Car Rental System – User Documentation

Overview

Version: 1.0.0 (Stable Release)

Car Rental System The purpose of the Car Rental System application is to automate the rental system.

It replaces the manual paperwork with a digital solution that enables user and car management, booking requests and rental life cycle management.

This platform has been implemented following OOP (Object-Oriented Programming), multilayer-based development and also applying Design Patterns (Repository, Service Layer, DTO's, Singleton for DB Manager).

Installation & Configuration.

The system follows **Object-Oriented Programming (OOP)** principles, **layered architecture**, and **design patterns** (Repository, Service Layer, DTOs, Singleton for DB Manager).

System Documentation
Coding Standard
Support And Maintenance

Installation & Configuration

1. Requirements

- Python 3.10+
- Virtual environment (recommended)
- Dependencies in requirements.txt

2. Setup Steps

```
# Clone the repository or extract the ZIP
git clone https://github.com/retiangson/MSE800_Car_Rental.git
cd CarRentalSystem

# Create virtual environment
python -m venv venv
source venv/bin/activate # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

3. Running the Application

You have multiple ways to run the system:

Option A: CLI Mode

```
python main.py
```

This launches the **Main Menu** in console mode.

Option B: Build as Executable

```
pyinstaller --onefile --console main.py
```

This creates a standalone executable in the dist/ folder.

Run it with:

```
dist/main.exe # On Windows
```

Option C: Run API Server (Admin Only)

From the Admin Main Menu → choose **Run API Server**, or run manually:

```
uvicorn api.main:app --reload
```

Server starts at: http://127.0.0.1:8000

API Docs:

- Swagger → http://127.0.0.1:8000/docs
- ReDoc → http://127.0.0.1:8000/redoc

Option D: Prebuilt Standalone Package

Download car_rental_standalone.zip, extract it, and run:

```
car_rental.exe
```

This version does not require Python or dependencies.

Initial Admin Access

At the first load, this system automatically creates an **Admin user** if no other users are found on initial startup of the system.

Username: adminPassword: admin

You should log in with Admin as soon as the site is up and running and create some **new users** to handle roles correctly.

Running Tests

Tests reside in the tests/ directory and use pytest.

Run all tests with:

```
pytest
```

Run tests with detailed output:

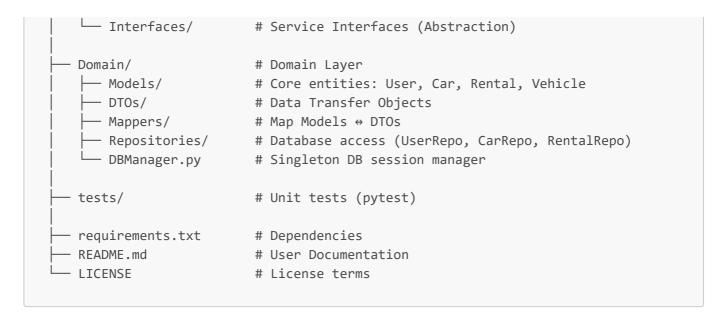
```
pytest -v
```

Run a specific test file:

```
pytest tests/test_car_service.py
```

Project Structure & File Purpose

```
MSE800_Car_Rental/
— main.py
                         # Entry point - loads Main Menu
 — ui/
                        # User Interface Layer
    ├── MainMenu.py
                       # Admin & Customer main navigation
    ├─ Car.py
                       # Add/List/Delete/Restore cars
    ── Rental.py
                       # Create/Approve/Return/Cancel rentals
    ├─ Login.py
                       # Login screen
     Customer.py
                       # Customer registration UI
    L— Employee.py
                       # Admin (user management)
  - Business/
                       # Service Layer
     — Services/
       ├─ CarService.py
         RentalService.py
         UserService.py
```



File Purposes:

- **UI** → Handles input/output and menus.
- **Business/Services** → Business logic (rules like "only available cars can be rented").
- **Domain** → Entities, DTOs, Mappers, Repositories, DB connection.
- **Tests** → Ensure reliability.
- main.py → Entry point.

Roles & Features

Admin Features

- Manage Users (register, list, soft-delete)
- Manage Cars (add, update, delete, restore, list available cars)
- Manage Rentals (approve, reject, start, cancel, return)
- Run API Server

Customer Features

- Register / Login
- Browse available cars
- Create rental & preview fees
- View rental history

System Design

- Use Case Diagram: shows Admin & Customer interactions
- **Sequence Diagram:** illustrates request → UI → Service → Database flow
- Class Diagram: represents layered architecture (UI, Services, Repositories, DTOs, Models)

License

MIT License ## License This project is released under the MIT license:

Free for personal and commercial use, but not for resale or inclusion in app / website!

No warranty provided.

See full text in LICENSE file.

Known Issues / Bugs

• UI currently runs in console only (no GUI frontend).

- Database defaults to SQLite (car_rental.db). Multi-user concurrency may need PostgreSQL or MySQL.
- Limited validation on rental dates (future enhancement).
- No payment integration yet (future feature).

Developer Credit

Car Rental System was developed by:

Name: Ronald Ephraim Tiangson

Programme: Master of Software Engineering (MSE800)

Institution: Yoobee College, New Zealand **Supervisor Name:** Mohammad Norouzifard

Date: September 2025

Contact: retiangson@gmail.com