B.Tech. BCSE497J - Project-I

IntelliNest: Smart Living Redefined

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering in Internet of Things

by

21BCT0358 – SURAJ JHA

21BCE0211- VIVEK PRASAD

21BDS0192 - RAGHAV AGARWAL

Under the Supervision of

Dr. Gladys Gnana Kiruba B

Assistant Professor Sr. Grade 1

Department of Software Systems

School of Computer Science and Engineering (SCOPE)

November 2024

DECLARATION

I hereby declare that the project entitled “IntelliNest: Smart Living Redefined” submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering in Internet of Things* to VIT is a record of bonafide work carried out by me under the supervision of Prof. Dr. Gladys Gnana Kiruba B.

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place : Vellore

Date : 13/11/2024

Suraj Jha.

Signature of the Candidate

CERTIFICATE

This is to certify that the project entitled “IntelliNest: Smart Living Redefined” submitted by SURAJ JHA Reg No. 21BCT0358 **School of Computer Science and Engineering** ,VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering in Internet of Things*, is a record of bonafide work carried out by him under my supervision during Fall Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The project fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 13/11/2024

[illegible]
13/11

Signature of the Guide

[illegible]

Examiner(s)

Dr. SHARMILA BANU K
B. TECH in Computer Science Engineering with specialization in IoT

ACKNOWLEDGEMENTS

I am deeply grateful to the management of Vellore Institute of Technology (VIT) for providing me with the opportunity and resources to undertake this project. Their commitment to fostering a conducive learning environment has been instrumental in my academic journey. The support and infrastructure provided by VIT have enabled me to explore and develop my ideas to their fullest potential.

My sincere thanks to Dr. Ramesh Babu K, the Dean of the School of Computer Science and Engineering (SCOPE), for his unwavering support and encouragement. His leadership and vision have greatly inspired me to strive for excellence. The Dean's dedication to academic excellence and innovation has been a constant source of motivation for me. I appreciate his efforts in creating an environment that nurtures creativity and critical thinking.

I express my profound appreciation to Dr. Sharmila Banu K, the Head of the Computer Science Department, for her insightful guidance and continuous support. Her expertise and advice have been crucial in shaping the direction of my project. The Head of Department's commitment to fostering a collaborative and supportive atmosphere has greatly enhanced my learning experience. Her constructive feedback and encouragement have been invaluable in overcoming challenges and achieving my project goals.

I am immensely thankful to my project supervisor, Dr. Gladys Gnana Kiruba B, for her dedicated mentorship and invaluable feedback. Her patience, knowledge, and encouragement have been pivotal in the successful completion of this project. My supervisor's willingness to share her expertise and provide thoughtful guidance has been instrumental in refining my ideas and methodologies. Her support has not only contributed to the success of this project but has also enriched my overall academic experience.

Thank you all for your contributions and support

Name of the Candidate

Suraj Jha .

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Full Form
IoT	Internet of Things
HVAC	Heating, Ventilation, and Air Conditioning
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
SSL/TLS	Secure Sockets Layer/Transport Layer Security
MFA	Multi-Factor Authentication
UI	User Interface
SPA	Single-Page Application
API	Application Programming Interface
EHR	Electronic Health Records
ML	Machine Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MRI	Magnetic Resonance Imaging
CT	Computed Tomography
DFD	Data Flow Diagram
PSU	Power Supply Unit
OTA	Over-the-Air
EDA	Exploratory Data Analysis

ABSTRACT

The Internet of Things (IoT) has revolutionized home automation, offering enhanced control, efficiency, and convenience in managing household devices. The IoT-Based Smart Home Control Switch is a system designed to allow users to remotely control and monitor home appliances via the internet. It leverages the ESP8266 microcontroller and a 4-channel relay module to enable seamless interaction between the user's mobile or web application and home devices. The system provides real-time status updates, secure communication, and energy-efficient control, making it a comprehensive solution for modern smart homes.

At the heart of this system is the ESP8266 microcontroller, known for its integrated Wi-Fi capabilities, affordability, and processing power. The microcontroller connects to a local Wi-Fi network and responds to user commands sent via MQTT, activating or deactivating connected appliances. The 4-channel relay module works in conjunction with the ESP8266, allowing for the safe and efficient control of high-power appliances such as lights, fans, and air conditioners.

Key features of the Smart Home Control Switch include remote control, real-time monitoring, scheduling, scalability, and security. Users can manage appliances from anywhere, receiving real-time updates on their operational status. The scheduling feature enables automation, allowing users to set appliances to turn on or off based on their routines, promoting energy efficiency. The system is designed to be easily scalable by adding additional relay modules to control more devices, making it suitable for both small homes and larger environments.

Security is a critical component of the system, incorporating SSL/TLS encryption for secure communication between the user and the ESP8266. Multi-factor authentication can also be integrated to ensure only authorized users can control the appliances. Cost-effectiveness is a priority, with the system targeting a retail price between ₹500 and ₹750. This makes it accessible to a wide consumer base while maintaining high functionality. Overall, the IoT-Based Smart Home Control Switch offers a flexible, secure, and affordable solution for home automation, contributing to improved convenience, security, and energy savings in everyday life.

1. INTRODUCTION

The ongoing technological revolution and the Internet of Things (IoT) are profoundly reshaping home automation, making it more accessible, efficient, and innovative than ever before. In today's society, smart homes are no longer considered futuristic but are quickly becoming integral to modern living. These systems enable users to remotely manage essential household operations such as lighting, security, heating, ventilation, and air conditioning (HVAC) systems, allowing for unprecedented control over personal living environments. This shift towards smart home technology not only elevates convenience but also promotes energy efficiency and enhanced security, making the smart home experience both practical and enjoyable.

Our project, titled "IntelliNest: Smart Living Redefined," seeks to design and deploy a cost-effective, user-friendly system that enables users to control various home appliances remotely via the internet. At the heart of IntelliNest is the ESP8266 microcontroller, a highly efficient, low-cost microcontroller that includes Wi-Fi capabilities, enabling robust communication between connected devices and the user interface. This system aims to empower users with control over their home environment, enabling the manipulation of appliances such as lights, fans, and security systems through an intuitive web-based or mobile application. IntelliNest demonstrates how accessible technology, like the ESP8266, can serve as a powerful foundation for addressing everyday needs in smart home environments. By focusing on wireless communication, minimal hardware, and high adaptability, IntelliNest not only addresses immediate user needs but also provides an excellent foundation for future upgrades, including advanced features like voice control, energy monitoring, and enhanced security. The IntelliNest project showcases the transformative potential of IoT by converting standard homes into interconnected, responsive living spaces.

1.1 Background

Home automation technology has undergone significant evolution, transitioning from simple, remote-controlled gadgets to fully interconnected systems capable of managing entire households. The rise of IoT has been central to this transformation, providing a means for household devices to communicate with each other and the user through internet connectivity. This technological leap has laid the groundwork for the development of smarter, more efficient, and highly user-friendly automation solutions. The IoT revolution has also altered our interaction with living spaces, making intelligent homes — once confined to science fiction — a tangible reality. These smart environments offer substantial benefits, such as heightened comfort, improved energy efficiency, and greater operational control. However, even with these advancements, there remains a substantial gap between the theoretical potential of smart home technology and its broad acceptance and integration within the consumer market. Many existing solutions face limitations due to high costs, complicated installation processes, and issues with compatibility with existing home infrastructure. These obstacles limit the widespread adoption of smart home technologies, leaving large segments of the market underserved.

1.2 Motivations

The IntelliNest: Smart Living Redefined project was conceived with the mission of addressing the limitations of existing smart home solutions and democratizing access to smart technology for a wider audience. Several key motivations drive this project. Firstly, affordability is a primary focus. By developing a cost-effective solution, IntelliNest aims to bridge the gap between high-end, costly systems and the budget constraints of average homeowners. Secondly, we prioritize simplicity, creating a system that requires minimal technical knowledge for installation and use, thereby lowering the barrier to entry for first-time smart home adopters. Additionally, IntelliNest places a strong emphasis on energy efficiency. Our system aims to provide users with tools to monitor and optimize their energy consumption, promoting sustainable living practices and contributing to both cost savings and environmental conservation.

Another critical motivation is enhanced control. IntelliNest enables users to exercise greater control over their living environment, allowing remote management and automation of appliances and household systems. Scalability is also essential; the system is designed to accommodate evolving user needs, from simple automation to comprehensive home integration as desired. Security is another priority. With the increasing concern over cybersecurity in IoT devices, IntelliNest incorporates robust security measures to protect user privacy and data, employing encryption protocols and multi-factor authentication. Finally, innovation drives us to improve upon existing market solutions by introducing unique features and contributing to the overall advancement of smart home technology.

1.3 Scope of the Project

The IntelliNest project aims to develop an IoT-based Smart Home Control Switch, serving as the cornerstone for home automation. This project’s scope covers various critical areas, including hardware development, software development, connectivity, user interface design, and security. The primary hardware component, the ESP8266 microcontroller, is compact and energy efficient. It will be integrated with a 4-channel relay module initially, enabling the control of up to four household appliances. In terms of software, the project will involve creating firmware for the ESP8266 and developing a user-friendly mobile or web-based application for remote control and monitoring.

Connectivity is another core component, as secure Wi-Fi connectivity will allow seamless communication between the control unit and the user interfaces. An accessible and intuitive interface will be designed for mobile and web platforms, ensuring ease of use across a diverse user base. Key features will include remote appliance control, real-time status monitoring, scheduling, automation, and energy consumption tracking, providing users with actionable insights into their energy use. To protect user data, IntelliNest incorporates encryption protocols and multi-factor authentication, safeguarding against unauthorized access and enhancing overall security.

The project's scalability ensures that users can expand the system to control additional appliances, supporting evolving household needs. Comprehensive testing and validation will be conducted to ensure the system meets high standards of reliability and performance, with user satisfaction as a priority. Detailed user manuals, installation guides, and technical documentation will be created to facilitate smooth deployment and usage. Cost optimization will be a major focus, as we aim to keep the retail price within a target range of ₹500 to ₹750 per unit, making IntelliNest highly accessible. Furthermore, we are committed to eco-friendly design principles, with a focus on energy efficiency and sustainable materials where feasible.

The scope of this project is intentionally focused on creating a versatile control system compatible with a wide range of existing household devices rather than developing proprietary smart appliances. The initial release will target residential applications, with the potential for future expansion into commercial and industrial settings. By focusing on these key areas, IntelliNest seeks to deliver a comprehensive, user-friendly, and affordable smart home solution that addresses current gaps in the market. It aims to pave the way for broader adoption of smart home technologies by meeting the needs of both novice users and experienced technophiles alike, creating a foundation for future advancements in IoT-driven home automation. Through IntelliNest, we aim to transform homes into truly interconnected, intelligent living spaces that align with modern standards of convenience, energy efficiency, and security.

2.PROJECT DESCRIPTION AND GOALS

2.1 Literature Review

In recent years, the field of smart home automation has evolved rapidly, spurred by the rise of the Internet of Things (IoT) and cost-effective microcontrollers such as the ESP8266. This compact, Wi-Fi-enabled microcontroller has enabled researchers and engineers to create affordable, efficient, and scalable solutions for IoT-based home automation, leading to numerous innovations in the domain. This section presents a detailed review of the existing literature, focusing on key areas such as device control, energy management, security systems, environmental monitoring, voice integration, lighting, healthcare monitoring, cloud integration, and multi-protocol interoperability.

The ESP8266 has proven instrumental in enabling Wi-Fi-based control of home devices. Durani et al. (2018) demonstrated a low-cost smart home solution utilizing the ESP8266, in conjunction with the Blynk App, allowing users to remotely control household appliances. Similarly, Patchava et al. (2015) combined ESP8266 with Raspberry Pi to enhance functionality, creating a system that allowed for device monitoring and control via a mobile application. These studies highlight the ESP8266's potential in establishing wireless, internet-connected home automation systems with minimal infrastructure requirements.

As energy efficiency becomes a priority in smart homes, researchers have explored the ESP8266's potential in creating sustainable energy solutions. Lavanya and Suresh (2017) developed a smart energy management system using ESP8266 that provided users with real-time power consumption data. Sahani et al. (2017) further advanced this concept by implementing an IoT-based smart electricity meter, offering remote monitoring and control capabilities. These applications showcase ESP8266's role in helping homeowners track and manage their energy usage more effectively.

Security is a critical component of smart home automation. Biswas et al. (2017) utilized the ESP8266 to develop a cost-effective home security system that alerts homeowners to security breaches in real-time using various sensors. Chandramohan et al. (2017) extended these capabilities with an IoT-based smart door lock system, granting users the ability to control access to their home remotely. These projects underscore the ESP8266's ability to improve home security affordably, using real-time connectivity and data transfer.

Environmental monitoring in smart homes has also gained traction, with the ESP8266 playing a crucial role. Singh et al. (2018) designed an environmental monitoring system using ESP8266 and ThingSpeak, which provided real-time measurements of temperature, humidity, and air quality. Zhou et al. (2017) applied ESP8266 in a smart garden watering system, adjusting watering levels based on soil moisture data. Such innovations highlight ESP8266's potential in automating environmental control and enhancing indoor and outdoor living conditions.

Voice-controlled smart homes are gaining popularity for their user-friendly interfaces. Asadullah and Ullah (2017) implemented a voice-controlled smart home system using ESP8266 integrated with Google Assistant, facilitating seamless control of home devices through spoken commands. In a similar vein, Dhruvajyoti et al. (2020) developed an ESP8266-based home automation system that used Amazon Alexa for voice recognition, demonstrating the microcontroller's compatibility with popular virtual assistants. These studies illustrate the importance of voice integration in enhancing user interaction within smart homes.

