Faculty of Computers and Data Science
Program of Computing and Data Science
Alexandria University

Leap: Smart Solutions for Smart Homes

Submitted by:

Ayatullah Nabil El-Sayed El-Banhawy	20221452375
Aysel Mahmoud Hassan Mohamed El-Masry	20221465749
Habiba Mohammad Attia Mohammad	20221379966
Abdelrahman Ahmed Mohamed Ibrahim	20221460100
Abdelrahman Mohamed Abd El-Fattah Sayed	20221459795
Mohamed Hassan Abdullah Hassouna	20221458620
Mohamed Mostafa Kamel Raslan Ouf	20221459902
Muhannad Khaled Abdelmohsen El-kholy	20221459941

Supervised by:

Dr. Magda Madbouly

Acknowledgment

Thanks God, because he's graced our lives with opportunities that we know are not of our hands or of any other hand. He's shown us that it's a scientific fact that gratitude reciprocates. This final moment doesn't just belong to one of us. We wouldn't be up here if it wasn't for some very important people in our lives. To the ones who stayed with us till the moment of our achievement, families, professors, and friends, after God for sure, we want to say

“Thank you all for your invaluable love, support and guidance.”

And for sure our special thanks go to Dr. Magda Madbouly for her supervision and assistance in this project.

Abstract

List of Figures

List of Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
BMS	Battery Management System
BILSTM	Bidirectional long short term memory
CCPA	California Consumer Privacy Act
CD	Continuous Delivery
CI	Continuous Integration
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DC	Direct Current
DOF	Degrees of Freedom
DOM	Document Object Model
DPDT	Double Pole Double Throw
DTW	Dynamic Time Warping
ECCV	European Conference on Computer Vision
EL	Electroluminescent
ELIZA	Early chatbot developed by Weizenbaum (1966)
EMG	Electromyography
ESP	Espressif (e.g., ESP8266/ESP32 microcontrollers)
GDPR	General Data Protection Regulation
GET	HTTP GET method
GND	Ground
GPT	Generative Pre-trained Transformer
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ID	Identifier
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IMU	Inertial Measurement Unit
IN	Input
IP	Internet Protocol
JPEG/JPG	Joint Photographic Experts Group
JS	JavaScript

JSON	JavaScript Object Notation
JST	Japan Standard Time or JST Connector (context-dependent)
LFW	Labeled Faces in the Wild
LLM	Large Language Model
LPU	Low Power Unit
LSTM	Long Short-Term Memory
MAC	Media Access Control
MB	Megabyte
MDPI	Multidisciplinary Digital Publishing Institute
MFA	Multi-Factor Authentication
NCBI	National Center for Biotechnology Information
NLP	Natural Language Processing
NN	Neural Network
NOW	ESP-NOW (Espressif Wireless Protocol)
OCR	Optical Character Recognition
ONNX	Open Neural Network Exchange
OTP	One-Time Password
OUT	Output
PCB	Printed Circuit Board
PH	Potential of Hydrogen (Acidity scale)
PIN	Personal Identification Number
PNG	Portable Network Graphics
POST	HTTP POST method
PUT	HTTP PUT method
QR	Quick Response (code)
RAG	Retrieval-Augmented Generation
RCE	Restricted Coulomb Energy
RESET	System Reset
REST	Representational State Transfer
RX	Receive
SCL	Serial Clock Line
SDA	Serial Data Line
SR	Speech Recognition
SVM	Support Vector Machine
TTS	Text-to-Speech
TX	Transmit
UI	User Interface
USB	Universal Serial Bus
UX	User Experience

Chapter 1

Introduction

1.1 Problem Statement

Today, most smart home systems are almost exclusively dependent on voice commands and mobile applications as their primary interface. These interfaces, effective for most users, create a big accessibility barrier for people with disabilities, such as speech impairments, or in noisy environments where voice recognition systems usually fail. Moreover, mobile applications need a smartphone and technical competence in using complicated interfaces, which could be an especially difficult task for an older adult or somebody with poor technological literacy.

Another major problem is the lack of interoperability and integration among different smart home systems. To be sure, most of the already existing solutions are built in silos; that is, their separate devices and applications work in isolation, governing different functionalities such as security, automation, and health monitoring. This leads to a fragmentation that causes inefficiencies, increases costs, and creates a poor user experience, making it hard for the users to manage their ecosystem in the smart home efficiently. This shows the requirement for an innovative solution able to address accessibility concerns, offer seamless interaction, and integrate diverse functionalities into one coherent system.

1.2 Motivation and Justification

The project is motivated by the requirement to design an inclusive, accessible, and user-friendly approach to smart home automation. It looks into the challenges posed by the current systems and seeks to provide a solution that combines innovation with practicality. Equipped with gesture recognition, this smart ring makes it possible for one to control his or her smart home devices most naturally and intuitively, especially for people who have difficulties with voice commands or even mobile interfaces.

The integrated facial recognition system also allows for added security in personalized access and sends instant alerts in the event of unauthorized entries. Meanwhile, the conversational AI-powered chatbot responds to the users with instant help, answers to questions, and personalized suggestions, hence enhancing the experience. To make all these features easily accessible and manageable, the project entails a website and mobile application as core platforms. These platforms serve as single-control dashboards where users can set up and monitor their devices easily, either at home or remotely.

1.3 Goals and Objectives

Leap is designed to redefine the field of smart home automation with the first integrated, user-centered system in merging gesture recognition, facial recognition, and conversational AI with IoT technologies. The ultimate goal is to provide a seamless, secure, and accessible solution, delivering a better experience in the interaction between users and their living environment.

The core part of this project will be the development of a smart Ring that can recognize and interpret hand gestures. This feature overcomes some of the drawbacks of using traditional control methods, such as voice commands and mobile apps, by offering an intuitive, hands-free alternative able to respond to a wide array of user needs, including persons with disabilities or being in noisy environments. With accuracy and responsiveness assured, this gesture recognition system will allow for the effortless control of smart home devices such as lighting, thermostats, and doors.

This would further enhance home security with a state-of-the-art facial recognition system. Using TensorFlow, OpenCV, and Keras, this module is going to be able to identify authorized persons in real time and alert the homeowner about any unauthorized entry; it would adapt to changing lighting conditions in order to guarantee consistent performance. All these really make the smart home experience quite safer and much more tailored.

The conversation will be even richer with the aid of a conversational AI-driven chatbot that will offer live help, responding to queries and giving recommendations as needed. Available via both voice and text, this chatbot will become the easy-to-use interface for all smart home devices and support requests.

The Leap project will also have its mobile application and website, developed for the management of the system in a centralized way. It would be used in the control of devices, interacting with the chatbot, setting of security, and monitoring the system's performance. A strong focus will be put on making the platforms user-friendly, accessible, and compatible on a wide variety of devices to make smart home management easy for everyone.

The architecture of IoT will be the backbone in ensuring seamless communication between the smart Ring, facial recognition system, chatbot, and connected devices. Firebase will be utilized in achieving real-time data synchronization, while ESP-NOW and Wi-Fi will be used for low-latency communication. The architecture will provide reliable, scalable operation with the possibility of adding more devices and features to the system in the future.

The Leap project is uniting all these technologies with the view to achieve a holistic and adaptive solution able to handle changing demands from today's smart home for which there must be accessibility, security, and usability for all types of users.

Chapter 2

Literature Review

The **Leap project** builds upon a rich foundation of research and technological advancements in the fields of gesture recognition, facial recognition, chatbot development, and smart home automation. This chapter provides a comprehensive review of the existing literature, highlighting the key technologies, methodologies, and challenges that have shaped the development of the Leap system. By examining prior work, this chapter positions the Leap project within the broader context of smart home technologies and identifies the gaps that the project aims to address.

2.1 Gesture Recognition in Smart Home Systems

2.1.1 Abstract

The *Leap Smart Ring* project introduces a gesture recognition module as a core component of a unified smart home automation system. Designed for accessibility and responsiveness, the system allows users to control household devices through intuitive hand movements captured by embedded IMU sensors. Initially based on a classical **RCE + DTW** approach, the project evolved into a robust **Bidirectional LSTM based deep learning model**, significantly enhancing accuracy and generalization across users. Leveraging real-time data streaming via **WebSocket**, inference through a **FastAPI** endpoint, global **StandardScaler** normalization, and synchronized smart device control via **Firebase**, the system ensures seamless operation in dynamic home environments. This report presents comprehensive coverage of the dataset structure, preprocessing pipeline, model selection and training, system integration, deployment architecture (local and cloud), encountered challenges, and future development directions including on-device inference, user-specific gesture calibration, and interface customization.

Smart home technologies have transformed the way we interact with our living environments. Traditionally, the primary interfaces for controlling smart devices have been voice assistants and mobile applications. While effective for many users, these methods present significant limitations in real-world usage. **Voice control systems** fail in noisy environments, such as kitchens or open public areas, and are inaccessible to users with speech impairments. Similarly, **mobile apps** require both device availability and a certain level of technological fluency, which may exclude elderly individuals or those with physical disabilities.

To overcome these limitations, the *Leap* project introduces a **gesture recognition system** embedded within a smart wearable ring. This system enables users to control smart home devices through intuitive hand movements without relying on speech or touch-based interfaces. **Gestures**

are a universal form of communication, inherently language-independent, and culturally adaptable, making them an ideal modality for inclusive design.

2.1.2 System Architecture & Evolution

At the core of the Leap ring lies the **MPU6050 Inertial Measurement Unit (IMU)**, which captures both acceleration and gyroscopic data from hand movements. This data is transmitted in real time via the **ESP8266** microcontroller over a **WebSocket** connection to a local or cloud-based server. Once received, the motion data is **preprocessed** (outlier removal, median filtering, global normalization, sliding-window segmentation) and classified into gesture types using a **Bidirectional LSTM model**.

2.1.3 Shift from Classical to Deep Learning Models

Initially, the project proposed using a classical model combining **Dynamic Time Warping (DTW)** for sequence alignment and **Restricted Coulomb Energy (RCE)** Neural Networks for classification. While theoretically sound for small, static datasets, this approach proved inadequate in practice. It was highly sensitive to variability in gesture speed, personal motion style, and sensor noise. Furthermore, **DTW-based systems** do not scale well to real-time applications due to computational overhead, and **RCE models** offer limited generalization beyond their training data.

These limitations prompted a shift toward a deep learning approach. The final system is built upon a **Bidirectional Long Short-Term Memory (LSTM)** neural network. This architecture enables the model to learn motion dynamics directly from time-series IMU data and adapt to variable gesture styles and speeds. Compared to DTW + RCE, the **Bidirectional LSTM model** provided a significant boost in:

- **Classification accuracy** (95–98% vs. ~81%)
- **Robustness to inter-user gesture variation**
- **Real-time responsiveness and scalability**

2.2 Face Recognition System for Home Security

Facial recognition has rapidly matured into a critical technology for home security, prized for its contact-free convenience, seamless user experience, and robust automated authentication. Unlike traditional mechanisms like keys, cards, or PINs, face-based systems offer a frictionless experience without sacrificing security.

Recent years have witnessed substantial advances driven by deep learning. One study on home security applications demonstrated a hybrid Fisher Linear Discriminant approach combined with deep learning and IoT for real-time smart-home surveillance, showing improved detection accuracy and robustness under varying conditions (Anandan et al. 2024) [5][6]. Another IoT-enabled smart-lock system utilized Dlib with SVM classification, enabling reliable face-based door access via Android devices [7]

Beyond application-specific systems, broader empirical studies highlight the growing traction of deep learning in facial recognition. A systematic review by MDPI in 2023 emphasized the performance gains of convolutional neural networks in security-sensitive environments, while flagging concerns such as privacy, dataset bias, and adversarial vulnerability [8][9]. Privacy-preserving methods have gained attention as well, ECCV 2024 introduced "P3-Mask," a personalized perturbation technique effective at thwarting unauthorized face recognition while preserving visual quality [10][11].

State-of-the-art models like InsightFace, built using massive datasets like Glint360K (17 million images, over 360,000 identities), now set the standard for embedding-based face recognition [12][13]. These models offer generalized, high-accuracy performance across varied demographic groups, addressing criticisms of classical methods like Eigenfaces, Fisherfaces, and Local Binary Patterns, which falter under occlusion, pose variation, or poor illumination.

Simultaneously, applications specifically tailored for home environments are emerging. A study published in NCBI (2024) combined anomaly detection with face recognition in IoT devices, enabling subtle yet effective detection of unauthorized presence [14]. This reflects an evolving industry trend: integrating biometric security with intelligent event detection.

The literature consistently shows that leveraging **YOLOv8** (or similar detectors) together with deep embedding models like **InsightFace** enables fast, accurate, and scalable recognition suitable for real-world deployment. This combination allows live-stream processing, multi-face detection in frames, and adaptability to diverse lighting and pose scenarios, while preserving user data locally and securing biometric templates.

Ethical and privacy issues data storage, consent, bias are also thoroughly studied. Surveys note risks of demographic bias and emphasize the adoption of encryption, local processing, and transparency policies to comply with Privacy by Design principles.

Recent literature supports the architecture of Phase 2 in the Leap project: a hybrid system combining high-performance detection (**YOLOv8**), deep identity embeddings (**InsightFace** with **Glint360K**), and secure local/cloud data handling, offering fast, scalable, and privacy-preserving face recognition tailored for smart home security.

2.3 Chatbots for Smart Home Assistance

Chatbots have become an integral part of modern smart home systems, providing users with real-time assistance and personalized recommendations using text or speech. Powered by natural language processing (**NLP**) models, chatbots can understand and respond to user queries in natural language, making them a valuable addition to the Leap project.

2.3.1 Evolution of Chatbots and the Rise of LLM-Powered Smart Assistants

The evolution of chatbot technology has followed a clear trajectory from simple, rule-based systems to sophisticated, language-model-driven conversational agents that now power smart assistants in diverse domains. Early systems such as **ELIZA** (Weizenbaum, 1966) operated on

hardcoded templates and keyword-matching logic. Although they demonstrated the basic principles of human-computer dialogue, their functionality was rigid and unable to scale to varied or unpredictable user inputs.

The next major step was the adoption of **retrieval-based** methods, where chatbots selected the most appropriate response from a predefined dataset using similarity metrics and shallow machine learning models. While these systems offered improved relevance and fluency, they remained constrained by a **limited response pool** and an inability to generalize or generate new content.

The transition to **generative models**, particularly with the introduction of **Seq2Seq architectures** and later **transformer-based models**, marked a breakthrough in conversational AI. Chatbots like Google's **Meena**, OpenAI's **GPT-2**, and Facebook's **BlenderBot** showcased the ability to produce context-aware, human-like responses. However, these early generative systems often struggled with consistency in multi-turn dialogues and the generation of factually accurate responses.

Recent advances in **Large Language Models (LLMs)** such as **ChatGPT**, **Claude**, and open-weight models like **LLaMA 3** have significantly advanced the capabilities of chatbots. These models are pre-trained on vast and diverse corpora and fine-tuned for conversation, allowing them to handle complex interactions, adapt to user preferences, and maintain contextual continuity across multiple dialogue turns. In the context of the Leap project, such LLMs form the backbone of the assistant's reasoning engine powering functionalities such as general question answering, smart home command interpretation, and recipe-based recommendation through Retrieval-Augmented Generation (RAG).

Modern LLM-powered chatbots offer **flexibility, domain adaptability, multilingual support, and real-time responsiveness**, making them particularly suited for **interactive environments like smart homes**. Their ability to integrate with external services (e.g., Firebase, APIs, IoT platforms) enables seamless voice and text-based control, personalization, and contextual decision-making. Compared to previous generations, these systems exhibit a paradigm shift not only in linguistic fluency but also in how they engage with dynamic user environments and domain-specific data sources.

This progression from static, rule-based logic to dynamic, LLM-driven architectures highlights the transformational impact of deep learning on conversational systems. For projects like Leap, the adoption of LLaMA 3-based pipelines represents a critical enabler for building smart, interactive, and scalable assistants that align with real-time user needs.

2.3.2 Language Model Selection: LLaMA 3

The language model selected for this project is **LLaMA 3** (Large Language Model Meta AI, version 3), specifically the **70 billion parameter variant**. Developed by Meta AI, LLaMA 3

represents the **state-of-the-art** in open-weight transformer-based models, offering **exceptional performance** in both general and specialized language understanding tasks. One of the primary motivations for adopting LLaMA 3 is its strong performance on standard benchmarks, where it competes closely with proprietary models such as **GPT-4** and **Claude 3**. This makes **LLaMA 3** a reliable alternative for high-level language reasoning, classification, summarization, and structured response generation in academic and industrial applications.

Technically, **LLaMA 3** utilizes a dense transformer architecture with extended **context length support up to 8192 tokens**, enabling it to **reason effectively over long-form input**, including **multi-turn dialogues** and **document-based prompts**. The model has been instruction-tuned to follow natural language queries with improved coherence and reduced hallucination, which is particularly advantageous for conversational agents. Additionally, its **open-source licensing** allows **full transparency, reproducibility, and local customization**, making it ideal for research-grade deployments in projects. Overall, LLaMA 3's ability to generalize across multiple domains and its compatibility with modern NLP frameworks like **LangChain** make it a powerful backbone for intelligent assistant systems.

2.3.3 Local vs. Cloud-Based Model Deployment

The deployment strategy of large language models (LLMs) is a critical design choice that significantly impacts system performance, scalability, security, and operational complexity. Two principal paradigms exist: **local (on-device or on-premise)** deployment and **cloud-based** inference. Local deployment involves running models directly **on edge devices** or **user-controlled servers**. This approach offers maximum control over data privacy and offline accessibility, which is particularly valuable in constrained or sensitive environments (Lin et al., 2022). Tools such as **Ollama** has recently emerged to simplify local deployment of transformer models like LLaMA, enabling real-time inference on consumer-grade **GPUs** and even **CPUs**. However, local deployment is typically limited by **hardware constraints**, making it less suitable for hosting large models (e.g., LLaMA 3–70B) due to high memory and computational demands.

In contrast, **cloud-based deployment** leverages high-performance infrastructure provided by cloud vendors to serve LLMs as APIs. Platforms such as **Groq Cloud**, OpenAI, and AWS SageMaker offer scalable environments optimized for transformer inference. Cloud deployment provides several advantages, including **low latency**, **high throughput**, **automatic scaling**, and simplified integration via RESTful APIs. Furthermore, it abstracts away infrastructure management, allowing developers to focus on application logic rather than hardware provisioning. However, cloud services may introduce concerns related to **data sovereignty**, **latency variability** depending on network conditions, and recurring costs for large-scale usage (Zhao et al., 2023).

Ultimately, the choice between local and cloud deployment depends on the **use case requirements**. Cloud platforms are generally preferred for **production-grade, real-time**

applications with high user concurrency, while local deployment remains advantageous for **development, prototyping, or privacy-preserving scenarios**. In academic or research projects—such as the one presented in this study cloud-based deployment via **Groq was selected due to its superior performance and integration with LangChain**, enabling seamless experimentation with large transformer models without the overhead of hardware management.

2.3.4 Key Challenges in Conversational AI for Smart Assistants

The development of intelligent conversational agents for smart environments introduces a range of technical and architectural challenges that must be addressed to ensure effectiveness and user satisfaction. Among these, **contextual understanding**, **personalization**, and **real-time performance** stand out as core dimensions influencing chatbot usability and reliability.

Contextual understanding refers to a chatbot's ability to maintain dialogue coherence across multiple conversational turns. Unlike single-turn models, multi-turn systems require maintaining state and long-term dependencies to generate relevant, non-repetitive responses. Previous studies have highlighted the importance of memory-augmented models and transformer-based architectures to capture this continuity (Wolf et al., 2019). In the case of the Leap project, this challenge is addressed through the integration of conversational memory buffers using LangChain's ConversationBufferMemory, which enables the chatbot to retain previous interactions and respond accordingly.

Personalization is another critical aspect, where the chatbot adapts its responses based on the user's history, preferences, and behavioral patterns. Personalized conversational agents have been shown to improve user engagement and satisfaction, particularly in smart home contexts where users have recurring routines and preferences (Zhou et al., 2020). The Leap system aims to implement user profiling features that allow it to tailor suggestions (e.g., preferred lighting settings or frequently used commands) to individual users, enhancing the perceived intelligence of the assistant.

Finally, **real-time performance** is essential for maintaining fluid interaction. High-latency responses can disrupt the user experience and reduce system trustworthiness. Efficient response generation depends on both optimized backend inference and fast data synchronization with IoT devices. The Leap chatbot addresses this by deploying its language model on Groq Cloud for ultra-low-latency inference and using Firebase for real-time data updates, ensuring that commands (e.g., turning lights and devices on/off) are executed with minimal delay.

Together, these challenges reflect the multidimensional complexity of building robust conversational AI systems. Addressing them requires the integration of state-of-the-art NLP models, adaptive user modeling, and low-latency infrastructure—a combination that the Leap project aims to achieve through its modular, cloud-enhanced architecture.

2.4 Text-to-Speech and Speech Recognition in Conversational AI

Speech recognition (SR) and text-to-speech (TTS) technologies play a pivotal role in enhancing the interactivity and accessibility of modern conversational AI systems. By enabling machines to both understand spoken input and respond with synthesized audio, these technologies establish a natural, bidirectional communication channel between users and machines. In web-based applications developed using frameworks like Flask, these components can be integrated seamlessly to support voice-based interaction over HTTP protocols.

On the input side, **speech recognition** is typically implemented using Python's speech_recognition library, which provides access to powerful cloud-based engines such as Google's Web Speech API. The process involves capturing user audio via a web interface, sending it to the Flask backend, and converting it to text using methods like recognize_google(). This enables real-time interpretation of English voice commands in lightweight setups, eliminating the need for local model hosting.

Conversely, the **text-to-speech** component is implemented using tools such as **Google Text-to-Speech (gTTS)**, which generates natural-sounding English audio from chatbot responses. Once a response is generated by the language model (e.g., LLaMA 3), it is passed to the TTS engine, synthesized into an .mp3 or .wav file, and streamed back to the user through the web interface.

This combined SR–TTS pipeline enables **end-to-end voice interaction**, where users can speak naturally to the system and receive spoken responses in return. Libraries such as pydub are used for audio preprocessing, including trimming, format conversion, and playback optimization. These technologies form a robust, interactive foundation for voice-controlled assistants, enhancing both usability and accessibility without the need for heavy client-side infrastructure.

2.5 Integration of Smart Home Technologies

The integration of gesture recognition, facial recognition, and chatbot technologies into a unified smart home system presents both opportunities and challenges. While each technology has its own strengths, combining them into a cohesive system requires careful design and implementation.

2.5.1 System Integration Challenges

- **Interoperability**: Ensuring that different components of the system, such as the smart Ring, facial recognition system, and chatbot, can communicate seamlessly is critical. This requires the use of standardized communication protocols and APIs [19]. The Leap project employs a modular architecture with layered communication protocols to ensure seamless interoperability.
- **Scalability**: The system must be scalable, allowing for the addition of new devices and functionalities in the future. This requires a modular architecture that can accommodate

changes without disrupting existing operations [20]. The Leap project is designed with scalability in mind, enabling future expansions such as voice assistant hardware and advanced health monitoring features.

- **User Experience**: The integration of multiple technologies must not compromise the user experience. The system should provide a consistent and intuitive interface across all components [21]. The Leap project prioritizes user-centric design, ensuring that the system is easy to use and accessible to all users.

2.5.2 Prior Work in Smart Home Integration

- **Unified Control Platforms**: Several smart home platforms, such as Google Home and Amazon Alexa, have attempted to integrate multiple control methods, including voice commands, mobile apps, and gesture recognition. However, these platforms often lack the depth of integration and personalization offered by the Leap project [22].
- **Wearable Devices for Smart Homes**: Wearable devices, such as smartwatches and fitness trackers, have been explored as control mechanisms for smart homes. However, these devices are often limited in their functionality and do not offer the same level of precision and customization as the smart Ring [23]. The Leap project addresses this limitation by introducing gesture-based rings and a smart band as the central hub for home automation.

2.6 Gaps in Existing Systems

Despite the advancements in gesture recognition, facial recognition, and chatbot technologies, several gaps remain in existing smart home systems:

1. **Limited Inclusivity**: Many smart home systems rely heavily on voice commands and mobile apps, which may not be suitable for users with disabilities or those in noisy environments. The Leap project addresses this gap by offering gesture-based control as an inclusive alternative.
2. **Fragmentation**: Existing systems often operate in silos, with separate devices and applications for different functionalities. This fragmentation leads to inefficiencies and a lack of integration. The Leap project integrates gesture recognition, facial recognition, and chatbot technologies into a unified system, providing a seamless user experience.
3. **Energy Efficiency**: Wearable devices and other hardware components must be energy-efficient to ensure continuous operation. Many existing systems fail to prioritize power management, leading to frequent recharging or battery replacements. The Leap project optimizes energy consumption through efficient hardware design and firmware optimization.

Chapter 3

System Design and Methodology

This chapter provides an in-depth overview of the system design, and the methodology employed in the Leap project. It elaborates on the integration of the gesture recognition system, facial recognition system, and the chatbot, each of which plays a critical role in the smart ring-based control system. The chapter explains the architecture, workflow, datasets, data preprocessing techniques, model selection, and tools used in developing the Leap system.

3.1 Overview of the System Architecture

The Leap system has a modular architecture that is scalable, flexible, and hence easy to integrate into the framework of smart home control. It contains three main components performing different roles in the framework:

First, Leap Smart Ring is equipped with an MPU6050 sensor (3-axis gyroscope and 3-axis accelerometer) mounted on the index finger to capture fine-grained hand movement data. This data is streamed in real-time over **WebSocket** using an **ESP8266 microcontroller**, which transmits structured IMU data to a central processing backend. The system buffers the data into 50-sample sliding windows (~0.5s) and processes it through a **BiLSTM + Attention** deep learning model trained on labeled motion sequences. The prediction output is interpreted as gesture commands that control smart devices such as lights and doors through Firebase Realtime Database. This pipeline ensures **low latency**, **on-demand responsiveness**, and **robust performance** across varying gesture speeds and styles.

Second, face recognition module is designed as a **real-time, on-device** security mechanism integrated into the smart home interface. It uses **YOLOv8** for high-speed and accurate face detection and **InsightFace's Glint360k ONNX model** for generating 512-dimensional face embeddings. The embeddings are compared using **cosine similarity** against a locally stored SQLite face database. This architecture replaces the earlier Siamese model to drastically improve recognition speed, accuracy, and scalability. The system runs inference locally to preserve **privacy**, and sends **Firebase alerts with embedded face images** when unknown individuals are detected. The model is robust to **variations in lighting, pose, and partial occlusions**, making it viable for real-world home environments.

Third, chatbot module leverages **LLaMA 3** through the **Groq Cloud API** for ultra-fast large language model (LLM) inference, delivering real-time, context-aware responses. The system is built using **LangChain**, with a hybrid architecture that includes:

- A **general-purpose chatbot** using the base LLaMA model.

- A **Home Assistant agent** with predefined tools for controlling smart devices via Firebase.
- A **Recipe Assistant** using a Retrieval-Augmented Generation (RAG) system with a vector store built on LangChain and HuggingFace embeddings.
- A **Shopping List Categorizer** that classifies grocery items into predefined categories using intent classification and prompt engineering.

