**UNIVERSITY OF MANAGEMENT AND TECHNOLOGY**

**Subject Name: Software Engineering [v20]**

**Submitted to: Shahbaz Qadeer**

**Submitted by: Maria Hanif**

**Student ID : F2023266779**

**BS Computer Science(3rd Semester)**

**Report :**

**Smart Home Energy Consumption Monitor**

**Overview:**

The Smart Home Energy Consumption Monitor is a real-time system designed to help users track and optimize their household energy usage. It provides insights into energy consumption patterns, identifies high-energy-consuming devices, and offers actionable recommendations for energy conservation. The system enhances convenience by integrating with smart home platforms, enabling remote monitoring and control.

**Objectives:**

* Provide **real-time energy tracking** and insights.
* Identify energy-intensive devices and inefficiencies.
* Offer **personalized conservation recommendations.**
* Enable **remote monitoring and control** via mobile/web apps.
* Integrate with smart home ecosystems for seamless use.
* Promote **cost savings and sustainability.**

**Scope:**

* **In-Scope:**
* User authentication and dashboard visualization.
* Real-time monitoring and device-wise consumption breakdown.
* Energy cost estimation and conservation tips.
* Alerts, notifications, and remote device control.
* **Out of Scope:**
* Industrial/commercial energy monitoring.
* Hardware development or direct utility billing integration.

**Key features:**

* **Real-Time Monitoring** – Track energy usage live with device-specific insights.
* **Consumption Analytics** – View historical data trends and forecasts.
* **Energy Cost Estimation** – Estimate electricity bills based on usage.
* **Smart Alerts & Notifications** – Get notified on unusual consumption patterns.
* **Remote Control** – Turn devices on/off via app or voice commands.
* **Smart Home Integration** – Works with Alexa, Google Home, etc.

**Stakeholders:**

* **Homeowners –** End-users who wish to track and reduce their energy costs.
* **Utility Providers** – Potential partners for data analytics and efficiency programs.
* **Smart Device Manufacturers** – Companies providing compatible IoT appliances.
* **Government/Environmental Agencies** – Interested in promoting sustainable energy practices.

**Benefits:**

**For Users:**

* Reduced energy bills by **10-30%.**
* Better awareness of household consumption habits.
* Convenience through remote monitoring and control.
* Lower environmental impact via optimized energy usage.

**Success Criteria:**

* System uptime of **99.9% ensuring reliability.**
* User engagement of at least **80% retention within 6 months.**
* Reduction in household **energy consumption by 15-25%.**
* Integration with at least 3 major smart home platforms.
* Positive feedback with a **90% user satisfaction rate.**

**Technical Overview:**

**Architecture:**

* **Front-end**: Web and mobile applications (React/Flutter).
* **Back-end:** Cloud-based service (AWS/Azure, Node.js/Python).
* **Database:** Scalable cloud storage (MongoDB, PostgreSQL).
* **IoT Integration:** Communication via Wi-Fi/Bluetooth/Smart Hub.
* **Security Measures**: OAuth 2.0 authentication, data encryption

(AES-256).

**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):**

* Secure authentication via email/password, social logins (Google, Facebook).
* Support for two-factor authentication (2FA).
* Response time: < 2 seconds for login/signup.

**(FR2) System Dashboard:**

* Overview of energy usage, cost, and efficiency.
* Load time: < 3 seconds.
* Widgets for real-time insights and alerts.

**(FR3) Real-Time Energy Monitoring:**

* Data refresh rate: every 5-10 seconds.
* Accuracy: ±2% deviation from actual usage.

Supports up to 100 devices simultaneously

**(FR4) Device-Wise Breakdown:**

* Granular consumption insights for each device.
* Energy consumption threshold alerts.
* Data retention for up to 1 year.

**(FR5) Consumption Trends and Analytics:**

* Historical data visualization (daily, weekly, monthly).
* Trend forecasting with ±5% accuracy.
* Export reports in PDF/CSV format.

**(FR6) Energy Cost Estimation:**

* Calculates based on real-time tariff rates.
* Supports multiple tariff plans and peak hours.
* Error margin: ±3% in estimated costs.

**(FR7) Customizable Alerts and Notifications:**

* Email, SMS, and push notifications.
* Response time for critical alerts: < 1 second.
* Supports custom thresholds and conditions.

**(FR8) Energy Conservation Recommendations:**

* AI-driven suggestions with 80%+ accuracy.
* Personalized based on user consumption patterns.
* Estimated savings: 10-20% monthly.

**(FR9) Remote Monitoring and Control:**

* Control devices remotely with < 2-second latency.
* Compatible with mobile and web apps.
* Supports scheduling and automation

**(FR10) Integration with Smart Home Platforms:**

* Compatible with Google Home, Alexa, Apple HomeKit.
* Response time: < 2 seconds for command execution.
* Supports API integrations for third-party apps.

