RANCANGAN APLIKASI *SMARTLOCK FOR DOOR* DENGAN SIDIK JARI BERBASIS ANDROID PADA MIN 2 BANGKA

PROYEK AKHIR

Laporan akhir ini dibuat dan diajukan untuk memenuhi salah satu syarat kelulusan Sarjana Terapan/Diploma IV Politeknik Manufaktur Negeri Bangka Belitung

Disusun Oleh :

Wahyu Rizky Algifari	NPM 1062028
Wisnu Fauzaan Pahlevi	NPM 1062002

POLITEKNIK MANUFAKTUR NEGERI BANGKA BELITUNG

TAHUN 2024

LEMBAR PENGESAHAN

JUDUL PROYEK AKHIR

RANCANGAN APLIKASI *SMARTLOCK FOR DOOR* DENGAN SIDIK JARI BERBASIS ANDROID PADA MIN 2 BANGKA

Oleh :

Wahyu Rizky Algifahri/1062028

Wisnu Fauzaan Pahlevi/1062002

Laporan akhir ini telah disetujui dan disahkan sebagai salah satu syarat kelulusan Program Sarjana Terapan Politeknik Manufaktur Negeri Bangka Belitung

Menyetujui,

Pembimbing 1	Pembimbing 2
Ahmat Josi, M.Kom	Riki Afriansyah, M.T
NIP. 198908202019031015	NIP. 199004042019031013

Penguji 1	Penguji 2
Sidhiq Andriyanto, M.Kom	Nur Khasanah, S.P., M. Si
NIP. 199007182019031011	NIP. 199110302022032012

PERNYATAAN BUKAN PLAGIAT

Yang bertanda tangan di bawah ini :

Nama Mahasiswa	:	Wahyu Rizky Algifari	NPM	:	1062028
Nama Mahasiswa	:	Wisnu Fauzaan Pahlevi	NPM	:	1062002

Dengan Judul : Rancangan Aplikasi *Smartlock For Door* Dengan Sidik Jari Berbasis Android Pada Min 2 Bangka

Menyatakan bahwa laporan akhir ini adalah hasil kerja saya sendiri dan bukan merupakan plagiat. Pernyataan ini saya buat dengan sebenarnya dan bila ternyata dikemudian hari ternyata melanggar pernyataan ini, saya bersedia menerima sanksi yang berlaku.

Sungailiat, 11 Januari 2024

Nama Mahasiswa	Tanda Tangan
1. Wahyu Rizky Algifari	.....
2. Wisnu Fauzaan Pahlevi	.....

ABSTRAK

Dalam era perkembangan teknologi yang pesat, penerapan teknologi memiliki peran krusial dalam memastikan keamanan berbagai aspek kehidupan manusia, termasuk pada sektor pendidikan seperti sekolah dasar. Salah satu inovasi yang menonjol adalah penggunaan smartlock for door dengan akses sidik jari berbasis Android. Penelitian ini dilakukan di MIN 2 Bangka dengan tujuan meningkatkan keamanan dan efisiensi akses. Kepala sekolah mengidentifikasi kebutuhan akan sistem keamanan yang lebih canggih untuk melindungi informasi sensitif dan aset sekolah. Melalui pendekatan studi kasus dan metode Agile, penelitian ini mengembangkan aplikasi yang memungkinkan pengontrolan pintu menggunakan sensor sidik jari dan alternatif kontrol manual berbasis Internet of Things (IoT). Aplikasi ini dilengkapi dengan fitur monitoring, setting, manajemen sidik jari, dan fungsi kontrol pintu. Hasil penelitian menunjukkan bahwa sistem mampu mengenali sidik jari dengan akurasi, menyimpan hingga 127 data sidik jari, dan membatasi akses hanya untuk pengguna terdaftar. Namun, tantangan yang muncul meliputi ketergantungan pada kualitas sinyal WiFi yang dapat mempengaruhi kinerja solenoid. Untuk mengatasi hal ini, penelitian mengusulkan solusi berupa penggunaan jaringan WiFi yang konsisten untuk memastikan koneksi yang stabil antara perangkat lunak dan perangkat keras, sehingga sistem dapat beroperasi dengan efektif dan efisien.

Kata kunci: *IOT, Pintu, Sidik jari, Wifi.*

ABSTRACT

In an era of rapid technological development, the application of technology has a crucial role in ensuring the security of various aspects of human life, including in the education sector such as elementary schools. One of the innovations that stands out is the use of smart locks for doors with Android-based fingerprint access. This research was conducted at MIN 2 Bangka with the aim of increasing security and access efficiency. The principal identified the need for a more sophisticated security system to protect sensitive information and school assets. Through a case study approach and Agile methods, this research develops an application that allows door control using fingerprint sensors and Internet of Things (IoT) based manual control alternatives. This application is equipped with monitoring, setting, fingerprint management and door control functions. The research results show that the system is able to recognize fingerprints with accuracy, stores up to 127 fingerprint data, and limits access to registered users only. However, challenges that arise include dependence on WiFi signal quality which can affect solenoid performance. To overcome this, research recommends a solution in the form of using a consistent WiFi network to ensure a stable connection between software and hardware, so that the system can operate effectively and efficiently.

Key word: Door, Fingerprint, IOT, Wifi.

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh. Segala puji bagi Allah SWT., yang telah melimpahkan taufiq dan hidayah-Nya kepada penulis, sehingga penulis dapat menyusun laporan proyek akhir ini dengan Judul "RANCANGAN APLIKASI *SMARTLOCK FOR DOOR* DENGAN SIDIK JARI BERBASIS ANDROID PADA MIN 2 BANGKA " dan dapat menyelesaikan Program Studi Diploma IV Teknologi Rekayasa Perangkat Lunak di Politeknik Manufaktur Negeri Bangka Belitung. Penulis menyadari sepenuhnya, bahwa dalam penyusunan laporan proyek akhir ini banyak terdpat kekurangan mengingat terbatasnya kemampuan penulis, namun berkat rahmat Allah SWT, serta pengarahan dari berbagai pihak. akhirnya laporan proyek akhir ini dapat diselesaikan. Harapan penulis semoga laporan proyek akhir ini dapat bermanfaat untuk kepentingan bersama. Sehubungan dengan itu, penulis menyampaikan ucapan terima kasih kepada:

1. Bapak I Made Andik Setiawan, M.Eng, Ph.D selaku Direktur Politeknik Manufaktur Negeri Bangka Belitung.
2. Bapak Irwan, M.Sc, Ph.D selaku Wakil Direktur I Politeknik Manufaktur Negeri Bangka Belitung.
3. Bapak Muhammad Subhan, M.T selaku Wakil Direktur II Politeknik Manufaktur Negeri Bangka Belitung.
4. Bapak Eko Sulistyo, M.T selaku Wakil Direktur III Politeknik Manufaktur Negeri Bangka Belitung.
5. Bapak Ahmat Josi, M.Kom selaku Ka. Prodi D4 Teknologi Rekayasa Perangkat Lunak dan juga Dosen Pembimbing Utama Proyek Akhir.
6. Bapak Zanu Saputra, S.ST, M.TR.T. selaku Ka. Jurusan Teknik Elektronika dan Informatika Politeknik Manufaktur Negeri Bangka Belitung.
7. Bapak Riki Afriansyah selaku Dosen Pembimbing kedua Proyek Akhir.

8. Teman-teman seperjuangan yang telah memberikan support dan dukungannya.
9. Ibu dan Ayah tercinta serta seluruh keluarga yang dengan penuh keikhlasan untuk dukungannya.

Setelah melalui proses yang panjang dan penuh tantangan, akhirnya penulis dapat menyelesaikan pembuatan alat dan laporan proyek akhir ini. Penulis menyadari bahwa penulisan makalah proyek akhir ini masih terdapat banyak kekurangan baik dari segi penulisan maupun isi makalah. Oleh karena itu, penulis mengharapkan segala petunjuk, kritik, dan saran yang membangun dari pembaca agar dapat menunjang pengembangan dan perbaikan penulisan selanjutnya. Akhir kata, penulis mohon maaf atas kekurangan dalam penulisan makalah ini dan penulis dengan senang hati menerima saran dan kritik yang membangun dari pembaca. Semoga makalah ini dapat berguna untuk menambah wawasan dan wacana bagi rekan-rekan mahasiswa. Waalaikumsalam warahmarullahi wabarakatuh.

Sungailiat, 11 Januari 2024

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

MIN 2 Bangka adalah sebuah sekolah Madrasah Ibtidaiyah Negeri yang terletak di wilayah Bangka tepatnya pada daerah sungailiat. Menurut Peraturan Pemerintah Nomor 66 Tahun 2010 tentang Pengelolaan dan Penyelenggaraan Pendidikan, Madrasah Ibtidaiyah, yang selanjutnya disingkat MI, adalah salah satu bentuk satuan pendidikan formal dalam binaan Menteri Agama yang menyelenggarakan pendidikan umum dengan kekhasan agama Islam pada jenjang pendidikan dasar[1].

Teknologi pada saat ini berkembang dengan sangat pesat serta merupakan salah satu bidang yang mempunyai peran yang sangat penting dibeberapa aspek kehidupan manusia, termasuk pada bidang *security*[2] suatu sekolahan. Penggunaan teknologi untuk masuk ke dalam sebuah ruangan telah berkembang dari menggunakan kunci manual menjadi menggunakan password atau sidik jari[3]. Salah satu contoh teknologi yang dapat diterapkan pada sekolah dasar dalam sistem keamanan adalah *smartlock for door*, yaitu sebuah sistem pengamanan pintu dengan akses sidik jari yang dapat dikontrol melalui aplikasi berbasis android.

Kepala sekolah MIN 2 Bangka memiliki kekhawatiran terhadap akses tanpa izin ke ruangan, yang dapat membahayakan informasi sensitif atau barang berharga. Selain itu, kepala sekolah juga memiliki tanggung jawab atas keamanan alat serta berkas-berkas karyawan di sekolah. Keamanan menjadi salah satu faktor yang mempengaruhi kehidupan manusia, karena jika pada suatu tempat atau ruangan kurang aman maka hal tersebut sangat berpengaruh terhadap segala aspek kehidupan pada lingkungan tersebut[27]. Pengunaan kunci konvensional juga tidak efektif karena waktu yang dibutuhkan untuk membukanya lebih lama dibandingkan penggunaan sidik

jari, serta kunci konvensional yang sering hilang menghambat proses pembukaan pintu.

Dengan begistu kami menawarkan *Smartlock for door* dengan sidik jari yang merupakan salah satu solusi yang tepat untuk mengatasi masalah keamanan di sekolah. Dengan adanya Rancangan Aplikasi *Smartlock for Door* dengan Sidik Jari Berbasis Android, diharapkan dapat membantu meningkatkan keamanan sekolah di MIN 2 Bangka. Dalam penelitian ini menerapkan dua mekanisme pengontrolan pintu rumah yaitu dengan memanfaatkan sensor sidik jari sebagai sistem kontrol *biometric*, dan sistem kontrol manual berbasis *Internet of Things* untuk langkah *alternative*[4], sehingga lebih praktis dan efisien dibandingkan dengan kunci konvensional. Selain itu, dengan menggunakan teknologi *smartlock for door*, pengguna dapat memonitor semua aktivitas pintu secara *real-time*.

1.2. Perumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, terdapat beberapa masalah yang perlu dipecahkan, yaitu:

a) Bagaimana merancang dan membuat aplikasi *smartlock for door* yang dapat diintegrasikan dengan sensor sidik jari dan sistem kontrol berbasis *Internet of Things*?
b) Bagaimana cara mengimplementasikan dan mengintegrasikan sistem *smartlock for door* dengan infrastruktur yang ada di MIN 2 Bangka?
c) Bagaimana mengoptimalkan keamanan pintu ruangan kepala sekolah dengan memanfaatkan teknologi *smartlock for door* dan fitur pemantauan aktivitas pintu secara *real-time*?
d) Bagaimana mengevaluasi keefektifan dan kehandalan sistem *smartlock for door* dengan sidik jari yang diimplementasikan dalam meningkatkan keamanan sekolah di MIN 2 Bangka?

1.3. Tujuan Proyek Akhir

Berdasarkan rumusan masalah yang telah dijelaskan, tujuan dari penelitian ini adalah :

a) Merancang dan membuat aplikasi *Smartlock for Door* dengan Sidik Jari Berbasis Android yang dapat diintegrasikan dengan sensor sidik jari dan sistem kontrol berbasis *Internet of Things*.
b) Mengimplementasikan dan mengintegrasikan sistem *Smartlock for Door* dengan Sidik Jari di MIN 2 Bangka.
c) Optimalisasi keamanan pintu ruangan kepala sekolah dengan memanfaatkan teknologi *Smartlock for Door* dengan pemantauan aktivitas pintu secara *real-time*.
d) Mengevaluasi keefektifan dan kehandalan sistem *Smartlock for Door* dengan Sidik Jari yang diimplementasikan di MIN 2 Bangka.

BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Penulis membuat tinjauan pustaka dengan merujuk pada penelitian sebelumnya yang relevan dengan topik penelitian ini. Di bawah ini terdapat hasil dari penelitian yang telah dilakukan sebelumnya:

Tabel 2 1 *Studi Literatur*

No	Judul	Hasil
a)	Perancangan Sistem *Doorlock* Menggunakan Aplikasi Blynk Berbasis IoT Studi Kasus Pada Rumah Tempat Tinggal Pribadi[5].	Perancangan sistem *doorlock* menggunakan aplikasi blynk berbasis IoT studi kasus pada rumah tempat tinggal pribadi diangkat berdasarkan permasalahan keamanan rumah yang belum terkontrol sehingga menggakibatkan kemalingan. Ditulis dengan metode *Design Science Research* dengan cara pengumpulan datanya dengan melakukan wawancara.
b)	Implementasi Sistem Kendali dan Monitoring Keamanan Pintu Berbasis IoT Menggunakan Perangkat Mobile[6].	Implementasi Sistem Kendali dan Monitoring Keamanan Pintu Berbasis IoT Menggunakan Perangkat Mobile mengangkat permasalahan penggunakan kunci konvensional yang kurang efektif serta kelalian yang memungkinkan terjadinya pencurian dan penggunaan ruangan tanpa izin yang disusun dengan metode studi literatur, perancangan sistem, implementasi, dan pengujian.
c)	Pengembangan Prototipe	Pengembangan prototype aplikasi

	Aplikasi Membuka Kunci Pintu Otomatis Menggunakan Metode Sidik Jari Berbasis Nodemcu[7].	membuka kunci pintu otomatis menggunakan metode sidik jari berbasis nodemcu yang diangkat karena proses pembukaan pintu masih terkesan jadul di era perkembangan IoT sekarang ini. Disusun dengan metode kepustakaan dan mengumpulkan data dengan mencari referensi dari buku maupun jurnal.
d)	Perancangan Sistem Pembuka Pintu Menggunakan Sidik Jari[8].	Perancangan sistem pembuka pintu menggunakan sidik jari dibuat karena permasalahan tindak criminal perampokan dan juga tingkat keamanan kunci yang mudah dibobol serta melakukan riset untuk pengumpulan datanya.
e)	Rancang Bangun Pengaman Pintu Menggunakan Sidik Jari (*Fingerprint*) Dan Smartphone Android Berbasis Mikrokontroler Atmega8[9].	Hasil dari penelitian yaitu, Sensor sidik jari yang digunakan dapat mengidentifikasi sidik jari dengan posisi yang berbeda, serta mampu membaca sidik jari dalam keadaan kotor. Jarak kontrol antara smartphone android dengan perangkat sistem maksimum sebesar 24 meter.
f)	Sistem Keamanan Pintu Menggunakan Sensor Sidik Jari Berbasis Mikrokontroller Arduino Uno R3[3].	Berdasarkan hasil penelitian maka dapat disimpulkan hasil sudah bisa digunakan tetapi masih banyak kesalahan, tetapi alat yang dibuat berjalan dengan semestinya dan untuk kesalahan bisa diharapkan untuk dikembangkan.
g)	Sistem Pengunci Pintu	Berdasarakan hasil penelitian maka

	Dengan Sidik Jari Menggunakan Arduino[10].	dapat disimpulkan sistem berfungsi dengan baik dan dapat digunakan untuk membuka pintu dengan menggunakan sensor sidik jari,
h)	Sistem Buka Tutup Pintu Otomatis Berbasis Android Dan Sidik Jari[11].	Berdasarkan hasil penelitian maka dapat disimpulkan hasil rancangan ini berhasil menerima inputan dari android sebagai *remote control* dan mengontrol pintu.
i)	Rancang Bagun Sistem Keamanan Pintu Ruangan Tata Usaha Fakultas Sains Dan Teknologi Universitas Islam As-Syafi'iyah Menggunakan Sidik Jari Dan Android[12].	Berdasarkan hasil penelitian maka dapat disimpulkan sistem kemanan pintu fakultas sains menggunakan *fingerprint* berhasil dibuat dan dapat meningkatakan keamanan dari fakultas sains.
j)	Sistem Keamanan Ganda Menggunakan *Fingerprint* dan *Keypad* Pada Pintu Rumah (*Smart Security System*)[13].	Berdasarkan hasil dari penelitian maka dapat disimpulkan Penelitian ini telah berhasil membangun sebuah aplikasi sistem keamanan ganda menggunakan *fingerprint* dan *keypad* pada pintu rumah yang secara keseluruhan sudah berfungsi dengan baik.

Berdasakan hasil kajian pustaka yang telah kami kaji, yakni mencakup berbagai penelitian terkait sistem keamanan pintu berbasis IoT dengan penggunaan sidik jari. Beberapa fitur unggulan dari penelitian-penelitian tersebut adalah penggunaan sidik jari sebagai metode akses, eksplorasi teknologi IoT, serta upaya meningkatkan keamanan dan kenyamanan akses pintu. Namun, ada beberapa kekurangan yang dapat diidentifikasi dari penelitian-penelitian tersebut.

Kelemahan yang dapat diidentifikasi dari tinjauan pustaka tersebut meliputi:

a) Beberapa penelitian mungkin masih menggunakan teknologi yang terkesan jadul atau belum sepenuhnya dioptimalkan untuk era IoT. Hal ini dapat mempengaruhi kualitas dan fungsionalitas sistem keamanan pintu.
b) Beberapa penelitian tidak memasukkan komponen pemantauan monitoring untuk meningkatkan keamanan. Ini dapat mengurangi responsivitas sistem terhadap aktivitas yang mencurigakan.
c) Beberapa penelitian mungkin kurang dalam hal kustomisasi atau manajemen data pengguna. Ini dapat membatasi fleksibilitas pengguna dalam mengelola dan mengonfigurasi sistem.
d) Beberapa penelitian memiliki fokus pada lingkungan tertentu, seperti rumah pribadi atau fasilitas khusus. Ini mungkin tidak mempertimbangkan aplikasi yang lebih luas atau relevan untuk lingkungan lain.

Sementara itu, penelitian kami memiliki beberapa keunggulan yang signifikan diantaranya adalah Fitur Sidik Jari dan IoT, Pemantauan dan Notifikasi, Manajemen Akun *Multilevel* serta Relevansi untuk MIN 2 Bangka. Sebagai hasilnya, penelitian kami menawarkan solusi yang lebih maju, aman, dan sesuai dengan lingkungan MIN 2 Bangka, memungkinkan tingkat keamanan dan kenyamanan yang lebih tinggi dalam mengakses pintu.

2.2. *Internet of Things* (IoT)

Internet of Things (IoT) adalah penerapan teknologi elektronika modern yang memungkinkan kontrol, komunikasi, dan kerjasama perangkat keras melalui internet[14]. Teknologi IoT ini merupakan bagian dari Revolusi Industri 4.0 yang bisa mengendalikan dan memantau perangkat konvensional melalui internet yang mana tentunya mempermudah pekerjaan manusia[15]. Kemudahan dan keefektivitas pengoperasian tersebut menjadi penentu kemajuan sistem, termasuk keamanan gedung Sekolahan. Sistem keamanan penguncian pintu penting dalam pengelolaan gedung Sekolahan[6]. IoT bisa digunakan untuk mengendalikan peralatan elektronik, termasuk kunci pintu.

Sistem keamanan pintu bia+sanya menggunakan kunci manual, namun keamanannya tidak terjamin[5].

Penelitian sebelumnya dilakukan oleh A. Hanafie dkk. yang membahas tentang kontrol akses pintu yang berjudul "Alat Kontrol Akses Pintu Menggunakan Sidik Jari dengan Notifikasi Telegram Berbasis IOT". Dalam penelitiannya menghasilkan sistem akses pintu dengan menggunakan sidik jari serta notifikasi yang akan dikirim melalui Telegram ke pemilik rumah bahwa telah terdeteksi sidik jari pada *fingerprint*[16]. Penelitian lain terkait IoT yang berjudul "Uji Perbandingan Kegiatan Laboratorium Berbasis IoT dengan *Virtual Laboratory* pada Materi Periode Bandul dengan HOTLAB" membahas tentang kegiatan laboratorium berbasis IoT pada materi periode bandul dengan HOTLAB. Metode yang digunakan dalam penelitian ini adalah dengan menguji perbandingan dari hasil analisis data percobaan dengan *Data Analytics* dan *Graphic Analytics*[17]. Lalu ada juga penelitian lain lagi tentang penentuan ciri sidik jari yang dapat membaca gender dengan judul "Penentuan Fitur Gender Pemilik Sidik Jari Berbasis Ketebalan *Ridge*". Penelitian ini membahas penentuan ciri sidik jari yang dapat digunakan sebagai pembeda gender pemiliknya. Pendekatan yang digunakan pada penelitian ini adalah menghitung ketebalan alur (*ridge*) dari sidik jari[18].