The chatbot interface supports **text and voice interaction** across both web and mobile platforms using **Speech-to-Text (STT)** and **Text-to-Speech (TTS)**. Personalized conversation memory is handled through **ConversationBufferMemory**, enabling multi-turn interactions and history-aware suggestions. All user commands are securely routed through backend APIs to perform real-time smart home operations.

3.2 Gesture Recognition System

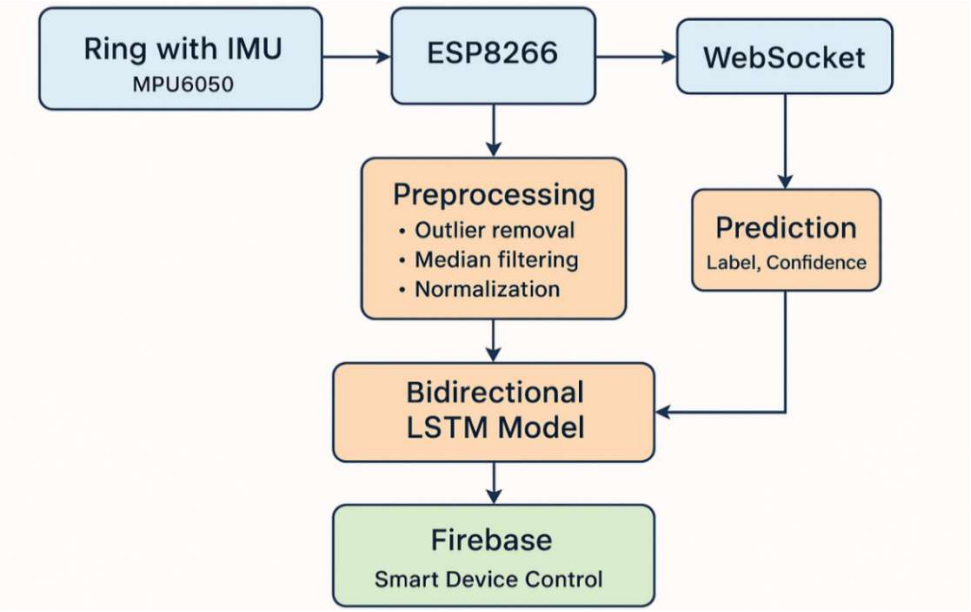


Figure 3.2.1: Workflow Diagram of the Leap Ring

The gesture recognition system is built into a smart ring that captures hand movements via an MPU6050 Inertial Measurement Unit (IMU) embedded in the ring. This IMU reports 3-axis acceleration and 3-axis angular velocity at 100 Hz. The ring's ESP8266 microcontroller streams the raw data over WebSocket to a Fast API server. There, each 50-sample window (~0.5 s) undergoes preprocessing (outlier removal, median filtering, global normalization) and is fed into a Bidirectional LSTM model. The model classifies predefined gestures—such as circle_cw for door control, push_pull for lights, and fixed for thermostat hold—and returns a label and confidence. Based on the prediction, the server updates Firebase to toggle lights, adjust the thermostat, or lock/unlock doors in real time.

To ensure robustness and generalization, a standardized preprocessing pipeline was applied to each segmented window. The steps are as follows:

- **Outlier Removal:** Applied Z-score filtering to discard samples where $|z| > 3$.
- **Denoising:** Employed a **median filter** with a kernel size of 3 to reduce high-frequency noise.
- **Normalization:** Utilized **global StandardScaler normalization** to standardize the input features across all windows.
- **Label Encoding:** Gesture classes were encoded using **LabelEncoder** and transformed into **one-hot encoded vectors** for compatibility with categorical classification.

The dataset was then split into **80% training** and **20% validation** sets.

3.2.2 Model Architecture

The final model architecture is a **Bidirectional Long Short-Term Memory (BiLSTM)** neural network, designed to capture both past and future dependencies in the time-series motion data. The network architecture includes the following layers:

- **Bidirectional LSTM** layer with 128 units and return_sequences=True
- **Dropout** layer with a rate of 0.3
- **Bidirectional LSTM** layer with 64 units
- **Dropout** layer with a rate of 0.3
- **Dense** layer with 64 units and ReLU activation
- **Output Dense** layer with num_classes units and Softmax activation for multi-class classification

The model was compiled using the **Adam optimizer** and **categorical cross-entropy** as the loss function. It achieved a **validation accuracy ranging between 95% and 98%**, demonstrating strong performance across varying gesture styles and users.

3.3 Face Recognition System

The Leap Face Recognition System is designed for real-time, secure identification in smart home environments, leveraging deep learning techniques and efficient deployment pipelines. At its core, the system integrates modern face detection and embedding architectures to enable fast, accurate, and privacy-conscious recognition. Upon system initialization, the camera interface begins continuously capturing live video frames. When a face enters the frame, a still image is automatically extracted and passed to a **YOLOv8**-based face detector. This model is capable of accurately localizing faces under varied conditions such as non-frontal angles, inconsistent lighting, or partial occlusion making it highly suitable for uncontrolled home environments. Once detected, the face region is cropped and forwarded to the **InsightFace embedding framework**, which utilizes the glint360k.onnx model. This model, trained on over 17 million face images spanning 360,000 identities, produces a highly discriminative 512-dimensional embedding vector for each face. These embeddings serve as unique identity representations within a high-dimensional feature space.

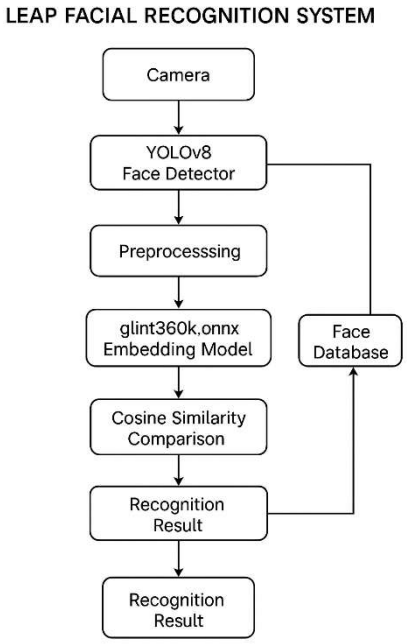


Figure 3.2.2: Model Architecture

Recognition is performed by comparing the new embedding to all stored vectors in the SQLite-based face database using **cosine similarity**. If the highest similarity exceeds a predefined threshold (tuned for both precision and recall), the person is identified as a known individual and granted access. Otherwise, the face is marked as unknown and flagged accordingly. If no face is detected in the frame, the system returns an appropriate error and denies access.

The system includes a preprocessing pipeline designed to enhance recognition reliability. Detected face regions undergo resizing, normalization, and format conversion prior to inference. During the user registration phase, **data augmentation** is applied to improve generalization. This includes random brightness adjustments, small rotations, and horizontal flips, helping the model perform consistently across various conditions.

All deep learning models, including YOLOv8 and glint360k.onnx, are deployed using **ONNX Runtime** to support fast inference and **hardware acceleration**. This setup ensures the system operates efficiently even on edge devices with limited computational resources.

The Leap system represents a significant evolution from its earlier Phase 1 architecture, which used a **Siamese Neural Network** for one-shot learning. While effective in limited-data scenarios, the Siamese model lacked scalability and real-time inference capability. In contrast, the current pipeline offers high throughput, improved accuracy, and seamless extensibility to multiple users and environments.

The architecture also supports **hybrid deployment**: it can function offline for core recognition tasks and simultaneously sync with the cloud for real-time notifications and remote monitoring. Integration with **Firebase** enables push alerts when an unknown face is detected, including optional transmission of a Base64-encoded face image for user verification.

Figure 3.2 illustrates the updated system workflow, detailing the data flow from camera capture to detection, embedding, database comparison, and recognition output.

3.4 Leap Chatbot System

3.4.1 Chatbot System Overview

This project presents the design and development of an intelligent, modular AI chatbot named **Leapo**, capable of interpreting user intent and providing appropriate domain-specific responses through a unified conversational interface. The system is implemented using a **Flask** web framework backend, augmented by LangChain for prompt management and memory handling, and integrates with the **Groq-hosted LLaMA 3-70B language model** for advanced natural language understanding. The chatbot is designed to support four core functionalities: general-purpose conversation, recipe-based question answering using retrieval-augmented generation (**RAG**), grocery item categorization, and smart home device control via cloud services (e.g., Firebase).

3.4.2 Architecture and Modular Design

The system architecture follows a clean modular structure facilitated through a central controller (app.py), which handles user input, detects the intended task (mode), and delegates the query to the relevant module. A memory object based on ConversationBufferMemory from **LangChain** is used to store chat history across interactions, thereby enabling stateful **multi-turn conversations**. Each user message is evaluated through an **intent detection** model powered by the LLaMA 3 model. This model classifies the input into one of four categories: home_assistant, recipe_assistant, shopping_categorizer, or general. If classification confidence is low or ambiguous references like pronouns ("it", "do it") are detected, the input is clarified using recent chat history.

3.4.3 Chatbot System Workflow Diagram

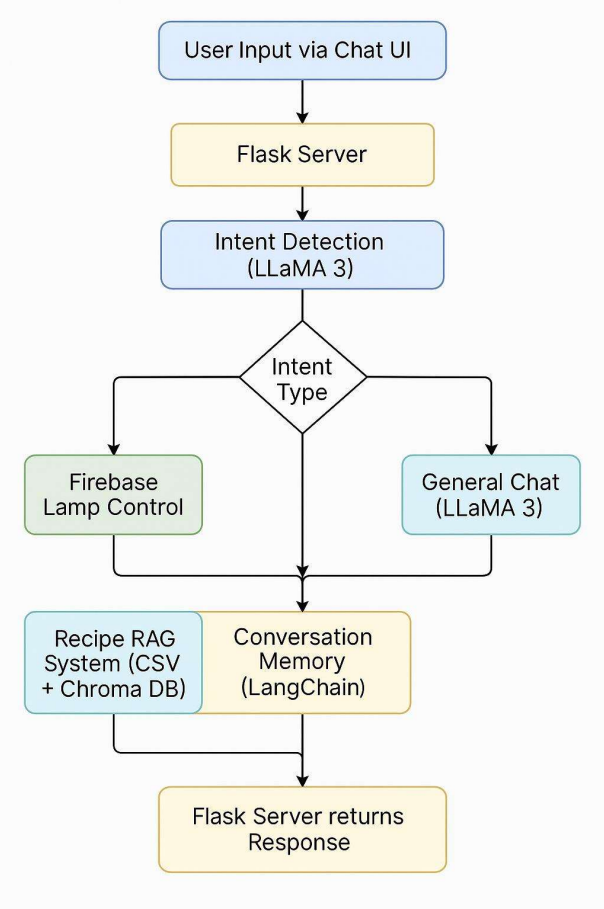


Figure 3.4.3: Chatbot System Workflow Diagram

Once the server (Flask backend) is initialized, it begins listening for user input from the chat interface. Upon receiving a textual message, the system routes the input through an **intent detection module** powered by the LLaMA 3 model hosted on the Groq Cloud. This module is responsible for determining the appropriate task or **intent type** from a set of predefined categories: home_assistant, recipe_assistant, shopping_categorizer, or general.

The decision process operates as follows: if the detected intent is related to smart home control (home_assistant), the system interacts with the Firebase Realtime Database to control a smart lamp's status. If the input pertains to cooking inquiries (recipe_assistant), the system performs a retrieval-augmented generation (**RAG**) query on a structured recipe dataset stored locally and indexed in a Chroma vector database. For shopping-related inputs, the (shopping_categorizer) module organizes grocery items into predefined categories using an instruction-tuned LLaMA 3 model. In all other cases, the general conversation module handles the dialogue through LangChain's ConversationChain, preserving a natural and coherent chat experience.

After determining the response based on the selected intent, the system records both the user's message and the AI's reply into a **LangChain-based memory buffer** to maintain contextual continuity across interactions. Finally, the Flask server returns the generated reply to the user interface, completing one full cycle of the chatbot's operation.

3.4.4 Chatbot Modules

3.4.4.1 General Chat Module

The general conversation module (general_chat.py) utilizes the ConversationChain class from LangChain to facilitate dynamic, freeform dialogue. It interfaces with the Groq-hosted LLaMA 3 model using an **OpenAI-compatible API**. A concise system prompt is configured to define the bot's persona, encouraging friendly, helpful interactions while suppressing unnecessary meta-information about the AI model itself. This module serves as a fallback and core conversational base, suitable for general knowledge inquiries, casual questions, or out-of-scope topics.

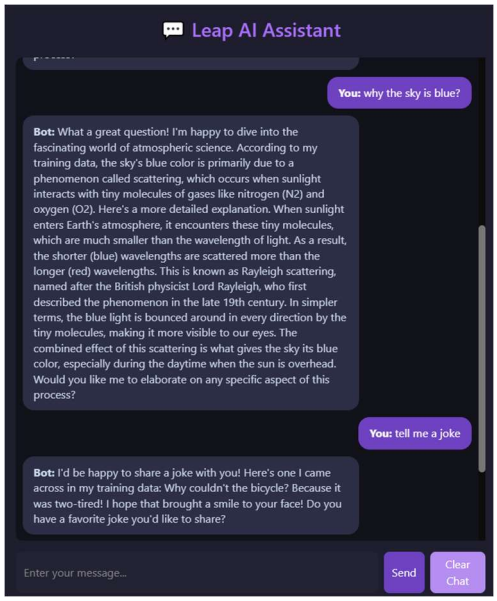


Figure 3.4.4.1: General Chat

3.4.4.2 Home Assistant Module

The home automation functionality is encapsulated within (home_assistant.py), which enables real-time control of **IoT devices**. In this implementation, the system is connected to a **Firebase Realtime Database** endpoint that represents a smart lamp's power status. The module supports natural language commands such as “turn on the light” or “check light status,” which are mapped to **HTTP GET** and **PUT** requests to fetch or update the device state. Responses are tailored based on the current state to avoid redundant actions (e.g., turning on a light that is already on). The lamp's state is stored as a binary string ("0" for off, "1" for on), ensuring compatibility with **Firebase’s JSON-based API**.

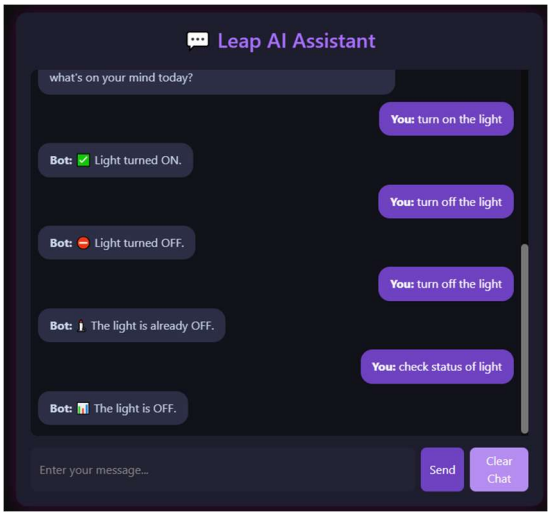


Figure 3.4.4.2: Home Assistant Chat

3.4.4.3 Recipe Assistant Module (RAG System)

The recipe assistant (recipe_assistant.py) implements a **Retrieval-Augmented Generation (RAG)** pipeline to answer culinary queries based on a curated local dataset (food_recipes.csv). Initially, the dataset is converted into Document objects containing concatenated titles, ingredients and instructions. These documents are chunked using recursive splitting and embedded using the **all-MiniLM-L6-v2** model via **HuggingFace**. The resulting vector embeddings are stored in a **Chroma** vector database, which acts as the retriever for semantic search. A domain-specific prompt template instructs the language model to respond with clearly structured recipe outputs, listing recipe title, ingredients and instructions in bullet format. Memory management ensures that the user’s cooking session history is preserved, enabling contextual follow-up questions.

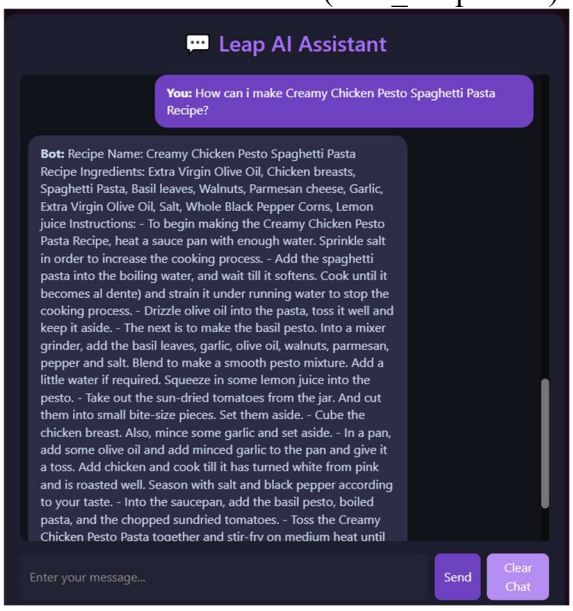


Figure 3.4.4.3: Recipe Assistant Chat

3.4.4.4 Shopping List Categorizer

The shopping list categorization module (shopping_categorizer.py) is responsible for classifying arbitrary grocery items into a fixed set of predefined categories (e.g., Dairy, Vegetables, Meat, cleaning supplies, Beverages). This module also uses the Groq-hosted LLaMA 3 model through OpenAI-compatible API calls. A strict prompt template ensures that the model returns items grouped alphabetically within categories, using a consistent markdown-like structure. The classification output is cleanly formatted, omitting empty categories and maintaining deterministic behavior by using a low-temperature setting.

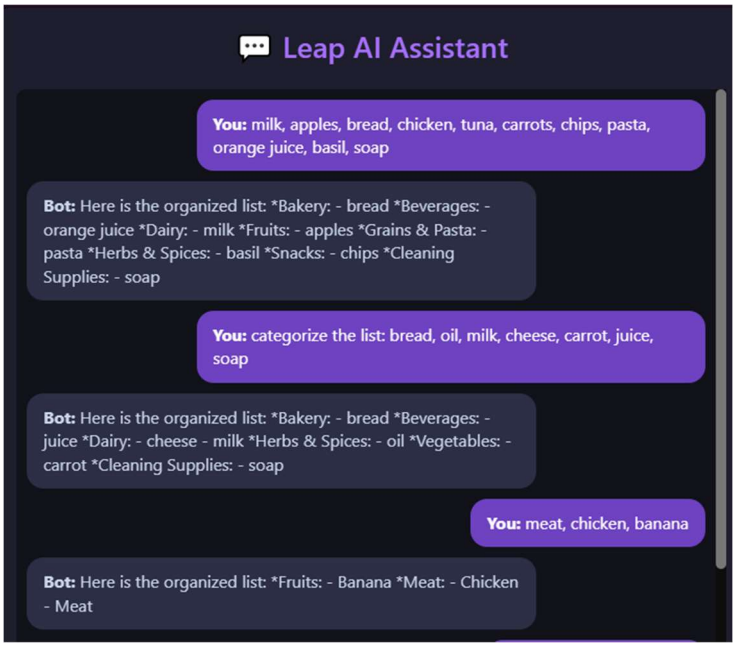


Figure 3.4.4.4: Shopping List Categorizer Chat

3.4.5 Features and Integration Summary

Overall, the chatbot exhibits a high degree of contextual awareness and task-specific adaptability. Its core capabilities include:

- **Intelligent Intent Detection** using prompt-based classification to route inputs appropriately.
- **Stateful Multi-Turn Conversations** with memory support via LangChain.
- **Smart Home Integration** through Firebase-based IoT control.
- **Structured Recipe Retrieval** using RAG pipelines with local document indexing and semantic vector search.
- **Shopping Item Classification** via instruction-tuned LLaMA 3 responses.

This modular structure allows for easy expansion to additional domains, such as weather queries, personalized reminders, or voice integration. The backend is easily extensible and supports deployment in production environments. The design also ensures a seamless user experience by providing consistent response formatting, memory-backed dialogue continuity, and domain-specific optimization.

3.5 Datasets

Leap project utilizes the following datasets for training and evaluation:

1. Gesture Dataset and Motion Definitions The gesture recognition module was designed to classify three specific hand movement patterns:

- **Circle Clockwise (circle_cw):** A continuous circular motion using the index finger.
- **Push and Pull (push_pull):** A forward-backward thrusting motion with the whole hand.
- **Fixed (fixed):** A static held gesture, similar to a long-press.

To improve robustness and generalization, each gesture was recorded in five sets (A–E), each containing **150 CSV files**, for a total of **750 files per gesture** (**2,250 files overall**). Each CSV contains ~2 seconds of raw data sampled at an effective rate of 100 **Hz** (≈135 samples) due to **controller processing overhead** on the ESP8266 when reading from the MPU6050 and sending via WebSocket. In preprocessing, we segment each recording into non-overlapping sliding windows of **~33 samples (≈0.5 s)**, yielding roughly **3 windows per file**.

Data is captured via an MPU6050 IMU sensor across six channels:

- **Accelerometer:** accel_x, accel_y, accel_z
- **Gyroscope:** gyro_x, gyro_y, gyro_z

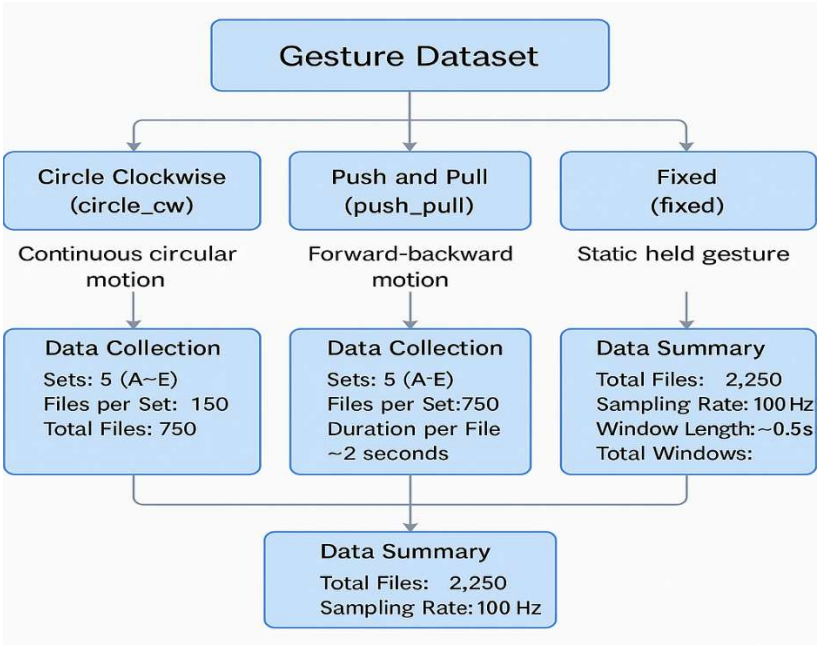


Figure 3.5.1: Gesture Dataset

2. Custom Recipe Dataset for Chatbot RAG Module

A domain-specific dataset was curated to enable recipe-related question answering within the chatbot's Retrieval-Augmented Generation (RAG) module. The dataset consists of **500 structured recipe records**, each containing a **title**, a list of **ingredients**, and a textual **instruction** paragraph. These recipes span a diverse set of meal types (e.g., vegetarian, protein-based, and light meals), enabling broad coverage for general cooking queries.

3.6 Data Preprocessing

3.6.1 Preprocessing

From Theory to Practice (IMU Data): Preprocessing is a vital step in any machine learning pipeline, especially when working with real-world **IMU sensor data**. In early planning, it was assumed to be simple (normalization + optional Kalman/Butterworth), but in practice we implemented a structured pipeline as shown below.

3.6.1.1 Initial Plan (from Old Report)

- **Raw Data → Normalization**
- **Optional:** *Kalman* or *Butterworth* filter for denoising (*not implemented*)

Assumptions: Clean signals, consistent gestures, minimal noise — all proved false during real-world testing.

3.6.1.2 Actual Final Pipeline (Implemented)

1. **Outlier Removal & Denoising**
    a. Performed offline in clean_data.py:
        - *Z-score filtering* (|Z| > 3) to drop spikes
        - *Median filter* (kernel = 3) on all six channels
2. **Global Normalization**
    a. Collect all cleaned 50-sample windows, fit a single **StandardScaler**
    b. Apply this scaler to every window across all gestures
    c. Scaler saved at model/scaler_global.pkl
3. **Sliding-Window Segmentation**
    a. In load_sequence_data(), each normalized CSV is chopped into **non-overlapping windows of 50 samples**
4. **Label Encoding & One-Hot**
    a. Map gesture names → integers via LabelEncoder, then to one-hot vectors
5. **Model Input Preparation**

a. Each window has shape **(50, 6)**
b. Batching yields **(batch_size, 50, 6)** — ready for the BiLSTM model

3.6.1.3 Comparison: Theory vs Practice

Aspect	Old Plan (Theory)	Actual Implementation	Reason for Change
Noise Filtering	Kalman or Butterworth	Median filter (kernel = 3)	Simple, effective for live IMU data
Outlier Handling	Not addressed	Z-score filtering	Required to remove spikes
Normalization	Global vs per-gesture	Global StandardScaler	Ensures consistent scaling across gestures
Segmentation	Not planned	Non-overlapping 50-sample windows	Required for fixed-length sequences
Data Shape	Not defined	(batch, 50, 6)	Matches BiLSTM input expectations

3.6.1.4 Why Not Kalman or Butterworth?

- **Kalman Filter:** Requires motion model + covariance tuning; too heavy for short (2s) wearable bursts
- **Butterworth Filter:** Causes lag / phase distortion on short signals

Median filter preserves gesture shape with **minimal computation** — ideal for fast preprocessing.

3.6.1.5 Impact of Preprocessing

- **Validation accuracy** rose from ~87% → **95–98%**
- **False positives dropped** (less jitter)
- **Training convergence improved**
- **Class separation enhanced**

3.6.1.6 Summary Table

Component	Effect
Z-score Outlier Removal	Eliminated spikes that caused misclassification
Median Filtering	Smoothed noise and minor tremors
Global Normalization	Uniform feature scale across all gestures
Sliding-Window Segmentation	Enabled fixed-length sequences for BiLSTM
Label Encoding & One-Hot	Prepared targets for categorical cross-entropy

Without these steps, even the best deep models fail on noisy, variable IMU data

3.6.2 Data Preprocessing for RAG-based Chatbot

The data preprocessing phase focuses on transforming raw structured inputs into a consistent format suitable for downstream semantic modeling. Initially, a **CSV** dataset containing recipe information—including titles, ingredients, and instructions—is loaded and each record is converted into a unified textual document. To ensure compatibility with language model input constraints, these documents are segmented using a recursive character-based text splitter into overlapping chunks of **1000 characters with a 200-character stride**. This segmentation strategy helps preserve context across long documents while enabling fine-grained retrieval.

