SHELL STATION MANAGEMENT SYSTEM

UNEXT Software Engineering Project Report

SE-Custom Bootcamp Evaluation

Submitted by:

ADITYA NARAYAN SINGH

PRABAL GAUTAM

PRIYA AGGARWAL

NAMAN JAIN

SARTHAK SANGAL

Batch No.: 15

Submitted to: Unext

Software Engineering Department

SBO, Bangalore

September 2023

1. Introduction

1.1 Purpose of this document

The purpose of this document is to give a deep insight about the working and the attributes of the product. It highlights the major tasks that the software performs. It also mentions all the software as well as hardware requirements of the product. This document helps the clients, teams understand the product in a more lucid manner.

1.2 Overview

The upcoming sections of this document take you through the general description of the software, including the product perspective, the main features of the product, the user characteristics and the major software and hardware requirements. The section henceforth talks about the detailed functional requirements of the product coupled with the external interface requirements, logical database requirements and the quality attributes.

2. Overall Description

2.1 Product Perspective

The proposed web application aims to provide users with a convenient and comprehensive solution for locating Electric Vehicle (EV) charging stations and identifying items available in the nearest shell select stores. By integrating mapping, geolocation, and real-time data, this application enables users to make informed decisions about charging their EVs and shopping for desired products. Key functionalities include user registration, location-based searches, filters for EV charging stations and store items, user preferences, item details, and navigation. The webapp strives to offer a seamless and user-friendly experience while adhering to strict security and privacy standards. However, several challenges must be addressed to ensure the success of this project.

2.2 Product Features

- Locate nearest shell station
- Get list of available products in shell select retail associated with a station.
- Submit feedback of station
- Raise complaint/grievance for station.
- Admin flow to manage station and perform CRUD operations.

2.3 General Constraints, Assumptions and Dependencies

2.3.1 Constraints:

1. Data Availability: The availability and accuracy of data related to EV charging stations and store inventory are critical. The webapp's effectiveness depends on the quality and timeliness of this data.

2. API Integration: Will need to integrate with external APIs or data sources to gather information about EV charging stations and store inventory. Dependence on these APIs can lead to disruptions if they change or experience downtime.

3. Scalability: As the user base grows, the app's infrastructure needs to handle increased traffic. Scalability concerns may require regular server maintenance and potentially additional resources.

4. User Privacy: Ensure compliance with data privacy regulations (e.g., GDPR, CCPA) when handling user data, location information, and any personally identifiable information.

5. Cost: Developing and maintaining the app can incur costs for hosting, data providers, and API usage. Cost management is crucial to keep the project sustainable.

2.3.2 Assumptions:

1. Location Services: Users will have location services enabled on their devices to find nearby EV charging stations and stores accurately.

2. Data Accuracy: Assume that the data from external sources, such as EV station locations and store inventory, is accurate and up-to-date. Regular data validation and updates may be necessary.

3. User Preferences: Users will have preferences for the type of charging stations (e.g., fast charging level) and stores they want to search for. These preferences should be customizable.

4. Network Connectivity: Users will have a stable internet connection for real-time data retrieval from external sources.

5. User Registration: Users may need to register or log in to access certain features, such as saving favorite locations or past searches.

2.3.3 Dependencies:

1. Mapping and Geolocation Services: The webapp will depend on mapping and geolocation services (e.g., Google Maps API) to display EV charging stations and stores on a map.

2. EV Charging Station Data Providers: Rely on data providers or APIs that offer information about EV charging station locations, availability, and types.

3. Store Inventory Data Providers: Integrate with data sources that provide store inventory data, including product availability, pricing, and location.

4. User Authentication Services: As user registration and login is required, it will depend on authentication services to manage user accounts securely.

6. Server Infrastructure: Depends on server infrastructure to host and run the web app, including databases for storing user data and preferences.

7. Front-end Frameworks and Libraries: Will use front-end technologies like HTML, CSS, JavaScript, and possibly front-end frameworks (e.g., React, Angular, Vue.js) for building the user interface.

8. Back-end Frameworks and Languages: Need to select a back-end technology stack (e.g., Node.js, Ruby on Rails, Django) for handling server-side logic and database interactions.

9. Hosting and Cloud Services: Need to Choose a hosting platform (e.g., AWS, Azure, Heroku) to deploy and manage the app in a scalable and secure manner.

10. Data Privacy Compliance Tools: If handling user data,need to incorporate tools and practices for ensuring compliance with data privacy regulations.

