**ARDHI UNIVERSITY**

****

**SCHOOL OF EARTH SCIENCES, REAL ESTATE, BUSINESS STUDIES AND INFORMATICS (SERBI)**

DEPARTMENT OF COMPUTER SYSTEMS AND MATHEMATICS (CSM)

BSC. INFORMATION SYSTEM MANAGEMENT (BSC. ISM)

IS 292: PROJECT II: INFORMATION SYSTEMS ANALYSIS, DESIGN AND IMPLEMENTATION

YEAR 2025/26

**PROJECT TITLE: WEB AND VOICE-BASED HOME ELECTRONIC DEVICES SMART SWITCHING SYSTEM**

**GROUP (6) MEMBERS**

|  |  |  |
| --- | --- | --- |
| **S/N** | **STUDENTS’ FULL NAME** | **REGISTRATION NUMBER** |
| 1 | BEDAH FREDRICK KABEHO | 31678/T.2023 |
| 2 | HEMED ZAHORO MOHAMED | 30352/T.2023 |
| 3 | MERCY JOHN HAULE | 32000/T.2023 |
| 4 | AMIN ISSA SAID | 31030/T.2023 |
| 5 | ILHAM ABDALLAH LUSINDE | 30667/T.2023 |
| 6 | ARNOLD MAGAI MASHAURI | 31101/T.2023 |

# **DECLARATION**

We, **Group 6,** hereby declare that this report is our own work and effort. The work in this report was carried out in accordance with the regulations of the Ardhi University.

We have properly acknowledged and given credit to all the researchers whose work we have used and mentioned in this report. We also confirm that we have not copied any paragraphs, text, data, results, or any other information from books, journals, reports, websites, or other sources and claimed it as our own work.

**S/No Student’s Name Signature Date**

1. BEDAH F KABEHO ……………………… ……………………
2. HEMED Z MOHAMED ……………………… …………………
3. MERCY JOHN HAULE ……………………… …………………
4. AMIN ISSA SAID ………………………. ..………………….
5. ILHAM ABDALLAH LUSINDE ………………………. ……………………
6. ARNOLD MAGAI MASHAURI ………………………. ……………………

# **CERTIFICATION**

The undersigned certify that they have read and hereby recommend for acceptance by Ardhi University a project titled: Web and voice-based home electronic devices smart switching system in fulfillment of the requirements for the accomplishment of first year studies at Ardhi University.

………………………………

Dr. Ali Ahmada

(Supervisor)

Date: ……………….

……………………………….

Mr. Ibrahim J Ryoba

(Supervisor)

Date: ……………….

# ACKNOWLEDGEMENT

We sincerely extend our innermost gratitude to the Almighty God, the creator and sustainer of life for his grace and love to us, our family, friends and all that are around us since they have been the reason for us to be more than we thought we could be.

We extend our appreciation to Dr. Ahmada O. Ali and Mr. Ibrahim J. Ryoba for their abundant support outstanding supervision and sacrifice throughout this study from the beginning to its completion, and their patience and tireless efforts to provide us with intellectual direction and criticisms were very inspirational and helpful for us to complete this project. We would like to acknowledge with gratitude, support from our families and thank people who indirectly participated in this project without forgetting our fellow students for their support and suggestions. Also, we extend our thanks to all staff of the Department of computer system and mathematics for their contribution in completion of our study.

# LIST OF ABBREVIATIONS

ARU Ardhi University

UML Unified Modeling Language

|  |  |
| --- | --- |
| API | Application Programming Interface |
| CSS | Cascading Style Sheets |
| ESP32 | Espressif Systems 32-bit Microcontroller |
| HTML | HyperText Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IDE | Integrated Development Environment |
| IoT | Internet of Things |
| IFTTT | If This Then That (Cloud Automation Tool) |
| IJERT | International Journal of Engineering Research &  Technology |
| MVT | Model-View-Template |
| MQTT | Message Queuing Telemetry Transport |
| OOD | Object-Oriented Design |
| Wi-Fi | Wireless Fidelity |
| SQL | Structured Query Language |

# ABSTRACT

Many people often forget to turn off electrical appliances such as lights and fans when leaving their rooms or houses. This leads to energy wastage, increased electricity costs, reduced device lifespan, and potential safety hazards such as overheating and fire outbreaks. To address this issue, users can remotely access their home appliances through the web to ensure they are turned on or off as needed.

This project presents the design and development of a smart home switching system that can be controlled via voice commands and the internet. The system allows users to control devices such as fans and lights from a distance using a web interface built with Django, an ESP32 microcontroller, and voice command. The development process followed an Agile approach, incorporating user feedback and iterative testing. Key functional requirements included real-time device status updates, user authentication, and voice command processing. Non-functional requirements specified at least 95% system uptime and a response time within three seconds.

The hardware components used include the ESP32 microcontroller, relay modules, breadboard, end devices like fans and bulbs. The software stack consists of HTML, CSS, Django, JavaScript, and HTTP communication with IoT devices. Integration testing confirmed that the hardware, software, and backend components functioned correctly together. The system supports both web and voice control interfaces, offering flexibility and ease of use.

This solution fills gaps in existing systems by providing an affordable and accessible smart home platform that functions reliably even with limited internet connectivity. Future improvements may involve integrating machine learning to enhance voice recognition accuracy and analyze energy consumption patterns. The project demonstrates the potential of open-source technologies to create user-friendly and scalable home automation systems.

**Table of Contents**

[DECLARATION i](#_Toc201309883)

[CERTIFICATION ii](#_Toc201309884)

[ACKNOWLEDGEMENT iii](#_Toc201309885)

[LIST OF ABBREVIATIONS iv](#_Toc201309886)

[ABSTRACT v](#_Toc201309887)

[LIST OF TABLES ix](#_Toc201309888)

[LIST OF FIGURES x](#_Toc201309889)

[CHAPTER ONE 1](#_Toc201309890)

[INTRODUCTION 1](#_Toc201309891)

[**1.1** **General Introduction** 1](#_Toc201309892)

[**1.2 Statement of the Problem** 2](#_Toc201309893)

[**1.3 Objectives** 2](#_Toc201309894)

[**1.3.1 General Objective** 2](#_Toc201309895)

[1.3.2 Specific objective 2](#_Toc201309896)

[1.3.3 Research Questions 3](#_Toc201309897)

[1.3.4 Significance of the Study 3](#_Toc201309898)

[CHAPTER TWO 4](#_Toc201309899)

[LITERATURE REVIEW 4](#_Toc201309900)

[**2.1 Introduction** 4](#_Toc201309901)

[**2.2 Related Studies** 4](#_Toc201309902)

[**2.3 Research Gap** 5](#_Toc201309903)

[CHAPTER THREE 6](#_Toc201309904)

[METHODOLOGY 6](#_Toc201309905)

[**3.1 Introduction** 6](#_Toc201309906)

[**3.2 Selected Methodology** 6](#_Toc201309907)

[3.2.1 General Methodology 6](#_Toc201309908)

[3.2.2 Methodology for Gathering User Requirement 6](#_Toc201309909)

[3.2.3 Methodology for System design 8](#_Toc201309910)

[3.2.4 Methodology for System implementation 8](#_Toc201309911)

[3.2.5 Methodology for System testing and Validation 9](#_Toc201309912)

[**3.3 Methodology table** 10](#_Toc201309913)

[CHAPTER FOUR 11](#_Toc201309914)

[SYSTEM ANALYSIS AND DESIGN 11](#_Toc201309915)

[**4.1 Introduction** 11](#_Toc201309916)

[**4.2 Requirement Analysis** 11](#_Toc201309917)

[4.2.1 System Requirement 11](#_Toc201309918)

[4.2.2 Data Requirement 11](#_Toc201309919)

[4.2.3 Software Requirement 12](#_Toc201309920)

[4.2.4 Hardware Requirement 12](#_Toc201309921)

[4.2.5 User Requirement 13](#_Toc201309922)

[4.2.6 Functional Requirement 13](#_Toc201309923)

[4.2.7 Non-functional Requirement 13](#_Toc201309924)