Lighting control has become a prominent aspect of smart homes, with ESP8266-enabled systems proving highly effective. Prabaharan et al. (2018) proposed an intelligent lighting control system that uses occupancy and ambient light levels to adjust lighting automatically, reducing energy consumption. Additionally, Lee and Hong (2019) developed a customizable LED lighting system controlled by ESP8266, allowing users to alter colors and set lighting scenarios via a smartphone application. These systems showcase how ESP8266 can contribute to energy-efficient and aesthetically pleasing lighting solutions.

ESP8266's application in healthcare is growing, especially in home settings for health monitoring and elderly care. Ghosh et al. (2020) created a health monitoring system with ESP8266, integrating sensors to track health metrics in real time. Rani et al. (2021) advanced this further with a fall detection system, specifically designed for elderly people, combining ESP8266 with accelerometer sensors to detect falls and alert caregivers. These studies highlight the potential of ESP8266 in enhancing health and safety within homes.

Integrating IoT-based smart home systems with cloud platforms has become a critical area of research. Kostić et al. (2019) explored ESP8266-based smart home systems connected to cloud platforms such as AWS and Google Cloud, enabling advanced data analytics and enhanced control capabilities. Shete and Agrawal (2021) implemented an ESP8266-based system using the Blynk IoT framework, demonstrating the microcontroller's compatibility with popular cloud services for rapid application development. These studies emphasize the importance of cloud integration for expanding functionality and accessibility in smart home automation.

The integration of multiple communication protocols has been another area of focus to address interoperability challenges in IoT. Chowdhury et al. (2020) developed a hybrid protocol smart home system that combined ESP8266 with Bluetooth and Zigbee, ensuring compatibility across various device types. Patel and Patel (2022) implemented an IoT gateway using ESP8266, bridging multiple IoT protocols and enabling seamless communication among diverse smart devices. This line of research indicates the potential of ESP8266 to act as a central hub, facilitating cohesive interactions between devices across different protocols.

The above literature review highlights the versatility and adaptability of the ESP8266 in the realm of smart home automation. From basic device control to advanced integration with cloud platforms and various IoT protocols, the ESP8266 has proven to be an invaluable tool for developing affordable, reliable, and efficient smart home solutions. These studies collectively underscore the broad potential of ESP8266 in addressing various facets of smart home automation, advancing from rudimentary control mechanisms to complex, integrated systems that enhance both functionality and user experience. As the demand for connected homes continues to grow, the ESP8266's role in facilitating innovation within this space remains significant.

2.2 Research Gap

Despite the advancements in ESP8266-based smart home automation systems, several notable gaps persist that could hinder broader adoption and real-world application. One prominent issue is the lack of standardization. The existing systems are often developed independently, which results in limited interoperability between devices from different manufacturers or across various platforms. A standardized architecture would allow for seamless integration and communication, facilitating the creation of a unified smart home ecosystem.

Security is another critical concern, as many studies prioritize system functionality over robust cybersecurity measures. This leaves systems vulnerable to potential cyber threats, raising privacy and data security issues. As the use of IoT expands in personal spaces, there is an urgent need for improved security protocols and data protection methods to prevent unauthorized access and safeguard user data. Moreover, most ESP8266-based smart home projects have limited scalability, typically designed to control only a few devices. Research on scaling these systems without compromising performance is essential, especially as homes incorporate increasing numbers of IoT devices. Optimizing power efficiency also remains an area of opportunity; while the ESP8266 is relatively efficient, further enhancements could benefit battery-powered devices, extending their usability in scenarios where frequent recharging is impractical.

User experience also presents challenges, as many systems lack intuitive, accessible interfaces that cater to non-technical users. Designing more user-friendly interfaces would increase the appeal of these systems and make smart home automation more accessible to a broader audience. Additionally, only a few studies explore integrating artificial intelligence and machine learning with ESP8266-based systems, which could enhance functionality through predictive automation and decision-making. Such integration could enable smart systems to anticipate user needs and adapt to patterns, thereby improving efficiency and user satisfaction.

Reliability and fault tolerance are other critical areas that require attention. Many current systems lack mechanisms to handle network or device failures, which undermines the stability of smart home solutions. Implementing fault-tolerant designs and improving reliability would ensure consistent performance, particularly in environments where continuous connectivity cannot be guaranteed. Privacy concerns also emerge as a gap, as there is limited focus on managing data collection and storage in a way that respects user privacy within smart home setups. Addressing these concerns will be vital for building trust and encouraging adoption.

Furthermore, many ESP8266-based systems lack contextual awareness, often operating without consideration for user behavior or environmental changes. Developing systems that are capable of adapting to the user's habits and surroundings would greatly enhance the relevance and usability of smart home automation. Lastly, few studies conduct a comprehensive cost-benefit analysis of ESP8266-based smart home systems in comparison to alternative solutions. Analyzing the financial and practical benefits of these systems would provide valuable insights into their feasibility and long-term sustainability for consumers.

Addressing these gaps could result in more resilient, secure, scalable, and user-centric smart home systems, positioning ESP8266-based automation as a leading solution in the evolving landscape of IoT technology. By focusing on these areas, future research and development efforts could significantly enhance the usability, effectiveness, and adoption of ESP8266 in smart home automation.

2.3 Objectives

The goal of this project is to develop a comprehensive and efficient smart home control system using the ESP8266 microcontroller, focusing on features such as remote control, real-time monitoring, energy efficiency, scalability, security, and user engagement. By leveraging Internet-of-Things (IoT) technologies, the system will provide users with a seamless, intuitive way to control and manage household appliances, while also ensuring that it remains accessible, energy-efficient, and secure.

The first key objective of this project is to enable remote control of home appliances. This will be achieved by developing a mobile or web application that allows users to interact with their appliances from anywhere with an internet connection. The application will act as the central control interface, facilitating communication between the user and various devices, such as lights, fans, and other electrical appliances. The goal is to ensure that users can easily monitor and manage these devices remotely, enhancing convenience and making it possible to adjust settings when away from home.

In addition to remote control, real-time monitoring will play a pivotal role in the system's design. Users will be able to view the operational status of their appliances in real-time, receiving instant notifications about whether devices are powered on or off. This feature will enable users to make informed decisions regarding their appliance usage, improving overall household energy management. By providing continuous feedback on the status of connected devices, users will be able to ensure that appliances are not left on unnecessarily, thus contributing to better energy efficiency.

Energy management will be another core focus of the system. To promote sustainability, the system will include scheduling features that allow users to automate the operation of appliances based on their daily routines. By setting specific times for devices to turn on or off, the system will help reduce unnecessary energy consumption, such as preventing appliances from being left on when not in use. This objective aims to not only enhance the convenience of controlling home devices but also to contribute to significant energy savings and environmental sustainability.

Scalability is an important consideration for the system's long-term use and adaptability. The system architecture will be designed to support the easy addition of additional relay modules, allowing users to expand their smart home setup as their needs grow. Whether for a small apartment or a large home, the system will be flexible enough to accommodate a range of devices, ensuring that it can scale effectively as the number of connected devices increases.

Security is a fundamental concern for any IoT-based system, and this project will prioritize the implementation of robust security measures. Secure communication protocols, such as SSL/TLS encryption, will be employed to protect the data exchanged between the user application and the ESP8266 microcontroller. Additionally, the system will incorporate multi-factor authentication to ensure that only authorized users can access and control the appliances, preventing unauthorized access and protecting user privacy.

Maintaining cost-effectiveness is another critical objective, particularly to ensure the system remains accessible to a wide consumer base. The system will be designed to operate within a target price range of ₹500 to ₹750, balancing affordability with functionality. This goal ensures that the system offers high value without sacrificing quality, making smart home technology accessible to a broader range of households.

The choice of hardware for this project will center around the ESP8266 microcontroller, which offers integrated Wi-Fi capabilities and sufficient processing power to handle the control of multiple devices. This hardware choice ensures reliable performance while minimizing energy consumption, enabling the efficient control of high-power appliances through a 4-channel relay module. The efficiency of the hardware will play a key role in the overall system's performance, ensuring smooth operation and reliable connectivity.

User engagement is another critical aspect of the system’s design. The user interface will be developed with simplicity and clarity in mind, ensuring that the system is easy to navigate and use. Clear instructions and support resources will be provided to help users maximize the benefits of the system. By enhancing the user experience, the system aims to encourage regular interaction, leading to higher user satisfaction and a greater overall adoption rate.

To ensure the system’s reliability and robustness, comprehensive testing will be conducted throughout the development process. This will include unit testing to verify the functionality of individual components, integration testing to ensure seamless communication between the hardware and software, and user acceptance testing (UAT) to confirm that the system meets user expectations and requirements. Rigorous testing will help identify and resolve any issues before deployment, ensuring the system is stable and performs well in real-world conditions.

Finally, after deployment, continuous feedback will be sought from users to identify areas for improvement and further development. This feedback loop will allow for ongoing refinement of the system, ensuring that it adapts to evolving user needs and remains effective in the long term. By incorporating user insights into future updates, the system can be continually enhanced, maintaining its relevance and effectiveness over time.

In conclusion, the objectives of this project aim to create a smart home control system that is convenient, energy-efficient, secure, and scalable. By integrating advanced features such as real-time monitoring, scheduling, and remote control, the system will empower users to manage their appliances more effectively. The combination of user-friendly design, robust security measures, and efficient hardware will ensure that the system provides both high functionality and a positive user experience, making smart home technology accessible to a broader audience while promoting sustainability and energy efficiency.

2.4 Problem Statement

The rapid adoption of Internet of Things (IoT) technology has led to significant advancements in home automation, but many of the existing smart home solutions fall short of meeting the diverse needs of average consumers. Despite the potential of IoT to revolutionize home management, several key issues persist in the current market.

One of the most significant barriers to widespread adoption is the high cost of commercial smart home products. Many solutions available today are prohibitively expensive, limiting access to only a small segment of the population and excluding average consumers from benefiting from the convenience and efficiency of smart home technology.

Additionally, many existing systems require technical expertise for installation and operation, making them difficult to use for non-tech-savvy individuals. This complexity discourages a large portion of the potential user base, hindering the growth of smart home adoption.

Another issue is the lack of flexibility and scalability in current systems. Many smart home products are closed systems, offering limited customization and expansion options. This inflexibility restricts users from tailoring their systems to suit their evolving needs, particularly as their household grows or changes.

Furthermore, energy inefficiency remains a significant concern. Many existing systems fail to provide users with effective control over their appliances, leading to unnecessary energy consumption. Without remote monitoring, scheduling, or automation features, households often waste energy, contributing to higher utility bills and environmental degradation.

Security vulnerabilities in IoT devices present another major challenge. Smart home

systems, if not adequately secured, can become targets for cyberattacks, posing serious risks to user privacy and safety. The lack of robust security measures in many current solutions leaves users exposed to unauthorized access and potential breaches.
Moreover, real-time feedback is often absent in many systems, leaving users in the dark about the status of their devices. This lack of instant information hampers effective management of household appliances, leading to inefficiencies and inconvenience.
Other problems include the limited control options offered by existing systems, lack of accessibility for individuals with disabilities, absence of meaningful data insights on energy consumption, and the environmental impact of the production and disposal of smart home devices. These issues further limit the appeal and usability of current smart home solutions, preventing them from achieving their full potential.
Addressing these problems requires an innovative solution that is affordable, easy to use, scalable, secure, energy-efficient, and accessible. The "IntelliNest: Smart Living Redefined" project seeks to tackle these challenges by creating a comprehensive smart home control system that empowers users to manage their homes effectively while promoting sustainability and inclusivity.

2.5 Project Plan

The development of the "IntelliNest: Smart Living Redefined" IoT-based smart home control system follows a detailed project plan designed to ensure efficient use of resources and meet academic timelines. This plan outlines the key phases and milestones that will guide the team throughout the project, with clear objectives and deliverables for each stage. The project spans 16 weeks, divided into six phases, ensuring the systematic development of both hardware and software components.
The first phase, *Research and Planning*, will span two weeks. During this phase, the team will conduct a comprehensive analysis of existing smart home solutions, identify their strengths and weaknesses, and pinpoint areas where the "IntelliNest" system can offer improvements. This stage will also involve defining the specific requirements for the project, outlining the system's core features, and establishing a preliminary system architecture. The primary deliverable from this phase will be a detailed project proposal that will guide subsequent development efforts.
Following the planning phase, the second phase, *Design*, will last three weeks. In this

phase, the team will focus on designing the system's hardware and software architectures. This will include creating the circuit design for integrating the ESP8266 microcontroller with the 4-channel relay module. Simultaneously, the team will design basic UI mockups for the mobile or web application and develop a plan for the overall software architecture. The output of this phase will be a comprehensive design documentation that will serve as the foundation for the hardware and software development phases.

The third phase, *Hardware Development*, will also span three weeks. This phase will begin with procuring the necessary components, including the ESP8266 microcontroller, relay modules, and other basic electronic parts. The team will assemble the hardware components on a breadboard and conduct basic functionality tests to ensure proper integration of the components. At the end of this phase, the team will have a working hardware prototype, which will be the basis for the subsequent software development.

In the fourth phase, *Software Development*, which will last four weeks, the team will develop the firmware for the ESP8266 microcontroller to enable the control and monitoring of appliances. A simple mobile or web application will be developed, which will allow users to remotely control their home appliances via the internet. During this phase, the team will implement basic control features such as device switching, real-time monitoring, and scheduling for energy management. The completion of this phase will result in a basic functional software system.

The fifth phase, *Integration and Testing*, will take place over two weeks. The team will integrate the hardware and software components into a unified system, testing for compatibility and functionality. Any issues that arise during integration will be debugged, and the system will be refined to ensure smooth operation. This phase will also include performance and reliability tests to verify that the system functions effectively under real-world conditions. The goal is to produce an integrated prototype that demonstrates the system's capabilities.

The sixth and final phase, *Documentation and Presentation*, will span the final two weeks. In this phase, the team will compile a comprehensive project report documenting all aspects of the development process, from initial research to the final prototype. The team will also prepare presentation slides and a simple user guide, ensuring that the system's features and usage instructions are clear. The deliverables from this phase will include the final report, presentation, and user guide, providing a complete record of the project’s outcomes.

