Technical Report: Car Rental Management System

Project Title: Car Rental Management System

Team Members:

علي محسن الحبشي 2221809256

عبدالعزيز اشرف الجازوي 2221808195

محمد المهدي اللوشة 2221810969

Course: Advanced Programming with Java

1. Overview of the System

The Car Rental Management System is a desktop application built to automate the core operations of a car rental business. It provides a user-friendly interface for managing vehicles, customer information, and rental transactions. The system enables staff to efficiently add cars and customers, process new rentals, complete returns, and generate business performance reports, thereby replacing manual processes with a centralized, digital solution.

2. Technologies and Tools Used

- **Language:** Java 11+
- **IDE:** NetBeans with Apache Ant
- **Database:** SQLite with sqlite-jdbc-3.36.0.3.jar
- **GUI Framework:** Java Swing (AWT)
- **Core Java APIs:** JDBC, Collections Framework, I/O Streams, Multithreading (SwingWorker)

3. Key Features Implemented

- **Car Management:** Add, update, delete, and view vehicles. Automatically tracks availability.
- **Customer Management:** Register, update, delete, and view customer information.
- **Rental Management:** Create new rentals with automatic cost calculation, complete rentals (which updates car availability), and view all rental records.
- **Reporting:** Generate summary, car usage, and customer activity reports.
- **Data Export:** Export cars, customers, and rental data to CSV files.
- **System Utilities:** Automatic action logging and a configurable auto-refresh feature for real-time data updates.

4. Explanation of Required Topics Implementation

This section details how each required advanced Java topic was integrated into the project.

1. Exception Handling

- Robust try-catch-finally blocks are used in all DAO (Data Access Object) classes to manage database connections and handle SQLExceptions, ensuring resources are always closed.
- The service layer validates user input (e.g., checking for empty fields) to prevent invalid data from reaching the database.
- File I/O operations in FileManager use try-catch to handle potential IOExceptions gracefully.

2. Collections Framework

- **ArrayList:** Used extensively in DAO classes to store and manage lists of objects retrieved from the database (e.g., List<Car>, List<Customer>).
- **HashMap:** Utilized in the reporting module to efficiently aggregate data, such as mapping car IDs to their total rental revenue.

3. JDBC - Java Database Connectivity

- A DatabaseManager class handles the connection to the local SQLite database and initializes the cars, customers, and rentals tables on first run.
- Full CRUD (Create, Read, Update, Delete) operations are implemented using PreparedStatement in all DAO classes to ensure security against SQL injection and improve performance.

4. GUI Using Swing + Event Handling

- The entire user interface is built with Swing components, including JFrame, JTabbedPane, JTable, JButton, and JComboBox.
- The application is event-driven. ActionListeners are attached to buttons to respond to user actions, triggering calls to the service and DAO layers to perform business logic and update the UI.

5. I/O Streams

- The FileManager class uses a BufferedWriter to implement a logging system that writes system actions to a text file (car_rental_log.txt).
- The same class provides functionality to export data from the application to structured CSV files using FileWriter.

6. Multithreading

- All potentially long-running database operations are executed in a background thread using a custom BackgroundTask class that extends SwingWorker. This prevents the GUI from freezing and provides a responsive user experience.
- An AutoRefreshService uses a ScheduledExecutorService to periodically refresh data in the JTable without user intervention.

5. Sample Screenshots

Figure 1: Main Application Window The main interface with the tabbed pane for navigating between Cars, Customers, Rentals, and Reports.