Secara keseluruhan, penelitian-penelitian sebelumnya menunjukkan bahwa IoT dan *smartlock* memiliki potensi besar dalam meningkatkan keamanan, efisiensi, dan kemudahan akses pada berbagai bidang, termasuk keamanan rumah tangga, pendidikan, dan identifikasi individu. Penggunaan teknologi IoT dan *smartlock* dapat memberikan solusi yang inovatif dan lebih cerdas dalam mengatasi berbagai tantangan yang dihadapi oleh masyarakat modern.

2.3. Teknologi Sidik Jari

Teknologi sidik jari adalah teknologi cara mengenali orang dengan sidik jari. Dipakai di berbagai hal, seperti identifikasi kriminal, absensi kerja, dan pemilihan umum. Dalam aplikasi identifikasi sidik jari yang mengacu pada teori *Galton-Henry*, teknologi sidik jari digunakan untuk menyatakan dua sidik jari identik jika mereka memiliki pola yang sama dan setidaknya 12 titik minusi yang serupa. Dalam implementasinya, *AFIS* (*Automated Fingerprint Identification System*) digunakan dengan pendekatan yang mengadopsi teori-teori pengolahan citra yang sesuai dengan jenis citra masukan dan tujuan dari teori *Galton-Henry*. Beberapa teori pengolahan citra yang digunakan meliputi *Interpolasi Bilinear*, *Histogram Equalization*, *Transformasi Fourier*, dan *Gabor Filter*[19].

Pada peneiltian sebelumnya yang berjudul "Sistem Kontrol Kunci Pintu Rumah Menggunakan *Fingerprint Smartphone* Android Berbasis Arduino Uno" membahas tentang pengembangan sistem kontrol kunci pintu menggunakan sensor sidik jari pada *smartphone* Android, Arduino Uno, *Bluetooth HC*-05, dan *Selenoid Doorlock*. Penelitian ini bertujuan dari untuk membuat sebuah alat yang dapat mengontrol kunci pintu dari dua arah[20]. Lalu ada juga penelitian yang membahas tentang Pengamanan Mesin ATM berbasis sidik jari dengan judul "Sistem Pengamanan Mesin ATM dengan Menggunakan Pengenalan Sidik Jari dan Wajah *Face Recognition* untuk Meminimalisir *Cyberbanking Crime*". Penelitian ini membahas tentang penggunaan keamanan *biometrik*, khususnya sidik jari dan pengenalan wajah, untuk mengamankan mesin ATM. Penelitian ini bertujuan untuk meminimalisir kejahatan *cyberbanking*, seperti pencurian data dan penggandaan kartu, dengan menggunakan keamanan *biometrik*. Metode yang digunakan dalam penelitian ini adalah merekam sidik jari dan wajah nasabah di bank[21].

Berdasarkan kedua penelitian diatas, teknologi sidik jari smartlock menggunakan teori pemrosesan gambar, pendekatan pembelajaran mendalam, dan keamanan *biometrik* untuk menyediakan kontrol akses yang aman. Teknologi tersebut memiliki keunggulan seperti akurasi dan keamanan

yang tinggi, tetapi juga memiliki beberapa keterbatasan seperti biaya komputasi yang tinggi dan kebutuhan pemeliharaan yang tepat.

2.4. Aplikasi Mobile Berbasis Android

Aplikasi Mobile Berbasis Android merujuk pada aplikasi yang secara spesifik dirancang untuk dijalankan pada sistem operasi Android. Aplikasi ini memiliki beragam kegunaan yang mencakup berbagai keperluan, seperti pengendalian dan pengawasan peternakan ayam broiler[18], pembelajaran pemrograman pada mahasiswa Teknik Informatika[17], pendeteksi kebakaran Gedung[18], pengembangan bahan ajar teori ekonomi mikro[17], pembelajaran matematika pada materi persamaan linear satu variable[20], dan aplikasi mobile driver online[21].

Aplikasi Mobile Berbasis Android dapat membantu meningkatkan keterampilan dan efektivitas pembelajaran, seperti pada pengembangan aplikasi Mobile Learning berbasis Android untuk meningkatkan keterampilan pemrograman pada mahasiswa Teknik Informatika[17], dan pengembangan media pembelajaran Mobile *Learning* berbasis Android yang layak digunakan untuk menunjang pembelajaran matematika pada materi persamaan linear satu variable[22]. Dalam pengembangan aplikasi Mobile Berbasis Android, digunakan berbagai metode dan teknik, seperti Model Siklus Hidup Perangkat Lunak[17], *ADDIE* (*analysis*, *design*, *development*, *implementation*, dan *evaluation*)[22], dan teknik prototype[23].

Terdapat penelitian yang berjudul "Pelatihan Pembuatan Aplikasi Mobile Berbasis Android Pada SMK TKJ Yapim I Medan". Penelitian ini memberikan pembahasan untuk memberikan pelatihan pembuatan aplikasi mobile berbasis Android pada siswa SMK TKJ Yapim I Medan. Materi-materi pelatihan sudah disesuaikan dengan kebutuhan dunia kerja saat ini[24]. Lalu ada jugxa penelitian yang berjudul "Pengembangan Bahan Ajar (Buku) Teori Ekonomi Mikro Berbasis Media Pembelajaran Aplikasi Android", yang membahas tentang meningkatkan kualitas pembelajaran pada mata kuliah teori ekonomi mikro di program studi Pendidikan Ekonomi,

Fakultas Ekonomi, Unimed. Penelitian ini mengembangkan media pembelajaran teori ekonomi mikro berbasis Android yang efektif dan layak untuk digunakan dalam pembelajaran[17]. Selain itu terdapat juga penelitian dengan judul "*Security Assessment* Pada Aplikasi Mobile Android XYZ Dengan Mengacu Pada Kerentanan *OWASP Mobile Top Ten* 2016". Penelitian ini melakukan *security assessment* pada aplikasi XYZ untuk mengidentifikasi kerentanan dan memberikan rekomendasi keamanan pada kerentanan yang teridentifikasi. Kerentanan yang diidentifikasi mengacu pada kerentanan dari *OWASP Mobile Top Ten* 2016[17]. Penelitian lain terkait dengan aplikasi mobile dengan judul "Efektifitas Penerapan *Flipped Learning Model* Berbasis Aplikasi Android Terhadap Hasil Belajar *Laboratory Skills* pada Mahasiswa Keperawatan di Kota Jambi". Penelitian ini bertujuan untuk mengidentifikasi perbedaan hasil belajar *laboratory skills* pada kelompok intervensi dan kontrol setelah dilakukan penerapan *flipped learning* model berbasis aplikasi Android pada mahasiswa/i keperawatan di Kota Jambi[22].

Dari penelitian-penelitian diatas, aplikasi mobile berbasis Android memiliki banyak potensi untuk digunakan dalam berbagai bidang, seperti pendidikan, keamanan, dan pelayanan publik. Aplikasi mobile berbasis Android juga dapat meningkatkan kualitas pembelajaran dan hasil belajar. Namun, perlu diperhatikan juga keamanan dari aplikasi tersebut agar tidak terjadi kerentanan yang dapat membahayakan pengguna.

Secara keseluruhan, dapat disimpulkan bahwa penggunaan sidik jari sebagai identifikasi pada pengunci pintu memiliki kelebihan karena sidik jari setiap orang unik, sehingga sulit untuk dipalsukan. Namun, kelemahan dari penggunaan sidik jari sebagai identifikasi adalah jika terjadi kerusakan pada sensor sidik jari, maka pengunci pintu tidak dapat digunakan.

Sementara itu, penelitian yang kami memiliki fitur yang menarik, yaitu bisa IOT, bisa membuka menggunakan sidik jari, bisa monitoring jika pintu terbuka, dan ada level user yang mana admin bisa mendaftarkan,

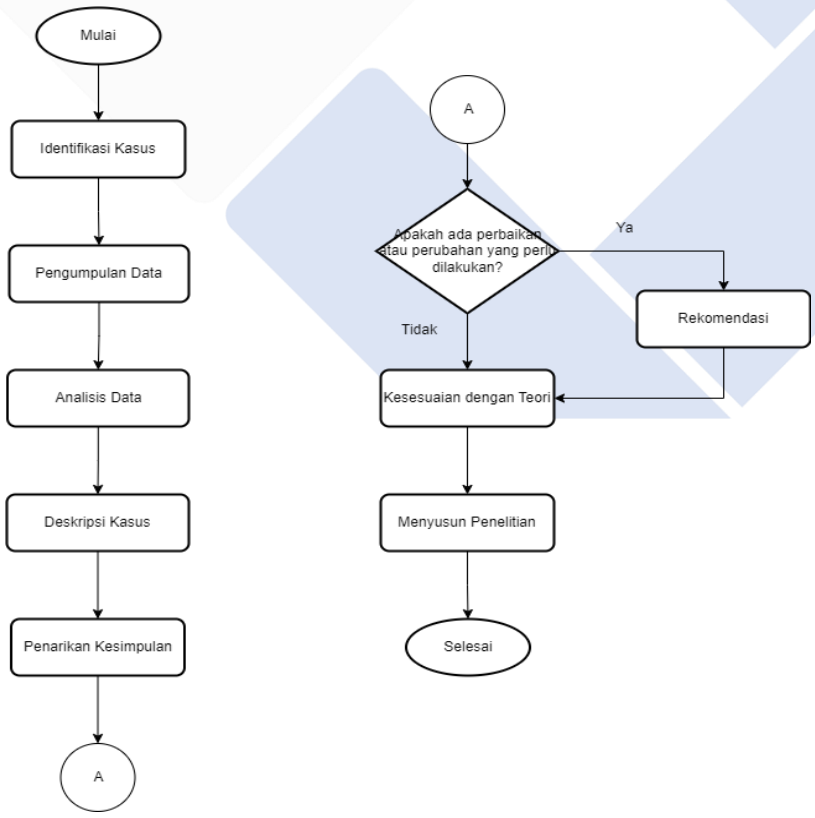
menghapus akun pada aplikasi yang sama. Hal ini membuat penelitian kami memiliki kelebihan dibandingkan dengan penelitian yang telah ada.

BAB III
METODE PELAKSANAAN

3.1. Model Rancangan

3.1.1. Metode Pelaksanaan Studi Kasus

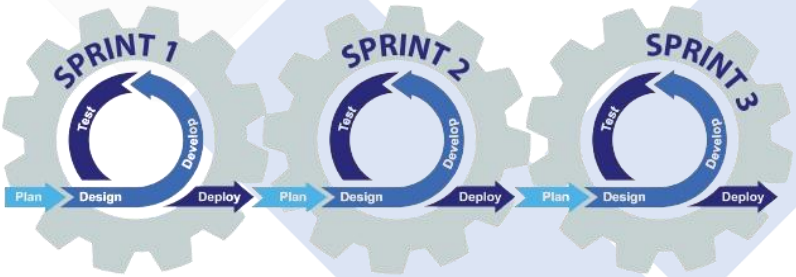
Metode pelaksanaan yang kami terapkan dalam pembuatan penelititian ini menggunakan metode studi kasus[25], Metode penelitian studi kasus adalah suatu pendekatan penelitian yang fokus pada pemahaman yang mendalam tentang suatu fenomena atau permasalahan tertentu di dalam konteks nyata. Penelitian studi kasus dilakukan dengan cara mendokumentasikan, menganalisis, dan menjelaskan kasus atau situasi yang terjadi dalam lingkungan yang sesungguhnya[26]. Untuk tahapan-tahapan umum dalam proses pengerjaan dapat diilustrasikan melalui *flowchart* berikut.



Gambar 3 1 *Flowchart* Studi Kasus

3.1.2. Metode Pengembangan *Agile*

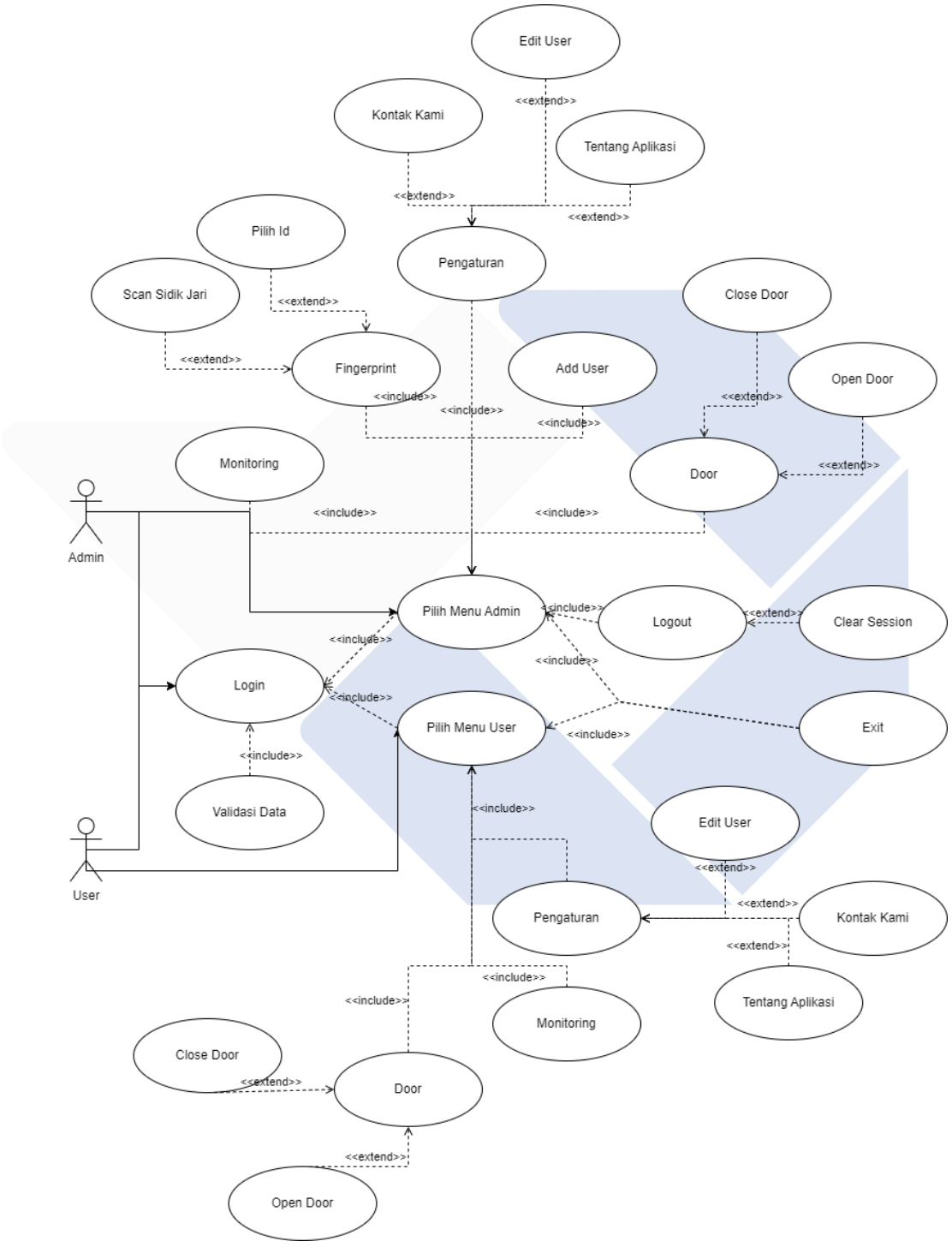
Metode pelaksanaan yang kami terapkan dalam pembuatan aplikasi pada penelitian ini didasarkan pada prinsip-prinsip metode *agile*[25], yang secara khusus menekankan sifat iteratif dan inkremental dalam pengembangan perangkat lunak. Dengan fokus pada kolaborasi intensif antara tim pengembang, pendekatan ini memungkinkan *respons* yang cepat terhadap perubahan dan menciptakan perangkat lunak yang adaptif terhadap perubahan kebutuhan. *Responsibilitas* terhadap dinamika perubahan menjadi integral dalam metode *agile*, memungkinkan penyesuaian yang fleksibel terhadap prioritas, tugas, dan fokus pengembangan. Proses pengerjaan yang disajikan melalui gambar di bawah ini memberikan gambaran yang jelas dan sederhana mengenai bagaimana perangkat lunak berkembang dari satu tahap ke tahap berikutnya.



Gambar 3 2. *Sprint Agile*

3.1.3. *Use Case Diagram*

Pemodelan diagram use *case* untuk admin dan user pada aplikasi *Smartlock* untuk mengidentifikasi interaksi antara pengguna dan sistem.



Gambar 3.1.3.1 *Use Case Diagram*

3.2. Analisis Kebutuhan

Terdapat beberapa *hardware* dan *software* yang digunakan untuk membuat aplikasi *smartlock for door* dengan sidik jari berbasis android pada min 2 bangka, berikut uraian hardware dan softwarenya.

3.2.1. *Hardware*

Aplikasi *smartlock* dengan sidik jari memerlukan beberapa komponen *hardware*, antara lain:

a) Sensor sidik jari

Komponen ini digunakan untuk membaca sidik jari pengguna dan membandingkannya dengan data sidik jari yang telah tersimpan.

b) Mikrokontroler

Komponen ini berfungsi sebagai pusat kendali aplikasi *smartlock*. Mikrokontroler akan memproses data dari sensor sidik jari dan mengirimkan sinyal untuk membuka atau mengunci pintu.

c) *Selenoid Door Lock*

Komponen ini digunakan untuk mengunci dan membuka pintu secara otomatis.

d) Kable Jumper

Komponen ini digunakan untuk menghubungkan komponen satu kekomponen lainnya

e) Baterai

Komponen ini digunakan untuk sumber daya dari perangkat *smartlock for door* agar bisa digunakan dan lebih simpel dibandingkan dengan sumber daya yang lain.

f) *Buzzer*

Komponen ini digunakan untuk mendeteksi ESP 8266 sudah terkoneksi atau belum, dan juga komponen ini bisa mendeteksi sidik jari yang salah maka komponen ini akan berbunyi.

g) *Relay*

Komponen ini digunakan untuk membuka atau menutup sirkuit listrik ketika menerima sinyal dari sumber tertentu.

3.2.2. Software

Selain kebutuhan *hardware*, aplikasi *smartlock* dengan sidik jari juga memerlukan beberapa komponen *software*, antara lain:

a) Sistem operasi Android

Aplikasi *smartlock* dengan sidik jari dirancang untuk berjalan pada sistem operasi Android.

b) Bahasa pemrograman Kotlin

Bahasa pemrograman Kotlin digunakan untuk membuat dan mengembangkan aplikasi smartlock dengan sidik jari.

c) Database

Aplikasi smartlock dengan sidik jari memerlukan database untuk menyimpan data sidik jari pengguna dan informasi lainnya, seperti akun waktu akses dan notifikasi.

d) Algoritma *Minutiae-Based Matching*

Algoritma ini digunakan untuk membandingkan sidik jari pengguna dengan data sidik jari yang telah tersimpan.

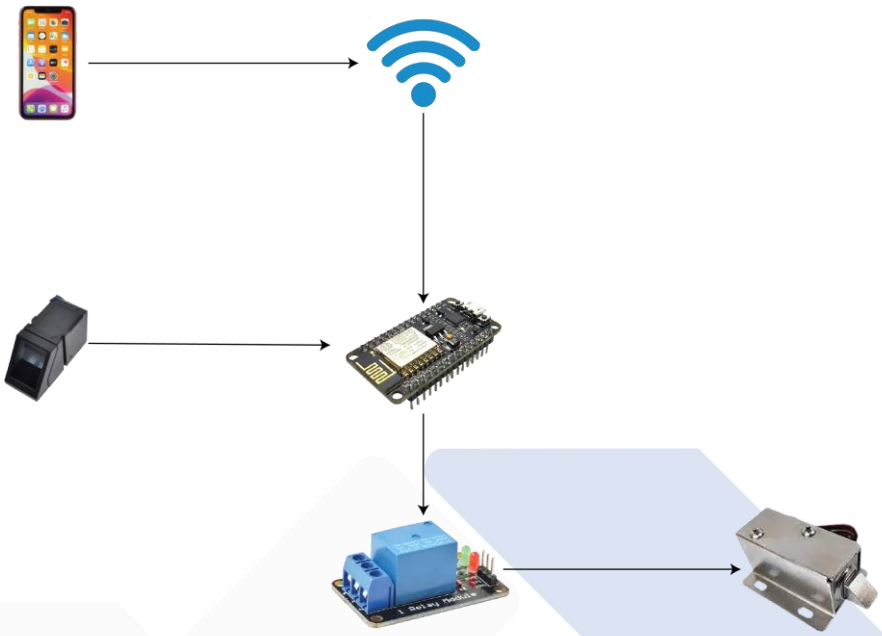
e) *Integrated Development Environment* (IDE)

IDE yang digunakan untuk membuat aplikasi ini adalah Android Studio, karena Android Studio sendiri menyediakan lingkungan pengembangan yang memfasilitasi pembuatan aplikasi *smartlock* dengan sidik jari menggunakan bahasa pemrograman Kotlin.

f) *Android Software Development Kit* (SDK)

SDK Android menyediakan alat dan API yang diperlukan untuk mengembangkan aplikasi Android, termasuk aplikasi *smartlock*. Untuk aplikasi kami menggunakan API versi 34.

