**Software Engineering (W6)**

**Activity no. 5**

**Submitted to:**

**Ma’am Kinza Sardar**

**Submitted by:**

**Eisha -tir- Raazia**

**F2023065346**

**BS Software Engineering (3rd Semester)**



Dated: 15/1/2025

**Problem/proposal:**

**Smart Home Energy Consumption Monitor -**A real-time energy monitoring system that helps users identify areas for energy conservation and comprehend their consumption trends.

**Step 1: Planning and Requirements Collection:**

**Identify Objectives:**

Your system need to monitor the amount of energy used by each device, examine use trends, and offer recommendations for ways to cut costs.

**Investigation:**

Examine related energy-tracking apps or Internet of Things gadgets to learn how they work and what makes them easy to use.

**Functional requirements (FR):**

**FR1:**

User Account (Sign up or login)

**FR2:**

System Dashboard

**FR3:**

Real-Time Energy Monitoring

**FR4:**

Device-Wise Breakdown

**FR5:**

Consumption Trends and Analytics

**FR6:**

Energy Cost Estimation

**FR7:**

Customizable Alerts and Notifications

**FR8:**

Energy Conservation Recommendations

**FR9:**

Remote Monitoring and Control

**FR10:**

Integration with Smart Home Platforms

**FR11:**

Multi-User Support

**FR12:**

Device management

**Non-Functional requirements (NFR):**

**NFR1:**

Performance

**NFR2:**

Scalability

**NFR3:**

Reliability

**NFR4:**

Usability

**NFR5:**

Security

**NFR6:**

Maintainability

**Set a Timeline:** Set reasonable due dates for each step of the work and divide it out over several months.

**Stakeholders**:

* Utilities
* Electric consumers (public)
* System integrator (Admin)
* IoT Technology providers (hardware, software)

**Step 2: Designing:  
Describe the primary elements:**   
**Front-end:** Data visualization dashboard  
**Back-end:** Data processing server  
**Database:** retains usage history  
**Device API:** To interact with devices when working with the Internet of Things, you will require APIs.

**Step 3: Choosing a Technology**

**Front-end**: For a responsive interface, think about utilizing HTML, CSS, JavaScript, or a framework like React.  
**Back-end:** Python using Django or Flask to process and handle data requests.  
**Database:** MySQL or MongoDB for more reliable data processing; SQLite for local development.  
**IoT Integration:** If genuine IoT devices aren't available, you can utilize modeling. For real-time data, think about using MQTT or REST APIs to interface with devices.

**Step 4: Improvement:**

**Back-end development:** Create a database to hold information about energy usage.  
Provide APIs to update consumption records and get device use.  
**Front-end development:** Create the dashboard interface to provide device data (graphics, use insights) in an approachable way.  
Make a section or page with suggestions for energy conservation.

**Source of IoT/Simulated Data:**

Configure APIs to retrieve device data if you're using the IoT.  
If simulating, generate fictitious statistics to show the energy consumption of the gadget over time.

**Step 5: Evaluation:**

**Unit Testing:** Examine each of your code's distinct operations, including data storage, retrieval, and computation techniques.  
**Integration testing:** Verify that front-end elements accurately retrieve and present data from the back end.  
**User Testing:** To make sure your dashboard is clear and simple to use, have others utilize it. Get input regarding usability.

