# Report Title

Dawoud Tormos  
Home Control System  
Lebanese International University  
1/3/2025

Table of Contents

[Report Title 1](#_Toc187317743)

[Dedication 2](#_Toc187317744)

[Acknowledgment 3](#_Toc187317745)

[Abstract 6](#_Toc187317746)

[Table of Contents 7](#_Toc187317747)

[Chapter 1: Introduction 8](#_Toc187317748)

[1.1 Introduction 8](#_Toc187317749)

[1.2 Objective 9](#_Toc187317751)

[1.3 Scope 9](#_Toc187317753)

[1.4 Technology Constraints 11](#_Toc187317783)

[1.5 Problem 11](#_Toc187317787)

[1.6 Solution 12](#_Toc187317788)

[Chapter 2: Requirements and Analysis 13](#_Toc187317789)

[2.1 Functional Requirements 13](#_Toc187317790)

[1. User Authentication: 13](#_Toc187317791)

[2. Device Management: 13](#_Toc187317794)

[3. Real-Time Device Control: 13](#_Toc187317798)

[4. Automation & Scheduling: 13](#_Toc187317801)

[5. Sensor Monitoring: 14](#_Toc187317804)

[6. Notifications: 14](#_Toc187317807)

[7. Device Status Feedback: 14](#_Toc187317810)

[2.2 Non-Functional Requirements 15](#_Toc187317813)

[2.3 UML Use Case Diagram 16](#_Toc187317822)

[16](#_Toc187317823)

[2.4 Use Case Scenarios 17](#_Toc187317824)

[Chapter 3: System Design 32](#_Toc187317825)

[3.1 ER Diagram 32](#_Toc187317826)

[3.2 Relational Diagram 32](#_Toc187317827)

[Chapter 4: Implementation 33](#_Toc187317828)

[DashBoard Page 38](#_Toc187317829)

[AI Assitant Page 39](#_Toc187317830)

[Add Device Pages 42](#_Toc187317831)

[42](#_Toc187317832)

[44](#_Toc187317833)

[45](#_Toc187317834)

[Profile Page 51](#_Toc187317835)

[51](#_Toc187317836)

[Chapter 5: Conclusion and Future Work 52](#_Toc187317837)

[5.1 Conclusion 52](#_Toc187317838)

[5.2 Future Work 52](#_Toc187317842)

[References 54](#_Toc187317843)

# Dedication

I was mostly inspired by other similar platforms like Alexa and google home.

# Acknowledgment

We would like to express our heartfelt gratitude to the individuals, organizations, and communities whose contributions and support made this project possible. In particular, we acknowledge the following:

**Core Frameworks and Technologies**

**Go:** For providing a fast, efficient, and robust programming environment for backend development. Its excellent performance stems from being a compiled language, producing highly optimized binaries with minimal runtime overhead.

Go's built-in support for asynchronous programming and multi-threading, through goroutines and channels, enables scalable and efficient handling of concurrent tasks, making it ideal for high-performance applications like this platform.

**sqlc**: For generating type-safe Go code from SQL queries, simplifying our database interactions.

**Gin Web Framework:** For delivering a lightweight and powerful toolkit for building our web APIs.

**PostgreSQL**: For serving as our reliable and feature-rich database system.

**pgAdmin 4**: For offering an intuitive interface to manage and maintain our PostgreSQL database.

**Flutter**: For enabling us to create beautiful, high-performance cross-platform applications effortlessly.

**JWT Authentication:** For enabling secure and efficient token-based user authentication across the platform.

**Bcrypt hashing (with salting):** For ensuring robust password security through reliable hashing and salting mechanisms.

**Flutter Dependencies**

Special thanks to the developers and maintainers of the following Flutter packages, which played a critical role in enhancing the functionality and user experience of our application:

**cupertino\_icons:** For providing iOS-style icons.

**persistent\_bottom\_nav\_bar:** For implementing consistent navigation in our application.

**speech\_to\_text:** For enabling speech recognition capabilities.

**permission\_handler:** For managing runtime permissions across platforms.

**mobile\_scanner:** For adding robust barcode scanning functionality.

**image\_picker:** For facilitating image selection and capture.

**flutter\_secure\_storage:** For secure data storage within the app.

**provider**: For efficient state management.

**web\_socket\_channel:** For seamless real-time communication via WebSocket.

**Special Thanks**

**Ollama**: For providing invaluable insights and tools that enhanced our project development.

**LLaMA 3.3 from Meta:** For offering state-of-the-art language modeling capabilities that inspired and supported key components of the project.

**Additional Contributions**

**Open Source Communities:** For their dedication to creating and maintaining these invaluable tools and libraries.

**Testers and Early Users:** For their invaluable feedback, which helped refine the application and improve its usability.

Your support and contributions have been instrumental to the success of this project, and we are deeply thankful for your efforts.

# Abstract

This project is an interactive platform inspired by smart home ecosystems like Alexa, designed to manage and control IoT devices through a unified interface. Built with a robust backend powered by Go, sqlc, Gin, PostgreSQL, and pgAdmin 4, the platform seamlessly integrates with a variety of smart devices, including cameras, sensors (temperature, power, motion), and switches (dimmer, ON/OFF). The frontend, developed using Flutter, provides an intuitive user experience across mobile and web interfaces.

Key features include user authentication, device connectivity, voice and GUI-based commands, and real-time device management. Users can perform tasks such as turning devices on or off, retrieving sensor data, configuring settings, managing schedules, and viewing camera streams or snapshots. Voice commands are processed using advanced AI (powered by LLaMA 3.3 from Meta), providing a natural interaction similar to Alexa

The platform ensures a seamless user experience by facilitating device linking through QR codes or manual input, offering push notifications for critical updates, and maintaining periodic logging of sensor data. Security and scalability are prioritized, enabling users to manage multiple devices with personalized schedules and preferences.