3.3. Perancangan Sistem

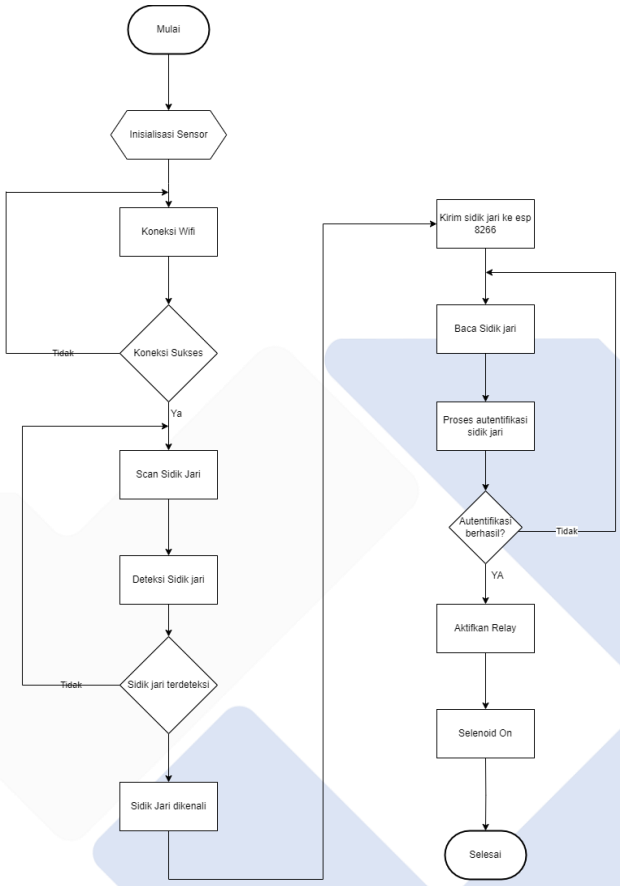


Gambar 3 3 Skema Sistem

Pada tahap ini berisi tentang rancangan sistem *Smartlock for door* menggunakan sensor sidik jari dan IOT. Diawali dengan Aplikasi *smartlock* dimana user akan melakukan login terlebih dahulu, setelah itu user bisa menggunakan menu *Door* dimana menu *Door* tersebut adalah untuk membuka pintu otomatis berbasis IOT. Jika user menekan *button open door* maka akan mengirim data melalui wifi lalu mengirim data ke mikorkontroller kemudian jika sesuai maka mikrokontroller memberikan perintah ke relay untuk membuka *solenoid door*. Kemudian untuk sensor sidik jari tersebut, jika sensor membaca data sidik jari maka sensor akan mulai mengirim data ke mikrokontroller lalu jika data sesuai mikrokontroller akan memberikan perintah ke relay untuk membuka *solenoid door*.

3.4. Rancangan *Hardware*

Perancangan *Hardware* dan alur kerja dari sistem *smartlock* menggunakan sidik jari berbasis iot.



Gambar 3 4 Skema Hardware

3.5. Pengumpulan dan Pengujian Data

Jenis metode pengujian data yang kami gunakan adalah metode gabungan. Untuk pengujian alat kami menggunakan metode kuantitatif berupa uji coba keberhasilan sidik jari dan untuk metode aplikasi kami menggunakan metode kualitatif berupa survey kuesioner. Berikut merupakan metode dari pengumpulan dan pengujian data yang diperlukan untuk Aplikasi *Smartlock for door* pada Min 2 Bangka:

3.5.1. Fungsionalitas

Pengujian fungsionalitas yang kami lakukan adalah suatu proses evaluasi yang bertujuan untuk memastikan bahwa alat yang diteliti beroperasi sesuai dengan fungsinya dan memenuhi standar yang ditetapkan.

3.5.2. Kuesioner

Pengujian kuesioner yang kami lakukan melibatkan evaluasi terhadap pertanyaan-pertanyaan yang diajukan kepada responden dan analisis terhadap jawaban yang diberikan. Dalam metode penelitian ini, terlihat bahwa kuesioner digunakan untuk mengukur kepuasan pengguna terhadap aplikasi *smartlock*

BAB IV
HASIL DAN PEMBAHASAN

4.1. Pembuatan Aplikasi

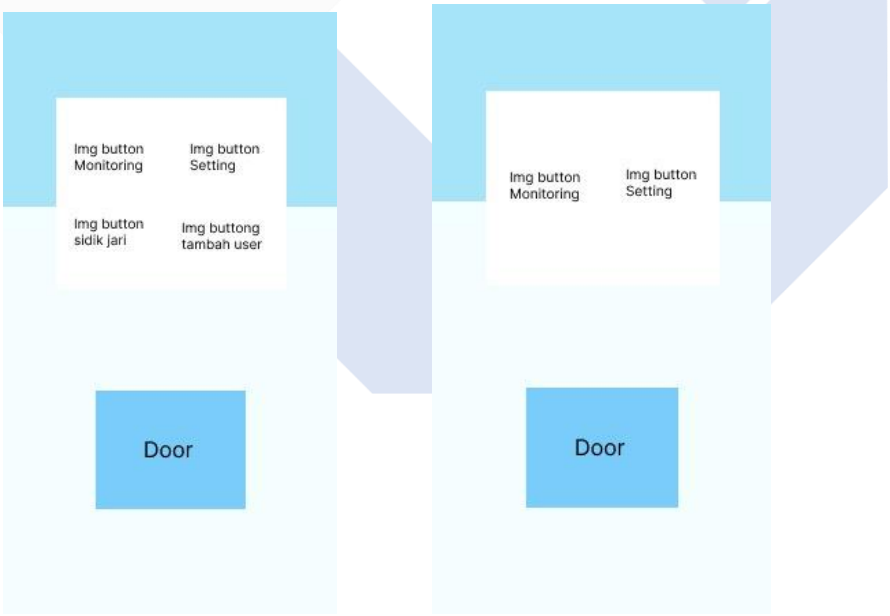
Pada kali ini peneliti akan melakukan pembuatan fitur pada aplikasi *smartlock*, berikut pengembangan fitur menggunakan *sprint agile* :

4.1.1. *SPRINT* 1 : Membuat Menu Admin dan *User*

a) Plan

Pada tahap ini yaitu tahap perancangan, kami fokus perencanaan proyek untuk menu admin dan menu user. Menu admin mencakupi tampilan yang berisikan 4 buah *image button* dan satu buah *button*, dimana 4 buah *image button* itu adalah *monitoring*, *setting*, daftar sidik jari serta tambah *user*, sedangkan untuk satu buah button yaitu button door untuk menjalankan IOT membuka tutup pintu dari aplikasi.

b) Design



Gambar 4 1 Desain menu Admin dan *User*

Desain antarmuka dan penempatan *resource* untuk admin berupa *image monitoring, image setting, image add fingerprint dan image add user*

pada satu bagian lalu di bawahnya kami buat *button* yang bertuliskan Door, sedangkan untuk antarmuka dan penempatan *resource* user mirip dengan admin, pembedanya ada pada bagian *resource* yang hanya ada image monitoring dan setting lalu di bawahnya ada button yang bertuliskan *door*.

c) Develop

Di tahap develop atau pengembangan ini kami mulai membuat aplikasi berdasarkan rancangan yang telah dibuat sebelumnya. Kami menuliskan kode xml untuk antarmuka layout login, lalu menggunakan Bahasa pemrograman kotlin untuk menjalankan logika *backend* program seperti mengarahkan ke *layout* mana jika button di klik.

```
class MainAdminFragment : Fragment() {

    private lateinit var doorButton: Button
    private lateinit var fingerButton: ImageButton
    private lateinit var adduserButton: ImageButton
    private lateinit var settingButton: ImageButton

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_main_admin, container, false)

        doorButton      = view.findViewById(R.id.B_Door)
        fingerButton    = view.findViewById(R.id.I_Finger)
        adduserButton   = view.findViewById(R.id.I_addUser)
        settingButton   = view.findViewById(R.id.I_Setting)

        doorButton.setOnClickListener{
            val destinationFragment = DoorFragment()
            val transaction = requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        fingerButton.setOnClickListener{
            val destinationFragment = FingerprintFragment()
            val transaction = requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        adduserButton.setOnClickListener{
            val destinationFragment = AddUserFragment()
            val transaction = requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        settingButton.setOnClickListener{
            val destinationFragment = SettingFragment()
            val transaction = requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        return view
    }
}
```

Gambar 4 2 Program menu admin

```
class HomeFragment : Fragment() {

    private lateinit var doorButton: Button
    private lateinit var settingButton: ImageButton

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_home, container, false)

        doorButton = view.findViewById(R.id.B_Door)
        settingButton   = view.findViewById(R.id.I_Setting)

        doorButton.setOnClickListener{
            val destinationFragment = DoorFragment()
            val transaction = requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        settingButton.setOnClickListener{
            val destinationFragment = SettingFragment()
            val transaction = requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        return view
    }

}
```

Gambar 4 3 Program menu *user*

d) Testing

Pada sesi pengujian kami memastikan bahwa sistem dari aplikasi berfungsi dengan baik dan lancer. Kami menguji apabila terdapat error atau *force close* saat menjalankan menu admin ataupun user.

e) Deploy

Sesi terakhir dari pembuatkan menu admin dan *user* ini kami *build* dan luncurkan untuk menambahkan fitur ini ke aplikasi kami. Kami akan memastikan lagi bahwa fitur kami siap untuk digunakan.

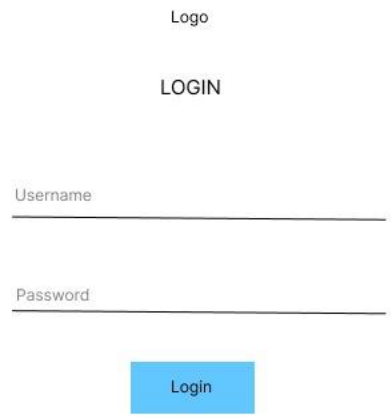
4.1.2. *SPRINT* 2 : *Login Multi Level*

a) Plan

Pada tahap ini yaitu tahap perancangan, kami fokus perencanaan proyek untuk sesi login. Sesi login ini mencakupi fitur *multilevel login*, dimana jika jenis *user* yang login adalah *user* atau level 2 dia akan mengarahkan ke menu atau tampilan *user*, sedangkan jika jenis *user* admin atau level 1 dia akan mengarah ke tampilan admin yang memiliki fitur lebih dibandingkan level 2 atau *user*. Pada tampilan *login* ini terdapat *text input* berupa *username* dan *password* dan satu buah *button*

login untuk menjalankan logika dan memvalidasi *username* serta *password* yang dimasukkan.

b) Design



Gambar 4 4 Desain *login*

Kami mendesain antarmuka dan penempatan *resource* berupa logo *user* di bagian paling atas, lalu di bawahnya kami meletakan teks bertuliskan *LOGIN*. Di bawah teks *LOGIN* terdapat 2 buah Teks *input* untuk *Username* dan *password*. Untuk *username* menggunakan *input* tipe teks sedangkan untuk *password* menggunakan *input* tipe *password*. Lalu di bagian paling bawah ada *button* yang bertuliskan *LOGIN* yang mana *button* ini berfungsi untuk menjalankan logika dan memvalidasi *username* serta *password* yang dimasukkan.

c) Develop

Di tahap *develop* atau pengembangan ini kami mulai membuat aplikasi berdasarkan rancangan yang telah dibuat sebelumnya. Kami menuliskan kode *xml* untuk antarmuka *layout login*, lalu menggunakan Bahasa pemrograman kotlin untuk menjalankan logika *backend* program seperti *button* yang apabila di klik akan mengirimkan logika ke API, lalu untuk API terhubung ke database. kami menggunakan Bahasa pemrograman PHP dan untuk database kami menggunakan hosting mysql.

```
class LoginActivity : AppCompatActivity() {

    lateinit var prefHelper: PrefHelper

    private var binding: ActivityLoginBinding? = null
    private var user: String = ""
    private var pass: String = ""
    private var lvl: Int = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding!!.root)

        prefHelper = PrefHelper(this)

        binding!!.buttonLogin.setOnClickListener {
            user = binding!!.editTextUsername.text.toString()
            pass = binding!!.editTextPassword.text.toString()

            when {
                user == "" -> {
                    binding!!.editTextUsername.error = "Username tidak boleh kosong"
                }

                pass == "" -> {
                    binding!!.editTextPassword.error = "Password tidak boleh kosong"
                }

                else -> {
                    binding!!.loading.visibility = View.VISIBLE
                    getData()
                }
            }
        }
    }

    private fun getData() {
        val api = RetrofitClient().getInstance()
        api.login(user, pass).enqueue(object : retrofit2.Callback<ResponseLogin> {
            override fun onResponse(call: Call<ResponseLogin>, response: Response<ResponseLogin>) {
                if (response.isSuccessful) {
                    if (response.body()?.response == true) {
                        val userLevel = response.body()?.payload?.level_user ?: "0"
                        val userId = response.body()?.payload?.id_user ?: "0"
                        saveSession(user, pass, true, userLevel.toInt(), userId.toInt())
                        binding!!.loading.visibility = View.GONE

                        if (userLevel == "1") {
                            startActivity(Intent(this@LoginActivity, MainActivityAdmin::class.java))
                        } else {
                            startActivity(Intent(this@LoginActivity, MainActivity::class.java))
                        }
                        finish()

                    } else {
                        binding!!.loading.visibility = View.GONE
                        Toast.makeText(
                            this@LoginActivity,
                            "Login gagal, Periksa kembali username dan password",
                            Toast.LENGTH_LONG
                        ).show()
                    }
                } else {
                    Toast.makeText(
                        this@LoginActivity,
                        "Login gagal, Terjadi Kesalahan",
                        Toast.LENGTH_LONG
                    ).show()
                }
            }

            override fun onFailure(call: Call<ResponseLogin>, t: Throwable) {
                Log.e("pesan error", "${t.message}")
            }

        })
    }

    override fun onStart() {
        super.onStart()

        if (prefHelper.getIsLogin()) {
            val userLevel = prefHelper.getUserLevel()
            val intent = if (userLevel == 1) {
                Intent(this@LoginActivity, MainActivityAdmin::class.java)
            } else {
                Intent(this@LoginActivity, MainActivity::class.java)
            }

            startActivity(intent)
            finish()
        }
    }

    private fun saveSession(
        username: String,
        password: String,
        isLogin: Boolean,
        userLevel: Int,
        userId: Int
    ) {
        prefHelper.saveSession(username, password, isLogin, userLevel, userId)
    }
}
```

Gambar 4 5 Program *login*

d) Testing

Dalam sesi pengujian yang kami lakukan, kami dengan memverifikasi semua fitur yang terdapat dalam aplikasi yang kami buat beroperasi sesuai dengan perencanaan yang telah ditetapkan. Pengujian kami melibatkan skenario-skenario tertentu, seperti memasukkan kombinasi *username* dan *password* yang salah, untuk memastikan *respons* sistem yang tepat dalam situasi-situasi tersebut. Selain itu, kami menguji apakah sistem merespons dengan benar ketika *username* dan *password* yang dimasukkan memiliki tingkat akses level 1 atau admin, dengan fokus pada pengalihan menu ke halaman admin. Begitu juga, kami mengevaluasi respons sistem ketika *username* dan *password* memiliki tingkat akses level 2 atau user, dengan penekanan pada pengalihan menu ke halaman user. Semua hasil pengujian ini kami dokumentasikan dengan cermat untuk memastikan kehandalan dan keefektifan keseluruhan dari aplikasi yang kami kembangkan.

e) Deploy

Sesi terakhir dari proses pembuatan fitur *login multilevel* ini adalah ketika kami berhasil membangun dan meluncurkannya ke dalam aplikasi kami, sebagai upaya untuk menambahkan dimensi baru pada fungsionalitas *login*. Untuk memastikan kesiapan fitur ini, kami akan melakukan verifikasi tambahan untuk menjamin bahwa fitur tersebut siap dan dapat digunakan dengan optimal dalam aplikasi kami.

4.1.3. *SPRINT* 3 : *IOT Doorlock*

a) Plan

Pada tahap perancangan ini, kami memusatkan perhatian pada perencanaan proyek untuk menu *IOT Doorlock*. Menu *Doorlock* ini mencakup tampilan yang terdiri dari 2 buah tombol, yang berfungsi untuk mengoperasikan *IOT* dalam membuka dan menutup pintu melalui aplikasi.

b) Design

Gambar 4 6 Desain *IOT*

Dalam proses perancangan, kami berfokus pada desain antarmuka yang optimal dan strategi penempatan resource yang efektif. Sebagai bagian dari implementasi ini, kami telah merancang dua buah tombol yang memiliki fungsi, yaitu "*OPEN DOOR*" dan "*CLOSE DOOR*". Keberadaan tombol-tombol pintu ini dirancang untuk memenuhi kebutuhan akses secara *Internet of Things (IoT)* dengan memberikan kontrol atas pembukaan dan penutupan pintu. Penting untuk dicatat bahwa tombol *door* ini telah ditempatkan secara khusus di dua tampilan utama, yaitu tampilan admin dan tampilan pengguna (user), sehingga memberikan hak akses yang sesuai kepada administrator dan pengguna untuk mengelola pintu dengan efisien dan sesuai dengan otoritas masing-masing.

c) *Develop*

Di tahap *develop* atau pengembangan ini kami mulai membuat aplikasi berdasarkan rancangan yang telah dibuat sebelumnya. Kami menuliskan kode xml untuk antarmuka *layout*, lalu menggunakan Bahasa pemrograman kotlin untuk menjalankan logika *backend* program seperti mengarahkan ke layout mana jika *button* di klik.

```
class DoorFragment : Fragment() {

    lateinit var prefHelper: PrefHelper

    private lateinit var btnOpenDoor: Button
    private lateinit var btnCloseDoor: Button
    private lateinit var controlViewModel: ControlViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view = inflater.inflate(R.layout.fragment_door, container, false)

        prefHelper = PrefHelper(requireActivity())

        controlViewModel = ViewModelProvider(requireActivity())[ControlViewModel::class.java]

        btnOpenDoor = view.findViewById(R.id.btnOpen)
        btnCloseDoor = view.findViewById(R.id.btnClose)

        btnOpenDoor.setOnClickListener {
            controlViewModel.controlSolenoid("close")
        }

        btnCloseDoor.setOnClickListener {
            controlViewModel.controlSolenoid("open")
        }

        controlViewModel.response.observe(viewLifecycleOwner) { response ->
            requireActivity().runOnUiThread {
                showToast(response)
            }
        }

        return view
    }

    private fun showToast(message: String) {
        Toast.makeText(requireContext(), message, Toast.LENGTH_SHORT).show()
    }

    class ControlViewModel : ViewModel() {

        private val _response = MutableLiveData<String>()
        val response: LiveData<String> get() = _response

        fun controlSolenoid(state: String) {
            GlobalScope.launch(Dispatchers.IO) {
                val result = sendControlRequest(state)
                withContext(Dispatchers.Main) {
                    _response.value = result
                }
            }
        }

        private fun sendControlRequest(state: String): String {
            val url = URL("http://192.168.125.99./control")
            val urlConnection = url.openConnection() as HttpURLConnection

            return try {
                urlConnection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded")
                urlConnection.requestMethod = "POST"
                urlConnection.doOutput = true

                val outputStream = DataOutputStream(urlConnection.outputStream)
                val postData = "state=$state"
                outputStream.writeBytes(postData)
                outputStream.flush()
                outputStream.close()

                val responseCode = urlConnection.responseCode
                println("Response Code: $responseCode")

                if (urlConnection.responseCode == HttpURLConnection.HTTP_OK) {
                    urlConnection.inputStream.bufferedReader().readText()
                } else {
                    "HTTP Error: ${urlConnection.responseCode}"
                }
            } catch (e: Exception) {
                e.printStackTrace()
                "Error: ${e.message}"
            } finally {
                urlConnection.disconnect()
            }
        }
    }
}
```

Gambar 4 7 Program *IOT*

d) *Testing*

Pada sesi pengujian kami memastikan bahwa fitur dari aplikasi yang kami buat berfungsi sesuai dengan apa yang direncanakan, jika menekan tombol maka solenoid atau slot pintu akan terbuka, jika tidak mungkin ada kesalahan pada bagian pengkodean.

e) Deploy

Sesi terakhir dari pembuatkan fitur login multilevel ini kami *build* dan luncurkan untuk menambahkan fitur ini ke aplikasi kami. Kami akan memastikan lagi bahwa fitur kami siap untuk digunakan.

