Software Requirements Specification

Wyatt Cosby

Sep 21, 2024

1 Introduction and Overview

The software system being specified is the BeAvis car rental service. As the name suggests, this system is designed to allow customers to rent cars through either an app or a website. The system is needed to make the car rental process easier for customers while also reducing the workload for employees. This document provides a detailed overview of the system's requirements. It includes user requirements, which describe how the system will function from a non-technical perspective, as well as details on the hardware being used, the look of the UI, how users will interact with the system, and any software constraints. Additionally, this document covers both the functional and non-functional requirements of the system.

2 User Requirements

Customer Requirements:

Customers will interact with the system to find and rent vehicles. They will start by selecting their pickup and drop-off locations from a list sorted by distance from their given location. A form will allow them to choose pickup and drop-off dates and times. Once this information is entered, a list of available vehicles will be displayed, with filters for vehicle type, price range (low-to-high or high-to-low), number of seats, and luggage space. After selecting a vehicle, customers can choose from various insurance options and accessibility features such as booster seats, steering wheel spinner knobs, and toll passes. They will then provide their personal and payment information to complete the booking process.

The UI for customers should be clean and user-friendly, featuring a prominent search bar on the homepage and a simple navigation bar with options like "Home," "Book a Car," "About Us," and "Contact." The design will be minimalist, focusing on functionality with clear fonts and high-quality car images.

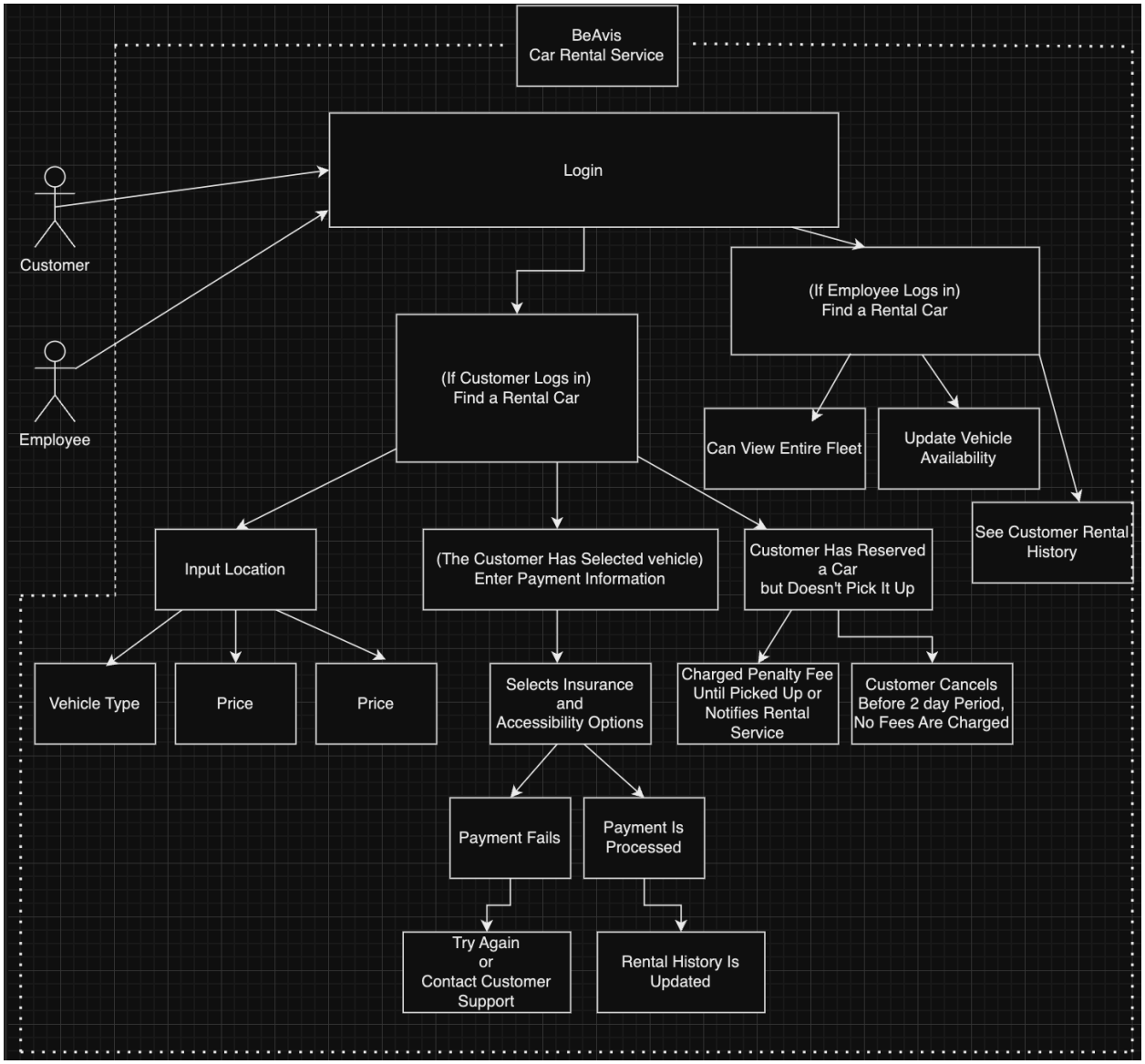
Employee Requirements:

Employees will need to manage and update the vehicle fleet and view customer rental histories. They will have access to a section where they can check the availability of vehicles, update their status, and see detailed rental history for customers. The employee interface should be straightforward and efficient, allowing quick access to fleet management tools and customer information without unnecessary complexity.

3 System Requirements

The BeAvisCar rental service system will have two types of users: customers and employees. Customers will interact with the system to search for, select, and reserve rental cars, while employees will manage the fleet and view customer rental histories. The system will be available on both web and app platforms, supporting multiple languages.

Use Case Diagram



3.2 Non-functional Requirements

The BeAvisCar system needs to be secure by encrypting important info like passwords and payment details when it's being sent or stored. We'll keep sensitive stuff in different databases to reduce the risk if there's a data breach. There's also an option for two-factor authentication for extra security, and payment info will follow industry standards. In terms of performance, the site should load fast — things like searching for cars should take less than 3 seconds. It also needs to be able to handle a lot of users at once, especially during busy times, and support at least 500 users without slowing down.

The system should be reliable, meaning it should be up and running 99.9% of the time. Daily backups will keep the data safe, and the system should recover in under an hour if something goes wrong. It should also show clear error messages when there's an issue. Usability is important too, so the design will be simple and easy to use. The system will support multiple languages like English, Spanish, Arabic, and Mandarin, and it'll work well on both mobile and desktop devices. Users shouldn't need a lot of training to figure out how to use it.

The system also needs to work across different platforms, including iOS, Android, and web. It should get yearly updates to keep it running smoothly with newer operating systems. For maintenance, the system should be built in a way that allows individual parts to be updated without messing up everything else. There should be solid documentation for developers, and updates and bug fixes should happen every few months.

As the user base grows, the system needs to scale by adding more servers when needed. It should also be able to use cloud services to handle extra traffic during peak times. Lastly, the system will follow legal guidelines like GDPR for international users, and even though accessibility isn't required, we'll still try to follow best practices for web accessibility.

4 Other

Risks: There are a few risks to keep in mind. Data security breaches are a concern, even with encryption and secure data storage, so regular security checks will be important. System downtime is another risk, which could interrupt service. To handle this, we'll need strong backup systems and a solid recovery plan. As the user base grows, performance might be an issue if the system isn't scaled properly, so keeping an eye on server capacity and adjusting as needed will be crucial.

Constraints: The project has a $250,000 budget, which might limit some features or technologies we can use. We also need to make sure the system works with current technologies like iOS, Android, and web browsers, which could restrict us from using the latest tech.

Assumptions: We're assuming that users will follow policies like showing ID at the store and paying for damages, even though this isn't handled directly through the system. We also assume that platforms like iOS and Android will keep supporting the tech we use for the system.

Potential Future Changes: Looking ahead, we might add new features, like GPS navigation integration or advanced vehicle analytics. We might also include more languages if needed and potentially improve accessibility features in future updates.