**Final Proposal for**

**The Project**

On

**IntelliHome**

**Group Number:** 8.

**Domain:** Internet of Things, Machine learning.

**Team Members**

| Roll No | Name | Role/Task assigned | email | mobile | sign |
| --- | --- | --- | --- | --- | --- |
| 0072 | Kavishwar Prashant Khankari | IoT | kavikhankari@gmail.com | 7828870528 |  |
| 0033 | Veeraja Prashant Ghate | ML | ghateveeraja@gmail.com | 9579502944 |  |
| 0067 | Aryan Vivek Desai | Frontend | aryan04vd@gmail.com | 8484064881 |  |
| 0008 | Mayur Maludeo Shriram | Backend |  |  |  |

**Type of Project:** Product

**Mentor:** None

**TE students associated with** : None

**Inner Pages:**

** Keywords:**

1. IoT (Internet of Things)
2. Home Automation
3. Machine Learning
4. Smart Homes
5. Energy Efficiency
6. Real-time Monitoring
7. Personalization
8. Home Security

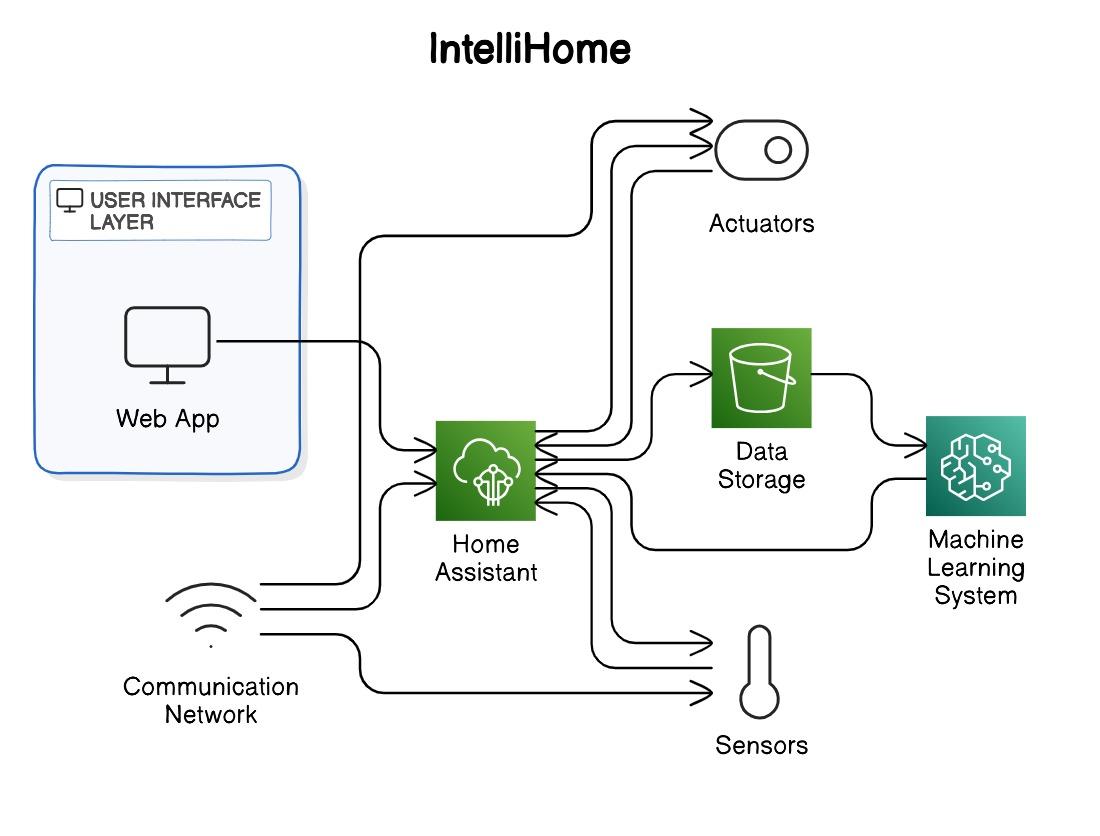
** Problem Definition:**

Existing home automation systems lack adaptability to individual preferences and require manual configuration. This project aims to develop an IoT-based home automation system that uses machine learning to analyze behavior patterns of each resident, automating functions like lighting, temperature, and security. The solution will provide personalized, dynamic control that evolves based on user habits, enhancing convenience, energy efficiency, and security.

