A PRELIMENERY REPORT ON

INTELLIHOME

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

SUBMITTED BY

STUDENT NAME	Exam No
Kavishwar Prashant Khankari	B1900704235
Veeraja Prashant Ghate	B1900704225
Aryan Vivek Desai	B1900704218
Mayur Maludeo Shriram	B1900704268

DEPARTMENT OF COMPUTER ENGINEERING

PVG's COLLEGE OF ENGINEERING AND TECHNOLOGY
& G K PATE(WANI) INSTITUTE OF MANAGEMENT

44, VIDYANAGARI, PARVATI, PUNE 411009

SAVITRIBAI PHULE PUNE UNIVERSITY
2024-25

CERTIFICATE

This is to certify that the project report entitles

“ INTELLIHOME”

Submitted by

STUDENT NAME	Exam No
Kavishwar Prashant Khankari	B1900704235
Veeraja Prashant Ghate	B1900704225
Aryan Vivek Desai	B1900704218
Mayur Maludeo Shriram	B1900704268

is a Bonafide student of this institute and the work has been carried out by him under the supervision of Prof. **A. M. Bhadgale** and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering).

Prof. A. M. Bhadgale
Guide
Department of Computer Engineering
Engineering

Prof. U. M. Kalshetti
Head,
Department of Computer

Dr. M R Tarambale
I/C Principal,
PVG’s College of Engineering and Technology & GKPIM, Pune – 09

Place: Pune
Date:

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to PVG's College of Engineering & Technology and G K Pate (Wani) Institute of Management, Pune, for providing us with the resources, infrastructure, and academic environment that made this project possible. We are especially grateful to our internal guide, A. M. Bhadgale, whose valuable guidance, technical expertise, and continuous support have been instrumental in shaping our project. Their encouragement and feedback throughout each stage of the project have allowed us to improve our work significantly.

Their guidance has not only contributed to our understanding of software planning but has also enhanced our technical and analytical skills.

Additionally, we would like to thank our faculty members and classmates for their constructive feedback, as well as our families for their unwavering support, patience, and encouragement, which motivated us to put forth our best efforts.

NAME OF THE STUDENTS

Mayur Maludeo Shriram
Aryan Vivek Desai
Veeraja Prashant Ghate
Kavishwar Prashant Khankari

ABSTRACT

In an era where technology is integral to daily life, there is a growing demand for home automation systems that offer convenience, efficiency, and enhanced security. **Intellihome** addresses this need by developing an intelligent, IoT-based home automation system that utilizes machine learning to personalize and dynamically manage household functions such as lighting, climate control, and security.

The current market for home automation systems often falls short in delivering adaptive solutions that can respond to individual user behaviours and preferences. Most existing systems operate on predefined settings and schedules, which limits their ability to provide a truly personalized experience and reduces their efficiency. Additionally, some systems lack robust security measures, creating vulnerabilities in data privacy and access control.

To solve these issues, our project proposes a modular home automation system that integrates IoT devices with machine learning algorithms. The system is designed to analyse and learn from resident behaviour patterns, enabling it to automatically adjust settings based on real-time and historical data. This adaptive automation minimizes energy waste, enhances user convenience, and strengthens security by intelligently controlling access and monitoring suspicious activities. The architecture incorporates sensors and actuators that communicate via a central processing unit, which then applies machine learning models to predict and respond to user needs.

This project aims to provide a foundation for the development of next-generation smart home systems, ultimately contributing to a future where technology seamlessly integrates with everyday life.

TABLE OF CONTENTS

LIST OF ABBREVATIONS vii
LIST OF FIGURES viii

Sr. No.	Title of Chapter	Page No.
01	**Introduction**	1
1.1	**Motivation**	1
1.2	**Problem Definition**	1
02	**Literature Survey**	2
2.1	**Human activity datasets for smart home applications**	2
2.2	**User behaviour prediction in smart homes**	3
2.3	**Face recognition-based visitor authentication**	3
03	**Software Requirements Specification**	5
3.1	**Introduction**	5
	3.1.1 Project Scope	5
	3.1.2 User Classes and Characteristics	6
	3.1.3 Assumptions and Dependencies	7
3.2	**Functional Requirements**	9
	3.2.1 Smart Lighting Control	9
	3.2.2 Temperature Automation	10
	3.2.3 Security Monitoring	10
	3.2.4 Personalized User Profiles	11
	3.2.5 Centralized Control System	12
	3.2.6 Integration of IoT Devices	12
	3.2.7 Remote Access	13
	3.2.8 Energy Efficiency Monitoring	14
3.3	**External Interface Requirements (If Any)**	14
	3.3.1 Hardware Interfaces	14
	3.3.2 Software Interfaces	15
	3.3.3 Communication Interfaces	15
3.4	**Nonfunctional Requirements**	16
	3.4.1 Performance Requirements	16
	3.4.2 Safety Requirements	17
	3.4.3 Security Requirements	17
	3.4.4 Software Quality Attributes	18
3.5	**System Requirements**	19
	3.5.1 Database Requirements	19
	3.5.2 Software Requirements	20
	3.5.3 Hardware Requirements	21
	3.5.4 Communication Interfaces	23
3.6	**Analysis Models: SDLC Model to be applied**	23
3.7	**System Implementation Plan**	25
04	**System Design**	29

	4.1	**System Architecture and module description**		29
	4.2	**Data Flow Diagrams**		30
		4.2.1	Data Flow Diagram – 0	30
		4.2.2	Data Flow Diagram – 1	31
		4.2.3	Data Flow Diagram – 2	31
	4.3	**Entity Relationship Diagrams**		32
	4.4	**UML Diagrams**		33
		4.4.1	Usecase Diagram	33
		4.4.2	Class Diagram	33
		4.4.3	Activity Diagram	34
		4.4.4	Sequence Diagram	35
		4.4.5	Component Diagram	36
05		**Other Specification**		37
	5.1	**Advantages**		37
	5.2	**Limitations**		37
	5.3	**Applications**		38
06		**Conclusions & Future Work**		39
		Appendix A:		40
		Appendix B:		42
		Appendix C (Plagiarism Report)		43
		References		44

I. LIST OF ABBREVATIONS

ABBREVIATION	ILLUSTRATION
IoT	Internet of Things
ML	Machine Learning
SRS	Software Requirements Specification
VPN	Virtual Private Network
IP	Internet Protocol
MQTT	Message Queuing Telemetry Transport
ANN	Artificial Neural Network
SSL/TLS	Secure Sockets Layer / Transport Layer Security
CPU	Central Processing Unit
DFD	Data Flow Diagram
ER	Entity-Relationship
BLE	Bluetooth Low Energy
SDLC	Software Development Life Cycle
SQL	Structured Query Language
GUI	Graphical User Interface

II. LIST OF FIGURES

FIGURE	ILLUSTRATION	PAGE NO.
1	Project Timeline	28
2	Data Flow Diagram – 0	30
3	Data Flow Diagram – 1	30
4	Data Flow Diagram – 2	31
5	Entity Relationship Diagram	32
6	Usecase Diagram	33
7	Class Diagram	33
8	Activity Diagram	34
9	Sequence Diagram	35
10	Component Diagram	36

1. INTRODUCTION

1.1. MOTIVATION

The evolution of technology has opened opportunities to create smarter, more connected home environments that enhance everyday living. With an increasing focus on convenience, security, and energy conservation, there is a strong demand for intelligent home automation solutions. Tasks that are routinely managed by residents, such as controlling lighting, adjusting temperature, and ensuring home security, could be handled more efficiently through automation. Integrating Internet of Things (IoT) devices with Machine Learning provides a powerful approach to building a system that can make real-time decisions, learn from resident behaviors, and reduce the need for human intervention, thereby making daily life smoother and more customized.

1.2. PROBLEM DEFINITION

Current home automation systems often lack adaptability to individual user preferences and typically require manual configuration, which limits their usability and flexibility. These systems generally do not analyze resident behavior to personalize control and optimize resource use. This project addresses this gap by developing an IoT-based home automation system that uses Machine Learning to study resident patterns and habits, allowing for automated control over aspects like lighting, temperature, and security. By implementing a solution that dynamically evolves based on user behavior, this project aims to enhance convenience, security, and energy efficiency in modern homes, creating a responsive and intelligent living environment.

2. LITERATURE SURVEY

2.1 LITERATURE REVIEW ON HUMAN ACTIVITY DATASETS FOR SMART HOME APPLICATIONS

1. Introduction to Activity Datasets in Smart Homes
    - Human activity datasets are crucial for developing machine learning models that recognize daily activities in smart homes, aiding in long-term health monitoring. This study presents the ARAS dataset, gathered from two multi-resident homes over two months, containing 27 different activities captured by 20 sensors.
2. Unique Features of ARAS Dataset
    - Unlike many existing datasets that focus on single-resident homes, ARAS records multi-resident interactions, offering a diverse activity set. The dataset also avoids intrusive sensors (e.g., cameras), instead using ambient sensors like contact, proximity, and temperature sensors to maintain privacy.
3. Comparison to Other Datasets
    - The ARAS dataset is notable for its naturalistic data collection, where residents perform unscripted daily activities, unlike controlled lab datasets. Compared to datasets such as Casas or UvA, ARAS has a broader range of activities and data points, making it valuable for real-world activity recognition models.
4. Preliminary Data Analysis and Model Testing
    - Initial tests with Hidden Markov Models on frequent activities (e.g., sleeping, eating) achieved up to 76% accuracy, indicating the dataset's potential for further refining machine learning algorithms.

