BUKU TUGAS AKHIR
CAPSTONE DESIGN

PERANCANGAN *SMART DORM KEY* BERBASIS *VOICE RECOGNITION DENGAN TAMBAHAN SENSOR FINGERPRINT* SEBAGAI VERIFIKASI DUA LANGKAH UNTUK MENINGKATKAN KEAMANAN

Oleh :

Muhammad Trisucipto /1101210011

Muhamad Hazbi Ashiddiqi /1101210070

DTM Faiq Zariaqwila /1101213370

PRODI S1 TEKNIK TELEKOMUNIKASI

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

BANDUNG

2025

LEMBAR PENGESAHAN
BUKU CAPSTONE DESIGN

PERANCANGAN *SMART DORM KEY* BERBASIS *VOICE RECOGNITION DENGAN TAMBAHAN SENSOR FINGERPRINT* SEBAGAI VERIFIKASI DUA LANGKAH UNTUK MENINGKATKAN KEAMANAN

(DESIGNING A VOICE RECOGNITION-BASED SMART DORM KEY WITH ADDITIONAL FINGERPRINT SENSOR AS A TWO-STEP VERIFICATION TO IMPROVE SECURITY)

Telah disetujui dan disahkan sebagai bagian dari Capstone Design

Program S1 Teknik Telekomunikasi

Fakultas Teknik Elektro

Universitas Telkom

Bandung

Disusun oleh:

Muhammad Trisucipto /1101210011

Muhamad Hazbi Ashiddiqi /1101210070

DTM Faiq Zariaqwila /1101213370

Bandung, Juli 2025

Menyetujui,

Pembimbing 1	Pembimbing 2
	
Dr. Rita Purnamasari, S.T., M.T.	Efri Suhartono, S.T., M.T.
NIP. 10860010	NIP. 99730010

LEMBAR PERNYATAAN ORISINALITAS

Saya, yang bertanda tangan di bawah ini

Nama : Muhammad Trisucipto

NIM : 1101210011

Alamat : Kost Megaria, Jl. Hj.Mansyur No.1, Bojongsoang

No. Telepon : 082349202215

Email : mtrisucipto204@gmail.com

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

PERANCANGAN *SMART DORM KEY* BERBASIS *VOICE RECOGNITION DENGAN TAMBAHAN SENSOR FINGERPRINT* SEBAGAI VERIFIKASI DUA LANGKAH UNTUK MENINGKATKAN KEAMANAN

(DESIGNING A VOICE RECOGNITION-BASED SMART DORM KEY WITH ADDITIONAL FINGERPRINT SENSOR AS A TWO-STEP VERIFICATION TO IMPROVE SECURITY)

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 11 Juli 2025

Muhammad Trisucipto

1101210011

LEMBAR PERNYATAAN ORISINALITAS

Saya, yang bertanda tangan di bawah ini

Nama : Muhamad Hazbi Ashiddiqi

NIM : 1101210070

Alamat : Bandung, Jl. Batununggal Gg. Batu Berkah No.20

No. Telepon : 081214116801

Email : hazbiashiddiqi@student.telkomuniversity.ac.id

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

PERANCANGAN *SMART DORM KEY* BERBASIS *VOICE RECOGNITION DENGAN TAMBAHAN SENSOR FINGERPRINT* SEBAGAI VERIFIKASI DUA LANGKAH UNTUK MENINGKATKAN KEAMANAN

(DESIGNING A VOICE RECOGNITION-BASED SMART DORM KEY WITH ADDITIONAL FINGERPRINT SENSOR AS A TWO-STEP VERIFICATION TO IMPROVE SECURITY)

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 11 Juli 2025

Muhamad Hazbi Ashiddiqi
1101210070

LEMBAR PERNYATAAN ORISINALITAS

Saya, yang bertanda tangan di bawah ini

Nama : DTM Faiq Zariaqwila

NIM : 1101213370

Alamat : Jln. Mangga Dua STT.Telkom Sukapura

No. Telepon : 085277920463

Email : faiqaqwila04@gmail.com

Menyatakan bahwa Buku Capstone Design ini merupakan karya orisinal saya sendiri bersama dengan kelompok Capstone Design saya, dengan judul:

PERANCANGAN *SMART DORM KEY* BERBASIS *VOICE RECOGNITION DENGAN TAMBAHAN SENSOR FINGERPRINT* SEBAGAI VERIFIKASI DUA LANGKAH UNTUK MENINGKATKAN KEAMANAN

(DESIGNING A VOICE RECOGNITION-BASED SMART DORM KEY WITH ADDITIONAL FINGERPRINT SENSOR AS A TWO-STEP VERIFICATION TO IMPROVE SECURITY)

Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila dikemudian ditemukan adanya pelanggaran terhadap kejujuran akademik atau etika keilmuan dalam karya ini, atau ditemukan bukti yang menunjukkan ketidak aslian karya ini.

Bandung, 11 Juli 2025

DTM Faiq Zariaqwila
1101213370

ABSTRAK

Keamanan asrama di Telkom University saat ini masih mengandalkan sistem konvensional seperti kunci manual dan *logbook*, yang dinilai kurang memadai, rentan, dan kurang aman. Keterbatasan ini memungkinkan akses yang tidak sah dan meningkatkan risiko kehilangan barang berharga mahasiswa. Untuk mengatasi permasalahan tersebut, dirancanglah sebuah "*Smart Dorm Key*" berbasis *Internet of Things* (IoT) dengan sistem verifikasi dua langkah menggunakan sidik jari (*fingerprint*) dan pengenalan suara (*voice recognition*) guna meningkatkan keamanan.

Sistem ini dikembangkan dengan mikrokontroler ESP32 sebagai pusat kendali. Proses dimulai dengan pengguna mendaftarkan sidik jari dan frasa sandi suara melalui aplikasi seluler. Untuk mengakses kamar, pengguna pertama-tama melakukan pemindaian suara pada aplikasi yang tersedia. Jika suara terverifikasi, yang dikonfirmasi melalui layar LCD dan *buzzer*, pengguna kemudian melakukan pemindaian sidik jari melalui sensor *fingerprint*. Aplikasi ini menggunakan *machine learning* dengan metode *Mel-Frequency Cepstral Coefficients* (MFCC) untuk pemrosesan ekstraksi suara dan menggunakan model *Convolutional Neural Networks* (CNN) untuk pengenalan suara. Jika kedua verifikasi berhasil, ESP32 akan mengaktifkan *solenoid door lock* untuk membuka pintu. Sistem juga mencakup fitur log aktivitas secara *real-time* yang tersimpan di *database Firebase*, tombol keluar tanpa sentuh (*No Touch Exit Sensor*), dan tombol fisik untuk manajemen data sidik jari.

Kata kunci : CNN, ESP32, *Fingerprint*, *Internet of Things*, Keamanan Asrama, MFCC, *Smart Dorm Key*, Verifikasi Dua Langkah, *Voice Recognition.*

ABSTRACT

Current dormitory security at Telkom University still relies on conventional systems like manual keys and logbooks, which are considered inadequate, vulnerable, and insecure. These limitations enable unauthorized access and increase the risk of students losing valuable belongings. To address these issues, a "Smart Dorm Key" based on the Internet of Things (IoT) has been designed. It features a two-step verification system using fingerprint and voice recognition to enhance security.

This system is developed with an ESP32 microcontroller as the central control unit. The process begins with the user registering their fingerprint and voice passphrase through a mobile application. To access the room, the user first performs a voice scan on the available application. If the voice is verified, which is confirmed via an LCD screen and a buzzer, the user then scans their fingerprint using the fingerprint sensor. The application utilizes machine learning with the Mel-Frequency Cepstral Coefficients (MFCC) method for voice feature extraction and employs a Convolutional Neural Networks (CNN) model for voice recognition. If both verifications are successful, the ESP32 will activate the solenoid door lock to open the door. The system also includes a real-time activity log feature stored in the Firebase database, a no-touch exit sensor, and a physical button for fingerprint data management.

Keywords : CNN, Dormitory Security, ESP32, Fingerprint, Internet of Things, MFCC, Smart Dorm Key, Two-Step Verification, Voice Recognition.

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT atas rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan tugas akhir ini dengan judul "Perancangan *Smart Dorm Key* Berbasis *Voice Recognition* dengan tambahan sensor *fingerprint* Sebagai Verifikasi Dua Langkah untuk Meningkatkan Keamanan". Tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Telekomunikasi, Fakultas Teknik Elektro, Telkom University.

Asrama sebagai tempat tinggal mahasiswa memiliki peran vital dalam mendukung kegiatan akademik dan non-akademik. Keamanan asrama menjadi prioritas utama demi kenyamanan dan perlindungan aset pribadi mahasiswa. Namun, sistem keamanan konvensional yang masih banyak digunakan saat ini seringkali menunjukkan keterbatasan, baik dari segi efektivitas maupun keandalannya. Kerap terjadi insiden kehilangan barang dan akses tidak sah yang mengindikasikan perlunya inovasi dalam sistem keamanan asrama.

Melalui tugas akhir ini, penulis berupaya menghadirkan solusi konkret berupa sistem kunci pintar asrama yang mengintegrasikan teknologi *Internet of Things* (IoT) dengan verifikasi dua langkah menggunakan sidik jari dan pengenalan suara. Harapannya, sistem ini dapat menjawab tantangan keamanan yang ada, memberikan rasa aman yang lebih baik bagi penghuni asrama, serta menjadi fondasi bagi pengembangan sistem keamanan yang lebih canggih di masa depan.

Penyusunan tugas akhir ini tidak lepas dari bantuan dan bimbingan berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dalam menyelesaikan tugas akhir ini, termasuk dosen pembimbing, teman – teman, dan keluarga yang selalu memberikan dukungan dan doa.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan di kemudian hari. Semoga tugas akhir ini dapat memberikan kontribusi positif bagi pengembangan ilmu pengetahuan dan teknologi, khususnya di bidang keamanan berbasis IoT.

UCAPAN TERIMAKASIH

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan Tugas Akhir ini. Terselesaikannya Tugas Akhir ini tentu tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih yang tulus kepada:

1. Puji syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat, anugerah, dan kelancaran yang diberikan, sehingga penulisan tugas akhir ini dapat diselesaikan dengan baik.
2. Kedua orang tua tercinta, atas doa, kasih sayang, dan dukungan tanpa henti yang selalu kalian curahkan. Keberhasilan ini adalah cerminan dari pengorbanan, cinta, dan motivasi yang tak pernah putus dari kalian setiap harinya.
3. Ibu Dr. Rita Purnamasari, S.T., M.T. sebagai dosen pembimbing 1 yang dengan penuh kesabaran dan senantiasa memberikan bimbingan, saran, serta jalan keluar atas setiap kendala yang mengantarkan penulis menyelesaikan tugas akhir ini sesuai jadwal.
4. Bapak Efri Suhartono, S.T., M.T., selaku Dosen Pembimbing 2 berkat bimbingan dan solusi yang selalu diberikan sehingga tugas akhir ini terselesaikan sesuai target waktu.
5. Para sahabat (PGA, Megaria, dan Ukhuwah), pasangan, dan rekan terdekat, terima kasih atas dukungan, dorongan, bantuan, dan kesempatan untuk saling berbagi pengetahuan.
6. Terima kasih tak terhingga untuk *playlist* lagu yang menjadi *soundtrack* wajib selama mengerjakan tugas akhir. Tanpa musik kalian, mungkin penulis hanya akan berteman dengan kafein dan tumpukan revisi. Kalian adalah penyelamat kewarasan di tengah tuntutan *deadline*!.

Ucapan terima kasih ini mungkin tidak cukup untuk menggambarkan seberapa besar penghargaan saya atas semua bantuan yang telah diberikan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN

KTM	: Kartu Tanda Mahasiswa
QoS	: Quality of Service
RFID	: Radio Frequency Identification
SNI	: Standar Nasional Indonesia
UI/UX	: User Interface/User Experience)
UPS	: Uninterruptible Power Supply
MFCC	: Mel-Frequency Cepstral Coefficients
NFC	: Near Field Communication
NIM	: Nomor Induk Mahasiswa
PIN	: Personal Identification Number
HMM	: Hidden Markov Model
I2C	: Inter-Integrated Circuit
IDE	: Integrated Development Environment
IoT	: Internet of Things
JVM	: Java Virtual Machine
CNN	: Convolutional Neural Networks
CCTV	: Closed Circuit Television
DSP	: Digital Signal Processor
UART	: Universal Asynchronous Receiver-Transmitter
WIFI	: Wireless Fidelity
LED	: Light Emitting Diode
LCD`	: Liquid Crystal Display
REST API	: Representational State Transfer Application Programming Interface.
DCT	: Discrete Cosine Transform
WAV	: Waveform Audio Format

BAB 1
USULAN GAGASAN

1.1 Deskripsi Umum Masalah

1.1.1 Latar Belakang Masalah

Asrama Telkom University merupakan salah satu sarana kampus yang dibangun sebagai tempat tinggal bagi mahasiswa/i baru pada satu tahun pertama mereka di Telkom University. Asrama merupakan tempat yang sempurna bagi mahasiswa/i baru untuk belajar banyak hal seperti, toleransi dengan sesama, kerjasama dengan tim, kekeluargaan dan banyak hal bermanfaat lainnya. Semua kegiatan yang dilakukan selama berada di asrama akan dibimbing oleh kakak asrama yang disebut dengan *Senior Resident* serta *Helpdesk* yang berada di setiap gedung asrama untuk menjaga rasa keamanan serta kenyamanan bagi seluruh mahasiswa/i yang berada di gedung asrama [1]. Keamanan menjadi salah satu bidang yang memegang peran penting dalam kehidupan manusia. Dengan adanya keamanan ini suatu gedung atau ruangan akan semakin terlindungi dan terhindar dari upaya pencurian barang berharga.

Untuk saat ini, solusi keamanan yang digunakan oleh asrama Telkom University masih terbilang tradisional, belum memadai, rentan, dan kurang aman. Karena keamanan tradisional yang digunakan hanya sebatas kunci manual dan *Closed Circuit Television (CCTV)* sedangkan untuk tamu yang hendak memasuki asrama diharuskan untuk mengisi *log book* manual di meja *helpdesk* serta melakukan penitipan Kartu Tanda Mahasiswa (KTM) jika ingin masuk ke dalam asrama sebagai pengunjung. Dengan keamanan yang terbilang masih cukup terbatas ini sangat memungkinkan bagi mahasiswa ataupun orang lain yang tidak seharusnya berada di asrama dapat dengan mudah melakukan akses untuk keluar masuk ke gedung asrama. Dengan sistem keamanan yang seperti ini, sangat wajar bagi mahasiswa jika harus menaruh perhatian lebih terkait keamanan di ruangan asrama mengingat sangat banyak barang berharga yang tersimpan di ruangan asrama seperti, *laptop, handphone,* dan berbagai macam barang berharga lainnya.

Seiring berkembangnya zaman, teknologi berkembang sangat pesat dan menjadi peran penting bagi kehidupan manusia sehari-hari. Dengan adanya teknologi tentu saja dapat dimanfaatkan untuk mempermudah pekerjaan pekerjaan manusia. *Smart Home* merupakan teknologi zaman sekarang dengan kepintaran teknologi yang semakin maju yang bertujuan untuk mempermudah pekerjaan manusia, kunci pintu otomatis adalah salah satunya [2]. Salah satu teknologi yang dilakukan dalam meningkatkan keamanannya yaitu dengan cara

menggunakan sistem *smart door* dengan beberapa metode seperti *Radio Frequency Identification (RFID*), *password*, kode *Personal Identification Number (PIN)*, dan *fingerprint*. Namun dari beberapa metode ini jika hanya menggunakan salah satu dari metode tersebut belum terlalu aman untuk digunakan, karena masih dapat dimanipulasi oleh orang lain. Oleh sebab itu sangat diperlukannya inovasi baru dengan cara menggabungkan dua atau lebih metode keamanan supaya dapat meningkatkan keamanan gedung atau ruangan, salah satunya yaitu dengan cara verifikasi dua langkah.

1.2 Analisis Masalah

Terdapat beberapa permasalahan yang dapat terjadi selama penelitian berlangsung. Untuk mengatasi permasalahan ini dapat dilakukan dengan menganalisis dari beberapa aspek terkait, yaitu :

1.2.1 Aspek Ekonomi

Dengan rencana pemasangan *smart dorm key* di asrama ini tentu membutuhkan biaya yang lebih besar di awal karena adanya proses perancangan dan pembelian alat untuk setiap kamar yang berada di asrama. Tetapi ketika pemasangan alat sudah berjalan, tentu kita bisa mengurangi biaya operasional yang digunakan untuk asrama karena kita tidak memerlukan *helpdesk* untuk mencatat tamu ataupun orang yang hendak masuk ke asrama melalui *logbook.* Dengan meningkatnya tingkat keamanan pada asrama ini tentunya akan ada kemungkinan bahwa harga asrama untuk tahun-tahun berikutnya meningkat dari yang sudah ada.

1.2.2 Aspek Teknologi

Dengan penggunaan kunci manual atau konvensional pada asrama masih memiliki kekurangan dari segi keamanan dan kenyamanan. Kunci konvesional ini dapat hilang, dicuri, atau diduplikasi dengan mudah, hal ini dapat berpotensi membahayakan keamanan gedung atau ruangan dan dapat terjadi kehilangan barang berharga bagi penghuni asrama. Selain itu juga, sistem dari kunci konvesional ini tidak memiliki catatan akses yang dapat dipantau atau dilacak, hal ini dapat menyulitkan pengurus asrama untuk memantau keluar masuknya penghuni asrama.

1.2.3 Aspek Efisiensi dan Kenyamanan

Penggunaan sistem keamanan manual seperti melakukan pencatatan *logbook* di *helpdesk* dan CCTV yang diawasi oleh petugas asrama sangat tidak efisien karena memerlukan banyak tenaga kerja dan potensi kelalaian yang dilakukan oleh petugas. Dengan adanya kemungkinan kelalaian yang dilakukan oleh petugas tentunya akan membuat mahasiswa menjadi kurang nyaman terhadap hal ini.

1.3 Analisis Solusi yang Ada

Ada beberapa solusi yang sudah ada pada asrama Telkom University untuk sistem keamanan, yaitu dengan cara menggunakan *logbook* manual dan pengawasan dari setiap *senior resident* yang ada pada setiap gedung asrama. Untuk *logbook* manual ini, dilakukan dengan cara mencatat akses keluar masuk dari setiap pengunjung. *Logbook* manual ini memiliki kekurangan yang sangat jelas, dimana *logbook* manual ini sangat memungkinkan untuk terjadinya kesalahan yang dilakukan oleh penjaga. Karena *logbook* manual ini tidak dapat mengenali siapa saja yang memiliki kunci akses untuk asrama. Untuk tugas *senior resident* ini bertanggung jawab untuk menjaga keamanan dari gedung tempat mereka tinggal juga bertanggung jawab untuk membantu mahasiswa ketika memerlukan bantuan dan *senior resident* juga harus bisa menjadi panutan bagi mahasiswa yang tinggal di asrama. Kekurangan dari *senior resident* ini adalah untuk menjadi *senior resident* ini tidak diberikan pelatihan khusus dan *senior resident* ini juga merupakan mahasiswa sehingga *senior resident* ini tidak bisa hadir setiap waktu dan memiliki kekurangan dalam hal keamanan yang sangat jelas ketika mereka tidak sedang berada di asrama.

Solusi lain yang telah ada terkait *smart dorm key*, salah satunya yaitu menggunakan metode *face recognition* dan sensor ultrasonik. Metode *face recognition* dan sensor ultrasonik. Pengenalan wajah ini menggunakan metode HOG dan *Haar Cascade*, metode ini terdapat keunggulan dan kekurangan yang harus diperhatikan. Keunggulan *face recognition* dan sensor ultrasonik yaitu metode ini sangat mudah untuk digunakan jika dalam kondisi normal, untuk membuka kunci hanya perlu memperlihatkan wajah untuk dideteksi, memiliki keamanan yang baik karena tidak semua orang mendapatkan akses keluar-masuk dan hanya orang yang terdaftar yang memiliki akses. Namun, metode *face recognition* dan sensor ultrasonik ini juga memiliki kekurangan. Salah satu kekurangan dari metode ini yaitu gagalnya sistem dalam mendeteksi untuk mengenali wajah yang disebabkan oleh kondisi pencahayaan yang redup, wajah yang tertutup rambut, dan adanya penggunaan aksesoris yang dapat mengurangi akurasi pengenalan wajah dan hambatan visual yang dapat mengubah tampilan wajah yang dapat menambah elemen baru yang menutupi wajah [3].

Ada juga solusi yang sudah tersedia yaitu *Smart lock door* dengan menggunakan metode sensor suara yang lebih tepatnya menggunakan sensor *piezzo* untuk dapat mendeteksi pola ketukan yang digunakan sebagai kunci untuk pintu. Pada metode ini juga menggunakan Blynk sebagai aplikasi yang akan dihubungkan dengan rangkaian tersebut agar dapat mempermudah pengaksesan pengunci pintu pada ponsel sebagai alternatif lain apabila sensor suara dirasa

sedang bermasalah. Namun pada metode ini terdapat banyak kekurangan, yaitu penggunaan dari sensor *piezzo* ini sangat tidak efektif dikarenakan sensor ini sangat sensitif terhadap getaran suara sehingga ketika digunakan sebagai media *input* untuk pengunci pintu sering kali terjadi kegagalan. Oleh karena itu, metode ini masih kurang efektif untuk digunakan [4]

1.4 Tujuan Tugas Akhir

Adapun tujuan dari penyusunan *Capstone Design* sebagai berikut :

- Sistem *Smart Dorm Key* ini dirancang dan diimplementasikan secara khusus untuk lingkungan asrama Telkom University. Konfigurasi izin akses aplikasi terbatas hanya untuk mahasiswa/i Telkom University yang menggunakan *E-mail* sebagai autentikasi.
- Merancang dan membangun sebuah prototipe sistem keamanan pintu asrama, yaitu *Smart Dorm Key*, dengan menggunakan mikrokontroler ESP32 sebagai pusat kendali.
- Mengimplementasikan sistem verifikasi dua langkah (*two-step verification*) yang mengintegrasikan teknologi sidik jari (*fingerprint*) dan pengenalan suara (*voice recognition*) untuk meningkatkan keamanan akses.
- Mengembangkan model machine learning menggunakan metode *Mel-Frequency Cepstral Coefficients* (MFCC) untuk ekstraksi fitur suara dan arsitektur *Convolutional Neural Network* (CNN) untuk proses klasifikasi dan pengenalan suara pengguna.
- Menciptakan aplikasi seluler yang berfungsi sebagai antarmuka bagi pengguna untuk melakukan verifikasi suara serta bagi admin untuk memantau *log* aktivitas masuk secara *real-time*.
- Mengintegrasikan seluruh sistem dengan *database Firebase* untuk penyimpanan data pengguna dan pencatatan log aktivitas secara *real-time*.

1.5 Batasan Tugas Akhir

Beberapa batasan masalah yang telah diatur pada tugas akhir adalah sebagai berikut :

- Sistem *Smart Dorm Key* ini dirancang dan diimplementasikan secara khusus untuk lingkungan asrama Telkom University. Konfigurasi izin akses aplikasi terbatas hanya untuk mahasiswa/i Telkom University yang menggunakan *E-mail* sebagai autentikasi.

- Sistem pemindaian identitas berfokus pada penggunaan sensor sidik jari (*fingerprint*) dan pengenalan suara (*voice recognition*). Sistem ini dirancang untuk mencapai tingkat akurasi identifikasi sebesar 90% dengan kecepatan respons yang terjaga. Pengujian akurasi akan dilakukan pada sampel 10 pengguna dalam berbagai kondisi (normal, serak, dan bising).
- Sistem dibatasi untuk menyediakan pemantauan aktivitas secara *real-time*. Informasi yang dicatat dalam *log* aktivitas meliputi waktu dan identitas penghuni yang mencoba mengakses pintu.
- Sistem menggunakan *database* untuk menyimpan data identitas dan *log* aktivitas penghuni. Kapasitas penyimpanan ini terbatas, tergantung pada jumlah data yang dapat disimpan.
- Keamanan perangkat keras dibatasi pada desain di mana komponen-komponen penting seperti modul komunikasi dan penyimpanan data diletakkan di dalam perangkat pada tempat yang tersembunyi agar sulit diakses secara fisik oleh pihak yang tidak sah.
- Batasan layanan berfokus pada pencapaian kualitas layanan (QoS) yang optimal, terutama dalam hal kecepatan respon data dan meminimalisir gangguan. Pengukuran akan dilakukan untuk parameter *delay* dan *jitter* dalam berbagai skenario jaringan.

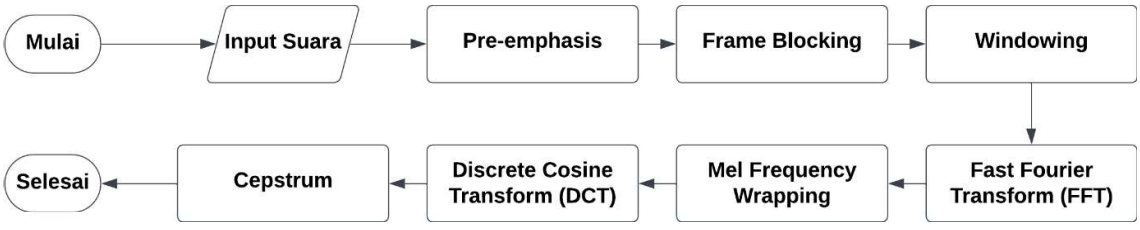
BAB 2
TINJAUAN PUSTAKA

2.1 *Machine learning*

Machine learning adalah teknologi fundamental yang menjadi mesin penggerak di balik sistem *voice recognition* (pengenalan suara). Peran dari *machine learning* ini adalah untuk melatih sistem agar mampu mengenali pola – pola yang sangat kompleks dan bervariasi dalam sinyal suara manusia, kemudia menerjemahkanya menjadi perintah yang dapat dieksekusi. Proses ini dimulai dengan metode seperti *Mel-Frequency Cepstral Coefficients* (MFCC), dengan mengubah rekaman suara analog menjadi data digital. Data ini kemudian diekstraksi, yaitu proses mengambil karakteristik unik yang menjadi pembeda setiap suara. Kumpulan karakteristik inilah yang pada akhirnya dianalisis oleh model *machine learning* untuk dilatih dan mengidentifikasi atau mengklasifikasikan suara tersebut dengan menggunakan kombinasi model *machine learning Convolutional Neural Networks* (CNN) yang efektif dalam mengenali pola lokal pada spektogram, seperti bentuk vokal dan konsonan [5].

2.1.1 *Mel-Frequency Cepstral Coefficients* (MFCC)

Mel-Frequency Cepstral Coefficients (MFCC) adalah sebuah teknik ekstraksi fitur yang esensial di bidang *voice recognition*, dirancang untuk mentransformasi sinyal suara menjadi vektor fitur yang ringkas dan informatif. Teknik ini didasarkan pada skala frekuensi *Mel*, sebuah model persepsi pendengaran manusia yang non-linear, di mana sensitivitas lebih tinggi diberikan pada frekuensi rendah. Secara prosedural, sinyal audio disegmentasi, dianalisis dalam domain frekuensi, lalu dilewatkan melalui *filterbank Mel*. Koefisien yang dihasilkan dari transformasi matematis akhir (DCT) secara efektif mengisolasi komponen linguistik dari sinyal (fonem, intonasi) dari variasi non-linguistik seperti amplitudo dan derau [5].



