Application Development Final Project Deliverable 1

18-10-2024

Matthew Macri

Danny Zhou

Car Reservation System

Scenario:

The Car Reservation System allows customers to reserve cars for rental. Users can browse available cars, select the date and time for the rental, and make a reservation. Admins can add, update, or remove cars from the system and manage reservations.

Design Paradigm:

The system will have two types of users: **Customers** and **Admins**.

- **Customers**:
    - Browse available cars.
    - Filter cars by type, brand, or availability.
    - Reserve a car for a specific time period.
    - View their current and past reservations.
- **Admins**:
    - Add new cars to the system.
    - Update or remove existing cars.
    - View all reservations.
    - Manage reservation statuses (confirm, cancel, or modify).

Expected Output:

- **For Customers**:
    - Ability to browse and filter cars.
    - Reserve a car with specific start and end dates.
    - View a list of current reservations and reservation history.
- **For Admins**:

- Ability to add, update, or remove cars.
- View and manage all reservations.

Project Features:

1. Core Classes:

- **User** (abstract):
  - Attributes: username, password, email.
  - Methods: login(), logout(), viewProfile().
- **Customer** (extends User):
  - Attributes: customerId, reservationHistory.
  - Methods: makeReservation(), viewReservations().
- **Admin** (extends User):
  - Attributes: adminId.
  - Methods: addCar(), updateCar(), removeCar(), manageReservations().
- **Car**:
  - Attributes: carId, model, brand, carType, availableFrom, availableTo.
  - Methods: isAvailable(), getCarDetails().
- **Reservation**:
  - Attributes: reservationId, car, customer, reservationStart, reservationEnd, status.
  - Methods: confirmReservation(), cancelReservation(), viewDetails().

2. Data Structures:

- **List**: To store the list of available cars.
- **List**: To store all reservations.
- LINQ can be used to filter cars based on availability, brand, or type.

3. GUI Features:

- **Login Screen**: A login system for both customers and admins.
- **Customer Dashboard**:
  - Browse cars with filtering options (brand, type, availability).

    - View current reservations and past reservation history.
- **Admin Dashboard**:
    - Add, update, or remove cars from the system.
    - View and manage all reservations.

4. Data Source:

- Use a **database** (SQLite or any relational DB) to store car information and reservations.
- **Optional**: File-based storage if a database isn't needed.

5. Internationalization (I18N and L10N):

- Support for **two languages** (e.g., English and French).
- Use **ResourceBundle** for storing translated strings.
- The UI should switch between languages based on user preference.

6. Test-Driven Development (TDD):

- Write unit tests for key methods like makeReservation(), addCar(), and isAvailable().
- Ensure the test coverage includes checking for availability conflicts and proper reservation creation.

