ÇANKAYA UNIVERSITY

Software Design Description

ChargeMind: AI Based E-charge-Management and Planning Application

Lorin Melek Vural - 202011035, Selin Deniz - 202011051, Hanife Müge Karakaya - 202011044, Meleksu Özdoğan - 202011054, Kaan Baydemir – 202011070

27/12/2024

Table of Contents

List of Figures

1. Introduction

1.1 Scope

The ChargeMind technology is intended to address one of the most serious problems for electric vehicle (EV) owners: battery depletion during long trips, or more generally, “range anxiety”. ChargeMind ensures that EV drivers and fleet owners may locate appropriate paths that match their energy and trip requirements by providing tailored and optimal route planning.

1.2 Purpose

The purpose of this document is to provide detailed descriptions of design concerns and design decisions, recording design information and explanations for the stakeholders. This SDD will further elaborate and explain the functional and non-functional requirements mentioned in the Software Requirement Specification document.

1.3 Glossary

Term	Definition
Activity Diagram	A flowchart showing how one activity leads to another activity.
Artificial Intelligence	Technology enabling machines or computers to mimic human-like intelligence and learning capabilities.
Backend	The part of a computer system or application that is not directly accessed by the user, typically responsible for storing and manipulating data.
Class Diagram	A graphical notation for building and visualizing object-oriented systems.
Data Flow Diagram	It is a way of representing a flow of data through a process or a system
EVRP	The EVRP is an optimization problem that focuses on determining the most efficient routes for electric vehicles while considering constraints like battery capacity, charging station availability, and energy consumption.
Fleet Owner	It refers to an individual or company that owns and operates a fleet of vehicles for various purposes.
Frontend	Relating to or denoting the part of a computer system or application with which the user interacts directly.
Hardware	The machines, wiring, and other physical components of a computer or other electronic system.
Hashing	The practice of transforming a given key or string of characters into another value for the

	purpose of security like one-way encryption [1].
Java	Java is a multi-platform, object-oriented, and network-centric programming language and computing platform for coding web applications [2].
JPype	JPype is a Python module to provide full access to Java from within Python [3].
Machine Learning	The ability of computer systems to learn from data and make predictions.
NoSQL	NoSQL is a type of database management system that is designed to handle and store large volumes of unstructured and semi-structured data [4].
OAuth 2.0	One established protocol for authorization procedures is OAuth.
Py4J	Py4J enables Python programs running in a Python interpreter to dynamically access Java objects in a Java Virtual Machine [5].
PyTorch	PyTorch is a machine learning library based on the Torch library, used for applications such as computer vision and natural language processing [6].
Range Anxiety	Worry on the part of a person driving an electric car that the battery will run out of power before the destination, or a suitable charging point is reached.
React Native	React Native is a popular JavaScript-based mobile app framework that allows you to build natively rendered mobile apps for iOS and Android [7].
REST API	An application programming interface that accesses and utilizes data via HTTP requests is known as a RESTful API. The GET, PUT, POST, and DELETE data types that stand for reading, updating, creating, and removing resource related operations can be utilized with that data [8].
SDD	A description of software created to facilitate analysis, planning, implementation, and decision-making.
Sequence Diagram	Shows process interactions in software engineering that are time-ordered.
Spring Boot	Java Spring Boot is an open-source tool that makes it easier to use Java-based frameworks to create microservices and web apps [9].
SSL	SSL is an encryption-based Internet security protocol for the purpose of ensuring privacy,

	authentication, and data integrity in Internet communications [10].
Stakeholder	Denoting a type of organization or system in which all the members or participants are seen as having an interest in its success.
Tensorflow	TensorFlow is a leading open-source library designed for developing and deploying state-of-the-art machine learning applications using data flow graphs [11].
TLS	TLS is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet [12].
User Interface Design	Techniques used by designers to construct interfaces in software or technological devices, focusing on appearances or style.

1.4 Definitions, Acronyms, and Abbreviations

- **AI:** Artificial Intelligence
- **EV:** Electric Vehicle
- **EVRP:** Electric Vehicle Routing Problem
- **GPS:** Global Positioning System
- **IEEE:** Institute of Electrical and Electronics Engineers
- **iOS:** iPhone Operating System
- **JSON:** JavaScript Object Notation
- **LTE:** Long-Term Evolution
- **OAuth:** Open Authorization
- **RAM:** Random Access Memory
- **REST API:** Representational State Transfer Application Programming Interface
- **SDD:** Software Design Description/Specification
- **SQL:** Structured Query Language
- **SSL:** Secure Sockets Layer
- **TLS:** Transport Layer Security
- **UI:** User Interface
- **UML:** Unified Modelling Language
- **UX:** User Experience

1.5 Intended Audience

The audience (i.e. stakeholders) of this document is intended to be technical and management staff of the ChargeMind project. These stakeholders include software developers, UI/UX designers, system architects, programmers, maintainers, testers, project managers, and product owners.

1.6 Conformance

The document conforms to standard naming conventions, terminology, and formatting guidelines specified by IEEE Software Design Description document [13] standards for clarity and consistency.

1.7 Overview of the Document

In this Software Design Document (SDD), the technical and architectural layout of the ChargeMind application, an AI-based e-charge route planning system, is described. It provides a general guide for stakeholders, developers, and maintainers by detailing the system's overall structure, components by UML diagrams, activity diagrams, class diagrams, sequence diagrams, database schema, and user interface design. The document also addresses key aspects such as security, fault tolerance, and scalability to ensure a robust and reliable solution. Additionally, it includes design considerations for system testing and validation, ensuring that the final implementation meets performance and usability expectations. By presenting a clear and structured overview of the system's design, this document serves as a foundational reference throughout the development lifecycle.

1.8 Motivation

The motivation of the ChargeMind project is to provide users with a more comfortable charging experience with the increasing use of electric vehicles today. The need for smarter charging solutions is also becoming increasingly important with the use of electric vehicles. Current charging applications may be inadequate in areas such as route planning, accessibility, and user interface design. This project aims to address these shortcomings with real-time data integration, advanced route planning algorithms, and user-friendly features. It also aims to provide a more environmentally friendly approach with carbon footprint tracking and real-time emission estimation. The goal of the project is to support sustainable transportation by not only providing users with a practical solution but also making them aware of their environmental impact.

1.9 References

[1] "What is hashing?," *Built In.* Available: https://builtin.com/articles/what-is-hashing#:~:text=What%20Is%20Hashing%3F-,Hashing%20is%20the%20practice%20of%20transforming%20a%20given%20key%20or,are%20very%20difficult%20to%20decode. (Accessed Dec. 23, 2024).

[2] "What is Java?," *Amazon Web Services (AWS)*. Available: https://aws.amazon.com/what-is/java/?nc1=h_ls. (Accessed Dec. 24, 2024).

[3] “JPype,” *PyPI*. Available: https://pypi.org/project/jpype1/#:~:text=JPype%20is%20a%20Python%20module,scientific%20computing%2C%20and%20much%20more. (Accessed Dec. 21, 2024).

[4] “Introduction to NoSQL,” *GeeksforGeeks*. Available: https://www.geeksforgeeks.org/introduction-to-nosql/. (Accessed Dec. 22, 2024).

[5] “Py4J,” *Py4J*. Available: https://www.py4j.org/. (Accessed Dec. 23, 2024).

[6] “PyTorch,” *Wikipedia*. Available: https://en.wikipedia.org/wiki/PyTorch. (Accessed Dec. 24, 2024).

[7] “React Native,” *Netguru*. Available: https://www.netguru.com/glossary/react-native. (Accessed Dec. 23, 2024).

[8] “RESTful API,” *TechTarget*. Available: https://www.techtarget.com/searchapparchitecture/definition/RESTful-API. (Accessed Dec. 25, 2024).

[9] “What is Java Spring Boot?,” *Microsoft Azure*. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-java-spring-boot#:~:text=Java%20Spring%20Boot%20is%20an,computing%20platforms%20for%20app%20development. (Accessed Dec. 24, 2024).

[10] “What is SSL?,” *Cloudflare*. Available: https://www.cloudflare.com/learning/ssl/what-is-ssl/. (Accessed Dec. 23, 2024).

[11] “What is TensorFlow?,” *NVIDIA*. Available: https://www.nvidia.com/en-us/glossary/tensorflow/. (Accessed Dec. 25, 2024).

