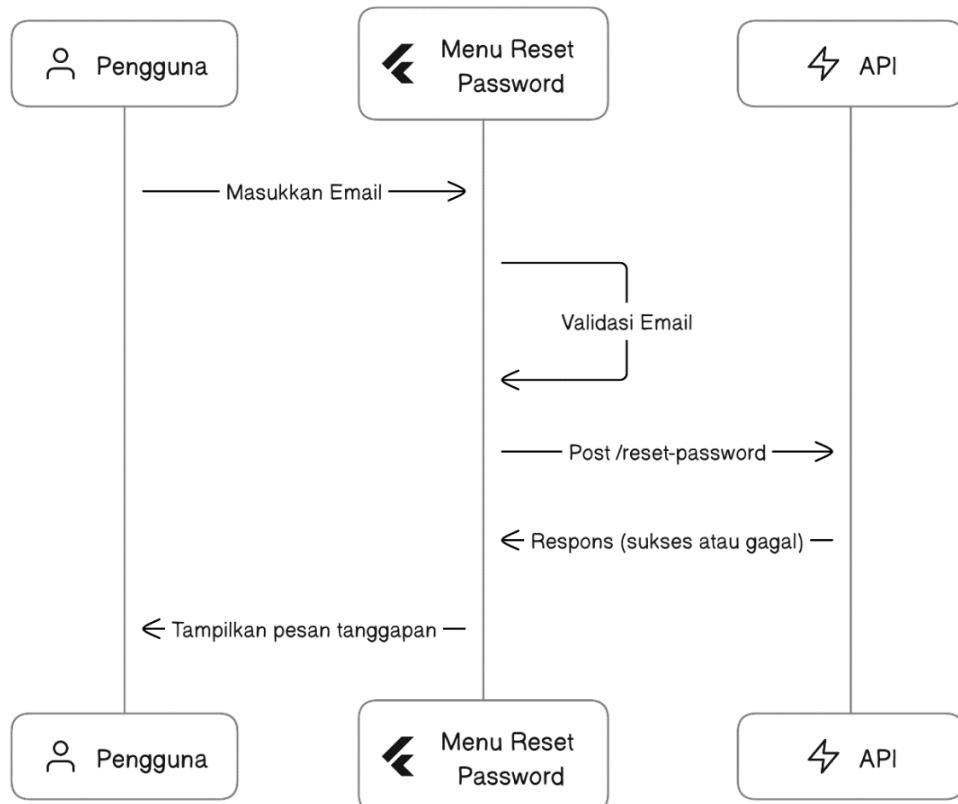
2. Verifikasi Email



Gambar 2.39 Diagram *sequence* tampilan verifikasi *email*

Gambar 2.39 di atas merupakan diagram *sequence* yang menggambarkan proses verifikasi email menggunakan kode OTP dalam sebuah aplikasi. Pertama, pengguna atau operator memasukkan kode OTP yang diterima melalui aplikasi berbasis *Flutter*. Setelah itu, kode OTP disimpan secara sementara dalam *SharedPreferences*, sebuah tempat penyimpanan lokal pada perangkat. Setelah itu, *SharedPreferences* mengembalikan token ke kode OTP. Kode OTP melakukan *post/verify email* ke *NetworkAPI* untuk memulai proses verifikasi. *NetworkAPI* melakukan *request* ke API untuk memverifikasi kode OTP. Setelah proses verifikasi selesai, API mengirimkan respons ke *NetworkAPI*. Respons tersebut kemudian diteruskan kembali oleh *NetworkAPI* ke kode OTP. Kode OTP mengatur *login* dan data pengguna pada *SharedPreferences*. Terakhir, *SharedPreferences* mengirimkan konfirmasi kumpulan data ke kode OTP dan kode OTP menampilkan pesan status ke operator atau pengguna.

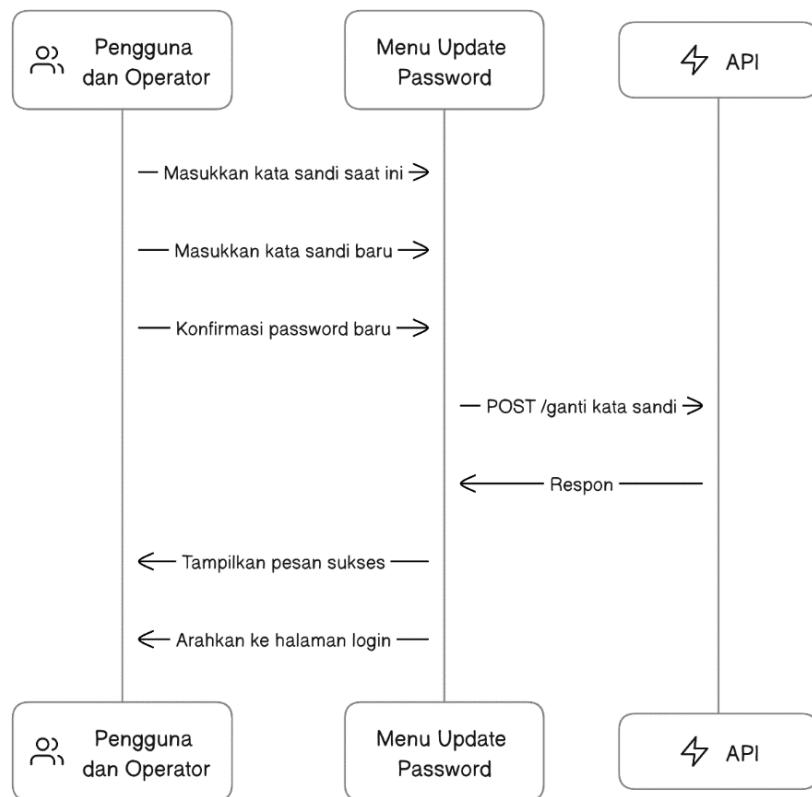
3. Reset Password



Gambar 2.40 Diagram *sequence* tampilan *reset password*

Gambar 2.40 di atas merupakan diagram *sequence* yang menggambarkan proses *reset password* dalam sebuah aplikasi. Pengguna mengajukan permintaan untuk *reset password* dengan memasukkan alamat email pada menu *reset password*. Yang dilanjutkan dengan melakukan validasi email yang dimasukkan untuk memastikan bahwa email tersebut valid dan terdaftar di dalam sistem. Setelah email divalidasi, menu *reset password* melakukan *post/reset password* ke *API server* melalui protokol HTTP POST dengan mengirimkan data permintaan *reset password* yang berisi alamat email pengguna. *API server* akan memproses permintaan tersebut dan mencoba me-*reset password* untuk akun yang sesuai dengan email yang diberikan. Kemudian, *API server* akan memberikan respons (sukses atau gagal) ke menu *reset password*. Terakhir, pengguna menerima pesan tanggapan yang ditampilkan oleh menu *reset password* pada aplikasi.

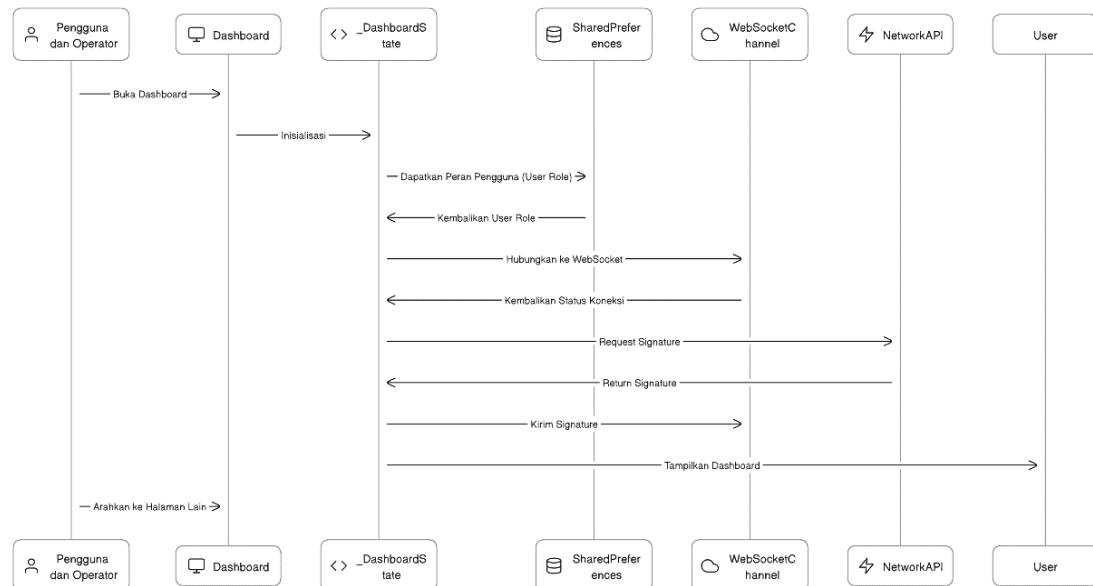
4. Update Password



Gambar 2.41 Diagram *sequence* tampilan menu *update password*

Gambar 2.41 di atas merupakan diagram *sequence* yang menggambarkan alur proses ketika pengguna atau operator ingin mengganti *password* melalui menu *update password* dalam sistem. Pertama, pengguna atau operator memasukkan kata sandi saat ini sebagai langkah verifikasi identitas pada menu *update password*. Setelah itu, pengguna atau operator memasukkan kata sandi baru yang ingin digunakan pada menu *update password*. Untuk memastikan keakuratan, sistem meminta pengguna untuk mengonfirmasi *password* baru yang telah dimasukkan sebelumnya. Kemudian, menu *update password* melakukan *post/ganti kata sandi* ke API. Setelah berhasil melakukan proses penggantian kata sandi, API memberikan respons ke menu *update password* untuk menunjukkan bahwa proses telah berhasil. Kemudian, menu *update password* menampilkan pesan sukses ke pengguna atau operator. Menu *update password* juga mengarahkan pengguna atau operator ke halaman *login*, sehingga pengguna atau operator dapat masuk kembali ke sistem dengan menggunakan kata sandi baru yang telah diperbarui.

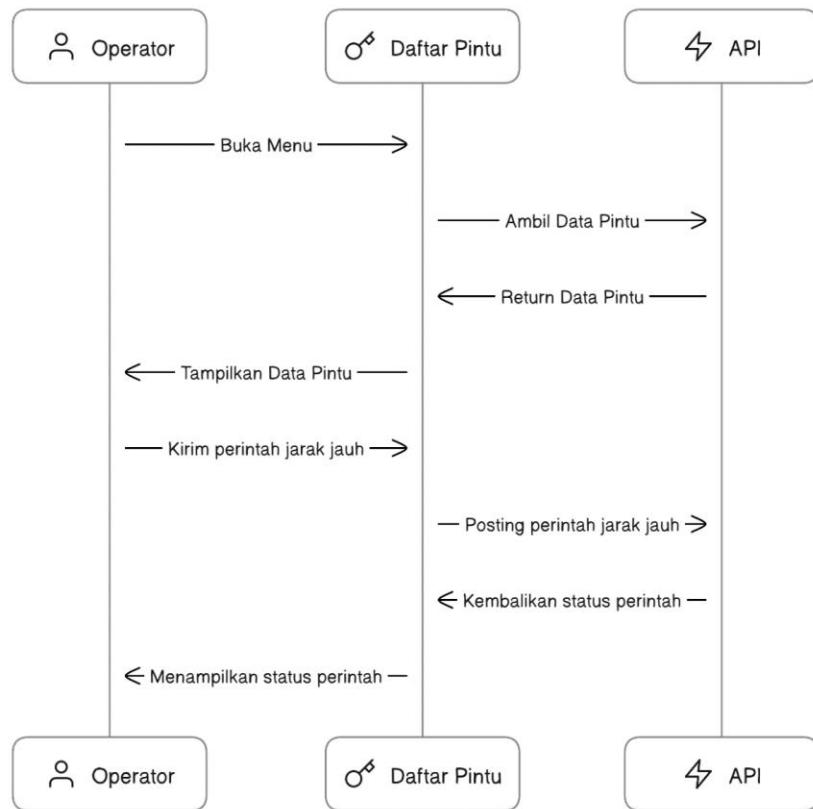
5. Tampilan Halaman *Dashboard*



Gambar 2.42 Diagrams *sequence* tampilan halaman *dashboard*

Gambar 2.42 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah ketika pengguna atau operator membuka *dashboard* dalam sistem. Saat pengguna atau operator membuka *dashboard*, sistem menampilkan tampilan awal. Kemudian, sistem memeriksa peran pengguna atau operator yang sedang aktif untuk menentukan hak aksesnya. Setelah itu, sistem memastikan koneksi ke *WebSocket* yang memungkinkan komunikasi *real-time*. Setelah koneksi berhasil, sistem akan meminta tanda tangan (*signature*) dari *server* melalui *NetworkAPI*. Setelah mendapatkan tanda tangan, sistem mengirimkan tanda tangan melalui *WebSocket*. Setelah menerima tanda tangan, sistem menampilkan *dashboard* dengan konten yang sesuai dengan peran pengguna atau operator. Jika ada halaman lain yang perlu diakses dari *dashboard*, sistem akan mengarahkan pengguna atau operator ke halaman lain. Dengan demikian, diagram tersebut memberikan gambaran tentang proses sederhana saat pengguna atau operator membuka *dashboard* dan bagaimana komponen-komponen dalam sistem berinteraksi untuk memberikan tampilan yang sesuai dengan peran pengguna dan menavigasikan ke halaman lain jika diperlukan.

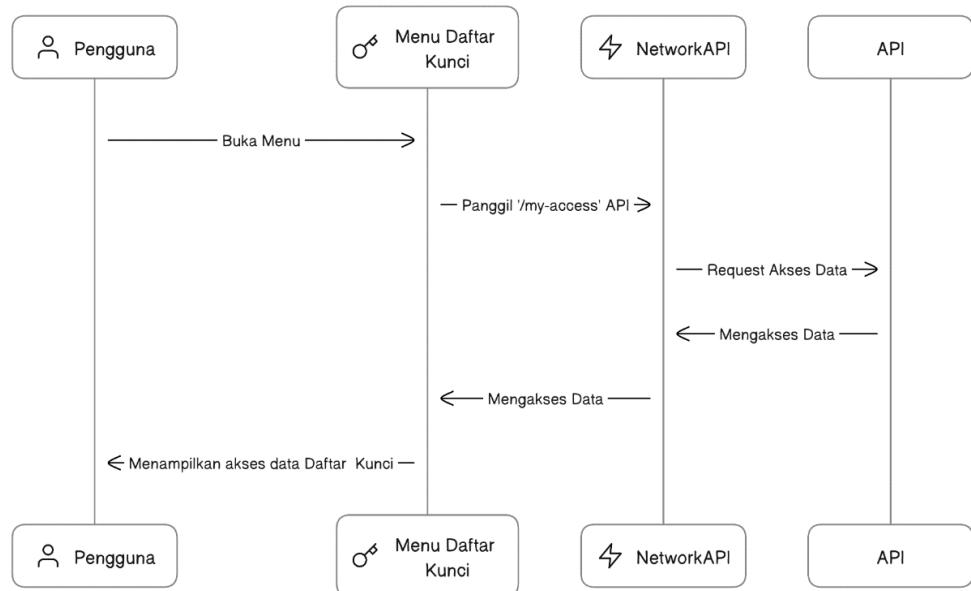
6. Tampilan Menu Daftar Pintu



Gambar 2.43 Diagram *sequence* tampilan menu daftar pintu

Gambar 2.43 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah yang terjadi ketika operator membuka daftar pintu dan menggunakan fitur jarak jauh untuk mengendalikan pintu-pintu dalam sistem. Pertama, operator membuka menu daftar pintu untuk melihat daftar pintu yang terhubung. Daftar pintu menghubungi API untuk mengambil data pintu. Setelah menerima permintaan, API *return* data pintu ke daftar pintu. Daftar pintu menampilkan data pintu tersebut ke operator, sehingga operator dapat melihat informasi tentang pintu-pintu yang ada. Selanjutnya, operator menggunakan fitur perintah jarak jauh untuk mengendalikan salah satu pintu dari jarak jauh. Daftar pintu menghubungi kembali API untuk melakukan *posting* perintah jarak jauh tersebut. Setelah menerima perintah, API memprosesnya dan mengembalikan status perintah ke daftar pintu. Kemudian, daftar pintu menampilkan status perintah tersebut ke operator, sehingga operator dapat melihat apakah perintah telah berhasil atau tidak.

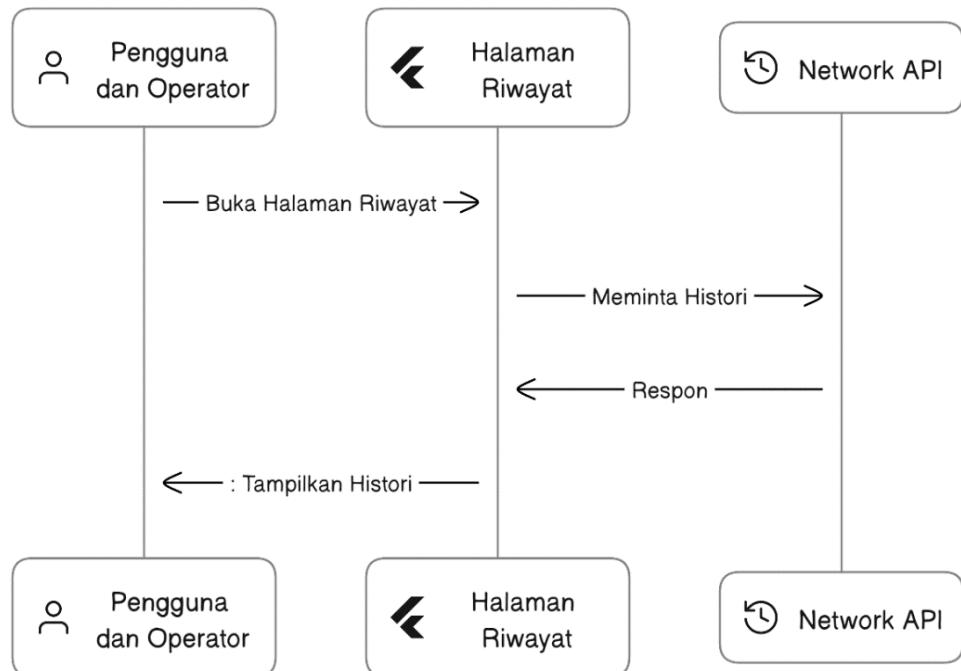
7. Tampilan Menu Daftar Kunci



Gambar 2.44 Diagram *sequence* tampilan menu daftar kunci

Gambar 2.44 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah ketika pengguna membuka menu daftar kunci untuk melihat data akses terkait kunci dalam sistem. Pengguna pertama kali membuka menu daftar kunci yang menampilkan daftar kunci yang ada. Selanjutnya, menu daftar kunci menghubungi *NetworkAPI* untuk mengambil data akses pengguna terkini dengan memanggil API '/my-access'. Setelah menerima permintaan, *NetworkAPI* meminta akses data dari API. API mengakses data yang diperlukan dan memberikannya kembali ke *NetworkAPI*. Selanjutnya, *NetworkAPI* memberikan data akses tersebut ke menu daftar kunci. Menu daftar kunci menampilkan data akses tersebut ke pengguna sehingga pengguna dapat melihat daftar kunci yang dimiliki. Proses tersebut memastikan bahwa pengguna dapat dengan mudah melihat informasi tentang akses kunci melalui menu daftar kunci tanpa perlu campur tangan manual. Dengan alur ini, pengguna dapat dengan cepat mengetahui status dan informasi penting terkait kunci yang dimilikinya dalam sistem.