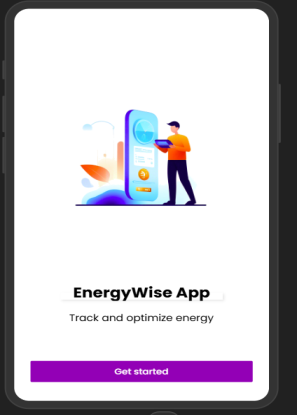
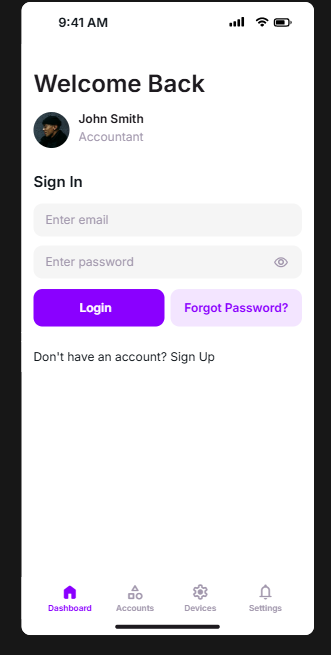
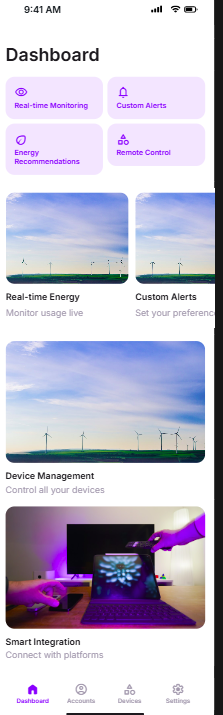
**Step 6: Records:**

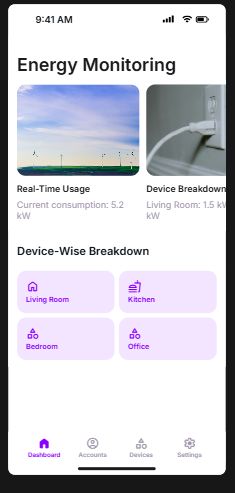
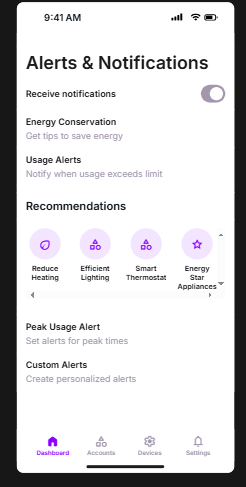
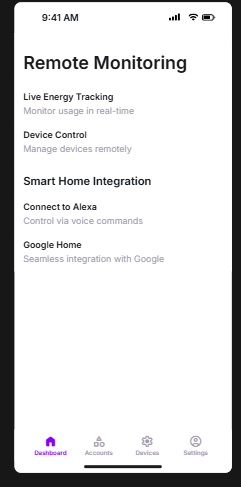
**Technological documentation: It i**ncludes any technological decisions you make as well as yourcode-base and APIs.  
**User Manual:** Compose a manual that explains how to utilize every dashboard function.  
**Upcoming Improvements:** Keep track of any future enhancements or new features that might be added.

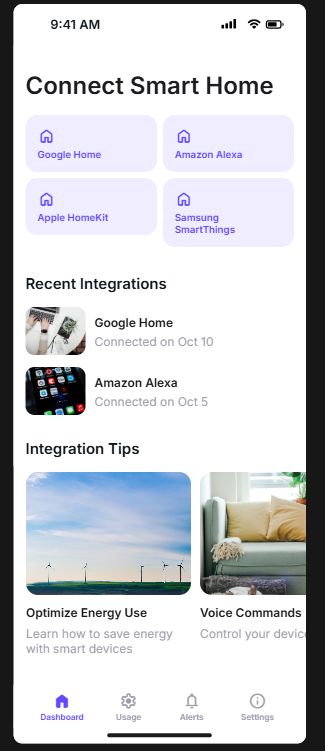
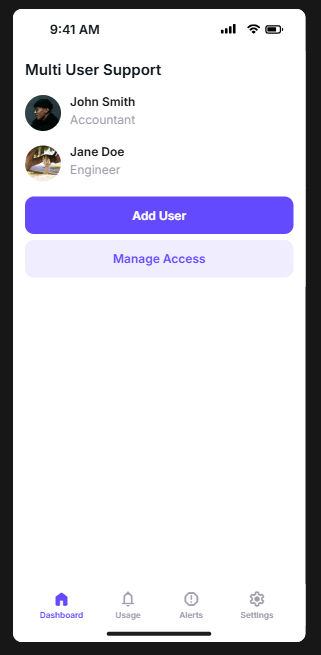
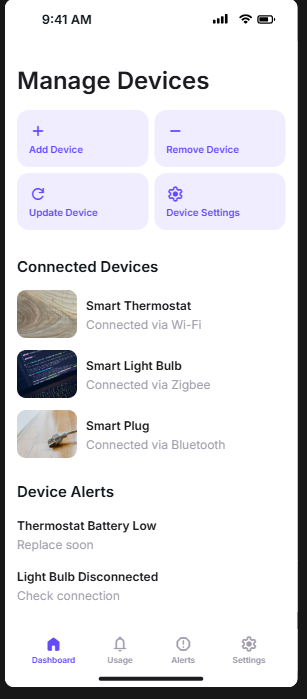
**Blackbox testing:**