To ensure efficient progress, the team will adopt a collaborative approach, with clearly defined roles and responsibilities. Vivek and Suraj will focus on hardware development, while Vivek and Raghav will handle software development. Suraj and Raghav will share responsibilities for documentation and testing. This distribution ensures that each team member’s strengths are leveraged, while also encouraging cross-functional collaboration.

The successful completion of the project will require a variety of resources, including the ESP8266 microcontroller, 4-channel relay module, basic electronic components, and a computer for programming and testing. An internet connection will be necessary to enable the IoT functionality of the system.

The project is expected to face a number of challenges, including the learning curve associated with IoT and embedded systems programming, integrating hardware and software components, ensuring reliable Wi-Fi connectivity, and managing time effectively alongside other coursework. To mitigate these risks, the team will hold weekly meetings to track progress, troubleshoot issues, and review goals. Regular check-ins with the project advisor will ensure the project stays aligned with academic expectations, while a shared online tool such as Trello or Google Sheets will help the team stay organized and on schedule.

3. TECHNICAL SPECIFICATION

3.1. Requirements

The primary challenge is to design and implement a secure, efficient, and user-friendly smart home automation system that integrates various IoT devices and sensors to control home appliances. The system must ensure privacy, data integrity, and energy efficiency while providing real-time monitoring and control capabilities.

Key issues to address:

- Secure communication between devices and central server

- User authentication and access control

- Real-time monitoring and control of appliances

- Energy efficiency through automated controls

- Integration of multiple sensor types for comprehensive home automation

- Scalability and maintainability of the system

3.1.1 Functional Requirements

1. **User Authentication and Access Control**: One of the key functional requirements is to implement multi-factor authentication (MFA) to provide an extra layer of security for the user login process. MFA requires users to not only input their password but also a second factor of authentication, such as a verification code sent to their phone or an app, significantly increasing the security of the system. A secure login interface is created using React.js, which offers a responsive, dynamic, and user-friendly interface for login. The Node.js backend is responsible for handling the authentication process by validating the user’s credentials against a database, ensuring that only

authorized users can access the system.

2. **Appliance Control**: The system must enable users to manually control appliances through a web-based dashboard. This feature allows users to turn on/off devices and adjust settings in real time from any web browser. Additionally, the system should be capable of automatic control based on sensor inputs. For example, motion sensors can automatically turn on lights when someone enters a room, while temperature sensors can adjust air conditioning settings when the room temperature rises. However, users should always have the ability to override automatic controls, giving them full manual control over the system when necessary. This combination of manual and automated control ensures flexibility and user empowerment.
3. **IoT Device Communication**: **ESP8266 microcontrollers** will be used for connecting appliances to the system. These are low-cost, energy-efficient IoT devices that can communicate wirelessly with the server to execute control commands. The system communicates securely using SSL-encrypted HTTP for device-server communication, ensuring that all data exchanged is protected. AWS IoT Core is employed to manage this communication, providing a reliable and scalable platform for connecting IoT devices with the cloud securely. This setup ensures that appliances can be controlled and monitored remotely with minimal risk of data breaches or malicious interference.
4. **Real-Time Monitoring**: The system must support real-time monitoring of appliance statuses and their power consumption. Users should be able to check the current operational state of any appliance and view the energy being consumed at any given moment. To enhance security and efficiency, the system should generate alerts for abnormal usage, unauthorized access, or system malfunctions. For example, if an appliance is drawing more power than usual or is operating in an unusual state, the system will notify the user immediately. These alerts help users take action quickly to prevent potential issues or reduce energy waste.
5. **Data Collection and Processing**: The system will aggregate data from various sensors embedded in the environment, such as motion sensors,

temperature sensors, humidity sensors, light sensors, power meters, and smoke/gas detectors. This sensor data is essential for understanding the current environmental conditions and making informed decisions. Real-time data processing ensures that the system can react instantly to changes in the environment, such as adjusting appliance settings based on the current room temperature or turning off lights if no motion is detected. By processing data as it is received, the system ensures that actions are taken immediately, maintaining a smooth and efficient user experience.

6. **Energy Management**: A key objective of the system is to provide users with detailed insights into their power consumption patterns. By analyzing usage trends over time, the system can help users understand which appliances consume the most energy and suggest potential ways to reduce power usage. The system should also include automated energy-saving features, such as turning off appliances when they are not in use or when sensors detect that a room is unoccupied. These features aim to optimize energy consumption, helping users reduce their carbon footprint and energy bills.

3.1.2 Non-Functional Requirements

1. **Security**: Security is a paramount concern in any IoT-based system, and this project is no exception. The system must implement end-to-end encryption to ensure that all communications between devices and the server are secure. This includes HTTPS for web-based interactions and SSL/TLS encryption for the IoT devices’ communications. Additionally, MQTT with SSL is used for lightweight, low-latency messaging between the devices and the server. The system must also ensure that firmware updates are securely delivered, with each update being digitally signed to prevent unauthorized modifications. To further enhance security, token-based API access is implemented, requiring authentication tokens for any API interaction, ensuring that only authorized users and systems can interact with the server.

2. **Performance**: The system is designed to deliver minimal latency (less than

one second) for device responses and user interface updates. This low-latency requirement is crucial to ensure that users experience near-instantaneous control over appliances and that any changes in sensor data are reflected immediately on the user interface. Additionally, real-time processing of sensor data and control commands is essential for an effective IoT system. The ability to process data as soon as it is received allows the system to make quick decisions, such as adjusting the temperature of an HVAC system or turning off lights when no motion is detected, improving user experience and system efficiency.

3. **Scalability**: As the number of IoT devices and sensors in the system grows, it is essential that the platform remains scalable. The system must be designed to handle an increasing number of connected devices without sacrificing performance or reliability. This means that as users add more appliances or sensors, the system should be able to integrate them easily without requiring significant changes to the architecture. The scalability also ensures that performance remains consistent even as the system expands, maintaining a smooth experience for users regardless of the size of the deployment.
4. **Reliability**: The reliability of the system is critical, particularly since it controls essential household appliances. The central control server should have high uptime, meaning it should be available and functional nearly all the time. In case of a failure, the system should offer fallback mechanisms, such as the ability to manually control appliances, ensuring that users are not left without control in the event of a system failure. These reliability measures ensure that the system can provide consistent service and remain operational even during unforeseen disruptions.
5. **Usability**: The usability of the system is a significant consideration, as it impacts how easily users can interact with the system to control appliances and monitor their energy consumption. The user interface (UI) is designed to be intuitive, with easy-to-understand controls, icons, and feedback, ensuring that even non-technical users can operate the system with ease. The UI must also provide clear feedback on actions being taken, such as a notification when an appliance is turned off or an alert if there is an issue with the system. This

ensures that users are always aware of the system's state and can respond appropriately to any issues.

6. **Maintainability**: **Maintainability** is a key non-functional requirement, as it ensures the system can be easily updated and improved over time. The system will support over-the-air (OTA) updates for both the server software and the IoT device firmware, allowing for remote upgrades without requiring physical access to the devices. This ensures that the system can be maintained and updated efficiently, minimizing downtime and ensuring that devices always have the latest features and security patches. Additionally, comprehensive documentation for both hardware and software components will be provided, which will be crucial for troubleshooting, updating, and expanding the system in the future.

3.2 Feasibility Study

The feasibility study for IntelliNest examines the technical, economic, and social viability of the project, considering the project's technological requirements, budgetary constraints, and potential social impact. The purpose of this study is to ensure that IntelliNest can be realistically developed, deployed, and maintained within the planned scope, while delivering a functional and user-centered experience.

3.2.1 Technical Feasibility

In assessing the technical feasibility of IntelliNest, we evaluated the compatibility, functionality, and scalability of the proposed technology stack. The system relies on the ESP8266 microcontroller, a well-established choice in IoT applications due to its integrated Wi-Fi capabilities, cost-effectiveness, and sufficient processing power for real-time operations. The ESP8266's capacity to serve as both a client and a server makes it particularly suitable for enabling home automation features, such as remote control and real-time monitoring, as it can readily handle commands sent from the backend server and promptly execute these commands at the device level. Additionally, the ESP8266's compact design and low energy consumption align with IntelliNest's objectives to create an energy-efficient system that can seamlessly fit into a variety of household environments.

The backend of IntelliNest is developed using Node.js, a JavaScript runtime known for its non-blocking, event-driven architecture, which efficiently manages concurrent requests in a real-time environment. Node.js is an optimal choice for a smart home system, where user commands, status updates, and device feedback must be handled with minimal latency. To facilitate the communication between devices and the backend server, we employ AWS IoT Core with MQTT protocols. MQTT is specifically designed for IoT environments, and its lightweight nature reduces network overhead, making it ideal for devices with constrained resources like the ESP8266. AWS IoT Core enhances this setup by providing robust management tools and security features, such as SSL/TLS encryption, which ensure data integrity and protect against unauthorized access.

On the data management side, MongoDB is used as the database solution. Known for its flexibility in handling both structured and unstructured data, MongoDB allows IntelliNest to store user information, device configurations, and historical logs efficiently. This flexibility is crucial for a system that may need to accommodate diverse data types and expand over time. MongoDB's scalability also enables the IntelliNest system to support an increasing number of users and devices without performance degradation, aligning with the project's long-term goals for scalability and adaptability.

The integration of these technologies confirms IntelliNest's technical feasibility, ensuring that the system architecture can support its functional requirements. Together, the ESP8266, Node.js backend, AWS IoT Core, and MongoDB form a cohesive and robust technical foundation capable of delivering a scalable, secure, and high-performing smart home solution.

3.2.2 Economic Feasibility

The economic feasibility of IntelliNest was assessed to determine whether the project's development, deployment, and maintenance costs align with the target retail price and intended market reach. The project aims to maintain affordability without compromising on quality, with a target retail price range between ₹500 and ₹750 per unit. This price point is designed to make IntelliNest accessible to a broad audience, especially in regions where premium smart home solutions may be economically prohibitive.

IntelliNest minimizes initial hardware costs by using the ESP8266 microcontroller and a 4-channel relay module, both of which are cost-effective yet reliable components for IoT applications. Bulk purchasing of these components further reduces costs, allowing us to offer a competitively priced product. Open-source frameworks like Node.js and React.js, selected for the backend and frontend, respectively, reduce software development expenses, as they eliminate licensing fees and offer extensive community support, which shortens the development cycle and minimizes costs associated with troubleshooting and maintenance.

In terms of operational expenses, the system leverages cloud-based infrastructure through AWS IoT Core, which provides a scalable and secure environment for device communication. AWS's pay-as-you-go model eliminates the need for significant upfront investment in server hardware, allowing the system to scale economically as demand grows. This approach also reduces maintenance costs, as AWS handles much of the server management and provides integrated security tools. Additionally, MongoDB's flexible licensing and its ability to efficiently store and retrieve data further optimize long-term operational costs.

By strategically selecting affordable hardware, open-source software, and cloud infrastructure, IntelliNest achieves economic feasibility, ensuring that the system can be developed and maintained within the target price range. This approach makes IntelliNest not only cost-effective for end-users but also financially sustainable as a product.

3.2.3 Social Feasibility

IntelliNest's social feasibility examines the project's alignment with societal needs and its potential impact on users' daily lives. In recent years, there has been a noticeable shift in consumer expectations toward products that offer convenience, sustainability, and security. IntelliNest is designed to meet these needs by providing an accessible, user-friendly solution for managing household devices. The system enables users to control appliances remotely, offering convenience for individuals with busy schedules, limited mobility, or those who prefer the flexibility of managing their home environment from any location.

The scheduling and automation features embedded in IntelliNest promote energy efficiency, an aspect that appeals to environmentally conscious consumers. By enabling users to set schedules for devices, IntelliNest helps reduce unnecessary energy consumption, leading to lower utility bills and contributing to environmental sustainability. These features also support a growing movement toward smart energy management, in which IoT devices play a pivotal role in reducing residential carbon footprints.

IntelliNest addresses another major societal concern: data security. As IoT devices become more common in households, there is an increasing awareness and concern about privacy risks. IntelliNest incorporates robust security measures, including multi-factor authentication and SSL/TLS encryption, to protect user data and prevent unauthorized access. This security focus is designed to build trust with users who are conscious of the privacy risks associated with IoT products, thus encouraging broader adoption.

IntelliNest's design also considers accessibility, making it suitable for a diverse user base, including elderly users who may benefit from automated scheduling and those who might struggle with traditional appliance management. The social feasibility of IntelliNest, therefore, is confirmed by its alignment with contemporary lifestyle needs, its potential to promote energy-efficient practices, and its commitment to user privacy and data security.

3.3 System Specification

The system specification for IntelliNest outlines the essential hardware and software components required to build a responsive, scalable, and user-centric smart home automation system. Each component was selected based on its compatibility, performance, and contribution to the overall functionality and security of the system, ensuring a reliable and effective user experience.

3.3.1 Hardware Specification

The hardware components of IntelliNest were chosen to provide reliable performance, seamless communication, and cost-effectiveness, meeting the functional requirements of a modern smart home system. At the core of IntelliNest's hardware architecture is the ESP8266 microcontroller, which acts as both the processing unit and communication hub. The ESP8266's built-in Wi-Fi module enables it to connect seamlessly to the home's Wi-Fi network, facilitating real-time communication with the backend server. The microcontroller's processing capabilities allow it to manage device states and execute commands efficiently, while its low power consumption supports the project's goal of energy efficiency.

Connected to the ESP8266 is a 4-channel relay module, which controls the electrical appliances linked to the system. This relay module allows IntelliNest to manage multiple devices simultaneously, supporting up to four appliances per module. Each relay can be triggered individually, enabling users to control different appliances independently. This modular approach provides flexibility, allowing for expansion as users add more devices to their smart home setup.