[12] “Transport Layer Security (TLS),” *Cloudflare*.Available: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/. (Accessed Dec. 27, 2024).

2. System Overview

The ChargeMind project is built on top of various core components to fit the needs of electric vehicle drivers. These core components of the application are database, backend, frontend, machine learning model. React Native is preferred for the front-end part to make it platform independent (i.e., both works on iOS and Android). Users of the application are offered miscellaneous such as basic route planning, advanced filtering for different concerns, and real-time status retrieval from charging stations. Each user can change preferences on the app, interact with the map, visualize paths, leading to a customized experience for each user.

On the backend, Java and Spring Boot are desired to be used. The design behind these technologies is to handle critical parts of the app such as process of data, optimization of routes, and execution of advanced machine learning models. The backend also plays the role of a bridge between third party services such as Google Maps, and various charge station service providers. This design leads the system to stay up to date with the rest of the world. For the storage purposes, a combinational usage of SQL and NoSQL is preferred. While keeping, for example, user credentials in a structured manner in SQL databases, the system will adapt to fresh data with a much more flexible method with the help of NoSQL. This approach shall make the system future-proof, flexible, and scalable.

Artificial Intelligence has a crucial role in the ChargeMind system. It is the backbone of the route planning feature, essential for effectively finding available stations while optimizing the route for keeping the user preferences in mind. This is achieved via not only traditional filtering methodologies but also machine learning models that the ChargeMind system is going to use. Tensorflow and/or PyTorch will be the preferred libraries for the development of the model. The machine learning model that will be developed shall find the shortest route for a user while pinpointing potentially available charging stations along the route. The Java backend shall communicate with the machine learning model via inter-communication solutions such as Py4J or JPype.

To meet its security requirements, ChargeMind uses industry-standard protocols like OAuth 2.0 for secure authentication, ensuring strong, token-based verification for both users and third-party services. ChargeMind also uses TLS/SSL encryption for its REST API to protect server-client communications, using secure cipher suites such as AES-256 and RSA with 2048-bit keys. User credentials are kept in hashed form using the bcrypt method, which includes a salt mechanism for enhanced security and forward secrecy, prohibiting unauthorized access even if the database credentials are compromised. Moreover, ChargeMind protects sensitive user and operational data by ensuring its confidentiality, integrity, and availability via extensive encryption. Data is secured at rest with AES-256, and encryption keys are handled via a Key Management System (KMS) to ensure regular key rotation. ChargeMind also implements Multi-Factor Authentication (MFA) and enforces strong password standards to improve user access security. Role-Based Access Control (RBAC) restricts access to important system functions while ensuring that users and services follow their assigned permissions. Overall, ChargeMind effectively mitigates security risks while maintaining system integrity and availability by continuously implementing advanced cryptographic techniques, adhering to best practices in data security, and incorporating regular security testing.

Finally, the mobile application is highly optimized for state-of-the-art smart phones. For network connections, a smartphone should have 4G LTE or higher cellular protocol

implemented in it, and a high precision GPS module is expected for determining drivers' location. The hardware prerequisites for these end-user devices are 8 GB of RAM, 2.5-3 GHz processor, and 64 GB of non-volatile storage.

3. System Design

This section contains the architectural design of the system, the problem description, the user interface design, and the hardware design. Additionally, it includes certain diagrams, such as Sequence Diagram, Activity Diagram, Data Flow Diagram, and Class Diagram.

3.1 Problem Description

ChargeMind aims to provide better charging experience for electric vehicle users by planning personalized routes, considering factors such as city, location, region, destinations to be visited before reaching the destination, cost optimization, possible duration of charging for specified range, and the selection of environmentally friendly or top-ranked stations. These factors are supported by the carbon emission tracking and station ranking features. Through the implementation of these features, it seeks to promote increased electric vehicle usage by delivering eco-friendly and autonomous solutions for problems like EVRP (Electric Vehicle Routing Problem), and Range Anxiety in the e-charging domain.

3.2 Architectural Design

To provide better comprehension of our architectural design, we explained the problem, technologies used and included numerous diagrams. As previously indicated, our system will perform three key roles. As a result, we will develop architecture that incorporates these responsibilities. Figure illustrates the fundamental structure of the ChargeMind System.