[**4.3 System Design** 14](#_Toc201309925)

[4.3.1 System Architecture 14](#_Toc201309926)

[4.3.2 Use Case Diagram 15](#_Toc201309927)

[4.3.3 Class Diagram 17](#_Toc201309928)

[CHAPTER FIVE 18](#_Toc201309929)

[**5.1 Introduction** 18](#_Toc201309930)

[**5.2 Implementation** 18](#_Toc201309931)

[5.2.1 System Overview 18](#_Toc201309932)

[5.2.2 Backend Development 18](#_Toc201309933)

[5.2.3 Frontend Development 18](#_Toc201309934)

[5.2.4 Registration Page 19](#_Toc201309935)

[5.2.5 User Dashboard 19](#_Toc201309936)

[5.2.6 Manage users page for Admin 20](#_Toc201309937)

[5.2.7 IoT Hardware Integration 21](#_Toc201309938)

[5.2.8 Voice Control 21](#_Toc201309939)

[5.2.9 Device Setup 22](#_Toc201309940)

[**5.3 Testing** 22](#_Toc201309941)

[5.3.1 Functional Testing 22](#_Toc201309942)

[5.3.2 Integration Testing 23](#_Toc201309943)

[5.3.3 Usability Feedback 23](#_Toc201309944)

[CHAPTER SIX 24](#_Toc201309945)

[CONCLUSION AND RECOMMENDATION 24](#_Toc201309946)

[**6.1 Introduction** 24](#_Toc201309947)

[**6.2 Conclusion** 24](#_Toc201309948)

[6.2.1 To identify and gather user requirements 24](#_Toc201309949)

[6.2.2 To design the system 24](#_Toc201309950)

[6.2.3 To implement the system 24](#_Toc201309951)

[6.2.4 To test and validate the system 25](#_Toc201309952)

[**6.3 Challenges and Limitations** 25](#_Toc201309953)

[**6.4 Recommendations** 25](#_Toc201309954)

[**References** 26](#_Toc201309955)

# LIST OF TABLES

[Table 1: Methodology Table 9](#_Toc201170348)

[Table 2: Functional Requirement Table 20](#_Toc201170349)

# LIST OF FIGURES

[Figure 1: Question 01 from Google form 7](#_Toc201169752)

[Figure 2: Question 02 from Google form 7](#_Toc201169753)

[Figure 3 : Use Case Diagram 14](#_Toc201169754)

[Figure 4: Class Diagram 15](#_Toc201169755)

[Figure 5: Registration Page 17](#_Toc201169756)

[Figure 6: User Dashboard 17](#_Toc201169757)

[Figure 7: User Management By Admin 18](#_Toc201169758)

[Figure 8: Connection Of Our Devices 19](#_Toc201169759)

# CHAPTER ONE

# INTRODUCTION

## **General Introduction**

In the era of rapid technological advancement, the integration of smart systems into daily life has transformed traditional methods of performing basic tasks. One of the most significant developments in this domain is home automation, which enables users to control electronic appliances remotely through digital interfaces. This study focuses on designing and implementing a web-based system that allows users to operate home appliances such as lights and fans using a browser interface from any location with internet access.

The concept of remotely controlling household devices is no longer futuristic. With the growth of the Internet of Things (IoT) and cloud computing, users can now interact with their home environments in more intelligent and efficient ways. Home automation enhances convenience, improves energy efficiency, and supports users with mobility impairments. Traditionally, home devices were manually controlled using physical switches, but modern approaches integrate voice assistance and web applications to simplify usage (Alam et al., 2021).

This project specifically aims to create a web-based switching system that connects to devices like lights and fans via a microcontroller and wireless network. Users can log into a secure webpage and send commands to control appliances in real time. The system is accessible via desktops, laptops, or mobile phones, enabling flexibility and continuous monitoring. The website acts as the central control unit, linking the user interface to the hardware through server-side scripts or APIs, depending on the implementation.

Such systems not only provide comfort but also help optimize energy use by allowing users to turn off forgotten appliances remotely. With features like real-time feedback and device status monitoring, web-based control platforms are becoming a preferred solution in modern smart homes.

This research builds upon existing studies and innovations in IoT and web technologies, such as those presented in the International Journal of Engineering Research & Technology (IJERT)*,* where similar web-based home automation systems have been implemented using microcontrollers and IoT frameworks (Patel, 2019).

## **1.2 Statement of the Problem**

In many households, people often forget to switch off electrical appliances such as lights and fans. This leads to unnecessary energy consumption, increased electricity bills, and accelerated wear and tear on devices. Moreover, it poses serious safety risks, such as fire outbreaks caused by overheating or electrical faults due to prolonged use without proper precautionary measures.

Many existing smart systems offer only short-range control via Bluetooth and lack essential user-friendly features such as remote accessibility, voice command functionality, and device usage tracking. For example, the Smart Home Automation System controlled appliances via Bluetooth and Wi-Fi using a mobile app but had limited control range and lacked features like user roles or authentication (Kumar & Gupta, 2020). Similarly, theSmart Switch Control System Using Arduino with Android App functioned only within a local Wi-Fi range and did not support user login or multi-user roles (Patel & Chauhan, 2019).

To address these limitations, this project proposes the development of a Web and Voice-Based Smart Switching System that enables users to control electronic devices such as lights and fans through a web interface or voice commands using IoT technology. The system will support personalized device control, activity tracking, and remote access from any location with internet connectivity.

## **1.3 Objectives**

## **1.3.1 General Objective**

The general objective of this project is to design and implement a smart system that enables users to remotely control and monitor home electronic devices using a web application and voice commands integrated with IoT hardware.

### 1.3.2 Specific objective

1. To identify and gather user requirements for the Web and Voice-Based Home Electronic Devices Switching System.
2. To design a responsive web-based interface for controlling devices in the Web and Voice-Based Home Electronic Devices Switching System.
3. To implement the Web and Voice-Based Home Electronic Devices Switching System using appropriate technologies.
4. To test and validate the functionality and usability of the Web and Voice-Based Home Electronic Devices Switching System.

### 1.3.3 Research Questions

1. Would you be interested in controlling electronic devices like lights and fans at home using your phone or laptop?
2. Do you think having voice control for switching devices ON\OFF is useful?
3. Would you like to receive a notification if a device stays ON for too long?
4. Would you like to view the ON\OFF history of your devices?

### 1.3.4 Significance of the Study

1. The system will allow users to control their home appliances from anywhere using a mobile phone or other internet-enabled devices.
2. Voice control will enhance accessibility, especially for users with disabilities, enabling full control over connected devices.
3. The system will provide a history log showing the ON/OFF status of devices, helping users verify actions when in doubt.
4. Users will receive notifications when a device has been ON for too long, allowing them to act and prevent energy waste or hazards.

# CHAPTER TWO

# LITERATURE REVIEW

## **2.1 Introduction**

A literature study looks at existing research on smart home automation in a critical way, focusing on IoT topologies, voice control systems and user interface design. This study looks at technology developments, the problems with present solutions, and the chances for new ideas Ford et al. (2021).

## **Related Studies**

**Iot-Based Home automation:**

Brock (2020) examined on how Http protocols could be utilized to make it easier for smart home devices to talk to each other in a light way. His research demonstrated that Http works well in low-bandwidth conditions, but it caused latency problems because it relied on cloud services.   
This approach gets around that problem by lowering latency and not relying on the cloud through local Wi-Fi.

**Devices that respond to voice**

Mozilla (2023) looked into voice recognition in browsers using the web speech API. The study found that accuracy was good in quiet places, but performance with a lot of noise and accents.

Gap Closed: This system uses both manual and voice controls to make sure it is always accessible and reliable, even if voice is not available.

**Making User Interface:**

Chao (2020) talked about how important responsive design is for making sure that websites work on all devices. Their expertise helped us design our frontend layout, which works on both desktop and mobile devices.

**ESP32 for Home Automation**:

Ford et al. (2021) talked about how flexible and cheap the ESP 32 microcontroller is for IOT projects. Still, they found security holes in the firmware level.