Gambar 2.1 Tahapan Metode MFCC

Tujuan dari metode *mel-frequency cepstral coefficients* (MFCC) ini adalah untuk melakukan ekstraksi pada filter dan menghapus bagian sumber data suara. Berikut adalah tahapan metode MFCC :

a. *Pre-Emphasis*

Pre-Emphasis yaitu proses untuk memperkuat komponen frekuensi tinggi pada sinyal suara yang di*input*. Hal ini dilakukan karena pada sinyal suara manusia, energi pada frekuensi tinggi cenderung lebih lemah akibat sifat alami pita suara dan resonansi rongga mulut. Informasi penting untuk mengenali karakter suara banyak terkandung pada frekuensi tinggi [6]. Proses pre-emphasis dilakukan dengan menerapkan filter high-pass sederhana pada sinyal input seperti yang terlihat pada persamaan (2.1) [6].

$$S'_n = S_n - \alpha S_{n-1} \tag{2.1}$$

Dimana :

S_n : nilai sampel ke-n

α : konstanta *pre-emphasis*, $0.9 \leq \alpha \leq 1.0$

b. *Frame Blocking*

Pada bagian *framing* sinyal suara dibagi menjadi beberapa segmen kecil yang disebut frame karena sinyal suara manusia bersifat *non-stationary* (karakteristiknya berubah terhadap waktu), sedangkan banyak metode analisis sinyal memerlukan sinyal yang stationer (karakteristiknya konstan pada periode tertentu). Dengan membagi sinyal menjadi *frame-frame* pendek berdurasi sekitar 20-30 milidetik. Selain itu, untuk memastikan informasi pada batas frame tidak hilang, *frame-frame* ini biasanya saling tumpeng tindih (*overlap*) sekitar 10-15 milidetik. Hasil dari tahap *framing* adalah serangkaian *frame* yang siap untuk diproses pada domain frekuensi [6].

c. *Windowing*

Setiap *frame* kemudian diberikan *window* untuk meredam efek diskontinuitas pada tiap ujung *frame* yang dapat menyebabkan distorsi pada spektrum frekuensi. Tanpa *windowing*, perbedaan nilai tiba-tiba anatara *frame* dapat menghasilkan efek kebocoran (*spectral leakage*) pada hasil transformasi frekuensi. Untuk itu digunakan fungsi *window function*, seperti *hamming window*, yang mereduksi amplitude sinyal di tepi *frame* menjadi nol secara halus. *Windowing* menhasilkan sinyal *frame* yang lebih mulus, sehingga Ketika dianalisis di domain

frekuensi hasilnya menjadi lebih bersih dan akurat [6]. Proses *hamming window* tersebut dapat dituliskan dalam persamaan sebagai berikut .seperti pada persamaan (2.2).

$$w(n) = 0{,}54 - 0{,}46\ cos = \frac{2\pi n}{N-1} \tag{2.2}$$

Dimana :

N : Jumlah sampel pada masing-masing *frame*.

n : 0, 1, 2, 3, …, $N-1$

d. *Fast Fourier Transform* (FFT)

Pada masing-masing *frame* yang sudah di-*windowing* diubah dari domain waktu ke domain frekuensi menggunakan *Fast Fourier Transform* (FFT). Perubahan ke domain frekuensi dilakukan karena ciri khas suara manusia lebih jelas terlihat pada pola distribusi frekuensinya daripada pada bentuk gelombang waktu. FFT menghitung komponen-komponen sinyal pada berbagai frekuensi dan menghasilkan spektrum magnitudo yang merepresentasikan seberapa kuat energi pada setiap frekuensi dalam *frame* tersebut. Hasil dari tahap ini adalah spektrum frekuensi dari masing-masing *frame* [6]. Proses FFT ini dapat ditulis pada persamaan (2.3).

$$f(n) = \sum_{K=0}^{N-1} (Y_k)\, e^{-\frac{2\pi jkn}{N}}, n = 0, 1, 2, \dots, N-1 \tag{2.3}$$

Dimana :

$f(n)$: Frekuensi.

k : 0, 1, 2, …, $(N-1)$

N : Jumlah sampel pada masing-masing *frame*.

j : Bilangan imajiner.

n : 0, 1, 2, 3, …, $(N-1)$

e. *Mel-Frequency Wrapping*

Spektrum frekuensi hasil FFT kemudian diubah ke skala *Mel* untuk menyesuaikan dengan cara telinga manusia mendengar suara. Penelitian psikofisik menunjukkan bahwa telinga manusia lebih sensitif terhadap perbedaan frekuensi di daerah rendah dibandingkan daerah tinggi, sehingga skala *Mel* digunakan untuk merefleksikan persepsi ini. Caranya, spektrum frekuensi dilewatkan melalui sekumpulan filter segitiga yang tersusun dalam skala *Mel*, yang

disebut *Mel* filter bank. Setiap filter mengakumulasi energi pada rentang frekuensi tertentu dalam skala *Mel*. Hasil tahap ini berupa vector energi dari masing-masing filter *Mel* yang menggambarkan intensitas suara pada rentang frekuensi yang relevan bagi pendengaran manusia [6]. Untuk menghitung skala *Mel* pada frekuensi dalam Hz ditulis dalam persamaan (2.4).

$$Mel\ f = 2595 \times \log_{10}\left(1 + \frac{f}{700}\right)$$

$$f = 700\ (10^{\frac{m}{2595}} - 1) \tag{2.4}$$

Dimana :

$Mel\ f$: Nilai frekuensi *Mel* dari f.

f. *Discrete* Cosine Transform (DCT)

Tahap akhir dari MFCC adalah DCT pada vektor *log* energi filter *Mel*. Tujuan dari DCT adalah memadatkan informasi ke dalam beberapa koefisien pertama dengan meminimalkan korelasi antar komponen. Koefisien pertama umumnya mewakili energi rata-rata, sementara koefisien-koefisien berikutnya merepresentasikan pola distribusi spektral yang menjadi ciri khas suara. Hasil akhir dari tahap ini adalah sekumpulan koefisien MFCC untuk masing-masing *frame* suara yang kemudian digunakan sebagai fitur dalam proses klasifikasi atau pengenalan suara [6]. Persamaan DCT ini dituliskan seperi pada (2.5).

$$C_n = \sum_{K=1}^{K}(logS_k)\cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{k}\right],\ n = 1, 2, \ldots, k \tag{2.5}$$

Dimana :

C_n : Koefisien *cepstrum mel-frequency*

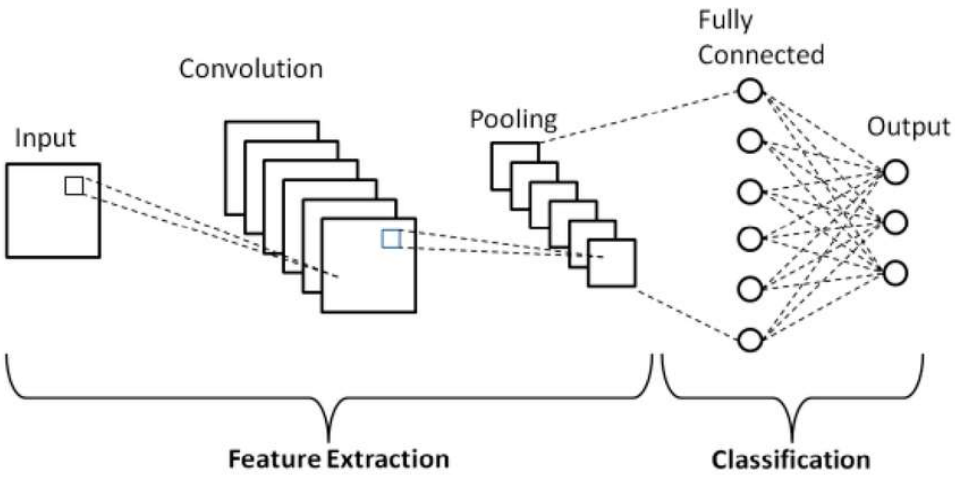
S_k : *Mel Frequency*

n : Bilangan bulat dari 1, …,N (jumlah total sampel).

k : Jumlah koefisien.r

2.1.2 Convolutional Neural Networks (CNN)

Convolutional Neural Network (CNN) adalah arsitektur *deep learning* yang andal untuk mengenali suara karena keahliannya dalam mengidentifikasi pola visual. Agar bisa memproses suara, CNN mengubah sinyal audio menjadi "gambar" yang disebut spektogram. Sama seperti saat mengenali objek dalam foto, CNN memindai spektogram untuk menemukan ciri-ciri khas ucapan. Jaringan ini belajar secara mandiri untuk mendeteksi komponen-komponen dasar (seperti vokal) dan kemudian menggabungkannya untuk memahami struktur yang lebih kompleks (seperti kata). Keunggulan utamanya adalah CNN dapat menangkap esensi sebuah kata tanpa terganggu oleh perbedaan kecil, misalnya jika diucapkan sedikit lebih cepat atau lebih tinggi [5].



Gambar 2.2 Arsitektur *Convolutional Neural Network*

Sumber : [19]

Berikut adalah penjelasan dari Gambar 2.2 sebagai berikut :

1. Input Sinyal Suara

Input awal adalah sinyal suara mentah dalam bentuk gelombang audio (*waveform*), seperti yang ditunjukkan di paling kiri [7]. Namun, CNN tidak bisa langsung memproses gelombang 1D ini. Oleh karena itu, sinyal ini pertama-tama diubah menjadi representasi visual 2D yang disebut spektogram (contohnya menggunakan MFCC). Spektogram ini adalah "gambar" dari suara, yang memetakan frekuensi terhadap waktu, sehingga bisa dianalisis oleh CNN [7].

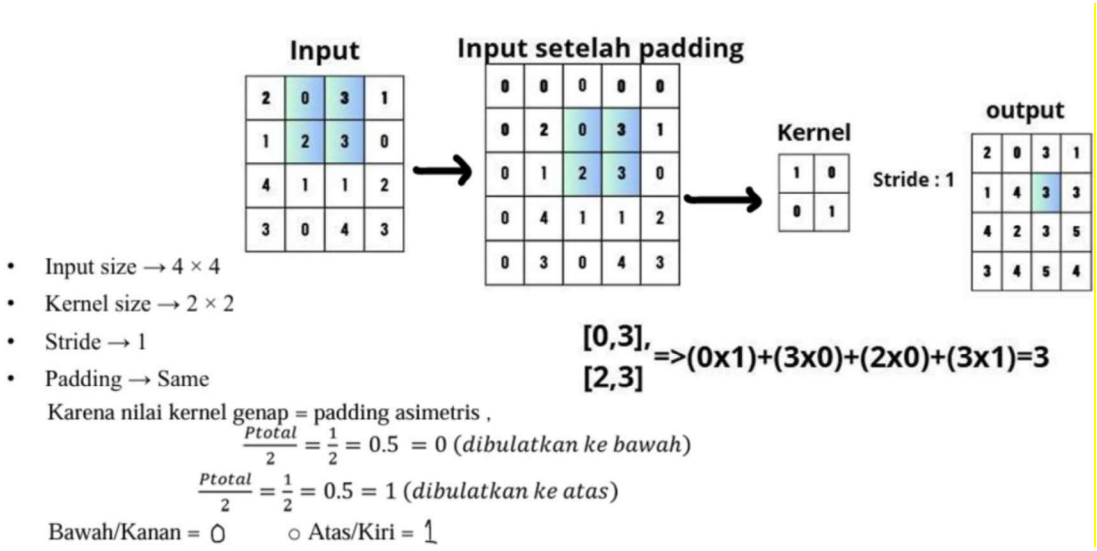
2. Tahap Ekstraksi Fitur

Ini adalah bagian inti di mana CNN menggunakan keahliannya untuk "melihat" pola dalam spektogram. Tahap ini terdiri dari dua lapisan utama yang diulang beberapa kali:

- *Convolution* (Konvolusi)

Lapisan ini bekerja seperti kaca pembesar yang digerakkan di seluruh "gambar" spektogram untuk menemukan pola-pola kecil. Filter-filter pada lapisan ini secara otomatis belajar untuk mendeteksi fitur-fitur akustik dasar, seperti bentuk vokal, konsonan letup (seperti 'p' atau 't'), atau suara desis (seperti 's') [7]. Hasil dari setiap filter adalah sebuah *Feature Map*, yaitu sebuah peta yang menyorot di mana saja fitur tersebut ditemukan dalam suara [7].

Konvolusi pada arsitektur CNN adalah operasi matematis fundamental yang berfungsi untuk mengekstraksi fitur dari data masukan, khususnya gambar. Proses ini melibatkan sebuah filter atau kernel, yaitu matriks kecil berisi bobot (nilai-nilai yang dipelajari selama pelatihan), yang digeser secara sistematis ke seluruh area gambar. Pada setiap posisi, dilakukan operasi perkalian *element-wise* antara nilai pada kernel dengan bagian gambar yang ditutupinya, lalu semua hasilnya dijumlahkan untuk menghasilkan satu nilai Tunggal seperti pada Gambar 2.3. Kumpulan dari semua nilai tunggal ini membentuk sebuah matriks baru yang disebut *feature map* atau peta fitur.



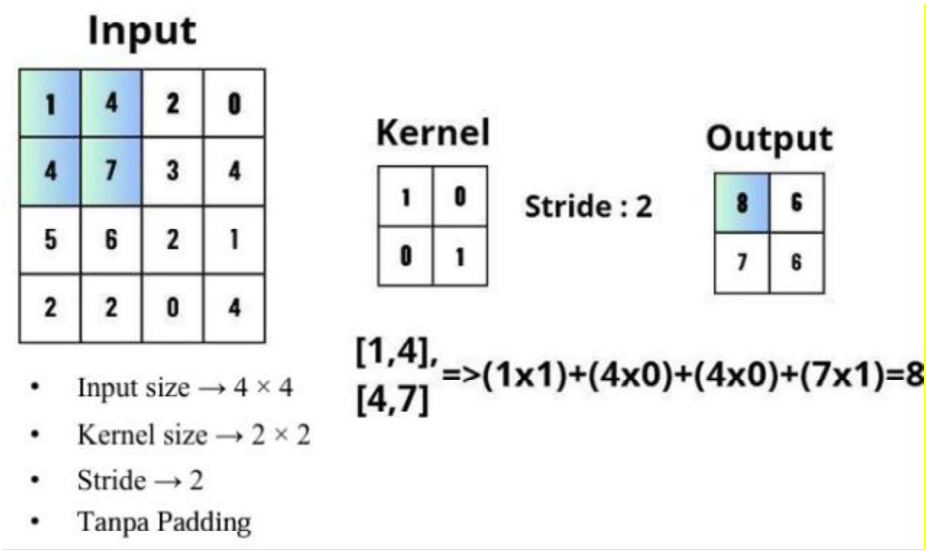
Gambar 2.3 Konvolusi Manual

Tujuan utama dari konvolusi adalah untuk mengidentifikasi pola-pola spesifik seperti tepi, sudut, tekstur, atau bentuk-bentuk yang lebih kompleks pada lapisan yang lebih dalam, sehingga memungkinkan jaringan untuk mengenali objek secara efektif.

- *Pooling* (Pengumpulan)

Setelah menemukan banyak fitur, lapisan *pooling* bertugas untuk meringkas dan mereduksi ukuran *feature map*. Cara kerjanya adalah dengan mengambil nilai terpenting (misalnya nilai maksimum atau rata-rata) dari sebuah area kecil [7]. Tujuannya ada dua: (1) Membuat komputasi lebih efisien, dan (2) Membuat model lebih tahan terhadap variasi kecil. Misalnya, jika sebuah kata diucapkan sedikit lebih cepat atau lambat, fitur suaranya mungkin sedikit bergeser di spektogram. *Pooling* membantu model tetap mengenali fitur tersebut meskipun posisinya tidak persis sama [7].

Pooling dalam arsitektur CNN adalah sebuah operasi *down-sampling* yang bertujuan untuk mengurangi dimensi spasial (lebar dan tinggi) dari *feature map* yang dihasilkan oleh lapisan konvolusi. Proses ini bekerja dengan cara meringkas informasi dari sekelompok kecil piksel menjadi satu nilai tunggal. Jenis yang paling umum adalah *Max Pooling*, yang mengambil nilai piksel maksimum dari setiap area (misalnya, area 2x2). Tujuan utama dari pooling adalah untuk mengurangi jumlah parameter dan beban komputasi jaringan, mengontrol *overfitting*, serta membuat representasi fitur lebih tahan (invarian) terhadap pergeseran atau distorsi kecil pada gambar.



Gambar 2.4 Proses *Pooling* pada *Feature Map*

Sebagai contoh, pada Gambar 2.4 proses *Max pooling* dengan filter 4x4 dan stride 2 akan mengambil sebuah blok nilai, misalnya [1, 4, 4, 7], lalu menghitung hasilnya menjadi 8. Nilai tunggal ini kemudian menjadi bagian dari peta fitur hasil *pooling* (*pooled feature map*) yang baru. Ketika operasi ini diterapkan ke seluruh peta

fitur, hasilnya adalah sebuah peta fitur dengan ukuran yang lebih kecil, sehingga lebih efisien untuk diolah oleh lapisan jaringan berikutnya. Dengan demikian, *Max pooling* bekerja dengan cara mengambil nilai piksel maksimum dari setiap blok atau area pada *feature map* serta mampu mengurangi kompleksitas komputasi dan membuat sistem lebih tahan terhadap pergeseran kecil pada gambar, sambil tetap mempertahankan informasi suara dari setiap fitur.

3. Tahap Klasifikasi

Setelah fitur-fitur penting dari suara diekstrak, tahap selanjutnya adalah mengklasifikasikan suara tersebut.

- *Fully Connected* (Terhubung Penuh)

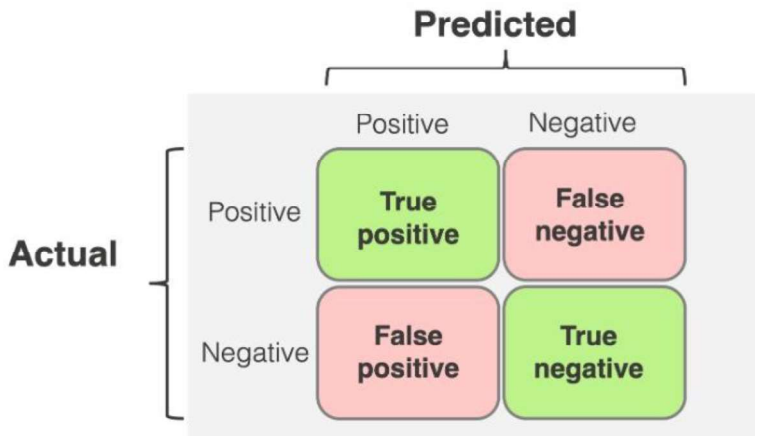
  Data yang sudah ringkas dari tahap *pooling* kemudian "diratakan" menjadi satu baris data panjang dan dimasukkan ke lapisan ini [7].

- *Output* (Keluaran)

  Ini adalah lapisan terakhir yang memberikan hasil akhir. Jumlah neuron (lingkaran) di lapisan ini sesuai dengan jumlah kelas yang ingin dikenali [7]. Misalnya, jika tujuannya adalah mengenali empat perintah ("maju", "mundur", "kiri", "kanan"), maka akan ada empat neuron di lapisan *output*. Neuron dengan nilai probabilitas tertinggi akan menjadi jawaban akhir dari system [7].

2.1.3 Model *Performance Evaluate Confusion Matrix*

Confusion Matrix merupakan sebuah matriks tabulasi yang berfungsi sebagai alat evaluasi komprehensif untuk mengukur performa sebuah model klasifikasi. Fungsinya adalah untuk memvisualisasikan kinerja model dengan cara membandingkan setiap hasil prediksi dengan kelas aktualnya. Tabel ini mengkuantifikasi prediksi ke dalam empat kategori: jumlah data yang diprediksi benar (*True Positive* dan *True Negative*) dan jumlah data yang diprediksi salah (*False Positive* dan *False Negative*). Dengan demikian, matriks ini tidak hanya menyimpulkan performa secara umum, tetapi juga memberikan wawasan detail mengenai jenis-jenis kesalahan yang sering dibuat oleh model [8].



Gambar 2.5 Struktur *Confusion Matrix*

Sumber gambar : https://www.evidentlyai.com/classification-metrics/confusion-matrix

Diperlukan metrik evaluasi untuk menilai performa klasifikasi dari model yang sudah dilatih. Metrik yang paling sering dipakai dalam tugas klasifikasi adalah *accuracy*. Namun, untuk memperoleh penilaian yang lebih menyeluruh, perlu juga digunakan metrik lain seperti *precision*, *recall*, dan *F1-score* [9].

a. *Recall* (*Sensitivity*)

Recall (*sensitivity*) adalah metrik yang mengukur kemampuan model dalam mendeteksi seluruh kasus positif yang sebenarnya, dengan membandingkan jumlah prediksi positif yang benar terhadap total kasus positif aktual [9]. Rumus untuk menghitung *recall* dapar dilihat pada (2.6).

$$Recall = \frac{TP}{TP + FN} \tag{2.6}$$

b. *Precision*

Precision adalah metrik yang menilai seberapa tepat prediksi positif yang dibuat oleh model, dengan membandingkan jumlah prediksi positif yang benar terhadap seluruh prediksi positif yang dihasilkan [9]. Rumus untuk menghitung *precision* dapat dilihat pada (2.7).

$$Precision = \frac{TP}{TP + FP} \tag{2.7}$$

c. *F1 Score*

F1 Score adalah rata-rata harmonis (*harmonic mean*) dari *precision* dan *recall*. Metrik ini digunakan untuk memberikan keseimbangan antara keduanya, khususnya ketika terjadi ketidakseimbangan jumlah data pada masing-masing kelas [9]. Adapun rumus untuk menghitung *F1 Score* seperti pada (2.8).

$$F1_{score} = 2 \times \frac{presisi\ x\ recall}{presisi\ +\ recall} \tag{2.8}$$

Keterangan :

TP: Prediksi positif, label asli positif.

TN: Prediksi negatif, label asli negatif.

FP: Prediksi positif, label asli negatif.

FN: Prediksi negatif, label asli positif.

2.2 Mobile Application

Sistem *Smart Dorm Key* dirancang dengan aplikasi seluler sebagai antarmuka utama yang dapat diakses melalui perangkat *smartphone*. Aplikasi ini memiliki peran sentral dalam proses verifikasi dua langkah, khususnya untuk melakukan otentikasi pengguna melalui pengenalan suara (*voice recognition*) sebelum memberikan akses ke pemindaian sidik jari pada perangkat keras. Teknologi ini memungkinkan aplikasi untuk mengenali pola-pola akustik yang unik dari frasa sandi yang diucapkan oleh pengguna, yang menjadi ciri khas dan pembeda setiap individu. Aplikasi ini dikembangkan agar kompatibel dengan sistem operasi *Android*, sehingga memberikan kemudahan akses bagi pengguna untuk melakukan verifikasi kapan saja dan di mana pun selama terhubung dengan sistem.

Dengan demikian, aplikasi ini menawarkan solusi praktis untuk meningkatkan keamanan asrama, memungkinkan proses otentikasi yang lebih modern, berlapis, dan sulit untuk dimanipulasi dibandingkan dengan metode konvensional seperti kunci manual. Dengan memanfaatkan *machine learning* untuk pengenalan suara, sistem ini mampu memberikan lapisan keamanan tambahan dan membantu mengurangi risiko akses yang tidak sah. Selain sebagai media verifikasi, aplikasi ini juga berfungsi untuk pendaftaran pengguna baru dan memungkinkan *admin* untuk memantau *log* aktivitas secara *real-time*.

BAB 3
SPESIFIKASI DAN DESAIN SISTEM

3.1 Spesifikasi Sistem

Penentuan spesifikasi sistem untuk "*Smart Dorm Key*" ini didasarkan pada analisis komprehensif terhadap kelemahan sistem keamanan asrama konvensional serta studi mendalam terhadap standar industri dan regulasi pemerintah yang berlaku. Sebagai landasan, tinjauan dilakukan pada produk komersial seperti Samsung SHS-P718 dan *IT Smart Door Lock* A230, yang menetapkan standar fitur meliputi multi-metode akses (seperti *fingerprint*, RFID, dan PIN), kapasitas penyimpanan data untuk ratusan pengguna, dan integrasi aplikasi seluler untuk notifikasi serta kontrol jarak jauh [10], [11], [12]. Selain itu, perancangan sistem wajib mematuhi regulasi pemerintah Indonesia terkait keamanan siber (Permen Kominfo No. 5/2020), perlindungan data pribadi (UU No. 27/2022), dan standar kualitas perangkat keras (SNI dan SDPPI) [13], [14], [15]. Berdasarkan pilar-pilar tersebut, sistem ini diimplementasikan dengan metode verifikasi dua langkah menggabungkan teknologi biometrik sidik jari dan pengenalan suara yang dikendalikan oleh mikrokontroler ESP32 dan terhubung ke aplikasi. Oleh karena itu, spesifikasi akhir dari proyek ini merupakan hasil sintesis dari kebutuhan untuk mengatasi masalah keamanan yang ada, selaras dengan tolak ukur industri dan mandat peraturan yang berlaku.

Tabel 3.1 Batasan dan Spesifikasi Alat yang digunakan

No	Hal	Rincian
1	Sistem Pemindaian Identitas	Sistem pemindaian meliputi sensor deteksi orang, suara dan jari. Serta menjaga tingkat akurasi sebesar 90% .
2	Konfigurasi Izin Akses	Sistem ini hanya bisa diakses oleh mahasiswa/i Telkom, karena memakai *E-mail* sebagai akses untuk dapat menggunakan aplikasi.
3	Sistem Pemantauan *Real-Time*	Memberikan informasi yang dapat diakses secara langsung dan memberikan laporan tentang aktivitas penghuni yang mencoba

		untuk masuk. Informasi ini meliputi waktu dan identitas penghuni.
4	Kapasitas Penyimpanan	*Database* diperlukan untuk menyimpan identitas dan log aktivitas dari penghuni. Kapasitas penyimpanan ini terbatas tergantung dari banyaknya jumlah data yang dapat disimpan.
5	Keamanan Perangkat	Komponen penting seperti modul komunikasi dan penyimpanan data diletakkan di tempat tersembunyi dalam perangkat, sehingga sulit diakses langsung oleh pelaku perusakan.
6	*Quality of Service*	Memiliki kualitas layanan yang optimal dalam proses penerimaan kecepatan respons data serta mengalami gangguan seminimal mungkin.

3.2 Desain Sistem

Desain sistem merupakan tahap perancangan teknis yang menjabarkan bagaimana seluruh komponen sistem akan terhubung dan bekerja sama. Tahapan ini bertujuan untuk membuat solusi teknis berdasarkan analisis kebutuhan pengguna dan batasan yang ada. Sebuah desain yang matang akan mempermudah implementasi serta memastikan aplikasi dapat berfungsi secara optimal dan efisien.

3.2.1 Deskripsi Umum Desain

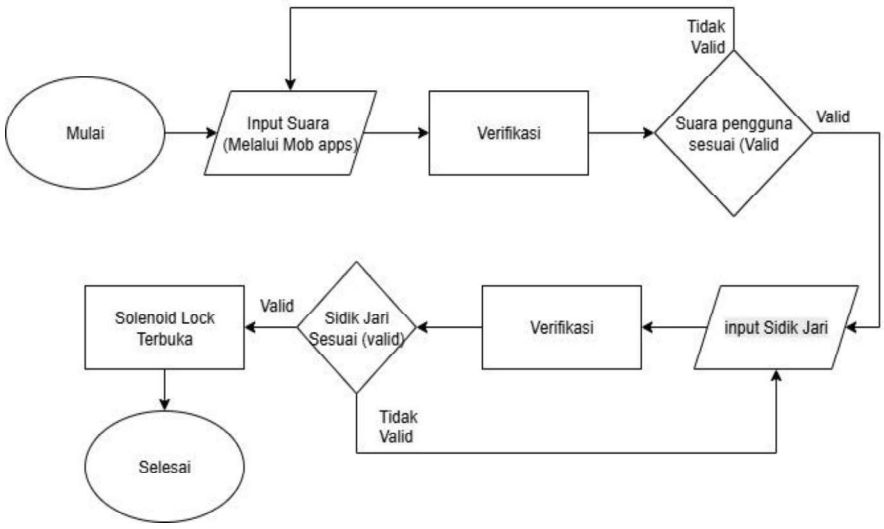
Desain sistem yang dirancang terdiri dari dua komponen utama, yaitu perangkat "*Smart Dorm Key*" yang berbasis IoT dan aplikasi berbasis *mobile* yang berfungsi sebagai antarmuka pengguna. Pada aplikasi *mobile* berfungsi sebagai media bagi pengguna untuk mendaftarkan data pengguna, serta untuk melakukan tahap pertama verifikasi yaitu pemindaian suara. Sistem ini didukung oleh machine learning yang menggunakan metode *Mel-Frequency Cepstral Coefficients (MFCC)* untuk ekstraksi fitur suara dan model *Convolutional Neural Networks (CNN)* untuk proses pengenalan suara.

Sementara itu, perangkat keras "*Smart Dorm Key*" menggunakan mikrokontroler ESP32 sebagai unit kendali pusat. Untuk verifikasi tahap kedua, sistem dilengkapi dengan sensor *fingerprint*. Sebagai pusat kendali utama, sistem menggunakan *solenoid door lock* yang akan aktif untuk membuka pintu jika kedua verifikasi berhasil. Untuk memberikan umpan balik kepada pengguna, sistem dilengkapi dengan layar LCD dan *buzzer*. Fitur tambahan mencakup *No Touch Exit Sensor* untuk membuka pintu dari dalam tanpa sentuhan dan tombol fisik untuk mengelola data sidik jari langsung pada perangkat. Integrasi antara aplikasi, *machine learning*, dan perangkat IoT ini bertujuan untuk menciptakan solusi keamanan asrama yang berlapis dan efisien.

3.2.2 Detail Desain Sistem Terpilih

3.2.2.1 Diagram Alir Sistem *Smart Dorm Key*

Diagram alir berikut merupakan deskripsi dari alat yang akan dirancang. Diagram alur alat yang dirancang tersebut disajikan pada Gambar 3.1 di bawah.



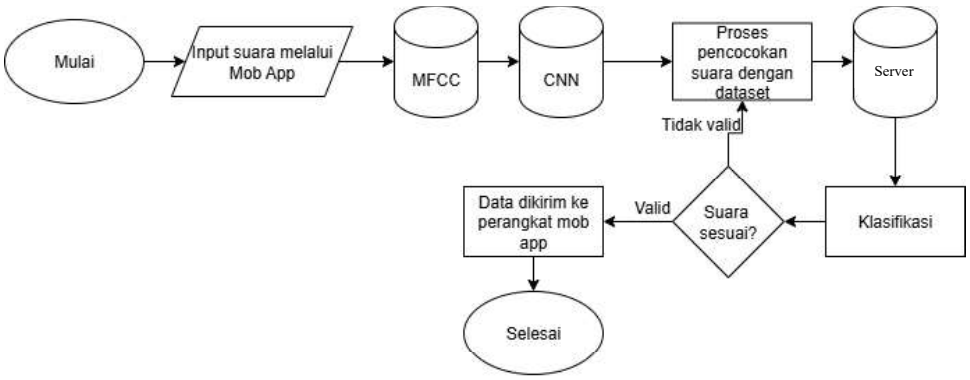
Gambar 3.1 Diagram Alir Sistem *Smart Dorm Key*

Diagram Alir tersebut menggambarkan bagaimana alur kerja dari sistem *smart dorm lock* ini menggunakan sensor sidik jari dan pengenalan suara melalui *mobile app*. Proses ini dimulai dengan menerima data sidik jari pengguna yang dimana jika data yang diterima tersebut sesuai maka data tersebut akan dikirim ke mikrokontroler ESP32 dan akan tersimpan di dalam *database*, kemudian pengguna harus memasukkan kode suara yang sudah terdaftar melalui *mobile app* dan jika suara yang dimasukkan sesuai dengan yang sudah terdaftar maka *solenoid door lock* akan terbuka dan pengguna dapat masuk ke asrama. Secara keseluruhan, *flowchart* ini memberikan penjelasan tentang bagaimana sistem dari *smart dorm lock*

melakukan proses pengenalan sidik jari dan pengenalan suara untuk menentukan apakah data sesuai atau tidak sehingga pengguna dapat diberikan akses ke tahap berikutnya.

3.2.2.2 Diagram Alir *Machine Learning*

Pada Gambar 3.2 yang disajikan dibawah ini merupakan diagram alir dari machine learning *Smart Dorm Key* yang akan dirancang.

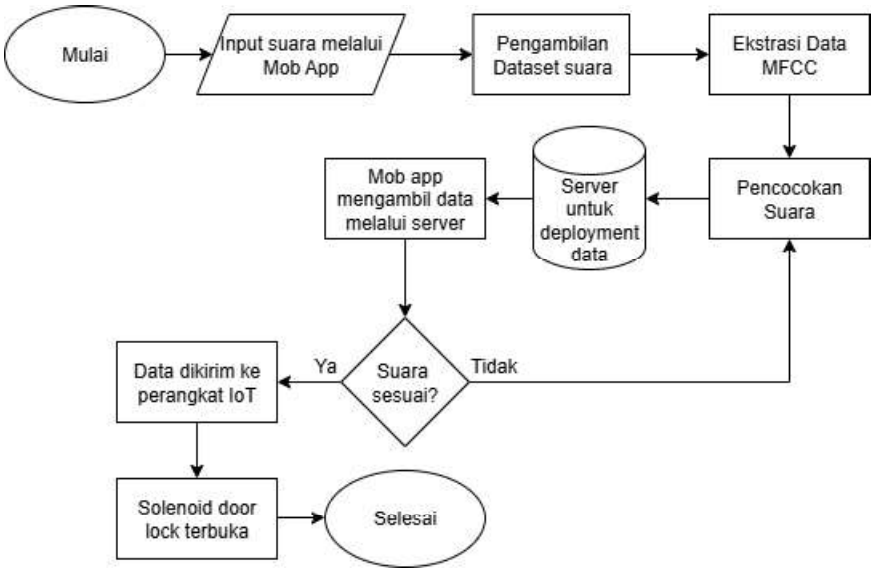


Gambar 3.2 Diagram Alir *Machine Learning*

Diagram alir ini merupakan alur yang menampilkan tahapan dari proses *machine learning* untuk sistem *smart dorm key*. Proses ini dimulai dengan pengambilan *dataset* suara yang menjadi input utama sistem dan kemudian dataset suara akan diproses menggunakan metode MFCC. Setelah data selesai diproses oleh kemudian akan dilakukan prosesan lebih lanjut menggunakan CNN. Kemudian akan dilakukan pencocokan suara, di mana *input* suara akan dibandingkan dengan *dataset* yang tersedia. Hasil dari proses pencocokan suara akan disimpan dalam *Server*. Selanjutnya, sistem melakukan klasifikasi suara berdasarkan hasil pencocokan untuk menentukan identifikasi suara sesuai dengan *dataset* yang ada. Kemudian ketika pengguna menginputkan suara, sistem akan memverifikasi apakah suara tersebut *valid* atau tidak. Jika suara *valid* maka data akan diteruskan ke perangkat *mobile application* tetapi jika suara tidak *valid* sistem akan kembali ke tahap pencocokan suara untuk melakukan verifikasi ulang.