**(FR11) Multi-User Support:**

* Role-based access (admin, user, guest).
* Supports up to 10 simultaneous users per account.
* Audit logs maintained for 6 months.

**(FR12) Device Management:**

* Add/remove devices within 10 seconds.
* Supports auto-discovery for smart devices.
* Stores device usage data for up to 1 year.

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

**(NFR1) Performance:**

* System response time: < 3 seconds under normal load.
* Handles 1,000 concurrent users with < 5% degradation.
* API response time: < 200 ms.

**(NFR2) Reliability:**

* System uptime: 99.9% (SLA).
* Failover mechanisms with auto-recovery in < 5 minutes.
* Redundant data backups every 24 hours

**(NFR3) Usability:**

* User-friendly design with accessibility features (WCAG 2.1).
* 95% of users should complete key tasks within 3 clicks.
* Onboarding time: < 5 minutes.

**(NFR4) Security:**

* Data encryption (AES-256) for sensitive information.
* Compliance with GDPR and ISO 27001 standards.
* Role-based access control (RBAC).

**(NFR5) Maintainability:**

* Modular architecture with microservices.
* Mean Time to Repair (MTTR): < 4 hours.
* Automated CI/CD pipeline for seamless updates.

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

**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:**

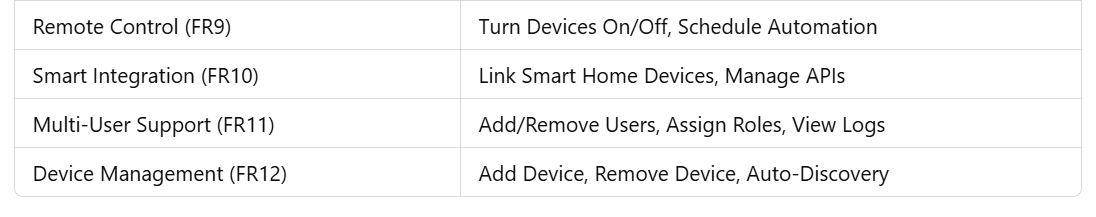
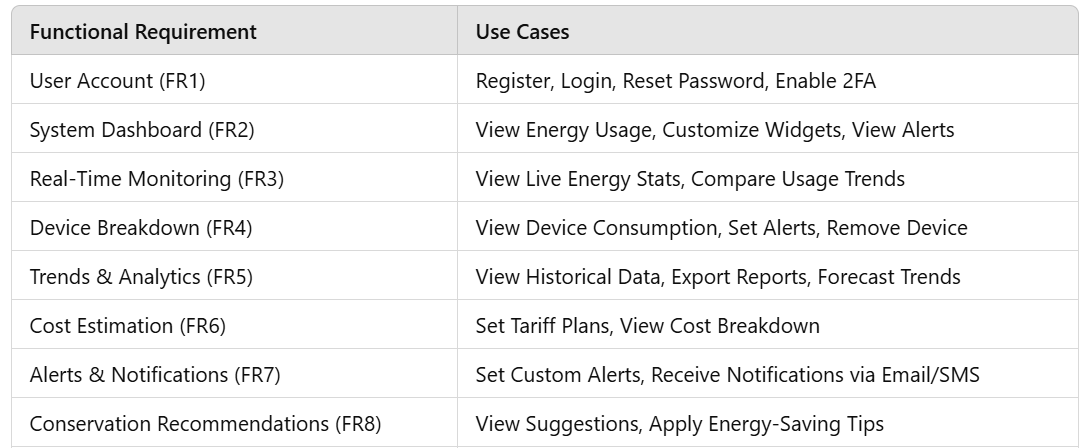
**Unit Testing:** Examine each of your code's distinct operations, including data storage, retrieval, and computation techniques.  
**Integration testing:** Verify that front-end elements accurately retrieve and present data from the back end.  
**User Testing:** To make sure your dashboard is clear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