The system also incorporates predictive capabilities for optimizing future interactions based on logged commands and device usage patterns, ensuring a smarter and more efficient smart home experience. This project leverages the power of modern technologies and frameworks to deliver a highly interactive, reliable, and scalable IoT solution.

# Chapter 1: Introduction

## 1.1 Introduction

## This project aims to create an intelligent platform for managing and controlling various IoT devices in a smart home environment. By integrating modern technologies, the system provides users with seamless control over devices through a mobile app, web interface, and voice commands, offering a user-friendly and secure experience.

## 1.2 Objective

## This project is a comprehensive smart home platform designed to streamline the management and control of IoT devices. By leveraging cutting-edge technologies, the platform enables users to interact with devices such as cameras, sensors, and switches through voice commands, a user-friendly mobile app, and a web interface. With a focus on performance, security, and scalability, the system integrates advanced AI, secure authentication, and real-time device connectivity to provide an efficient and interactive smart home experience.

## 1.3 Scope

## This project aims to develop a smart home platform for controlling IoT devices such as cameras, sensors (temperature, power, motion), and switches (dimmer, ON/OFF). The platform will offer an interactive user interface for mobile and web applications, enabling seamless control of devices through both voice commands and direct GUI actions.

## Features Included:

## User authentication (sign in and sign up) for account access.

## Device management, including linking devices to user accounts and connecting them to the internet.

## Real-time control of devices such as cameras, sensors, and switches.

## Voice command integration for controlling devices using AI-powered instructions.

## Real-time sensor data viewing (e.g., temperature, power, motion).

## Camera viewing (live video streaming or snapshots).

## Push notifications for critical updates or device status changes.

## Scheduling and configuring device states (e.g., switches turning ON/OFF based on a schedule).

## Secure user data management using JWT authentication and bcrypt hashing with salt.

## Technologies Used:

## Backend: Go, Gin, PostgreSQL, sqlc

## Frontend: Flutter (for mobile and web interfaces)

**API:** Rest Api using Json , WebSocket

## Database: PostgreSQL

## Database Management Tool: pgAdmin 4

## AI: LLaMA 3.3 for voice command processing

## Database JWT Authentication, bcrypt hashing with salt

## Target Audience:

## Homeowners looking for an efficient and secure way to control smart home devices.

## Businesses or enterprises with IoT device management needs.

## Limitations:

## Limited device compatibility based on supported models and standards.

## The platform requires a stable internet connection for device connectivity and remote control.

## AI capabilities may be limited in terms of voice command recognition and response accuracy.

## Exclusions:

## Support for non-IoT devices.

## Integration with external third-party smart home ecosystems (e.g., Amazon Alexa, Google Home).

## This platform aims to provide a comprehensive solution for managing and interacting with IoT devices while ensuring a secure, reliable, and user-friendly experience.

## 1.4 Technology Constraints

## At the outset, the project was initially constrained by hardware limitations, particularly in developing a robust language model for AI-driven voice commands. The lack of sufficient computational power and resources made it challenging to implement an in-house language model capable of understanding and processing complex commands

## However, this challenge was overcome through the integration of LLaMA 3.3 via Ollama, which provided access to a powerful inference platform. Ollama's platform leverages C++-based code to efficiently utilize both CPU and GPU resources, making it possible to process language models at scale. By combining VRAM (Video RAM) and RAM (system memory), Ollama's platform enabled the project to run advanced language models with improved performance and responsiveness, even on limited hardware.

## This integration allowed the project to bypass hardware constraints and utilize state-of-the-art AI capabilities for voice command processing without the need for extensive in-house model development. The ability to run LLaMA 3.3 on Ollama's platform also provided the flexibility to scale the language model inference, making it more suitable for real-time applications like smart home control.

## 1.5 Problem

One of the main challenges in creating a smart home system is developing an intuitive and reliable method for controlling various devices within a home. As homes become increasingly connected with IoT devices—such as cameras, lights, sensors, and appliances—the need for a centralized platform that can efficiently manage and automate these devices becomes essential. However, achieving this requires overcoming several obstacles, such as:

1. **Device Compatibility**: Integrating devices from different manufacturers, each with its own protocols and communication standards, can be complex and time-consuming.
2. **User Experience**: Ensuring that users can easily interact with and control devices from a variety of interfaces, including mobile apps, web interfaces, and voice commands, without facing a steep learning curve.
3. **Security**: Safeguarding user data and ensuring secure communication between devices to prevent unauthorized access.
4. **Automation Logic**: Implementing smart automation that can respond to user preferences or environmental changes without requiring constant manual input, ensuring seamless and efficient device control.
5. **Scalability**: Allowing for the addition of new devices and functionalities over time without significant redevelopment of the platform.

## 1.6 Solution

The solution our project offers to these challenges involves building an integrated platform that supports various IoT devices and allows users to control and automate their home environment through multiple interfaces. Key components of the solution include:

1. **Unified Device Management**: Developing a central system capable of linking and managing different types of IoT devices, ensuring compatibility through a common communication protocol, and providing an easy way to add new devices as they become available.
2. **User-Friendly Interface**: Creating mobile and web applications with intuitive and simple controls for device management, as well as incorporating voice command capabilities for hands-free interaction.
3. **Secure Authentication**: Implementing robust security measures, such as JWT authentication and bcrypt hashing with salt, to protect user data and ensure that only authorized individuals can control devices.
4. **Smart Automation**: Using sensor data and user preferences to create automation rules (e.g., turning on lights when motion is detected or adjusting the thermostat based on the time of day), making the system responsive and autonomous.
5. **Scalable Architecture**: Designing the platform to be scalable and flexible, allowing for the easy integration of new devices and features over time, and ensuring that the system can grow as the smart home ecosystem evolves.