3.2.2.3 Diagram Alir *Mobile Application*

Diagram alir yang ditampilkan pada gambar 3.3 di bawah ini merupakan penjelasan alur dari *mobile application* yang akan dirancang untuk sistem *smart dorm key*.

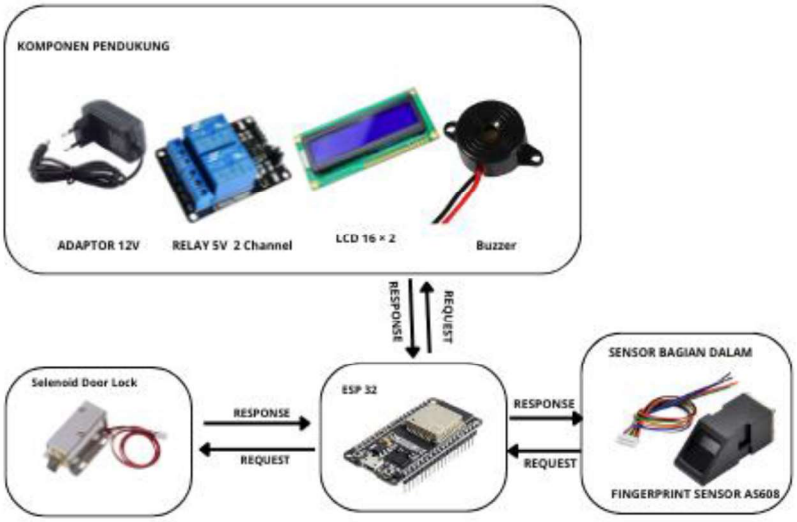


Gambar 3.3 Diagram Alir *Mobile Application*

Pada Diagram alir diatas menggambarkan sistem kerja dari *mobile application* dimana, saat kita menggunakan aplikasi tersebut kita akan melakukan proses *input* suara melalui aplikasi. Setelah melakukan penginputan suara, data suara akan diteruskan menuju proses *Machine Learning* MFCC yang berfungsi sebagai Ekstraksi data suara. Setelah proses ekstraksi selesai nantinya akan ada pencocokan suara melalui *server* untuk *development* data, lalu aplikasi akan mengambil data suara yang berada pada *firebase*. Jika suara *valid* nantinya akan diteruskan menuju perangkat IoT, dan jika data suara tidak *valid* maka data akan dikembalikan ke pencocokan suara.

3.2.2.4 Arsitektur Perangkat Keras

Untuk melakukan perancangan dari *smart dorm key* diperlukan rangkaian arsitektur perangkat keras dari smart dorm key yang membantu dalam perincian komponen-komponen yang akan digunakan sesuai dengan kebutuhan untuk pembuatan *smart dorm key*. Dalam perancangan ini terdapat beberapa jenis komponen yang digunakan yaitu sensor bagian dalam, mikrokontroler, *solenoid* dan komponen pendukung yang akan ditampilkan pada gambar di bawah.

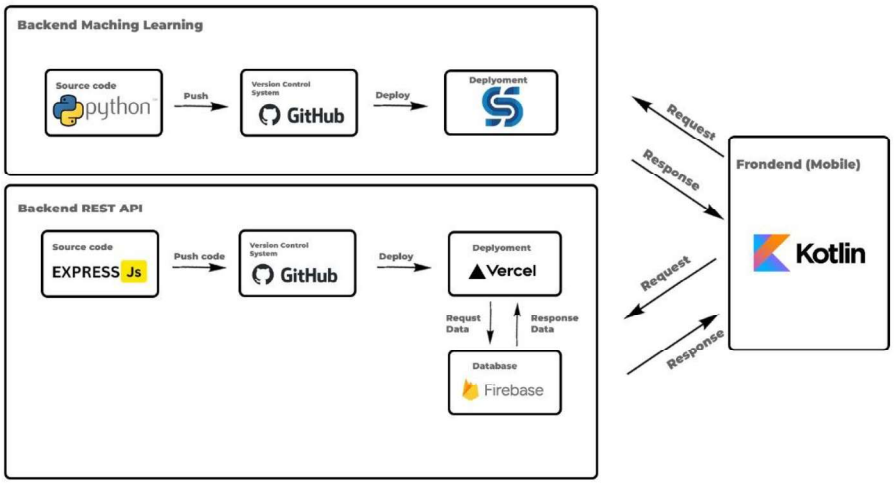


Gambar 3.4 Arsitektur Perangkat Keras

Arsitektur perangkat keras sistem *smart door key* terdiri dari beberapa komponen yang saling terhubung, seperti yang ditunjukkan pada Gambar 3.4. Di bagian atas gambar terdapat komponen pendukung, termasuk adaptor 12V yang berfungsi sebagai sumber daya, *relay* 5v2 yang berfungsi untuk mengalirkan arus yang diterima dari adaptor ke komponen lain, LCD 16 x 2 yang berfungsi untuk menampilkan informasi, dan *buzzer* yang berfungsi sebagai pemberi sinyal suara. Sensor bagian dalam di sebelah kanan atas adalah *fingerprint* sensor AS608 yang dapat membaca sidik jari pengguna. Mikrokontroler ESP32 yang terletak di tengah gambar berfungsi sebagai pusat kontrol sistem. ESP32 berfungsi sebagai pusat, memproses semua *input* dan mengendalikan *output* sistem. ESP32 dan komponen pendukung di atasnya memiliki alur komunikasi dua arah, yang menunjukkan proses pengiriman dan penerimaan data. Ada *solenoid door lock* di sebelah kanan ESP32. Ini berfungsi sebagai pengunci pintu elektrik dan akan aktif atau nonaktif tergantung pada instruksi ESP32.

3.2.2.5 Arsitektur Perangkat Lunak

Perancangan *smart dorm key* ini memerlukan rangkaian arsitektur dari perangkat lunak untuk membantu optimalisasi aplikasi yang akan digunakan. Terdapat beberapa bagian dalam perancangan perangkat lunak ini, yaitu *backend* dari *machine learning, backend REST API* dan *frontend* untuk *mobile*. Dalam perancangan bagian tersebut digunakan beberapa *software* pendukung yang akan ditampilkan pada gambar di bawah.



Gambar 3.5 Arsitektur Perangkat Lunak

Pada gambar 3.5 menampilkan arsitektur perangkat lunak yang akan digunakan untuk sistem *smart dorm key*. Terdapat beberapa bagian yaitu pada *backend REST API* memiliki fungsi untuk mengelola seluruh data yang dibutuhkan oleh aplikasi serta terhubung ke database. *Backend* dari *machine learning* yang berfungsi sebagai sistem utama untuk memproses data untuk fitur pengenalan suara. Kedua *backend* ini akan disambungkan ke *frontend (mobile)* dengan sistem *request* dan *response,* yaitu *frontend (mobile)* akan melakukan *request* untuk mengambil data pada *backend* dan setelah *request* diterima oleh *backend*, maka *backend* akan mengirimkan *response* berupa data yang diinginkan oleh *frontend (mobile). Frontend (mobile)* ini juga memiliki *interface* yang berfungsi sebagai perantara dari aplikasi dengan pengguna.

3.3 Metode Pengukuran yang Sesuai dengan Solusi Terpilih

Pada pengukuran/verifikasi spesifikasi ini dirancang untuk memastikan bahwa aspek yang digunakan dapat memenuhi standar dan sesuai dengan kebutuhan yang diperlukan.

Tabel 3.2 Pengukuran/Verifikasi Spesifikasi Alat yang Digunakan

No	Spesifikasi	Metode Pengukuran/Pengujian	Prosedur Pengujian
1	Sistem Pemindaian Identitas	Uji akurasi pemindaian fingerprint dan pengenalan suara.	Menggunakan sampel data kurang lebih 10 pengguna untuk memindai sidik jari dan suara mereka. Catat tingkat keberhasilan pemindaian dalam kondisi normal, basah, dan bising.
2	Konfigurasi Izin Akses	Simulasi akses menggunakan aplikasi dengan autentikasi berbasis *E-mail*.	Buat skenario autentikasi pengguna menggunakan data *E-mail* yang valid dan tidak *valid*. Catat waktu respons sistem dan keberhasilan validasi data.
3	*Log* Aktivitas *Real-Time*	Uji pencatatan aktivitas dengan *wireshark*	Lakukan simulasi akses masuk pada sistem dengan berbagai identitas pengguna. Verifikasi *log* aktivitas melalui aplikasi *mobile* dan catat waktu pencatatan.
4	Kapasitas Penyimpanan	Uji batas kapasitas penyimpanan *database* sistem.	Simulasikan penyimpanan data untuk 25 pengguna dengan *log* aktivitas masing-masing 12 entri. Periksa apakah sistem masih dapat menyimpan data baru tanpa *error* setelah kapasitas maksimum tercapai.

5	Keamanan Perangkat	Uji ketahanan perangkat terhadap akses fisik yang tidak sah.	Simulasikan percobaan membuka casing perangkat tanpa izin. Periksa apakah data atau komponen penting tetap aman dari manipulasi fisik.
6	*Quality of Service*	Uji kecepatan layanan *Quality of Service* dalam berbagai kondisi.	Melakukan simulasi skenario pengambilan data selama 5 menit dan lakukan perhitungan *delay* dan *jitter*

BAB 4
IMPLEMENTASI

4.1 Deskripsi Umum Implementasi

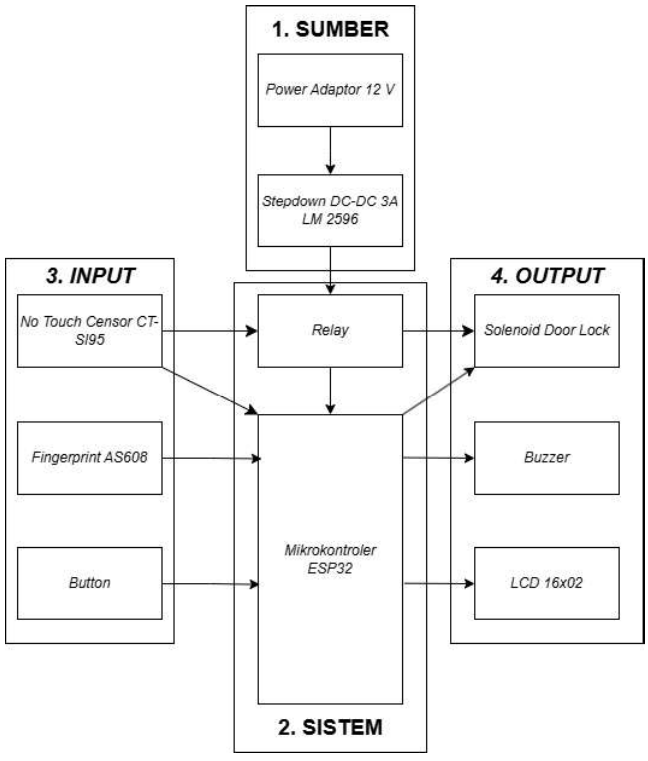
Implementasi pada proyek *Smart Dorm Key* Berbasis IoT adalah bertujuan untuk meningkatkan keamanan dan kenyamanan penghuni asrama dengan memanfaatkan teknologi *Internet of Things*. Proses kerja dari sistem aplikasi dan alat *Smart Dorm Key* ini dimulai dengan aplikasi yang terdiri dari *admin* dan *user*, untuk *admin* pada aplikasi berperan untuk melihat *logbook* akses masuk dan membantu user apabila terdapat kendala dalam melakukan akses masuk. Lalu untuk *user* yaitu membuat akun untuk masuk ke dalam aplikasi, setelah berhasil membuat akun dan *login* ke aplikasinya *user* mengisi identitas dan mendaftarkan *fingerprint* dan juga kata kunci suara yang akan dipakai untuk membuka pintu dan pada aplikasi untuk *user* digunakan untuk memasukan kunci suara pada saat membuka pintu.

Selanjutnya pada proses kerja dari alat *Smart Dorm Key* ini menggunakan verifikasi dua langkah, yaitu *fingerprint* dan *voice recognition*, untuk memastikan hanya penghuni yang sah dapat mengakses ruangan. Sistem ini dibangun menggunakan mikrokontroler ESP32 sebagai otak pengendali utama yang terhubung dengan berbagai komponen. Saat pengguna ingin membuka pintu, mereka pertama-tama harus melakukan verifikasi suara yang telah terdaftar melalui aplikasi, yang kemudian hasil verifikasi suara tersebut akan dikirimkan ke *server* sehingga perangkat ESP32 yang sudah terhubung ke jaringan melalui *Wi-Fi* bisa menerima hasil verifikasi. Jika verifikasi berhasil sistem akan memberikan konfirmasi awal melalui LCD 16x2 sebagai tanda verifikasi tahap pertama berhasil. Selanjutnya, pengguna harus menempelkan sidik jari pada sensor *fingerprint* dengan jari yang sudah didaftarkan sebelumnya. Jika sidik jari sesuai maka akan muncul pesan pada LCD 16x2 dan *buzzer* akan berbunyi yang menandakan bahwa sidik jari sesuai juga sebagai tanda bahwa verifikasi tahap kedua dianggap berhasil dan kemudian sistem akan mengaktifkan *solenoid door* untuk membuka kunci pintu. Namun, jika salah satu dari kedua verifikasi gagal, pintu tidak akan terbuka dan *buzzer* akan berbunyi sebagai tanda peringatan kegagalan dan akan terekam pada *logbook* secara *real-time* dalam aplikasi. Sistem ini dirancang untuk meningkatkan keamanan dengan memadukan biometrik dan autentikasi suara secara berlapis.

4.1.1 Blok Diagram *Smart Dorm Key*

Pada gambar 4.1 menampilkan sebuah blok diagram yang menggambarkan alur kerja dan hubungan antar komponen yang diperlukan untuk membentuk sistem *smart dorm key*.

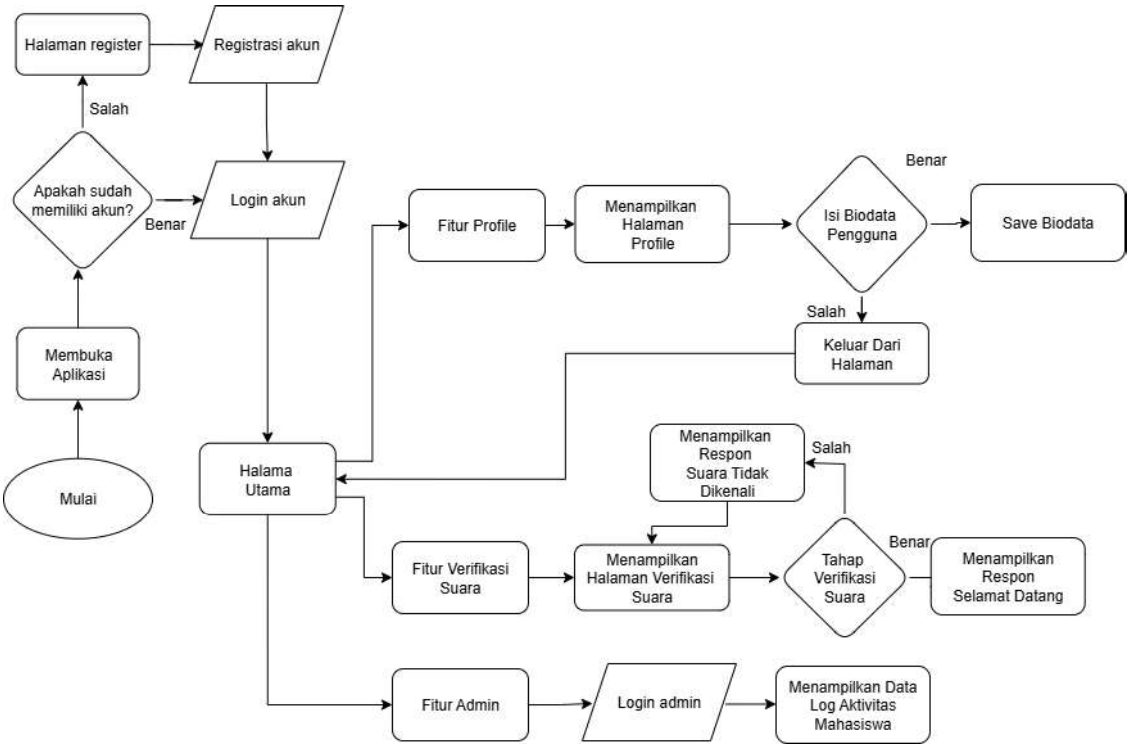
Proses dari sistem ini dimulai dengan menggunakan *power adaptor* sebagai sumber tegangan utama yang dihubungkan ke *step down* untuk mengatur besar tegangan yang diperlukan untuk keseluruhan komponen yang digunakan. ESP 32 bekerja sebagai mikrokontroler yang berfungsi sebagai pusat kontrol untuk komponen lainnya, dimana mikrokontroler akan menerima input berupa sidik jari melalui *fingerprint AS608* dan jika sidik jari benar maka *relay* akan mengaktifkan *solenoid door lock* agar pintu terbuka dan *buzzer* akan mengeluarkan bunyi tertentu jika sidik jari benar, tetapi jika sidik jari tidak sesuai maka *relay* tidak akan mengaktifkan *solenoid door lock* dan *buzzer* akan berbunyi dengan pola yang berbeda. N*o touch censor CT-SI95* ini juga terhubung ke *relay* dengan tujuan jika N*o touch censor CT-SI95* ini mendeteksi sesuatu di jarak 6 cm maka akan otomatis mengaktifkan *solenoid door lock.* Sistem ini juga menggunakan 2 *button,* dimana button 1 digunakan untuk menambahkan sidik jari dan button 2 untuk menghapus sidik jari yang sudah terdaftar. Pada *Smart Door Lock* ini tentunya menggunakan LCD 16x02 yang berfungsi untuk menampilkan segala kegiatan yang dilakukan oleh sistem, seperti ketika sedang menempelkan sidik jari, ketika pintu berhasil dibuka, menghapus dan menambahkan sidik jari.



Gambar 4.1 Blok Diagram *Smart Dorm Key*

4.1.2 Aplikasi *Smart Dorm Key*

Gambar 4.2 merupakan *flowchart* dari aplikasi *"Smart Dorm Key"*. *Flowchart* ini membantu dalam memberikan gambaran singkat dari aplikasi yang akan dirancang.



Gambar 4.2 Flowchart Aplikasi *Smart Dorm Key*

Alur *Flowchart* ini dimulai ketika pengguna membuka aplikasi dan sistem akan melakukan pengecekan untuk mengetahui apakah pengguna sudah memiliki akun. Apabila pengguna belum memiliki akun, sistem akan mengarahkan *user* ke halaman *register* untuk melakukan pembuatan akun dan jika sudah membuat akun maka *user* bisa melakukan proses *login*. Namun, jika pengguna sudah memiliki akun maka pengguna dapat langsung melanjutkan ke tahap *login* akun.

Setelah berhasil melakukan *login*, pengguna akan langsung menuju ke halaman utama aplikasi. Pada halaman utama ini tersedia dua fitur utama yang dapat diakses. Fitur pertama adalah fitur *profile* yang berfungsi untuk menampilkan halaman *profile*. Di dalam halaman ini, pengguna memiliki opsi untuk mengisi serta memperbarui biodata pengguna. Jika pengguna memutuskan untuk menyimpan perubahan, biodata tersebut akan di-*Save* Biodata. Namun, jika pengguna memilih untuk membatalkan atau kembali, maka akan dibawa keluar dari halaman profil dan dapat kembali ke halaman utama.

Fitur kedua adalah Fitur Verifikasi Suara. Pada fitur ini, aplikasi akan Menampilkan halaman verifikasi suara, di mana pengguna akan melalui tahap verifikasi suara. Hasil dari

tahap verifikasi ini akan menentukan respons sistem yang dimana jika verifikasi berhasil, sistem akan menampilkan respon selamat datang, yang mengindikasikan akses telah diberikan atau pintu berhasil dibuka. Akan tetapi, jika suara tidak dikenali, sistem akan menampilkan respon suara tidak dikenali.

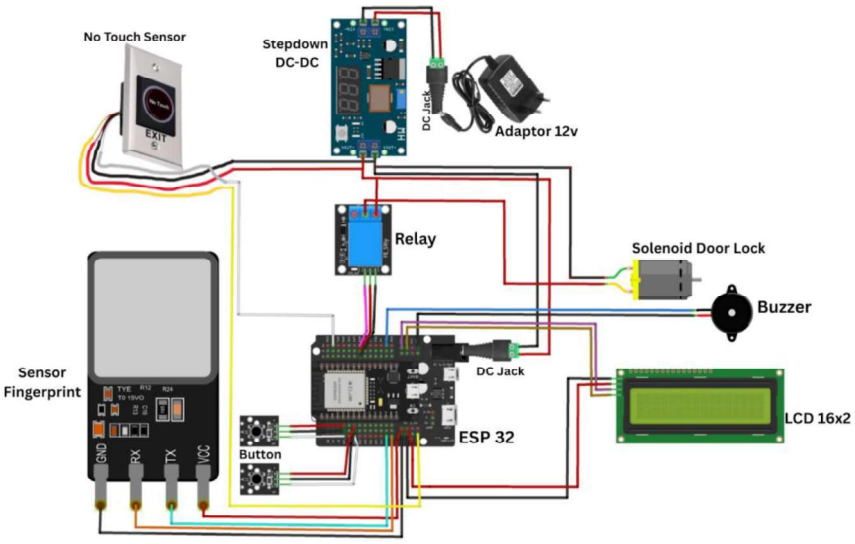
Fitur ketiga yang tersedia di halaman utama adalah fitur *admin*. Fitur ini dirancang khusus untuk administrator sistem. Untuk mengaksesnya, admin harus melalui halaman *login admin* terlebih dahulu untuk autentikasi. Setelah berhasil *login*, sistem akan langsung menampilkan data yang berupa total *user* dan *log* aktivitas mahasiswa yang melakukan verifikasi suara sehingga memungkinkan *admin* untuk memantau atau meninjau catatan aktivitas yang dilakukan oleh pengguna mahasiswa di dalam aplikasi.

4.2 Detil Implementasi

Pada detail implementasi ini akan memberikan penjelasan dari perancangan sistem *Smart Dorm Key* yang terdiri dari dua bagian utama yaitu perangkat keras dan perangkat lunak serta proses dan luaran terkait implementasi dari sistem *smart dorm key* secara detail.

4.2.1 Perangkat Keras

Smart Dorm Key menggunakan beberapa komponen utama dan komponen pendukung yang dirancang sedemikian rupa dan saling terintegrasi satu sama lain. Pada perangkat ini menggunakan mikrokontroler ESP32 sebagai pusat kontrol sistem dan beberapa sensor yang berfungsi untuk menjalankan sistem dari *Smart Dorm Key.*



Gambar 4.3 Skema *Wiring* Diagram *Smart Dorm Key*

Dalam perancangan perangkat keras yang akan digunakan, dilakukan sebuah skema untuk *wiring* diagram dari alat *Smart Dorm Key* yang dapat membantu dalam perancangan seperti

yang terlihat pada Gambar 4.3. Perancangan dimulai menggunakan adaptor 12v sebagai sumber tegangan yang kemudian dihubungkan dengan *stepdown* untuk mengatur seberapa besar tegangan yang dibutuhkan. ESP32 kemudian terhubung dengan sensor *fingerprint* untuk otentikasi sidik jari, modul *relay* yang mengatur kunci pintu *solenoid*, *buzzer*, *button*, serta LCD berfungsi sebagai alat untuk bisa berinteraksi dengan pengguna. Sistem ini juga menggunakan *No Touch Sensor* yang digunakan untuk membuka pintu dari dalam dengan menempelkan tangan dengan jarak maksimal 6cm.

4.2.1.1 Mikrokontroler ESP32

ESP32 adalah mikrokontroler yang berfungsi sebagai pusat kendali dalam sistem elektronik berbasis IoT, termasuk pada *Smart Dorm Key*. Dengan fitur konektivitas *Wi-Fi* dan *Bluetooth* bawaan, ESP32 mampu menghubungkan berbagai komponen seperti sensor *fingerprint*, LCD, *buzzer*, dan modul pengendali pintu, serta menerima input suara dari aplikasi eksternal. Selain itu, ESP32 memiliki performa pemrosesan yang cukup tinggi dan konsumsi daya yang rendah, menjadikannya ideal untuk sistem otomatisasi yang efisien dan responsif.

Gambar 4.4 Mikrokontroler ESP32

Sumber gambar : https://ozami.co.id/mengenal-esp32-mikrokontroler-iot/

4.2.1.2 Sensor Fingerprint AS608

Pada Gambar 4.5 merupakan sensor *fingerprint* AS608 modul pemindai sidik jari yang umum digunakan dalam berbagai proyek sistem keamanan berbasis mikrokontroler. Sensor ini memiliki kemampuan untuk mengambil, menyimpan, dan mencocokkan data sidik jari secara mandiri tanpa memerlukan pemrosesan eksternal yang kompleks,

Gambar 4.5 sensor *Fingerprint AS608*.

Sumber gambar : https://id.aliexpress.com/i/32825192875.html

Sensor *Fingerprint* AS608 bekerja dengan menggunakan optik untuk menangkap pola sidik jari, kemudian mengubahnya menjadi data digital yang dapat disimpan dalam memori internal sensor. Modul ini terhubung ke mikrokontroler seperti ESP32 melalui antarmuka UART (TX dan RX), dan menyediakan perintah-perintah standar untuk proses pendaftaran (*enrollment*), pencocokan (*matching*), serta penghapusan data sidik jari. Dengan ukuran yang ringkas dan akurasi yang cukup tinggi, AS608 sangat cocok digunakan dalam sistem akses kontrol seperti *Smart Dorm Key*.

```
#include <Adafruit_Fingerprint.h>
#include <Wire.h>
#include <WiFi.h>
// Informasi WiFi
const char* ssid = "PGAUniversity";
const char* password = "Minyakkita";

// Pin konfigurasi
#define FINGERPRINT_TX 16 //memberitahu kita pin mana pada mikrokontroler TX
#define FINGERPRINT_RX 17 //memberitahu kita pin mana pada mikrokontroler RX

// Objek
HardwareSerial fingerprintSerial(2); //objek komunikasi serial (UART) dibuat, menggunakan port serial ke-2 pada mikrokontroler.
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&fingerprintSerial); //untuk mengontrol sensor sidik jari dibuat di sini

// Status
uint8_t scanFingerprint(); //kode bahwa akan ada fungsi bernama scanFingerprint yang akan mengembalikan hasil scan

void setup() {  //mengaktifkan dan mengkonfigurasi komunikasi serial (fingerprintSerial) ke sensor.
  fingerprintSerial.begin(57600, SERIAL_8N1, FINGERPRINT_RX, FINGERPRINT_TX);
  delay(1000);
}

// Pindai sidik jari
void loop() {

  uint8_t result = scanFingerprint(); //Setiap kali loop berjalan, ia akan memanggil fungsi scanFingerprint()

  if (result == 1 && !doorOpen) { // struktur pengambilan keputusan pertama apakah sidik jari sesuai.
    openDoor();
  } else if (result == 2) { // struktur pengambilan keputusan pertama apakah sidik jari salah.
    Serial.println("Sidik jari salah!");
    if (doorOpen) lockDoor();
  }
}
```

Gambar 4.6 Potongan Program Sensor *Fingerprint AS608*

4.2.1.3 *Solenoid Door Lock*

Gambar 4.7 menampilkan *solenoid door lock* yang merupakan jenis kunci elektronik yang bekerja menggunakan prinsip elektromagnetik untuk mengunci dan membuka pintu secara otomatis. Ketika arus listrik dialirkan ke dalam *solenoid*, medan magnet yang dihasilkan akan menarik atau mendorong pin pengunci, sehingga memungkinkan pintu untuk terbuka.

Gambar 4.7 *Solenoid Door Lock*

Sumber gambar : https://www.blibli.com/p/solenoid-door-lock-12v-automatic-electronic-arduino-rfid-besar/ps--ALA-47809-00587

Dalam sistem *Smart Dorm Key*, *solenoid door lock* dikendalikan oleh ESP32, yang hanya akan mengaktifkan kunci jika kedua tahap verifikasi—*fingerprint* dan *voice recognition*—berhasil dilewati. Ketika tidak dialiri listrik, *solenoid* biasanya berada dalam posisi mengunci (*fail-secure*) atau membuka (*fail-safe*), tergantung jenisnya. Keunggulan dari kunci *solenoid* adalah responnya yang cepat, mekanisme yang sederhana, dan integrasinya yang mudah dengan sistem otomatis berbasis mikrokontroler.

```
#include <Wire.h>

// Pin konfigurasi
#define RELAY_PIN 26 //untuk memberi nama pin pada mikrokontroller

// Status
void lockDoor(); //deklarasi fungsi pintu tertutup
void openDoor(); //deklarasi fungsi pintu terbuka

void setup() { //deklarasi untuk mengontrol dan mengaktifkan relay
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
}

void loop() { // untuk melakukan penguncian pintu secara otomatis setelah dibuka
  if (doorOpen 77 (millis() -lastActionTime > LOCK_TIMEOUT)) {
    LOCKdOOR();
  }


void openDoor() { // instruksi yang dijalankan ketika kita ingin membuka pintu
  digitalWrite(RELAY_PIN, HIGH);
  doorOpen = true;
  lastActionTime = millis();
  Serial.println("Pintu TERBUKA");
}

void lockDoor() { // instruksi yang dijalankan ketika mengunci pintu
  digitalWrite(RELAY_PIN, LOW);
  doorOpen = false;
  Serial.println("Pintu TERKUNCI");
 }
}
```

Gambar 4.8 Potongan Program *Solenoid Door Lock.*

4.2.1.4 *Buzzer*

Pada gambar 4.9 *merupakan buzzer*, yaitu komponen yang digunakan untuk menghasilkan suara sebagai bentuk notifikasi atau peringatan dalam suatu sistem.

Gambar 4.9 *Buzzer*

Sumber gambar : https://indonesian.uttransducer.com/sale-13611395-12mm-dc-5v-2-terminals-loud-active-and-passive-piezo-buzzer.html

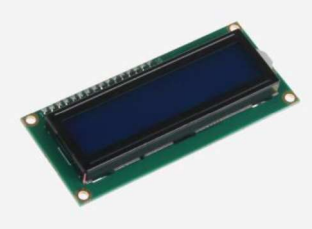
Dalam proyek *Smart Dorm Key*, *buzzer* berfungsi sebagai indikator audio yang memberi tahu pengguna tentang status verifikasi—baik itu berhasil maupun gagal. Misalnya, jika *fingerprint* atau *voice recognition* tidak dikenali, *buzzer* akan mengeluarkan bunyi pendek atau panjang sebagai tanda kesalahan. Sebaliknya, jika kedua tahap verifikasi sukses, *buzzer* akan mengeluarkan bunyi berbeda yang menandakan akses diterima. *Buzzer* yang digunakan biasanya tipe *piezoelektrik* atau elektromagnetik dan dapat dikendalikan langsung oleh mikrokontroler seperti ESP32 melalui sinyal digital.

```
#include <Wire.h>

// Pin konfigurasi
#define BUZZER_PIN 4 // Pin yang terhubung ke buzzer

// Status
void beepBuzzer(int duration, int delayTime); // Deklarasi Fungsi (memberitahu kompiler bahwa fungsi ini ada)
void beepBuzzerPattern();

void setup() { //mengirim sinyal keluar melalui pin ini untuk menyalakan atau mematikan buzzer.
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

void beepBuzzer(int duration, int delayTime) { // mengirimkan sinyal untuk mengatur buzzer
  digitalWrite(BUZZER_PIN, HIGH); // Nyalakan buzzer
  delay(delayTime); //Biarkan menyala selama 'delayTime'
  digitalWrite(BUZZER_PIN, LOW); // Matikan buzzer
  delay(duration - delayTime); // Biarkan mati selama sisa 'duration'
}

void beepBuzzerPattern() {
  for (int i = 0; i < 3; i++) { // Ulangi blok kode ini 3 kali
    digitalWrite(BUZZER_PIN, HIGH); // Nyalakan buzzer
    delay(200); // Biarkan menyala selama 200 milidetik
    digitalWrite(BUZZER_PIN, LOW); // Matikan buzzer
    delay(200); // Biarkan mati selama 200 milidetik
  }
}
```

Gambar 4.10 Potongan Program *Buzzer*

4.2.1.5 LCD 16x2

Gambar 4.11 menampilkan LCD 16x2 yang merupakan sebuah modul *display* yang terdiri dari dua baris dengan masing-masing 16 karakter, digunakan untuk menampilkan informasi dalam bentuk teks.