**Step 6: Records:**

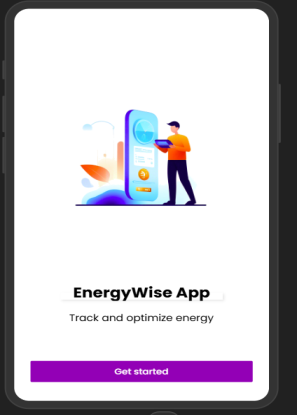
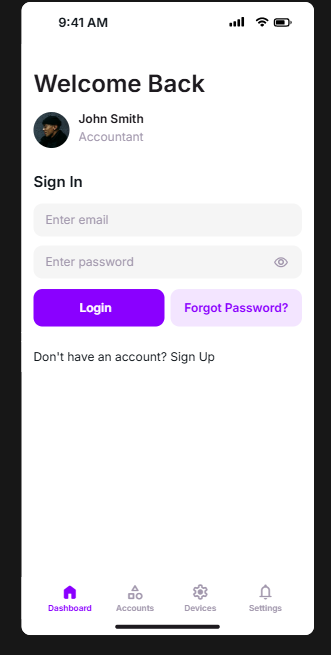
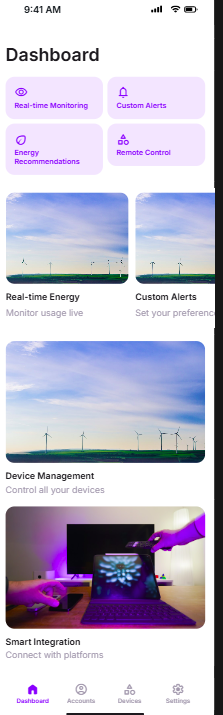
**Technological documentation: It i**ncludes any technological decisions you make as well as yourcode-base and APIs.  
**User Manual:** Compose a manual that explains how to utilize every dashboard function.  
**Upcoming Improvements:** Keep track of any future enhancements or new features that might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

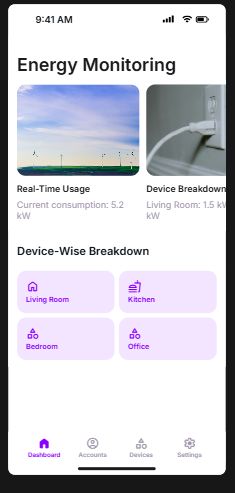
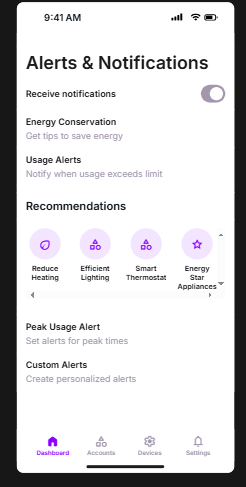
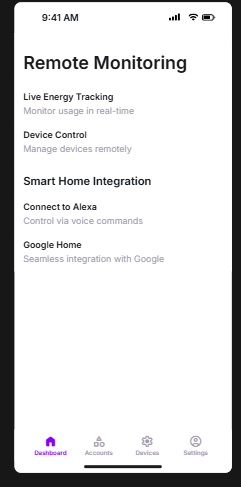
**Blackbox testing:**

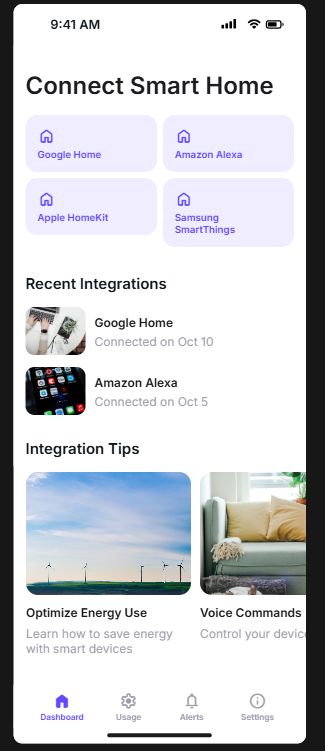
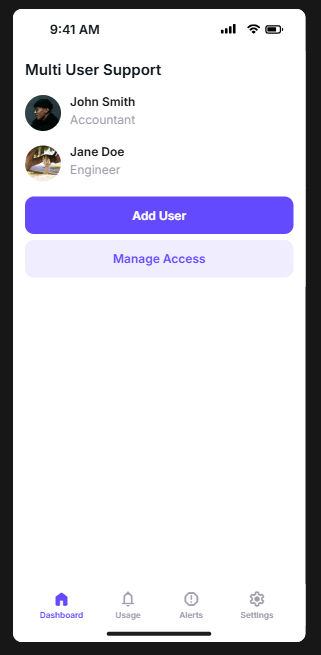
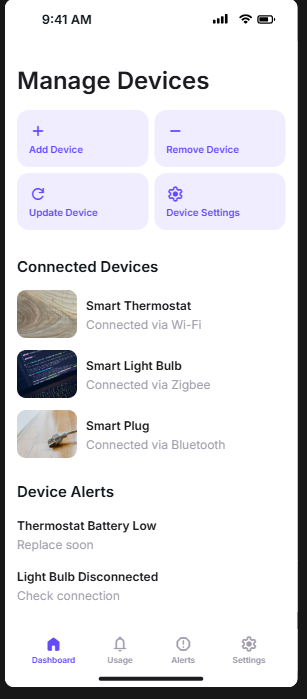
1. **User-Centric Approach**
   * Validates the system from the end-user's perspective.
   * Ensures features like energy monitoring, notifications, analytics, and integrations work as intended.
2. **Focus on Functional Requirements**
   * Tests core features such as real-time monitoring, device-wise 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.