1. **User-Centric Approach**
   * Validates the system from the end-user's perspective.
   * Ensures features like energy monitoring, notifications, analytics, and integrations work as intended.
2. **Focus on Functional Requirements**
   * Tests core features such as real-time monitoring, device-wise breakdown, multi-user support, and platform integrations.
   * Verifies that functional requirements are fully met.
3. **Scalability Testing**
   * Evaluates the system's performance with increased devices and users.
   * Ensures smooth handling of higher loads and interactions.
4. **No Need for Code Access**
   * Testers only need input and expected output to perform tests.
   * Ideal for scenarios where testers and developers are separate or external testers are involved.

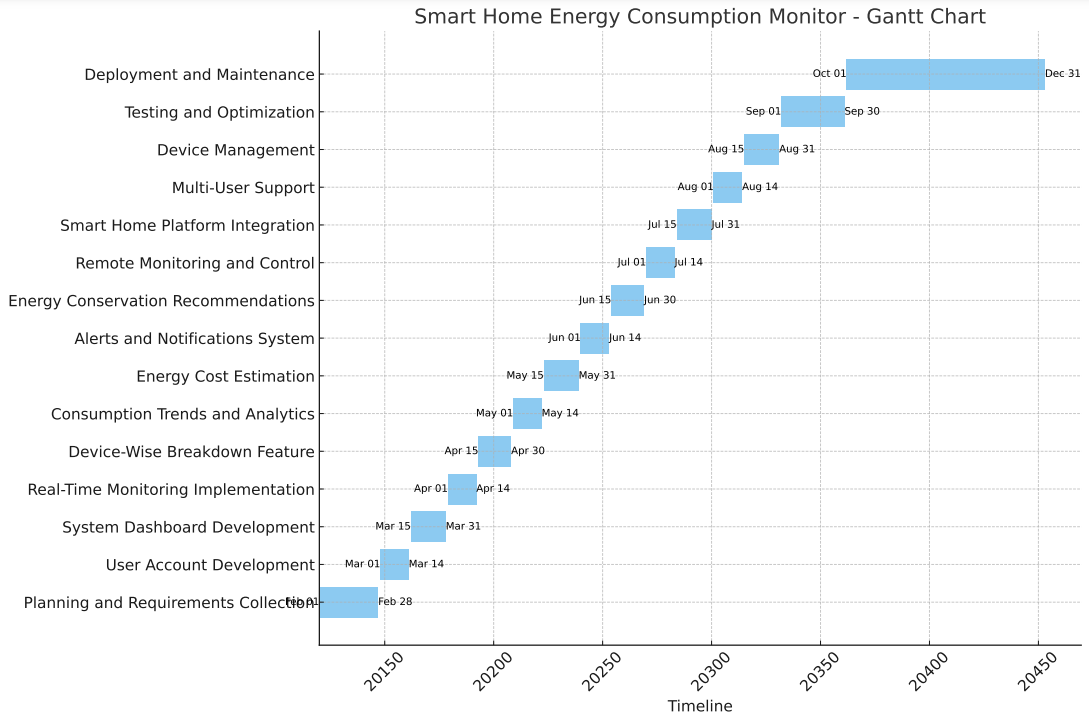
Prototypes:

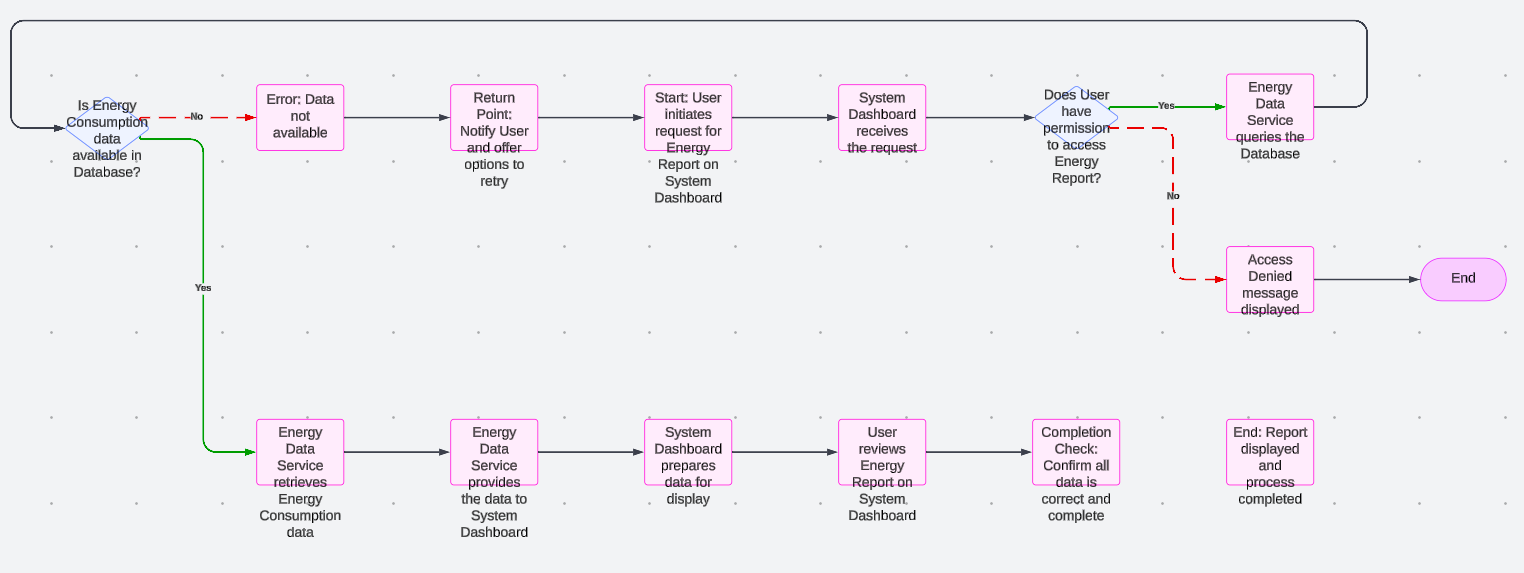
  

**  **

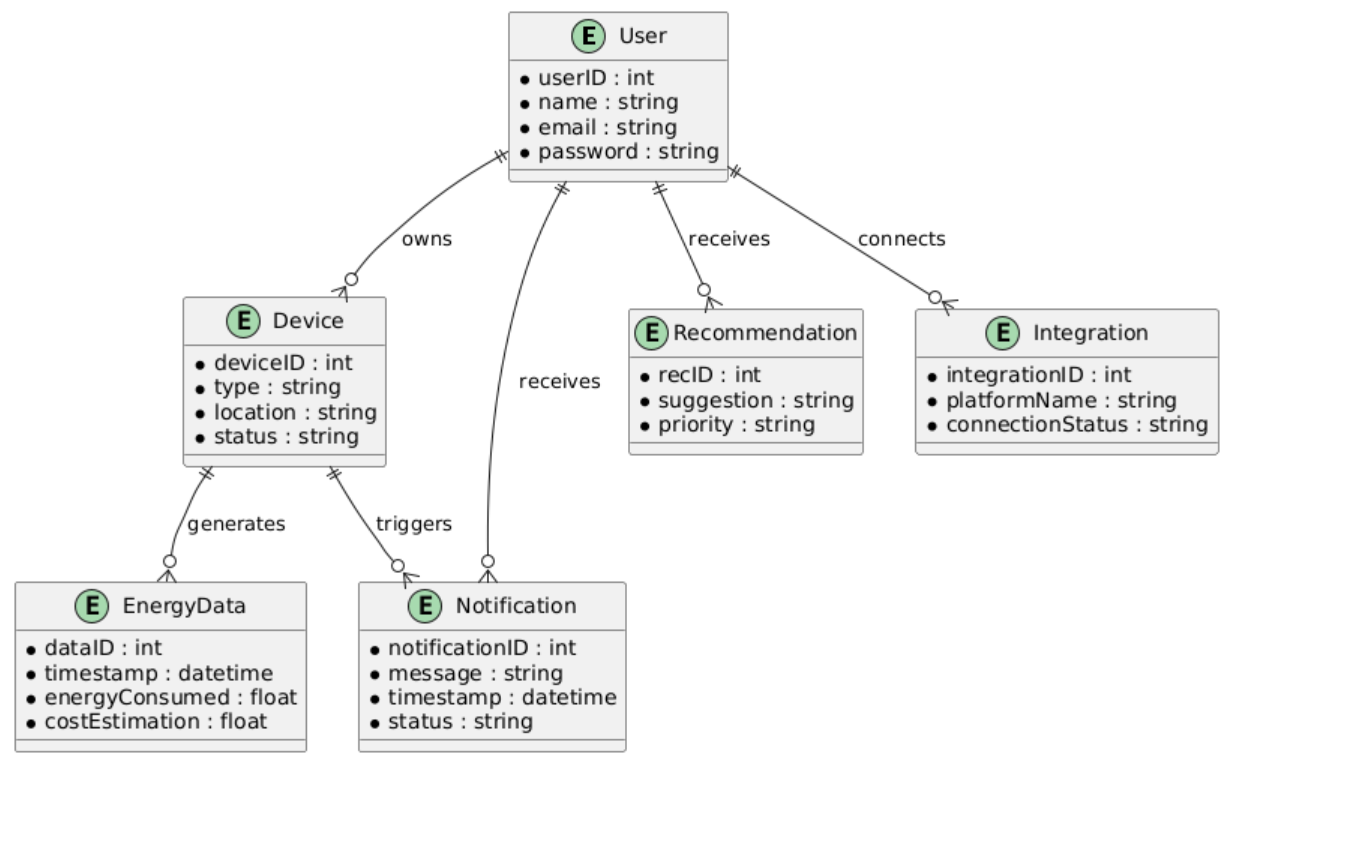