Gambar 4.11 LCD 16x2

Sumber gambar : https://translate.google.com/translate?u=https://joy-it.net/en/products/SBC-LCD16x2&hl=id&sl=en&tl=id&client=imgs

Dalam sistem *Smart Dorm Key*, LCD 16x2 berfungsi sebagai antarmuka visual untuk memberi tahu pengguna mengenai status sistem, seperti "Tempelkan Jari Anda", "Verifikasi

Berhasil", "Kata Sandi Salah", atau "Akses Ditolak". Modul ini menggunakan *controller HD44780* dan dapat dihubungkan ke mikrokontroler seperti ESP32 melalui sambungan paralel atau menggunakan modul I2C untuk menghemat pin. Keunggulan LCD 16x2 adalah kemudahan dalam penggunaan, konsumsi daya rendah, dan kemampuannya untuk menyampaikan informasi secara langsung kepada pengguna.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pin konfigurasi
#define SDA_PIN 21 //Mendefinisikan pin SDA untuk I2C adalah pin 21
#define SCL_PIN 22 //Mendefinisikan pin SCL untuk I2C adalah pin 22

// Objek
LiquidCrystal_I2C lcd(0x27, 16, 2); //Membuat objek untuk LCD I2C
HardwareSerial fingerprintSerial(2); //Membuat objek Serial untuk sensor sidik jari (UART2)

void setup() {
  Wire.begin(SDA_PIN, SCL_PIN); //Memulai komunikasi I2C dengan pin SDA dan SCL yang ditentukan
}
```

Gambar 4.12 Potongan Program LCD 16x2

4.2.1.6 *No Touch Exit Button*

Pada Gambar 4.13 menampilkan sebuah sebuah inovasi dalam sistem kontrol akses, yaitu *no Touch Exit Button* yang merupakan sebuah tombol keluaran berbasis sensor inframerah yang memungkinkan pengguna membuka pintu tanpa menyentuh permukaan fisik, cukup dengan mendekatkan tangan dalam jarak tertentu dan Implementasi teknologi ini bertujuan meningkatkan higienitas dan kemudahan akses bagi pengguna.

Gambar 4.13 *Sensor No Touch*

Sumber gambar : https://shopee.co.id/NO-TOUCH-BUTTON-ACCESS-DOOR-LIFT-TOUCHLESS-INFRARED-SENSOR-12V-24V-i.561590001.14821864024

Dalam sistem *Smart Dorm Key*, tombol ini digunakan untuk memudahkan penghuni asrama keluar ruangan secara higenis dan efisien, tanpa perlu menggunakan *fingerprint* atau *voice recognition* lagi. Saat tangan terdeteksi oleh sensor, tombol akan mengirim sinyal ke ESP32 untuk mengaktifkan *solenoid door lock* dan membuka pintu. Teknologi ini sangat cocok

digunakan di lingkungan yang mengutamakan kebersihan, seperti ruang publik atau kamar bersama, karena mengurangi kontak langsung yang berpotensi menyebarkan kuman atau virus.

```
#include <Wire.h>

// Pin konfigurasi
#define IR_SENSOR_PIN 33  // Mendefinisikan pin untuk sensor IR adalah pin 33

void setup() {
  pinMode(IR_SENSOR_PIN, INPUT_PULLUP);  // Setup pin sensor IR sebagai INPUT dengan PULLUP internal
}

// Sensor IR membuka pintu dari dalam
void loop() {
  if (digitalRead(IR_SENSOR_PIN) == HIGH && !doorOpen) {
    Serial.println("Sensor IR aktif: Buka pintu dari dalam"); // Cetak pesan
    openDoor(); // Panggil fungsi untuk membuka pintu
  }
}
```

Gambar 4.14 Potongan Program Sensor *No Touch*

4.2.1.7 *Button*

Pada Gambar 4.15 memperlihatkan sebuah *button module* yang merupakan komponen elektronik. Fungsi utama dari modul *button* ini untuk mendeteksi *input* berupa tekanan tombol fisik terutama yang melibatkan mikrokontroler, seperti *ESP-32*.

Gambar 4.15 *Button*

Sumber gambar : https://www.amazon.in/Luroze-Compatible-Personal-Computer-Instrument/dp/B0B1WHBZ54

Sistem *"Smart Dorm Key"* menggunakan dua *button* yang memiliki fungsi dalam pengelolaan sidik jari pengguna. *Button* 1 berfungsi untuk menambahkan sidik jari baru ke dalam sistem, memungkinkan *user* untuk mendaftarkan sidik jari mereka agar dapat mengakses pintu. Sementara itu, *button* 2 memiliki fungsi untuk menghapus sidik jari yang sudah terdaftar, berguna dalam kasus di mana pengguna perlu mencabut akses atau memperbarui data sidik jari mereka. Kedua *button* ini terhubung langsung ke mikrokontroler ESP32, yang memproses *input* dari *button* dan menjalankan fungsi sistem *Smart Dorm Key*. Dengan adanya kedua *button* ini, pengguna memiliki kontrol langsung untuk mengelola data sidik jari mereka tanpa harus melalui aplikasi atau antarmuka lain.

```
#include <Wire.h>

// Pin konfigurasi
#define INPUT_BUTTON_PIN 27 //Mendefinisikan pin untuk tombol input/daftar
#define DELETE_BUTTON_PIN 14 //Mendefinisikan pin untuk tombol hapus

// Status
void enrollFingerprint(); //Deklarasi fungsi untuk mendaftarkan sidik jari
void deleteAllFingerprints(); //Deklarasi fungsi untuk menghapus semua sidik jari

//Setup pin tombol sebagai INPUT dengan PULLUP internal
void setup() {
  pinMode(INPUT_BUTTON_PIN, INPUT_PULLUP);
  pinMode(DELETE_BUTTON_PIN, INPUT_PULLUP);
}

void loop() {
  // Tombol daftar
  if (enrollButtonPressed && (currentMillis - lastButtonPress > DEBOUNCE_DELAY)) {
    Serial.println("Daftarkan sidik jari...");
    enrollFingerprint(); //Panggil fungsi pendaftaran
    finger.getTemplateCount(); //Dapatkan jumlah template (perlu objek 'finger')
    lastButtonPress = currentMillis; //Update waktu tekan tombol terakhir
  }

  // Tombol hapus
  if (deleteButtonPressed && (currentMillis - lastButtonPress > DEBOUNCE_DELAY)) {
    Serial.println("Menghapus semua sidik jari...");
    deleteAllFingerprints(); //Panggil fungsi penghapusan
    delay(2000); //Beri jeda 2 detik
    lastButtonPress = currentMillis; //Update waktu tekan tombol terakhir
  }
}
```

Gambar 4.16 Potongan program *Button*

4.2.1.8 Power Adaptor 12v

Power Adaptor 12v (Adaptor Daya 12V) adalah komponen yang digunakan sebagai sumber tegangan utama dalam sistem "*Smart Dorm Key*". Adaptor ini menyediakan daya listrik dengan tegangan 12 volt yang kemudian akan didistribusikan ke berbagai komponen elektronik dalam sistem. Namun, karena beberapa komponen seperti mikrokontroler ESP32 dan sensor-sensor lainnya membutuhkan tegangan yang lebih rendah, tegangan 12 volt dari adaptor akan diturunkan menggunakan *stepdown LM2596* agar sesuai dengan kebutuhan masing-masing komponen. Dengan demikian, *Power Adaptor 12v* berfungsi sebagai sumber energi awal yang vital, memastikan sistem "*Smart Dorm Key*" dapat beroperasi dengan stabil dan efisien.

Gambar 4.17 *Power Adaptor 12V*

Sumber gambar : https://jadistore.com/product/adaptor-power-12v-1a-switching-power-supply-adapter/

4.2.1.9 *Stepdown* LM2596

Pada gambar 4.18 menampilkan sebuah *stepdown* LM2596 yang menjadi salah satu komponen yang digunakan pada sistem *smart dorm key*.

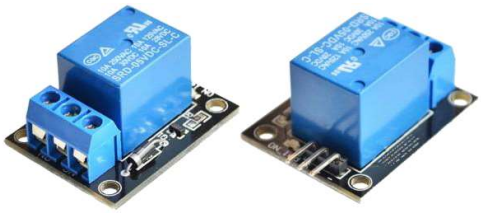
Gambar 4.18 *Stepdown LM2596*

Sumber gambar : https://botland.store/converters-step-down/4871-step-down-voltage-regulator-with-display-lm2596-32v-35v-3a-5904422359560.html

Modul *StepDown* LM2596 DC-DC merupakan konverter untuk menurunkan tingkat tegangan masukan DC menjadi tingkat tegangan yang lebih rendah daripada tegangan masukan. *Stepdown LM2596* digunakan dalam perancangan perangkat keras "*Smart Dorm Key*" berbasis *fingerprint* dan *voice recognition*. Pada sistem ini menggunakan *power adaptor* 12V yang digunakan sebagai sumber tegangan utama, namun komponen-komponen elektronik seperti ESP32 dan sensor-sensor lainnya membutuhkan tegangan yang lebih rendah. *Stepdown LM2596* berfungsi menurunkan tegangan dari 12V menjadi tegangan yang sesuai kebutuhan komponen-komponen tersebut, memastikan daya yang tepat dan stabilitas sistem.

4.2.1.10 *Relay*

Gambar 4.19 menampilkan *Modul Relay 5V* yang merupakan sebuah komponen dengan fungsi sebagai sakelar listrik yang dapat dikontrol secara elektronik melalui *mikrokontroler*.



Gambar 4.19 *Relay*

Sumber gambar : https://apmonitor.com/dde/index.php/Main/ElectromagneticRelay

Dalam sistem *"Smart Dorm Key"*, *relay* memainkan peran kunci sebagai antarmuka. Ia terhubung antara *mikrokontroler* ESP32 dan komponen *solenoid door lock.* Fungsi utama relay adalah sebagai saklar yang dioperasikan secara elektrik. *Relay* ini bertugas mengendalikan aktivasi dari *solenoid door lock* tersebut. Proses dimulai saat ESP32 melakukan verifikasi sidik jari dan suara pengguna. Jika kedua verifikasi tersebut berhasil dan pengguna dinyatakan sah, ESP32 akan mengirimkan sinyal *output* logika ke terminal input *relay*. Menerima sinyal ini, *relay* akan mengubah kondisi kontaknya, mengaktifkan *solenoid*. *Solenoid door lock* kemudian mendapatkan daya listrik dan membuka kunci pintu. Sebaliknya, jika salah satu atau kedua verifikasi mengalami kegagalan, ESP32 tidak akan mengirimkan sinyal pemicu tersebut ke *relay*. Akibatnya, *relay* tidak akan aktif, dan *solenoid door lock* pun tetap mati. Pintu akan tetap dalam kondisi terkunci, menjaga keamanan akses.

```
#include <Wire.h>

// Pin konfigurasi
#define RELAY_PIN 26 //Mendefinisikan pin untuk relay adalah pin 26

// Deklarasi Fungsi
void lockDoor();
void openDoor();

void setup() {
  pinMode(RELAY_PIN, OUTPUT); //Atur RELAY_PIN sebagai OUTPUT
  digitalWrite(RELAY_PIN, LOW); //Pastikan relay mati (pintu terkunci) saat awal
}

void loop() {
  // Timeout otomatis kunci pintu
  if (doorOpen && (millis() - lastActionTime > LOCK_TIMEOUT)) {
    lockDoor(); //Jika ya, panggil fungsi untuk mengunci pintu
  }

void openDoor() {
  digitalWrite(RELAY_PIN, HIGH); //Aktifkan relay (misalnya, HIGH membuka kunci)
  doorOpen = true; //Set status pintu menjadi terbuka
  lastActionTime = millis(); //Catat waktu saat pintu dibuka
  Serial.println("Pintu TERBUKA");
  }

void lockDoor() {
  digitalWrite(RELAY_PIN, LOW); //Matikan relay (misalnya, LOW mengunci pintu)
  doorOpen = false; //Set status pintu menjadi terkunci
  Serial.println("Pintu TERKUNCI");
  }
}
```

Gambar 4.20 Potongan Program *relay*

4.2.2 Perangkat Lunak

4.2.2.1 Arduino IDE

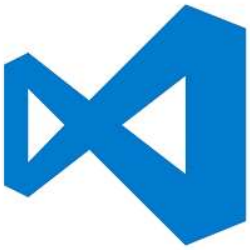
Arduino IDE (*Integrated Development Environment*) adalah perangkat lunak resmi yang digunakan untuk menulis, mengedit, dan mengunggah kode program ke papan mikrokontroler seperti Arduino maupun ESP32. Dalam proyek *Smart Dorm Key*, Arduino IDE digunakan untuk memprogram ESP32 agar dapat mengontrol berbagai komponen seperti sensor *fingerprint*, *solenoid door lock*, *buzzer*, LCD 16x2, dan menerima *input* dari aplikasi *voice recognition*. Arduino IDE menyediakan antarmuka yang sederhana dengan bahasa pemrograman berbasis C/C++, serta berbagai pustaka (*library*) yang memudahkan integrasi perangkat keras. Selain itu, Arduino IDE mendukung *debugging* dasar dan komunikasi serial yang sangat berguna dalam pengujian sistem secara *real-time*.

Gambar 4.21 Logo *Arduino IDE*

Sumber gambar : https://worldvectorlogo.com/logo/arduino-1

4.2.2.2 Visual Studio Code

Visual Studio Code adalah editor kode sumber (*source code editor*) gratis dan ringan yang dikembangkan oleh Microsoft. VS Code mendukung banyak bahasa pemrograman dan dilengkapi dengan berbagai fitur canggih yang menjadikannya pilihan populer di kalangan *developer*. Dalam proyek *Smart Dorm Key* itu sendiri, *Visual Studio Code* digunakan sebagai Aplikasi pemrograman untuk membuat dan menghubungkan *API*.

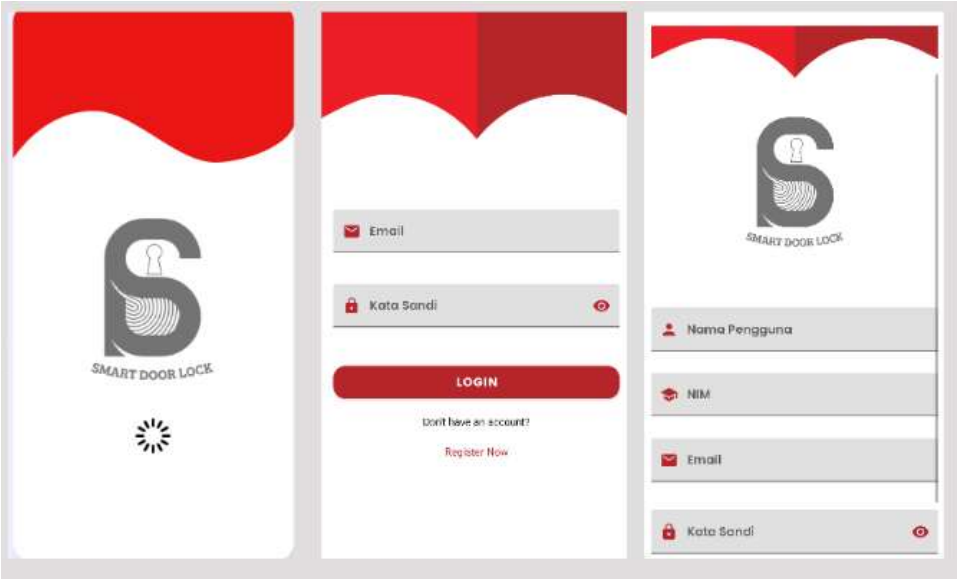


Gambar 4.22 Logo *Visual Studio Code*

Sumber gambar : https://www.clipartmax.com/middle/m2i8Z5b1i8Z5i8H7_visual-studio-code-logo/

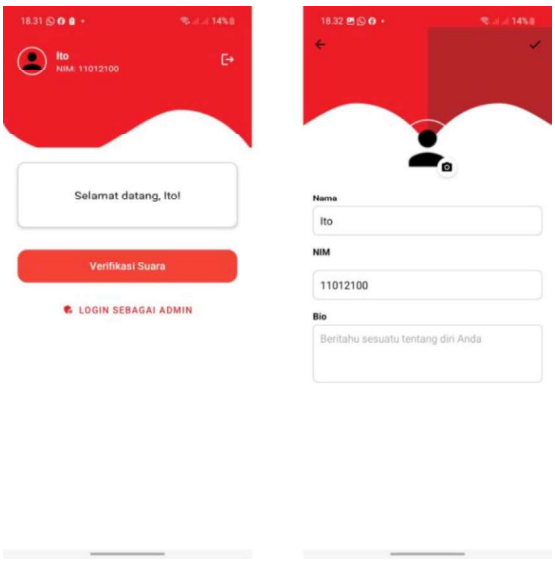
4.2.2.3 Aplikasi

Aplikasi adalah program atau perangkat lunak yang dirancang untuk membantu pengguna dalam melakukan tugas atau aktivitas tertentu. Aplikasi biasanya dibangun untuk memenuhi kebutuhan spesifik dalam kehidupan sehari-hari, baik itu untuk penggunaan pribadi, profesional, maupun pendidikan.



Gambar 4.23 Tampilan awal Aplikasi

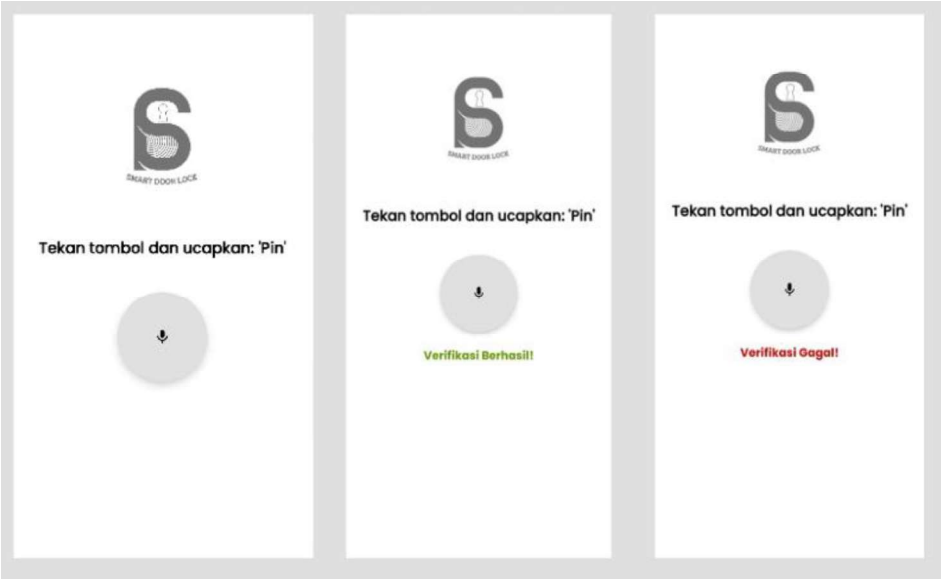
Pada gambar 4.23 merupakan tampilan awal aplikasi yang ditunjukkan kepada pengguna sebelum ke halaman Utama. Pada gambar sebelah kiri merupakan logo aplikasi yang akan muncul ketika pengguna membuka aplikasi. Pada gambar yang ada ditengah menampilkan halaman selanjutnya setelah animasi logo selesai, yaitu halaman bagi pengguna untuk melakukan *login* ketika sudah mempunyai akun, akan tetapi jika pengguna belum memiliki akun, maka pengguna dapat melakukan registrasi seperti yang ditampilkan pada gambar sebelah kiri dengan mengisi biodata diri.



Gambar 4.24 Tampilan Dalam Aplikasi

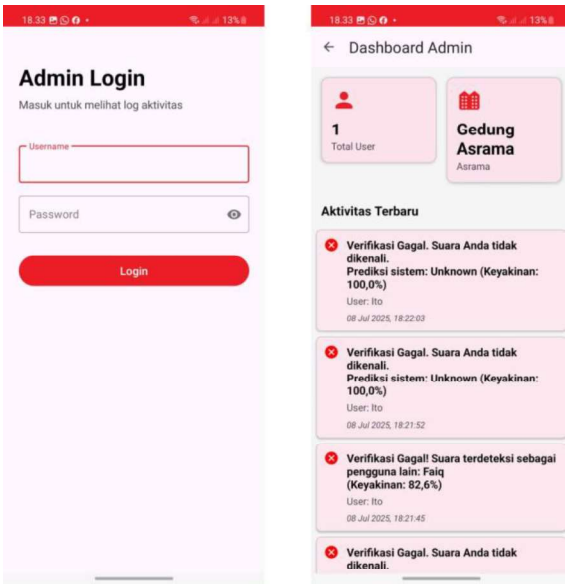
Pada Gambar 4.24 terdapat dua halaman gambar. Pada gambar sebelah kiri menunjukkan tampilan halaman utama dari aplikasi setelah berhasil *login* dan jika menekan

profil yang ada di kiri atas, maka akan beralih ke halaman profil seperti gambar yang ada di sebelah kanan, yang di mana halaman profil berfungsi untuk melengkapi profil. Halaman utama berfungsi untuk meneruskan tahap verifikasi suara yang akan dilakukan nanti dan juga pada halaman utama terdapat *button* untuk melakukan *login* sebagai *admin*, dimana untuk akun dari *admin* sendiri sudah didaftarkan melalui *local database.*



Gambar 4.25 Tampilan Verifikasi Suara

Pada gambar 4.25 adalah, halaman verifikasi suara aplikasi *Smart Door Lock*, yang merupakan fitur inti untuk mengotentikasi pengguna melalui perintah suara. Kondisi awal halaman verifikasi (kiri) menginstruksikan pengguna dengan tulisan Tekan tombol dan ucapkan: Pin di bawah logo aplikasi, disertai tombol mikrofon besar di tengah layar yang siap menerima input suara. Selanjutnya, skenario ketika verifikasi suara berhasil (tengah) digambarkan dengan munculnya teks Verifikasi Berhasil berwarna hijau di bawah tombol mikrofon setelah pengguna mengucapkan kata kunci yang benar, menandakan akses diberikan atau pintu telah terbuka. Sebaliknya, jika verifikasi suara gagal karena pengguna mengucapkan kata kunci yang salah atau suara tidak dikenali (kanan), akan tampil teks Verifikasi Gagal berwarna merah di bawah tombol mikrofon, yang mengindikasikan bahwa akses ditolak.



Gambar 4.26 Tampilan *Logbook Admin*

Pada Gambar 4.26 merupakan tampilan ketika melakukan *login* sebagai *admin* yang ada pada halaman utama aplikasi. Pada gambar sebelah kiri merupakan halaman untuk memasukkan akun yang berupa *username* dan *password* yang telah didaftarkan. Pada gambar sebelah kanan merupakan tampilan setelah *admin* berhasil melakukan *login*, maka akan muncul halaman yang menampilkan total *user* yang terdaftar dan gedung asrama serta inti dari halaman ini, yaitu tampilan log aktivitas mahasiswa ketika memasukkan verifikasi suara. Tampilan dari *log* ini sendiri berupa verifikasi suara yang gagal dan berhasil, tanggal ketika mahasiswa melakukan verifikasi suara serta waktu mahasiswa melakukan verifikasi suara. Dengan adanya *log* aktivitas ini tentunya akan memudahkan penjaga asrama untuk memantau kapan mahasiswa kembali ke kamar asrama.

4.2.2.4 Kotlin

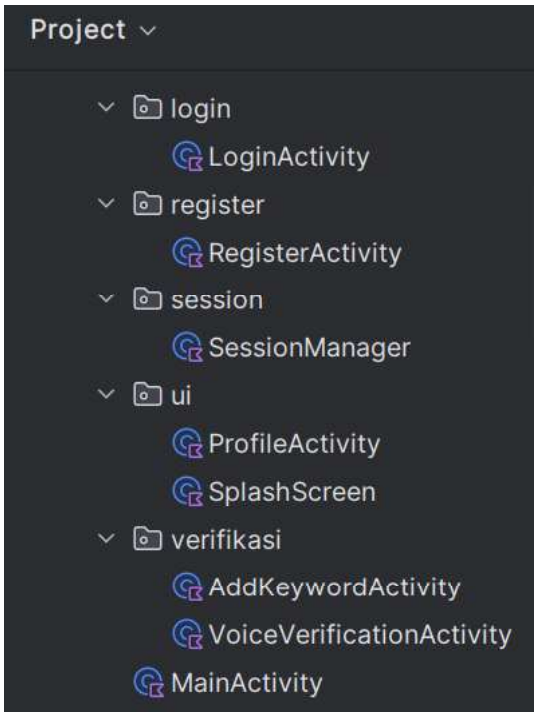
Pada Gambar 4.27 menampilkan tampilan dari kotlin.

Gambar 4.27 Logo Kotlin

Sumber gambar : https://www.pngwing.com/en/free-png-arbnl

Kotlin adalah bahasa pemrograman yang berjalan di atas *Java Virtual Machine* (*JVM*), dan dikembangkan oleh *JetBrains*, perusahaan yang dikenal dengan produk IDE seperti *IntelliJ IDEA*. Kotlin dirancang untuk menjadi lebih modern, ekspresif, dan aman dibandingkan Java,

serta sepenuhnya kompatibel dengan Java. *Platform* Kotlin menyediakan berbagai fitur yang menjadikannya pilihan populer dalam pengembangan aplikasi *Android* dan aplikasi *backend*. Dalam proyek *Smart Dorm Key* itu sendiri, Kotlin digunakan sebagai bahasa pemrograman untuk membuat aplikasi *Android*.



Gambar 4.28 *Project Activity* Aplikasi

A. Halaman Pendukung

- *Main Activity*

  Pada Tabel 4.1 terdapat dua potongan kode. Untuk kode pertama mendefinisikan *MainActivity*, layar utama aplikasi Android, yang berisi berbagai elemen antarmuka pengguna dan pengelola sesi. Terdapat *CircleImageView* untuk foto profil (*imgProfile*), *ImageView* untuk latar belakang (*imgBackground*), beberapa *TextView* untuk menampilkan nama (*tvName*), NIM (*tvNIM*), dan pesan selamat datang (*tvWelcome*). Ada juga tiga tombol: *Button* untuk verifikasi (*btnVerify*), *ImageButton* untuk *logout* (*btnLogout*), dan *Button* lain untuk fungsi "Tambah Suara" (*btnTambahSuara*). Terakhir, *SessionManager* adalah objek yang akan menangani data sesi pengguna.

  Potongan kode kedua ini adalah bagian dari metode *onCreate* di *MainActivity*, yang dijalankan saat layar pertama kali dibuat. Anotasi

@SuppressLint digunakan untuk mengabaikan peringatan terkait internasionalisasi teks. Di dalam metode ini, pertama-tama layout antarmuka pengguna (*activity_main.xml*) ditetapkan sebagai tampilan layar. Kemudian, *SessionManager* diinisialisasi untuk mengelola sesi pengguna. Setelah itu, semua variabel untuk komponen UI (seperti gambar profil, teks nama, dan tombol-tombol) dihubungkan dengan elemen-elemen yang ada di *file layout* menggunakan *findViewById*. Terakhir, kode ini memeriksa status *login* pengguna melalui *sessionManager*; jika pengguna ternyata belum *login*, ia akan diarahkan ke layar *login* (*redirectToLogin*()) dan eksekusi lebih lanjut di *onCreate* ini dihentikan.

Tabel 4.1 Potongan Kode *main activity*

```
class MainActivity : AppCompatActivity() {


    private lateinit var imgProfile: CircleImageView
    private lateinit var imgBackground: ImageView
    private lateinit var tvName: TextView
    private lateinit var tvNIM: TextView
    private lateinit var tvWelcome: TextView
    private lateinit var btnVerify: Button
    private lateinit var btnLogout: ImageButton
    private lateinit var btnTambahSuara: Button
    private lateinit var sessionManager: SessionManager
```

```
@SuppressLint("SetTextI18n")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)


    sessionManager = SessionManager(this)


    // Inisialisasi View
    imgProfile = findViewById(R.id.imgProfile)
    imgBackground = findViewById(R.id.imgBackground)
    tvName = findViewById(R.id.tvName)
    tvNIM = findViewById(R.id.tvNIM)
    tvWelcome = findViewById(R.id.tvWelcome)
    btnVerify = findViewById(R.id.btnVerify)
```

```
        btnLogout = findViewById(R.id.btnLogout)
        btnTambahSuara = findViewById(R.id.btnTambahSuara)


        if (!sessionManager.isLoggedIn()) {
            redirectToLogin()
            return
        }
```

- *Profile Activity*

Pada Tabel 4.2 terdapat dua potongan kode. Tentu. Kode tersebut mendefinisikan *ProfileActivity*, sebuah layar dalam aplikasi *Android* yang kemungkinan digunakan untuk menampilkan dan mengedit profil pengguna. Di dalamnya, dideklarasikan berbagai komponen antarmuka seperti *ImageView* untuk tombol kembali (*backButton*) dan simpan (*saveButton*), *CircleImageView* untuk foto profil (*profileImage*), dan *ImageView* lain untuk memicu unggah gambar (*uploadImage*). Terdapat juga beberapa *EditText* untuk input nama (*userName*), bio (*userBio*), dan NIM (*userNim*). Selain itu, ada *SessionManager* untuk mengelola data sesi pengguna, dan variabel *currentImageUri* untuk menyimpan URI gambar yang dipilih. Di dalam *companion object*, didefinisikan konstanta *PICK_IMAGE_REQUEST* yang biasanya digunakan sebagai kode permintaan saat memilih gambar dari galeri.

Kode Kedua merupakan metode *onCreate* dalam sebuah Aktivitas (*Activity*) *Android*. Saat Aktivitas ini dibuat, metode ini akan dipanggil. Pertama, ia memanggil implementasi *onCreate* dari kelas induknya (*super.onCreate(savedInstanceState)*), kemudian mengatur tata letak antarmuka pengguna (UI) dengan *setContentView(R.layout.activity_profile)*. Selanjutnya, kode ini menginisialisasi *SessionManager* untuk manajemen sesi pengguna, dan menghubungkan variabel-variabel seperti *backButton*, *saveButton*, *profileImage, uploadImage*, *userName*, *userBio*, dan *userNim* dengan elemen-elemen yang ada di *file layout* XML melalui *findViewById*. Setelah itu, fungsi *loadProfileData*() dipanggil, yang kemungkinan bertugas untuk memuat data profil pengguna. Terakhir, sebuah *OnClickListener* dipasang pada *backButton* sehingga ketika tombol tersebut diklik, aplikasi akan kembali ke layar sebelumnya menggunakan *onBackPressedDispatcher.onBackPressed()*.

Tabel 4.2 Potongan Kode *profile activity*

```
class ProfileActivity : AppCompatActivity() {


    private lateinit var backButton: ImageView
    private lateinit var saveButton: ImageView
    private lateinit var profileImage: CircleImageView
    private lateinit var uploadImage: ImageView
    private lateinit var userName: EditText
    private lateinit var userBio: EditText
    private lateinit var userNim: EditText
    private lateinit var sessionManager: SessionManager
    private var currentImageUri: String? = null


    companion object {
        private const val PICK_IMAGE_REQUEST = 1
    }
```

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


    sessionManager = SessionManager(this)
    backButton = findViewById(R.id.back_button)
    saveButton = findViewById(R.id.save_button)
    profileImage = findViewById(R.id.profile_image)
    uploadImage = findViewById(R.id.upload_image)
    userName = findViewById(R.id.user_name)
    userBio = findViewById(R.id.user_bio)
    userNim = findViewById(R.id.user_nim)


loadProfileData()



    backButton.setOnClickListener {
        onBackPressedDispatcher.onBackPressed()
    }
```

- *Splash Screen*

Pada Tabel 4.3 merupakan kode yang mendefinisikan *SplashScreen*, sebuah Aktivitas (*Activity*) dalam aplikasi *Android* yang berfungsi sebagai layar pembuka. Di dalam metode *onCreate*, yang dipanggil saat Aktivitas dibuat, tata letak antarmuka dari *R.layout.activity_splash_screen* diterapkan. Kemudian, sebuah *VideoView* dengan *ID videoSplash* diinisialisasi dari *layout*. Kode selanjutnya menentukan lokasi video yang akan diputar, yaitu *splashvideo* yang tersimpan dalam *folder raw* sumber daya aplikasi, dengan membuat sebuah URI. URI video ini kemudian diatur ke *VideoView*. Akhirnya, sebuah *setOnPreparedListener* ditambahkan ke *VideoView*; *listener* ini akan memastikan video mulai diputar *(videoView.start())* secara otomatis segera setelah *VideoView* siap untuk memainkannya.

