GALGOTIAS UNIVERSITY

CAR RENTAL SYSTEM

2024-2028

DEPARTMENT OF COMPUTER SCIENCE

SUBMITTED BY-
PARV KAJLA
AARUSH PRATAP SINGH
BHVAN GANDHI
ANSHUL PANDEY

Under the guidance of-
MUKESH KUMAR JHA SIR

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to our guide, Mukesh Kumar Jha Sir, for their valuable guidance and encouragement throughout the project.

We are also thankful to our family, friends, and everyone who supported us during this project work.

- Parv Kajla(24SCSE1180173)
- Aarush Pratap Singh(24SCSE1180512)
- Bhuvan Gandhi(24SCSE1180275)
- Anshul Pandey(24SCSE1180540)

ABSTRACT

The Car Rental Management System is a Java-based mini project developed using OOPs by Java and MySQL.

This system helps to manage customers, cars, and rental records efficiently through a simple console interface.

It supports full CRUD operations (Create, Read, Delete) for managing rental business data.

This project is an academic submission designed to demonstrate the practical implementation of Java programming, JDBC,

and database integration skills.

Table of Contents

Introduction

The *Car Rental Management System* is a software application designed to simplify and automate the process of renting cars. Traditionally, car rental businesses maintain records manually, which can lead to errors, duplication, and inefficiencies. This project aims to provide a digital solution that allows car rental businesses to manage cars, customers, and rental records efficiently.

Our system is built using **Java** and **MySQL**, following a structured design with separate modules for data handling, user interface, and database operations. The system provides functionality to add, view, update, and delete car and customer records and track rental transactions in real-time. By automating these processes, the system minimizes human errors, enhances productivity, and ensures better customer service.

This project has been developed by a team of four members as part of our coursework, with each member contributing to different aspects of design, development, and testing.

This project serves as a practical application of database management, object-oriented programming, and software design principles learned during our academic course.

Project Objectives

- To develop a user-friendly **Car Rental Management System**.
- To automate the process of managing **car availability**, **customer details**, and **rental records**.
- To implement basic **CRUD operations** (Create, Read, Update, Delete) for all modules.
- To ensure data is securely stored and managed in the **MySQL database**.
- To practice **object-oriented programming** concepts in Java.
- To provide easy retrieval and updating of records to avoid manual errors.
- To separate concerns using **DAO (Data Access Objects)** and **Model classes** for clean code structure.
- To collaborate as a team and divide responsibilities among members for efficient project completion.
- To demonstrate real-world application development using **Java and SQL**.
- Designing the system with possible future GUI/Web interface in mind.

Technology Used

- Programming Language: Java (OOPs by JAva)

- Database: MySQL

- Tools: IntelliJ IDEA

- Connector: MySQL JDBC Driver

System Features

- ➢ Add/View/Delete Customers
- ➢ Add/View/Delete Cars
- ➢ Add/View/Delete Rents
- ➢ Fully connected with MySQL database

Work Division Among Team Members

❖ Review 1 Contributions

➢ Aarush Pratap Singh:

- Project Setup(JDK &IDE).
- JDBC & DO implementation.

➢ Parv Kajla:

- Defined Initial Project Structure.
- Project Structure and Module Organization.

➢ Bhuvan Gandhi:

- Data connectivity support.
- Assisted in model class design

➢ Anshul Pandey:

- Component placement in UI.
- UI alignment and visual appeal.

❖ Review 2 Contributions

➢ Parv Kajla:

- Code Quality and Innovation.
- Project Documentation.

➢ Bhuvan Gandhi

- Event handling.
- Testing and robustness.

- Aarush Pratap Singh:
    - Core Feature Implementaion
    - Error handling.

- Anshul Pandey:
    - Data Validation.
    - Supported event processing modules.

GitHub And LinkedIn Links

Parv Kajla:

LinkedIn- https://www.linkedin.com/in/parv-kajla-676b45364?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app

GitHub- parvkajla

Aarush Pratap Singh:

LinkedIn- https://www.linkedin.com/in/aarush-pratap-singh-723169365?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app

GitHub- Aarush-17

Bhuvan Gandhi:

LinkedIn- https://www.linkedin.com/in/bhuvan-gandhi-064527365

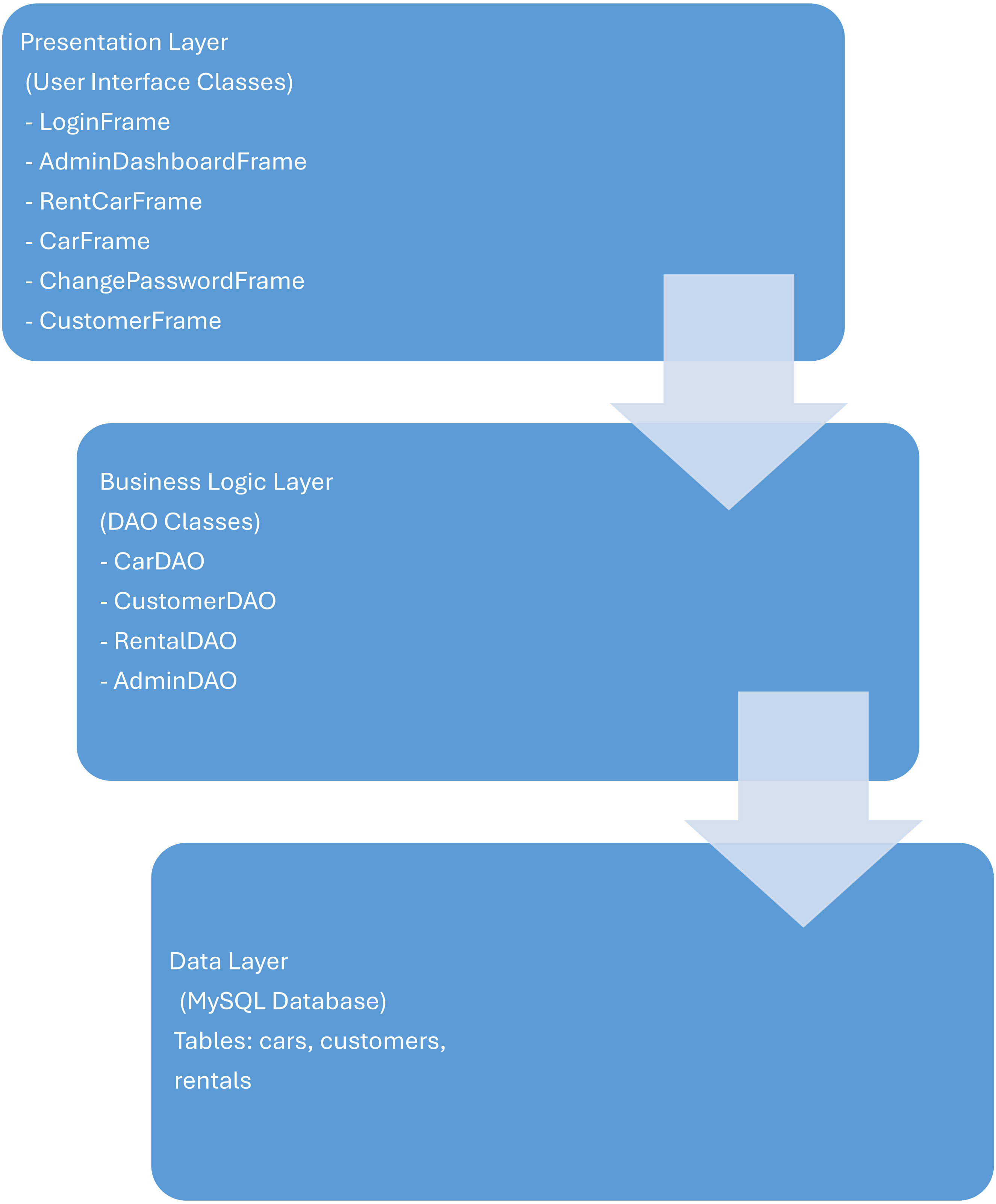
GitHub- Bhuvan-jpg

Anshul Pandey:

LinkedIn- https://www.linkedin.com/in/anshul-pandey-546167365?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=android_app

GitHub- Anshulpandey286

System Architecture



Presentation Layer

(User Interface Classes)

- LoginFrame
- AdminDashboardFrame
- RentCarFrame
- CarFrame
- ChangePasswordFrame
- CustomerFrame

Business Logic Layer

(DAO Classes)

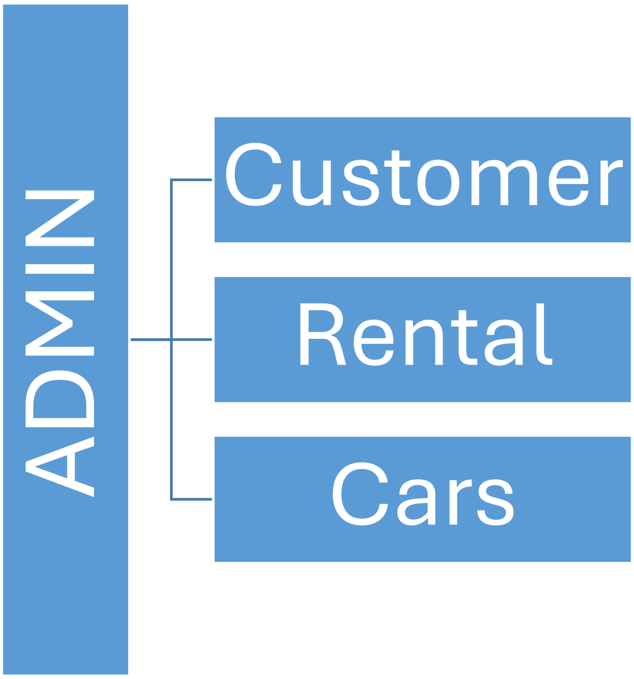
- CarDAO
- CustomerDAO
- RentalDAO
- AdminDAO

Data Layer

(MySQL Database)

Tables: cars, customers, rentals

Database Design



Cars

- ID
- Brand
- Model

Rental

- Cus_ID
- Rent Date
- Return Date

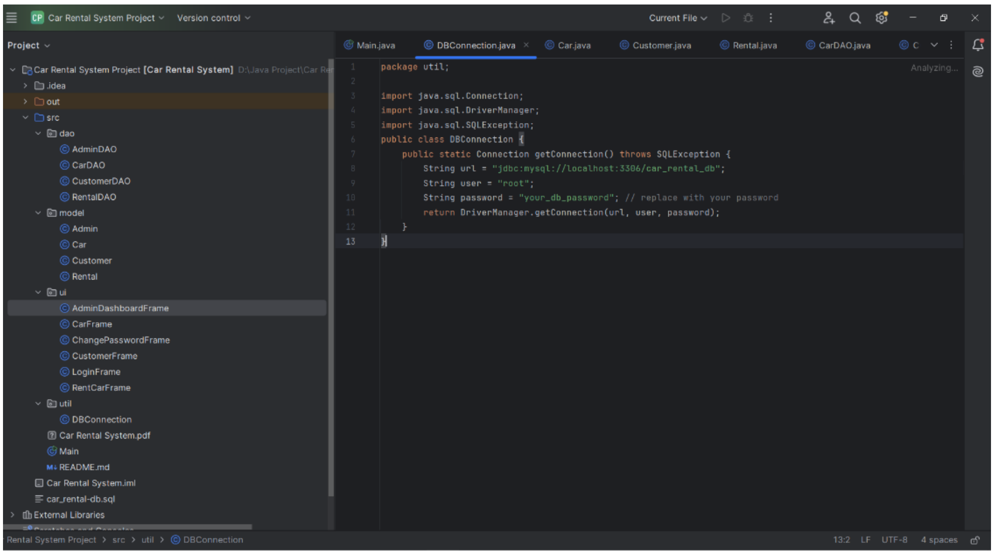
Customer

- Name
- Id
- Number

Code Explanation

❖ DBConnection.java

- This class provides a method to **connect to MySQL database**.
- Uses JDBC to establish connection.
- Connection string includes url, username, and password.

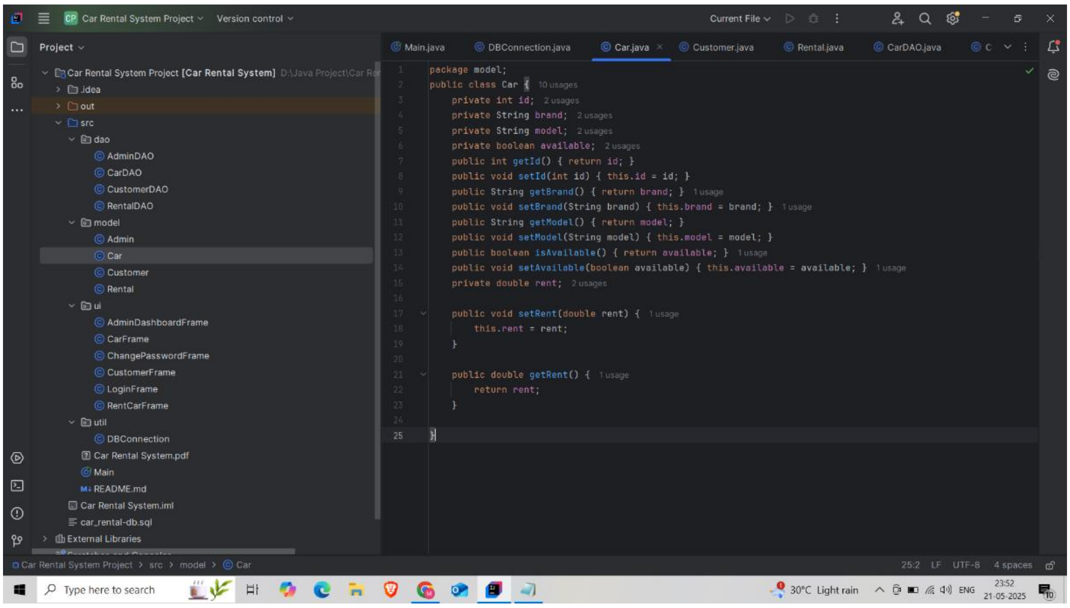


```java
package util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBConnection {
    public static Connection getConnection() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/car_rental_db";
        String user = "root";
        String password = "your_db_password"; // replace with your password
        return DriverManager.getConnection(url, user, password);
    }
}
```