Our response: The system uses Django’s built in authentication to make security level better.

**Energy efficiency in smart homes:**

Alam et al. (2021) proved that real time feedback from devices reduces the usage of unnecessary energy.   
Implementation*:* Our technology includes live status updates that let users easily turn off devices that aren't being used.

**Waterfall vs Agile in the development for IoT**

It was found that Agile worked better for IoT initiatives that were focused on the user and had multiple iterations.

Solutions argument: this system was built using Agile development methods, which let it be tested and improved all the time.

The Django Software Foundation (2023) found that unencrypted HTTP traffic was a major weakness in IoT Systems.

Response: Even if testing is done over a local network, plans are in place for future HTTPS deployment and secure coding methods.

**Accessibility of Voice Interface:**

According to Meszaros (2021), voice-only interfaces may not work for people with disabilities or speech problems.

Inclusion Strategy: This system has both manual and voice commands so that it can fulfil the demands of many different users.

## **2.3 Research Gap**

Existing studies predominantly rely on cloud-based infrastructures (Brock, 2020) or proprietary voice assistants (Mozilla, 2023), limiting affordability and offline functionality. This project bridges these gaps by:

1. Developing a locally hosted system with open-source tools.
2. Combining voice and manual controls for inclusivity.
3. Prioritizing real-time feedback without cloud latency.

# CHAPTER THREE

# METHODOLOGY

## **3.1 Introduction**

Methodology can be defined as a proper study or analysis of all the methods used in the

particular study or activity (Merriam-Webster, 2023). Also, it involves the use of various tools such as software and programming languages so as to achieve the general objective of the study (Dawson, 2019). Here, we will see the methods and actions we used to implement the Web and Voice-Based Home Electronic Devices Smart Switching System. The aim is to ensure the project is achieved and completed in a structural and systematic approach.

## **3.2 Selected Methodology**

### 3.2.1 General Methodology

The selected methodology for this project is the Agile Software Development Methodology, which involves breaking down of projects into smaller, manageable parts called "sprints" or "iterations", allowing for continuous improvement based on feedback given either after testing or user feedback.

The choice of this methodology is due to its feature that allows modular developments which improves collaboration between members, and hence improve teamwork and lets the development become faster. Also, through agile methodology we can easily turn back to previous step for the maintenance or improvement due to user feedback or after testing thus improves flexibility of a system.

### 3.2.2 Methodology for Gathering User Requirement

To gather user requirements, we have used two methodologies which are literature reviews and Google Forms (Online questionnaire).

literature reviews *-*It involves reading Existing projects, academic papers, and online resources to review and understand current trends, common features, and shortcomings in the already developed smart switching systems. This will help to get full picture of the smart system as well as to not repeat other challenges or shortcomings facing other smart switching systems, also to improve our system from those challenges. Some of the articles we have read and their challenges they face are like:

“Smart Switch Control System Using Arduino with Android App” by Anish Kumar and Neha Sharma published in International Journal of Engineering Research and Technology (IJERT)**:** what we find is that in the system it Only supported local Wi-Fi range, No admin/user roles or device logs

“Smart Home Automation System Using IoT” By Dr. Prashant Bhagat et al., published in International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering (IJIREEICE)**:** what we found is that in the system they Only used Bluetooth and had limited range also, No voice control, no real-time logging

Google Forms: It involve a digital form that we shared with our potential users to collect information about their needs, preferences, and expectations concerning the needs to control their devices remotely. This method is quick, cost-effective, as well as reaches many users. Some of the questions and user feedbacks are:

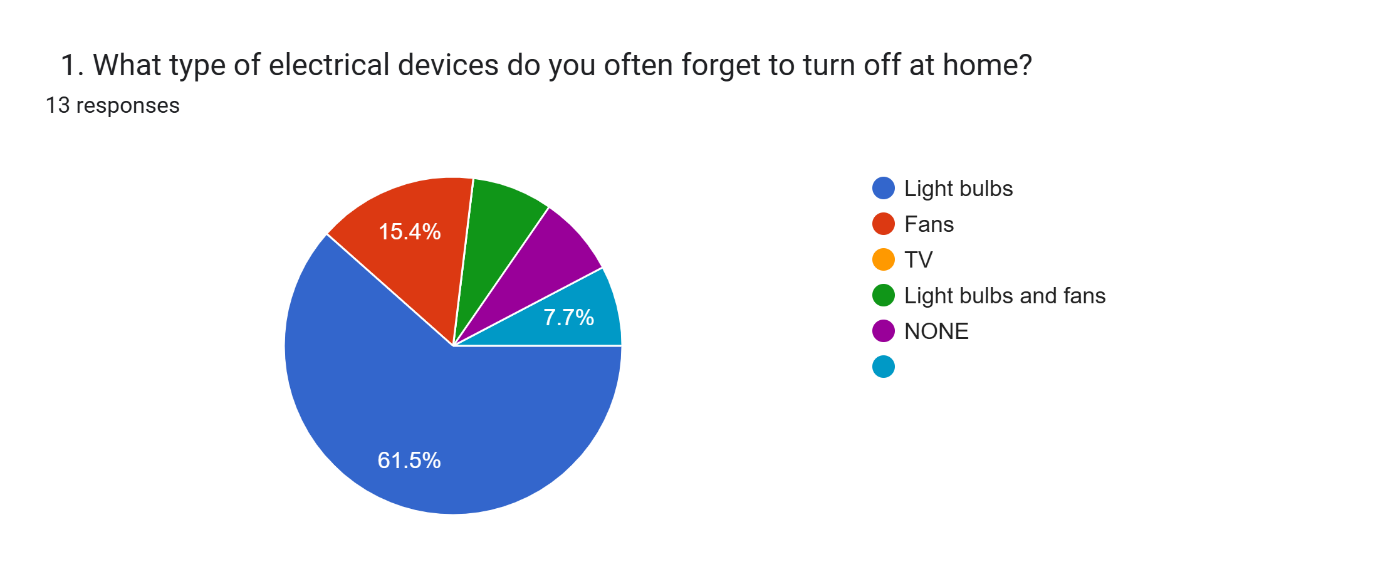


Figure 1: Question 01 from Google form

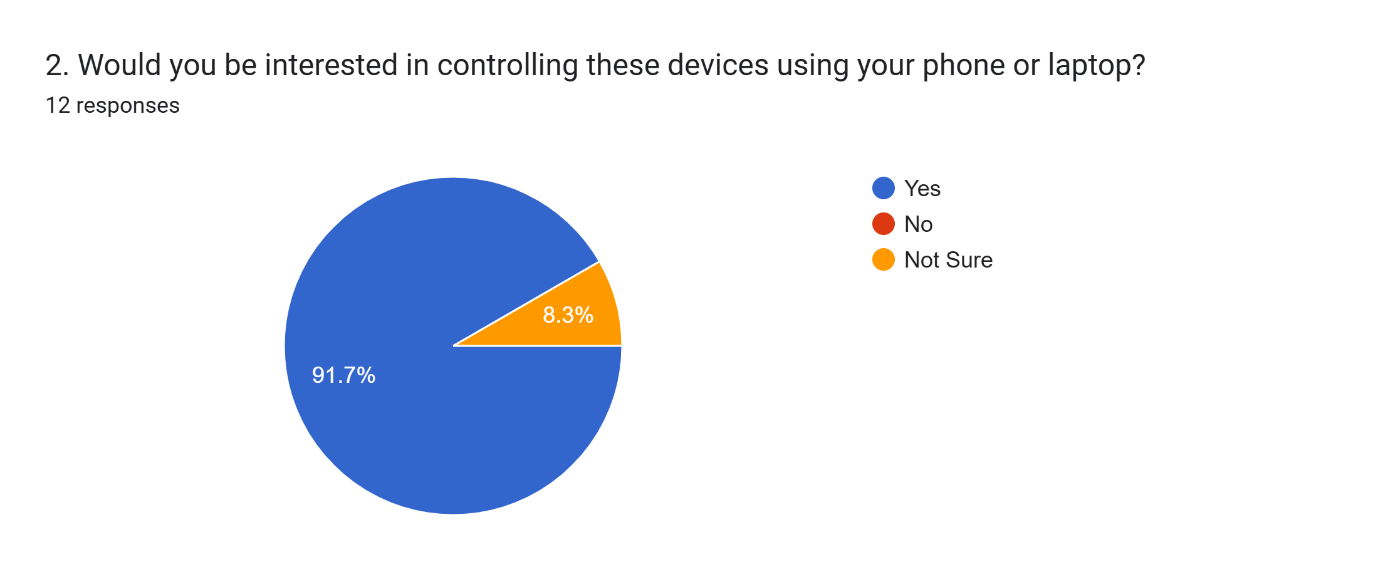


Figure 2: Question 02 from Google form

### 3.2.3 Methodology for System design

The design methodology we decided to use is Object Oriented Design (OOD) model, as it focuses on designing the system using system using objects, classes, attributes, and methods that map to real world entities. We decided to use this approach since it promotes modularity (dividing a system into smaller components), code reusability and easy maintenance of code. Also, through using OOD it helps a lot when project grows as we can only add some new Objects, attributes or methods without changing the structure of our codes.