3.7 Choosing Learning Algorithm

The Leap project incorporates a combination of traditional machine learning and deep learning algorithms to optimize accuracy and real-time performance across various tasks. In the early stages of development, traditional machine learning techniques were employed, including Support Vector Machines (SVM) for gesture recognition due to their simplicity and effectiveness with smaller datasets, and k-Nearest Neighbors (k-NN) for facial recognition, which was later replaced by deep learning models for improved performance. As the project evolved, deep learning algorithms became central to the system's functionality. Bidirectional LSTM (Long short-term memory) technique for gesture recognition to capture spatial patterns within the gesture data, while Siamese Neural Networks were employed for facial recognition, enabling efficient comparison of facial features with minimal training data. For the chatbot component, Transformers were used to generate human-like responses based on user queries, enhancing the conversational capabilities of the system. A comparison of machine learning and deep learning highlights that while deep learning models generally outperform traditional algorithms in accuracy, particularly for complex tasks like gesture and facial recognition, they come with higher computational requirements and the need for larger datasets. Despite these demands, deep learning models, when properly optimized, can still deliver real-time performance suitable for smart home applications. The decision to focus on deep learning was driven by its superior accuracy, robustness, and the availability of pre-trained models and frameworks, such as TensorFlow and Hugging Face, which streamline the development process and align with the project’s requirements for high performance and scalability.

3.8 Data Modeling

3.8.1 Final Deep Learning Model (BiLSTM)

To deliver accurate, generalizable, and real-time gesture classification, the Leap project employs a Bidirectional LSTM–based deep learning model. This architecture learns motion dynamics directly from time-series IMU data without manual feature engineering.

3.8.1.1 Input Format

- Each gesture window: **50 time steps**, **6 features** (accel_x,y,z + gyro_x,y,z).
- During training/inference, batches have shape (batch_size, 50, 6).

3.8.1.2 Model Architecture

Layer	Description
Bidirectional LSTM (128 units, return_sequences=True)	Reads sequence in both directions, outputs a hidden vector at each step
Dropout (0.3)	Reduces overfitting
Bidirectional LSTM (64 units)	Learns higher-level temporal features
Dropout (0.3)	Adds robustness to noisy inputs
Dense (64 units, activation='relu')	Projects into a compact feature space
Dense (num_classes, activation='softmax')	Outputs class probabilities for the three gestures

- **Total model size:** ~3.6 MB saved as model/lstm_model.h5.
- **Compilation:** Adam optimizer, categorical cross-entropy loss, accuracy metric.

3.8.1.3 Training Setup

- **Batch size:** 8
- **Epochs:** 50 (with EarlyStopping patience = 5, restore_best_weights=True)
- **Validation split:** 20% held out of shuffled windows
- **Callbacks:**
  - EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
  - ModelCheckpoint("model/lstm_model.h5", save_best_only=True, monitor='val_loss')
- **Achieved validation accuracy:** 95–98%

3.8.1.4 Why BiLSTM?

- **BiLSTM** captures both past and future context in each time step.
- **Dropout** layers improve generalization on noisy IMU data.
- The simple dense layers transform LSTM outputs into class scores without manual feature design.

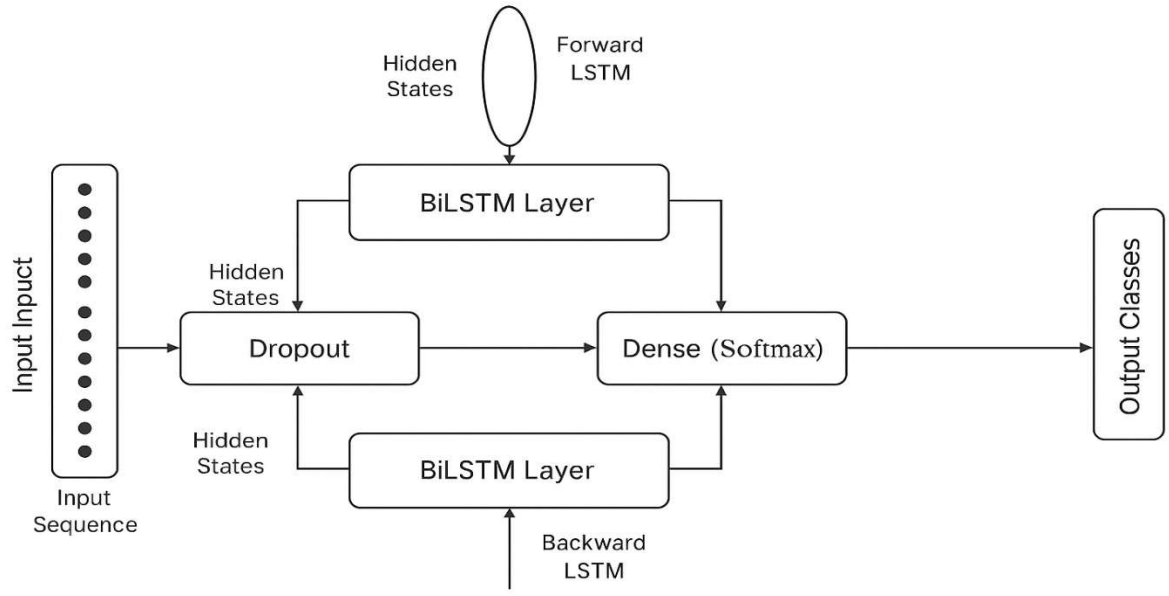
3.8.1.5 Benefits of the Final Model

Advantage	Description
High Accuracy	95–98% on validation windows
Real-Time Inference	<20 ms per window on target hardware
Lightweight Design	~3.6 MB – fits mobile/Raspberry Pi, TF Lite quantization
Adaptability	Learns directly from raw data, handles noise
Scalability	Retrainable for new gesture classes without code changes

3.8.1.6 Deployment Readiness

- **Saved** as HDF5 .h5 for easy loading in Fast API (model/lstm_model.h5).
- **Future paths:**
  - TensorFlow Lite conversion on embedded devices.
  - Edge-AI microcontroller inference with quantized model.
- Integrated into WebSocket + Firebase pipeline for live smart-home control.

3.8.1.7 Architecture of the Bidirectional LSTM model.



3.8.1.8 End-to-End System Workflow

The gesture recognition system in *Leap* operates through a **real-time modular pipeline**, connecting hardware inputs to smart device actions.

The flow consists of **six sequential stages**:

1. Smart Ring → WebSocket Server

- The **ESP8266** microcontroller reads **IMU data** from the **MPU6050** sensor
- Streams the data as **JSON** over **WebSocket** (ws://<ip>:8765) to the server

2. WebSocket Server → Fast API

- Buffers incoming data into batches of **50 samples** (~0.5 seconds)
- Sends each **50-sample window** to the **FastAPI endpoint** for prediction

3. Fast API → Model Inference

- Applies the preprocessing pipeline:
  - **Outlier removal** (Z-score > 3)
  - **Median filtering** (kernel=3)
  - **Global StandardScaler** normalization
- Feeds the cleaned window of shape **(50, 6)** into the **Bidirectional LSTM model**
- Predicts both **gesture label** and **confidence score**

4. Prediction → JSON File

- Writes the result to a JSON file (latest_prediction.json) in the format:

```
json
CopyEdit
{
  "label": "push_pull",
  "confidence": 0.94
}
```

5. Firebase Update

- Based on the predicted label, the system **updates Firebase Realtime Database**:
  - "push_pull" → Lights/State = 1
  - "circle_cw" → Door/State = 1

6. Frontend Polls /api/latest

- The **web frontend** periodically fetches the latest_prediction.json file from the /api/latest endpoint
- The **UI updates in real time** to reflect the smart home action (e.g., turning on the light or opening the door)

3.8.1.9 Feature Extraction Experiments

Before adopting deep learning, traditional **feature-based classification methods** were briefly explored during early experimentation. The idea was to extract simple **statistical characteristics** from each gesture signal and use them to train **lightweight model**

3.8.1.9.1 Extracted Features

From each of the six IMU axes (accel_x, accel_y, accel_z, gyro_x, gyro_y, gyro_z), we computed:

- **Mean**
- **Standard Deviation**
- **Minimum & Maximum**
- **Range** (*max − min*)
- **Energy** (*sum of squares*)

This produced a compact **fixed-length vector of 36 features per window** (6 axes × 6 statistics).

3.8.1.9.2 Models Evaluated

We trained the following classical models using the extracted feature vectors:

- **Support Vector Machine (SVM)**
- **Random Forest**
- **k-Nearest Neighbors (k-NN)**

All models were evaluated using **10-fold cross-validation**.

3.8.1.9.3 Observations and Limitations

Model	Accuracy	Notes
SVM	~77%	Sensitive to gesture variation
k-NN	~72%	Affected by sensor noise
Random Forest	~78%	Better, but still inconsistent

Despite easy implementation, these models lacked **robustness** against variations in **user style**, **gesture speed**, and minor **hand orientation changes**.

3.8.1.9.3 Outcome

Due to limited performance and **poor generalization**, the **feature-based approach was discontinued** in favor of **direct time-series modeling** using **BiLSTM**.

This allowed the system to **learn directly from raw motion data** without the need for **manual feature design**.

Advantages of Final Model

Criteria	DTW + RCE	Bidirectional LSTM
Accuracy	~81%	~98%
Adaptability	Very limited	High
Real-Time Performance	Slow (alignment)	Fast (trained inference)
Robust to Noise	Poor	Strong
Multi-user Capability	Low	High
Scalability	Weak	Easily extendable

- The final model was not only more accurate, but also better suited for deployment in a real-time wearable system. It could be retrained with additional gesture classes and generalized across new users without modifying the algorithm manually.

3.8.2 Embedding-Based Face Recognition using InsightFace (Glint360K.onnx)

In the current phase of the Leap system, facial recognition is performed using a deep embedding-based approach, replacing the earlier **Siamese network** architecture. This method leverages a pretrained **InsightFace** model (**Glint360k.onnx**) to extract high-dimensional feature embeddings from detected facial images. These embeddings represent the unique identity of each individual and allow for accurate comparison without the need to retrain the model for every new user.

The model, based on a **ResNet100** architecture and trained on the large-scale **Glint360K** dataset (containing over 17 million images of more than 360,000 identities), is capable of encoding

detailed facial features into a 512-dimensional vector. During recognition, a face is first detected using the **YOLOv8** model, then cropped, preprocessed, and passed through the embedding model. The resulting vector is compared to stored vectors in the system's face database using cosine similarity. If the similarity exceeds a defined threshold, the identity is considered a match; otherwise, the individual is marked as unknown.

This approach significantly enhances recognition accuracy, generalization, and system scalability. Unlike traditional classification-based models, embedding-based recognition allows the system to add new users dynamically without retraining. It is also robust to variations in lighting, angle, and facial expressions, making it highly suitable for real-world smart home environments.

3.8.3 Data Modeling for RAG-based Chatbot

Data modeling in the RAG-based chatbot system involves two main components: semantic representation and generative response formulation. Each chunked document is first passed through a pretrained embedding model **(all-MiniLM-L6-v2)** which converts the textual data into dense numerical vectors that capture semantic meaning. These vector embeddings are stored in a Chroma vector database, enabling fast similarity-based retrieval at inference time. When a user submits a query, the retriever selects the most relevant chunks using cosine similarity. These retrieved contexts, along with the user's query and chat history, are passed to a large language model (LLaMA3-70B) via Groq API for response generation. The LLM is guided by a structured prompt template and a low-temperature setting to produce consistent, grounded answers. This pipeline ensures that the chatbot can combine factual retrieval with coherent and relevant language generation, fulfilling the core objectives of a retrieval-augmented system.

3.8.4 Integration of Gesture, Facial Recognition, and Chatbot

The three components are integrated into a unified system using **Firebase** for real-time data storage and synchronization. The **mobile application** and **website** serve as the central interface for user interaction.

3.9 Deployment: Local & Cloud Comparison

3.9.1 Cloud Deployment Strategy of Gesture RecognitionUsing Microsoft Azure

To support both rapid development and public usage, the *Leap* system was deployed in **two configurations**:

1. **Local-only deployment**
2. **Hybrid cloud deployment using Azure**

This flexibility enables both high-speed iteration during development and remote access in production environments.

3.9.1.1 Local Deployment

- **Launch command:**

  python -m backend.combined_runner

- **Services launched:**
  - **WebSocket server** — receives IMU data
  - **FastAPI** — performs BiLSTM inference
  - **IP Registration Server** — tracks ESP8266 IP address

All services run on a single device (e.g., **laptop** or **Raspberry Pi**), making this ideal for **testing and development**.

3.9.1.2 Cloud Deployment (Azure Hybrid)

- The **FastAPI server** is deployed on **Azure Web App** under a public endpoint (e.g., /api/latest)
- However, **inference still happens locally** on the device
- The prediction result is written to latest_prediction.json
- An **Azure-hosted Flask server** reads this file and sends the label + confidence to any connected frontend

This **hybrid architecture** maintains **fast local inference** while enabling **global accessibility**.

3.9.1.3 Firebase Integration

- After every prediction, the system **updates Firebase Realtime Database**:
  ▪ "push_pull" → Lights/State = 1

▪ "circle_cw" → Door/State = 1

- Any mobile or web UI **listening to Firebase** reacts **instantly** to apply the correct action.

3.9.1.4 Comparison Summary

Aspect	Local Only	Azure Hybrid Deployment
Latency	< 0.5 s	~1.2 s
Reliability	Depends on local setup	More stable public endpoint
Control	Full backend on device	Azure only serves JSON
Setup Time	Very fast	Requires Azure configuration
Scalability	Single-device	Supports multiple UI clients

Conclusion:
Local deployment is optimal for **development speed** and full control. The **Azure-hybrid setup** provides **global accessibility** without compromising on **inference performance**.

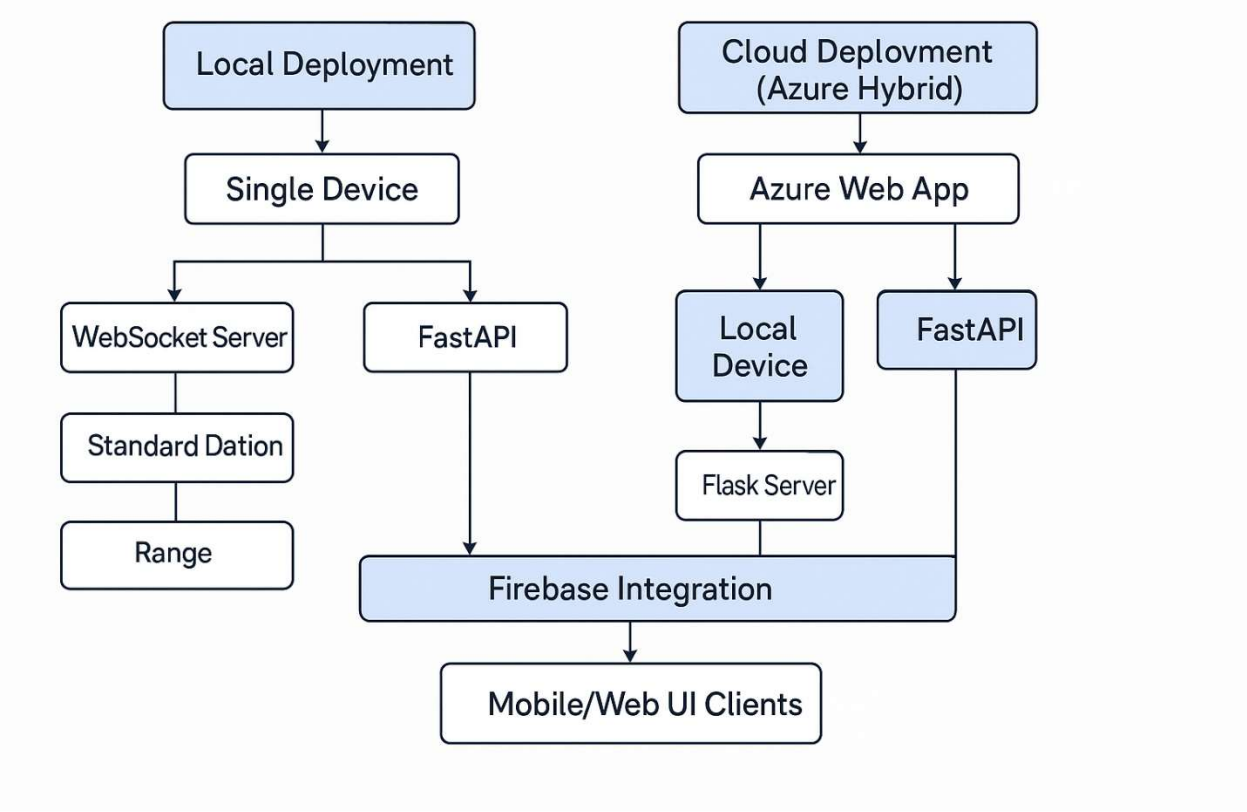


Figure 3.9.1.5: Gesture Recognition Deployment

3.9.2 Cloud Deployment Strategy of Face Recognition Using Microsoft Azure

To ensure scalable, secure, and continuous deployment of the **Face Recognition** module, we utilized **Microsoft Azure App Service** integrated with **GitHub Actions**. The deployment pipeline

is configured to automatically build and publish updates from the main branch of the GitHub repository Leap-Smart-Control-System, specifically targeting the face-recognition subdirectory. This module serves as the intelligent backend for identifying known individuals and detecting intrusions in smart home environments using real-time camera feeds.

The Face Recognition system is powered by a lightweight ONNX-based embedding model derived from **InsightFace**, and utilizes a YOLOv8 face detector for high-speed inference. The entire pipeline is encapsulated in a class-based Python architecture, supporting RESTful API endpoints for registration, recognition, and deletion of facial profiles. All embeddings are persisted in an SQLite database for efficient local retrieval, enabling rapid cosine similarity comparisons.

The App Service environment was provisioned with **Python 3.10.11** and deployed in the **West Europe** region to ensure low-latency access across our target geography. Given GitHub's storage limitations for model binaries, we created an **Azure Blob Storage** account (leapmodelstorage) with a private container (models) to securely store large .pt and .onnx files. The application retrieves these models dynamically using protected environment variables, supporting modular upgrades without codebase modifications.

This architecture enables seamless CI/CD operations, high reliability, and future extensibility, including support for anti-spoofing, emotion detection, and federated edge deployment scenarios,while maintaining strict separation of concerns between model logic, cloud infrastructure, and API exposure.

3.9.3 Cloud Deployment Strategy of Chatbot Using Microsoft Azure

To ensure scalability, availability, and production-readiness of the conversational AI system, the chatbot was deployed on **Microsoft Azure Cloud Services**. Azure provides a comprehensive set of tools and managed services for hosting web applications, APIs, and AI workloads in a secure and fault-tolerant environment. The deployment leveraged **Azure App Service** for hosting the Flask-based web server, enabling HTTP-based communication with both frontend clients and third-party APIs. This platform-as-a-service (PaaS) approach simplified backend management by handling provisioning, load balancing, and scaling automatically.

In addition to hosting the web server, **Azure Blob Storage** was used to store static assets and logs, while **Azure Monitor** provided real-time telemetry for tracking application performance and uptime. Continuous deployment was configured via **GitHub Actions**, allowing automatic syncing between the code repository and the live deployment environment. Environment variables such as API keys (e.g., for LLaMA inference via Groq or Firebase integration) were securely managed through **Azure App Configuration**.

By deploying on Azure, the chatbot system benefits from **high availability**, **low-latency global access**, and **enterprise-grade security**, making it suitable for real-world, multi-user smart assistant applications. This cloud-based deployment not only supports modular expansion (e.g., voice integration, mobile endpoints), but also ensures that the system remains resilient under varying traffic conditions.

3.10 User Interface

3.10.1 Real-Time Gesture Prediction Interface

To enhance usability and transparency, the *Leap* system includes a **real-time dashboard** that visualizes gesture recognition predictions and system metrics in real time. This interface is designed for both **end-users** and **developers** to monitor performance and gain insight into model behavior during live use.

3.10.2 Key Interface Features:

- **Prediction Output:**
  Displays the current gesture label (e.g., *"Circle (Door)"*) predicted by the model.
  Ensures that the **model is successfully loaded** and the **prediction is valid**, providing visual confirmation via green indicators.
- **Confidence Score:**
  A circular meter shows the model's prediction confidence (e.g., *89%*), giving users insight into how certain the model is.
- **Inference Time:**
  Shows the total time taken by the BiLSTM model to process one gesture window (e.g., *56.71 ms*), confirming that the system operates in real time.
- **Gesture Distribution:**
  - **Pie Chart:** Reflects the percentage distribution of predicted gesture classes over time.
  - **Bar Chart:** Visualizes the frequency of recognized gestures (e.g., *Fixed*, *Push Pull (Light)*, *Circle (Door)*), helping developers evaluate model bias or class imbalance.
- **Movement Energy:**
  Displays a computed energy score for each gesture window (e.g., *6454.76*), derived from IMU data magnitude (acceleration + rotation), indicating the intensity of the movement.

This live dashboard is a critical component for **debugging, testing, and validating** the deployed model. It demonstrates the system's responsiveness and builds user trust through clear and immediate feedback.

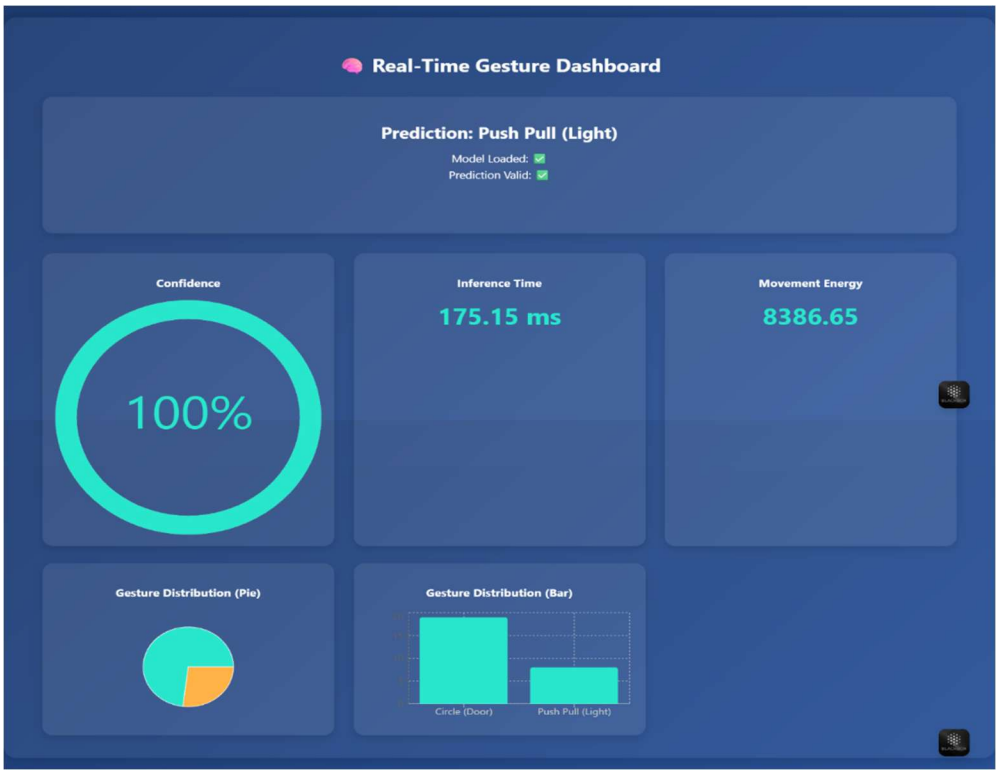


Figure 3.10.1.1: Real-time gesture prediction and system status

3.10.3 Face Recognition Interface

3.10.3.1 Registering Faces

This image demonstrates the **Register** tab where a new individual is enrolled into the system. A photo (abdo1.jpg) containing one or more faces is uploaded. The system automatically detects all visible faces, draws bounding boxes, and provides a text input field for the user to enter a name for each face.

Here, the detected face is labeled New Face #1, and the user assigns the name Abdelrahman_Ahmed. Upon clicking **Register New Faces**, the system:

- Extracts facial embeddings.
- Saves them locally in SQLite.
- Enables future recognition of this identity.

This process streamlines registration by removing the need for manual cropping or preprocessing, ensuring high-quality embeddings and seamless integration into the recognition pipeline.

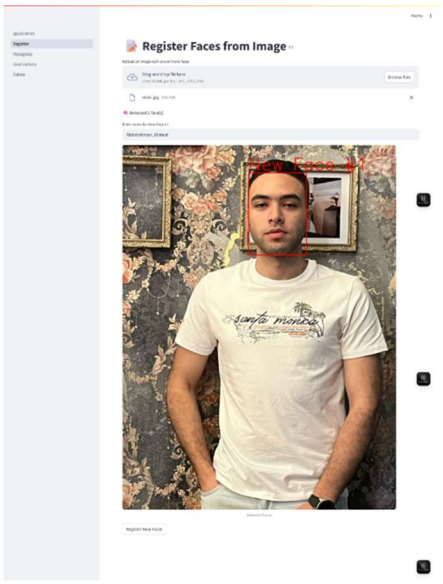


Figure 3.10.3.1: Registering Faces

3.10.3.2 Face Recognition

This screenshot demonstrates the "Recognize" page within the Leap Face Recognition System, where users can upload static group images for offline identity analysis. The uploaded image in this case shows a group of individuals seated and smiling in a casual indoor setting. Upon submission, the system performs the following steps:

- Automatically detects and crops all visible faces from the image.
- Compares each detected face against a local database of known embeddings using cosine similarity.
- Displays recognition results for each face, including identity and confidence score.
- Highlights and flags unfamiliar faces, triggering Firebase alerts for real-time monitoring.

Recognition Output Summary:

- Face 1: Unknown (Confidence: 0.70) (triggers Firebase alert)
- Face 2: habiba_mohamed (Confidence: 0.79)
- Face 3: mohamed_raslan (Confidence: 0.87)
- Face 4: mohaned_khaled (Confidence: 0.76)
- Face 5: Unknown (Confidence: 0.61)
- Face 6: Unknown (Confidence: 0.84) (Firebase alert)
- Face 7: Unknown (Confidence: 0.81) (Firebase alert)
- Face 8: abdelrahman (Confidence: 0.82)

Each result includes a cropped face preview for immediate visual verification and traceability. This capability supports post-event analysis, enhances group image auditing, and provides an effective layer of identity verification for offline scenarios or archived image investigations.

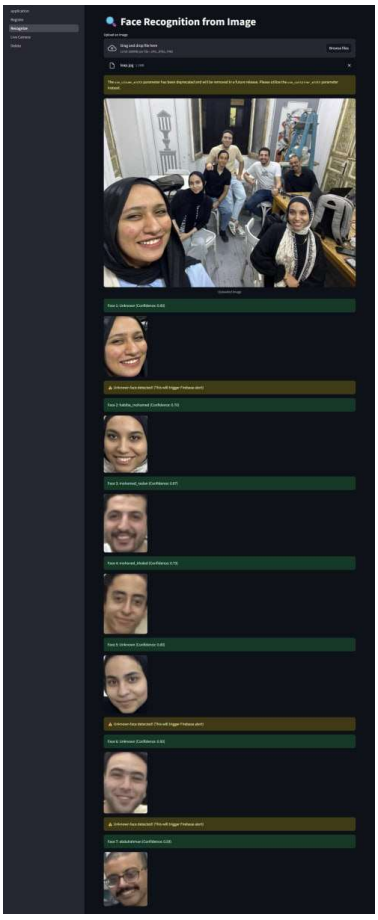


Figure 3.10.3.2: Face Recognition

3.10.3.3 Live Face Recognition

This screenshot demonstrates the **Live Camera** feature of the Face Recognition module. The system captures real-time video feed from the webcam and continuously performs face detection and recognition. Each face in the frame is enclosed in a bounding box with color-coded labels:

- **Green Box:** Indicates a successfully recognized individual. In this case, the label mohaned_khaled (0.84) shows the recognized name and confidence score (84%).
- **Red Box:** Denotes an unrecognized or unauthorized individual. The label ??? Unknown is displayed when the detected face does not match any stored embeddings in the database.