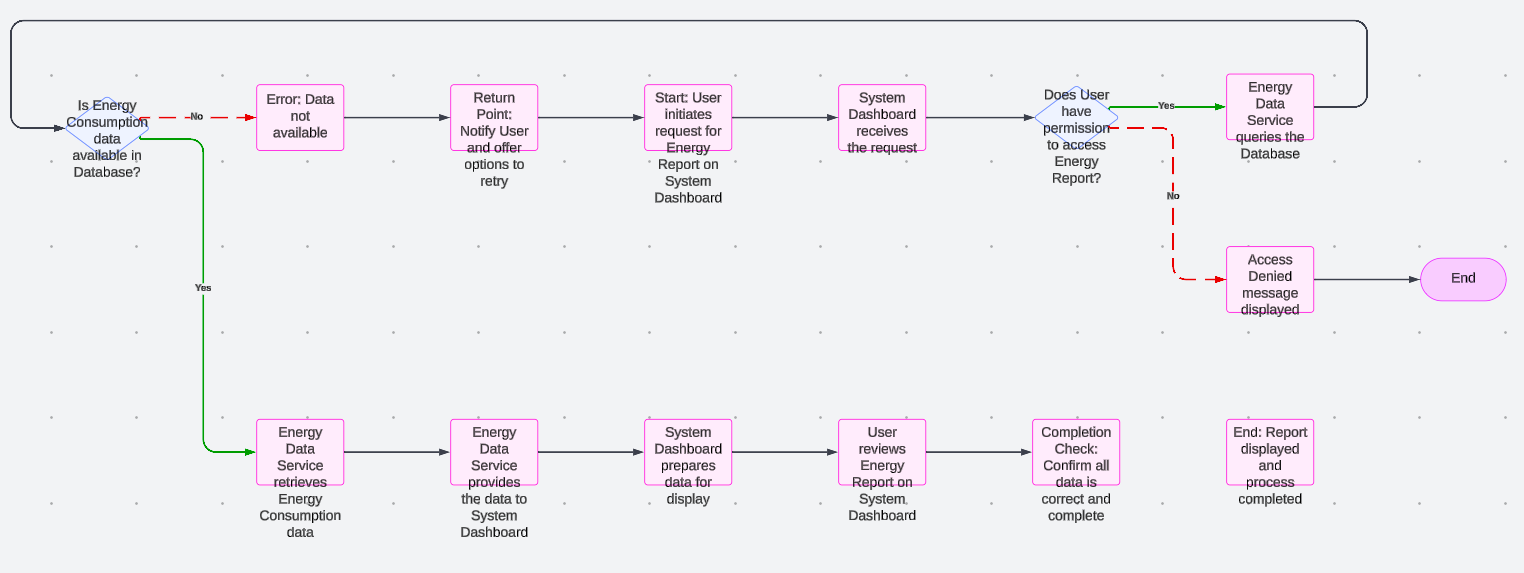
**Prototypes:**

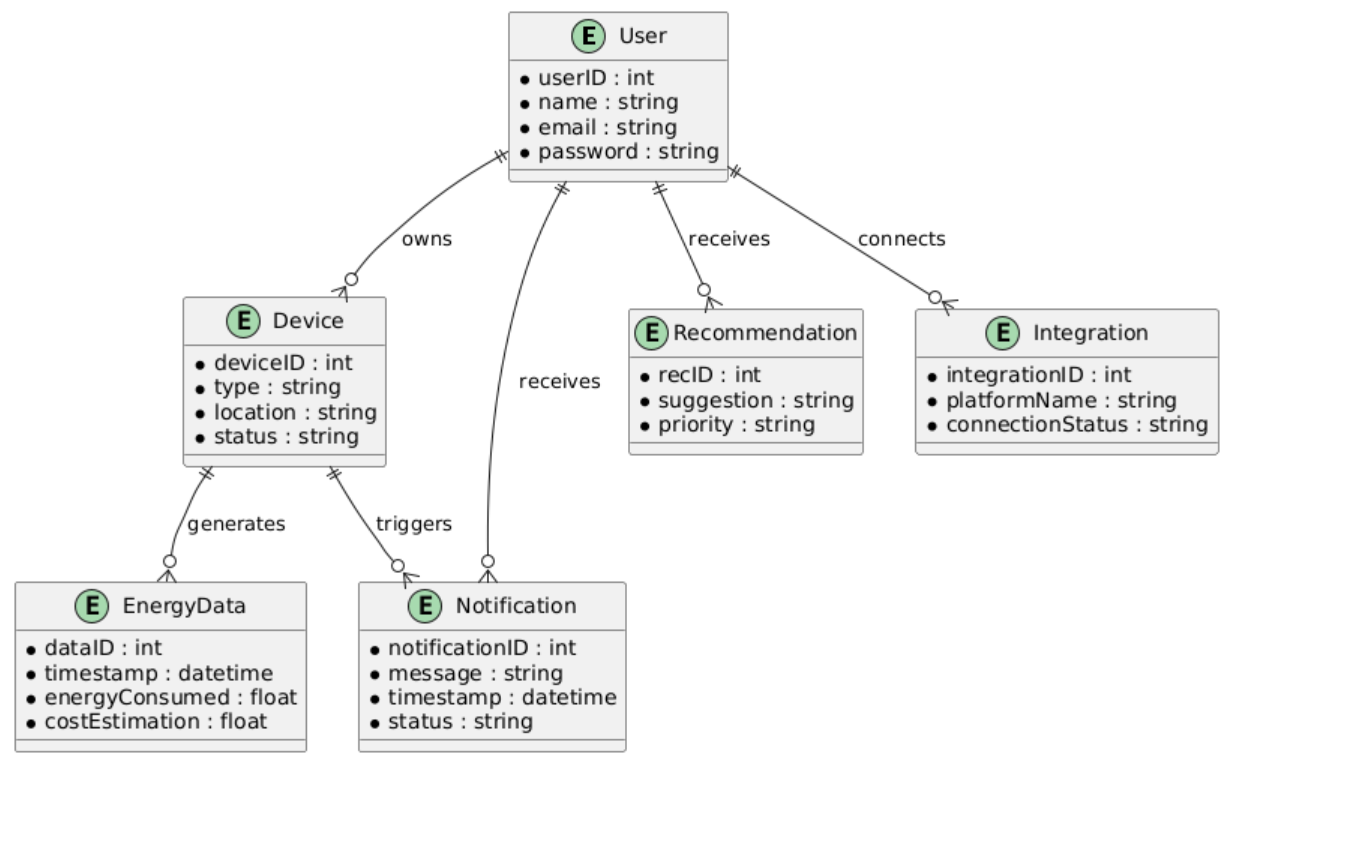