2.2 LITERATURE REVIEW ON USER BEHAVIOR PREDICTION IN SMART HOMES

1. Background
    - Advances in machine learning enable the use of extensive smart home data for predicting user behaviour, enhancing automation. This paper introduces the Unsupervised User Behaviour Prediction (UUBP) algorithm, which uses neural networks for adaptive, unsupervised learning.
2. Core Methodology
    - *ANN Initialization*: The UUBP algorithm initializes behaviour clusters without preset requirements.
    - *Forgetting Curve*: By prioritizing recent data, the algorithm maintains relevant predictions over time.
3. Comparison with Previous Methods
    - Traditional models, like K-means, struggle with high-dimensional data and static clustering. More recent methods improve prediction but often lack flexibility for changing behaviours, which UUBP addresses effectively.
4. Results and Future Work
    - Testing showed UUBP outperforms other clustering algorithms in adapting to user behaviour changes. Future work aims to enhance inter-device behaviour prediction for a more seamless smart home experience

2.3 LITERATURE REVIEW ON FACE RECOGNITION-BASED VISITOR AUTHENTICATION

1. Face Recognition in Security Systems
    - This study implements a face recognition system on a Jetson Nano, integrating CCTV for real-time visitor authentication in smart homes. This approach addresses rising demands for secure, automated access control.
2. System Design and Technology
    - *Tiny-YOLOv3 Deep Learning Model*: Used for fast, efficient face detection, Tiny-YOLOv3 provides a balance of accuracy and speed.

- *Hardware Comparison*: Jetson Nano outperformed Raspberry Pi with 86.3% accuracy at 6.5 FPS, making it suitable for real-time detection.
- *Key Feature Detection*: Recognizes facial features even with variations like glasses or masks, reducing false positives.

3. Authentication and Security Protocols
    - *Process*: Visitor authentication relies on face matching against stored images, triggering alarms and additional security steps if unauthorized.
    - *Anti-Peeking Measures*: Detects and alerts against password-peeking attempts, requiring a complex password if suspicious behaviour is identified.
4. Performance and Practical Use
    - Jetson Nano's performance confirms its practicality for home security, effectively recognizing and logging authorized visitors.
5. Future Directions
    - Suggested improvements include refining accuracy and scalability for larger security applications.

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1. INTRODUCTION

The Software Requirements Specification (SRS) document describes the intended functionality and operational characteristics of the IoT-based Home Automation System. This system is designed to enhance the convenience, security, and energy efficiency of a modern home by integrating IoT devices with machine learning capabilities to create a smart, adaptive environment. This document includes details about the scope, target users, assumptions, and dependencies essential to the success of the project.

3.1.1 Project Scope

The scope of this project is to develop an intelligent home automation solution that uses IoT and Machine Learning to optimize household management and safety. The primary goals are to provide personalized automation, reduce human intervention, and deliver a seamless user experience for residential control. The system will feature:

1. **Smart Lighting and Temperature Control**
    - Adaptive automation that adjusts lighting and temperature based on user behaviours, such as presence and time of day.
    - Allows for manual overrides and scheduled settings through a mobile or web app.
2. **Security Monitoring and Threat Detection**
    - Continuous monitoring of security cameras with machine learning algorithms for threat detection, including unauthorized access alerts.
    - Real-time notifications sent to the user in case of a potential security breach.
3. **Centralized Control and Monitoring Platform**
    - An app that consolidates control over all connected devices, including lighting, temperature, and security systems.
    - Provides live updates on home conditions, system status, and usage statistics.

4. **Remote Access and Control**
    - Cloud-based functionality enabling users to access and control their home systems remotely from any location.
    - Ensures secure data access with encrypted connections.
5. **Energy Usage Optimization**
    - Monitors real-time energy consumption of connected appliances and provides insights into reducing unnecessary power usage.
    - Includes automated control to turn off devices when not in use, contributing to energy savings.

Future Scalability: The system architecture is modular, allowing for easy upgrades and integration of new devices or functionalities, such as additional IoT sensors, machine learning features, or enhanced user settings. This design ensures longevity and adaptability as technology advances.

3.1.2 User Classes and Characteristics

The project considers three primary user classes with specific characteristics and requirements:

1. **Home Residents (Primary Users)**
    - **Characteristics**:
        - Residents are the main users of the system and vary in technical expertise, ranging from tech-savvy individuals to those with limited knowledge of smart technology.
        - They have regular access to mobile or web applications and are familiar with operating basic app functions.
    - **Requirements**:
        - **Intuitive Interface**: Easy-to-navigate app that enables simple control of home automation.
        - **Personalization**: The ability to set custom preferences for lighting, temperature, and security.
        - **Real-Time Monitoring**: Access to real-time updates on home conditions, security status, and energy consumption.

        - **Alerts and Notifications**: Instant notifications in the event of security threats or system malfunctions.

2. **System Administrators**
    - **Characteristics**:
        - System administrators are responsible for the maintenance, troubleshooting, and configuration of the system. They may have access to backend controls.
    - **Requirements**:
        - **Advanced Control and Configuration**: Access to device settings, logs, and configurations.
        - **Remote Diagnostics**: Tools for remote monitoring and problem-solving.
        - **User Management**: Ability to manage user permissions and control access to various system features.
3. **Guests**
    - **Characteristics**:
        - Temporary users who might occasionally require access to basic functionalities within the system, such as family members or visitors.
        - They may have limited permissions granted by residents.
    - **Requirements**:
        - **Basic Access**: Control of certain features, like lighting, without access to sensitive controls or data.
        - **Time-Restricted Access**: Permissions that can be granted temporarily and revoked as needed by the primary users.

3.1.3 Assumptions and Dependencies

This section defines the assumptions and dependencies required for the system to function optimally.

- **Assumptions**:
    - **Reliable Internet Connection**: The system relies on a stable internet connection for real-time data communication, cloud integration, and

remote access. In the case of connection interruptions, basic local control may be available, but full functionality could be limited.

- **Availability of Compatible IoT Devices**: The setup assumes the presence of compatible IoT devices like motion sensors, temperature sensors, cameras, and other connected devices. These devices need to adhere to the standards and communication protocols supported by the system.
- **User Access to Mobile or Web Applications**: Users are expected to have access to smartphones, tablets, or computers to interact with the control application. The application should be compatible with both Android and iOS, as well as with major web browsers.

- **Dependencies**:
  - **Third-Party IoT Hardware**: The system relies on external IoT hardware, such as sensors and cameras, which must be compatible with the core platform. Changes in the availability, firmware, or specifications of these devices may impact functionality.
  - **Machine Learning Libraries (TensorFlow, OpenCV)**: Machine learning models for behavior analysis, threat detection, and personalized automation rely on libraries like TensorFlow and OpenCV. Updates or changes in these libraries may affect performance or require adjustments to the software.
  - **Data Security and Encryption Standards**: The system must comply with security protocols to protect user data during transmission and storage. This includes SSL/TLS for encrypted communication and secure storage practices for sensitive information.
  - **Software Compatibility**: The application relies on certain software and operating systems (e.g., Raspbian for Raspberry Pi, or other OS for PCs) to support IoT devices and run machine learning models. Regular updates may be required to ensure compatibility and security.

3.2. FUNCTIONAL REQUIREMENTS

The following section describes the functional features and requirements of the IoT-based Home Automation System, detailing the expected behaviour of the system and how it meets user needs.

3.2.1. System Feature 1: Smart Lighting Control

- **Description**: This feature automates the lighting of the home based on occupancy, time of day, and user preferences. It aims to reduce energy consumption and provide convenience to the user.
- **Functional Requirements**:
    1. **Occupancy-Based Lighting**:
        - Use motion sensors to detect presence in a room. When motion is detected, lights turn on, and when no motion is detected for a set period, lights turn off.
    2. **Time-Based Lighting Control**:
        - Allow the user to set schedules for when the lights should be on or off based on the time of day (e.g., automatic dimming at night).
    3. **Manual Override**:
        - Enable users to manually control lights via a centralized mobile/web application.
    4. **Scene Control**:
        - Allow the user to create lighting scenes (e.g., "movie night" with dimmed lights).
    5. **Integration with Other Automation Features**:
        - Sync lighting settings with temperature automation for energy efficiency (e.g., turning lights off when HVAC is off).

3.2.2. System Feature 2: Temperature Automation

- **Description**: This feature ensures the home's temperature is adjusted automatically based on user preferences, room occupancy, and external environmental conditions to enhance comfort and energy savings.
- **Functional Requirements**:
    1. **User-Defined Temperature Preferences**:
        - Allow users to set preferred temperature ranges for each room (e.g., 22°C for living room, 24°C for bedroom).
    2. **Occupancy-Based Temperature Control**:
        - Use motion sensors or occupancy data to adjust the temperature only in rooms that are in use, minimizing energy consumption.
    3. **Adaptive Temperature Adjustments**:
        - Automatically adjust the temperature based on the time of day, external weather conditions (e.g., increasing heating during cold weather), and user behaviour patterns.
    4. **Manual Control via App**:
        - Users can manually adjust the temperature settings through the mobile/web application at any time.
    5. **Energy Optimization**:
        - Automatically turn off or adjust heating/cooling when rooms are unoccupied or when windows/doors are open.

3.2.3. System Feature 3: Security Monitoring

- **Description**: The system provides continuous monitoring of the home through video surveillance and threat detection using machine learning algorithms, ensuring security and peace of mind for residents.
- **Functional Requirements**:
    1. **Real-Time Video Surveillance**:
        - Stream live video feeds from security cameras to the centralized app, with real-time monitoring capabilities.

2. **Motion Detection and Alerting**:
    - Use machine learning to detect unusual movement (e.g., unauthorized access) and send notifications/alerts to the user.
3. **Anomaly Detection**:
    - Identify abnormal behaviours using ML algorithms (e.g., detecting broken windows or forced entry) and trigger appropriate alerts.
4. **Secure Cloud Storage**:
    - Store video footage securely in the cloud, providing the user with easy access and the ability to review footage.
5. **Access Control**:
    - Implement a system to control access points (e.g., doors, gates) and notify users of any unauthorized attempts to breach the premises.

3.2.4. System Feature 4: Personalized User Profiles

- **Description**: The system learns the behaviours and preferences of each resident, enabling personalized automation settings that adapt to individual needs over time.
- **Functional Requirements**:
    1. **Profile Creation**:
        - Users can create individual profiles within the system (e.g., for each family member).
    2. **Personalized Automation**:
        - The system will learn each user's habits (e.g., preferred room temperature, lighting preferences) and adjust the home environment accordingly.
    3. **Behavioural Adaptation**:
        - As the system gathers more data over time, it refines its behaviour and automation rules to better match the user's routine (e.g., adjusting lighting as per time of day or user activity).