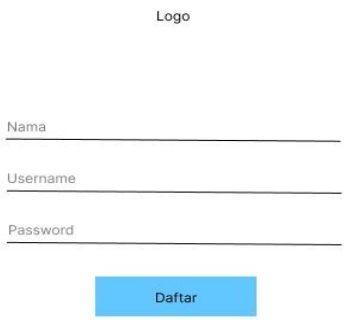
4.1.4. *SPRINT* 4 : Tambah *User*

a) Plan

Pada tahap ini yaitu tahap perancangan, kami fokus perencanaan proyek untuk sesi tambah user. Sesi tambah user ini mencakupi fitur *multilevel login*, dimana jika jenis *user* yang login adalah *user* atau level 2 dia akan mengarahkan ke menu atau tampilan *user*, sedangkan jika jenis *user* admin atau level 1 dia akan mengarah ke tampilan admin yang memiliki fitur lebih dibandingkan level 2 atau *user*. Pada tampilan *login* ini terdapat *text* input berupa nama, *username* dan *password* dan satu buah *button* daftar untuk menjalankan logika dan memvalidasi *username* serta *password* yang dimasukkan.

b) Design

.



Gambar 4 8 Desain *Add User*

Kami mendesain antarmuka dan penempatan resource berupa textview yang harus diisi nama, *username* dan *password* dan juga *button* yang bernama Daftar. Dimana button ini akan menginput data yang telah diisi ke *database*.

c) Develop

Di tahap *develop* atau pengembangan ini kami mulai membuat aplikasi berdasarkan rancangan yang telah dibuat sebelumnya. Kami menuliskan kode xml untuk antarmuka *layout*, lalu menggunakan Bahasa pemrograman kotlin untuk menjalankan logika *backend* program seperti mengarahkan ke login jika *button* registrasi diklik.

```
class AddUserFragment : Fragment() {

    private val BASE_URL = "https://scrptdoorlock.000webhostapp.com/login/"

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_add_user, container, false)

        val etName           : EditText = view.findViewById(R.id.etName)
        val etUsername       : EditText = view.findViewById(R.id.etUsername)
        val etPassword       : EditText = view.findViewById(R.id.etPassword)
        val etConfirmPassword: EditText = view.findViewById(R.id.etConfirmPassword)
        val btnSignUp        : Button   = view.findViewById(R.id.btnRegister)

        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create(GsonBuilder().setLenient().create()))
            .build()

        val service = retrofit.create(ApiService::class.java)

        btnSignUp.setOnClickListener {
            val name = etName.text.toString()
            val username = etUsername.text.toString()
            val password = etPassword.text.toString()
            val confirmPassword = etConfirmPassword.text.toString()

            if (username.isEmpty()) {
                etUsername.setError("Username harus diisi")
                etUsername.requestFocus()
                return@setOnClickListener
            }

            if (password.isEmpty()) {
                etPassword.setError("Password harus diisi")
                etPassword.requestFocus()
                return@setOnClickListener
            }

            if (password.length < 8) {
                etPassword.setError("Password harus memiliki minimal 8 karakter")
                etPassword.requestFocus()
                return@setOnClickListener
            }

            if (!password.matches(".*\\d.*".toRegex())) {
                etPassword.setError("Password harus memiliki minimal satu angka")
                etPassword.requestFocus()
                return@setOnClickListener
            }

            if (confirmPassword.isEmpty()) {
                etConfirmPassword.setError("Konfirmasi Password harus diisi")
                etConfirmPassword.requestFocus()
                return@setOnClickListener
            }

            if (password != confirmPassword) {
                etConfirmPassword.setError("Konfirmasi Password tidak sesuai dengan Password")
                etConfirmPassword.requestFocus()
                return@setOnClickListener
            }

            val call = service.signup(name, username, password)

            call.enqueue(object : Callback<ApiResponse> {
                override fun onResponse(call: Call<ApiResponse>, response: Response<ApiResponse>) {
                    if (response.isSuccessful) {
                        val apiResponse = response.body()
                        if (apiResponse != null && apiResponse.status == "success") {
                            Log.d("SignUp", "Success")
                            showToast("SignUp Successful")
                        } else {
                            Log.d("SignUp", "Failed: ${apiResponse?.message}")
                            showToast("Sign Up gagal: ${apiResponse?.message}")
                        }
                    } else {
                        Log.d("SignUp", "Failed: ${response.message()}")
                        showToast("Sign Up gagal: ${response.message()}")
                    }
                }

                override fun onFailure(call: Call<ApiResponse>, t: Throwable) {
                    Log.d("SignUp", "Error: ${t.message}")
                    showToast("Terjadi kesalahan: ${t.message}")
                }
            })
        }
        return view
    }

    private fun showToast(message: String) {
        Toast.makeText(requireContext(), message, Toast.LENGTH_SHORT).show()
    }
}
```

Gambar 4 9 Program *Add User*

d) Testing

Pada sesi pengujian kami memastikan bahwa fitur dari aplikasi yang kami buat berfungsi sesuai dengan apa yang direncanakan. Kami menguji apabila memasukkan nama, *username* dan *password* yang salah bagaimana hasilnya, jika memasukkan nama, *username* dan *password* yang mempunyai level 1 atau admin apakah akan mengarahkan ke menu admin, lalu apabila memasukkan nama, *username* dan *password* yang mempunyai level 2 atau *user* apakah akan mengarahkan ke menu *user*.

e) Deploy

Sesi terakhir dari proses pembuatan fitur *login multilevel* ini telah menjadi langkah signifikan dalam pengembangan aplikasi kami, di mana kami berhasil membangun dan meluncurkan fitur tersebut dengan penuh dedikasi. Tujuan utama kami adalah untuk memperkaya pengalaman pengguna dengan menambahkan opsi *login multilevel* yang inovatif dan aman ke dalam aplikasi kami. Kami akan terus memastikan bahwa fitur ini tidak hanya terintegrasi secara mulus, tetapi juga siap untuk digunakan oleh pengguna dengan performa optimal dan keamanan yang terjamin.

4.1.5. *SPRINT* 5 :Tambah sidik jari

a) Plan

Pada tahap ini yaitu tahap perancangan, kami fokus perencanaan proyek untuk sesi tambah sidik jari. Sesi tambah sidik jari ini mencakupi pilih id dan scan sidik jari dimana pilih id betujuan untuk membedakan data jari jika sudah selesai dimasukan id maka user akan melakukan scan pada sensor sidik jari agar sensor bisa membaca data jari user yang baru mendaftar.

b) Design

Gambar 4 10 Tambah sidik jari

Kami mendesain antarmuka dan penempatan resource berupa *button* yang bernama Pilih id dan *Scan* sidik jari. Dimana untuk tampilan dalam dari kedua button tersebut berbeda-beda.

c) Develop

Pada tahap *develop* atau pengembangan ini, kami memulai proses pembuatan aplikasi dengan merinci rancangan yang telah disusun sebelumnya. Tindakan awal melibatkan penulisan kode *XML* untuk menyusun antarmuka *layout*, dengan langkah selanjutnya menggunakan Bahasa pemrograman Kotlin untuk mengimplementasikan logika *backend* program. Ini mencakup aspek-aspek seperti menentukan perilaku ketika tombol diklik, mengarahkan navigasi ke *layout* yang sesuai, dan menjalankan berbagai fungsi logika yang mendukung fungsionalitas aplikasi secara menyeluruh. Proses ini merupakan bagian dari langkah-langkah pengembangan yang kami tempuh untuk mewujudkan aplikasi sesuai dengan visi dan kebutuhan yang telah diidentifikasi sebelumnya.

```
class AddFingerprintFragment : Fragment() {

    private lateinit var B_Next1: Button
    private lateinit var V_Flipper: ViewFlipper
    private lateinit var gifImageView: ImageView
    private lateinit var viewModel: FingerprintViewModel

    lateinit var prefHelper: PrefHelper

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_add_fingerprint, container, false)

        prefHelper = PrefHelper(requireActivity())

        V_Flipper = view.findViewById(R.id.V_Flipper)
        B_Next1 = view.findViewById(R.id.B_Next1)
        gifImageView = view.findViewById(R.id.gifImageView)

        val repository = FingerprintRepository()
        viewModel = ViewModelProvider(
            this,
            FingerprintViewModelFactory(repository)
        )[FingerprintViewModel::class.java]

            B_Next1.setOnClickListener {
                viewModel.addFingerprint { result ->
                    requireActivity().runOnUiThread {
                        showToast(result)
                    }
                }
                Glide.with(this@AddFingerprintFragment).asGif().load(R.drawable.fingerprint)
                    .into(gifImageView)
                gifImageView.visibility = View.VISIBLE

                Handler(Looper.getMainLooper()).postDelayed({
                    gifImageView.visibility = View.GONE
                    V_Flipper.displayedChild = 2

                    Handler(Looper.getMainLooper()).postDelayed({
                        Glide.with(this@AddFingerprintFragment).asGif().load(R.drawable.fingerprint)
                            .into(gifImageView)
                        gifImageView.visibility = View.VISIBLE
                    }, 1000)

                    Handler(Looper.getMainLooper()).postDelayed(
                        {
                            V_Flipper.displayedChild = 1
                        },
                        3000
                    )
                }, 1000)
            }

            return view
        }

    private fun sendToESP(enteredNumber: Int) {
        val ipAddress = "192.168.125.99"
        val port = 80

        GlobalScope.launch {
            val url = URL("http://$ipAddress:$port/?data=$enteredNumber")
            with(url.openConnection() as HttpURLConnection) {
                requestMethod = "GET"
                inputStream.bufferedReader().use {
                    val response = it.readText()
                    println("Response from ESP8266: $response")
                }
            }
        }
    }

    private fun showToast(message: String) {
        Toast.makeText(requireContext(), message, Toast.LENGTH_SHORT).show()
    }

    private fun showResponse(message: String) {
        requireActivity().runOnUiThread {
            Toast.makeText(requireContext(), message, Toast.LENGTH_SHORT).show()
        }
    }
}
```

Gambar 4 11 Program tambah sidik jari

d) Testing

Pada sesi pengujian kami memastikan bahwa fitur dari aplikasi yang kami buat berfungsi sesuai dengan apa yang direncanakan. jika menekan tombol *button* pilih id maka akan masuk ke tampilan untuk memasukan id lalu jika menekan *button scan* sidik jari maka akan memunculkan tampilan jari lalu *user* diharapkan untuk *scan* sidik jari pada *fingerprint* sensor. Jika hal kedua tersebut tidak bisa atau masih ada terjadi kesalahan mungkin peneliti terjadi kekeliruan terhadap pengkodean.

e) Deploy

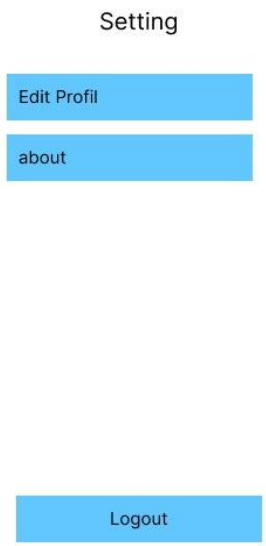
Sesi terakhir dari pembuatkan fitur login multilevel ini kami build dan luncurkan untuk menambahkan fitur ini ke aplikasi kami. Kami akan memastikan lagi bahwa fitur kami siap untuk digunakan.

4.1.6. *SPRINT* 6 : *Setting*

a) Plan

Pada tahap ini yaitu tahap perancangan, kami fokus perencanaan proyek untuk sesi *setting*. Sesi *setting* ini mencakupi fitur *edit profil*, *about* dan *logout*. Dimana jika kita mengakses *edit profil* maka user bisa mengubah nama, *username* dan *password*, lalu jika *user* mengakses *about* maka akan muncul tampilan biodata dari peneliti yang melakukan penelitian, kemudian yang terakhir fitur *logout*, untuk keluar dari akun yang *user* gunakan.

b) Design



Gambar 4 12 Desain *setting*

Kami mendesain antarmuka dan penempatan resource berupa *button* yang bernama *Edit profil*, *about* dan *logout*.

c) *Develop*

Di tahap *develop* atau pengembangan ini kami mulai membuat aplikasi berdasarkan rancangan yang telah dibuat sebelumnya. Kami menuliskan kode xml untuk antarmuka *layout*, lalu menggunakan Bahasa pemrograman kotlin untuk menjalankan logika *backend* program seperti mengarahkan ke *layout* mana jika *button* di klik.

```
class SettingFragment : Fragment() {

    lateinit var prefHelper: PrefHelper

    private lateinit var profileImageView: ImageView
    private lateinit var usernameTextView: TextView
    private lateinit var listView: ListView
    private lateinit var linearLayout: LinearLayout


    private val apiService: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl("https://scriptdoorlock.000webhostapp.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(ApiService::class.java)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_setting, container, false)

        profileImageView    = view.findViewById(R.id.profileImageView)
        usernameTextView    = view.findViewById(R.id.usernameTextView)
        listView            = view.findViewById(R.id.listView)
        linearLayout        = view.findViewById(R.id.linearEditUser)
        prefHelper          = PrefHelper(requireActivity())

        val loggedInUserId  = prefHelper.getUserId()
        val items           = arrayOf("Kontak Kami", "Tentang Aplikasi")

        val adapter = ArrayAdapter<String>(
            requireContext(),
            R.layout.list_item_example,
            R.id.textView,
            items
        )

        listView.adapter = adapter

        listView.onItemClickListener = AdapterView.OnItemClickListener { _, _, position, _ ->
            val selectedItem = items[position]
            startIntent(selectedItem)
        }

        linearLayout.setOnClickListener {
            onLinearLayoutClick(it)
        }

        apiService.getUser(loggedInUserId).enqueue(object : Callback<User> {
            override fun onResponse(call: Call<User>, response: Response<User>) {
                Log.d("SettingsActivity", "onResponse")
                if (response.isSuccessful) {
                    val user = response.body()
                    if (user != null) {
                        val initial = getInitials(user.nama)
                        val bitmap = textAsBitmap(initial)
                        profileImageView.setImageBitmap(bitmap)
                        Handler(Looper.getMainLooper()).post {

profileImageView.setContentDescription(getString(R.string.profile_image_description, user.nama))
                            usernameTextView.text = user.nama
                        }
                    }
                }
                else {
                    Log.e("SettingsActivity", "Failed to get user. Code: ${response.code()}")
                }
            }

            override fun onFailure(call: Call<User>, t: Throwable) {
                Log.e("SettingsActivity", "Failed to get user. Error: ${t.message}")
            }

        })
        return view
    }

    private fun getInitials(name: String): String {
        val names = name.split(" ")
        return if (names.size > 1) {
            val firstInitial = names[0].first()
            val lastInitial = names[1].first()
            "$firstInitial$lastInitial"
        } else {
            name.first().toString()
        }
    }

    private fun onLinearLayoutClick(view: View) {
        val fragmentManager = requireActivity().supportFragmentManager
        val fragmentTransaction = fragmentManager.beginTransaction()
        val tujuanFragment = EditUserFragment()

        fragmentTransaction.replace(R.id.fragment_container, tujuanFragment)
        fragmentTransaction.addToBackStack(null)
        fragmentTransaction.commit()
    }

    private fun textAsBitmap(text: String): Bitmap {
        val paint = Paint(Paint.ANTI_ALIAS_FLAG)
        paint.color = Color.BLUE
        paint.textSize = 50f
        paint.textAlign = Paint.Align.CENTER

        val baseline = -paint.ascent()
        val width = (paint.measureText(text) + 0.5f).toInt()
        val height = (baseline + paint.descent() + 0.5f).toInt()

        val image = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888)
        val canvas = Canvas(image)
        canvas.drawText(text, (width / 2).toFloat(), baseline, paint)

        return image
    }

    private fun startIntent(selectedItem: String) {
        when (selectedItem) {
            "Kontak Kami" -> {
            }
            "Tentang Aplikasi" -> {
            }
        }
    }
}
```

Gambar 4 13 Program *Setting*

d) Testing

Pada sesi pengujian kami memastikan bahwa fitur dari aplikasi yang kami buat berfungsi sesuai dengan apa yang direncanakan. Kami menguji apabila mengakses *button edit profil* apa yang akan keluar, dan jika peneliti mengakses fitur *about* apa yang akan muncul, lalu yang terakhir jika peneliti mengakses *logout* apakah bisa keluar dari akun atau tidak.

e) Deploy

Sesi terakhir dari pembuatkan fitur *setting* ini kami *build* dan luncurkan untuk menambahkan fitur ini ke aplikasi kami. Kami akan memastikan lagi bahwa fitur kami siap untuk digunakan.

4.1.7. *SPRINT* 7 : *Monitoring*

a) Plan

Pada tahap ini yaitu tahap perancangan, kami fokus perencanaan proyek untuk sesi *monitoring*. Sesi *monitoring* ini mencakupi fitur dimana kita bisa memonitoring kapan pintu terbuka, jika pintu terbuka maka akan muncul notif pada tampilan *monitoring*.

b) Design

Gambar 4 14 Desain *Monitoring*

Kami mendesain antarmuka dan penempatan resource berupa *textview* dan elemen kubus sebagai *background* dari *textview*.

c) Develop

Di tahap *develop* atau pengembangan ini kami mulai membuat aplikasi berdasarkan rancangan yang telah dibuat sebelumnya. Kami menuliskan kode xml untuk antarmuka *layout*, lalu menggunakan Bahasa pemrograman kotlin untuk menjalankan logika *backend* program seperti mengarahkan ke *layout* mana jika *button* di klik.

```
class MonitoringFragment : Fragment() {

    private lateinit var monitoringViewModel: MonitoringViewModel
    private lateinit var monitoringAdapter: MonitoringAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_monitoring, container, false)

        monitoringViewModel = ViewModelProvider(this)[MonitoringViewModel::class.java]

        val recyclerView: RecyclerView = view.findViewById(R.id.recyclerView)
        monitoringAdapter = MonitoringAdapter()

        recyclerView.adapter = monitoringAdapter
        recyclerView.layoutManager = LinearLayoutManager(requireContext())

        monitoringViewModel.monitoringList.observe(viewLifecycleOwner) { monitoringList ->
            monitoringAdapter.submitList(monitoringList)
        }

        return view
    }
}
```

Gambar 4 15 Program *Monitoring*

d) Testing

Pada sesi pengujian kami memastikan bahwa fitur dari aplikasi yang kami buat berfungsi sesuai dengan apa yang direncanakan. Kami menguji apabila membuka pintu apakah muncul notifikasi *monitoring* atau tidak.

e) Deploy

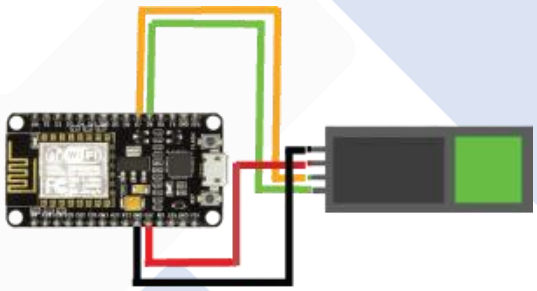
Sesi terakhir dari pembuatkan fitur *login multilevel* ini kami *build* dan luncurkan untuk menambahkan fitur ini ke aplikasi kami. Kami akan memastikan lagi bahwa fitur kami siap untuk digunakan.

4.2. Pembuatan Alat dan Pengkodean Sistem

Pada tahap ini peneliti ingin melakukan tahap-tahap pembuatan hardware dan pengkodean hardware.

4.2.1. Rangkaian Sensor Sidik jari

Rangkaian Sensor Sidik jari pada perangkat ini berfungsi untuk menggerakan *selenoid door*. Sensor sidik jari ini adalah rangkaian system keamanan utama, agar sensor sidik bekerja maka harus disambungkan oleh esp 8266 dimana *sensor fingerprint* sendiri bisa dinyalakan menggunakan 3V dan untuk inputan sensor sendiri menggunakan RX TX Seperti rangkaian dibawah ini:

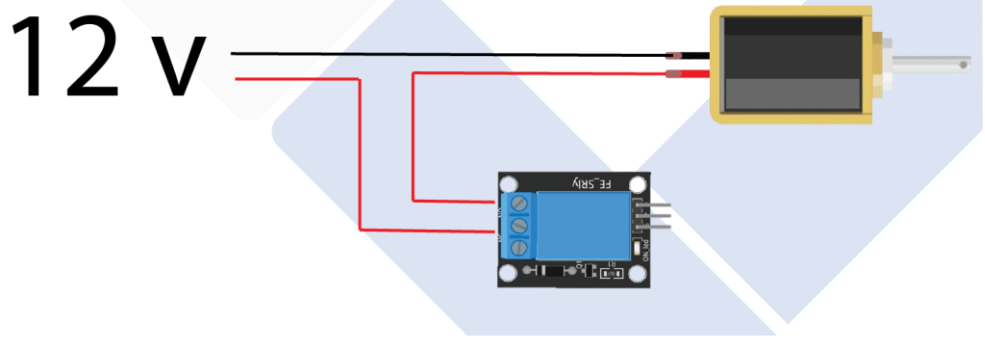


Gambar 4 16 Skema *Fingerprint Sensor*

Fingerprint	ESP 8266
VCC	3.3v
GND	GND
RX	D7
TX	D8

4.2.2. Rangkaian *Selenoid*

Rangkaian di bawah ini memberikan pandangan terperinci mengenai langkah-langkah yang diperlukan untuk mengoperasikan *selenoid*. *Selenoid* yang terintegrasi dalam rangkaian ini memiliki spesifikasi tegangan operasional sekitar 9V hingga 12V, dan untuk menjalankannya secara optimal, diperlukan sumber daya setara, yaitu sekitar 12V. Dalam pemilihan sumber daya, pilihan diantaranya adalah menggunakan *power supply* atau baterai. Dalam konteks penelitian ini, kami memutuskan untuk memanfaatkan baterai sebagai sumber daya utama. Penggerakan *selenoid* dan pengaktifan *mikrokontroler* dilakukan dengan menggunakan energi yang disediakan oleh baterai, memberikan fleksibilitas dan mobilitas dalam pelaksanaan eksperimen, serta memungkinkan analisis yang dinamis terhadap kinerja *selenoid* dan *mikrokontroler* yang sedang diteliti..