** Abstract:**

With the rapid advancement of technology, smart home automation systems are becoming increasingly essential for enhancing modern living. However, many existing systems lack the adaptability needed to cater to individual user preferences and require significant manual configuration. This project focuses on the development of an IoT-based home automation system integrated with machine learning to provide an intelligent, personalized solution. The system automates key household functions, including lighting, temperature control, and security monitoring, by analyzing resident behavior patterns and environmental conditions. Leveraging IoT sensors, actuators, and cameras, the system collects real-time data to enable dynamic control of the home environment. A centralized mobile or web application provides users with real-time monitoring and remote control of various functions, allowing them to manage their homes seamlessly from any location. Machine learning algorithms are used to analyze data and optimize energy consumption by automating tasks like adjusting lights or temperature based on occupancy and user preferences. Additionally, the system enhances security through real-time surveillance, threat detection, and instant notifications in case of suspicious activity. Designed with scalability in mind, the architecture supports the integration of additional IoT devices and future upgrades without major disruptions. Reliability and security are prioritized through encrypted communication protocols and local processing to minimize latency. This project aims to deliver a smart home system that not only improves convenience and safety but also optimizes energy efficiency, reducing human intervention in routine tasks. By combining the power of IoT and machine learning, the proposed system paves the way for a smarter, more adaptive, and sustainable living environment, addressing the growing demand for intelligent home automation solutions.

** System architecture:**

**System Architecture Description:**

**1. User Interface Layer**

Provides a centralized platform where users can interact with the system. The web application allows control over smart home features like lighting, temperature, and security.

**2. Home Assistant**

Acts as the central hub for communication between sensors, actuators, and the cloud server. It gathers real-time data from various smart sensors and relays it to the cloud server.

**3. Data Storage**

Used for storing data and enabling remote access. It holds historical data about sensor inputs and user interactions, which is used to train machine learning models and optimize performance over time.

**4. Machine Learning System**

Machine learning algorithms analyze user habits and preferences based on data collected from smart sensors. This enables personalized automation for each user, adjusting settings dynamically to improve comfort, energy efficiency, and security.

**5. Sensor Layer**

Collect real-time data from the environment, such as room temperature, light levels, and motion detection. These sensors provide the necessary inputs for automation and machine learning models to adjust home settings accordingly. Cameras, Door/Window Sensors capture video footage for surveillance and monitor entry points to enhance home security. Machine learning models process this data for threat detection, identifying unusual activities and alerting the homeowner.

**6. Actuation Layer**

Perform physical tasks such as turning on/off devices or adjusting heating and lighting levels based on commands received from the system. Actuators are controlled via the Home Assistant in real-time based on user preferences or system predictions. Automation includes integration with home appliances like smart refrigerators, washing machines, and entertainment systems, offering comprehensive control over the entire home.

**7. Communication Layer**

Facilitates seamless communication between all IoT devices, sensors, actuators, and the gateway.

**List of modules:**

**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:

* Connect and 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:

Database: NoSQL (MongoDB) for sensor data, SQL (PostgreSQL) for user management.

Cloud Storage options like AWS S3 for large data handling.

**4. Machine Learning System**

Tasks:

* Develop and train models based on sensor data (temperature, occupancy, etc.).
* Perform predictive tasks (e.g., energy usage prediction, smart heating control).
* 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 Temparature, 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.

**Integration Plan:**

* Web App development and testing in parallel with Home Assistant backend.
* Create simple data storage mechanisms and gradually move to cloud-based or larger solutions.
* Develop a basic machine learning model with dummy data and progressively train with real sensor data.
* Integrate sensors and actuators to start communicating with the Home Assistant and Web App.
* Test full system connectivity and flow using the Communication Network module.