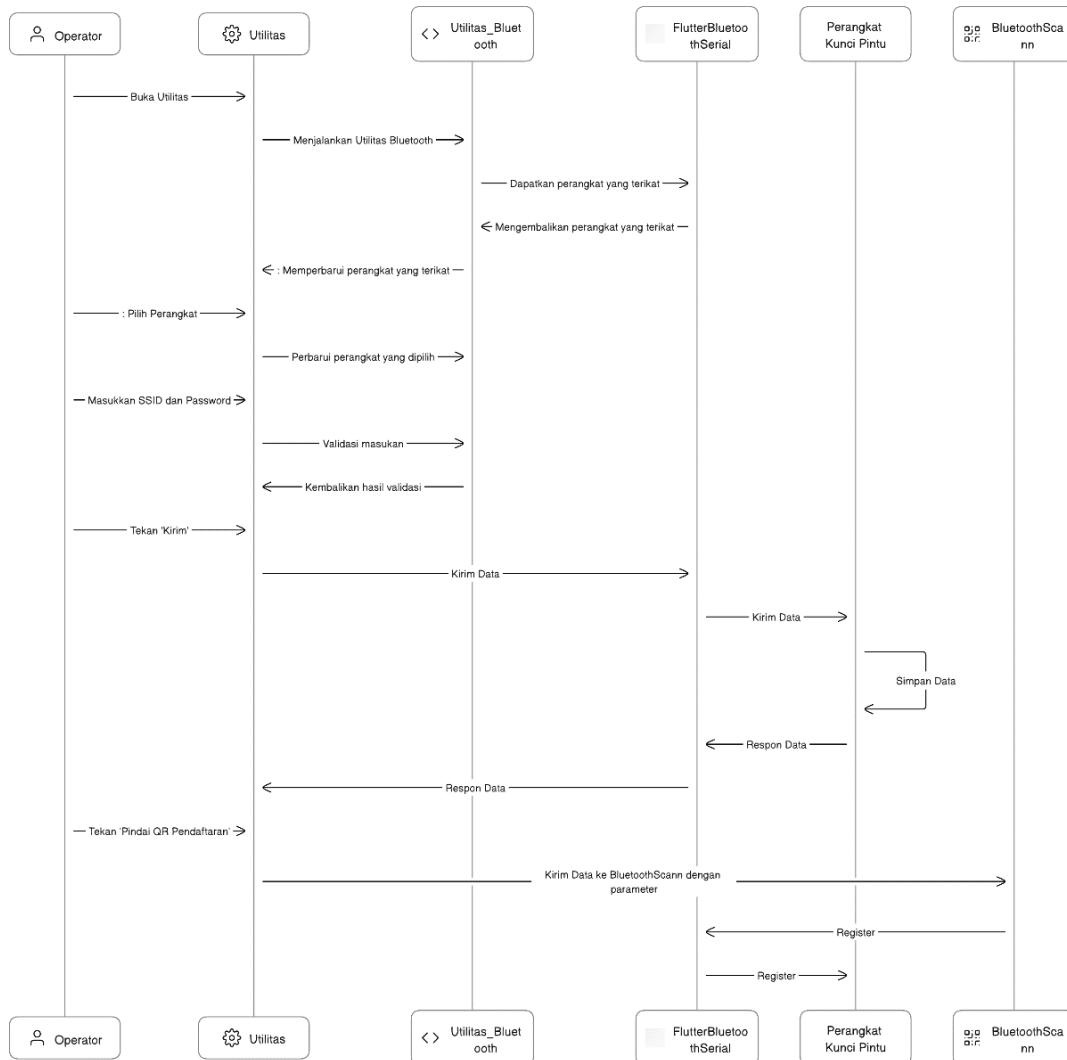
8. Tampilan Menu Riwayat Akses



Gambar 2.45 Diagram *sequence* tampilan halaman riwayat

Gambar 2.45 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah ketika pengguna atau operator membuka halaman riwayat untuk melihat histori aktivitas dalam sistem. Pertama, pengguna atau operator membuka halaman riwayat yang menampilkan daftar histori aktivitas yang ada. Halaman riwayat menghubungi *Network API* untuk meminta histori aktivitas. Setelah menerima permintaan, *Network API* memproses, mengambil data histori dari *server*, dan memberikan respons yang berisi data histori tersebut ke halaman riwayat. Halaman riwayat menampilkan data histori aktivitas ke pengguna atau operator, sehingga pengguna atau operator dapat melihat daftar aktivitas yang telah terjadi dalam sistem. Proses tersebut memastikan bahwa pengguna atau operator dapat dengan mudah melihat dan menelusuri histori aktivitasnya melalui halaman riwayat tanpa kesulitan. Dengan adanya alur tersebut, pengguna atau operator dapat dengan cepat memahami dan melacak aktivitas yang terjadi dalam sistem.

9. Tampilan Menu Utilitas

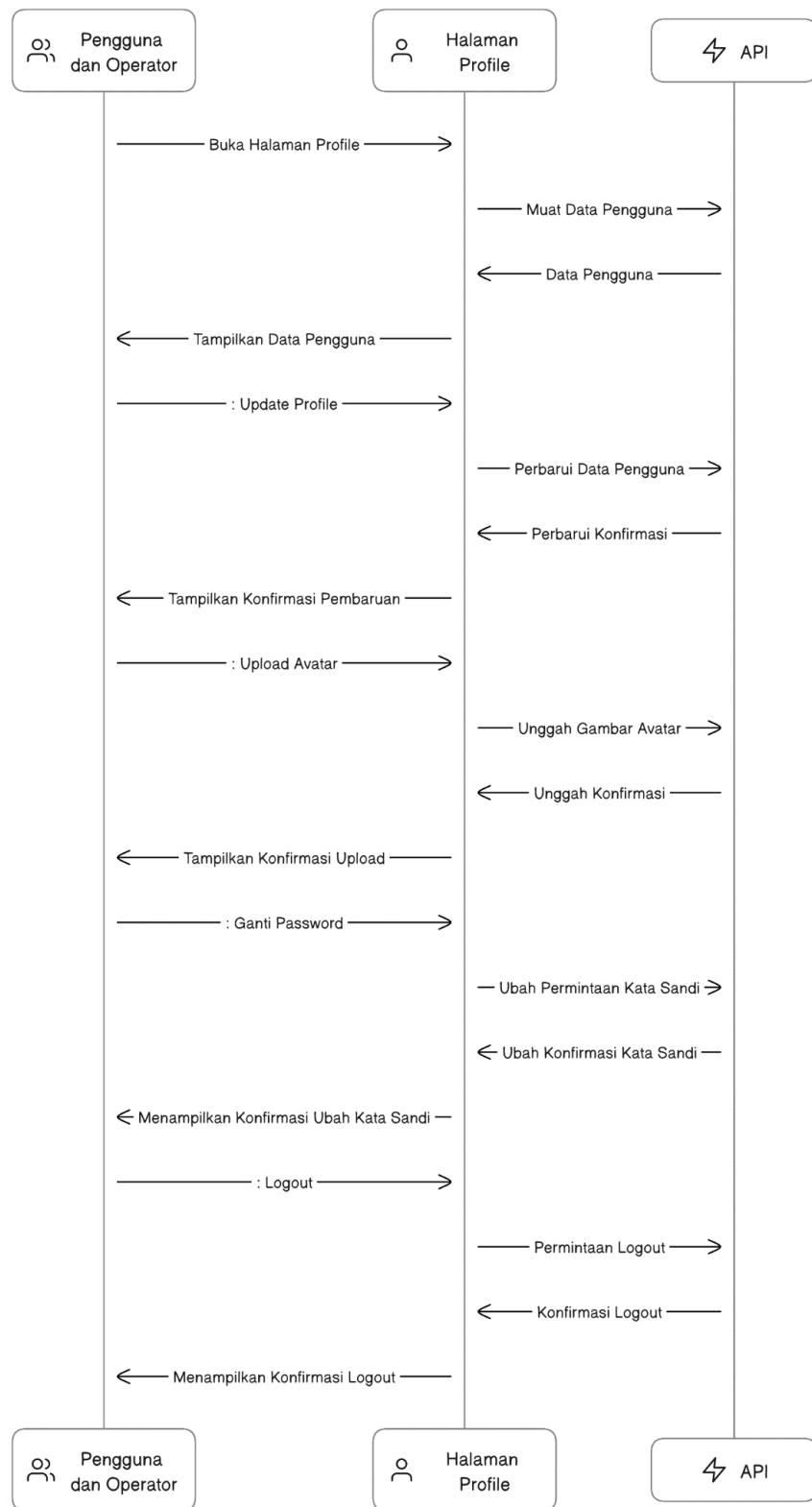


Gambar 2.46 Diagram *sequence* tampilan halaman utilitas

Gambar 2.46 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah yang terjadi ketika operator membuka menu utilitas dan menggunakan fitur utilitas *bluetooth* untuk menghubungkan dan mengatur perangkat kunci pintu. Pertama, operator membuka menu utilitas untuk mengakses fitur utilitas *bluetooth*. Selanjutnya, utilitas *bluetooth* berinteraksi dengan *flutter bluetooth serial* untuk mendapatkan daftar perangkat yang terikat. *Flutter bluetooth serial* mengembalikan daftar perangkat yang terikat ke utilitas *bluetooth*. Kemudian, operator memilih perangkat dari daftar dan menu utilitas meminta utilitas *bluetooth* untuk memperbarui perangkat yang dipilih. Utilitas *bluetooth* berkomunikasi dengan

operator untuk meminta SSID dan *password* untuk konfigurasi perangkat. Setelah operator memasukkan SSID dan *password*, utilitas *bluetooth* melakukan validasi masukan dan mengembalikan hasil validasi ke menu utilitas. Kemudian, operator menekan tombol kirim dan menu utilitas mengirimkan data melalui *flutter bluetooth serial*. *Flutter bluetooth serial* mengirim data ke perangkat kunci pintu untuk melakukan pengaturan. Perangkat kunci pintu menerima data dan menyimpannya. Kemudian, perangkat kunci pintu memberikan respons kembali melalui *flutter bluetooth serial*. *Flutter bluetooth serial* meneruskan respons ke menu utilitas dan operator menerima respons tersebut. Selanjutnya, operator menekan tombol pindai QR pendaftaran, dan menu utilitas mengirim data dengan parameter ke *bluetooth scann*. *Bluetooth scann* melakukan registrasi perangkat kunci pintu. *Flutter bluetooth serial* bertindak sebagai penghubung untuk melakukan registrasi dengan perangkat kunci pintu.

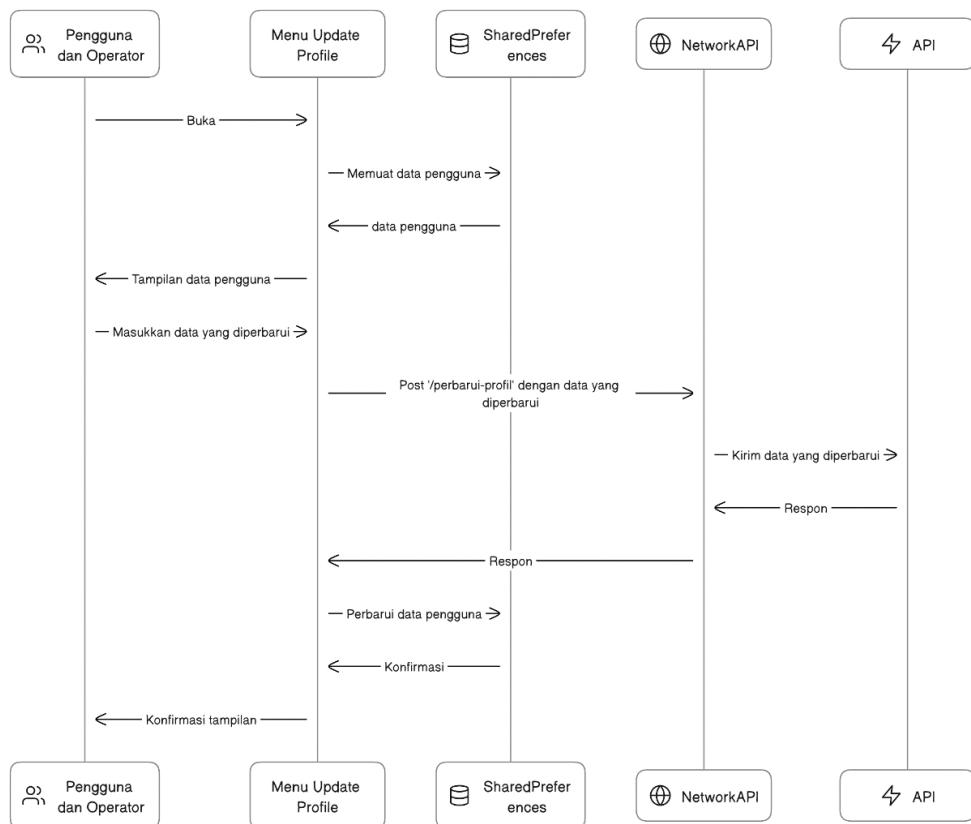
10. Tampilan Menu Profil



Gambar 2.47 Diagram *sequence* tampilan halaman *profile*

Gambar 2.47 di atas merupakan diagram *sequence* yang menggambarkan langkah-langkah yang terjadi ketika pengguna atau operator membuka halaman *profile* untuk mengelola profilnya dalam sistem. Pertama, pengguna atau operator membuka halaman *profile* yang menampilkan informasi profil. Halaman *profile* menghubungi API untuk memuat data pengguna dari *server*. Setelah menerima permintaan, API mengirimkan data pengguna kembali ke halaman *profile*. Halaman *profile* menampilkan data pengguna ke pengguna atau operator sehingga informasi profil ditampilkan. Selanjutnya, pengguna atau operator memiliki opsi untuk memperbarui profil. Jika memilih untuk melakukan perubahan, halaman *profile* akan menghubungi kembali API untuk memperbarui data pengguna. API mengonfirmasi perubahan data pengguna dan mengembalikan konfirmasi ke halaman *profile*. Halaman *profile* menampilkan konfirmasi pembaruan sehingga pengguna dapat melihat bahwa profilnya telah diperbarui. Selain itu, pengguna atau operator juga dapat mengunggah foto untuk profil. Jika memilih untuk melakukannya, halaman *profile* menghubungi API untuk mengunggah gambar foto. API mengonfirmasi proses unggah gambar foto dan mengembalikan konfirmasi ke halaman *profile*. Halaman *profile* menampilkan konfirmasi unggah foto sehingga pengguna atau operator dapat melihat bahwa foto telah berhasil diunggah. Selanjutnya, pengguna atau operator juga dapat mengganti kata sandi. Jika memilih untuk melakukannya, halaman *profile* menghubungi API untuk mengubah kata sandi. API mengonfirmasi proses perubahan kata sandi dan mengembalikan konfirmasi ke halaman *profile*. Halaman *profile* menampilkan konfirmasi perubahan kata sandi sehingga pengguna atau operator dapat melihat bahwa kata sandi telah berhasil diubah. Terakhir, pengguna atau operator dapat melakukan *logout* dari aplikasi. Jika memilih untuk keluar, halaman *profile* menghubungi API untuk melakukan permintaan *logout*. API mengonfirmasi proses *logout* dan mengembalikan konfirmasi ke halaman *profile*. Halaman *profile* menampilkan konfirmasi *logout* sehingga pengguna atau operator dapat melihat bahwa telah berhasil keluar dari aplikasi.

11. Update Profile

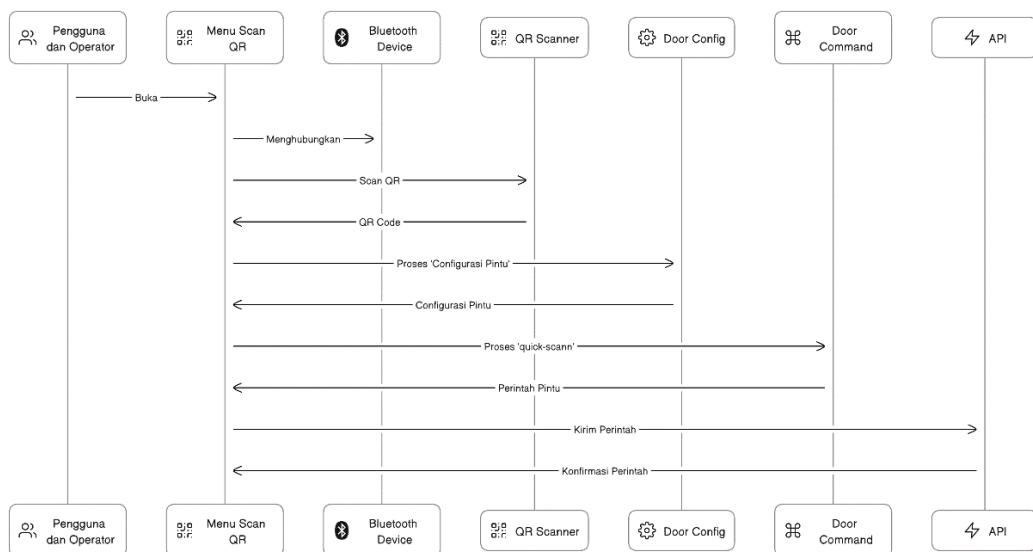


Gambar 2.48 Diagram *sequence* tampilan menu *update profile*

Gambar 2.48 di atas merupakan diagram yang menggambarkan langkah-langkah yang terjadi ketika pengguna atau operator membuka menu *update profile* untuk memperbarui profil dalam sistem. Pertama, pengguna atau operator membuka menu *update profile* yang memuat data pengguna dari *SharedPreferences*. Setelah data pengguna dimuat dari *SharedPreferences*, menu *update profile* menampilkan data tersebut ke menu *update profile* sehingga pengguna atau operator dapat melihat informasi profil yang sudah ada. Selanjutnya, pengguna atau operator memasukkan data yang ingin diperbarui pada menu *update profile*. Setelah data diperbarui, menu *update profile* menghubungi *NetworkAPI* untuk melakukan permintaan POST dengan data yang diperbarui menggunakan *endpoint* '/perbarui-profil'. *NetworkAPI* menerima permintaan dari menu *update profile* dan mengirimkan data yang telah diperbarui ke *API*. *API* memproses data tersebut dan memberikan respons kembali ke *NetworkAPI*. *NetworkAPI* meneruskan respons dari *API* ke menu *update profile*. Setelah menerima respons, menu *update profile* memperbarui data pengguna yang

ada di *SharedPreferences* untuk mencerminkan perubahan yang telah dilakukan. Setelah data pengguna diperbarui di *SharedPreferences*, menu *update profile* memberikan konfirmasi kepada pengguna atau operator bahwa data telah berhasil diperbarui.

12. Tampilan Menu Scan QR Code



Gambar 2.49 Diagram sequence tampilan menu scan QR Code

Gambar 2.49 di atas merupakan diagram yang menggambarkan langkah-langkah yang terjadi ketika pengguna atau operator membuka menu *scan QR* untuk melakukan proses *scan QR code* pada aplikasi. Pertama, pengguna atau operator membuka menu *scan QR* yang akan menghubungkan dengan perangkat *bluetooth*. Setelah terhubung dengan perangkat *bluetooth*, menu *scan QR* menggunakan *QR scanner* melakukan pemindaian *QR code*. *QR scanner* mengambil data dari *QR code* yang dipindai dan mengirimkan hasilnya kembali ke menu *scan QR*. Menu *scan QR* menerima data *QR code* dan menghubungkan ke *door config* untuk memproses '*configurasi pintu*' berdasarkan *QR code* yang dipindai. *Door config* melakukan proses konfigurasi pintu sesuai dengan data *QR code* dan mengembalikan hasilnya kembali ke menu *scan QR*. Selanjutnya, menu *scan QR* menggunakan *door command* untuk melakukan '*quick-scan*' atau pemindaian cepat pada pintu yang telah dikonfigurasi. *Door command* mengirimkan perintah pintu berdasarkan konfigurasi yang diterima dari *door config* dan mengembalikan

hasilnya kembali ke menu *scan QR*. Menu *scan QR* menggunakan API untuk mengirimkan perintah pintu yang telah dihasilkan oleh *door command*. API memproses perintah tersebut dan memberikan konfirmasi bahwa perintah pintu telah berhasil dikirimkan kembali ke menu *scan QR*.

3. PENUTUP

Dokumen B400 memaparkan proses implementasi yang dilakukan untuk membangun sistem keamanan kunci pintu gedung berbasis IoT. Proses implementasi akan menentukan hasil akhir dari sistem yang dikembangkan. Hasil implementasi juga akan dijadikan acuan untuk proses pengujian sistem selanjutnya.



Dokumen Pengembangan Produk Lembar Sampul Dokumen

Judul Dokumen

TUGAS AKHIR:
Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*

Jenis Dokumen

IMPLEMENTASI

Catatan: Dokumen ini dikendalikan penyebarannya oleh Dept. Teknik Elektro Undip

Nomor Dokumen

B500-01-TA2223.2.19012

Nomor Revisi

01

Nama File

B500-2-TA2223

Tanggal Penerbitan

8 September 2023

Unit Penerbit

Departemen Teknik Elektro Undip

Jumlah Halaman

15 (termasuk lembar sampul ini)

Data Pengusul				
Pengusul	Nama NIM	Henric Dhiki Wicaksono 21060119120011	Jabatan	Anggota
	Nama NIM	Novi Dianasari 21060119120039	Jabatan	Anggota
	Nama NIM	Muhammad Khoiril Wafi 21060119140133	Jabatan	Anggota
Pembimbing Utama	Nama NIP	M. Arfan, S.Kom., M.Eng. 198408172015041002	Tanda Tangan	
Pendamping	Nama NIP	Imam Santoso, S.T., M.T. 197012031997021001	Tanda Tangan	

DAFTAR ISI

Catatan Sejarah Perbaikan Dokumen.....	3
1. PENDAHULUAN.....	4
1.1 Ringkasan Isi Dokumen	4
1.2 Aplikasi Dokumen.....	4
1.3 Referensi.....	4
1.4 Daftar Singkatan.....	5
2. PENGUJIAN.....	6
2.1 Pengujian Fungsionalitas Komunikasi <i>Bluetooth</i>	6
2.2 Pengujian Performa Komunikasi <i>Bluetooth</i>	7
2.3 Pengujian Fungsionalitas Komunikasi WiFi	11
2.4 Pengujian Performa Komunikasi WiFi	12
2.5 Pengujian Fungsionalitas Penguncian.....	18
2.6 Pengujian Fungsionalitas API	23
2.7 Pengujian Performa API.....	36
2.8 Pengujian Fungsionalitas <i>Website</i>	39
2.9 Pengujian Performa Website	56
2.10 Pengujian Fungsionalitas Aplikasi <i>Mobile</i>	60
2.11 Pengujian Performa Aplikasi Mobile	71
3. PENUTUP	74

Catatan Sejarah Perbaikan Dokumen

VERSI, TGL, OLEH	PERBAIKAN
01, 8 September 2023, oleh Henric Dhiki Wicaksono, Novi Dianasari dan Muhammad Khoiril Wafi.	<i>Draft Dokumen B500</i>

1. PENDAHULUAN

1.1 Ringkasan Isi Dokumen

Dokumen ini berisikan proses pengujian yang dilakukan pada proses pengembangan “Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*” proses pengujian yang dilakukan meliputi pengujian secara fungsional dan pengujian untuk mengukur performa sistem. Dokumen ini menjelaskan metode serta alat yang digunakan dalam proses pengujian. Proses pengujian dilakukan untuk melakukan verifikasi dari hasil proses implementasi dengan spesifikasi dan standarisasi yang telah ditentukan pada proses desain.

