TQS: Product specification report

Gabriel Silva [113786], Henrique Teixeira [114588], João Roldão [113920], Sebastião Teixeira [114624]
v2025-05-09

1 Introduction

1.1 Overview of the project

As the world transitions to sustainable transportation, electric vehicles (EVs) are becoming more common. However, EV adoption is often hindered by a disjointed and inconsistent charging experience. SparkFlow is a next-generation EV charging platform designed to bridge the gap between drivers and station operators by unifying discovery, booking, usage, and payment into a single platform. This platform aims to simplify and enrich the EV charging experience, promote station utilization, and encourage sustainable mobility.

1.2 Known limitations

While the project successfully implements the core functionality of station discovery, booking, and user management, there are a few limitations worth noting, particularly regarding features that were planned but not fully realized during the main development phase.
Station Pricing Management by Operators
Although the system allows station operators to register new stations and view their status, the ability for operators to change the pricing of charging sessions was not implemented in the initial version. This

functionality was part of the original plan but was deprioritized to focus on delivering a stable core booking and user management system within the timeline.

Stripe Payment Integration

The integration with Stripe for handling user payments is still in its early stages. While the system architecture and external service modules were designed to support it, the payment flow has not been fully implemented or tested in a production-like environment at the time of the delivery.

RoadTrip Functionality

Despite the logic being developed in the backend it isn't integrated into the frontend. With this the feature is not ready for a production environment but its logic is implemented and tested.

1.3 Post-Presentation Improvements

After the project presentation, we continued development and implemented user authentication, enabling secure login and session control. Additionally, we began progressing further with the Stripe integration, laying the groundwork for supporting payment flows in upcoming iterations and started with the RoadTrip Functionality. Even more in terms of User Session Data we give more of that information to the user.

1.4 References and resources

Frameworks & Libraries

These are essential components used in your codebase.

- Spring Boot – Primary backend framework for building REST APIs.
- Spring Data JPA – For abstracting database operations and working with relational databases.
- Spring Security – Used for role-based access and user authentication.
- MongoDB Java Driver – To connect and interact with the MongoDB database.
- Stripe Java SDK – For integrating payment handling via Stripe.
- Lombok – To reduce boilerplate code in Java classes

Databases and Tools

Database systems and tools that helped manage or visualize data.

- MongoDB – NoSQL database used for user/session-related data.
- PostgreSQL – Relational database used for station and booking data.

APIs & Web Services

External services integrated into your platform.

- Stripe API – Used for processing payments and generating receipts.
- Open Charge Map API – Used to enrich charging station data and extend coverage.

Development Tools

Helpful tools or environments used during development.

- Postman – For testing and debugging RESTful APIs.
- Docker & Docker Compose – For containerizing services and simulating the production environment.
- GitHub – Source code hosting, version control, and CI/CD (if used).
- Draw.io / diagrams.net – For creating architectural and deployment diagrams.

Key References & Learning Materials

Blogs, tutorials, or documentation that were particularly helpful.

- Baeldung – Spring Boot Tutorials – Extremely helpful for understanding Spring Boot, Spring Data, and security configurations.
- Stripe API Documentation – Official Stripe docs used to implement payment workflows.
- Open Charge Map API Docs – For integrating public EV station data.

- Docker Documentation – For building and configuring containers.

2 Product concept and requirements

2.1 Vision statement

SparkFlow is a unified EV charging platform that connects electric vehicle drivers and station operators through a centralized and data-driven experience. It was designed to provide a seamless, smart, and user-friendly environment for both end-users and infrastructure managers.
For EV drivers, the platform offers a powerful discovery system that enables users to search for charging stations based on filters such as price per kilowatt-hour, charging speed, connector type, and geographic proximity. While real-time data integration and dynamic rerouting logic were envisioned, these features have been scoped for future implementation. The current version includes booking functionality with time conflict management, recurring bookings, and a personal dashboard where users can track their past charging sessions. Users are also able to initiate charging sessions and, following the project presentation, we added secure user authentication to enable protected access to their data and session details.
The platform was initially designed to support payment processing through Stripe. However, this integration is currently in progress and not yet available in the production workflow. The architecture supports Stripe's API, and development has progressed post-presentation to establish the payment foundation. Future versions will incorporate a complete end-to-end payment flow within the app interface.
For station operators, SparkFlow delivers administrative tools to register new charging stations with metadata such as location, quantity of chargers, connector types, and operational status. Operators can view the health and availability of stations via their dashboards. Although the system was intended to include dynamic pricing management, this specific functionality was postponed to prioritize core features. This means operators currently cannot change pricing through the interface, though the platform architecture supports such extensibility in the future.
SparkFlow draws inspiration from platforms like Chargemap, PlugShare, and Tesla's Supercharger Network, while offering a provider-neutral experience that supports both users and operators. It emphasizes automation capabilities such as recurring bookings and intelligent station filtering, with planned features like rerouting and advanced pricing logic to be developed iteratively.
The concept and requirements for SparkFlow were established through user-centered design practices during the early stages of the project. Role-based scenarios and persona development shaped feature prioritization, and defining the MVP scope in line with academic guidelines enabled us to focus on delivering a robust, working system within the time constraints. While some advanced features were deferred, the project's architectural design ensures a clear path for future iterations and enhancements.

