Car Rental Agency Operations Simulation

INFO263-Web Design and Development

Group number: 13

Group members: Jordan Withell (67307220)

Xanthe Waldon (26978489)

Shaohua Zang (58583660)

# Introduction

This project focuses on developing a web-based front-end for a car rental agency simulation. The application provides an interface for users to view and interact with data generated by the simulation, which includes real-world geospatial, physical, and financial data. The goal is to enable users to access key information such as vehicle trips, maintenance records, relocations, and financial summaries in a user-friendly manner. Additionally, the frontend allows users to modify financial data and view the effects of these changes through visualizations.

This report outlines the structure of the solution, its components and features, and provides instructions for running and testing the application.

# Solution Structure

The project is structured as a React application, with key components organized across several directories. Below is an overview of the solution's structure (blue text indicate file names):

App.js: The central file that defines the application's routes and page structure​.

src/components: Contains reusable components such as navigation bars, lists, maps, and charts.

src/pages: Manages the individual pages of the application, such as the home page, vehicle pages, relocations, and trips.

src/utils: Holds utility functions and assets (such as the city data in nz.json​) that are essential for map and data operations.

App.css and index.css: CSS files for styling the main layout and specific components of the application​.

The application routes are set up in App.js, using React Router to define paths for various pages, such as vehicles, trips, relocations, and charts​. This ensures that each section of the application has a dedicated route, and users can navigate easily between pages.

# Components and Features

### Home Page:

File name: Home.js

Functionality: Displays a summary of car rental data, such as the number of completed trips, upgrades, refused bookings, and vehicles relocated or serviced. It also features the vehicle type list component, which shows vehicle categories and prices.

### Navigation Bar:

File name: AppBar.js

Functionality：The navigation bar provides links to the home page, vehicles, relocations, trips, and charts. It ensures users can switch between different sections of the application seamlessly.

### Vehicle Pages：

File name: Vehicles.js, Vehicle.js, VehicleList.js, and VehicleListObj.js

Functionality: List all vehicles in the system, displaying essential information like registration number, category, and odometer readings. Pagination is implemented to handle large datasets. They also provide detailed information for a specific vehicle, including trip history, maintenance records, and relocations.

### Relocation Pages:

File name: Relocations.js, Relocation.js, RelocationList.js, and RelocationListObj.js

Functionality: Display a list of vehicle relocations, including details such as origin, destination, and distance. Also provide detailed information about a specific relocation, along with a map showing the origin and destination using Map.js.

### Trip Pages:

File name: Trips.js, Trip.js, TripsList.js, and TripsListObj.js

Functionality: List all vehicle trips, showing details like registration number, origin, destination, and trip distance. Provide detailed information about a specific trip, including a map that shows the route from origin to destination.

### Charts and Data Visualization:

File name: Charts.js

Functionality: Displays interactive charts, including quarterly data and daily location traffic. The charts are rendered using the Chart.js library, with data fetched from the backend to create visual representations of important metrics such as vehicle usage and trip revenue.

### 404 Page:

File name: NotFound.js

Functionality: Displays a simple 404 "Not Found" page for invalid URLs, ensuring smooth user experience even for incorrect paths.

## **Advanced Features**

### Login System Implementation：

File name: Login.js

Functionality: The login page allows users to authenticate using their email and password. The interface includes input validation and an option for password recovery via a **Forgot Password** link.

Key features include:

**user input validation:** Email and password inputs are validated to ensure correct formatting. And If inputs are invalid, error messages are displayed dynamically.

**Authentication Request**: Upon successful input validation, the login data is sent to the backend via an API call to UserModal.php. If the login is successful, a token is stored in the session to manage user authentication status.

**Session Management**:

* On successful login, an authentication token is saved in **sessionStorage** to maintain the logged-in state.
* The **useNavigate** hook from React Router is used to redirect users to the homepage upon successful login.

**Password Recovery**: A link to the **Forgot Password** feature opens a dialog where users can reset their password if forgotten.

# How to run

To run the project locally, follow these steps:

1 Install Node.js: Ensure that Node.js is installed on your machine.

2 Clone the repository: Download the project files or extract them from the provided ZIP.

3 Navigate to the project folder and run: “npm install”

4 Place the provided RentalSimV8Logging.2024-08-09.db database file in the root directory and configure the database connection in PHPStorm or your preferred IDE.

5 Start the development server with:

“npm start”

The application will be accessible at http://localhost:3000.

# Testing

Once on ‘localhost:3000’, you can use the appBar at the top to navigate or press on a car type to search for those cars.

An account is already made for you, press login and type in your UC email address, press ‘forgot your password’ to set your password for the first time, because passwords are hashed, the server needs to set the password.

To validate the functionality of the web application:

1. **Home Page**: Navigate to the home page (/) and ensure that key metrics (trips completed, vehicles relocated, etc.) are displayed correctly.
2. **Vehicle Pages**:
   * Test the vehicle list page (/vehicles) and ensure that vehicles are displayed with pagination.
   * Select a vehicle to view its detailed information, including trip and maintenance records.
3. **Relocation Pages**:
   * On the relocation list page (/relocations), verify that relocations are displayed and paginated correctly.
   * Click on a relocation to view detailed information and the map visualization of the route.
4. **Trip Pages**:
   * Test the trips list page (/trips) and verify that trip data is displayed and paginated.
   * Click on a trip to view detailed information and the route on the map.
5. **Charts**: Navigate to the charts page (/charts) and ensure that the quarterly and daily location data is visualized correctly using charts.
6. **404 Page**: Enter an invalid URL to verify that the 404 page is displayed.

# Conclusion

The project successfully implements a front-end solution for interacting with car rental simulation data. Core features such as navigation, real-time data fetching, search/filtering, pagination, and data visualization are well integrated using React components. The application is built with scalability in mind, ensuring that the user interface remains responsive even with large datasets.

Future enhancements could include additional data visualization features and more advanced filtering options for users to explore data in greater detail.