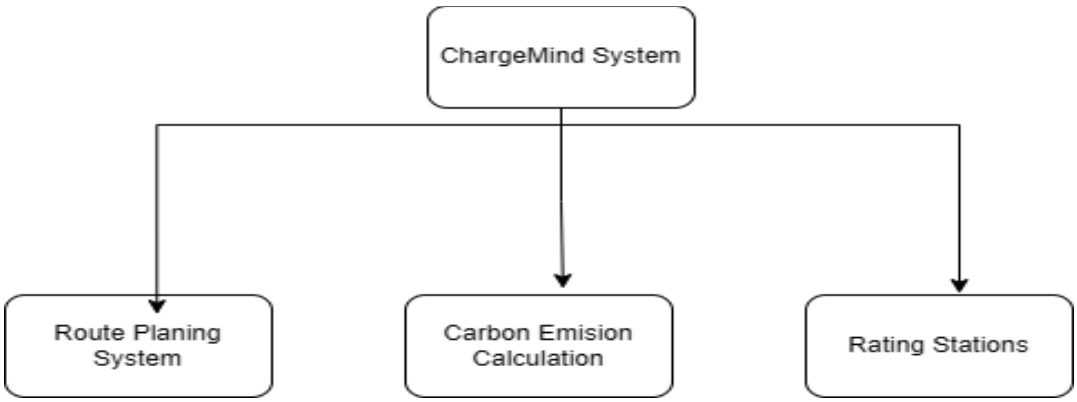

Figure 1: Brief of System Architectural Design

3.2.1 Data Flow Diagram

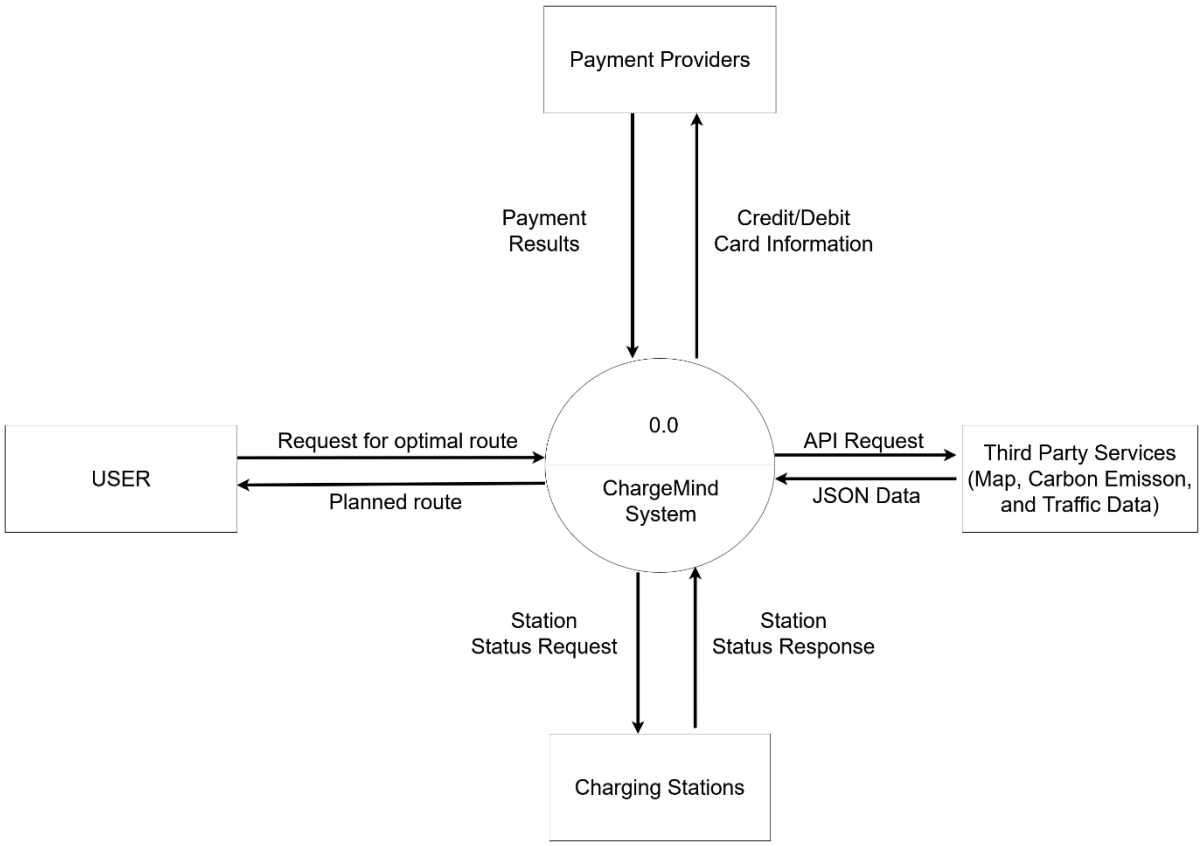

Figure 2: Context Diagram

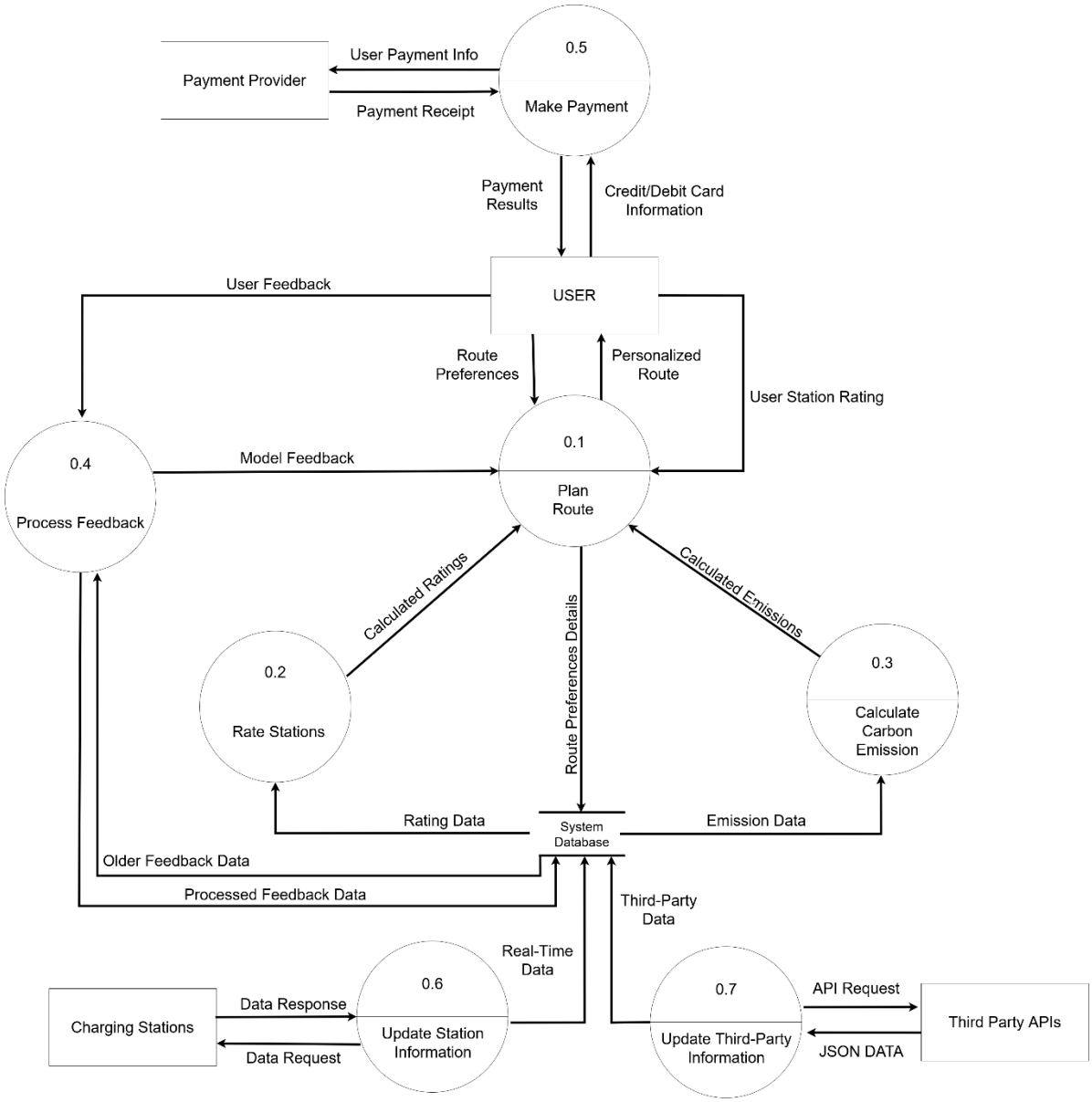

Figure 3: DFD Level - 1

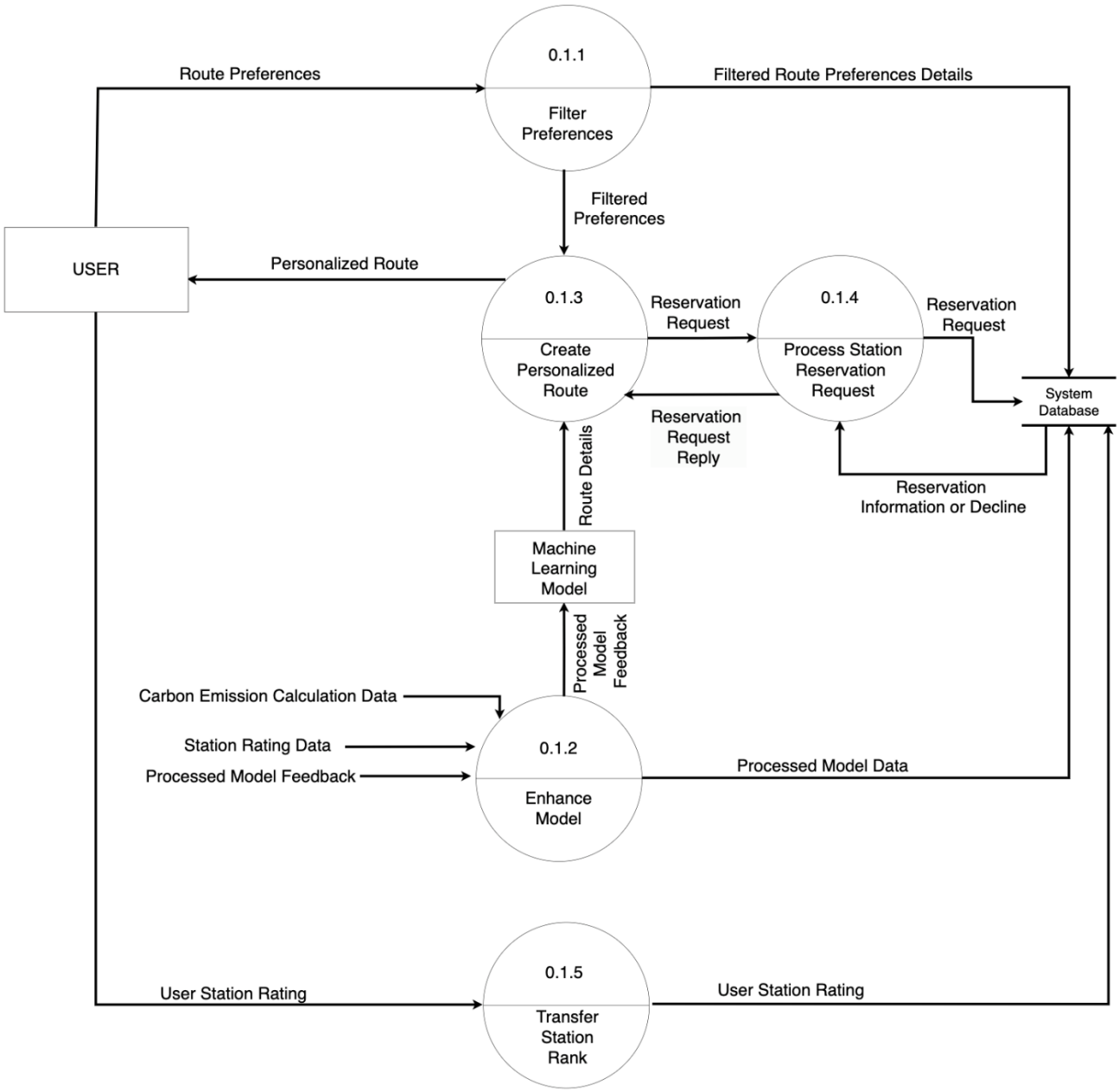

Figure 4: DFD Level - 2 for Personalized Route Creation

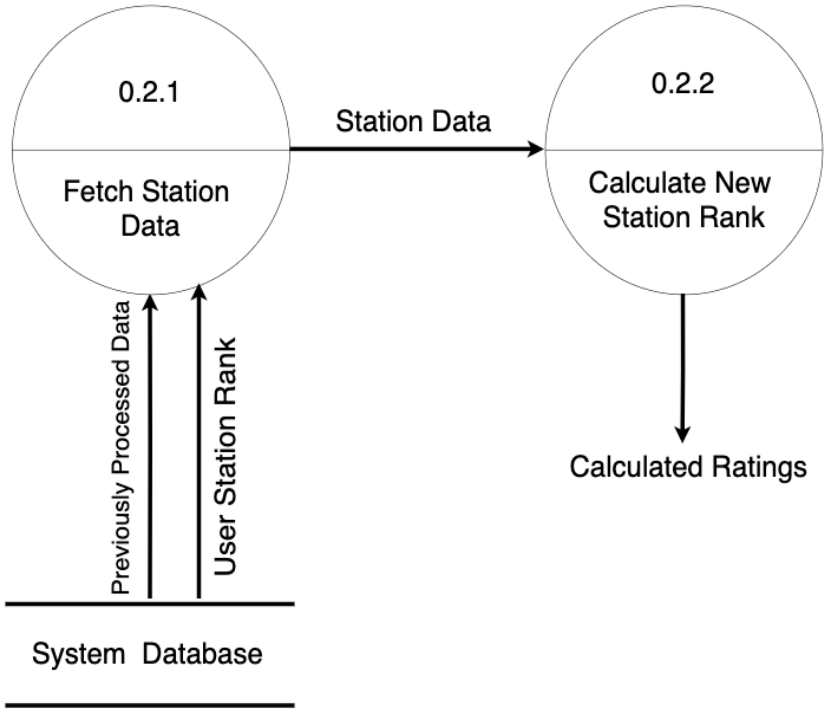

Figure 5: DFD Level - 2 for Station Rank Calculation