This approach ensures a comprehensive, user-friendly, and secure solution for controlling and automating smart home devices, offering users a seamless and efficient way to manage their home environment.

# Chapter 2: Requirements and Analysis

## 2.1 Functional Requirements

## 1. User Authentication:

## The system should allow users to sign up, sign in, and manage their accounts using secure authentication methods (e.g., email/password, JWT authentication).

## The system should support password recovery and two-factor authentication for enhanced security.

## 2. Device Management:

## The platform should allow users to add, remove, and manage IoT devices (e.g., cameras, sensors, switches).

## Devices should be linked to user accounts and must be able to connect to the internet using WiFi.

## The platform should support both manual entry and QR code scanning for device linking.

## 3. Real-Time Device Control:

## Users should be able to control devices (e.g., turning switches on/off, viewing camera footage, adjusting sensor settings) via mobile or web interfaces.

## Voice commands should be supported for controlling devices, processed by an AI model integrated into the system.

## 4. Automation & Scheduling:

## The platform should allow users to create schedules for devices (e.g., turning a switch on/off at specific times).

## The system should support automation based on sensor inputs (e.g., motion detection triggering lights or security cameras).

## 5. Sensor Monitoring:

## The system should provide real-time sensor data (e.g., temperature, power, motion) to users through the app or web interface.

## The data should be updated and displayed in an easy-to-read format.

## 6. Notifications:

## The platform should send notifications to users for critical events (e.g., motion detected, device status changes, system errors).

## Notifications should be sent via push notifications to mobile devices.

## 7. Device Status Feedback:

## The system should provide feedback on the current status of devices (e.g., whether a camera is streaming, a switch is on/off, or a sensor is active).

## Users should receive real-time updates on their devices' state.

## 2.2 Non-Functional Requirements

## 1. Performance The system should provide quick responses for user commands, with a maximum response time of 3 seconds for device control actions. Real-time data updates (e.g., sensor readings, device status) should be reflected immediately on user interfaces.

## 2. Scalability The system should be able to handle the addition of new devices and users without significant performance degradation. The architecture should support future expansion, such as the addition of more device types, sensors, or automation rules.

## 3. Availability The platform should be available 99.9% of the time, with minimal downtime for maintenance. The system should be able to handle high numbers of concurrent users, especially during peak times.

## 4. Security User data should be encrypted both in transit (using HTTPS) and at rest (using secure storage). Access to devices and user data should be restricted based on user roles, ensuring that only authorized users can control devices and modify settings. The platform should implement protection against common security vulnerabilities (e.g., SQL injection, cross-site scripting).

## 5. Usability The mobile and web interfaces should be user-friendly, with clear instructions and intuitive navigation. Voice command functionality should be easy to use, with clear feedback to users for successful or failed commands.

## 6. Maintainability The codebase should be well-documented and modular, allowing for easy updates and feature additions. The system should allow for remote diagnostics and debugging in case of errors or issues.

## 7. Interoperability The platform should be able to integrate with a wide range of IoT devices, following standard protocols (e.g., MQTT, HTTP, RESTful APIs). The system should be able to work across different platforms (mobile and web) and support major operating systems (iOS, Android, Windows, macOS).

## 8. Data Integrity The platform should ensure that all device data and user information are accurately captured and stored in the database. Any changes made to device settings, schedules, or user preferences should be consistently reflected across all user interfaces.

## 2.3 UML Use Case Diagram

## 

## 2.4 Use Case Scenarios

|  |
| --- |
| Use Case Name: Sign in |
| **Actors:** User |
| **Brief Description:**  User enters his account’s credentials to enter his account. |
| **Basic Flow:**   1. Enters password and username 2. Credentials are validated 3. If valid the user is logged in to his account |
| **Alternative Flow:**  **The credentials are invalid.** A error message is shown. |
| **Pre-conditions:**  Connected to internet |
| **Post-conditions:**  The user is logged in  He has now the GUI opened |

|  |
| --- |
| Use Case Name: Sign up |
| **Actors:** User |
| **Brief Description:**  User enters the required input fields. |
| **Basic Flow:**   1. Enters input fields 2. Data entered are checked and validated. 3. If valid the user is logged in to his new account |
| **Alternative Flow:**  **The data entered are invalid.** A error message is shown. |
| **Pre-conditions:**  Connected to internet |
| **Post-conditions:**  The user is logged in  He has now the GUI opened |

|  |
| --- |
| Use Case Name: Connect a device to the internet |
| **Actors:** User, Device |
| **Brief Description:**  The User boot the device in the hotspot/configuration mode  Hotspot is to have a direct connection between mobile and the device to give it the WiFi's credentials. |
| **Basic Flow:**   1. User boot the device in the hotspot/configuration mode 2. The device is now mini server and an access point of close range 3. A simple webpage is present 4. Through the page the user can enter WIFI credentials that the device will use to access internet |
| **Alternative Flow:**  **None** |
| **Pre-conditions:**  Having the device |
| **Post-conditions:**  The device has internet access now |

|  |
| --- |
| Use Case Name: Link a device to his account |
| **Actors:** User, Device |
| **Brief Description:**  User scans a secret QR code off the device if using mobile app **or** manually enters id and passcode of the device. |
| **Basic Flow:**   1. User scans or enters id and passcode of the device. 2. Now your account has the id and passcode of the device. |
| **Alternative Flow:**  **The QR code are invalid.** A error message is shown. |
| **Pre-conditions:**  Having the device near you and already with internet access. |
| **Post-conditions:**  You have access to the device you linked to your account through mobile app or web interface. |

