Car Rental System

SSE 657

Erin Cargin Dylan DeVries
Jarod Miller Joel Seepersaud

1. Scope

1.1. Identification

This project involves creating an application for *Car Rental Company* that operates as a car rental system to reduce the company's operation cost. The car rental system will include a system for customers of *Car Rental Company* to create new reservations and view their current vehicle reservations, as well as a database to store all customers, reservations, and vehicles that *Car Rental Company* handles.

1.2. Overview

The Car Rental System application is a tool that centralizes inventory and booking organization for a car rental company. The application is created based on the company's given requirements including some improvement proposals.

2. Requirements

2.1. Application Requirements

2.1.1. Functional Application Requirements

2.1.1.1. R01: LOGIN
The application shall allow the customer to login with an account connected to *Car Rental Company*.

2.1.1.2. R01: VIEW_CURRENT_RESERVATIONS
The application shall allow the customer to view a “Current Reservations” screen, which shall display a list of reservations belonging to the customer, or a “No Current Reservations” message if the customer has not made any reservations.

2.1.1.3. R03: CREATE_NEW_RESERVATIONS
The application shall allow the customer to create new reservation queries by collecting information from the user such as start and end date, number of passengers, pickup and return location, and make/model of vehicle.

2.1.1.4. R04: BOOK_NEW_RESERVATIONS
The application shall allow the customer to book a new reservation from their list of valid reservations returned by the reservation query, which will then update the *Car Rental Company*’s vehicle database with the reservations dates, locations, and other information.

2.1.2 Non-functional Application Requirements

2.1.2.1. APPLICATION_POWER_UP
Upon power up, the application shall display a login screen with fields for a username and password.

2.1.2.2. VERIFY_LOGIN_DATA_FILLED
Upon user selection of the “Login” button, the application shall validate that a password and username has been inputted.

2.1.2.3. SEND_VALIDATION_REQUEST
Upon user selection of the “Login” button, the application shall send a validation_request message to the database.

2.1.2.4. LOGIN_SUCCESS
Following receipt of a successful “login_response” message, the application shall display the home screen.
2.1.2.5. LOGIN_FAILURE
Following receipt of an unsuccessful “login_response” message, the application shall display an error message and highlight the error fields in red.

2.1.2.6. DISPLAY_UPCOMING_RESERVATIONS
Upon displaying the home screen, the application shall display the user’s current/upcoming car reservations or “No reservations”.

2.1.2.7. NEW_RESERVATION_FIELDS
Upon user selection of the “Create New Reservation” button, the application shall display a date range selection field, number of passengers selection field, car make/model selection field, and pick-up/return location fields.

2.1.2.8. DISPLAY_PRICE
Upon user selection of the “Check Availability” button, the application shall display the total price of any available reservations, including all applicable discounts and taxes.

2.1.2.9. SEND_BOOKING_VALIDATION
Upon user selection of a valid reservation and of the “Book Reservation” button, the application shall send a “booking_validation” message to the database.

2.1.2.10. BOOKING_RESPONSE_SUCCESS
Following receipt of a successful “booking_resonse” message from the database, the application shall inform the user and display the booking in the "Upcoming Bookings" section.

2.1.2.11. BOOKING_RESPONSE_FAILURE
Following receipt of an unsuccessful “booking_resonse” message, the application shall display the error and highlight the errored fields in red (such as no cars available at this location).

2.1.2.12. ENCRYPTED_PASSWORD
The application shall not transmit a plain-text password.

2.2. Application Specifications

2.2.1. Vehicles can be taken from one location and can be returned to the same or different location with an additional cost.

2.2.2. The application shall accept mainly passenger cars, but future work may include making the system expandable to other types of vehicle rental.

2.2.3. The application shall be able to handle at least 10 different makes of vehicles from different manufacturers, and each make may have at least 3 models (Toyota has Corolla, Camry, and more). Models are grouped into a small number of price classes.

2.2.4. Rental prices may be different for different user input options, in which customers should be informed when reserving a car.

2.3. Database Requirements

2.3.1. Functional Database Requirements

2.3.1.1. R05: ADD_NEW_VEHICLE_INFORMATION
The database shall allow a *Car Rental Company* employee to add a new vehicle with its information, such as make, model, year, mileage, pricing, etc.

2.3.1.2. R06: REMOVE_CURRENT_VEHICLE_INFORMATION
The database shall allow a *Car Rental Company* employee to remove any existing vehicle that may not be in use anymore.

2.3.1.3. R07: UPDATE_CURRENT_VEHICLE_INFORMATION
The database shall allow a *Car Rental Company* employee to update any current vehicle information such as mileage and pricing that may change overtime.

2.3.2. Non-functional Database Requirements

2.3.2.1. DATABASE_TYPE
The database will use PostgreSQL.

2.3.2.2. RESPONSE_TIMING
Following receipt of a query, the database must respond within 5 seconds or return an error message.

2.3.2.3. STORE_ENCRYPTED_PASSWORD
The database shall not store a plain-text password.