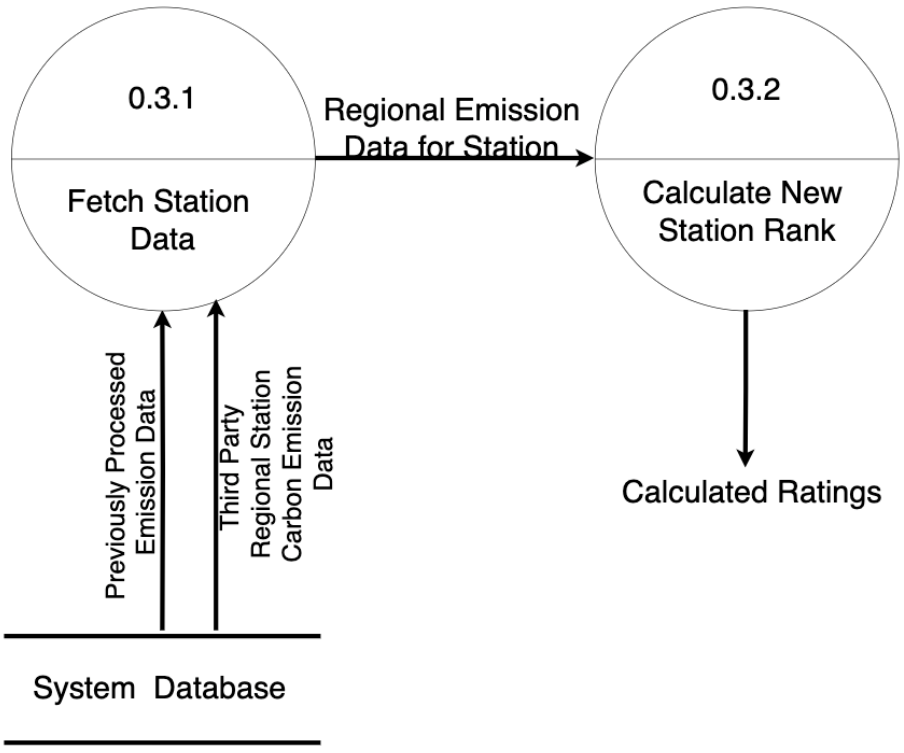

Figure 6: DFD Level - 2 for Carbon Emission Calculation

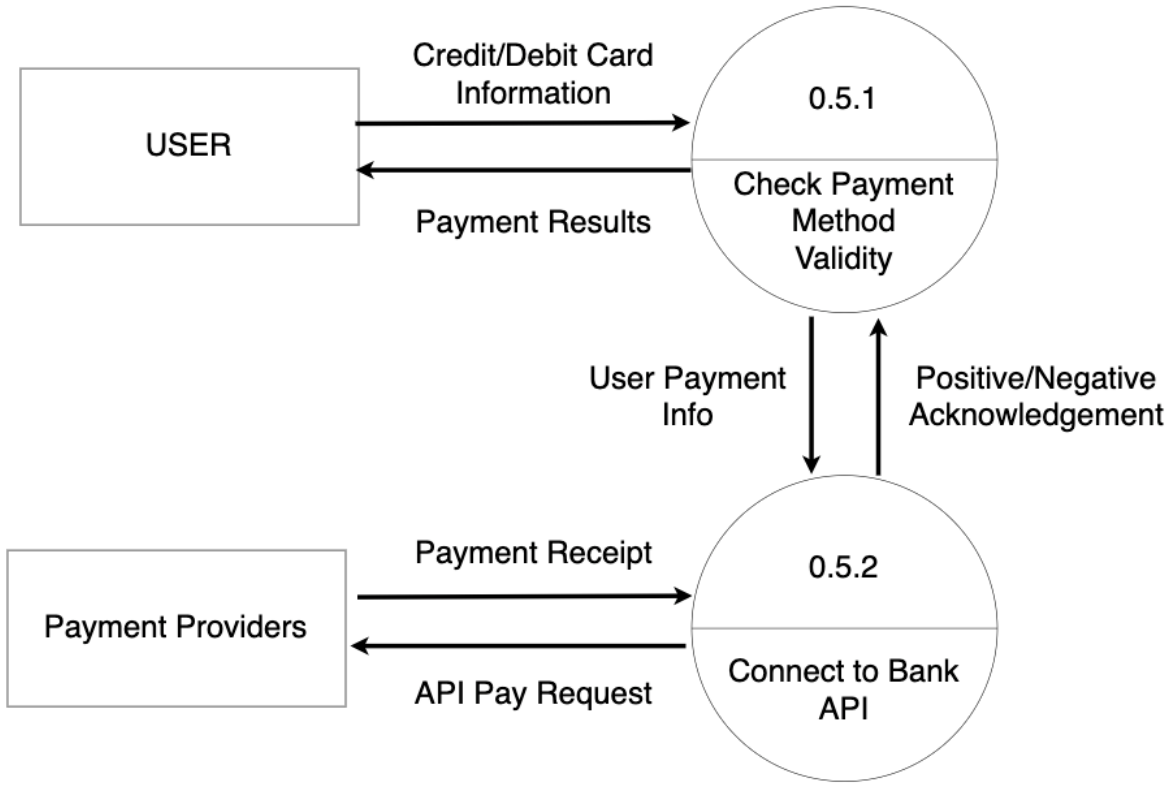

Figure 7: DFD Level - 2 for Payment

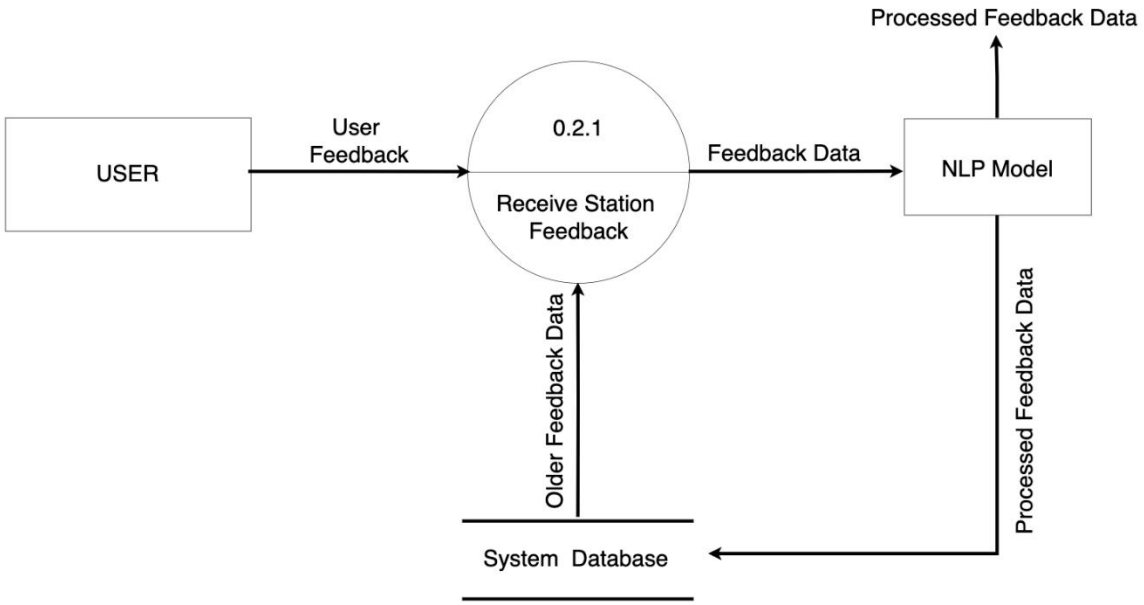

Figure 8: DFD Level - 2 for Feedback Processing

3.2.2 Activity Diagram

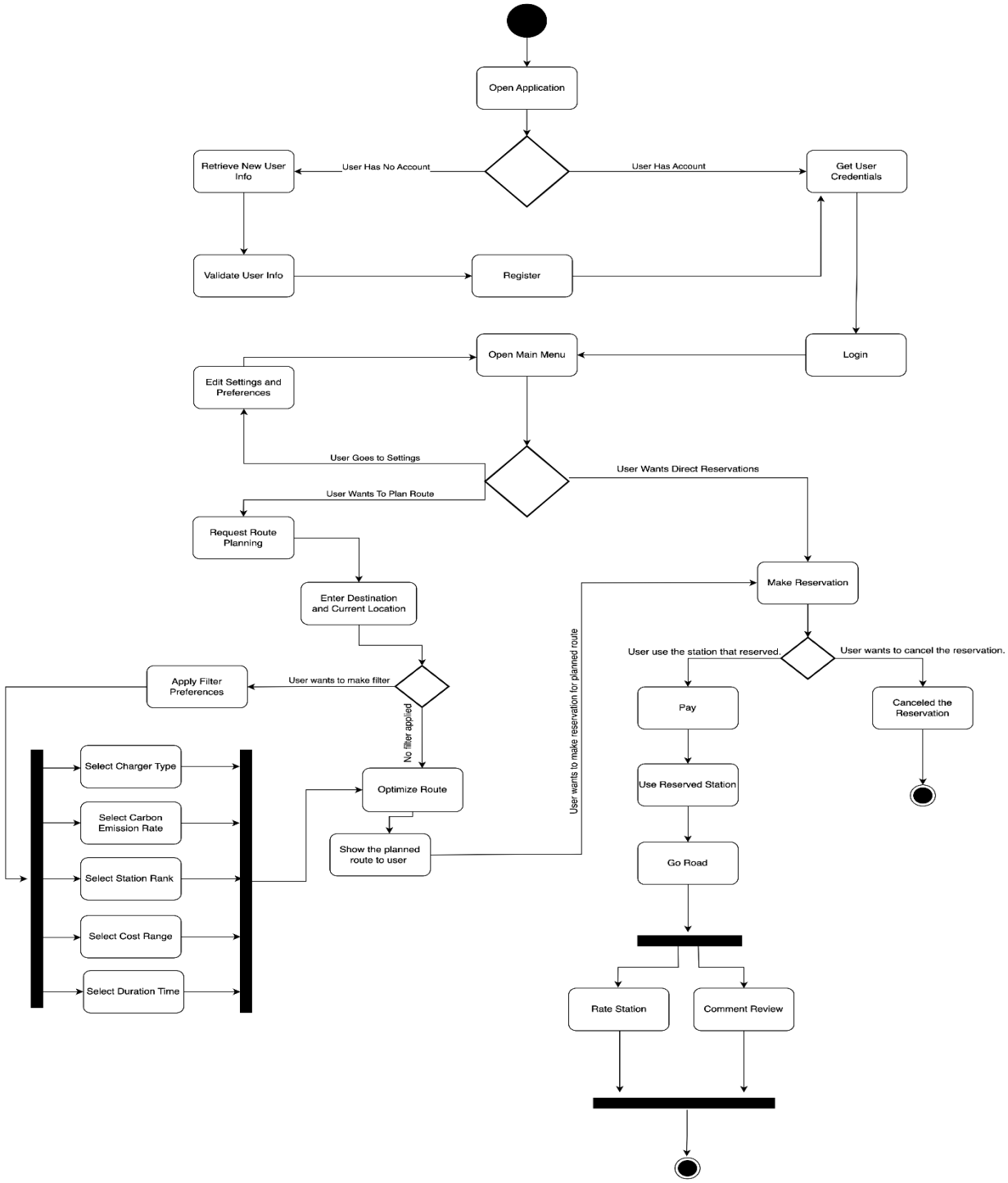

Figure 9: Activity Diagram for General System

3.2.3 Class Diagram

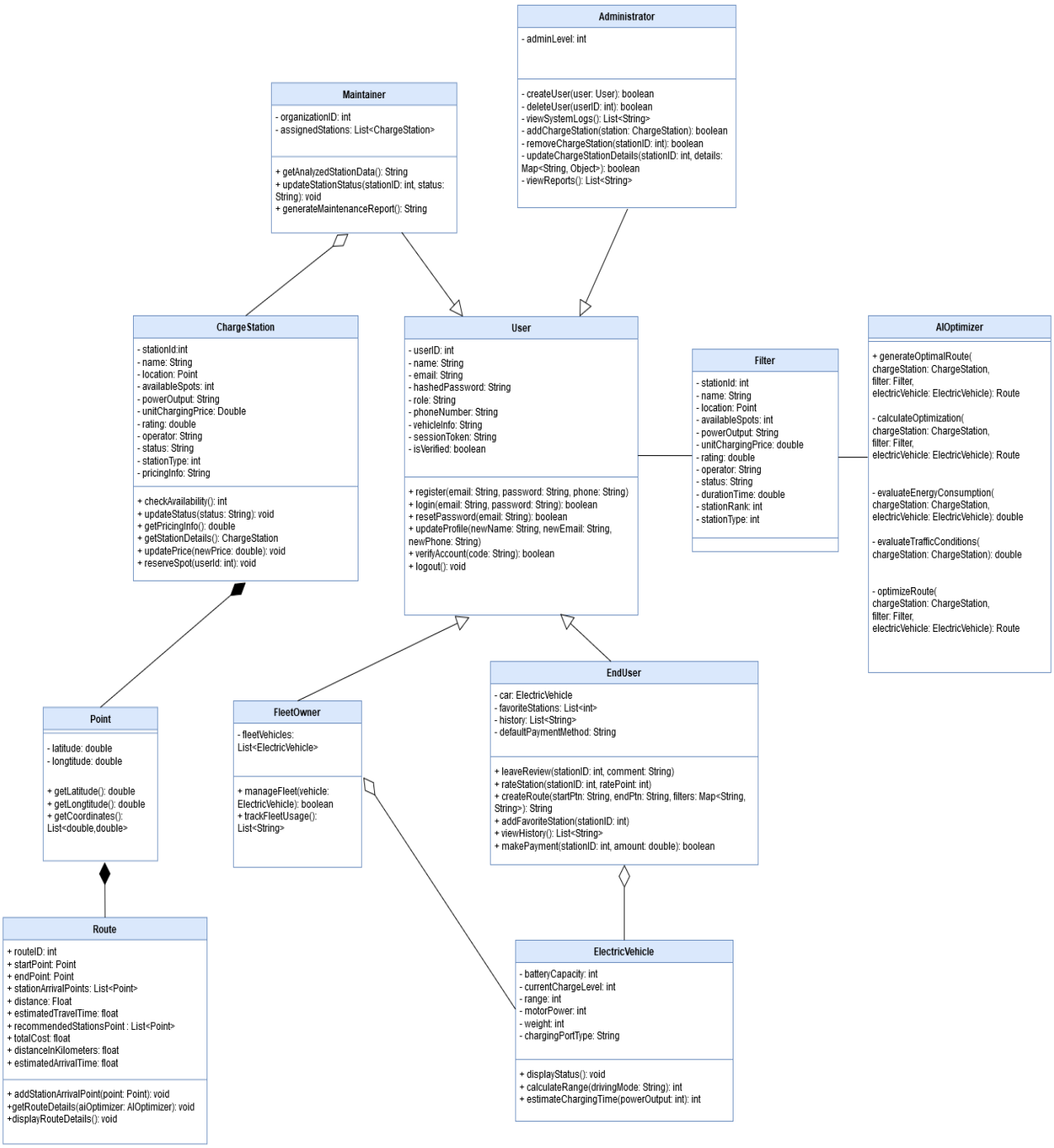