Tabel 4.3 Potongan Kode *splash screen*

```
class SplashScreen : AppCompatActivity() {


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


        val videoView: VideoView =
findViewById(R.id.videoSplash)


        // Uri video dari folder raw
        val videoUri =
Uri.parse("android.resource://${packageName}/${R.raw.splashvideo}")
        videoView.setVideoURI(videoUri)


        // Mulai putar video
        videoView.setOnPreparedListener {
            videoView.start()
        }
```

B. Halaman Autentikasi

- *Login Activity*

Pada Tabel 4.4 terdapat dua kode. Kode pertama untuk *LoginActivity* di aplikasi *Android*. Saat aktivitas ini dibuat (*onCreate*), ia pertama-tama mengatur tampilan antarmukanya dari *R.layout.activity_login* dan menginisialisasi *SessionManager*. Hal yang paling penting di sini adalah pengecekan status *login* pengguna: jika *sessionManager.isLoggedIn()* mengembalikan *true* (artinya pengguna sudah *login* sebelumnya), aplikasi akan langsung mengarahkan pengguna ke *MainActivity* menggunakan *Intent* dan kemudian memanggil *finish*() untuk menutup *LoginActivity* ini. Ini mencegah pengguna kembali ke halaman *login* setelah berhasil masuk. Jika pengguna belum *login*, kode setelah *blok if* akan dieksekusi, yang biasanya berisi logika untuk menampilkan *form login*.

Kode kedua Fungsi *performLogin* ini bertugas untuk melakukan proses *login* pengguna dengan mengirimkan kredensial (*email* dan *password*) ke *server*. Pertama, fungsi ini membuat objek JSON yang berisi *email* dan *password*. Kemudian, objek JSON tersebut dikonversi menjadi *RequestBody* dengan tipe media *"application/json"*. Selanjutnya, sebuah permintaan HTTP POST (Request) dibuat, menargetkan URL *https://backendapi-roan.vercel.app/auth/login*, dengan *RequestBody* yang telah dibuat sebelumnya dan header "*Content-Type*" yang disetel ke "*application/json*". Permintaan ini kemudian dieksekusi secara *asynchronous* menggunakan *client.newCall(request).enqueue()*. Jika terjadi kegagalan koneksi (misalnya, tidak ada internet atau *server* tidak merespons), bagian *onFailure* akan dijalankan: *ProgressBar* akan disembunyikan, pesan *error* akan dicatat (*log*), dan sebuah pesan *Toast* akan ditampilkan kepada pengguna yang memberitahukan bahwa koneksi ke *server* gagal.

Tabel 4.4 Potongan Kode *login activity*

```
class LoginActivity : AppCompatActivity() {
    private val client = OkHttpClient()
    private lateinit var sessionManager: SessionManager
    override fun onCreate(savedInstanceState: Bundle?)
{
        super.onCreate(savedInstanceState)
```

```
            setContentView(R.layout.activity_login)

            sessionManager = SessionManager(this)

            if (sessionManager.isLoggedIn()) {

                startActivity(Intent(this,
MainActivity::class.java))
                finish()
                return
            }
```

```
    private fun performLogin(email: String, password:
String, progressBar: ProgressBar) {
        val json = JSONObject().apply {
            put("email", email)
            put("password", password)
        }

        val mediaType = "application/json".toMediaType()
        val requestBody =
json.toString().toRequestBody(mediaType)

        val request = Request.Builder()
            .url("https://backendapi-
roan.vercel.app/auth/login")
            .post(requestBody)
            .addHeader("Content-Type", "application/json")
            .build()

        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e:
IOException) {
                runOnUiThread {
                    progressBar.visibility = View.GONE
                    Log.e("LoginError", "Koneksi gagal:
${e.message}")
                    Toast.makeText(
                        this@LoginActivity,
```

```
                        "Gagal terhubung ke server. Cek
koneksi internet Anda.",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }
```

- *Register Activity*

Pada Tabel 4.5 terdapat dua kode. Kode pertama Kode ini adalah bagian dari *Register Activity* dalam aplikasi *Android*. Saat aktivitas ini dimulai (*onCreate*), ia pertama-tama mengatur tampilan antarmukanya menggunakan *layout R.layout.activity_register.* Kemudian, kode ini menginisialisasi berbagai elemen antarmuka pengguna (UI) yang ada di *layout* tersebut: sebuah *ImageButton* (*btnBackToLogin*) untuk kembali ke halaman *login*, beberapa *TextInputEditText* untuk *input* nama (*edtName*), NIM (*edtNim*), *email* (*edtEmail*), dan *password* (*edtPassword*), sebuah *Button* (*btnRegister*) untuk melakukan *registrasi*, serta sebuah *ProgressBar* untuk menunjukkan proses yang sedang berjalan. Terakhir, sebuah *OnClickListener* dipasang pada *btnBackToLogin*, yang berarti akan ada aksi tertentu yang dijalankan ketika tombol kembali ini diklik, meskipun aksi spesifiknya belum terlihat dalam potongan kode yang diberikan.

Untuk kode kedua fungsi *registerUser* ini bertugas untuk mendaftarkan pengguna baru dengan mengirimkan data pengguna (*username*, NIM, *email*, dan *password*) ke *server*. Pertama, fungsi ini membuat sebuah objek JSON yang berisi semua data pengguna tersebut. Objek JSON ini kemudian dikonversi menjadi *RequestBody* dengan tipe media "*application/json; charset=utf-8*". Selanjutnya, sebuah permintaan HTTP POST (*Request*) dibuat, yang ditujukan ke URL *https://backendapi-roan.vercel.app/auth/register, dengan RequestBody* yang sudah disiapkan dan header "*Content-Type*" yang disetel ke *"application/json"*. Permintaan ini lalu dieksekusi secara *asynchronous* menggunakan *client.newCall(request).enqueue()*. Jika terjadi kegagalan saat koneksi ke *server* (misalnya, tidak ada internet atau *server* tidak bisa dijangkau), bagian *onFailure* akan dijalankan: *ProgressBar* akan disembunyikan, dan

sebuah pesan *Toast* akan ditampilkan kepada pengguna yang memberitahukan bahwa koneksi ke *server* gagal, beserta pesan *error* spesifiknya.

Tabel 4.5 Potongan Kode *register activity*

```
class RegisterActivity : AppCompatActivity() {
    private val client = OkHttpClient()

    private lateinit var edtNim: TextInputEditText

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

        val btnBackToLogin: ImageButton =
findViewById(R.id.btn_back_to_login)
        val edtName: TextInputEditText =
findViewById(R.id.edt_name)
        edtNim = findViewById(R.id.edt_nim)         val
edtEmail: TextInputEditText = findViewById(R.id.edt_Email)
        val edtPassword: TextInputEditText =
findViewById(R.id.edt_Password)
        val btnRegister: Button =
findViewById(R.id.btn_register)
        val progressBar: ProgressBar =
findViewById(R.id.progress_bar)

        btnBackToLogin.setOnClickListener {
            finish()
        }
```

```
    private fun registerUser(username: String, nim: String,
email: String, password: String, progressBar: ProgressBar)
{
    val json = JSONObject().apply {
        put("username", username)
        put("nim", nim)
        put("email", email)
        put("password", password)
    }
```

```
        val mediaType = "application/json; charset=utf-
f".toMediaType()
    val                    requestBody                    =
json.toString().toRequestBody(mediaType)

        val request = Request.Builder()
            .url("https://backendapi-
roan.vercel.app/auth/register")
            .post(requestBody)
            .addHeader("Content-Type", "application/json")
            .build()

        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e:
IOException) {
                runOnUiThread {
                    progressBar.visibility = View.GONE
                    Toast.makeText(
                        this@RegisterActivity,
                        "Gagal terhubung ke server:
${e.message}",
                        Toast.LENGTH_LONG
                    ).show()
                }
            }
```

- *Session Manager*

Pada Tabel 4.6 terdapat dua kode. Kode pertama ini adalah kelas *SessionManager* dalam Kotlin untuk aplikasi *Android*, yang berfungsi untuk mengelola dan menyimpan data sesi pengguna secara lokal di perangkat. Kelas ini menggunakan *SharedPreferences* (disetel sebagai prefs) untuk menyimpan data secara persisten. Di dalam *companion object*, didefinisikan berbagai kunci (konstanta) seperti *PREFS_NAME (nama file preferensi), USER_TOKEN, USER_EMAIL, USER_NAME, USER_NIM, dan IS_LOGIN* untuk menyimpan informasi terkait *login* dan data pengguna. Selain itu, ada juga kunci untuk data profil seperti *PROFILE_NAME, PROFILE_BIO, PROFILE_NIM, dan*

PROFILE_IMAGE_URI yang digunakan untuk menyimpan detail profil pengguna.

Untuk Kode kedua fungsi *saveLoginSession* ini bertujuan untuk menyimpan informasi sesi *login* pengguna ke dalam *SharedPreferences*. Ketika dipanggil, fungsi ini menerima empat parameter: token (token autentikasi), *email* pengguna, *name* (nama pengguna), dan nim (Nomor Induk Mahasiswa). Menggunakan *prefs.edit*(), fungsi ini menyimpan setiap nilai tersebut ke *SharedPreferences* dengan kunci yang telah didefinisikan sebelumnya (seperti *USER_TOKEN, USER_EMAIL, USER_NAME, dan USER_NIM*). Selain itu, ia juga menyimpan nilai *true* untuk kunci *IS_LOGIN,* menandakan bahwa pengguna telah berhasil *login*. Terakhir, *apply()* dipanggil untuk menyimpan perubahan ini secara asinkron ke *SharedPreferences.*

Tabel 4.6 Potongan Kode *session manager*

```
    class SessionManager(context: Context) {
        private var prefs: SharedPreferences =
context.getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE)

        companion object {
            const val PREFS_NAME = "user_session"
            const val USER_TOKEN = "user_token"
            const val USER_EMAIL = "user_email"
            const val USER_NAME = "user_name"
            const val USER_NIM = "user_nim"
            const val IS_LOGIN = "is_login"

            const val PROFILE_NAME = "profile_name"
            const val PROFILE_BIO = "profile_bio"
            const val PROFILE_NIM = "profile_nim"
            const val PROFILE_IMAGE_URI =
"profile_image_uri"
        }
```

```
    fun saveLoginSession(token: String, email: String,
name: String, nim: String) {
        with(prefs.edit()) {
```

```
            putString(USER_TOKEN, token)
            putString(USER_EMAIL, email)
            putString(USER_NAME,  name)  //  Nama  dari  sesi
login
            putString(USER_NIM, nim)   // NIM dari sesi login
            putBoolean(IS_LOGIN, true)
            apply()
        }
    }
```

C. Halaman Verifikasi

- *Add Keyword Activity*

Pada Tabel 4.7 terdapat dua kode. Kode pertama Kode ini mendefinisikan kelas *AddKeywordActivity* untuk sebuah layar (*Activity*) di aplikasi *Android*, yang kemungkinan besar berfungsi untuk menambahkan kata kunci melalui input suara. Di dalamnya, dideklarasikan beberapa variabel privat: tvStatus (sebuah *TextView* untuk menampilkan status), *btnRecord* (sebuah *ImageButton* untuk memulai atau menghentikan perekaman suara), dan *btnSaveKeyword* (sebuah *Button* untuk menyimpan kata kunci). Selain itu, ada *speechRecognizer* untuk fungsionalitas pengenalan suara, *recognizedKeywordText* untuk menyimpan teks yang dikenali, dan *isListening* sebagai penanda apakah perekaman sedang berlangsung. Terdapat juga konstanta *RECORD_AUDIO_PERMISSION_CODE* untuk kode permintaan izin rekaman *audio* dan TAG untuk *logging*.

Untuk kode kedua Fungsi *startListeningForKeyword* ini bertugas untuk memulai proses perekaman dan pengenalan suara untuk mendapatkan sebuah kata kunci. Pertama, ia akan mengosongkan *recognizedKeywordText* yang mungkin berisi teks sebelumnya dan menonaktifkan tombol *btnSaveKeyword*. Kemudian, sebuah Intent disiapkan menggunakan *RecognizerIntent.ACTION_RECOGNIZE_SPEECH* yang dikonfigurasi untuk model bahasa bebas *(LANGUAGE_MODEL_FREE_FORM)*, bahasa Indonesia *(Locale("id", "ID"))*, dan menampilkan *prompt* "Ucapkan keyword Anda...". Setelah itu, *speechRecognizer.startListening(intent)* dipanggil untuk memulai

proses mendengarkan. Status pada tvStatus diubah menjadi "Mendengarkan...", ikon pada *btnRecord* diubah menjadi ikon jeda (*pause*), dan variabel *isListening* disetel menjadi *true* untuk menandakan bahwa proses perekaman sedang aktif.

Tabel 4.7 Potongan Kode halaman verifikasi

	`` class AddKeywordActivity : AppCompatActivity() { private lateinit var tvStatus: TextView private lateinit var btnRecord: ImageButton private lateinit var btnSaveKeyword: Button private lateinit var speechRecognizer: SpeechRecognizer private var recognizedKeywordText: String? = null private var isListening = false private val RECORD_AUDIO_PERMISSION_CODE = 101 private val TAG = "AddKeywordActivity" } ```
	`` private fun startListeningForKeyword() { recognizedKeywordText = null btnSaveKeyword.isEnabled = false val intent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH).apply { putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM) putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale("id", "ID")) // Bahasa Indonesia putExtra(RecognizerIntent.EXTRA_PROMPT, "Ucapkan keyword Anda...") } speechRecognizer.startListening(intent) tvStatus.text = "Mendengarkan..." btnRecord.setImageResource(android.R.drawable.ic_media_pause) isListening = true } ```

- *Voice Verification Activity*

Pada Tabel 4.8 terdapat dua kode. Kode pertama Kelas *VoiceVerificationActivity* ini merupakan sebuah layar (*Activity*) dalam aplikasi *Android* yang dirancang untuk melakukan verifikasi pengguna melalui suara. Di dalamnya, dideklarasikan beberapa komponen antarmuka pengguna (UI) yang akan diinisialisasi nanti: *micButton* (sebuah *ImageButton* yang kemungkinan berfungsi sebagai tombol mikrofon untuk memulai atau menghentikan perekaman), *instructionText* (sebuah *TextView* untuk menampilkan instruksi kepada pengguna), *resultText* (sebuah *TextView* untuk menampilkan hasil verifikasi), dan *progressBar* (untuk menunjukkan progres proses verifikasi). Selain elemen UI, terdapat juga *speechRecognizer* yang akan digunakan untuk menangkap dan memproses *input* suara dari pengguna.

Kode kedua Fungsi *startVoiceVerification* ini memulai proses verifikasi suara pengguna. Awalnya, fungsi ini mengatur antarmuka pengguna dengan menyembunyikan teks hasil (*resultText*), menampilkan *progressBar*, menonaktifkan tombol mikrofon (*micButton*), dan mengubah teks instruksi (*instructionText*) menjadi "Mendengarkan...". Kemudian, sebuah Intent disiapkan untuk layanan pengenalan suara (*RecognizerIntent.ACTION_RECOGNIZE_SPEECH*), dikonfigurasi untuk menggunakan model bahasa bebas (*LANGUAGE_MODEL_FREE_FORM*), bahasa *default* perangkat (*Locale.getDefault()*), dan menampilkan pesan "Ucapkan *keyword* Anda...". Selanjutnya, sebuah *RecognitionListener* diatur ke *speechRecognizer* untuk menangani berbagai tahapan proses pengenalan suara: saat siap menerima suara, *instructionText* diubah menjadi "Silakan berbicara...", dan setelah pengguna selesai berbicara (*onEndOfSpeech*), teksnya diubah menjadi "Memproses...". Fungsi ini menyiapkan semua yang diperlukan sebelum pengenalan suara benar-benar dimulai oleh *speechRecognizer*.

Tabel 4.8 Potongan Kode *verification activity*

```
class VoiceVerificationActivity : AppCompatActivity()
{
    private lateinit var micButton: ImageButton
    private lateinit var instructionText: TextView
    private lateinit var resultText: TextView
    private lateinit var progressBar: ProgressBar
```

```
    private lateinit var speechRecognizer: SpeechRecognizer
```

```
    private fun startVoiceVerification() {
        resultText.visibility = View.GONE
        progressBar.visibility = View.VISIBLE
        micButton.isEnabled = false
        instructionText.text = "Mendengarkan..."



        val intent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH).apply {
            putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM)

            putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault())
            putExtra(RecognizerIntent.EXTRA_PROMPT, "Ucapkan keyword Anda...")
        }

        speechRecognizer.setRecognitionListener(object : RecognitionListener {
            override fun onReadyForSpeech(params: Bundle?) {
                instructionText.text = "Silakan berbicara..."
            }
            override fun onBeginningOfSpeech() {}
            override fun onRmsChanged(rmsdB: Float) {}
            override fun onBufferReceived(buffer: ByteArray?) {}
            override fun onEndOfSpeech() {
                instructionText.text = "Memproses..."
            }
```

4.2.3 Cara Kerja *Machine Learning*

Kombinasi CNN dengan MFCC (*Mel-Frequency Cepstral Coefficients*) sering digunakan dalam aplikasi pemrosesan sinyal *audio*, seperti pengenalan ucapan, identifikasi

pembicara, atau klasifikasi suara. MFCC adalah representasi fitur standar dalam pemrosesan *audio* yang menggambarkan spektrum daya logaritmik sinyal pada skala frekuensi *Mel*. Fitur MFCC mengekstraksi karakteristik penting dari *timbre* suara manusia, sehingga sangat cocok untuk analisis suara. Ketika MFCC dikombinasikan dengan CNN, fitur MFCC yang biasanya berbentuk vektor atau matriks waktu-frekuensi diperlakukan sebagai "gambar" masukan 1D atau 2D untuk CNN. CNN kemudian dapat mempelajari pola spasial dan temporal dalam fitur MFCC tersebut, seperti perubahan frekuensi atau durasi yang penting untuk mengidentifikasi kata, emosi, atau pembicara. Pendekatan ini memanfaatkan kemampuan MFCC untuk merepresentasikan fitur *audio* yang relevan dengan persepsi manusia, sekaligus memanfaatkan kemampuan CNN untuk belajar fitur kompleks dari representasi tersebut secara otomatis [5], [16].

4.2.3.1 Pengumpulan Dataset

Proses awal dalam pengembangan sistem *machine learning* untuk verifikasi suara pada "*Smart Dorm Key*" adalah pengumpulan dataset suara. *Dataset* ini harus mencakup beragam sampel suara dari pengguna yang sah dan tidak sah, dengan mempertimbangkan variasi seperti intonasi, kecepatan bicara, dan kondisi lingkungan yang berbeda (misalnya, adanya *noise* latar belakang). Pengumpulan data dilakukan melalui aplikasi khusus yang memungkinkan pengguna merekam frasa kunci mereka. Untuk setiap pengguna, beberapa sampel suara frasa kunci (misalnya, "*Pin*") direkam untuk tujuan pendaftaran, sementara sampel suara dari individu yang tidak terdaftar juga dikumpulkan untuk melatih model agar dapat mengenali upaya akses yang tidak sah. Kualitas dan kuantitas *dataset* sangat krusial untuk memastikan robustnya model dalam mengenali suara yang sah dan menolak suara yang tidak dikenal.

4.2.3.2 Anotasi Data

Setelah pengumpulan, setiap sampel suara dalam *dataset* akan melalui proses anotasi data. Anotasi ini melibatkan pemberian label yang akurat untuk setiap rekaman suara, mengidentifikasi apakah suara tersebut berasal dari pengguna yang terdaftar ("sah") atau tidak terdaftar (tidak sah). Selain itu, anotasi juga dapat mencakup informasi meta data lainnya seperti identitas pembicara, waktu rekaman, dan kondisi lingkungan saat rekaman dibuat. Proses anotasi ini memastikan bahwa data yang digunakan untuk pelatihan model *machine learning* memiliki label yang benar, sehingga model dapat belajar membedakan antara suara yang sah dan yang tidak sah dengan tepat.

Tabel 4.9 *Class* Berdasarkan Jenis Suara

No.	Class	Jumlah Dataset
1	Hazbi	475
2	Ito	712
3	Faiq	477
4	*Unknown*	988

Tabel 4.9 merupakan daftar dari *class* suara yang telah didaftarkan berdasarkan jenis suara pengguna dan juga total dari dataset pada setiap label. Pada *class* ini sendiri menggunakan suara Hazbi, Ito, Faiq, sebagai suara pengguna yang terdaftar (sah) dan suara *unknown* sebagai suara pengguna yang tidak terdaftar (tidak sah). Untuk suara dari kelas Hazbi menggunakan *dataset* sebanyak 475 rekaman suara, pada suara kelas Ito menggunakan *dataset* sebanyak 712 rekaman suara, dan untuk kelas Faiq menggunakan 477 dataset rekaman suara, sehingga total *dataset* rekaman suara untuk pengguna yang terdaftar ada sebanyak 1664 rekaman suara dan untuk kelas yang tidak terdaftar (*Unknown*) menggunakan dataset sebanyak 988 rekaman suara.

4.2.3.3 Transformasi Data (Ekstraksi Fitur MFCC)

Transformasi data merupakan langkah penting di mana sinyal suara mentah diubah menjadi representasi yang lebih informatif dan mudah dianalisis oleh model *machine learning*. Dalam konteks verifikasi suara, *Mel-Frequency Cepstral Coefficients* (MFCC) adalah fitur yang paling umum digunakan. Proses ekstraksi fitur MFCC dimulai dengan membagi sinyal suara menjadi *frame-frame* pendek, kemudian setiap *frame* dianalisis untuk mendapatkan spektrum dayanya. Spektrum daya ini kemudian dilewatkan melalui filter *bank* skala *Mel*, dan dilanjutkan dengan transformasi *Discrete Cosine Transform* (DCT) untuk menghasilkan koefisien MFCC. Representasi MFCC ini secara efektif menangkap karakteristik *timbre* suara manusia, menjadikannya sangat relevan untuk tugas pengenalan suara.

4.2.3.4 Pembagian Data

Dataset yang telah diekstraksi fitur MFCC-nya kemudian dibagi menjadi dua kategori utama, yaitu sampel dikenal (*Known*) dan sampel tidak dikenal (*Unknown*). Pembagian ini dilakukan dengan rasio 80% untuk data dikenal dan 20% untuk data tidak dikenal. Data dari kategori dikenal yang mencakup sampel 'Hazbi WAV', 'Ito WAV', dan 'Faiq WAV' akan

digunakan untuk melatih model. Sementara itu, kategori tidak dikenal digunakan untuk mengevaluasi kinerja akhir model terhadap data yang sepenuhnya baru dan tidak termasuk dalam kelas yang telah dipelajari. Pembagian ini memastikan bahwa evaluasi model tidak bias dan memberikan gambaran akurat tentang kemampuannya pada data baru.

4.2.3.5 Struktur *Dataset*

Struktur *dataset* setelah transformasi fitur MFCC akan berupa matriks atau tensor. Setiap *entry* dalam dataset akan terdiri dari urutan koefisien MFCC yang mewakili sampel suara, bersama dengan label yang sesuai (misalnya, ID pengguna atau kategori "sah"/"tidak sah"). Untuk model CNN, urutan MFCC dari setiap sampel suara dapat diperlakukan sebagai "gambar" 1D atau 2D, di mana sumbu waktu dan frekuensi menjadi dimensi spasial yang dapat dieksplorasi oleh lapisan konvolusi. Struktur ini memungkinkan CNN untuk secara otomatis mempelajari pola temporal dan frekuensi yang relevan untuk membedakan antara suara yang berbeda.

4.2.3.6 Instalasi *Library*

Dalam mengimplementasikan ekstraksi fitur MFCC dan membangun model CNN, beberapa *library machine learning* populer diperlukan. Untuk membangun, melatih, dan mengevaluasi model, kita akan menggunakan fungsi-fungsi penting dari *library* yang diimpor. Berikut merupakan daftar dari *library* utama yang digunakan untuk *machine learning.*

Tabel 4.10 Daftar *Library* Utama

Library	Deskripsi Fungsi
Numpy	Digunakan untuk menyimpan dan memproses data dalam bentuk array multidimensi, seperti fitur *audio.*
Librosa	*Library* utama untuk pemrosesan sinyal *audio* seperti memuat file .wav, menghitung MFCC, dll.
Tensorflow.keras	Digunakan untuk membuat arsitektur model (*Dense, Conv1D/2D, Attention*), melatih model, dan melakukan prediksi.
sklearn.model_selection	Berfungsi untuk membagi data menjadi data latih dan data uji.

sklearn.preprocessing	Mengubah label dari bentuk teks menjadi angka dan *one-hot encoding*.
matplotlib.pyplot	Digunakan untuk menampilkan kurva *loss/accuracy*, dan visualisasi data seperti *waveform*, *spectrogram*.
pandas	Digunakan untuk memproses dan menampilkan data hasil prediksi atau evaluasi model dalam format yang terstruktur.

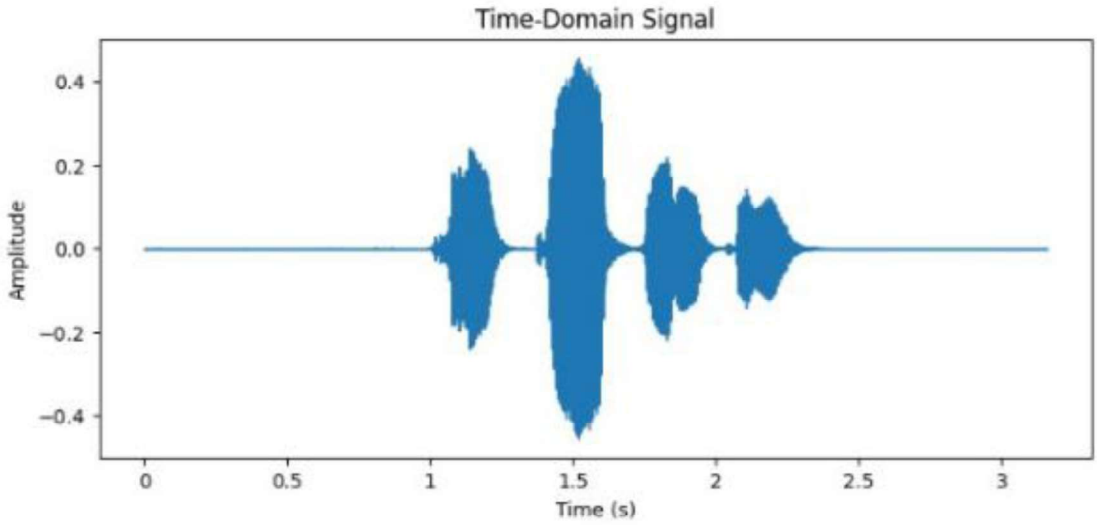
4.2.3.7 Pelatihan *Machine Learning*

Pelatihan model melibatkan penggunaan fitur MFCC yang telah diekstraksi sebagai input untuk *Convolutional Neural Network* (CNN). Dalam konteks ini, CNN akan belajar untuk mengidentifikasi pola-pola unik dalam koefisien MFCC yang berkaitan dengan suara pengguna yang sah. Proses pelatihan dimulai dengan menginisialisasi bobot *neural network* secara acak. Kemudian, data pelatihan (MFCC beserta labelnya) dimasukkan ke dalam CNN secara berulang melalui beberapa *epoch*. Selama setiap *epoch*, model akan melakukan *forward pass* untuk memprediksi label, menghitung *loss* (perbedaan antara prediksi dan label sebenarnya), dan kemudian melakukan *backward pass* (*backpropagation*) untuk memperbarui bobotnya menggunakan algoritma optimasi.

Pelatihan model melibatkan penggunaan fitur MFCC yang telah diekstraksi sebagai input untuk *Convolutional Neural Network* (CNN). Dalam konteks ini, CNN akan belajar untuk mengidentifikasi pola-pola unik dalam koefisien MFCC yang berkaitan dengan suara pengguna yang sah. Proses pelatihan dimulai dengan menginisialisasi bobot *neural network* secara acak. Kemudian, data pelatihan (MFCC beserta labelnya) dimasukkan ke dalam CNN secara berulang melalui beberapa *epoch*. Selama setiap *epoch*, model akan melakukan *forward pass* untuk memprediksi label, menghitung *loss* (perbedaan antara prediksi dan label sebenarnya), dan kemudian melakukan *backward pass* (*backpropagation*) untuk memperbarui bobotnya menggunakan algoritma optimasi.