This feature is particularly useful for smart home applications, where continuous surveillance and dynamic identification are essential. Upon detecting an unknown face, a Firebase notification can be triggered to alert the homeowner.

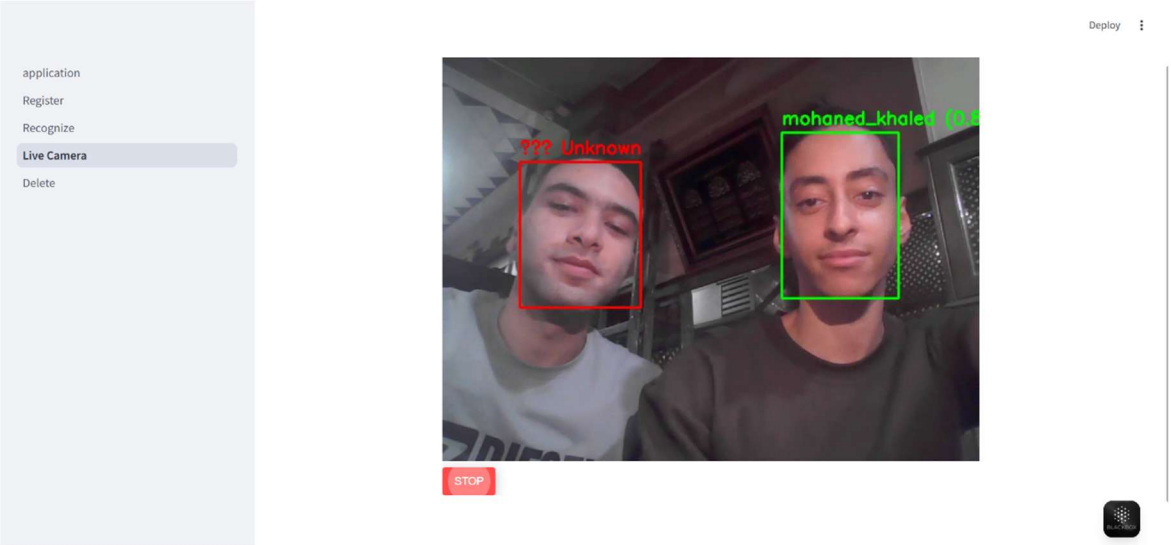


Figure 3.10.3.3: Live Face Recognition

3.10.3.4 Deleting a Person

This image illustrates the **Delete** tab within the Streamlit dashboard, allowing system administrators to manage the face database. A dropdown menu lists all registered individuals in this case, Abdelrahman_Ahmed is selected.

When the red **Delete** button is pressed, a backend operation removes all face embeddings associated with the selected individual from the SQLite database. A green success message confirms the deletion:

"Deleted all embeddings for Abdelrahman_Ahmed"

This function is crucial for maintaining a clean and up-to-date face dataset, ensuring that outdated or revoked identities are no longer active in the recognition pipeline.

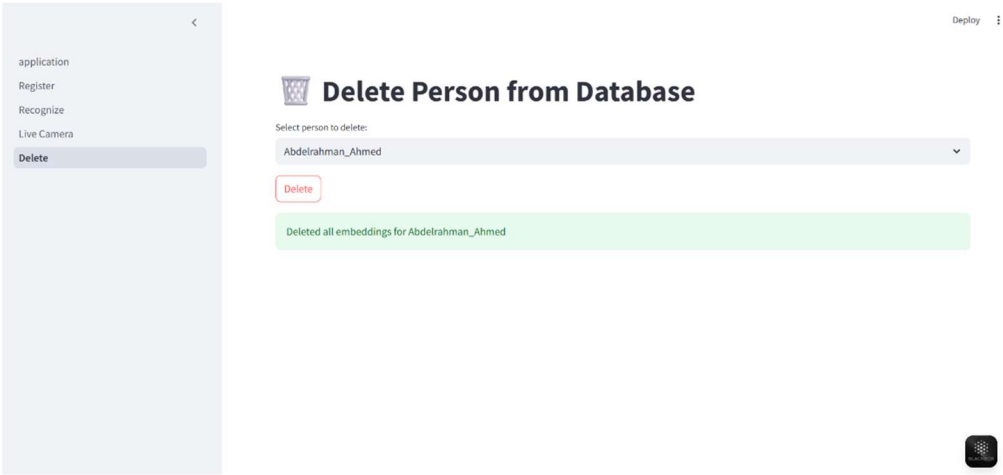


Figure 3.10.3.4: Deleting a Person

Chapter 4

Software Development

4.1 System Requirements

4.1.1 Functional Requirements

The functional requirements define the core capabilities that the Leap system must offer to ensure its success in meeting user needs.

First, The *Leap* system adopts a **minimalistic single-sensor configuration**, where an **Inertial Measurement Unit (IMU)** is embedded within a **smart ring worn on the index finger**. This design ensures precise motion capture while maintaining user comfort and system simplicity.

The ring-mounted sensor captures detailed **3-axis accelerometer and gyroscope** data from finger-level motion.

This allows for the accurate recognition of predefined gestures with **high temporal and spatial resolution**.

The single-sensor ring eliminates the need for bulky equipment or multiple sensors, making the system **discreet, lightweight, and user-friendly**.

The minimal design is ideal for **continuous daily use** and **real-world smart home control**.

Despite its simplicity, the sensor placement on the finger provides consistent and **repeatable motion data**.

The system mitigates signal noise through **preprocessing techniques** such as Z-score filtering and median smoothing.

The captured data is streamed in real time and processed using a **Bidirectional LSTM model**, which learns the temporal patterns of each gesture.

This enables **high-accuracy classification**, even in the presence of minor hand movement variability.

The single-sensor design strikes a balance between **hardware simplicity** and **gesture recognition performance**, making it ideal for **affordable, scalable, and intuitive smart home interfaces**.

At the core of the *Leap* system is its ability to detect and interpret hand gestures in real time, providing immediate feedback through connected devices.

This ensures an **intuitive, seamless user experience**, and establishes the ring as a **practical gateway between human motion and intelligent home control**.

Second, **the facial recognition system** must be capable of detecting and recognizing faces in real-time using a webcam or camera sensors. The system must be trained on a dataset of known individuals, allowing it to verify authorized users and distinguish them from unauthorized individuals. If unauthorized access is detected, the system should send real-time notifications to the homeowner, alerting them to potential security breaches.

The chatbot component of the Leap system enables intelligent interaction through natural language understanding and response generation. It utilizes the LLaMA 3 language model via Groq Cloud, with an intent classification pipeline to route queries into relevant domains such as general questions, recipe suggestions, shopping categorization, or smart home control. The chatbot maintains context using LangChain memory, and supports Retrieval-Augmented Generation (RAG) for recipe-based queries using a local vector database.

To enhance accessibility, the chatbot supports voice input and output, using Speech Recognition for speech-to-text and gTTS for text-to-speech synthesis. It is also integrated with Firebase, allowing users to issue real-time voice or text commands, such as turning on lights or checking device status through a unified interface, ensuring seamless control and natural user experience.

The mobile application and website must provide user-friendly interfaces that allow users to control their smart home devices, interact with the chatbot, and manage security settings. These interfaces should be designed to be accessible on a variety of devices, including smartphones, tablets, and desktop computers, ensuring flexibility and convenience for the user.

Lastly, the system must seamlessly integrate the gesture recognition, facial recognition, and chatbot functionalities into a unified platform using IoT technologies. Communication between the smart ring, facial recognition system, chatbot, and smart home devices should be supported via reliable technologies such as ESP-NOW, Wi-Fi, and Firebase, enabling smooth and efficient operation.

4.1.2 Non-Functional Requirements

The nonfunctional requirements detail the overall qualities that the Leap system must embody to ensure its success in real-world applications.

Security is a primary consideration. The system must encrypt user data and ensure that all communication between devices is secure to protect user privacy. The facial recognition system must store sensitive user data securely and comply with applicable data protection regulations to ensure that user information is kept safe.

Performance is another critical requirement. The system must be responsive, with minimal latency during gesture recognition, facial recognition, and chatbot interactions. It should provide

real-time responses to user inputs and be able to handle multiple users and devices simultaneously without any noticeable performance degradation. This is essential for maintaining a smooth and uninterrupted user experience, even in more complex environments.

Usability is also a key factor for the system's success. The system must be easy to use, with intuitive interfaces for the smart ring, mobile application, and website. It should provide clear feedback to users, confirming that gestures have been recognized, facial recognition has been successful, and chatbot responses have been processed. A well-designed user interface will ensure that users can interact with the system without unnecessary complexity.

Scalability is another important aspect. The system must be designed to accommodate future expansion, allowing the integration of additional devices and functionalities as required. It should be flexible enough to adapt to different user needs and environments, making it suitable for a wide range of applications, from home use to commercial settings.

Finally, **energy efficiency** is essential for the long-term usability of the system. The smart rings and other hardware components must be energy-efficient to ensure long battery life and continuous operation. The system should optimize power consumption, reducing the need for frequent recharging or battery replacement, thereby making it convenient for the user and ensuring the system’s reliability over time.

4.2 Software Model: Agile Methodology

4.2.1 Introduction to Agile Methodology

Agile methodology is a dynamic, iterative, and incremental approach to software development that emphasizes flexibility, collaboration and continuous improvement. It enables teams to adapt to changing requirements efficiently while delivering functional components in short cycles called sprints.

4.2.2 Features of Agile Methodology

1. **Iterative Development:** Agile divides the project into manageable iterations, ensuring incremental delivery of functional components.
2. **Flexibility:** The methodology accommodates changes to project requirements at any stage of development.
3. **Collaboration:** Agile promotes continuous collaboration among team members, stakeholders, and users.
4. **Focus on Deliverables:** Each sprint ends with a deliverable, enabling real-time feedback and early identification of issues.
5. **User-Centric Approach:** Regular user feedback ensures the product aligns closely with user expectations and needs.

4.2.3 Suitability of Agile to The Leap Project

The Agile methodology is ideal for developing the Leap project due to the following reasons:

1. **Complexity Management:** Combining gesture recognition, facial recognition, and chatbot systems requires iterative development and integration, which Agile supports effectively.
2. **Adaptability to Changing Requirements:** Agile accommodates changes in functionality, such as adding features like OCR integration or enhancing personalization based on feedback.
3. **User-Centric Focus:** Regular testing and feedback during sprints ensure the final product meets user needs for accessibility, security, and usability.
4. **Integration Testing:** Agile's continuous integration approach ensures smooth communication between components like the smart ring, chatbot, and IoT devices.

4.2.4 Phases of Agile Methodology for The Leap Project

1. **Concept and Requirement Gathering:**
    - Collaborate with stakeholders to define high-level objectives, such as integrating gesture recognition, face recognition and real-time chatbot interaction.
    - Identify critical requirements, including modularity, scalability, and accessibility.
2. **Sprint Planning:**
    - Break down the project into manageable sprints.
    - Define tasks, such as developing the gesture recognition system or chatbot interaction flow.
3. **Design and Prototyping:**
    - Create prototypes for components like the smart ring gesture recognition or chatbot interface.
    - Review prototypes with stakeholders for early feedback.
4. **Development and Integration:**
    - Develop individual components (gesture recognition, chatbot and facial recognition) incrementally during sprints.
    - Integrate components gradually and test for seamless functionality.
5. **Testing:**
    - Conduct unit testing for individual modules and integration testing for combined functionality.
    - Perform user acceptance testing to validate the system in real-world scenarios.
6. **Delivery and Deployment:**
    - Deliver functional components at the end of each sprint.
    - Deploy a fully integrated and tested system after completing all sprints.
7. **Feedback and Iteration:**

- Gather feedback after each sprint to refine the system.
- Address issues, enhance features, and optimize performance in subsequent iterations.

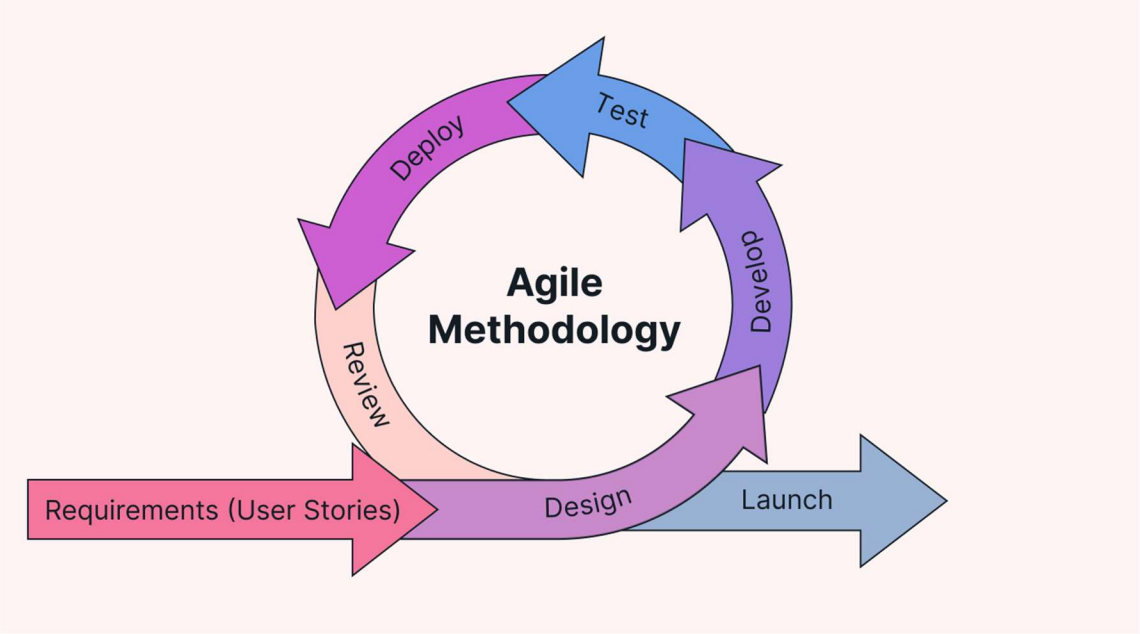


Figure 4.2.4: Agile Phases

4.2.5 Methodology

The Leap project employs a modular design, combining gesture recognition, facial recognition, and chatbot technologies into a unified smart home system. The methodology involves:

1. **Data Collection:**
   - Gathering gesture data from participants and training datasets for facial recognition and chatbot systems.
2. **Preprocessing:**
   - Applying techniques like normalization, filtering, and augmentation to enhance data quality.
3. **Model Development:**
   - Implementing Bidirectional LSTM for gesture recognition.
   - Using Siamese Networks for facial recognition.
   - Leveraging Hugging Face Transformers for chatbot responses.
4. **Integration:**
   - Connecting all components using Firebase for real-time data storage and synchronization.
5. **Testing and Feedback:**
   - Conducting iterative testing and refining based on user feedback to ensure a user-friendly, secure, and efficient system.

4.3 Website

4.3.1 Introduction

There are many things to consider when developing a website, from functionality and appearance to navigation and coding integrity, a lot goes into creating an eye-catching, user-friendly website. Determining a database to store in data and data types, a frontend that will loosely mirror the wireframe/prototype and building the backend to provide HTTP endpoints for the frontend and using also Firebase, authenticate users, authorize, and serve the front end. The purpose of Firebase is to make web application development easier and faster than coding a web app from scratch.

In our website we have three roles:

Admin	User	Visitor
✓ Manage users	× Manage users	× Manage users
✓ Manage issues	× Manage issues	× Manage issues
✓ Manage orders	× Manage orders	× Manage orders
✓ Manage, Create items	× Create, Manage items	× Manage, Create items
✓ Manage, Create inventory Reorder parts	× Manage, Create inventory Reorder parts	× Manage, Create inventory Reorder parts
✓ Manage profile	✓ Manage profile	× Manage profile
✓ Add issue	✓ Add issue	× Add issue
✓ Place an order	✓ Place an order	× Place an order
✓ Control smart devices	✓ Control smart devices	× Control components
✓ Can contact Leap Support	✓ Can contact Leap Support	✓ Can contact Leap Support

4.3.1.1 Admin Workflow

This flowchart outlines the admin portal login logic. It starts with an "Admin Portal Login" step and checks if the user is an admin. If not, the flow redirects to the "Users Login" path. If the user is an admin and the login is valid, they gain access to the admin dashboard functionalities including managing users, orders, issues, items, inventory, and settings.

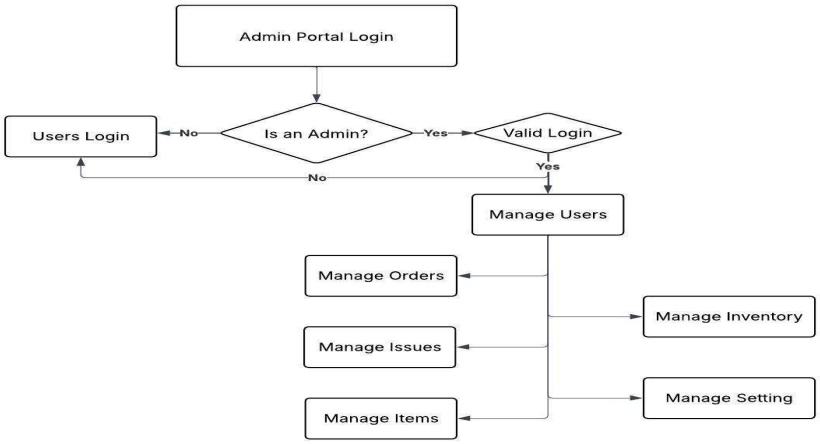


Figure 4.3.1.1 Admin Workflow

4.3.1.1 Admin Workflow

his diagram visualizes the user flow for an e-commerce platform starting from the landing page. It branches based on whether the user has an account or not. After signup or login, the system checks for completed orders. Based on order status, users are directed to either continue shopping or proceed to the "Living Room"(Where User Can Manage Home Components).

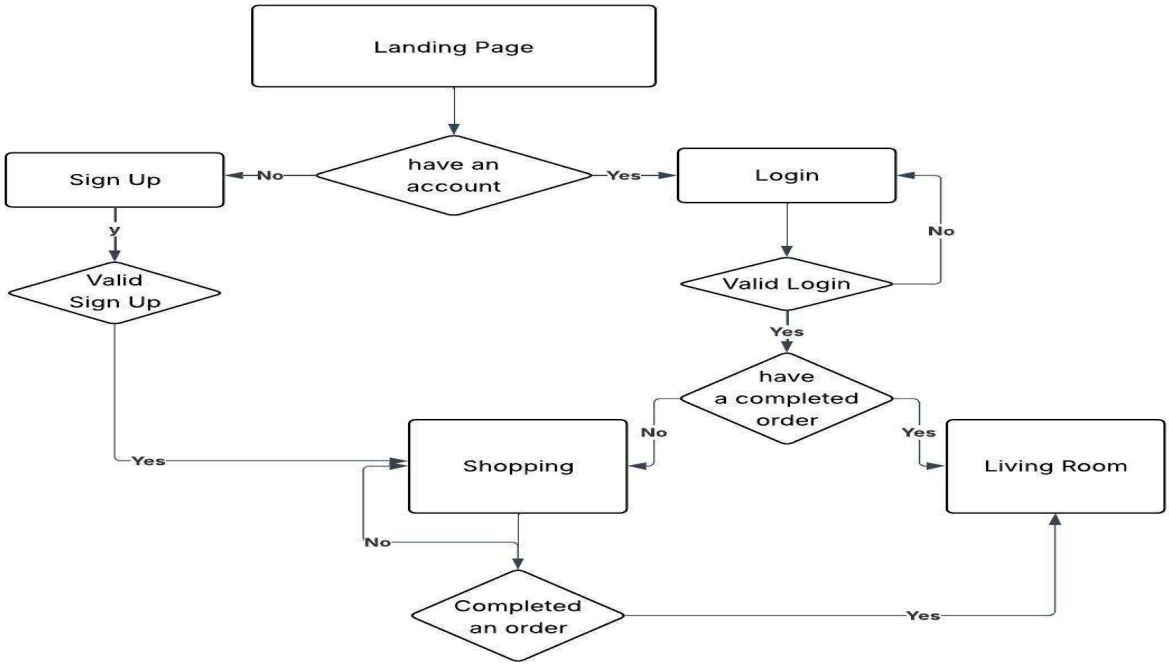


Figure 4.3.1.2 User Workflow

4.3.2 Front-End Development

The web life cycle will be presented in this part from beginning to end. In this part, specific parts of the web app implementation will be shown and discussed.

4.3.2.1 Tools

This project uses the three foundational technologies of the web: **HTML**, **CSS**, and **JavaScript**. Below is a brief overview of each.

HTML(HyperText Markup Language) is the standard markup language used to create the structure and content of web pages. It defines elements such as headings, paragraphs, links, images, forms, and more.

CSS(Cascading Style Sheets) is used to control the **presentation and layout** of HTML content. It allows you to apply styles such as colors, fonts, spacing, and positioning.

JavaScript(JS) is a high-level programming language used to add **interactivity and dynamic behavior** to web pages. It can manipulate the DOM, handle events, and interact with servers.

4.3.2.2 Admin Side

Admin Can login to the Website when the owner add admin email when he was just a user. And he can login as an admin.

4.3.2.2.1 Admin login

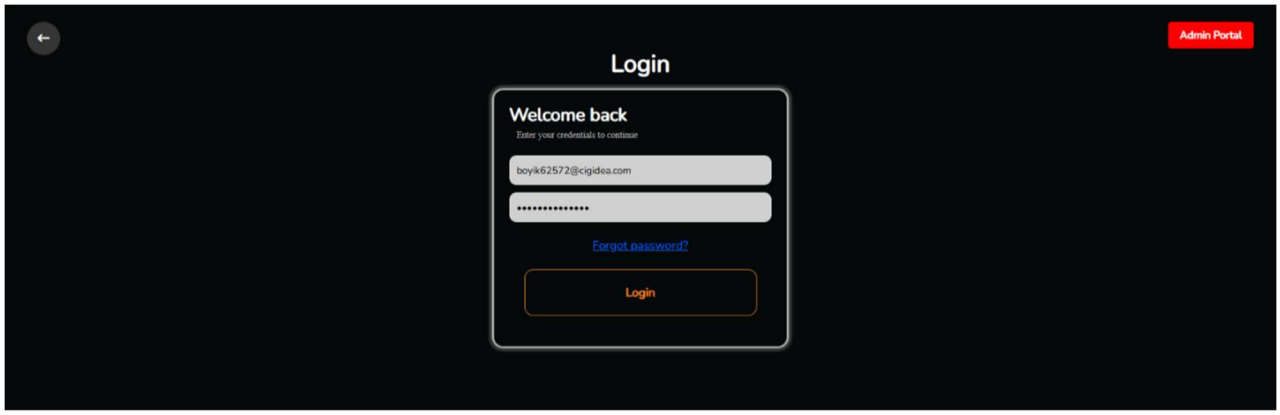


Figure 4.3.2.2.1 Admin login

4.3.2.2.2 Admin |Users

Once Admin login the Users Page will show up. In this page admin can read and manage users(update, delete).

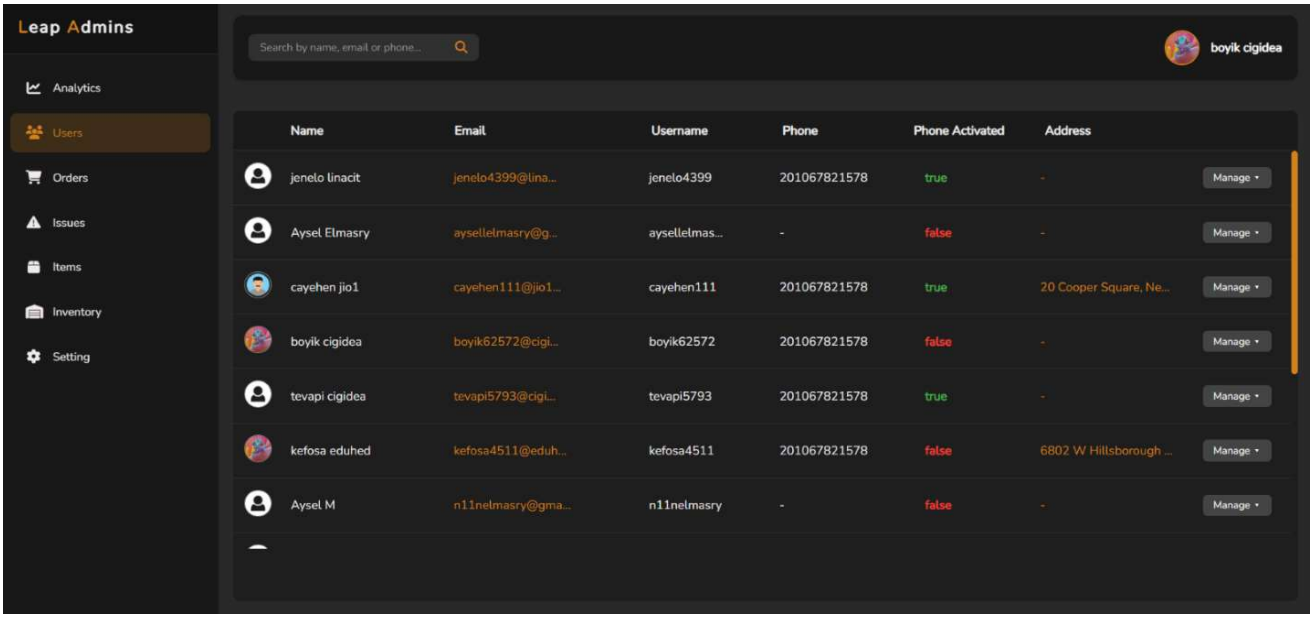


Figure 4.3.2.2.2 Admin |Users

4.3.2.2.2 Admin |Orders

In order page admin can update the status of the orders that users ordered and also filter orders based on status(pending, completed and canceled).