The IntelliNest system also includes a 5V DC power supply unit (PSU) to ensure stable and reliable operation of the ESP8266 and relay module. This power configuration is widely available, economical, and compatible with the system's components, reducing the overall cost while ensuring that the system operates smoothly. Additional components, such as optional sensors, can be integrated to enhance IntelliNest's functionality. For instance, motion sensors can be used to trigger security features, while temperature sensors can automate HVAC controls based on real-time conditions. These add-ons increase the system's adaptability, allowing IntelliNest to cater to a broader range of user preferences and household needs.

3.3.2 Software Specification

The IntelliNest software ecosystem was designed to ensure efficient data processing, secure communication, and a user-friendly experience. The frontend of IntelliNest is built using React.js, a JavaScript library known for creating dynamic, responsive user interfaces. React.js allows IntelliNest to deliver a single-page application (SPA) experience, where users can control appliances, set schedules, and receive updates without reloading pages, resulting in faster and smoother interactions. The UI is designed to be intuitive and accessible, with a layout that accommodates various devices, including desktops, tablets, and smartphones. This responsiveness ensures that users can manage their home automation system from any location, at any time.

The backend of IntelliNest is developed with Node.js, which processes user commands, manages device states, and enforces security protocols. Node.js's asynchronous programming model enables the system to handle multiple requests simultaneously, essential for real-time operation in a multi-user environment. The backend also supports automation rules and scheduling, which are managed through predefined user parameters. These features allow users to customize appliance control, ensuring that IntelliNest can meet a range of usage scenarios and preferences.

To facilitate secure communication between the ESP8266 devices and the backend, IntelliNest uses AWS IoT Core, leveraging the MQTT protocol with SSL/TLS encryption. MQTT's lightweight nature is ideal for the ESP8266, minimizing bandwidth and power consumption while ensuring reliable data transmission. AWS IoT Core also includes identity management and access control, which are crucial for maintaining the security and integrity of device communication, especially as the system scales.

For data management, MongoDB serves as IntelliNest's database solution. MongoDB's document-based architecture supports a variety of data types, enabling it to efficiently store user information, device configurations, usage logs, and automation rules. This flexibility is essential for IntelliNest, as it allows the system to accommodate diverse data formats and adapt to new data types as the system expands. MongoDB also facilitates fast data retrieval, which is critical for delivering real-time updates and insights to users.

Together, these hardware and software components establish a robust, scalable, and secure foundation for IntelliNest, supporting its objectives of user convenience, energy efficiency, and data security. Through careful selection and integration of each element, the system specifications ensure that IntelliNest can deliver a reliable and versatile smart home experience.

Component	Description	Specifications	Rationale
ESP8266	Microcontroller	Wi-Fi enabled, low power consumption, sufficient processing power	Cost-effective, integrated Wi-Fi, adequate processing for real-time control
4-Channel Relay Module	Controls appliances	5V DC, 4 independent relays	Affordable, allows control of multiple appliances
Node.js	Backend server runtime	Non-blocking, event-driven	Efficient handling of concurrent requests, suitable for real-time operation
React.js	Frontend UI library	Component-based, creates dynamic and responsive UIs	Enhanced user experience, ease of development
MongoDB	Database	Document-based, scalable, flexible	Efficient data storage and retrieval, adaptable to diverse data types
AWS IoT Core	Cloud platform for IoT device management	Secure, scalable, supports MQTT	Secure communication, remote device management, scalability

Table 3.3.1 Hardware and Software Component Specifications

4. DESIGN APPROACH AND DETAILS

The IntelliNest smart home automation system is designed with a layered architecture that emphasizes modularity, scalability, and high performance. This design approach ensures a flexible, secure, and efficient platform for managing smart home devices, handling a large number of concurrent users, and accommodating complex automation scenarios. The goal is to provide a seamless user experience, real-time control, and continuous operation, all while maintaining system security and performance.

4.1 System Architecture

The IntelliNest system is structured using five primary layers: Presentation Layer, Application Layer, Communication Layer, Data Layer, and IoT Device Layer. Each layer has a specific function and interacts with other layers to deliver a complete smart home automation solution. Below, we delve into the details of each layer, explaining their roles and how they contribute to the overall architecture.

4.1.1 Architectural Overview

The IntelliNest system consists of several critical layers that ensure its operation. The Presentation Layer serves as the user-facing interface, offering various ways for users to interact with the system. The Web Application Interface provides users with a browser-based dashboard for monitoring and controlling devices, managing settings, and viewing system analytics. The Mobile Application Interface extends this functionality, enabling users to control and monitor their devices on the go through their smartphones or tablets. Additionally, the Administrative Dashboard is designed for administrators to oversee system operations, including managing users, devices, and configurations. This layer ensures that the system is accessible both for everyday users and for administrators responsible for system management.

The Application Layer houses the core business logic and services required for the functioning of IntelliNest. This layer is essential for tasks such as ensuring secure access, efficient device management, processing automation rules, and sending notifications. The Authentication Service is a crucial component in this layer, managing user authentication and ensuring that only authorized individuals can access the system. The Device Management Service handles all aspects of IoT devices, including their registration, configuration, monitoring, and control. The Automation Rules Engine is responsible for evaluating user-defined automation rules and triggering actions based on inputs from sensors or user commands. The Data Analytics Service gathers and analyzes data from devices, generating insights on system performance, usage patterns, and efficiency. Finally, the Notification Service ensures that users receive timely alerts and notifications regarding important events or system status updates.

The Communication Layer is responsible for managing data exchanges between devices, users, and services using secure and efficient communication protocols. This layer includes MQTT/HTTP Protocol Handlers, which facilitate communication between IoT devices and the system. MQTT is particularly suited for real-time messaging in low-bandwidth environments, while HTTP enables broader communication for web-based interactions. The WebSocket Service ensures that users receive real-time updates on device statuses and other system interactions, making the experience dynamic and responsive. The API Gateway is another critical component, acting as the entry point for external communications, routing requests to appropriate backend services while enforcing security policies.

The Data Layer manages the storage and retrieval of data used by the system, ensuring that all critical information is available when needed. The Time-Series Database is a key part of this layer, storing time-stamped sensor data and device status changes, allowing for historical analysis and trend tracking. The User Management Database stores all user-related information, including profiles, preferences, roles, and access permissions. The Device Configuration Store contains configuration settings and metadata for all connected devices. The Logging and Monitoring component ensures that detailed logs are maintained for auditing purposes and troubleshooting system issues.

Finally, the IoT Device Layer encompasses the physical IoT devices that interact with the IntelliNest system. This includes ESP8266 controllers, which serve as microcontrollers that interface with sensors, relays, and other connected components, enabling both data collection and control actions. Sensor Networks collect environmental data, such as temperature, humidity, and motion, and feed this information into the system for analysis or automation triggers. Relay Modules are used to control the on/off states of connected electrical appliances, allowing for automation sequences that enhance convenience and energy efficiency.

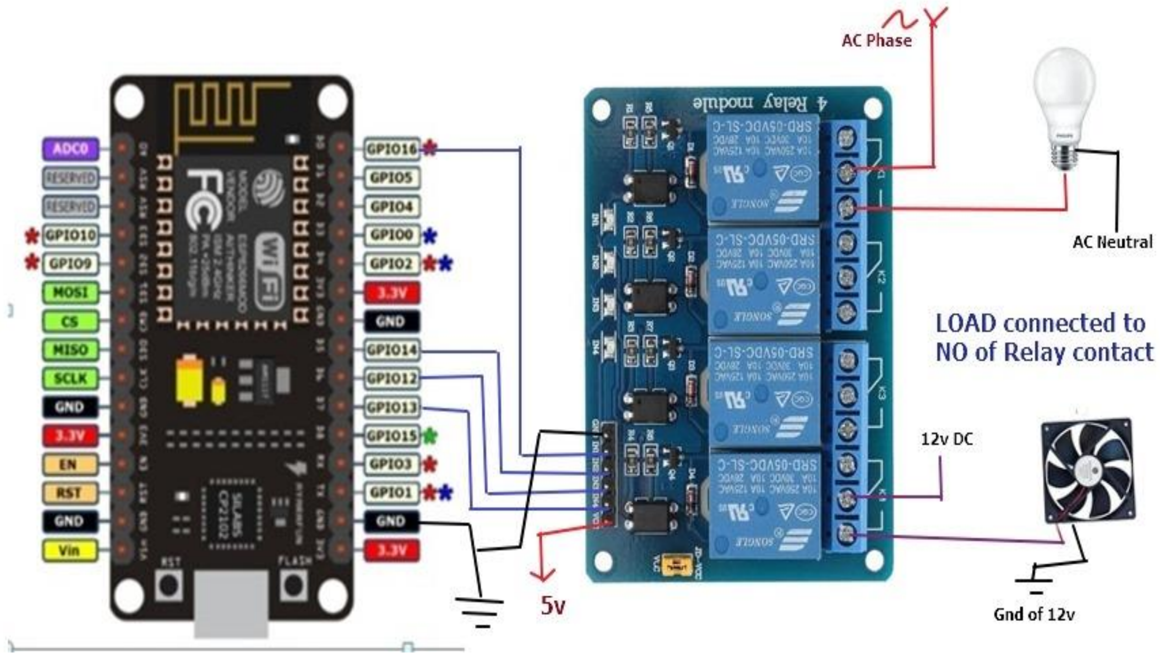


Figure 4.1.1 IntelliNest Architecture

4.1.2 Component Interaction

The interaction between various components in the IntelliNest system is designed to be seamless and efficient. RESTful APIs facilitate communication between the client interfaces (e.g., web and mobile applications) and the backend services. These APIs support key operations such as device control, rule management, and data retrieval. The system also uses WebSocket connections to provide real-time updates for device statuses, user actions, and incoming alerts. This ensures that users receive immediate feedback for their interactions with the system.

The MQTT Protocol plays a crucial role in communication between IoT devices and the backend services. This lightweight protocol is ideal for scenarios that require low latency and minimal bandwidth usage, such as reporting sensor data or sending control commands. Additionally, the system employs Event-Driven Messaging to manage asynchronous communication among services, ensuring that tasks such as automation rule execution and notification delivery are handled efficiently, without blocking other processes.

4.1.3 Scalability Considerations

As IntelliNest is designed to support a growing number of users and devices, scalability is a critical aspect of its architecture. Several strategies are employed to ensure that the system can handle increased demand while maintaining performance and reliability.
One key consideration is the use of a Microservices-Based Architecture. The IntelliNest system is divided into smaller, independent microservices, each responsible for specific tasks. This modular approach allows the system to scale horizontally, with new instances of services being added as needed. This ensures that system performance remains consistent even during periods of high demand. Additionally, each microservice can be maintained, deployed, and updated independently, which ensures that the system remains available even during updates.
To support this, IntelliNest employs Load Balancing, which distributes incoming requests across multiple server instances. This prevents bottlenecks and ensures that the system can handle spikes in user activity. Load balancers dynamically adjust to traffic patterns, ensuring responsive performance at all times.
The system also utilizes a Distributed Database Architecture to handle increasing amounts of data generated by sensors, users, and automation rules. Data is stored across multiple database nodes, which enhances both read and write performance while ensuring data availability in case of failures. This distribution also supports horizontal scaling of the database layer, allowing the system to accommodate growing volumes of data without compromising performance.
Finally, Caching Mechanisms are employed to reduce the load on the database and improve response times. Frequently accessed data, such as user preferences, device configurations, and status updates, is stored in caching systems, enabling faster retrieval and reducing the need for repeated database queries.

4.2 Design

The Design section of the IntelliNest project documentation provides a comprehensive analysis of the system’s structural and functional architecture, depicting how various components work together to achieve core functionality. This section includes detailed diagrams—Data Flow Diagram (DFD), Use Case Diagram, Class Diagram, and Sequence Diagram—that collectively outline data flow, user interactions, class structure, and process sequences within the system.

4.2.1 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) is a crucial part of the IntelliNest design, illustrating how data moves through the system and the interactions between various processes, data stores, and external entities. This diagram enables a clearer understanding of the overall information flow and shows dependencies between different system components, which is essential for analyzing data movement and identifying any potential bottlenecks.

At the highest level, the Level 0 DFD (also known as the Context Diagram) provides an overview of the system as a single process and represents its primary inputs and outputs, defining the boundaries between the system and external entities. In this diagram, key interactions include user commands to control devices, sensor data inputs for monitoring, system-generated control commands for device management, notifications sent to users, and administrative operations. User interactions form the core input, as users issue commands to control devices and retrieve status updates. Sensor data, constantly fed into the system, allows for monitoring and environmental adaptation. The system, in response to user or automation inputs, issues control commands to connected devices, facilitating real-time actions. Additionally, notifications inform users of system statuses, such as alerts for environmental changes or device malfunctions. Administrative operations represent another layer, enabling authorized users to configure devices, manage user access, and adjust system settings.

The Level 1 DFD breaks down the high-level system process into several more detailed sub-processes, providing an in-depth view of specific functions. Key processes include

User Authentication, which verifies user credentials and manages sessions to ensure secure access; Device Management, responsible for registering and monitoring the status of devices and processing commands issued by users or automation rules; Data Collection, which gathers, validates, and pre-processes data from sensors; Automation, which evaluates predefined rules, triggers actions based on conditions, and manages scheduled tasks; and Notification, which handles the creation of alerts, sends real-time notifications to users, and updates them on device statuses. Each of these processes interacts with external entities and each other, ensuring that data flows in a logical and organized manner throughout the system.