2.2 Personas and scenarios

SparkFlow is built with two main actors in mind, the EV Driver and the Station Operator.
The EV Driver is the primary end-user of SparkFlow, representing individual consumers who use electric vehicles for commuting, travel, or everyday transportation. This actor interacts with the platform through a user-facing interface. The goals of this actor with this software solution are to find available charging stations nearby or along planned routes, reserve a charging time slot and pay for this reservation seamlessly. Finally, being able to monitor historical charging data, costs, and environmental impact is also of interest to this actor.
The Station Operator can represent either an individual or an organization responsible for managing EV charging infrastructure. They may own a few private chargers or operate a network of public stations.

They use SparkFlow via a back office web dashboard. The objective of the Operator is to maximize usage of the charging infrastructure as well as maintaining station uptime and service quality. By monitoring consumption and performance metrics, they can also offer promotional pricing to attract customers during off-peak times.

EV Driver Personas

Daily Urban Commuter - Claúdio Martins

Cláudio Martins is a 29-year-old HR Coordinator living in Lisbon, Portugal. Tech-savvy and environmentally conscious, he relies on his Nissan Leaf to commute daily between his suburban home and his downtown office. His routine is consistent: he uses his electric vehicle twice every weekday, and efficiency is always top of mind.

His primary goals revolve around convenience and cost-efficiency. He wants to quickly find affordable charging stations near both his home and workplace and aims to minimize the time spent waiting at or using these stations. In addition, he carefully tracks his monthly EV expenses to ensure he stays within his transport budget.

However, his experience is far from seamless. Charging stations near his office are frequently occupied, making it frustrating to plan her day. The need to juggle multiple apps, each corresponding to a different charging network, only adds to her frustration.

Weekend Road-Tripper - Roberto Silva

Roberto Silva is a 38-year-old freelance photographer based in Porto, Portugal, with a passion for weekend road trips and capturing the Portuguese landscape. He drives a Tesla Model 3 and uses it primarily for long-distance travel, heading out two to three times a week for shoots in scenic, often remote locations.

When it comes to his electric vehicle, Roberto has clear priorities. He wants to plan routes that include charging stops optimized not just for speed, but also for convenience. He looks for chargers located near cafés and rest areas, places where he can relax or even get some editing done while his car powers up. Beyond convenience, he also tracks his environmental impact, taking satisfaction in knowing how much CO_2 he's saved.

But the road isn't always smooth. Roberto often finds himself frustrated by the lack of a unified planning tool. He has to manually cross-reference multiple apps and websites just to map out a route with dependable charging points. Worse, he sometimes arrives at a charger only to find it offline or already in use, wasting valuable time.

Late-Night Gig Worker - Bruna Lopes

Bruna Lopes is a 24-year-old food delivery driver working the night shift in Coimbra, Portugal. She zips through the city with precision and speed on her electric scooter. As a digital native, Bruna is highly tech-savvy and depends on smart tools to keep her nightly routine running smoothly.

Her vehicle is her livelihood, and keeping it charged is non-negotiable. Bruna is constantly on the lookout for fast, compatible chargers during off-peak hours when demand is low and time is precious.

Despite her resourcefulness, the current charging experience presents daily hurdles. Too often, she arrives at a station only to discover it doesn't support his EV type, wasting both time and energy. Pricing can be another headache. She finds herself constantly comparing rates across different providers, which is frustrating and inefficient.

Station Operator Personas

Regional Infrastructure Technician - João Faria

João Faria is a 37-year-old regional infrastructure technician working for the municipal transport department in Faro, Portugal. He oversees a network of over 20 public EV charging stations and plays a key role in maintaining their functionality, accessibility, and the public's confidence in the region's growing electric mobility infrastructure.

His mission is to ensure that every charger under his watch is operational and accessible at all times. To do this effectively, João needs real-time monitoring capabilities and instant alerts for outages, errors, or unusual usage patterns. He relies heavily on an admin dashboard, which serves as his central hub for managing the entire network. From here, he checks the health of each unit, runs diagnostics, and exports weekly performance data for internal KPIs.
Despite his technical skills, João often finds himself battling outdated systems that don't provide real-time status updates. On top of that, switching between different tools for diagnostics, reporting, and management slows him down and fragments his workflow.

Green Tech Startup Operator - Rita Neves
Rita Neves is a 32-year-old entrepreneur based in Lisbon, Portugal, at the helm of a green tech startup focused on providing EV charging solutions. With a network of 50 chargers spread across three cities, Rita is building a business that merges sustainability with smart technology, aiming to turn EV charging into a seamless, customer-centric experience.
Her business runs on data. Rita is eager to implement dynamic pricing models that respond to variables like demand, time of day, and location. She also wants to track user behavior and build meaningful loyalty programs that keep EV drivers coming back.
But the road to full control has its challenges. Most platforms she works with offer limited flexibility when it comes to customizing pricing tiers, which hampers her ability to test different business models. She's also frustrated by the lack of detailed customer segmentation data, essential for crafting targeted offers and personalized user experiences.