  

**  **

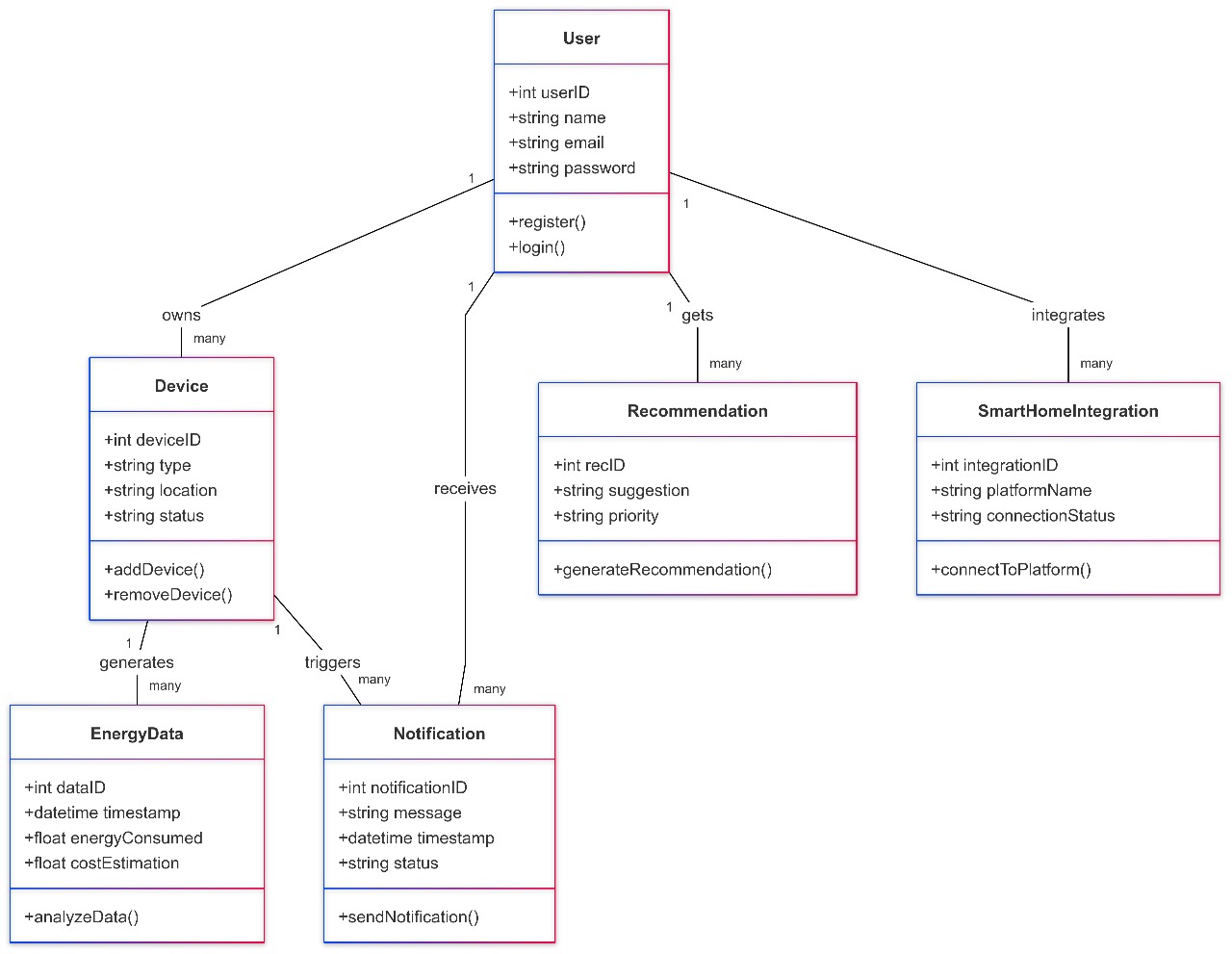
**Sequence Diagram:**

