Design Report

Table of Contents

1 Title

AutoRentNexus: Vehicle Rental Management System

2 Summary

AutoRentNexus enhances the efficiency and reliability of car rental operations by automating essential functions such as vehicle management, customer service, and rental transactions. Developed using Java and modeled with Lucidchart, this system allows administrators to manage vehicles and customer profiles, search for available vehicles, book rentals, calculate booking costs, generate invoices, and send email notifications to customers. AutoRentNexus aims to streamline the car rental process, reduce manual workloads, and improve overall customer satisfaction.

3 Introduction

Car rental agencies face several operational challenges, including inefficient vehicle tracking, delayed customer service, and cumbersome transaction processes. These issues result in customer dissatisfaction, increased operational costs, and potential revenue loss. AutoRentNexus is designed to address these challenges by providing a comprehensive solution that automates vehicle management, customer service, and booking processes. By leveraging modern software engineering principles and a modular architecture, AutoRentNexus ensures reliable, scalable, and user-friendly operations for car rental agencies.

4 Problem Definition

- Ineffective Car Tracking Systems: Manual tracking of vehicle availability and status leads to inefficiencies and increased operational costs. AutoRentNexus provides a centralized vehicle management system that allows administrators to add, update, and remove vehicles, and view vehicle information and status in real-time.
- Customer Service Delays: Manual booking and check-in processes result in long wait times, negatively impacting customer satisfaction. AutoRentNexus automates the booking process, enabling quick searches for available vehicles, immediate booking confirmations, and seamless customer interactions.
- Complex Transaction Processes: Paperwork and manual verification during rental transactions are time-consuming and prone to errors. AutoRentNexus automates these processes, generating booking confirmations and invoices, and sending them directly to customers via email.

5 System Requirements

5.1 Functional Requirements

1. Vehicle Management:
    - Admin:
        - Add, update and remove vehicles.
        - Access vehicle information (model, rental price, availability).
        - Access a list of all vehicles and their current status (available, rented).

2. Booking System:
    - Admin:
        - Search for available vehicles based on dates and vehicle type specified by the customer.
        - Book vehicles for a specific time period for customers, ensuring no double bookings occur.
        - Calculate the cost of booking based on the rental period and vehicle type.

Vehicle Types and Prices per Day	
SUV	$60/day
Truck	$80/day
Van	$100/day
Convertible	$70/day
Wagon	$50/day
Electric Vehicle (EV)	$50/day
Economy Car	$40/day
Sports Car	$100/day

        - Generate booking confirmations and invoices for customers and provide them to the customers via email.
    - Customer:
        - Receive booking confirmation and invoice after the admin completes the booking process.

3. User Management:
    - Admin:
        - Create new customer profiles with details such as name, contact information, and payment details.
        - Edit customer profiles to update information or correct errors.

    - Remove customer profiles from the system.
    - View customer rental history, including past and current rentals, and payment details.

5.2 Non-Functional Requirements

- Performance:
    - Pages should load within a short time frame (2-3 seconds).
    - The system should provide quick response times for searching, booking, and adding/removing vehicles.
- Usability:
    - The interface should be user-friendly, intuitive, and easy to navigate.
- Reliability:
    - The system should have an uptime of 99.9% or higher.
- Maintainability:
    - Use version control (GitHub) to manage code changes and collaboration.

6 System Architecture

6.1 Three Main Components Of The System Architecture

1. Vehicle Management Component
    - Admin Interface
        - Provides a UI for admins to manage vehicle details.
        - Sends requests to the Vehicle Database to perform CRUD operations.
    - Vehicle Database
        - Stores vehicle details including model, rental price, and availability.
        - Interacts with the Admin Interface for adding, updating, and removing vehicles.
2. Booking System Component
    - Admin Interface
        - Provides a user interface for admins to manage bookings, search for available vehicles, and view booking details.
        - It interacts with the Booking Engine to initiate and confirm bookings.
    - Booking Engine
        - Handles the logic for searching available vehicles, booking vehicles, and calculating booking costs.
        - Interacts with the Admin Interface and Vehicle Database.
    - Vehicle Database
        - Stores vehicle availability and booking status.
        - It provides data to the Booking Engine for checking availability and updating booking status.
    - Invoice Generator

        - Generates booking confirmations and invoices.
        - Sends generated documents to the Email Service for delivery.
    - Email Service
        - Handles sending booking confirmation and invoice emails to customers.
        - Receives documents from the Invoice Generator.
3. Customer Management Component
    - Admin Interface
        - Provides a UI for admins to manage customer details.
        - Sends requests to the Customer Database to perform CRUD operations.
    - Customer Database
        - Stores customer details including name, contact information, and rental history.
        - Interacts with the Admin Interface for adding, updating, and removing customer profiles.

6.2 Table of Interfaces

Top-Level Component	Internal Structure	Interface Method	Purpose
Vehicle Management	Admin Interface → Vehicle Database	+addVehicle(vehicle: Vehicle)	Adds a new vehicle to the database
		+updateVehicle(vehicle: Vehicle)	Updates details of an existing vehicle
		+removeVehicle(vehicle: Vehicle)	Removes a vehicle from the database
		+viewVehicleList()	Retrieves the list of all vehicles
Booking System	Admin Interface → Booking Engine	+searchAvailability(startDate: Date, endDate: Date): List<Vehicle>	Searches for available vehicles within a specified date range
		+confirmBooking(booking: Booking)	Confirms a booking after checking availability
		+cancelBooking(bookingID)	Cancels an existing booking
	Booking Engine → Vehicle Database	+getAvailability(vehicleID: int, startDate: Date, endDate: Date): boolean	Checks if a vehicle is available for the specified date range