|  |
| --- |
| Use Case Name: Voice Command through AI |
| **Actors:** User, Mobile App |
| **Brief Description:**  User sends a voice command in a manner similar to the well known Alexa. |
| **Basic Flow:**   1. Voice command made 2. AI (LLM) see if an instruction is valid for the command and responds through text as well 3. Server deals with the instruction if present |
| **Alternative Flow:**  **The Command are invalid.** AI (LLM) responds accordingly |
| **Pre-conditions:**   * Connected to internet and logged in * Having device(s) linked to send commands |
| **Post-conditions:**  Command performed  \*Command is logged for later predictions |

|  |
| --- |
| Use Case Name: Direct Command through GUI |
| **Actors:** User, Mobile App, Web interface |
| **Brief Description:**  User does a command through GUI |
| **Basic Flow:**   1. **Command issued** 2. **Server deals with the command by sending/updating the required data** 3. **GUI receives responds the presents the data or response needed** |
| **Alternative Flow:**  **The Command are invalid.** GUI responds accordingly |
| **Pre-conditions:**   * Connected to internet and logged in * Having device(s) linked to send commands |
| **Post-conditions:**  Command performed  \*Command is logged for later predictions |

|  |
| --- |
| Use Case Name: See sensor measurement |
| **Actors:** User, Mobile App or Web Interface ,Server |
| **Brief Description:**  The user sees the sensor’s measurement on his phone or web interface. |
| **Basic Flow:**   1. Voice Command or direct GUI Command 2. Command done 3. Sensor measurement is sent back. |
| **Alternative Flow:**  There is a problem in the sensor or server  Response is invalid or an error message is shown |
| **Pre-conditions:**  Connected to internet  Logged in  A sensor is linked to the account  Command sent |
| **Post-conditions:**  sensor measurement presented to user  \*Command is logged for later predictions |

|  |
| --- |
| Use Case Name: See camera shot or to video stream |
| **Actors:** User, Mobile App or Web Interface ,Server |
| **Brief Description:**  The user sends a command to see a camera shot or a camera video stream. |
| **Basic Flow:**   1. Voice Command or direct GUI Command 2. Command done 3. Camera shot or video stream is sent back. |
| **Alternative Flow:**  There is a problem in the Camera or server  Response is invalid or an error message is shown |
| **Pre-conditions:**  Connected to internet  Logged in  A Camera is linked to the account  Command sent |
| **Post-conditions:**  Camera shot or video stream is presented  \*Command is logged for later predictions |

|  |
| --- |
| Use Case Name: Turn ON/OFF a switch |
| **Actors:** User, Mobile App or Web Interface ,Server |
| **Brief Description:**  The user changes a switch state. |
| **Basic Flow:**   1. Voice Command or direct GUI Command 2. Command done 3. Response of success is sent back. |
| **Alternative Flow:**  There is a problem in the sensor  Response is invalid or an error message is shown |
| **Pre-conditions:**  Connected to internet  Logged in  A switch is linked to the account  Command sent |
| **Post-conditions:**  switch state changed  \*Command is logged for later predictions |

|  |
| --- |
| Use Case Name: Configure Settings |
| **Actors:** User, Mobile App or Web Interface ,Server |
| **Brief Description:**  User edits in settings. |
| **Basic Flow:**   1. Voice Command or direct GUI Command 2. Command done 3. Setting changed. |
| **Alternative Flow:**  There is a problem in the interface used, the server or internet connection  Response is invalid or an error message is shown |
| **Pre-conditions:**  Connected to internet  Logged in  A switch is linked to the account  Command sent |
| **Post-conditions:**  Settings changed and the effect of the change is show  \*Command is logged for later predictions |

|  |
| --- |
| Use Case Name: Edit/add schedule |
| **Actors:** User, Mobile App or Web Interface ,Server |
| **Brief Description:**  User edits a schedule that is typically linked to a switch. The schedule is when a switch is ON or OFF |
| **Basic Flow:**   1. Voice Command or direct GUI Command 2. Command done 3. Schedule added or edited . |
| **Alternative Flow:**  There is a problem in the Switch or server  Schedule not changed  Response is invalid or an error message is shown |
| **Pre-conditions:**  Connected to internet  Logged in  A switch is linked to the account  Command sent |
| **Post-conditions:**  Schedule added or edited .  \*Command is logged for later predictions |

|  |
| --- |
| Use Case Name: Send Notifications |
| **Actors:** User, Mobile App ,Server |
| **Brief Description:**  The server have a data change, an alarm to show or anything that requires the user to be alerted. A push notification is sent to the mobile app. |
| **Basic Flow:**   1. Servers sees a change or data that requires the user to be alerted 2. Server sends a push notification to the mobile app. |
| **Alternative Flow:**  Mobile with no internet access  The mobile app cannot see a new notification |
| **Pre-conditions:**  Connected to internet  Logged in  Notification present |
| **Post-conditions:**  Users is now alert to the change or the urgent data  \*Change is logged for later predictions |

|  |
| --- |
| Use Case Name: Sensor Logs its measurement |
| **Actors:** Device(sensor), Server |
| **Brief Description:**  The device periodically logs its measurement to the database |
| **Basic Flow:**   1. Device checks for the passing a set period of time 2. Device send a measurement record to the server 3. Server adds it to the database |
| **Alternative Flow:**  There is a problem in the sensor  No periodic logging is happening  In case of motion sensor:  It logs only the time instance of motion detection |
| **Pre-conditions:**  Connected to internet  Linked to an account |
| **Post-conditions:**  Server’s database is updated to latest sensor data |