### 3.2.4 Methodology for System implementation

The system will be implemented using the following technologies:

**Frontend**: HTML, CSS, JavaScript, Bootstrap, and Web Speech API (for voice commands) These tools are used to create a responsive and user-friendly interface, so that a user can perform their actions seamlessly and correctly.

**Backend**: Python with Django Framework. The backend is responsible for handling and processing all actions submitted by users. Also, it manages communication between the user interface and the database, processes different types of requests, and provides appropriate responses. We decided to choose Django due to its powerful features, scalability, and many built in components.

**Database**: SQLite. The database stores important information such as user details and device data, also we use SQLite as we don’t have much data to be stored and later we can upgrade to MySQL or PostgreSQL once system become complex and needs to store much data.

**Microcontroller**: ESP32, connected via WIFI and programmed using Arduino IDE with C++ (It serves as the main control unit, receiving and processing commands.)

Relay module*:* Acts as a switch to turn end devices ON or OFF, based on signals received from the ESP32.

End devices*:* Involves all devices need to be switched ON/OFF or those can be controlled by the system, but we decided to use fan and light as a prototype for our system.

### 3.2.5 Methodology for System testing and Validation

The testing methodology we use is: Alpha testing and User feedback;

Alpha testing*:* This is the initial testing of the system done by developers and a small group of test users to find bugs and validate system functionalities before releasing the system to normal users. Here it involves both types of testing including functional and non-functional of testing to make sure it meets user requirements.

User feedback*:* This is another method to validate the system functionality by accepting feedback from users after using them and be able to collect different updates provided by the users about the system.

## **3.3 Methodology table**

Table 1: Methodology Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Specific Objective** | **Methodology** | **Tools** | **Deliverable** |
| 1. To identify and gather user requirements. | |  |  | | --- | --- | |  | Google Forms, Literature Review | | Google Forms, Research Articles | |  |  | | --- | --- | |  | Requirements Analysis Document | |
| 1. To design a responsive web-based interface | |  |  | | --- | --- | |  | Object Oriented Design | | Draw.io, Figma | Use case and class diagrams |
| |  | | --- | |  | | 1. To implement the smart switching system | | Modular developments approach | HTML, CSS, JavaScript, Django, Arduino IDE, ESP 32 | Working system  (Web Pages + device switching functionality) |
| 1. To test and validate the system | Alpha Testing, Black Box Testing | |  |  | | --- | --- | |  | Browsers, Smartphones, User Feedback | | Bug Reports, User Feedback, Verified Features |

# CHAPTER FOUR

# SYSTEM ANALYSIS AND DESIGN

## **4.1 Introduction**

Systems analysis represents a stage within the system development lifecycle that encompasses gathering and analyzing information, recognizing issues, and breaking down systems into their constituent parts (Kendall & Kendall, 2020). Systems design can be characterized as the methodology for creating or substituting current systems through the specification of their elements to meet particular needs (Ford et al., 2021). This section describes the implementation methods employed in developing our system.

## **4.2 Requirement Analysis**

Requirements gathering means studying what users need to create clear plans for systems, hardware, or software. This step is very important in software development because it creates the foundation that developers use to build and design software systems (Django Software Foundation, 2023). The process involves collecting information from users and understanding their needs before any development work begins. There are two main types of requirements that need to be analyzed: user requirements, which describe what people want the system to do and how they expect it to work, and system requirements, which explain the technical details of how the system should be built and what resources it needs to operate properly.

### 4.2.1 System Requirement

System requirements are the technical specifications that describe what a computer system needs to work properly (Alzubi et al., 2022). When analyzing system requirements, there are two main types to consider. Hardware requirements specify the physical components needed, such as memory, processor speed, and storage space. Software requirements identify the programs and operating systems that must be present for the system to function correctly.

### 4.2.2 Data Requirement

The data collection process for our smart home system involves gathering information needed to control fans and lights using voice commands and smart switching. We collected data from user actions, including voice commands and how people turn fans and lights on and off, which helped the system learn what users want and work better. We also collected information about device status, which means checking if fans and lights were turned on or off and how well they were working. We stored setup information like user choices, device settings, and rules that tell fans and lights how to respond when people give voice commands or use switches. This way of collecting data helped our system provide good control for basic home items like fans and lights through both talking to the system and using smart switches.

### 4.2.3 Software Requirement

Software requirements describe the programs and applications that a computer system needs to work properly (Django Software Foundation, 2023). For our smart home system that controls fans and lights through voice commands and switching, we used several key software programs. Arduino IDE software was used to write and upload code to the microcontroller that controls the electrical components. Python programming language created the voice recognition system that understands user commands and processes speech input. We also used basic operating system software to manage the overall system operations and ensure all components work together smoothly. These software tools worked together to create a complete system that responds to voice commands and controls home appliances effectively.

### 4.2.4 Hardware Requirement

Hardware requirements describe the physical components and equipment that a computer system needs to operate correctly and perform its intended functions (Silberschatz et al., 2018). For our smart home system that controls fans and lights using voice commands and switching, we used several important hardware components. We used an Arduino microcontroller board as the main control unit that processes commands and controls the electrical devices. Relay modules were connected to manage the switching of fans and lights by controlling the electrical power flow to these appliances. A microphone was used to capture voice commands from users, allowing the system to receive audio input for processing. We also used to connect wires and breadboards to link all components together and create proper electrical connections. These hardware components worked together to build a functional system that can listen to voice commands and control home appliances through smart switching.

### 4.2.5 User Requirement

User requirements are the needs and expectations that people have for a system, describing what they want it to do and how it should work (Alzubi et al., 2022). There are two main types of user requirements: Functional requirements and Non-functional requirements system must perform and what features it should have. Non-functional requirements describe how well the system should work, including speed, ease of use, and reliability.

### 4.2.6 Functional Requirement

Functional requirements are the essential operations and features that a system must perform to meet user expectations and fulfill its objectives (Django Software Foundation, 2023). They define the core functionalities of the system, derived from identified user needs and system goals. For this project, the functional requirements focus on providing remote and efficient control of household electronic devices such as bulbs and fans. These requirements were selected to ensure usability, accessibility, and relevance to real-world domestic applications.

1. The system shall allow users to log in through a secure authentication interface to ensure access control.
2. The system shall enable users to switch lights and fans ON or OFF via a web interface.
3. The system shall display and update the current status of each connected device in real time.
4. The system shall support voice command input through the browser to control connected devices.
5. The system shall maintain a log of all device actions for monitoring and analytics purposes.
6. The system shall use Wi-Fi and HTTP protocols to synchronize communication between the frontend interface and the ESP32 microcontroller.

### 4.2.7 Non-functional Requirement

Non-functional requirements define the quality attributes of a system, including performance, usability, reliability, and security. These characteristics ensure that the system is not only functional but also efficient, user-friendly, and robust (Kendall & Kendall, 2020). For this project, the non-functional requirements were selected to ensure responsiveness, flexibility, and scalability for future enhancements.