**Functionalities:**

* **Personalized Lighting Control**
  + Functionality: The system will adjust lighting automatically based on user behavior and preferences. For example, lights will turn on when a user enters a room and adjust brightness based on ambient light detected by sensors.
  + Automation: Using motion and light sensors, the system will learn patterns (e.g., when and where lights are typically used) and adjust them accordingly, saving energy by turning off lights when not needed.
* **Smart Temperature Regulation**
  + Functionality: Automatically adjusts heating, ventilation, and air conditioning (HVAC) systems based on room occupancy and temperature sensors. The system learns user preferences over time (e.g., preferred temperatures for different times of the day).
  + Automation: Machine learning algorithms will predict when the user is likely to need heating or cooling, optimizing temperature control to reduce energy usage while maximizing comfort.
* **Security and Surveillance**
  + Functionality: Real-time surveillance through security cameras with motion detection and video analytics. The system will detect potential threats, recognize familiar faces, and notify the homeowner of any suspicious activities.
  + Automation: Cameras will be activated when motion is detected, and the system will alert users or automatically lock doors if a security breach is identified. Intelligent access control will allow authorized users to enter without manual intervention (e.g., via facial recognition).
* **Appliance Control and Automation**
  + Functionality: Control home appliances like coffee makers, TVs, or washing machines. The system learns usage patterns and automates appliance control based on historical behavior.
  + Automation: Automatically turns off unused devices to save power, or turns on appliances when needed, like brewing coffee in the morning based on user schedules.
* **Energy Optimization**
  + Functionality: Monitors the energy consumption of devices and optimizes usage by turning off devices when not in use or during low-usage periods. The system identifies patterns of energy use and provides suggestions for reducing waste.
  + Automation: Automatically turns off high-power appliances (e.g., HVAC or lighting) when they are not needed, based on user absence or reduced activity.
* **Centralized Control via Web Interface**
  + Functionality: Users can control all home systems from a centralized platform (mobile app or web dashboard). This includes manually adjusting lights, temperature, security settings, and appliances from anywhere.
  + Automation: Allows remote monitoring and control, providing status updates, alerts, and detailed reports on energy usage, system performance, and security events.
* **Real-Time Alerts and Notifications**
  + Functionality: Users receive real-time notifications for important events, such as a security breach, a device malfunction, or unusual energy consumption.
  + Automation: Alerts are sent based on predefined rules, user preferences, or the detection of anomalies by the machine learning model. The system can also trigger certain actions automatically in response to these alerts (e.g., locking doors or adjusting temperature).
* **Behavioral Learning and Predictive Automation**
  + Functionality: The system uses machine learning to track and analyze user behavior over time. It makes predictions based on this data to automate future actions (e.g., adjusting lights or appliances before the user even interacts with them).
  + Automation: The system predicts and automates daily routines (e.g., adjusting the thermostat when the user typically wakes up) and evolves its behavior based on new data and habits.

** Literature survey:**

**Current market survey:**

**1. SmartThings by Samsung**

Pros:

* Integration: Offers seamless integration with a wide range of IoT devices, including lights, cameras, and thermostats.
* Automation: Supports user-defined automation rules through routines (e.g., motion triggers lights, or door sensors activate security cameras).
* Real-time Monitoring: Features real-time monitoring through connected devices like sensors and cameras.

Cons:

* Behavior Adaptation: Does not inherently learn from user behavior patterns; automation is largely rule-based and requires manual configuration.
* Complex Setup: Though powerful, the setup and management of multiple devices can be cumbersome, especially for non-tech-savvy users.
* Cloud Dependency: Relies on the cloud for advanced features, leading to privacy concerns and potential delays during internet outages.

**2. Apple HomeKit**

Pros:

* Privacy and Security: High emphasis on user privacy, with end-to-end encryption of data.
* Integration: Integrates smoothly with Apple ecosystem devices, including iPhones, iPads, and Macs for centralized control.
* Automation: Offers location-based automation (e.g., lights turning on as users arrive home).

Cons:

* Limited Compatibility: Less flexible with third-party devices compared to competitors like Google Home or Alexa.
* Learning Capabilities: Lacks machine learning capabilities for behavior-based automation, relying on preset conditions instead of evolving based on user habits.

**3. Nest (Google Nest)**

Pros:

* Adaptive Learning: Learns user preferences for temperature control, adjusting heating/cooling systems over time to optimize comfort and energy usage.
* Real-time Monitoring: Features real-time video surveillance with intelligent motion detection.
* Voice Control: Offers integrated voice control via Google Assistant for seamless command execution.

Cons:

* Limited Custom Automation: While Nest's thermostat offers learning capabilities, other areas (e.g., lighting or security) lack deep personalization or behavior-based adaptability.
* Subscription Dependency: Advanced security features (e.g., continuous video recording) require a subscription, adding ongoing costs.

**4. Vivint Smart Home**

Pros:

* Custom Automation: Offers extensive automation capabilities that integrate with security systems, including automatic door locking/unlocking and smart lighting.
* Real-time Security: Provides real-time video monitoring with AI-powered object detection (e.g., distinguishing between people, pets, or packages).
* Professional Monitoring: Includes 24/7 security monitoring, offering immediate alerts to emergencies.