1.2 Aplikasi Dokumen

Dokumen ini digunakan dalam proses pengembangan “Rancang Bangun Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*” untuk:

- 1) Sebagai penjelasan proses pengujian yang dilakukan.
- 2) Sebagai penjelasan mengenai metode dan alat yang digunakan dalam proses pengujian.
- 3) Sebagai acuan keberhasilan sistem sesuai dengan spesifikasi dan standarisasi yang telah ditentukan.
- 4) Sebagai dokumentasi dan pencatatan perubahan.

Dokumen B400 ini diajukan kepada dosen pembimbing tugas akhir dan tim tugas akhir Program Studi Sarjana Teknik Elektro Undip sebagai bahan penilaian tugas akhir.

1.3 Referensi

- [1] E. Novrizza Alam and F. Dewi, “Performance Testing Analysis of Bandungtanginas Application With Jmeter,” *Int. J. Innov. Enterp. Syst.*, vol. 6, no. 01, pp. 85–94, 2022.

1.4 Daftar Singkatan

Tabel 1.1 Daftar Singkatan

SINGKATAN	ARTI
IoT	<i>Internet of Things</i>
WiFi	<i>Wireless Fidelity</i>
JSON	<i>Javascript Object Notation</i>
PC	<i>Personal Computer</i>
UI	<i>User Interface</i>
QR-Code	<i>Quick Response Code</i>

2. PENGUJIAN

2.1 Pengujian Fungsionalitas Komunikasi *Bluetooth*

Pengujian komunikasi *bluetooth* bertujuan untuk melakukan pemeriksaan dan verifikasi fungsionalitas dari sistem komunikasi bluetooth yang sudah diimplementasikan pada perangkat kunci pintu. Pengujian dilakukan dengan menggunakan aplikasi *serial bluetooth terminal*. Aplikasi *serial bluetooth terminal* merupakan sebuah aplikasi yang menyediakan sebuah terminal yang dapat digunakan untuk mengirimkan dan menerima data dengan menggunakan komunikasi *bluetooth*. Proses pengujian fungsionalitas *bluetooth* dilakukan dalam 3 bagian yaitu :

1. Pengujian koneksi

Pengujian koneksi dilakukan dengan melakukan *pairing* pada aplikasi *serial bluetooth terminal*. Hasil proses *pairing* dapat dilihat pada gambar 2.1 di bawah.



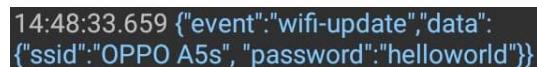
Connecting to HC-05 ...
Connected

Gambar 2.1 Status *smartphone* terhubung ke HC-05

Berdasarkan Gambar 2.1 di atas, proses *pairing* berhasil dilakukan dengan status koneksi pada aplikasi *serial bluetooth terminal* yaitu “Connected”.

2. Pengujian pengiriman data

Pengujian pengiriman data dilakukan dengan mengirimkan data JSON yang disajikan dalam sebuah *string*, data yang dikirimkan sesuai dengan format data pada perangkat kunci pintu. Hasil dari proses pengujian pengiriman data dapat dilihat pada Gambar 2.2 di bawah.



```
14:48:33.659 {"event": "wifi-update", "data": {"ssid": "OPPO A5s", "password": "helloworld"} }
```

Gambar 2.2 Pengujian pengiriman data dari *smartphone* ke modul *bluetooth*

Pada Gambar 2.2 di atas, data `{ "event": "wifi-update", "data": { "ssid": "OPPO A5s", "password": "helloworld" } }` berhasil terkirim ke perangkat kunci pintu ditandai dengan pesan yang berwarna biru.

3. Pengujian balasan data

Pengujian balasan data dilakukan dengan menunggu balasan dari setiap perintah yang dikirimkan dari *smartphone* ke perangkat kunci pintu. Hasil dari pengujian balasan data dapat dilihat pada Gambar 2.3 di bawah.

```
14:48:33.779 {"event":"wifi-update","data":  
{"ssid":"OPPO A5s", "password":"helloworld"}}
```

Gambar 2.3 Pengujian balasan data dari modul *bluetooth* ke *smartphone*

Pada Gambar 2.3 di atas, terlihat bahwa data { "event": "wifi-update", "data": { "ssid": "OPPO A5s", "password": "helloworld" } } berhasil diterima oleh aplikasi dan sesuai dengan balasan yang diharapkan.

Hasil pengujian di atas membuktikan bahwa komunikasi data dua arah melalui *bluetooth* yang diuji coba menggunakan aplikasi *serial bluetooth terminal* berjalan dengan baik. Perangkat *bluetooth* mampu menerima data dari *smartphone* dan mengirimkan balasan kembali ke *smartphone* melalui koneksi *bluetooth*. Hasil pengujian menunjukkan bahwa protokol komunikasi dan konfigurasi *serial* telah berhasil diatur dengan benar. Selama pengujian tersebut, tidak ada gangguan atau kesalahan dalam komunikasi yang terdeteksi. Pesan dapat dikirim dengan sukses dan balasan diterima dengan benar. Hal tersebut menunjukkan keandalan komunikasi antara perangkat *bluetooth* dan *smartphone*.

2.2 Pengujian Performa Komunikasi *Bluetooth*

Pada pengujian kinerja komunikasi data dua arah melalui *bluetooth* ini dilakukan dengan memperhatikan parameter QoS yang didefinisikan oleh TIPHON. Pengujian ini bertujuan untuk mengukur dan menganalisis karakteristik kinerja komunikasi *bluetooth*, termasuk *delay*, *throughput*, dan *packet loss*. Pengujian yang dilakukan yaitu :

1. Pengujian *delay*

Pengujian *delay* dilakukan dengan cara mengirimkan paket data dari *smartphone* ke modul *bluetooth* melalui koneksi *Bluetooth*. Kemudian, mencatat waktu yang dibutuhkan untuk paket data mencapai modul *bluetooth* setelah dikirim dari

smartphone. Pengujian ini dilakukan beberapa kali dan rerata *delay* dihitung. Hasil pengujian *delay* komunikasi *bluetooth* ditunjukkan pada Tabel 2.1 di bawah.

Tabel 2.1 Hasil pengujian *delay* komunikasi *bluetooth*

Jarak	Waktu A	Waktu B	RTT	Delay	Keterangan
1 m	15:24:43.762	15:24:44.304	542 ms	271 ms	Bagus
	15:24:56.354	15:24:56.708	354 ms	177 ms	
	15:25:00.749	15:25:00.858	109 ms	54,5 ms	
Rata-rata <i>delay</i> (ms)		167,5			
5 m	15:29:20.865	15:29:20.994	129 ms	64,5 ms	Sangat Bagus
	15:29:24.604	15:29:24.722	118 ms	59 ms	
	15:29:27.581	15:29:27.742	161 ms	80,5 ms	
Rata-rata <i>delay</i> (ms)		68 ms			
10 m	15:32:05.030	15:32:05.174	144 ms	72 ms	Sangat Bagus
	15:32:08.974	15:32:09.095	121 ms	60,5 ms	
	15:32:11.982	15:32:12.109	127 ms	63,5 ms	
Rata-rata <i>delay</i> (ms)		65,3 ms			

Keterangan : Waktu A = waktu ketika data dikirim
 Waktu B = waktu ketika data diterima kembali oleh pengirim
 RTT = *Round-Trip Time*

Dari Tabel 2.1 hasil pengujian *delay* komunikasi *bluetooth* antara *smartphone* ke modul *bluetooth* terjadi penurunan rata-rata *delay* seiring dengan meningkatnya jarak. Pada jarak 1 m rata-rata hasil dari pengujian *delay* komunikasi *bluetooth* untuk tiga kali percobaan adalah sebesar 167,5 ms sehingga tergolong bagus. Pada jarak 5 m rata-rata hasil dari pengujian *delay* komunikasi *bluetooth* untuk tiga kali percobaan adalah sebesar 68 ms sehingga tergolong sangat bagus. Pada jarak 10 m rata-rata hasil dari pengujian *delay* komunikasi *bluetooth* untuk tiga kali percobaan

adalah sebesar 65,3 ms sehingga tergolong sangat bagus. Dengan demikian, *delay* tertinggi ditunjukkan pada saat pengujian pada jarak 1 m dan terendah pada jarak 10 m. Hal tersebut bisa saja terjadi karena faktor sinyal yang tidak stabil ataupun adanya proses pada perangkat sehingga terjadi waktu tunda pada sistem.

2. Pengujian *throughput*

Pengujian *throughput* dilakukan untuk mengetahui kecepatan rata-rata transfer data. Untuk mengukur *throughput*, dilakukan dengan cara mengirimkan sejumlah besar data dari *smartphone* ke modul *bluetooth* dalam satu sesi komunikasi *bluetooth*. Kemudian, mencatat jumlah data yang berhasil dikirimkan dan waktu yang diperlukan untuk mengirimkan data tersebut. Dari informasi tersebut, *throughput* (jumlah data per satuan waktu) dihitung. Hasil pengujian *throughput* komunikasi *bluetooth* ditunjukkan pada Tabel 2.2 di bawah.

Tabel 2.2 Hasil Pengujian *throughput* komunikasi *bluetooth*

Jarak (m)	Total Paket (bit)	Waktu Pengukuran (s)	Throughput (Kbps)
1 m	592	0,271	2,184
5 m	592	0,0645	9,178
10 m	592	0,072	8,222

Pada Tabel 2.2 hasil pengujian *throughput* komunikasi *bluetooth* dapat dilihat bahwa hasil *throughput* paling tinggi pada jarak 5 m dan paling rendah pada jarak 1 m. Pada jarak 1 m hasil dari pengujian *throughput* komunikasi *bluetooth* sebesar 2,184 Kbps. Pada jarak 5 m hasil dari pengujian *throughput* komunikasi *bluetooth* sebesar 9,178 Kbps. Pada jarak 10 m hasil dari pengujian *throughput* komunikasi *bluetooth* sebesar 8,222 Kbps.

3. Pengujian *packet loss*

Pengujian *packet loss* dilakukan untuk mengetahui apakah terjadi *loss* pada sistem yang dibuat ataukah tidak. Untuk mengukur *packet loss*, dilakukan dengan cara mengirimkan sejumlah paket data dari *smartphone* ke modul *bluetooth*. Kemudian, mencatat jumlah paket yang berhasil dikirimkan dan jumlah paket yang hilang atau

tidak diterima oleh modul *bluetooth*. Dari informasi tersebut, *packet loss rate* (persentase paket yang hilang) dihitung. Hasil pengujian *packet loss* komunikasi *bluetooth* dapat dilihat pada Tabel 2.3 di bawah.

Tabel 2.3 Hasil pengujian *packet loss* komunikasi *bluetooth*

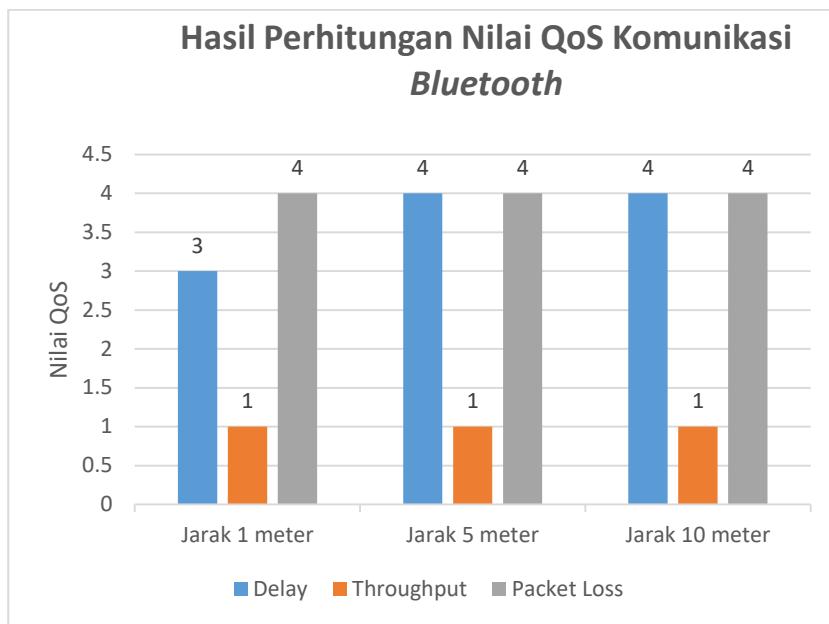
Jarak	Data Terkirim	Data Diterima	Persentase Loss
1 m	3	3	0%
5 m	3	3	0%
10 m	3	3	0%

Dari Tabel 2.3 hasil pengujian *packet loss* komunikasi *bluetooth* terlihat bahwa pada jarak pengujian sampai 10 m tidak terjadi *packet loss*, semua data pada pengujian yang dikirim sebanyak 3 kali pada setiap pengujian dapat diterima dengan baik yaitu 3 data, sehingga perangkat dapat berkomunikasi dengan baik sampai pada jarak 10 meter.

Berdasarkan pengujian yang telah dilakukan maka selanjutnya dilakukan analisa QoS dengan cara menghitung nilai *delay*, *throughput*, dan *packet loss* terhadap perubahan jarak agar didapatkan nilai QoS seperti yang ditunjukkan pada Tabel 2.4 di bawah.

Tabel 2.4 Indeks parameter QoS komunikasi *bluetooth*

Jarak	Delay	Throughput	Packet Loss	Nilai QoS	Indeks
1 m	3	1	4	2,67	Kurang Memuaskan
5 m	4	1	4	3	Memuaskan
10 m	4	1	4	3	Memuaskan



Gambar 2.4 Grafik hasil perhitungan nilai QoS komunikasi *bluetooth*

Berdasarkan Tabel 2.4 dan Gambar 2.4 dapat dilihat bahwa hasil dari pengujian parameter QoS komunikasi *bluetooth* seperti *delay*, *throughput*, dan *packet loss* dihitung dan dianalisis sesuai dengan nilai pada indeks parameter QoS. Hasil yang didapat bahwa pada jarak 1 m memiliki *indeks* kurang memuaskan dengan nilai QoS sebesar 2,67, pada jarak 5 m memiliki indeks memuaskan dengan nilai QoS sebesar 3, dan pada jarak 10 m memiliki indeks memuaskan dengan nilai QoS sebesar 3.

2.3 Pengujian Fungsionalitas Komunikasi WiFi

Pada pengujian ini, digunakan aplikasi *serial monitor* dalam Arduino IDE untuk menguji fungsionalitas komunikasi data dua arah melalui WiFi. Tujuan utamanya adalah untuk mengirimkan data dari ESP32 ke *server* dan menerima balasan dari *server* kembali ke ESP32 melalui koneksi WiFi. Proses pengujian fungsionalitas WiFi dilakukan dalam 2 bagian yaitu :

1. Pengujian koneksi ke *router*

```

pengujian komunikasi data dua arah melalui wifi
menghubungkan ...
terhubung
waktu yang dibutuhkan: 3105 ms

```

Gambar 2.5 Status perangkat WiFi terhubung ke jaringan WiFi

Berdasarkan pada Gambar 2.5, proses pengujian koneksi dilakukan dengan menghubungkan perangkat WiFi (ESP32) ke jaringan WiFi yang tersedia. Kemudian, status koneksi WiFi “Terhubung” ditampilkan pada aplikasi *serial monitor* dalam Arduino IDE. Adapun waktu yang dibutuhkan ESP32 untuk terhubung ke jaringan WiFi adalah 3105 ms.

2. Pengujian pengiriman dan penerimaan data

```
pengujian fungsionalitas wifi untuk komunikasi data dua arah
mengirim data
data diterima: {"date":"2023-08-09","time":"15:19:04"}
waktu yang dibutuhkan: 9625 ms
```

Gambar 2.6 Pengujian pengiriman dan penerimaan data

Berdasarkan pada Gambar 2.6, hasil pengujian pengiriman dan penerimaan data menggunakan aplikasi *serial monitor* dalam Arduino IDE memperlihatkan bahwa ESP32 mengirim data ke *server*. Kemudian, data diterima kembali oleh ESP32 pada `{"date":"2023-08-09","time":"15:19:04"}`. Adapun waktu yang dibutuhkan dalam proses pengiriman dan penerimaan data tersebut adalah 9625 ms.

Hasil pengujian tersebut berhasil membuktikan bahwa komunikasi data dua arah melalui WiFi yang diuji menggunakan aplikasi *serial monitor* dalam Arduino IDE berjalan dengan baik. *Server* mampu menerima data dari ESP32 melalui koneksi WiFi dan mengirimkan balasan kembali ke ESP32 melalui jalur yang sama. Pesan berhasil dikirim dan diterima, serta balasan berhasil diterima dan ditampilkan di *Serial Monitor* komputer.

2.4 Pengujian Performa Komunikasi WiFi

Pada pengujian performa kinerja komunikasi data dua arah melalui WiFi ini dengan memperhatikan parameter QoS yang didefinisikan oleh TIPHON. Pengujian ini bertujuan untuk mengukur dan menganalisis karakteristik kinerja komunikasi WiFi, termasuk *delay*, *throughput*, dan *packet loss*. Pengujian yang dilakukan yaitu :

1. Pengujian *delay*

Pengujian *delay* dilakukan dengan cara mengirimkan paket data dari ESP32 ke *server* melalui koneksi WiFi. Kemudian, mencatat waktu yang dibutuhkan untuk paket data mencapai *server* setelah dikirim dari ESP32. Pengukuran ini dilakukan beberapa kali dan rerata *delay* dihitung. Hasil pengujian *delay* komunikasi WiFi ditunjukkan pada Tabel 4.5 di bawah.

Tabel 2.5 Hasil Pengujian *delay* komunikasi WiFi

Kecepatan Internet (Mbps)	Waktu A	Waktu B	RTT (ms)	Delay (ms)	Ket.
U = 11.90 D = 16.63	17:39:22.808	17:39:24.885	2077	1038,5	
	17:41:36.089	17:41:37.756	1667	833,5	
	17:43:02.389	17:43:04.408	2019	1009,5	Buruk
	Rata-rata <i>delay</i> (ms)			960,5	
U = 41.97 D = 62.93	07:17:04.068	07:17:05.303	1235	617,5	
	07:24:38.336	07:24:39.568	1232	616	
	07:28:05.561	07:28:06.815	1254	627	Buruk
	Rata-rata <i>delay</i> (ms)			620,167	
U = 2.82 D = 8.30	09:35:04.975	09:35:07.876	2901	1450,5	
	09:36:32.482	09:36:35.992	3510	1755	
	09:37:57.434	09:37:59.472	2038	1019	Buruk
	Rata-rata <i>delay</i> (ms)			1408,167	
U = 17.52 D = 15.14	10:30:19.570	10:30:21.237	1667	833,5	
	10:33:17.431	10:33:18.943	1512	756	
	10:34:28.657	10:34:30.289	1632	816	Buruk
	Rata-rata <i>delay</i> (ms)			801,83	

Tabel 2.5 (lanjutan)

Kecepatan	Internet	Waktu A	Waktu B	RTT (ms)	Delay (ms)	Ket.
	(Mbps)					
		10:56:44.493	10:56:46.075	1582	791	
U = 13.86		10:58:27.993	10:58:29.344	1351	675,5	
D = 9.99		10:59:54.064	10:59:56.081	2017	1008,5	Buruk
		Rata-rata <i>delay</i> (ms)			825	

Ket.: Waktu A = Waktu ketika data dikirim
 Waktu B = Waktu ketika data diterima kembali oleh pengirim
 RTT = *Round-Trip Time*
 U = *Upload*
 D = *Download*

Dari Tabel 2.5 hasil pengujian *delay* komunikasi WiFi antara ESP32 ke *server* terjadi kenaikan atau penurunan rata-rata *delay* seiring dengan meningkat atau menurunnya kecepatan internet. Pada kecepatan internet sebesar (U = 11.90 Mbps dan D = 16.63 Mbps), rata-rata hasil dari pengujian *delay* komunikasi WiFi untuk tiga kali percobaan adalah sebesar 960,5 ms sehingga tergolong buruk. Pada kecepatan internet sebesar (U = 41.97 Mbps dan D = 62.93 Mbps), rata-rata hasil dari pengujian *delay* komunikasi WiFi untuk tiga kali percobaan adalah sebesar 620,167 ms sehingga tergolong buruk. Pada kecepatan internet sebesar (U = 2.82 Mbps dan D = 8.30 Mbps), rata-rata hasil dari pengujian *delay* komunikasi WiFi untuk tiga kali percobaan adalah sebesar 1408,167 ms sehingga tergolong buruk. Pada kecepatan internet sebesar (U = 17.52 Mbps dan D = 15.14 Mbps), rata-rata hasil dari pengujian *delay* komunikasi WiFi untuk tiga kali percobaan adalah sebesar 801,83 ms sehingga tergolong buruk. Pada kecepatan internet sebesar (U = 13.86 Mbps dan D = 9.99 Mbps), rata-rata hasil dari pengujian *delay* komunikasi WiFi untuk tiga kali percobaan adalah sebesar 825 ms sehingga tergolong buruk.