EV Driver Scenarios
Morning Efficiency Boost - Daily Urban Commuter
Cláudio wakes up at 7:00 AM and is preparing to drive to work, knowing that his car needs to recharge. Opening the SparkFlow app from home, he sets his work location as the destination and filters stations by availability in a 1km radius and lowest price per kWh. Having real-time availability, the application shows the unoccupied chargers and offers a "Book Now" feature. He reserves a spot for a charger near his office building. Upon arrival, he unlocks the charging station via the app and plugs in the car. When charging is complete, SparkFlow notifies him and provides a receipt.
Budget Watchdog - Daily Urban Commuter
At the end of the month, Cláudio wants to check his transport budget. He opens the Driver Dashboard and reviews the Monthly Summary that has information like the number of charging sessions, total kWh consumed, total cost and average price per session. SparkFlow also shows a progress bar comparing actual expenses to his preset monthly budget.
Routine Charging Plan - Daily Urban Commuter
Wanting a more hands-off approach to his weekly charging needs, Cláudio configures a weekly recurring booking in the SparkFlow app. He selects Monday through Friday at 8:00 AM at the Praça Liberdade Parking Garage. The app checks availability and books recurring slots. If a slot becomes unavailable, Claudio receives a Smart Reroute alert with the nearest alternative.
Route Planner with Charging Stops - Weekend Road-Tripper
Roberto is planning a weekend trip to Serra da Estrela for some landscape shoots. He opens the SparkFlow Route Planner and enters the starting point as Porto, the destination Serra da Estrela and the vehicle info Tesla Model 3. SparkFlow then calculates his best routes and inserts optimized charging stops based on the battery range and charger availability and also pre-books his preferred chargers along the route.
Environmental Impact Tracker - Weekend Road-Tripper
After his road trips, Roberto likes to reflect on his environmental contribution. For that information, he opens the Trip Summary in the dashboard, presenting him with information like distance driven, energy consumed and CO2 saved compared to a gasoline car.

Report a Broken Charger - Weekend Road-Tripper

Arriving at a rural charger he booked through the SparkFlow app, he finds it to be offline and unresponsive. He opens the app and taps in the Report Issue button directly on the charger's profile. He selects a charger offline from a quick list of issues presented and adds a short comment along with the report "Screen frozen and not accepting charge".

Fast Match - Late-Night Gig Worker

It's 11:30 PM. Bruna has just completed a round of deliveries and needs a quick recharge to finish her shift. She opens the app and taps in Quick Charge Finder. The system auto-detects her location and EV type (electric scooter), then filters by connector compatibility, availability, speed and lowest price. SparkFlow highlights three nearby chargers, she picks one and books it.

Instant Reroute When a Station Fails - Late-Night Gig Worker

Bruna arrives at the station she booked, but it's showing an error on the charger screen. She taps Report Issue in SparkFlow, marks the charger as Out of Service and leaves a quick comment. The system flags the station in real-time for others and automatically suggests the next closest working charger within 1 km, offering to transfer the reservation. Bruna accepts the transition.

Station Operator Scenarios

Registering a New Station - Regional Infrastructure Technician

The municipality has installed two new chargers at a park-and-ride location on the outskirts of Faro. João logs into the SparkFlow Admin Dashboard and selects "Add New Station". For the first one, he enters details like Station Name (Faro Norte P&R), Charger Count (2), Connector types (CCS and Type 2), Location (map pin or GPS), Operating Hours, Pricing Tier, and Accessibility Info. The new chargers appear on the public map within minutes.

Real-Time Fault Alert and Fast Response - Regional Infrastructure Technician

It's 9:15 AM. João is in a team meeting when an alert pops up on his phone. SparkFlow detects that Charger #12 (Downtown Bus Terminal) has failed to respond to routine pings. He opens the SparkFlow Admin Dashboard and sees the charger status, last session and recent logs. João attempts a remote soft reboot from the dashboard. If successful, the charger returns to green, no further action needed. If failed SparkFlow auto-creates a maintenance ticket.

Weekly Station Performance Review - Regional Infrastructure Technician

Every Monday morning, João prepares a KPI report for his department. He logs into SparkFlow and opens the Reports Dashboard. João generates a weekly performance summary with total uptime per station, number of sessions per location, average charging time and utilization rates and maintenance events and downtime logs. The report is automatically formatted as a PDF export. He downloads the files and attaches them to his email to the department head, all done in under 5 minutes.

Deactivating or Removing a Station - Green Tech Startup Operator

Rita is decommissioning two underperforming chargers in a private coworking space in Porto. She opens the Station Management panel and selects the 2 underperforming stations, and in the context menu, she chooses Mark as Inactive. The system notifies users with active reservations and excludes them from the map.

Pricing Strategy Rollout - Green Tech Startup Operator

Rita wants to increase charger usage at less busy stations during off-peak hours across her network. She opens the SparkFlow Operator Dashboard and selects the "Pricing Rules" module. She sets a pricing rule in Coimbra by selecting the time range and discount. Rita reviews the data after one week, sees increased midday usage, and decides to scale the model to Lisbon stations.

2.3 Project epics and priorities

SparkFlow will be developed iteratively over a series of timeboxed implementation periods. Each iteration introduces new functionality aligned with prioritized epics, enabling continuous integration, validation, and feedback. The plan is structured to ensure core user journeys are operational early in the development cycle, while backend tools and advanced features are layered in progressively.