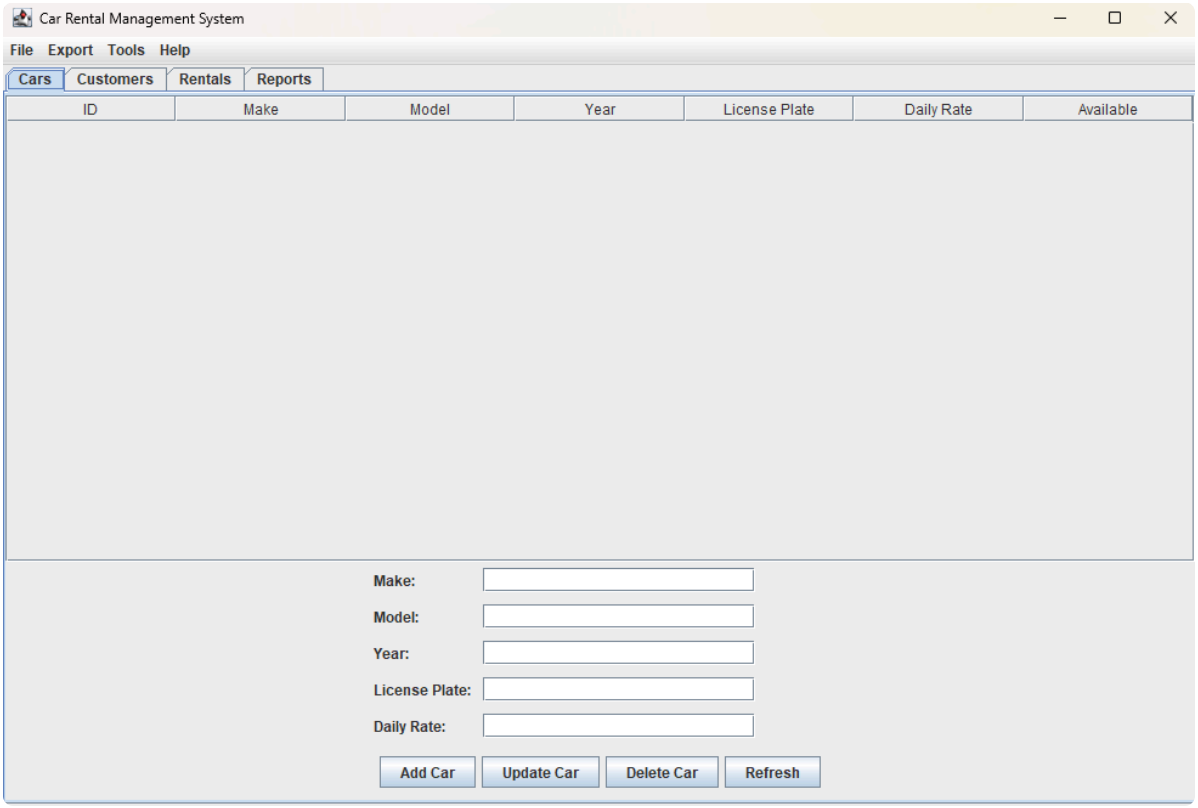


Figure 2: Car Management Panel The form for adding a new car and the table displaying the current car inventory.