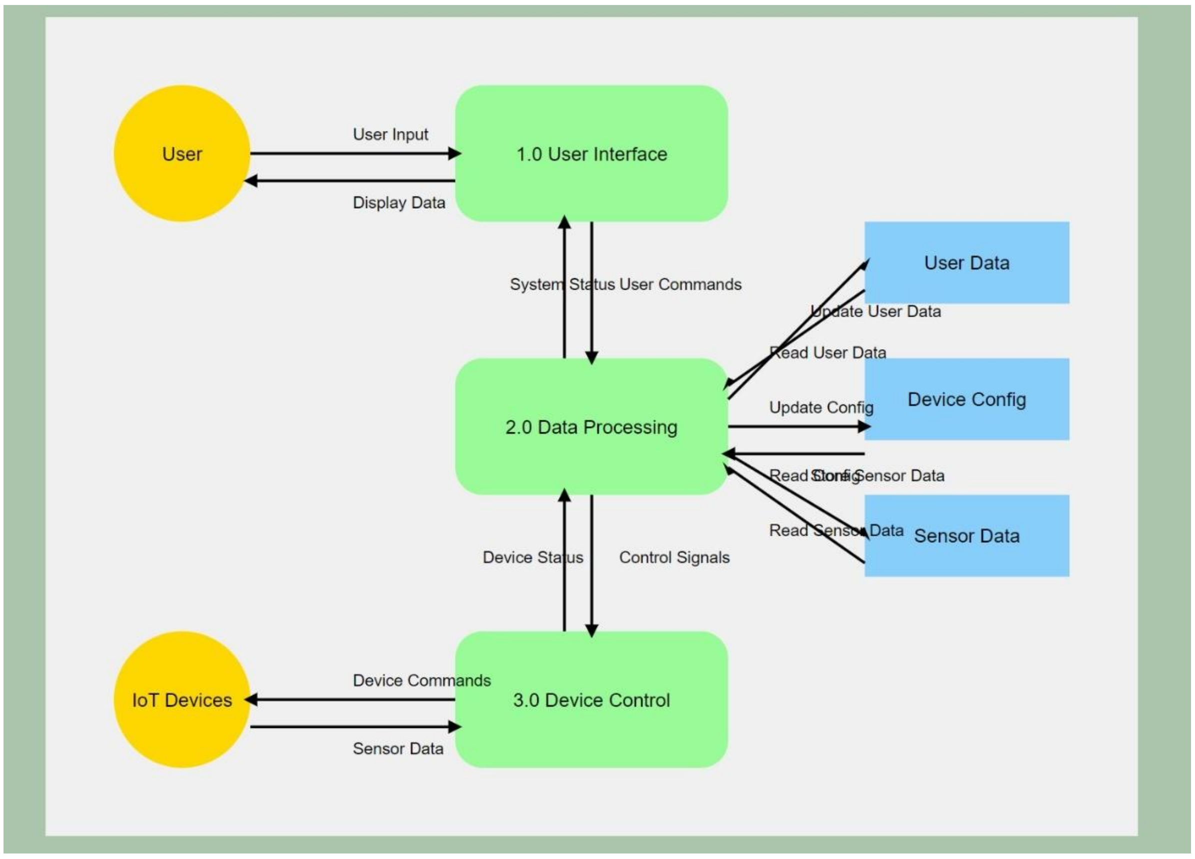


Figure 4.1.2 Level 1 Data Flow Diagram

4.2.2 Use Case Diagram

The Use Case Diagram represents the system’s functionality from the perspective of various user roles, illustrating the interactions between different actors and the primary use cases of the IntelliNest system. This diagram provides insight into the system’s capabilities by highlighting the actions available to each user type.

In this system, Regular Users represent the general audience, having access to basic functionalities like appliance control, monitoring, and automation setup. They can manage devices, monitor their statuses, and create basic automation rules. Administrative Users hold elevated permissions, enabling them to perform management tasks, including configuring devices, adjusting user permissions, and overseeing system configurations. IoT Devices, such as connected sensors or appliances, interact with the system to execute user commands or respond to automation rules. External Systems represent any third-party services or integrations, like cloud storage or monitoring systems, that may interact with the IntelliNest system. The primary use cases are divided into key functional areas: User Management (where users can create accounts, log in, manage profiles, and reset passwords); Device Control (which allows users to add and configure new devices, control their states, and monitor performance in real-time); Automation Management (enabling users to create and manage automation rules, schedule device operations, set event triggers, and define complex scenarios for home automation); and System Monitoring (where users can check device statuses, track system health, monitor energy usage, and view activity logs). These use cases provide a structured overview of the system's capabilities, showing how each user role can interact with the IntelliNest system.

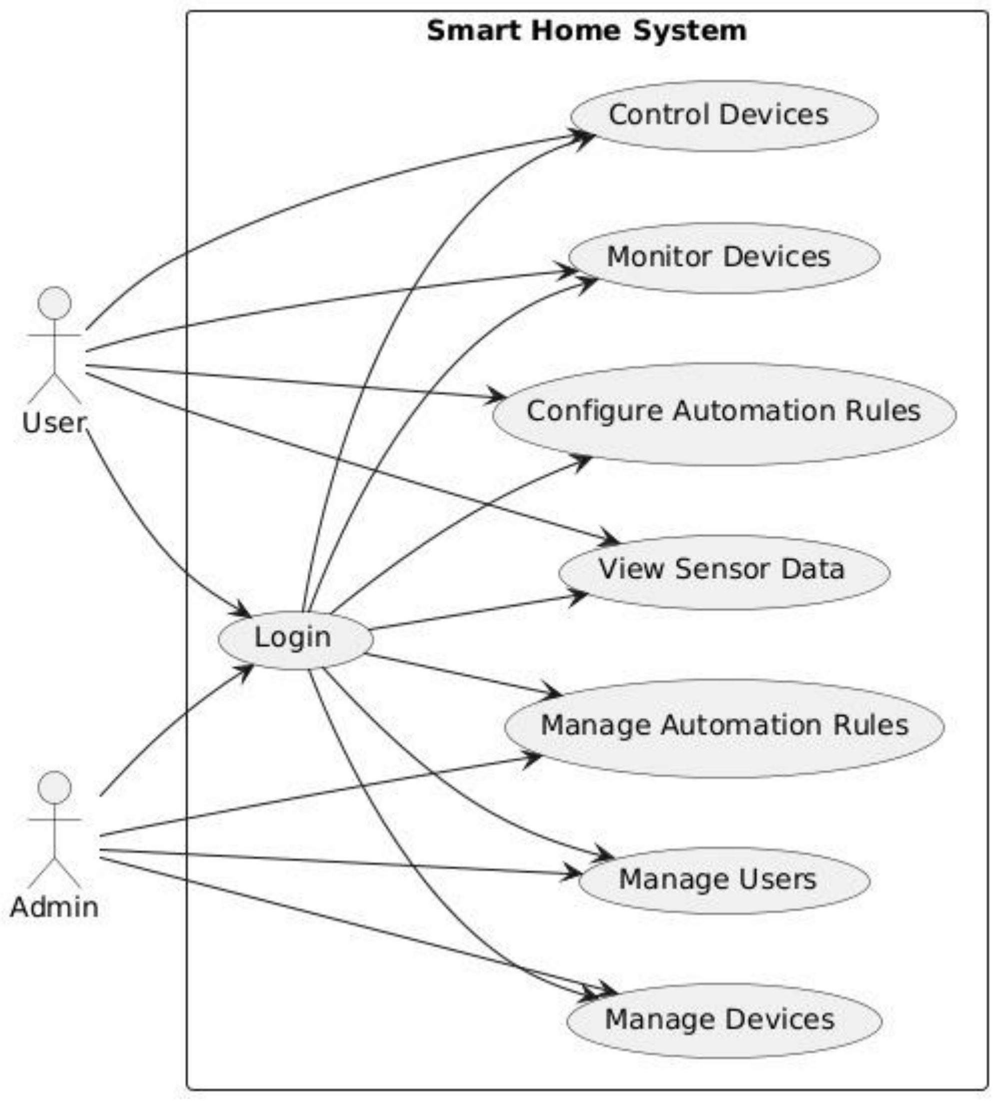


Figure 4.1.3 **Use Case Diagram**

4.2.3 Class Diagram

The Class Diagram defines the structure of classes within the IntelliNest system, focusing on the key attributes, methods, and relationships between classes. This diagram follows object-oriented design principles, essential for understanding the internal structure of the IntelliNest system and how various objects collaborate.

Key classes in this diagram include the User Class, which encompasses essential attributes like user ID, name, contact details, and access level, with methods such as login(), logout(), and updateProfile(). The Device Class represents connected IoT devices and includes attributes like device ID, type, status, and settings, with methods such as turnOn(), turnOff(), and configure() to control device behavior. The Sensor Class encapsulates all sensors in the system, detailing sensor type, status, and recorded values, along with methods like readData() and validateData(). The AutomationRule Class represents automation rules that trigger actions based on specific conditions; its attributes include rule ID, trigger condition, and actions, with methods for evaluating rules and executing actions. Additionally, the Notification Class manages alerts and notifications with attributes like notification ID, message content, and type, along with methods for generating and sending notifications. Finally, the Admin Class includes methods and attributes for managing users, devices, and configurations, making it essential for system administration.

Relationships between these classes are also vital for understanding system interactions. For instance, the User Class is associated with Device Class and Sensor Class for monitoring and control purposes, while Device Class and Sensor Class are linked, allowing devices to retrieve sensor data as needed. AutomationRule Class interacts with Device Class for task automation, and the Admin Class has control over User Class and Device Class to perform administrative functions.

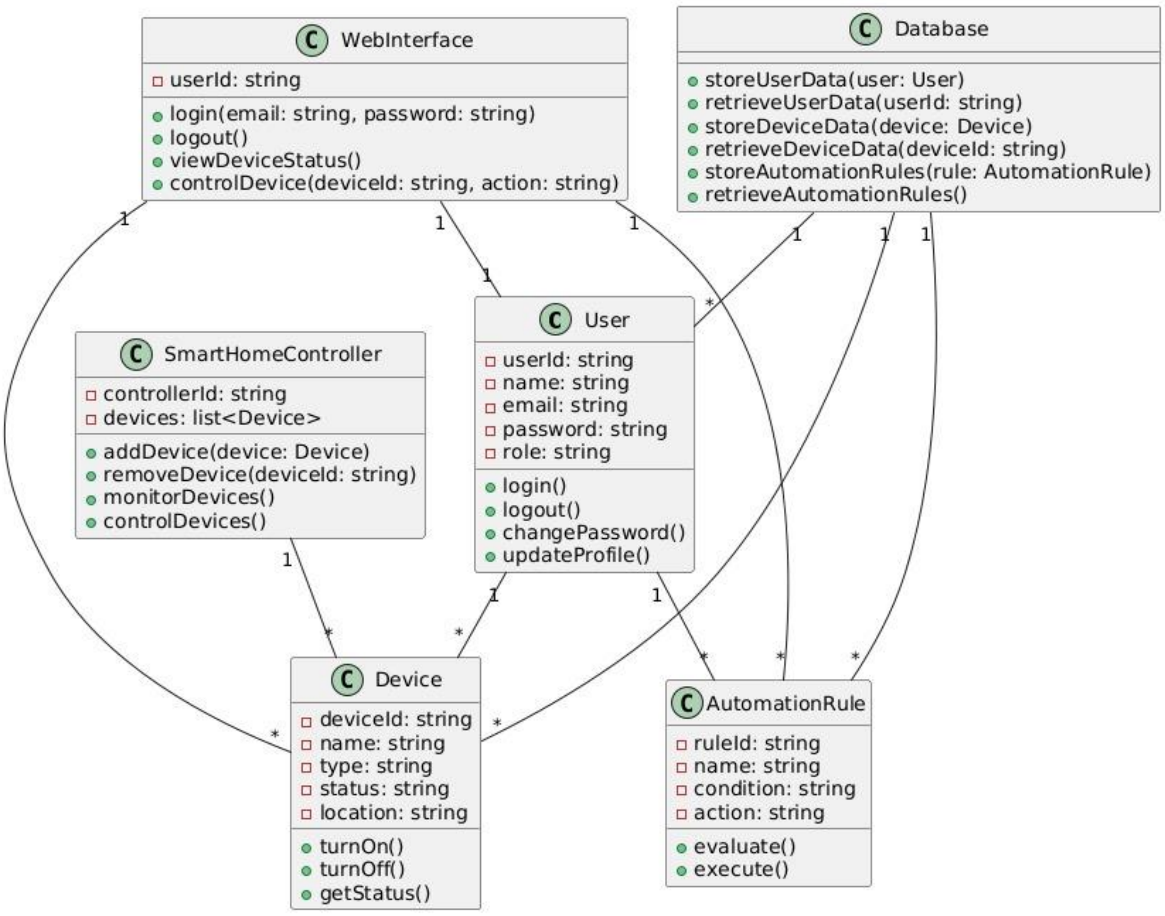


Figure 4.1.4 Class Diagram

4.2.4 Sequence Diagram

The Sequence Diagram provides a step-by-step view of how specific processes are executed over time within the IntelliNest system. This type of diagram is essential for understanding the sequence and timing of interactions between various system components.

For example, in a Device Control Request scenario, the process begins when a user sends a command (such as turning on a device) via the mobile or web application. The user interface then forwards the command to the Device Controller component, which first checks the User Authentication process to verify the user's permissions. Once authenticated, the Device Controller issues the appropriate command to the device, which then performs the requested action, such as turning on a light. After executing the action, the device sends a status update back to the Device Controller, which then records this status in the system database. The Notification System subsequently informs the user of the action's success or failure, and the User Interface displays the updated device status, completing the process.

This diagram highlights the sequence of interactions between components, showing how data and control messages flow in response to a user request.