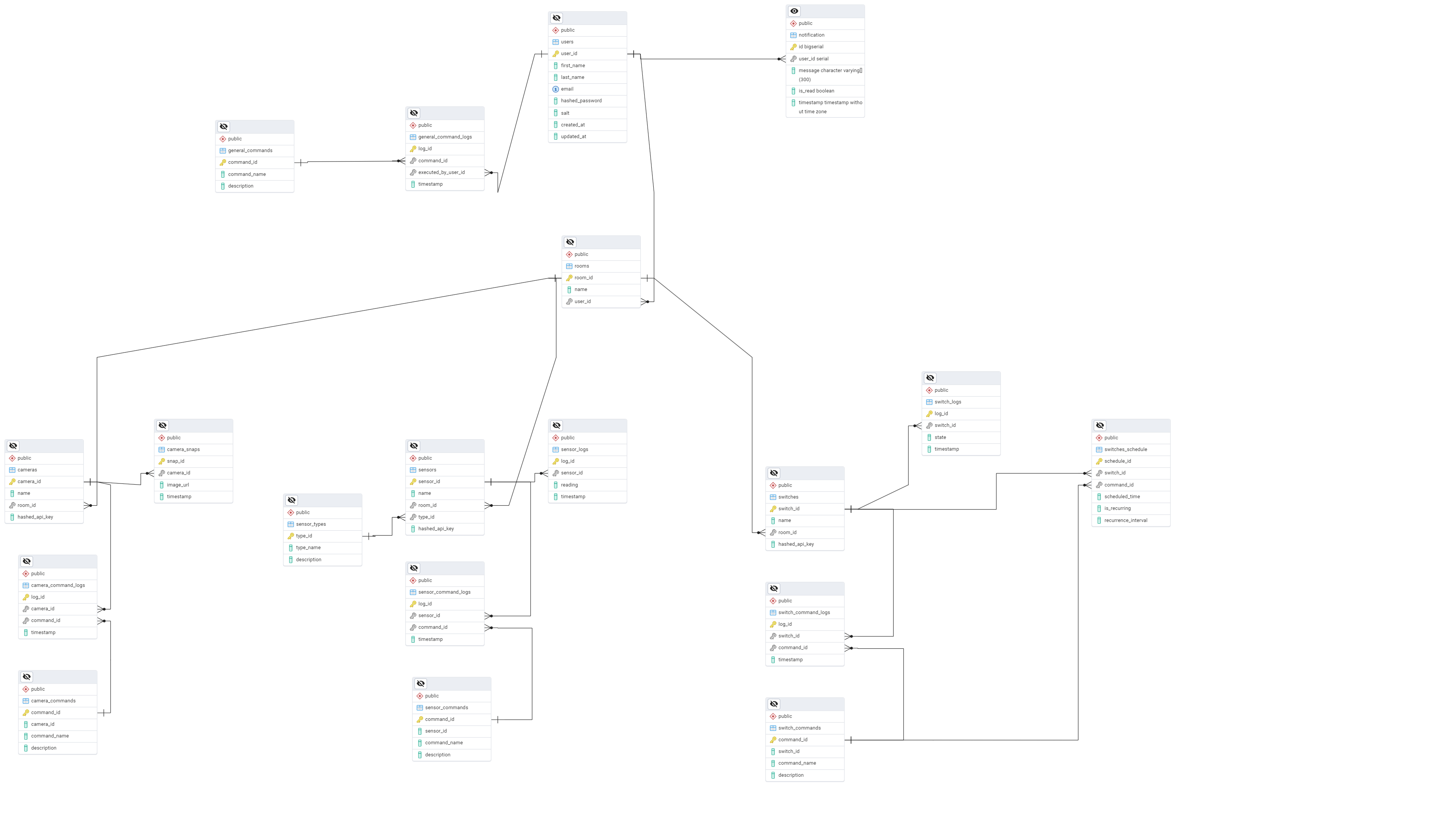
|  |
| --- |
| Use Case Name: Camera check for streaming request or camera shot Request |
| **Actors:** Device(Camera), Server |
| **Brief Description:**  The camera works by detecting a pending request in the server where then it sends a camera shot or starts streaming |
| **Basic Flow:**   1. Server has a request pending from the user 2. A camera device sees the request 3. Response sent as needed |
| **Alternative Flow:**  **None** |
| **Pre-conditions:**  Device is connected to internet  A request is present and pending |
| **Post-conditions:**  The users can see the response |

|  |
| --- |
| Use Case Name: Switch checks for a state (ON/OFF) change to apply |
| **Actors:** Device(Switch), Server |
| **Brief Description:**  The Switch works by detecting the latest state to be different from its current state. Then it applies the latest state.  In case of a schedule, the server checks the schedule periodically to update the state. For then then the switch device to see it and apply it/ |
| **Basic Flow:**   1. Switch state is changed 2. A Switch device sees the change 3. Switch change accordingly |
| **Alternative Flow:**  **None** |
| **Pre-conditions:**  Device is connected to internet  Switch state is changed |
| **Post-conditions:**  The home appliance state is changed now. |

# Chapter 3: System Design

## 3.1 ER Diagram

## 3.2 Relational Diagram



# Chapter 4: Implementation

**1. Overview of the Implementation**

The home automation system was designed to integrate multiple smart devices such as cameras, sensors, and switches, enabling users to control and monitor their homes efficiently through a mobile app or web interface. The implementation leverages Go, Gin, PostgreSQL, and Flutter, alongside powerful technologies like Llama 3.3 for AI-based voice commands. The backend focuses on scalable device communication, while the frontend ensures an intuitive user experience across different platforms. Security and performance were key considerations, with JWT authentication and secure data handling ensuring user privacy.

**2. System Architecture**

The system follows a client-server architecture. The mobile and web interfaces serve as clients, interacting with a backend built with Go and Gin, which handles requests from the user and communicates with the PostgreSQL database. The backend processes device commands, stores sensor data, and provides the necessary logic for device control. A communication layer connects the devices, which include cameras, sensors, and switches. Voice command functionality is powered by Llama 3.3 hosted on Ollama’s inference platform.