Dengan demikian dapat disimpulkan bahwa dengan adanya kecepatan internet yang semakin meningkat, maka *delay* semakin menurun.

2. Pengujian *throughput*

Pengujian *throughput* dilakukan untuk mengetahui kecepatan rata-rata transfer data. Untuk mengukur *throughput*, dilakukan dengan cara mengirimkan sejumlah besar data dari ESP32 ke *server* dalam satu sesi komunikasi WiFi. Kemudian, mencatat jumlah data yang berhasil dikirimkan dan waktu yang diperlukan untuk mengirimkan data tersebut. Dari informasi tersebut, *throughput* (jumlah data per satuan waktu) dihitung. Hasil pengujian *throughput* komunikasi WiFi ditunjukkan pada Tabel 2.6 di bawah.

Tabel 2.6 Hasil pengujian *throughput* komunikasi WiFi

Kecepatan Internet (Mbps)	Total Paket (Bytes)	Waktu Pengukuran (s)	Throughput (Mbps)
U = 11.90 D = 16.63	882523	1,0385	6,79836
U = 41.97 D = 62.93	882523	0,6175	11,43349
U = 2.82 D = 8.30	882523	1,4505	4,86741
U = 17.52 D = 15.14	882523	0,8335	8,47052
U = 13.86 D = 9.99	882523	0,791	8,92564

Pada Tabel 2.6 hasil pengujian *throughput* komunikasi WiFi dapat dilihat bahwa terjadi kenaikan atau penurunan *throughput* seiring dengan meningkat atau menurunnya kecepatan internet. Pada kecepatan internet sebesar (U = 11.90 Mbps dan D = 16.63 Mbps), hasil dari pengujian *throughput* komunikasi WiFi adalah

sebesar 6,79836 Mbps. Pada kecepatan internet sebesar ($U = 41.97$ Mbps dan $D = 62.93$ Mbps), hasil dari pengujian *throughput* komunikasi WiFi adalah sebesar 11,43349 Mbps. Pada kecepatan internet sebesar ($U = 2.82$ Mbps dan $D = 8.30$ Mbps), hasil dari pengujian *throughput* komunikasi WiFi adalah sebesar 4,86741 Mbps. Pada kecepatan internet sebesar ($U = 17.52$ Mbps dan $D = 15.14$ Mbps), hasil dari pengujian *throughput* komunikasi WiFi adalah sebesar 8,47052 Mbps. Pada kecepatan internet sebesar ($U = 13.86$ Mbps dan $D = 9.99$ Mbps), hasil dari pengujian *throughput* komunikasi WiFi adalah sebesar 8,92564 Mbps. Dengan demikian dapat disimpulkan bahwa dengan adanya kecepatan internet yang semakin meningkat, maka *throughput* juga semakin meningkat.

3. Pengujian *packet loss*

Pengujian *packet loss* dilakukan untuk mengetahui apakah terjadi *loss* pada sistem yang dibuat ataukah tidak. Untuk mengukur *packet loss*, dilakukan dengan cara mengirimkan sejumlah paket data dari ESP32 ke *server*. Kemudian, mencatat jumlah paket yang berhasil dikirimkan dan jumlah paket yang hilang atau tidak diterima oleh *server*. Dari informasi tersebut, *packet loss rate* (persentase paket yang hilang) dihitung. Hasil pengujian *packet loss* komunikasi WiFi dapat dilihat pada Tabel 2.7 di bawah.

Tabel 2.7 Hasil pengujian *packet loss* komunikasi WiFi

Kecepatan Internet (Mbps)	Data Terkirim	Data Diterima	Persentase Loss
$U = 11.90$ $D = 16.63$	3	3	0%
$U = 41.97$ $D = 62.93$	3	3	0%
$U = 2.82$ $D = 8.30$	3	3	0%
$U = 17.52$ $D = 15.14$	3	3	0%

Tabel 2.7 (lanjutan)

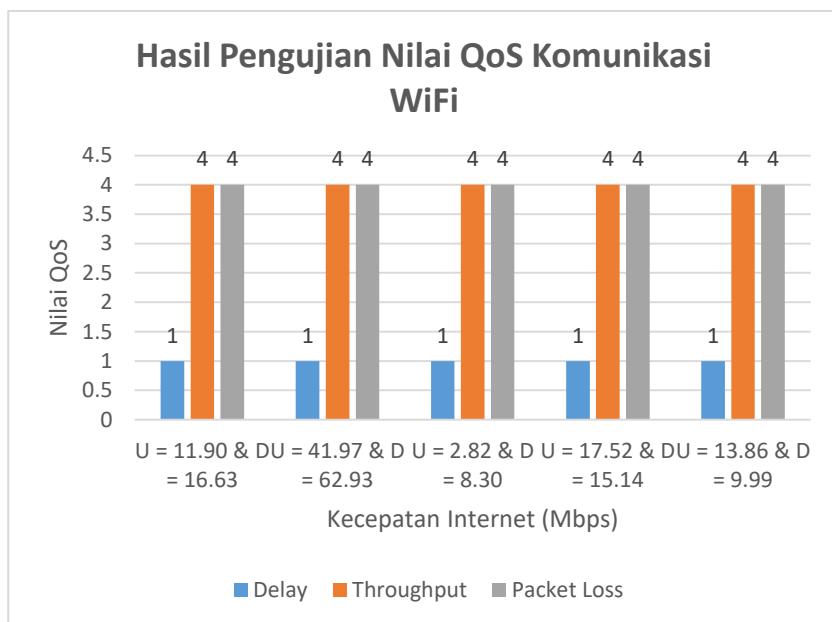
Kecepatan Internet (Mbps)	Data Terkirim	Data Diterima	Percentase Loss
$U = 13.86$	3	3	0%
$D = 9.99$			

Dari Tabel 2.7 hasil pengujian *packet loss* komunikasi WiFi terlihat bahwa semua data pada pengujian yang dikirim sebanyak 3 kali pada setiap pengujian dapat diterima dengan baik yaitu 3 data untuk beberapa variasi kecepatan internet. Dengan demikian dapat disimpulkan bahwa perangkat dapat berkomunikasi dengan baik.

Berdasarkan pengujian yang telah dilakukan maka selanjutnya dilakukan analisa QoS dengan menghitung nilai *delay*, *throughput*, dan *packet loss* terhadap perubahan kecepatan internet agar didapatkan nilai QoS seperti yang ditunjukkan pada Tabel 2.8 di bawah.

Tabel 2.8 Indeks parameter QoS komunikasi WiFi

Kecepatan Internet (Mbps)	Delay	Throughput	Packet Loss	Nilai QoS	Indeks
$U = 11.90$	1	4	4	3	Memuaskan
$D = 16.63$					
$U = 41.97$	1	4	4	3	Memuaskan
$D = 62.93$					
$U = 2.82$	1	4	4	3	Memuaskan
$D = 8.30$					
$U = 17.52$	1	4	4	3	Memuaskan
$D = 15.14$					
$U = 13.86$	1	4	4	3	Memuaskan
$D = 9.99$					



Gambar 2.7 Grafik hasil pengujian nilai QoS komunikasi WiFi

Berdasarkan Tabel 2.8 dan Gambar 2.7 dapat dilihat bahwa hasil dari pengujian parameter QoS komunikasi WiFi seperti *delay*, *throughput*, dan *packet loss* dihitung dan dianalisis sesuai dengan nilai pada indeks parameter QoS. Dapat disimpulkan bahwa semua percobaan dengan kecepatan internet yang berbeda memiliki hasil nilai QoS sebesar 3 yang berarti memuaskan.

2.5 Pengujian Fungsionalitas Penguncian

Pengujian fungsional perangkat penguncian ini dengan menggunakan metode *blackbox* untuk membantu mengidentifikasi bagaimana perangkat penguncian berfungsi dari perspektif pengguna atau pemakai, tanpa perlu mengetahui detail teknis internal perangkat. Hasil pengujian ini memberikan pemahaman tentang sejauh mana perangkat mendukung kebutuhan dan fungsionalitas yang telah ditetapkan.

Pengujian melibatkan penggunaan antarmuka yang tersedia (seperti aplikasi *mobile* atau antarmuka *website*) untuk mengendalikan perangkat penguncian. Perintah-perintah yang berkaitan dengan perangkat penguncian diuji untuk memastikan perangkat berfungsi sesuai harapan. Tabel 2.9 berikut ini adalah hasil pengujian yang dilakukan.

Tabel 2.9 Hasil pengujian fungsional perangkat penguncian

No.	Komponen	Skenario	Hasil yang Diharapkan	Keterangan
1.	Solenoid	Memberikan perintah kunci pintu dari website	Mengunci Pintu: Ketika perangkat mendapatkan perintah untuk mengunci pintu, solenoid akan di-set ke LOW. Ini akan menghasilkan gaya magnetik yang akan mengunci pintu secara fisik.	Berhasil
2.	Solenoid	Memberikan perintah membuka pintu melalui penggunaan aplikasi mobile, website, push button, dan touch sensor.	Membuka Pintu: Ketika perangkat mendapatkan perintah untuk membuka pintu, solenoid akan diaktifkan (HIGH). Hal ini akan menghilangkan gaya magnetik dan memungkinkan pintu untuk dibuka.	Berhasil
3.	Solenoid	Mencoba membuka kunci pintu secara paksa	Pengaman Pintu: Solenoid juga digunakan untuk memberikan pengamanan tambahan. Jika pintu terdeteksi terbuka tanpa adanya autentikasi yang sah (seperti perintah dari server atau tindakan tombol), solenoid akan diaktifkan untuk mengunci pintu kembali.	Berhasil

Tabel 2.9 (lanjutan)

No.	Komponen	Skenario	Hasil yang Diharapkan	Keterangan
4.	Solenoid	Memberikan perintah penjadwalan buka kunci pintu melalui website	Pengendalian Jadwal: Solenoid juga digunakan untuk mengendalikan pintu berdasarkan jadwal. Jika jadwal kunci pintu berakhir, solenoid akan diaktifkan untuk mengunci pintu sesuai dengan pengaturan jadwal.	Berhasil
5.	Push Button	Menekan tombol pada perangkat penguncian	Membuka Pintu Manual: Ketika tombol ditekan, program akan mengenali aksi tersebut dan membuka pintu secara manual (<i>unlock</i>) jika kondisi memungkinkan. Ini memungkinkan pengguna untuk membuka pintu secara langsung tanpa harus menggunakan autentikasi tambahan.	Berhasil
6.	Saklar Magnetik	Melakukan aksi buka tutup pintu	Deteksi Status Pintu: Sensor magnetik akan mendeteksi apakah pintu dalam keadaan terbuka atau tertutup berdasarkan perubahan medan magnet. Jika pintu terbuka, sensor akan memberikan sinyal LOW (0), dan jika pintu tertutup, sensor akan memberikan sinyal HIGH (1).	Berhasil

Tabel 2.9 (lanjutan)

No.	Komponen	Skenario	Hasil yang Diharapkan	Keterangan
7.	Saklar Magnetik	Melakukan aksi buka tutup pintu dan melihat status kondisi pintu melalui aplikasi <i>mobile</i> atau <i>website</i> .	Update Status Pintu: Program akan membaca nilai dari pin dan yang terhubung dengan sensor magnetik secara berkala. Jika pintu memberikan sinyal LOW, program akan menganggap pintu dalam keadaan terbuka, dan jika sensor memberikan sinyal HIGH, program akan menganggap pintu dalam keadaan tertutup.	Berhasil
8.	<i>Buzzer</i>	Menekan tombol dengan cepat, membuka pintu secara paksa, membuka pintu terlalu lama dalam mode penguncian.	Tanda Bunyi Notifikasi: <i>Buzzer</i> akan memberikan tanda bunyi sebagai respons terhadap berbagai peristiwa.	Berhasil
9.	<i>Buzzer</i>	Melakukan buka tutup kunci pintu melalui aplikasi <i>mobile</i> atau <i>website</i> .	Indikator Status: <i>Buzzer</i> dapat digunakan sebagai indikator status untuk memberi tahu pengguna tentang kondisi sistem.	Berhasil
10.	<i>Buzzer</i>	Membuka pintu dalam waktu lama.	Tanda Bunyi Pengingat: <i>Buzzer</i> dapat digunakan sebagai pengingat.	Berhasil

Tabel 2.9 (lanjutan)

No.	Komponen	Skenario	Hasil yang Diharapkan	Keterangan
11.	LED	Menghubungkan perangkat penguncian ke jaringan WiFi.	Indikasi Status WiFi: LED dapat berkedip dengan pola tertentu untuk menunjukkan status koneksi WiFi. Kedipan LED menunjukkan apakah perangkat terhubung ke jaringan WiFi atau tidak.	Berhasil
12.	LED	Memberikan perintah melalui aplikasi <i>mobile</i> atau <i>website</i> agar penguncian berkomunikasi dengan <i>server</i> .	Indikasi Status Data: LED dapat berkedip dengan pola tertentu untuk menunjukkan aktivitas data atau komunikasi agar perangkat dengan <i>server</i> .	Berhasil
13.	LED	Memberikan perintah buka tutup kunci pintu melalui aplikasi <i>mobile</i> atau <i>website</i> .	Indikasi Status Lock: LED dapat berkedip dengan pola tertentu untuk menunjukkan status penguncian pintu atau apakah solenoid sedang aktif atau tidak.	Berhasil
14.	Sensor Sentuh	Melakukan aksi memegang gagang pintu saat adanya penjadwalan pintu.	Deteksi Sentuhan: Komponen <i>touch sensor</i> akan mendeteksi apakah ada sentuhan atau sentuhan jari pada area yang diindikasikan. Jika ada sentuhan, nilai sensor akan bervariasi sesuai dengan intensitas sentuhan.	Berhasil

Tabel 2.9 (lanjutan)

No.	Komponen	Skenario	Hasil yang Diharapkan	Keterangan
15.	Sensor Sentuh	Melakukan aksi memegang gagang pintu saat adanya penjadwalan pintu.	Pengendalian Nilai sensor sentuh akan dipantau dan dihitung dalam kode program. Jika nilai sensor mencapai ambang tertentu yang mengindikasikan sentuhan, program dapat mengambil tindakan tertentu, seperti mengaktifkan solenoid untuk membuka pintu.	Tindakan: Berhasil

2.6 Pengujian Fungsionalitas API

Pada pengujian fungsional API berfokus pada fungsi dari fitur-fitur yang ada pada sistem API yang telah dibangun. Pengujian fungsional dilakukan menggunakan metode *blackbox* dengan bantuan *postman* sebagai alat pengujian. Pengujian *blackbox* merupakan sebuah metode pengujian perangkat lunak yang dilakukan tanpa memperhatikan struktur kode program didalamnya. Tabel 2.10 menjelaskan beberapa fitur hasil implementasi dari kebutuhan fungsional pada sistem keamanan kunci pintu gedung.

Tabel 2.10 Hasil pengujian fungsional

No	Kebutuhan	Hasil
1	Autentikasi <i>login</i> dan <i>logout</i>	Tersedia
2	Ganti <i>password</i>	Tersedia
3	Reset <i>password</i>	Tersedia
4	Verifikasi <i>email</i>	Tersedia
5	Ganti profil	Tersedia
6	Lihat daftar akses	Tersedia
7	Verifikasi akses	Tersedia
8	Lihat riwayat aktifitas	Tersedia
9	Lihat daftar pintu	Tersedia

Tabel 2.10 (lanjutan)

No	Kebutuhan	Hasil
10	Membuka pintu jarak jauh	Tersedia
11	Register pintu baru	Tersedia
12	Koneksi <i>websocket</i> untuk pintu	Tersedia
13	Mendapatkan <i>signature</i>	Tersedia
14	<i>Update</i> status pintu	Tersedia
15	Peringatan pintu	Tersedia

Terlihat pada Tabel 2.10 di atas, beberapa fitur yang telah diimplementasikan pada sistem API. Pengujian fungsional dari masing-masing API adalah sebagai berikut:

1. *Login*

Pengujian pada fungsi *login* dilaksanakan untuk memeriksa kinerja dari fungsi *login*, *login* dikatakan berhasil jika client mendapatkan respon *success* dari server disertai dengan dikirimkannya data *client* dan token akses. Hasil dari pengujian fungsi *login* dapat dilihat pada Tabel 2.11.

Tabel 2.11 Hasil pengujian fungsi *login*

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Login</i> dengan data benar	Melakukan <i>login</i> menggunakan <i>email</i> dan <i>password</i> yang sesuai	<i>success</i>	Berhasil
2	<i>Login</i> dengan data salah	Melakukan <i>login</i> dengan menggunakan <i>email</i> atau <i>password</i> yang salah	<i>failed</i>	Gagal
3	<i>Login</i> dengan data kurang	Melakukan <i>login</i> dengan menggunakan <i>email</i> saja atau <i>password</i> saja	<i>missing_parameter</i>	Gagal
4	<i>Login</i> data tidak terdaftar	Melakukan <i>login</i> dengan menggunakan <i>email</i> yang belum terdaftar	<i>missing_parameter</i>	Gagal
5	<i>Login</i> dengan format tidak sesuai	Melakukan <i>login</i> dengan menggunakan <i>username</i> bukan <i>email</i>	<i>missing_parameter</i>	Gagal

Tabel 2.11 (lanjutan)

No	Nama	Bentuk Pengujian	Respon	Hasil
6	<i>Login</i> dengan <i>email</i> belum terverifikasi	Melakukan <i>login</i> dengan menggunakan <i>email</i> yang belum terverifikasi	<i>email_unverified</i>	Gagal

Dapat dilihat pada Tabel 2.11 proses *login* akan berhasil jika menggunakan email dan *password* yang sesuai, proses *login* juga memastikan semua parameter yang digunakan pada autentikasi tersedia dan juga sesuai. Pada proses pengujian menggunakan email yang belum terverifikasi *login* akan tertahan dengan status *email_unverified* dan menunggu *client* untuk melakukan verifikasi *email*.

2. *Logout*

Pengujian pada fungsi *logout* dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. *Logout* dikatakan berhasil jika *client* menerima respon *success* dari *server*. Hasil dari pengujian fungsi *logout* dapat dilihat pada Tabel 2.12.

Tabel 2.12 Hasil Pengujian fungsi logout

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Logout</i> dengan token	Melakukan <i>logout</i> menggunakan token yang sesuai	<i>success</i>	Berhasil
2	<i>Logout</i> tanpa token	Melakukan <i>logout</i> tanpa menggunakan token	<i>Unauthenticated</i>	Gagal
3	<i>Login</i> dengan token salah	Melakukan <i>logout</i> dengan menggunakan token yang sudah terhapus	<i>Unauthenticated</i>	Gagal

Dapat dilihat pada Tabel 2.12 proses *logout* hanya berhasil jika menggunakan token yang sesuai, jika *client* melakukan *logout* tanpa menggunakan token atau menggunakan token yang sudah terhapus maka *logout* akan gagal dengan respon *unauthenticated* atau tidak terautentikasi.

3. Verifikasi *email*

Pengujian pada fungsi verifikasi *email* dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Verifikasi email dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. Hasil dari pengujian fungsi verifikasi *email* dapat dilihat pada Tabel 2.13.

Tabel 2.13 Hasil pengujian fungsi verifikasi *email*

No	Nama	Bentuk Pengujian	Respon	Hasil
1	Verifikasi <i>email</i> tanpa token	Melakukan verifikasi <i>email</i> tanpa menggunakan token	<i>Unauthenticated</i>	Gagal
2	Verifikasi <i>email</i> tidak sesuai	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang salah	<i>otp_not_match</i>	Gagal
3	Verifikasi <i>email</i> kadaluarsa	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang sudah kadaluarsa	<i>otp_expired</i>	Gagal
4	Verifikasi <i>email</i> sesuai	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang sesuai	<i>success</i>	Berhasil
5	Verifikasi <i>email</i> token salah	Melakukan verifikasi <i>email</i> menggunakan token yang sudah terhapus	<i>Unauthenticated</i>	Gagal

Dapat dilihat pada Tabel 2.13 proses verifikasi *email* hanya berhasil jika *client* mengirimkan kode OTP yang sesuai disertai dengan token yang sesuai. Jika proses verifikasi *email* menggunakan kode OTP yang salah atau sudah kadaluarsa maka proses verifikasi *email* akan gagal.

4. Ganti *password*

Pengujian pada fungsi ganti *password* dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses ganti *password* dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian ganti *password* dapat dilihat pada Tabel 2.14.