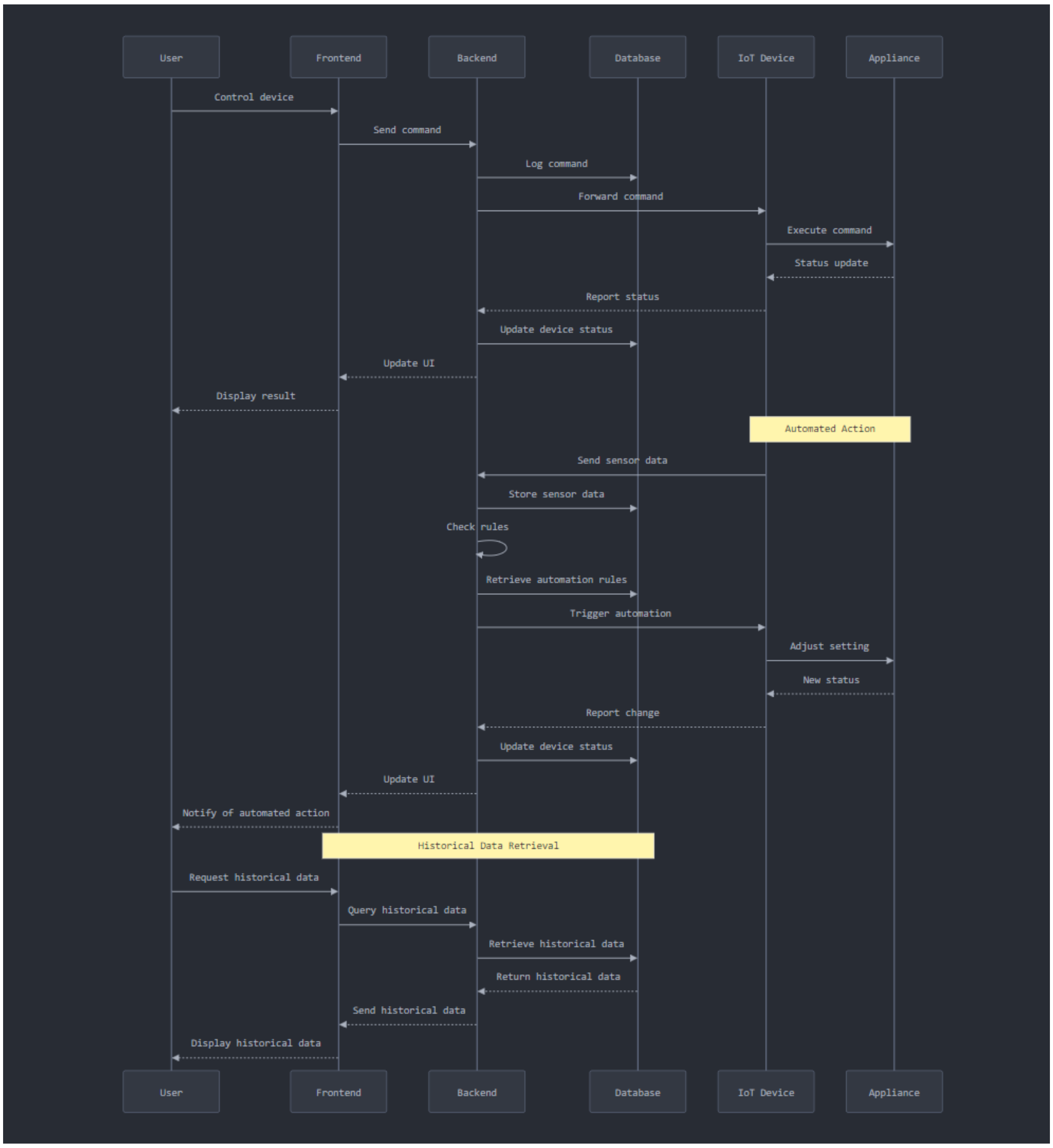


Figure 4.1.5 Sequence Diagram

5. Methodology and Testing

The IntelliNest project was conceptualized and implemented with a focus on creating a scalable, secure, and user-friendly IoT-based smart home automation system. Each module within the system was meticulously designed to support seamless user-device interaction, real-time monitoring, and remote control of appliances, all while prioritizing security and ease of use. The development process incorporated a modular approach, with rigorous testing at every stage to ensure each component met high standards for functionality, interoperability, and performance.

5.1 Module Description

The IntelliNest system consists of four core modules, each addressing distinct functional requirements. A modular design allowed for focused development on each part and enabled scalability, such that additional devices and functionalities can be easily integrated in the future.

5.1.1 User Interface (UI) Module

The User Interface serves as the main interaction point between users and the IntelliNest system. Designed with React.js, the UI aims to provide a visually appealing, responsive, and easy-to-navigate experience, empowering users to manage their home appliances through an intuitive web-based dashboard.

The interface design prioritizes accessibility, with large, clearly labeled controls, live status indicators, and configurable automation options. To enhance usability, the UI includes a control dashboard, where users can toggle devices on and off, set timers, view real-time device status, and receive alerts. This module also integrates scheduling features, allowing users to set routines for their appliances, which promotes energy efficiency and user convenience by automatically managing devices according to user preferences.

A critical aspect of the UI module is its responsive design, enabling consistent functionality across desktops, tablets, and mobile devices. User interactions are communicated to the backend in real-time, leveraging HTTP and WebSocket protocols for instantaneous feedback, ensuring that the system’s responsiveness meets users’ expectations for a smart home environment.

5.1.2 Backend Server Module

The backend server is the cornerstone of IntelliNest's operational framework, handling requests from the UI, processing data, executing commands, and ensuring secure data management. Developed with Node.js, the backend enables robust processing capabilities, including user authentication, data handling, device control logic, and automation rule execution.

User authentication is implemented with industry-standard security protocols, including multi-factor authentication (MFA) and encrypted session management, safeguarding the system against unauthorized access. This authentication mechanism is tightly integrated with the user data stored in MongoDB, providing a seamless yet secure login experience.

The backend further incorporates an API gateway and MQTT client to facilitate device communication. Using these components, the backend can manage device states, execute commands in response to user interactions, and monitor device health. Data processing functions aggregate usage metrics, which are then logged for historical analysis. Automation rules, defined by users through the UI, are executed by the backend based on data received from connected devices, offering a cohesive solution for real-time and scheduled appliance management.

5.1.3 IoT Device Communication Module

The IoT Device Communication Module is implemented using ESP8266 microcontrollers, which serve as the bridge between the user interface and physical devices. The ESP8266 devices connect to the internet via Wi-Fi and communicate with the backend using MQTT, with AWS IoT Core providing secure, scalable management for IoT protocols.

This module's primary responsibility is to relay commands from the backend to the

appropriate device and to send device status updates back to the server. Each device is programmed to subscribe to specific MQTT topics corresponding to control commands. By leveraging SSL/TLS encryption, the IoT devices ensure that data transmitted between devices and the server remains confidential and tamper resistant.
Additionally, the IoT module’s configuration allows for scalability. With its modular design, additional devices can be incorporated by assigning them unique MQTT topics. Each device can be programmed for specific functions, such as motion detection, temperature monitoring, or appliance control, which can be expanded to create an integrated smart home ecosystem.

5.1.4 Database Module

The Database Module, implemented using MongoDB, serves as the data management layer, storing all relevant user information, device configurations, usage logs, and automation rules. MongoDB was chosen for its scalability, flexibility, and compatibility with JSON-like documents, allowing for efficient data storage and retrieval across various types of user and device data.
This module logs each interaction, such as user commands and device status changes, facilitating historical analysis and user feedback on energy consumption patterns. Stored data also supports real-time monitoring by providing immediate access to device states and configurations, which the backend uses to respond to user actions promptly. Moreover, the database is secured with access control policies to prevent unauthorized access to sensitive data. Role-based access is implemented to ensure that only authenticated users and specific server processes can access or modify stored data, adding an extra layer of security to the system.

5.2 Testing

Testing for the IntelliNest system encompassed multiple stages and techniques to ensure functionality, interoperability, security, and user satisfaction. Testing methodologies included unit testing, integration testing, system testing, performance testing, usability testing, and security testing, with each approach focusing on distinct aspects of the system's performance and reliability.

5.2.1 Testing

Unit testing was conducted for each module to verify individual functionalities. For the UI module, tests focused on button interactions, responsiveness, and real-time status displays, using Jest to automate test cases. Backend testing targeted data validation, user authentication, API endpoint functionality, and data processing, utilizing Mocha to ensure each backend component handled requests accurately. MQTT.fx was used for testing device communication, confirming that each command sent from the backend to the ESP8266 devices was correctly processed and executed.

5.2.2 Integration Testing

Integration testing assessed the interoperability of modules and validated data flows across different components. By simulating real user interactions, we verified that the UI, backend, database, and IoT devices operated cohesively. The testing focused on:

1. UI-Backend Synchronization: Ensuring user commands initiated from the UI were accurately interpreted and processed by the backend.
2. Backend-Database Synchronization: Confirming the backend accurately stored and retrieved data from MongoDB.
3. Backend-IoT Device Communication: Testing MQTT communication to verify that commands were received by IoT devices promptly and without data loss.

Integration testing was essential to guarantee that the IntelliNest system provided a seamless experience across modules, ensuring that data integrity and user commands were maintained throughout the entire architecture.

5.2.3 System Testing

System testing validated end-to-end functionality under realistic conditions, assessing the IntelliNest system’s ability to execute remote control commands, provide real-time feedback, and execute scheduled tasks. Key system tests included:

- Remote Control Functionality: Users could turn devices on or off remotely, with instant feedback on device status.
- Scheduling and Automation: Devices activated or deactivated according to user-defined schedules, verified through direct observation and log analysis.
- Real-Time Feedback: Device status updates were reliably reflected in the UI, confirming real-time synchronization between modules.

System testing ensured the system was robust enough to support the core functionalities that make IntelliNest a practical and efficient smart home solution.

5.2.4 Performance Testing

Performance testing measured system responsiveness and reliability under various operational loads. Using JMeter, we conducted latency tests to monitor the time taken between user command initiation and device response. The system consistently achieved sub-second response times, demonstrating high responsiveness. Load tests evaluated backend stability when processing multiple user requests simultaneously, confirming that the system maintained performance under high traffic conditions, critical for scalability.

5.2.5 Usability Testing

Usability testing focused on the interface's intuitiveness, clarity, and accessibility. Test participants provided feedback on ease of navigation, the clarity of control labels, and the responsiveness of control elements. Based on user input, adjustments were made to improve the layout, ensuring a user-friendly experience that meets diverse needs. The final UI design incorporated visual cues and simplified control flows, enhancing the accessibility of the IntelliNest system for users of varying technical proficiency.

5.2.6 Security Testing

Security testing ensured that the IntelliNest system adhered to best practices in data protection and access control. Penetration testing identified potential vulnerabilities in the user authentication process, while vulnerability assessments were conducted on the backend to secure sensitive endpoints. SSL/TLS encryption on MQTT and HTTP protocols was rigorously tested, confirming secure data transmission between the backend and IoT devices. Multi-factor authentication was validated to ensure effective access control, reinforcing the security of the IntelliNest system against unauthorized access.

5.3 Conclusion

The IntelliNest project’s Methodology and Testing approach involved a structured and rigorous process, ensuring that each module met the necessary standards for functionality, interoperability, performance, and security. By employing comprehensive testing methods, the IntelliNest system was validated as a scalable, reliable, and user-friendly smart home solution. The detailed approach to each module’s design and testing provides a solid foundation for further development and scalability, allowing for future integration of advanced features like voice control, AI-driven automation, and enhanced energy monitoring. The IntelliNest system not only addresses key gaps in existing smart home solutions but also offers a user-centered experience that aligns with the needs of modern households.

6. Project Demonstration

The IntelliNest: Smart Living Redefined project provides a comprehensive, real-world demonstration of how accessible IoT technology can transform conventional home environments into interconnected smart homes. This system integrates the ESP8266 microcontroller, a compact yet powerful unit known for its Wi-Fi capabilities, with a 4-channel relay module to provide precise control over multiple home appliances. The primary aim of the demonstration is to showcase the system's core functionalities, security features, and adaptability in real-time control and monitoring of home appliances, all while emphasizing its low-cost advantage.

Setting up the system begins with configuring the ESP8266 microcontroller, which is programmed to connect to the user's home Wi-Fi network. This connection establishes an internet link between the microcontroller and a user interface accessible on mobile devices or desktop computers. As the gateway for remote communication, the ESP8266 allows users to send commands that control each appliance independently, turning them on or off based on specific needs. The hardware configuration itself is streamlined: the ESP8266 is wired to the 4-channel relay module, with each relay controlling a separate appliance. This setup enables users to manage multiple devices from one central unit, reducing the complexity and cost associated with individual control units for each appliance.

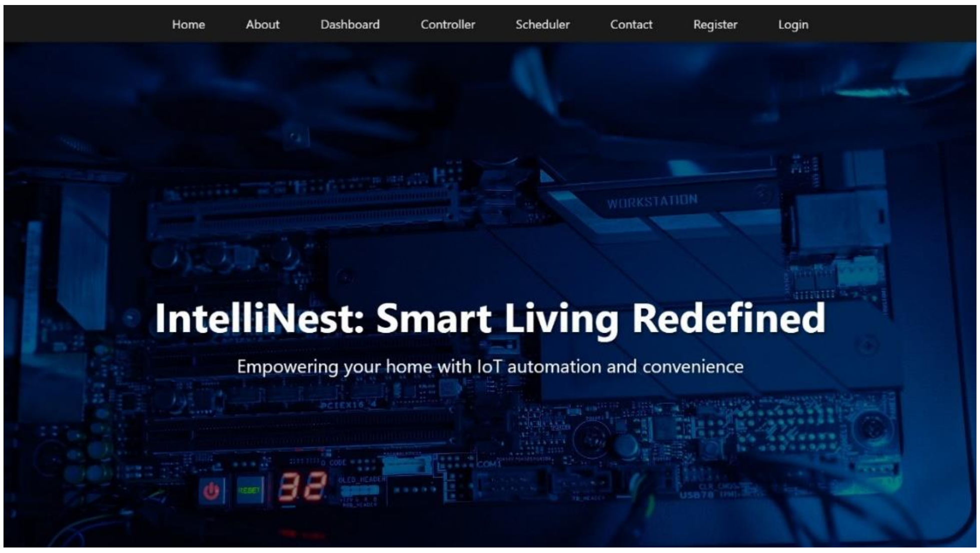
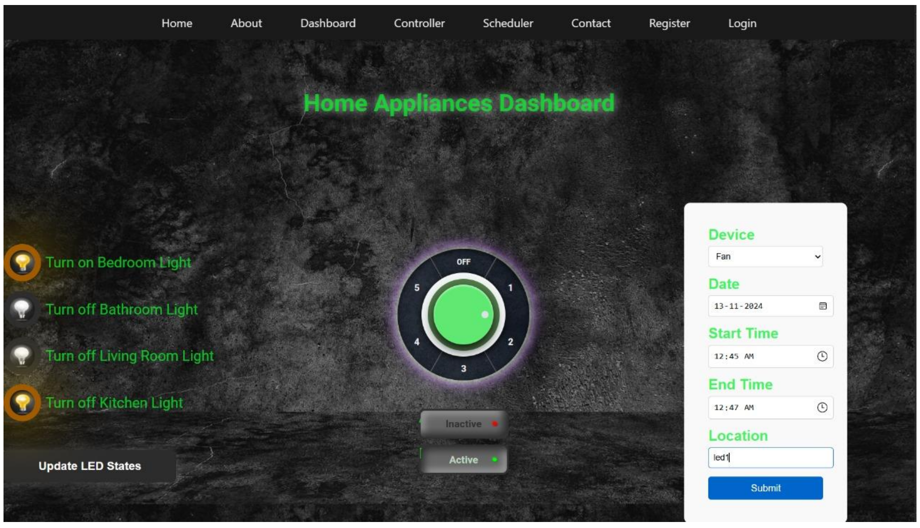

Home
About
Dashboard
Controller
Scheduler
Contact
Register
Login
IntelliNest: Smart Living Redefined
Empowering your home with IoT automation and convenience

Figure 6.1 Home Page

The user interface, developed as a mobile or web application, plays a vital role in the IntelliNest system by providing users with an intuitive, real-time dashboard for appliance control and monitoring. This interface displays the status of each connected device, updating instantly to reflect any change in appliance state. With the remote-control feature, users can operate appliances from any location with internet access, granting them greater flexibility and convenience. Additionally, the scheduling feature allows users to program appliances to operate automatically based on predefined routines. For instance, users might schedule lights to turn on at sunset and off at bedtime, or they might program heating systems to activate only during specific hours, ultimately enhancing energy efficiency and providing a level of convenience traditionally associated with premium smart home systems.

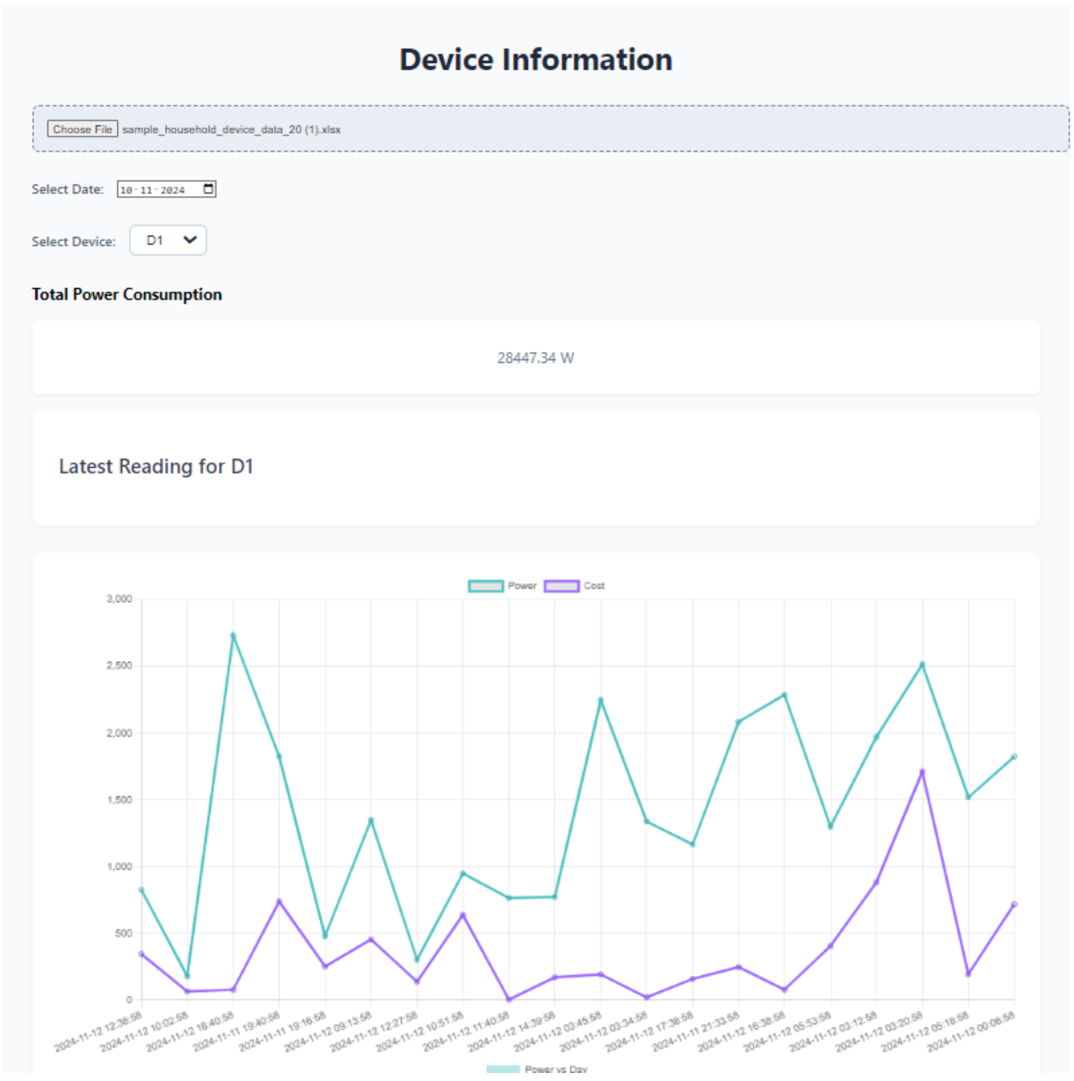


Device: Fan
Date: 13-11-2024
Start Time: 12:45 AM
End Time: 12:47 AM
Location: led1

Figure 6.2 Appliances Dashboard

Security is integral to the IntelliNest system, addressing the privacy and cybersecurity concerns that often accompany IoT devices. The use of SSL/TLS encryption ensures that data transmitted between the user interface and the ESP8266 is secure from interception or tampering. This encryption establishes a secure communication channel that protects user commands and device data from potential cyber threats. The project also includes plans for multi-factor authentication (MFA), where users would have an additional verification step—such as an OTP or biometric confirmation—adding an extra layer of security to the system. These security measures underscore IntelliNest's commitment to user privacy and data protection, providing a more secure alternative to many low-cost IoT systems that lack robust encryption.

Scalability is a defining feature of the IntelliNest system, making it adaptable to both small and large home environments. Through a modular design, users can add additional relay modules to control more devices as needed, thus transforming IntelliNest from a basic, four-appliance controller into a comprehensive home automation solution capable of managing lighting, climate control, security systems, and other essential home functions. This modularity is particularly beneficial in enabling gradual upgrades, where users can expand their smart home setup over time without needing to overhaul the existing system. As a result, IntelliNest caters to diverse user needs, from single-room control to whole-house automation.


Device Information

sample_household_device_data_20 (1).xlsx

Select Date: 10-11-2024

Select Device: D1

Total Power Consumption

28447.34 W

Latest Reading for D1

Figure 6.3 Power vs Cost Graph

In its demonstration, the IntelliNest project has successfully validated the functionality, reliability, and user-friendliness of its smart home control system. The feedback received indicates that users appreciate the real-time control, security measures, and scalability, all of which contribute to an enhanced living experience. With additional testing and iterative improvements, IntelliNest has the potential to set new standards in affordable, accessible smart home technology, offering a practical solution that democratizes access to IoT-based automation.

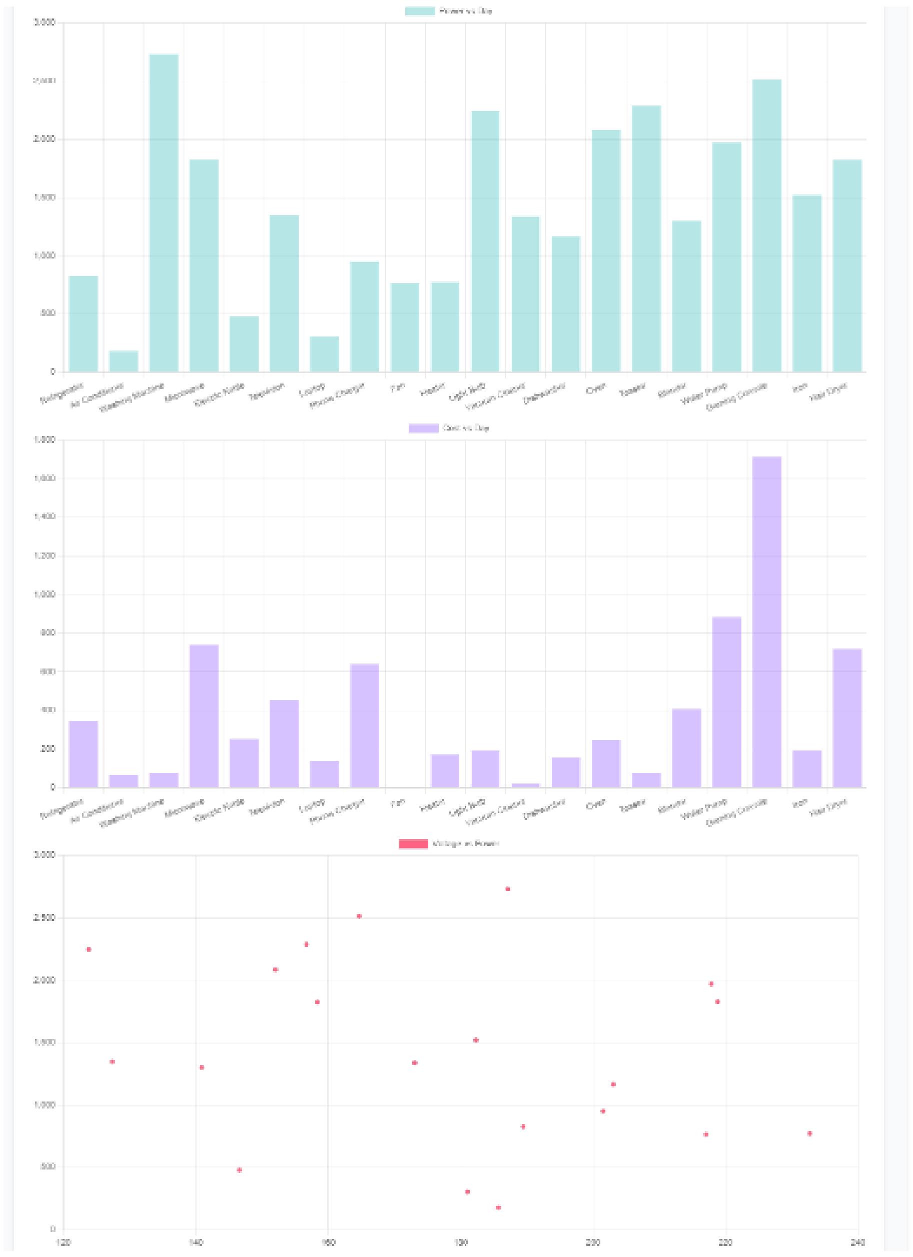


Figure 6.4 Analysis Graphs

7. Results and Discussion

Cost analysis is a crucial aspect of the IntelliNest project, as its goal is to deliver a competitive smart home system at a fraction of the cost of conventional market offerings. Targeting a retail price between ₹500 and ₹750, IntelliNest offers an affordable solution that retains core functionalities seen in higher-end smart home products, such as remote control, scheduling, and secure communication. This section provides an in-depth analysis of the project’s cost structure and explores how it compares to commercial alternatives.

The ESP8266 microcontroller is the cornerstone of the IntelliNest system, chosen for its affordability and versatility in IoT applications. Costing around ₹200 to ₹300, the ESP8266 combines Wi-Fi connectivity, processing power, and compatibility with relay modules, making it a highly efficient option for a low-cost smart home setup. By using the ESP8266, the project eliminates the need for additional controllers or hubs, keeping the design streamlined and cost-effective. The 4-channel relay module, costing between ₹100 and ₹150, further enhances the IntelliNest’s affordability by allowing one microcontroller to manage multiple appliances. This multi-channel approach significantly reduces the per-device cost, providing users with more control options without increasing the system’s complexity or price.

Additional components, such as connectors, wiring, and the enclosure, add roughly ₹50 to ₹100 to the overall cost, bringing the total material expenses to around ₹500. In mass production, economies of scale could reduce this cost even further, making it feasible to achieve the target price point of ₹500 to ₹750 per unit. This makes IntelliNest particularly accessible to a broader consumer base, including those who might be priced out of traditional smart home solutions.

In comparison to existing products on the market, IntelliNest offers substantial cost savings without compromising on essential features. High-end systems like Google Nest and Amazon Alexa Smart Home range from ₹5,000 to ₹20,000, depending on additional functionalities like voice control and proprietary smart home ecosystem integration. These systems, while highly integrated, often require specific hardware or subscriptions for advanced features, adding to the long-term expenses for users. In contrast, IntelliNest’s open-source design enables greater customization and expansion flexibility, allowing users to incorporate additional devices or modify functionalities based on their specific needs. Mid-range systems such as TP-Link’s Kasa Smart, priced between ₹3,000 and ₹6,000, also lack the modularity and customization options that IntelliNest provides, limiting users to pre-defined device compatibility and configurations.