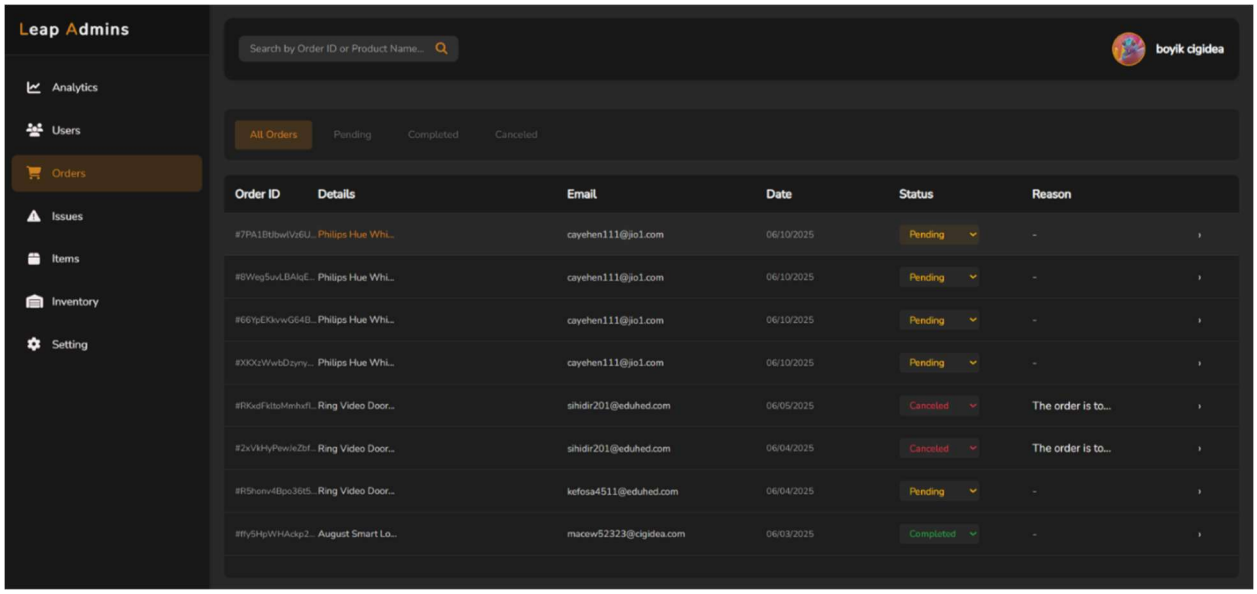


Figure 4.3.2.2.2 Admin |Orders

4.3.2.2.3 Admin |Issues

In issues page admin can handle the users issues and give each issue priority and also filter issues based on status(open, pending and closed).

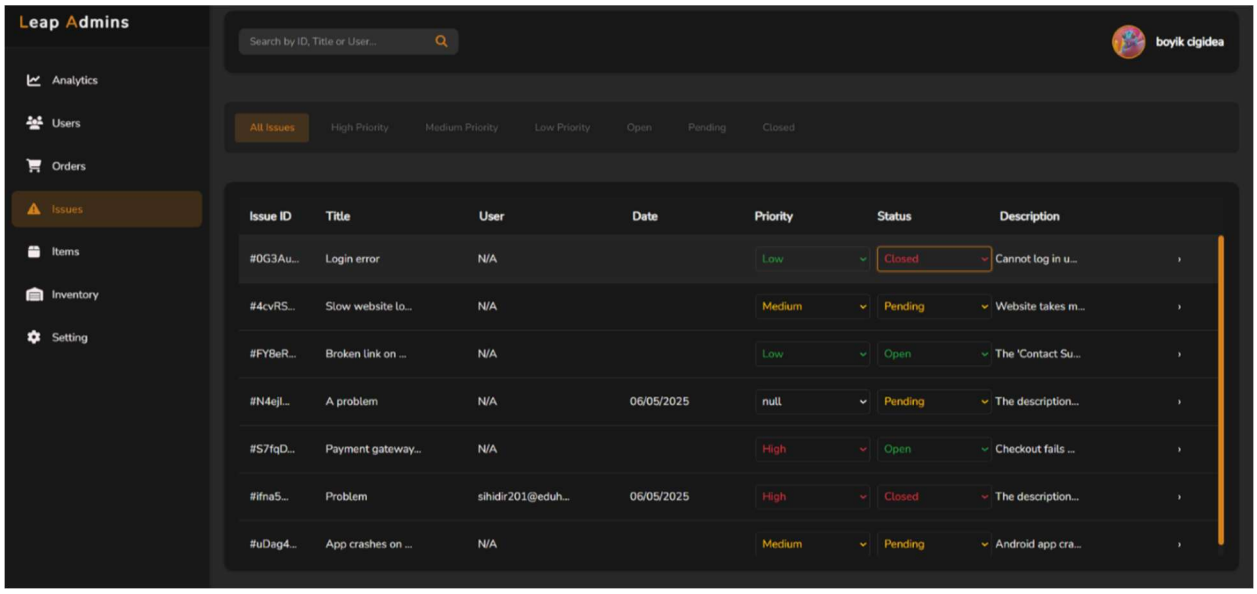


Figure 4.3.2.2.3 Admin |Issues

4.3.2.2.4 Admin |Items

In item page admin can add new item(ring, switch, etc..) that will show in the users part of the website.

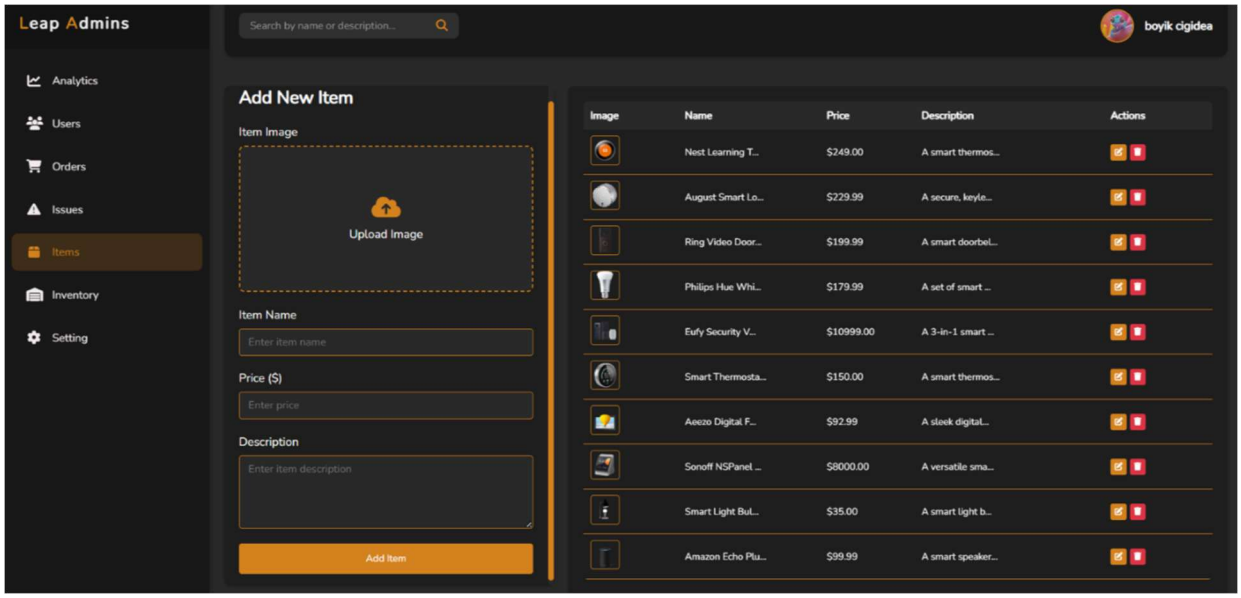


Figure 4.3.2.2.4 Admin |Items

4.3.2.2.5 Admin |Inventory

In inventory page admin can add new component(ESP32, PCB, etc..) and can reorder this parts if it is out of stock.

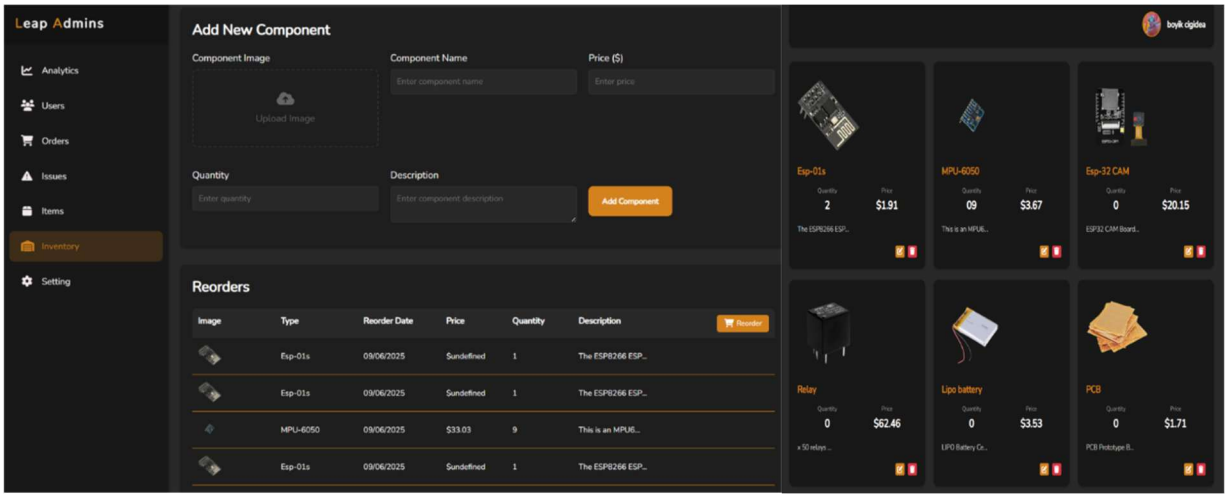


Figure 4.3.2.2.5 Admin |Inventory

4.3.2.2.6 Admin |Settings

In the setting page admin can update his profile by change profile photo or change password.

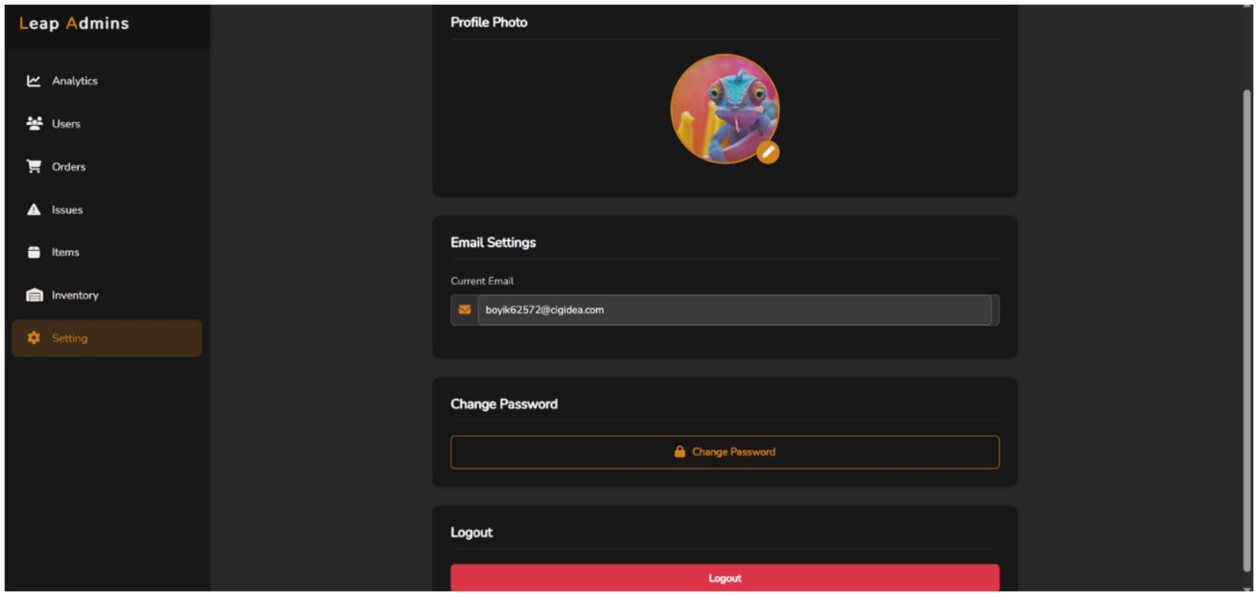


Figure 4.3.2.2.6 Admin |Settings

4.3.2.3 User Side

In user side we will focus on how user login , signup and verification of his email and phone number using WhatsApp OTP and other User Pages like Shopping and profile etc...

4.3.2.3.1 Landing Page

Landing page for user how will visit us, show the signup, login buttons for authentication etc...

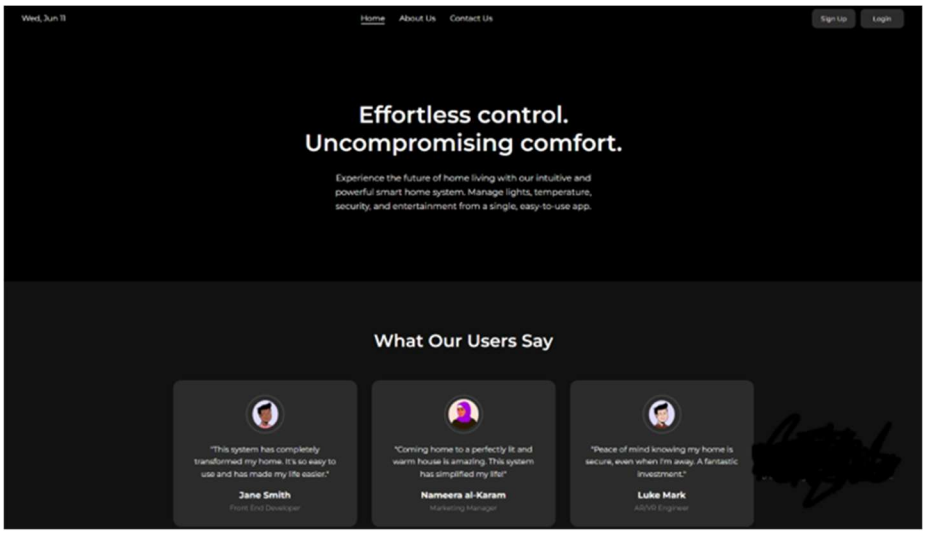


Figure 4.3.2.3.1 Landing Page

4.3.2.3.2 Signup Page, email and phone number verification

User signup and fill an input fields then Email and phone verification will show up and users can verify their email and phone number.

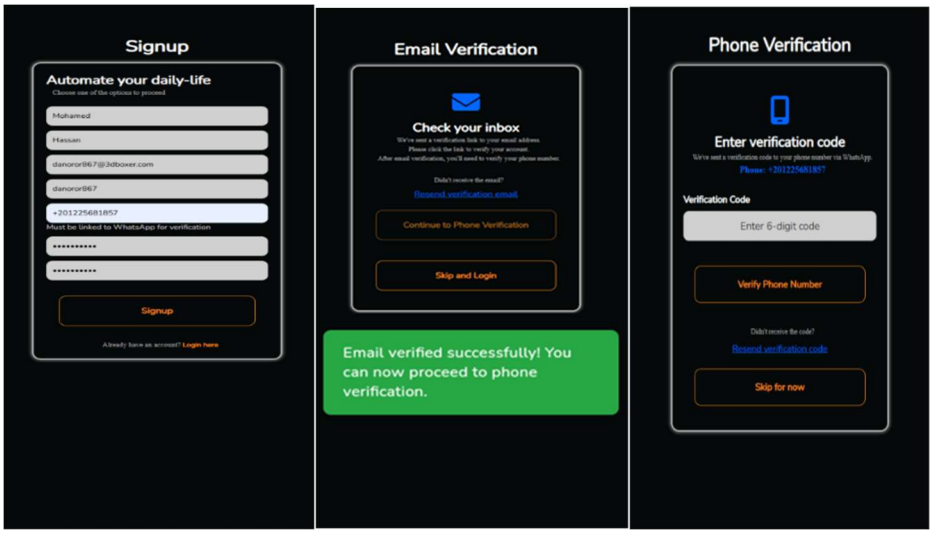
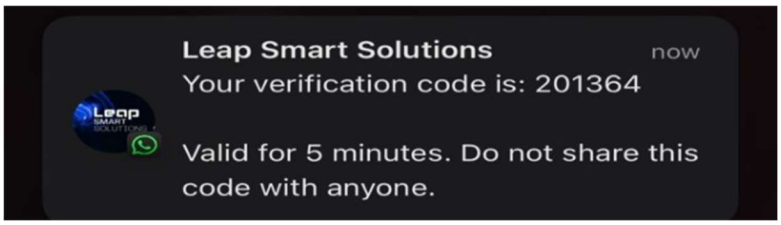


Figure 4.3.2.3.2 Signup Page, email and phone number verification

How OTP delivered by WhatsApp :



4.3.2.3.3 Login

Once the user Signup to the website he can usually login as a user.

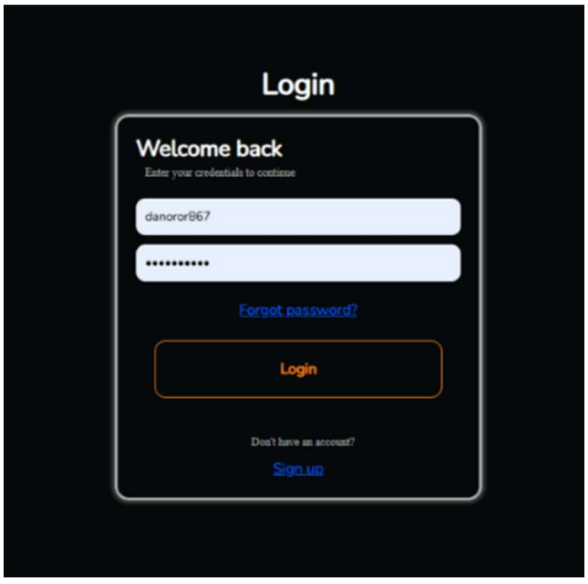


Figure 4.3.2.3.3 Login

4.3.2.3.4 Shopping Page

User can buy an a product by add them to the cart in the shopping cart then click the “Chek Out”

Button to buy the product

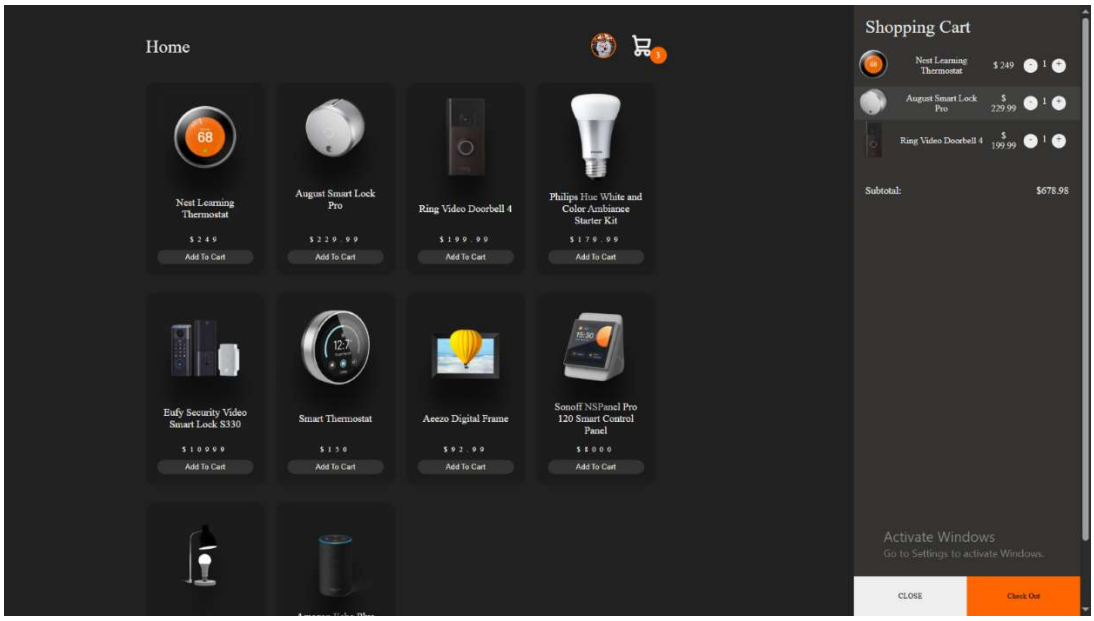


Figure 4.3.2.3.4 Shopping Page

4.3.2.3.5 Order Details

In product details page show the information of specific product and details.

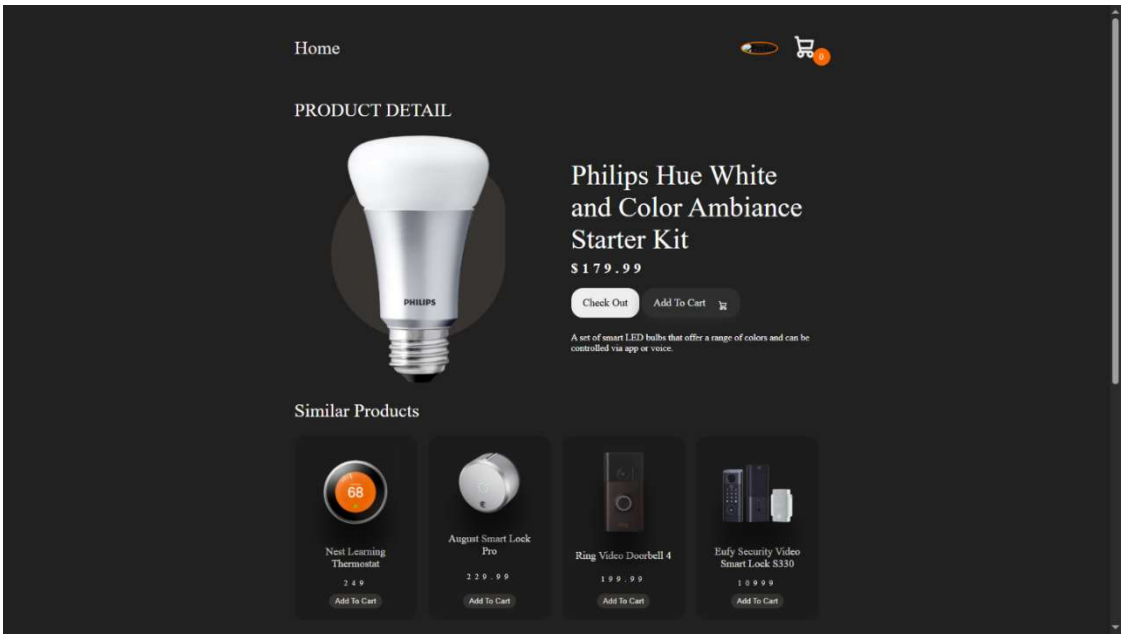


Figure 4.3.2.3.5 Order Details

4.3.2.3.6 Chek Out Page

User continue to fill the input fields(address, city, etc..) and once the user click on the “place order” button, the order will send to the admin orders page.

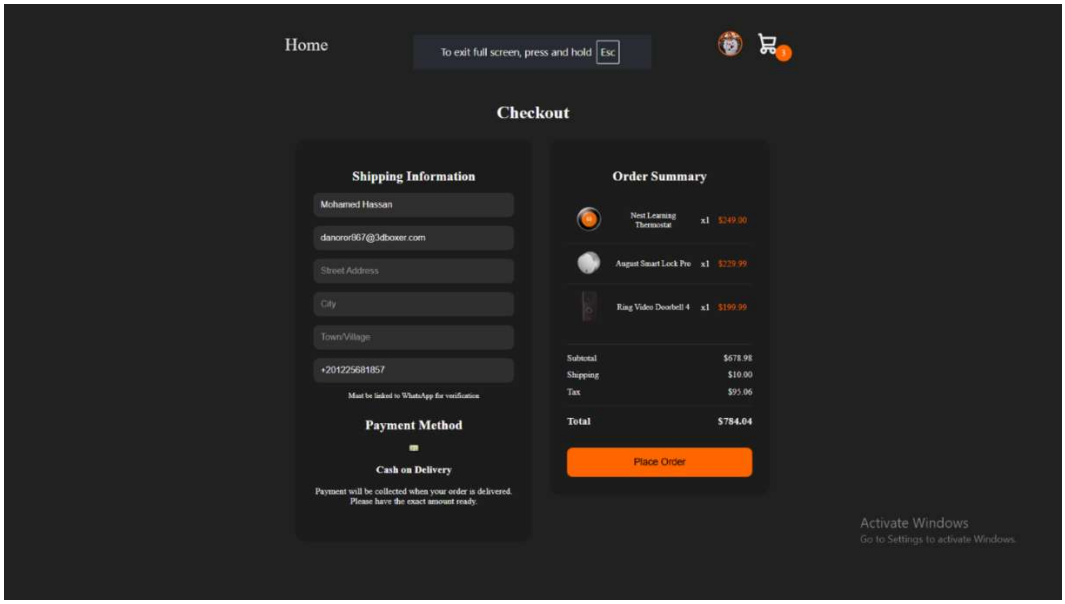
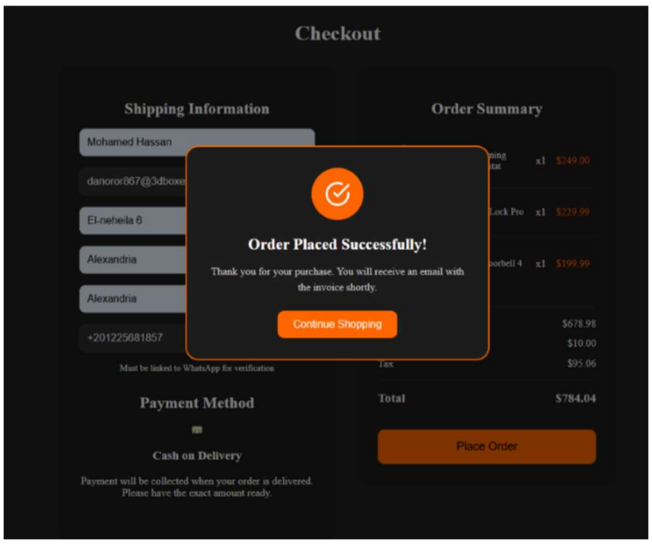


Figure 4.3.2.3.6 Chek Out Page

And Order Placed Successfully.



4.3.2.3.7 Create issues Page

User can create an issues that will delivered to the admin for what he needs.

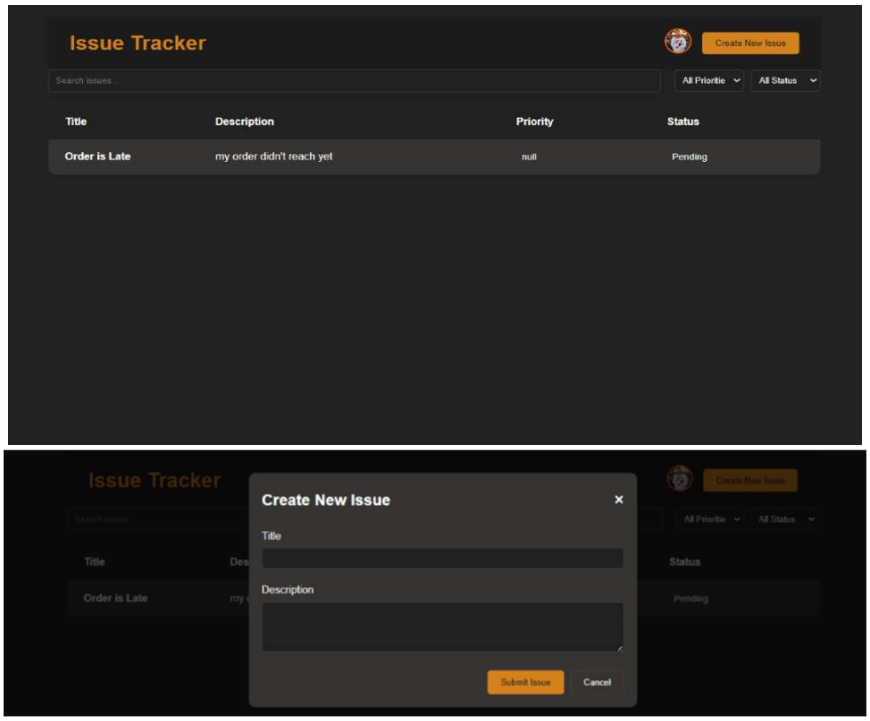


Figure 4.3.2.3.7 Create issues Page

4.3.2.3.8 Livingroom Page

Once admin select order completed the user will be able to visit the Home components.