**Gantt Chart:**



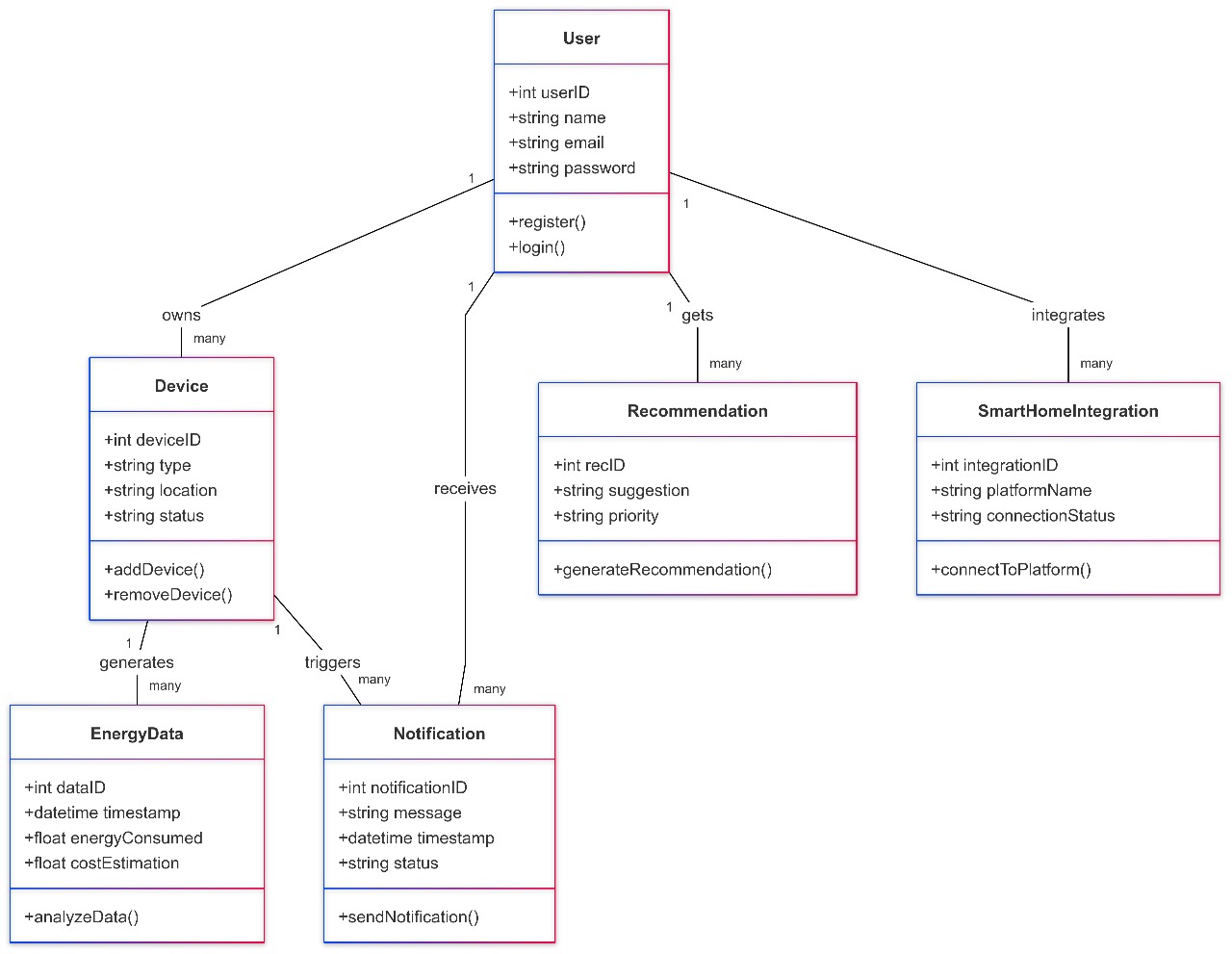
**Sequence Diagram:**

