# Kaushal Kishore Rai

Atlantis University

MIT 563: Programming and Applications Development

Fall “A” 2024

Professor: Alex Lima

10/06/2024

Final Project

A) Business Requirements for Home Automation System

# 1. Introduction

The Home Automation System is designed to provide an integrated and comprehensive solution for managing and automating various devices within a household environment. This system enables users to control their devices remotely, automate device operations based on predefined schedules, and monitor device activities for convenience, security, and energy efficiency.

# 2. Business Objectives

A diagram of a diagram

Description automatically generated

The primary objective of the Home Automation System is to enhance the user’s living experience by enabling the automation and remote control of smart devices in their home. The system is designed to be scalable, user-friendly, and customizable to meet the diverse needs of different households. Key objectives include:

* • Automating household devices (e.g., lights, thermostats, cameras, etc.).
* • Providing remote control via mobile or web interfaces.
* • Optimizing energy consumption through automated schedules.
* • Ensuring security through smart locks, cameras, and alert systems.
* • Allowing users to monitor and manage devices across multiple rooms.
* • Generating logs and reports of device activities for user transparency.

# 3. Scope of the System

The Home Automation System will support the following functionalities:

* • Device control: Users can control devices such as lights, thermostats, cameras, and smart plugs.
* • Scheduling: Users can set schedules for device operations (e.g., turning on/off lights at certain times).
* • Monitoring: The system will track and display the status of devices and their activities.
* • User management: Multiple users with different access levels (e.g., admin, user) can be created.
* • Security management: Alerts for security breaches, and integration with smart locks and cameras.
* • Energy management: Monitoring and optimizing energy usage of devices.

# 4. Stakeholders

The following stakeholders are identified for the Home Automation System:

* • Homeowners: Primary users who will interact with the system to control their home devices.
* • Administrators: Users responsible for managing the system, configuring devices, and setting up schedules.
* • Technical Support: Personnel responsible for troubleshooting and maintaining the system.

# 5. Functional Requirements

The Home Automation System should provide the following functionalities:

* • Ability to add, remove, and configure devices.
* • Real-time device monitoring and status updates.
* • Scheduling functionality for automating device actions.
* • User authentication and role-based access control.
* Logs and reports for device activities and user interactions.

# 6. Non-Functional Requirements

The system should meet the following non-functional requirements:

* • Usability: The system should provide a user-friendly interface for easy navigation.
* • Scalability: The system should support multiple users and a wide range of devices.
* • Security: The system must ensure secure communication between devices and the application, including user data protection.
* • Performance: The system should provide real-time responses to user commands and maintain optimal performance during high traffic.

# 7. Diagrams

The following diagrams should be included to illustrate the architecture and workflows of the Home Automation System:

* Use Case Diagram: To depict the interactions between users and the system.

A diagram with text and arrows

Description automatically generated with medium confidence

* **Explanation:**
* **Homeowner**: Interacts with the **Website** to schedule a device (e.g., Light) to turn on.
* **Website**: Receives the scheduling request, stores the schedule in the **Database**, and confirms back to the **Homeowner**.
* **Database**: Stores the schedule details.
* **Device**: Receives the command from the website to turn on at the scheduled time, then sends a confirmation back to the website.

# 8. Conclusion

The Home Automation System aims to provide a seamless and efficient way for users to control and manage their home devices. With features such as scheduling, real-time monitoring, and security management, the system will enhance the convenience and comfort of users while promoting energy efficiency. The integration of various smart devices into one system will allow for centralized control, ensuring an enhanced living experience.

B) High-Level Design of the Solution

The high-level design of the Home Automation System focuses on the overall architecture of the system. The key components of the system include the user interface (UI), backend services, the database, and the IoT devices. The user interface will be a web or mobile application through which users (homeowners, admins) can interact with the system to control devices, monitor statuses, and set schedules. The backend services will handle the business logic and communicate with the database and devices. The system will use APIs to communicate with IoT devices, such as lights, thermostats, and security cameras. Data related to user actions, device statuses, and schedules will be stored in a PostgreSQL database. The system will be scalable to accommodate new devices and users.

**High Level Diagram-**

A diagram of a computer system

Description automatically generated

This **Architecture Diagram** for a **Home Automation System** shows how data flows between different components. The **portal** or web interface, accessible via mobile or desktop, allows users to control and monitor IoT devices. The communication passes through **Azure Cloud**, which includes a PostgreSQL database for storing data and an **Azure IoT Hub** for managing device interactions. The IoT Hub handles communication between the cloud and devices, with **Cloud-to-Device (C2D)** and **Device-to-Cloud (D2C)** messaging. Commands are routed through a **Firewall home router**, ensuring security, before reaching home IoT devices like lights, thermostat, home security camera, alarm system, and doorbell notifications.

