Project Report

1. Introduction This project presents the design and implementation of a smart home automation system focusing on intelligent lighting and fan control. With increasing demand for energy efficiency and user convenience, the proposed system offers a centralized, IoT-enabled solution that automates climate and lighting controls based on environmental and user-specific factors.

2. Objectives To automate lighting and fan systems based on real-time environmental data. To minimize energy consumption through AI-driven adaptive algorithms. To provide a modular, scalable architecture that supports user preferences and third-party integrations. To ensure seamless user interaction via mobile interfaces and voice control (Amazon Alexa).

3. System Architecture The system architecture is built around the following core components:

a. ESP32 Microcontrollers Act as the core embedded hardware. Interface with lights, fans, sensors, and cloud services. Communicate via the MQTT protocol for low-latency data transfer.

b. MQTT Communication Protocol Enables lightweight, real-time communication between devices and servers. Ensures reliable messaging for control signals and sensor data.

c. Sensors PIR motion sensors detect room occupancy. DHT22 temperature sensors monitor room temperature. d. Control Elements Dimmable LED lights allow brightness and colour temperature control. Variable-speed fans adjust airflow based on temperature.

4. Intelligent Automation Logic The system uses AI-based automation rules for intelligent control:

Occupancy-Based Lighting: Lights dim or turn off when no motion is detected. Temperature-Based Fan Speed: Fan speed adjusts according to ambient temperature. Time-Based Automation: Automated transitions such as dimming lights at sunset and increasing fan speed during peak heat hours.

5. Technologies Used:

The Smart Home Automation Web App is built using a modern, real-time, and AI-powered tech stack. On the frontend, it uses React for UI, Tailwind CSS for styling, Shadcn UI for customizable components, and Wouter for routing. Form handling and validation are managed by React Hook Form and Zod, while React Query ensures efficient data fetching and caching. Visual elements are enhanced with Lucide React icons, Recharts for data visualization, Framer Motion for animations, and React Day Picker for calendar inputs. Backend operations are powered by Express.js with PostgreSQL as the database and Drizzle ORM for schema management. Real-time communication with IoT devices is handled via MQTT and WebSockets. The app also integrates the OpenAI API to enable natural language commands, energy-saving suggestions, and intelligent automation rules. Development tools like TypeScript, Vite,

ESBuild, and Drizzle Kit ensure a fast and scalable development process. Utilities like date-fns, clsx/tailwind-merge, and drizzle-zod further streamline logic, styling, and validation. Key highlights include: real-time device control, AI integration, a responsive and elegant UI, and a scalable backend architecture.

6. User Interface

a. Mobile/Web Dashboard Allows users to manually control lights and fans. View real-time device status and sensor data. Schedule or override automation rules. b. Voice Control Integration Full compatibility with Amazon Alexa for hands-free operation. Voice commands control individual devices or activate automation scenes.

7. Backend System

a. Database Design MySQL relational database with modular schema. Tables include: Users, Devices, Rooms, Schedules, AutomationRules, and DeviceAssignments. Stores user preferences, room mappings, and automation logic. b. Python Backend Core logic for user authentication (signup/login), device management, and rule processing. Interfaces with MySQL and ESP32 via MQTT