Car Rental Management System(CRMS)

Codebase and API

The Car Rental Management System (CRMS) provides a comprehensive set of functionalities for managing car rentals. The system is implemented in Java using Spring Boot for the backend and React for the frontend. Below are the documented API endpoints and the corresponding models used in the system.

Table of Content

File Descriptions

Car Model

Represents a car in the system.

Fields:
- car_ID: Unique identifier for the car.
- make: Make of the car.
- model: Model of the car.
- years: Year of manufacture.
- color: Color of the car.
- image: Image of the car.
- mileage: Mileage of the car.
- rental_price_per_day: Rental price per day.

- car_status: Status of the car (e.g., Available, Rented).
- maintenance: Maintenance history of the car.
- rental: Rental history of the car.
- insurance: Insurance details of the car.

Customer Model

Represents a customer in the system.

Fields:
- customer_ID: Unique identifier for the customer.
- first_name: First name of the customer.
- last_name: Last name of the customer.
- contact_information: Contact information of the customer.
- email: Email address of the customer.
- address: Address of the customer.
- driving_license_number: Driving license number of the customer.
- rental: Rental history of the customer.

Employee Model

Represents an employee in the system.

Fields:
- employee_ID: Unique identifier for the employee.
- first_name: First name of the employee.
- last_name: Last name of the employee.
- telephoneNumber: Telephone number of the employee.
- position: Position of the employee.
- rentals: Rental history of the employee.
- location: Location where the employee is stationed.
- maintenances: Maintenance tasks performed by the employee.

Insurance Model

Represents insurance details for cars in the system.

Fields:
- insurance_ID: Unique identifier for the insurance.
- coverage_type: Type of coverage.
- insurance_company: Insurance company providing coverage.
- premium_amount: Premium amount.
- car: Car associated with this insurance.

Location Model

Represents rental locations in the system.

Fields:
- location_ID: Unique identifier for the location.
- address: Address of the location.
- contact_information: Contact information of the location.
- operating_hours: Operating hours of the location.
- employees: Employees stationed at the location.

Maintenance Model

Represents maintenance records for cars in the system.

Fields:
- maintenance_ID: Unique identifier for the maintenance record.
- maintenance_type: Type of maintenance performed.
- maintenance_date: Date when maintenance was performed.
- car: Car for which maintenance was performed.
- maintenance_cost: Cost of maintenance.
- employee: Employee who performed the maintenance.

Payment Model

Represents payment details for rentals in the system.

Fields:
- payment_ID: Unique identifier for the payment.
- payment_date: Date of the payment.
- payment_method: Payment method used.
- rental: Rental associated with this payment.
- amount_paid: Amount paid.

User Model

Represents user details for the system.

Fields:
- userId: Unique identifier for the user.
- firstName: First name of the user.
- lastName: Last name of the user.

- email: Email address of the user.
- position: Position of the user.

Repository Interfaces

These interfaces define the methods for interacting
Repository Interfaces

These interfaces define the methods for interacting with the database tables.

- CarRepository: Interface for CRUD operations on Car entities.
- CustomerRepository: Interface for CRUD operations on Customer entities.
- EmployeeRepository: Interface for CRUD operations on Employee entities.
- InsuranceRepository: Interface for CRUD operations on Insurance entities.
- LocationRepository: Interface for CRUD operations on Location entities.
- MaintenanceRepository: Interface for CRUD operations on Maintenance entities.
- PaymentRepository: Interface for CRUD operations on Payment entities.
- RentalRepository: Interface for CRUD operations on Rental entities.
- UserRepository: Interface for CRUD operations on User entities.

API Documentation

CarController

Handles operations related to cars in the system.

Endpoints:
- POST /cars/save: Save a new car to the system.
- GET /cars/all: Retrieve all cars from the system.
- GET /cars/{id}: Retrieve a car by its ID.
- PUT /cars/update/{id}: Update details of a car.
- DELETE /cars/delete/{id}: Delete a car by its ID.
- GET /cars/search/{car_id}: Search for a car by its ID.

CustomerController

Handles operations related to customers in the system.

Endpoints:
- POST /Customer/addcustomers: Add a new customer to the system.
- GET /Customer/getcustomer/{id}: Retrieve a customer by their ID.
- PUT /Customer/updatecustomer/{id}: Update details of a customer.
- DELETE /Customer/deletecustomer/{id}: Delete a customer by their ID.

- GET /Customer/getallcustomers: Retrieve all customers from the system.

EmployeeController

Handles operations related to employees in the system.

Endpoints:
- POST /employees/add: Add a new employee to the system.
- GET /employees/get/{id}: Retrieve an employee by their ID.
- PUT /employees/update/{id}: Update details of an employee.
- DELETE /employees/delete/{id}: Delete an employee by their ID.
- GET /employees/getall: Retrieve all employees from the system.

InsuranceController

Handles operations related to insurance in the system.