Tabel 2.14 Hasil pengujian fungsi ganti *password*

No	Nama	Bentuk Pengujian	Respon	Hasil
1	Ganti <i>password</i> tanpa token	Melakukan penggantian <i>password</i> tanpa menggunakan token	<i>Unauthenticated</i>	Gagal
2	Ganti <i>password</i> dengan token salah	Melakukan penggantian <i>password</i> menggunakan token yang sudah terhapus	<i>Unauthenticated</i>	Gagal
3	Ganti <i>password</i> sesuai	Melakukan penggantian <i>password</i> sesuai	<i>success</i>	Berhasil
4	Ganti <i>password</i> minimal	Melakukan penggantian <i>password</i> dengan 3 karakter	<i>missing_parameter</i>	Gagal
5	Ganti <i>password</i> maksimal	Melakukan penggantian <i>password</i> dengan 100 karakter	<i>missing_parameter</i>	Gagal
6	Ganti <i>password</i> konfirmasi salah	Melakukan penggantian <i>password</i> dengan konfirmasi <i>password</i> berbeda	<i>missing_parameter</i>	Gagal

Pada Tabel 2.14 terlihat proses ganti *password* berhasil jika *client* mengirimkan *password* dan konfirmasi *password* sesuai, *client* juga harus mengirimkan token yang sesuai juga. Jika salah satu parameter tidak terpenuhi maka proses ganti *password* akan gagal.

5. *Reset password*

Pengujian pada fungsi *reset password* dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses *reset password* dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian *reset password* dapat dilihat pada Tabel 2.15.

Tabel 2.15 Hasil pengujian fungsi *reset password*

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Reset password email</i> sesuai	Melakukan <i>reset password</i> menggunakan <i>email</i> dengan format sesuai dan terdaftar	<i>success</i>	Berhasil
2	<i>Reset password email</i> tidak sesuai	Melakukan <i>reset password</i> menggunakan <i>email</i> yang belum terdaftar	<i>failed</i>	Gagal
3	<i>Reset password</i> format <i>email</i> salah	Melakukan <i>reset password</i> menggunakan <i>email</i> dengan format salah	<i>missing_parameter</i>	Gagal

Pada Tabel 2.15 terlihat proses *reset password* berhasil jika *client* mengirimkan alamat *email* yang sesuai, Jika *email* tidak sesuai atau belum terdaftar maka *reset password* akan gagal.

6. *Update* profil

Pengujian pada fungsi *update* profil dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses *update* profil dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian *update* profil dapat dilihat pada Tabel 2.16.

Tabel 2.16 Hasil pengujian fungsi *update* profil

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Update nama minimal</i>	Melakukan update profil menggunakan nama 3 karakter	<i>missing_parameter</i>	Gagal
2	<i>Update nama sesuai</i>	Melakukan update profil menggunakan nama sesuai	<i>success</i>	Berhasil
3	<i>Update email sesuai</i>	Melakukan update profil menggunakan email yang sesuai	<i>success</i>	Berhasil

4	Update email format salah	Melakukan update profil menggunakan email dengan format tidak sesuai	<i>missing_parameter</i>	Gagal
5	Update jenis kelamin sesuai	Melakukan update jenis kelamin dengan format sesuai	<i>success</i>	Berhasil
6	Update nomor hp sesuai	Melakukan update nomor hp dengan format sesuai	<i>success</i>	Berhasil
7	Update nomor hp tidak sesuai	Melakukan update nomor hp dengan format salah (digit angka kurang/lebih)	<i>missing_parameter</i>	Gagal

Pada Tabel 2.16 terlihat proses *update* profil akan berhasil jika semua data yang dikirimkan sesuai dengan format sehingga mendapatkan respon *success*, jika ada salah satu data yang tidak sesuai format misalnya format *email* tidak sesuai atau nomor hp kurang maka *update* profil akan gagal.

7. *Update* avatar

Pengujian pada fungsi *update* avatar dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses *update* avatar dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian *reset password* dapat dilihat pada Tabel 2.17.

Tabel 2.17 Hasil pengujian fungsi *update* avatar

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Update</i> avatar format sesuai	Melakukan <i>update</i> gambar avatar sesuai dengan format	<i>success</i>	Berhasil
2	<i>Update</i> avatar file besar	Melakukan <i>update</i> gambar menggunakan gambar dengan ukuran lebih dari 1 MB	<i>request entity too large</i>	Gagal

Tabel 2.17 (lanjutan)

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Update</i> avatar format sesuai	Melakukan <i>update</i> gambar avatar sesuai dengan format	<i>success</i>	Berhasil
2	<i>Update</i> avatar file besar	Melakukan <i>update</i> gambar menggunakan gambar dengan ukuran lebih dari 1 MB	<i>request entity too large</i>	Gagal
3	<i>Update</i> avatar bukan gambar	Melakukan <i>update</i> avatar menggunakan file selain gambar	<i>missing_parameter</i>	Gagal

Pada Tabel 2.17 terlihat proses *update* avatar hanya berhasil jika file yang dikirim adalah gambar dengan ukuran kurang dari 1 MB, jika data yang dikirim bukan merupakan gambar atau ukuran gambar lebih dari 1 MB maka proses *update* avatar akan gagal.

8. Lihat akses

Pengujian pada fungsi lihat akses dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses lihat akses dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian fungsi lihat akses dapat dilihat pada Tabel 2.18.

Tabel 2.18 Hasil pengujian fungsi lihat akses

No	Nama	Bentuk Pengujian	Respon	Hasil
1	Lihat akses dengan token	Melakukan <i>request</i> lihat akses menggunakan token yang sesuai	<i>success</i>	Berhasil
2	Lihat akses tanpa token	Melakukan <i>request</i> lihat akses tanpa menggunakan token	<i>Unauthenticated</i>	Gagal

Pada Tabel 2.18 terlihat bahwa proses lihat akses akan berhasil jika permintaan dilakukan dengan menambahkan token, jika permintaan tidak menggunakan token maka permintaan akan ditolak.

9. Lihat pintu

Pengujian pada fungsi lihat pintu dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses lihat pintu dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian fungsi lihat pintu dapat dilihat pada Tabel 2.19.

Tabel 2.19 Hasil pengujian fungsi lihat pintu

No	Nama	Bentuk Pengujian	Respon	Hasil
1	Lihat pintu tanpa token	Melakukan <i>request</i> lihat akses tanpa menggunakan token yang sesuai	<i>Unauthenticated</i>	Gagal
2	Lihat pintu dengan token pengguna	Melakukan <i>request</i> lihat akses menggunakan token yang dimiliki pengguna	<i>Unauthenticated</i>	Gagal
3	Lihat Pintu dengan token operator	Melakukan <i>request</i> lihat akses menggunakan token yang dimiliki operator	<i>success</i>	Berhasil

Pada Tabel 2.19 terlihat bahwa proses lihat pintu berhasil jika permintaan disertai dengan token operator, jika permintaan tidak menggunakan token atau menggunakan token yang dimiliki oleh pengguna biasa maka permintaan akan ditolak.

10. Riwayat akses

Pengujian pada fungsi lihat riwayat akses dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses lihat riwayat akses dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian fungsi lihat riwayat akses dapat dilihat pada Tabel 2.20.

Tabel 2.20 Hasil pengujian riwayat akses

No	Nama	Bentuk Pengujian	Respon	Hasil
1	Lihat riwayat dengan token	Melakukan <i>request</i> lihat riwayat menggunakan token yang sesuai	<i>success</i>	Berhasil
2	Lihat riwayat tanpa token	Melakukan <i>request</i> lihat riwayat tanpa token menggunakan token	<i>Unauthenticated</i>	Gagal

Pada Tabel 2.20 terlihat bahwa proses lihat riwayat akses berhasil jika permintaan disertai dengan token yang sesuai, jika permintaan tidak menggunakan token maka permintaan akan ditolak.

11. Verifikasi akses

Pengujian pada fungsi verifikasi akses dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses verifikasi akses dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian fungsi lihat riwayat akses dapat dilihat pada Tabel 2.21.

Tabel 2.21 Hasil pengujian fungsi verifikasi akses

No	Nama	Bentuk Pengujian	Respon	Hasil
1	Verifikasi akses tanpa token	Melakukan <i>request</i> lihat verifikasi akses tanpa menggunakan token	<i>Unauthenticated</i>	Gagal
2	Verifikasi akses dengan token	Melakukan <i>request</i> verifikasi akses menggunakan token yang sesuai	<i>success</i>	Berhasil
3	Verifikasi akses dengan id pintu salah	Melakukan <i>request</i> verifikasi akses menggunakan identitas pintu salah	<i>no_data</i>	Gagal

Pada Tabel 2.21 terlihat bahwa proses verifikasi akses berhasil jika permintaan disertai dengan token dan identitas pintu sesuai, jika identitas pintu tidak sesuai

maka proses verifikasi akses akan gagal karena pintu tidak ditemukan. Jika permintaan tidak disertai dengan token maka permintaan tersebut akan ditolak.

12. *Signature*

Pengujian pada fungsi *signature* dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses pembuatan *signature* dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian fungsi buat signature dapat dilihat pada Tabel 2.22.

Tabel 2.22 Hasil Pengujian Fungsi Signature

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Signature</i> tanpa token	Melakukan <i>request</i> <i>signature</i> tanpa menggunakan token	<i>Unauthenticated</i>	Gagal
2	<i>Signature</i> dengan token	Melakukan <i>request</i> <i>signature</i> menggunakan token dan data lengkap	<i>success</i>	Berhasil
3	<i>Signature</i> data kurang	Melakukan <i>request</i> <i>signature</i> menggunakan data yang kurang	<i>missing_parameter</i>	Gagal

Pada Tabel 2.22 terlihat bahwa proses fungsi *signature* berhasil jika permintaan yang dikirimkan disertai dengan token dan data yang dikirimkan lengkap, jika permintaan yang dikirimkan tanpa menggunakan token atau ada data yang kurang maka permintaan akan gagal.

13. *Websocket*

Pengujian komunikasi *websocket* dilakukan untuk mengetahui kinerja dari koneksi *subscription* pada *channel websocket*. Hasil dari pengujian *websocket* dapat dilihat pada Tabel 2.23.

Tabel 2.23 Hasil pengujian *websocket*

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Subscription</i> dengan <i>signature</i>	Melakukan <i>subscription</i> ke <i>channel websocket</i> menggunakan <i>signature</i> yang sesuai	<i>Subscription Succeed</i>	Berhasil
2	<i>Subscription</i> tanpa <i>signature</i>	Melakukan <i>subscription</i> ke <i>channel websocket</i> tanpa menggunakan <i>signature</i>	<i>Invalid Signature</i>	Gagal
3	<i>Subscription</i> dengan <i>signature</i> salah	Melakukan <i>subscription</i> ke <i>channel websocket</i> menggunakan <i>signature</i> yang tidak sesuai	<i>Invalid Signature</i>	Gagal

Pada Tabel 2.23 terlihat bahwa proses *subscription* pada *channel websocket* berhasil jika menggunakan kode *signature* yang sesuai, jika tidak menggunakan kode *signature* atau menggunakan kode *signature* yang salah maka proses *subscription* akan gagal.

14. *Update* status pintu

Pengujian pada fungsi *update* status pintu dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses *update* status pintu dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian fungsi *update* status pintu dapat dilihat pada Tabel 2.24.

Tabel 2.24 Hasil pengujian fungsi *update* status pintu

No	Nama	Bentuk Pengujian	Respon	Hasil
1	<i>Update</i> status tanpa token	Melakukan <i>update</i> status pintu tanpa menggunakan token	<i>Unauthenticated</i>	Gagal
2	<i>Update</i> status dengan token	Melakukan <i>update</i> status pintu dengan menggunakan token dan data lengkap	<i>success</i>	Berhasil

Tabel 2.24 (lanjutan)

No	Nama	Bentuk Pengujian	Respon	Hasil
3	<i>Update</i> status data tidak lengkap	Melakukan <i>update</i> status pintu dengan menggunakan data yang tidak lengkap	<i>missing_parameter</i>	Gagal
4	<i>Update</i> status id pintu salah	Melakukan <i>update</i> status pintu menggunakan data identitas pintu yang salah	<i>failed</i>	Gagal

Pada Tabel 2.24 terlihat bahwa proses *update* status pintu berhasil jika *request* dikirimkan dengan token dan data yang lengkap, jika ada data yang kurang lengkap atau tidak disertai dengan token akan *request* tersebut akan gagal.

15. Peringatan pintu

Pengujian pada fungsi peringatan pintu dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. Proses peringatan pintu dikatakan berhasil jika *client* mendapatkan respon *success* dari *server*. hasil dari pengujian fungsi peringatan pintu dapat dilihat pada Tabel 2.25.

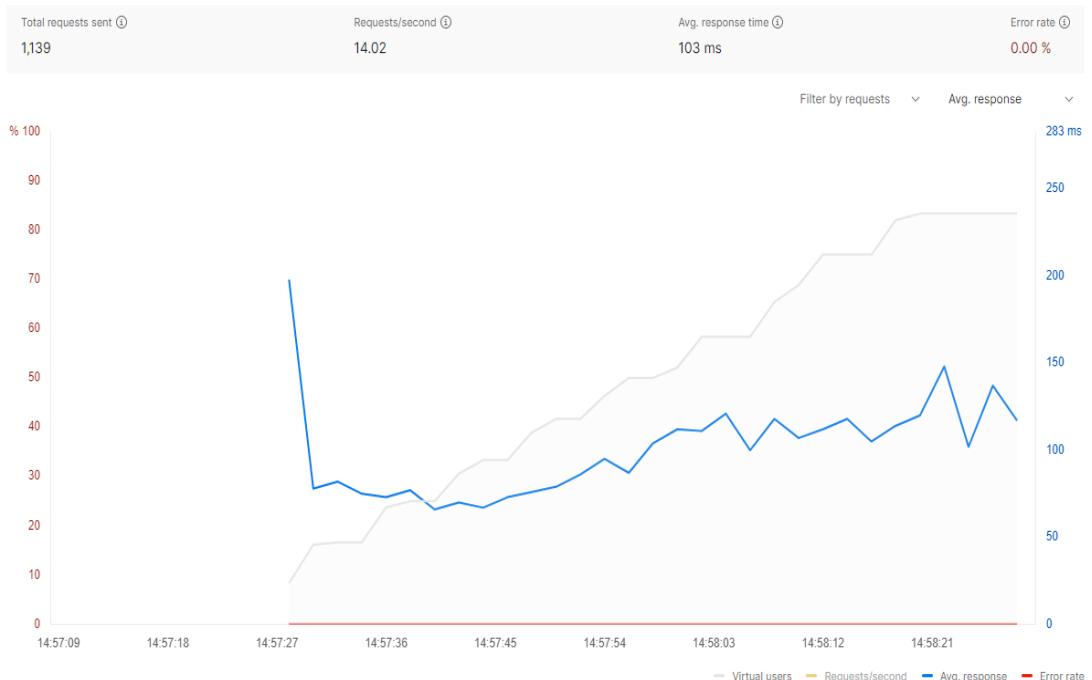
Tabel 2.25 Hasil pengujian fungsi peringatan pintu

No	Nama	Bentuk Pengujian	Respon	Hasil
1	Peringatan pintu tanpa token	Mengirim peringatan pintu tanpa menggunakan token	<i>Unauthenticated</i>	Gagal
2	Peringatan pintu sesuai	Mengirim peringatan pintu dengan menggunakan token dan data lengkap	<i>success</i>	Berhasil
3	Peringatan pintu data tidak lengkap	Mengirim peringatan pintu dengan menggunakan data yang tidak lengkap	<i>missing_parameter</i>	Gagal
4	Peringatan pintu id pintu salah	Mengirim peringatan pintu menggunakan data identitas pintu yang salah	<i>failed</i>	Gagal

Pada Tabel 2.25 terlihat bahwa proses peringatan pintu berhasil jika *request* dikirimkan dengan token dan data yang lengkap, jika ada data yang kurang lengkap atau tidak disertai dengan token akan *request* tersebut akan gagal.

2.7 Pengujian Performa API

Pengujian pengujian performa API berfokus pada karakteristik dari sistem API yang telah dibuat seperti rasio *error*, waktu respon dan lain sebagainya yang menjadi indikator performa dari sistem tersebut. Pada pengujian performa ini dilakukan dengan menggunakan Postman dan Jmeter. Jmeter merupakan sebuah perangkat lunak yang digunakan untuk melakukan pengujian terutama pengujian beban dengan memberikan beberapa *request* secara bersamaan[1].

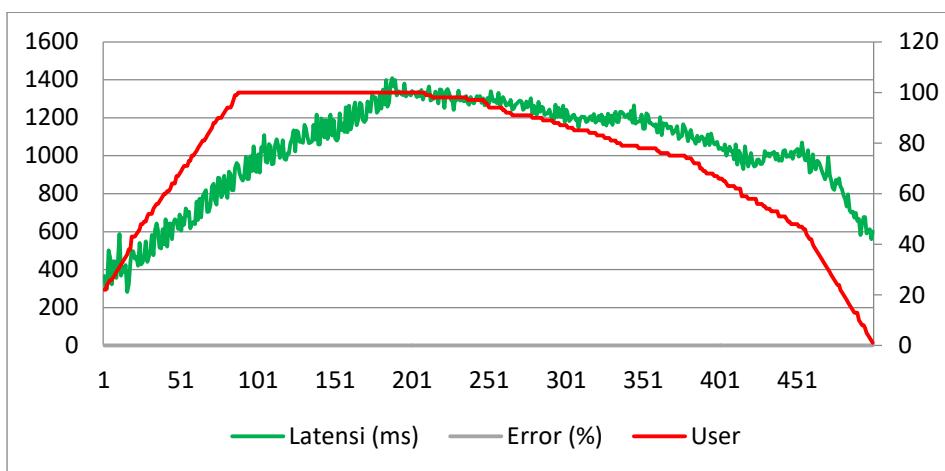


Gambar 2.8 Hasil Pengujian Performa API

Dapat dilihat pada Gambar 2.8, pada pengujian API secara keseluruhan dengan menggunakan 10 pengguna secara bersamaan menunjukkan waktu respon rata-rata 104 milidetik dengan waktu respon tertinggi 148 milidetik dan rasio error 0%. Pengujian performa juga dilakukan dengan melakukan simulasi beban pengguna pada API yang kemungkinan besar akan diakses secara bersamaan dan menentukan kinerja dari sistem keamanan kunci pintu tersebut, yaitu :

1. Performa verifikasi akses

Pada sistem penguncian pintu gedung ini kemungkinan pengguna melakukan permintaan akses secara bersamaan, secara teori semakin banyak pengguna yang mengirimkan *request* maka waktu respon akan semakin meningkat. Oleh karena itu dilakukan pengujian untuk melihat kemampuan sistem dalam menangani permintaan akses tersebut.

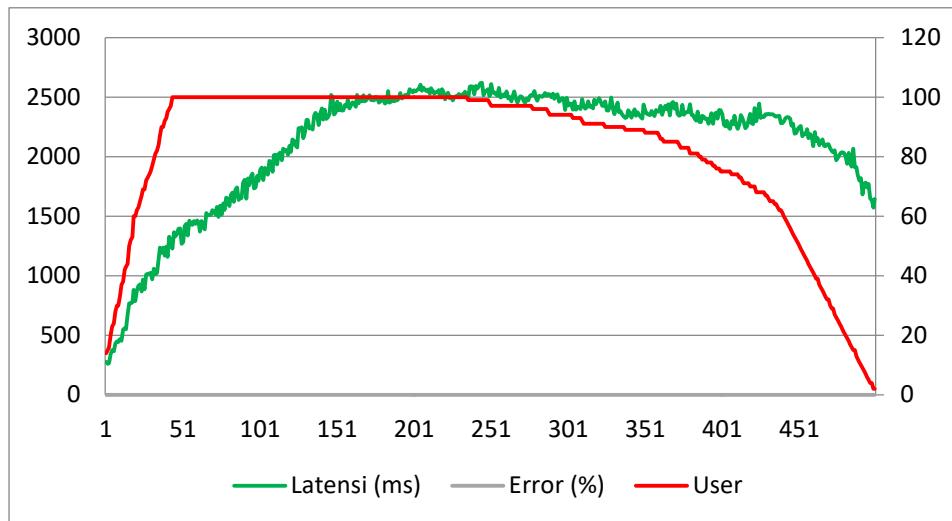


Gambar 2.9 Grafik hasil pengujian performa verifikasi akses

Gambar 2.9 merupakan hasil pengujian fungsi verifikasi akses menggunakan Jmeter. Pengujian dilakukan dengan melakukan simulasi beban permintaan verifikasi akses sebanyak 100 pengguna berbeda secara bersamaan. Dari hasil pengujian didapatkan bahwa pada puncak jumlah pengguna sistem membutuhkan waktu rata-rata 1.3 detik untuk memberikan respon ke pengguna dengan rasio *error* 0.0%.

2. Performa *update* status pintu

Setiap terjadi perubahan status pada pintu maka perangkat penguncian akan langsung mengirimkan status perubahan ke server. Dalam prosesnya dimungkinkan beberapa pintu mengirimkan status perubahan secara bersamaan sehingga dapat mempengaruhi kinerja dari sistem.