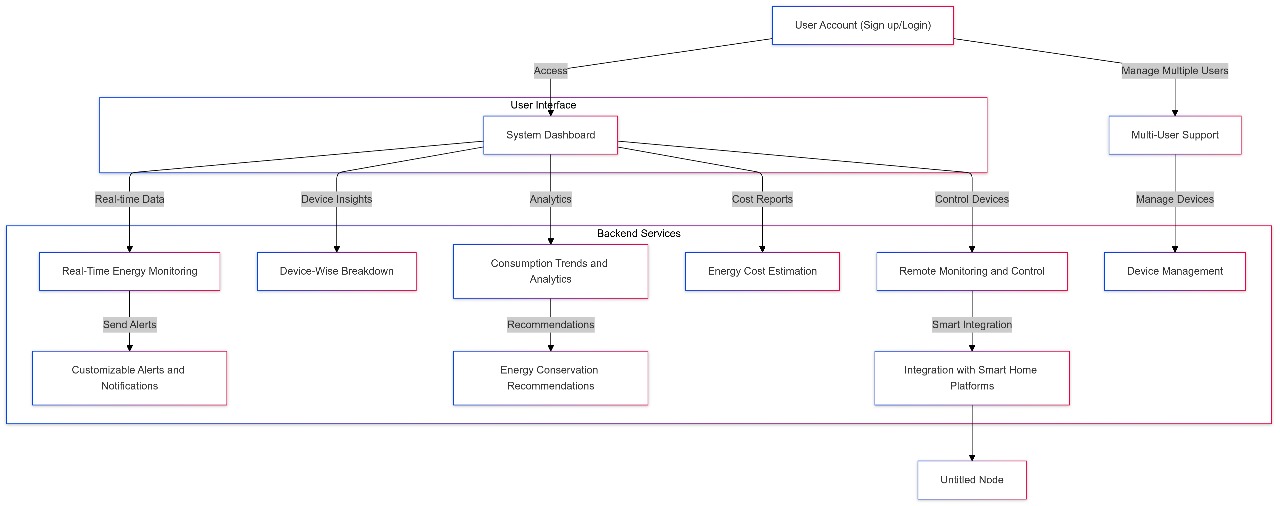
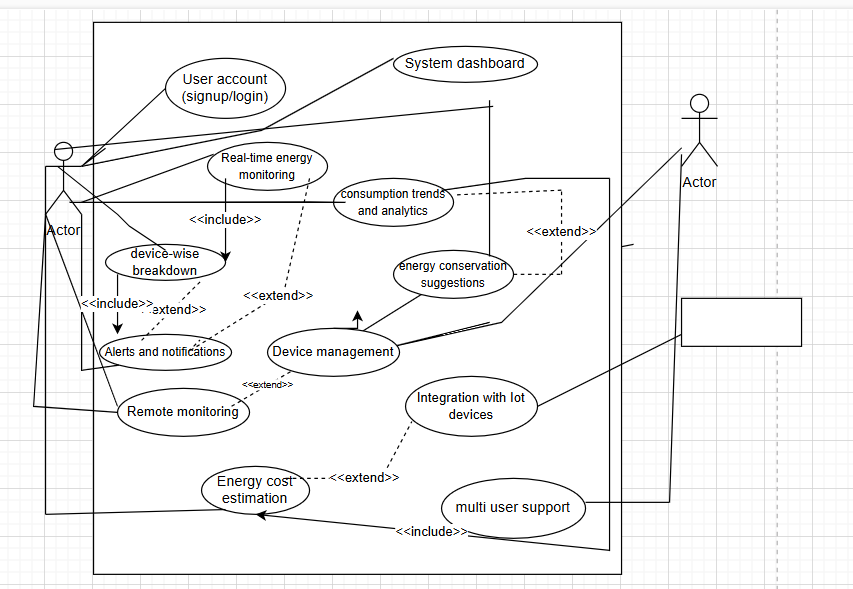