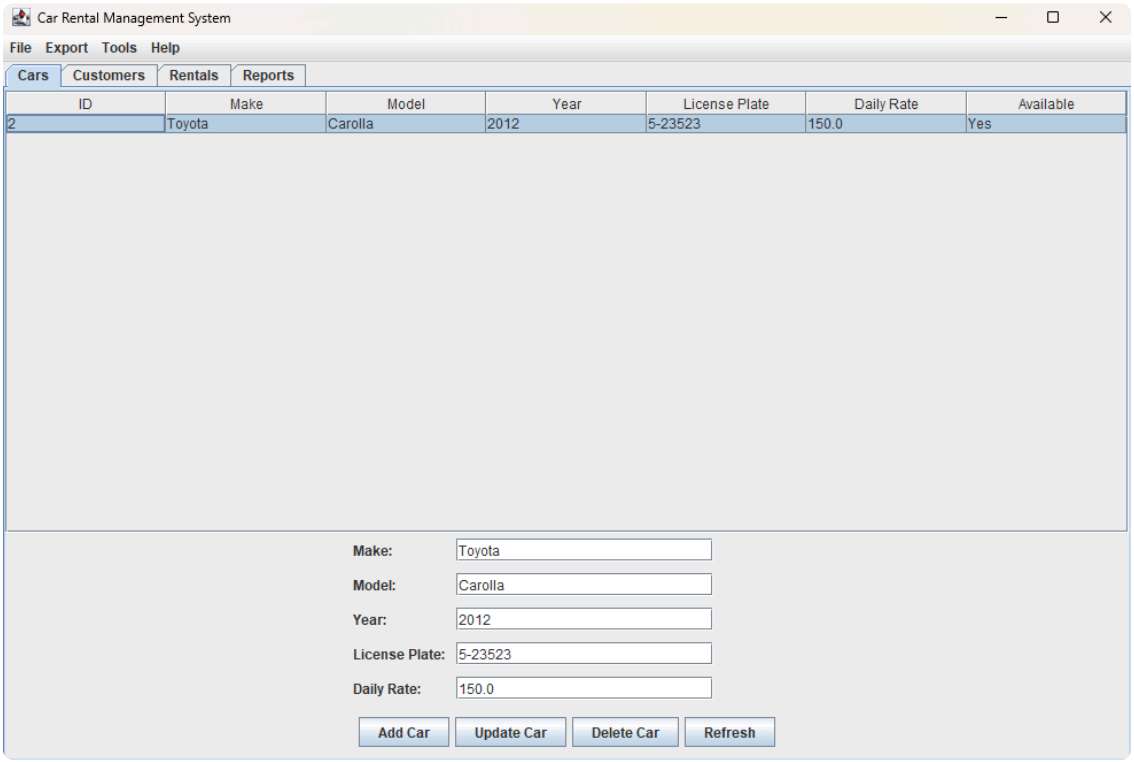


Figure 3: Creating a New Rental The process of creating a new rental by selecting a car, customer, and rental dates.