Iteration 2 (May 14 - May 20)
Goal: Deliver a usable foundation for drivers and station operators, enabling discovery and station registration.
Involved Epics:

- Epic 1 – Charging Station Discovery
  - Implementation of map and basic station filters (location, distance, availability).
- Epic 5 – Station Onboarding and Configuration
  - Operator interface to register a new station with metadata. Station visibility toggling included.

Outcome: Users can search for chargers, operators can register them.

Iteration 3 (May 21 - May 27)
Goal: Finalize core driver interaction loop and introduce driver-facing history tools as well as booking.
Involved Epics:

- Epic 1 – Charging Station Discovery
  - Complete advanced filters (price, connector type) and rating/review display.
    Initial Quick Charge Finder prototype.
- Epic 2 – Slot Booking and Management
  - Basic booking system with time-slot selection and reservation storage. Users can view and cancel bookings.
- Epic 3 – Charging Session Interaction
  - Implement remote unlock, in-session status updates, and error reporting.
- Epic 4 – Station Onboarding and Configuration
  - Initial dashboard with session history, usage summaries, and cost breakdown.

Outcome: Drivers can now complete a full charging journey from discovery to usage, and review their consumption.

Iteration 4: May 18 – May 31
Goal: Introduce payments and initial station monitoring features for operators.
Involved Epics:

- Epic 6 – Station Monitoring and Maintenance
  - Operators view session logs, and are alerted to faults.
- Epic 9 – Payment and Billing Integration
  - Chargers now display pricing, and a basic in-app payment process is connected to a sandbox gateway

Outcome: SparkFlow adds transactional value. Operators begin monitoring their infrastructure. Users can now pay for charging sessions within the app.

Iteration 4: May 22 – June 4
Goal: Deliver business tools and location-aware user enhancements.
Involved Epics:

- Epic 7 – Business Rules Management
  - Operators configure time-based or location-based pricing rules. System tracks usage impact.
- Epic 8 – Map and Location Services Integration
  - First version of a route planner.
- Epic 9 – Payment and Billing Integration
  - Expand payment features to include receipts, stored billing history.

Outcome: Operators gain tools to optimize pricing and charger utilization. Users benefit from smarter navigation and full billing visibility.

2.3.1 Detailed Epics

Epic 1 - Charging Station Discovery
As a driver, I want to search and filter charging stations so I can quickly find the most suitable option based on my needs.
Acceptance Criteria:
- Real-time availability map
- Filters (price, distance, speed, connector type)
- “Quick Charge Finder” feature
- View charger details, reviews, and ratings

Epic 2 - Slot Booking and Management
As a driver, I want to reserve a charger ahead of time so I can avoid waiting and ensure availability.
Acceptance Criteria:
- Booking system with time slots
- Recurring bookings
- Booking modification and cancellation
- Conflict management and rerouting

Epic 3 - Charging session Interaction
As a driver, I want to unlock, use, and end a charging session smoothly using the app.
Acceptance Criteria:
- Remote unlock via app
- Charging status updates
- End-of-session alerts
- Error handling and reporting

Epic 4 - Driver Dashboard & History
As a driver, I want to view my charging history and performance stats to manage costs and track usage.
Acceptance Criteria:
- Monthly summary: sessions, kWh, cost, average rate
- Environmental impact (CO_2 savings)
- Session history and receipts

Epic 5 - Station Onboarding and Configuration
As an operator, I want to register and configure charging stations so they are available to drivers.
Acceptance Criteria:
- Register a new station
- Add station data (connector types, location, pricing)
- Station visibility toggling (active/inactive)
- Station removal

Epic 6: Station Monitoring and Maintenance
As an operator, I want to monitor and maintain my stations to ensure high availability and user satisfaction.
Acceptance Criteria:
- Monitoring of Stations
- Fault alert system
- Session logs

Epic 7: Business Rules Management

As an operator, I want to set pricing and rules to improve charger utilization and profitability.
Acceptance Criteria:

- Set pricing logic and rules
- View the impact of pricing changes

Epic 8: Map and Location Services Integration
As a driver, I want to view stations on a map and get location-based suggestions.
Acceptance Criteria:

- Interactive map interface for all users
- Location-aware suggestions
- Route planner
- Geolocation tagging for stations

Epic 9: Payment and Billing Integration
As a user, I want to handle payments smoothly and securely through the app.
Acceptance Criteria:

- Pricing display per charger
- In-app payments (wallet or gateway)
- Session billing and receipts

3 Domain model

3.1 Domain Overview

The domain of this project centers around the management of electric vehicle (EV) charging infrastructure, allowing users to locate and book charging stations, manage their charging sessions, and track station availability. The system distinguishes between normal users and operators (who may manage or oversee charging stations).
This domain model emphasizes clarity and extensibility. User, Booking, and ChargingSession are separated to reflect distinct phases of interaction (reservation vs. actual use). The Station is central and interacts with both time-based entities, enabling flexible queries, operational reporting, and future expansion (e.g., adding pricing, maintenance reports, or dynamic power availability).

3.2 Core Information Concepts

User: Represents a registered person using the platform.

- id
- username
- email
- password
- isOperator

Station: Represents a physical EV charging station.

- id
- name
- address
- city
- country
- latitude

- longitude
- status
- quantityOfChargers
- power
- isOperational

Booking: Represents a reservation for a charging station.
- id
- stationId
- userId
- startTime
- endTime
- recurringDays
- status

ChargingSession: Represents an active or past charging session.
- id
- stationId- userId
- finished
- startTime
- endTime

3.2.1 Relations

A User can create many Bookings.
A User can initiate many ChargingSessions.
A Station can have many Bookings and ChargingSessions.
Each Booking and ChargingSession is associated with one User and one Station.

3.3 UML Class Diagram (Logical Model)

4 Architecture notebook

4.1 Key requirements and constraints

4.1.1 Functional Requirements

The system is intended to provide a comprehensive platform for managing electric vehicle (EV) charging infrastructure. The core features are designed around the needs of EV drivers, station operators.

User Authentication and Management
- Users must be able to register, log in, and maintain their profiles securely.
- Distinction is made between regular users and operator users, each having different privileges.

Station Discovery and Management
- Users should be able to browse, search, and filter available charging stations by location and status.
- Operators can create, update, or deactivate stations as needed.
- Each station must be associated with metadata such as address, geolocation, operational status, and number of chargers.

Booking System
- Users must be able to create bookings for specific time slots, with validation to avoid overlapping or conflicting reservations.
- Recurring bookings should be supported to accommodate frequent usage patterns.
- Bookings must be modifiable or cancellable under certain conditions.

Real-Time Charging Session Tracking
- Users can start and stop charging sessions.
- The system must track when a session begins and ends and record whether it was completed successfully.
- Session data may later be used for analytics or billing.

4.1.2 Non-Functional Requirements

Characteristic	Importance	Expanded Description
Scalability	High	The system should gracefully handle increases in user count, station registrations, and charging session volume. This includes scaling the database, services, and infrastructure components horizontally.

Availability	High	EV charging systems operate around the clock. Users expect 24/7 availability for station access, booking services, and session monitoring. Redundancy and fault-tolerant mechanisms are critical.
Responsiveness	Medium	Users expect quick feedback when searching for stations, booking slots, or starting sessions. UI responsiveness must be supported by backend performance optimization.
Integration	Medium	The platform may need to interface with third-party services like map APIs (Google Maps), payment providers (Stripe, PayPal), utility APIs (for smart grid integration), or IoT networks (for physical station updates).
Testability	Medium	Unit and integration tests should be feasible across modules. Decoupled components and clear domain logic support automated testing.

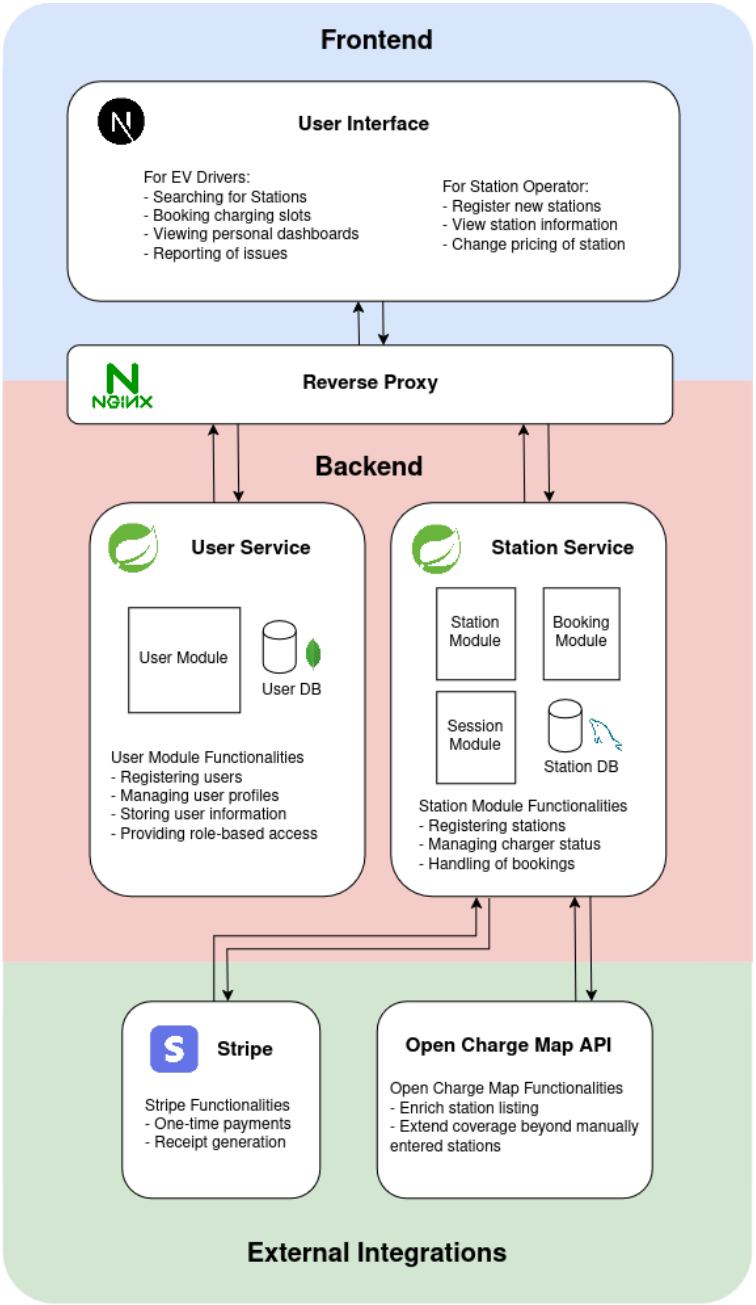
4.2 Architecture view

4.2.1 Overview & Component Logic

This is the proposed system architecture. It adopts a modular service-based architecture organized into 3 distinct layers, frontend interface, backend services and external integrations. This logical separation improves maintainability and enables independent scalability of architecture components which aligns with system goals. In the backend services we have divided it into logical business concepts modules, like user, station, booking and sessions.

4.2.2 Component Logic

User Interface (Frontend)



Built for both EV drivers and station operators, it provides interfaces for station search, bookings, user dashboards, and station management tasks. It communicates with the backend exclusively through REST APIs.

Reverse Proxy (Nginx)
Acts as a routing layer, forwarding client requests to appropriate backend services, handling load balancing and enabling future scalability via service isolation.

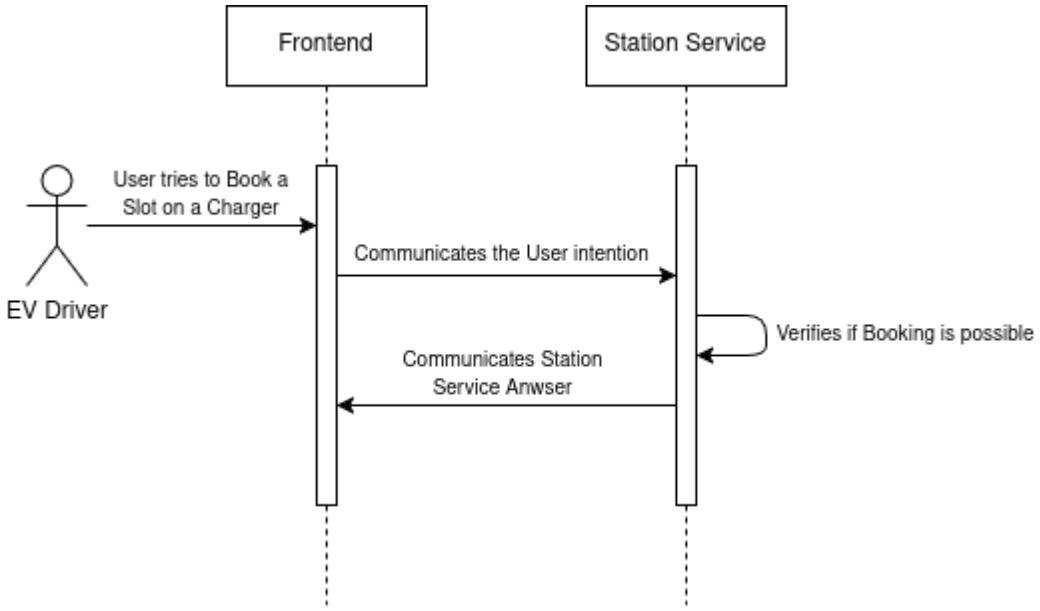
User Service
Encapsulates the user module and manages:
- Registration and authentication
- Profile data and preferences
- Role-based access control (users, operators, future admins)

Station Service
Manages three core domains:
Station Module – CRUD operations on station metadata and geo-coordinates
Booking Module – Time slot validation, booking creation, recurring bookings
Charging Session Module – Real-time tracking of charging activities

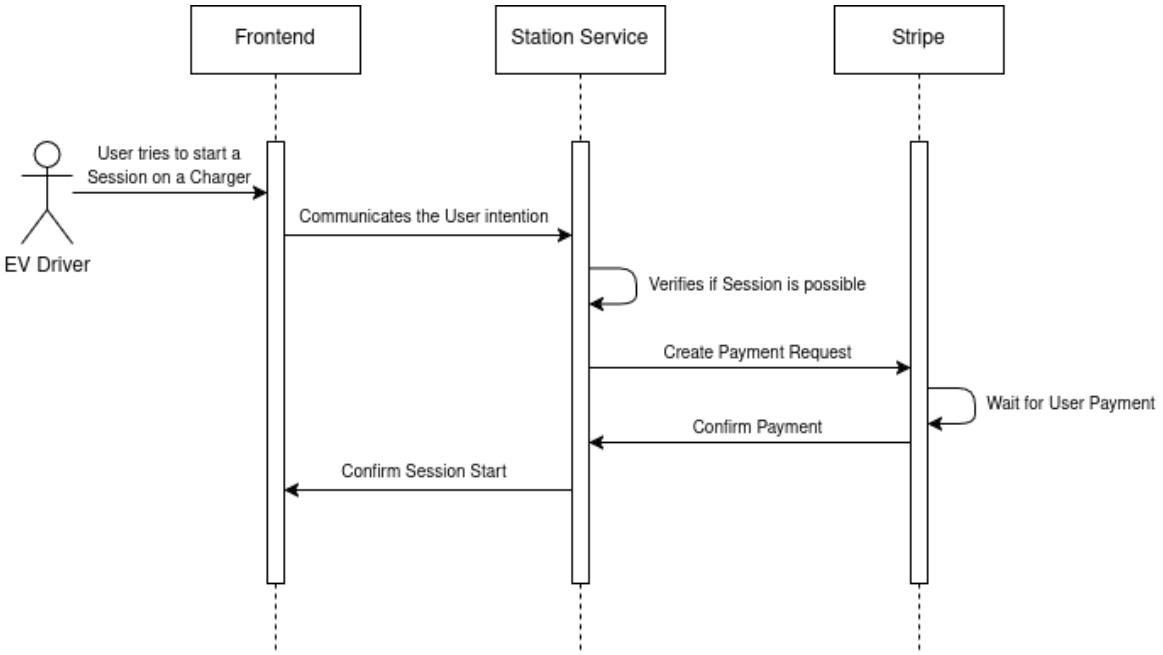
External Integrations
The architecture includes connections with:
Stripe – For one-time payment processing and receipt generation
Open Charge Map API – To enrich the station dataset and provide broader geographical coverage

4.2.3 Sequence Diagrams

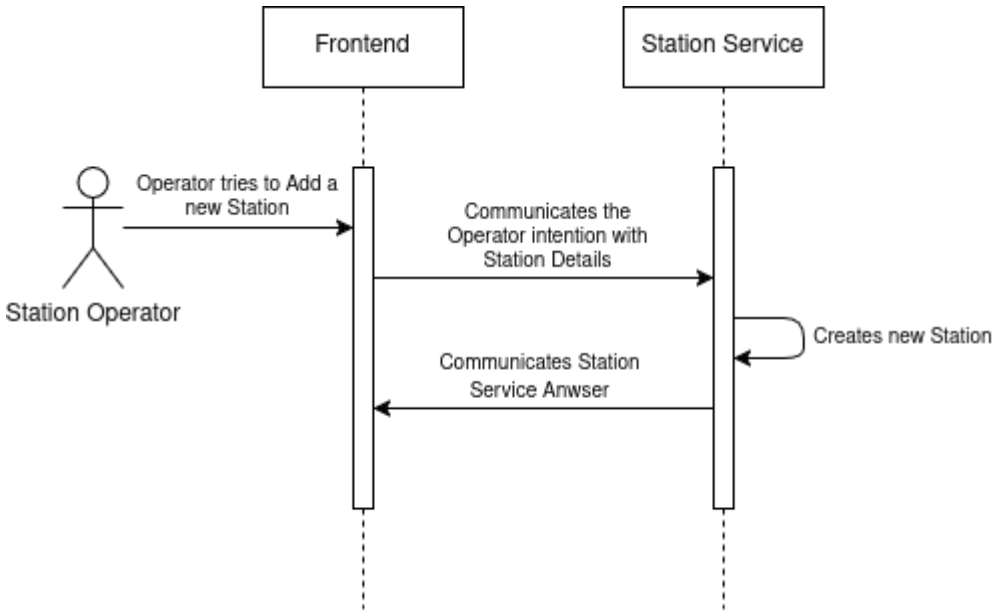
Booking a Charging Slot



Starting a Charging Session with Payment

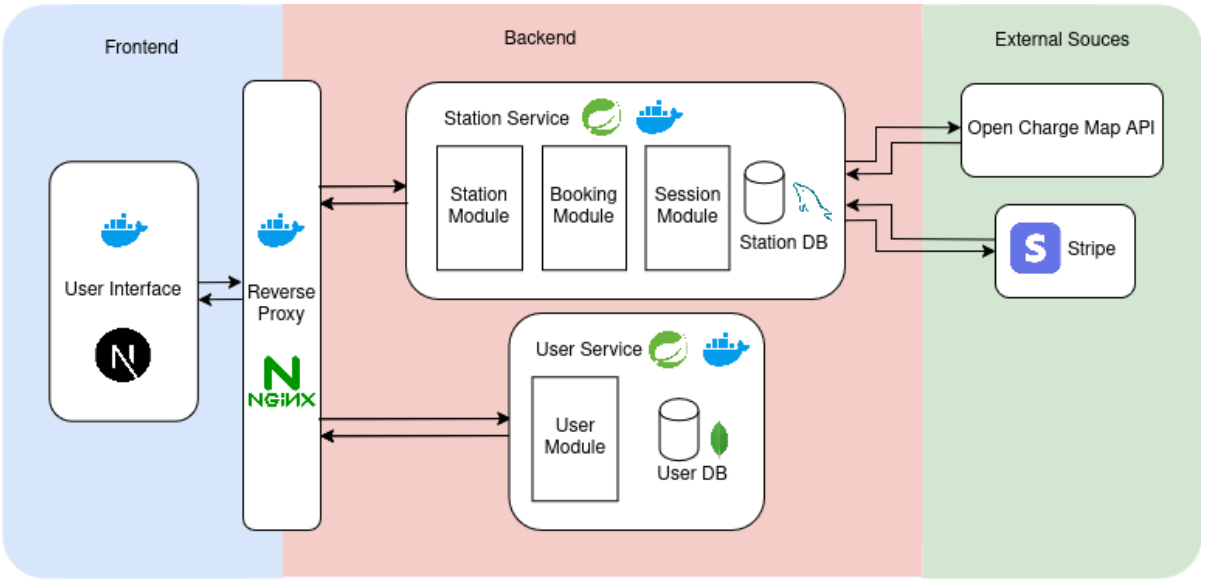


Registering a New Station



4.3 Deployment view

The deployment architecture of the system follows a microservices-based model, where the backend is split into multiple Spring Boot services that communicate via REST APIs. Each service is containerized using Docker and managed via Docker Compose. The frontend, built with a modern JavaScript framework, is served through an Nginx reverse proxy which also handles routing to backend services. Each database is in the corresponding service container.
External integrations, such as the Stripe API and Open Charge Map API, are accessed over the internet. Proper API keys and environment variables are configured in Docker .env files.