**ERD:Top of Form**

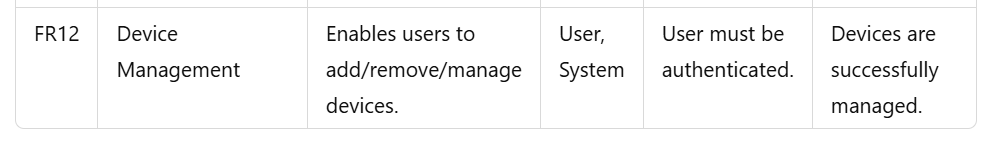
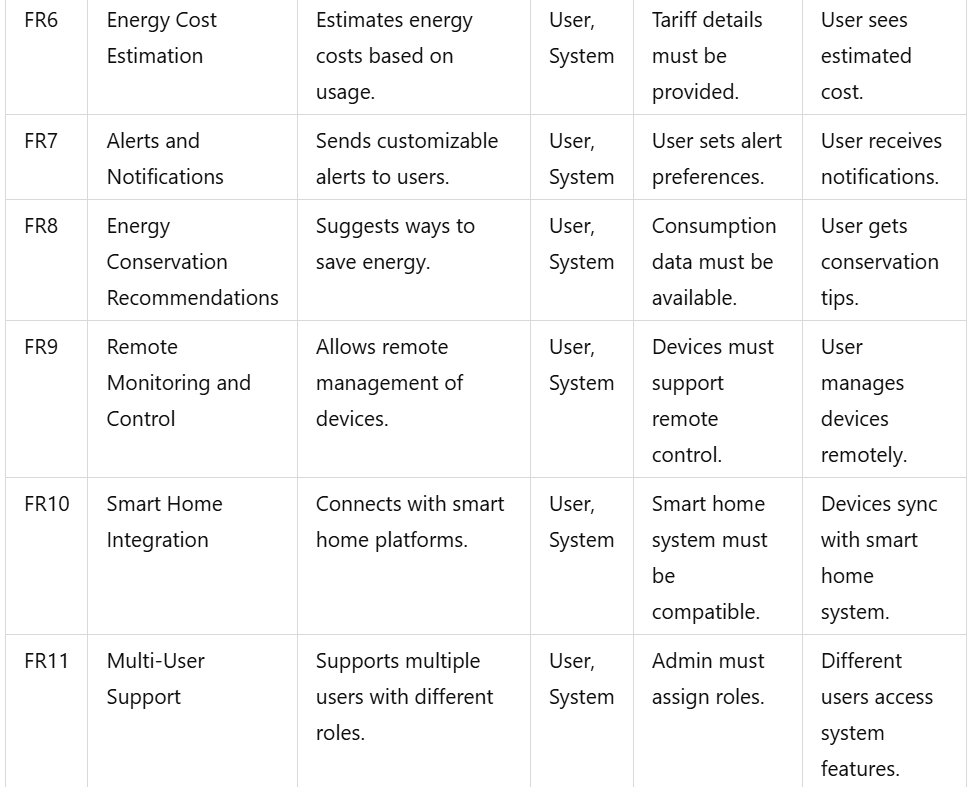
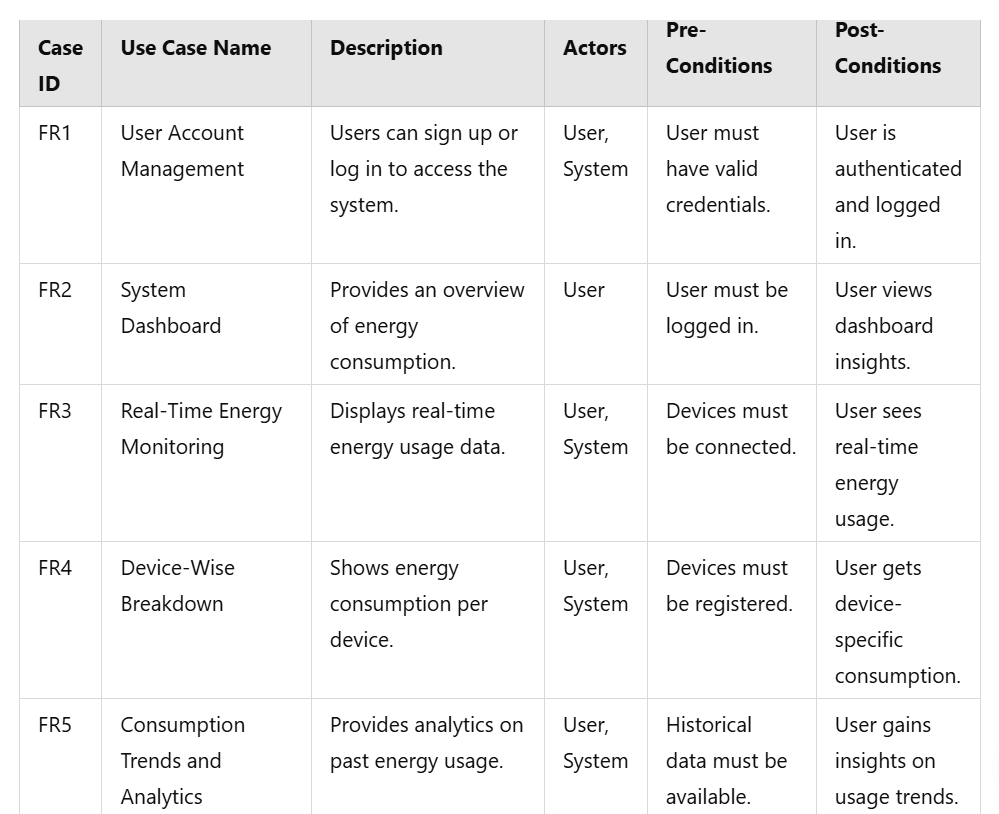