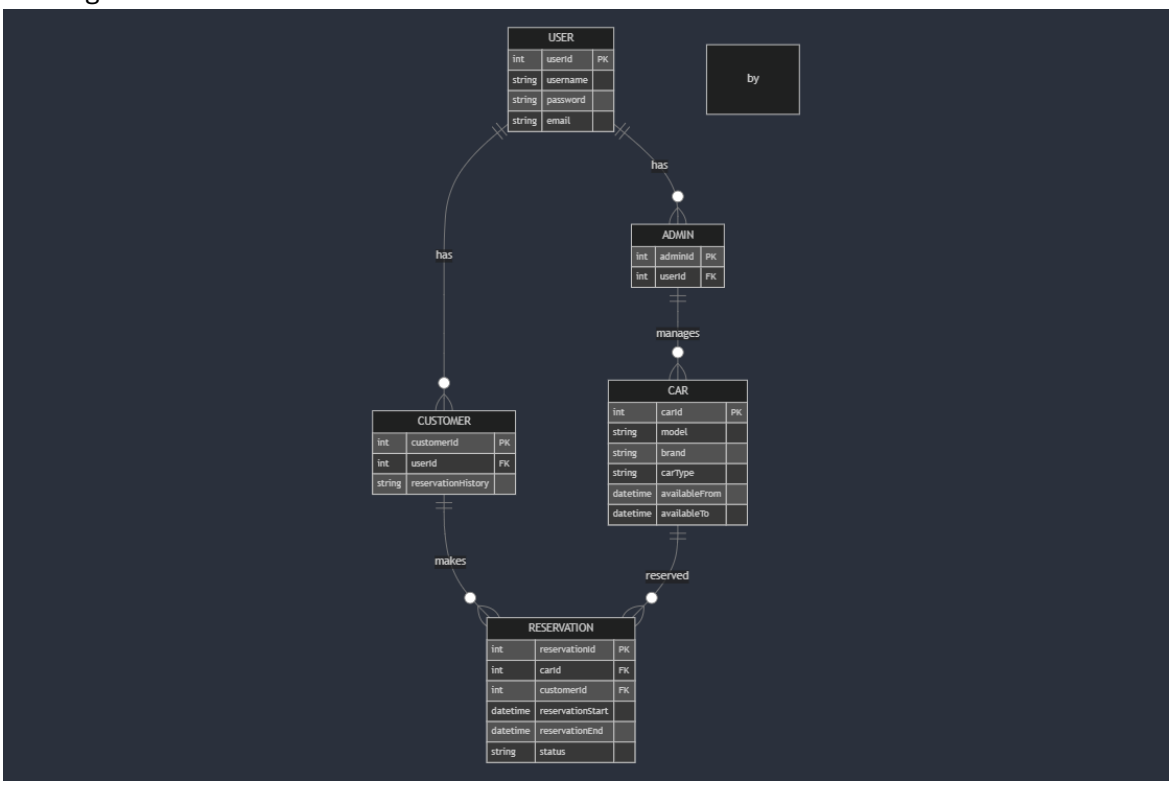
Created By

- Danny Zhou
- Matthew Macri

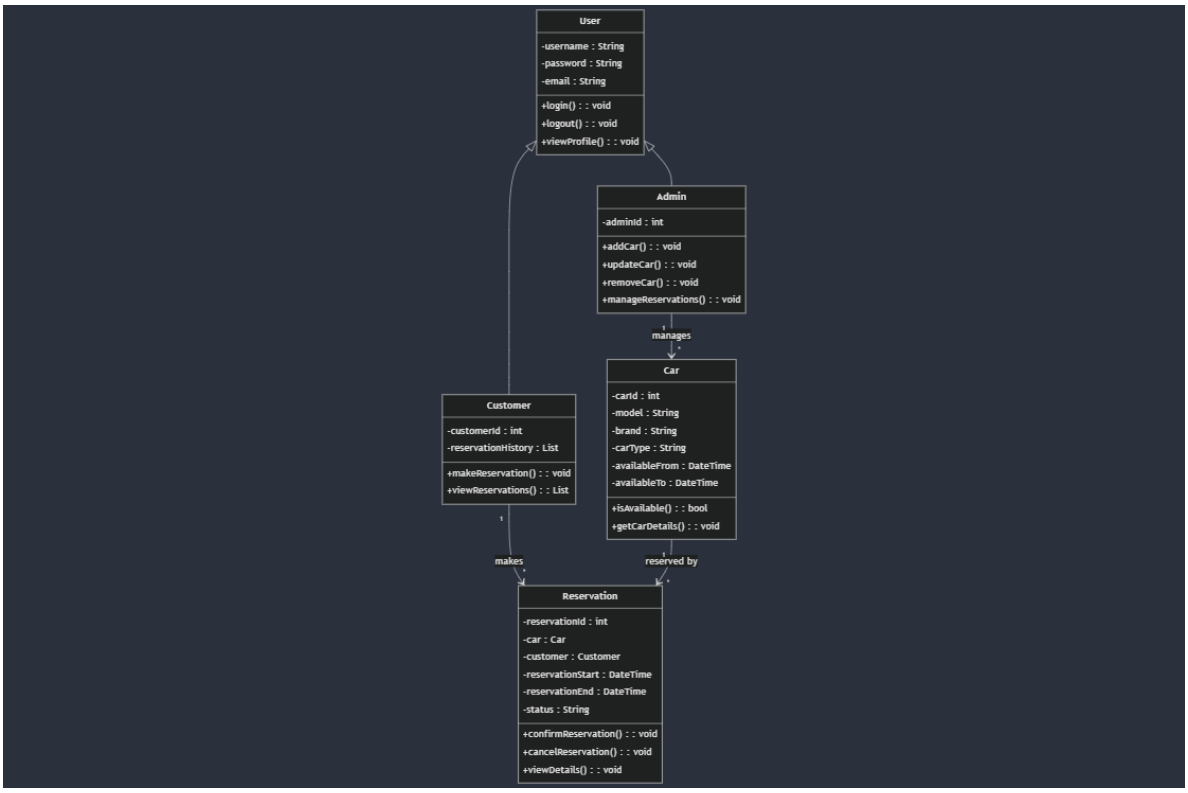
Git Repository: https://github.com/MatthewMacri/Application-Development-Project

Diagrams

ER Diagram



Class Diagram



Activity Diagram