		+setAvailability(vehicleID: int, status: boolean)	Updates the availability status of a vehicle
	Booking Engine → Invoice Generator	+generateInvoice(bookingID: int, totalCost: double): Invoice	Generates an invoice for the specified booking
	Invoice Generator → Email Service	+sendEmail(customer: Customer, invoice: InvoiceDetails): void	Sends an email with the booking confirmation and invoice
Customer Management	Admin Interface → Customer Database	+addCustomer(customer: Customer)	Adds a new customer to the database
		+editCustomer(customer: Customer)	Updates details of an existing customer
		+removeCustomer(customer: Customer)	Removes a customer from the database
		+viewCustomerProfile(customerID: int): Customer	Retrieves the profile details of a specified customer
		+viewCustomerRentalHistory(customerID: int): List<Booking>	Retrieves the rental history of a specified customer

The top-level components (Vehicle Management, Booking System, and Customer Management) interact through well-defined interfaces that facilitate smooth communication and data exchange. For example, the Admin Interface in the Vehicle Management component uses methods like *+addVehicle(vehicle: Vehicle)* and *+viewVehicleList()* to interact with the Vehicle Database, enabling the addition, updating, and removal of vehicles. Similarly, the Booking System's Admin Interface interacts with the Booking Engine to confirm bookings and check vehicle availability through methods like *+confirmBooking(booking: Booking)* and *+searchAvailability(startDate: Date, endDate: Date)*. The Booking Engine also interfaces with the Invoice Generator and Email Service to generate invoices and send booking confirmations to customers. In the Customer Management component, the Admin Interface uses methods like *+addCustomer(customer: Customer)* and *+viewCustomerProfile(customerID: int)* to manage customer data within the Customer Database. These interfaces ensure that each component can function independently while still communicating effectively with other parts of the system, promoting modularity and ease of maintenance.

7 Solution Features

7.1 Top-Level Components and Their Functionalities

1. Vehicle Management Component:
    - Add Vehicle
    - Update Vehicle
    - Remove Vehicle
    - View Vehicle Information (VehicleType)
2. Booking System Component:
    - Search Available Vehicles
    - Book Vehicle
    - Calculate Booking Cost
    - Generate Booking Confirmation
    - Generate Invoice
3. Customer Management Component:
    - Create Customer Profile
    - Edit Customer Profile
    - Remove Customer Profile
    - View Customer Profile
    - View Customer Rental History

7.2 Product Features Table

Feature	Description	Component involved
Add Vehicle	Admin can add a new vehicle to the system.	Admin Interface, Vehicle Database
Update Vehicle	Admin can update details of an existing vehicle.	Admin Interface, Vehicle Database
Remove Vehicle	Admin can remove a vehicle from the system.	Admin Interface, Vehicle Database
View Vehicle Information	Admin can view details of all vehicles.	Admin Interface, Vehicle Database
Search Available Vehicles	Admin can search for vehicles based on availability and type.	Admin Interface, Booking Engine, Vehicle Database

Book Vehicles	This feature allows admins to book vehicles for customers, marking them as unavailable to ensure no double-booking occurs.	Admin Interface, Booking Engine, Vehicle Database
Calculate Booking Costs	The system calculates the cost based on rental period and vehicle type.	Booking Engine
Generate Booking Confirmation	The system generates a booking confirmation for the customer.	Booking Engine, Invoice Generator
Generate Invoice	The system generates an invoice for the customer.	Booking Engine, Invoice Generator
Create Customer Profile	Admin can create a new customer profile.	Admin Interface, Customer Database
Edit Customer Profile	Admin can edit details of an existing customer profile.	Admin Interface, Customer Database
Remove Customer Profile	Admin can remove a customer profile from the system.	Admin Interface, Customer Database
View Customer Profile	Admin can view details of a customer profile.	Admin Interface, Customer Database
View Customer Rental history	Admin can view the rental history of a customer.	Admin Interface, Customer Database
Send Email	The system sends booking confirmation and invoice emails to customers.	Email Service, Invoice Generator

8 Solution Limitations & Assumptions

8.1 Limitations

- Data Protection and Security:
    - Limited Security Features: The current implementation does not include advanced security features such as encryption for sensitive customer data or secure payment processing. Handling of personal and payment information may not comply with all relevant data protection regulations (e.g., GDPR, CCPA).

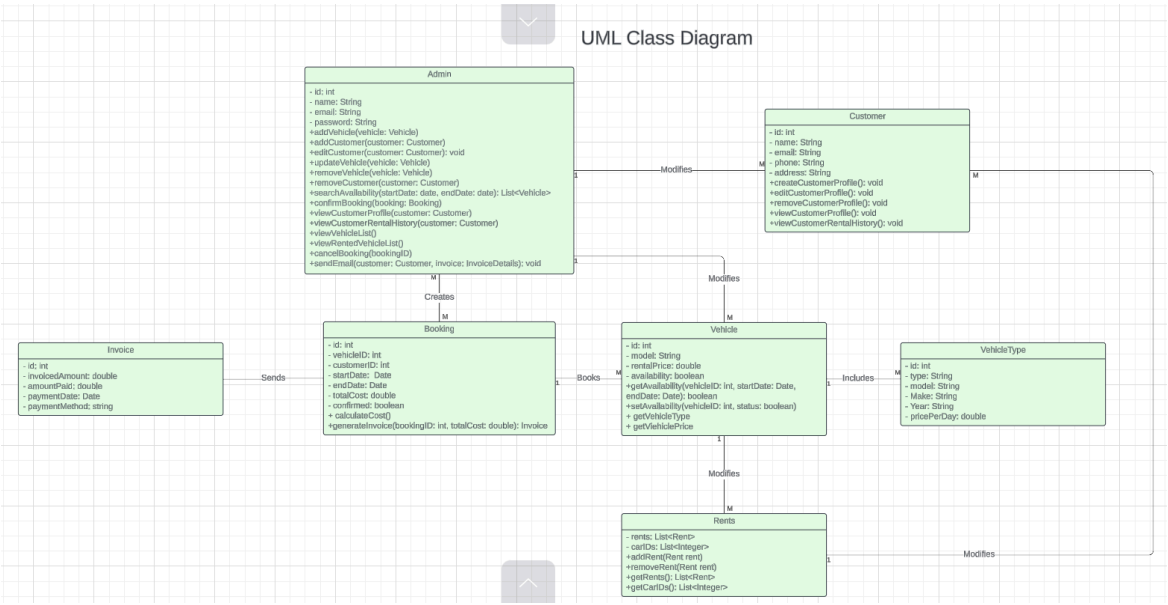
    - No Authentication Mechanisms: The system does not include user authentication or authorization mechanisms beyond basic admin login functionality.
- Integration with External Systems:
    - Payment Gateway Integration: The system does not integrate with payment gateways for processing payments, requiring manual handling of payment transactions outside the system.
    - Limited Email Functionality: Email functionality is simulated through console output. Actual email sending capabilities are not implemented, requiring integration with an email service provider.
- Scalability and Performance:
    - Performance Constraints: The system may experience performance bottlenecks as the number of vehicles, customers, and bookings increase, especially due to the lack of optimizations for handling large datasets.
    - Scalability Limitations: The system is not designed to scale horizontally across multiple servers or to handle a high volume of concurrent users.