**Technologies Used**:

* Backend: Go, Gin
* Frontend: Flutter (Mobile & Web)
* Database: PostgreSQL
* AI: Llama 3.3 via Ollama
* Device Communication: HTTP, WebSockets
* Authentication: JWT, Bcrypt hashing with salt

**3. Frontend Implementation**

The frontend is implemented in Flutter to support both mobile and web platforms, providing a responsive design for controlling devices and viewing data. The user interface allows users to log in, link devices, issue commands, and view sensor readings. The UI is designed to be intuitive, supporting common mobile gestures and responsive layouts for web users. The frontend interacts with the backend via API calls to manage authentication, retrieve device data, and control devices.

**UI/UX Design**:

* A clean, minimalistic interface with intuitive device control features.
* User flows for logging in, controlling devices, and viewing sensor data.
* Real-time device state updates to reflect changes in the home environment.

**4. Backend Implementation**

The backend is developed using Go and Gin, offering a high-performance, scalable solution for handling API requests and managing device interactions. Each request is authenticated via JWT, ensuring that users can only control their devices. The backend also communicates with PostgreSQL for storing and retrieving device information, user data, and sensor readings. It includes functionality for scheduling tasks, logging user actions, and handling voice commands.

**API Design**:

1. RESTful APIs for login, device control, and sensor data retrieval.
2. Authentication endpoints that validate user credentials and provide access tokens.
3. Device management APIs for adding, removing, and linking devices to the user account.

**Business Logic**:

* The backend processes device commands and sends instructions to the devices (e.g., turn on/off switches, request camera shots).
* The backend integrates with the Llama 3.3 AI model to process voice commands and execute corresponding device actions.

**5. Database Implementation**

The database is implemented using PostgreSQL to store user accounts, device configurations, sensor data, and command logs. The database schema is designed with efficiency and scalability in mind, allowing quick access to device statuses, user preferences, and historical sensor readings. Key tables include users, devices, commands, and schedules. Data integrity is ensured through careful validation of inputs and foreign key relationships between tables.

**Database Schema**:

* **Users Table**: Stores user credentials, profiles, and linked devices.
* **Devices Table**: Stores device configurations, IDs, and connection details.
* **Commands Table**: Logs all user-issued commands for auditing and prediction purposes.
* **Sensors Table**: Stores sensor measurements such as temperature, motion, and power readings.

**6. Device Integration and Communication**

Devices like cameras, temperature sensors, and switches are connected to the system through a variety of communication protocols such as HTTP and WebSockets. The devices expose APIs for status updates and command execution. When a user sends a command through the frontend, the backend relays this command to the relevant device, which then executes the action and provides feedback to the system.

**Device Communication Protocol**:

* **HTTP**: Used for synchronous communication with devices, such as turning switches on or off.
* **WebSockets**: Utilized for real-time data updates, such as streaming camera feeds or sending sensor measurements.

**7. Voice Command Implementation**

Voice command functionality is implemented using Llama 3.3, an advanced AI model running on Ollama’s platform. This system processes user voice inputs to generate meaningful commands for controlling devices. The voice commands are captured by the mobile app, then sent to the backend, where they are interpreted by the AI model. The model checks whether the command is valid and performs the necessary action (e.g., turning a device on or off).

**Command Processing**:

* Voice commands are converted into text and passed to the backend for processing.
* The backend uses Llama 3.3 to analyze the command and verify its validity.
* Commands are executed if valid; otherwise, an error message is returned to the user.

**8. Security Implementation**

Security is a critical part of the system. To ensure that user data and device control are kept secure, we implemented the following measures:

* **Data Encryption**: All sensitive data, such as user credentials and device commands, is encrypted using SSL/TLS during transmission.
* **Authentication**: JWT tokens are used for stateless user authentication, and bcrypt hashing with salt ensures password security.
* **Authorization**: Role-based access control (RBAC) is employed to grant users permission only to devices they own or have been granted access to.

**9. Testing**

The system underwent rigorous testing to ensure it meets performance and reliability standards:

* **Unit Tests**: Individual components such as device controllers, API endpoints, and authentication mechanisms were tested.
* **Integration Tests**: The interaction between the frontend, backend, and devices was thoroughly tested to ensure seamless communication.
* **End-to-End Tests**: Full user workflows, such as logging in, linking devices, issuing commands, and receiving sensor data, were validated.

**10. Deployment**

The system was deployed on a cloud-based infrastructure, utilizing Docker containers for portability and ease of scaling. The backend runs in a Kubernetes cluster for high availability, while the frontend is deployed using CI/CD pipelines to ensure continuous delivery and updates. Monitoring tools are integrated to track the health of the system, and logs are collected for troubleshooting and analysis.

**Deployment Architecture**:

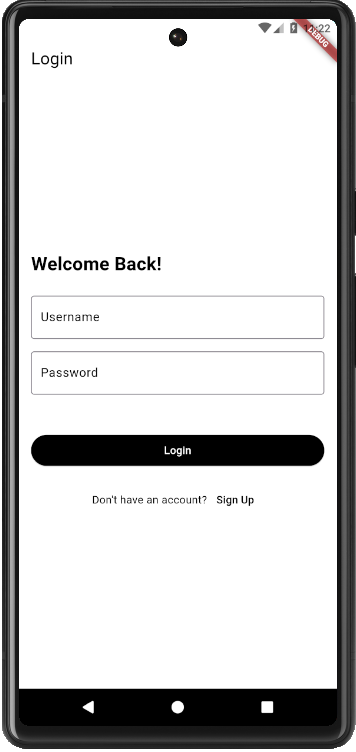
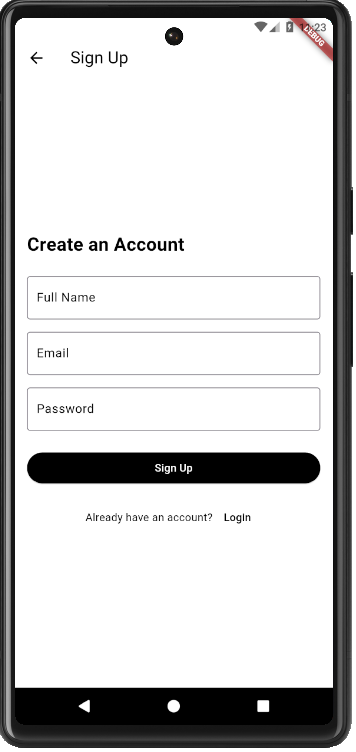
* **Backend**: Hosted on cloud servers with auto-scaling enabled.
* **Frontend**: Deployed on a web server and distributed via app stores for mobile.
* **Database**: PostgreSQL hosted on managed cloud infrastructure.