Figure 10: Class Diagram for General System

3.2.4 Sequence Diagrams

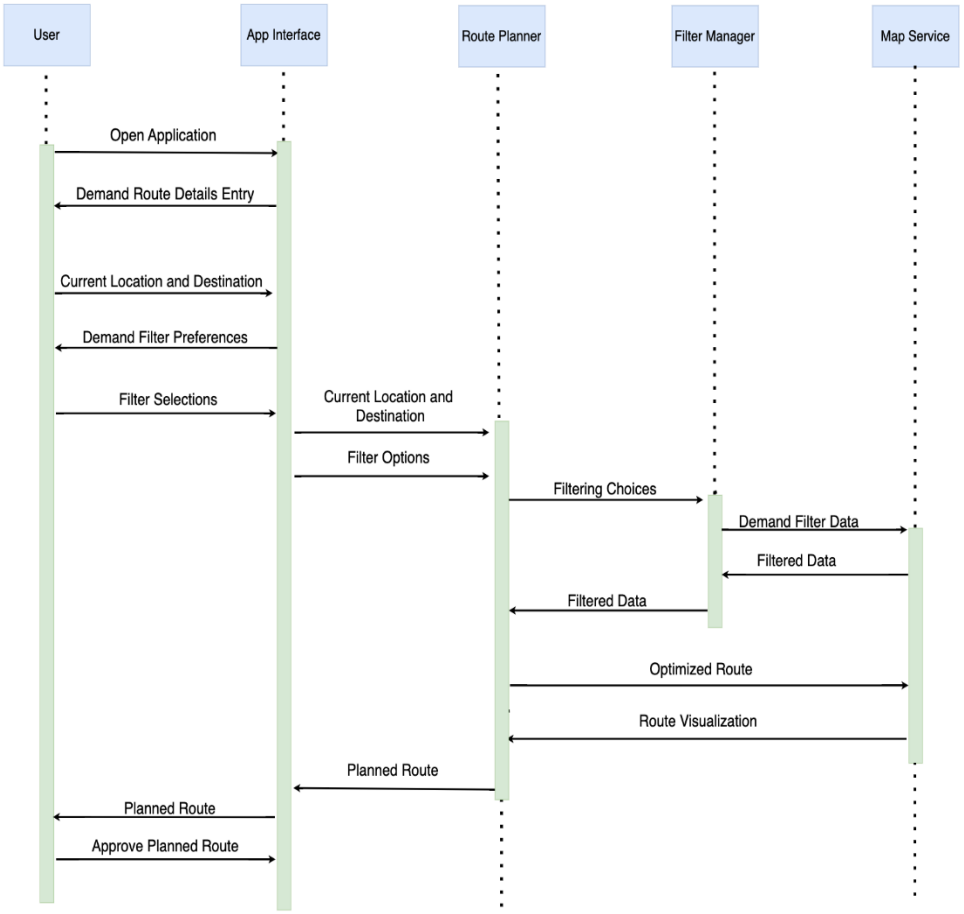

Figure 11: Sequence Diagram for Route Planning

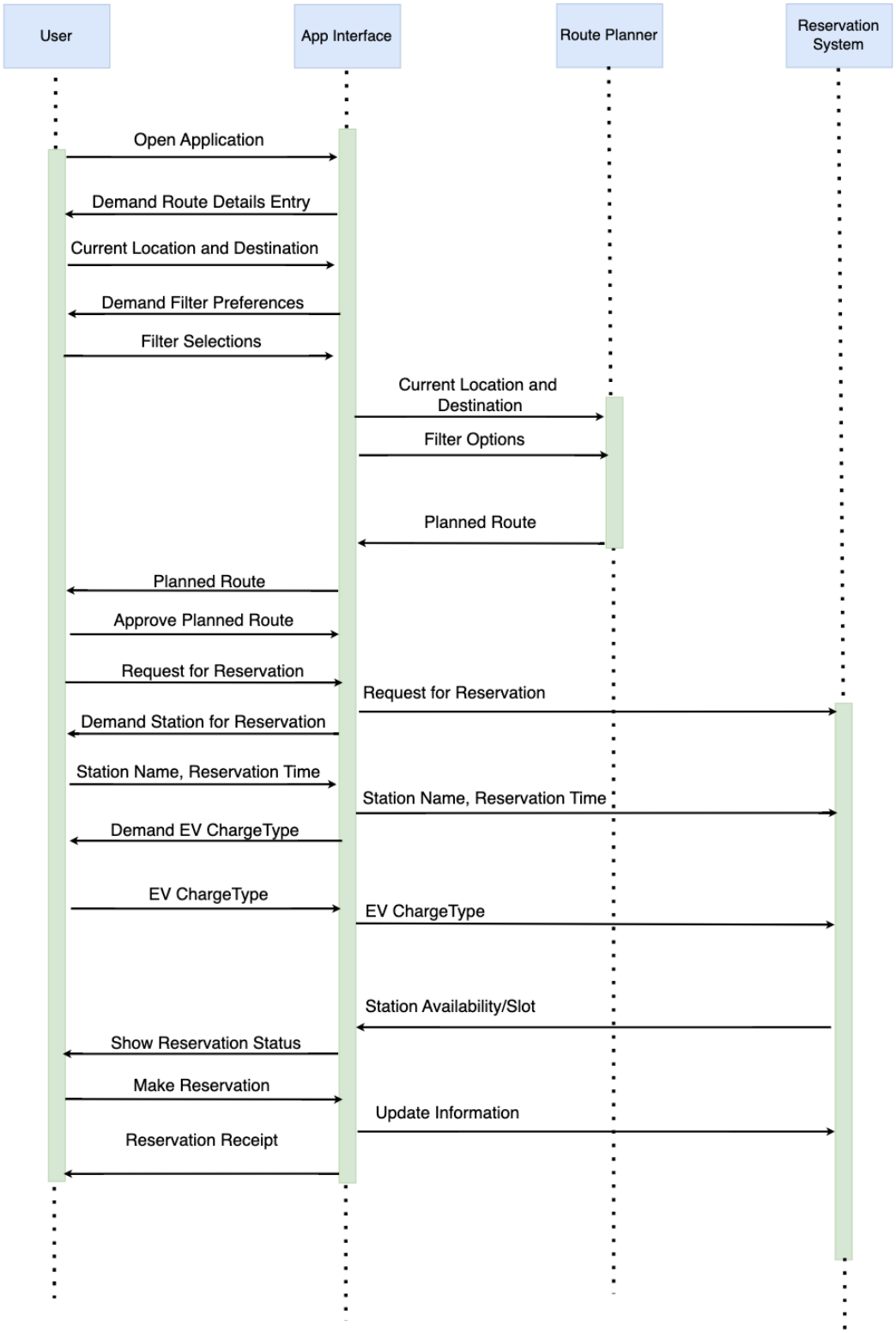

Figure 12: Sequence Diagram for Reservation System

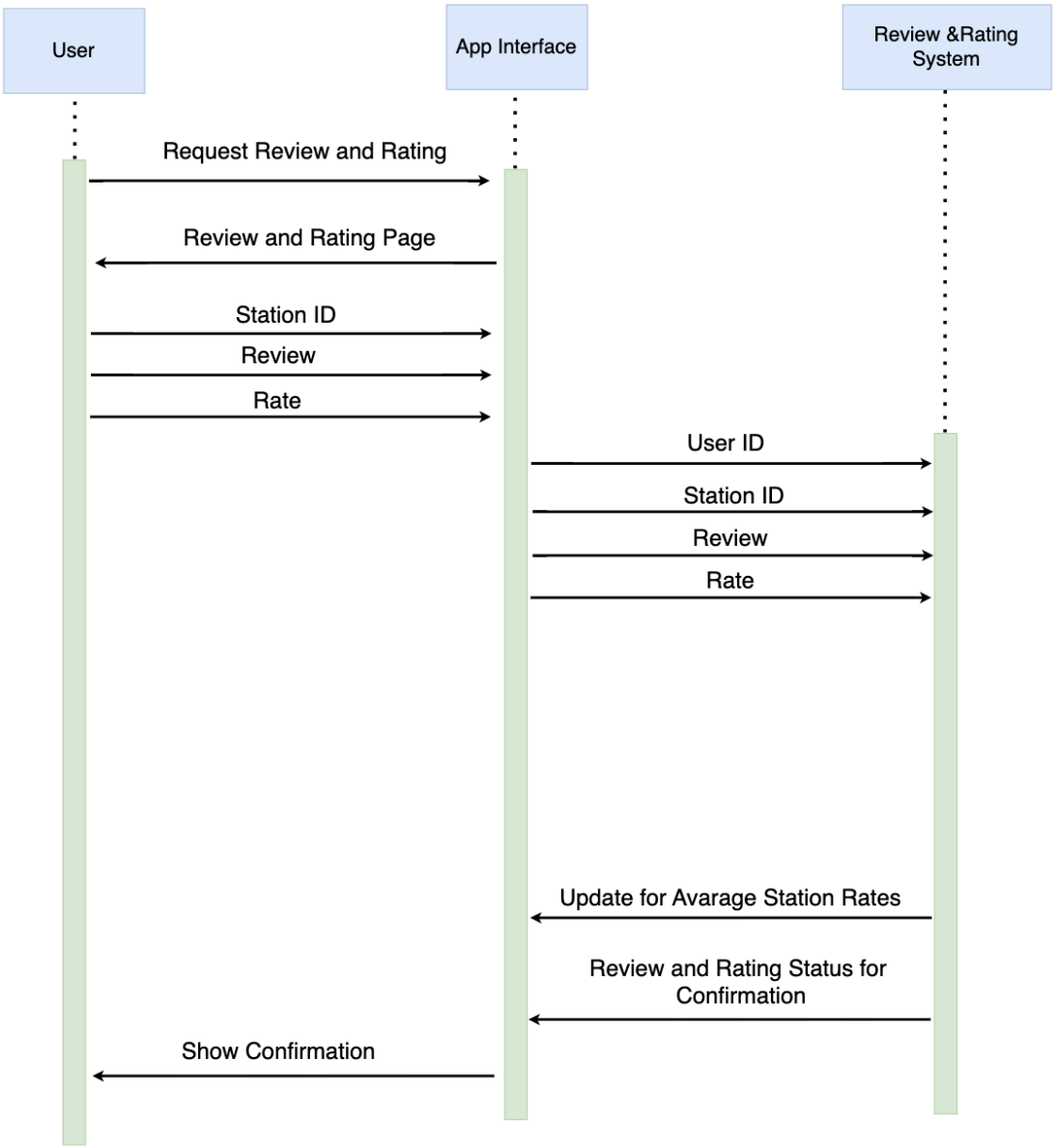

Figure 13: Sequence Diagram for Review and Rating Systems

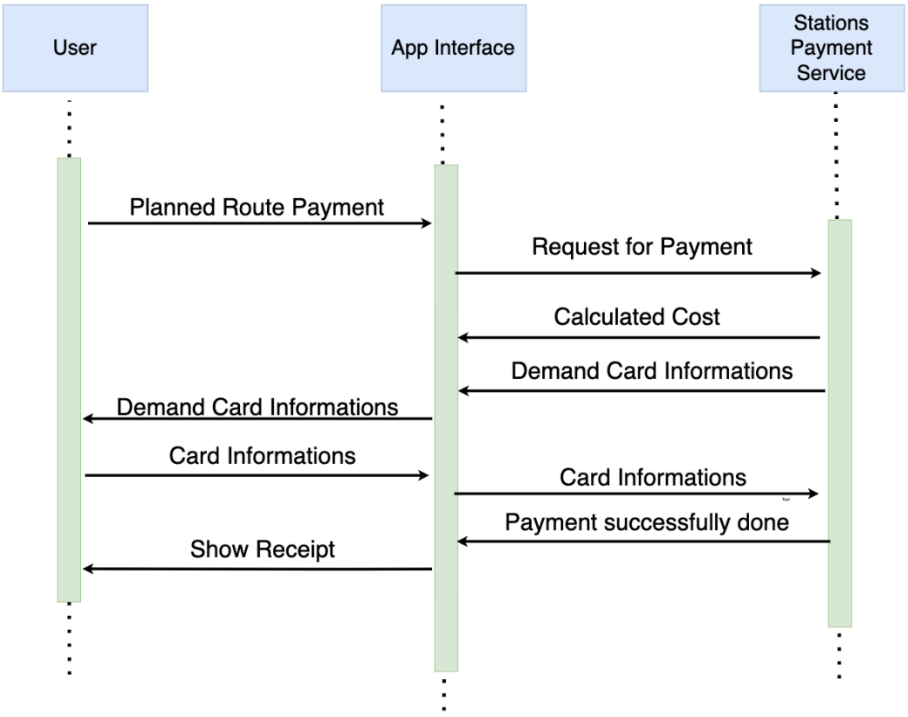

Figure 14: Sequence Diagram for Payment Service