Cons:

* Learning Limitations: Automation is largely rules-based and lacks deep adaptive learning of individual behaviors over time.
* High Cost: Premium pricing with additional fees for monitoring services and equipment.
* Complex Setup: Requires professional installation, making it less accessible for DIY enthusiasts.

**Related Reference Papers:**

**An Unsupervised User Behavior Prediction Algorithm Based on Machine Learning and Neural Network for Smart Home**

**Summary:**

* Objective:

The study aims to improve the intelligence of smart home systems by predicting user behavior using large datasets of operational data. By leveraging unsupervised machine learning and neural networks, the system can enhance automation and offer personalized services without extensive manual intervention.

* Algorithm Proposed:

The Unsupervised User Behavior Prediction (UUBP) algorithm incorporates a novel Artificial Neural Network (ANN) and a forgetting factor inspired by the Ebbinghaus Forgetting Curve. This factor helps the system prioritize recent user behavior while "forgetting" older, infrequent actions.

* Key Features:

Unsupervised Learning: The system can predict user behaviors autonomously, using mass historical and real-time operation records.

Forgetting Factor: Helps reduce the influence of outdated or infrequent actions to focus on recent and relevant behaviors.

Self-Organizing Learning: Minimal manual intervention is required to set parameters or clustering numbers, unlike older algorithms like K-means.

* Data Handling:

The UUBP algorithm preprocesses the dataset by transforming it into a format compatible with machine learning algorithms.

It uses an ANN to cluster user behaviors, identifying relevant patterns that inform predictive actions.

* Experimental Setup:

The algorithm was evaluated using real user data from a smart home setup with 10 devices. The performance was compared against traditional algorithms such as K-means and DBSCAN.

**Findings:**

* Performance:

The UUBP algorithm demonstrated superior performance over traditional clustering algorithms (K-means, SOMNN, and DBSCAN).

Metrics such as Compactness Index (CP), Separation Index (SP), and Davies-Bouldin index (DB) showed that UUBP offers better clustering compactness, separation, and overall predictive accuracy.

* Real-Time Learning:

The use of a forgetting factor allows the system to adapt to real-time changes in user behavior, reducing the influence of older behaviors and improving the relevance of predictions.

It outperformed algorithms that did not prioritize recent user actions, showing better recognition of current user needs.

* Higher Autonomy:

UUBP requires significantly less manual intervention compared to traditional models. The ANN-based clustering mechanism allowed for automatic detection of behavior patterns, improving ease of use.

* Enhanced Smart Home Intelligence:

The system moves beyond simple remote control and pre-defined responses by enabling predictive intelligence. This results in more dynamic, context-aware smart home interactions that adjust to evolving user behaviors.

**ARAS Human Activity Datasets in Multiple Homes with Multiple Residents**

**Summary:**

The paper "ARAS Human Activity Datasets in Multiple Homes with Multiple Residents" by Hande Alemdar et al. presents a dataset designed for human activity recognition in real-world environments. The ARAS dataset was collected from two homes, each equipped with 20 ambient binary sensors over a two-month period. It includes ground truth labels for 27 distinct activities performed by multiple residents, offering a diverse set of data that reflects natural human behavior in a domestic setting. The authors emphasize the significance of publicly available annotated datasets for developing machine learning models capable of recognizing human activities, particularly in the context of healthcare monitoring systems.

**Findings:**

* Dataset Composition:

The ARAS dataset comprises sensor readings and activity labels, capturing various everyday activities (e.g., cooking, sleeping) in a natural environment.

It includes data collected over two months from multiple residents, enhancing its applicability for real-world scenarios.

* Sensor Deployment:

The study utilized non-intrusive sensors such as force-sensitive resistors, proximity sensors, and temperature sensors, ensuring privacy and minimal disruption to residents.

The authors successfully selected and deployed sensors that accurately captured the targeted activities.

* Machine Learning Experiments:

Preliminary experiments using Hidden Markov Models (HMMs) for activity recognition achieved average accuracies of 61.5% for one home and 76.2% for the other.

The findings indicate potential improvements in recognition performance with the application of more advanced machine learning techniques.

* Challenges Identified:

The paper discusses challenges related to sensor selection, data quality, and the accuracy of activity labeling. The authors emphasize the trade-off between maintaining natural behavior and achieving precise activity annotation.

* Importance for Future Research:

The ARAS dataset is deemed valuable for the development of robust human activity recognition systems, particularly for smart health monitoring applications. The authors encourage the research community to leverage this dataset for future advancements in the field.

**Design for Visitor Authentication Based on Face Recognition Technology Using CCTV**

**Summary:**

* Introduction:

The study explores the growing need for enhanced security systems, focusing on face recognition technologies for smart home services. The proposed system uses CCTV to detect and authenticate visitors in real-time, addressing limitations in current security systems.

* System Design:

The system architecture is divided into four parts:

* Deep Learning Model Training: The model is trained on various face datasets, labeling 7 key facial features.
* Face Detection: A webcam captures images for face recognition, applying deep learning models to detect visitors.
* Recognition and Access Control: The system compares detected faces with a pre-stored database to either allow or deny access.
* Monitoring: Log information like visitor images, times, and number of visitors is stored for post-processing and security audits.
* Face Recognition:

The model used for face detection and recognition is Tiny-YOLOv3, optimized for real-time performance on low-power devices like Jetson Nano. Key facial features, such as the eyes and mouth, are compared against stored data to authenticate visitors. The system implements a multi-factor authentication protocol, including alerts for potential security breaches like password peeking.

* Hardware:

The system's hardware includes a Jetson Nano board, a webcam, and sensors. This setup facilitates real-time face detection and recognition.

* Performance:

The system’s performance was evaluated on two devices—Jetson Nano and Raspberry Pi. While Raspberry Pi struggled with real-time video processing, the Jetson Nano showed superior performance with 86.3% accuracy and 6.5 frames per second (FPS). Tiny-YOLOv3 was chosen for its balance between speed and accuracy.

* Security Features:

The system includes mechanisms to detect multiple visitors, check for shoulder surfing attacks (password peeking), and ensure robust user authentication through multiple levels of verification.

**Findings:**

* Real-Time Efficiency:

Tiny-YOLOv3 on the Jetson Nano demonstrated effective real-time processing, balancing speed (6.5 FPS) and accuracy (86.3%) in face detection.

* Cost Efficiency:

The system was designed to be low-cost and standalone, using hardware like the Jetson Nano that can operate without an external server, reducing deployment costs.

* Security Enhancement:

The system successfully enhances home security through face detection, peeking prevention, and access control. It records visitor logs for potential audits and enhances security by detecting unauthorized individuals in real-time.

* Hardware Comparison:

The Jetson Nano outperformed Raspberry Pi in terms of speed and accuracy, making it more suitable for real-time security applications.

** Objectives:**

1. Develop a smart home automation system using IoT and Machine Learning
2. Ensure real-time monitoring and automation of household activities
3. Enhance home security through intelligent access control and surveillance
4. Provide users with seamless control over their home environment through a centralized platform

** Scope of the project:**

1. Automation of lighting, heating, and home appliances
2. Real-time security monitoring with threat detection using Machine Learning
3. Integration of various sensors and IoT devices for real-time data collection
4. Modular architecture allowing future upgrades and scalability.

** Software and hardware requirements:**

**Hardware Requirements:**

* **IoT Sensors**
* Motion Sensors: Used to detect any movement within the home, triggering actions such as turning on lights or activating security cameras. These sensors help automate the process of turning off unnecessary devices when no one is around, improving energy efficiency.
* Temperature Sensors: These sensors monitor indoor temperature and provide feedback to the system to adjust heating, ventilation, and air conditioning (HVAC) accordingly. By analyzing data from these sensors, the system can maintain an optimal environment based on user preferences, improving both comfort and energy usage.
* Light Sensors: Light sensors monitor ambient lighting conditions to adjust artificial lighting accordingly. They can dim or brighten lights based on the time of day or the current lighting levels, ensuring optimal energy consumption.
* **Cameras for Surveillance**

Cameras serve as the core component for security in the smart home system. They will monitor key areas in real-time, providing video feeds and detecting potential security breaches or unauthorized access. Machine learning algorithms can analyze the camera footage to detect unusual behavior or threats, triggering alerts or automated responses like locking doors or turning on alarms.

* **Actuators for Controlling Devices**

Actuators are essential for converting the digital commands from the automation system into physical actions. For instance, when a command is issued to turn on a light or adjust the thermostat, the actuators (often using relays) will enable these devices to respond. These components allow for the real-time execution of tasks such as opening blinds, controlling appliances, or locking doors.