1. The system shall be responsive and load all control elements within 3 seconds.
2. The system shall maintain a reliable connection between the web interface and the ESP32 microcontroller, ensuring at least 95% uptime during operation.
3. The system shall support mobile responsiveness and display correctly on both desktop and mobile browsers.
4. The system shall process and respond to voice commands within 2 seconds.
5. The system shall follow secure coding practices to ensure safe and protected communication, especially in production environments.
6. The user interface shall include accessibility features such as high contrast mode and a simplified layout for ease of use.

## **4.3** **System Design**

System design involves creating a blueprint that defines how the system’s components interact, including software modules, hardware interfaces, and data storage mechanisms (Pressman & Maxim, 2014). In this project, the system design serves to clearly illustrate the architecture, relationships between major components, and the flow of data across modules, ensuring a well-organized and scalable IoT-based solution.

### 4.3.1 System Architecture

System architecture refers to the high-level structure of the system and describes the interactions among key components, such as the user interface, backend server, microcontroller, and connected electronic devices (Meszaros, 2021). The goal of the architectural design is to provide a clear visualization of how user actions translate into hardware responses. This structure supports modular development, simplifies troubleshooting, and facilitates future system expansion and upgrades.

### 4.3.2 Use Case Diagram

Use case diagrams illustrate the interaction between users (actors) and the system in order to achieve specific goals. These diagrams are typically developed during the analysis phase to help identify the system’s functionalities from the user’s perspective (Meszaros, 2021)

In this project, the primary actor is the User, who interacts with the system through a web interface or voice commands. The main use cases include:

1. Login: User authenticates to access the system.
2. Control Devices: User turns devices (e.g., lights and fans) ON or OFF.
3. View Status: User checks the real-time status of connected devices.
4. Use Voice Commands: User issues voice-based commands to control devices.

These use cases ensure that all functional requirements are covered during the design, development, and testing phases.

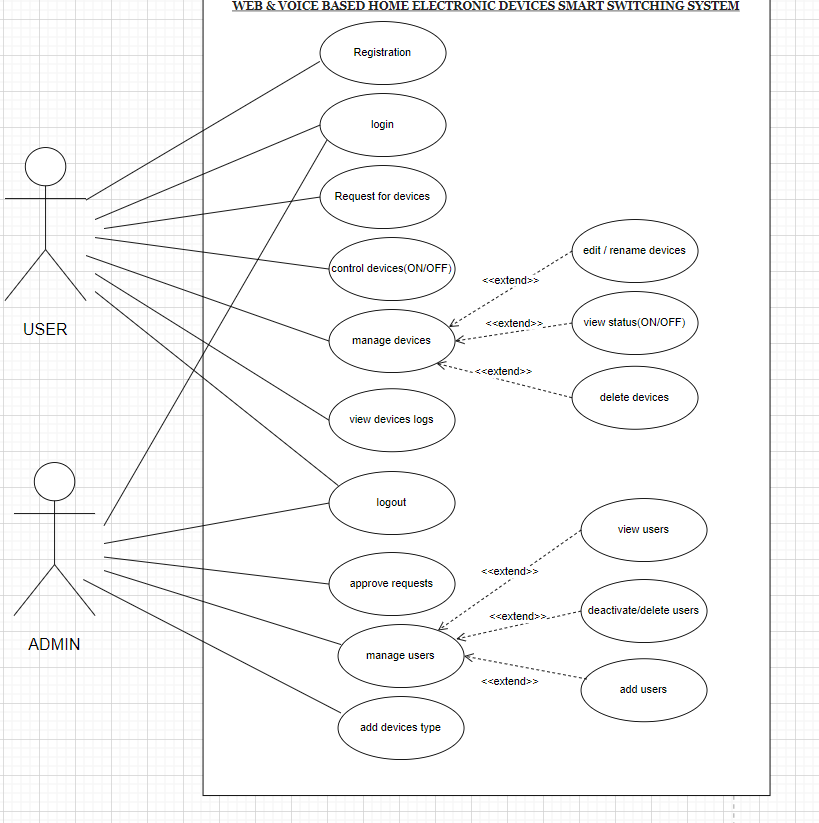


Figure 3 : Use Case Diagram

### 4.3.3 Class Diagram

A class diagram is a structural diagram in the Unified Modeling Language (UML) that defines the structure of a system by illustrating its classes, attributes, methods, and the relationships between objects (Ford et al., 2021). It represents the static view of the application and is especially helpful during the development phase to ensure that all components and their interactions are well-defined.

In this project, the class diagram models’ key components such as User, Device, Device Controller, and Logger. It shows how these classes interact with one another and how responsibilities are distributed across modules. For instance, the Device Controller class handles the logic for sending commands to devices, while the Logger class records status updates for monitoring and analysis.

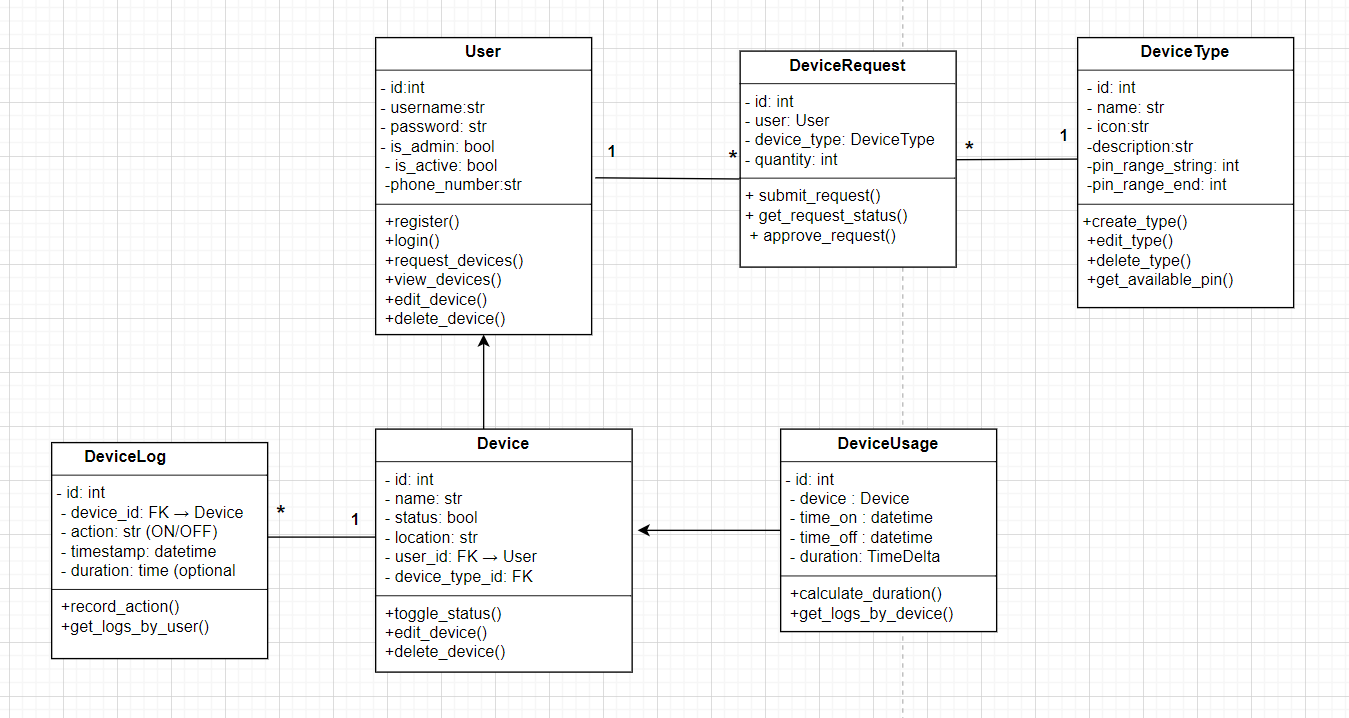


Figure 4: Class Diagram

# CHAPTER FIVE

**IMPLEMENTATION AND TESTING**

## **5.1 Introduction**

This chapter covers the implementation and testing of the Web and Voice-Based Home Electronic Devices Switching System. It outlines the integration of software and hardware components, along with the testing methods used to validate performance. Implementation involved setting up the web interface, backend logic, ESP32 microcontroller configuration, and voice command integration. Testing assessed the system's functionality and whether it met the predefined objectives (Ford et al. (2021)).