Certainly, here are both functional and non-functional requirements for the web app that allows users to search for EV charging stations and find items available in the nearest stores

2.4 Detailed Description of Functional Dependencies

2.4.1 Functional Requirements:

1. User Registration and Authentication:

- Users must be able to register for an account.

- Registered users can log in securely.

- Registered users can recover their passwords if forgotten.

2. Location Services:

- The webapp should access the user's location to provide accurate search results.

- Users can manually input their location if they prefer not to use GPS.

3. EV Charging Station Search:

- Users can search for nearby EV charging stations.

- Users can filter charging stations based on criteria such as charging speed, availability, and compatibility with their EV model.

- The webapp displays a list of nearby charging stations on a map.

4. Store Item Search:

- Users can search for items available in nearby shell select stores.

- Users can filter store items by category, price range, and brand.

- The app displays a list of nearby shell stores offering the searched items on a map.

5. Item Details and Reviews:

- Users can view detailed information about store items, including prices, descriptions, and user reviews.

- Users can rate and write reviews for ev stations they have visited.

7. Directions and Navigation:

- The app provides directions from the user's current location to selected charging stations and stores.

- Users can choose between walking, driving, or public transportation directions.

2.4.2 Non-Functional Requirements:

1. Performance:

- The webapp should load quickly and respond to user actions promptly.

- It must handle concurrent user requests efficiently, especially during peak usage times.

2. Security:

- User data, including personal information and payment details, must be securely stored and transmitted using encryption.

- Implement user authentication and authorization to protect user accounts.

3. Reliability:

- The webapp should have high availability and minimal downtime.

- Regular backups of user data and app configurations should be performed.

4. Scalability:

- The app must be able to scale horizontally to accommodate a growing user base and increasing data volumes.

5. Usability and User Experience:

- The user interface should be intuitive, user-friendly, and accessible.

- Ensure that the app is responsive and works smoothly on various devices and screen sizes.

6. Compliance:

- Ensure compliance with data privacy regulations (e.g., GDPR, CCPA) and adhere to industry-specific standards and guidelines.

7. Data Accuracy and Integrity:

- Regularly update and verify the data from external sources, such as EV charging station locations and store inventory, to maintain accuracy.

8. Cost-Efficiency:

- Optimize resource usage and minimize unnecessary costs associated with server hosting, data providers, and API usage.

9. Cross-Browser Compatibility:

- The app should work consistently across different web browsers (e.g., Chrome, Firefox, Safari, Edge).

11. Support and Maintenance:

- Provide ongoing support and maintenance to address bugs, add new features, and ensure the webapp remains up to date with the latest technologies.

2.5 Challenges

1. Data Integration: One of the primary challenges is integrating and synchronizing data from various external sources, such as EV charging station databases and store inventories. Ensuring the accuracy and real-time availability of this data is crucial for delivering reliable search results.

2. Data Privacy and Security: Handling user data, including location information and payment details, requires robust security measures. Ensuring data privacy and protection against potential breaches is paramount, especially in compliance with data privacy regulations.

3. API Reliability: The application heavily relies on external APIs for mapping, geolocation, and data retrieval. Ensuring the continuous availability and reliability of these APIs is essential for uninterrupted service.

4. Scalability: As the user base grows, the webapp must scale gracefully to accommodate increased traffic and data volumes. Scalability planning is necessary to avoid performance bottlenecks.

5. User Experience: Balancing feature-rich functionality with an intuitive and user-friendly interface is challenging. The app must provide a seamless and enjoyable user experience across different devices and platforms.

6. Data Accuracy: Maintaining the accuracy and freshness of data, especially concerning charging station availability and store inventory, requires constant monitoring and validation processes.

7. Regulatory Compliance: Adhering to data privacy regulations, as well as local and industry-specific standards, adds complexity to the development process. Ensuring that the app complies with these regulations is non-negotiable.

8. Cost Management: Optimizing costs associated with hosting, data providers, and API usage while delivering quality service can be challenging. Efficient resource utilization is critical for sustainability.

9. Cross-Browser Compatibility: Ensuring that the app functions consistently and without issues across various web browsers can be technically demanding due to differences in browser capabilities and standards.

11. Support and Maintenance: Providing ongoing support, updates, and maintenance to address bugs, improve performance, and add new features is a long-term commitment that requires resource allocation.

In summary, developing a web application for locating EV charging stations and nearby store items offers significant benefits to users. However, addressing the outlined challenges is crucial to ensuring the app's success and delivering a reliable and user-friendly solution. Thorough planning, attention to detail, and a commitment to data accuracy, security, and user experience are essential to overcoming these challenges.