Gambar 4 17 Skema *Selenoid*

4.2.3. Pengkodean Alat

4.2.4. Penkodean Daftar Sidik jari

Pada Tahap ini penulis mengkode untuk mendaftar sidik jari yang akan didaftar.

```
uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
tln("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  p = finger.image2Tz(1);
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  Serial.print("Creating model for #");  Serial.println(id);

  p = finger.createModel();
  if (p == FINGERPRINT_OK) {
    Serial.println("Prints matched!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_ENROLLMISMATCH) {
    Serial.println("Fingerprints did not match");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  return true;
}
```

Gambar 4 18 Program Sidik jari

a) *uint8_t getFingerprintEnroll()* ini merupakan sebuah fungsi yang mengembalikan nilai tipe data uint8_t (*byte*). Fungsi ini bertujuan untuk melakukan proses pendaftaran sidik jari.
b) *int p = -1;* Membuat variabel p bertipe data *integer* yang digunakan untuk menyimpan status hasil operasi dari fungsi-fungsi yang dipanggil. Nilainya diinisialisasi dengan -1.
c) Program mulai dengan menampilkan pesan "*Waiting for valid finger to enroll as #*".
d) Selama sidik jari belum diambil dengan benar (status p bukan *FINGERPRINT_OK*), program akan berada dalam *loop* dan menampilkan pesan sesuai dengan status saat ini.
e) Proses ini berlangsung hingga berhasil mengambil gambar sidik jari.
f) Setelah gambar sidik jari diambil, dilakukan konversi gambar ke template sidik jari menggunakan fungsi *finger.image2Tz(1).*
g) Hasil konversi dicatat dalam variabel p.
h) Setelah konversi, program akan mengevaluasi nilai p dan memberikan *output* sesuai dengan kondisi, seperti jika konversi berhasil *(FINGERPRINT_OK),* gambar terlalu rumit *(FINGERPRINT_IMAGEMESS),* terjadi kesalahan komunikasi *(FINGERPRINT_PACKETRECIEVEERR*), atau tidak dapat menemukan fitur sidik jari *(FINGERPRINT_FEATUREFAIL).*
i) Jika pembuatan model berhasil, program menyimpan model sidik jari ke dalam database menggunakan *finger.storeModel(id).*
j) Program mengevaluasi nilai p dan memberikan *output* sesuai dengan kondisi, seperti jika penyimpanan berhasil*(FINGERPRINT_OK),* terjadi kesalahan komunikasi *(FINGERPRINT_PACKETRECIEVEERR),* model tidak dapat disimpan di lokasi tersebut *(FINGERPRINT_BADLOCATION),* terjadi kesalahan pada flash *(FINGERPRINT_FLASHERR),* atau terjadi kesalahan tidak diketahui.

4.2.5. Pengkodean Pengenalan Sidik Jari

Pada Tahap ini peneliti ingin mencoba apakah sidik jari yang tadi didaftar dikenali atau tidak oleh sistem.

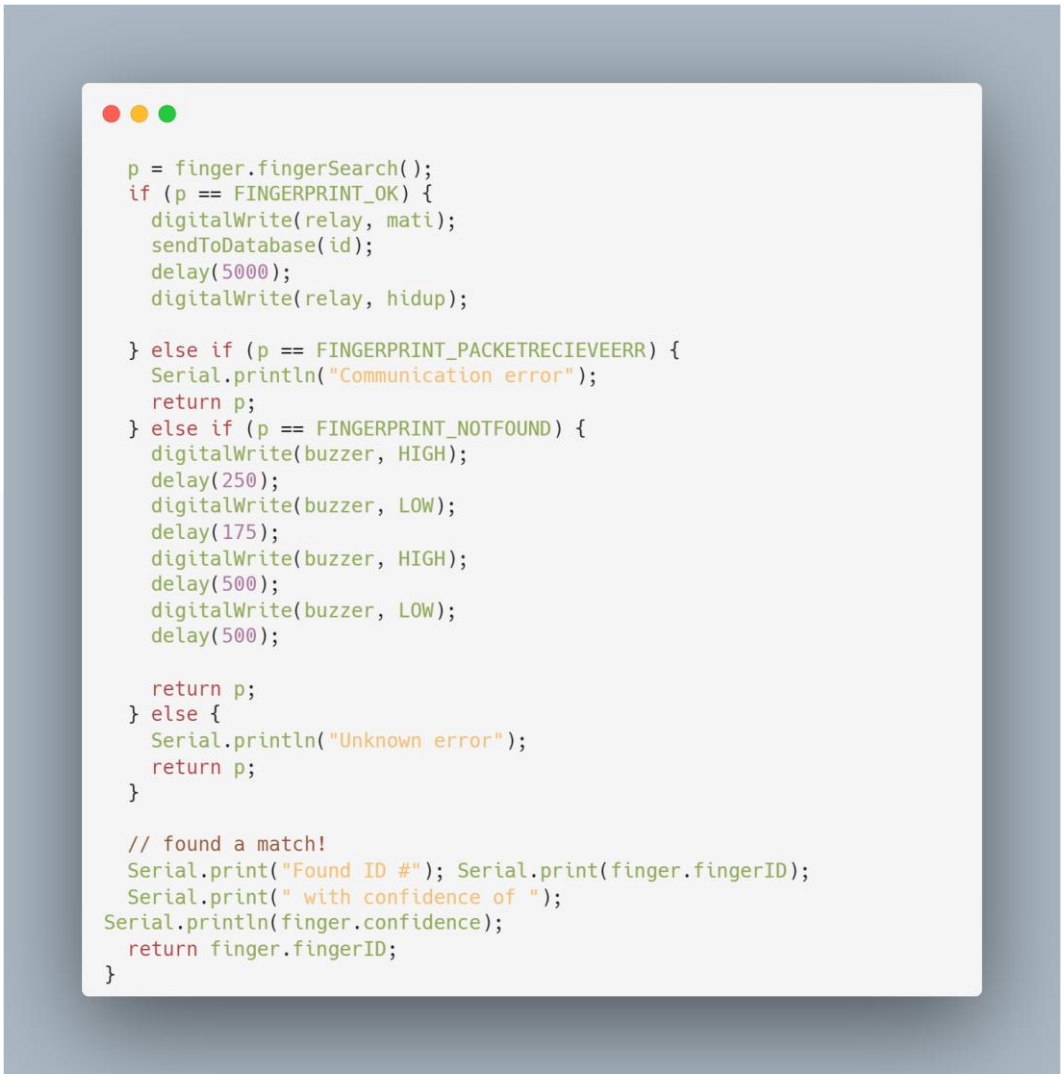
```
uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      //Serial.println("No finger detected");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  p = finger.image2Tz();
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  p = finger.fingerSearch();
  if (p == FINGERPRINT_OK) {
    digitalWrite(relay, mati);
    sendToDatabase(id);
    delay(5000);
    digitalWrite(relay, hidup);

  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    digitalWrite(buzzer, HIGH);
    delay(250);
    digitalWrite(buzzer, LOW);
    delay(175);
    digitalWrite(buzzer, HIGH);
    delay(500);
    digitalWrite(buzzer, LOW);
    delay(500);

    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);
  return finger.fingerID;
}

int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);
}
```

Gambar 4 19 Program Pengenalan Sidik Jari

a) Fungsi *getImage()* dipanggil untuk mengambil gambar sidik jari.
b) Hasilnya disimpan dalam variabel p.
c) Dilakukan pengecekan dengan *switch* untuk menangani hasil seperti jika gambar berhasil diambil *(FINGERPRINT_OK),* tidak ada sidik jari terdeteksi *(FINGERPRINT_NOFINGER),* atau terjadi kesalahan komunikasi *(FINGERPRINT_PACKETRECIEVEERR).*
d) Jika terjadi kesalahan atau tidak ada sidik jari yang terdeteksi, fungsi langsung mengembalikan nilai p. Jika berhasil, program melanjutkan ke langkah berikutnya.
e) Fungsi *image2Tz()* digunakan untuk mengonversi gambar sidik jari ke dalam *template* sidik jari.
f) Kembali dilakukan pengecekan hasil dengan switch.
g) Jika berhasil *(FINGERPRINT_OK),* program melanjutkan ke langkah selanjutnya. Jika terjadi kesalahan, nilai p dikembalikan.
h) Fungsi *fingerSearch()* digunakan untuk mencocokkan sidik jari dengan template yang sudah tersimpan.
i) Jika sidik jari ditemukan *(FINGERPRINT_OK),* dilakukan beberapa tindakan seperti membuka *solenoid door*, mengirim data ke database, dan menunggu beberapa saat sebelum menutup *solenoid door* kembali.
j) Jika terjadi kesalahan komunikasi, program mengembalikan nilai p.
k) Jika sidik jari terdeteksi dan cocok, informasi ID ditampilkan melalui *Serial Monitor.*
l) Fungsi mengembalikan nilai ID sidik jari.
m) Jika sidik jari tidak ditemukan, beberapa tindakan spesifik dilakukan, seperti menyalakan *buzzer* dengan pola tertentu. Nilai p juga dikembalikan.
n) Jika sidik jari terdeteksi dan cocok, informasi ID ditampilkan melalui *Serial Monitor.*
o) Fungsi mengembalikan nilai ID sidik jari.

4.2.6. Pengkodean membuka *selenoid* menggunakan sidik jari

Selanjutnya peneliti akan mencoba membuka *selenoid* menggunakan sidik jari yang sudah terdaftar tadi



```
  p = finger.fingerSearch();
  if (p == FINGERPRINT_OK) {
    digitalWrite(relay, mati);
    sendToDatabase(id);
    delay(5000);
    digitalWrite(relay, hidup);

  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    digitalWrite(buzzer, HIGH);
    delay(250);
    digitalWrite(buzzer, LOW);
    delay(175);
    digitalWrite(buzzer, HIGH);
    delay(500);
    digitalWrite(buzzer, LOW);
    delay(500);

    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);
  return finger.fingerID;
}
```

Gambar 4 20 Program membuka *Selenoid*

a) Melakukan pencarian sidik jari menggunakan fungsi *fingerSearch()* dari modul sidik jari.
b) Hasil pencarian disimpan dalam variabel p.
c) Jika pencarian berhasil *(FINGERPRINT_OK)*, maka kode di dalam blok akan dieksekusi.
d) Jika terjadi kesalahan komunikasi *(FINGERPRINT_PACKETRECIEVEERR)*, pesan kesalahan

"Communication error" akan dicetak di Serial dan nilai p akan dikembalikan.

e) Jika sidik jari tidak ditemukan *(FINGERPRINT_NOTFOUND)*, modul *fingerprint* akan memberikan sinyal pada *buzzer* dengan pola tertentu, dan nilai p akan dikembalikan.
f) Jika terjadi kesalahan yang tidak diketahui, pesan *"Unknown error"* akan dicetak di Serial dan nilai p akan dikembalikan.
g) Jika sidik jari ditemukan, *relay* akan dimatikan (menandakan bahwa *solenoid door* terbuka).
h) Data ID sidik jari akan dikirim ke database melalui fungsi *sendToDatabase(id).*
i) Terdapat *delay* 5000 milidetik (5 detik) sebelum *relay* dihidupkan kembali (menandakan bahwa *solenoid door* tertutup kembali).
j) Jika sidik jari ditemukan, informasi ID sidik jari akan dicetak di Serial.
k) Fungsi akan mengembalikan nilai ID sidik jari.

4.3. Perangkat Keras (*Hardware*)

Perangkat keras yang sudah berhasil dirancang pada penelitian ini adalah sistem *smartlock for door* menggunakan sidik jari berbasis Iot.

Gambar 4 21 Hasil Perangkat Keras

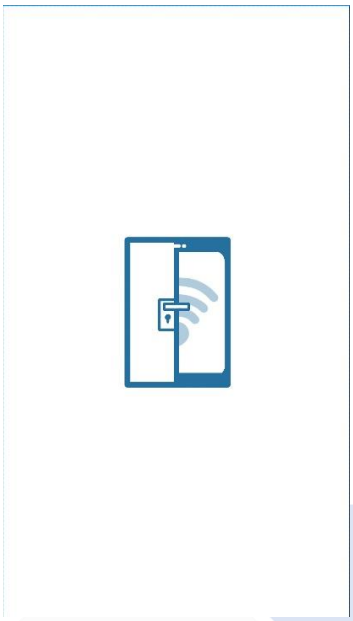
Hardware ini terdiri dari esp 8266, *module relay*, *sensor fingerprint*, *baterai*, *selenoid door*, kabel *jumper* beserta komponen pendukung lainnya.

4.4. Tampilan Aplikasi

Pada tahap kali ini peneliti membuat tampilan aplikasi dari *Smartlock* dimana peneliti kali ini membuat aplikasi ini menggunakan *software* android studio.

4.4.1. Tampilan *Splash Screen*

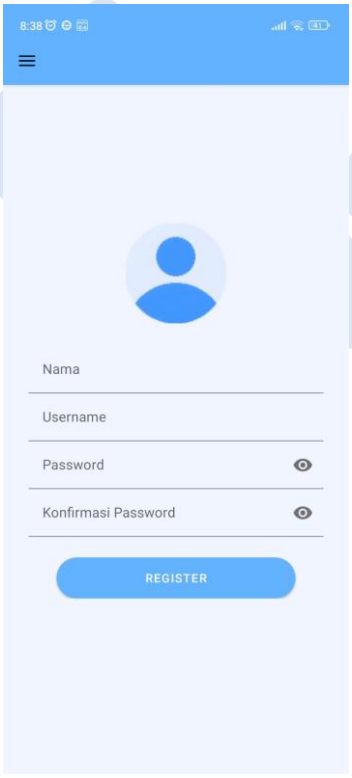
Berikut tampilan *splash screen* untuk aplikasi *smartlock for door* dan juga logo aplikasi *smartlock for door* untuk penelitian kami, *splash screen* adalah gambar atau animasi yang muncul sebentar saat aplikasi atau program dimulai.



Gambar 4 22 Tampilan *Splash Screen*

4.4.2. Tampilan *Add User*

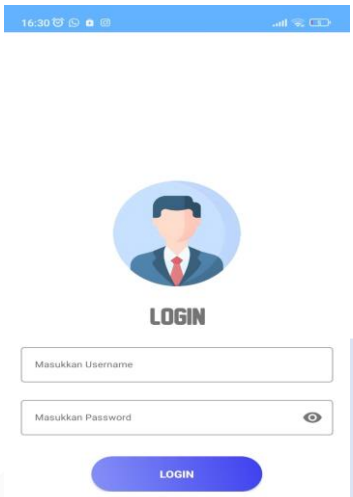
User bisa menambahkan atau membuat akun dengan memasukkan nama, *username*, *password* dan konfirm *password*.



Gambar 4 23 Tampilan *Add User*

4.4.3. Tampilan Login

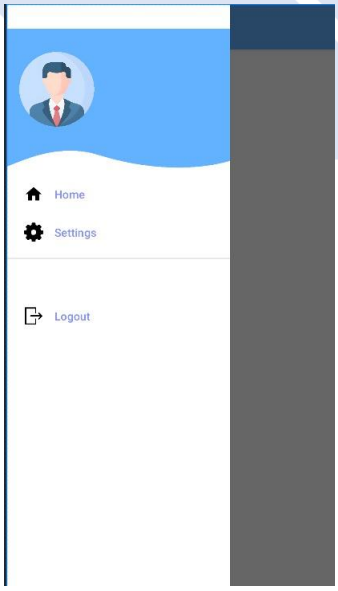
Untuk tampilan login sendiri seperti pada umumnya user diharapkan memasukkan *username* dan *password*.



Gambar 4 24 Tampilan *Login*

4.4.4. Tampilan *Nav Bar*

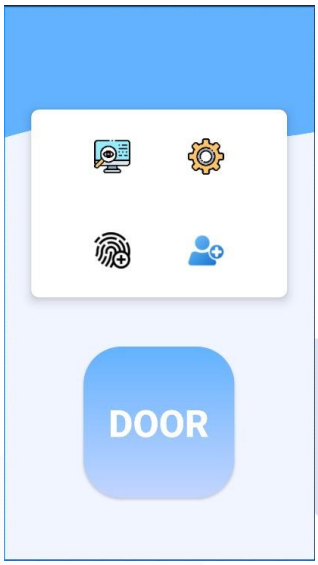
Tampilan *Nav bar* ini berguna untuk membuka halaman *setting* dan jika ingin melakukan *logout*.



Gambar 4 25 Tampilan *NavBar*

4.4.5. Tampilan *Main* Admin

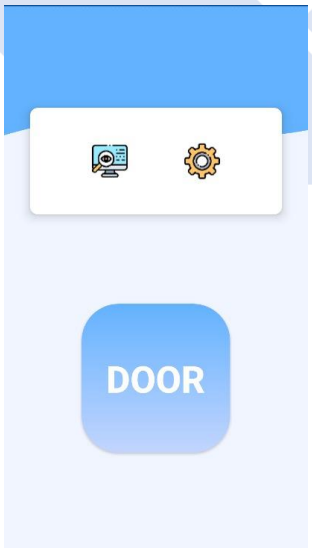
Berikut untuk tampilan admin dimana ada menu *monitoring, setting*, daftar sidik jari, dan tambah *user*



Gambar 4 26 Tampilan *Main Admin*

4.4.6. Tampilan *Main User*

Untuk tampilan *user* sendiri lebih sedikit dari tampilan admin dimana untuk tampilan *user* ada Menu *monitoring, setting dan door* dimana *door* ini digunakan untuk IOT atau membuka pintu secara IOT.



Gambar 4 27 Tampilan *Main User*

4.4.7. Tampilan *Add Fingerprint*

Berikut tampilan daftar sidik jari yang ada ditampilan admin dimana *button* PILIH ID itu untuk mendata jari yang akan didaftar jika sudah selesai maka user akan diharapkan menempelkan jari pada sensor sidik jari.

Gambar 4 28 Tampilan *Add Fingerprint*

4.4.8. Tampilan *Scan Fingerprint*

Lalu untuk tampilan didalam *Scan* sidik jari tersebut seperti dibawah ini jika admin sudah menekan tombol *scan* maka *user* diharapkan menempelkan sidik jari yang akan didaftar.