5 API for developers

In this project, Swagger was used as the hosted API documentation tool to provide detailed method information, parameter specifications, and interactive testing capabilities. This method keeps the technical documentation separate from the general API overview while making sure developers have access to up-to-date information.

5.1.1 API Organization and Resource Collections

SparkFlow follows REST common practices by using a resource-oriented design where APIs are built around key entities rather than actions. The system organizes its functionality into the two main services, referred to in architecture, that expose distinct resource collections.

The Station Service manages charging infrastructure and related operations. Its main resource collections include stations for charging point management, bookings for reservation handling, sessions for active charging processes, payments for transaction processing, statistics for operational metrics, and routes for journey planning with charging stops. The accompanying image shows an example of some endpoints available for the Booking resource.

The User Service handles user management and authentication. It exposes users as the primary resource for account management and auth for authentication operations including login, logout, and token management.

5.1.2 REST-design Principle

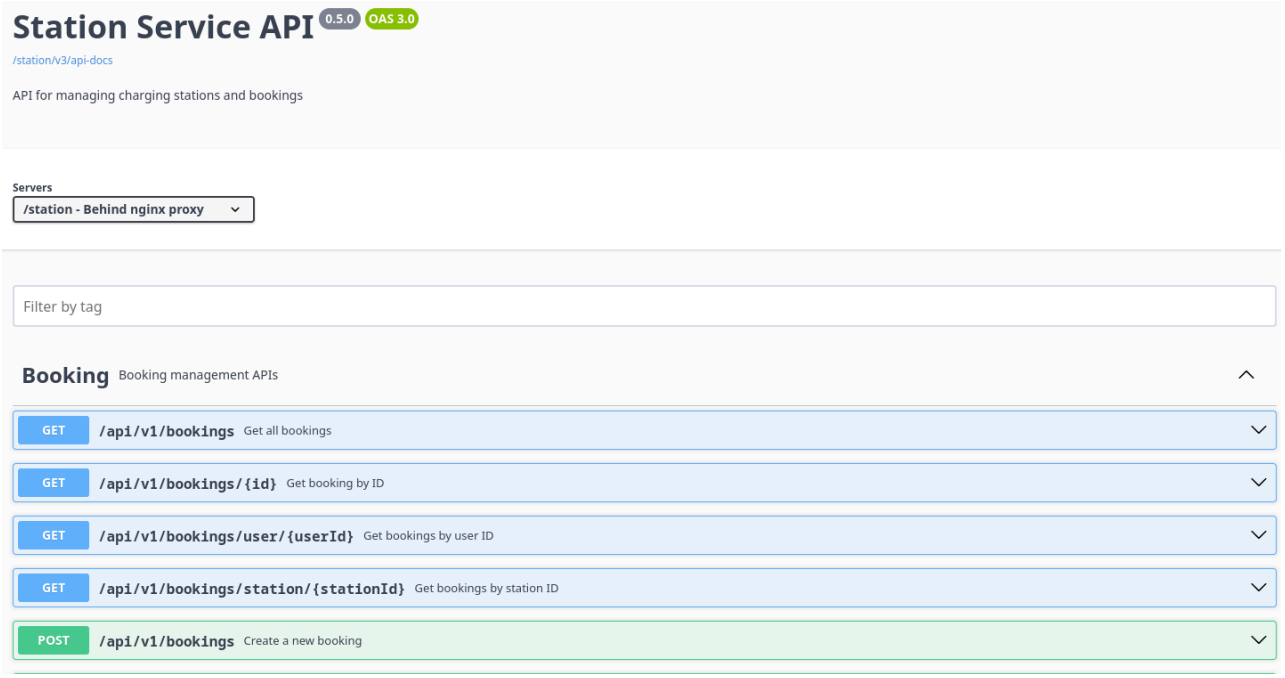
The APIs follow resource-oriented design by using URIs that represent entities rather than actions. For example, /api/stations/{id} retrieves a specific station rather than using action-based endpoints like /api/getStation. Standard HTTP methods define operations on these resources:

- GET retrieves existing resources
- POST creates new resources
- PUT updates complete resources
- DELETE removes resources

Each request operates independently and includes all necessary information, keeping stateless communication between client and server.

5.1.3 Documentation Access

Developers can access detailed method documentation through Swagger UI at http://<host>/station/swagger-ui.html for the Station Service and, similarly /user/swagger-ui.html for the User Service. These interfaces provide interactive testing, request/response schemas, authentication examples, and parameter validation details.
In the image below, we can see part of the Station Service API swagger interface, where the presence of the Booking entity with some of its GET and POST endpoints is evident.



5.1.4 Authentication

The system uses JWT-based authentication where clients include Bearer tokens in the Authorization header. Tokens are obtained by posting credentials to /api/auth/login and must be included in subsequent requests, through “Authorization” HTTP header with “Bearer ” prefix, to protected resources.