4. **Preference-Based Adjustments**:
    - Automatically set lighting, temperature, and other conditions based on user profile preferences (e.g., when a user arrives home, lights and temperature adjust to their preset values).

3.2.5. System Feature 5: Centralized Control System

- **Description**: The system provides a centralized platform (mobile/web app) to manage and monitor all aspects of home automation, including lighting, temperature, and security, from any location.
- **Functional Requirements**:
    1. **Unified App Interface**:
        - The app will provide a unified interface for users to control all automation aspects (lighting, temperature, security) from a single platform.
    2. **Real-Time Data Display**:
        - Display real-time data about the home's conditions (e.g., current temperature, lighting status, camera feeds) on the app.
    3. **Energy Usage Monitoring**:
        - Show energy consumption statistics to help users optimize energy usage.
    4. **Remote Access**:
        - Users can remotely monitor and control home devices when they are away from the premises via secure cloud access.
    5. **User Notifications**:
        - Notify users of significant events (e.g., security alerts, system malfunctions) via push notifications or emails.

3.2.6. System Feature 6: Integration of IoT Devices

- **Description**: Seamless communication between different IoT devices (e.g., sensors, actuators) is essential for real-time data collection and automation.

- **Functional Requirements**:
    1. **IoT Sensor Integration**:
        - Integrate various types of sensors (motion, temperature, light, etc.) to collect data and trigger automation.
    2. **IoT Actuator Control**:
        - Use actuators (e.g., relays, switches) to control devices such as lights, HVAC systems, or other appliances.
    3. **Communication Protocols**:
        - Support wireless communication protocols (e.g., Wi-Fi, Zigbee, Z-Wave) and wired protocols (e.g., Ethernet) for seamless integration.
    4. **Data Syncing and Reliability**:
        - Ensure real-time data syncing between devices to maintain smooth operation and high reliability of the automation system.

3.2.7. System Feature 7: Remote Access

- **Description**: The system allows users to monitor and control their home devices remotely through secure cloud-based access, offering convenience and peace of mind.
- **Functional Requirements**:
    1. **Secure Remote Monitoring**:
        - Allow users to access real-time data and device status remotely through a web or mobile app with secure login and encrypted communication.
    2. **Cloud-Based Data Access**:
        - Store system data (e.g., logs, usage statistics) in the cloud for easy retrieval and analysis.
    3. **Push Notifications**:
        - Send notifications to users for critical events (e.g., security breach, appliance malfunction) even when they are away from home.

4. **Authentication**:
    - Implement user authentication and authorization mechanisms to ensure only authorized individuals can control or view the system remotely.

3.2.8. System Feature 8: Energy Efficiency Monitoring

- **Description**: The system helps users optimize their energy usage by monitoring appliance consumption and automating the operation of energy-consuming devices.
- **Functional Requirements**:
    1. **Energy Usage Statistics**:
        - Provide detailed statistics on energy consumption of various devices and appliances.
    2. **Automation for Energy Efficiency**:
        - Automatically turn off unused appliances (e.g., lights, fans) to reduce energy waste.
    3. **Energy-Saving Suggestions**:
        - Provide recommendations to users on how to optimize their energy consumption based on usage patterns (e.g., reduce heating when the house is empty).
    4. **Scheduled Energy Usage**:
        - Set schedules for devices to operate at optimal energy-efficient times (e.g., running the washing machine during off-peak hours)

3.3. EXTERNAL INTERFACE REQUIREMENTS

3.3.1 Hardware Interfaces

The system will need various hardware interfaces to interact with IoT devices and the central controller (Raspberry Pi/ESP32/Arduino). These include:

- **Microcontroller Interfaces**: The microcontrollers (e.g., ESP32, Arduino) will communicate with sensors (e.g., motion, temperature, light) and actuators (e.g., relays) via GPIO pins or I2C/SPI communication protocols.
    - **Sensors**: Motion (PIR), temperature (DHT11/22), and light (LDR) sensors will send data to the microcontrollers for further processing.
    - **Actuators**: Relays will control connected devices (lights, appliances) through electrical signals.
- **Cameras**: IP cameras will be connected to the Raspberry Pi or cloud services, with video data sent to the cloud for real-time surveillance and machine learning analysis.
- **Power Supply Interfaces**: Proper power management interfaces for supplying consistent power to microcontrollers, sensors, and cameras.

3.3.2 Software Interfaces

The system will interact with several software components to achieve the desired functionality:

- **Machine Learning Libraries**: The system will use TensorFlow, OpenCV, or similar libraries for processing sensor data, surveillance video, and implementing intelligent automation. These libraries will run on the central processing unit (e.g., Raspberry Pi or cloud server).
- **Third-Party Integrations**: The system will integrate with third-party applications such as voice assistants (Google Assistant, Alexa) and home automation platforms (e.g., SmartThings) for voice and automated control.

3.3.3 Communication Interfaces

The communication interfaces are crucial for data exchange between IoT devices, sensors, actuators, and the central controller, as well as between the system and user interfaces.

- **Wi-Fi**: IoT devices and microcontrollers will use Wi-Fi for communication with the central server and cloud services. This will enable remote access to devices and real-time data synchronization.

- **Bluetooth Low Energy (BLE)**: For short-range communication with certain devices (e.g., door locks, light bulbs), BLE will be used for energy-efficient communication.
- **Zigbee/Z-Wave**: For communication between home automation devices that are compatible with these standards, ensuring low-power and reliable data exchange.
- **Ethernet**: For wired communication where Wi-Fi is not available or stable, especially for cameras and critical devices that require consistent data transfer.
- **MQTT Protocol**: MQTT will be used as the messaging protocol to communicate between devices and the cloud service. It is lightweight and suitable for IoT applications requiring fast and reliable messaging.

3.4. NONFUNCTIONAL REQUIREMENTS

3.4.1 Performance Requirements

Performance requirements define the expected system efficiency and responsiveness. For this home automation system, the following are key performance expectations:

- **Real-Time Processing**: The system should process sensor data (e.g., temperature, motion) and make real-time decisions for automation. Response time for device control commands (e.g., turning on/off lights) should be under 1 second.
- **Real-Time Video Streaming**: Surveillance cameras should stream video in real-time with minimal latency, ensuring that video footage can be viewed live from the mobile app or web interface with minimal buffering.
- **Scalability**: The system should be able to handle the addition of new devices (e.g., more sensors, cameras, appliances) without degrading performance. It should support at least 50 connected devices simultaneously.
- **Throughput**: The system should be able to process and transmit sensor data at a rate of 10-20 data points per second, ensuring that automation rules and monitoring remain consistent even under high load.

3.4.2 Safety Requirements

Safety requirements address how the system ensures the well-being of users and prevents potential harm in case of failure or misuse. For this system:

- **Electrical Safety**: The system should ensure that all electrical devices controlled through the automation system, such as lights and appliances, adhere to safety standards. Devices like relays should be rated for the expected voltage and current to prevent electrical hazards.
- **Hardware Failure Protection**: If any hardware (e.g., sensors, microcontrollers, cameras) malfunctions, the system should fail gracefully, alerting the user to the issue without causing unsafe operation or device damage.
- **Emergency Override**: In case of a malfunction in automation or control systems, users should have the ability to manually override all automated systems (e.g., lights, temperature, security) via the mobile or web app to ensure continued safety.

3.4.3 Security Requirements

Security is a critical component of any IoT-based system, especially one controlling sensitive areas like home automation. The following security requirements should be met:

- **Data Encryption**: All communication between devices, mobile app, web interface, and cloud services should be encrypted using secure protocols such as SSL/TLS to prevent unauthorized data interception or tampering.
- **Authentication & Authorization**: Users must authenticate themselves securely to access the system (e.g., via username/password, two-factor authentication). Different user roles (e.g., admin, guest) should be supported to restrict access to sensitive system settings and data.
- **Secure Cloud Storage**: Sensitive data, such as user preferences, usage patterns, and video footage, should be securely stored on the cloud using encryption. This ensures that unauthorized access to personal data is prevented.
- **Intrusion Detection**: The system should implement intrusion detection mechanisms, using machine learning algorithms to identify abnormal behaviors

or unauthorized access attempts. In the event of a security breach, the system should notify the user immediately.

- **Firmware Updates**: The system should support secure firmware updates for IoT devices to patch vulnerabilities and improve system security over time.

3.4.4 Software Quality Attributes

These are the characteristics that the software should exhibit in terms of reliability, maintainability, and usability:

- **Reliability**: The system should maintain high availability and be able to recover from failures (e.g., power outages, network issues) without significant downtime. It should also be able to handle system failures without corrupting user data.
- **Maintainability**: The software should be modular and well-documented to support future maintenance and feature upgrades. Clear documentation should include instructions for debugging, troubleshooting, and performing system updates.
- **Usability**: The user interfaces (mobile app and web interface) should be intuitive, making it easy for users with varying technical skills to operate the system. The system should also offer easy configuration for automation rules, security settings, and notifications.
- **Portability**: The system should be able to run on various hardware platforms (e.g., Raspberry Pi, Arduino, cloud) without requiring significant modifications. It should also support multiple operating systems (e.g., Linux for Raspberry Pi, Android/iOS for mobile apps).
- **Scalability**: The system architecture should be designed for scalability to support additional devices and features with minimal adjustments. The cloud infrastructure should be capable of scaling up or down based on usage patterns.
- **Interoperability**: The system should be compatible with various third-party devices and services. This includes compatibility with other IoT platforms (e.g., Google Home, Alexa) and third-party APIs (e.g., weather services for automatic temperature adjustment).

- **Efficiency**: The system should optimize resource usage, minimizing power consumption, CPU load, and network bandwidth, especially for IoT devices that are always active, like sensors and cameras.