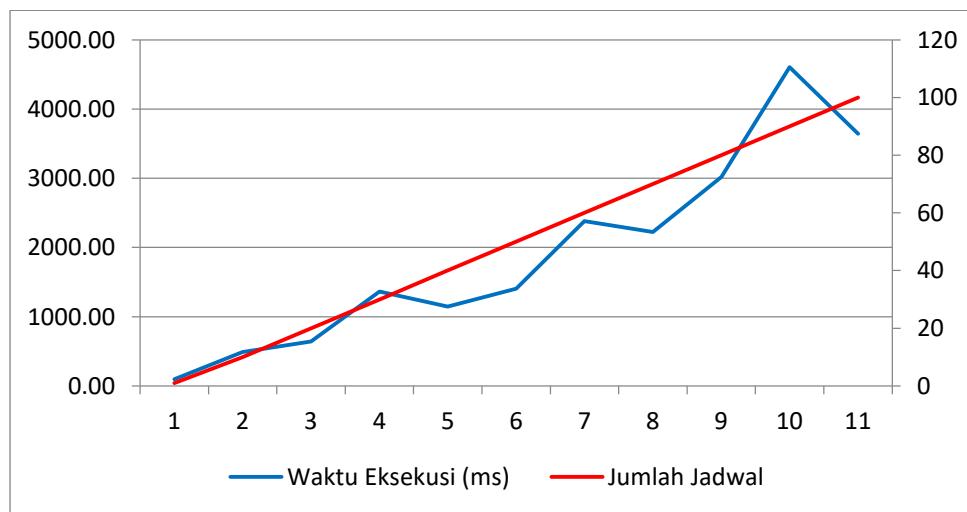


Gambar 2.10 Hasil Pengujian Performa Update Status Pintu

Gambar 2.10 merupakan hasil pengujian performa *update* status pintu menggunakan Jmeter. Proses pengujian mendapatkan hasil bahwa pada beban 100 pintu berbeda melakukan *update* status secara bersamaan maka sistem memerlukan waktu rata-rata 2.5 detik dengan rasio error 0.0%.

3. Performa penjadwalan

Pada proses penjadwalan maka sistem akan melakukan pemeriksaan data jadwal pada *database* setiap 1 menit, oleh karena itu metode pengecekan jadwal diharuskan selesai dilaksanakan sebelum proses pemeriksaan selanjutnya dijalankan. hasil dari proses pengujian performa penjadwalan dapat dilihat pada Gambar 2.11 di bawah.



Gambar 2.11 Hasil Pengujian Performa Penjadwalan

Pengujian dilakukan dengan mencatat waktu pemeriksaan untuk setiap jadwal, yaitu terdapat 11 titik pengujian dimulai dari 1 jadwal sampai 100 jadwal dengan masing-masing jadwal terdapat 20 pintu. Dapat dilihat pada Gambar 4.34 di atas, semakin besar jumlah jadwal yang ada maka waktu yang diperlukan pada proses penjadwalan akan semakin lama dimana untuk 1 jadwal memerlukan waktu 96.00 milidetik dan 100 jadwal memerlukan waktu 3641.14 milidetik. Dari hasil pengujian menggunakan 100 jadwal waktu yang diperlukan yaitu 3.6 detik dimana nilai tersebut masih di bawah dari periode pengecekan jadwal (1 menit) sehingga sistem masih dapat menangani 100 proses penjadwalan dengan aman.

2.8 Pengujian Fungsionalitas Website

Proses pengujian fungsionalitas *website* berfokus pada fungsi dari setiap fitur yang telah diimplementasikan. Pengujian dilakukan dengan menggunakan metode *blackbox* dengan menggunakan *test-case* yang disesuaikan dengan berbagai kemungkinan interaksi pengguna pada fitur tersebut. Hasil dari proses pengujian fungsionalitas *website* dapat dilihat pada Tabel 2.26 di bawah.

Tabel 2.26 Hasil pengujian kebutuhan *website*

No	Kriteria Kebutuhan	Hasil
1.	Tersedia Halaman <i>Login</i> dan <i>logout website</i>	Tersedia
2.	Tersedia halaman <i>Reset Password</i>	Tersedia
3.	Tersedia halaman untuk mengisi kode OTP	Tersedia
4.	Tersedia menu dan halaman daftar pintu	Tersedia
5.	Tersedia menu dan halaman daftar pengguna	Tersedia
6.	Tersedia menu dan halaman penjadwalan	Tersedia
7.	Tersedia menu dan halaman Riwayat akses	Tersedia
8.	Tersedia menu dan halaman pengaturan akun	Tersedia
9.	Tersedia menu dan halaman daftar Gedung	Tersedia
10	Tersedia menu dan halaman daftar operator	Tersedia

Dapat dilihat pada Tabel 2.26 di atas, semua fitur atau menu yang dimiliki oleh *website* telah berhasil diimplementasikan menggunakan metode dan pemrograman yang sesuai. Proses pengujian lebih lanjut dilakukan pada semua fitur yang ada dengan hasil sebagai berikut:

1. Autentikasi

Pengujian pada halaman autentikasi dilaksanakan untuk menguji keberhasilan dari fungsi *login* dan *logout* yang bekerja pada halaman *website*. Hasil pengujian halaman autentikasi ditunjukkan pada tabel 2.27 di bawah.

Tabel 2.27 Hasil pengujian fungsi autentikasi

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	<i>Login</i> dengan data benar	Melakukan <i>login</i> menggunakan <i>email</i> dan <i>password</i> yang sesuai	masuk ke halaman <i>dashboard</i>	Berhasil
2.	<i>Login</i> dengan data salah	Melakukan <i>login</i> menggunakan <i>email</i> atau <i>password</i> yang salah	Pesan <i>login</i> gagal	Gagal
3.	<i>Login</i> dengan data kurang	Melakukan <i>login</i> menggunakan <i>email</i> saja atau <i>password</i> saja	Tampil peringatan untuk mengisi kolom yang masih kosong.	Gagal
4.	<i>Login</i> dengan data tidak terdaftar	Melakukan <i>login</i> menggunakan <i>email</i> yang belum terdaftar	Tampil peringatan “ <i>login</i> gagal”. Data tidak ditemukan.	Gagal

Tabel 2.27 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
5.	<i>Login</i> dengan format tidak sesuai	Melakukan <i>login</i> menggunakan <i>username</i> bukan <i>email</i>	Tampil peringatan untuk menyertakan “@” pada format penulisan <i>email</i> .	Gagal
6.	<i>Login</i> dengan <i>email</i> belum terverifikasi	Melakukan <i>login</i> menggunakan <i>email</i> yang berlum terverifikasi	Tampil peringatan <i>login</i> gagal dan data tidak ditemukan	Gagal
7.	<i>Logout</i> dengan menekan tombol <i>logout</i>	Melakukan Tindakan untuk <i>logout</i> dengan menekan tombol <i>logout</i>	Akan Kembali keluar ke halaman <i>login</i>	Berhasil

Terlihat pada Tabel 2.27 bahwa proses pengujian yang dilakukan pada fungsi autentikasi *website*, fungsi autentikasi akan berhasil jika data yang dimasukkan benar sesuai dengan ketentuan dan proses autentikasi akan gagal jika data tidak sesuai.

2. *Reset password*

Pengujian pada halaman *reset password* dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses yang bekerja pada halaman *reset password* pada *website*. Hasil pengujian halaman *reset password* ditunjukkan pada Tabel 2.28 di bawah.

Tabel 2.28 Hasil pengujian *reset password*

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Menampilkan halaman <i>reset password</i>	Melakukan akses dengan menekan tombol lupa <i>password</i>	Halaman <i>reset password</i> berhasil ditampilkan	Berhasil
2.	<i>reset password</i> sesuai	Mengisi formulir <i>reset password</i> dengan <i>email</i> yang terdaftar	Mendapat <i>email</i> balasan berupa <i>link</i> untuk mengakses <i>reset password</i>	Berhasil
3.	Mengganti <i>password</i>	Mengisi <i>password</i> baru dan konfirmasi <i>password</i> baru	Terkonfirmasi dan masuk ke halaman <i>login</i> .	Berhasil
4.	<i>email</i> tidak terdaftar	Memasukkan <i>email</i> yang tidak terdaftar atau salah memasukkan alamat <i>email</i>	Tampil peringatan bahwa alamat <i>email</i> tidak ditemukan	Gagal

Dapat dilihat pada Tabel 2.28 bahwa proses *reset password* akan berhasil jika prosesnya menggunakan alamat *email* yang terdaftar serta *password* baru yang dimasukkan sesuai dengan format yang telah ditentukan.

3. Verifikasi *email*

Pengujian pada halaman verifikasi *email* dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses untuk melakukan verifikasi *email* dengan memasukkan kode OTP yang bekerja pada halaman verifikasi *email* pada *website*. Hasil pengujian halaman verifikasi *email* ditunjukkan pada tabel 2.29 di bawah.

Tabel 2.29 Hasil pengujian verifikasi email

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Verifikasi <i>email</i> tidak sesuai	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang salah	Tampil peringatan kode OTP tidak sesuai	Gagal
2.	Verifikasi <i>email</i> kadaluwarsa.	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang sudah kadaluwarsa	Tampil peringatan OTP sudah	Gagal
3.	Verifikasi <i>email</i> sesuai	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang sesuai	Verifikasi <i>email</i> berhasil	Berhasil
4.	Verifikasi <i>email token</i> salah	Melakukan verifikasi <i>email</i> menggunakan <i>token</i> yang sudah terhapus	Kode OTP tidak diketahui	Gagal
5.	Verifikasi <i>email token</i> lebih dari 6 digit	Melakukan verifikasi <i>email</i> menggunakan <i>token</i> yang lebih dari 6 digit	Tampil peringatan bahwa <i>token</i> lebih dari 6 digit	Gagal

Dapat dilihat pada Tabel 2.29 bahwa proses verifikasi *email* akan berhasil jika menggunakan kode OTP yang sesuai dan masih aktif yang telah diterima pada alamat *email* yang terdaftar.

4. Daftar pintu

Pengujian pada halaman daftar pintu dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses pada halaman daftar pintu pada *website*. Hasil pengujian halaman daftar pintu ditunjukkan pada Tabel 2.30 di bawah.

Tabel 2.30 Hasil pengujian daftar pintu

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu daftar pintu pada halaman <i>dashboard</i>	Melakukan akses dengan menekan menu daftar pintu	Halaman daftar pintu berhasil ditampilkan	Berhasil
2.	Menampilkan daftar pintu yang terdaftar dalam aplikasi	Melakukan akses untuk melihat daftar pintu	Daftar pintu berhasil ditampilkan	Berhasil
3.	Menambahkan daftar pintu	Melakukan akses untuk menambah daftar pintu dengan menekan tombol tambah pada halaman daftar pintu	Tampil formulir untuk mengisi nama pintu yang akan ditambahkan	Berhasil
4.	Menambahkan daftar pintu dengan nama yang sama	Melakukan akses untuk menambah daftar pintu dengan menggunakan nama yang sudah ada dalam daftar pintu sebelumnya	Daftar pintu baru berhasil ditambahkan	Gagal
5.	Mencari daftar pintu yang tersedia	Mencari daftar pintu yang sudah	Daftar pintu berhasil	Berhasil

		terdaftar dengan cepat menggunakan menu cari pintu	ditemukan setelah dicari dengan kata kunci yang dituliskan	
6.	Membuka dan menutup akses penguncian pintu	Melakukan akses untuk membuka dan menutup daftar pintu dengan menekan tombol buka atau tutup	Muncul formulir untuk konfirmasi buka pintu atau batal buka pintu	Berhasil

Tabel 2.30 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
5.	Mencari daftar pintu yang tersedia	Mencari daftar pintu yang sudah terdaftar dengan cepat menggunakan menu cari pintu	Daftar pintu berhasil ditemukan setelah dicari dengan kata kunci yang dituliskan	Berhasil
6.	Membuka dan menutup akses penguncian pintu	Melakukan akses untuk membuka dan menutup daftar pintu dengan menekan tombol buka atau tutup	Muncul formulir untuk konfirmasi buka pintu atau batal buka pintu	Berhasil

5. Daftar pengguna

Pengujian pada halaman daftar pengguna dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses pada halaman daftar pengguna pada *website*. Hasil pengujian halaman daftar pengguna ditunjukkan pada Tabel 2.31 di bawah.

Tabel 2.31 Hasil pengujian daftar pengguna

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu daftar pengguna pada halaman <i>dashboard</i>	Melakukan akses dengan menekan menu daftar pengguna	Halaman daftar pengguna berhasil ditampilkan	Berhasil
2.	Menampilkan daftar pengguna yang terdaftar dalam aplikasi	Melakukan akses untuk melihat daftar pengguna	Daftar pengguna berhasil ditampilkan	Berhasil
3.	Menampilkan daftar pengguna ketika belum ada pintu yang terdaftar	Melakukan akses untuk melihat daftar pengguna	Daftar pengguna tidak ditemukan	Gagal
4.	Menambahkan daftar pengguna	Menambah daftar pengguna dengan menekan tombol tambah pada halaman daftar pintu	Tampil formulir untuk mengisi nama pengguna yang akan ditambahkan	Berhasil
5.	Menambahkan daftar pengguna dengan nama yang berbeda	Melakukan akses untuk menambah daftar pengguna dengan menggunakan nama yang belum ada dalam daftar pengguna sebelumnya	Daftar pengguna baru berhasil ditambahkan	Berhasil
6.	Menambahkan	Melakukan akses	Daftar pengguna	Gagal

	daftar pengguna dengan nama yang sama	untuk menambah daftar pengguna dengan menggunakan nama yang sudah ada dalam daftar pengguna sebelumnya	baru berhasil ditambahkan	
7.	Mencari daftar pengguna yang tersedia	Mencari daftar pengguna yang sudah terdaftar dengan cepat menggunakan menu cari pintu	Daftar pengguna berhasil ditemukan setelah dicari dengan kata kunci yang dituliskan	Berhasil
8.	Mencari daftar pengguna yang tidak tersedia	Mencari daftar pengguna yang belum terdaftar dengan cepat menggunakan menu cari pintu	Daftar pengguna tidak ditemukan setelah dicari dengan kata kunci yang dituliskan	Gagal
9.	Melihat daftar pengguna	Melakukan akses untuk melihat daftar pengguna dengan menekan tombol “lihat”	Dapat melihat <i>profile</i> pengguna dan melihat akses pintu pengguna	Berhasil
10.	Menghapus daftar pengguna	Melakukan akses untuk menghapus daftar pengguna dengan menekan tombol “hapus”	Muncul formulir untuk mengkonfirmasi penghapusan	Berhasil

Tabel 2.31 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
6.	Menambahkan daftar pengguna dengan nama yang sama	Menambah daftar pengguna dengan menggunakan nama yang sudah ada dalam daftar pengguna sebelumnya	Daftar pengguna baru berhasil ditambahkan	Gagal
7.	Mencari daftar pengguna yang tersedia	Mencari daftar pengguna yang sudah terdaftar dengan cepat menggunakan menu cari pintu	Daftar pengguna berhasil ditemukan setelah dicari dengan kata kunci yang dituliskan	Berhasil
8.	Mencari daftar pengguna yang tidak tersedia	Mencari daftar pengguna yang belum terdaftar dengan cepat menggunakan menu cari pintu	Daftar pengguna tidak ditemukan setelah dicari dengan kata kunci yang dituliskan	Gagal
9.	Melihat daftar pengguna	Melakukan akses untuk melihat daftar pengguna dengan menekan tombol “lihat”	Dapat melihat <i>profile</i> pengguna dan melihat akses pintu pengguna	Berhasil
10.	Menghapus daftar pengguna	Melakukan akses untuk menghapus daftar pengguna dengan menekan tombol “hapus”	Muncul formulir untuk mengkonfirmasi penghapusan	Berhasil

6. Penjadwalan

Pengujian pada halaman penjadwalan dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses pada halaman penjadwalan pada *website*. Hasil pengujian halaman penjadwalan ditunjukkan pada Tabel 2.32 di bawah.

Tabel 2.32 Hasil pengujian penjadwalan

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu daftar pengguna pada halaman <i>dashboard</i>	Melakukan akses dengan menekan menu daftar pengguna	Halaman daftar pengguna berhasil ditampilkan	Berhasil
2.	Menambahkan penjadwalan pada kunci pintu	Melakukan akses untuk menambah penjadwalan menekan tombol tambah	Tampipl formulir untuk mengatur jadwal penguncian	Berhasil
3.	Mencari daftar pintu yang sudah terjadwal	Mencari daftar pintu yang sudah terjadwal pada sistem	Daftar pintu yang terjadwal berhasil ditemukan menggunakan kata kunci yang sesuai	Berhasil
4.	Pengaturan jadwal kunci pintu	Mengatur penjadwalan pada formulir dengan memasukkan data nama pintu, tanggal, durasi harian, dan perulangan hari	Penjadwalan penguncian terkonfirmasi dan dapat terjadwal sesuai pengaturan	Berhasil

Tabel 2.32 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
5.	Memilih tanggal dan durasi pengaturan jadwal pintu	Menentukan tanggal dan durasi penguncian pintu sesuai permintaan	Pintu hanya dapat dibuka sesuai jadwal yang telah ditetapkan	Berhasil
6.	Konfirmasi Penjadwalan	Melakukan akses dengan menekan tombol “Tambahkan”	Penjadwalan yang telah diatur berhasil terkonfirmasi	Berhasil
7.	Pembataran penjadwalan akses kunci pintu	Melakukan akses dengan menekan tombol “Batal”	Penjadwalan yang telah diatur batal terkonfirmasi	Berhasil

7. Riwayat akses

Pengujian pada halaman riwayat akses dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses pada halaman riwayat akses pada *website*. Hasil pengujian halaman riwayat akses ditunjukkan pada Tabel 2.33 di bawah.

Tabel 2.33 Hasil pengujian riwayat akses

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu riwayat akses	Melakukan akses dengan menekan menu riwayat akses	Halaman riwayat akses berhasil ditampilkan	Berhasil
2.	Menampilkan riwayat akses yang terdaftar dalam sistem	Melakukan akses untuk melihat halaman riwayat akses	Halaman riwayat akses berhasil ditampilkan	Berhasil

Tabel 2.33 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
4.	Mencari riwayat pintu yang telah diakses oleh pengguna	Melakukan akses dengan mengetikkan kata kunci pada kolom untuk mencari riwayat pintu yang telah diakses oleh pengguna	Riwayat pintu yang telah diakses berhasil ditemukan	Berhasil

8. Pengaturan akun

Pengujian pada halaman riwayat akses dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses pada halaman riwayat akses pada website. Hasil pengujian halaman riwayat akses ditunjukkan pada Tabel 2.34 di bawah.

Tabel 2.34 Hasil pengujian pengaturan akun

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka halaman <i>profile</i>	Melakukan akses dengan menekan menu <i>profile</i>	Halaman <i>profile</i> berhasil ditampilkan	Berhasil
2.	Melakukan <i>upload</i> foto	Melakukan akses dengan menekan tombol <i>upload</i> foto	Tersedia formulir untuk menambahkan foto profil	Berhasil
3.	Memilih file foto	Memilih file foto dengan menekan tombol pilih file dan memilih file yang sesuai dengan format dan ukuran	Tampil ke halaman penyimpanan perangkat untuk memilih foto profil	Berhasil

Tabel 2.34 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
4.	Memilih foto dengan format dan ukuran yang tidak sesuai	Memilih file foto dengan format dan ukuran yang tidak sesuai dengan ketentuan	Foto profil tidak berhasil di <i>upload</i>	Gagal
5.	Mengedit profil	Menekan tombol “edit profil”	Berhasil muncul formulir untuk melakukan update profil	Berhasil
6.	Konfirmasi edit profil	Melakukan akses dengan menekan tombol “simpan”	Berhasil menyimpan pembaruan profil	Berhasil
7.	Membatalkan edit profil	Melakukan akses dengan menekan tombol “batal”	Pembaruan profil dibatalkan	Berhasil
8.	Mengganti <i>password</i>	Melakukan akses dengan menekan tombol ganti <i>password</i>	Berhasil muncul formulir untuk melakukan ganti <i>password</i>	Berhasil
9.	Memasukkan <i>password</i> saat ini dengan benar	Mengetikkan <i>password</i> saat ini dengan benar	Berhasil tampil formulir untuk mengisi <i>password</i> baru	Berhasil
10.	Salah memasukkan <i>password</i> saat ini	Mengetikkan <i>password</i> saat ini dengan salah	Tidak dapat untuk mengganti <i>password</i> baru	Gagal

Tabel 2.34 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
11.	Konfirmasi <i>password</i> baru	Memasukkan <i>password</i> baru dan konfirmasi <i>password</i> baru pada form	<i>Password</i> baru berhasil terkonfirmasi	Berhasil
12.	Membuka tombol menu <i>logout</i>	Melakukan akses dengan menekan tombol <i>logout</i>	Berhasil untuk keluar dari aplikasi	Berhasil

9. Daftar gedung

Pengujian pada halaman daftar gedung dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses pada halaman daftar gedung pada *website*. Hasil pengujian halaman daftar gedung ditunjukkan pada Tabel 2.35 di bawah.