Endpoints:
- POST /insurance/add: Add insurance details for cars in the system.
- GET /insurance/get/{id}: Retrieve insurance details by ID.
- PUT /insurance/update/{id}: Update insurance details.
- DELETE /insurance/delete/{id}: Delete insurance details.
- GET /insurance/getall: Retrieve all insurance details.

LocationController

Handles operations related to rental locations in the system.

Endpoints:
- POST /locations/add: Add a new rental location to the system.
- GET /locations/get/{id}: Retrieve a rental location by ID.
- PUT /locations/update/{id}: Update details of a rental location.
- DELETE /locations/delete/{id}: Delete a rental location by ID.
- GET /locations/getall: Retrieve all rental locations from the system.

MaintenanceController

Handles operations related to maintenance records for cars in the system.

Endpoints:
- POST /maintenance/add: Add a new maintenance record to the system.
- GET /maintenance/get/{id}: Retrieve a maintenance record by ID.
- PUT /maintenance/update/{id}: Update details of a maintenance record.

- DELETE /maintenance/delete/{id}: Delete a maintenance record by ID.
- GET /maintenance/getall: Retrieve all maintenance records from the system.

PaymentController

Handles operations related to payment details for rentals in the system.

Endpoints:
- POST /payments/add: Add a new payment record to the system.
- GET /payments/get/{id}: Retrieve a payment record by ID.
- PUT /payments/update/{id}: Update details of a payment record.
- DELETE /payments/delete/{id}: Delete a payment record by ID.
- GET /payments/getall: Retrieve all payment records from the system.

RentalController

Handles operations related to rentals in the system.

Endpoints:
- POST /rentals/adds: Add a new rental to the system.
- GET /rentals/get/{id}: Retrieve a rental by its ID.
- PUT /rentals/updates/{id}: Update details of a rental.
- DELETE /rentals/delete/{id}: Delete a rental by its ID.
- GET /rentals/getall: Retrieve all rentals from the system.

UserController

Handles operations related to users in the system.

Endpoints:
- POST /users/adds: Add a new user to the system.
- GET /users/get/{id}: Retrieve a user by their ID.
- PUT /users/update/{id}: Update details of a user.
- DELETE /users/delete/{id}: Delete a user by their ID.
- GET /users/getall: Retrieve all users from the system.

Functionalities of this command interpreter

Features

- Browse and search for available cars
- Make car rentals

- rental history
- Manage car inventory, maintenance records, and insurance details
- Process payments for rentals

Environment

- Backend: Java, Spring Boot
- Frontend: React
- Database: (Hibernate(MySQL))

Installation

Prerequisites

- Java Development Kit (JDK) version 8 or later
- Node.js and npm (for running the React frontend)

Backend Setup

1. Open the folder: Car-Rent-Management-System
2. Navigate to the project directory: cd Car-Rent-Management-System
3. Build the project: ./mvn clean install
4. Run the application: ./mvn spring-boot:run

The backend server should now be running at http://localhost:8080.

Frontend Setup

1. Open the folder: Car-Rental-Management-System
2. Navigate to the project directory: cd car-rental-management-system
3. Install dependencies: npm install
4. Start the development server: npm start

The frontend application should now be running at http://localhost:3000.

API Documentation

The backend API endpoints are documented using postman and Also using thunder client. You can access at http://localhost:8080/when the backend server is running.

User manuals

Car Rental Management System (CRMS) User Manual

Welcome to the Car Rental Management System (CRMS) User Manual! This guide will walk you through the steps to create an account, log in, rent a car, and manage your rentals effectively.

1. Creating an Account

To get started with CRMS, you need to create an account by following these steps:

1. Navigate to the CRMS website and click on the "Sign Up" or "Create Account" button.
2. Fill out the registration form with your personal information, including your first name, last name, email address, and desired password.
3. Click on the "Create Account" button to submit your registration information.
4. You will receive a confirmation email to verify your email address. Click on the verification link provided in the email to activate your account.

2. Logging In

Once your account is activated, you can log in to CRMS using the following steps:

1. Navigate to the CRMS website and click on the "Log In" button.
2. Enter your registered email address and password in the login form.
3. Click on the "Log In" button to access your account.

3. Renting a Car

After logging in to CRMS, you can rent a car by following these steps:

1. From the homepage, browse through the available cars listed.
2. Click on the car you wish to rent to view more details.
3. Select the rental dates and any additional preferences (if applicable).
4. Click on the "Rent Now" or "Book Now" button to proceed with the rental.
5. Fill out the required payment information to confirm your rental.
6. Once payment is successfully processed, you will receive a confirmation email with your rental details.

4. Managing Rentals

After renting a car, you can manage your rentals by accessing your account dashboard. Here, you can:

- View your current and past rentals.

5. Contact Support

If you encounter any issues or have any questions while using CRMS, please contact our support team at (carrental7@gmail.com). We are available to assist you with any inquiries or concerns you may have.

Thank you for choosing CRMS for your car rental needs!

Bugs

No known bugs at this time.

Authors

Marius Bayizere
Umuhire Gatesi Lyse

License