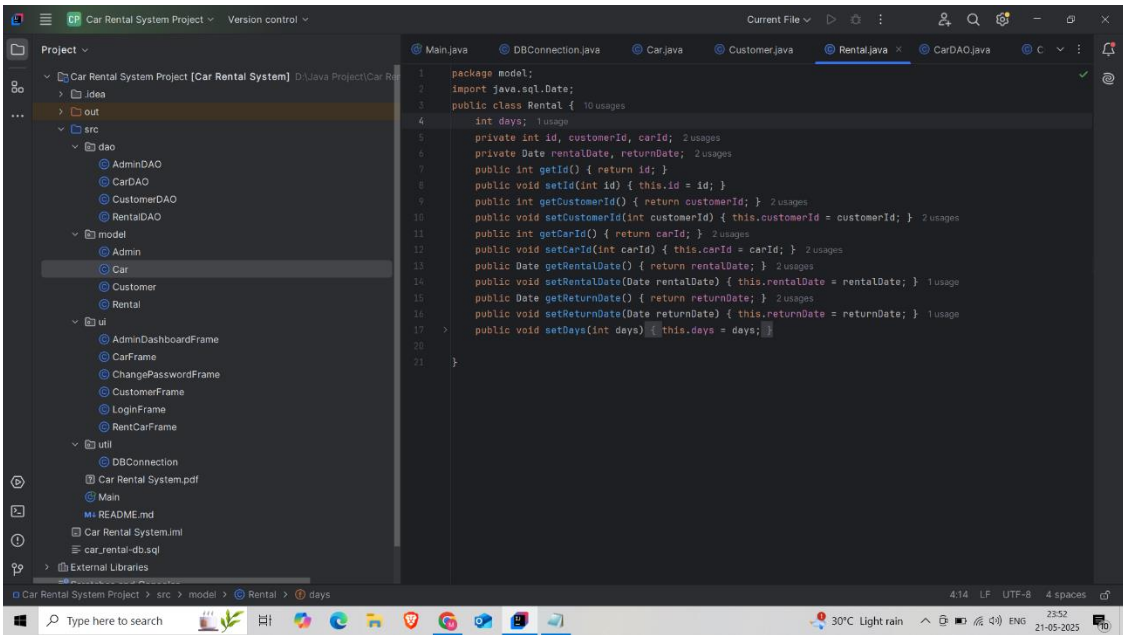
❖ Model Classes

- **Car** → defines car properties: id, brand, model, available.
- **Admin** → Defines admin properties: id, username, password. Used for login authentication and secure access to admin dashboard functionalities.
- **Customer** → defines customer properties: id, name, email, phone.

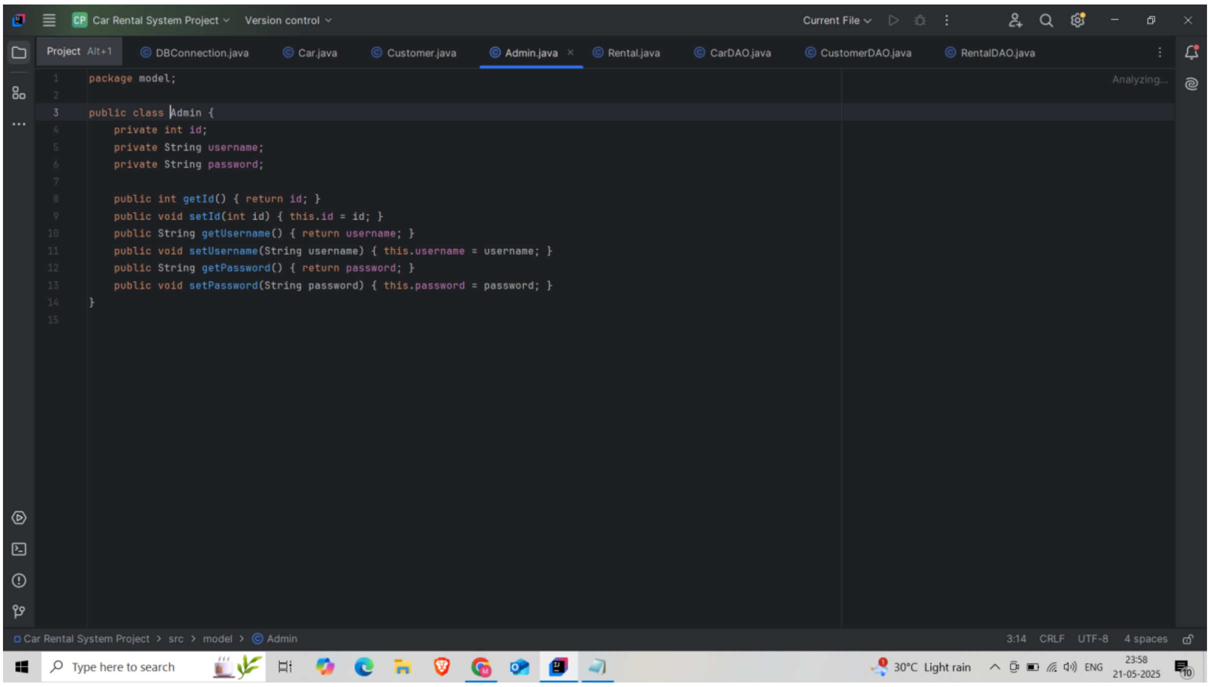
- **Rental.** → defines rental record: id, customerId, carId, rentalDate, returnDate.



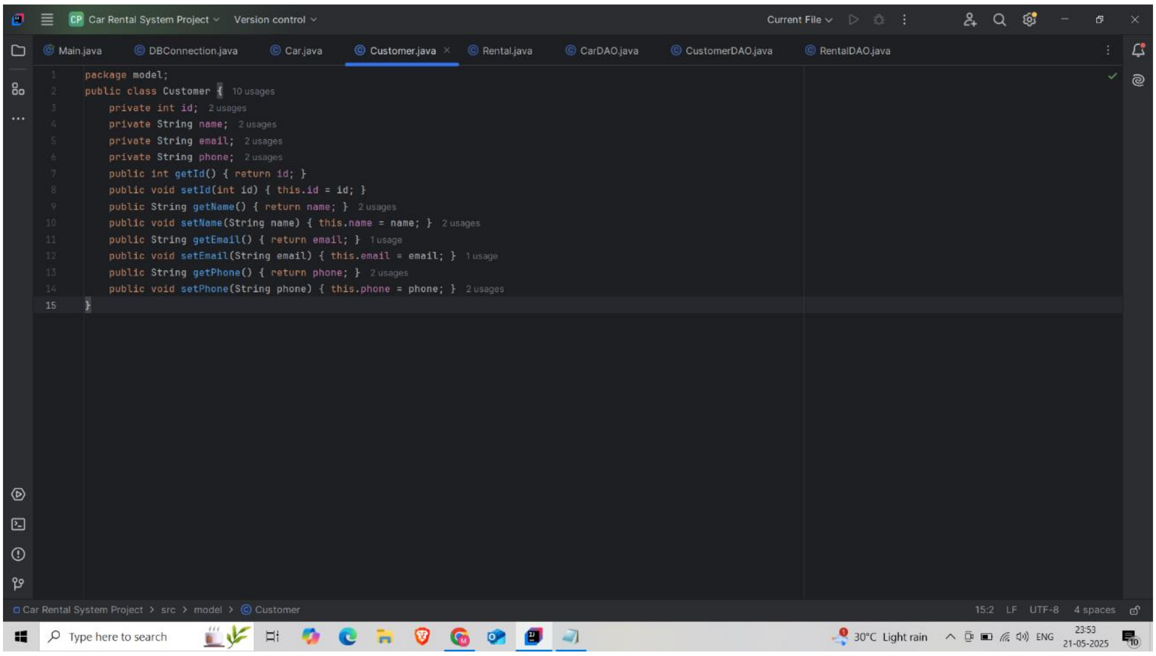
```java
package model;
public class Car {
    private int id;
    private String brand;
    private String model;
    private boolean available;
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getBrand() { return brand; }
    public void setBrand(String brand) { this.brand = brand; }
    public String getModel() { return model; }
    public void setModel(String model) { this.model = model; }
    public boolean isAvailable() { return available; }
    public void setAvailable(boolean available) { this.available = available; }
    private double rent;

    public void setRent(double rent) {
        this.rent = rent;
    }

    public double getRent() {
        return rent;
    }

}
```



```java
package model;
import java.sql.Date;
public class Rental {
    int days;
    private int id, customerId, carId;
    private Date rentalDate, returnDate;
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public int getCustomerId() { return customerId; }
    public void setCustomerId(int customerId) { this.customerId = customerId; }
    public int getCarId() { return carId; }
    public void setCarId(int carId) { this.carId = carId; }
    public Date getRentalDate() { return rentalDate; }
    public void setRentalDate(Date rentalDate) { this.rentalDate = rentalDate; }
    public Date getReturnDate() { return returnDate; }
    public void setReturnDate(Date returnDate) { this.returnDate = returnDate; }
    public void setDays(int days) { this.days = days; }

}
```