Tabel 2.35 Hasil pengujian daftar gedung

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu daftar gedung pada halaman <i>dashboard</i>	Melakukan akses dengan menekan menu daftar gedung	Halaman daftar gedung berhasil ditampilkan	Berhasil
2.	Menambahkan daftar gedung dengan nama yang berbeda	Melakukan akses untuk menambah daftar gedung dengan menggunakan nama yang belum ada dalam daftar gedung sebelumnya	Daftar gedung baru berhasil ditambahkan	Berhasil

Tabel 2.25 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
3.	Menambahkan daftar gedung dengan nama yang sama	Menambah daftar gedung dengan menggunakan nama yang sudah ada dalam daftar pintu sebelumnya	Daftar gedung baru berhasil ditambahkan	Gagal
4.	Mencari daftar gedung yang tersedia	Mencari daftar gedung yang sudah terdaftar dengan cepat menggunakan menu cari gedung	Daftar gedung berhasil ditemukan setelah dicari dengan kata kunci yang dituliskan	Berhasil
5.	Mencari daftar gedung yang tidak tersedia	Mencari daftar gedung yang belum terdaftar dengan cepat menggunakan menu cari gedung	Daftar gedung tidak ditemukan setelah dicari dengan kata kunci yang dituliskan	Gagal
6.	Membuka dan menutup akses penguncian pintu	membuka dan menutup daftar pintu dengan menekan tombol buka atau tutup	Muncul formulir untuk konfirmasi buka pintu atau batal buka pintu	Berhasil
7.	Mengedit daftar gedung	Melakukan akses untuk mengedit daftar gedung dengan menekan tombol “edit”	Muncul formulir untuk mengedit nama dan operator gedung	Berhasil

Tabel 2.35 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
8.	Menghapus daftar gedung	Melakukan akses untuk menghapus daftar gedung dengan menekan tombol “hapus”	Muncul formulir untuk mengkonfirmasi penghapusan daftar gedung	Berhasil

10. Daftar operator

Pengujian pada halaman daftar operator dilaksanakan untuk menguji keberhasilan dari fungsi tampilan dan akses pada halaman daftar operator pada *website*. Hasil pengujian halaman daftar operator ditunjukkan pada Tabel 2.36 di bawah.

Tabel 2.36 Hasil pengujian daftar operator

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	akses <i>dashboard</i> khusus moderator	Melakukan akses dengan menekan menu daftar operator	Halaman daftar operator berhasil ditampilkan	Berhasil
2.	tampilan daftar operator	Melakukan akses untuk melihat daftar operator	Daftar operator berhasil ditampilkan	Berhasil
3.	Menambahkan daftar operator	Menambah daftar operator dengan menekan tombol tambah pada halaman daftar operator	Tampil formulir untuk mengisi nama operator yang akan ditambahkan	Berhasil

Tabel 2.36 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
4.	Menambahkan daftar operator dengan nama yang sama	Menambah daftar operator dengan menggunakan nama yang sudah ada dalam daftar operator sebelumnya	Daftar operator baru berhasil ditambahkan	Gagal
5.	Mencari daftar operator yang tersedia	Mencari daftar operator yang sudah terdaftar dengan cepat menggunakan menu cari pintu	Daftar operator berhasil ditemukan setelah dicari dengan kata kunci yang dituliskan	Berhasil
6.	Mencari daftar operator yang tidak tersedia	Mencari daftar operator yang belum terdaftar dengan cepat menggunakan menu cari operator	Daftar operator tidak ditemukan setelah dicari dengan kata kunci yang dituliskan	Gagal
7.	Menghapus daftar operator	Melakukan akses untuk menghapus daftar operator dengan menekan tombol “hapus”	Muncul formulir untuk mengkonfirmasi penghapusan	Berhasil

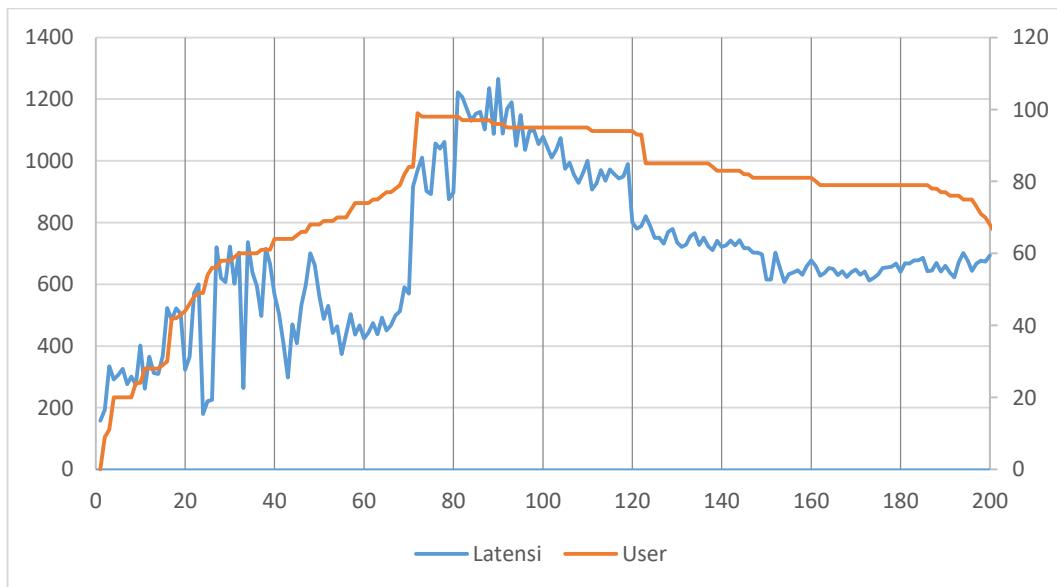
2.9 Pengujian Performa Website

Pengujian performa website bertujuan untuk mengetahui kerakteristik dari *website* yang telah dikembangkan seperti rasio *error*, waktu respon dan lain sebagainya.

Pengujian dilakukan dengan menggunakan metode load testing dengan menggunakan Jmeter. Hasil dari pengujian performa *website* adalah sebagai berikut:

1. Performa *landing page*

Halaman *landing page* diuji untuk memastikan bahwa tujuan utamanya tercapai dengan efektif dan memberikan pengalaman yang positif kepada pengunjung. Pengujian halaman website melibatkan analisis interaksi antara variabel-variabel penting seperti *latensi* (waktu respons *server*), pengguna (jumlah pengunjung), sampel (data yang dimuat), dan waktu (kecepatan pemuatan). Pengujian ini bertujuan untuk memahami bagaimana kinerja halaman berubah saat beban pengguna meningkat, yang dapat mempengaruhi latensi dan waktu pemuatan halaman. Hasil dari pengujian pada halaman *landing page* aplikasi *website* dapat dilihat pada Gambar 2.12 di bawah.



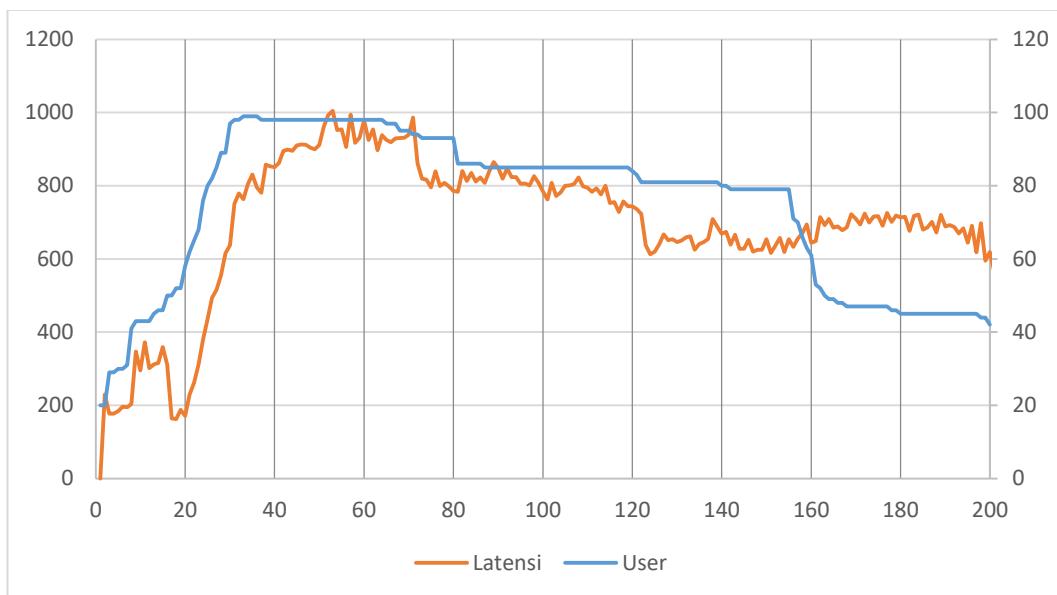
Gambar 2.12 Performa landing page

Gambar 2.12 di atas menampilkan grafik yang menggambarkan interaksi antara latensi (waktu respons *server*), pengguna (jumlah pengunjung), sampel (data yang dimuat), dan waktu (kecepatan pemuatan) pada *landing page*. Pada pengujian ini menggunakan 200 sampel pengujian. Ketika jumlah pengunjung naik maka waktu respons *server* juga akan naik, waktu yang dibutuhkan untuk mengakses data juga naik. Pada Gambar 4.46 di atas puncak dari latensi ditunjukkan ketika sample di angka 90 *user* dan waktu yang dibutuhkan sekitar 1200 milidetik. Hal itu

menunjukkan bahwa website dapat diakses oleh banyak pengguna secara bersamaan dengan waktu respon yang masih wajar.

2. Performa *login*

Halaman *login* diuji untuk memahami bagaimana interaksi antara latensi, pengguna, sampel, dan waktu mempengaruhi pengalaman pengguna saat mencoba masuk ke dalam sistem. Pengujian ini penting karena halaman *login* adalah titik masuk utama bagi pengguna untuk mengakses aplikasi atau *platform*, dan kinerjanya dapat sangat mempengaruhi kesan awal pengguna terhadap sistem. Hasil dari pengujian pada halaman *login* aplikasi *website* dapat dilihat pada Gambar 2.13 di bawah.



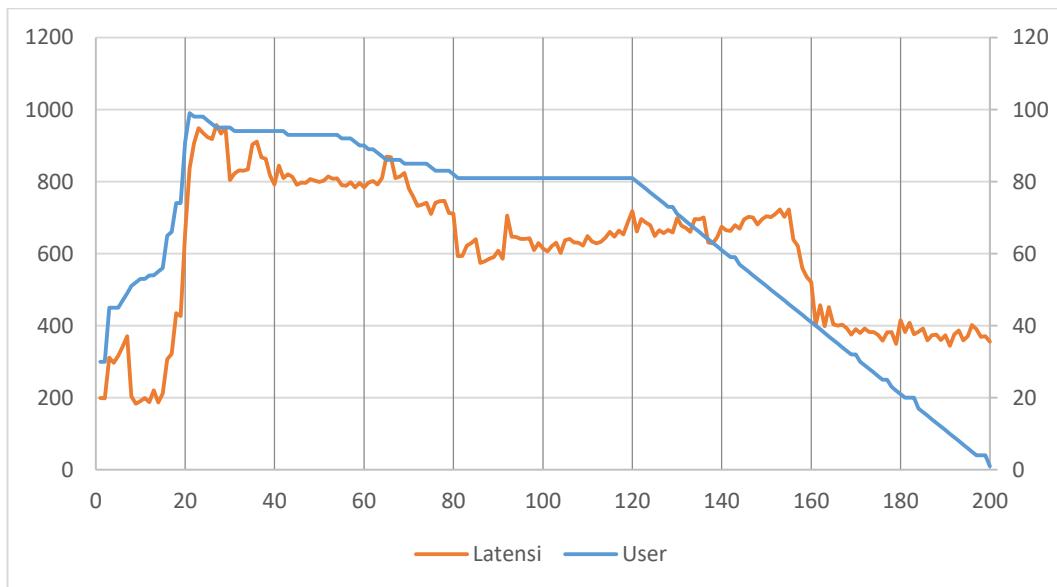
Gambar 2.13 Performa halaman *login*

Gambar 2.13 di atas menampilkan grafik yang menggambarkan interaksi antara latensi (waktu respons *server*), pengguna (jumlah pengunjung), sampel (data yang dimuat), dan waktu (kecepatan pemuatan) pada halaman *login*. Dalam uji coba ini, dilibatkan sebanyak 200 sampel pengujian. Saat jumlah pengunjung meningkat, terlihat peningkatan waktu respons dari *server*, serta durasi untuk mengakses data. Dalam Diagram 4.47 yang tercantum di atas, titik puncak latensi mencapai nilai tertinggi ketika sampel mencapai 60 pengguna, dengan durasi respons *server* hanya sekitar 1000 milidetik. Situasi ini mengindikasikan kemampuan halaman *login* dalam mengelola proses dengan performa pemuatan dan respon yang optimal. Hal

itu menunjukkan bahwa *website* dapat diakses oleh banyak pengguna secara bersamaan dengan waktu respon yang masih wajar.

3. Performa *reset password*

Halaman lupa *password* diuji untuk memahami bagaimana interaksi antara latensi (waktu respons *server*), pengguna (jumlah pengguna yang mengakses halaman tersebut), sampel (data yang diakses untuk mengelola permintaan lupa *password*), dan waktu mempengaruhi pengalaman pengguna saat menggunakan fitur ini. Pengujian ini penting karena halaman lupa *password* adalah bagian penting dari pengalaman pengguna dalam mengatasi masalah akses ke akun mereka. Hasil dari pengujian pada halaman *login* aplikasi *website* dapat dilihat pada Gambar 2.14 di bawah.



Gambar 2.14 Performa halaman *reset password*

Gambar 2.14 di atas menampilkan grafik yang menggambarkan interaksi antara latensi (waktu respons *server*), pengguna (jumlah pengunjung), sampel (data yang dimuat), dan waktu (kecepatan pemuatan) pada halaman lupa *password*. Dalam uji coba ini, dilibatkan sebanyak 200 sampel pengujian. Dalam skenario lonjakan pengunjung, terlihat lonjakan waktu respons *server* dan waktu akses data. Pada Diagram 4.48, puncak latensi terjadi saat sampel mencapai 20 pengguna, dengan respons *server* sekitar 1000 milidetik. Situasi ini mencerminkan performa baik

halaman *reset password* dalam mengelola pemuatan dan respon secara optimal dan menunjukkan bahwa *website* dapat diakses oleh banyak pengguna secara bersamaan dengan waktu respon yang masih wajar.

2.10 Pengujian Fungsionalitas Aplikasi Mobile

Pengujian fungsionalitas aplikasi mobile bertujuan untuk melihat kinerja dari fitur yang disediakan oleh aplikasi apakah sudah sesuai atau belum. Proses pengujian menggunakan metode blackbox dengan menggunakan berbagai kemungkinan interaksi pengguna terhadap aplikasi mobile. Hasil dari proses pengujian adalah sebagai berikut:

1. Autentikasi

Pengujian pada fungsi autentikasi bertujuan untuk mengidentifikasi potensi kerentanan, kesalahan, atau masalah keamanan yang dapat dieksloitasi oleh pihak yang tidak berwenang. Pengujian pada halaman autentikasi dilaksanakan untuk menguji keberhasilan dari fungsi *login* dan *logout* yang bekerja pada halaman tersebut. Hasil pengujian halaman *authentication* ditunjukkan pada Tabel 4.3 di bawah.

Tabel 2.37 Hasil pengujian autentikasi

No.	Nama	Bentuk Pengujian	Hasil Pengujian	Status
1.	<i>Login</i> dengan data benar	Melakukan <i>login</i> menggunakan <i>email</i> dan <i>password</i> yang sesuai	Berhasil melakukan <i>login</i> dan mendapatkan akses untuk masuk ke halaman <i>dashboard</i>	Berhasil
2.	<i>Login</i> dengan data salah	Melakukan <i>login</i> dengan menggunakan <i>email</i> atau <i>password</i> yang salah	Tidak mendapatkan akses masuk ke halaman <i>dashboard</i>	Gagal

Tabel 2.37 (lanjutan)

No.	Nama	Bentuk Pengujian	Hasil Pengujian	Status
3.	<i>Login</i> dengan data kurang	Melakukan <i>login</i> dengan menggunakan <i>email</i> saja atau <i>password</i> saja	Tampil peringatan untuk mengisi kolom yang masih kosong.	Gagal
4.	<i>Login</i> dengan data tidak terdaftar	Melakukan <i>Login</i> dengan menggunakan <i>email</i> yang belum terdaftar	Tampil peringatan “ <i>Login Gagal</i> ”. Data tidak ditemukan.	Gagal
5.	<i>Login</i> dengan format tidak sesuai	Melakukan <i>Login</i> dengan menggunakan <i>email</i> . Username bukan <i>email</i>	Tampil peringatan untuk menyertakan “@” pada format penulisan <i>email</i> .	Gagal
6.	<i>Login</i> dengan <i>email</i> belum terverifikasi	Melakukan <i>Login</i> dengan menggunakan <i>email</i> yang belum terverifikasi	Tampil peringatan <i>login Gagal</i> dan data tidak ditemukan	Gagal
7.	<i>Logout</i> dengan menekan tombol <i>logout</i>	Melakukan Tindakan untuk <i>logout</i> dengan menekan tombol <i>logout</i>	Akan Kembali keluar ke halaman <i>login</i>	Berhasil

Pada Tabel 2.37 terlihat bahwa proses autentikasi *login* berhasil apabila *email* dan *password* sesuai dengan format yang ditentukan. Proses *login* dilakukan dengan menekan tombol *login* pada formulir dan untuk proses *logout* dilakukan dengan menekan tombol *logout*.

2. Verifikasi *email*

Pengujian pada halaman verifikasi *email* dilaksanakan untuk menguji keberhasilan dari fungsi verifikasi *email* dengan memasukkan token yang sesuai. Selain itu juga untuk menguji keberhasilan *fungsi* yang bekerja pada halaman tersebut. Hasil pengujian halaman verifikasi *email* ditunjukkan oleh Tabel 2.38 di bawah.

Tabel 2.38 Hasil Pengujian verifikasi email

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Verifikasi <i>email</i> tanpa <i>token</i>	Melakukan verifikasi <i>email</i> tanpa menggunakan <i>token</i>	Tampil peringatan untuk mengisi <i>token</i>	Gagal
2.	Verifikasi <i>email</i> tidak sesuai	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang salah	Tampil peringatan kode OTP tidak sesuai	Gagal
3.	Verifikasi <i>email</i> kadaluwarsa.	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang sudah kadaluwarsa	Tampil peringatan OTP sudah kadaluwarsa	Gagal
4.	Verifikasi <i>email</i> sesuai	Melakukan verifikasi <i>email</i> menggunakan kode OTP yang sesuai	Verifikasi <i>email</i> berhasil	Berhasil
5.	Verifikasi <i>email</i> <i>token</i> salah	Melakukan verifikasi <i>email</i> menggunakan <i>token</i> yang sudah terhapus	Kode OTP tidak diketahui	Gagal

Tabel 2.38 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
6.	Verifikasi <i>email token</i> lebih dari 6 digit	Melakukan verifikasi <i>email token</i> yang lebih dari 6 digit menggunakan	Tampil peringatan bahwa <i>token</i> lebih dari 6 digit	Gagal

Dapat dilihat pada Tabel 2.38 di atas bahwa untuk melakukan verifikasi email pengguna diwajibkan mengisi email yang sesuai pada halaman kirim email. Kemudian email akan mengirimkan token yang akan diisi ke halaman isi token. Verifikasi akan berhasil jika token yang dimasukkan benar dan juga pengisian token tidak melewati batas waktu.

3. *Reset password*

Pengujian fungsi *reset password* pada aplikasi Android bertujuan memverifikasi kemampuan pengguna untuk dengan berhasil mengubah kata sandi mereka dengan aman dan mudah, khususnya dalam situasi di mana mereka lupa atau ingin mengganti kata sandi. Hasil pengujian halaman *reset password* ditunjukkan oleh Tabel 2.39 di bawah.

Tabel 2.39 Pengujian Fungsi *Reset Password*

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Menampilkan halaman <i>Reset Password</i>	Melakukan akses dengan menekan tombol lupa <i>password</i>	Halaman <i>Reset Password</i> berhasil ditampilkan	Berhasil
2.	Pengujian <i>Reset Password</i>	Mengisi form <i>Reset Password</i> dengan <i>email</i> yang terdaftar	Mendapat <i>email</i> balasan berupa link untuk mengakses <i>Reset Password</i>	Berhasil

Tabel 2.39 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
3.	Pengujian melakukan <i>Reset Password</i>	Mengisi <i>password baru</i> dan konfirmasi <i>password baru</i>	Terkonfirmasi dan masuk ke halaman <i>Login</i> untuk melakukan akses <i>Login</i> .	Berhasil
4.	Menggunakan <i>email</i> yang tidak terdaftar	Memasukkan <i>email</i> yang tidak terdaftar atau salah memasukkan <i>Alamat email</i>	Tampil peringatan bahwa alamat <i>email</i> tidak ditemukan	Gagal

Dapat dilihat pada Tabel 2.39 di atas bahwa pengguna berhasil melakukan akses untuk *reset password* dengan menekan tombol lupa *password* kemudian mengisi formulir dengan *email* yang sesuai. Apabila format *email* tidak sesuai maka *reset password* gagal.