8.2 Assumptions

- Admin Training and Usage:
    - Admin Competence: It is assumed that administrators using the system are adequately trained and possess the necessary technical skills to operate the console-based interface and perform administrative tasks.
    - Single Admin Usage: The system assumes that only one admin will be using the system at any given time, reducing the risk of data inconsistencies due to concurrent modifications.
- Customer Interaction:
    - Email Accessibility: It is assumed that all customers have access to email and can receive booking confirmations and invoices via email.
    - Manual Payment Handling: Customers will handle payments outside the system, and admins will manually update payment statuses within the system.
- Operational Environment:
    - Stable Runtime Environment: The system assumes a stable and consistent runtime environment where it is deployed, with adequate system resources (CPU, memory) to handle its operations.
    - Data Backup and Recovery: Regular data backups are assumed to be performed by the admin to prevent data loss, as the system does not include built-in backup and recovery mechanisms.
- System Maintenance:
    - Version Control Usage: The development and maintenance of the system will utilize version control (e.g., GitHub) to manage code changes and collaboration among developers.

- Periodic Updates: It is assumed that periodic updates and maintenance will be performed to address bugs, improve performance, and enhance system features over time.

9 Class Diagram

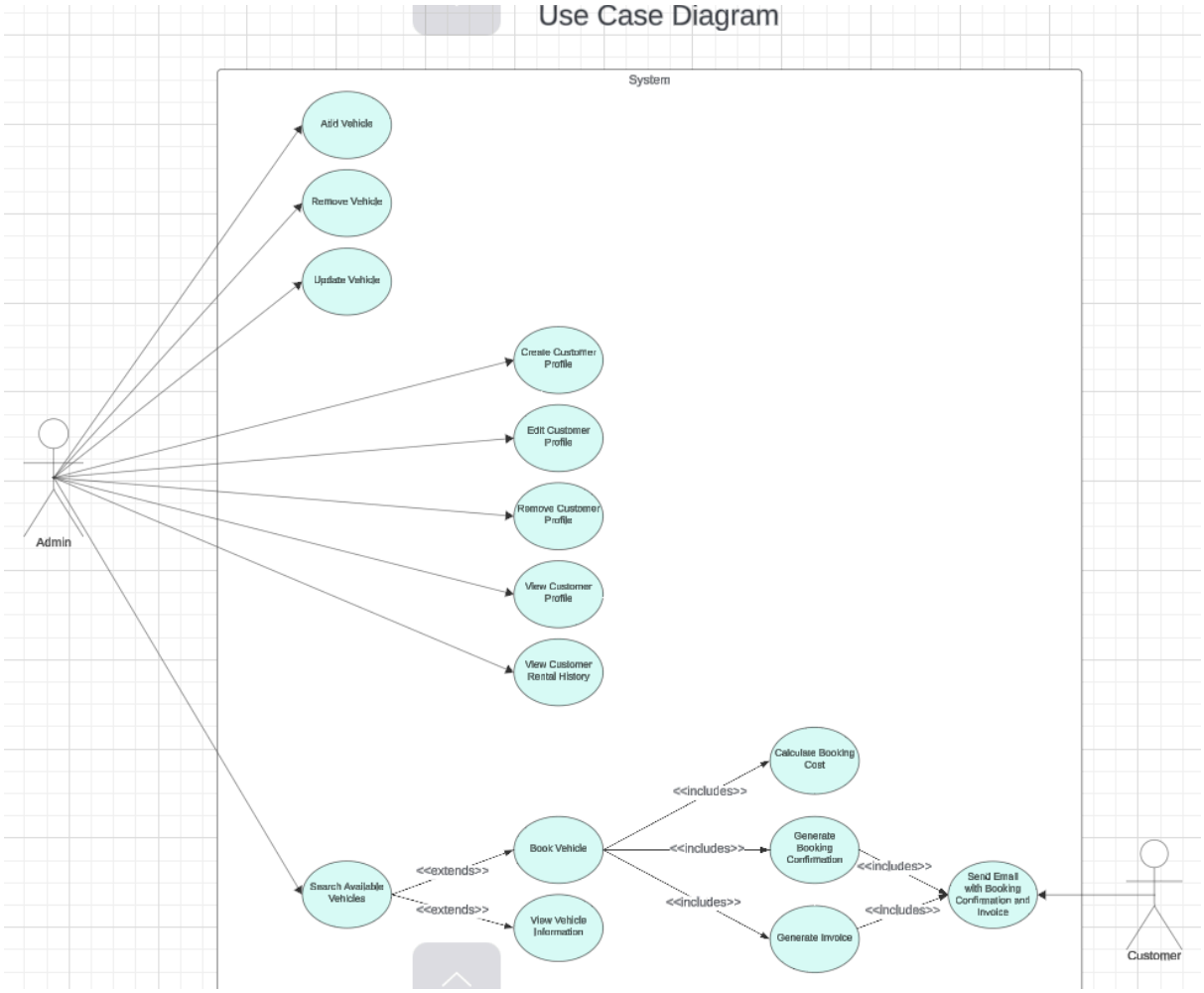


Class Diagram Lucidchart Link:
https://lucid.app/lucidchart/2b8c774a-7675-4473-a9fe-934f429cf506/edit?invitationId=inv_1b013c63-c4b8-483c-8679-8babc33bd4b9&page=HWEp-vi-RSFO#

The UML class diagram represents the structure and relationships of the various classes within AutoRentNexus. It includes key classes such as Admin, Customer, BookingEngine, Vehicle, VehicleType, Invoice, and Rents.

- Admin: Manages vehicle, customer, and booking information. It includes methods for adding, updating, and removing vehicles and customers, as well as booking management functionalities.
- Customer: Represents customers with attributes like name, email, phone, and address, and methods for managing customer profiles and viewing rental history.
- Booking: Handles the booking process, including searching for available vehicles, confirming bookings, and generating invoices.
- Vehicle: Represents vehicles with details such as model, rental price, and availability, including methods for managing vehicle status and booking information.
- VehicleType: Defines the type and pricing details of various vehicles available for rent.
- Invoice: Manages invoicing details, including invoiced amount, amount paid, payment date, and payment method.
- Rents: Manages the list of rented vehicles and their associated IDs.

10 Use Case Diagram



Use Case Diagram Lucidchart Link:
https://lucid.app/lucidchart/76412a25-e674-4318-9f86-be0b4696eef4/edit?invitationId=inv_6d2fd5cd-4b1a-403e-b68d-7035eb36644c&page=HWEp-vi-RSFO#

10.1 Actors

1. **Admin:**
   - Responsibilities: Managing vehicles, customers, and bookings.
   - Interactions: The Admin performs tasks such as adding/removing vehicles, managing customer profiles, and confirming bookings.
2. **Customer:**
   - Responsibilities: Receiving booking confirmations and invoices via email.
   - Interactions: The Customer interacts with the system indirectly by receiving emails with booking details and invoices.

10.2 Use Cases

1. **Add Vehicle**
    - Actor: Admin
    - Description: Admin adds a new vehicle into the system's vehicle list
    - Steps: Admin logs in, adds vehicle details, submits form.
    - Product Features Mapped: Vehicle Management - Add vehicle details.
    - Relationship: N/A
2. **Remove Vehicle**
    - Actor: Admin
    - Description: Admin removes a vehicle from the system's vehicle list
    - Steps: Admin logs in, selects vehicle, removes vehicle.
    - Product Features Mapped: Vehicle Management - Remove vehicle details.
    - Relationship: N/A
3. **Update Vehicle**
    - Actor: Admin
    - Description: Admin updates a vehicle's details in the system's vehicle list
    - Steps: Admin logs in, selects vehicle, updates details, submits form.
    - Product Features Mapped: Vehicle Management - Update vehicle details.
    - Relationship: N/A
4. **Create Customer Profile**
    - Actor: Admin
    - Description: Admin creates a new customer profile into the system's customer database
    - Steps: Admin logs in, enters customer details, submits form.
    - Product Features Mapped: Customer Management - Create customer profile.
    - Relationship: N/A
5. **Edit Customer Profile**
    - Actor: Admin
    - Description: Admin edits a customer profile into the system's customer database
    - Steps: Admin logs in, selects customer, updates details, submits form.
    - Product Features Mapped: Customer Management - Edit customer profile.
    - Relationship: N/A
6. **Remove Customer Profile**
    - Actor: Admin
    - Description: Remove customer profile from system.
    - Steps: Admin logs in, selects customer, removes profile.
    - Product Features Mapped: Customer Management - Remove customer profile.
    - Relationship: N/A
7. **View Customer Profile**
    - Actor: Admin

- Description: View details of a customer profile.
- Steps: Admin logs in, selects customer, views profile.
- Product Features Mapped: Customer Management - View customer profile.
- Relationship: N/A

8. **View Customer Rental History**
   - Actor: Admin
   - Description: View customers rental history.
   - Steps: Admin logs in, selects a customer, views rental history.
   - Product Features Mapped: Customer Management - View customer rental history.
   - Relationship: N/A
9. **Search Available Vehicles**
   - Actor: Admin
   - Description: Search for vehicles based on criteria.
   - Steps: Admin logs in, enters search criteria, views available vehicles.
   - Product Features Mapped: Booking System - Search available vehicles.
   - Relationship: This use case has an <<extends>> relationship with Book Vehicle and View Vehicle Information as entering the search criteria and viewing results are mandatory steps before viewing the vehicles' information and booking it. Book Vehicle and View Vehicle Information are optional scenarios that could happen after this use case.
10. **Book Vehicle**
    - Actor: Admin
    - Description: Book a vehicle for a customer.
    - Steps: Admin selects vehicle, enters booking details, confirms booking.
    - Product Features Mapped: Booking System - Book vehicles.
    - Relationship: This use case <<extends>> the Search Available Vehicles and has an <<includes>> relationship with Calculate Booking Cost, Generate Booking Confirmation and Generate Invoice. Selecting the vehicle, entering booking details, and confirming the booking are necessary steps for booking a vehicle. These actions ensure the booking is accurately processed.
11. **View Vehicle Information**
    - Actor: Admin
    - Description: View details of a specific vehicle.
    - Steps: Admin logs in, selects vehicle, views details.
    - Product Features Mapped: Vehicle Management - View vehicle details.
    - Relationship: This use case <<extends>> the Search Available Vehicles as selecting the vehicle is essential to view its information. This step ensures the correct vehicle details are displayed. View Vehicle Information is an optional scenario that could happen after the Search Available Vehicles use case.

12. Calculate Booking Cost

- Actor: Admin
- Description: Calculate the cost of a booking.
- Steps: Admin enters booking details, system calculates cost.
- Product Features Mapped: Booking System - Calculate booking cost.
- Relationship: Book Vehicle <<includes>> this use case and this use case has an <<includes>> relationship with Send Email with Booking Confirmation and Invoice. Entering booking details and viewing the cost are mandatory for calculating the booking cost. These steps ensure the cost calculation is accurate.

13. Generate Booking Confirmation

- Actor: Admin
- Description: Generate booking confirmation for a customer.
- Steps: Admin confirms booking, system generates confirmation.
- Product Features Mapped: Booking System - Generate booking confirmation.
- Relationship: Book Vehicle <<includes>> this use case and this use case has an <<includes>> relationship with Send Email with Booking Confirmation and Invoice. Confirming the booking is essential to generate a booking confirmation. This action ensures the confirmation is accurate and complete.

14. Generate Invoice

- Actor: Admin
- Description: Generate invoice for a customer.
- Steps: Admin confirms booking, system generates invoice.
- Product Features Mapped: Booking System - Generate invoice.
- Relationship: Book Vehicle <<includes>> this use case as generating an invoice is a necessary step after the booking process. This use case also has an <<includes>> relationship with Send Email with Booking Confirmation and Invoice as generating the invoice is a necessary step before sending the email.

15. Send Email with Booking Confirmation and Invoice

- Actor: Customer
- Description: Customer receives an email with the booking confirmation and invoice
- Steps: Admin confirms booking, system sends email with confirmation and invoice.
- Product Features Mapped: Booking System - Send email notifications.
- Relationship: This use case has an <<includes>> relationship with Generate Booking Information and Generate Invoice as generating the booking confirmation and invoice are necessary steps to send the email. These steps ensure the email contains accurate information.

10.3 Details of 2 Use Cases

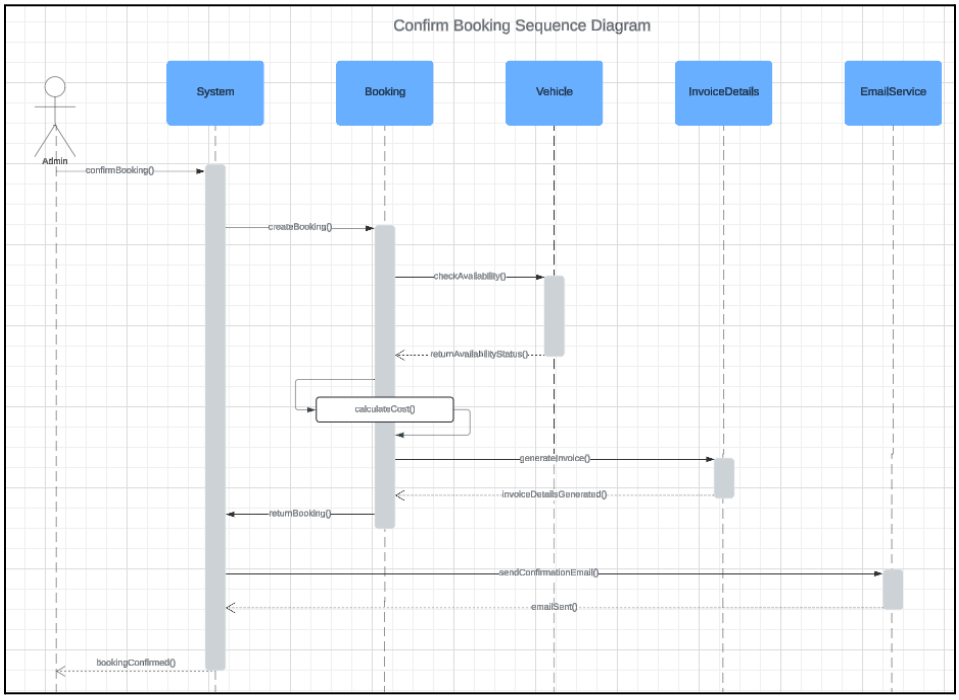
1. **Add/Remove Vehicle:**
    - Actor: Admin
    - Description: Admin adds or removes vehicles in the system.
    - Steps: Admin logs in, navigates to the vehicle management section, adds or removes vehicle details, and submits the form.
    - Product Features Mapped: Vehicle Management - Add or Remove vehicle details.
    - Preconditions:
        - Admin must be logged into the system.
        - Admin must have the necessary permissions to add or remove vehicles.
    - Postconditions:
        - The vehicle is successfully added to or removed from the system.
        - The vehicle list is updated accordingly.

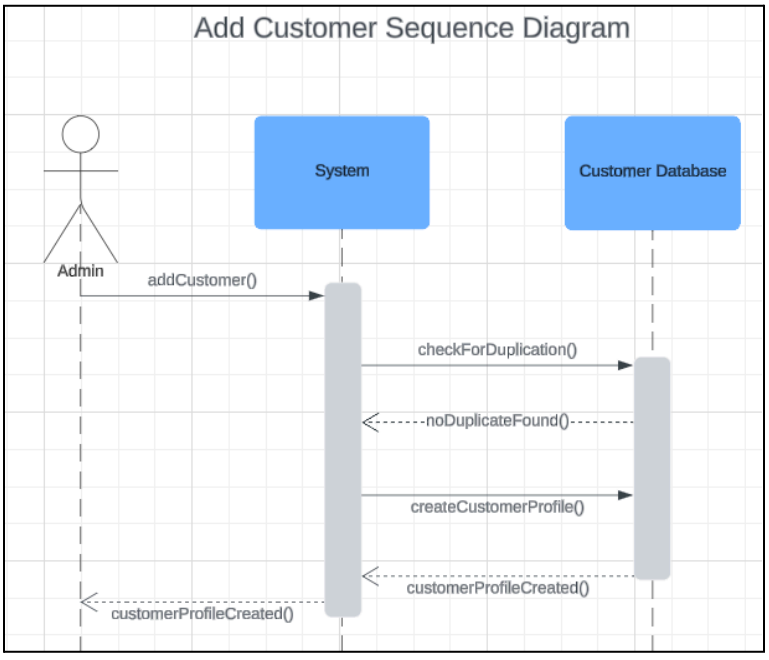
2. **Confirm Booking:**
    - Actor: Admin
    - Description: Admin confirms a booking after checking availability and calculating the cost.
    - Steps: Admin searches for available vehicles, selects a vehicle, confirms the booking, and generates an invoice.
    - Product Features Mapped: Booking System - Confirm bookings, generate invoices.
    - Preconditions:
        - Admin must be logged into the system.
        - Admin must have the necessary permissions to confirm bookings.
        - The vehicle must be available for the selected dates.
    - Postconditions:
        - The booking is successfully confirmed.
        - The customer receives an email with the booking confirmation and invoice.
        - The vehicle's availability status is updated.

11 Sequence Diagrams

11.1 Two Non-Trivial Use Cases

1. Confirm Booking
2. Add Customer





Sequence Diagrams Lucidchart Link:
https://lucid.app/lucidchart/1d27b7f9-5cc6-409c-b8d7-0c23659a32e7/edit?viewport_loc=-1548%2C-845%2C4448%2C1822%2C0_0&invitationId=inv_80289ab0-9d60-4352-a2b7-73303d682ef6

11.2 Confirm Booking Sequence Diagram Relationships

- Admin to System: Admin initiates the booking confirmation process by calling *confirmBooking()*.
- System to Booking: System creates a new booking entry by calling *createBooking()*.
- Booking to Vehicle: Booking checks the availability of the selected vehicle by calling *checkAvailability()*.
- Vehicle to Booking: Vehicle returns the availability status to Booking.
- Booking to Booking: Booking calculates the total cost by calling *calculateCost()*.
- Booking to InvoiceDetails: Booking generates an invoice for the booking by calling *generateInvoice()*.
- InvoiceDetails to Booking: InvoiceDetails returns the generated invoice details to Booking.
- Booking to System: Booking returns the booking details to the system.
- System to EmailService: System sends a confirmation email to the customer by calling *sendConfirmationEmail()*.
- EmailService to System: EmailService confirms that the email has been sent.
- System to Admin: Admin receives confirmation that the booking has been completed with *bookingConfirmed()*.

11.3 Add Customer Sequence Diagram Relationships

- Admin to System: Admin initiates the process to add a new customer by calling *addCustomer()*.
- System to CustomerDatabase: System checks if the customer already exists in the database by calling *checkForDuplication()*.
- CustomerDatabase to System: CustomerDatabase returns the result of the duplication check as a dotted line (*noDuplicateFound()*).
- System to CustomerDatabase: If no duplicate is found, System creates the new customer profile by calling *createCustomerProfile()*.
- CustomerDatabase to System: CustomerDatabase returns the result indicating the customer profile has been created as a dotted line (*customerProfileCreated()*).
- System to Admin: Admin receives confirmation that the customer profile has been created (*customerProfileCreated()*).

11.4 Rationale For Essential Design Decisions

1. **Confirm Booking Sequence Diagram:**
    - Increasing Cohesion: Each component performs a specific role within the booking confirmation process, such as checking availability, calculating costs, generating invoices, and sending emails.

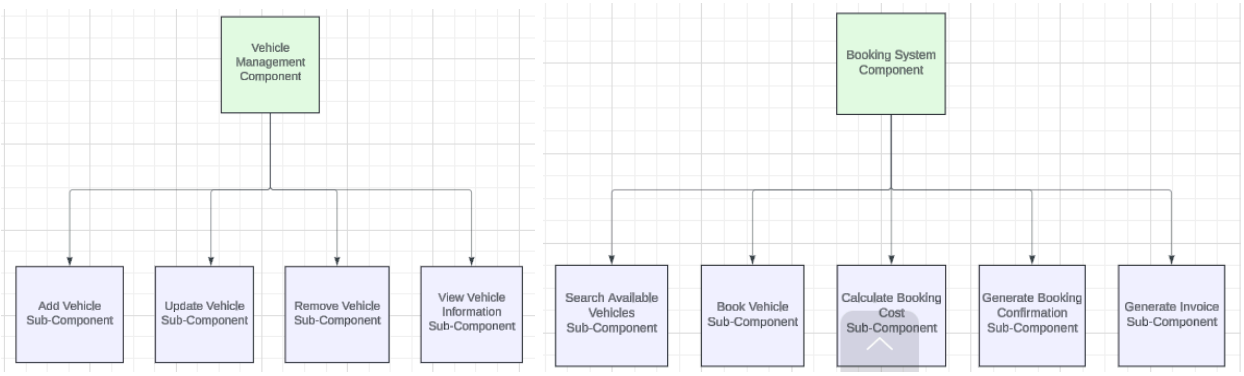
- Reducing Coupling: Clearly defined interactions between components (e.g., System, Booking, Vehicle, InvoiceDetails, EmailService) minimize dependencies, making the system more modular.
- Efficiency Considerations: Efficient data flow and interaction between components enhance overall system performance and reduce redundancy.

2. **Add Customer Sequence Diagram:**
   - Increasing Cohesion: Each component focuses on a single responsibility within the customer addition process, such as creating customer profiles and confirming the creation.
   - Reducing Coupling: The interaction between Admin, System, and CustomerDatabase is clearly defined, promoting modularity and ease of maintenance.
   - Efficiency Considerations: The process ensures efficient creation and confirmation of customer profiles with minimal steps and clear data flow.

12 Components Diagram

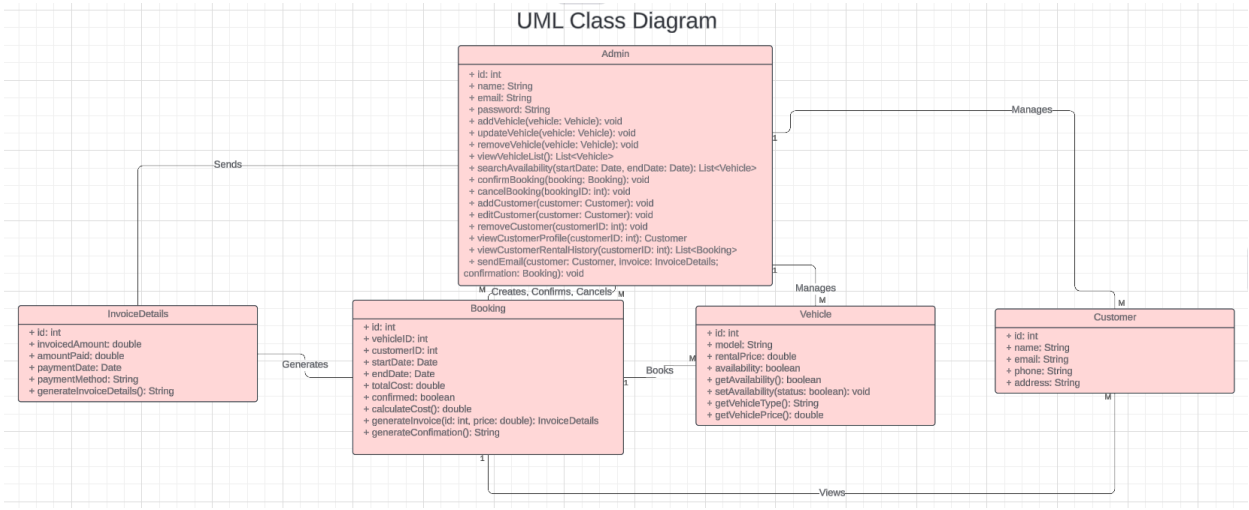
12.1 Two Major Components

1. Vehicle Management Component Diagram
2. Booking System Component Diagram



Component Diagrams Lucidchart Link:
https://lucid.app/lucidchart/893a6634-22f0-433b-aaa7-748cca5819ed/edit?viewport_loc=-2256%2C-653%2C7400%2C3706%2C0_0&invitationId=inv_1a15eb93-7793-49f3-9e58-f52f094f9074

13 Two Non-Trivial Software System Components UML Class Diagram



UML Class Diagram For Two Non-Trivial Software Components Lucidchart Link: https://lucid.app/lucidchart/4ca9c54a-e399-45bb-815d-ab789918c55b/edit?invitationId=inv_534956ea-d5de-4c9e-b0ce-86ca3aa22b3b

1. Booking System Component
    - Classes:
        - Admin
        - Booking
        - Vehicle

2. Customer Management Component Classes
    - Classes:
        - Admin
        - Customer

13.1 Responsibilities of Each Class

- Admin Class:
    - Manages vehicles, bookings, and customer profiles.
    - Methods include adding, updating, and removing vehicles, searching vehicle availability, confirming and canceling bookings, and managing customer profiles.
- Booking Class:
    - Represents a vehicle booking.
    - Attributes include booking details (id, vehicleID, customerID, startDate, endDate, totalCost, confirmed).
    - Methods include calculating cost and generating invoices.
- Vehicle Class:
    - Represents a vehicle.

    - Attributes include vehicle details (id, model, rentalPrice, availability).
    - Methods include getting and setting availability, and retrieving vehicle type and price.
- Customer Class:
    - Represents a customer.
    - Attributes include customer details (id, name, email, phone, address).
- InvoiceDetails Class:
    - Represents invoice details.
    - Attributes include invoice details (id, invoicedAmount, amountPaid, paymentDate, paymentMethod).
    - Methods include generating invoice details.

13.2 Relationships and Rationales

- **Admin to Booking** → Creates, Confirms, Cancels: Admin manages bookings, including creating new bookings, confirming existing bookings, and canceling bookings.
- **Admin to Vehicle** → Manages: Admin manages vehicles, including adding, updating, and removing vehicle details.
- **Admin to Customer** → Manages: Admin manages customer profiles, including adding, editing, and removing customer details.
- **Admin to InvoiceDetails** → Sends: Admin sends invoices to customers.
- **Booking to InvoiceDetails** → Generates: Booking generates invoices based on booking details.
- **Vehicle to Booking**: Books → Vehicle is booked as part of the booking process.
- **Customer to Booking** → Views: Customer can view booking details related to their rental history.

13.3 Rationale For Critical Design Decisions

- Increasing Cohesion: Each class is designed with a single responsibility, ensuring that all methods and attributes within a class are closely related to its primary function.
- Reducing Coupling: By defining clear interfaces and interactions, the design minimizes dependencies between classes, making the system more modular and easier to maintain.
- Code Reuse: Common functionalities like managing vehicles, bookings, and customers are centralized in the Admin class, promoting code reuse and reducing redundancy. The use of methods for specific actions (e.g., addCustomer, confirmBooking) ensures that code can be reused across different parts of the system without duplication.

14 System Input/Output

GitHub Repository: https://github.com/manhamalik/AutoRentNexusCode

14.2 User Interaction Flow

When the program starts, it presents a menu with options for admin login, creating an admin account, and exiting the program. The user is prompted to enter their choice.

```
1. Admin Login
2. Create Admin Account
0. Exit
Enter choice:
```

Choice [2] Create Admin Account

- If the user chooses to create an admin account, the program asks for a username, email, and password.
- An Admin object is created and added to the admins list.
- A success message is displayed: "Admin account created successfully."

```
1. Admin Login
2. Create Admin Account
0. Exit
Enter choice: 2
Enter username: group4_cp317
Enter email: group4_cp317@gmail.com
Enter password: 123456
Admin account created successfully.
```

Choice [1] Admin Login

- The user selects the admin login option.
- The program prompts for the admin username and password.
- The credentials are verified against the existing admins list.
- If the credentials match, the admin is logged in, and a success message is displayed.
- If the credentials do not match, an error message is displayed, and the user is prompted to try again.

```
Admin account created successfully.
1. Admin Login
2. Create Admin Account
0. Exit
Enter choice: 1
Enter username: group4_cp317
Enter password: 123456.
Login successful!
```

After logging in, the admin is presented with a menu with options to add, update, remove, and view vehicles and customers, book a vehicle, view bookings, logout, and exit the program.