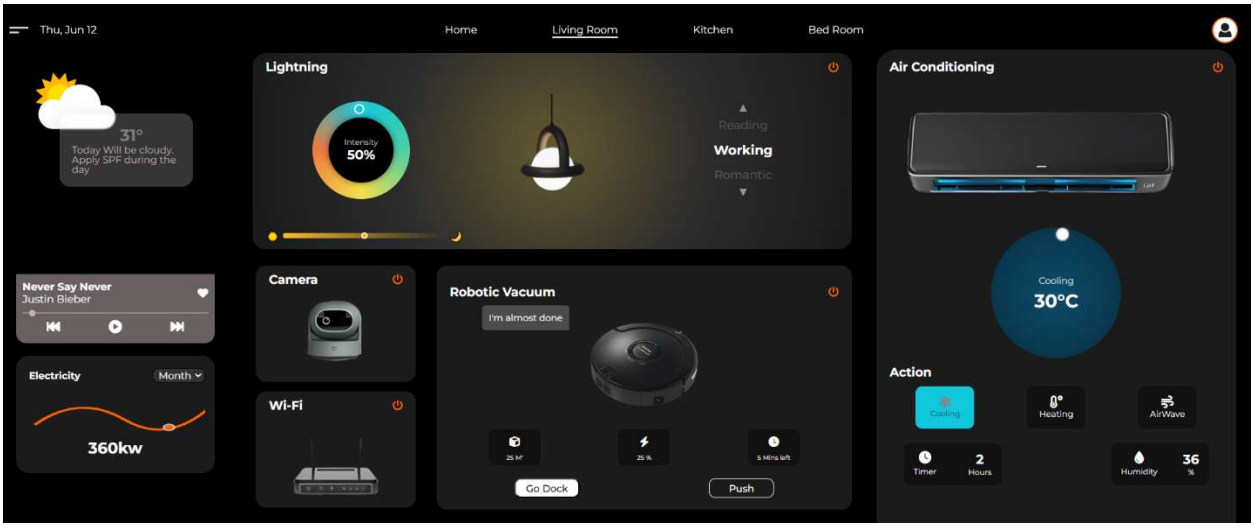


Figure 4.3.2.3.8 Livingroom Page

4.3.2.3.9 Profile Page

In the profile user can reset or change password.

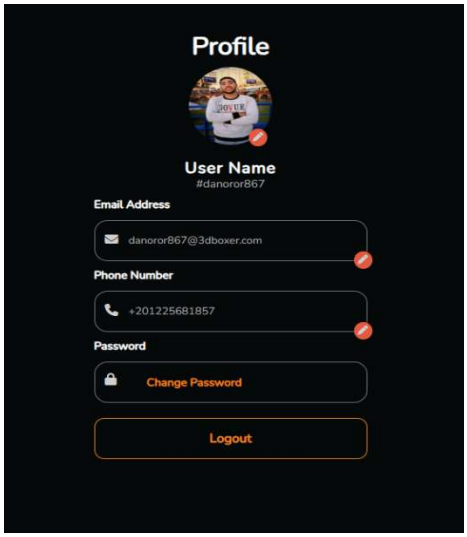


Figure 4.3.2.3.9 Profile Page

4.3.3 Backend Development

This section details the server-side architecture, technologies, and functionalities that power the Leap smart home system. The backend is responsible for handling user authentication, managing data, processing business logic, and ensuring seamless communication between the user interfaces (web and mobile) and the IoT devices.

4.3.3.1 Tools and Technologies

The backend is built using a modern JavaScript stack, chosen for its performance, scalability, and rich ecosystem. The primary tools include:

Node.js: A runtime environment for executing JavaScript on the server-side.

Express.js: A minimal and flexible Node.js web application framework used to build our API endpoints.

Firebase: A comprehensive platform from Google that provides several crucial backend services.

4.3.3.2 Backend Design Pattern

The Leap system's backend is initially designed using a monolithic architecture. This approach centralizes the codebase, which simplifies development, testing, and deployment in the early stages of the project. All core functionalities including user management, device control, and API endpoints are managed within a single, unified application. This pattern was chosen for its

straightforwardness, allowing for rapid development and iteration, which aligns with our Agile methodology.

As the system grows in complexity and user load, we plan to evolve towards a microservices architecture to enhance scalability and maintainability, as mentioned in our future work.

4.3.3.3 Authentication and Authorization

Secure access is managed through a multi-layered authentication and authorization process:

User Authentication: We use Firebase Authentication to manage user sign-up and login processes. It provides a secure and easy-to-implement solution for handling user credentials, including email/password authentication.

Phone Verification: To enhance security and validate user identity, we have implemented a phone number verification step during signup using a custom WhatsApp OTP (One-Time Password) service. This service is an open-source project available at https://github.com/Abd-El-Rahman-Mohamed/whatsapp-otp

and is deployed on Railway at https://whatsapp-otp-production.up.railway.app. Upon sign-up, a One-Time Password (OTP) is sent to the user's WhatsApp number. This OTP is used to verify their phone, enhancing the confidentiality of future WhatsApp communications.

Email Verification: To enhance security and validate user identity, we have implemented an email verification step during signup using Firebase Authentication. When a user signs up, a verification email is automatically sent to their registered email address. The user must click the verification link in the email to complete their registration. This ensures that only users with valid email addresses can access the system. The verification status is tracked in Firebase, and users cannot access certain features until their email is verified. This is followed by phone verification using WhatsApp OTP for additional security.

Password Management: The system implements comprehensive password management features to ensure account security:

- **Change Password**: Authenticated users can change their password through their profile settings:
  1. Requires current password verification
  2. Updates password in Firebase Authentication
  3. Maintains session security during password change

- **Reset Password**: Users who have forgotten their password can request a reset:
  1. Accessible from the login page

2. Requires email verification
3. Sends password reset link to user's registered email
4. Link expires after 24 hours for security
5. Allows setting a new password without knowing the old one
6. Requires email verification before allowing password reset
7. Sends confirmation email after successful reset

- **Security Features**:
  1. Rate limiting on password attempts
  2. Secure password storage using Firebase Authentication
  3. Automatic session invalidation on password change

Profile Image Management: The system provides a user-friendly interface for managing profile pictures:

- **Image Upload**: Users can update their profile picture through their profile settings:
  1. Supports common image formats (JPG, PNG, JPEG)
  2. Drag-and-drop interface for easy upload
  3. Using Imgbb image hosting platform for Image hosting
- **Image Processing:**
  1. Automatic image resizing to standard dimensions
  2. Maintains aspect ratio
  3. Stores images urls in Firebase Firestore database

- **Storage and Retrieval:**
  1. Secure storage in Firebase Storage
  2. Image hosting platform delivery for fast image loading
- **User Experience:**
  1. Real-time preview of selected image
  2. Immediate update across all user interfaces
  3. Fallback to default avatar if upload fails
  4. Error handling for failed uploads

Role-Based Access Control: The system distinguishes between two main roles: 'User' and 'Admin'. By default, all new sign-ups are assigned the 'User' role. To elevate a user's privileges to 'Admin', the user must contact the system owner directly. This manual process ensures a high level of security for administrative functions.

Order-Based Access Control: To ensure that only verified customers can access certain features, the system implements an order-based access control mechanism. Specifically, access to the Living Room, Bedroom, and Kitchen control pages is restricted to users who have completed at least one order. This is implemented through a dual-check system:

- The system verifies the user's authentication status
- It then checks both the main orders collection and the user's subcollection for completed orders
- Access is granted only if the user has at least one completed order
- Users without completed orders are redirected to the Products to make an order This ensures that only legitimate customers who have purchased the system can access the smart home control features.

4.3.3.4 Users Management System

The backend implements a secure user management system that is exclusively accessible to administrators:

- **User Management**: Admins can manage user accounts through a dedicated interface, with each user's data stored in Firebase with the following structure:
  1. User ID
  2. User profile image
  3. Email (verified)
  4. Phone number (verified/not verified)
  5. Username
  6. Address
  7. Full Name
- **Admin Controls:** The system provides comprehensive administrative capabilities:
  1. View all registered users and their details
  2. Delete user accounts
  3. Monitor user order history
  5. Manage user order status
  6. Access user issues
- **Security Features:**
  Role-based access control ensures only admins can access the management interface

4.3.3.5 Order Management System

The backend implements a comprehensive order management system that handles the entire order lifecycle:

- **Order Creation**: Users can place orders through the shopping interface, with each order stored in Firebase with the following structure:
  1. Order ID and timestamp
  2. User ID and shipping information
  3. Product details and quantities

4. Total price and payment status
5. Current status (pending, completed, canceled) pending is the default

- **Order Processing**: The system provides different views for users and admins:
  1. Users can view their order history and the order status via the email address
  2. Admins can view all orders, update their status, and add tracking information
  3. Admins can filter orders based on their status, and add tracking information
  4. Updates ensure both users and admins are viewing the status changes
  5. Order completion triggers access to smart home control features

4.3.3.6 Issue Management System

The backend implements a comprehensive issue management system that allows users to create and track support requests:

- **Issue Creation**: Users can create support tickets through a dedicated interface, providing details about their concerns or requirements. Each issue is stored in Firebase with the following structure:
  1. Issue title and description
  2. User ID
  3. Creation timestamp
  4. Current status (open, pending, closed) pending is the default
  5. Priority level (set by admin)
- **Issue Processing:** The system provides different views for users and admins:
  1. Users can create new issues and view their own issue history
  2. Admins can view all issues, update their status, set priority levels
  3. Updates ensure both users and admins are viewing the status changes
  4. Admins can filter issues based on their priority, and status

4.3.3.7 Items Management System

The backend implements a secure items management system that is exclusively accessible to administrators:

- **Product Management:** Admins can manage the product catalog through a dedicated interface, with each product stored in Firebase with the following structure:
  1. Product ID and name
  2. Description and specifications
  3. Price and availability status
  4. Product images is Hosted at Imgbb for remote access instead of local hosting
- **Admin-Only Features**: The system provides comprehensive management capabilities exclusively for administrators:
  1. Create new products with detailed specifications
  2. Update existing product information and prices
  3. Setting reorders

4. Upload and manage product images

- **Security Measures:**
  1. Admin-only access enforced through Firebase Authentication
  2. Input validation to prevent data corruption

4.3.3.8 Inventory Management System

The backend implements a secure inventory management system exclusively for administrators to manage IoT component parts:

- **Inventory Control:** Admins can manage the parts inventory through a dedicated interface, with each item stored in Firebase with the following structure:
  1. Part ID and name
  2. Current stock quantity
  3. Last restock date
  4. Cost per unit
- **Admin-Only Features:**
  1. Add new parts to inventory
  2. Update stock quantities
  3. Track part usage in IoT components
  4. View inventory and reorder history
- **Firebase Integration**: Inventory is stored in a secure Fire store collection with strict access control:
  1. Only admin users can access and modify inventory data
  2. Reorders monitoring
  3. Audit trail of all inventory changes
- **Security Measures:**
  1. Role-based access control ensures only admins can access inventory
  2. All inventory changes are logged with admin ID and timestamp
  3. Regular inventory audits can be performed

4.3.3.9 Automated Email Invoicing

After a user successfully places an order through the website, the backend automatically sends a detailed invoice to the user's registered email address. This functionality is implemented using EmailJS, a service that allows sending emails directly from the client-side or server-side code without needing a full backend server for email management.

4.3.3.10 Real-Time Database and IoT Integration

The core of our smart home control functionality relies on real-time data synchronization between the user interfaces and the connected IoT devices.

- **Firebase Real-Time Database:** We utilize Firebase's Real-Time Database, a NoSQL cloud database, to store and sync data instantly. This is crucial for real-time control of smart devices.
- Living Room Lighting Control: For example, the lighting in the 'Livingroom' is controlled by a specific entry in the database. The frontend application updates a value in the database, and the corresponding smart light (connected via an ESP module) listens for changes to that value.
    1. Setting the value to 1 turns the light on.
    2. Setting the value to 0 turns the light off. This ensures that commands from the web or mobile app are executed with minimal latency.

4.3.3.11 Backend Testing

To ensure the reliability and robustness of our backend, we have planned a comprehensive testing strategy that will be implemented in future iterations:

- **Unit Testing:** We plan to write unit tests for individual functions and modules (e.g., authentication logic, API route handlers) to verify their correctness in isolation.
- Integration Testing: We plan to conduct integration tests to ensure that different parts of the backend (like the API, database, and external services) work together seamlessly. This will include testing the end-to-end flow of user requests, from the API endpoint to the database and back.

4.4 Mobile Application

4.4.1 Overview of Application Architecture

The Leap smart home application represents a revolutionary comprehensive security and automation system that seamlessly integrates cutting-edge mobile development technologies with advanced IoT capabilities. Built upon the robust Flutter framework, the application harnesses Firebase's enterprise-grade backend services to deliver an unparalleled user experience that combines security, convenience, and intelligent automation. The architecture embraces a sophisticated modular design pattern that elegantly separates concerns across authentication, multi-modal device management, advanced security surveillance, intelligent facial recognition, and AI-powered chatbot interactions.

The application's architectural foundation establishes a new paradigm in smart home control by implementing a multi-layered security approach that encompasses biometric authentication, real-time surveillance monitoring, and intelligent threat detection. This comprehensive system transforms traditional home automation into an intelligent security ecosystem that adapts to user behavior patterns while maintaining vigilant protection against unauthorized access and security threats.

4.4.2 Multi-Modal Authentication System Implementation

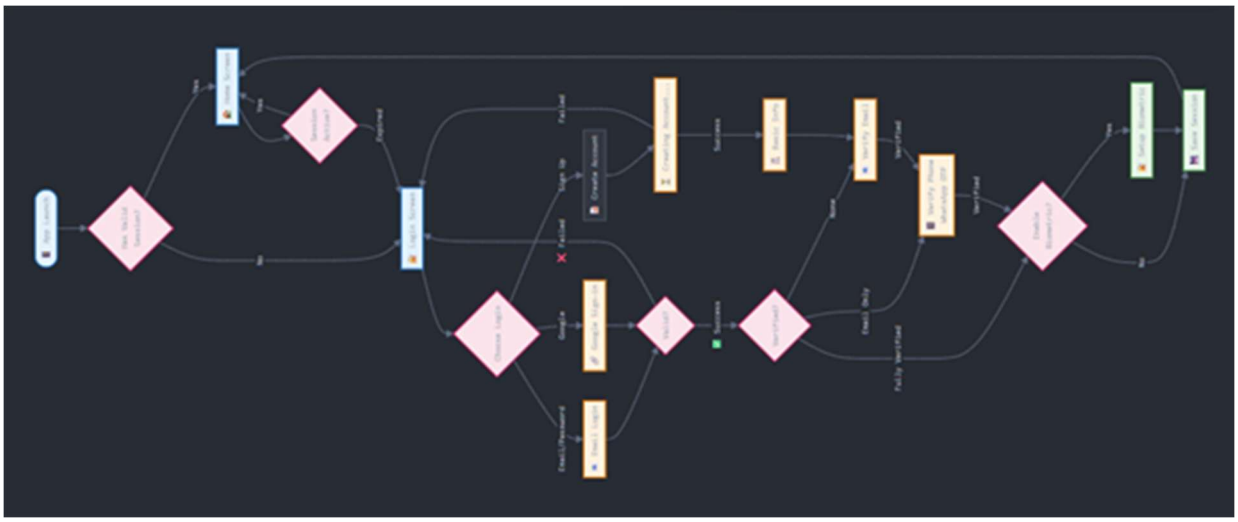
The authentication framework revolutionizes user security through a comprehensive multi-modal verification system that accommodates diverse user preferences while maintaining enterprise-grade security standards. The Firebase Authentication integration provides a robust backend foundation that supports multiple authentication pathways, ensuring seamless user experiences across different verification methods.

The email and password authentication system implements advanced form validation with real-time error handling and comprehensive security measures including password strength requirements and breach detection. Users navigate through an intuitive registration process that guides profile creation while implementing security protocols that protect against common vulnerabilities.

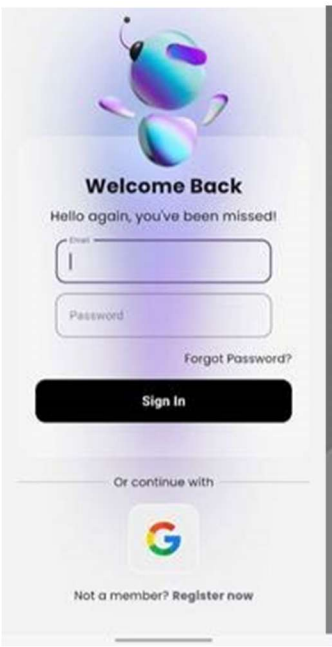
Google Sign-in integration leverages OAuth 2.0 protocols to provide users with familiar single sign-on experiences that eliminate password fatigue while maintaining secure access credentials. This implementation significantly reduces onboarding friction while ensuring enterprise-level security through Google's authentication infrastructure.

The innovative WhatsApp OTP verification system transforms traditional phone verification by utilizing users' preferred messaging platform for security confirmation. This approach demonstrates the application's adaptability to modern communication preferences while maintaining robust security verification standards.

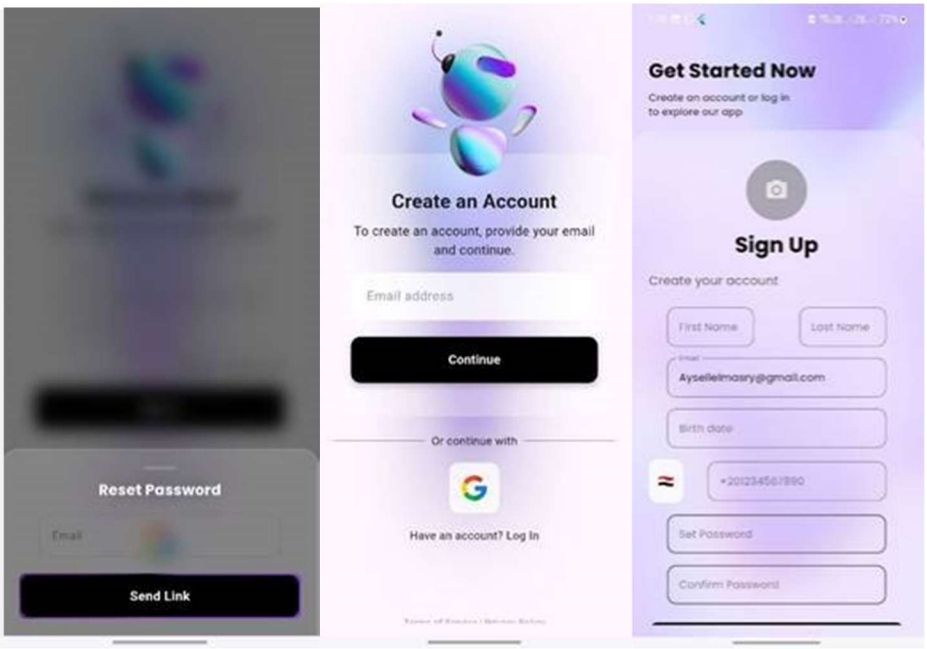
Additional security enhancements include biometric authentication capabilities that enable fingerprint and facial recognition for rapid device access, password reset functionality through secure email verification, and comprehensive session management with automatic logout and token refresh mechanisms.



[Figure 4.4.2.1: Authentication Flow Diagram]



[Figure 4.4.2.2: Login Interface Screenshot]



[Figure 4.4.2.3: Registration Process Screenshots]

[light Mode]

4.4.3 Advanced Device Management and Integration System

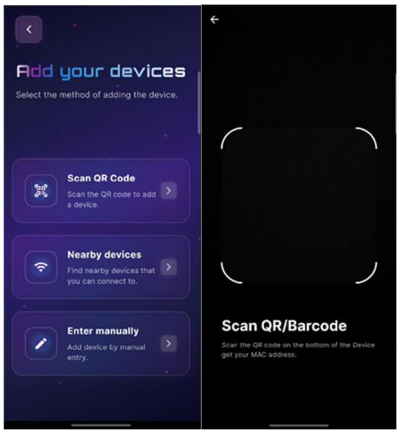
The device management system pioneers innovative approaches to IoT device integration through three sophisticated connection methodologies that accommodate diverse device types and user technical capabilities. The QR code and barcode scanning functionality employs advanced OCR technology to instantly recognize device information with perfect accuracy, eliminating manual data entry errors while dramatically accelerating the setup process. This automated approach transforms traditionally complex device configuration into a simple scanning operation.

Bluetooth discovery capabilities showcase intelligent device detection algorithms that automatically identify nearby IoT devices while filtering based on compatibility, proximity, and security protocols. The system presents users with curated lists of available devices, ensuring only relevant and secure connection options appear during the setup process.

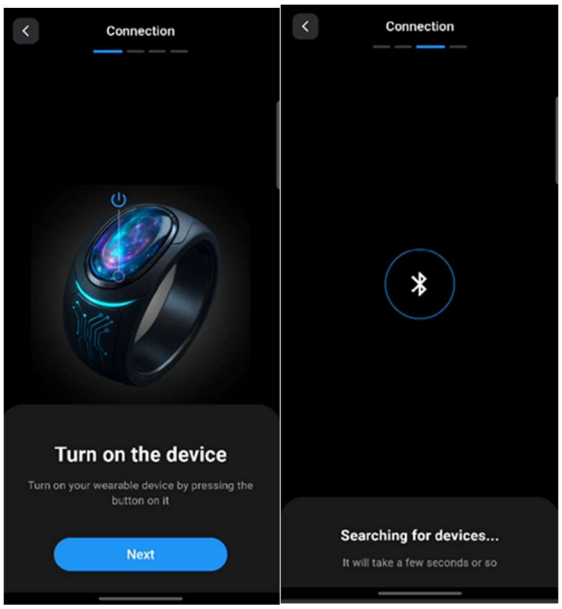
Manual device entry maintains system inclusivity by accommodating advanced users and unique device configurations that may not support automated discovery methods. This comprehensive approach ensures universal device compatibility while maintaining the sophisticated automation features that define the smart home experience.

The device registration process implements meticulous data collection protocols that capture essential device information including MAC addresses (the critical identifier for network security and device control), user-defined device names, logical room assignments, and comprehensive device type classifications. All device configurations are securely stored in Firebase under individual user profiles, ensuring personalized device ecosystems that remain accessible across multiple devices and sessions.

Device control functionality provides real-time on/off toggle capabilities with instant status feedback, comprehensive device status monitoring with health diagnostics, individual device detail pages with advanced control options, secure device removal with confirmation safeguards, and bulk operation capabilities for managing multiple devices simultaneously.



[Figure 4.4.3.1: Device Addition Methods Interface] [Figure 4.4.3.2: QR Scanner Implementation]



[Figure 4.4.3.3: Bluetooth Discovery Interface] [Figure 4.4.3.4: Device Configuration]

4.4.4 Comprehensive Security Monitoring and Intelligent Surveillance

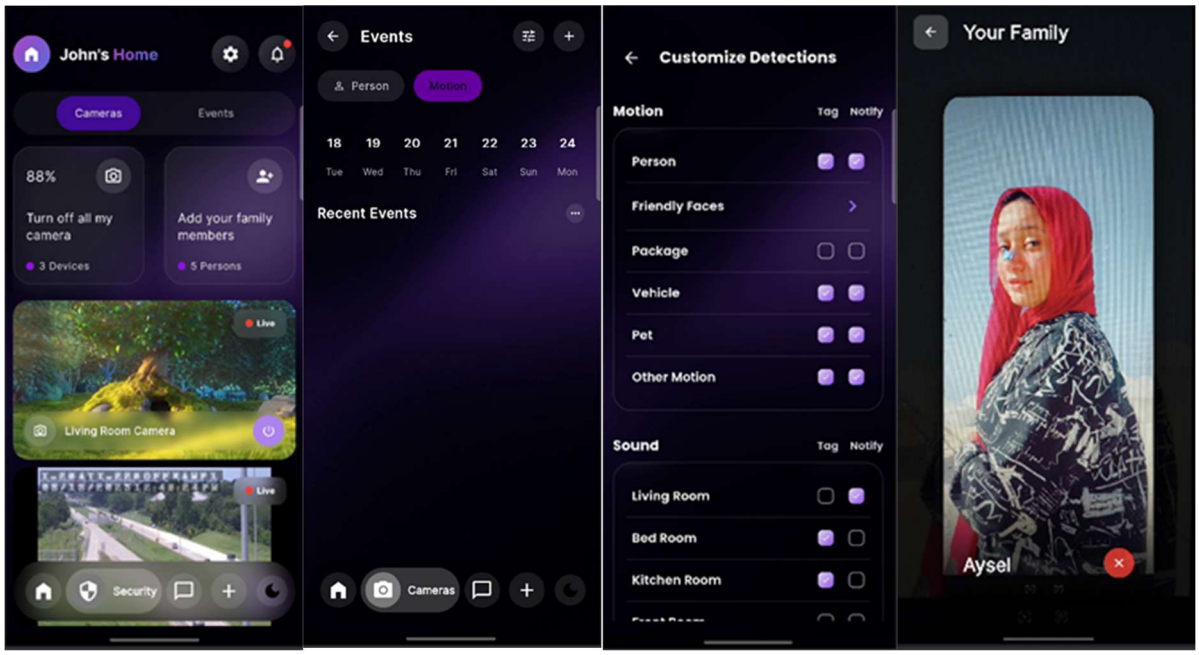
The security monitoring system transforms the application into a sophisticated surveillance and threat detection platform through advanced camera integration, intelligent facial recognition, and real-time security analytics. The live camera streaming system provides continuous real-time monitoring capabilities across multiple rooms with adaptive streaming quality that automatically adjusts based on network conditions to ensure consistent performance.

The revolutionary facial recognition system employs state-of-the-art CNN and Siamese neural networks to deliver precision identification capabilities that accurately distinguish between family members and unknown individuals. This sophisticated AI implementation creates a personalized security ecosystem that continuously learns and adapts to each household's unique composition while maintaining exceptional accuracy with minimal false positives.

Advanced camera control features enable users to pause and resume live streams for bandwidth management, capture instant screenshots with automatic Firebase storage for security documentation, and initiate event-triggered recording sessions that provide comprehensive visual evidence of security events. The system intelligently manages storage resources while ensuring critical security footage remains accessible for review and analysis.

The intelligent event management system maintains comprehensive security logs over a seven-day historical period, providing users with detailed timeline analysis of their home's security status. The system employs advanced algorithms to categorize events by type, severity, and context, enabling users to quickly identify genuine security concerns while filtering routine household activities.

Unknown person detection capabilities represent a breakthrough in residential security technology, automatically identifying unfamiliar individuals and triggering immediate notification protocols. The system provides users with multiple response options including adding the person to the family database, initiating emergency contact procedures, or dismissing false alarms, ensuring appropriate responses to various security scenarios.