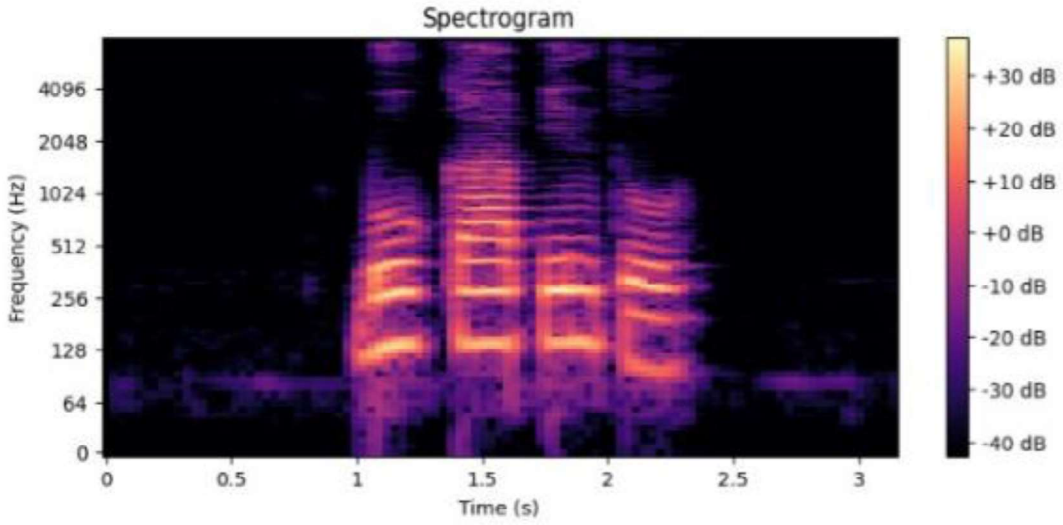
A. Pelatihan *Mel-Frequency Cepstral Coefficients*

Implementasi pelatihan dataset suara menggunakan metode MFCC dimulai dari data suara mentah, yang merupakan sinyal domain waktu (seperti pada grafik kiri atas), yang menunjukkan amplitudo suara seiring waktu.



Gambar 4.29 Data Suara Mentah

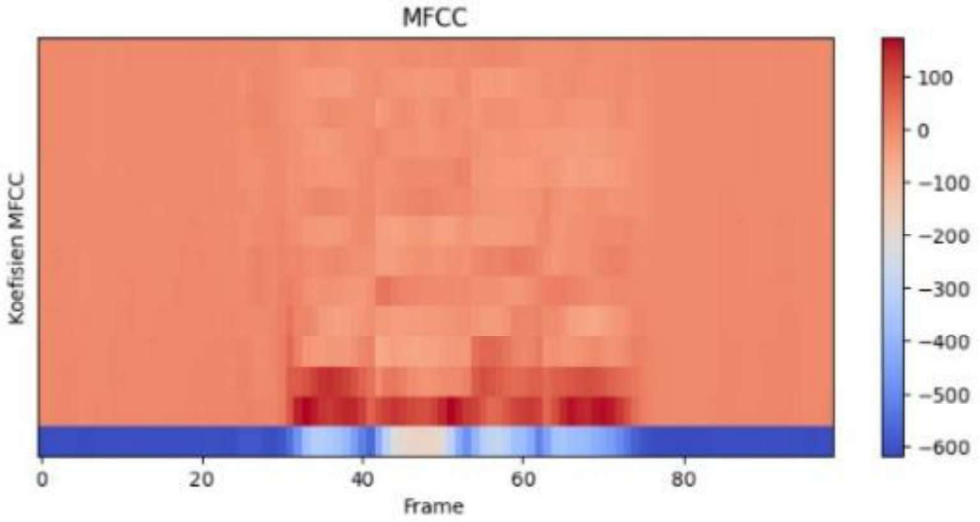
Langkah pertama adalah mengubah sinyal ini menjadi representasi frekuensi-waktu. Hal ini dilakukan dengan proses *framing*, di mana sinyal dibagi menjadi bingkai-bingkai pendek yang tumpang tindih, lalu pada setiap bingkai diterapkan transformasi *Fast Fourier Transform* (FFT) untuk menganalisis kandungan frekuensinya. Hasil dari seluruh bingkai ini kemudian digabungkan untuk membentuk spektrogram (grafik kanan atas), yang memvisualisasikan energi pada berbagai frekuensi dari waktu ke waktu. Spektrogram ini kemudian diolah lebih lanjut untuk mengekstrak fitur yang lebih relevan dengan persepsi pendengaran manusia.



Gambar 4.30 Spektogram

Sumbu frekuensi pada spektrogram dipetakan ke skala *Mel*, yang meniru cara telinga manusia memproses frekuensi (lebih detail pada frekuensi rendah). Setelah mengambil

logaritma dari energi pada setiap filter *Mel*, transformasi *Discrete Cosine Transform* (DCT) diterapkan untuk menghasilkan serangkaian koefisien yang disebut *Mel-Frequency Cepstral Coefficients* (MFCC) (grafik kiri bawah).



Gambar 4.31 Koefisien MFCC

Koefisien MFCC ini menjadi "sidik jari" akustik dari setiap *frame* suara, yang secara efisien menangkap karakteristik penting untuk pengenalan ucapan. Kumpulan fitur MFCC dari seluruh *dataset* inilah yang kemudian digunakan sebagai *input* untuk melatih model *Convolutional Neural Network.*

B. Pelatihan *Convolutional Neural Network*

Setelah melakukan pelatihan MFCC yang di mana *dataset* suara mentah diekstraksi menjadi dalam bentuk spectrogram, *dataset* hasil MFCC ini baru bisa dilakukan pelatihan menggunakan model CNN 5 *layer* untuk klasifikasi. Arsitektur CNN ini dirancang untuk memproses fitur MFCC, yang dianggap sebagai gambar 2D (waktu vs. koefisien MFCC), guna mengekstrak pola-pola akustik secara hierarkis.

```python
def create_cnn_model(input_shape, num_classes):
    model = Sequential([
        Input(shape=input_shape),

        Conv2D(32, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D((2, 2)),
        Dropout(0.25),

        Conv2D(64, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D((2, 2)),
        Dropout(0.25),

        Conv2D(128, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D((2, 2)),
        Dropout(0.25),

        GlobalAveragePooling2D(),

        Dense(512, activation='relu'),
        BatchNormalization(),
        Dropout(0.5),

        Dense(num_classes, activation='softmax')
    ])
    return model
```

Gambar 4.32 Potongan Skrip Label CNN

Implementasinya dimulai dengan tiga blok konvolusi (Conv2D) yang berfungsi sebagai pengekstrak fitur otomatis. Lapisan pertama mencari pola dasar, dan lapisan-lapisan berikutnya (dengan jumlah filter yang meningkat dari 32, 64, hingga 128) belajar mengidentifikasi pola yang lebih kompleks dan abstrak dari kombinasi fitur sebelumnya. Setiap blok konvolusi diikuti oleh MaxPooling2D untuk mereduksi dimensi dan membuat model lebih efisien, serta *Dropout* dan BatchNormalization untuk mencegah *overfitting* dan menstabilkan proses pelatihan.

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 128, 128, 32)
batch_normalization (BatchNormalization)	(None, 128, 128, 32)
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)
dropout (Dropout)	(None, 64, 64, 32)
conv2d_1 (Conv2D)	(None, 64, 64, 64)
batch_normalization_1 (BatchNormalization)	(None, 64, 64, 64)
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)
dropout_1 (Dropout)	(None, 32, 32, 64)
conv2d_2 (Conv2D)	(None, 32, 32, 128)
batch_normalization_2 (BatchNormalization)	(None, 32, 32, 128)
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 128)
dropout_2 (Dropout)	(None, 16, 16, 128)
global_average_pooling2d (GlobalAveragePooling2D)	(None, 128)
dense (Dense)	(None, 512)
batch_normalization_3 (BatchNormalization)	(None, 512)
dropout_3 (Dropout)	(None, 512)
dense_1 (Dense)	(None, 4)

Gambar 4.33 Hasil Output Label CNN

Setelah fitur-fitur penting diekstraksi, lapisan GlobalAveragePooling2D merangkum setiap peta fitur menjadi satu nilai tunggal, yang secara drastis mengurangi jumlah parameter dan mempersiapkan data untuk klasifikasi. Data yang telah diringkas ini kemudian dimasukkan ke lapisan *Dense* (512 neuron) untuk interpretasi tingkat tinggi, sebelum akhirnya masuk ke lapisan *Dense* terakhir dengan aktivasi *softmax*. Lapisan *softmax* inilah yang menghasilkan probabilitas untuk setiap kelas suara yang mungkin, sehingga model dapat membuat prediksi akhir. Implementasi arsitektur ini terbukti sangat efektif, yang ditunjukkan oleh hasil akhir berupa akurasi tes sebesar 92.75%, artinya model mampu mengenali dan mengklasifikasikan data suara yang belum pernah dilihat sebelumnya dengan sangat baik.

```
=== DOWNLOADING MODEL FILES ===

⬇️ Downloading 4 production-ready files...
✅ Downloaded: model_suara_cnn.h5
✅ Downloaded: model_suara_cnn.tflite
✅ Downloaded: label_encoder_cnn.pkl
✅ Downloaded: labels_cnn.txt

================================================================================
🎉 SIMPLIFIED VOICE RECOGNITION TRAINING COMPLETED!
================================================================================
📊 FINAL RESULTS:
   🎯 Model: CNN with Mel-Spectrograms
   ✅ Final Test Accuracy: 92.75%
   📁 4 production-ready files have been generated and downloaded.
```

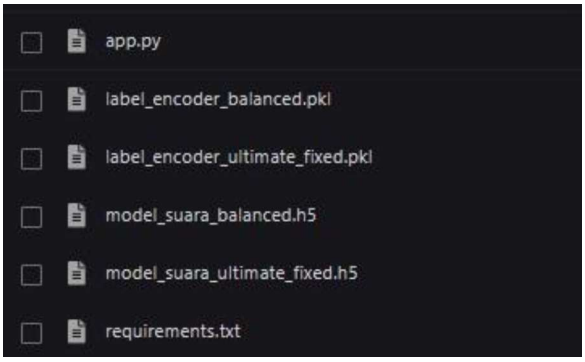
Gambar 4.34 Hasil Training Model CNN

4.2.3.8 Penerapan Model *Machine Learning*

Berdasarkan hasil perancangan sistem, model *machine learning* untuk verifikasi suara terpilih untuk dijalankan pada aplikasi *Smart Dorm Key*. Sebelum dapat digunakan, model ini disiapkan menjadi sebuah *endpoint* pada layanan *backend*. *Endpoint* ini bertugas melakukan proses verifikasi suara secara otomatis ketika dipanggil oleh aplikasi untuk mengautentikasi pengguna melalui layanan *Stregtipe server*.

A. Struktur Penyimpanan File

Untuk keperluan *deployment*, digunakan susunan file seperti yang terlihat pada Gambar 4.30. Di dalamnya terdapat semua file yang dibutuhkan untuk men*deploy* model *machine learning*, yaitu sebuah model CNN yang sudah terlatih.



Gambar 4.35 Struktur Penyimpanan *File Deployment*

Rincian mengenai masing-masing file tersebut dijelaskan sebagai berikut pada Tabel 4.11.

Tabel 4.11 Rincian Struktur Penyimpanan *File Deployment*

Nama File	Keterangan
app.py	Kode berikut merupakan bagian inti yang menangani pemanggilan hasil dari proses deteksi.
label_encoder_balanced.pkl	Menunjukkan fungsinya untuk mengkodekan label menandakan bahwa *encoder* ini dibuat dan dilatih menggunakan dataset yang seimbang.

label_encoder_ultimate_fixed.pkl	Sebagai kamus untuk mengubah label teks menjadi angka dan sebaliknya yang Menandakan bahwa ini kemungkinan adalah versi model yang dianggap final atau paling canggih oleh pengembangnya.
model_suara_balanced.h5	Bagian ini untuk memproses atau mengenali "suara" yang dilatih menggunakan *dataset* yang seimbang dengan format file HDF5 (*Hierarchical Data Format* 5).
model_suara_ultimate_fixed.h5	*File* model untuk analisis suara, yang dianggap sebagai versi terbaik (*ultimate*) dan telah melalui perbaikan (*fixed*) dari versi sebelumnya.
requirements.txt	*File* ini berisi daftar semua paket (*library*) yang diperlukan agar app.py dapat berjalan.

B. Pendeteksian Suara

Sebuah model telah berhasil dibuat dengan melatih arsitektur CNN pada dataset suara seperti pada gambar 4.31 dibawah. Model ini selanjutnya akan dimuat melalui API *voice recognition* untuk menjalankan fungsi deteksi suara pengguna.

```
def __init__(self):
    """
    Initialize voice detector with TensorFlow model and smart predictions
    Compatible with both model_suara.h5 and model_suara_balanced.h5
    """
    try:
        # Try to load TensorFlow model (multiple possible names)
        model_files = ['model_suara_ultimate_fixed.h5', 'model_suara_balanced.h5']
        self.model = None

        for model_file in model_files:
            if os.path.exists(model_file):
                try:
                    self.model = tf.keras.models.load_model(model_file)
                    logger.info(f"TensorFlow model loaded from: {model_file}")
                    break
                except Exception as e:
                    logger.warning(f"Failed to load {model_file}: {e}")

        if self.model is None:
            raise Exception("No valid TensorFlow model found!")

        # Try to load label encoder (multiple possible names)
        encoder_files = ['label_encoder_ultimate_fixed.pkl', 'label_encoder_balanced.pkl']
        self.label_encoder = None
```

Gambar 4.36 Skrip untuk *Load Model CNN*

Setelah model dimuat melalui API, akan dijalankan fungsi untuk melakukan prediksi suara seperti pada gambar 4.32 dibawah. Alur kerja fungsi deteksi suara ini dimulai dengan model AI akan menganalisis *input* suara. Kemudian, dari hasil deteksi, sistem akan mengekstrak informasi dari *bounding boxes* (kotak pembatas), yang mencakup ID & nama kelas, skor kepercayaan, serta posisi dan ukurannya. Semua data ini dikumpulkan ke dalam satu daftar (*list*) dan dikembalikan sebagai *output* fungsi.

```
def smart_predict(self, features):
    """Smart prediction with bias correction towards known voices"""
    if features is None:
        return {"error": "Couldn't process audio file", "confidence": 0.0}

    try:
        # Reshape features for model input
        features_reshaped = features.reshape(1, -1)

        # Make prediction
        raw_predictions = self.model.predict(features_reshaped, verbose=0)[0]

        # Create probability dictionary
        all_probabilities = {}
        for i, class_name in enumerate(self.label_encoder.classes_):
            all_probabilities[class_name] = float(raw_predictions[i])

        # Apply smart logic to reduce unknown bias
        adjusted_predictions = self._apply_smart_logic(raw_predictions, all_probabilities)

        # Get final prediction
        predicted_class_idx = np.argmax(adjusted_predictions)
        confidence = float(np.max(adjusted_predictions))
        predicted_class = self.label_encoder.classes_[predicted_class_idx]

        # Create detailed result
        result = {
            "prediction": predicted_class,
            "confidence": confidence,
            "confidence_percent": confidence * 100,
            "raw_probabilities": all_probabilities,
            "adjusted_probabilities": {
                name: float(adjusted_predictions[i])
                for i, name in enumerate(self.label_encoder.classes_)
            },
            "prediction_logic": self._explain_prediction(predicted_class, confidence, all_probabilities),
            "certainty_level": self._get_certainty_level(confidence)
        }

        # Final confidence check
        if confidence < self.thresholds['absolute_min']:
            result["prediction"] = "uncertain"
            result["warning"] = "Very low confidence prediction"
```

Gambar 4.37 Skrip untuk Prediksi Suara

Selanjutnya diperlukan *endpoint* seperti yang terlihat pada gambar 4.33 dibawah yang berfungsi sebagai pintu untuk layanan deteksi suara. Saat pengguna mengirim *audio*, maka sistem akan menerimanya. Kemudian, sistem akan memanggil fungsi lain (detector.smart_predict()) untuk menganalisis suara itu secara mendalam. Informasi hasil analisis tersebut, seperti objek apa yang terdeteksi, akan disusun dalam format JSON dan dikirim kembali ke pengguna.

```python
@app.route('/api/detect/suara', methods=['POST'])
def predict_voice():
    """Predict voice from uploaded audio file with smart logic"""
    if not detector:
        return jsonify({'error': 'Voice detector not initialized'}), 500

    try:
        if 'audio' not in request.files:
            return jsonify({'error': 'No audio file provided'}), 400

        file = request.files['audio']
        username = request.form.get('username', 'unknown')

        if file.filename == '':
            return jsonify({'error': 'No file selected'}), 400

        # Save uploaded file
        original_path = os.path.join(UPLOAD_FOLDER, f"{username}_{file.filename}")
        file.save(original_path)

        try:
            # Convert to WAV format if needed
            if not original_path.lower().endswith('.wav'):
                audio = AudioSegment.from_file(original_path)
                audio = audio.set_frame_rate(16000).set_channels(1).set_sample_width(2)
                wav_path = original_path.rsplit('.', 1)[0] + '.wav'
                audio.export(wav_path, format='wav')
                os.remove(original_path)
                original_path = wav_path
        except Exception as e:
            if os.path.exists(original_path):
                os.remove(original_path)
            return jsonify({'error': f'Audio conversion error: {str(e)}'}), 400

        # Extract features and predict
        features = detector.extract_features(original_path)
        result = detector.smart_predict(features)

        # Clean up temporary file
        if os.path.exists(original_path):
            os.remove(original_path)

        # Prepare response
        response = {
            'status': 'success',
            'username_requested': username,
            'model_type': 'tensorflow_smart',
            **result
        }

        return jsonify(response)
```

Gambar 4.38 Skrip untuk API Hasil AKhir

C. Tahapan *Deployment*

Proses dari *deployment* untuk fitur *voice recognition* ini menggunakan *Stegripe Server* dan memiliki alur yang sistematis. Pada langkah pertama adalah mengunggah *source code* ke repositori di GitHub. Setelah itu, kode yang sudah ada di GitHub akan di-*deploy* atau diterapkan ke *Stegripe Server*. Tujuan dari proses ini adalah untuk menjalankan semua fungsi yang diperlukan agar fitur *voice recognition* dapat beroperasi di *server*. Sebagai konfirmasi bahwa proses *deployment* telah selesai dan berhasil, sistem akan menghasilkan sebuah URL *endpoint*. URL inilah yang nantinya

diintegrasikan dengan aplikasi *smart dorm lock* sehingga aplikasi tersebut dapat mengakses dan menggunakan fitur pengenalan suara secara fungsional.

4.3 Prosedur Pengoperasian

Prosedur pengoperasian ini menjelaskan pengoperasian alat yang diuji untuk mencakup skenario pengukuran kinerja alat. Tujuan dari prosedur ini adalah untuk menjamin kelancaran penggunaan dan pengoperasian sistem bagi seluruh pihak, termasuk para penghuni gedung. Berikut adalah prosedur pengoperasian sistem dari *smart dorm key* berbasis *fingerprint* dan *voice recognition* sebagai verifikasi dua langkah :

A. Menghubungkan Sistem ke Jaringan

Hubungkan perangkat ESP32 ke jaringan *Wi-Fi.* Diperlukan koneksi jaringan yang stabil guna memastikan sistem dapat berkomunikasi dengan perangkat lain dan aplikasi.

B. Persiapan Perangkat Keras

- Periksa kembali konektivitas seluruh komponen perangkat keras untuk memastikan semuanya terpasang dengan tepat.
- Pastikan pemasangan sensor *fingerprint* sudah tepat dan bersih sehingga mampu memindai sidik jari dengan akurat.
- Pastikan ESP32 terhubung dengan aplikasi sehingga dapat melakukan verifikasi suara untuk membuka pintu.
- Pastikan sensor IR *no touch button* terhubung dan berfungsi untuk membuka pintu dari dalam.
- Uji coba fungsi *solenoid door lock* untuk memastikan sistem penguncian pintu bekerja sebagaimana mestinya.

C. Menjalankan Sistem

Untuk mengaktifkan sistem, hidupkan ESP32 lalu jalankan kode programnya. Setelah aktif, program akan menggunakan sensor *fingerprint*, terhubung dengan aplikasi untuk melakukan verifikasi suara dan sensor IR *no touch button* untuk membuka *solenoid door lock*.

D. Proses Akses Masuk dari Luar

Untuk akses masuk dari luar pengguna akan melakukan verifikasi dua langkah, yaitu melakukan pemindaian sidik jari dan pengenalan suara lalu sistem akan memeriksa apakah sidik jari dan suara tersebut ada dalam *dataset* :

1) *Voice Recognition*

- Suara Terdaftar

Apabila suara terverifikasi sebagai penghuni gedung, sistem akan mengirimkan pesan ke ESP32 untuk memasukkan *fingerprint* yang terdaftar

- Suara Tidak Terdaftar

Jika terjadi kegagalan verifikasi suara, informasi tersebut akan direkam secara otomatis oleh sistem. Secara bersamaan, sebuah pesan "Tidak Dikenal" akan ditampilkan pada layar LCD dan notifikasi akan dikirimkan melalui aplikasi kepada *admin*. Setelah gagal melakukan verifikasi maka sistem akan menyuruh pengguna untuk melakukan verifikasi ulang.

2) *Fingerprint*

- Sidik Jari Terdaftar

Saat sistem mengenali Sidik jari sebagai penghuni, *solenoid door lock* secara otomatis akan terbuka untuk mengizinkan akses. Mekanisme penguncian ulang akan aktif setelah jeda 5 detik,

- Sidik Jari Tidak Terdaftar

Jika terjadi kegagalan pemindaian sidik jari, informasi tersebut akan direkam secara otomatis oleh sistem. Secara bersamaan, sebuah pesan "Tidak Dikenal" akan ditampilkan pada layar LCD dan notifikasi akan dikirimkan melalui aplikasi kepada admin.

E. Proses Akses Masuk dari Dalam

Mekanisme akses keluar bagi penghuni memanfaatkan sensor IR *no touch button*. Deteksi objek oleh sensor ini akan secara otomatis memicu terbukanya *solenoid door lock*, yang selanjutnya memberikan akses bagi penghuni untuk keluar dari ruangan.

Di bawah ini adalah tahapan untuk setiap jenis akses: melalui pengenalan sidik jari dan suara, serta menggunakan sensor IR *no touch button* :

1. Akses dari Luar *Prototype*

- Subjek uji utama (siapapun) melakukan percobaan akses pada pintu prototipe dengan membuka aplikasi dan melakukan verifikasi untuk pengenalan suara.
- Kemudian, menempelkan jari pada sensor *fingerprint* untuk memulai proses pemindaian jari.
- Setelah berhasil dikenali oleh sistem, maka pintu akan terbuka

2. Akses dari Dalam Gedung/Pintu
    - Pengujian kemudian dilanjutkan dari sisi dalam ruangan. Pengguna mencoba membuka pintu dengan mendekatkan tangan ke sensor IR *no touch button*.
    - Pintu akan terbuka secara otomatis saat sensor mendeteksi keberadaan objek pada jarak 10 cm.

Pada intinya, prosedur pengoperasian *Smart Dorm Key* ini diciptakan untuk memberikan kemudahan bagi penghuni, sekaligus meningkatkan standar keamanan dan kenyamanan di lingkungan gedung.

BAB 5
PENGUJIAN

5.1 Skenario Umum Pengujian

Skenario pengujian untuk *Smart Dorm Key* Berbasis *Fingerprint* dan *Voice Recognition* sebagai Verifikasi Dua Langkah akan dirancang untuk mengevaluasi akurasi sistem dalam berbagai kondisi lingkungan dan penggunaan. Pengujian akan melibatkan sejumlah partisipan yang akan berperan sebagai pengguna, dengan setiap partisipan menjalani serangkaian pengujian dalam beberapa skenario utama, masing-masing diulang sebanyak 15 kali. Melakukan 15 kali pengujian untuk setiap partisipan di setiap skenario memungkinkan kami mengumpulkan data kuantitatif yang kuat. Ini akan menunjukkan seberapa akurat sistem *Smart Dorm Key* kami dalam berbagai kondisi, sekaligus menjadi tolak ukur ketangguhan dan keandalan perangkat yang telah dikembangkan. Berikut adalah detail skenario pengujian yang akan dilakukan:

5.1.1 Pengujian Verifikasi *Voice Recognition*

Pada pengujian *voice recognition* ini dilakukan secara dua kali, dimana pada pengujian pertama dilakukan dengan menggunakan suara dari tiga orang sebagai subjek yang terdaftar, yaitu Hazbi, Trisucipto, dan Faiq. Pengujian *voice recognition* ini dilakukan sebanyak 3 kali pengujian dimana pada setiap pengujian dilakukan pengambilan suara untuk dilakukan percobaan sebanyak 5 kali, sehingga untuk total percobaan dalam *voice recognition* ini mencapai 45 kali percobaan. Pada pengujian kedua dilakukan dengan menggunakan suara 3 orang berbeda yang tidak terdaftar (*unknown)*, yaitu Ikratul, Naufal, dan Rayzi. Pengujian *voice recognition* ini dilakukan sebanyak 3 kali pengujian dimana pada setiap pengujian dilakukan pengambilan suara untuk dilakukan percobaan sebanyak 5 kali, sehingga untuk total percobaan dalam *voice recognition* ini mencapai 45 kali percobaan.

Pengujian ini akan mengevaluasi kemampuan sistem dalam mengenali suara pengguna dalam berbagai kondisi :

- Pengujian *Voice Recognition* Kondisi Normal: Verifikasi suara akan dilakukan di lingkungan yang tenang dan minim kebisingan, dengan partisipan mengucapkan frasa kunci secara jelas. Ini akan menjadi *baseline* untuk akurasi suara.
- Pengujian *Voice Recognition* Kondisi Berisik: Kami akan menguji akurasi sistem di tengah kebisingan latar belakang, seperti suara percakapan atau musik. Ini mensimulasikan lingkungan asrama yang ramai.

- Pengujian *Voice Recognition* Kondisi Serak: Kami akan menguji kemampuan sistem untuk mengenali suara pengguna meskipun ada sedikit perubahan pada kualitas suara, seperti saat suara agak serak atau lelah.

5.1.2 Pengujian Sensor *Fingerprint*

Pengujian sensor f*ingerprint* ini dilakukan dengan tiga orang sebagai subjek untuk melakukan pengujian, yaitu Hazbi, Trisucipto, dan Faiq. Pengujian sensor *fingerprint* ini dilakukan sebanyak 3 kali percobaan dimana pada percobaan pertama dilakukan dengan menggunakan 5 jari tangan kanan berbeda untuk setiap orang, percobaan kedua menggunakan 5 jari tangan kiri untuk setiap orang, dan percobaan ketiga dilakukan dilakukan dengan menggunakan 5 jari secara acak dari setiap orang.

Pengujian ini akan fokus pada akurasi pengenalan sidik jari dalam tiga kondisi berbeda:

- Pengujian *Fingerprint* Kondisi Normal: Verifikasi sidik jari akan dilakukan di lingkungan yang ideal, tanpa gangguan pada jari atau sensor. Ini akan menjadi *baseline* untuk akurasi sistem.
- Pengujian *Fingerprint* Kondisi Kotor: Kami akan menguji bagaimana sistem mengenali sidik jari yang sedikit kotor atau berminyak, seperti setelah memegang makanan ringan. Ini mensimulasikan penggunaan sehari-hari yang mungkin tidak selalu bersih.
- Pengujian *Fingerprint* Kondisi Basah: Pengujian ini akan mengevaluasi kinerja sensor saat jari sedikit basah, misalnya setelah mencuci tangan atau berkeringat.

5.1.3 Pengujian *No Touch* Sensor

Pada pengujian untuk *no touch* sensor ini dilakukan dengan mengukur pada jarak berapa saja sensor ini dapat diaktifkan. Pengujian ini dilakukan dengan cara mendekatkan objek atau tangan pada sensor *no touch* button dari jarak yang dekat hingga jarak yang membuat sensor ini tidak aktif atau merespon objek.

5.1.4 Pengujian *Machine Learning*

Pengujian *machine learning* dirancang untuk mengevaluasi kinerja dan akurasi dari model *ensemble* pengenalan suara yang telah dilatih, dengan tujuan utama memvalidasi kemampuan sistem dalam membedakan secara akurat antara suara pengguna yang terdaftar (Faiq, Hazbi, Ito, beserta variasinya) dan suara pengguna yang tidak terdaftar (*unknown*). Evaluasi ini dilakukan menggunakan metrik standar industri yang dianalisis melalui Laporan Klasifikasi (C*lassification Report*) dan *Confusion Matrix*. Pengujian akurasi klasifikasi berfokus pada pengukuran kuantitatif menggunakan metrik utama seperti *Precision* untuk

mengukur tingkat ketepatan prediksi, *Recall* untuk menilai kemampuan model menemukan semua sampel yang relevan, dan *F1-Score* sebagai skor penyeimbang keduanya. Di sisi lain, analisis kesalahan dengan *Confusion Matrix* bertujuan untuk memvisualisasikan performa model secara rinci, mengidentifikasi di mana model berhasil melakukan prediksi dengan benar, serta di mana terjadi kesalahan klasifikasi sehingga dapat dipahami apakah ada kelas suara tertentu yang cenderung tertukar.

5.1.5 Pengujian *Quality of Service* (QoS)

Selain pengujian akurasi dalam berbagai kondisi, kami juga akan melakukan pengujian Kualitas Layanan (*Quality of Service*) untuk memastikan sistem *Smart Dorm Key* tidak hanya akurat tetapi juga responsif dan dapat diandalkan. Pengujian QoS ini akan fokus pada latensi (*delay*) dan *jitter*. Latensi adalah waktu yang dibutuhkan sistem untuk merespons sejak sidik jari atau suara dimasukkan hingga keputusan verifikasi diberikan. *Jitter* akan diukur untuk melihat variasi atau fluktuasi dalam *delay* tersebut, memastikan respons sistem konsisten dan tidak berfluktuasi secara signifikan. Analisis QoS ini akan memberikan gambaran lengkap tentang kinerja operasional sistem *Smart Dorm Key* secara keseluruhan terkait aspek waktu respons.

5.2 Proses Pengujian dan Analisis Hasil

Pada bagian detil pengujian ini akan dijelaskan proses dan hasil pengujian secara rinci. Penjelasan akan dilakukan secara berurutan untuk setiap spesifikasi yang diuji. Materi uji yang dibahas dalam setiap spesifikasi meliputi skenario detail, proses pengujian, dan hasil pengujian. Hasil pengujian yang bersifat kuantitatif dapat disajikan dalam bentuk tabel atau bagan.