**11. Challenges and Solutions**

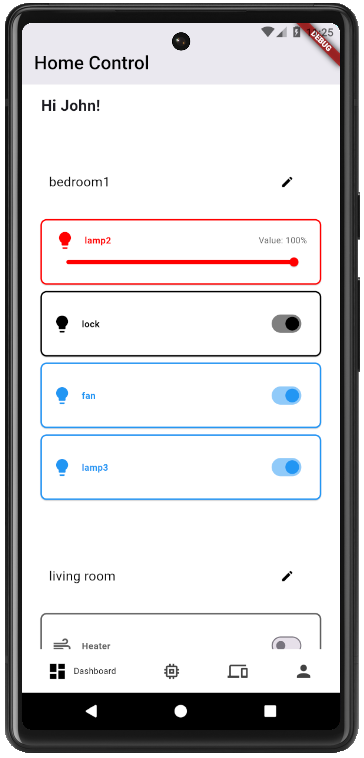
Several challenges were encountered during the implementation:

* **Device Communication**: Ensuring reliable communication between the backend and various devices was initially challenging due to different communication protocols. This was resolved by implementing a flexible communication layer that supports HTTP and WebSockets.
* **Voice Command Processing**: Integrating voice commands with the backend was complex due to the need for AI processing. The solution was to use Llama 3.3 on Ollama’s inference platform, which provided the necessary AI capabilities with efficient CPU and GPU usage.