3.5. SYSTEM REQUIREMENTS

3.5.1 Database Requirements:

- **Database Type and Integration**:
    - Home Assistant primarily uses **SQLite** for its internal database, which is sufficient for small-scale systems and basic operations like logging states, events, and configurations. However, for systems with extensive logging requirements or larger-scale installations with numerous IoT devices and cameras, you may opt to use **MySQL** or **PostgreSQL** for more robust data management and faster querying capabilities.
    - **External Database Integration**: For advanced features like long-term data analytics, it is recommended to integrate Home Assistant with an external database, especially for storing large volumes of sensor data, video surveillance footage metadata, or energy consumption data that Home Assistant itself cannot manage efficiently in SQLite.
- **Data Retention and Backup**:
    - **Log and History Data**: You may need to establish retention policies for logs and historical data, particularly for data that could grow large over time, such as sensor readings, video surveillance recordings, or automation logs. This can help maintain system performance and avoid storage overrun.
    - **Backup and Recovery**: Data should be regularly backed up to prevent loss due to system failures or hardware malfunctions. Scheduled backups for both Home Assistant configurations and external databases (if used) should be performed periodically.

- **Scalability**:
    - As the system scales with more devices, the database should be able to handle an increasing number of entries, particularly with real-time IoT data processing. A scalable database solution should support automatic scaling or allow for manual scaling as the system grows.

3.5.2 Software Requirements:

- **Home Assistant Platform**:
    - **Home Assistant Core**: This is the core software that handles device integration, automation, and control. It runs on various platforms like Raspberry Pi, Docker, Linux, Windows, or cloud instances.
    - **Operating System**: For local installations, the recommended OS is **Raspbian** for Raspberry Pi devices. If using more robust hardware or virtual machines, **Ubuntu** or any Linux-based OS can also be used. For cloud-based installations, **Home Assistant OS** (a minimal OS designed for Home Assistant) can be used.
- **Programming Languages**:
    - Home Assistant allows for extensive customization and integrations using **YAML** for automation configuration and **Python** for writing custom components or integrating machine learning models.
    - **MQTT Protocol**: For real-time communication between IoT devices, using **MQTT (Message Queuing Telemetry Transport)** is common. The MQTT broker can be Mosquitto or any other lightweight broker. MQTT supports real-time communication, ideal for IoT applications like lighting, temperature control, and security monitoring.
    - **Machine Learning Libraries**: For analysing user behaviour patterns and detecting security anomalies:
        - **TensorFlow** and **Keras** for training predictive models.
        - **OpenCV** for computer vision and image processing, especially for camera-based surveillance systems.

- **Integration with Third-Party Services**:
    - Home Assistant supports integrations with a wide range of third-party cloud platforms like **AWS IoT**, **Google Cloud**, and **Azure IoT Hub**. For cloud-based storage or remote device management, consider integrating with these platforms to extend functionality.
    - For remote monitoring, **Home Assistant Cloud** or **Google Assistant**/ **Amazon Alexa** integrations can be used for voice control.
- **Add-Ons and Custom Components**:
    - Home Assistant has a robust system of add-ons to extend functionality. Add-ons for **Node-RED** (for advanced automation flows), **MariaDB** (for external database support), **Let's Encrypt** (for SSL certificates), and **ESPHome** (for custom firmware for ESP8266/ESP32 devices) can all be used to enhance the system's capabilities.

3.5.3 Hardware Requirements:

- **Home Assistant Server**:
    - **Raspberry Pi 4 (4GB RAM)**: The most cost-effective and recommended option for running Home Assistant on a single device. It provides sufficient processing power for most home automation tasks and can be easily upgraded to higher memory options (e.g., 8GB RAM models) for larger setups.
    - **Intel NUC / Server Hardware**: For more demanding installations (e.g., with high-resolution cameras or advanced machine learning), you may require more powerful hardware. An Intel NUC or server-class hardware running Home Assistant OS could offer better performance, especially when handling multiple integrations or intensive computational tasks like real-time object detection.
    - **Storage**: Sufficient storage is essential for logs, historical data, and video footage. For Raspberry Pi, you would typically use an **SSD** for better reliability and speed. For other setups, an **HDD** or **SSD** can be used depending on the scale.

- **IoT Devices and Sensors**:
    - **Sensors**: A range of IoT sensors for motion, temperature, humidity, and light can be used. For instance:
        - **PIR Motion Sensors**: Approx. $5–$10 per unit.
        - **DHT11/22 Temperature & Humidity Sensors**: Approx. $5–$10 per unit.
        - **Light Sensors**: Approx. $2–$5 per unit.
    - **Cameras for Surveillance**:
        - IP Cameras: Approx. $50–$100 for basic models, or higher-end cameras for HD/4K support (e.g., $100–$200).
        - Consider integration with **ONVIF** or other camera protocols supported by Home Assistant.
- **Relays and Actuators**:
    - **Relays** for controlling devices like lights, fans, or security systems. These range from **$5–$10** per relay, and you may need multiple relays depending on the number of devices you plan to automate.
- **Networking**:
    - **Wi-Fi/Ethernet**: Reliable network connectivity is required for communication between the devices, sensors, and Home Assistant. A good Wi-Fi router (or Ethernet switch) is essential to ensure seamless communication between devices.
    - **Zigbee/Z-Wave Integration**: If using Zigbee or Z-Wave devices, you'll need a Zigbee/Z-Wave USB stick (e.g., **ConBee II** or **Aeotec Z-Wave Stick**) to connect these devices to Home Assistant.
- **Power Supply**:
    - Ensure adequate **power supply units (PSUs)** for the Raspberry Pi, sensors, and cameras. Power-over-Ethernet (PoE) can also be an option for cameras or devices located far from power outlets.
- **Backup Power**:
    - For maintaining operations in case of power outages, a **UPS (Uninterruptible Power Supply)** is recommended, especially for the Home Assistant server and critical devices (e.g., cameras and alarms).

3.5.4 Communication Interfaces:

- **Protocol Support**:
  - **Wi-Fi**: Most IoT devices and sensors communicate over Wi-Fi, ensuring compatibility with standard home networks.
  - **Zigbee and Z-Wave**: For device-to-device communication in smart home setups, Zigbee and Z-Wave protocols may be used. These require specific hubs or adapters (e.g., **ConBee II** for Zigbee).
  - **MQTT**: MQTT is widely used for communication between sensors, actuators, and the Home Assistant server. It is lightweight, efficient, and well-supported within the IoT ecosystem.
  - **Bluetooth**: For short-range communication, Bluetooth can be used with compatible IoT devices.
  - **Cloud Communication**: For remote control, Home Assistant supports cloud integration with services like **Google Assistant**, **Amazon Alexa**, and **Apple HomeKit**.
  - **Local Communication**: For privacy and security, it's recommended to use local communication protocols where possible to ensure that data does not need to be sent to the cloud.

3.6 ANALYSIS MODELS: SDLC MODEL TO BE APPLIED

We are using the Waterfall Model for this project, following a structured sequence of phases that provide clear documentation and milestone completion at each stage. This approach allows us to address each development aspect thoroughly, which aligns well with our project's systematic approach to integrating machine learning with IoT for home automation.

1. **Requirements:**

   Define specific system requirements, including both functional and non-functional needs to guide system capabilities and limitations.

2. **Planning:**

   Select the software tools that best support IoT integration, data management, and ML functionalities.

   Explore appropriate machine learning models for automation and predictive analysis.

   Choose reliable data sources, both internal (sensors) and external (weather, environmental factors), for system learning and decision-making.

3. **Design:**

   Create the model architecture design, laying out the relationships between IoT devices, data processing units, and machine learning models.

   Set up the hardware, including sensors and microcontrollers, ensuring compatibility and integration.

   Integrate software and hardware components, establishing communication protocols between IoT devices and the core control system.

4. **Implementation:**

   Develop the core functionality of the system, implementing primary automation features like lighting, temperature, and security control.

   Conduct data trial runs to verify data flow, accuracy, and reliability, and to calibrate machine learning models.

5. **Testing and Maintenance:**

   Perform preliminary testing and debugging to identify and resolve any issues in system operations.

   Integrate advanced output features such as customized alerts and enhanced control options.

   Fine-tune and optimize the system for efficiency, ensuring smooth, reliable, and secure operation.

By following these steps, we aim for a robust and efficient deployment of an intelligent home automation system. This Waterfall approach supports thorough documentation and structured progress, from requirements to final system optimization.

3.7 SYSTEM IMPLEMENTATION PLAN

The implementation plan for the IoT-based home automation project outlines a phased approach to ensure a smooth and structured deployment. This plan includes key steps for setting up the system, configuring components, testing, and monitoring. Each phase is designed to reduce risks, manage resources effectively, and ensure that each part of the system functions as expected.

1. Initial Setup and Preparation

- **Objective**: Set up foundational hardware, software, and cloud environments.
- **Tasks**:
    - Set up Raspberry Pi (or equivalent) with the necessary OS (e.g., Raspbian) and configure it as the central controller.
    - Install and configure Home Assistant on the Raspberry Pi, which will serve as the main integration and control platform.
    - Establish the network setup and verify connections for IoT devices, ensuring they are accessible on the local network.
- **Expected Outcome**: A configured system environment ready to host and control IoT devices through Home Assistant.

2. Integration of Core IoT Devices and Sensors

- **Objective**: Integrate essential IoT devices (e.g., sensors, actuators) and establish their functionality within the Home Assistant platform.
- **Tasks**:
    - Connect motion sensors, temperature sensors, and light sensors to the microcontrollers (Arduino or ESP32).
    - Configure actuators (e.g., relays) for lighting and appliance control.
    - Integrate cameras for security surveillance and ensure video feeds are accessible through Home Assistant.
    - Establish communication between sensors, actuators, and Home Assistant using MQTT or HTTP protocols.
- **Expected Outcome**: All core IoT devices are connected and communicate successfully with Home Assistant, providing basic automation functions.