[Figure 4.4.4.1: Security Dashboard Interface] [Figure 4.4.4.3: Facial Recognition Management]

[Figure 4.4.4.4: Event History Timeline]

4.4.5 AI-Powered Dual-Mode Chatbot Integration

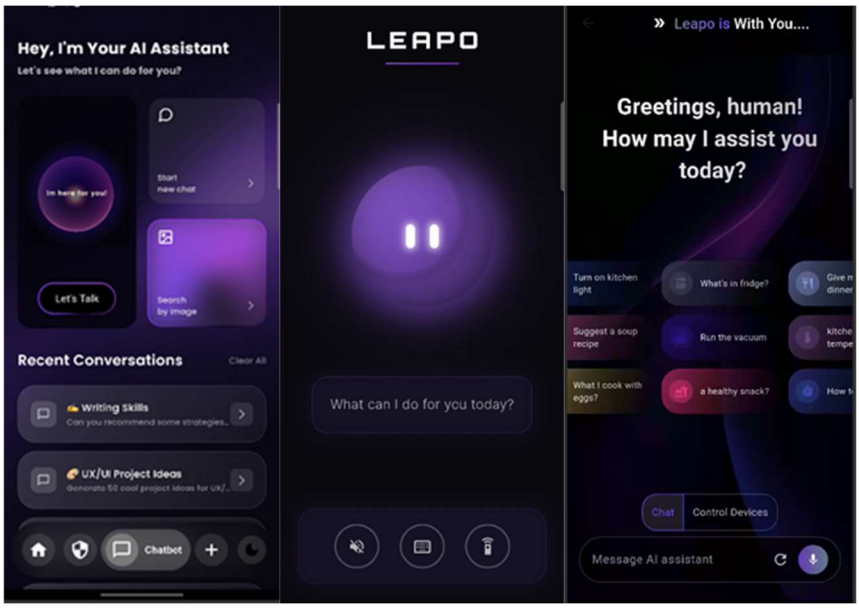
The chatbot system represents a paradigm shift in smart home interaction through revolutionary dual-mode communication capabilities that seamlessly blend text-based conversation with advanced voice recognition technology. The text-based chat interface provides users with a familiar messaging experience that processes natural language commands and translates them into precise device actions, making sophisticated smart home control accessible to users regardless of their technical expertise or familiarity with IoT systems.

Voice interaction capabilities elevate the user experience to unprecedented levels through cutting-edge Speech Recognition and Text-to-Speech technologies that enable completely hands-free control of the entire smart home ecosystem. Users can engage in natural conversational exchanges with their smart home system, receiving intelligent audio responses that confirm actions and provide status updates in real-time.

The chatbot's artificial intelligence extends far beyond simple command processing to include sophisticated contextual understanding that learns from user interaction patterns, preferences, and behavioral habits. This intelligent system proactively suggests optimal device configurations, energy-saving opportunities, and security enhancements based on historical usage data and environmental conditions.

Comprehensive API integration ensures seamless real-time communication between the chatbot interface and all connected IoT devices, enabling instant control responses and status updates through natural language interactions. The system maintains detailed chat history that users can reference for previous commands and system responses, creating a comprehensive interaction log that enhances user experience and system transparency.

Advanced conversation management includes intelligent response generation, contextual memory retention, and personalized interaction adaptation that makes each conversation feel natural and productive while maintaining the sophisticated functionality that defines professional smart home automation systems.



[Figure 4.4.5.1: Chatbot Interface Design] [Figure 4.4.5.2: Voice Command Processing]

[Figure 4.4.5.3: Chat History Management]

4.4.6 Intelligent Real-Time Notification and Alert System

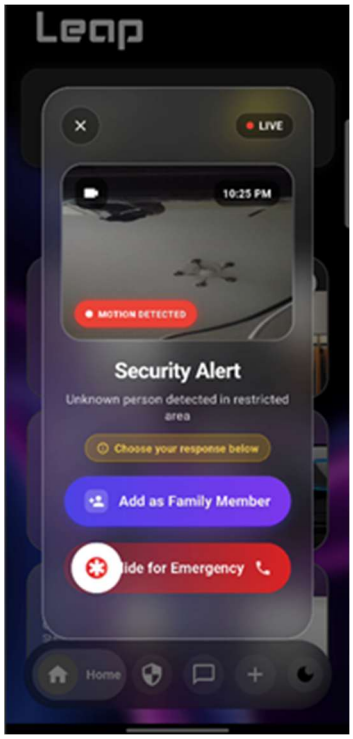
The notification system delivers a sophisticated communication framework that intelligently prioritizes critical security alerts while providing comprehensive device status updates and system notifications. The advanced Firebase-powered notification engine ensures users remain informed about their home's security status and device functionality regardless of their physical location or the application's active state.

The system's revolutionary unknown person detection feature represents a breakthrough in residential security technology, automatically triggering immediate notifications when unfamiliar individuals are detected by the facial recognition system. These critical alerts provide users with multiple intelligent response options including adding the detected person to the family database for future recognition, initiating emergency contact procedures through integrated police calling functionality, or dismissing false alarms with detailed reasoning logs.

Sophisticated alert categorization algorithms distinguish between urgent security notifications requiring immediate attention, routine device status updates, informational system messages, and preventive maintenance alerts. This intelligent classification system prevents notification fatigue while ensuring critical security events receive immediate priority attention through distinctive alert tones, visual indicators, and priority delivery mechanisms.

Interactive notification functionality revolutionizes emergency response procedures by enabling users to respond directly to security alerts without opening the application interface. This streamlined approach ensures rapid response times during critical security events while maintaining detailed logs of user responses and actions taken.

Cross-platform notification support ensures consistent functionality and feature parity across Android and iOS devices, maintaining uniform security coverage regardless of users' mobile platform preferences. The system includes customizable notification settings that allow users to tailor alert preferences, quiet hours, priority contacts, and emergency response protocols according to their specific security requirements and lifestyle preferences.



[Figure 4.4.6.1: Notification Center Interface]

4.4.7 User Interface Design and Experience Philosophy

The application's visual design philosophy embraces a cutting-edge futuristic aesthetic that perfectly reflects the innovative nature of advanced smart home technology while maintaining exceptional usability and intuitive navigation. The interface employs sophisticated modern design principles including elegant frosted glass effects, fluid micro-animations, carefully orchestrated transition sequences, and meticulously crafted color palettes that create an engaging and premium user experience that feels both familiar and revolutionary.

Adaptive theming capabilities provide users with comprehensive customization options that extend beyond simple dark and light modes to include personalized color schemes, typography preferences, and interface density options. The expertly implemented dark mode significantly reduces eye strain during low-light usage scenarios while maintaining perfect visual hierarchy, contrast ratios, and readability across all interface elements and interaction states.

Responsive layout design ensures optimal functionality and visual appeal across the complete spectrum of mobile devices, from compact smartphones to large tablets, while maintaining consistent user experiences across different screen sizes, orientations, and aspect ratios. The interface dynamically adapts to device capabilities while preserving functionality and aesthetic appeal.

Gesture-based navigation enhances interaction efficiency through intuitive shortcuts for common actions including swipe gestures for quick device control, pinch-to-zoom for camera feeds, pull-to-refresh for status updates, and long-press for contextual menus. These advanced gestures complement traditional navigation methods without replacing them, offering power users enhanced control while maintaining universal accessibility.

The UI/UX design prioritizes ease of use through logical information architecture, consistent interaction patterns, clear visual feedback for all user actions, intuitive icon design, and streamlined workflows that minimize cognitive load while maximizing functionality. Professional-grade design elements include subtle shadows, consistent spacing systems, harmonious color relationships, and sophisticated typography that creates a premium user experience rivaling the best consumer applications.

4.4.8 Device Control and Management Implementation

Individual device control interfaces demonstrate the application's exceptional versatility in managing diverse IoT ecosystems through specialized, purpose-built control panels that optimize user interaction for each device category. The sophisticated lighting control system provides comprehensive brightness adjustment with precise dimming capabilities, advanced color temperature modification for circadian rhythm optimization, intelligent scheduling features that adapt to users' daily routines, and energy consumption monitoring that promotes sustainable usage patterns.

Air conditioning management interfaces offer precision environmental control through intelligent temperature regulation systems that learn from user preferences and external weather conditions. The system provides comprehensive scheduling capabilities with vacation modes, energy optimization suggestions based on occupancy patterns, humidity control integration, and predictive maintenance alerts that ensure optimal performance while minimizing energy consumption.

Vacuum cleaner integration showcases the application's capability to manage sophisticated autonomous devices through comprehensive cleaning schedule management, zone-specific cleaning commands with detailed room mapping, maintenance status monitoring with filter replacement notifications, and remote cleaning initiation with real-time progress tracking. Users can monitor cleaning progress, receive completion notifications, and access detailed cleaning reports that track coverage patterns and efficiency metrics.

Smart door lock controls provide advanced security management through multiple authentication methods including biometric verification, secure PIN code entry, proximity-based unlocking using smartphones, and temporary access codes for guests. The system maintains comprehensive access logs with timestamps, user identification, and entry methods while providing users with flexible control over their home's security perimeter and access management protocols.

Each device control interface includes real-time status indicators, historical usage analytics, energy consumption tracking, maintenance scheduling, and predictive failure analysis that helps users optimize device performance while extending operational lifespan through proactive maintenance recommendations.

[Figure 4.4.8.1: Lighting Control Interface] [Figure 4.4.8.2: Air Conditioning Management] [Figure 4.4.8.3: Vacuum Control Panel]

4.4.9 Advanced Face Recognition and Family Management System

The face recognition system represents a revolutionary advancement in residential security technology through sophisticated family member management capabilities that combine convenience with comprehensive security monitoring. The system supports multiple face registration methods including high-quality gallery photo uploads and real-time camera capture sessions that ensure optimal facial recognition accuracy across various lighting conditions and angles.

Family member database management provides intuitive interfaces for adding new family members with detailed profile information, editing existing profiles with updated photos and personal details, and removing unwanted or outdated facial recognition data. The system maintains comprehensive facial recognition logs that track recognition events, accuracy scores, and system confidence levels for continuous improvement and security auditing.

Advanced facial recognition algorithms employ cutting-edge machine learning techniques that adapt to facial changes over time, accommodate different lighting conditions, handle partial face occlusion scenarios, and distinguish between family members with similar facial features. The system continuously improves recognition accuracy through ongoing learning algorithms that analyze successful and failed recognition attempts.

The unknown person detection system triggers immediate security protocols when unfamiliar faces are detected, providing users with high-resolution images of detected individuals, confidence scores for recognition accuracy, and comprehensive context information including time, location, and associated security events. This intelligent detection system forms the foundation of the application's proactive security monitoring capabilities.

Family face management includes privacy protection measures that ensure facial recognition data remains secure and encrypted, user consent management for facial recognition usage, and comprehensive data retention policies that respect user privacy while maintaining security effectiveness. The system provides users with complete control over their facial recognition data including export, deletion, and privacy setting modifications.

4.4.10 Comprehensive Settings and Configuration Management

The settings system provides users with granular control over every aspect of their smart home security and automation experience through intuitive configuration interfaces that balance comprehensive functionality with ease of use. The notification settings panel enables users to customize alert preferences with detailed options for security notifications, device status updates, maintenance reminders, and emergency protocols.

Security configuration options include comprehensive camera management settings that control recording quality, storage duration, privacy zones, and motion detection sensitivity. Users can configure facial recognition accuracy thresholds, unknown person detection protocols, emergency contact procedures, and automated response systems that enhance security while minimizing false alerts.

Family management settings provide comprehensive control over facial recognition databases including adding new family members through multiple registration methods, editing existing profiles with updated information and photos, removing outdated or unwanted facial recognition data, and managing privacy settings for facial recognition usage.

Device control settings enable users to customize automation rules, scheduling preferences, energy optimization parameters, and maintenance alert thresholds for each connected device category. The system includes comprehensive preference management for lighting schedules, climate control automation, vacuum cleaning routines, and security device activation protocols.

Theme and interface customization options provide users with extensive personalization capabilities including dark and light mode preferences, color scheme selections, font size adjustments, animation preferences, and accessibility options that ensure optimal user experience across diverse needs and preferences.

Privacy and data management settings offer comprehensive control over personal information sharing, data retention policies, facial recognition consent management, device data access permissions, and comprehensive export options for user data portability and transparency.

4.4.11 Advanced Event Management and Historical Analytics

The event management system provides comprehensive historical analysis capabilities that transform raw security data into actionable insights through sophisticated timeline visualization and intelligent event categorization. The seven-day event history maintains detailed records of all security events, device status changes, user interactions, and system notifications with precise timestamps, event classifications, and associated visual documentation.

Event categorization algorithms intelligently distinguish between security-related events including unknown person detection, motion alerts, and access violations, routine system events such as device status changes and scheduled maintenance, user-initiated events including manual device control and system configuration changes, and automated system events such as energy optimization adjustments and predictive maintenance notifications.

Visual documentation integration automatically captures and stores high-resolution screenshots and video segments associated with security events, providing users with comprehensive visual evidence for security analysis and incident documentation. The system intelligently manages

storage resources while ensuring critical security footage remains accessible for the full seven-day retention period.

Timeline visualization presents event history through intuitive calendar interfaces that enable users to quickly navigate between days, filter events by category and severity, search for specific event types or keywords, and access detailed event information including full-resolution images, contextual information, and related system activities.

Advanced analytics capabilities provide users with insights into security patterns, device usage trends, energy consumption patterns, and system performance metrics that help optimize their smart home configuration while identifying potential security vulnerabilities or system inefficiencies.

Export functionality enables users to generate comprehensive reports for insurance purposes, security auditing, or personal record-keeping, ensuring that critical security information remains accessible beyond the standard retention period.

4.4.12 Data Management, Privacy Protection, and Cloud Integration

The application implements enterprise-grade data management strategies that prioritize user privacy while enabling sophisticated smart home functionality through secure Firebase cloud integration and intelligent local storage optimization. The comprehensive data architecture ensures seamless synchronization between cloud storage and local device storage while maintaining strict privacy controls and data encryption standards.

Firebase integration provides robust cloud infrastructure for secure storage of user profiles, device configurations, facial recognition databases, security event logs, and system preferences while maintaining full compliance with international privacy standards including GDPR and CCPA regulations. The cloud architecture ensures data availability across multiple devices while providing automated backup and disaster recovery capabilities.

Local storage mechanisms ensure critical functionality remains available during network disruptions while sensitive data receives military-grade encryption protection. The system intelligently balances cloud synchronization with local availability to optimize performance, reduce bandwidth usage, and ensure system reliability under various network conditions.

Advanced privacy protection measures include end-to-end encryption for all sensitive communications, secure token-based authentication with automatic refresh capabilities, granular privacy controls that enable users to customize data sharing preferences, and comprehensive audit logs that track all data access and modification activities.

User data protection extends to facial recognition data through secure biometric template storage, encrypted transmission of all security footage, anonymized analytics data collection, and user-

controlled data retention policies that respect individual privacy preferences while maintaining system security effectiveness.

Comprehensive data portability features enable users to export their complete data sets including device configurations, security settings, event histories, and personal preferences, ensuring user control over their information while facilitating easy migration between devices or platforms.

4.4.13 Professional Implementation Excellence and Future Innovation Roadmap

The Leap smart home application successfully establishes a new paradigm in residential security and automation through the seamless integration of cutting-edge mobile development techniques, advanced AI technologies, and comprehensive IoT ecosystem management. The application's sophisticated multi-faceted approach to authentication, intelligent device management, advanced security monitoring, facial recognition, and conversational AI interaction represents a significant breakthrough in smart home control applications that sets new industry standards for functionality, security, and user experience.

The implementation demonstrates exceptional professional excellence through its comprehensive feature integration that addresses real-world challenges in home automation, security monitoring, and intelligent device control. The application's success in balancing sophisticated AI-powered functionality with intuitive user interfaces showcases masterful human-centered design principles that make advanced technology accessible to users across all technical proficiency levels.

The revolutionary combination of advanced facial recognition, real-time security monitoring, intelligent chatbot interaction, and comprehensive device management creates a unified smart home ecosystem that transcends traditional automation limitations. The system's ability to integrate diverse technologies including CNN neural networks, real-time streaming, natural language processing, and Firebase cloud services into a cohesive user experience represents a substantial advancement in mobile application development for IoT environments.

Future development opportunities identified through this comprehensive implementation provide a robust foundation for continued innovation in smart home technologies, ensuring the application remains at the forefront of technological advancement as the IoT ecosystem continues to evolve. The modular architecture and API extensibility ensure seamless integration with emerging technologies while protecting user investment in the platform.

The application's professional-grade implementation, combining sophisticated security features, intelligent automation, intuitive user experience, and robust cloud integration, positions it as a significant contribution to the smart home technology landscape that will influence future development approaches in residential automation and security systems.

Chapter 5

IoT Development

This Chapter presents the hardware and software integration of the Smart Ring system, covering the gesture recognition unit, circuit design, PCB layout, charging module, and smart home actuators. It details the use of the ESP8266-01s and MPU6050 for real-time motion tracking and wireless data transmission. A custom PCB ensures compactness, while the TP4056-based charging unit provides safe recharging. Additionally, smart actuators like switches and door locks are integrated using the same architecture, allowing remote and manual control via Firebase. The section emphasizes modular design, power efficiency, and real-time connectivity across all subsystems.

5.1 Smart Ring

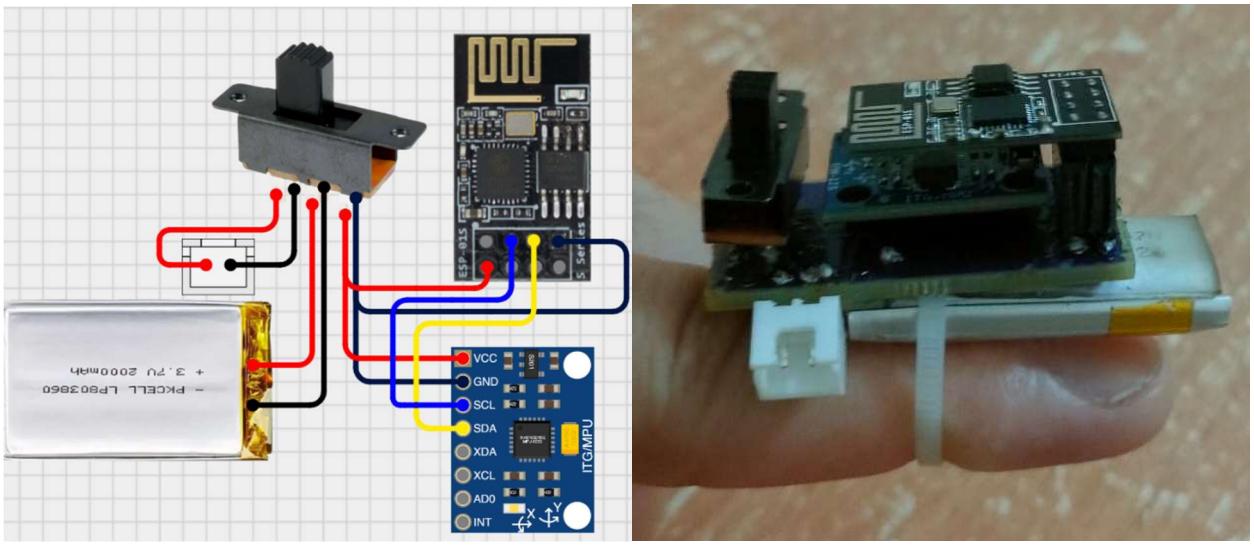


Figure 5.1: Ring circuit

5.1.1 Use Case

The circuit and wiring system forms the physical backbone of the Smart Ring, enabling real-time gesture recognition and wireless communication with the smart home infrastructure. This unit ensures seamless integration between the microcontroller, motion sensor, and power source while maintaining low energy consumption and a compact footprint. The primary use case is to support continuous motion tracking and trigger commands wirelessly, thereby allowing the user to control smart devices through simple hand gestures without physical interaction with switches or screens.

5.1.2 Hardware Components and Justification

The hardware of the Smart Ring consists of four main components:the ESP8266-01s module, the MPU6050 motion sensor, a 3.7V Li-Po battery, and a DPDT slide switch. The ESP8266-01s was selected for its compact size, integrated Wi-Fi capability, and low power consumption, making it ideal for wearable devices with limited space. The MPU6050 combines a 3-axis gyroscope and 3-axis accelerometer, providing precise motion sensing required for gesture recognition. The Li-Po battery supplies lightweight and rechargeable power suitable for extended operation in wearable applications. A DPDT (Double Pole Double Throw) switch is included to control the power routing, allowing the user to manually switch between active and charging states, enhancing safety and usability. Additionally, a JST 2-pin connector is incorporated as a charging interface, providing a compact and secure connection for power input during recharging.

5.1.3 Software

The software component of the Smart Ring is implemented using the Arduino environment with the C++ programming language, and it is deployed on the ESP8266-01s Wi-Fi module. The firmware integrates real-time sensor data acquisition, wireless communication via WebSocket, and IP auto-registration via HTTP to a local Flask server.

Upon startup, the ESP8266 establishes a connection to a predefined Wi-Fi network and automatically sends its local IP address to the Flask server using an HTTP POST request. This enables dynamic device discovery and registration without manual configuration, simplifying integration with the backend infrastructure.

For motion tracking, the MPU6050 sensor is interfaced over the I2C bus using GPIO0 (SCL) and GPIO2 (SDA) of the ESP-01s. The sensor is configured with appropriate ranges for accelerometer and gyroscope readings to capture precise motion patterns. Sensor data is continuously polled and structured as a JSON object containing six features: three-axis acceleration and three-axis angular velocity.

The ESP8266 hosts a local HTTP web server on port 80 that serves a live HTML dashboard, displaying the latest 50 sensor readings in real-time via a WebSocket connection on port 81. This WebSocket stream broadcasts the JSON-encoded sensor data to any connected web client, enabling real-time visualization without the need for refreshing the page.

The software is optimized to run efficiently on the limited resources of the ESP-01s, using lightweight libraries such as ESP8266WiFi, ESP8266WebServer, WebSocketsServer, HTTPClient, and ArduinoJson. This modular software architecture ensures responsive performance, low latency, and seamless interaction with both the user interface and the backend services.

5.1.4 PCB Design and Layout

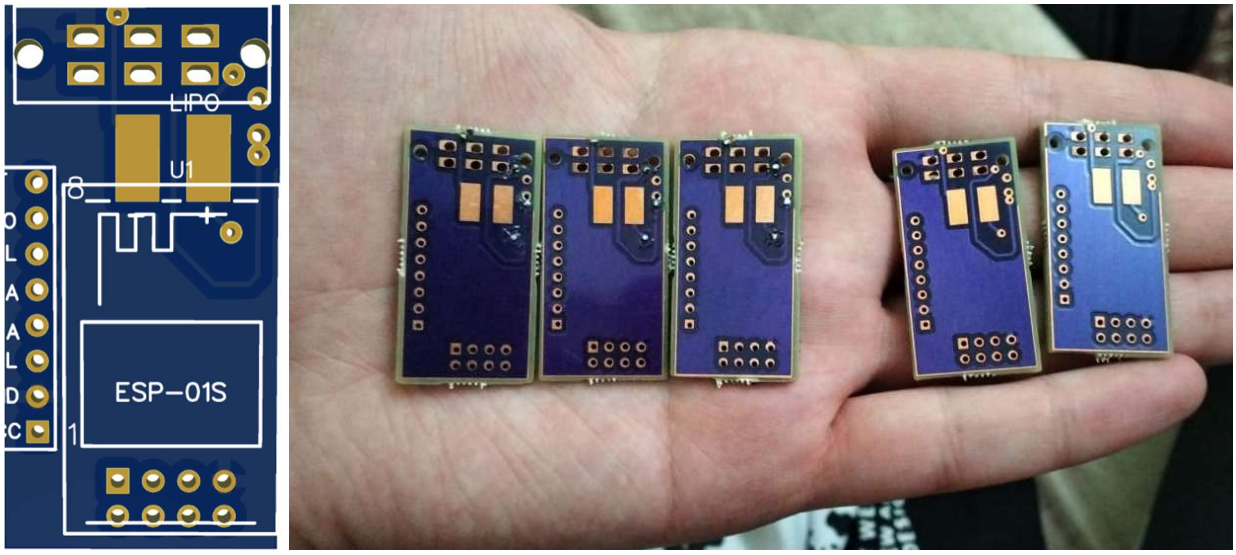


Figure 5.2: Ring PCB

To support compact and reliable deployment, a custom-designed Printed Circuit Board (PCB) was developed. The layout integrates all components—ESP8266-01, MPU6050, Li-Po battery connector, DPDT switch, and JST charging port—within a minimal footprint suitable for wearable applications. The PCB includes designated headers for the ESP8266-01's eight pins (RX, TX, GPIO0, GPIO2, GND, 3V3, CH_PD, RESET) and clear routing for I2C communication lines (SCL and SDA) to the MPU6050 sensor.

The power path is managed through the DPDT switch, which directs the battery's positive and negative terminals either to the system for operation or to the JST connector for charging. This ensures safe and controlled charging behavior. The JST 2-pin connector serves as the primary charging port and is wired directly to the switch, allowing external chargers to power the battery without directly energizing the system.

The PCB's trace design is optimized for current flow and signal integrity, with thicker power lines for battery current and well-isolated signal traces for stable communication. Mounting holes are included to facilitate integration into wearable enclosures. Overall, the PCB design reduces wiring complexity, enhances mechanical reliability, and ensures efficient use of space, contributing to a sleek and robust Smart Ring implementation.

5.2 Charging Unit

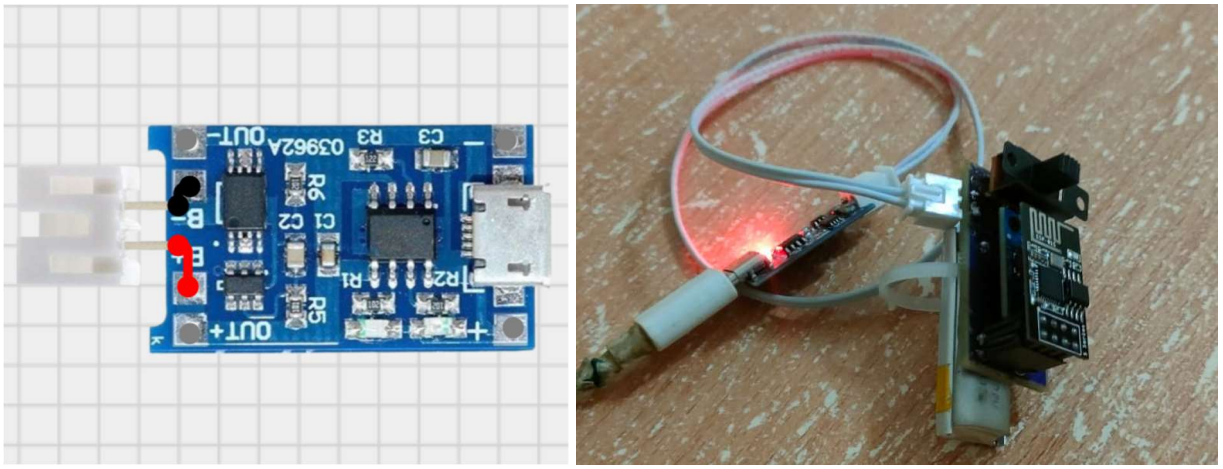


Figure 5.3: Ring Charger

5.2.1 Use Case

The charging unit serves as the external power management system for safely recharging the Smart Ring's internal Li-Po battery. It is designed to connect via a JST 2.54 mm 2-pin connector, allowing easy and secure connection to the Smart Ring. The unit provides a safe, compact, and reusable method of charging using a standard 5V USB input and is based on the TP4056 charging module. This enables the ring to be charged externally without the need to remove the battery, ensuring user convenience and safe energy transfer during recharging cycles.