**Sign-Up and Sign-In Pages**



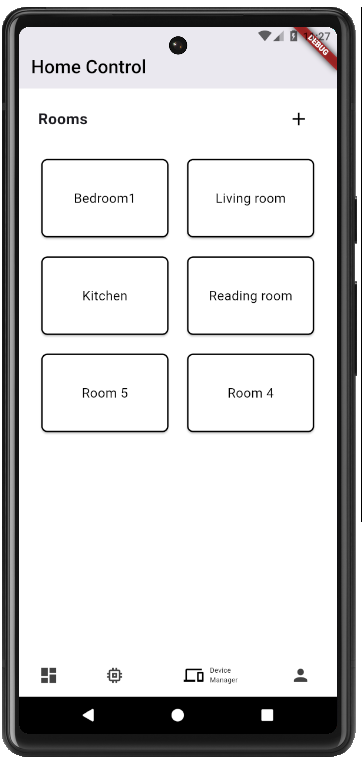
# DashBoard Page

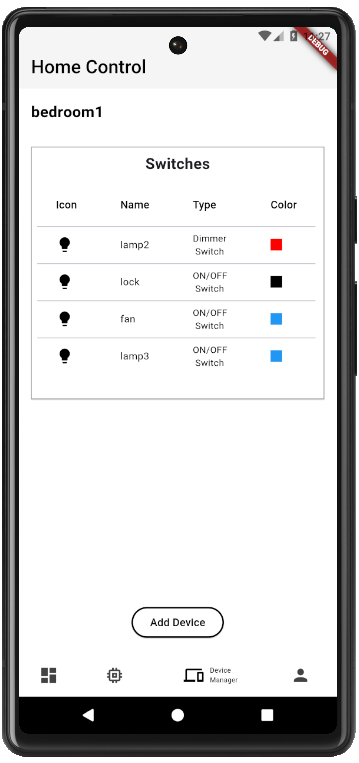


# AI Assitant Page



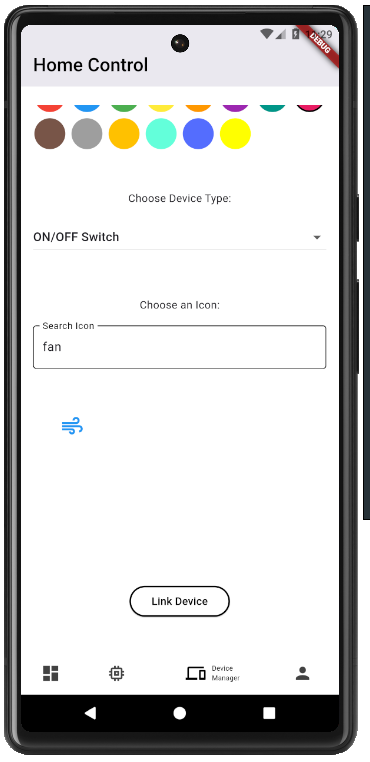
**Device Manager Pages**

****

****

# Add Device Pages

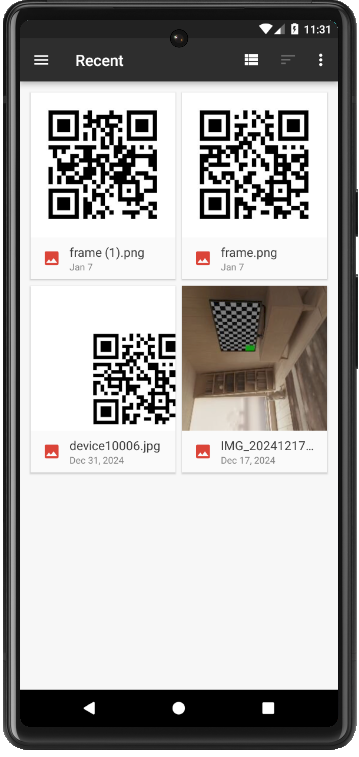
# 

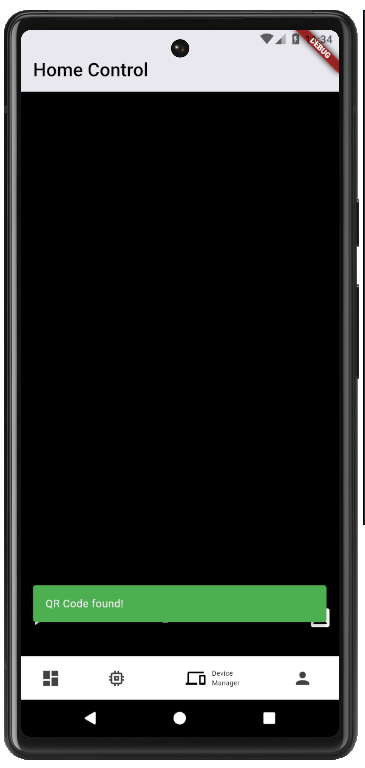


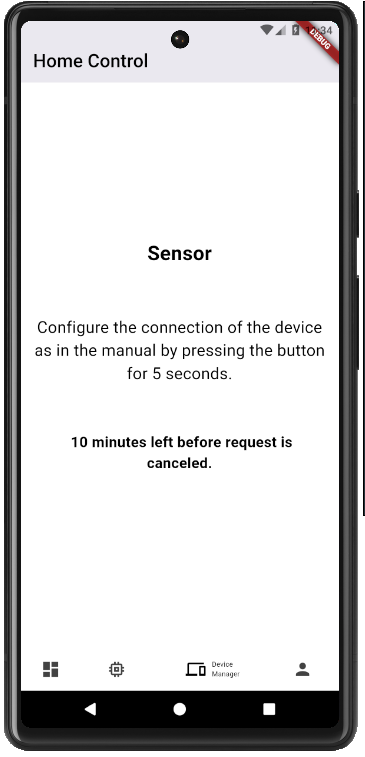
# 

# 

Option to pick image from the gallery







The user should press a button on the device for 5 seconds. In this case device is a Sensor. The Button is a type of authentication where token is sent from the device to the backed to allow the linking.

After this the device is linked and controlled by the account

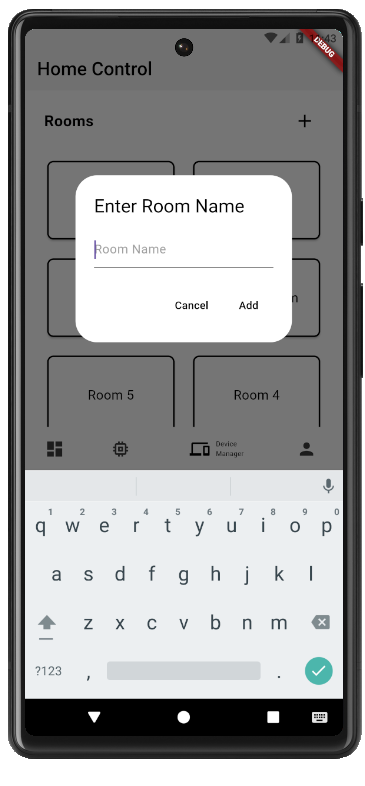
This the user has the request available in the Backend’s memory (RAM) for 10 minutes only.

If several requests are made by mistake, the button when pressed will authenticate the first request.



Linking request successful after the button was pressed

**Add Room**

****

# Profile Page

# 

# Chapter 5: Conclusion and Future Work

## 5.1 Conclusion

## The home automation system implemented in this project successfully integrates a range of smart devices, such as cameras, sensors, and switches, offering users an intuitive way to control and monitor their homes. By utilizing a robust technology stack consisting of Go, Gin, PostgreSQL, Flutter, and Llama 3.3, the project leverages both efficient backend processes and a seamless frontend experience for mobile and web users.

## The system enables features like device management, real-time data viewing, voice command functionality, and secure user authentication, all backed by a highly scalable architecture. Through strategic use of AI for voice commands and multi-threaded processing, the platform delivers both speed and flexibility, ensuring users have full control over their connected devices at any time.

## Overall, this project represents a significant step toward creating a user-friendly, secure, and scalable home automation platform. Future improvements may include additional device integrations, expanded AI capabilities, and more advanced predictive features based on user behavior. The foundation laid here provides a versatile base for further innovation in the realm of smart home technology.

## Future Work

1. **Device Integration and Expansion**

Integrate additional smart devices into the system, such as thermostats, smart locks, smart appliances, and home entertainment systems. This would further expand the range of control the platform offers, making it a comprehensive smart home solution.

1. **AI and Voice Command Enhancement**

Improve the AI capabilities for voice commands, enabling the system to understand more complex instructions and user contexts. Incorporating machine learning to predict user behavior and automate tasks based on historical data will also enhance system intelligence.

1. **Energy Efficiency and Automation**

Implement features to optimize energy usage by controlling devices based on real-time data, such as electricity pricing or weather conditions. Adding more advanced scheduling and automated energy-saving suggestions would help users reduce their environmental footprint.

1. **Security and Multi-user Access**

Enhance security by adding multi-user support with different permission levels and introducing biometric authentication for user logins. Implementing advanced security features, such as motion detection and real-time alerts, would also improve home safety.

# References