4. *Update password*

Pengujian fungsi *update password* pada pengujian aplikasi Android memiliki tujuan untuk memastikan bahwa pengguna dapat mengganti kata sandi yang ada dengan sukses, dan bahwa perubahan tersebut terjadi dengan aman dan sesuai dengan harapan. Hasil pengujian halaman *update password* ditunjukkan oleh Tabel 2.40 di bawah.

Tabel 2.40 Hasil pengujian update password

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Menampilkan formulir <i>Update Password</i>	Melakukan akses dengan menekan tombol ganti <i>password</i>	Halaman <i>Update Password</i> berhasil ditampilkan	Berhasil

Tabel 2.40 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
2.	Pengujian <i>Update Password</i>	Mengisi form konfirmasi <i>password</i> dengan <i>password</i> saat ini, <i>password</i> baru, dan konfirmasi <i>password</i> dengan benar.	<i>Update Password</i>	Berhasil berhasil
3.	Konfirmasi <i>Update Password</i>	Mengisi form konfirmasi <i>password</i> dengan <i>password</i> yang tidak sesuai	Tampil peringatan bahwa ada yang salah dengan itu <i>password</i> tidak dapat diperbarui dan konfirmasi <i>password</i> tidak sesuai	Gagal

Dapat dilihat pada Tabel 2.40 di atas bahwa pengguna berhasil melakukan akses untuk *update password* dengan menekan tombol ganti *password* kemudian mengisi formulir dengan email yang sesuai. Apabila format email tidak sesuai maka *update password* gagal.

5. Halaman *dashboard*

Pengujian fungsi halaman *dashboard* dalam pengujian aplikasi Android memiliki tujuan untuk memastikan bahwa halaman ini berfungsi dengan baik, memberikan informasi yang relevan, dan menawarkan pengalaman pengguna yang baik. Hasil pengujian halaman *dashboard* ditunjukkan oleh Tabel 2.41 di bawah.

Tabel 2.41 Hasil pengujian halaman dashboard

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Menampilkan halaman <i>Dashboard</i>	Melakukan akses dengan menyelesaikan tahapan <i>authentication</i> dengan benar	Halaman <i>Dashboard</i> berhasil ditampilkan	Berhasil
2.	Menampilkan gambar informasi tentang aplikasi dan sistem penguncian	Menggeser gambar informasi dengan melakukan scroll horizontal	Halaman gambar informasi dapat discroll secara horizontal saat halaman dibuka	Berhasil

Dapat dilihat pada Tabel 2.41 di atas bahwa pengguna berhasil menampilkan halaman *dashboard* dengan menyelesaikan tahapan *authentication* dengan benar. Setelah menyelesaikan tahap *authentication* maka akan masuk ke halaman *dashboard* dengan beberapa menu di dalamnya dan juga papan informasi yang berada di bagian dashboard.

6. Daftar pintu

Pengujian fungsi halaman daftar pintu pada pengujian aplikasi Android memiliki tujuan untuk memastikan bahwa pengguna dapat dengan efektif melihat dan mengelola daftar pintu yang ada dalam aplikasi tersebut. Hasil pengujian halaman daftar pintu ditunjukkan oleh Tabel 2.42 di bawah.

Tabel 2.42 Hasil pengujian daftar pintu

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu daftar pintu pada halaman <i>Dashboard</i>	Melakukan akses dengan menekan menu daftar pintu	Halaman daftar pintu berhasil ditampilkan	Berhasil

Tabel 2.42 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
2.	Menampilkan daftar pintu yang terdaftar dalam aplikasi	Melakukan akses untuk melihat daftar pintu	Daftar pintu berhasil ditampilkan	Berhasil
3.	Menampilkan daftar pintu Ketika belum ada pintu yang terdaftar	Melakukan akses untuk melihat daftar pintu	Daftar pintu tidak ditemukan	Gagal

Dapat dilihat pada Tabel 2.42 di atas bahwa pengguna berhasil menampilkan halaman daftar pintu dengan menekan menu daftar pintu. Daftar pintu akan ditampilkan apabila daftar pintu sudah terdaftar dalam aplikasi. Apabila belum terdaftar maka daftar pintu tidak dapat ditampilkan.

7. Daftar kunci

Pengujian fungsi halaman daftar kunci pada pengujian aplikasi Android bertujuan untuk memverifikasi bahwa pengguna mampu dengan sukses mengakses, mengamati, dan mengelola daftar kunci secara efisien dalam aplikasi tersebut. Hasil pengujian halaman daftar pintu ditunjukkan oleh Tabel 2.43 di bawah.

Tabel 2.43 Hasil pengujian daftar kunci

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu daftar kunci	Menekan menu daftar kunci	daftar kunci berhasil ditampilkan	Berhasil
2.	Menampilkan daftar kunci ketika belum ada daftar kunci yang terdaftar	Melakukan akses untuk melihat daftar kunci	Daftar kunci tidak ditemukan	Gagal

Dapat dilihat pada Tabel 2.43 di atas bahwa pengguna berhasil menampilkan halaman daftar kunci dengan menekan menu daftar kunci. Daftar kunci akan ditampilkan apabila sudah terdaftar dalam aplikasi. Apabila belum terdaftar maka daftar kunci tidak dapat ditampilkan.

8. Riwayat akses

Pengujian fungsi halaman riwayat akses pada pengujian aplikasi Android memiliki tujuan untuk memastikan bahwa pengguna dapat mengakses dan melihat riwayat akses yang telah terjadi dengan akurat dan mudah di dalam aplikasi. Hasil pengujian halaman daftar pintu ditunjukkan oleh Tabel 2.44 di bawah.

Tabel 2.44 Hasil pengujian riwayat akses

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu riwayat akses	Melakukan akses dengan menekan menu riwayat akses	Halaman riwayat akses berhasil ditampilkan	Berhasil
2.	Menampilkan Riwayat akses yang terdaftar dalam aplikasi	Melakukan akses untuk melihat halaman riwayat akses	Halaman riwayat akses berhasil ditampilkan	Berhasil
3.	Menampilkan halaman riwayat akses ketika belum dilakukan akses ke pintu.	Melakukan akses untuk melihat halaman riwayat akses	Halaman riwayat akses tidak ditemukan	Gagal

Dapat dilihat pada Tabel 2.44 di atas bahwa pengguna berhasil menampilkan halaman riwayat akses dengan menekan menu riwayat akses. Riwayat akses akan ditampilkan apabila sudah terdaftar dalam aplikasi. Apabila belum terdaftar maka riwayat akses tidak dapat ditampilkan.

9. Utilitas

Pengujian fungsi halaman utilitas pada pengujian aplikasi Android memiliki tujuan untuk memastikan bahwa pengguna dapat menggunakan berbagai utilitas atau alat bantu untuk mengatur kunci IoT, pengaturan wifi dan melakukan register ke server. Hasil pengujian halaman daftar pintu ditunjukkan oleh Tabel 2.45 di bawah.

Tabel 2.45 Hasil pengujian utilitas

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka menu utilitas	Melakukan akses dengan menekan menu utilitas	Halaman utilitas berhasil ditampilkan	Berhasil
2.	Melakukan pengaturan wifi	Mengisi SSID dan <i>password</i> dengan benar	Pengaturan wifi terkonfirmasi dan terkirim	Berhasil
3.	Melakukan pengaturan wifi dengan SSID dan <i>password</i> salah	Melakukan akses pengaturan dengan memasukkan SSID dan <i>Password</i> yang salah	Peringatan bahwa perangkat kunci kosong	
4.	Melakukan register ke <i>server</i>	Menekan tombol Pindai QR Pindai QR Pendaftaran	Pindai QR pendafataran terkonfirmasi	Berhasil

Dapat dilihat pada Tabel 2.45 di atas bahwa pengguna berhasil menampilkan halaman utilitas akun dengan menekan menu utilitas. Halaman utilitas akun berfungsi untuk melakukan akses pengaturan dengan memasukkan SSID dan *password*. Pengaturan wifi berhasil apabila SSID dan *password* yang dimasukkan sesuai.

10. *Update profile*

Pengujian fungsi halaman *update profile* pada pengujian aplikasi Android adalah untuk memverifikasi bahwa pengguna dapat mengubah dan memperbarui informasi

dalam profil mereka dengan akurat dan mudah. Hasil pengujian halaman daftar pintu ditunjukkan oleh Tabel 2.46 di bawah.

Tabel 2.46 Pengujian fungsi update profile

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
1.	Membuka halaman <i>update profil</i>	Melakukan akses dengan menekan tombol <i>update profile</i>	Berhasil masuk ke halaman <i>update profile</i>	Berhasil
2.	Memasukkan <i>email</i> dengan benar	Memasukkan alamat <i>email</i> ke formulir dengan format yang benar	Berhasil memperbarui <i>email</i>	Berhasil
3.	Memasukkan <i>email</i> dengan salah	Memasukkan alamat <i>email</i> ke formulir dengan format yang salah	<i>Email</i> gagal diperbarui dan terdapat peringatan ada yang salah dari sistem	Gagal
4.	Mengganti nama pengguna menjadi nama yang baru	Memasukkan nama pengguna dengan nama baru yang berbeda	Nama berhasil diperbarui menjadi nama baru	berhasil
5.	Mengganti jenis kelamin pengguna	Mengganti jenis kelamin pengguna dengan memilih jenis kelaamin yang berbeda	Jenis kelamin berhasil terganti dengan.	Berhasil

Tabel 2.46 (lanjutan)

No.	Nama Pengujian	Bentuk Pengujian	Hasil Pengujian	Status
6.	Mengganti nomor HP pengguna dengan nomor yang salah	Mengganti nomor dengan nomor yang salah dan belum terdaftar	Nomor HP pengguna tidak dapat diperbarui dan terdapat peringatan ada yang salah dari sistem	Gagal
7.	Mengganti nomor HP pengguna dengan nomor yang benar	Mengganti nomor dengan nomor benar dan sudah terdaftar	Nomor HP pengguna berhasil diperbarui	Berhasil

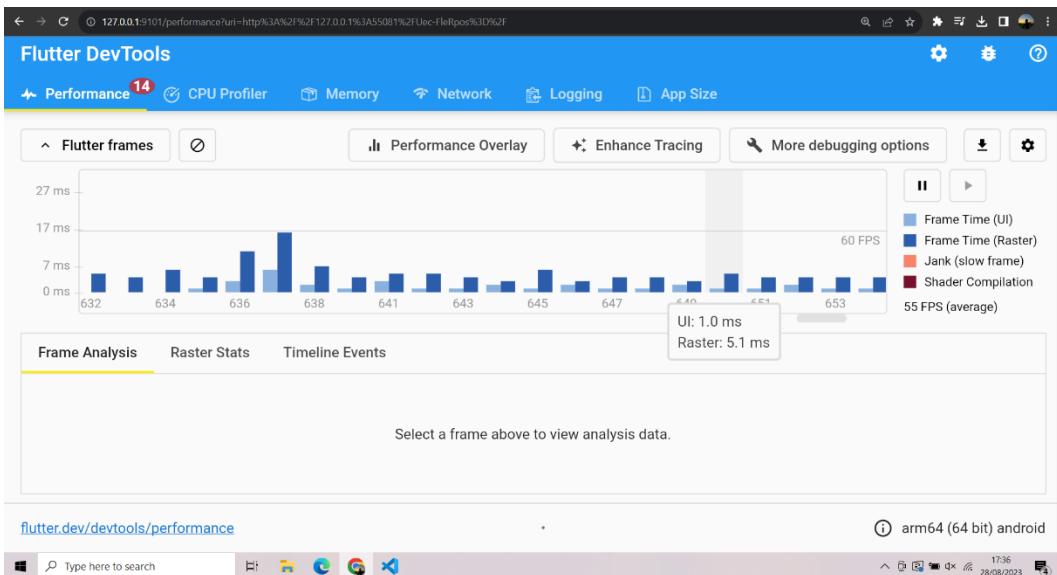
Dapat dilihat pada Tabel 2.46 di atas bahwa pengguna berhasil menampilkan halaman *update profile* dengan menekan menu *update profile*. Halaman profil berisi menu untuk mengganti *email*, nama pengguna, jenis kelamin, mengganti nomor HP. Beberapa menu tersebut berhasil diganti apabila menggunakan format dan data yang sesuai.

2.11 Pengujian Performa Aplikasi Mobile

Pengujian ini berfokus pada performa tampilan, alokasi memori, dan koneksi internet dalam aplikasi mobile yang memiliki peran sentral dalam memastikan responsivitas antarmuka, manajemen memori yang efisien, serta performa yang konsisten dalam beragam kondisi jaringan. Hasil dari proses pengujian performa aplikasi mobile adalah sebagai berikut:

1. Performa tampilan

Pengujian performa tampilan pada aplikasi Android adalah proses untuk memastikan bahwa aplikasi Android dapat berjalan dengan lancar, responsif, dan memberikan pengalaman pengguna yang baik. Hasil dari pengujian performa tampilan aplikasi android dapat dilihat pada Gambar 2.15 di bawah.

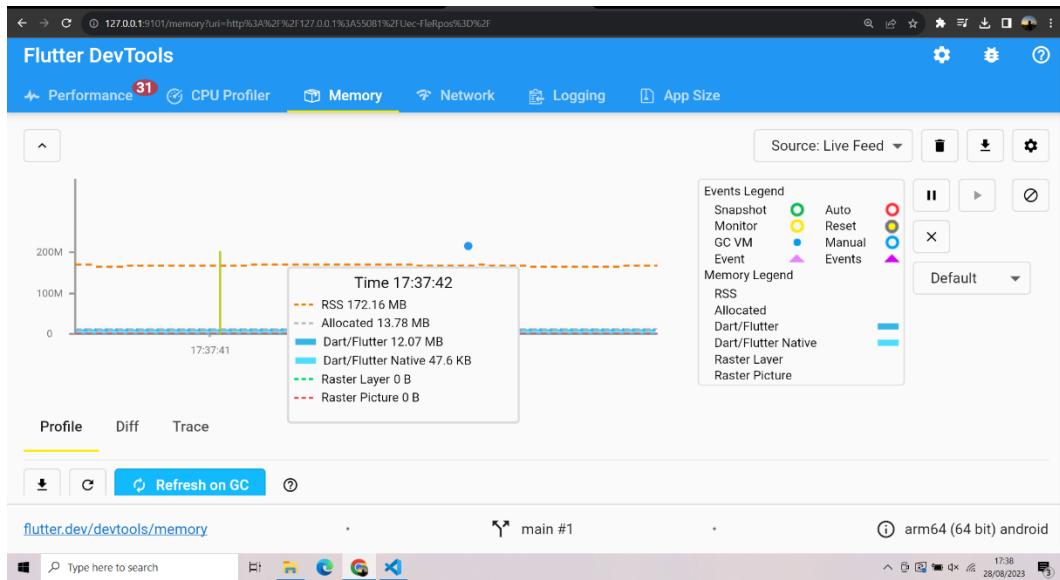


Gambar 2.15 Performa tampilan aplikasi

Gambar 2.15 di atas menampilkan hasil pengujian performa tampilan untuk mengukur kinerja aplikasi yang mengacu pada waktu yang dibutuhkan untuk memproses dan merender elemen-elemen antarmuka pengguna (UI) yang berupa *frame time* (UI) dan juga *frame time* (raster) untuk mengukur waktu yang diperlukan untuk merender elemen grafis menjadi gambar *raster* yang siap ditampilkan di *layer*. Selain itu juga menampilkan nilai rata-rata FPS (*Frame Per Second*) yang merupakan jumlah rata-rata *frame* yang ditampilkan per detik dalam suatu aplikasi. Pada gambar pengukuran tersebut ditampilkan *frame time* (UI) dengan rata-rata waktu 3 milisecond, *frame time* (raster) dengan rata-rata waktu 5 milisecond dan untuk nilai rata-rata FPS berkisar pada 55 FPS.

2. Penggunaan memori

Pengujian performa memori pada aplikasi Android merupakan langkah untuk mengamati dan mengukur bagaimana aplikasi menggunakan memori. Masalah dalam performa memori bisa menghasilkan konsekuensi seperti respons yang lambat, kegagalan (*crash*), atau penutupan aplikasi yang tiba-tiba. Tujuan dari pengujian ini adalah untuk memastikan bahwa aplikasi menggunakan sumber daya memori dengan efisien dan mencegah potensi permasalahan yang mungkin muncul karena penggunaan memori yang berlebihan. Hasil dari pengujian performa memori aplikasi Android dapat dilihat pada Gambar 2.16 di bawah.



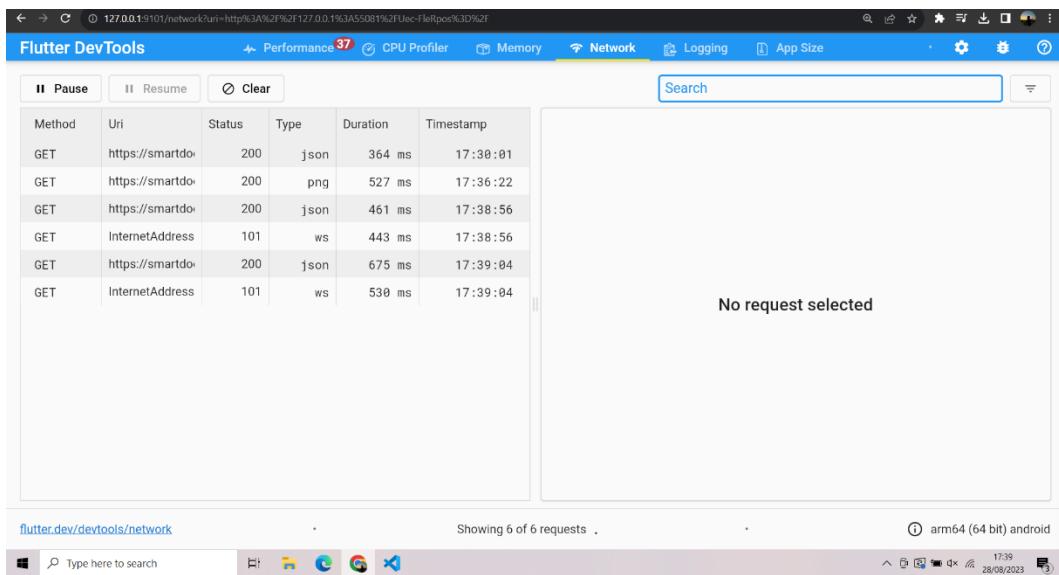
Gambar 2.16 Penggunaan memori

Gambar 2.16 di atas menampilkan hasil pengujian performa memory aplikasi. Pada pengujian performa tersebut, data yang ditampilkan seperti RSS (*Resident Set Size*), *allocated* (alokasi memori), *dart/flutter*, *dart/flutter native*, *raster layer*, dan *raster picture* menggambarkan berbagai aspek penggunaan memori dalam aplikasi. RSS menunjukkan seberapa banyak memori fisik yang aktif digunakan, sedangkan *allocated* mengukur total alokasi memori termasuk yang aktif dan tidak aktif. *dart/flutter* dan *dart/flutter native* mengacu pada alokasi memori yang terkait dengan bahasa pemrograman dan kerangka kerja. Penggunaan memori dalam *raster layer* dan *raster picture* mengindikasikan penggunaan memori oleh elemen grafis yang ditampilkan. Pada Gambar 4.44 di atas menampilkan nilai dari RSS (*Resident Set Size*) sebesar 172.16 MB, *allocated* (alokasi memori) sebesar 13.78 MB, penggunaan memori yang terkait dengan dart dan flutter sebesar 12.7 MB, alokasi memori untuk kode *Dart* yang dijalankan dalam *mode native* sebesar 47.6 KB dan untuk *raster layer* dan *raster picture* sebesar 0 B.

3. Performa internet

Hasil pengujian performa aplikasi Android terkait dengan kinerja internet mencakup sejumlah informasi penting yang menggambarkan bagaimana aplikasi

berperilaku saat berinteraksi dengan jaringan. Hasil dari pengujian performa internet aplikasi Android dapat dilihat pada Gambar 2.17 di bawah.



Gambar 2.17 Performa koneksi internet

Gambar 2.17 di atas menampilkan hasil performa internet saat aplikasi digunakan. Pada gambar 4.45 di atas menampilkan tipe data yang diakses, durasi yang dibutuhkan dan juga waktu saat melakukan akses. Dapat disimpulkan untuk rata-rata durasi waktu yang dibutuhkan untuk mengakses data adalah 500 ms.

3. PENUTUP

Proses pengujian pada produk tugas akhir “Sistem Keamanan Kunci Pintu Gedung Berbasis *Internet of Things*” berhasil dijalankan dengan hasil yang sudah dijelaskan. Proses pengujian memberikan informasi terkait presentase keberhasilan yang telah dicapai serta kemampuan dari sistem yang dihasilkan.

Berdasarkan pengujian yang telah dilakukan, secara keseluruhan sistem yang telah diimplementasikan telah sesuai desain awal yang telah ditentukan. Dibuktikan dengan proses pengujian yang dilakukan. Proses pengujian juga membuktikan bahwa sistem yang dikembangkan memiliki performa yang bagus baik dari segi tampilan, penanganan pengguna dan lain sebagainya.