3. Implementation of Automation and Control Features

- **Objective**: Develop and configure automation routines to enable intelligent home control.
- **Tasks**:
    - Set up rules in Home Assistant for automation based on sensor data (e.g., turn on lights based on motion and time of day).
    - Program temperature control automation based on occupancy, user preferences, and external weather conditions.
    - Enable manual overrides for each automated function through the Home Assistant interface.
    - Configure security alerts and notifications for real-time monitoring of surveillance cameras and sensors.
- **Expected Outcome**: A functioning automated environment that adjusts lighting, temperature, and security based on user behaviour and sensor data.

4. Machine Learning Model Integration

- **Objective**: Integrate machine learning models for advanced personalization and threat detection.
- **Tasks**:
    - Train a machine learning model to analyse user behaviour patterns and make predictive adjustments to the environment (e.g., adjust lighting or temperature based on time of day and user presence).
    - Implement a computer vision model for real-time security monitoring and anomaly detection in video feeds.
    - Integrate these models with Home Assistant to enable dynamic decision-making.
- **Expected Outcome**: Personalized automation and security monitoring that evolves based on user behaviour and environmental conditions.

5. User Interface Development and Testing

- **Objective**: Ensure that users can control and monitor the system efficiently through a user-friendly interface.

- **Tasks**:
    - Customize the Home Assistant frontend to display essential controls and data for lighting, temperature, and security.
    - Test the interface on both mobile and web platforms for ease of use, responsiveness, and clarity.
    - Implement remote access with secure authentication so that users can control their home environment from anywhere.
- **Expected Outcome**: A reliable, accessible interface that allows users to monitor and control home settings both locally and remotely.

6. Testing and Quality Assurance

- **Objective**: Conduct comprehensive testing to verify system reliability, performance, and security.
- **Tasks**:
    - Perform unit tests, integration tests, and user acceptance tests for each functional component.
    - Test the machine learning models to ensure they provide accurate predictions and detections.
    - Conduct security testing on data transmissions and remote access points.
    - Monitor system performance to identify potential issues with latency, response time, and network reliability.
- **Expected Outcome**: A fully tested system with validated performance, security, and reliability standards.

7. Deployment and Final Integration

- **Objective**: Deploy the system for real-world use and ensure smooth operation.
- **Tasks**:
    - Install the system in the target environment and connect all IoT devices.
    - Conduct final tests on-site to confirm all components work as expected.
    - Provide training or documentation for users on how to operate the system and troubleshoot basic issues.
- **Expected Outcome**: A live system in the intended environment with all features functional and accessible.

8. Ongoing Maintenance and Monitoring

- **Objective**: Ensure the system's long-term performance, security, and scalability.
- **Tasks**:
    - Regularly monitor system logs and performance metrics to detect issues early.
    - Update software, firmware, and machine learning models as needed.
    - Provide support for adding new devices or upgrading features based on user feedback.
- **Expected Outcome**: A well-maintained system that remains efficient, secure, and adaptive to new requirements over time.

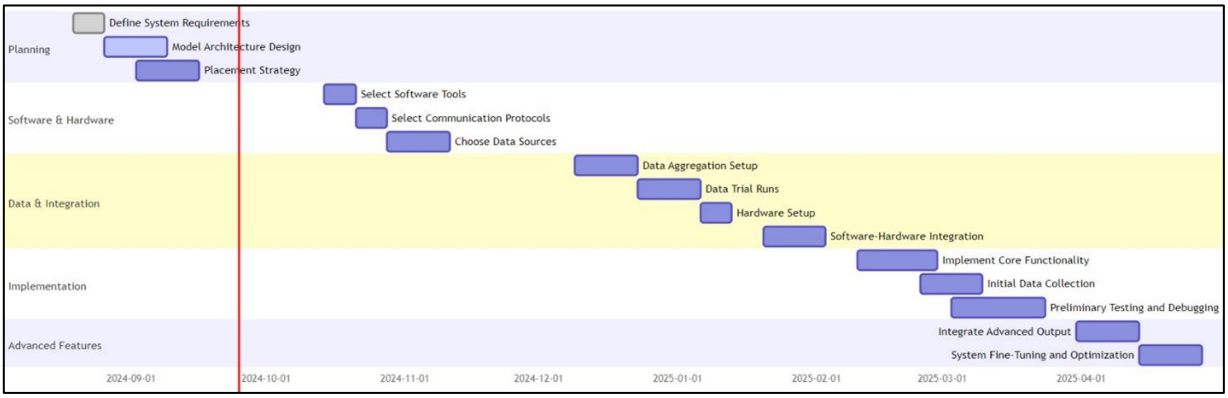

Figure 1: Project Timeline

4 SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE AND MODULE DESCRIPTION

The system architecture for the IoT-based home automation system consists of seven layers:

1. **Web App (User Interface Layer)**

- Tasks:
    - Develop a responsive interface for user control and interaction.
    - Implement account management (user login, settings, etc.).
    - Real-time status updates from sensors and actuators.
    - Provide insights based on machine learning predictions.
- Technology: MERN stack

2. **Home Assistant (Logic and Control)**

- Tasks:
    - Manage communication between sensors, actuators, and other modules.
    - Control routines and home automation processes.
    - Interface with the Machine Learning system for decision-making.
- Technology: Home Assistant

3. **Data Storage (Backend)**

- Tasks:
    - Store sensor data, user preferences, and logs.
    - Create databases to support machine learning training.
    - Ensure secure and reliable storage of information.
- Technology: Cloud Storage options like AWS S3 for large data handling.
    - Database: NoSQL (MongoDB) for sensor data, SQL (PostgreSQL) for user management.

4. **Machine Learning System**

- Tasks:
    - Develop and train models based on sensor data

- Perform predictive tasks
- Implement feedback loops to improve model accuracy.

- Technology: Frameworks: TensorFlow, PyTorch.
  - Integration: Flask API for the model service.

5. Sensors (Input Devices)

- Tasks:
  - Collect environmental data (temperature, humidity, light, etc.).
  - Transmit data to the Home Assistant for further processing.
- Technology: Use hardware like Temperature, light, humidity, presence, cameras, and biometric sensors.

6. Actuators (Output Devices)

- Tasks:
  - Receive commands from the Home Assistant.
  - Control physical devices (lights, thermostat, door locks, etc.)
- Technology: Use hardware like Relays, motors, servo, solenoid.

7. Communication Network

- Tasks:
  - Ensure reliable data flow between all components.
  - Manage latency, packet loss, and security in communication.
- Technology: Protocols like Wi-Fi