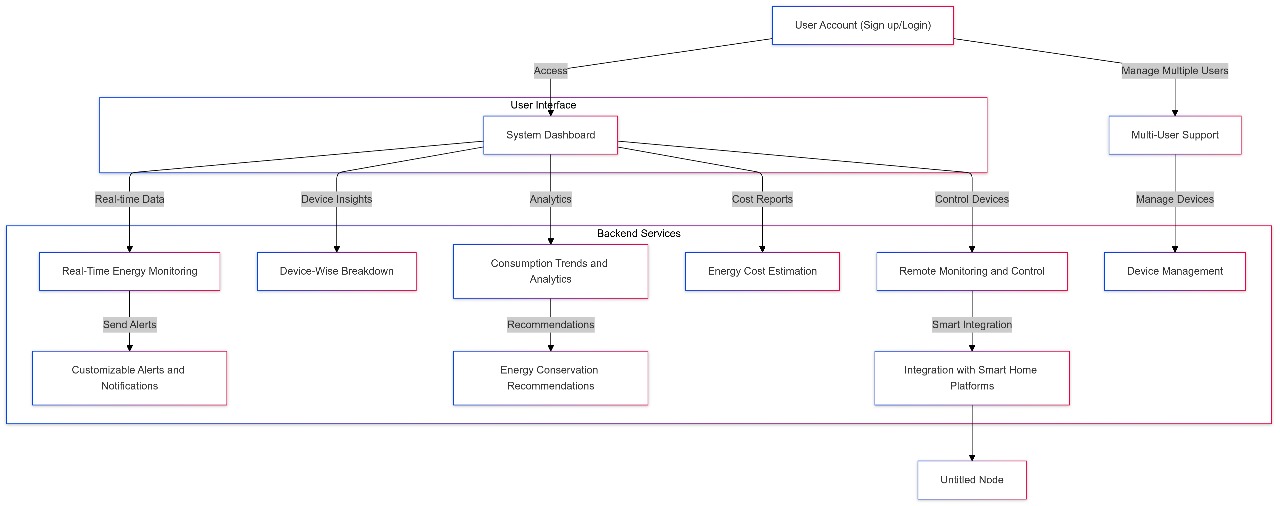
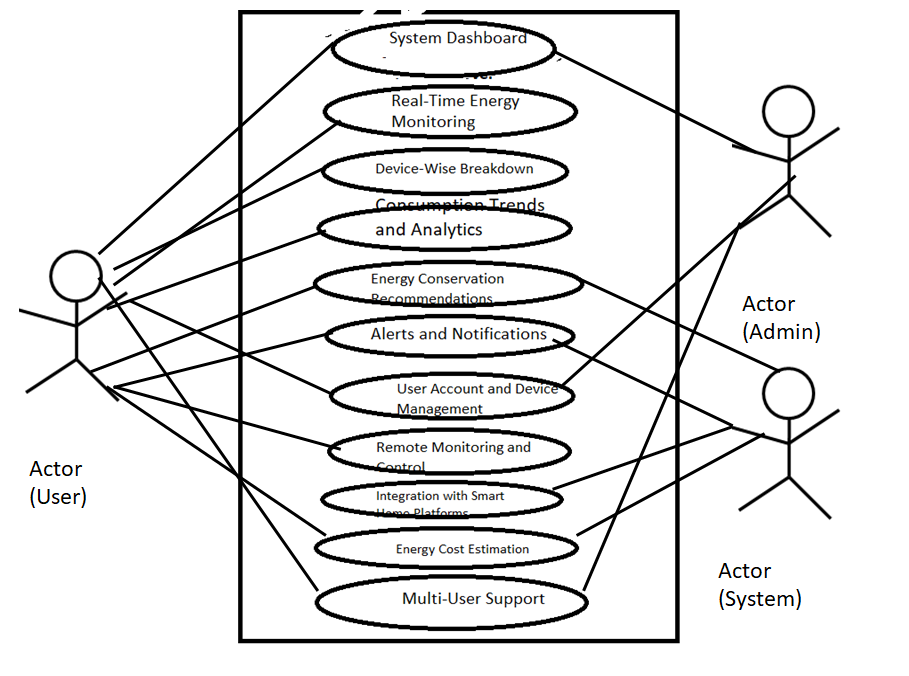


**ERD:Top of Form**



Class diagram:

****

Architecture diagram:Use Case:

Use case Diagram

Use case table:

