242370022 :Abdullah 242370041: Usama

2423700101:Ahmad Fozan 242370040: Ahmad

2423700102: Ahmad Aziz

**Project Name: Car Rental Management System**

### ****Core Functionalities (11 standard)****

1. **User Registration/Login** (Admin, Customer, Employee)
2. **Add Cars** (Admin only)
3. **View Cars**
4. **Book Car**
5. **Return Car**
6. **Delete Car** (Admin only)
7. **View Booking History** (DLL)
8. **Search Cars**
9. **Sort Cars**
10. **Customer Management**
11. **Feedback System** (Customer → Employee ratings)

Unique Features:

1. **Priority Booking** using **PriorityQueue** (VIP customers first)
2. **Employ Priority Based on Feedback**
3. **Admin Logs Track**
4. **Tree for Luxary/Cheap Cars**
5. **Booking Cancelation Using Stack**
6. **Panality on Late return**

**Project Summary:**

The **Car Rental Management System** is a Java-based application designed to automate and streamline the operations of a car rental business. The system allows three types of users—**Admin**, **Employee**, and **Customer**—each with role-specific access and functionalities.

The **Admin** has full control over the system, including adding and deleting cars, registering employees, managing customer priority queues, viewing booking history, and monitoring premium customers. **Employees** assist with day-to-day operations such as calculating bills, processing returns, and managing bookings. **Customers** can register, log in, book cars, cancel bookings, rate employees, and view available cars.

The system incorporates **data structures** like Linked Lists, Queues, Stacks, and Trees for efficient data management and retrieval. Special features include **priority-based booking** for VIP customers, **Admin logs** , Best Employ and Booking Cancelation And Tree for Classification of Cars

This project is designed to provide a realistic simulation of a professional car rental service, making it suitable for academic evaluation as well as potential adaptation for real-world usage.

**Functional Requirements**

The **Car Rental Management System** will fulfill the following functional requirements, categorized by user roles:

### 1. ****Admin Functions****

* **Login** using a fixed admin password.
* **Add New Car** to the fleet with details like ID, brand, model, and daily rental price.
* **Delete Car** from the system by ID.
* **View All Cars** and **sort cars by brand**.
* **Find Available Cars** for rental.
* **Register Employees** and ensure only registered employees can log in.
* **Manage Customer Priority Queue** (VIP and normal customers).
* **View Premium Customers** stored in the premium queue.
* **View Booking History** using a doubly linked list.
* **View System Logs** for admin actions.
* **Logout** from the admin account.

### 2. ****Employee Functions****

* **Login** using registered credentials.
* **Calculate Total Bill** based on rental days and late return penalties.
* **Mark Car as Returned** and update availability.
* **View All Bookings** and individual customer bookings.
* **Assist Customers** with booking and cancellation requests.

### 3. ****Customer Functions****

* **Register** as a new customer with username and password.
* **Login** to access personal account.
* **Book a Car** by selecting from available options.
* **Cancel Last Booking** (stack-based undo functionality).
* **Rate Employees** to provide service feedback.
* **View Available Cars** by brand and availability.

### 4. ****System Features & Data Structure Usage****

* **Linked Lists** for storing cars, customers, employees, and bookings.
* **Priority Queues** for VIP and normal customer handling.
* **Stacks** for booking cancellation and payment rollback.
* **Binary Search Tree** for car classification by price range (cheap vs. luxury).
* **Late Return Penalty** system based on return date vs. due date.

INDIVIDUAL CONTRIBUTIONS:

The **Car Rental Management System** was developed collaboratively by five team members, each responsible for specific modules and features of the system:

### ****1. Abdullah (Roll No: 242370022)****

* Developed **Employee-related classes** (Employ, EmployNode, EmployServices) including employee registration, login, and performance rating system.
* Designed and implemented the **Main file** with a role-based menu (Admin, Employee, Customer).
* Implemented **Customer-related classes** (Customer, CustomerNode, CustomerServices) with **VIP queue handling** and employee rating feature.
* Worked on **Tree classes** (ManagingTree, TreeNode) in collaboration with Ahmad for classifying cars as cheap or luxury.

### ****2. Ahmad (Roll No: 242370040)****

* Developed **Payment-related classes** (Payment, PaymentNode, PaymentServices) with support for recording and canceling payments.
* Created **Bill-related classes** (Bill) for rental cost calculation.
* Worked on **Tree classes** in collaboration with Abdullah to store and retrieve cars based on price range.

### ****3. Ahmad Aziz (Roll No: 2423700102)****

* Developed **Car-related classes** (Car, CarNode, CarServices) with functionalities such as adding, deleting, searching, and sorting cars, and checking availability.

### ****4. Usama (Roll No: 242370041)****

* Developed **Booking-related classes** (Booking, BookingNode, BookingDoubleNode, BookingServices) including booking creation, cancellation (stack-based), marking returns, and maintaining booking history using a doubly linked list.

### ****5. Ahmad Fozan (Roll No: 2423700101)****

* Developed **Admin-related classes** (Admin, AdminLogNode) with functionalities for managing cars, employees, customers, and system logs.
* Contributed to **Customer-related classes** in collaboration with Abdullah for customer registration/login features.

# Time Complexity Analysis

This section provides the Big-O notation for each major method implemented in the Car Rental Management System, organized by class.

## 1. CarServices

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| addCar(Car) | O(n) | Checks for duplicate IDs and traverses to the end of the linked list before adding. |
| displayAllCars() | O(n) | Traverses and prints all cars. |
| findAvailableCar(String) | O(n) | Searches for an available car by brand. |
| sortCarsByBrand() | O(n²) | Uses insertion sort on a linked list. |
| deleteCar(String) | O(n) | Searches and removes the car from the list. |
| searchById(String) | O(n) | Linear search for the car ID. |
| markAsRented(String) | O(n) | Finds the car and updates availability. |
| markAsReturned(String) | O(n) | Finds the car and marks it as available. |

## 2. Admin

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| login(String) | O(1) | Compares input password with the stored one. |
| addCar(CarServices, Car) | O(n) | Delegates to CarServices.addCar. |
| deleteCar(CarServices, String) | O(n) | Delegates to CarServices.deleteCar. |
| registerEmploy(EmployServices, Employ) | O(n) | Delegates to EmployServices.register. |
| addCusPriorityVise(CustomerServices) | O(n) | Delegates to CustomerServices.insertCustomerPriorityVise. |
| logAction(String) | O(1) | Inserts log entry at head of a doubly linked list. |
| viewLogs() | O(m) | Traverses all admin log entries. |

## 3. CustomerServices

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| register(Customer) | O(n) | Checks for duplicate and appends at end. |
| userExits(String) | O(n) | Searches for existing username. |
| login(String, String) | O(n) | Searches for matching username/password. |
| insertCustomerPriorityVise() | O(n) | Classifies customers into premium and regular queues. |
| rateEmploy(...) | O(n + m) | Finds customer (O(n)), updates employee rating (O(m)). |
| viewPremiumCustomers() | O(p) | Traverses only the premium queue. |
| displayCusList() | O(n) | Displays all customers. |

## 4. EmployServices

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| register(Employ) | O(n) | Checks for duplicates before adding. |
| EmpExist(String) | O(n) | Searches employee list. |
| login(String, String) | O(n) | Matches username/password. |
| updatePerformance(String, double) | O(n) | Finds and reorders employee based on performance. |
| insertInPriorityOrder(EmployNode) | O(n) | Inserts employee in sorted position. |
| displayAdminList() | O(n) | Displays all employees. |

## 5. BookingServices

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| bookCar(...) | O(n + m) | Searches car list (O(n)), updates customer booking list (O(m)). |
| markAsReturned(...) | O(n + m) | Searches booking list and customer list. |
| cancelLastBooking() | O(n) | Pops booking stack and updates car availability. |
| removeBookingFromList(String) | O(n) | Traverses list to remove booking. |
| viewAllBookings() | O(n) | Displays all bookings. |
| viewCustomerBookings(String) | O(n) | Displays bookings for one customer. |

## 6. PaymentServices

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| recordPayment(...) | O(n) | Appends payment at end of list. |
| cancelLastPayment() | O(1) | Pops from stack. |
| viewAllPayments() | O(n) | Displays all payments. |

## 7. TotalBillCalculator

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| calculateBillByBooking(...) | O(n) | Searches booking and car lists. |
| calculateBillByCustomer(...) | O(n × m) | For each booking by customer, finds matching car. |

## 8. ManagingTree

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Explanation |
| insert(Car) | O(h) | Inserts based on tree height (h). |
| displayCheapCars() | O(n) | Traverses all nodes. |
| displayLuxuryCars() | O(n) | Traverses all nodes. |
| searchByPricePerDay(...) | O(n) | Traverses all nodes (no BST property by price). |