4.2 DATA FLOW DIAGRAMS

4.2.1 Data Flow Diagram - 0

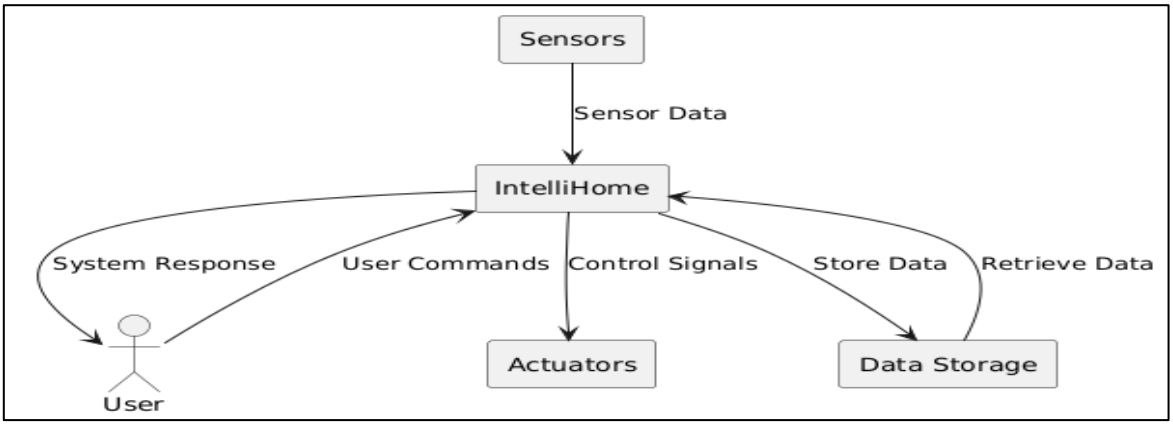

Figure 2: Data Flow Diagram – 0

4.2.2 Data Flow Diagram – 1

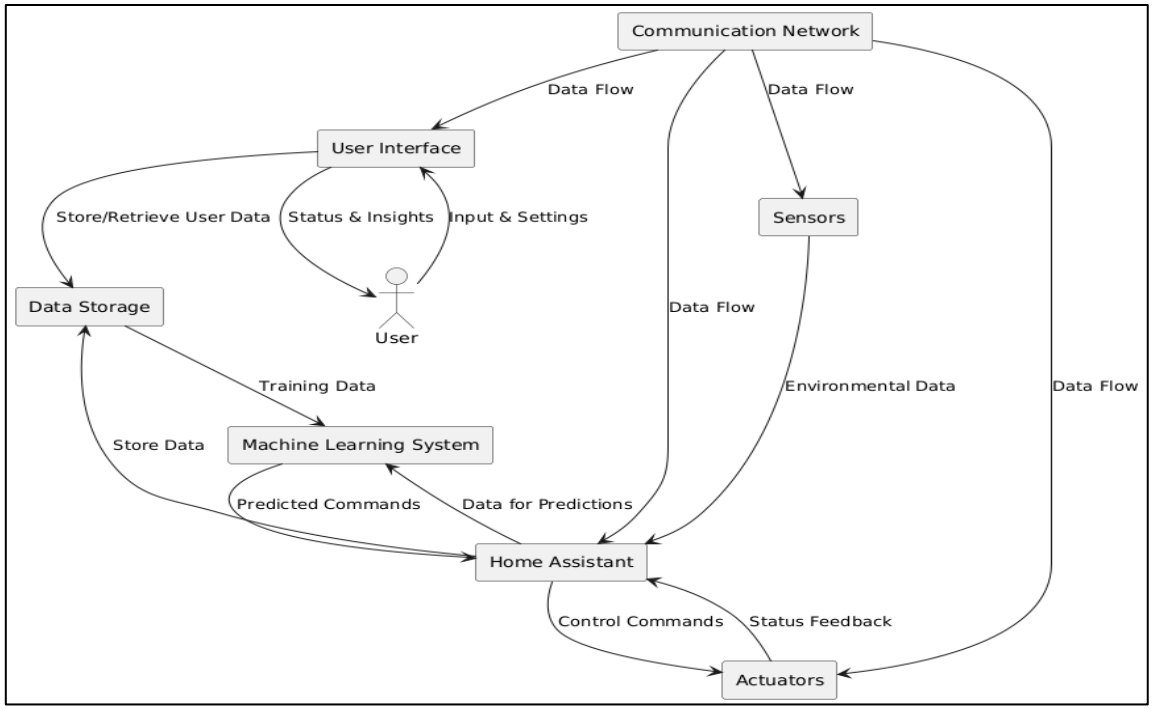

Figure 3: Data Flow Diagram - 1

4.2.3 Data Flow Diagram – 2

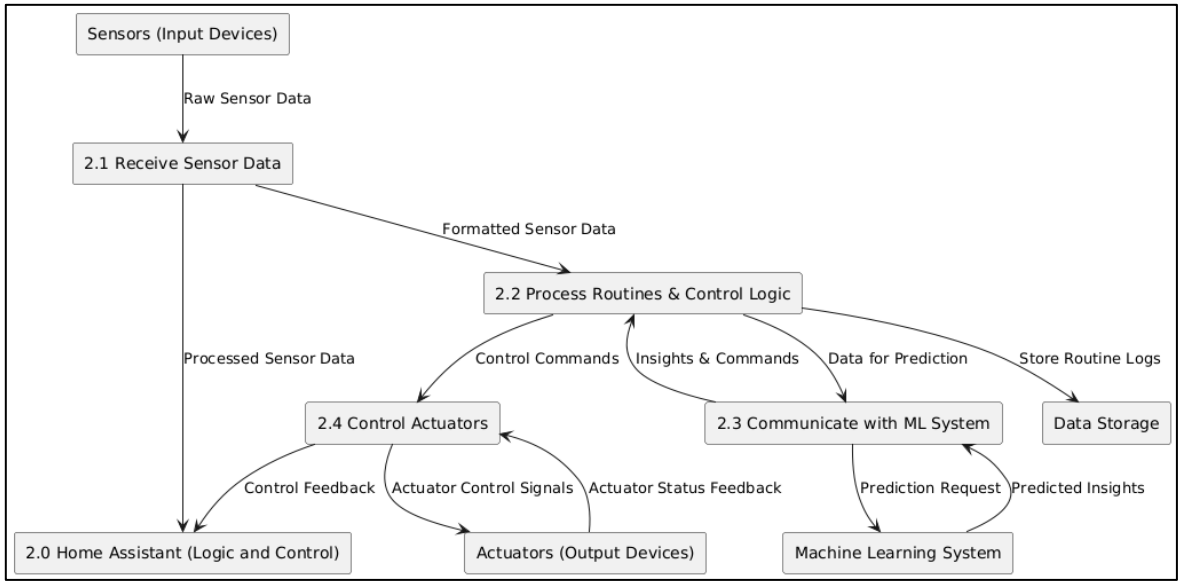

Figure 4: Data Flow Diagram - 2

4.3 ENTITY RELATIONSHIP DIAGRAM

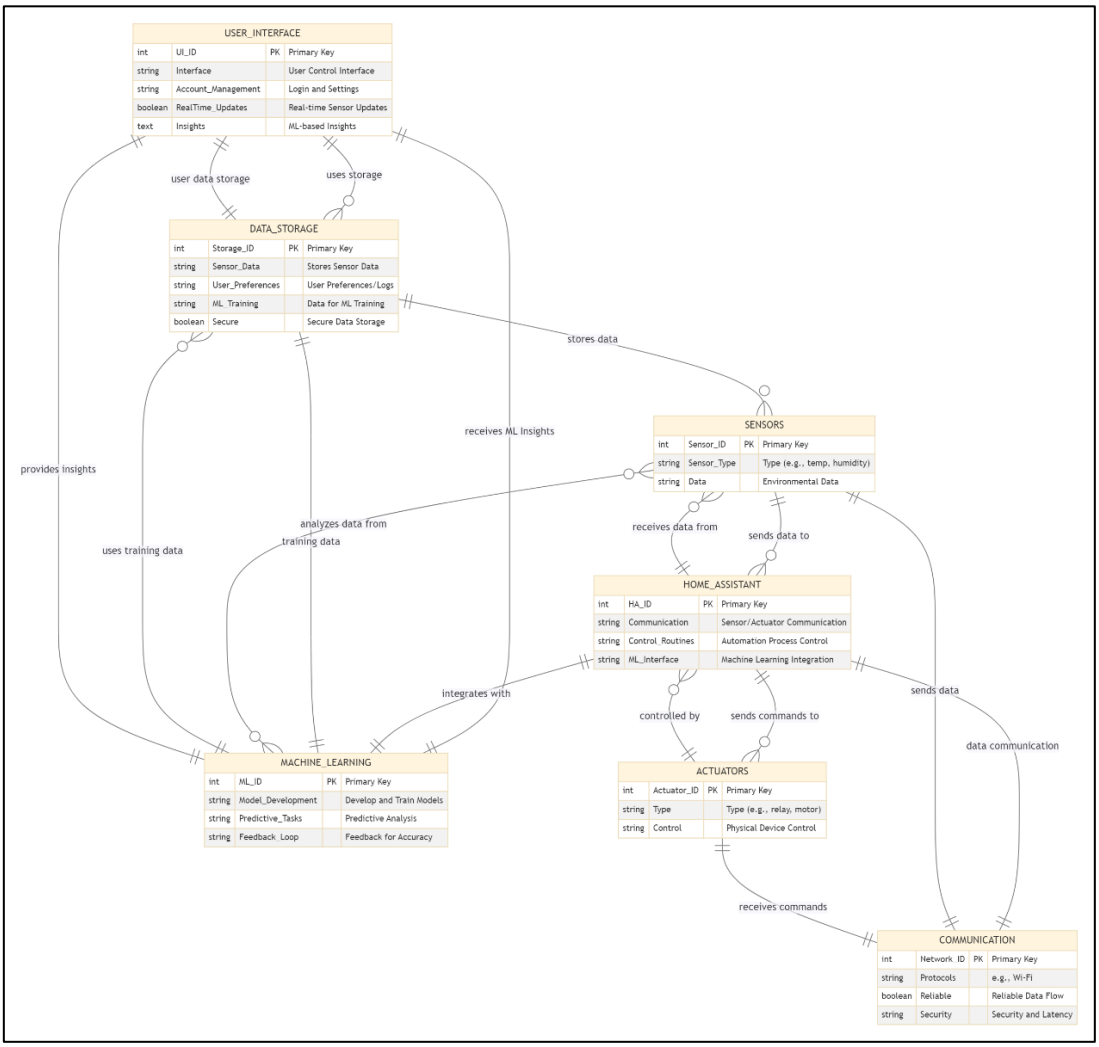

Figure 5: Entity Relationship Diagram

4.4 UML DIAGRAMS

4.4.1 Usecase Diagram

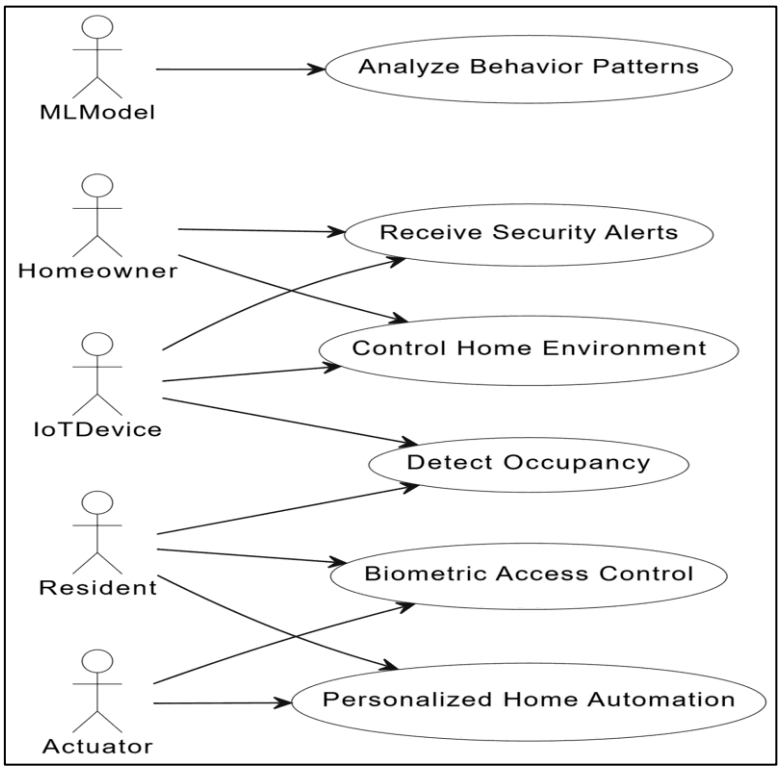

Figure 6: Usecase Diagram

4.4.2 Class Diagram

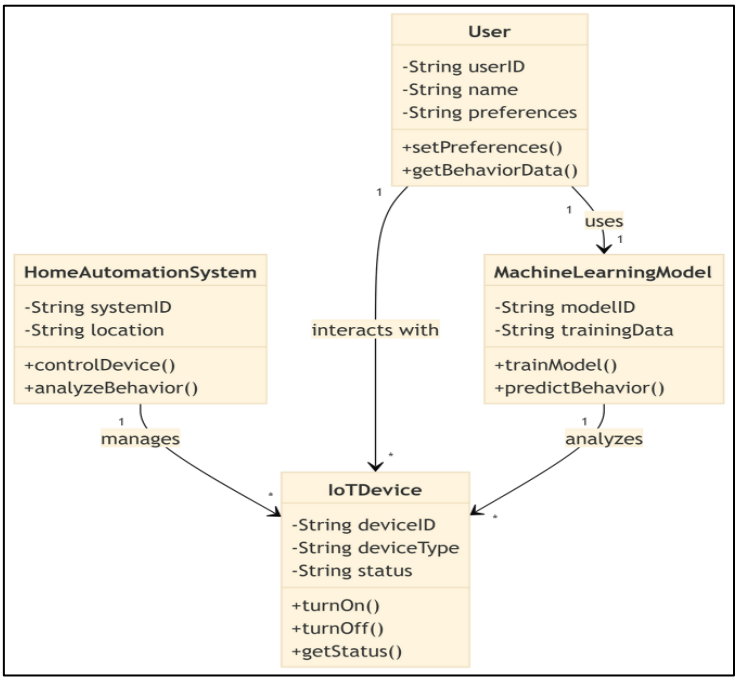

Figure 7: Class Diagram

4.4.3 Activity Diagram

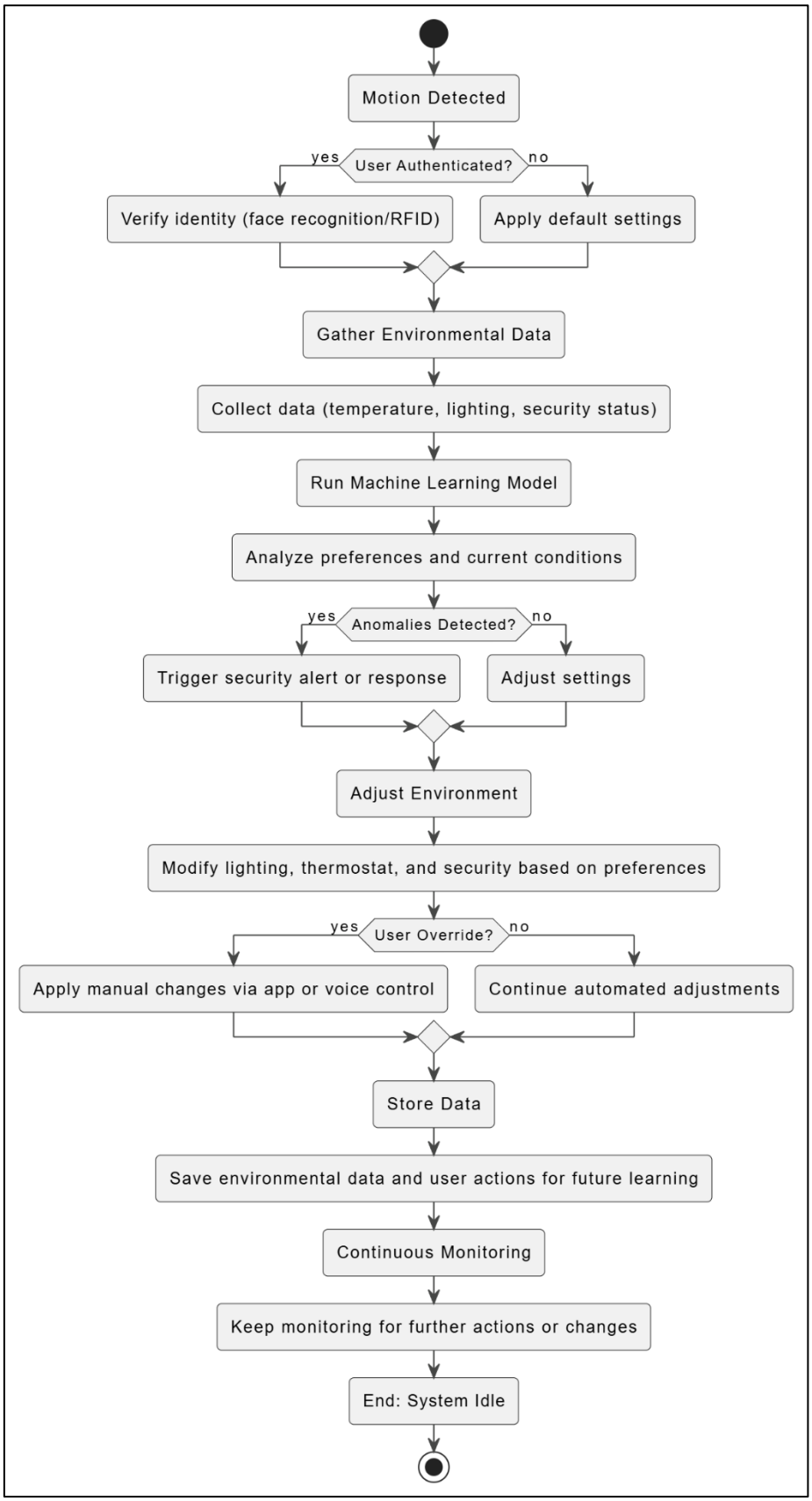

Figure 8: Activity Diagram

4.4.4 Sequence Diagram

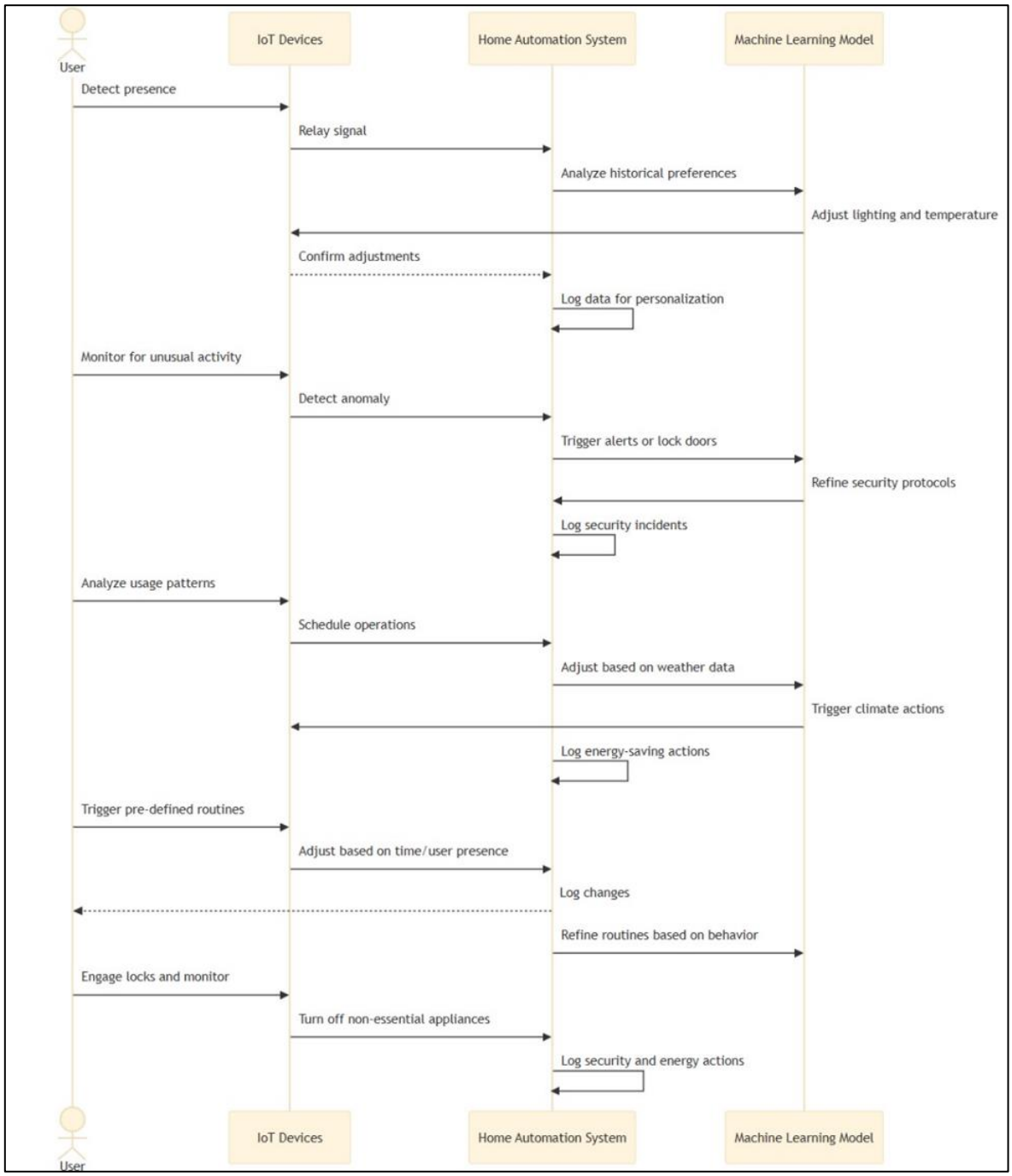

Figure 9: Sequence Diagram

4.4.5 Component Diagram

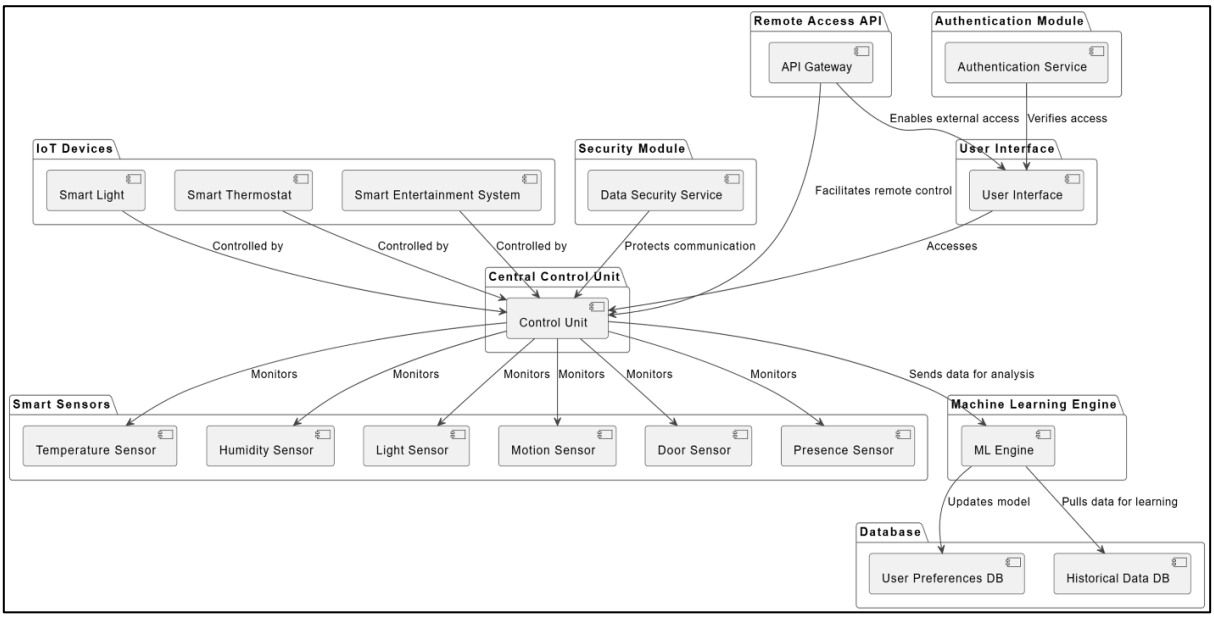

Figure 10: Component Diagram

5 OTHER SPECIFICATIONS

5.1 ADVANTAGES

1. **Enhanced Convenience**: Automates routine tasks like lighting, temperature control, and security, improving user comfort.
2. **Energy Efficiency**: Optimizes energy usage by adjusting lights and temperature based on occupancy, resulting in cost savings.
3. **Improved Security**: Real-time monitoring and alerts provide immediate notifications of unusual activity, enhancing home safety.
4. **Personalization**: Learns user preferences over time and adjusts settings, creating a tailored environment.
5. **Remote Access**: Allows users to control and monitor the home remotely, ensuring accessibility and convenience.
6. **Scalability**: Supports additional devices and functionalities, making it adaptable for future requirements.
7. **Data Insights**: Provides detailed energy consumption and usage patterns, helping users make informed decisions.

5.2 LIMITATIONS

1. **Initial Setup Cost**: Can be expensive due to hardware, installation, and potential cloud service fees.
2. **Complexity of Maintenance**: May require periodic software updates and maintenance, especially for ML models and IoT components.
3. **Privacy and Security Risks**: Data breaches or unsecured connections could compromise user data and privacy.
4. **Dependency on Internet Connectivity**: Some features may be limited or unavailable without a stable internet connection.

5. **Limited Compatibility**: Some IoT devices may not integrate smoothly with Home Assistant or other systems used.
6. **Power Dependency**: The system relies on continuous power; a power failure can disrupt automation.

5.3 APPLICATIONS

1. **Smart Homes**: Provides automation and control in residential settings, enhancing comfort, security, and efficiency.
2. **Commercial Buildings**: Useful for automating lighting, HVAC, and security in offices, hotels, and retail spaces.
3. **Healthcare Facilities**: Can monitor and automate systems for patient comfort, such as temperature and lighting.
4. **Educational Institutions**: Can be used to control and monitor energy usage in classrooms, laboratories, and libraries.
5. **Elderly Care**: Assists in monitoring activity patterns and providing security for elderly individuals living alone.
6. **Energy Management Solutions**: Useful for buildings aiming to reduce their carbon footprint by optimizing energy usage.
7. **Agriculture**: Can be adapted to monitor environmental conditions and control systems like irrigation and lighting in smart farming setups.

6 CONCLUSIONS & FUTURE WORK

CONCLUSION:

This project demonstrates the successful integration of Machine Learning (ML) with home automation to enhance efficiency, personalization, and intelligence in smart homes. By using ML algorithms, the system learns user behavior patterns and adapts over time, optimizing tasks like lighting, temperature, and security settings for better energy efficiency and user convenience. The combination of IoT and AI allows for more responsive, autonomous, and secure control of home devices, reducing the need for manual input while enabling predictive maintenance and anomaly detection.

FUTURE WORK:

- Enhanced Personalization: Broaden the data scope to include more behavioral and environmental factors for tailored user experiences.
- Real-Time Data Processing: Implement edge computing to improve adaptability and response time.
- Predictive Maintenance: Use ML to monitor device health and anticipate failures, ensuring reliable system uptime.
- Expanded IoT Integration: Integrate additional smart devices to expand automation capabilities.
- Security and Privacy: Strengthen encryption and authentication for enhanced data protection.
- Energy Optimization: Develop ML algorithms to forecast and reduce energy usage costs.
- Voice and Gesture Control: Add support for voice and gesture interfaces for hands-free interaction.
- Cross-Platform Compatibility: Expand support for mobile and multi-device control for improved accessibility.
- Addressing these areas could make the system more intelligent, efficient, and user-friendly, advancing towards a fully automated smart home environment.

APPENDIX A:

Problem Statement Feasibility Assessment

Problem Overview

This project focuses on integrating Machine Learning (ML) into a home automation system to optimize lighting, climate control, security, and energy usage. The goal is to design a system that learns from user preferences and adjusts automatically while maintaining energy efficiency and security.

Satisfiability Analysis

Satisfiability analysis examines whether the system can find configurations (e.g., device settings, sensor data) that meet predefined objectives like energy efficiency, comfort, and security. This is similar to the Boolean satisfiability problem (SAT), where we determine if a set of conditions can be satisfied by the system.

Computational Complexity

NP-hard Problems:

Energy Optimization: Minimizing energy usage while maximizing comfort is similar to the knapsack problem, which is NP-hard.

Scheduling and Task Management: Managing multiple devices in real-time (e.g., when to activate or deactivate) is akin to the job scheduling problem, also NP-hard.

NP-complete Problems:

Constraint Satisfaction: Balancing conflicting constraints (e.g., comfort vs. energy savings) is a constraint satisfaction problem (CSP), which is NP-complete.

Device Routing: Optimizing how commands are routed to devices based on constraints is an NP-complete problem.

P-type Problems:

Single Device Control: Deciding whether to turn a device on or off based on basic sensor data is solvable in polynomial time.

Basic Sensor Processing: Processing and reacting to sensor data (e.g., temperature) is a straightforward task that can be done in polynomial time.

Mathematical Models

Graph Theory: Devices and sensors can be modeled as a graph, with relationships forming edges. Task scheduling and optimization become graph traversal or optimization problems, which can be NP-complete.

Linear Programming: Energy optimization can be modeled using linear programming, solvable in polynomial time for certain problems but potentially NP-hard with mixed-integer constraints.