Choice [1] Add Vehicle

- The admin selects the option to add a vehicle.
- The program prompts for the vehicle model, rental price, and vehicle type.
- A Vehicle object is created with a unique ID and added to the vehicles list.
- A success message is displayed: "Vehicle added successfully."
- The program also displays the ID of the newly added vehicle.

```
1. Admin Login
2. Create Admin Account
0. Exit
Enter choice: 1
Enter username: group4_cp317
Enter password: 123456
Login successful!

1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 1
Enter vehicle model: 911 Carrera Cabriolet
Enter vehicle type (SUV, TRUCK, VAN, CONVERTIBLE, WAGON, EV, ECO
NOMY, SPORTS): CONVERTIBLE
Vehicle added successfully.
Vehicle added successfully.
The vehicle ID for 911 Carrera Cabriolet is: 1
```

Choice [2] Update Vehicle

- The admin selects the option to update a vehicle.
- The program prompts for the vehicle ID to update.
- The vehicle is retrieved from the vehicles list using the ID.
- The program prompts for the new model, rental price, and vehicle type.
- The vehicle details are updated, and a success message is displayed.

```
The vehicle ID for 911 Carrera Cabriolet is: 1

1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 2
Enter vehicle ID to update: 1
Enter new model: 911 Carrera GTS Cabriolet
Vehicle updated successfully.
```

Choice [4] View Vehicles

- The admin selects the option to view all vehicles.
- The program iterates through the vehicles list and displays the details of each vehicle.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 4
Vehicles:

Vehicle ID: 1
---------------
Model: '911 Carrera GTS Cabriolet'
Price/day: $70.0
Available: true
Type: Convertible
```

Choice [5] Add Customer

- The admin selects the option to add a customer.
- The program prompts for the customer name, email, phone, and address.
- A Customer object is created with a unique ID and added to the customers list.
- A success message is displayed: "Customer profile created successfully."
- The program also displays the ID of the newly added customer.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 5
Enter customer name: Emad
Enter customer email: emohammed@wlu.ca
Enter customer phone: 1234567890
Enter customer address: 136 waterloo st
Customer profile created successfully.
Customer profile created successfully.
The customer ID for Emad is: 1
```

Choice [6] Update Customer

- The admin selects the option to update a customer.
- The program prompts for the customer ID to update.
- The customer is retrieved from the customers list using the ID.
- The program prompts for the new name, email, phone, and address.
- The customer details are updated, and a success message is displayed.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 6
Enter customer ID to update: 1
Enter new name: Mohammed
Enter new email: emohammed@wlu.ca
Enter new phone: 7673737388
Enter new address: 9892 waterloo Dr
Customer profile updated successfully.
```

Choice [8] View Customers

- The admin selects the option to view all customers.
- The program iterates through the customers list and displays the details of each customer.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 8
Customers:

Customer ID: 1
----------------
Name: Mohammed
Email: emohammed@wlu.ca
Phone: 7673737388
Address: 9892 waterloo Dr
```

Choice [9] Book Vehicle

- The admin selects the option to book a vehicle.
- The program prompts for the customer ID and vehicle ID.
- The customer and vehicle are retrieved from their respective lists using the IDs.
- The program prompts for the start date and end date of the booking.
- The booking is created, and the details are stored in the bookings list.
- A success message is displayed, and the invoice is generated and sent to the customer.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 9
Enter customer ID: 1
Enter vehicle ID: 1
Enter start date (yyyy-MM-dd): 2024-08-02
Enter end date (yyyy-MM-dd): 2024-08-04
Booking confirmed. Total cost: $140.0
Sending email to emohammed@wlu.ca:
Dear Mohammed,

Thank you for your booking. Here are your invoice details:
Invoice ID: 1
----------------
Invoiced Amount: $140.0
Amount Paid: $140.0
Payment Date: Fri Aug 02 23:14:35 GMT 2024
Payment Method: Credit Card

Booking ID: 1
----------------
Vehicle ID: 1
Customer ID: 1
Start Date:Fri Aug 02 00:00:00 GMT 2024
End Date:Sun Aug 04 00:00:00 GMT 2024
Total Cost: $140.0
Confirmed: true


Best regards,
Your Car Rental Service
```

Choice [10] View Bookings

- The admin selects the option to view all bookings.
- The program iterates through the bookings list and displays the details of each booking.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 10
Bookings:

Booking ID: 1
---------------
Vehicle ID: 1
Customer ID: 1
Start Date:Fri Aug 02 00:00:00 GMT 2024
End Date:Sun Aug 04 00:00:00 GMT 2024
Total Cost: $140.0
Confirmed: true
```

Choice [3] Remove Vehicle

- The admin selects the option to remove a vehicle.
- The program prompts for the vehicle ID to remove.
- The vehicle is retrieved from the vehicles list using the ID.
- The vehicle is removed from the list, and a success message is displayed.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 3
Enter vehicle ID to remove: 1
Vehicle removed successfully.
```

Choice [7] Remove Customer

- The admin selects the option to remove a customer.
- The program prompts for the customer ID to remove.
- The customer is retrieved from the customers list using the ID.
- The customer is removed from the list, and a success message is displayed.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 7
Enter customer ID to remove: 1
Customer profile removed successfully.
```

Choice [11] Logout

- The admin selects the logout option.
- The current admin session ends, and the program returns to the initial menu.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 11
1. Admin Login
2. Create Admin Account
0. Exit
Enter choice:
```

Choice [0] Exit

- The user selects the exit option.
- The program terminates.

```
1. Add Vehicle
2. Update Vehicle
3. Remove Vehicle
4. View Vehicles
5. Add Customer
6. Update Customer
7. Remove Customer
8. View Customers
9. Book Vehicle
10. View Bookings
11. Logout
0. Exit
Enter choice: 11
1. Admin Login
2. Create Admin Account
0. Exit
Enter choice: 0
```

14.3 Interactions Between the User and the Software System

AutoRentNexus allows users to manage vehicle rentals through a series of interactive steps. Upon starting the program, users can either create an admin account or log in. Admins can then manage vehicles by adding, updating, or removing vehicle details and viewing the vehicle list. They can also manage customer profiles similarly, with options to add, update, remove, and view customer information. The booking process involves selecting a vehicle, entering booking dates, and confirming the booking, generating an invoice and sending it to the customer via email. Admins can view all bookings, ensuring efficient rental management. Finally, admins can log out or exit the system, returning to the main menu or terminating the program.