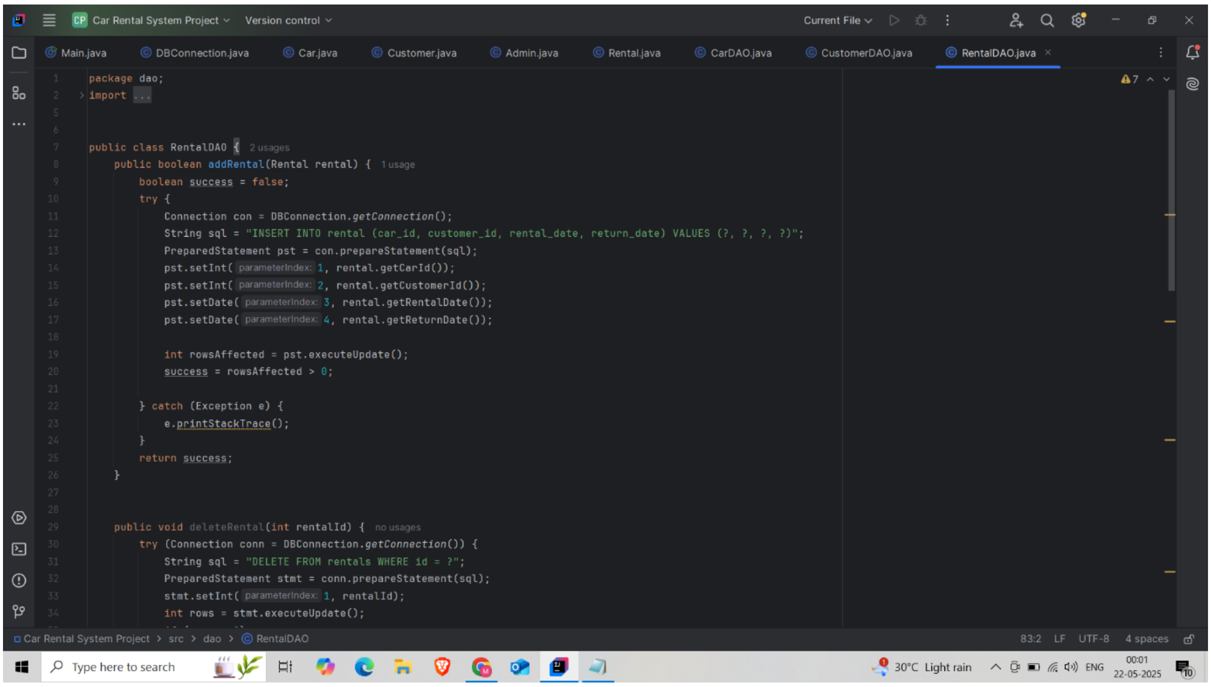
```java
package model;

public class Admin {
    private int id;
    private String username;
    private String password;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}

```



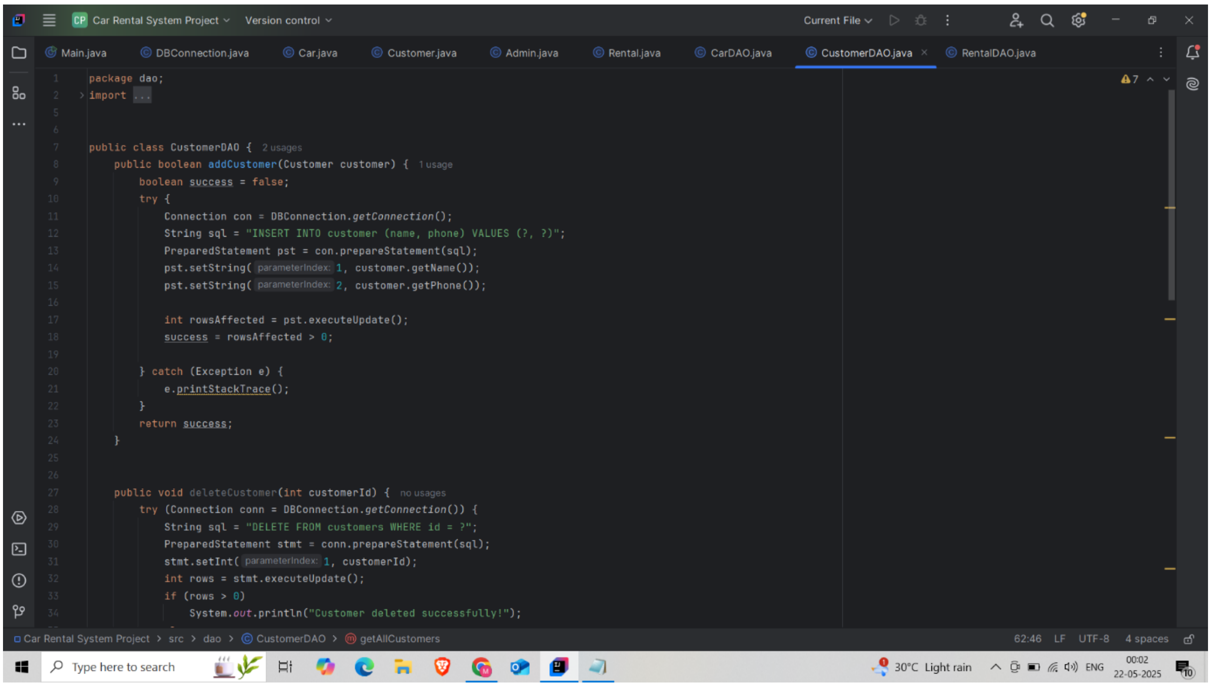
```java
package model;
public class Customer {  10 usages
    private int id;  2 usages
    private String name;  2 usages
    private String email;  2 usages
    private String phone;  2 usages
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }  2 usages
    public void setName(String name) { this.name = name; }  2 usages
    public String getEmail() { return email; }  1 usage
    public void setEmail(String email) { this.email = email; }  1 usage
    public String getPhone() { return phone; }  2 usages
    public void setPhone(String phone) { this.phone = phone; }  2 usages
}
```