4. User Interface Design

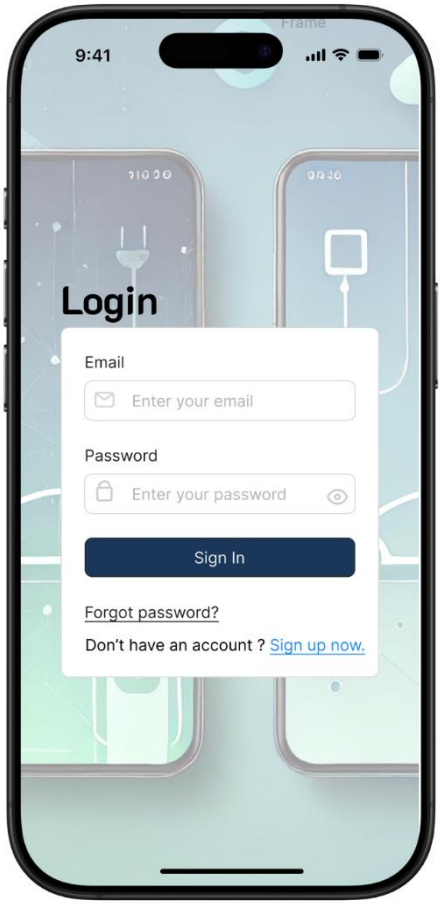

Figure 15: UI Design-1 for Login Page

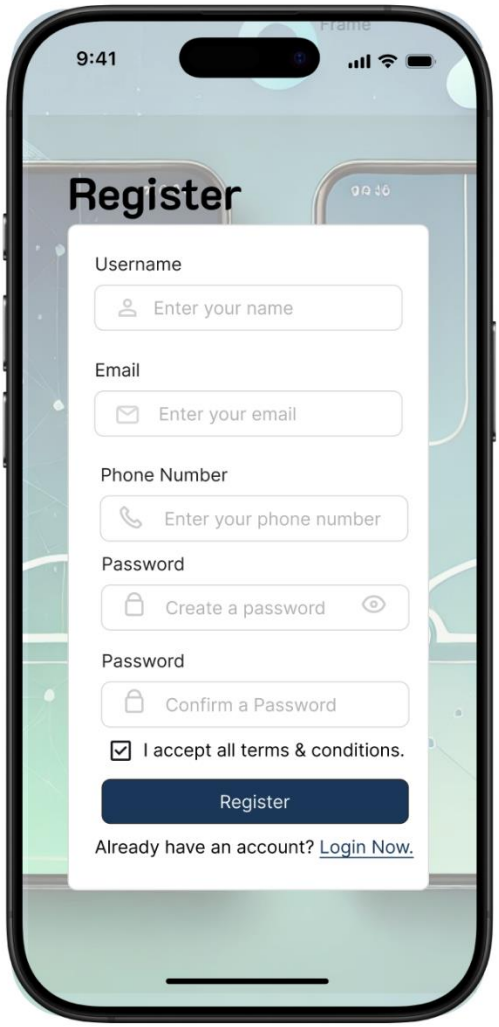

Figure 16: UI Design-2 for Register Page

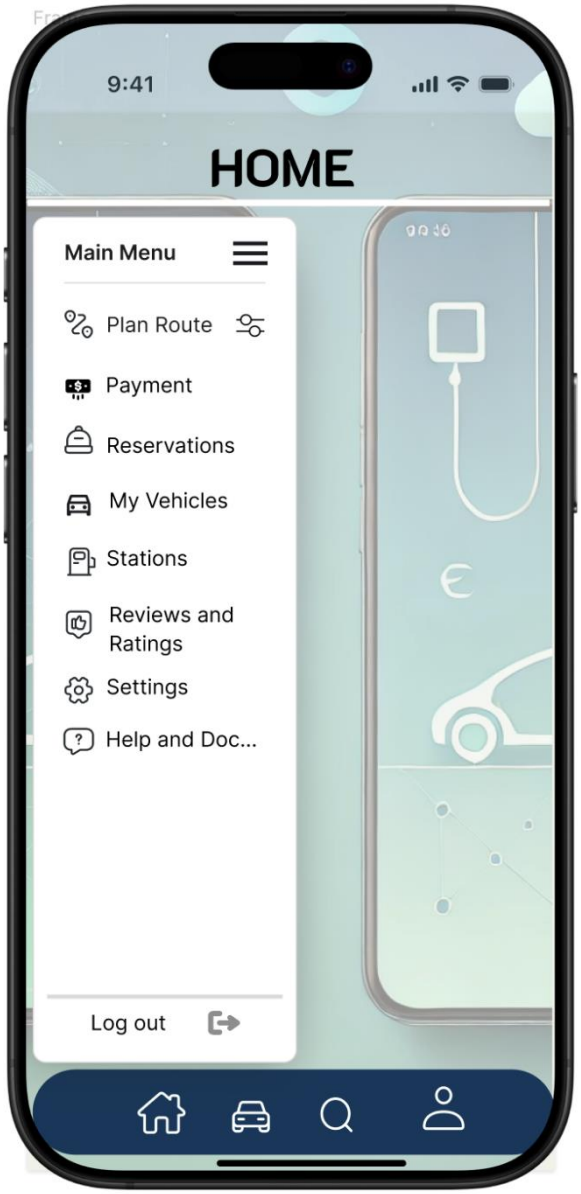

Figure 17: UI Design-3 for Home Page

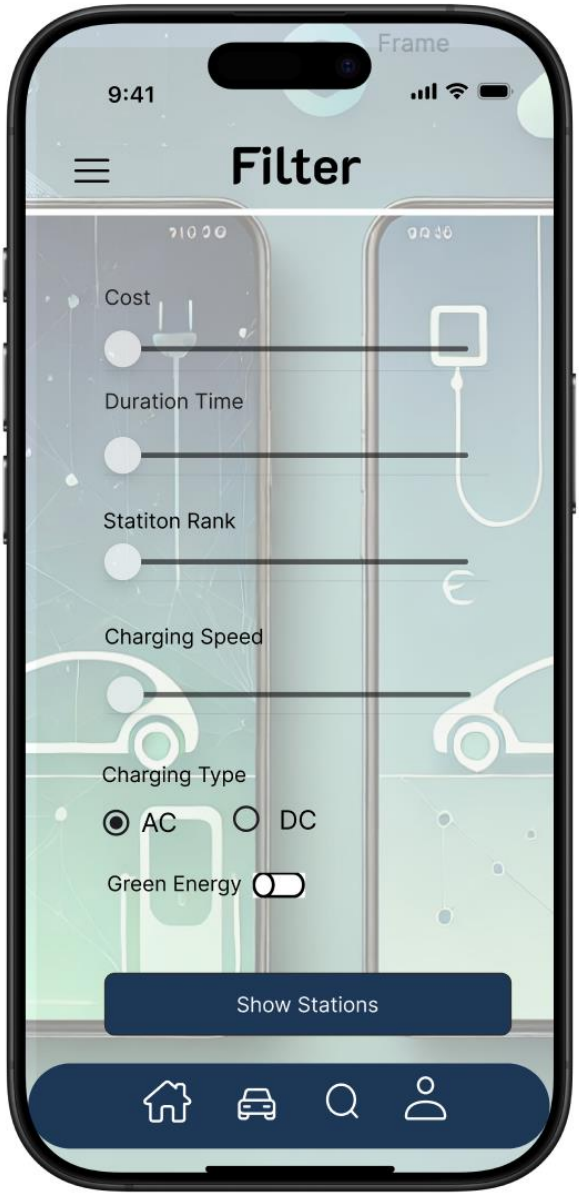

Figure 18: UI Design-4 for Filter Page

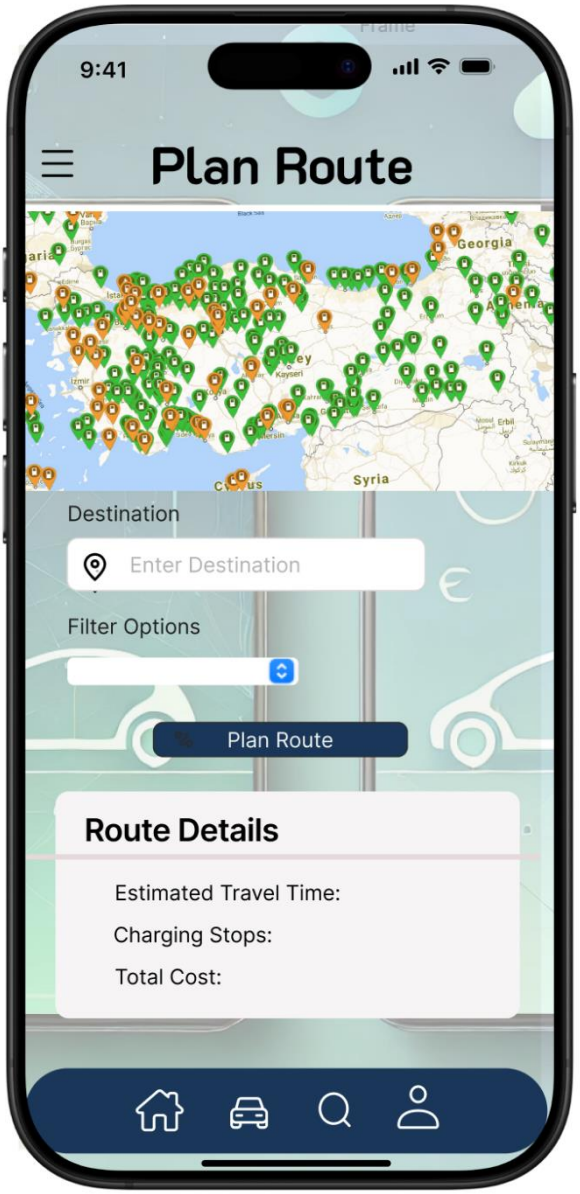

Figure 19: UI Design-5 for Plan Route Page

Figure 20: UI Navigation Schema

5. Requirements Matrix

Requirement Description	Design Component	Verification Method
User registration and login	Frontend (React Native)	Unit Testing, UI Testing
Route planning with station availability	Backend (Java/Spring Boot), ML Model	Integration Testing
Real-time station status retrieval	Backend, API Integration	System Testing, API Testing
Secure payment processing	Backend, Payment Gateway	Security Testing, Penetration Testing, Functional Test
Response time under 2 seconds	Backend, Load Balancer	Performance Testing
High availability (99.7% uptime)	Redundant Servers, Failover	Stress Testing, Monitoring
Data encryption during transmission	TLS/SSL, OAuth 2.0	Penetration Testing
User-friendly interface	Frontend (React Native)	Usability Testing
Compatibility with Android and iOS devices	Frontend	Compatibility Testing