**Capstone Project Requirement Specification Document**

Project Title: Smart Home Energy Management System (SHEMS)

**User Perspective**

**Objective:** The Smart Home Energy Management System (SHEMS) is designed to help homeowners efficiently manage and reduce their energy consumption through a user-friendly interface and intelligent automation.

**User Interface**

**1. Dashboard:**

* **Overview:** Provides a real-time summary of energy consumption across different appliances and areas of the home.
* **Graphical Representation:** Displays graphs and charts showing daily, weekly, and monthly energy usage.
* **Notifications:** Alerts users about unusual energy consumption or potential savings opportunities.

**2. Control Panel:**

* **Appliance Management:** Allows users to control (turn on/off) and schedule operations of individual appliances.
* **Smart Scheduling:** Users can set routines and automate the operation of appliances based on their preferences and daily habits.

**3. Reports:**

* **Usage Reports:** Detailed reports on energy consumption by device and time period.
* **Cost Analysis:** Estimates the cost associated with the energy consumption and suggests ways to save money.

**4. Settings:**

* **User Preferences:** Allows users to configure their preferences, including notification settings and appliance control options.
* **Energy Goals:** Users can set monthly energy consumption goals and track their progress.

**Example User Scenario:**

1. **Login:** The user logs into SHEMS via a secure login page.
2. **Dashboard View:** The user is greeted with a dashboard displaying their current energy usage.
3. **Appliance Control:** The user navigates to the Control Panel to schedule their washing machine to run during off-peak hours.
4. **Set Goals:** The user sets a goal to reduce their monthly energy consumption by 10%.
5. **View Reports:** The user checks their energy usage report at the end of the month to assess their progress towards the goal.

**Designer Perspective**

**System Architecture:**

**1. Frontend:**

* **Framework:** Developed using React for a responsive and dynamic user interface.
* **Components:**
  + **Dashboard Component:** Displays real-time data and charts.
  + **Control Panel Component:** Interface for controlling and scheduling appliances.
  + **Reports Component:** Provides detailed usage reports.
  + **Settings Component:** Manages user preferences and energy goals.

**2. Backend:**

* **Framework:** Node.js for handling server-side operations.
* **APIs:**
  + **Data Retrieval API:** Fetches real-time energy usage data from IoT sensors.
  + **Control API:** Sends commands to smart appliances for control and scheduling.
  + **Notification API:** Manages alert and notification services.
* **Database:** MongoDB for storing user data, appliance status, and energy consumption records.

**3. IoT Integration:**

* **Sensors:** Smart energy meters installed on appliances to measure and transmit energy usage data.
* **Communication Protocol:** Uses MQTT for efficient and reliable data transmission between sensors and the backend server.

**4. Security:**

* **Authentication:** Implements OAuth 2.0 for secure user authentication.
* **Data Encryption:** Ensures all data in transit and at rest is encrypted using AES-256.

**Behavior Specifications:**

1. **Energy Data Collection:** Sensors measure energy consumption every minute and send data to the backend.
2. **Real-time Updates:** The frontend dashboard updates every five seconds to reflect the latest data.
3. **Appliance Control:** Users can send commands to appliances, which are executed within two seconds.
4. **Notifications:** The system sends notifications instantly when unusual consumption patterns are detected.
5. **Report Generation:** Usage reports are generated on-demand and within five seconds of the request.

**Constraints**

**1. Cost:**

* **Development:** Estimated at $50,000, including labor and hardware.
* **Maintenance:** Annual maintenance costs estimated at $5,000.

**2. Design/Implementation Time:**

* **Total Duration:** Six months, including planning, development, testing, and deployment.
* **Milestones:**
  + **Month 1:** Requirement analysis and system design.
  + **Month 2-4:** Development of frontend and backend.
  + **Month 5:** Integration and testing.
  + **Month 6:** Deployment and user training.

**3. Performance:**

* **Scalability:** The system should handle up to 10,000 concurrent users.
* **Latency:** Data updates and control commands should have a latency of no more than two seconds.
* **Reliability:** 99.9% uptime required for the backend server.

**4. Compliance:**

* **Standards:** The system must comply with local energy regulations and data protection laws (e.g., GDPR).

**5. User Constraints:**

* **Device Compatibility:** The frontend must be compatible with major browsers and mobile devices (iOS and Android).
* **Ease of Use:** The system should be intuitive, requiring minimal training for average users.