❖ DAO (Data Access Object) Classes

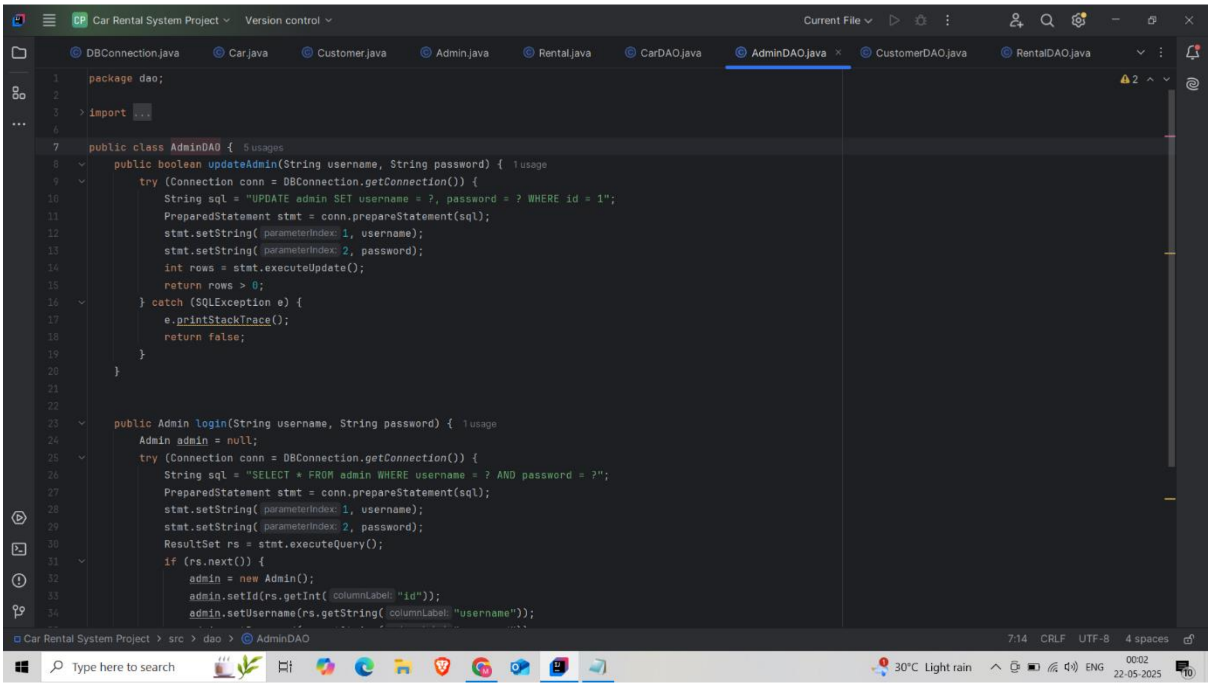
- CarDAO
- CustomerDAO
- RentalDAO
- AdminDAO



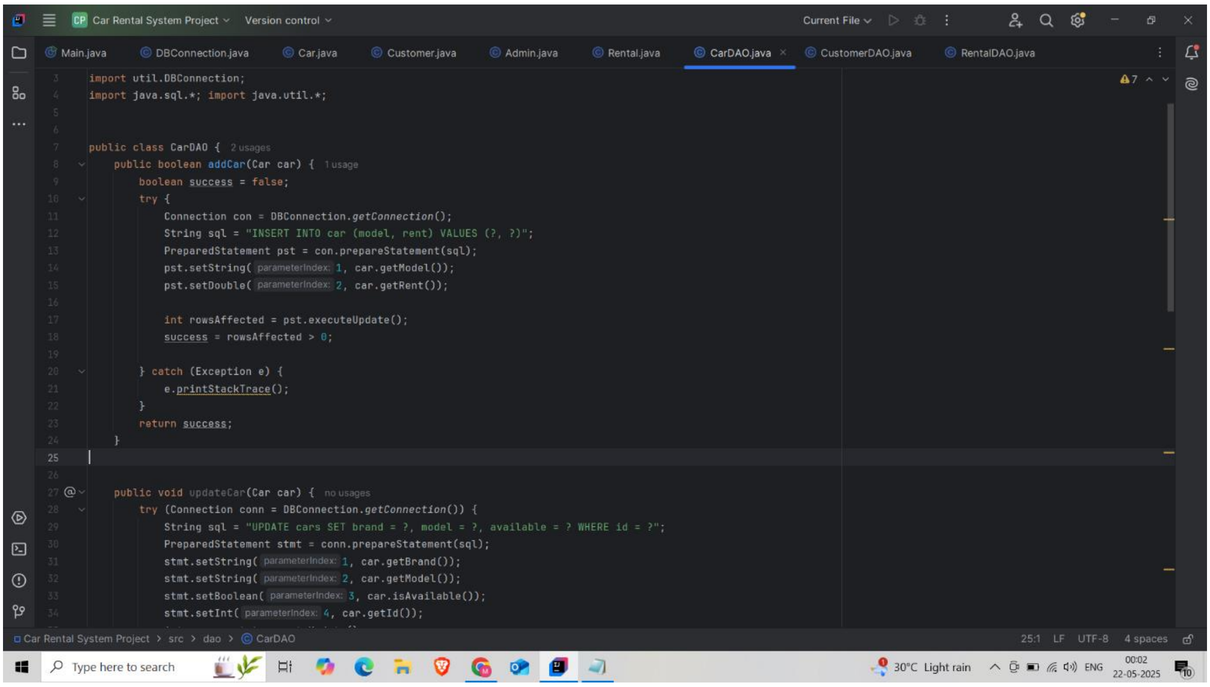
```
package dao;
import ...


public class RentalDAO {  2 usages
    public boolean addRental(Rental rental) {  1 usage
        boolean success = false;
        try {
            Connection con = DBConnection.getConnection();
            String sql = "INSERT INTO rental (car_id, customer_id, rental_date, return_date) VALUES (?, ?, ?, ?)";
            PreparedStatement pst = con.prepareStatement(sql);
            pst.setInt( parameterIndex: 1, rental.getCarId());
            pst.setInt( parameterIndex: 2, rental.getCustomerId());
            pst.setDate( parameterIndex: 3, rental.getRentalDate());
            pst.setDate( parameterIndex: 4, rental.getReturnDate());

            int rowsAffected = pst.executeUpdate();
            success = rowsAffected > 0;

        } catch (Exception e) {
            e.printStackTrace();
        }
        return success;
    }


    public void deleteRental(int rentalId) {  no usages
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "DELETE FROM rentals WHERE id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt( parameterIndex: 1, rentalId);
            int rows = stmt.executeUpdate();
```



```
package dao;
import ...


public class CustomerDAO {  2 usages
    public boolean addCustomer(Customer customer) {  1 usage
        boolean success = false;
        try {
            Connection con = DBConnection.getConnection();
            String sql = "INSERT INTO customer (name, phone) VALUES (?, ?)";
            PreparedStatement pst = con.prepareStatement(sql);
            pst.setString( parameterIndex: 1, customer.getName());
            pst.setString( parameterIndex: 2, customer.getPhone());

            int rowsAffected = pst.executeUpdate();
            success = rowsAffected > 0;

        } catch (Exception e) {
            e.printStackTrace();
        }
        return success;
    }


    public void deleteCustomer(int customerId) {  no usages
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "DELETE FROM customers WHERE id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt( parameterIndex: 1, customerId);
            int rows = stmt.executeUpdate();
            if (rows > 0)
                System.out.println("Customer deleted successfully!");
```



```
package dao;

import ...

public class AdminDAO {  5 usages
    public boolean updateAdmin(String username, String password) {  1 usage
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "UPDATE admin SET username = ?, password = ? WHERE id = 1";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString( parameterIndex: 1, username);
            stmt.setString( parameterIndex: 2, password);
            int rows = stmt.executeUpdate();
            return rows > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }


    public Admin login(String username, String password) {  1 usage
        Admin admin = null;
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "SELECT * FROM admin WHERE username = ? AND password = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString( parameterIndex: 1, username);
            stmt.setString( parameterIndex: 2, password);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                admin = new Admin();
                admin.setId(rs.getInt( columnLabel: "id"));
                admin.setUsername(rs.getString( columnLabel: "username"));
```



```
import util.DBConnection;
import java.sql.*; import java.util.*;


public class CarDAO {  2 usages
    public boolean addCar(Car car) {  1 usage
        boolean success = false;
        try {
            Connection con = DBConnection.getConnection();
            String sql = "INSERT INTO car (model, rent) VALUES (?, ?)";
            PreparedStatement pst = con.prepareStatement(sql);
            pst.setString( parameterIndex: 1, car.getModel());
            pst.setDouble( parameterIndex: 2, car.getRent());

            int rowsAffected = pst.executeUpdate();
            success = rowsAffected > 0;

        } catch (Exception e) {
            e.printStackTrace();
        }
        return success;
    }


    public void updateCar(Car car) {  no usages
        try (Connection conn = DBConnection.getConnection()) {
            String sql = "UPDATE cars SET brand = ?, model = ?, available = ? WHERE id = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString( parameterIndex: 1, car.getBrand());
            stmt.setString( parameterIndex: 2, car.getModel());
            stmt.setBoolean( parameterIndex: 3, car.isAvailable());
            stmt.setInt( parameterIndex: 4, car.getId());
```

❖ UI Classes

- **LoginFrame.java** → Displays login interface for admin authentication.

