Software Requirements Specification (SRS)
For: Locate a Socket Application

1. Introduction:

Overview:

Locate a Socket is a platform that uses GPS and maps to help electric vehicle drivers find nearby charging stations and easily and securely pay for their charges.

Target Audience:

- Developers and business analyst
- Product owners and sponsors
- Software testers

2. Purpose:

The major goal of the "Locate a Socket" application is to improve the reach and ease of EV charging stations.

Primary objective:

- Provide route-based charging station recommendations.
- Provide real-time information on charging station availability.
- Ensure safe payment processing for charging services.

Intended Benefits:

- Reduced range anxiety among EV drivers.
- Increased efficiency in locating and using charging stations.

3. Audience:

- **EV Drivers:**
  Users who rely on the application to find charging stations.
- **Charging Station Operators**:
  Secondary users are responsible for managing and updating charging station data.

4. Overall Description:

The "Locate a Socket" application is an easy-to-use tool that assists drivers of electric vehicles in finding and making use of charging stations. For users who need to charge their cars while on the go, the app offers a simplified experience by fusing location-based services with safe payment options.

• Purpose and Function:

    - Using their present location or intended path, users can use the application to look for charging stations in their area.
    - It shows the current charging station availability in real-time, along with the charging point status.
    - Additional information, like the kind of charging connector, cost, and working hours, is visible to users.
    - Integrated payment gateways allow customers to safely pay for their charging sessions after choosing a station.
- **Operation:**
    - **Input**: To locate charging stations along their path, users provide their current position or destination.
    - **Results Shown**: The app provides a list of charging stations in the area along with important information like availability, station type, connector type, and cost.
    - **Navigation**: Customers may select a charging station, see where it is on a map, and get instructions for it.
    - **Payment Process**: After utilizing a station, customers may safely finish the transaction by utilizing digital wallets or credit/debit cards.

5. External Interfaces:

- **Map Services:** Integration for navigation and location searches using the Google Maps API.
- **Payment gateways:** Using services like PayPal or Stripe to handle payments securely.
- **Charging Station APIs:** These show station availability and cost based on real-time data from outside suppliers.
- **User Authentication:** Manage user accounts using a secure OAuth login.
- **Notification Services:** Using services like Twilio or Firebase to notify users of updates.

6. System Features:

- **Search for Charging Stations**:
    - Users can look for stations by entering their destination or location.
    - Station type (e.g., fast chargers), availability, and cost filters.
- **Real-Time Availability:**

    - Gives current information about the availability and occupancy of charging stations.
- **Secure Payment Transactions:**
    - Allows customers to pay for charging services using digital wallets and credit/debit cards securely.
    - After every payment, a receipt is sent.
- **Account Management for Users:**
    - For later usage, users can store payment methods, stations they regularly visit, and preferences.

7. Non-functional Requirements:

- **Performance:**
    - High traffic volumes should not cause the system to crash.
- **Security**:
    - Secure authentication should be used by the application, and data must be encrypted during payment transactions.
- **Reliability**:
    - To ensure minimal service interruption, the program should support planned maintenance and notify users in advance.
- **Usability**:
    - Both desktop and mobile devices should be able to use it.

8. Other Requirements:

- **Scalability**:
    - To handle future increases in data volume and user counts, the program should be scalable.

- **Localization**:
    - To serve users internationally, the program should support a broad variety of languages.
- **Interoperability**:
    - The system needs to function on a range of gadgets, such as tablets, smartphones running iOS, Android, and Windows, and desktop PCs.

References:

1. Software Engineering TENTH edition Ian Sommerville
2. Google Map Documentation Derived from https://developers.google.com/maps/documentation
3. Stripe API documentation. Derived from https://docs.stripe.com/api