5.2.1 Pengujian *Voice Recognition*

Pada pengujian untuk *voice recognition* dilakukan dengan melakukan pengetesan suara dari tiga anggota kelompok dan pengetesan suara ini dilakukan dalam tiga kondisi berbeda, yaitu ketika kondisi suara dan lingkungan normal, kondisi lingkungan berisik dan ketika suara pengguna sedang serak. Untuk pengetesan ini dilakukan sebanyak tiga kali pengujian dimana untuk setiap pengujian dilakukan sebanyak 5 kali percobaan. Berdasarkan hasil percobaan tersebut kemudian akan dihitung untuk tingkat keberhasilan sensor dalam bentuk persentase menggunakan perhitungan untuk mengukur persentase keberhasilan dengan menggunakan rumus (5.1).

$$\text{Tingkat Akurasi (\%)} = \left(\frac{\text{Jumlah Keberhasilan}}{\text{Total Percobaan}}\right) \text{x } 100\% \quad (5.1)$$

5.2.1.1 Pengujian Suara dalam Keadaan Normal

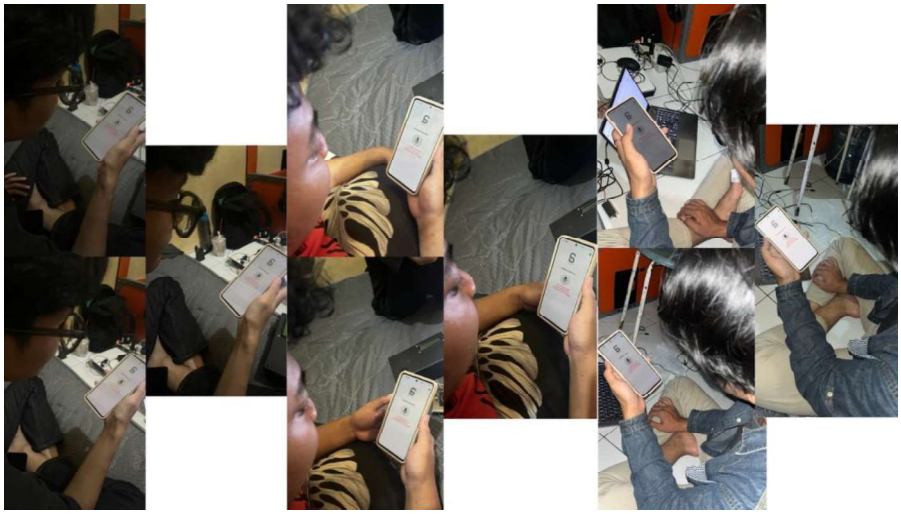
Pengujian suara dalam keadaan normal merupakan bagian dari evaluasi sistem *voice recognition* yang dilakukan untuk mengukur akurasi pengenalan suara dalam kondisi lingkungan ideal. Pengujian ini bertujuan untuk menetapkan *baseline* kinerja sistem di lingkungan yang tenang dan minim kebisingan, sebelum melakukan pengujian dalam kondisi yang lebih menantang. Proses pengujian melibatkan partisipan yang mengucapkan frasa kunci secara jelas dan teratur, dengan setiap pengujian terdiri dari beberapa percobaan untuk memastikan validitas data yang terkumpul.

Tabel 5.1 Hasil Pengujian Suara Terdaftar Keadaan Normal

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal
Hazbi	✓		✓		✓	
		✓	✓		✓	
	✓		✓		✓	
	✓		✓			✓
	✓		✓		✓	
Trisucipto	✓		✓		✓	
	✓		✓		✓	
		✓	✓		✓	
	✓		✓		✓	
	✓		✓		✓	
Faiq	✓		✓		✓	
	✓		✓		✓	
	✓			✓	✓	
	✓		✓		✓	
	✓		✓		✓	
Hasil suara yang terverifikasi berhasil						41

Hasil suara yang gagal terverifikasi	4
Akurasi Keakuratan (%)	91%

Pada Tabel 5.1 menyajikan hasil pengujian suara dalam keadaan normal untuk tiga partisipan: Hazbi, Trisucipto, dan Faiq. Setiap partisipan melakukan tiga kali pengujian, dengan masing-masing pengujian terdiri dari lima percobaan. Berdasarkan data, dari total 45 percobaan, sebanyak 41 suara berhasil diverifikasi, sementara 4 suara gagal diverifikasi. Hal ini menghasilkan akurasi keakuratan sistem sebesar 91% dalam kondisi suara dan lingkungan normal. Kesimpulannya, sistem *voice recognition* menunjukkan kinerja yang sangat baik dan akurat dalam kondisi lingkungan yang tenang dan minim kebisingan.



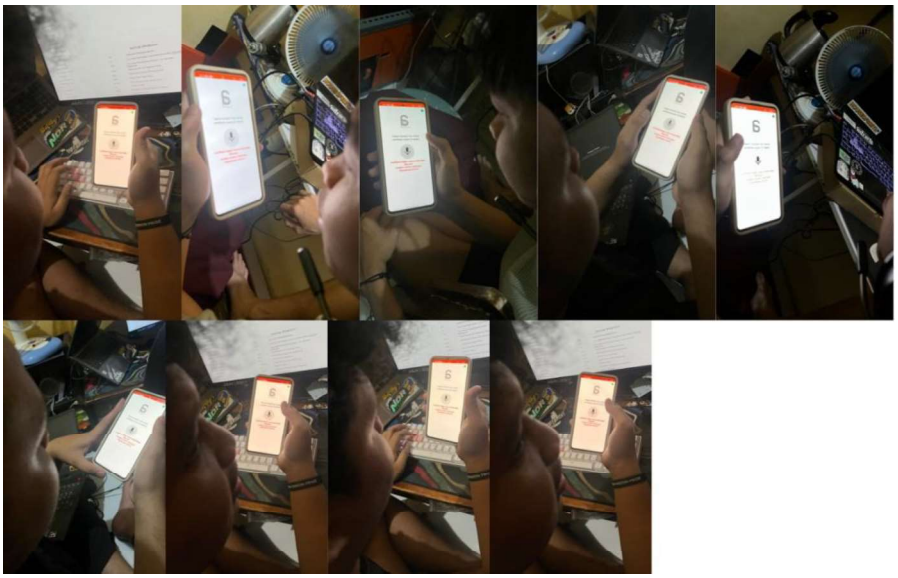
Gambar 5.1 Sampel Pengujian Suara Terdaftar Keadaan Normal.

Tabel 5.2 Hasil Pengujian Suara Tidak Terdaftar Keadaan Normal

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal
Ikratul	✓		✓		✓	
		✓	✓		✓	
	✓			✓		✓
	✓		✓		✓	
	✓		✓		✓	
Naufal	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	
	✓		✓			✓
	✓		✓		✓	
Rayzi	✓		✓		✓	
	✓		✓		✓	
	✓			✓	✓	
	✓		✓		✓	
	✓		✓		✓	
Hasil suara yang terverifikasi unknown						40
Hasil suara yang tidak terverifikasi unknown						5
Akurasi Keakuratan (%)						88%

Pada Tabel 5.2 menampilkan hasil pengujian suara tidak terdaftar dalam keadaan normal untuk tiga partisipan: Ikratul, Naufal, dan Rayzi. Setiap partisipan melakukan tiga kali pengujian, dengan masing-masing pengujian terdiri dari lima percobaan. Berdasarkan data, dari total 45 percobaan, sebanyak 40 suara berhasil terverifikasi sebagai *unknown*, sementara 5 suara gagal diverifikasi sebagai *unknown*. Hal ini menghasilkan tingkat keakuratan sistem

sebesar 88% dalam kondisi suara dan lingkungan normal. Kesimpulannya, sistem *voice recognition* menunjukkan kinerja yang sangat baik dan akurat dalam kondisi lingkungan yang tenang dan minim kebisingan.



Gambar 5.2 Sampel Pengujian Suara Tidak Terdaftar Keadaan Normal.

5.2.1.2 Pengujian Suara Normal dalam Keadaan Berisik

Pengujian suara dalam keadaan berisik bertujuan untuk mengevaluasi akurasi sistem *voice recognition* saat dihadapkan pada gangguan suara dari lingkungan sekitar, seperti di lingkungan asrama yang ramai dengan percakapan atau musik. Proses pengujian akan melibatkan partisipan yang mengucapkan frasa kunci dalam kondisi kebisingan latar belakang yang terkontrol, dengan tujuan untuk mengukur seberapa baik sistem dapat menyaring suara yang tidak relevan dan tetap mengidentifikasi suara pengguna dengan benar.

Tabel 5.3 Hasil Pengujian Suara Normal Terdaftar dalam Keadaan Berisik

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal
Hazbi		✓	✓			✓
	✓		✓		✓	
	✓			✓	✓	
		✓	✓		✓	
	✓		✓			✓

Trisucipto		✓	✓		✓	
	✓			✓		✓
	✓			✓	✓	
	✓		✓		✓	
		✓	✓		✓	
Faiq	✓			✓	✓	
	✓		✓		✓	
		✓	✓		✓	
		✓	✓			✓
	✓		✓		✓	
Hasil suara yang terverifikasi berhasil						31
Hasil suara yang gagal terverifikasi						14
Akurasi Keakuratan (%)						68%

Pada Tabel 5.3 menampilkan hasil pengujian suara dalam keadaan berisik untuk tiga partisipan: Hazbi, Trisucipto, dan Faiq, masing-masing melakukan tiga kali pengujian dengan lima repetisi. Dari total 45 percobaan, sebanyak 31 suara berhasil terverifikasi sebagai *unknown*, sementara 14 suara gagal diverifikasi. Hal ini menghasilkan akurasi sistem *voice recognition* sebesar 68% dalam kondisi berisik. Kesimpulannya, akurasi sistem *voice recognition* mengalami penurunan yang signifikan ketika dihadapkan pada kebisingan latar belakang, meskipun masih menunjukkan tingkat keberhasilan yang moderat.

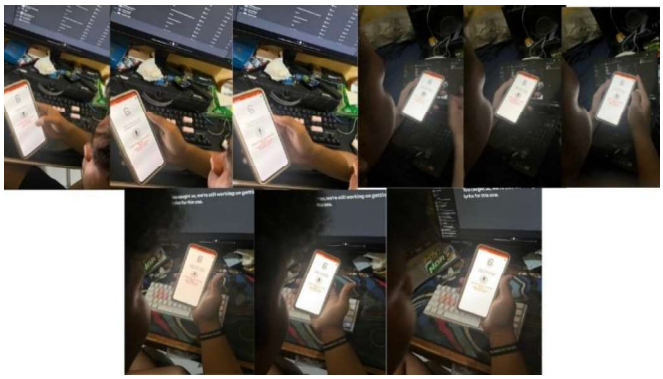
Gambar 5.3 Sampel Pengujian Suara Terdaftar Keadaan Berisik.

Tabel 5.4 Hasil Pengujian Suara Tidak Terdaftar Keadaan Berisik

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal
Ikratul			✓			✓
	✓		✓		✓	
	✓			✓	✓	
		✓	✓		✓	
	✓		✓			✓
Naufal		✓	✓		✓	
	✓			✓		✓
	✓			✓	✓	
	✓		✓		✓	
		✓	✓		✓	
Rayzi	✓			✓	✓	
	✓		✓		✓	
		✓	✓		✓	
		✓	✓			✓
	✓		✓		✓	
Hasil suara yang terverifikasi unknown						28
Hasil suara yang tidak terverifikasi unknown						17
Akurasi Keakuratan (%)						62%

Pada Tabel 5.4 menampilkan hasil pengujian suara dalam keadaan berisik untuk tiga partisipan: Ikratul, Naufal, dan Rayzi. Setiap orang melakukan tiga kali pengujian dengan lima repetisi. Dari total 45 percobaan, sebanyak 28 suara berhasil terverifikasi sebagai *unknown*, sementara 17 suara gagal terverifikasi sebagai *unknown*. Hal ini menghasilkan akurasi sistem *voice recognition* sebesar 62% dalam kondisi berisik. Kesimpulannya, akurasi sistem *voice*

recognition mengalami penurunan yang signifikan ketika dihadapkan pada kebisingan latar belakang, meskipun masih menunjukkan tingkat keberhasilan yang moderat.



Gambar 5.4 Sampel Pengujian Suara Tidak Terdaftar Keadaan Berisik.

5.2.1.3 Pengujian ketika Suara dalam Keadaan Serak

Pengujian suara dalam keadaan serak dirancang untuk mengevaluasi ketahanan dan akurasi sistem *voice recognition* ketika kualitas suara pengguna tidak optimal. Skenario ini mensimulasikan kondisi di mana pengguna mungkin sedang tidak sehat atau mengalami kelelahan suara, yang dapat mengubah karakteristik vokal. Pengujian ini melibatkan partisipan yang mengucapkan frasa kunci dengan suara yang sengaja dibuat serak, dan sistem akan mengidentifikasi apakah suara tersebut masih dapat diverifikasi dengan benar. Hasil dari pengujian ini akan memberikan wawasan penting tentang fleksibilitas dan adaptabilitas sistem terhadap variasi suara alami pengguna.

Tabel 5.5 Hasil Pengujian ketika Suara Terdaftar dalam Keadaan Serak

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal
Hazbi	✓		✓			✓
	✓		✓		✓	
		✓		✓	✓	
	✓			✓	✓	
	✓		✓		✓	
Trisucipto	✓		✓		✓	
		✓	✓		✓	

		✓	✓			✓
	✓		✓			✓
	✓			✓	✓	
Faiq		✓	✓		✓	
	✓		✓			✓
	✓		✓		✓	
	✓			✓	✓	
	✓		✓		✓	
Hasil suara yang terverifikasi berhasil						33
Hasil suara yang gagal terverifikasi						12
Akurasi Keakuratan (%)						73%

Pada Tabel 5.5 menampilkan hasil pengujian suara dalam keadaan serak untuk tiga partisipan: Hazbi, Trisucipto, dan Faiq. Masing-masing partisipan melakukan tiga kali pengujian, dengan setiap pengujian terdiri dari lima repetisi. Dari total 45 percobaan, sebanyak 33 suara berhasil diverifikasi, sementara 12 suara gagal diverifikasi. Hal ini menghasilkan akurasi sistem *voice recognition* sebesar 73% dalam kondisi suara serak. Kesimpulannya, meskipun suara dalam keadaan serak dapat menurunkan akurasi sistem *voice recognition*, sistem masih mampu mengidentifikasi sebagian besar suara dengan benar, menunjukkan tingkat adaptabilitas yang cukup baik terhadap variasi kualitas suara pengguna.

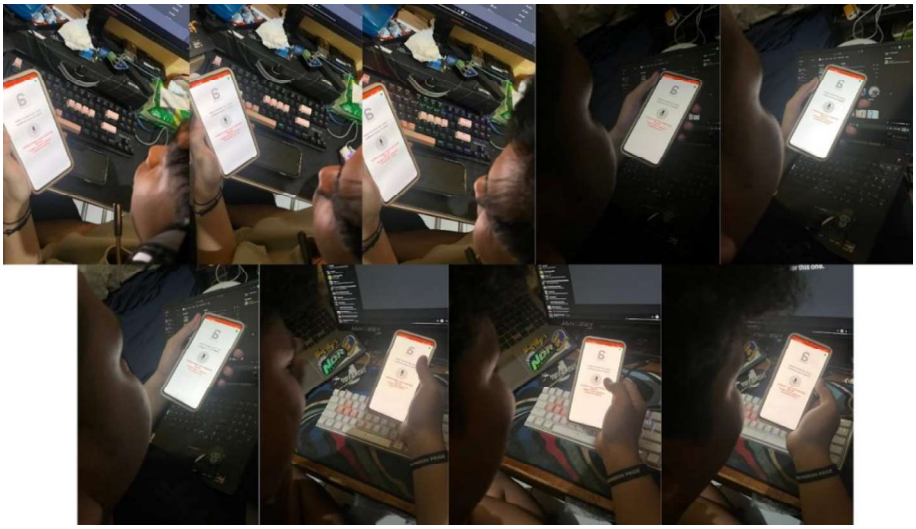
Gambar 5.5 Sampel Pengujian Suara Terdaftar Keadaan Serak.

Tabel 5.6 Hasil Pengujian Suara Tidak Terdaftar Keadaan Serak

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal
Ikratul	✓		✓			✓
		✓		✓	✓	
	✓			✓		✓
		✓	✓		✓	
	✓		✓		✓	
Naufal	✓			✓	✓	
	✓			✓	✓	
	✓		✓		✓	
	✓		✓		✓	
		✓	✓			✓
Rayzi	✓			✓	✓	
		✓	✓		✓	
		✓	✓		✓	
	✓		✓			✓
	✓		✓		✓	
Hasil suara yang terverifikasi unknown						30
Hasil suara yang tidak terverifikasi unknown						15
Akurasi Keakuratan (%)						66%

Pada Tabel 5.6 menampilkan hasil pengujian suara dalam keadaan serak untuk tiga partisipan: Ikratul, Naufal, dan Rayzi. Masing-masing partisipan melakukan tiga kali pengujian, dengan setiap pengujian terdiri dari lima repetisi. Dari total 45 percobaan, sebanyak 30 berhasil terverifikasi sebagai *unknown*, sementara 15 suara gagal terverifikasi sebagai *unknown*. Hal ini menghasilkan akurasi sistem *voice recognition* sebesar 66% dalam kondisi

suara serak. Kesimpulannya, meskipun suara dalam keadaan serak dapat menurunkan akurasi sistem *voice recognition*, sistem masih mampu mengidentifikasi sebagian besar suara dengan benar, menunjukkan tingkat adaptabilitas yang cukup baik terhadap variasi kualitas suara pengguna.



Gambar 5.6 Sampel Pengujian Suara Tidak Terdaftar Keadaan Serak.

5.2.2 Pengujian Sensor *Fingerprint* AS608

Untuk pengujian dari sensor *fingerprint* ini dilakukan dalam tiga kondisi yang berbeda, yaitu ketika kondisi jari dalam keadaan normal, basah dan juga kotor. Untuk setiap kondisi yang berbeda dilakukan sebanyak 3 percobaan, yang dimana setiap percobaan dilakukan sebanyak 5 repetisi.

5.2.2.1 Pengujian Sidik Jari dalam Keadaan Normal

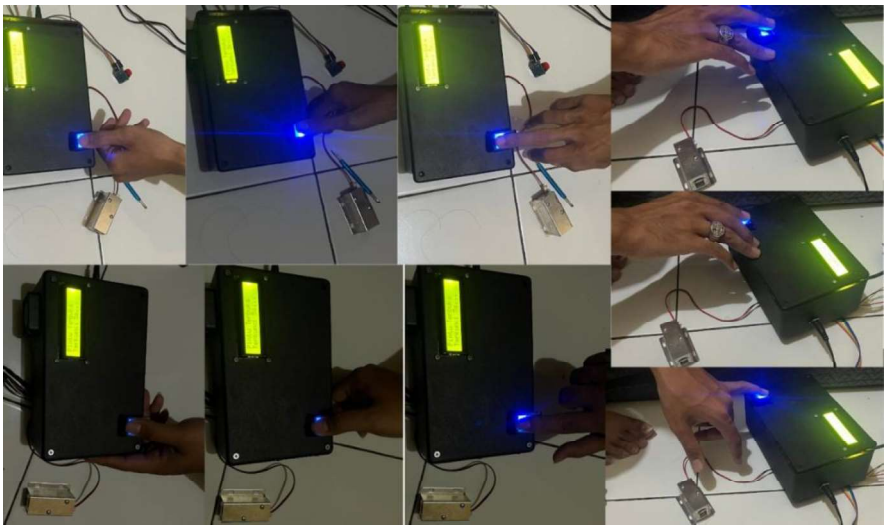
Pengujian sidik jari dalam keadaan normal dilakukan untuk mengevaluasi akurasi sensor sidik jari AS608 dalam kondisi ideal. Proses pengujian ini melibatkan tiga percobaan, di mana setiap percobaan terdiri dari lima repetisi. Hasil dari pengujian ini akan menjadi *baseline* atau acuan untuk membandingkan kinerja sensor dalam kondisi jari yang tidak normal.

Tabel 5.7 Hasil Pengujian Sidik Jari Keadaan Normal

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Terbaca	Tidak	Terbaca	Tidak	Terbaca	Tidak
Hazbi	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	

	✓		✓		✓	
	✓		✓		✓	
Trisucipto	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	
Faiq	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	
	✓		✓		✓	
Hasil Sidik Jari Terbaca						45
Hasil Sidik Jari Tidak Terbaca						0
Akurasi Keakuratan Sensor (%)						100%

Pada Tabel 5.7 Hasil Pengujian Sidik Jari Keadaan Normal menunjukkan hasil dari pengujian sidik jari dalam keadaan normal untuk tiga partisipan: Hazbi, Trisucipto, dan Faiq. Pengujian dilakukan sebanyak tiga kali percobaan, dan untuk setiap percobaan, hasil verifikasi sidik jari dicatat sebagai "Terbaca" atau "Tidak Terbaca". Berdasarkan data, seluruh 45 percobaan sidik jari berhasil terbaca, dengan 0 percobaan tidak terbaca. Oleh karena itu, akurasi sensor sidik jari dalam keadaan normal mencapai 100%. Ini menunjukkan bahwa sensor sidik jari AS608 sangat akurat dalam kondisi ideal tanpa gangguan pada jari atau sensor.



Gambar 5.7 Sampel Pengujian Sidik Jari Keadaan Normal.

5.2.2.2 Pengujian Sidik Jari dalam Keadaan Basah

Pengujian sidik jari dalam keadaan basah dilakukan untuk mengevaluasi akurasi sensor sidik jari AS608 dalam kondisi basah. Proses pengujian ini melibatkan tiga percobaan, di mana setiap percobaan terdiri dari lima repetisi. Partisipan akan melakukan verifikasi sidik jari dalam keadaan jari yang basah atau lembab.

Tabel 5.8 Hasil Pengujian Sidik Jari Keadaan Basah

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Terbaca	Tidak	Terbaca	Tidak	Terbaca	Tidak
Hazbi		✓		✓		✓
		✓		✓		✓
		✓		✓		✓
		✓		✓		✓
		✓		✓		✓
Trisucipto	✓			✓		✓
		✓		✓		✓
		✓		✓		✓
		✓	✓			✓
		✓		✓		✓

Faiq		✓		✓		✓
		✓		✓		✓
	✓			✓		✓
		✓		✓		✓
		✓	✓			✓
Hasil Sidik Jari Terbaca						4
Hasil Sidik Jari Tidak Terbaca						40
Akurasi Keakuratan Sensor (%)						8%

Pada Tabel 5.8 menunjukkan hasil pengujian sidik jari dalam keadaan basah untuk tiga partisipan: Hazbi, Trisucipto, dan Faiq, yang masing-masing melakukan tiga percobaan dengan lima repetisi. Berdasarkan data, dari total 45 percobaan, hanya 4 sidik jari yang berhasil terbaca, sementara 40 percobaan gagal terbaca. Hal ini menghasilkan akurasi sensor sidik jari sebesar 8% dalam kondisi basah, yang jauh lebih rendah dibandingkan kondisi normal. Kesimpulannya, keakuratan sensor sidik jari AS608 sangat menurun ketika jari berada dalam keadaan basah atau lembab.

Gambar 5.8 Sampel Pengujian Sidik Jari Keadaan Basah.

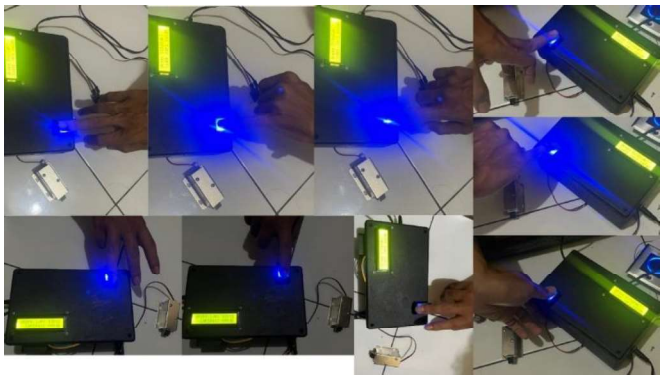
5.2.2.3 Pengujian Sidik Jari dalam Keadaan Kotor

Pengujian sidik jari dalam keadaan kotor dilakukan untuk mengevaluasi akurasi sensor sidik jari AS608 dalam kondisi kotor. Proses pengujian ini melibatkan tiga percobaan, di mana setiap percobaan terdiri dari lima repetisi. Partisipan akan melakukan verifikasi sidik jari dalam keadaan jari yang berdebu dan berpasir.

Tabel 5.9 Hasil Pengujian Sidik Jari Keadaan Kotor

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Terbaca	Tidak	Terbaca	Tidak	Terbaca	Tidak
Hazbi		✓		✓		✓
	✓			✓		✓
		✓		✓		✓
		✓	✓			✓
		✓		✓		✓
Trisucipto		✓		✓		✓
		✓	✓			✓
		✓		✓		✓
		✓		✓		✓
		✓		✓		✓
Faiq		✓		✓		✓
	✓			✓		✓
		✓		✓		✓
		✓		✓	✓	
		✓		✓		✓
Hasil Sidik Jari Terbaca						5
Hasil Sidik Jari Tidak Terbaca						39
Akurasi Keakuratan Sensor (%)						11%

Pada Tabel 5.9 menyajikan hasil pengujian sidik jari dalam keadaan kotor untuk tiga partisipan: Hazbi, Trisucipto, dan Faiq. Masing-masing partisipan melakukan tiga kali percobaan, dengan setiap percobaan terdiri dari lima repetisi. Berdasarkan data, dari total 45 percobaan, hanya 5 sidik jari yang berhasil terbaca, sementara 39 percobaan gagal terbaca. Hal ini menghasilkan akurasi sensor sidik jari sebesar 11% dalam kondisi kotor. Kesimpulannya, seperti halnya kondisi basah, keakuratan sensor sidik jari AS608 juga sangat menurun ketika jari dalam keadaan kotor atau berminyak.



Gambar 5.9 Sampel Pengujian Sidik Jari Keadaan Kotor.

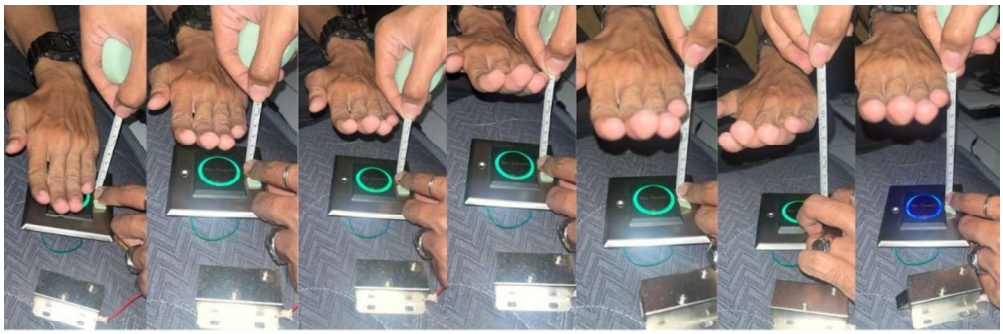
5.2.3 Pengujian Sensor *No Touch*

Proses pengujian untuk sensor *No Touch* dilakukan dengan melakukan pengukuran jarak maksimal sensor dapat membaca objek. Pengukuran ini dilakukan dengan melakukan percobaan di beberapa jarak dan pengujian dilakukan sebanyak tiga kali percobaan.

Tabel 5.10 Hasil Pengujian Jarak Maksimal Sensor *No Touch*

	Pengujian ke-1		**Pengujian ke-2**		**Pengujian ke-3**	
	Terbaca	Tidak	Terbaca	Tidak	Terbaca	Tidak
Jarak 2 cm	✓		✓		✓	
Jarak 4 cm	✓		✓		✓	
jarak 6 cm	✓		✓		✓	
Jarak 8 cm	✓		✓		✓	
Jarak 10 cm	✓		✓		✓	
Jarak 10.5 cm	✓		✓		✓	
Jarak 11 cm		x		x		x

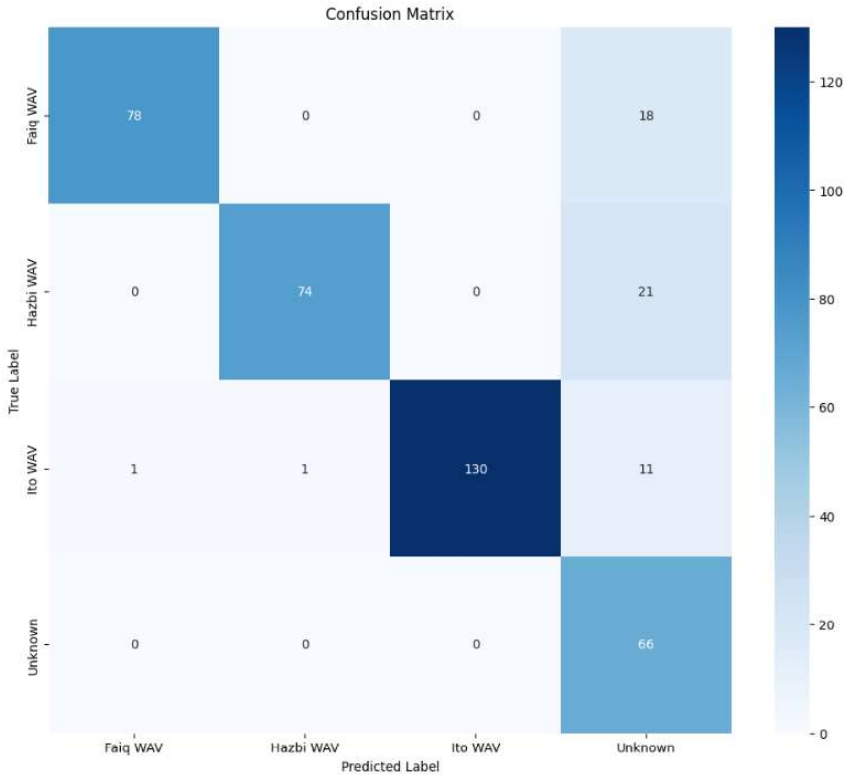
Berdasarkan hasil dari tabel 5.10 menjelaskan bahwa pengujian sensor *no touch* dilakukan untuk menentukan jarak maksimal sensor dapat mendeteksi objek atau tangan. Hasil percobaan menunjukkan bahwa sensor no *touch aktif* dan dapat mendeteksi objek hingga jarak sekitar 10.5 cm. Pada jarak 11 cm, sensor tidak lagi merespons objek. Ini mengindikasikan bahwa jarak operasional efektif sensor *no touch* adalah dalam rentang 0 cm hingga 10.5 cm.



Gambar 5.10 Sampel Pengujian Jarak Sensor *No Touch*.

5.2.4 Pengujian *Machine Learning*

Pengujian untuk *machine learning* ini dilakukan dengan mengumpulkan *dataset* suara dari Hazbi, Trisucipto, Faiq, dan suara dari orang tidak dikenal sebanyak kurang lebih 300 suara dengan kondisi yang berbeda, seperti pengambilan suara ketika keadaan sedang berisik dan perubahan intonasi pada pengambilan *dataset*. Pengambilan *dataset* yang beragam ini diperlukan agar proses *training* untuk model dari *machine learning* dapat membaca dengan baik perbedaan suara dari setiap subjek. *Dataset* yang telah diperoleh akan *di training* oleh model *machine learning* yang digunakan dan dari hasil *training* tersebut akan dilakukan pengujian menggunakan *confusion matrix* dimana seperti yang terlihat pada gambar 5.11.



Gambar 5.11 Hasil Pengujian Menggunakan *Confusion Matrix Voice Recognition*

Dari gambar 5.11, dapat dilihat hasil *confusion matrix voice recognition* yang merupakan visualisasi dari training model *machine learning,* yaitu dengan membandingkan label sebenarnya yang berada pada posisi vertikal dengan label prediksi yang berada pada sumbu horizontal.

Hasil dari matriks ini menunjukkan bahwa model memiliki kinerja yang secara umum sangat baik dan andal. Kekuatan utama model ini terletak pada kemampuannya untuk membedakan antar pengguna terdaftar, seperti 'Faiq WAV', 'Hazbi WAV', dan 'Ito WAV', dengan tingkat kesalahan yang sangat rendah di antara ketiganya. Selain itu, model juga terbukti sempurna dalam mengidentifikasi sampel 'Unknown', di mana tidak ada satupun sampel 'Unknown' yang salah diklasifikasikan sebagai pengguna terdaftar. Meskipun demikian, matriks ini juga mengungkap adanya kelemahan spesifik, yaitu kecenderungan model untuk kurang percaya diri dan mengklasifikasikan suara pengguna terdaftar sebagai 'Unknown', yang paling signifikan terlihat pada 21 sampel 'Hazbi WAV' dan 18 sampel 'Faiq WAV' yang salah diidentifikasi. Secara keseluruhan, confusion matrix ini mengonfirmasi bahwa model pengenalan suara tersebut efektif dan andal dalam tugas identifikasi dan penolakan akses, namun memerlukan perbaikan lebih lanjut untuk mengurangi kesalahan pada pengguna yang seharusnya dikenali.