Gambar 4 29 Tampilan *Scan Fingerprint*

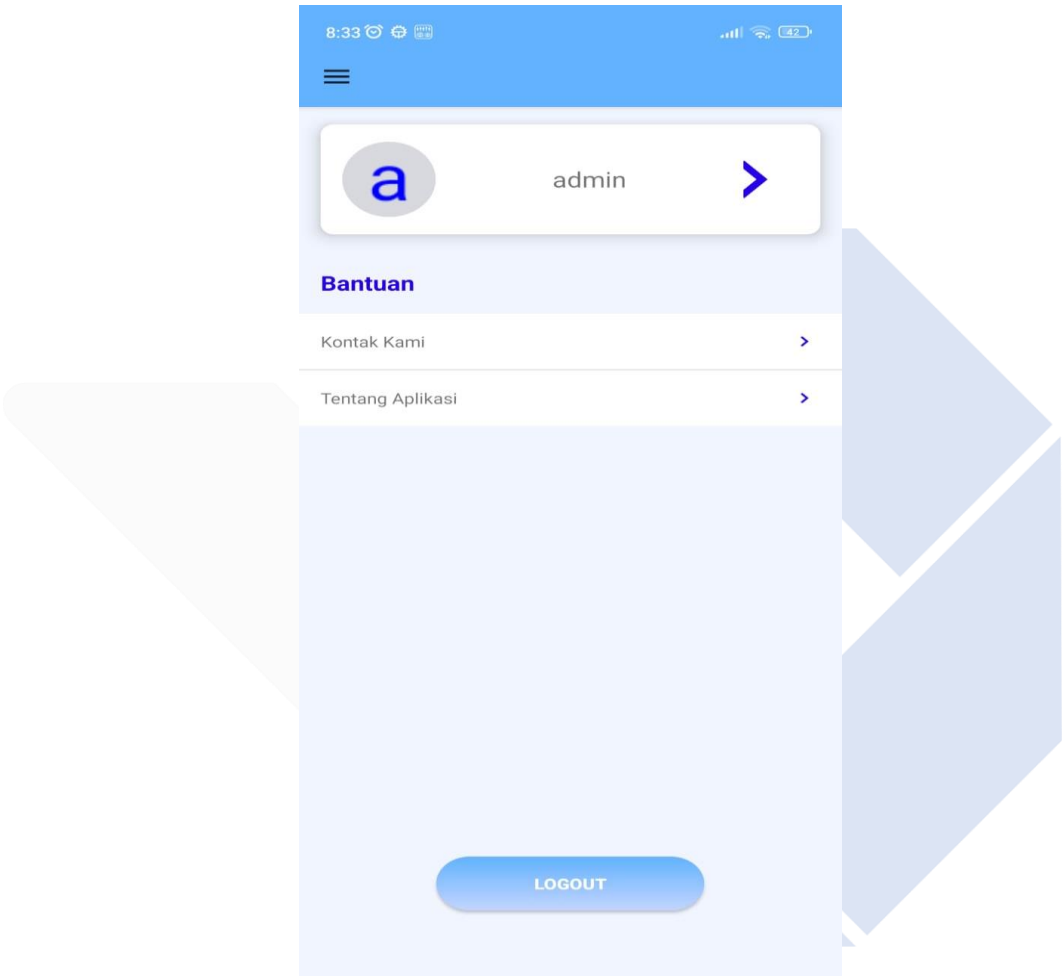
4.4.9. Tampilan *IOT Door*

Tampilan *IOT door* sendiri ada pada 2 tampilan yaitu tampilan *user* dan admin menu IOT door ini digunakan untuk membuka pintu melalui *handphone*

gambar 4 30 Tampilan *Iot Door*

4.4.10. Tampilan *Setting*

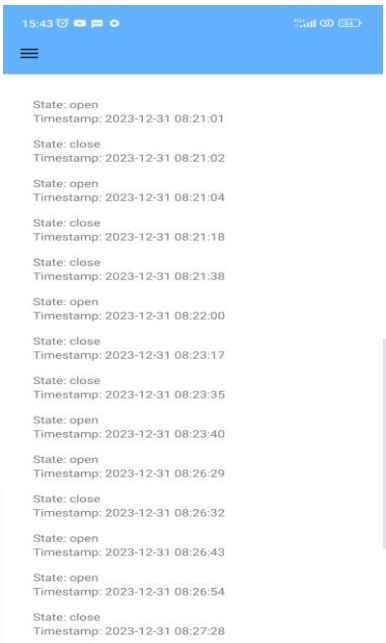
Tampilan didalam *Setting* sendiri terdapat *about, contact person*, dan *logout*. Dimna about berisi tentang aplikasi *smartlock*, lalu untuk edit profil *user* bisa mengedit *password* atau *username user*.



Gambar 4 31 Tampilan *Setting*

4.4.11. Tampilan *Monitoring*

Tampilan *monitoring* sendiri seperti nama menunya yaitu monitoring dimana *user* bisa memonitoring kapan saja pintu terbuka.



Gambar 4 32 Tampilan *Setting*

4.5. Pengujian

Pada tahap ini peneliti melakukan pengujian pada alat yang diteliti guna memastikan kinerjanya optimal dan sesuai dengan standar yang ditetapkan.

4.5.1. Pengujian Sidik Jari

Tabel 4 1 Tabel hasil pengujian sidik jari

No	Status Jari	Respon Sensor	Presentase
1	Terdaftar	Valid	100%
2	Terdaftar	Valid	100%
3	Terdaftar	Valid	100%
4	Terdaftar	Valid	100%

5	Terdaftar	Valid	100%
6	Tidak Terdafatar	Tidak Valid	0%
7	Tidak Terdaftar	Tidak Valid	0%
8	Tidak Terdaftar	Tidak Valid	0%
9	Tidak Terdaftar	Tidak Valid	0%
10	Tidak Terdaftar	Tidak Valid	0%

Pada kali ini kami menguji pengenalan sensor dimana kami menguji jari yang sudah terdaftar dengan jari yang belum terdaftar apakah sensor valid atau tidak menteksi jari yang terdaftar.

4.5.2. Pengujian Sensor

Tabel 4 2 Tabel hasil pengujian sensor

No	User	Jari	Status	Waktu
1	Penguji 1	Jempol	Terdeteksi	0,89 ms
2	Penguji 1	Jempol	Terdeteksi	1 detik
3	Penguji 1	Jempol	Terdeteksi	1 detik
4	Penguji 1	Jempol	Tidak Terdeteksi	Tidak Terdeteksi
5	Penguji 1	Jempol	Terdeteksi	0,9 ms
6	Penguji 1	Jempol	Tidak Terdeteksi	Tidak Terdeteksi
7	Penguji 1	Jempol	Terdeteksi	0,75 ms
8	Penguji 1	Jempol	Terdeteksi	1 detik
9	Penguji 1	Jempol	Terdeteksi	0,87 ms
10	Penguji 1	Jempol	Terdeteksi	0,9 ms

11	Penguji 2	Jempol	Terdeteksi	1 detik
12	Penguji 2	Jempol	Tidak Terdeteksi	Tidak Terdeteksi
13	Penguji 2	Jempol	Terdeteksi	0,7 ms
14	Penguji 2	Jempol	Terdeteksi	0,34 ms
15	Penguji 2	Jempol	Terdeteksi	2 detik
16	Penguji 2	Jempol	Tidak Terdeteksi	Tidak Terdeteksi
17	Penguji 2	Jempol	Tidak Terdeteksi	Tidak Terdeteksi
18	Penguji 2	Jempol	Terdeteksi	0,6 ms
19	Penguji 2	Jempol	Terdeteksi	1 detik
20	Penguji 2	Jempol	Terdeteksi	0,79 ms

Disini kami menguji terhadapat 2 orang yang dimana kami 10 kali melakukan pengujian terhadap sensor apakah berjalan dengan baik atau tidak, dan untuk hasil pengujiannya penguji 1 mencoba sensor sebanyak 10 kali dan terdapat 2 kali sensor tidak dapat mendeteksi jari dari penguji 1, untuk penguji 2 juga melakukan percobaan sensor 10x dan terdapat 3x sensor tidak dapat mendeteksi jari dari penguji 2.

4.5.3. Pengujian Perbandingan

Table 4 3 Hasil pengujian perbandingan

No	Penggunaan	Waktu Pintu Terbula
1	Kunci Konvensional	3 detik
2	Kunci Konvensional	2 detik
3	Kunci Konvensional	4 detik
4	*Smartlock*	0,89 ms
5	*Smartlock*	1 detik
6	*Smartlock*	0,75 ms

Disini peneliti melakukan perbandingan antara kunci konvensional dan *Smartlock* mana yang lebih efektif dalam pembukaan pintu, peneliti melakukan pengujian 3 kali setiap kunci konvensional dan *smartlock*.

4.5.4. Pengujian Keseluruhan

Tabel 4 4 Tabel hasil pengujian keseluruhan

No	Uji Coba Jari	Status Sensor	Status *Selenoid*	Status *Buzzer*
1	Jempol	Terdeteksi	Terbuka	Mati
2	Telunjuk	Tidak Terdeteksi	Tidak Terbuka	Hidup

Peneliti melakukan pengujian keseluruhan dari hardware berjalan dengan semestinya atau tidak dimana peneliti melakukan pengujian pada jari yang terdaftar apakah sensor bisa mendeteksi jari tersebut dan apakah *selenoid* terbuka atau tidak, dan jika jari tidak terdeteksi maka *buzzer* akan berbuyi 2 kali

4.5.5. Hasil Pengujian Kuesioner

Pengujian kuesioner telah dilakukan dengan memberikan evaluasi terhadap pertanyaan yang disajikan pada tabel 4, yang diberikan kepada responden secara acak.

Table 4 5 Tabel Pertanyaan Kuisioner

No	Pertanyaan
1	Apakah tampilan aplikasi smartlock mudah dipahami?
2	Apakah fitur aplikasi smartlock mudah digunakan?
3	Apakah aplikasi Smartlock mudah digunakan?
4	Apakah aplikasi Smartlock memudahkan dalam pengendalian pintu?
5	Apakah aplikasi smartlock dapat mengatasi masalah saat lupa mengunci pintu?
6	Apakah aplikasi Smartlock memudahkan pendaftaran sidik jari?
7	Apakah rancangan smartlock for door dapat meningkatkan keamanan pintu ruangan kepala sekolah?
8	Apakah Aplikasi Smartlock dapat memudahkan anda dalam memonitoring ruangan kepala sekolah?

1 (Sangat Tidak Setuju)

2 (Tidak Setuju)

3 (Netral)

4 (Setuju)

5 (Sangat Setuju)

Dari pertanyaan kuesioner diatas, diperoleh responden sebanyak 5 orang. Jawaban responden dapat dilihat pada table dibawah ini

Table 4 6 Tabel hasil jawaban kuisioner

No	Nama	Hasil Jawabam							
		1	2	3	4	5	6	7	8
1	Responden 1	5	4	4	5	4	5	4	4
2	Responden 2	5	5	4	5	4	5	4	5
3	Responden 3	5	5	5	4	4	5	4	5
4	Responden 4	5	4	4	4	4	5	4	4
5	Responden 5	5	3	4	5	4	5	3	5

Hasil perhitungan presentase hasil jawaban:

Total = 177

Nilai Presentase = 177/200 * 100%

= 88,5%

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dengan merancang dan membuat aplikasi Smartlock for Door berbasis Android yang terintegrasi dengan sensor sidik jari dan sistem kontrol Internet of Things (IoT), MIN 2 Bangka telah berhasil meningkatkan keamanan pintu ruangan kepala sekolah. Sistem ini tidak hanya menyediakan akses yang aman berbasis sidik jari, tetapi juga memungkinkan pemantauan aktivitas pintu secara real-time. Penerapan Smartlock for Door bertujuan optimalisasi keamanan dan pengendalian pintu ruangan kepala sekolah, memberikan solusi efisien melalui teknologi canggih. Evaluasi sistem ini memastikan keefektifan dan kehandalan dengan cara melakukan pengujian kunci konvensional dan *Smartlock* yang hasilnya *smartlock* lebih efektif dan handal dalam waktu pembukaan pintu, yang dimana *smartlock* lebih cepat 1-2 detik dari kunci konvensional.

5.2. Saran

a) Alat ini masih dapat dikembangkan dari sisi desain androidnya.
b) Perlu adanya tempat cas baterai ditempat yang dekat dimana hal ini bisa menjadi suatu masalah jika sumber daya dari baterai tersebut habis.

DAFTAR PUSTAKA

[1] P. Indonesia, "Peraturan Pemerintah Nomor 17 Tahun 2010 tentang Pengelolaan dan Penyelenggaraan Pendidikan," L. Negara, ed., Lembaran Negara, 2010.

[2] F. Kurniawan, and A. Surahman, "Sistem Keamanan Pada Perlintasan Kereta Api Mengunakan Sensor Infrared Berbasis Mikrokontroller Arduino Uno," *Jurnal Teknologi Dan Sistem Tertanam,* vol. 2, no. 1, pp. 7-12, 2021.

[3] P. E. S. Dita, A. Al Fahrezi, P. Prasetyawan *et al.*, "Sistem Keamanan Pintu Menggunakan Sensor Sidik Jari Berbasis Mikrokontroller Arduino UNO R3," *Jurnal Teknik Dan Sistem Komputer,* vol. 2, no. 1, pp. 121-135, 2021.

[4] A. Mude, and L. B. F. Mando, "Implementasi Keamanan Rumah Cerdas Menggunakan Internet of Things dan Biometric Sistem," *MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer,* vol. 21, no. 1, pp. 179-188, 2021.

[5] A. S. Bakhri, K. Suhada, and K. Kamaludin, "Perancangan Sistem Doorlock Menggunakan Aplikasi Blynk Berbasis IoT Studi Kasus Pada Rumah Tempat Tinggal Pribadi." pp. 1-10.

[6] T. N. Murti, I. Ruslianto, and U. Ristian, "Implementasi Sistem Kendali dan Monitoring Keamanan Pintu Berbasis IoT Menggunakan Perangkat Mobile," *JURIKOM (Jurnal Riset Komputer),* vol. 9, no. 6, pp. 1760-1769, 2022.

[7] F. Hady, M. Sholeh, and D. Andayati, "Pengembangan Prototipe Aplikasi Membuka Kunci Pintu Otomatis Menggunakan Metode Sidik Jari Berbasis Nodemcu," *Journal of Computer Science and Technology,* vol. 2, no. 1, pp. 1-7, 2022.

[8] D. Dahlan, Y. Supriadi, and R. Iskandar, "Perancangan Sistem Pembuka Pintu Menggunakan Sidik Jari," *Journal of Informatics and Electronics Engineering,* vol. 2, no. 1, pp. 6-9, 2022.

[9] S. L. Tobing, "Rancang Bangun Pengaman Pintu Menggunakan Sidik Jari (Fingerprint) Dan Smartphone Android Berbasis Mikrokontroler Atmega8," *Jurnal Teknik Elektro Universitas Tanjungpura,* vol. 1, no. 1, 2014.

[10] S. Sahril, R. Suppa, and M. Muhallim, "Sistem Pengunci Pintu Dengan Sidik Jari Menggunakan Arduino," *Jurasik (Jurnal Riset Sistem Informasi dan Teknik Informatika),* vol. 7, no. 1, pp. 19-26, 2022.

[11] R. B. Indak, "Sistem Buka Tutup Pintu Otomatis Berbasis Android Dan Sidik Jari," *Jurnal Cosphi,* vol. 2, no. 1, 2018.

[12] W. Rijanto, and S. Warsito, "RANCANG BAGUN SISTEM KEAMANAN PINTU RUANGAN TATA USAHA FAKULTAS SAINS DAN TEKNOLOGI UNIVERSITAS ISLAM AS-SYAFI'IYAH

MENGGUNAKAN SIDIK JARI DAN ANDROID," *KOCENIN SERIAL KONFERENSI (E) ISSN: 2746-7112*, no. 1, pp. 1.3. 1-1.3. 7, 2021.
[13] M. Iqbal, R. H. Hardyanto, and P. W. Ciptadi, "Sistem Keamanan Ganda Menggunakan Fingerprint dan Keypad Pada Pintu Rumah (Smart Security System)."
[14] S. Ibadurrahman, "RANCANG BANGUN SMART HOME DENGAN KONSEP INTERNET OF THINGS (IOT) BERBASIS ANDROID," University of Technology Yogyakarta, 2020.
[15] S. Ulum, and M. Budiyanto, "Prototipe Pengaman Pintu Rumah Menggunakan Voice Recognition dengan EasyVR Berbasis Mikrokontroler," *Jurnal Listrik, Instrumentasi dan Elektronika Terapan (JuLIET)*, vol. 1, no. 2, 2020.
[16] A. Hanafie, N. P. Husain, S. Sukirman *et al.*, "ALAT KONTROL AKSES PINTU MENGGUNAKAN SIDIK JARI DENGAN NOTIFIKASI TELEGRAM BERBASIS IOT," *ILTEK: Jurnal Teknologi*, vol. 18, no. 01, pp. 1-5, 2023.
[17] F. H. Ramadhan, I. E. Maulani, N. F. Zuhriyah *et al.*, "Pengembangan Aplikasi Mobile Learning Berbasis Android untuk Meningkatkan Keterampilan Pemrograman pada Mahasiswa Teknik Informatika," *Jurnal Sosial dan Teknologi*, vol. 3, no. 2, pp. 108-113, 2023.
[18] R. Putri, and R. Theis, "EFFECT OF PSYCHOLOGICAL FACTORS IN LEARNING TO LEARNING OUTCOMES ON REAL ANALYSIS," *Prosiding SEMIRATA 2017 Bidang MIPA BKS-PTN-Barat*, pp. 534-541, 2017.
[19] R. Setiawan, "SISTEM IDENTIFIKASI SIDIK JARI PADA DINAS KEPOLISIAN WILAYAH SIDOARJO DENGAN KOMBINASI METODE GALTON HENRY DAN TRANSFORMASI FOURIER," *Kurawal-Jurnal Teknologi, Informasi dan Industri*, vol. 4, no. 1, pp. 31-42, 2021.
[20] H. T. Saputra, A. Muhaimin, and B. Kurniawan, "Sistem Kontrol Kunci Pintu Rumah Menggunakan Fingerprint Smartphone Android Berbasis Arduino Uno," *Jurnal Ilmu Komputer*, vol. 11, no. 1, pp. 5-9, 2022.
[21] M. Arifin, "Sistem Pengamanan Mesin Atm Dengan Menggunakan Pengenalan Sidik Jari Dan Wajah Face Recognition Untuk Meminimalisir Cyberbanking Crime," *OISAA Journal of Indonesia Emas*, vol. 5, no. 1, pp. 35-42, 2022.
[22] A. S. Ayu, A. T. Hanuranto, and A. Novianti, "Desain Dan Implementasi Purwarupa Pendeteksi Kebakaran Gedung Menggunakan Aplikasi Mobile Berbasis Android Dan Internet Of Things (IOT)," *eProceedings of Engineering*, vol. 7, no. 2, 2020.
[23] P. S. M. Silaban, A. E. Putriku, S. D. Siahaan *et al.*, "Pengembangan Bahan Ajar (Buku) Teori Ekonomi Mikro Berbasis Media Pembelajaran Aplikasi Android," *Cetta: Jurnal Ilmu Pendidikan*, vol. 4, no. 1, pp. 1-17, 2021.
[24] G. I. Lintjewas, S. J. Sumarauw, and R. J. Pulukadang, "Pengembangan Aplikasi Mobile Learning Berbasis Android pada Materi Persamaan

Linear Satu Variabel," *MARISEKOLA: Jurnal Matematika Riset Edukasi dan Kolaborasi*, vol. 3, no. 1, pp. 57-64, 2022.
[25] R. P. Dewi, "Studi Kasus-Metode Penelitian Kualitatif," 2019.
[26] M. Rahardjo, "Studi kasus dalam penelitian kualitatif: konsep dan prosedurnya," 2017.
[27] SIANTURI, Agus Tinus. "Rancang Bangun Sistem Keamanan Ruangan dengan Sensor Cahaya Berbasis Arduino Uno Menggunakan SMS Gateway". 2019. PhD Thesis. Universitas Sumatera Utara.