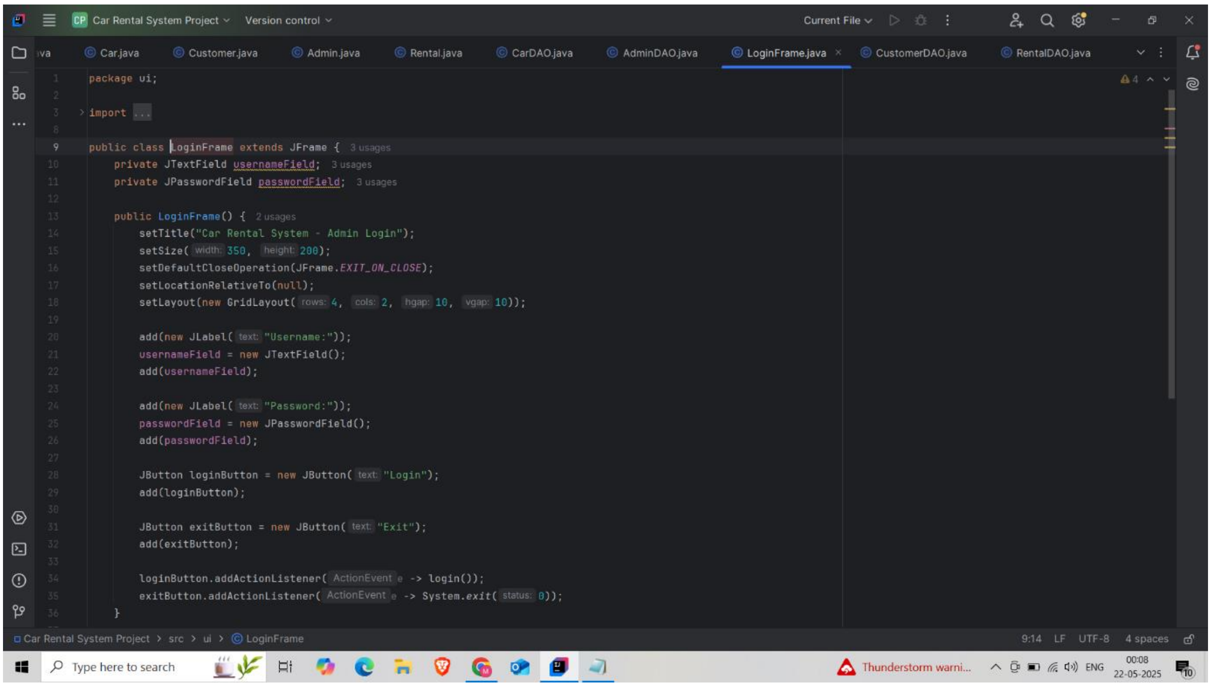
- **AdminDashboardFrame.java** → Main dashboard after login, provides navigation to different modules.

- **RentCarFrame.java** → UI for renting a car to a customer.

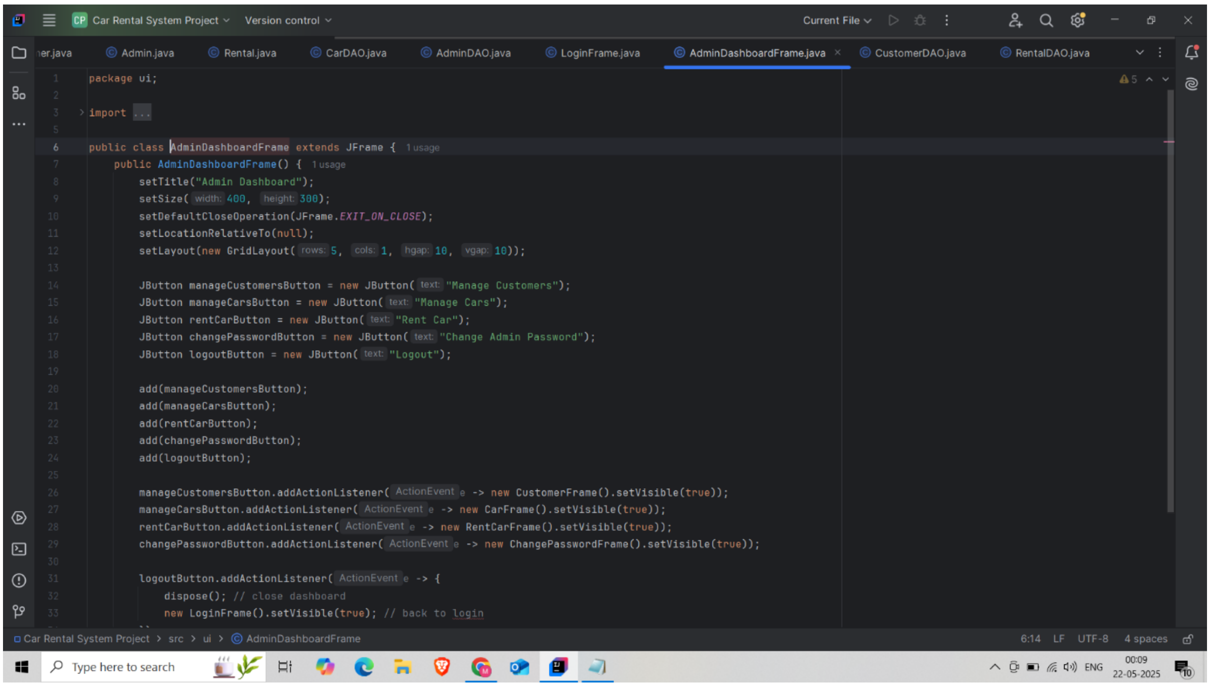
- **CarFrame.java** → Interface for adding, updating, or viewing car details.

- **ChangePasswordFrame.java** → Allows the admin to securely change their password.

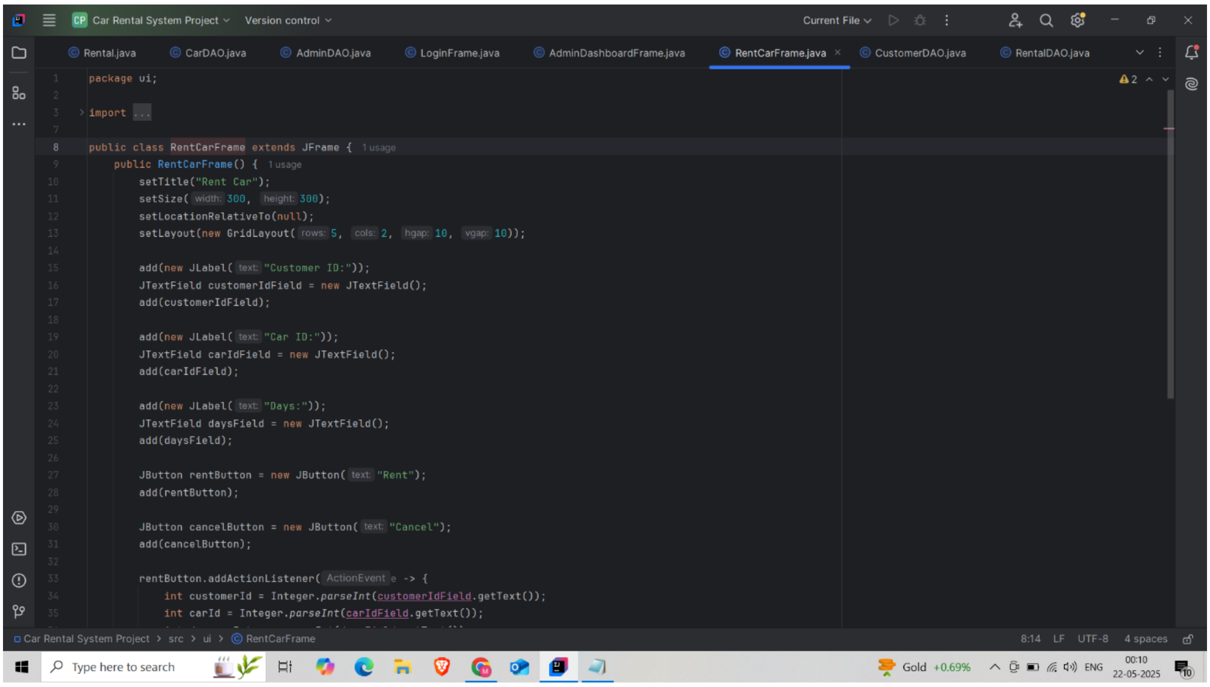
- **CustomerFrame.java** → Interface for managing customer records like add, update, and view.