Tabel 5.11 Hasil Laporan Klasifikasi Setiap Label

Label	Precision	Recall	F1-Score
Faiq WAV	0.98	0.81	0.89
Hazbi WAV	0.98	0.78	0.87
Trisucipto WAV	0.90	1.00	0.95
Unknown	0.57	1.00	0.72

Pada Tabel 5.11, disajikan laporan klasifikasi yang merinci performa model melalui metrik Precision, Recall, dan F1-Score untuk setiap kelas suara. Hasil ini menunjukkan kinerja model yang memiliki keunggulan spesifik sekaligus area yang memerlukan perhatian. Nilai Precision yang sangat tinggi untuk kelas pengguna terdaftar, seperti 'Faiq WAV' (0.98) dan 'Hazbi WAV' (0.98), mengindikasikan bahwa ketika model membuat prediksi identitas, prediksi tersebut sangat akurat dan dapat dipercaya. Di sisi lain, nilai Recall yang sempurna (1.00) untuk kelas 'Trisucipto WAV' dan 'Unknown' menunjukkan kemampuan superior model dalam dua aspek krusial: tidak pernah gagal mengenali pengguna 'Trisucipto' dan secara konsisten berhasil mengidentifikasi setiap suara asing yang sebenarnya.

Selanjutnya, laporan klasifikasi ini juga memperlihatkan potensi optimasi yang jelas pada model. Nilai Recall untuk 'Faiq WAV' (0.81) dan 'Hazbi WAV' (0.78) mengindikasikan bahwa model menerapkan standar kecocokan yang tinggi sebelum mengonfirmasi sebuah identitas, yang merupakan karakteristik dari sistem yang memprioritaskan keamanan. Sifat 'hati-hati' inilah yang membuat beberapa sampel vokal yang sedikit berbeda untuk sementara diklasifikasikan sebagai 'Unknown', yang menjelaskan nilai Precision (0.57) pada kelas tersebut. Hal ini bukanlah sebuah kekurangan fundamental, melainkan sebuah indikasi bahwa dengan menambah variasi data latih untuk kedua pengguna, konsistensi pengenalan dapat dengan mudah disempurnakan tanpa perlu mengorbankan tingkat keamanan yang sudah sangat tinggi.

5.2.5 Pengujian Aplikasi *Smart Door Lock* Melalui QoS

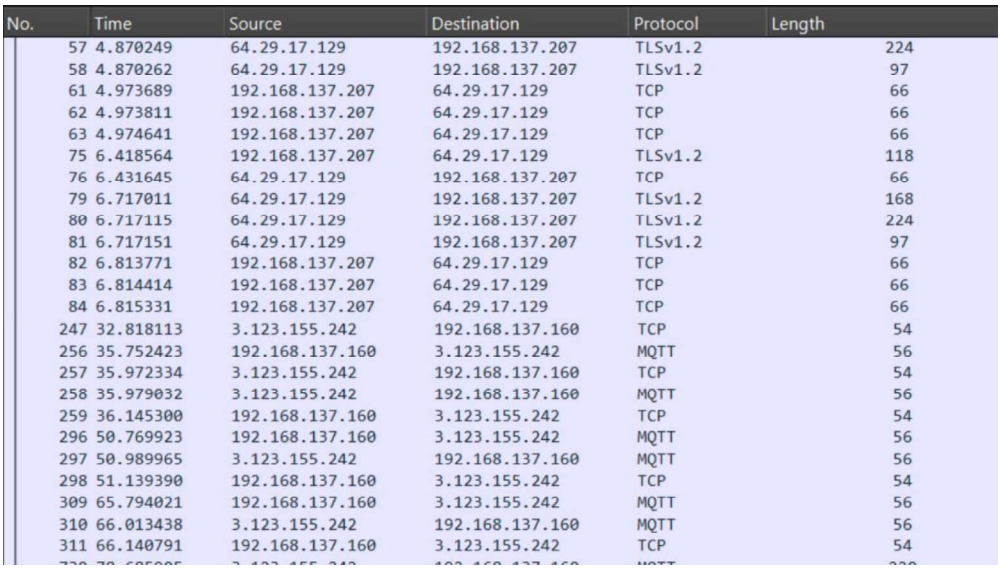
Quality of Service (QoS) merupakan indikator penting dalam mengevaluasi performa jaringan, yang mencerminkan sejauh mana layanan jaringan mampu memenuhi kebutuhan pengguna [17], [18]. Dalam konteks ini, parameter utama yang dianalisis adalah *delay* dan *jitter*. *Delay* menggambarkan waktu yang diperlukan oleh data untuk berpindah dari sumber ke tujuan, sedangkan *jitter* menunjukkan variasi waktu kedatangan antar paket data yang diterima

secara berturut-turut [17]. Proses pengujian ini dilaksanakan dengan bantuan perangkat lunak analisis jaringan, *Wireshark*. Untuk melakukan pengujian, *handphone* dihubungkan ke jaringan *WiFi* yang sama dengan laptop yang sedang menjalankan *wireshark.* Tahapan pengumpulan data dimulai dengan mengoperasikan aplikasi *smart dorm key* pada *handphone*. Setelah data berhasil direkam, selanjutnya data tersebut diolah dan dianalisis menggunakan Microsoft Excel, di mana perhitungan *delay* dan *jitter* dilakukan berdasarkan rumus yang telah ditentukan. Setelah aktivitas penangkapan paket data selama 5 menit, didapatkan hasil bahwa sebanyak 10511 paket berhasil ditransmisikan. Dari paket-paket yang berhasil ditransmisikan ini, terukur nilai *delay* dan nilai *jitter* seperti yang terdapat pada tabel 5.12.

Tabel 5.12 Hasil Perhitungan Nilai QoS Aplikasi *Smart Door Lock*

Parameter	**Delay**	**Jitter**
Hasil	65.84 ms	0.017 ms
Kategori Tiphon	Sangat Bagus	Bagus
Indeks	4	3

Dari hasil pengujian *QoS* aplikasi *smart dorm key* menggunakan *wireshark* seperti yang ada pada tabel 5.12 menampilkan bahwa aplikasi memiliki performa sistem yang unggul dikarenakan nilai dari parameter *delay* ini sendiri mencapai hasil sebesar 65.84 ms, dimana nilai dari *delay* tersebut berada pada indeks 4 yang berarti sangat bagus pada standar TIPHON. Nilai ini menandakan responsivitas sistem yang tinggi dalam memproses kode *IoT* dan *machine learning*, serta efisiensi pengiriman data dengan waktu tunggu yang minim. Di sisi lain, parameter *jitter* juga menunjukkan hasil yang cukup bagus yaitu di angka 0,017 ms dan nilai tersebut berada pada indeks 3 yang berarti termasuk kategori bagus pada standar TIPHON yang berarti transmisi data berjalan stabil karena variasi waktu antar paket data yang tidak signifikan.



No.	Time	Source	Destination	Protocol	Length
57	4.870249	64.29.17.129	192.168.137.207	TLSv1.2	224
58	4.870262	64.29.17.129	192.168.137.207	TLSv1.2	97
61	4.973689	192.168.137.207	64.29.17.129	TCP	66
62	4.973811	192.168.137.207	64.29.17.129	TCP	66
63	4.974641	192.168.137.207	64.29.17.129	TCP	66
75	6.418564	192.168.137.207	64.29.17.129	TLSv1.2	118
76	6.431645	64.29.17.129	192.168.137.207	TCP	66
79	6.717011	64.29.17.129	192.168.137.207	TLSv1.2	168
80	6.717115	64.29.17.129	192.168.137.207	TLSv1.2	224
81	6.717151	64.29.17.129	192.168.137.207	TLSv1.2	97
82	6.813771	192.168.137.207	64.29.17.129	TCP	66
83	6.814414	192.168.137.207	64.29.17.129	TCP	66
84	6.815331	192.168.137.207	64.29.17.129	TCP	66
247	32.818113	3.123.155.242	192.168.137.160	TCP	54
256	35.752423	192.168.137.160	3.123.155.242	MQTT	56
257	35.972334	3.123.155.242	192.168.137.160	TCP	54
258	35.979032	3.123.155.242	192.168.137.160	MQTT	56
259	36.145300	192.168.137.160	3.123.155.242	TCP	54
296	50.769923	192.168.137.160	3.123.155.242	MQTT	56
297	50.989965	3.123.155.242	192.168.137.160	MQTT	56
298	51.139390	192.168.137.160	3.123.155.242	TCP	54
309	65.794021	192.168.137.160	3.123.155.242	MQTT	56
310	66.013438	3.123.155.242	192.168.137.160	MQTT	56
311	66.140791	192.168.137.160	3.123.155.242	TCP	54

Gambar 5.12 Potongan Data Pengujian Nilai *QoS* di *Wireshark*

Seperti yang ditampilkan pada gambar 5.12 merupakan ptongan sampel dari alamat IP yang telah difilter untuk dilakukan pengujian transmisi paket, dimana 192.168.137.160 merupakan alamat IP dari *smart dorm key,* 192.168.137.207 merupakan alamat IP dari aplikasi *smart dorm key* dan 64.29.17.129 merupakan alamat IP dari *firebase,* dengan tujuan untuk mengirim data ke sistem *smart dorm key* melalui infrastruktur *Google Cloud Platform*.

Dengan demikian, hasil pengujian QoS ini menunjukkan bahwa aplikasi *smart dorm key* menunjukkan performa transmisi data yang optimal. Hal ini dapat dilihat melalui nilai delay dari sistem yang berada dalam kategori "sangat bagus", menandakan sistem dari *smart dorm key* memiliki kemampuan untuk memberikan respons yang cukup stabil saat memproses data IoT dan *machine learning*. Selain itu, nilai dari jitter yang tergolong dalam kategori "bagus" membuktikan adanya kestabilan pengiriman data antar paket yang konsisten selama proses transmisi menjadi indikator utama bahwa aplikasi *smart dorm key* yang dikembangkan mampu beroperasi secara optimal. Secara keseluruhan, hasil pengujian ini mengonfirmasi bahwa *quality of service* jaringan aplikasi ini sangat memadai, terutama untuk menunjang komunikasi *real-time* yang difasilitasi oleh *firebase*.

BAB 6
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan serangkaian pengujian yang telah dilakukan pada proyek "Perancangan *Smart Dorm Key* Berbasis *Fingerprint* dan *Voice Recognition* Sebagai Verifikasi Dua Langkah untuk Meningkatkan Keamanan", dapat ditarik beberapa simpulan sebagai berikut:

1. Implementasi verifikasi dua langkah yang fungsional dimana menggunakan sidik jari dan pengenalan suara di mana pengguna harus melakukan verifikasi suara melalui aplikasi seluler terlebih dahulu sebelum melanjutkan ke pemindaian sidik jari, berjalan sesuai dengan rancangan.
2. Sensor sidik jari AS608 menunjukkan akurasi sempurna 100% dalam kondisi normal. Namun, kinerjanya menurun secara drastis pada kondisi jari basah dengan akurasi 8% dan kotor akurasi 11%. Hal ini menandakan adanya keterbatasan signifikan untuk penggunaan di dunia nyata jika kondisi kebersihan dan kelembaban jari tidak ideal.
3. Sensor *No Touch* untuk akses keluar dari dalam ruangan berfungsi dengan sangat baik dan sesuai ekspektasi, dengan jarak deteksi efektif hingga 10.5 cm.
4. Performa verifikasi suara dengan sistem pengenalan suara menunjukkan akurasi tinggi, yaitu 91% untuk suara terdaftar dan 88% dalam mendeteksi suara tidak terdaftar (*unknown*). Namun, performanya menurun pada kondisi lingkungan yang lebih menantang: akurasi turun menjadi 68% (suara terdaftar) di lingkungan berisik dan 73% saat suara pengguna serak.
5. Kinerja model *machine learning* yang sangat andal menggunakan metode MFCC untuk ekstraksi fitur dan arsitektur CNN untuk klasifikasi suara menunjukkan kinerja yang sangat tinggi. Berdasarkan laporan klasifikasi, model mencapai nilai Precision yang cukup tinggi (0.98) untuk kelas 'Faiq WAV' dan 'Hazbi WAV', namun nilai Recall lebih rendah (0.81 dan 0.78) untuk kedua kelas tersebut, yang berbanding terbalik dengan kelas Ito WAV' dan Unknown' dimana nilai dari recall yang cukup tinggi yaitu (1.00) dan nilai precision (0.90) untuk ‘Ito WAV’ dan (0.57) untuk ‘Unknown’.

6. Kualitas layanan QoS pada aplikasi menunjukkan performa transmisi data yang sangat responsif dan stabil. Dengan nilai *delay* rata-rata 65.84 ms (kategori "Sangat Bagus") dan *jitter* 0.017 ms (kategori "Bagus") menurut standar TIPHON, sistem ini terbukti andal untuk komunikasi *real-time* antara aplikasi, *backend*, dan perangkat IoT melalui Firebase.

6.2 Saran

Berdasarkan simpulan dari hasil proyek yang telah dilaksanakan, terdapat beberapa saran untuk pengembangan lebih lanjut agar sistem "*Smart Dorm Key*" menjadi lebih andal, aman, dan ramah pengguna:

1. Peningkatan Robustitas Sensor Sidik Jari : Mengingat kinerja sensor sidik jari AS608 yang menurun drastis pada kondisi basah dan kotor, disarankan untuk mengeksplorasi penggunaan sensor sidik jari dengan teknologi yang lebih canggih dan tahan terhadap kondisi lingkungan, seperti sensor kapasitif atau ultrasonik. Alternatif lainnya adalah menambahkan fitur pelindung pada perangkat keras, seperti penutup sensor atau pemanas mini untuk mengeringkan permukaan sensor.
2. Pengembangan Model Pengenalan Suara yang Lebih Adaptif : Untuk mengatasi penurunan akurasi pada lingkungan berisik dan saat kualitas suara pengguna berubah, disarankan untuk melakukan:
3. Augmentasi *Dataset* : Memperkaya *dataset* pelatihan dengan lebih banyak sampel suara yang direkam dalam berbagai kondisi bising (misalnya, suara keramaian, musik) dan dengan variasi vokal yang lebih beragam (lelah, serak, berbisik).
4. Implementasi Algoritma *Noise Reduction* : Menerapkan teknik penyaringan derau (*noise reduction*) pada tahap pra-pemrosesan sinyal suara sebelum diekstraksi fitur MFCC-nya.
5. Eksplorasi Arsitektur Model Lanjutan : Menguji arsitektur *deep learning* lain yang lebih kompleks dan dirancang khusus untuk pengenalan suara di lingkungan bising, seperti model yang menggunakan mekanisme *attention* atau jaringan saraf berulang (*Recurrent Neural Networks*).
6. Penambahan Sistem Catu Daya Cadangan : Prototipe saat ini bergantung pada sumber daya listrik eksternal melalui adaptor 12V. Untuk mengantisipasi pemadaman listrik yang dapat melumpuhkan sistem keamanan, sangat disarankan untuk mengintegrasikan sistem catu daya cadangan seperti *Uninterruptible Power*

Supply (UPS) atau modul baterai yang dapat diisi ulang (*rechargeable*), sehingga perangkat tetap dapat beroperasi dalam keadaan darurat.

7. Peningkatan Antarmuka dan Pengalaman Pengguna (UI/UX):
    - Aplikasi Seluler: Mengembangkan fitur tambahan pada aplikasi seperti kemampuan admin untuk memberikan akses sementara (*temporary guest access*), melihat riwayat log aktivitas dengan filter yang lebih detail, dan memberikan notifikasi *push* yang lebih informatif jika terjadi kegagalan akses berulang kali.
    - Perangkat Keras: Memberikan umpan balik yang lebih deskriptif pada layar LCD, misalnya, menampilkan pesan seperti "Jari Terlalu Basah" atau "Suara Tidak Jelas" untuk memandu pengguna saat terjadi kegagalan verifikasi.
8. Pengujian Skalabilitas dan Integrasi Sistem: Prototipe ini dirancang untuk satu pintu. Untuk implementasi skala besar di seluruh gedung asrama, perlu dilakukan pengujian skalabilitas untuk memastikan *backend*, *database* Firebase, dan infrastruktur jaringan mampu menangani beban dari ratusan perangkat dan pengguna secara bersamaan. Selain itu, perlu dijajaki kemungkinan integrasi dengan sistem informasi akademik universitas untuk otomatisasi pendaftaran dan penghapusan data pengguna (mahasiswa) berdasarkan status aktif mereka di asrama.

DAFTAR PUSTAKA

[1] Telkom University, “Asrama.” Diakses: 11 Juli 2025. [Daring]. Tersedia pada: https://telkomuniversity.ac.id/asrama/

[2] I. T. Nugraha, R. Patmasari, and A. I. Irawan, "Implementasi Membuka Kunci Pintu Otomatis Menggunakan Face Recognition pada Raspberry Pi Berbasis Internet of Thing," *e-Proceeding of Engineering*, vol. 7, no. 1, pp. 707-715, Apr. 2020.

[3] M. I. Yoren, R. Purnamasari, and E. Suhartono, "Penerapan Metode Histogram Oriented Of Gradients Dan Haar-Cascad Pada Pintu Asrama Pintar Telkom University," *e-Proceeding of Engineering*, vol. 11, no. 6, pp. 6487-6489, Dec. 2024.

[4] D. Gultom dan M. F. Susanto, "Studi Aplikasi Smartlock Pada Pintu Rumah Dengan Arduino Berbasis IoT Dengan Sensor Suara," dalaikhm *Prosiding The 11th Industrial Research Workshop and National Seminar*, Bandung, 2020, pp. 239-245.

[5] R. Arunashmi, S. S. Gouda, dan K. Agrawal, “Speech Recognition Using MFCC & CNN,” *International Journal of Scientific Research and Engineering Development*, vol. 4.

[6] F. D. Adhinata, D. P. Rakhmadani, and A. J. T. Segara, "Pengenalan Jenis Kelamin Manusia Berbasis Suara Menggunakan MFCC dan GMM," *JURNAL DINDA*, vol. 1, no. 1, pp. 1-6, 2021.

[7] Y. LeCun, Y. Bengio, dan G. Hinton, “Deep Learning,” *Nature*, vol. 521, hlm. 436–444, Mei 2015, doi: 10.1038/nature14539.

[8] S. Swaminathan dan B. R. Tantri, “Confusion Matrix-Based Performance Evaluation Metrics,” *African Journal of Biomedical Research*, vol. 27, hlm. 4023–4031, Nov 2024, doi: 10.53555/AJBR.v27i4S.4345.

[9] D. A. Pramudhita, F. Azzahra, I. K. Arfat, R. Magdalena, dan S. Saidah, “Strawberry Plant Diseases Classification Using CNN Based on MobileNetV3-Large and EfficientNet-B0 Architecture,” *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 9, no. 3, hlm. 522–534, Jul 2023, doi: 10.26555/jiteki.v9i3.26341.

[10] Lumbatech News, “Menjadi Pelopor Tipe Push Pull, Digital Smart Lock Samsung SHS-P718 Masih Diminati,” Lumbatech. Diakses: 11 Juli 2025. [Daring]. Tersedia pada: https://www.lumbatech.com/news/read/109/menjadi-pelopor-tipe-push-pull-digital-smart-lock-samsung-shs-p718-masih-diminati

[11] Nuraini Desyinta, “Rekomendasi Smart Door Lock Anti Bobol, Punya Fitur Alarm sampai Pemantauan Jarak Jauh,” Hypeabis.id. Diakses: 11 Juli 2025. [Daring]. Tersedia pada: https://hypeabis.id/read/38536/rekomendasi-smart-door-lock-anti-bobol-punya-fitur-alarm-sampai-pemantauan-jarak-jauh

[12] Eraspace, “IT Smart Door Lock A230 Handle Wifi Fingerprint RFID NFC IP55,” Eraspace. Diakses: 11 Juli 2025. [Daring]. Tersedia pada: https://livewithit.eraspace.com/product/it-smart-door-lock-a230-handle-wifi-fingerprint-rfid-nfc-ip55

[13] Kementerian Komunikasi dan Informatika Republik Indonesia, “Peraturan Menteri Komunikasi dan Informatika Republik Indonesia No. 5 Tahun 2020 tentang Tata Kelola Penyelenggaraan Sistem Elektronik,” 2020.

[14] Dewan Perwakilan Rakyat Republik Indonesia, “Undang-Undang Republik Indonesia No. 27 Tahun 2022 tentang Perlindungan Data Pribadi,” 22M.

[15] R. A. Aprilianto, “Systematic Literature Review: Smart and Secure Locker System”, doi: 10.25105/jyy1vs09.

[16] D. Nagajyothi and P. Siddaiah, "Speech Recognition Using Convolutional Neural Networks," *International Journal of Engineering & Technology*, vol. 7, no. 4.6, pp. 133-137, 2018.

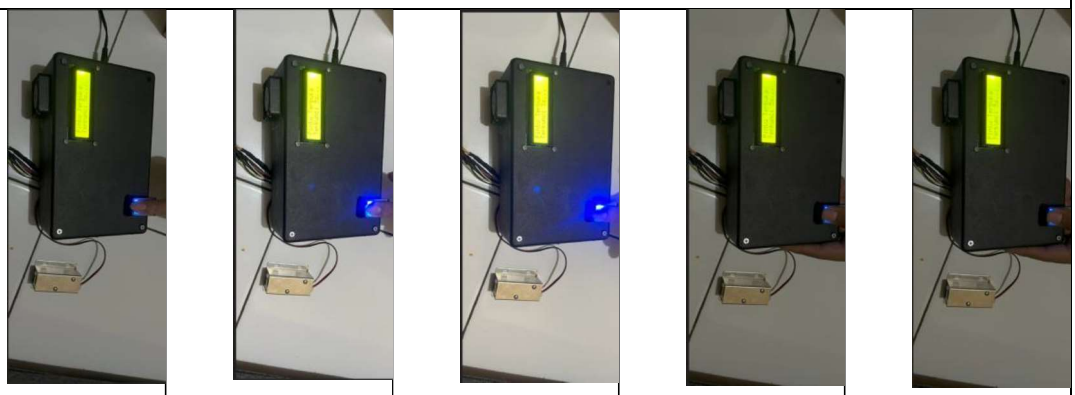
[17] R. Wulandari, “ANALISIS QoS (QUALITY OF SERVICE) PADA JARINGAN INTERNET (STUDI KASUS : UPT LOKA UJI TEKNIK PENAMBANGAN JAMPANG KULON-LIPI),” 2016.

[18] A. Surahman, F. Trias Pontia,) Jurusan, dan T. Elektro, “ANALISIS QUALITY OF SERVICE (QOS) VIDEO CONFERENCE PADA JARINGAN INTERNET DENGAN MENGGUNAKAN AKSES WIMAX (WORLD WIDE INTEROPERABILITY FOR MICROWAVE ACCESS).”

[19] V. H. Phung and E. J. Rhee, "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets," *Appl. Sci.*, vol. 9, no. 21, p. 4500, Oct. 2019.

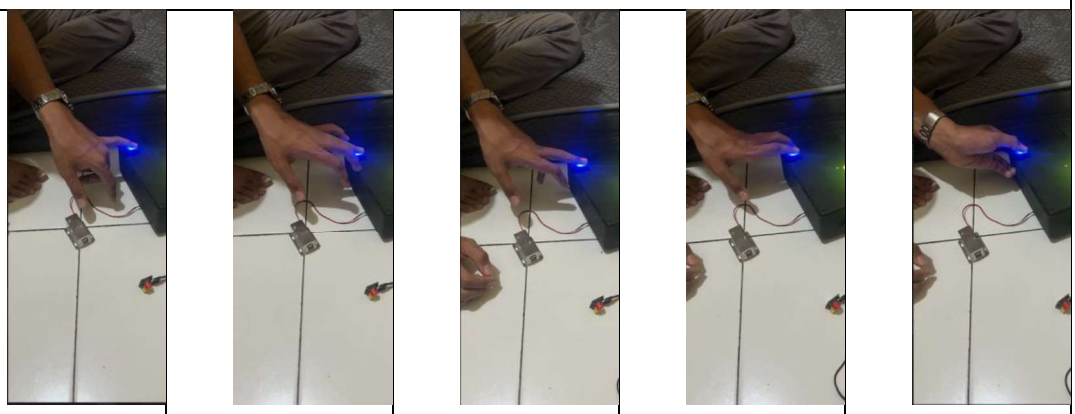
LAMPIRAN 1

M.Trisucipto



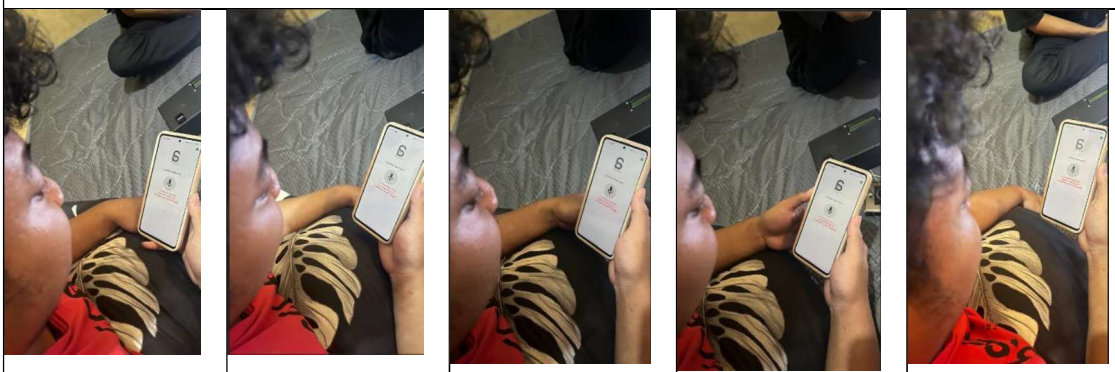
M.Hazbi Ashidiqi

DTM Faiq Zariaqwila

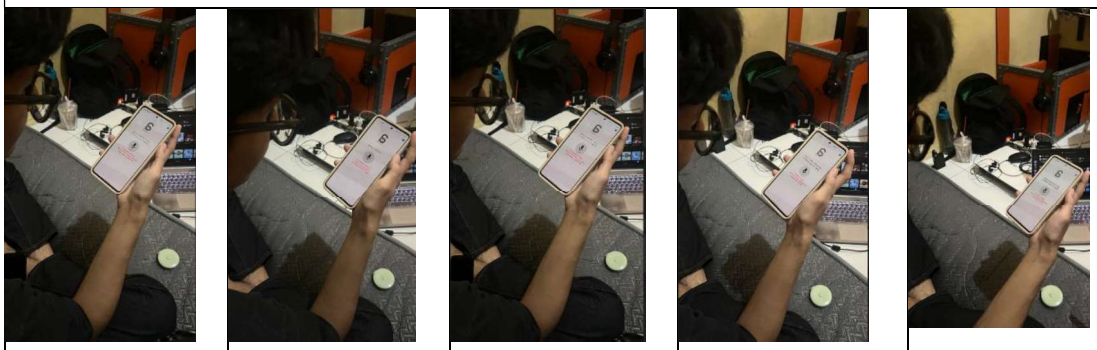


LAMPIRAN 2

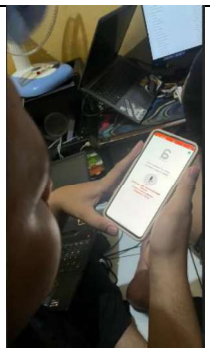
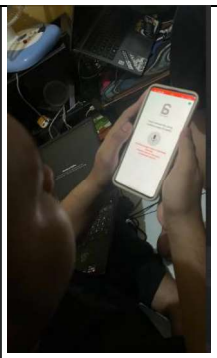
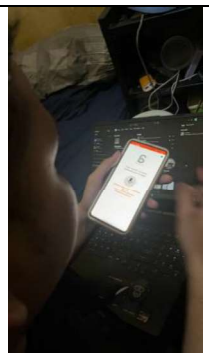
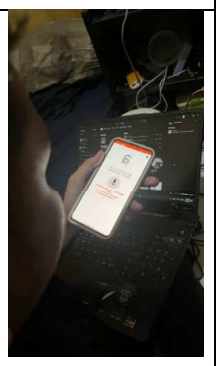
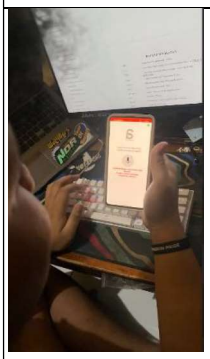
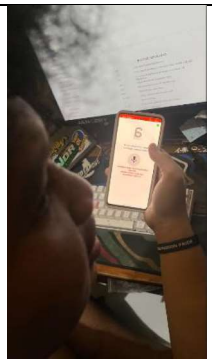
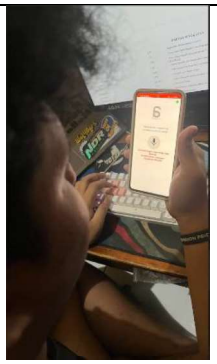
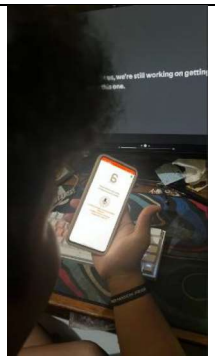
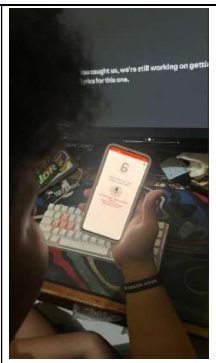
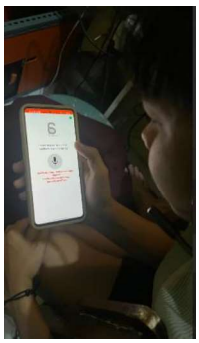
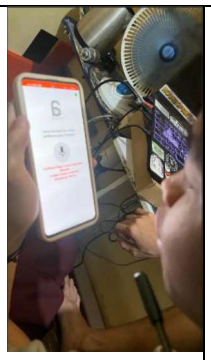
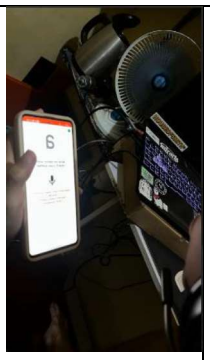
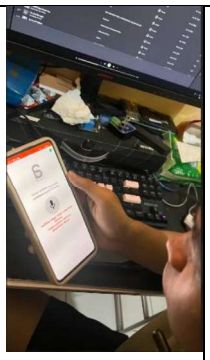
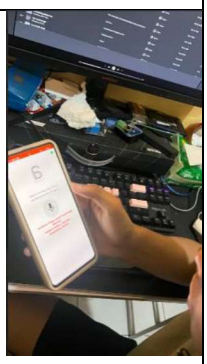
M.Trisucipto



M.Hazbi Ashidiqi



DTM Faiq Zariaqwila

Rayzi Derlamta Ginting				
				
Andi Muh Naufal Dzaky				
				
Ikratul Sarjono				
				

LAMPIRAN 3

Demo Video Hasil Pegujian :

https://drive.google.com/drive/u/0/folders/1msr47_rotY_LPLRwdF9YBFNKH_ZsumLy