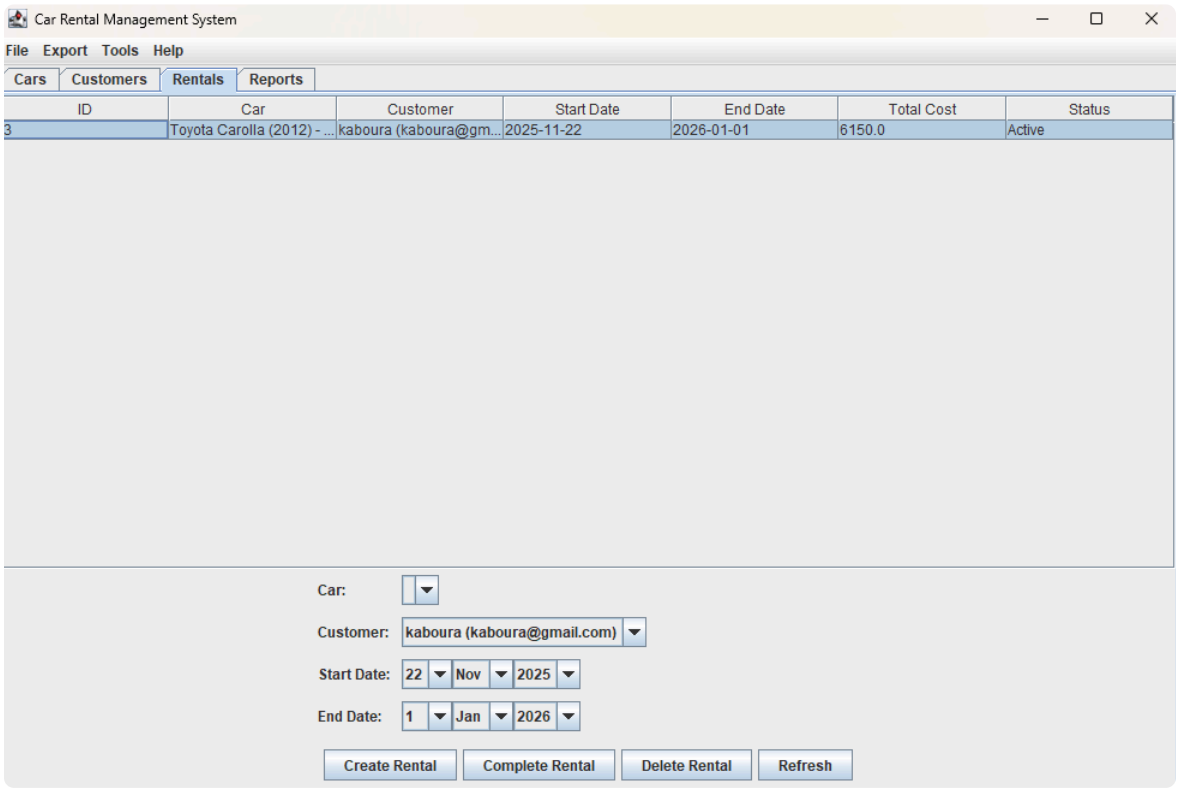
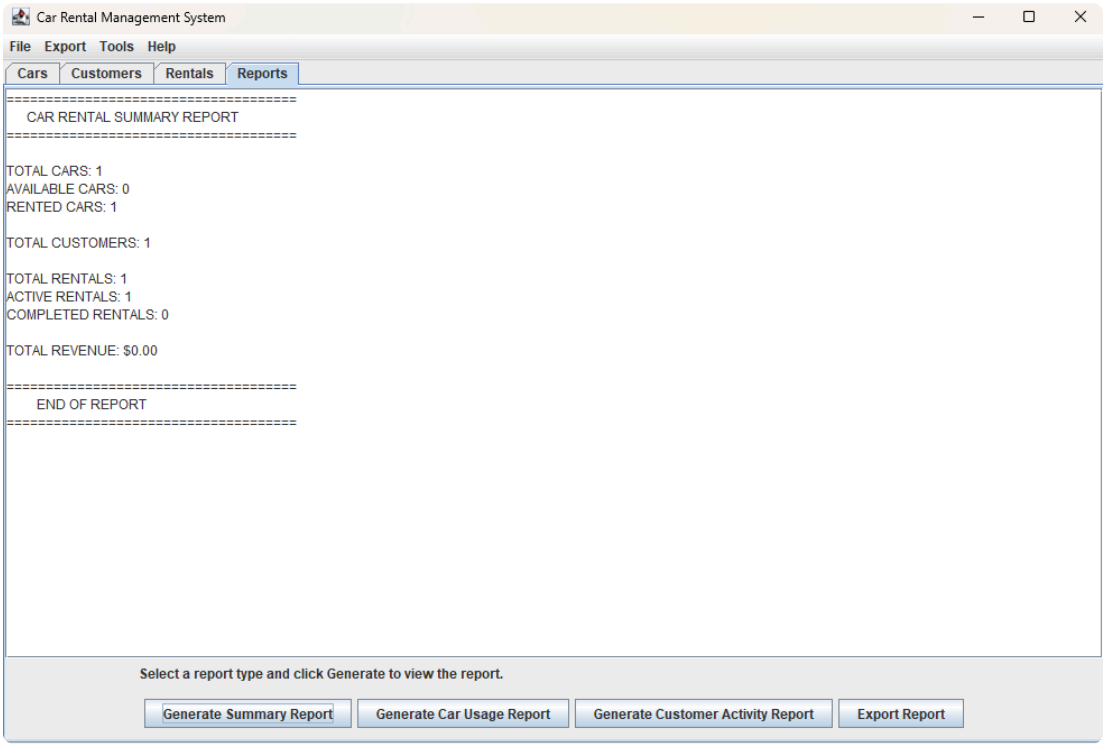


Figure 4: Generated Summary Report The output of the summary report, showing key business metrics.



6. Task Distribution Among Team Members

Team Member	Responsibilities
علي محسن الحبشي	- Project Lead & Database Design - Implemented DatabaseManager and all DAO classes - Integrated JDBC and handled database connectivity
عبدالعزيز اشرف الجازوي	- GUI Development - Designed and implemented all UI panels - Implemented event handling and user input validation
محمد المهدي اللوشة	- Advanced Features & Reporting - Implemented multithreading (BackgroundTask, AutoRefreshService) - Developed reporting module and FileManager for I/O operations - Prepared technical documentation

7. Challenges and Lessons Learned

- **Database Design:** A key challenge was designing a logical schema that properly linked cars, customers, and rentals. We learned that a well-planned database with clear relationships is the foundation of a stable application.
- **UI Responsiveness:** We encountered the common problem of the GUI freezing during database queries. This led us to implement SwingWorker for all background tasks, which was crucial for a smooth user experience.
- **Layered Architecture:** Coordinating the different parts of the system was a challenge. Adopting a layered architecture (UI -> Service -> DAO) made the code more organized, easier to test, and simplified team collaboration.