TASK 3.1P

1. **Use Cases:**

Use Case 1: Locate Nearby Charging Stations

- Actor: EV Driver
- Preconditions: The user has location services enabled.
- Postconditions: The user is navigated to the charging station.
- Main Flow:
  - When the application is launched, the user grants location access.
  - The system receives real-time data from the charging station APIs.
  - A list of nearby stations is displayed together with details about their sort, distance, and availability.
  - The user selects a station to receive directions or further information.

Use Case 2: Secure Payment for Charging

- Actor: EV Driver
- Preconditions: The user is logged in and has a valid payment method linked.
- Postconditions: The session is initiated, and payment details are logged.
- Main Flow:
  - After selecting a charging station, the user begins a charging session.
  - Once the system has connected to the payment gateway, it requests confirmation.
  - An encrypted connection is used to securely process payments.
  - Once the payment has been confirmed by the system, the charge session starts.

Use case 3: View Station Information

- Actor: EV Driver
- Preconditions: The user has selected a station from the list.
- Postconditions: The user gains insights into the station's offerings.
- Main Flow:
  - The user taps on a station's name from the list that displays.
  - Comprehensive station data, including charging rate, connector types, and cost, is retrieved and displayed by the system.

2. **User Stories:**

- As an EV driver, I hope to locate nearby charging stations on a map and to be able to swiftly charge my EV.
- As a commuter, I want to be able to check the availability of a station in real time to avoid having to wait a long time.

- As a user, I would like to pay for my charging session securely so that I don't have to carry cash.
- As a station operator, I want to make sure that the data from my station is updated on the app so that customers can access accurate information.

3. User Requirements:

Functional Requirements:

- The system must retrieve and display charging stations based on the user's journey or location.
- The program must integrate with payment gateways such as PayPal and Stripe to facilitate secure payments.
- The system must offer real-time updates on station availability and operational status.
- The app must allow users to filter stations based on charging type, price, and proximity.

Non-Functional Requirements:

- Performance: It should take two seconds for search results to show up.
- Scalability: Capable of supporting 50,000 users concurrently.
- Security: Encrypt all communications using the TLS/SSL protocols.
- Usability: The UI should follow WCAG for accessibility.

4. Design Specifications:

High-Level Architecture:

The "Locate a Socket" application is composed of:

- Frontend: Use React or Angular to create a responsive user interface.
- Backend: Node.js for API management.
- Database: MongoDB for user and station data.
- Google Maps for navigation, Stripe/PayPal for payments, and EV charging are examples of third-party APIs.

Diagram:

[User] → [Frontend] ↔ [Backend] ↔ [Database]

↔ [Google Maps API]

↔ [Payment Gateway API]

↔ [Charging Station API]

5. Connections Between Use Cases, User Stories, and Requirements:

Use Case	User Story	Requirement
Locate Nearby Charging Stations	As an EV driver, I want to locate nearby stations.	Functional Req: Display stations using GPS.
View Station Information	As a commuter, I want real-time availability updates.	Non-Functional Req: 2-second response time.
Secure Payment for Charging	As a user, I want to securely pay for sessions.	Functional Req: Integrate secure payments.

6. References:

- Google Maps API Documentation: https://developers.google.com/maps/documentation
- Stripe Payment Gateway: https://stripe.com/docs
- PCI DSS Standards: https://www.pcisecuritystandards.org/
- Web Content Accessibility Guidelines (WCAG): https://www.w3.org/WAI/