**Data flow diagram**

A diagram of a software process

Description automatically generated

A **Data Flow Diagram (DFD)** visually represents the flow of data within a system. In the context of the **Home Automation System**, it shows how data moves between different entities, such as users, the website, the database, and IoT devices. The DFD focuses on illustrating the interactions and the transfer of information across components rather than their physical structure.

**Key Components of the Data Flow Diagram:**

1. **External Entities (Users)**
   * **Homeowner**: The primary user who interacts with the system to control devices, schedule activities, and monitor their status.
   * **Admin**: A user who has privileges to manage the system, including user management, logs, and device control.
   * These users send requests (commands) to the system and receive responses (device status, confirmation of actions).
2. **Processes (System Components)**
   * **Website Interface**: The frontend through which the homeowner and admin interact with the system. The website acts as a mediator between users and the backend logic.
   * **Backend Logic**: The core processing unit that handles requests from the website, processes them, and communicates with the database and IoT devices. It processes user actions, fetches the data, and sends it back to the users.
   * **IoT Devices**: Devices like lights, thermostats, and security cameras that receive commands (e.g., turn on/off, adjust settings) from the backend. The devices send status updates back to the backend.
3. **Data Stores**
   * **PostgreSQL Database**: Stores data related to users, devices, schedules, and logs. It manages user accounts, device configurations, schedule settings, and logs every action (e.g., a user turning on a light).

**Explanation of the Data Flow:**

1. **User Requests (Control and Scheduling)**
   * The **Homeowner** or **Admin** initiates a request from the **Website** to control devices or set schedules.
   * The website forwards the request to the **Backend Logic**, which processes the request.
2. **Processing the Request**
   * The **Backend Logic** interacts with the **PostgreSQL Database** to retrieve relevant information (e.g., device status, user settings).
   * For scheduling, the backend stores the schedule in the database and sets up timers or events to execute at the specified time.
3. **Device Control and Feedback**
   * The **Backend Logic** sends the control command to the appropriate **IoT Device** (e.g., turning a light on or adjusting the thermostat).
   * The **IoT Device** executes the command and sends a confirmation back to the **Backend**.
4. **Real-Time Monitoring**
   * The system constantly updates device statuses in the database. The **Backend** retrieves the latest data from the devices and updates the **Website** in real-time so that users can monitor their devices' status.
5. **Log Storage**
   * Every action (device control, scheduling) is logged in the **PostgreSQL Database**. The **Admin** can access these logs through the website interface to track system usage and ensure proper functioning.

**Example Data Flow for "Turn On Light":**

1. The **Homeowner** clicks the "Turn On" button for a light on the **Website**.
2. The **Website** sends the request to the **Backend**.
3. The **Backend** queries the **PostgreSQL Database** to ensure the user has the right permissions and retrieves the current device status.
4. The **Backend** sends a "Turn On" command to the **Light**.
5. The **Light** responds with a confirmation.
6. The **Backend** updates the status in the **PostgreSQL Database** and sends the updated status back to the **Website**.
7. The **Homeowner** sees the light's status as "On" on the website.

**Visual Representation in the DFD:**

* **Processes** are represented by circles or ovals (e.g., Website, Backend, IoT Device).
* **Data Stores** (like the database) are represented by two horizontal lines.
* **External Entities** (like users) are shown as squares or rectangles.
* **Data Flows** between entities, processes, and data stores are depicted with arrows, indicating the direction of the data movement.

In the Home Automation System, the **DFD** helps in understanding how the system processes user requests, handles data, and communicates with IoT devices. It provides a clear, high-level view of the interactions and helps in identifying key components and data exchanges in the system.

C) Detailed Solution Design

In the detailed solution design, we break down each system component and describe how they work together. The frontend will be developed using HTML, CSS, and JavaScript for the web interface. The backend will be built using Flask (Python), which will handle user requests, interact with the database, and send commands to devices. PostgreSQL will be used to store user data, schedules, and logs. IoT devices will communicate with the backend using secure protocols (such as Wi-Fi, Zigbee, or Bluetooth). Devices will be identified and controlled via unique IDs, and the backend will manage the execution of user commands. Real-time updates will be pushed to the frontend, allowing users to monitor device statuses in real-time. Additionally, user authentication and role-based access control will ensure that only authorized users can perform certain actions.  
  