Moreover, IntelliNest’s design for energy efficiency through automated scheduling contributes to long-term cost savings. By allowing users to program appliance operation based on their routines, the system reduces unnecessary energy usage and helps lower electricity bills. This energy optimization is particularly beneficial in the context of rising energy costs and environmental concerns, aligning with global trends toward more sustainable living practices. For instance, users can set the system to turn off lights in unoccupied rooms or power down appliances during off-peak hours, contributing to overall energy conservation and cost-effectiveness.

In conclusion, the cost analysis of IntelliNest demonstrates its value proposition as a low-cost, feature-rich smart home solution that combines affordability with flexibility and security. Its modularity, energy efficiency, and open-source design make it a sustainable and adaptable alternative to expensive commercial smart home systems, positioning it as an ideal choice for a broad range of consumers looking to integrate smart technology into their daily lives without significant financial investment.

Feature	**IntelliNest**	**Google Nest**	**Amazon Smart Home**	**TP-Link Kasa Smart**
Remote Control	Yes	Yes	Yes	Yes
Scheduling	Yes	Yes	Yes	Yes
Voice Control	Planned	Yes	Yes	Some Models
Security (SSL/TLS)	Yes	Yes	Yes	Yes
Multi-Factor Auth	Planned	Yes	Yes	Some Models
Energy Monitoring	Yes	Yes	Yes	Some Models
Open Source	Yes	No	No	No
Expandability	Modular	Limited	Limited	Limited
Approximate Cost (₹)	**500-750**	**5000-20000**	**5000-20000**	**3000-6000**

Table 7.1 Feature Comparison with Existing Smart Home Systems

8. Conclusion

The IntelliNest: Smart Living Redefined project represents a pioneering approach to affordable, scalable, and secure smart home automation. By utilizing the ESP8266 microcontroller and a 4-channel relay module, IntelliNest effectively bridges the gap between high-cost commercial smart home systems and the growing demand for accessible IoT solutions. Through this project, we have demonstrated that essential smart home functionalities—such as real-time remote control, scheduling, and security—can be achieved within a budget-friendly framework, enabling more users to enjoy the benefits of a smart home.

One of the key accomplishments of IntelliNest lies in its ability to provide real-time monitoring and control over household appliances, granting users unprecedented flexibility and convenience. The system's interface allows users to view and manage appliance statuses from anywhere with internet access, transforming how they interact with their home environments. This level of control is typically reserved for premium smart home systems, which can be prohibitively expensive. IntelliNest, however, brings these capabilities to a wider audience through its open-source and modular design, allowing users to build and customize their smart home systems according to their specific needs and budget constraints.

Security remains a top priority in the IntelliNest system. By integrating SSL/TLS encryption and multi-factor authentication, the project addresses critical concerns about privacy and data protection in IoT environments. These measures safeguard user data from unauthorized access, ensuring that only verified users can control the system. This approach sets IntelliNest apart from many low-cost IoT solutions, which often overlook security in favor of affordability. By prioritizing security alongside affordability, IntelliNest provides users with a reliable and trustworthy smart home solution that upholds industry standards for data protection.

The scalability of IntelliNest is another crucial feature that makes it suitable for a diverse range of users. The modular architecture allows for easy expansion, enabling users to add more devices over time and adapt the system as their needs evolve. This scalability is essential for users in larger homes or those seeking more comprehensive control over various appliances and systems. Furthermore, the open-source nature of IntelliNest fosters community contributions and customization, encouraging users to integrate additional features like voice control, predictive analytics, and advanced automation. Such flexibility ensures that IntelliNest can keep pace with technological advancements and remain relevant in the rapidly evolving smart home market.

Looking ahead, IntelliNest has the potential to expand further, integrating machine learning algorithms for predictive automation, enabling voice control through platforms like Google Assistant or Amazon Alexa, and providing detailed energy monitoring for optimized household efficiency. These enhancements would elevate IntelliNest from a basic control system to an advanced, intelligent home automation solution capable of anticipating user needs and making autonomous decisions based on real-time data. By adopting a sustainable, user-centric approach to smart home technology, IntelliNest has the capacity to reshape smart living, making it not only attainable but also adaptable and forward-looking.

In summary, the IntelliNest project redefines smart home technology by making it accessible, secure, and adaptable to individual needs. Through its focus on affordability, security, and scalability, IntelliNest democratizes access to IoT-based automation, empowering users to take control of their home environments while promoting energy-conscious living. By filling the gaps left by existing market solutions, IntelliNest paves the way for more inclusive smart home technology, offering a sustainable model that aligns with the future of intelligent, connected living spaces. This project thus serves as a testament to the potential of affordable IoT innovation, making smart living a realistic goal for diverse households worldwide.

7. REFERENCES

[1] Buyya, R., & Dastjerdi, A. V. (Eds.). (2016). *Internet of Things: Principles and Paradigms*. Morgan Kaufmann.

[2] Kranz, M. (2016). *Building the Internet of Things: Implement New Business Models, Disrupt Competitors, Transform Your Industry*. Wiley.

[3] Goodwin, S. (2013). *Smart Home Automation with Linux and Raspberry Pi*. Apress.

[4] Lea, P. (2018). *Programming the Internet of Things*. O'Reilly Media.

[5] Arduino. (n.d.). Arduino - Home. Retrieved September 17, 2024, from https://www.arduino.cc

[6] ESP8266 Community Forum. (n.d.). ESP8266 Community Forum - Index. Retrieved September 17, 2024, from https://www.esp8266.com

[7] Instructables. (n.d.). Instructables. Retrieved September 17, 2024, from https://www.instructables.com

[8] Hackster.io. (n.d.). Hackster.io - The community dedicated to learning hardware. Retrieved September 17, 2024, from https://www.hackster.io

[10] SparkFun Electronics. (n.d.). SparkFun Electronics. Retrieved September 17, 2024, from https://www.sparkfun.com

[11] IoT For All. (n.d.). IoT For All - Your Authoritative Internet of Things Resource. Retrieved September 17, 2024, from https://www.iotforall.com

[12] IEEE Xplore. (n.d.). IEEE Xplore Digital Library. Retrieved September 17, 2024, from https://ieeexplore.ieee.org

[13] ResearchGate. (n.d.). ResearchGate | Find and share research. Retrieved September 17, 2024, from https://www.researchgate.net

[14] Google Scholar. (n.d.). Google Scholar. Retrieved September 17, 2024, from https://scholar.google.com

APPENDIX A – Sample Code

ESP8266 FLASHED CODE:

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include "env.h"

// WiFi credentials
const char WIFI_SSID[] = "motoedge50fusion";
const char WIFI_PASSWORD[] = "01011010";

// Device name from AWS
const char THINGNAME[] = "ESP8266";

// MQTT broker host address from AWS
const char MQTT_HOST[] = "a2dqhotbywifsc-ats.iot.eu-north-1.amazonaws.com";

// MQTT topics
const char AWS_IOT_PUBLISH_TOPIC[] = "esp8266/status";
const char AWS_IOT_SUBSCRIBE_TOPIC[] = "esp8266/control";

// Define LED pins
const int LED_1 = D1;
const int LED_2 = D2;
const int LED_3 = D3;
const int LED_4 = D4;

// Timezone offset from UTC
const int8_t TIME_ZONE = -5;

// WiFiClientSecure object for secure communication
WiFiClientSecure net;

// X.509 certificate for the CA
BearSSL::X509List cert(cacert);

// X.509 certificate for the client
BearSSL::X509List client_crt(client_cert);

// RSA private key
BearSSL::PrivateKey key(privkey);
```

```
// MQTT client instance
PubSubClient client(net);

// LED states
bool led_states[4] = {false, false, false, false};

void NTPConnect() {
  Serial.print("Setting time using SNTP");
  configTime(TIME_ZONE * 3600, 0, "pool.ntp.org", "time.nist.gov");
  time_t now = time(nullptr);
  while (now < 1510592825) {
    delay(500);
    Serial.print(".");
    now = time(nullptr);
  }
  Serial.println("done!");
}

// Process incoming messages and control LEDs
void messageReceived(char *topic, byte *payload, unsigned int length) {
  Serial.print("Received [");
  Serial.print(topic);
  Serial.print("]: ");

  // Create a buffer for the payload
  char message[length + 1];
  for (int i = 0; i < length; i++) {
    message[i] = (char)payload[i];
    Serial.print(message[i]);
  }
  message[length] = '\0';
  Serial.println();

  // Parse JSON message
  StaticJsonDocument<200> doc;
  DeserializationError error = deserializeJson(doc, message);

  if (error) {
    Serial.println("Failed to parse JSON message");
    return;
  }
 // Process LED controls
  if (doc.containsKey("led1")) {
    led_states[0] = doc["led1"].as<bool>();
    digitalWrite(LED_1, led_states[0] ? LOW : HIGH);
  }
  if (doc.containsKey("led2")) {
    led_states[1] = doc["led2"].as<bool>();
    digitalWrite(LED_2, led_states[1] ? LOW : HIGH);
  }
  if (doc.containsKey("led3")) {
    led_states[2] = doc["led3"].as<bool>();
    digitalWrite(LED_3, led_states[2] ? LOW : HIGH);
  }
  if (doc.containsKey("led4")) {
    led_states[3] = doc["led4"].as<bool>();
    digitalWrite(LED_4, led_states[3] ? LOW : HIGH);
  }

  // Publish LED states back to AWS
  publishStatus();
}

void connectAWS() {
  delay(3000);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.println(String("Attempting to connect to SSID: ") + String(WIFI_SSID));

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(1000);
  }

  NTPConnect();

  net.setTrustAnchors(&cert);
  net.setClientRSACert(&client_crt, &key);

  client.setServer(MQTT_HOST, 8883);
  client.setCallback(messageReceived);

  Serial.println("Connecting to AWS IoT");

  while (!client.connect(THINGNAME)) {
    Serial.print(".");
    delay(1000);
  }

  if (!client.connected()) {
    Serial.println("AWS IoT Timeout!");
    return;
  }

  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);
  Serial.println("AWS IoT Connected!");
}

// Publish LED status to AWS
void publishStatus() {
  StaticJsonDocument<200> doc;
  doc["led1"] = led_states[0];
  doc["led2"] = led_states[1];
  doc["led3"] = led_states[2];
  doc["led4"] = led_states[3];

  char jsonBuffer[200];
  serializeJson(doc, jsonBuffer);

  client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}

void setup() {
  Serial.begin(115200);

  // Initialize LED pins
  pinMode(LED_1, OUTPUT);
  pinMode(LED_2, OUTPUT);
  pinMode(LED_3, OUTPUT);
  pinMode(LED_4, OUTPUT);

  // Set initial LED states
  connectAWS();
  digitalWrite(LED_1, LOW);
  digitalWrite(LED_2, LOW);
  digitalWrite(LED_3, LOW);
  digitalWrite(LED_4, LOW);
}

void loop() {
  if (!client.connected()) {
    connectAWS();
  }
  client.loop();
}
```

SERVER CODE: (AWS integration)

```
const AWS = require('aws-sdk');
const DeviceData = require('../model/device_model'); // Import the DeviceData model

// Constants for calculations
const VOLTAGE = 5; // Example: LED Voltage
const CURRENT = 0.02; // Example: LED Current in Amps
const COST_PER_KWH = 0.12; // Cost per kWh in USD

AWS.config.update({
  region: 'eu-north-1',
  accessKeyId: 'AKIAYKFQQOR4T6NGAJMY',
  secretAccessKey: 'CNoOO60QlQ3xQushQxsDY7Z1s0ft6dPLrsIq96B1'
});

const iotData = new AWS.IotData({
  endpoint: 'a2dqhotbywifsc-ats.iot.eu-north-1.amazonaws.com'
});

// Calculate wattage
const calculateWatt = (voltage, current) => voltage * current;

// Calculate cost based on usage
const calculateCost = (watt, hours, costPerKWh) => (watt * hours) / 1000 * costPerKWh;

const controlLed = async (req, res) => {
  try {
    const { led1, led2, led3, led4 } = req.body;

    if (led1 === undefined || led2 === undefined || led3 === undefined || led4 === undefined) {
      return res.status(400).json({ message: 'Invalid LED data' });
    }

    // Create an object to track the on/off state of each LED
    const ledData = { led1, led2, led3, led4 };
    const timestamp = new Date();

    // Publish data to AWS IoT (keep as is)
    const params = {
      topic: 'esp8266/control',
      payload: JSON.stringify(ledData),
      qos: 1
    };

    iotData.publish(params, async (err, data) => {
      if (err) {
        console.error('Error publishing message', err);
        return res.status(500).json({ message: 'Error sending LED control command', error:
err.message });
      }

      // Process each LED
      const leds = ['led1', 'led2', 'led3', 'led4'];
      for (let i = 0; i < leds.length; i++) {
        if (ledData[leds[i]] !== undefined) {
```

```
        const watt = calculateWatt(VOLTAGE, CURRENT);
        const cost = calculateCost(watt, 1, COST_PER_KWH); // Assuming 1 hour of usage
for simplicity
        const state = ledData[leds[i]] ? 'on' : 'off'; // Set state as 'on' or 'off'

        const deviceData = new DeviceData({
          device: leds[i],
          time: timestamp,
          voltage: VOLTAGE,
          power: watt,
          watt: watt,
          cost: state === 'on' ? cost : 0, // Only store cost if the LED is on
          state: state
        });

        await deviceData.save();
      }
    }

    console.log('Message sent successfully', data);
    return res.status(200).json({ message: 'LED control command sent successfully', data:
data });
  });
 } catch (error) {
   console.error('Error in controlLed:', error);
   return res.status(500).json({ message: 'Unable to control LEDs', error: error.message });
 }
};

// Controller function to get device data
const getDeviceData = async (req, res) => {
 try {
   const deviceData = await DeviceData.find();

   if (deviceData.length === 0) {
     return res.status(404).json({ message: 'No device data found' });
   }

   return res.status(200).json({ data: deviceData });
 } catch (error) {
   console.error('Error fetching device data', error);
   return res.status(500).json({ message: 'Error fetching device data', error: error.message });
 }
};

module.exports = { controlLed, getDeviceData };
```