* **Microcontrollers**
* Raspberry Pi: A versatile and powerful microcontroller capable of handling multiple devices and sensors. It’s used for data processing, controlling devices, and interfacing with the system’s central hub.
* Arduino/ESP32: Microcontrollers used for lightweight tasks like interfacing with sensors and controlling actuators. They provide the computational power to handle device-level commands and process sensor data.
* **GPU Powered Local Inference Machine**

A local machine equipped with a Graphics Processing Unit (GPU) is critical for processing machine learning models efficiently, especially for tasks like real-time image recognition and behavioral pattern analysis. This ensures that the system can run complex algorithms locally, without relying on cloud resources, providing faster responses and improving privacy since data doesn’t need to be continuously uploaded.

* **Wireless/Wired Connection Spanning All Sensor Locations**

The system will use both wired and wireless connections to ensure reliable communication between all IoT devices and sensors. For stable connections in high-traffic or critical areas (like security cameras or access control), wired connections will be preferred. Wireless communication (using Wi-Fi, Zigbee, or Bluetooth) will be used for mobility and convenience in non-critical devices. A robust network infrastructure ensures the system operates seamlessly with low latency.

**Software Requirements**

* **Operating System**
  + Raspbian: A Linux-based operating system optimized for Raspberry Pi. It will serve as the main operating system for managing the IoT devices, microcontrollers, and local machine learning inference. Raspbian provides a lightweight environment that’s perfect for handling the system’s central hub, sensor communication, and processing tasks.
  + Windows: Used on more powerful machines, especially if you plan to leverage specific machine learning tools or run more complex simulations. It can also serve as the interface for development and cloud integration.
* **Programming Languages** 
  + Python: The primary language for writing machine learning algorithms, handling cloud interaction, and controlling IoT devices. Python’s extensive libraries (e.g., TensorFlow, OpenCV) and support for IoT frameworks make it ideal for building a robust home automation system.
  + C++: Used for low-level device control, particularly on microcontrollers like Arduino. C++ allows for precise control of hardware components, ensuring optimal performance in the communication between sensors and actuators.
* **Machine Learning Libraries**
  + TensorFlow: A powerful machine learning framework used to train and deploy the algorithms that will analyze user behavior patterns and control automation. TensorFlow can handle tasks like predicting user preferences, identifying anomalies in home security footage, and optimizing energy usage based on past data.
  + OpenCV: A computer vision library that will process the camera feeds for tasks like facial recognition, motion detection, and real-time surveillance. OpenCV can be integrated with TensorFlow to enhance the machine learning capabilities of the system, enabling it to identify objects and detect security threats.
* **Cloud Platforms for Data Storage and Remote Access**

AWS (Amazon Web Services): AWS provides scalable storage solutions for storing user data, camera footage, and IoT sensor data. It also allows remote access to the system, enabling users to monitor and control their homes from anywhere.

** Expected Outcomes:**

**Currently Working On**

* **Integration of Sensors with Home Assistant:**

**Task:** We are in the process of integrating various IoT sensors into our chosen platform, Home Assistant.

**Challenges:** Ensuring compatibility and smooth data transmission from sensors to the platform.

**Approach:** Researching and testing different protocols and methods to link the sensors effectively to Home Assistant, ensuring real-time data is captured.

* **Machine Learning Algorithm Selection:**

**Task:** We are evaluating machine learning algorithms for predicting user behavior and usage patterns based on the sensor data.

**Challenges:** Identifying an algorithm that balances prediction accuracy with computational efficiency.

**Approach:** Exploring algorithms such as decision trees, random forests, or neural networks that fit our project’s scope. Initial testing is being done using sample datasets to compare performance.

* **Next Steps:**

Finalizing sensor integration with Home Assistant.

Completing comparative analysis of ML algorithms and implementing the most suitable one for prediction.

** Probable date of completion:**

15-02-2025

** References:**

* [Design for Visitor Authentication Based on Face Recognition Technology Using CCTV | IEEE Journals & Magazine | IEEE Xplore](https://ieeexplore.ieee.org/document/9955529)
* [An Unsupervised User Behavior Prediction Algorithm Based on Machine Learning and Neural Network For Smart Home | IEEE Journals & Magazine | IEEE Xplore](https://ieeexplore.ieee.org/document/8458105)