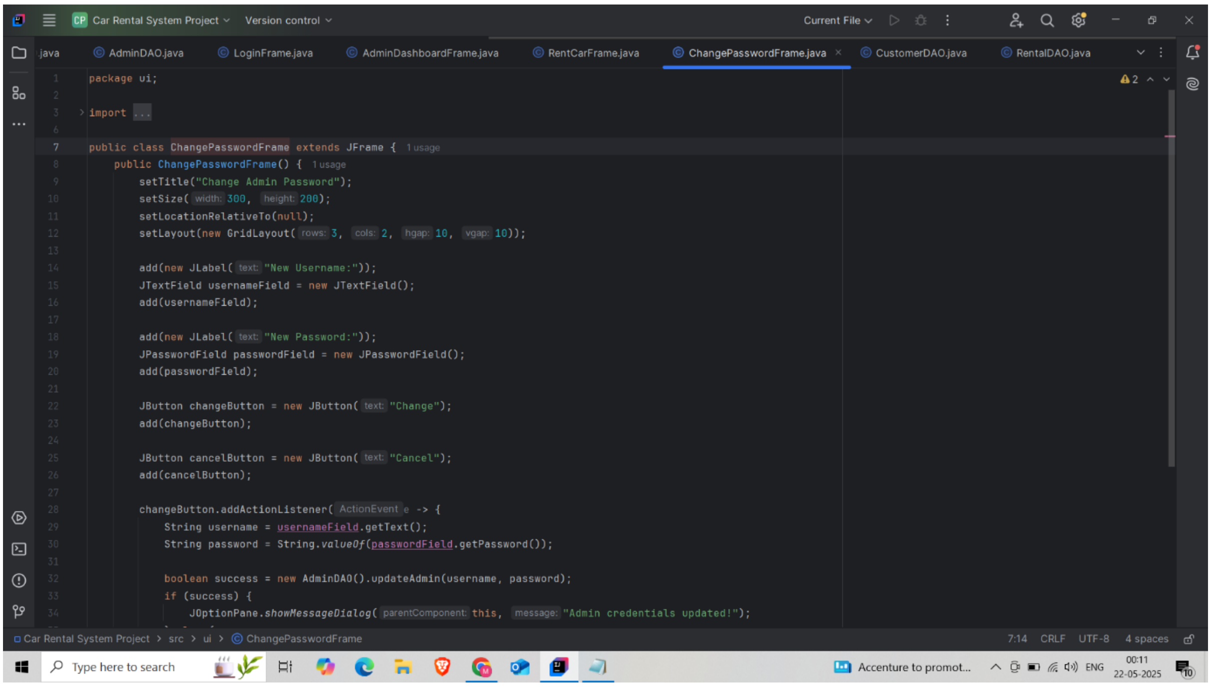
```
package ui;

import ...

public class LoginFrame extends JFrame {  3 usages
    private JTextField usernameField;  3 usages
    private JPasswordField passwordField;  3 usages

    public LoginFrame() {  2 usages
        setTitle("Car Rental System - Admin Login");
        setSize( width: 350,  height: 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new GridLayout( rows: 4,  cols: 2,  hgap: 10,  vgap: 10));

        add(new JLabel( text: "Username:"));
        usernameField = new JTextField();
        add(usernameField);

        add(new JLabel( text: "Password:"));
        passwordField = new JPasswordField();
        add(passwordField);

        JButton loginButton = new JButton( text: "Login");
        add(loginButton);

        JButton exitButton = new JButton( text: "Exit");
        add(exitButton);

        loginButton.addActionListener( ActionEvent e -> login());
        exitButton.addActionListener( ActionEvent e -> System.exit( status: 0));
    }
```