Scope for diagrams:   
• Component Diagram   
• Sequence Diagrams for User Actions   
• Class Diagrams

D) Data Requirements

The data requirements for the Home Automation System include storing user profiles, device configurations, schedules, and system logs. The system will store details for each device, including its type, status, and room location. Each schedule will record the time and duration for which a device is expected to operate. Logs will track user actions, device changes, and any errors that occur. The database will need to be scalable to handle multiple devices and users, and backups will be implemented for disaster recovery. PostgreSQL will be used as the database management system due to its reliability and support for complex queries.  
  
Scope for diagrams:   
• Entity-Relationship Diagram (ERD)   
• Database Schema

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Initial Database scripts**

-- Create the home\_automation database

CREATE DATABASE home\_automation;

-- Switch to the home\_automation database

\c home\_automation;

-- Create the rooms table

CREATE TABLE rooms (

id SERIAL PRIMARY KEY,

room\_name VARCHAR(50) NOT NULL,

floor INT

);

-- Create the devices table

CREATE TABLE devices (

id SERIAL PRIMARY KEY,

device\_name VARCHAR(50) NOT NULL,

device\_type VARCHAR(50) NOT NULL, -- Light, Thermostat, Camera, etc.

status VARCHAR(20) DEFAULT 'off', -- on/off

room\_id INT,

connection\_type VARCHAR(20), -- Wi-Fi, Zigbee, Bluetooth, etc.

battery\_level INT, -- Battery percentage (if applicable)

energy\_usage DECIMAL(5,2), -- Energy usage in kWh (if applicable)

last\_activity TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT fk\_room

FOREIGN KEY(room\_id)

REFERENCES rooms(id)

);

-- Create the schedules table

CREATE TABLE schedules (

id SERIAL PRIMARY KEY,

device\_id INT,

start\_time TIMESTAMP,

end\_time TIMESTAMP,

CONSTRAINT fk\_device

FOREIGN KEY(device\_id)

REFERENCES devices(id)

);

-- Create the users table

CREATE TABLE users (

id SERIAL PRIMARY KEY,

username VARCHAR(50) NOT NULL,

password VARCHAR(50) NOT NULL, -- In production, store hashed passwords

role VARCHAR(20) DEFAULT 'user' -- admin/user

);

-- Create the logs table

CREATE TABLE logs (

id SERIAL PRIMARY KEY,

device\_id INT,

user\_id INT,

action VARCHAR(50), -- e.g., 'turn on', 'turn off'

timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT fk\_device\_log

FOREIGN KEY(device\_id)

REFERENCES devices(id),

CONSTRAINT fk\_user\_log

FOREIGN KEY(user\_id)

REFERENCES users(id)

);

-- Insert sample data into rooms

INSERT INTO rooms (room\_name, floor) VALUES

('Living Room', 1),

('Bedroom', 2),

('Kitchen', 1),

('Garage', 1);

-- Insert sample data into devices

INSERT INTO devices (device\_name, device\_type, status, room\_id, connection\_type, battery\_level, energy\_usage) VALUES

('Smart Light Bulb', 'Light', 'off', 1, 'Wi-Fi', NULL, 0.05),

('Thermostat', 'Thermostat', 'off', 2, 'Wi-Fi', NULL, 0.10),

('Security Camera', 'Camera', 'on', 3, 'Wi-Fi', 85, 0.03),

('Smart Plug', 'Plug', 'on', 4, 'Bluetooth', NULL, 0.08);

-- Insert sample data into schedules

INSERT INTO schedules (device\_id, start\_time, end\_time) VALUES

(1, '2024-10-07 06:00:00', '2024-10-07 08:00:00'),

(2, '2024-10-07 09:00:00', '2024-10-07 18:00:00'),

(3, '2024-10-07 18:00:00', '2024-10-07 22:00:00');

-- Insert sample data into users

INSERT INTO users (username, password, role) VALUES

('admin', 'adminpassword', 'admin'),

('user1', 'userpassword', 'user'),

('user2', 'password123', 'user');

-- Insert sample data into logs

INSERT INTO logs (device\_id, user\_id, action) VALUES

(1, 1, 'turn on'),

(2, 2, 'turn off'),

(3, 1, 'turn on'),

(4, 3, 'turn off');

-- End of script

E) High-Level Program (The Logic)

At a high level, the program logic revolves around handling user commands and translating them into device actions. The main operations include turning devices on or off, adjusting settings, and scheduling actions to happen automatically at certain times. Users interact with the system through the website, which sends requests to the backend. The backend processes these requests and communicates with the IoT devices using predefined protocols. The system will also handle conflict resolution for overlapping schedules, ensuring that device commands do not interfere with each other. Additionally, user authentication and authorization will determine the level of access users have.  
  
