Software Requirements Specification (SRS) for "Locate a Socket"

1. Introduction

1.1 Document Purpose

This document specifies requirements for the "Locate a Socket" web application that enables EV drivers to find, reserve, and pay for charging stations. It is a binding agreement between stakeholders and developers.

1.2 Product Scope

Locate a Socket is an online system that helps electric vehicle drivers find, reserve, and pay for charging stations. The service provides real-time station availability, trip planning with charging stops, and secure payment processing. The primary goal is to reduce range anxiety by making charging stations easily available.

1.3 Document Overview

This document lays out the software requirements for Locate a Socket, which includes:

- Section 2: Overall description of the system
- Section 3: Specific requirements
- Section 4: Supporting information

1.4 Definitions and Abbreviations

- EV: Electric Vehicle
- EVSE: Electric Vehicle Supply Equipment
- OCPP: Open Charge Point Protocol
- API: Application Programming Interface
- kWh: Kilowatt-hour

2. Overall Description

2.1 Product Perspective

Locate a Socket is a web application integrating with mapping services, payment processors, and charging station networks. The system relies on cloud-based infrastructure for location services, payment processing, and user data management.

2.2 Product Functions

Core features are:

- User sign-up and login
- Searching and filtering charging stations
- Checking real-time availability
- Route planning including charging stops
- Secure payment handling
- Station rating and reviews
- Admin panel for managing stations

2.3 User Characteristics

- EV Drivers: Main users who need to search and use charging stations. They range from tech-savvy early adopters to mass EV drivers.
- Station Operators: Entrepreneurs operating charging stations. They need functionality to alter station status and prices.
- Administrators: System administrators responsible for user accounts, payments, and content moderation.

2.4 Constraints

- Must be accessible by all major web browsers (Chrome, Firefox, Safari, Edge)
- Mobile responsiveness for tablets and smartphones
- Must be PCI-DSS compliant to process payments
- Dependent on third-party charging station APIs for real-time data

2.5 Assumptions and Dependencies

- Users have modern web browsers and access to the internet
- Charging station networks provide accurate, real-time data
- Payment processors (Stripe/PayPal) are accurate in high availability
- Google Maps API will be available for mapping purposes

3. Specific Requirements

3.1 External Interfaces

- **User Interface:** Web application with mobile-responsive design

- **Hardware Interfaces:** Compatible with mobile devices that have GPS capability
- **Software Interfaces:**
    1. Google Maps API for geolocation services
    2. Stripe/PayPal API for payment services
    3. OCPP for station communication
- **Communication Interfaces**: RESTful API for third-party applications

3.2 Functional Requirements

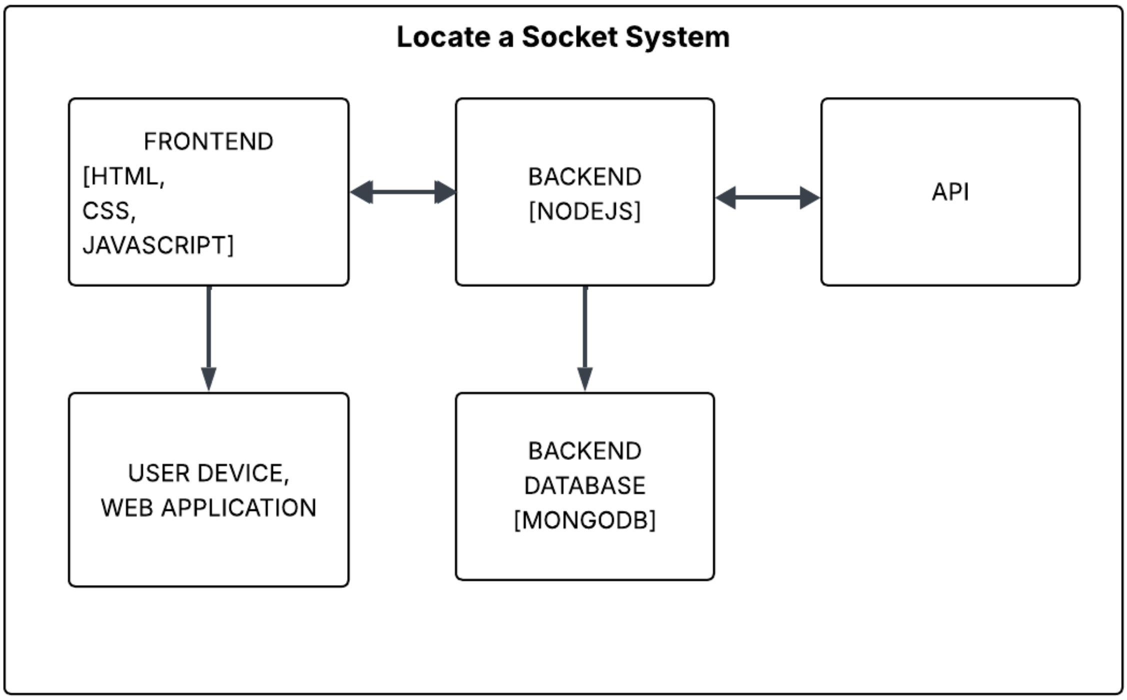
- **User Authentication:** Users can register with email or OAuth 2.0 (Google, Apple)
- **Station Search:** Users can search with location, connector type, and availability
- **Route Planning:** System recommends best routes with charging stops given the vehicle range
- **Payment Processing**: Secure credit card processing using Stripe API (PCI-DSS certified)
- **Station Management:** Operators can alter price, status, and maintenance schedules
- **Rating System:** Users can rate stations (1-5 stars) and comment