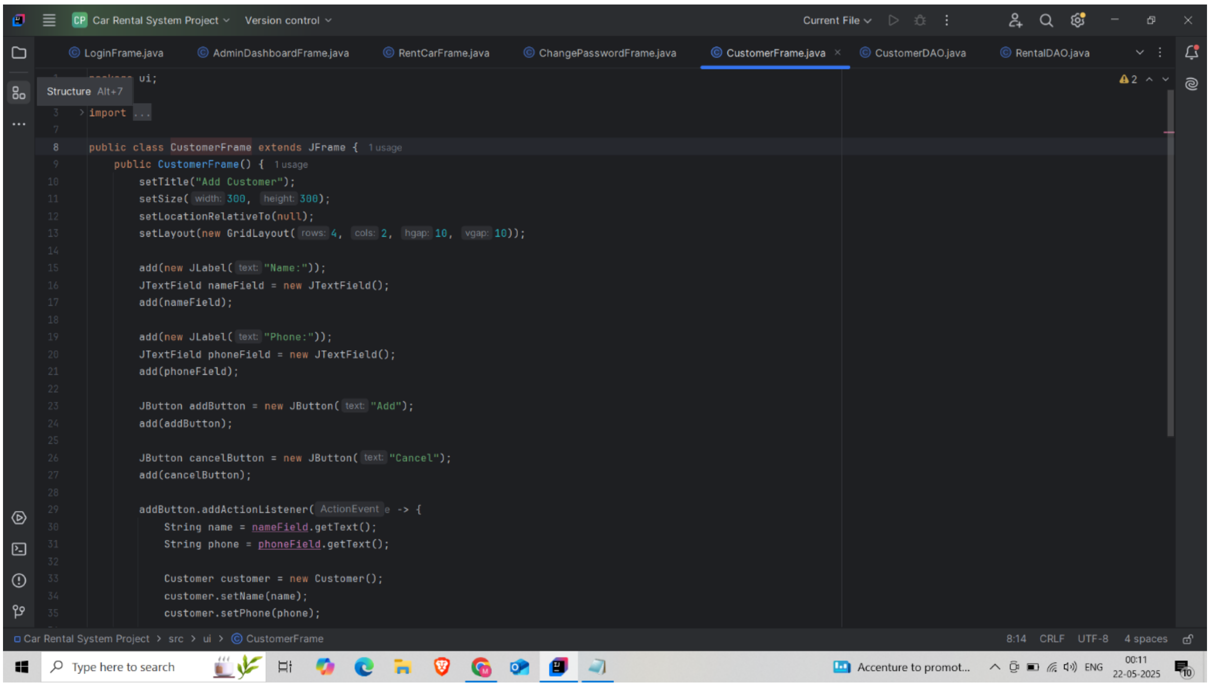
```java
package ui;

import ...

public class AdminDashboardFrame extends JFrame {  1 usage
    public AdminDashboardFrame() {  1 usage
        setTitle("Admin Dashboard");
        setSize( width: 400,  height: 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new GridLayout( rows: 5,  cols: 1,  hgap: 10,  vgap: 10));

        JButton manageCustomersButton = new JButton( text: "Manage Customers");
        JButton manageCarsButton = new JButton( text: "Manage Cars");
        JButton rentCarButton = new JButton( text: "Rent Car");
        JButton changePasswordButton = new JButton( text: "Change Admin Password");
        JButton logoutButton = new JButton( text: "Logout");

        add(manageCustomersButton);
        add(manageCarsButton);
        add(rentCarButton);
        add(changePasswordButton);
        add(logoutButton);

        manageCustomersButton.addActionListener( ActionEvent e -> new CustomerFrame().setVisible(true));
        manageCarsButton.addActionListener( ActionEvent e -> new CarFrame().setVisible(true));
        rentCarButton.addActionListener( ActionEvent e -> new RentCarFrame().setVisible(true));
        changePasswordButton.addActionListener( ActionEvent e -> new ChangePasswordFrame().setVisible(true));

        logoutButton.addActionListener( ActionEvent e -> {
            dispose(); // close dashboard
            new LoginFrame().setVisible(true); // back to login
```



```java
package ui;

import ...

public class RentCarFrame extends JFrame {  1 usage
    public RentCarFrame() {  1 usage
        setTitle("Rent Car");
        setSize( width: 300,  height: 300);
        setLocationRelativeTo(null);
        setLayout(new GridLayout( rows: 5,  cols: 2,  hgap: 10,  vgap: 10));

        add(new JLabel( text: "Customer ID:"));
        JTextField customerIdField = new JTextField();
        add(customerIdField);

        add(new JLabel( text: "Car ID:"));
        JTextField carIdField = new JTextField();
        add(carIdField);

        add(new JLabel( text: "Days:"));
        JTextField daysField = new JTextField();
        add(daysField);

        JButton rentButton = new JButton( text: "Rent");
        add(rentButton);

        JButton cancelButton = new JButton( text: "Cancel");
        add(cancelButton);

        rentButton.addActionListener( ActionEvent e -> {
            int customerId = Integer.parseInt(customerIdField.getText());
            int carId = Integer.parseInt(carIdField.getText());
```



```java
package ui;

import ...

public class ChangePasswordFrame extends JFrame {  1 usage
    public ChangePasswordFrame() {  1 usage
        setTitle("Change Admin Password");
        setSize( width: 300,  height: 200);
        setLocationRelativeTo(null);
        setLayout(new GridLayout( rows: 3,  cols: 2,  hgap: 10,  vgap: 10));

        add(new JLabel( text: "New Username:"));
        JTextField usernameField = new JTextField();
        add(usernameField);

        add(new JLabel( text: "New Password:"));
        JPasswordField passwordField = new JPasswordField();
        add(passwordField);

        JButton changeButton = new JButton( text: "Change");
        add(changeButton);

        JButton cancelButton = new JButton( text: "Cancel");
        add(cancelButton);

        changeButton.addActionListener( ActionEvent e -> {
            String username = usernameField.getText();
            String password = String.valueOf(passwordField.getPassword());

            boolean success = new AdminDAO().updateAdmin(username, password);
            if (success) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Admin credentials updated!");
```



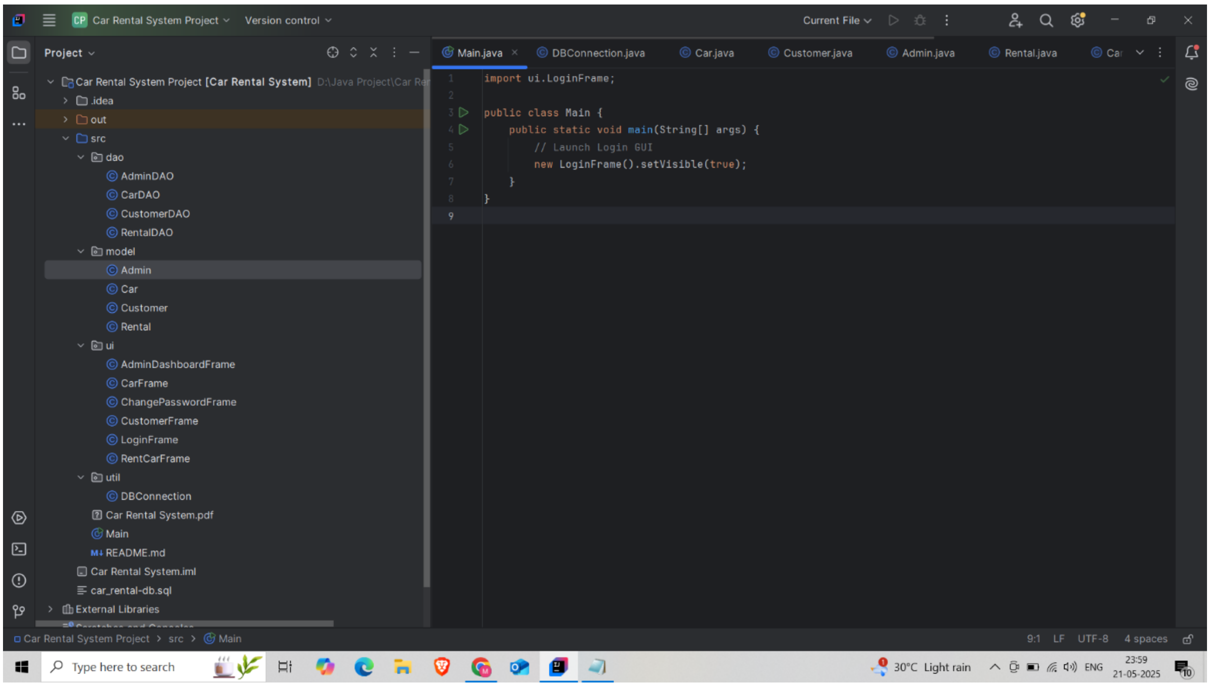
```java
package ui;
import ...

public class CustomerFrame extends JFrame {  1 usage
    public CustomerFrame() {  1 usage
        setTitle("Add Customer");
        setSize( width: 300,  height: 300);
        setLocationRelativeTo(null);
        setLayout(new GridLayout( rows: 4,  cols: 2,  hgap: 10,  vgap: 10));

        add(new JLabel( text: "Name:"));
        JTextField nameField = new JTextField();
        add(nameField);

        add(new JLabel( text: "Phone:"));
        JTextField phoneField = new JTextField();
        add(phoneField);

        JButton addButton = new JButton( text: "Add");
        add(addButton);

        JButton cancelButton = new JButton( text: "Cancel");
        add(cancelButton);

        addButton.addActionListener( ActionEvent e -> {
            String name = nameField.getText();
            String phone = phoneField.getText();

            Customer customer = new Customer();
            customer.setName(name);
            customer.setPhone(phone);
```

❖ Main.java

This is the **entry point** of the project.

- Creates DAO objects.
- Calls methods to fetch data from DB.
- Prints the list of **customers**, **cars**, and **rentals**.

This is the **entry point** of the project.



```java
import ui.LoginFrame;

public class Main {
    public static void main(String[] args) {
        // Launch Login GUI
        new LoginFrame().setVisible(true);
    }
}
```

Conclusion

The **Car Rental System** project was successfully developed and implemented by our team with the objective of automating and managing car rental operations efficiently. By applying concepts like **3-tier architecture**, **object-oriented programming**, and **JDBC database connectivity**, we have created a system that performs essential operations such as **adding**, **updating**, **deleting**, and **viewing** records for cars, customers, and rentals.

Through this project, we:

- Enhanced our understanding of **Java programming** and **MySQL database management**.
- Practiced modular coding with **separation of concerns** using Model, DAO, and UI layers.
- Experienced collaborative teamwork by dividing tasks among members effectively.
- Gained practical knowledge of **CRUD operations** and **system architecture design**.