ZENTRA – SMART HOME AUTOMATION SYSTEM

Chapter 1: Introduction

1.1 Overview of the Project

Zentra is a modern IoT-based Smart Home Automation System that enables seamless, real-time control and monitoring of smart devices from a centralized web dashboard. Designed with a futuristic visual theme, Zentra integrates real-time communication using MQTT, a powerful backend using Spring Boot, and an elegant frontend built with React JS. The system supports device toggling, scheduling, scene management, and IoT device simulation.

1.2 Problem Statement

Traditional home appliances require manual interaction and lack centralized control. Existing smart home systems are often proprietary, expensive, and not customizable. Users need an affordable, customizable, and scalable system that offers real-time monitoring and automation.

1.3 Objectives

- Provide a unified dashboard to control home appliances.
- Enable real-time device updates using MQTT.
- Implement secure login and authentication using JWT.
- Support device scheduling and automation routines.
- Ensure scalability for adding unlimited devices.
- Deliver a visually modern and responsive interface.

1.4 Scope

The project covers: - Smart device control (lights, AC, fans, simulated devices). - MQTT real-time communication. - Scheduling system with repeat options. - Secure backend with JWT authentication. - Device simulator using Python. - Cross-device responsive design.

Chapter 2: System Analysis

2.1 Existing System

Manual switching of devices lacks automation and remote control capability. No centralized interface exists, and users have no real-time feedback about device status. Commercial IoT systems are costly and limited in customization.

2.2 Proposed System

Zentra provides a customizable smart home automation platform featuring: - Instant device response via MQTT. - Central dashboard for all devices. - Scheduling and scenes. - Secure authentication and modular device management.

2.3 Feasibility Study

Technical Feasibility: Built with widely used technologies (React, Spring Boot, MySQL, MQTT).
Operational Feasibility: Intuitive UI ensures easy operation.
Economic Feasibility: Free and open-source tools minimize cost.

Chapter 3: System Design

3.1 Architecture Overview

- **Frontend:** React JS application.
- **Backend:** Spring Boot REST API.
- **Database:** MySQL for storing users, devices, and schedules.
- **Communication:** MQTT for real-time updates.
- **IoT Simulation:** Python scripts mimicking device responses.

3.2 Module Breakdown

3.2.1 Authentication Module

- User registration & login.
- JWT token-based authentication.

3.2.2 Dashboard Module

- Display of widgets (Weather, Fan, AC, Scenes).
- Welcome user info.
- Live device count.

3.2.3 Device Management Module

- Add, delete, and toggle devices.
- View all devices.
- MQTT real-time status updates.

3.2.4 Schedule Module

- Create schedules with repeat days.
- Assign multiple actions to schedules.

3.2.5 MQTT Module

- Live subscription to device status topics.
- Publishing device commands.

Chapter 4: System Implementation

4.1 Frontend (React)

- Built using functional components and hooks.
- Axios used for API calls.
- MQTT.js used for real-time updates.
- Pages: Login, Register, Dashboard, Devices, Schedules, Settings.
- Reusable UI components for cards, widgets, and modals.
- Modern UI styling using CSS animations, transitions, and neon highlights.

4.2 Backend (Spring Boot)

- REST controllers for devices, schedules, and users.
- JWT authentication handled via filters.
- MySQL persistence with Spring JPA.
- MQTT publisher service for device toggle commands.
- Schedule execution logic.

4.3 IoT Simulation (Python)

- Python script using Paho MQTT client.
- Simulates real device feedback by publishing ON/OFF state.
- Subscribes to command topics from backend.

Chapter 5: Database Design

5.1 Database Used

MySQL database manages persistent storage for users, devices, schedules, and actions.

5.2 Key Tables

- **users** – id, email, password, roles
- **devices** – id, name, room, type, status
- **schedules** – id, name, time, repeatDays
- **schedule_actions** – id, scheduleId, deviceId, action

5.3 Relationships

• One user can have many devices.
• One schedule can contain multiple device actions.

Chapter 6: Testing

6.1 Unit Testing

• Tested API endpoints using Postman.
• Verified JWT login cookie and header authentication.
• MQTT message tests for publish/receive.

6.2 Functional Testing

• Tested device toggle UI.
• Verified schedule creation and updates.
• Confirmed responsiveness on mobile and desktop.

6.3 Integration Testing

• Frontend ↔ Backend API link validation.
• Backend ↔ MQTT broker message tests.
• End-to-end simulation with Python devices.

Chapter 7: Output Screenshots

(Add your actual screenshots here) - Login Page - Dashboard - Devices Page - Add Device Modal - Create Schedule Page - MQTT Logs

---

Chapter 8: Conclusion

Zentra Smart Home Automation System demonstrates the seamless merging of IoT, web technologies, and real-time communication. It successfully provides a centralized, responsive, and futuristic interface for controlling home devices. MQTT integration ensures instant feedback, and the modular approach allows easy scalability.

Chapter 9: Future Enhancements

• Mobile application version (Android/iOS).
• Voice assistant integration.
• AI-based energy usage analytics.

- Geolocation-based automations.
- Integration with smart sensors (temperature, motion).

References

- MQTT Documentation
- ReactJS Documentation
- Spring Boot Documentation
- MySQL Documentation
- Paho MQTT Library

End of Report