Boolean Algebra: Logical relationships between devices and constraints are often expressed using Boolean algebra, which leads to NP-complete satisfiability problems.

Conclusion

The system involves both NP-hard and NP-complete problems, especially for energy optimization, task scheduling, and multi-constraint decision-making. While simple tasks (like controlling individual devices or processing sensor data) are P-type and solvable in polynomial time, as the system grows in complexity, more sophisticated approaches (e.g., heuristics or approximation algorithms) will be needed to handle the increased computational demands.

APPENDIX B:

Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy, "ARAS Human Activity Datasets in Multiple Homes with Multiple Residents," In Proceedings of the ICST PervasiveHealth Conference, Venice, Italy, May 2013. IEEE.
Summary: This paper presents the ARAS dataset, a valuable collection for human activity recognition in smart environments. Collected over two months from two homes with multiple residents, the dataset includes sensor data from 20 binary sensors across 27 activities. This work aims to enhance machine learning models for activity recognition, with applications in continuous health monitoring in realistic home environments.

Hyung-Jin Mun and Min-Hye Lee, "Design for Visitor Authentication Based on Face Recognition Technology Using CCTV," IEEE Access, vol. 10, pp. 124604-124618, Dec. 2022, doi:10.1109/ACCESS.2022.3223374.
Summary: This paper introduces a visitor authentication system that uses face recognition technology with CCTV, implemented on a Jetson Nano platform. The system uses Tiny-YOLOv3 for real-time face detection and multi-factor authentication to enhance security against unauthorized access. Experimental results indicate an effective accuracy rate of 86.3% with a detection speed of 6.5 FPS, demonstrating the system's viability for smart home security applications.

Tiankai Liang, Bi Zeng, Jianqi Liu, Linfeng Ye, and Caifeng Zou, "An Unsupervised User Behavior Prediction Algorithm Based on Machine Learning and Neural Network for Smart Home," IEEE Access, vol. 6, pp. 49237-49247, Sept. 2018, doi:10.1109/ACCESS.2018.2868984.
Summary: This paper proposes an unsupervised user behavior prediction (UUBP) algorithm that leverages machine learning and neural networks to enhance smart home systems. The UUBP algorithm uses a neural network for self-initialization and incorporates a "forgetting factor" to prioritize recent user behaviors, effectively predicting user actions and improving system responsiveness to evolving patterns. The method was validated through experiments demonstrating its superiority over traditional algorithms in adapting to user behavior changes.

APPENDIX C:

Plagiarism Report

Word count: 8611
Page count: 44
Character count: 60508
Unique: 86%
Plagiarism: 14%

REFERENCES:

- Hande Alemdar, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy, "ARAS Human Activity Datasets in Multiple Homes with Multiple Residents," In Proceedings of the ICST PervasiveHealth Conference, Venice, Italy, May 2013. IEEE.
- Hyung-Jin Mun and Min-Hye Lee, "Design for Visitor Authentication Based on Face Recognition Technology Using CCTV," IEEE Access, vol. 10, pp. 124604-124618, Dec. 2022, doi:10.1109/ACCESS.2022.3223374.
- Tiankai Liang, Bi Zeng, Jianqi Liu, Linfeng Ye, and Caifeng Zou, "An Unsupervised User Behavior Prediction Algorithm Based on Machine Learning and Neural Network for Smart Home," IEEE Access, vol. 6, pp. 49237-49247, Sept. 2018, doi:10.1109/ACCESS.2018.2868984.