Project Documentation for Energy Management System

1. Introduction The Energy Management System (EMS) is a distributed architecture project designed to manage smart energy devices with real-time monitoring, notification, and security mechanisms. This documentation consolidates the project phases, emphasizing WebSockets, microservice architecture, security, and Docker/Traefik usage.

2. Microservices Usage The EMS is built using multiple microservices to ensure modularity and scalability:

- **User Microservice:** Manages user accounts and role-based access control.
- **Device Microservice:** Manages devices and their energy data.
- **Monitoring and Communication Microservice:** Handles energy data monitoring and WebSocket notifications.
- **Chat Microservice:** Facilitates user-admin communication using WebSockets.

Each microservice has its own database, controllers, services, and repositories, ensuring loose coupling and high cohesion. RabbitMQ serves as the message broker for inter-service communication, facilitating event-driven architecture.

3. WebSockets Implementation WebSockets are crucial for real-time updates in the EMS. They are utilized in both the Monitoring and Chat microservices:

- **Monitoring WebSockets:**
    - Real-time alert notifications for energy consumption breaches.
    - Subscribes clients to /topic/alerts for immediate feedback.
- **Chat WebSockets:**
    - Real-time messaging between users and admins.
    - Notifications for message delivery, read receipts, and typing indicators using /topic/messages and /topic/notifications/read.

WebSocketConfig and SimpMessagingTemplate classes in Spring Boot handle the configurations and message broadcasting.

4. Security Implementation Security is implemented using Spring Security and JWT tokens for authentication and authorization:

- **JWT Authentication:**
    - The AuthTokenFilter and JwtUtilsService manage token validation and parsing.
- **Role-Based Access Control:**
    - Users have USER and ADMIN roles with restricted access based on permissions.
- **Database Security:**

    - User data is stored in hashed form with secure handling of sensitive information.

- **CORS and CSRF Protection:**
    - Configured in the SecurityConfig class to allow cross-origin requests securely.

5. Docker and Traefik Usage The project uses Docker for containerization and Traefik for service routing:

- **Docker:**
    - Each microservice runs in a separate container using Docker Compose.
    - The docker-compose.yml defines services for User, Device, Monitoring, and Chat microservices along with RabbitMQ.
- **Traefik:**
    - Acts as a reverse proxy and load balancer.
    - Configured to expose backend services with URLs like user.localhost and device.localhost.
    - Ensures smooth routing and SSL termination where applicable.

6. Conclusion This documentation covers the microservices architecture, WebSockets, security strategies, and Docker/Traefik configurations used in the Energy Management System. The modular design ensures scalability, while real-time capabilities and robust security mechanisms make it reliable for energy monitoring and user interaction.