Implementing an Agile model for the development of a web application that helps users locate EV charging stations and nearby store items involves breaking the project into iterative phases. Here are the Agile phases for this project:

3. Specific Requirements

3.1 External Interface Requirements

Hardware Interfaces:

The system shall run on any smartphone or personal computer having a web browser and stable internet connection.

Software Interfaces:

The system shall interface with the Access database.

Communications Interfaces:

Web browsers are the interfaces to communicate with the application. Supported browsers:

1. Google Chrome

2. Firefox

3. Internet Explorer

3.2 Performance Requirements

1. The application designed is a web application which can be accessed using a web browser on both smartphones and personal computers. Users should have a stable internet connection.

2. The application should be able to scroll down instead of scrolling side-ways. 3. The application must be light and quick to load as well as use enough memory of both RAM and storage.

4. Multiple users should be able to access their respective accounts simultaneously i.e., the server should be able to process multiple services simultaneously.

3.3 Quality Attributes

This part highlights the quality attributes of the application such as accuracy, performance, availability and security.

1. The application must allow authentication of users who have existing accounts for day to day use of the application.

2. The application must be in position to provide security for user data stored in the database. 3. The application must assign user rights to its respective users such as a laundry employee and students have different user rights even though they all login through the same interface.

4. The application must be accessible at all times allowing users to update their information and place orders at any given time.

5. The application must be able to respond to the changes made by users in a reasonable time.

4. Analysis Phase

4.1 Development Plan

4.1.1 Phase 1: Project Initiation (Sprint 0)

Sprint Goal: Define project scope, objectives, and initial backlog.

Activities:

- Stakeholder identification and engagement.
- High-level requirements gathering.
- Create a project vision and initial product backlog.
- Establish the project team and roles.
- Set up development environment and tools.

Deliverables:

- Project charter.
- High-level requirements.
- Initial product backlog.
- Team assignments.
- Development environment configured.

4.1.2 Phase 2: MVP Development (Multiple Sprints)

Sprint Goal: Build a Minimum Viable Product (MVP) with core features.

Activities:

- Prioritize and refine items in the product backlog.
- Develop and test core functionalities.
- Conduct regular sprint planning, daily stand-up meetings, and sprint reviews.
- Iteratively release increments of the MVP.

Deliverables:

- Incremental releases of the MVP.
- User stories, code, and documentation.
- User feedback and sprint review reports.

4.1.3 Phase 3: Feature Enhancements (Multiple Sprints)

Sprint Goal: Add additional features and improve existing ones.

Activities:

- Continuously refine and prioritize the product backlog.
- Develop and test new features based on user feedback.
- Address any issues or bugs identified in the MVP.
- Monitor and respond to user feedback and evolving requirements.

Deliverables:

- Incremental feature additions and improvements.
- User stories, code, and documentation.
- User feedback and sprint review reports.

4.1.4 Phase 4: WebApp Optimization (Multiple Sprints)

Sprint Goal: Ensure webapp responsiveness and optimize the user experience on various devices.

Activities:

- Implement responsive design changes.
- Iteratively improve webapp user experience.
- Test and validate changes with users.

Deliverables:

- Responsive web application.
- User feedback and validation reports.

4.1.5 Phase 5: Security and Compliance (Multiple Sprints)

Sprint Goal: Enhance security, privacy, and regulatory compliance.

Activities:

- Identify and address security vulnerabilities.
- Implement encryption and secure data handling.
- Ensure compliance with data privacy regulations.
- Perform regular security audits and testing.

Deliverables:

- Secure and compliant web application.
- Security audit reports.

4.1.6 Phase 6: Scaling and Performance (Multiple Sprints)

Sprint Goal: Optimize for scalability and performance.

Activities:

- Implement scaling solutions for increased user traffic.
- Optimize database and server performance.
- Conduct load testing and performance tuning.

Deliverables:

- Scalable and high-performance web application.

4.1.7 Phase 7: Ongoing Maintenance and Support (Continuous)

Sprint Goal: Provide ongoing support, bug fixes, and updates.

Activities:

- Address user-reported issues and bugs.

- Implement minor feature enhancements.

- Ensure server maintenance and backups.

Deliverables:

- Regular bug fixes and updates.

- Documentation for support and maintenance.

4.1.8 Phase 8: Continuous Improvement (Continuous)

Sprint Goal: Continuously improve the application based on user feedback and emerging needs.

Activities:

- Gather and analyze user feedback.

- Prioritize and plan improvements.

- Iteratively implement changes.