## **5.2 Implementation**

### 5.2.1 System Overview

The system allows users to control household appliances such as lights and fans through a web interface or browser-based voice commands. It uses the ESP32 microcontroller to establish communication between the web server and the physical devices. The backend was developed using Django, while the frontend used HTML, CSS, and JavaScript. All components were tested within a local Wi-Fi environment, and real-time response performance was observed.

### 5.2.2 Backend Development

The backend was built using the Django framework, which follows the Model-View-Template (MVT) architecture. Django was chosen for its simplicity, scalability, and built-in administrative tools (Django Software Foundation, 2023). It manages user authentication, device state handling, and communication with the ESP32 via HTTP requests

### 5.2.3 Frontend Development

The frontend was developed using HTML, CSS, and JavaScript. A responsive design ensured compatibility across different screen sizes. JavaScript played a key role in capturing user inputs and triggering control events. Real-time status updates of connected devices were displayed, providing a seamless user experience (Chao, 2020). Consider below pages

### 5.2.4 Registration Page

The **Registration Page** allows new users to sign up and create an account in the system. It collects essential user information such as username, email, password, and phone number. The design emphasizes simplicity and clarity to make the process accessible for all users, including those with limited technical skills. Input validations are performed on the frontend, and the submitted data is securely handled by the backend system using Django’s user authentication framework

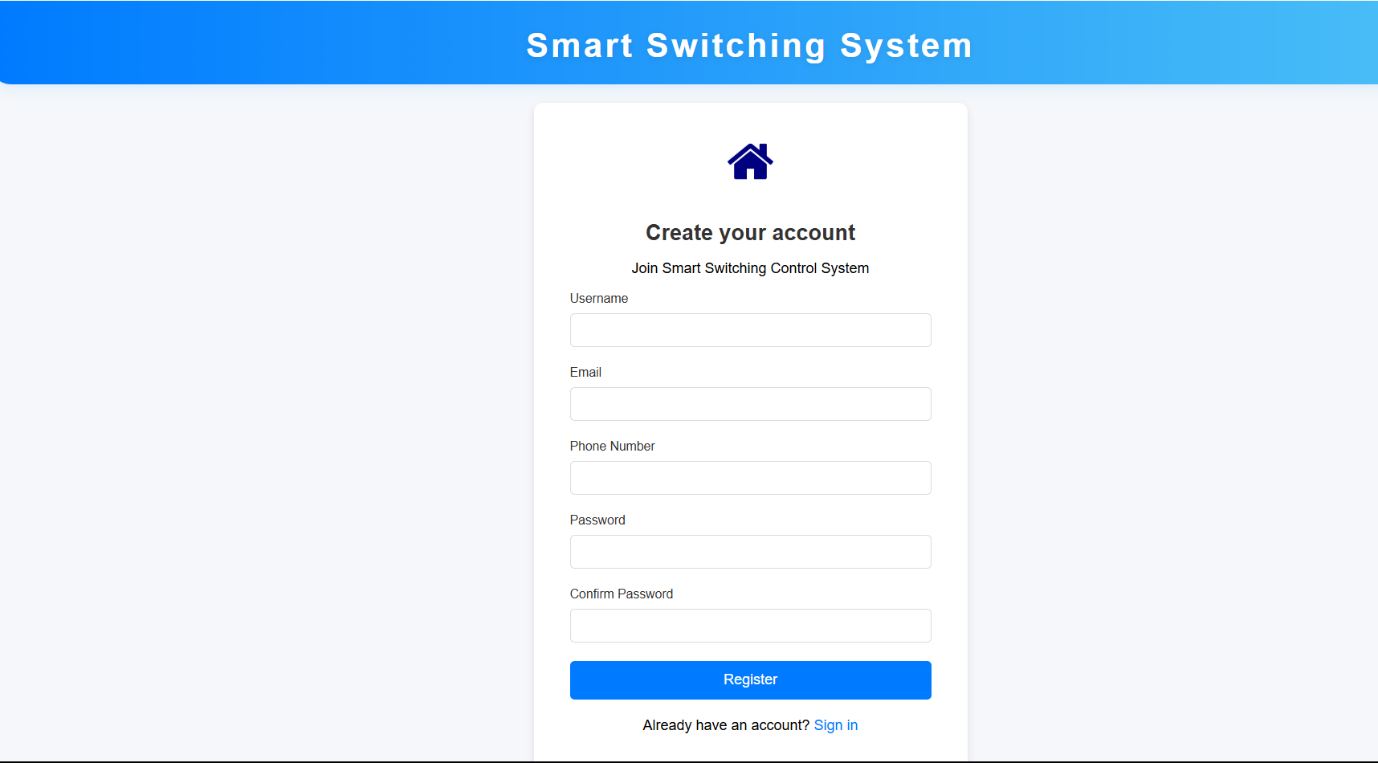


Figure 5: Registration Page

### 5.2.5 User Dashboard

The **User Dashboard** serves as the primary control panel for authenticated users. Upon logging in, users are presented with a personalized dashboard that shows the current status of their connected devices (e.g., lights and fans). Each device is represented by a card with real-time ON/OFF status and control buttons. The dashboard also includes a navigation sidebar that allows users to access device management, view logs, and log out. The interface is responsive and supports both manual and voice-based control using the Web Speech API.

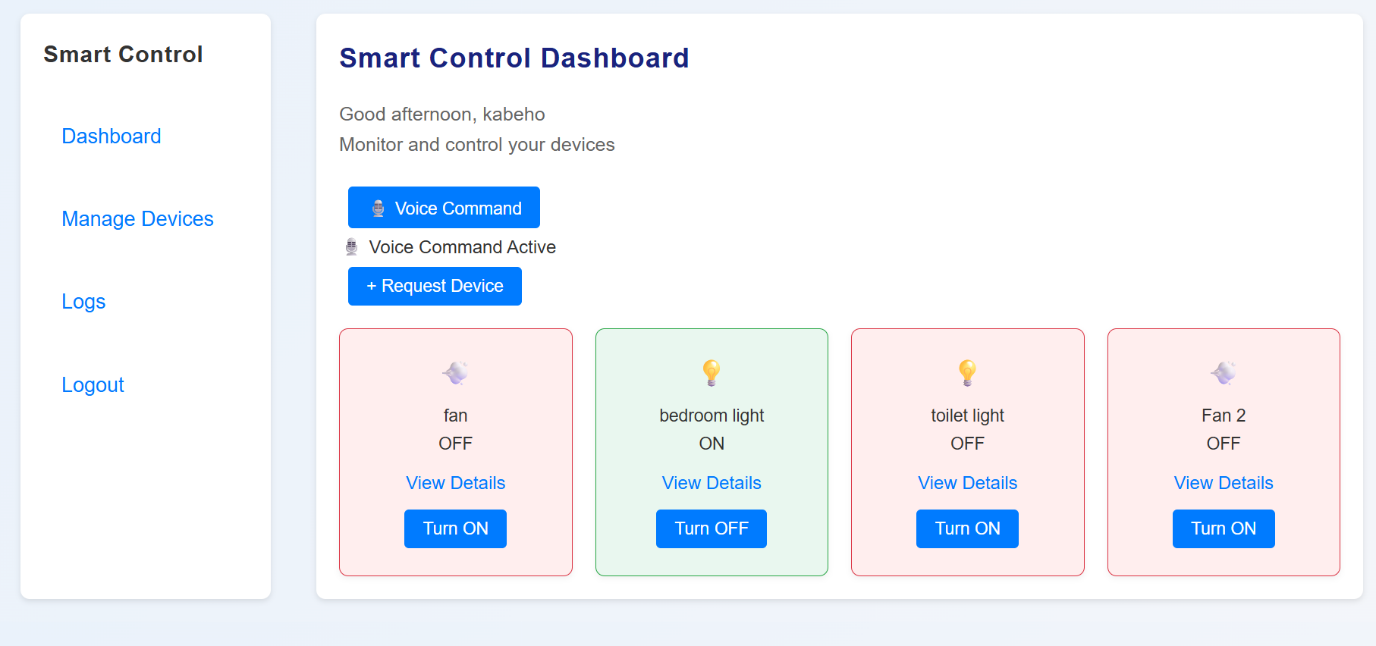


Figure 6: User Dashboard

### 5.2.6 Manage users page for Admin

This page provides functionality to view, add, edit, activate/deactivate, and delete user accounts. A search bar and filter options are included to make it easier to manage large numbers of users. Device count per user and current status (active/inactive) are clearly displayed. This section is crucial for maintaining user control and ensuring system usage.

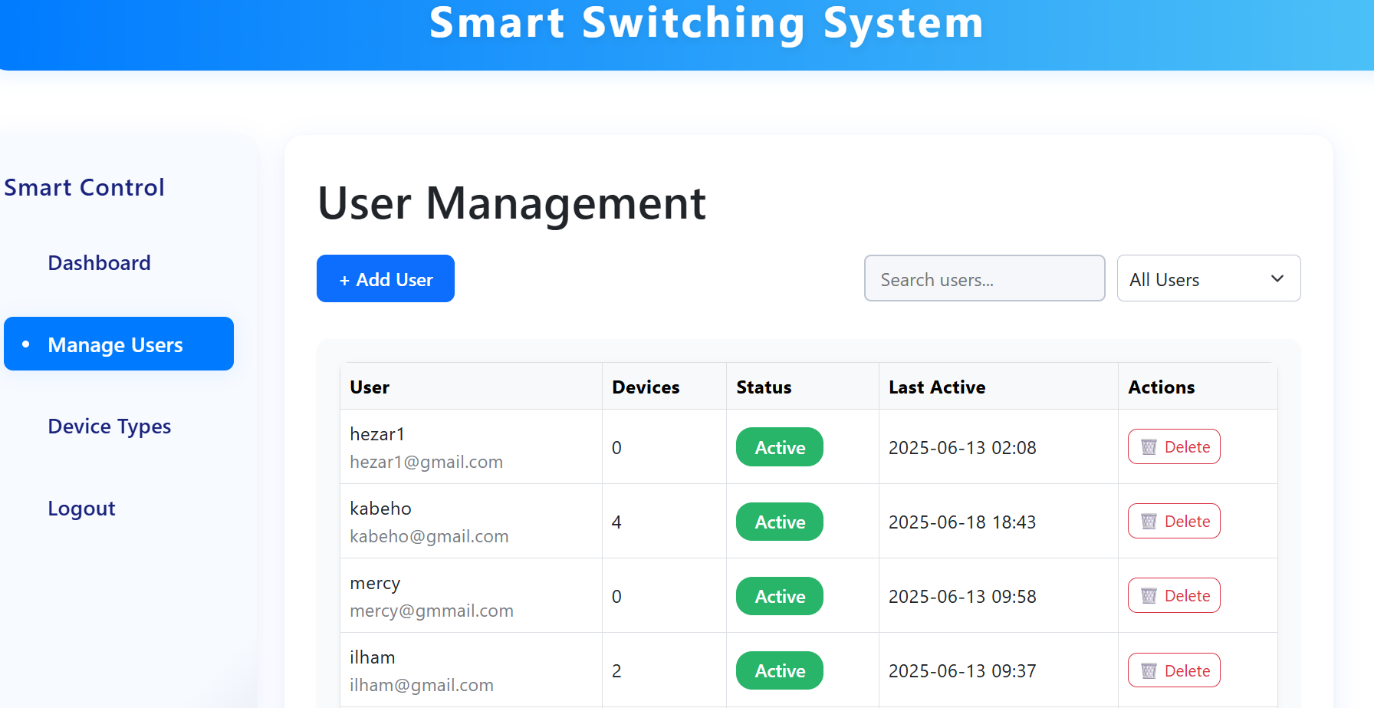


Figure 7: User Management by Admin

### 5.2.7 IoT Hardware Integration

The system uses an ESP32 microcontroller, which connects to a Wi-Fi network and interacts with the backend server. Relay modules connected to the ESP32 control the power supply to the bulbs and fan. HTTP requests were used to send ON/OFF commands, and Http was considered for future enhancements due to its lightweight, publish-subscribe messaging pattern ideal for IoT systems (Brock, 2020).

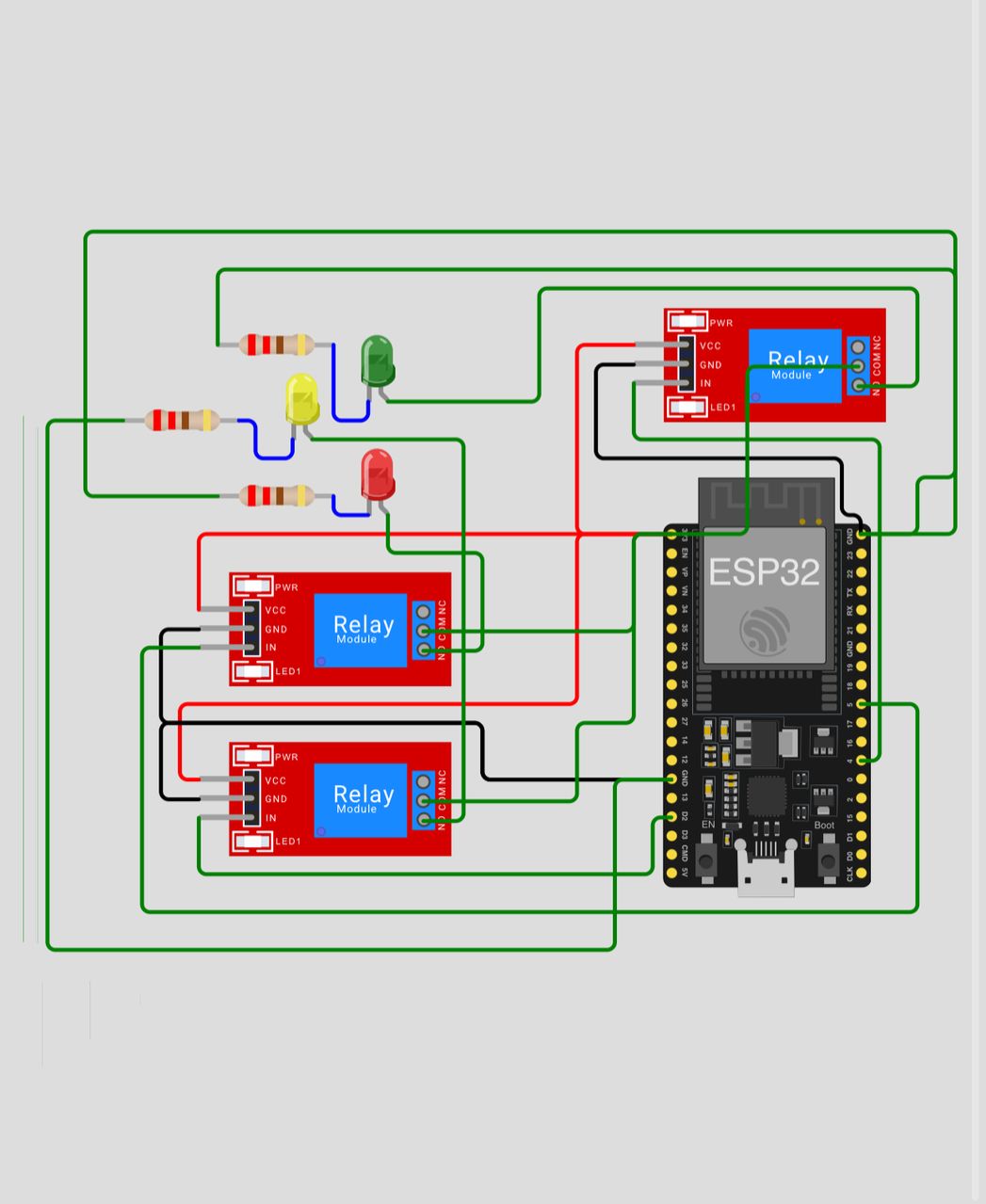


Figure 8: Connection of our devices

### 5.2.8 Voice Control

Voice control was implemented using the Web Speech API, which enables browser-based voice recognition (Mozilla, 2023). When a user speaks a command such as “turn on the fan,” the system interprets it and sends the appropriate signal to the ESP32. Additionally, the use of Google Assistant and IFTTT was considered to allow remote voice control through cloud automation (Meszaros, 2021).