LAMPIRAN

LAMPIRAN 1: DAFTAR RIWAYAT HIDUP PENULIS

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	:	Wahyu Rizky Algifari
Tempat & Tanggal Lahir	:	Sungailiat, 26 Juni 2002
Alamat Rumah	:	Parit Padang, Sungailiat.
		Hp: 081272166181
		Email: wahyurizky2662@gmail.com
Jenis Kelamin	:	Laki-Laki
Agama	:	Islam

2. Riwayat Pendidikan

MIN 2 BANGKA	:	2008 – 2014
SMPN 2 SUNGAILIAT	:	2014 – 2017
SMKN 1 SUNGAILIAT	:	2017 – 2020

Sungailiat, 27 Desember 2023

Wahyu Rizky Algifari

DAFTAR RIWAYAT HIDUP

1. Data Pribadi

Nama Lengkap	:	Wisnu Fauzaan Pahlevi
Tempat & Tanggal Lahir	:	Pangkalpinang, 26 Maret 2003
Alamat Rumah	:	Jl. Nilam V
		Hp: 082130694164
		Email: wisnufauzaan@gmail.com
Jenis Kelamin	:	Laki-Laki
Agama	:	Islam

2. Riwayat Pendidikan

SD SWADAYA PANGKALPINANG	:	2008 – 2014
MTSN NEGERI PANGKALPINANG	:	2014 – 2017
SMKN 2 PANGKALPINANG	:	2017 – 2020

Sungailiat, 27 Desember 2023

Wisnu Fauzaan Pahlevi

LAMPIRAN 2: KODE PROGRAM

PrefHelper.kt

```
#include <Base64.h>
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <Adafruit_Fingerprint.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WebServer.h>
#include <SoftwareSerial.h>

#define FINGERPRINT_SENSOR_RX 13
#define FINGERPRINT_SENSOR_TX 15

#define WIFI_SSID "Red"
#define WIFI_PASSWORD "12093487"
#define SERVER_PORT 80

#define hidup HIGH
#define mati LOW

SoftwareSerial mySerial(FINGERPRINT_SENSOR_RX,
FINGERPRINT_SENSOR_TX);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

ESP8266WebServer server(80);

uint8_t id;

int relay = 4;
int buzzer = 5;
float batasTegangan = 7.0;

bool solenoidState = false;

void setup() {
  Serial.begin(115200);
  mySerial.begin(57600);

  connectToWiFi();

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
```

```
while (1);
  }

  finger.getTemplateCount();
  Serial.print("Sensor contains ");
  Serial.print(finger.templateCount);
  Serial.println(" templates.");

  pinMode(buzzer, OUTPUT);
  pinMode(relay, OUTPUT);
  digitalWrite(relay, hidup);
  delay(5);

  setupServerRoutes();
}

void loop() {

  int nilaiADC = analogRead(analogInputPin);
  float tegangan = (nilaiADC / 1023.0) * 5.0;

  if (tegangan < batasTegangan) {
    tone(buzzer, 4000);
    delay(1000);
    noTone(buzzer);
  }
  delay(1000);

  getFingerprintID();
  delay(50);
  server.handleClient();
}

void connectToWiFi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(250);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi");

  tone(buzzer, 4000);
  delay(2000);
  noTone(buzzer);
```

```
delay(1000);

  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

void setupServerRoutes() {
  server.on("/", HTTP_GET, handleRoot);
  server.on("/addFingerprint", HTTP_POST, handleAddFingerprint);
  server.on("/status", HTTP_GET, handleStatus);
  server.on("/control", HTTP_POST, handleControl);
  server.begin();
}

void handleRoot() {
  String data = server.arg("data");
  id = data.toInt();
  Serial.print("Received data: ");
  Serial.println(id);

  server.send(200, "text/plain", "Hello from ESP8266!");
}

void handleStatus() {
  String statusMessage = (solenoidState) ? "Solenoid terbuka" :
"Solenoid tertutup";
  server.send(200, "text/plain", statusMessage);
}

// Fungsi untuk mengontrol solenoid
void handleControl() {
  String state = server.arg("state");

  // Lakukan tindakan sesuai dengan nilai state (buka/tutup
solenoid)
  if (state == "open") {
    digitalWrite(relay, HIGH); // Aktifkan solenoid
    solenoidState = true;
  } else if (state == "close") {
    digitalWrite(relay, LOW); // Matikan solenoid
    solenoidState = false;
  } else {
    server.send(400, "text/plain", "Permintaan tidak valid");
    return;
  }
```

```
 // Kirim respons ke aplikasi Kotlin
  server.send(200, "text/plain", "OK");
}

void handleAddFingerprint() {
  Serial.println("Ready to enroll a fingerprint!");
  Serial.println(id);

  while (!  getFingerprintEnroll() );
  server.send(200, "text/plain", "Fingerprint obtained
successfully");

  String fingerprintTemplate = server.arg("template");

  // Convert the fingerprint template string to a non-const buffer
  uint8_t* fingerprintData = new
uint8_t[fingerprintTemplate.length()];
  memcpy(fingerprintData, fingerprintTemplate.c_str(),
fingerprintTemplate.length());



  // Cleanup the dynamically allocated buffer
  delete[] fingerprintData;
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #");
Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
```

```
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  // OK success!

  p = finger.image2Tz(1);
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

Serial.println("Remove finger");
  delay(2000);
  p = 0;
  while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
  }
  Serial.print("ID "); Serial.println(id);
  p = -1;
  Serial.println("Place same finger again");
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
```

```
break;
    case FINGERPRINT_NOFINGER:
      Serial.print(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  // OK success!

  p = finger.image2Tz(2);
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  // OK converted!
  Serial.print("Creating model for #");  Serial.println(id);

  p = finger.createModel();
  if (p == FINGERPRINT_OK) {
```

```
Serial.println("Prints matched!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_ENROLLMISMATCH) {
    Serial.println("Fingerprints did not match");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  return true;
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      //Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      //Serial.println("No finger detected");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
```

```
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK success!

p = finger.image2Tz();
switch (p) {
  case FINGERPRINT_OK:
    //Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
p = finger.fingerSearch();
if (p == FINGERPRINT_OK) {
  digitalWrite(relay, mati);
  sendToDatabase(id);
  delay(5000);
  digitalWrite(relay, hidup);

} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_NOTFOUND) {
  digitalWrite(buzzer, HIGH);
```

```
    delay(250);
    digitalWrite(buzzer, LOW);
    delay(175);
    digitalWrite(buzzer, HIGH);
    delay(500);
    digitalWrite(buzzer, LOW);
    delay(500);

    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);

  return finger.fingerID;
}

// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of ");
Serial.println(finger.confidence);
  return finger.fingerID;
}

void sendToDatabase(uint8_t id) {
  WiFiClient client;

    Serial.println("Sending data to the database...");

    String server = "scrptdoorlock.000webhostapp.com";
```

```
    String url = "/insert_data.php?id=" + String(id);
    Serial.print("Requesting URL: ");
    Serial.println(url);

    if (client.connect(server.c_str(), 80)) {
        client.print(String("GET ") + url + " HTTP/1.1\r\n" +
                     "Host: " + server + "\r\n" +
                     "Connection: close\r\n\r\n");
        delay(500);

        while (client.available()) {
            String line = client.readStringUntil('\r');
            Serial.print(line);
        }
        client.stop();
        Serial.println("\nData sent to the database.");
    } else {
        Serial.println("Connection to the server failed.");
    }
}
```

PrefHelper.kt

```
package com.example.apksmartdoor.helper

import android.content.Context
import android.content.SharedPreferences

class PrefHelper(context: Context) {

    private val sharedPreferences: SharedPreferences =
        context.getSharedPreferences(PREF_NAME,
Context.MODE_PRIVATE)

    companion object {
        private const val PREF_NAME = "MyPreferences"
        private const val PREF_USERNAME = "username"
        private const val PREF_PASSWORD = "password"
        private const val PREF_IS_LOGIN = "isLogin"
        private const val PREF_USER_LEVEL = "userLevel"
        private const val PREF_USER_ID = "userId"
    }

    fun saveSession(username: String, password: String, isLogin:
Boolean, userLevel: Int, userId: Int) {
        val editor = sharedPreferences.edit()
        editor.putString(PREF_USERNAME, username)
        editor.putString(PREF_PASSWORD, password)
        editor.putBoolean(PREF_IS_LOGIN, isLogin)
        editor.putInt(PREF_USER_LEVEL, userLevel)
        editor.putInt(PREF_USER_ID, userId)
        editor.apply()
    }

    fun getUsername(): String? {
        return sharedPreferences.getString(PREF_USERNAME, null)
    }

    fun getPassword(): String? {
        return sharedPreferences.getString(PREF_PASSWORD, null)
    }

    fun getIsLogin(): Boolean {
        return sharedPreferences.getBoolean(PREF_IS_LOGIN, false)
    }

    fun getUserLevel(): Int {
        return sharedPreferences.getInt(PREF_USER_LEVEL, 0)
    }

    fun getUserId(): Int {
        return sharedPreferences.getInt(PREF_USER_ID, 0)
    }

    fun clearSession() {
        val editor = sharedPreferences.edit()
        editor.clear()
```

FingerprintViewModel.kt

```
package com.example.apksmartdoor.model

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.apksmartdoor.repository.FingerprintRepository
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch

class FingerprintViewModel(private val repository:
FingerprintRepository) : ViewModel() {

    fun addFingerprint(onResult: (String) -> Unit) {
        viewModelScope.launch(Dispatchers.IO) {
            val result =
repository.sendHttpRequest("http://192.168.134.99/addFingerprint")
            onResult(result)
        }
    }

}
```

FingerprintViewModelFactory.kt

```
package com.example.apksmartdoor.model

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.apksmartdoor.repository.FingerprintRepository

class FingerprintViewModelFactory(private val repository:
FingerprintRepository) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if
(modelClass.isAssignableFrom(FingerprintViewModel::class.java)) {
            return FingerprintViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

MonitoringAdapter.kt

```
package com.example.apksmartdoor.model

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView
import com.example.apksmartdoor.R

class MonitoringAdapter : ListAdapter<MonitoringItem,
MonitoringAdapter.MonitoringViewHolder>(DiffCallback()) {

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): MonitoringViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        val view = inflater.inflate(R.layout.item_monitoring,
parent, false)
        return MonitoringViewHolder(view)
    }

    override fun onBindViewHolder(holder: MonitoringViewHolder,
position: Int) {
        val monitoringItem = getItem(position)
        holder.bind(monitoringItem)
    }

    class MonitoringViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val stateTextView: TextView =
itemView.findViewById(R.id.textState)
        private val timestampTextView: TextView =
itemView.findViewById(R.id.textTimestamp)

        fun bind(monitoringItem: MonitoringItem) {
            stateTextView.text = "State: ${monitoringItem.state}"
            timestampTextView.text = "Timestamp:
${monitoringItem.timestamp}"
        }
    }

    private class DiffCallback :
DiffUtil.ItemCallback<MonitoringItem>() {
        override fun areItemsTheSame(oldItem: MonitoringItem,
newItem: MonitoringItem): Boolean {
            return oldItem.id == newItem.id
        }

        override fun areContentsTheSame(oldItem: MonitoringItem,
newItem: MonitoringItem): Boolean {
            return oldItem == newItem
        }
```

MonitoringItem.kt

```
package com.example.apksmartdoor.model

data class MonitoringItem(
    val id: Int,
    val state: String,
    val timestamp: String
)
```

MonitoringViewModel.kt

```
package com.example.apksmartdoor.model

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.apksmartdoor.network.RetrofitClient
import kotlinx.coroutines.launch

class MonitoringViewModel : ViewModel() {

    private val _monitoringList =
MutableLiveData<List<MonitoringItem>>()
    val monitoringList: LiveData<List<MonitoringItem>> get() =
_monitoringList

    init {
        // Load monitoring data from the API
        loadMonitoringData()
    }

    private fun loadMonitoringData() {
        viewModelScope.launch {
            try {
                val apiClient = RetrofitClient.getInstance()
                val monitoringData = apiClient.getMonitoringData()
                _monitoringList.value = monitoringData
            } catch (e: Exception) {
                // Handle error
                e.printStackTrace()
            }
        }
    }
}
```

PayloadLogin.kt

```
package com.example.apksmartdoor.model

data class PayloadLogin (
        val nama : String,
        val username : String,
        val level_user : String,
        val id_user : String
    )
```

ResponseLogin.kt

```
package com.example.apksmartdoor.model

import com.google.gson.annotations.SerializedName

class ResponseLogin (
    var response : Boolean,
    var payload : PayloadLogin
    )

data class ApiResponse(
    @SerializedName("status")
    val status: String?,

    @SerializedName("message")
    val message: String?
)
```

User.kt

```
package com.example.apksmartdoor.model

import com.google.gson.annotations.SerializedName

class ResponseLogin (
    var response : Boolean,
    var payload : PayloadLogin
    )

data class ApiResponse(
    @SerializedName("status")
    val status: String?,

    @SerializedName("message")
    val message: String?
)
```

ApiClient.kt

```
package com.example.apksmartdoor.network

import com.example.apksmartdoor.model.MonitoringItem
import com.example.apksmartdoor.model.ResponseLogin
import retrofit2.http.Field
import retrofit2.http.FormUrlEncoded
import retrofit2.http.GET
import retrofit2.http.POST

interface ApiClient {

    @GET("monitoring.php")
    suspend fun getMonitoringData(): List<MonitoringItem>

    @FormUrlEncoded
    @POST("login/login.php")
    fun login(
        @Field("username") username : String,
        @Field("password") password : String
        ): retrofit2.Call<ResponseLogin>
}
```

ApiService.kt

```
package com.example.apksmartdoor.network

import com.example.apksmartdoor.model.ApiResponse
import com.example.apksmartdoor.model.User
import retrofit2.Call
import retrofit2.http.Field
import retrofit2.http.FormUrlEncoded
import retrofit2.http.GET
import retrofit2.http.POST
import retrofit2.http.Query

interface ApiService {
    @FormUrlEncoded
    @POST("signup.php")
    fun signup(
        @Field("name") name: String,
        @Field("username") username: String,
        @Field("password") password: String
    ): Call<ApiResponse>

    @GET("user.php")
    fun getUser(@Query("user_id") userId: Int): Call<User>
}
```

RetrofitClient.kt

```
package com.example.apksmartdoor.network

import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class RetrofitClient {

    companion object {
        private const val BASE_URL =
"https://scrptdoorlock.000webhostapp.com/"
        private var retrofit: Retrofit? = null

        fun getInstance(): ApiClient {
            if (retrofit == null) {
                retrofit = Retrofit.Builder()
                    .baseUrl(BASE_URL)

.addConverterFactory(GsonConverterFactory.create())
                    .build()
            }
            return retrofit!!.create(ApiClient::class.java)
        }
    }

    private fun getRetofitClient(): Retrofit {
        return Retrofit.Builder()
            .baseUrl("https://scrptdoorlock.000webhostapp.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    }

    fun getInstance(): ApiClient {
        return getRetofitClient().create(ApiClient::class.java)
    }
}
```

UserRepository.kt

```
package com.example.apksmartdoor.network

import com.example.apksmartdoor.model.User
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class UserRepository {

    private val retrofit = Retrofit.Builder()
        .baseUrl("https://scrptdoorlock.000webhostapp.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    private val userService =
retrofit.create(UserService::class.java)

    fun updateUser(user: User, onSuccess: () -> Unit, onError:
(String) -> Unit) {
        val call = userService.updateUser(user)
        call.enqueue(object : retrofit2.Callback<Void> {
            override fun onResponse(call: retrofit2.Call<Void>,
response: retrofit2.Response<Void>) {
                if (response.isSuccessful) {
                    onSuccess.invoke() // Panggil callback
onSuccess jika berhasil
                } else {
                    onError.invoke("Gagal menyimpan data ke
server") // Panggil callback onError jika gagal
                }
            }

            override fun onFailure(call: retrofit2.Call<Void>, t:
Throwable) {
                onError.invoke("Terjadi kesalahan jaringan") //
Panggil callback onError jika terjadi kesalahan jaringan
            }
        })
    }
}
```

UserService.kt

```
package com.example.apksmartdoor.network

import com.example.apksmartdoor.model.User
import retrofit2.Call
import retrofit2.http.Body
import retrofit2.http.PUT

interface UserService {
    @PUT("editUser.php")
    fun updateUser(@Body user: User): Call<Void>
}
```

AddFingerprintFragment.kt

```
package com.example.apksmartdoor

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.Toast
import android.widget.ViewFlipper
import androidx.lifecycle.ViewModelProvider
import com.bumptech.glide.Glide
import com.example.apksmartdoor.helper.PrefHelper
import com.example.apksmartdoor.model.FingerprintViewModel
import com.example.apksmartdoor.model.FingerprintViewModelFactory
import com.example.apksmartdoor.repository.FingerprintRepository

class AddFingerprintFragment : Fragment() {

    private lateinit var B_Next1: Button
    private lateinit var V_Flipper: ViewFlipper
    private lateinit var gifImageView: ImageView
    private lateinit var viewModel: FingerprintViewModel

    lateinit var prefHelper: PrefHelper

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view =
inflater.inflate(R.layout.fragment_add_fingerprint, container,
false)

        prefHelper = PrefHelper(requireActivity())

        V_Flipper = view.findViewById(R.id.V_Flipper)
```

AddUserFragment.kt

```
package com.example.apksmartdoor

import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import com.example.apksmartdoor.model.ApiResponse
import com.example.apksmartdoor.network.ApiService
import com.google.gson.GsonBuilder
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class AddUserFragment : Fragment() {

    private val BASE_URL =
"https://scrptdoorlock.000webhostapp.com/login/"

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_add_user,
container, false)

        val etName            : EditText =
view.findViewById(R.id.etName)
        val etUsername        : EditText =
view.findViewById(R.id.etUsername)
        val etPassword        : EditText =
view.findViewById(R.id.etPassword)
        val etConfirmPassword: EditText =
view.findViewById(R.id.etConfirmPassword)
        val btnSignUp         : Button   =
view.findViewById(R.id.btnRegister)

        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)

.addConverterFactory(GsonConverterFactory.create(GsonBuilder().set
Lenient().create()))
            .build()

        val service = retrofit.create(ApiService::class.java)

        btnSignUp.setOnClickListener {
            val name = etName.text.toString()
```

```
            val username = etUsername.text.toString()
            val password = etPassword.text.toString()
            val confirmPassword =
etConfirmPassword.text.toString()

            if (username.isEmpty()) {
                etUsername.setError("Username harus diisi")
                etUsername.requestFocus()
                return@setOnClickListener
            }

            if (password.isEmpty()) {
                etPassword.setError("Password harus diisi")
                etPassword.requestFocus()
                return@setOnClickListener
            }

            if (password.length < 8) {
                etPassword.setError("Password harus memiliki
minimal 8 karakter")
                etPassword.requestFocus()
                return@setOnClickListener
            }

            if (!password.matches(".*\\d.*".toRegex())) {
                etPassword.setError("Password harus memiliki
minimal satu angka")
                etPassword.requestFocus()
                return@setOnClickListener
            }

            if (confirmPassword.isEmpty()) {
                etConfirmPassword.setError("Konfirmasi Password
harus diisi")
                etConfirmPassword.requestFocus()
                return@setOnClickListener
            }

            if (password != confirmPassword) {
                etConfirmPassword.setError("Konfirmasi Password
tidak sesuai dengan Password")
                etConfirmPassword.requestFocus()
                return@setOnClickListener
            }

            val call = service.signup(name, username, password)

            call.enqueue(object : Callback<ApiResponse> {
                override fun onResponse(call: Call<ApiResponse>,
response: Response<ApiResponse>) {
                    if (response.isSuccessful) {
                        val apiResponse = response.body()
                        if (apiResponse != null &&
apiResponse.status == "success") {
                            Log.d("SignUp", "Success")
```

DoorFragment.kt

```
package com.example.apksmartdoor

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.Toast
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.apksmartdoor.helper.PrefHelper
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import java.io.DataOutputStream
import java.net.HttpURLConnection
import java.net.URL

class DoorFragment : Fragment() {

    lateinit var prefHelper: PrefHelper

    private lateinit var btnOpenDoor: Button
    private lateinit var btnCloseDoor: Button
    private lateinit var controlViewModel: ControlViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view = inflater.inflate(R.layout.fragment_door,
container, false)

        prefHelper = PrefHelper(requireActivity())

        controlViewModel =
ViewModelProvider(requireActivity())[ControlViewModel::class.java]

        btnCloseDoor = view.findViewById(R.id.btnOpen)
        btnOpenDoor = view.findViewById(R.id.btnClose)

        btnOpenDoor.setOnClickListener {
            controlViewModel.controlSolenoid("open")
        }

        btnCloseDoor.setOnClickListener {
            controlViewModel.controlSolenoid("close")
        }
```

```
        controlViewModel.response.observe(viewLifecycleOwner) {
response ->
            requireActivity().runOnUiThread {
                showToast(response)
            }
        }

        return view
    }

    private fun showToast(message: String) {
        Toast.makeText(requireContext(), message,
Toast.LENGTH_SHORT).show()
    }

    class ControlViewModel : ViewModel() {

        private val _response = MutableLiveData<String>()
        val response: LiveData<String> get() = _response

        fun controlSolenoid(state: String) {
            GlobalScope.launch(Dispatchers.IO) {
                val result = sendControlRequest(state)
                withContext(Dispatchers.Main) {
                    _response.value = result
                }
                val result2 = sendControlRequest2(state)
                withContext(Dispatchers.Main) {
                    _response.value = result2
                }
            }
        }

        private fun sendControlRequest(state: String): String {
            val url = URL("http://192.168.134.99/control")
            val urlConnection = url.openConnection() as
HttpURLConnection

            return try {
                urlConnection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded")
                urlConnection.requestMethod = "POST"
                urlConnection.doOutput = true

                val outputStream =
DataOutputStream(urlConnection.outputStream)
                val postData = "state=$state"
                outputStream.writeBytes(postData)
                outputStream.flush()
                outputStream.close()

                val responseCode = urlConnection.responseCode
                println("Response Code: $responseCode")

                if (urlConnection.responseCode ==
HttpURLConnection.HTTP_OK) {
```

```
urlConnection.inputStream.bufferedReader().readText()
                } else {
                    "HTTP Error: ${urlConnection.responseCode}"
                }
            } catch (e: Exception) {
                e.printStackTrace()
                "Error: ${e.message}"
            } finally {
                urlConnection.disconnect()
            }
        }

        private fun sendControlRequest2(state: String): String {
            val url =
URL("https://scrptdoorlock.000webhostapp.com/control.php")
            val urlConnection = url.openConnection() as
HttpURLConnection

            return try {
                urlConnection.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded")
                urlConnection.requestMethod = "POST"
                urlConnection.doOutput = true

                val outputStream =
DataOutputStream(urlConnection.outputStream)
                val postData = "state=$state"
                outputStream.writeBytes(postData)
                outputStream.flush()
                outputStream.close()

                val responseCode = urlConnection.responseCode
                println("Response Code: $responseCode")

                if (urlConnection.responseCode ==
HttpURLConnection.HTTP_OK) {

urlConnection.inputStream.bufferedReader().readText()
                } else {
                    "HTTP Error: ${urlConnection.responseCode}"
                }
            } catch (e: Exception) {
                e.printStackTrace()
                "Error: ${e.message}"
            } finally {
                urlConnection.disconnect()
            }
        }
    }
}
```

EditUserFragment.kt

```
package com.example.apksmartdoor

import android.graphics.Bitmap
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import com.example.apksmartdoor.helper.PrefHelper
import com.example.apksmartdoor.model.User
import com.example.apksmartdoor.network.ApiService
import com.example.apksmartdoor.network.UserRepository
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class EditUserFragment : Fragment() {

    private val BASE_URL =
"https://scrptdoorlock.000webhostapp.com/"
    private val userRepository = UserRepository()

    lateinit var prefHelper: PrefHelper

    private lateinit var profileImageView           : ImageView
    private lateinit var usernameTextView           : TextView
    private lateinit var editnameTextView           : TextView
    private lateinit var editpasswordTextView       : TextView
    private lateinit var editConfirmPasswordTextView: TextView
    private lateinit var btnSave                    : Button

    private val apiService: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(ApiService::class.java)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
```