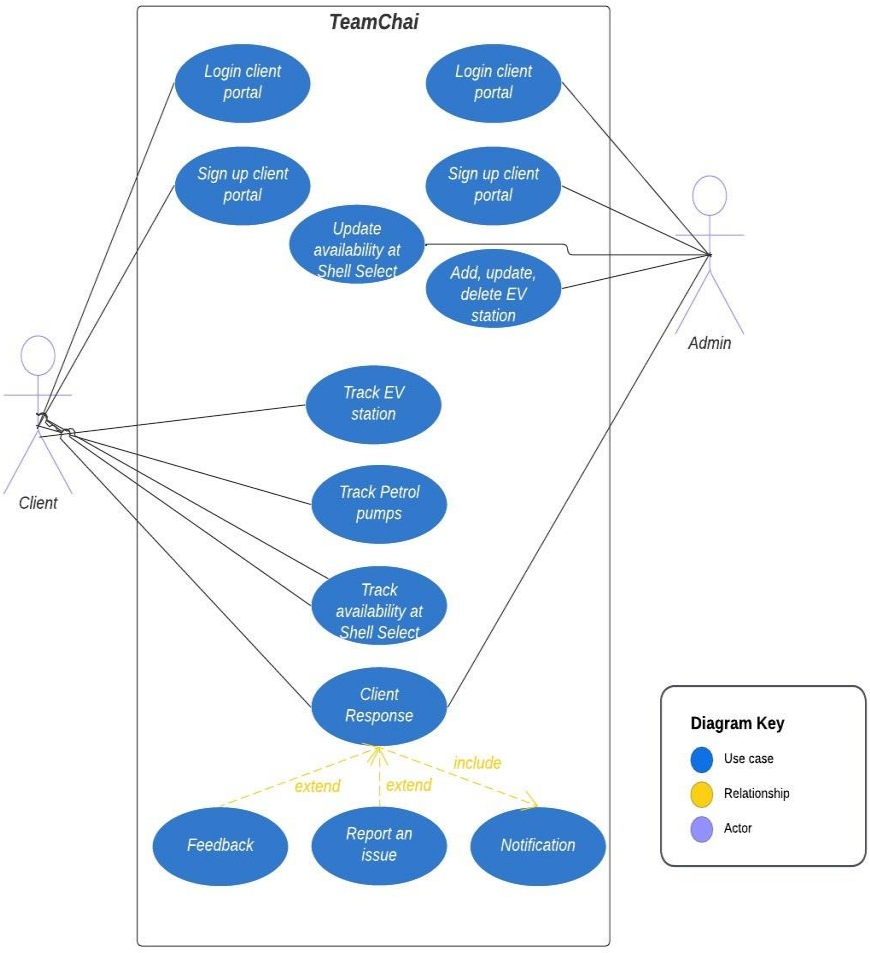
Deliverables:

- Incremental updates and improvements based on user feedback.

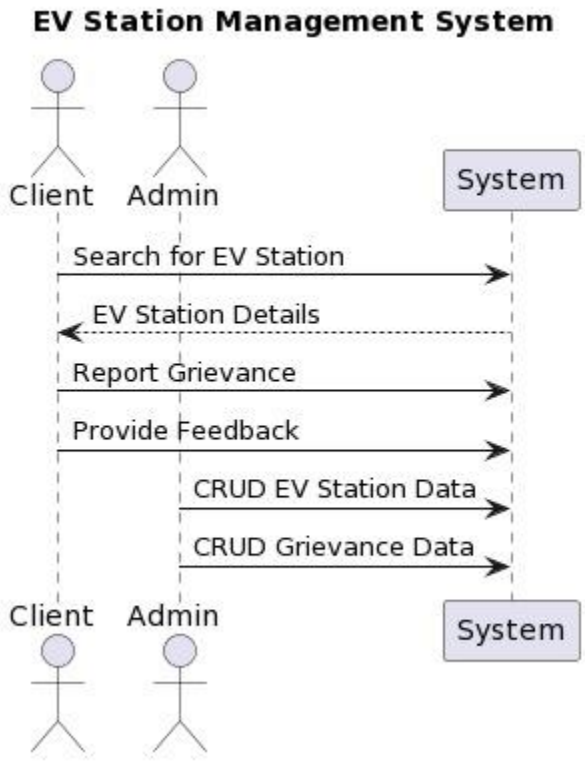
In an Agile model, each phase is a series of sprints, with each sprint typically lasting 2-4 weeks. The project team continually collaborates with stakeholders and adapts to changing requirements throughout the development process. This iterative approach allows for flexibility and ensures that the application evolves to meet user needs effectively.

5. Design Phase

5.1 Use case Diagram



5.2 Sequence Diagram



5.3 ER Diagram