5.2.2 Hardware Components and Justification

The charging unit is built around the TP4056 Lithium Battery Charging Module, which integrates a linear charger circuit specifically designed for single-cell Li-Ion batteries. The TP4056 was chosen due to its high reliability, integrated overvoltage protection, current regulation, and compact footprint. The module includes clear terminal pads for input (IN+ and IN-), battery connection (B+ and B−), and output (OUT+ and OUT−).

A JST PH 2.54 mm 2-pin female connector is used as the interface for connecting to the Smart Ring. Pin 1 of the JST connector is connected to the B− terminal of the TP4056, while Pin 2 is connected to the B+ terminal. This arrangement allows seamless charging through the same connector used for battery connection in the ring, reducing complexity and enhancing mechanical robustness. The use of USB Micro-B for input power makes the module widely compatible with standard USB chargers.

5.3 Smart Switch

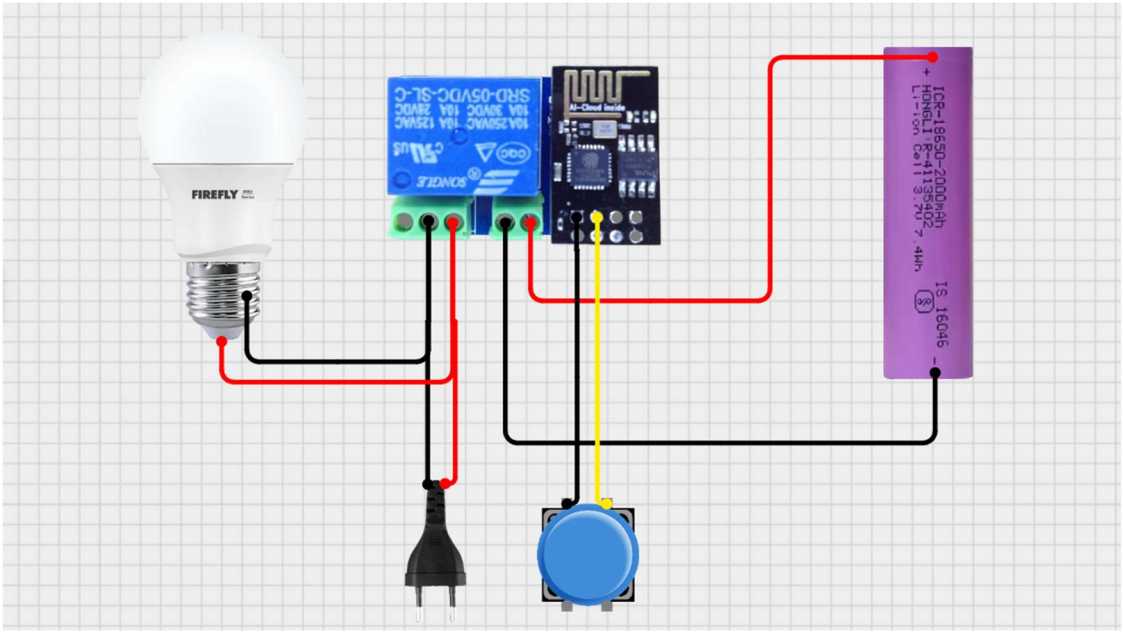


Figure 5.4: Smart Switch circuit

5.3.1 Use Case

The Smart Switch is designed to provide users with flexible control over AC-powered appliances, such as lighting, through multiple interfaces. These include a mobile application, website dashboard, Smart Ring, and manual pushbutton. This multi-channel control ensures ease of use and accessibility across various user scenarios and preferences.

5.3.2 Hardware Components and Justification

The system includes an ESP-01S relay module, a pushbutton, a 3.7V 18650 Li-ion battery, an AC-powered lamp, and an AC plug. The ESP-01S manages connectivity and switching logic. The relay module safely controls the flow of current to the lamp. The pushbutton connected to GPIO2 provides local manual override. The use of a Li-ion battery ensures portability and low-power operation, while the AC plug supplies power to the lamp through the relay.

5.3.3 Software

The firmware for the Smart Switch is developed using the C++ programming language within the Arduino IDE, ensuring efficient and modular code development. For real-time data exchange and remote control, the Firebase ESP Client Library is utilized. This integration enables seamless communication with Firebase, allowing users to monitor and control connected appliances remotely through the mobile application or website dashboard.

5.4 Smart Door Lock

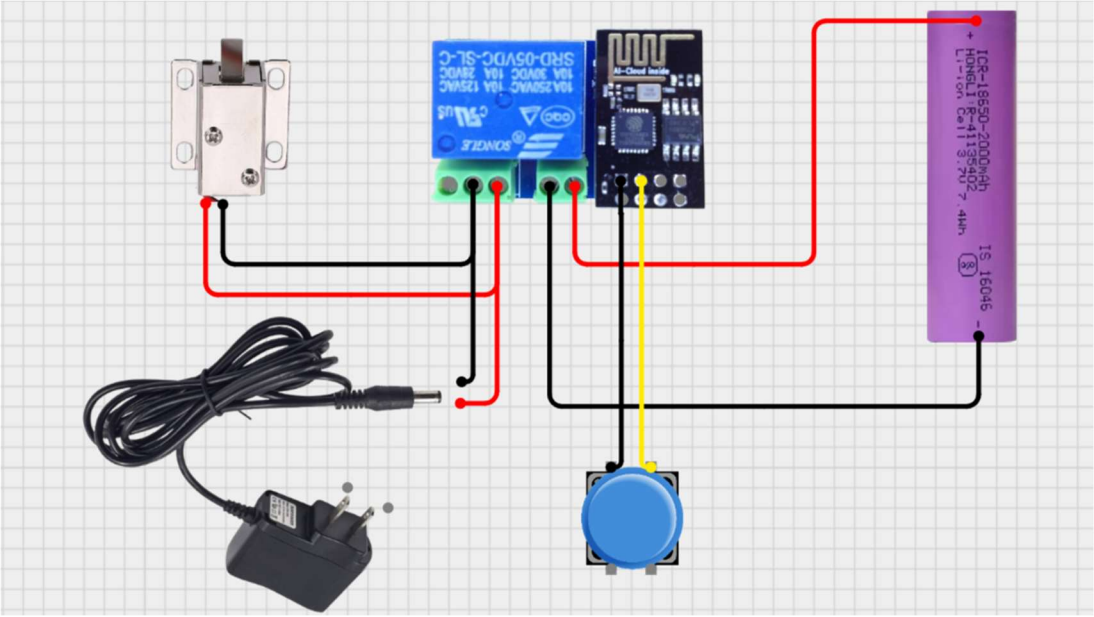


Figure 5.5: Smart Door Lock circuit

5.4.1 Use Case

The smart door lock system is designed to provide secure access control using both manual and wireless inputs. Users can unlock doors via remote commands from connected devices or through a local pushbutton. The system integrates a relay-controlled solenoid lock powered by a 12V adapter, with logic managed by an ESP-01S module. This dual-interface setup ensures flexibility, enabling remote or manual control depending on user needs or network availability.

5.4.2 Hardware Components and Justification

The system consists of an ESP-01S relay module, a 12V solenoid lock, a pushbutton, a 12V DC adapter, and a 3.7V 18650 Li-ion battery. The ESP-01S handles communication and logic control. The relay switches the high-power solenoid lock, while the pushbutton offers manual override. A compact PCB may be used to simplify wiring and integration.

5.4.3 Software

The firmware for the smart door lock is developed using the C++ programming language within the Arduino IDE, ensuring efficient and modular code development. For real-time data exchange and remote control, the Firebase ESP Client Library is utilized. This integration enables seamless communication with Firebase, allowing users to monitor and control door access remotely through the mobile application or website dashboard.

Chapter 6

Experimental Analysis

This chapter presents a detailed analysis of the experimental results obtained during the development and testing phases of the Leap project. It provides insights into the challenges encountered during system evaluation, focusing on the risks identified and the strategies employed to mitigate them. The primary goal of this chapter is to ensure that the Leap system can operate effectively across diverse environments, maintain robust security, and provide a seamless user experience.

6.1 Risk Analysis

The Leap project involves several technical, operational, and security risks that could hinder the overall performance, user experience, and adoption. A thorough risk analysis is critical to identifying potential issues and formulating strategies to address them. In this section, we examine key risks related to system performance, data privacy, and user adoption, and propose strategies to mitigate these risks in subsequent phases of development.

6.1.1 Technical Risks

System Reliability in Diverse Environments

One significant risk to the system's performance is its ability to function accurately in various environments, especially those with fluctuating lighting, background noise, or different user movement patterns. The system's accuracy may degrade in such conditions, leading to poor user experiences. To address this, adaptive machine learning models will be implemented, enabling the system to adjust dynamically to environmental changes. Real-world testing will be conducted in diverse settings to refine the system's performance, ensuring that it can handle different lighting conditions and noise levels. Additionally, the integration of environmental sensors, such as light sensors and microphones, will provide the system with real-time context to adapt its behavior accordingly.

Battery Life and Power Consumption

Another challenge arises from the limited battery life of the wearable components, particularly the smart ring, which may require frequent recharging. To mitigate this risk, the battery management system (BMS) will be further optimized to reduce power consumption during idle and active states. Moreover, energy-efficient hardware components such as low-power inertial measurement units (IMUs) and Bluetooth Low Energy (BLE) will be incorporated to extend battery life. Power-saving modes will also be implemented to reduce energy usage during periods of inactivity, contributing to longer operational times between charges.

Data Latency

Delays in processing and transmitting data could result in significant user experience issues, particularly in applications that require real-time interaction. To reduce latency, local processing capabilities on the smart ring will be enhanced, minimizing dependence on cloud-based systems and thus reducing transmission delays. Additionally, edge computing techniques will be utilized to process data closer to the source, enabling faster decision-making. Communication protocols, such as ESP-NOW and Bluetooth, will be optimized to ensure rapid data transmission between devices, further reducing latency and improving system responsiveness.

Speech Recognition Accuracy

Variations in accents, background noise, and audio quality may reduce speech-to-text accuracy, impacting the user experience. To mitigate this: Fine-tune models for diverse accents and dialects, implement noise reduction techniques to enhance audio clarity, use adaptive algorithms to learn user-specific speech patterns and conduct real-world testing in varied acoustic environments.

6.1.2 Security Risks

Data Privacy

Sensitive user data, including facial recognition embeddings and gesture patterns, is susceptible to unauthorized access, posing significant privacy risks. To safeguard this data, end-to-end encryption will be implemented for all data transmission and storage. This encryption will protect sensitive information from being intercepted or accessed by unauthorized parties. Furthermore, biometric data will be securely stored using secure enclaves or hardware-based security modules, reducing the likelihood of data exposure. Regular updates to the system's security protocols will be essential to address emerging threats and vulnerabilities, ensuring that the system remains secure over time.

Unauthorized Access

Another security concern is the potential for unauthorized users to gain access to the smart home system. To counter this risk, the facial recognition system will be enhanced with liveness detection, which can detect spoofing attempts using photos or videos. In addition, multi-factor authentication (MFA) will be implemented, combining facial recognition with other forms of authentication, such as PIN codes or gesture-based systems, to provide an additional layer of security. An intrusion detection system (IDS) will also be developed to monitor for unusual access patterns and alert users to potential security breaches, ensuring that the system remains secure from unauthorized access.

6.1.3 Usability and Adoption Risks

User Experience and Learning Curve

A major risk to the system's success is the potential for users to find the technology difficult to use, leading to low adoption rates. If the system's interface is unintuitive or complex, users may abandon it before realizing its full potential. To address this, user-centered design workshops will be conducted to gather feedback and refine the system's interface. Additionally, interactive tutorials and onboarding processes will be developed to guide users in learning how to use the system effectively. Adaptive user interfaces will also be implemented, personalizing the experience based on individual preferences and behaviors to ensure a smooth interaction with the system.

Resistance to Adoption

Some users may resist adopting the Leap system, particularly if they perceive it as unnecessary or overly complex. To overcome this resistance, the tangible benefits of the system, such as improved accessibility, convenience, and security, will be emphasized through real-life use cases and testimonials. Offering trial periods or demo versions will allow users to experience the system's capabilities firsthand, helping them understand its value. Furthermore, comprehensive customer support will be available to address any concerns and ensure a smooth transition to using the new technology.

Prompt Engineering Challenges

Prompt engineering is critical to ensuring the relevance and accuracy of AI chatbot responses, as the quality of input directly determines output performance. Challenges arise from ensuring prompts provide sufficient context for understanding, offering clear and unambiguous instructions, and defining response formats to meet user expectations. Poorly designed prompts can lead to irrelevant, incomplete, or incoherent outputs, impacting the chatbot's reliability and user satisfaction. Addressing these issues requires iterative prompt optimization, dynamic context incorporation, and the use of structured templates tailored to domain-specific needs. By mitigating these challenges, the chatbot can deliver precise, relevant, and well-structured responses, enhancing user engagement and functionality.

Chapter 7

Conclusion and Future Work

This chapter provides a detailed conclusion to the Leap project, summarizing the progress made during its initial phase, the challenges encountered, and the lessons learned. It also highlights potential areas for future enhancements to improve the system's functionality, scalability, and user experience. By structuring the chapter in this manner, it seeks to offer a comprehensive understanding of the project's current impact and its possibilities for future advancements.

7.1 Conclusion

The Leap project represents the completion of its first phase, focusing on laying the groundwork for a modular and scalable smart home automation system. This phase has aimed to address key aspects of the system, particularly gesture recognition and facial recognition, while conceptualizing the integration of a conversational AI chatbot. Although the system has not been fully implemented or integrated into web and mobile platforms yet, the progress made in this phase has established a robust foundation for subsequent development.

The development of the facial recognition system has significantly progressed in this phase. Instead of training a custom Siamese network, we adopted a more robust pipeline by integrating YOLOv8 for real-time face detection and InsightFace for face recognition. The Glint360k.onnx model, pretrained on a large-scale face dataset, was utilized to extract high-dimensional facial embeddings, providing superior accuracy and generalization. Real-time image capture was used to register and compare faces, allowing the system to identify house members based on live video input. OpenCV facilitated video stream processing and face preprocessing, ensuring smooth integration between detection and recognition modules. This updated approach has enhanced system performance, simplified deployment, and ensured scalability for real-world use cases.

Bidirectional LSTM model to classify hand motions with **95–98% accuracy**.

The full pipeline includes:

- Capturing motion data from the **MPU6050** via the **ESP8266**
- Streaming over **WebSocket** to the local server
- **Preprocessing + inference** using **FastAPI**
- Updating results to **Firebase**
- Triggering actions in **smart home devices**

Deployment is supported in both:

- **Local mode** (combined_runner) for development
- **Azure hybrid mode** for global accessibility.

The chatbot component, while not yet developed, has been conceptually planned as an integral part of the system. Powered by conversational AI, it is intended to provide users with personalized assistance and seamless interaction with smart home devices. Future phases will focus on implementing and refining this component.

Although the integration of these components into a unified web and mobile platform has not yet occurred, this remains a key objective for the next phase. The aim is to create an intuitive and accessible interface that allows users to control and monitor their smart home devices seamlessly.

7.2 Future Work

As the Leap project advances, several key areas of improvement and expansion will be prioritized to enhance functionality, usability, and scalability.

Gesture Recognition Future Roadmap

The system's architecture enables continuous improvement. Planned enhancements include:

- **UI-based gesture customization**
- **On-device deployment using TensorFlow Lite**
- **Personalized gesture calibration for each user**
- **Real-time alert notifications on unknown gestures or safety events**

For the **facial recognition system**, future improvements will focus on handling varying lighting and crowded environments by integrating sensors like light detectors. The system will support multi-user recognition and introduce liveness detection to prevent spoofing using techniques such as blink detection and depth analysis.

The **chatbot component** will be enhanced with contextual understanding and personalization features. By fine-tuning the model on larger and more diverse datasets, the chatbot will deliver more coherent and relevant responses. Its seamless integration into web and mobile platforms will ensure a unified user experience. New functionalities, such as **Daily Task Scheduling and Reminders** and **Bill Analysis and Spending Reports**, will be developed to expand the system's usability. The latter will utilize OCR technology to extract costs from bill photos, analyze monthly expenses, and generate detailed spending reports with actionable recommendations.

On the **technical front**, the system will be designed for scalability and cross-platform compatibility. A **microservices architecture** and load balancing proxies will be employed to handle increased user demand efficiently. The development of a desktop application will further ensure cross-platform compatibility. Integration with IoT devices, such as smart armbands and rings, will be optimized using Firebase's real-time database, while lightweight machine learning models will be deployed on edge devices to enhance real-time performance and energy efficiency.

Finally, **user experience and accessibility** will be improved through the integration of voice control capabilities, haptic feedback, and voice-guided navigation. These features will ensure the system remains inclusive and practical for a diverse range of users, while future enhancements, such as additional IoT device integrations and advanced AI functionalities, will further solidify the system's position as a versatile and user-centric solution.

Backend Testing and Quality Assurance To ensure the reliability and robustness of the system, a comprehensive backend testing strategy will be implemented:

- **Unit Testing Framework**
  1. Implement automated unit tests for individual functions and modules
  2. Focus on testing authentication logic, API route handlers, and database operations
  3. Use testing frameworks like Jest or Mocha for JavaScript/Node.js components
  4. Establish continuous integration to run tests automatically on code changes
- **Integration Testing**
  1. Develop end-to-end tests for complete user flowsTest interactions between different backend components (API, database, external services)
  2. Implement automated API testing using tools like Postman or Supertest
  3. Create test environments that mirror production settings
- **Performance Testing**
  1. Conduct load testing to ensure system stability under heavy usage
  2. Implement stress testing to identify system breaking points
  3. Monitor and optimize database query performance
  4. Test real-time data synchronization with Firebase

Security Testing

Implement automated security testing for authentication and authorization

Conduct penetration testing for API endpoints

Test data encryption and secure storage practices

Regular security audits and vulnerability assessments

7.3 Broader Implications

The Leap project's technologies have potential applications beyond smart home automation. Gesture recognition could be employed in healthcare to control medical devices or assist patients with mobility impairments. Facial recognition systems could enhance security and efficiency in industrial settings by controlling access to sensitive areas. The chatbot component could serve as a virtual teaching assistant in education, providing personalized learning experiences for students. These broader implications highlight the versatility and potential impact of the technologies developed in the Leap project.

7.4 Final Thoughts

As smart home automation continues to evolve, the Leap project's first phase serves as a foundation for future advancements in human-computer interaction and IoT technologies. The insights gained and challenges addressed during this phase will guide subsequent development efforts, paving the way for a more innovative, accessible, and secure smart home solution. By addressing current limitations and leveraging emerging opportunities, the Leap project aims to redefine smart home standards and enhance the quality of life for its users.

Chapter 8

Tools and References

8.1 Tools

Gesture Recognition

Machine Learning & Deep Learning

- **TensorFlow / Keras:** Used to build, train, and export the Bidirectional LSTM model (.h5 format).
  https://www.tensorflow.org/
- **Scikit-learn:** Employed for classical models (SVM, Random Forest, k-NN), feature extraction, normalization, and label encoding.
  https://scikit-learn.org/
- **NumPy & Pandas:** Essential for numeric operations and dataset preprocessing.
  https://numpy.org/ | https://pandas.pydata.org/

Microcontroller Communication & Streaming

- **ESP8266 with Arduino IDE:** Used to stream real-time IMU data (MPU6050) over WebSocket.
  https://www.arduino.cc/en/software
- **MPU6050 Library (I2Cdevlib):** To read acceleration and gyroscopic data from the IMU sensor.
  https://github.com/jrowberg/i2cdevlib

Server & Backend Technologies

- **FastAPI:** Python framework to serve the trained BiLSTM model via REST API.
  https://fastapi.tiangolo.com/
- **WebSocket Server (Python / asyncio):** Real-time stream handler that receives IMU data from the ESP8266.
  https://websockets.readthedocs.io/
- **Flask:**Used on Azure to read predictions and deliver frontend responses.
  https://flask.palletsprojects.com/

Cloud & Deployment

- **Microsoft Azure Web App:** Hosts the public FastAPI endpoint (/api/latest) for remote predictions.
  https://azure.microsoft.com/en-us/products/app-service
- **Firebase Realtime Database:** Used to sync predicted gesture actions (e.g., toggle lights or open door) with smart home UIs.
  https://firebase.google.com/products/realtime-database

Facial Recognition Tools

- **OpenCV**: Used for real-time image capture, face preprocessing (cropping, resizing, normalization), and visualization.

- **YOLOv8**: Employed for fast and accurate face detection in both image and live video streams, enabling real-time response in dynamic environments.
- **InsightFace (glint360k.onnx model)**: Provides deep face embeddings using a pretrained model trained on the VGGFace2 dataset to ensure accurate identity representation.
- **ONNX Runtime**: Used to run YOLOv8 and InsightFace models efficiently across different platforms with hardware acceleration support.
- **NumPy/ scikit-learn**: Utilized for calculating cosine similarity between facial embeddings to determine identity matches.

Chatbot Development Tools

- **Groq Cloud API (OpenAI-compatible) :** https://www.groq.com

  Groq Cloud provides high-speed, low-latency access to the **LLaMA 3** language model using an OpenAI-compatible API. It enables real-time inference for intent classification and chatbot response generation.

- **Hugging Face Sentence Transformers:** https://huggingface.co/sentence-transformers

  Used to generate vector embeddings (e.g., via all-MiniLM-L6-v2) from recipe documents. These embeddings are stored in Chroma for fast and accurate retrieval in RAG-based responses.

- **LangChain:** https://www.langchain.com
  LangChain is a powerful framework used for building context-aware chatbots. It manages prompt templates, conversational memory, and chaining logic, enabling structured interaction with large language models.

- **Flask:** https://flask.palletsprojects.com
  A lightweight web framework used to serve the chatbot's backend. Flask handles HTTP requests from the front-end, routes user input to the appropriate chatbot module, and returns the generated response.
- **Firebase Realtime Database:** https://firebase.google.com/products/realtime-database
  Used for real-time communication with smart home devices. The chatbot updates and reads device status (e.g., lamp on/off) instantly, enabling responsive smart environment control.
- **Microsoft Azure (Cloud Deployment Platform):** https://azure.microsoft.com
  Microsoft Azure is used to deploy the chatbot application in a scalable, production-ready environment. Through **Azure App Service**, the Flask backend is hosted with continuous deployment via GitHub. Azure ensures high availability, secure API access, and integration with services such as App Configuration and Azure Monitor for logging and diagnostics.

Speech Recognition

- **SpeechRecognition (sr):** https://pypi.org/project/SpeechRecognition/2.1.3/

  This module is used to convert audio speech into text. It enables speech-to-text functionality in our project, which is valuable for applications like voice-controlled assistants, transcription services and enhancing the accessibility of our system for users with disabilities.

- **gTTS (Google Text-to-Speech):** https://pypi.org/project/gTTS/

  Used to synthesize natural-sounding English speech from text responses. It enhances the accessibility and interactivity of the chatbot by enabling voice-based output.

8.2 References

Govindarajulu, Y., & Kumar, R.R.R. (2023).
Gesture Recognition based on Long-Short Term Memory Cells using Smartphone IMUs. arXiv preprint. arXiv

Toro-Ossaba, A., Jaramillo-Tigreros, J., Tejada, J.C., Peña, A., López-González, A., & Castanho, R.A. (2022).
LSTM Recurrent Neural Network for Hand Gesture Recognition Using EMG Signals. Applied Sciences, 12(19), 9700. MDPI

Tai, T.-M., Jhang, Y.-J., Liao, Z.-W., Teng, K.-C., & Hwang, W.-J. (2020).
Sensor-Based Continuous Hand Gesture Recognition by Long Short-Term Memory. arXiv preprint. arXiv

M Kim, J Cho, S Lee, Y Jung. (2019). IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces. [MPDI] [Google scholar]
Mouser Electronics Blog (2023). IMU Basics. [Mouser].
Pololu Robotics and Electronics (2023). 6-DOF IMU Data Guide. [Pololu].
P Senin (2008). Dynamic Time Warping Algorithm Review. [ResearchGate] [Google Scholar].
Asif Rahim, **Yanru Zhong**, Enhancing Smart Home Security: Anomaly Detection and Face Recognition in Smart Home IoT Devices Using Logit-Boosted CNN Models (2023) [https://pmc.ncbi.nlm.nih.gov/articles/PMC10422449/]
D. Anandan. (2024), Enhanced Approach of Face Recognition in Home Security Using Deep Learning Techniques [ResearchGate]
M. Marimuthu, G. Mohanraj (2025). Facial Recognition Enabled Smart Security Lock System Using Machine Learning Approach [https://publications.eai.eu/index.php/IoT/article/view/5657]
Nazar EL Fadel, (2025), Facial Recognition Algorithms: A Systematic Literature Review [https://www.mdpi.com/2313-433X/11/2/58]
Yiming Yao, (2024), Research on facial recognition system based on deep learning [ResearchGate]
Ka-Ho Chow, Sihao Hu (2024), Diversity-driven Privacy Protection Masks Against Unauthorized Face Recognition [https://petsymposium.org/popets/2024/popets-2024-0122.pdf]
Ka-Ho Chow, Sihao Hu, (2024), Personalized Privacy Protection Mask Against Unauthorized Facial Recognition [https://www.ecva.net/papers/eccv_2024/papers_ECCV/papers/10846.pdf]
An et al, (2024), Glint360K [https://paperswithcode.com/dataset/glint360k]
DeepInsight, insightface [GitHub]

Asif Rahim, Yanru Zhong, Tariq Ahmad, Sadique Ahmad (2023), Enhancing Smart Home Security: Anomaly Detection and Face Recognition in Smart Home IoT Devices Using Logit-Boosted CNN Models [https://pmc.ncbi.nlm.nih.gov/articles/PMC10422449/]

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Jegou, H. (2024). LLaMA 3: Open and Efficient Foundation Language Models. Meta AI Research. https://ai.meta.com/blog/meta-llama-3/

Groq Inc. (2024). Groq LPU Inference Performance for LLaMA 3. Retrieved from https://www.groq.com/llama3

Google Developers. (2024). Web Speech API Documentation. Voice Driven Web App

D. Sebastián Cabezas, R. Fonseca-Delgado. (2024). Integrating a LLaMa-based Chatbot with Augmented Retrieval Generation. [ResearchGate].

Sumit Kumar Dam, Choong Seon Hong. (2024). A Complete Survey on LLM-based AI Chatbots. [ResearchGate].

Stanford. Agile Development Methodology [Article].