Scope for diagrams:   
• Flowcharts depicting high-level logic

F) Detailed Program (Detailed Logic)

The detailed program logic will define how each function of the Home Automation System is executed. This includes how device control commands are issued, how schedules are stored and executed, and how logs are generated. For example, when a user sets a schedule for a device, the system will store the schedule in the database and trigger an event at the specified time. Device status updates will be fetched in real-time and displayed to the user. Error handling will be included to manage potential failures, such as device disconnections or invalid user inputs. Additionally, performance optimizations will be made to ensure that the system remains responsive even when managing many devices.  
  
Scope for diagrams:   
• Detailed Sequence Diagrams

G) Complete Program

The complete program includes all the necessary components for the system to function, including the frontend, backend, database, and device integration. The frontend (built using HTML, CSS, and JavaScript) provides the interface for users to interact with devices and set schedules. The backend (developed with Flask) handles all business logic, user authentication, and database interactions. The database (PostgreSQL) stores information about users, devices, schedules, and logs. Devices will communicate with the backend using secure protocols to execute commands in real-time. Together, these components provide a fully functional home automation system.

H) Results from Running the Program

After running the program, the user will be able to control their home devices from the website interface. The system will successfully execute schedules, automatically turning devices on and off based on user-defined times. Real-time status updates will allow users to monitor device operations. Logs will be generated, tracking every interaction with the system, and will be available for review by administrators. Users will also receive notifications when device status changes, enhancing the overall user experience.

I) Considerations about the Solution Proposed

The proposed solution must account for scalability, security, and reliability. The system must support adding new devices and users without significant performance degradation. Security measures, such as encryption for data transmission and secure authentication, are critical to prevent unauthorized access. The system should be designed to handle network interruptions gracefully, allowing devices to reconnect automatically when the network is restored. Additionally, the user interface must be intuitive, ensuring a smooth experience for all users.

J) Limitations and Further Developments

One limitation of the current system is its reliance on a stable network connection. In areas with poor internet connectivity, device commands may experience delays or failures. Another limitation is that the system is designed for common household devices and may not support specialized devices without additional integration efforts. Future developments could include integration with voice-controlled systems (e.g., Alexa, Google Assistant), advanced analytics for energy usage optimization, and the use of machine learning to predict user behavior.

K) General Ideas about Software Quality Management

Software quality management will involve continuous testing, code reviews, and monitoring of system performance. Automated tests will be run during each build to catch bugs early, while manual tests will be used to verify new features. Code quality will be ensured through peer reviews, and security audits will be conducted regularly to identify potential vulnerabilities. Performance metrics, such as response time and system uptime, will be tracked to ensure the system meets user expectations.

L) General Ideas about How to Test the Solution

Testing will cover all aspects of the system, including functionality, performance, security, and usability. Unit tests will verify the correctness of individual components, while integration tests will ensure that the components work together seamlessly. Load testing will simulate high user traffic to ensure the system remains responsive under stress. Security tests will check for vulnerabilities, such as SQL injection or unauthorized access. User acceptance testing (UAT) will ensure the system meets user requirements.

M) General Ideas about Implementation

The system will be implemented in a series of iterative development cycles following Agile methodology. Each cycle will focus on developing a specific set of features, which will be tested and reviewed before moving on to the next set. Continuous integration and deployment (CI/CD) pipelines will be used to automate the build and testing process. The system will be deployed locally for initial testing, and once stable, it can be deployed on cloud infrastructure for production use.

N) Conclusion

The Home Automation System is designed to provide a convenient and efficient way for users to control their household devices. By integrating a scalable architecture, secure communication, and an intuitive user interface, the system offers a complete solution for modern home automation. Future developments will further enhance the system's capabilities, ensuring it remains relevant and useful as new technologies emerge.

O) References

1. Flask Documentation: https://flask.palletsprojects.com/

2. PostgreSQL Documentation: https://www.postgresql.org/docs/

3. Zigbee Protocol: https://www.zigbee.org/

4. Agile Methodology: https://www.agilealliance.org/agile101/

5. CI/CD Pipelines: https://aws.amazon.com/devops/continuous-integration/