Class diagram:

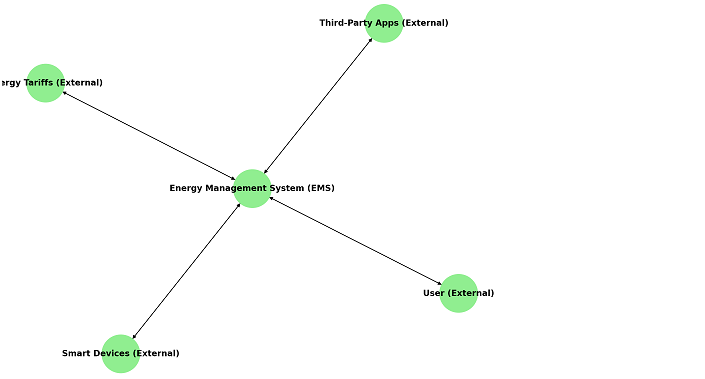
****

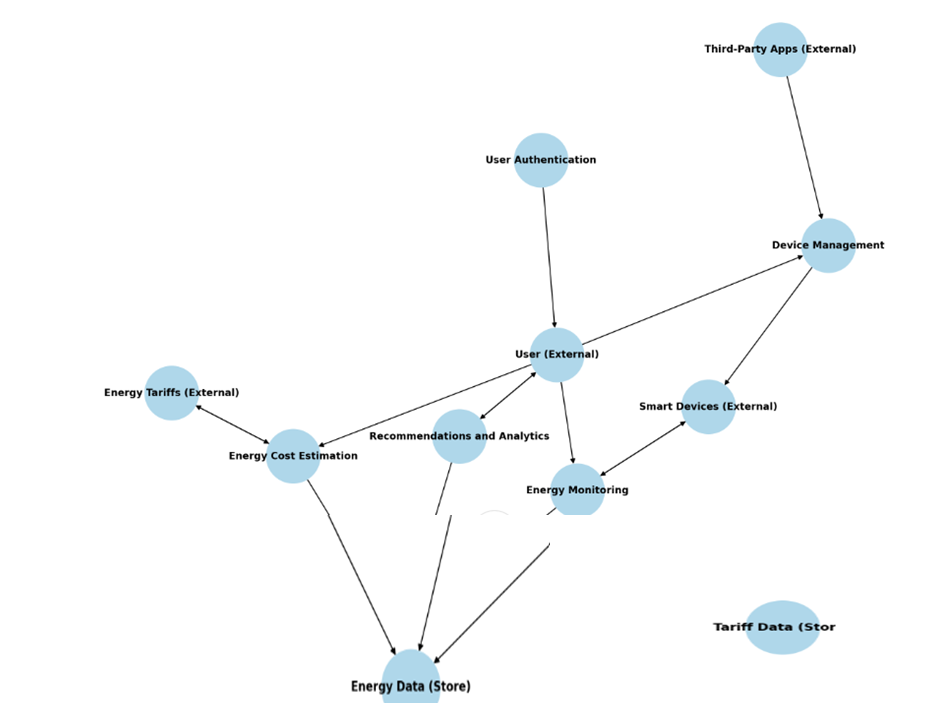
Architecture diagram:Use Case Diagram : 

Use case table:



DFD Level 0:



DFD Level 1: 

**Gantt Chart:**