3.3 Non-Functional Requirements

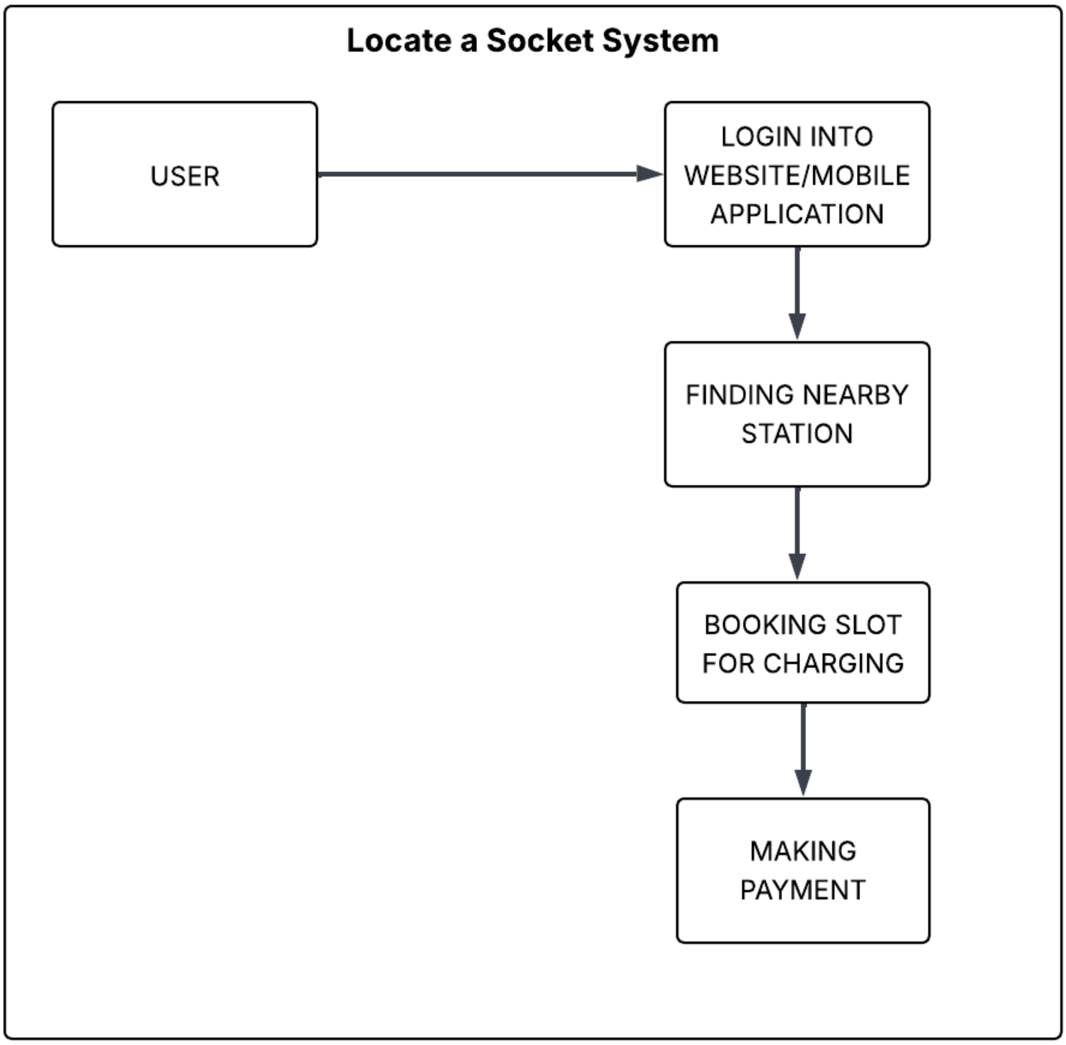
- **Performance:**
    - Search results are loaded within 2 seconds
    - Payment processing within 5 seconds
- **Security:**
    - All user data encrypted with AES-256
    - Passwords hashed and stored using bcrypt
- **Availability:** 99.5% uptime SLA
- **Usability:**
    - WCAG 2.1 AA compliant
    - Key features accessible within 3 clicks

4. Supporting Information

- **System Architecture**
    - High-level architecture diagram (web client → API server → database → external APIs)
    - Deployment architecture (AWS EC2, RDS, S3)
    - Sequence diagrams for booking flow


Locate a Socket System
FRONTEND [HTML, CSS, JAVASCRIPT]
BACKEND [NODEJS]
API
USER DEVICE, WEB APPLICATION
BACKEND DATABASE [MONGODB]

- **Use Cases**
  - Primary use cases: Find station, Reserve station, Pay for charging
  - Use case diagrams showing interactions between users, system, and external services


Locate a Socket System
USER
LOGIN INTO WEBSITE/MOBILE APPLICATION
FINDING NEARBY STATION
BOOKING SLOT FOR CHARGING
MAKING PAYMENT

- **API Documentation**
  - REST API specifications for all endpoints
  - Sample requests/responses for station search and payment processing
- **Legal Compliance**
  - PCI-DSS compliance for payments
  - GDPR compliance for EU users
  - Terms of Service and Privacy Policy
- **References**

1. IEEE Computer Society. (1998). *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications*. IEEE.https://standards.ieee.org/ieee/830/1222/
2. Google Developers. (2023). Google Maps Platform Documentation. Google LLC. https://developers.google.com/maps/documentation

3. PCI Security Standards Council. (2022). *Payment Card Industry Data Security Standard (PCI DSS) v4.0*.https://www.pcisecuritystandards.org/document_library/