```
): View? {
        // Inflate the layout for this fragment
        val view = inflater.inflate(R.layout.fragment_edit_user,
container, false)

        profileImageView            =
view.findViewById(R.id.profileImageView)
        usernameTextView            =
view.findViewById(R.id.usernameTextView)
        editnameTextView            =
view.findViewById(R.id.etName)
        editpasswordTextView        =
view.findViewById(R.id.etPassword)
        editConfirmPasswordTextView =
view.findViewById(R.id.etConfirmPassword)
        btnSave                     =
view.findViewById(R.id.btnSave)
        prefHelper                  =
PrefHelper(requireActivity())


        val userId = prefHelper.getUserId()

        btnSave.setOnClickListener {
            val name = editnameTextView.text.toString()
            val password = editpasswordTextView.text.toString()
            val confirmPassword =
editConfirmPasswordTextView.text.toString()

            when {
                name.isNotEmpty() && password.isEmpty() -> {
                    // Hanya Nama yang diinputkan
                    val user = User(userId, name, "")
                    userRepository.updateUser(user,
                        onSuccess = {
                            showToast("Nama pengguna berhasil
diperbarui")
                            // Tambahkan operasi lain yang perlu
dilakukan setelah keberhasilan
                        },
                        onError = { errorMessage ->
                            showToast(errorMessage)
                        }
                    )
                }

                name.isEmpty() && password.isNotEmpty() -> {
                    // Hanya Password yang diinputkan
                    val user = User(userId, "", password)
                    userRepository.updateUser(user,
                        onSuccess = {
                            showToast("Kata sandi pengguna
berhasil diperbarui")
                        },
                        onError = { errorMessage ->
                            showToast(errorMessage)


                    }
```

```
                    }
                }

                name.isNotEmpty() && password.isNotEmpty() &&
password == confirmPassword -> {
                    // Data pengguna valid, simpan ke database
                    val user = User(userId, name, password)
                    userRepository.updateUser(user,
                        onSuccess = {
                            showToast("Data pengguna berhasil
diperbarui")
                        },
                        onError = { errorMessage ->
                            showToast(errorMessage)
                        }
                    )
                }

                else -> {
                    // Tampilkan pesan kesalahan
                    showToast("Isian tidak valid")
                }
            }
        }

        apiService.getUser(userId).enqueue(object : Callback<User>
{
            override fun onResponse(call: Call<User>, response:
Response<User>) {
                Log.d("SettingsActivity", "onResponse")
                if (response.isSuccessful) {
                    val user = response.body()
                    if (user != null) {
                        // Set inisial pengguna sebagai teks pada
ImageView
                        val initial = getInitials(user.nama)

                        // Membuat bitmap dengan inisial
                        val bitmap = textAsBitmap(initial)

                        // Menetapkan bitmap sebagai gambar profil
                        profileImageView.setImageBitmap(bitmap)
                        Handler(Looper.getMainLooper()).post {

profileImageView.setContentDescription(getString(R.string.profile_
image_description, user.nama))
                            usernameTextView.text = user.nama
                        }
                    }
                }
                else {
                    // Log jika respons tidak berhasil
                    Log.e("SettingsActivity", "Failed to get user.
Code: ${response.code()}")
                }
            }
```

```
            override fun onFailure(call: Call<User>, t: Throwable)
{
                // Handle kesalahan ketika mengambil data pengguna
                Log.e("SettingsActivity", "Failed to get user.
Error: ${t.message}")
            }

        })

        return view
    }

    private fun getInitials(name: String): String {
        val names = name.split(" ")
        return if (names.size > 1) {
            val firstInitial = names[0].first()
            val lastInitial = names[1].first()
            "$firstInitial$lastInitial"
        } else {
            name.first().toString()
        }
    }

    private fun textAsBitmap(text: String): Bitmap {
        val paint = Paint(Paint.ANTI_ALIAS_FLAG)
        paint.color = Color.BLUE
        paint.textSize = 50f
        paint.textAlign = Paint.Align.CENTER

        val baseline = -paint.ascent() // ascent() is negative
        val width = (paint.measureText(text) + 0.5f).toInt()
        val height = (baseline + paint.descent() + 0.5f).toInt()

        val image = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888)
        val canvas = Canvas(image)
        canvas.drawText(text, (width / 2).toFloat(), baseline,
paint)

        return image
    }

    private fun showToast(message: String) {
        Toast.makeText(requireContext(), message,
Toast.LENGTH_SHORT).show()
    }
}
```

FingerprintFragment.kt

```
package com.example.apksmartdoor

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import android.widget.ViewFlipper
import com.example.apksmartdoor.helper.PrefHelper
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.net.HttpURLConnection
import java.net.URL

class FingerprintFragment : Fragment() {

    lateinit var prefHelper: PrefHelper
    private lateinit var V_Flipper: ViewFlipper
    private lateinit var editTextNumber: EditText
    private lateinit var sendButton: Button

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val view = inflater.inflate(R.layout.fragment_fingerprint,
container, false)

        prefHelper = PrefHelper(requireActivity())

        V_Flipper = view.findViewById(R.id.V_Flipper)
        editTextNumber = view.findViewById(R.id.editTextNumber)
        sendButton = view.findViewById(R.id.sendButton)

        val btnAddFingerprint: Button =
view.findViewById<Button>(R.id.btnAddFingerprint)
        val btnQWE: Button =
view.findViewById<Button>(R.id.btnQWE)

        btnAddFingerprint.setOnClickListener {
            V_Flipper.displayedChild = 1
        }

        btnQWE.setOnClickListener {
            val destinationFragment = AddFingerprintFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
```

```
transaction.commit()
        }

        sendButton.setOnClickListener {
            val enteredNumber = editTextNumber.text.toString()
            if (enteredNumber.isNotEmpty()) {
                val number = enteredNumber.toIntOrNull()
                if (number != null && number in 1..127) {
                    sendToESP(number)
                    V_Flipper.displayedChild = 1
                } else {
                    showToast("Enter a number between 1 and 127")
                }
            } else {
                showToast("Please enter a number")
            }
        }

        return view
    }

    private fun sendToESP(enteredNumber: Int) {
        val ipAddress = "192.168.134.99" // Ganti dengan IP
ESP8266
        val port = 80 // Ganti dengan port yang digunakan di
ESP8266

        GlobalScope.launch {
            val url =
URL("http://$ipAddress:$port/?data=$enteredNumber")
            with(url.openConnection() as HttpURLConnection) {
                requestMethod = "GET"
                inputStream.bufferedReader().use {
                    // Baca response dari ESP8266 jika diperlukan
                    val response = it.readText()
                    println("Response from ESP8266: $response")
                }
            }
        }
    }

    private fun showToast(message: String) {
        Toast.makeText(requireContext(), message,
Toast.LENGTH_SHORT).show()
    }

}
```

HomeFragment.kt

```
package com.example.apksmartdoor

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageButton

class HomeFragment : Fragment() {

    private lateinit var doorButton: Button
    private lateinit var settingButton: ImageButton
    private lateinit var monitoringButton: ImageButton

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_home,
container, false)

        doorButton = view.findViewById(R.id.B_Door)
        settingButton   = view.findViewById(R.id.I_Setting)
        monitoringButton   = view.findViewById(R.id.I_Monitoring)

        doorButton.setOnClickListener{
            val destinationFragment = DoorFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        settingButton.setOnClickListener{
            val destinationFragment = SettingFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        settingButton.setOnClickListener{
            val destinationFragment = MonitoringFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
```

LoginActivity.kt

```
package com.example.apksmartdoor

import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import android.view.View
import android.widget.Toast
import com.example.apksmartdoor.databinding.ActivityLoginBinding
import com.example.apksmartdoor.helper.PrefHelper
import com.example.apksmartdoor.model.ResponseLogin
import com.example.apksmartdoor.network.RetrofitClient
import retrofit2.Call
import retrofit2.Response

class LoginActivity : AppCompatActivity() {

    lateinit var prefHelper: PrefHelper

    private var binding: ActivityLoginBinding? = null
    private var user: String = ""
    private var pass: String = ""
    private var lvl: Int = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding!!.root)

        prefHelper = PrefHelper(this)

        binding!!.buttonLogin.setOnClickListener {
            user = binding!!.editTextUsername.text.toString()
            pass = binding!!.editTextPassword.text.toString()

            when {
                user == "" -> {
                    binding!!.editTextUsername.error = "Username
tidak boleh kosong"
                }

                pass == "" -> {
                    binding!!.editTextPassword.error = "Password
tidak boleh kosong"
                }

                else -> {
                    binding!!.loading.visibility = View.VISIBLE
                    getData()
                }
            }

    }
```

```
    }

    private fun getData() {
        val api = RetrofitClient().getInstance()
        api.login(user, pass).enqueue(object :
retrofit2.Callback<ResponseLogin> {
            override fun onResponse(call: Call<ResponseLogin>,
response: Response<ResponseLogin>) {
                if (response.isSuccessful) {
                    if (response.body()?.response == true) {
                        val userLevel =
response.body()?.payload?.level_user ?: "0"
                        val userId =
response.body()?.payload?.id_user ?: "0"
                        saveSession(user, pass, true,
userLevel.toInt(), userId.toInt())
                        binding!!.loading.visibility = View.GONE

                        if (userLevel == "1") {

startActivity(Intent(this@LoginActivity,
MainActivityAdmin::class.java))
                        } else {

startActivity(Intent(this@LoginActivity,
MainActivity::class.java))
                        }
                        finish()

                    } else {
                        binding!!.loading.visibility = View.GONE
                        Toast.makeText(
                            this@LoginActivity,
                            "Login gagal, Periksa kembali username
dan password",
                            Toast.LENGTH_LONG
                        ).show()
                    }
                } else {
                    Toast.makeText(
                        this@LoginActivity,
                        "Login gagal, Terjadi Kesalahan",
                        Toast.LENGTH_LONG
                    ).show()
                }
            }

            override fun onFailure(call: Call<ResponseLogin>, t:
Throwable) {
```

MainActivity.kt

```
package com.example.apksmartdoor

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.content.Intent
import android.view.MenuItem
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.widget.Toolbar
import androidx.core.view.GravityCompat
import androidx.drawerlayout.widget.DrawerLayout
import com.example.apksmartdoor.helper.PrefHelper
import com.google.android.material.navigation.NavigationView

class MainActivity : AppCompatActivity(),
NavigationView.OnNavigationItemSelectedListener {

    private lateinit var drawerLayout: DrawerLayout
    lateinit var prefHelper: PrefHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        prefHelper = PrefHelper(this)
        drawerLayout =
findViewById<DrawerLayout>(R.id.drawer_layout)

        val toolbar = findViewById<Toolbar>(R.id.toolbar)

        val navigationView =
findViewById<NavigationView>(R.id.nav_view)
        navigationView.setNavigationItemSelectedListener(this)

        val toggle = ActionBarDrawerToggle(
            this,
            drawerLayout,
            toolbar,
            R.string.open_nav,
            R.string.close_nav
        )
        drawerLayout.addDrawerListener(toggle)
        toggle.syncState()

        if (savedInstanceState == null) {
            supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container,
HomeFragment()).commit()
            navigationView.setCheckedItem(R.id.nav_home)
        }
    }

    override fun onNavigationItemSelected(item: MenuItem): Boolean
{
        when (item.itemId) {
```

```
| R.id.nav_home -> {
                supportFragmentManager.beginTransaction()
                    .replace(R.id.fragment_container,
HomeFragment()).commit()
            }

            R.id.nav_logout -> {
                // Clear user authentication information or
session (you need to implement this method)
                logoutUser()

                // Navigate to the login screen
                val intent = Intent(this,
LoginActivity::class.java)
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP or
Intent.FLAG_ACTIVITY_NEW_TASK)
                startActivity(intent)
                finish()
            }
        }
        drawerLayout.closeDrawer(GravityCompat.START)
        return true

    }

    private fun logoutUser() {
        // Implement your logic to clear user authentication
information or session
        // For example, you can use PrefHelper to clear stored
data
        prefHelper.clearSession()
    }
}
```

MainActivityAdmin.kt

```
package com.example.apksmartdoor

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.MenuItem
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.widget.Toolbar
import androidx.core.view.GravityCompat
import androidx.drawerlayout.widget.DrawerLayout
import com.example.apksmartdoor.helper.PrefHelper
import com.google.android.material.navigation.NavigationView

class MainActivityAdmin : AppCompatActivity(),
NavigationView.OnNavigationItemSelectedListener {

    private lateinit var drawerLayout: DrawerLayout
    lateinit var prefHelper: PrefHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main_admin)

        prefHelper = PrefHelper(this)
        drawerLayout =
findViewById<DrawerLayout>(R.id.drawer_layout)

        val toolbar = findViewById<Toolbar>(R.id.toolbar)

        val navigationView =
findViewById<NavigationView>(R.id.nav_view)
        navigationView.setNavigationItemSelectedListener(this)

        val toggle = ActionBarDrawerToggle(
            this,
            drawerLayout,
            toolbar,
            R.string.open_nav,
            R.string.close_nav
        )
        drawerLayout.addDrawerListener(toggle)
        toggle.syncState()

        if (savedInstanceState == null) {
            supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container,
MainAdminFragment()).commit()
            navigationView.setCheckedItem(R.id.nav_home)
        }
    }

    override fun onNavigationItemSelected(item: MenuItem): Boolean
{
        when (item.itemId) {
```

```
 R.id.nav_home -> {
                supportFragmentManager.beginTransaction()
                    .replace(R.id.fragment_container,
MainAdminFragment()).commit()
            }

            R.id.nav_logout -> {
                // Clear user authentication information or
session (you need to implement this method)
                logoutUser()

                // Navigate to the login screen
                val intent = Intent(this,
LoginActivity::class.java)
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP or
Intent.FLAG_ACTIVITY_NEW_TASK)
                startActivity(intent)
                finish()
            }
        }
        drawerLayout.closeDrawer(GravityCompat.START)
        return true
    }

    private fun logoutUser() {
        // Implement your logic to clear user authentication
information or session
        // For example, you can use PrefHelper to clear stored
data
        prefHelper.clearSession()
    }
}
```

MainAdminFragment.kt

```
package com.example.apksmartdoor

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageButton

class MainAdminFragment : Fragment() {

    private lateinit var doorButton: Button
    private lateinit var fingerButton: ImageButton
    private lateinit var adduserButton: ImageButton
    private lateinit var settingButton: ImageButton
    private lateinit var monitoringButton: ImageButton

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_main_admin,
container, false)

        doorButton       = view.findViewById(R.id.B_Door)
        fingerButton     = view.findViewById(R.id.I_Finger)
        adduserButton    = view.findViewById(R.id.I_addUser)
        settingButton    = view.findViewById(R.id.I_Setting)
        monitoringButton = view.findViewById(R.id.I_Monitoring)

        doorButton.setOnClickListener{
            val destinationFragment = DoorFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        fingerButton.setOnClickListener{
            val destinationFragment = FingerprintFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        adduserButton.setOnClickListener{
            val destinationFragment = AddUserFragment()
            val transaction =
```

```
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        settingButton.setOnClickListener{
            val destinationFragment = SettingFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        monitoringButton.setOnClickListener{
            val destinationFragment = MonitoringFragment()
            val transaction =
requireActivity().supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container,
destinationFragment)
            transaction.addToBackStack(null)
            transaction.commit()
        }

        return view
    }
}
```

MonitoringFragment.kt

```
package com.example.apksmartdoor

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.lifecycle.ViewModelProvider
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.apksmartdoor.model.MonitoringAdapter
import com.example.apksmartdoor.model.MonitoringViewModel

class MonitoringFragment : Fragment() {

    private lateinit var monitoringViewModel: MonitoringViewModel
    private lateinit var monitoringAdapter: MonitoringAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_monitoring,
container, false)

        monitoringViewModel =
ViewModelProvider(this)[MonitoringViewModel::class.java]

        val recyclerView: RecyclerView =
view.findViewById(R.id.recyclerView)
        monitoringAdapter = MonitoringAdapter()

        recyclerView.adapter = monitoringAdapter
        recyclerView.layoutManager =
LinearLayoutManager(requireContext())


monitoringViewModel.monitoringList.observe(viewLifecycleOwner) {
monitoringList ->
            monitoringAdapter.submitList(monitoringList)
        }

        return view
    }
}
```

SettingFragment.kt

```
package com.example.apksmartdoor

import android.graphics.Bitmap
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.ListView
import android.widget.TextView
import androidx.fragment.app.Fragment
import com.example.apksmartdoor.helper.PrefHelper
import com.example.apksmartdoor.model.User
import com.example.apksmartdoor.network.ApiService
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class SettingFragment : Fragment() {

    lateinit var prefHelper: PrefHelper

    private lateinit var profileImageView: ImageView
    private lateinit var usernameTextView: TextView
    private lateinit var listView: ListView
    private lateinit var linearLayout: LinearLayout


    private val apiService: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl("https://scrptdoorlock.000webhostapp.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(ApiService::class.java)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_setting,
container, false)
```

```
        profileImageView    =
view.findViewById(R.id.profileImageView)
        usernameTextView    =
view.findViewById(R.id.usernameTextView)
        listView            = view.findViewById(R.id.listView)
        linearLayout        =
view.findViewById(R.id.LinearEditUser)
        prefHelper          = PrefHelper(requireActivity())

        val loggedInUserId  = prefHelper.getUserId()
        val items           = arrayOf("Kontak Kami", "Tentang
Aplikasi")

        val adapter = ArrayAdapter<String>(
            requireContext(),
            R.layout.list_item_example,
            R.id.textView,
            items
        )

        listView.adapter = adapter

        listView.onItemClickListener =
AdapterView.OnItemClickListener { _, _, position, _ ->
            val selectedItem = items[position]
            startIntent(selectedItem)
        }

        linearLayout.setOnClickListener {
            onLinearLayoutClick(it)
        }

        apiService.getUser(loggedInUserId).enqueue(object :
Callback<User> {
            override fun onResponse(call: Call<User>, response:
Response<User>) {
                Log.d("SettingsActivity", "onResponse")
                if (response.isSuccessful) {
                    val user = response.body()
                    if (user != null) {
                        val initial = getInitials(user.nama)
                        val bitmap = textAsBitmap(initial)
                        profileImageView.setImageBitmap(bitmap)
                        Handler(Looper.getMainLooper()).post {

profileImageView.setContentDescription(getString(R.string.profile_
image_description, user.nama))
                            usernameTextView.text = user.nama
                        }
                    }
                }
                else {
                    Log.e("SettingsActivity", "Failed to get user.
Code: ${response.code()}")
                }
            }
```

```
            override fun onFailure(call: Call<User>, t: Throwable)
{
                Log.e("SettingsActivity", "Failed to get user.
Error: ${t.message}")
            }

        })
        return view
    }

    private fun getInitials(name: String): String {
        val names = name.split(" ")
        return if (names.size > 1) {
            val firstInitial = names[0].first()
            val lastInitial = names[1].first()
            "$firstInitial$lastInitial"
        } else {
            name.first().toString()
        }
    }

    private fun onLinearLayoutClick(view: View) {
        val fragmentManager =
requireActivity().supportFragmentManager
        val fragmentTransaction =
fragmentManager.beginTransaction()
        val tujuanFragment = EditUserFragment()

        fragmentTransaction.replace(R.id.fragment_container,
tujuanFragment)
        fragmentTransaction.addToBackStack(null)
        fragmentTransaction.commit()
    }

    private fun textAsBitmap(text: String): Bitmap {
        val paint = Paint(Paint.ANTI_ALIAS_FLAG)
        paint.color = Color.BLUE
        paint.textSize = 50f
        paint.textAlign = Paint.Align.CENTER

        val baseline = -paint.ascent()
        val width = (paint.measureText(text) + 0.5f).toInt()
        val height = (baseline + paint.descent() + 0.5f).toInt()

        val image = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888)
        val canvas = Canvas(image)
        canvas.drawText(text, (width / 2).toFloat(), baseline,
paint)

        return image
    }

    private fun startIntent(selectedItem: String) {
```

SplashScreenActivity.kt

```
package com.example.apksmartdoor

import android.app.ActivityOptions
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View

class SplashScreenActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash_screen)

        val s_screen: View = findViewById(R.id.s_screen)
        s_screen.alpha = 0f

s_screen.animate().setDuration(1500).alpha(1f).withEndAction {
            val i = Intent(this, LoginActivity::class.java)

            // Use ActivityOptions for custom animation
            val options = ActivityOptions.makeCustomAnimation(
                this,
                android.R.anim.fade_in,
                android.R.anim.fade_out
            )

            // Start activity with options
            startActivity(i, options.toBundle())

            // Finish the current activity
            finish()
        }
    }
}
```

LAMPIRAN 3: DOKUMENTASI


RUANG
BK
KEMENTERIAN AGAMA
ayo...
DUKUNG
KAMI
DALAM PEMBANGUNAN
ZONA INTEGRITAS
MENUJU WILAYAH
BEBAS DARI KORUPSI
VISI, MISI DAN TUJUAN
MIN 2 BANGKA


RUANG
BK
VISI, MISI DAN TUJUAN
MIN 2 BANGKA