### 5.2.9 Device Setup

The hardware included two 5V bulbs and one fan, each connected to individual relay channels controlled by the ESP32. The setup was built on a breadboard using jumper wires, and a 5V adapter powered the system. The ESP32 was programmed using the Arduino IDE, and commands were sent via Wi-Fi to toggle the devices.

## **5.3 Testing**

The system was tested through alpha testing, which involves developers testing the functionality internally before exposing it to real users. This method helped us identify and fix errors during early development phases (Meszaros, 2021).

### 5.3.1 Functional Testing

Functional testing focused on whether the system features worked as expected.

Table 2: Functional Requirement Table

|  |  |  |  |
| --- | --- | --- | --- |
| S/N | Functional Requirement | Expected Outcome | Test Result |
| 1 | |  | | --- | |  |  |  | | --- | | The system shall allow users to log in through a secure authentication interface to ensure access control | | User successfully logs in using correct credentials; | Pass |
| 2 | The system shall enable users to switch lights and fans ON or OFF via a web interface. | Devices toggle correctly with minimal delay when buttons are clicked. | Pass |
| 3 | |  | | --- | |  |  |  | | --- | | The system shall display and update the current status of each connected device in real time. | | Device status is reflected and accurately on the dashboard. | Pass |
| 4 | |  | | --- | |  |  |  | | --- | | The system shall support voice command input through the browser to control connected devices. | | Devices respond accurately to spoken commands via Web Speech API. | Pass |
| 5 | The system shall maintain a log of all device actions for monitoring and analytics purposes | All ON/OFF events are recorded with timestamps and records | Pass |
| 6 | |  | | --- | |  |  |  | | --- | | The system shall use Wi-Fi and HTTP protocols to synchronize communication between the frontend interface and the ESP32 microcontroller. | | Successful communication between frontend and ESP32 is established via HTTP requests over Wi-Fi. | Pass |

### 5.3.2 Integration Testing

Integration testing ensured proper communication between the frontend interface, backend server, and ESP32 hardware. No breakdowns were observed, and the commands flowed through the system smoothly (Ford et al. (2021)).

### 5.3.3 Usability Feedback

Group members tested the system and provided informal feedback. The interface was described as simple and user-friendly, and the voice command feature worked reliably in a quiet environment, the overall experience met user expectations on our system.

# CHAPTER SIX

# CONCLUSION AND RECOMMENDATION

## **6.1 Introduction**

This chapter summarizes the key outcomes of the Web and Voice-Based Home Electronic Devices Switching System. It presents the conclusions drawn from each project objective, highlights challenges faced during development and testing, and provides actionable recommendations for future enhancements and research. The project was designed to respond directly to the identified problem of energy waste and safety risks caused by unattended electrical appliances, and to offer an accessible, IoT-based solution with both web and voice controls.

## **6.2 Conclusion**

### 6.2.1 To identify and gather user requirements

User requirements were successfully collected through literature reviews and online questionnaires using Google Forms. Users expressed interest in controlling home appliances remotely, using voice commands, receiving notifications for prolonged device usage, and accessing device ON/OFF history. These needs directly informed the system design, which prioritized simplicity, real-time feedback, and dual (web and voice) control methods (Ford et al., 2021).

### 6.2.2 To design the system

The design phase produced a modular and scalable architecture, combining frontend and backend components with ESP32-based hardware. Design tools such as UML class and use case diagrams, along with the system architecture diagram, clearly outlined the relationships between modules, ensuring an organized and expandable solution (Meszaros, 2021).

### 6.2.3 To implement the system

The system was implemented using Django for backend logic, HTML/CSS/JavaScript for the user interface, and the ESP32 microcontroller for IoT integration. The system allows users to log in securely, control individual devices, issue voice commands, and monitor device status in real-time. The integration of Web Speech API allowed browser-based voice control, effectively aligning with the functional expectations of users.

### 6.2.4 To test and validate the system

The system underwent alpha testing and functional validation. All test cases—including device toggling via web, voice control, and real-time status updates—passed successfully. Integration testing confirmed proper communication between the frontend, backend, and ESP32 microcontroller. Feedback from users confirmed that the system was simple, responsive, and accessible even to individuals with limited technical skills (Django Software Foundation, 2023; Alzubi et al., 2022).

## **6.3 Challenges and Limitations**

1. **Voice Accuracy in Noisy Environments**: Voice recognition accuracy declined in environments with background noise or strong accents.
2. **Wi-Fi Dependency**: The system requires a stable Wi-Fi connection, which limits functionality in areas with poor connectivity.
3. **Hardware Constraints**: The prototype setup could only accommodate a limited number of devices due to ESP32 and relay limitations.
4. **Security Gaps**: Although user login was implemented, encryption and multi-factor authentication were not yet integrated, leaving the system vulnerable to potential threats in real deployments.

## **6.4 Recommendations**

To enhance the system’s performance, usability, and security in future versions, the following recommendations are proposed:

1. Offline Voice Support: Implement offline voice recognition libraries or modules (e.g. Pocket Sphinx) to reduce dependence on internet-based APIs.
2. Mobile Application Support: Develop a cross-platform mobile app (e.g., using Flutter or React Native) for improved accessibility and device control.
3. Scalability: Expand hardware compatibility to support more device types and greater numbers of devices, possibly using HTTPS for better communication handling.
4. Advanced Security: Introduce HTTPS for secure data transmission, along with user data encryption and multi-factor authentication.

**References**

Alam, M. R., Reaz, M. B. I., & Ali, M. A. M. (2021). A review of smart homes—Past, present, and future. IEEE Access, 9, 23565–23585. https://doi.org/10.1109/ACCESS.2021.3051812

Alzubi, J. A., et al. (2022). IoT-based Smart Home using ESP32 and Blynk App. International Journal of Interactive Mobile Technologies, 16(5), 75–83.

Brock, J. (2020). MQTT Essentials – A Lightweight IoT Protocol. Retrieved from https://mqtt.org

Chao, L. (2020). User Interface Design for Modern Applications. Springer.

Dawson, C. (2019). Projects in Computing and Information Systems: A Student's Guide (2nd ed.). Pearson Education.

Django Software Foundation. (2023). Django 4.x Documentation. https://docs.djangoproject.com/en/4.2/

Ford, N., Parsons, M., Richards, P., & Kua, P. (2021). Software Architecture: The Hard Parts. O’Reilly Media.

Kendall, K. E., & Kendall, J. E. (2020). Systems Analysis and Design (10th ed.). Pearson.

Kumar, P., & Gupta, S. (2020). Smart Home Automation System Using IoT. International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, 8(2), 45–48.

Meszaros, G. (2021). Modern Software Testing: Foundations and Practices. Addison-Wesley.

Mozilla. (2023). Web Speech API Documentation. Mozilla Developer Network. Retrieved from https://developer.mozilla.org

Patel, N., & Chauhan, R. (2019). Smart Switch Control System Using Arduino with Android App. International Journal of Engineering Research & Technology (IJERT), 8(6), 59–63.

Patel, P. (2019). Smart Home Automation Using IoT. International Journal of Engineering Research & Technology (IJERT), 7(4), 104–108.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). John Wiley & Sons.

APPENDIX

QUESTIONS USED IN DATA COLLECTIONS

