CarConnect, a Car Rental Platform

Creating Tables:-

```
import mysql.connector

def create_tables():
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="HARSHA1@singh",
        database="carrental"
    )
    cursor = conn.cursor()

    create_customer_table = """
    CREATE TABLE Customer (
        CustomerID INT AUTO_INCREMENT PRIMARY KEY,
        FirstName VARCHAR(255),
        LastName VARCHAR(255),
        Email VARCHAR(255),
        PhoneNumber VARCHAR(20),
        Address VARCHAR(255),
        Username VARCHAR(50) UNIQUE,
        Password VARCHAR(255),
        RegistrationDate DATE
    )
    """

    create_vehicle_table = """
    CREATE TABLE Vehicle (
        VehicleID INT AUTO_INCREMENT PRIMARY KEY,
        Model VARCHAR(255),
        Make VARCHAR(255),
        Year INT,
        Color VARCHAR(50),
        RegistrationNumber VARCHAR(50) UNIQUE,
        Availability BOOLEAN,
        DailyRate DECIMAL(10, 2)
    )
    """

    create_reservation_table = """
    CREATE TABLE Reservation (
        ReservationID INT AUTO_INCREMENT PRIMARY KEY,
        CustomerID INT,
        VehicleID INT,
        StartDate DATETIME,
        EndDate DATETIME,
        TotalCost DECIMAL(10, 2),
        Status VARCHAR(50),
        FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
        FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)
    )
    """

    create_admin_table = """
    CREATE TABLE Admin (
        AdminID INT AUTO_INCREMENT PRIMARY KEY,
        FirstName VARCHAR(255),
        LastName VARCHAR(255),
        Email VARCHAR(255),
```

```
        PhoneNumber VARCHAR(20),
        Username VARCHAR(50) UNIQUE,
        Password VARCHAR(255),
        Role VARCHAR(50),
        JoinDate DATE
    )
    """

    try:

        cursor.execute(create_customer_table)
        cursor.execute(create_vehicle_table)
        cursor.execute(create_reservation_table)
        cursor.execute(create_admin_table)
        conn.commit()
        print("Tables created successfully!")
    except mysql.connector.Error as err:
        print("Error:", err)
    finally:

        cursor.close()
        conn.close()

if __name__ == "__main__":
    create_tables()
```

Model/ Entity Classes:-

```
class Customer:
    def __init__(self, customer_id, first_name, last_name, email, phone_number,
address, username, password,
                 registration_date):
        self.__customer_id = customer_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__email = email
        self.__phone_number = phone_number
        self.__address = address
        self.__username = username
        self.__password = password
        self.__registration_date = registration_date


    def get_customer_id(self):
        return self.__customer_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_email(self):
        return self.__email

    def get_phone_number(self):
        return self.__phone_number

    def get_address(self):
        return self.__address
```

```
    def get_username(self):
        return self.__username

    def get_password(self):
        return self.__password

    def get_registration_date(self):
        return self.__registration_date

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_email(self, email):
        self.__email = email

    def set_phone_number(self, phone_number):
        self.__phone_number = phone_number

    def set_address(self, address):
        self.__address = address

    def set_username(self, username):
        self.__username = username

    def set_password(self, password):
        self.__password = password


class Vehicle:
    def __init__(self, vehicle_id, model, make, year, color, registration_number,
availability, daily_rate):
        self.__vehicle_id = vehicle_id
        self.__model = model
        self.__make = make
        self.__year = year
        self.__color = color
        self.__registration_number = registration_number
        self.__availability = availability
        self.__daily_rate = daily_rate

    def get_vehicle_id(self):
        return self.__vehicle_id

    def get_model(self):
        return self.__model

    def get_make(self):
        return self.__make

    def get_year(self):
        return self.__year

    def get_color(self):
        return self.__color

    def get_registration_number(self):
        return self.__registration_number

    def is_available(self):
        return self.__availability
```

```
    def get_daily_rate(self):
        return self.__daily_rate

    def set_model(self, model):
        self.__model = model

    def set_make(self, make):
        self.__make = make

    def set_year(self, year):
        self.__year = year

    def set_color(self, color):
        self.__color = color

    def set_registration_number(self, registration_number):
        self.__registration_number = registration_number

    def set_availability(self, availability):
        self.__availability = availability

    def set_daily_rate(self, daily_rate):
        self.__daily_rate = daily_rate


class Reservation:
    def __init__(self, reservation_id, customer_id, vehicle_id, start_date,
end_date, total_cost, status):
        self.__reservation_id = reservation_id
        self.__customer_id = customer_id
        self.__vehicle_id = vehicle_id
        self.__start_date = start_date
        self.__end_date = end_date
        self.__total_cost = total_cost
        self.__status = status

    # Getters
    def get_reservation_id(self):
        return self.__reservation_id

    def get_customer_id(self):
        return self.__customer_id

    def get_vehicle_id(self):
        return self.__vehicle_id

    def get_start_date(self):
        return self.__start_date

    def get_end_date(self):
        return self.__end_date

    def get_total_cost(self):
        return self.__total_cost

    def get_status(self):
        return self.__status

    def set_customer_id(self, customer_id):
        self.__customer_id = customer_id

    def set_vehicle_id(self, vehicle_id):
        self.__vehicle_id = vehicle_id
```

```
    def set_start_date(self, start_date):
        self.__start_date = start_date

    def set_end_date(self, end_date):
        self.__end_date = end_date

    def set_total_cost(self, total_cost):
        self.__total_cost = total_cost

    def set_status(self, status):
        self.__status = status


class Admin:
    def __init__(self, admin_id, first_name, last_name, email, phone_number,
username, password, role, join_date):
        self.__admin_id = admin_id
        self.__first_name = first_name
        self.__last_name = last_name
        self.__email = email
        self.__phone_number = phone_number
        self.__username = username
        self.__password = password
        self.__role = role
        self.__join_date = join_date

    def get_admin_id(self):
        return self.__admin_id

    def get_first_name(self):
        return self.__first_name

    def get_last_name(self):
        return self.__last_name

    def get_email(self):
        return self.__email

    def get_phone_number(self):
        return self.__phone_number

    def get_username(self):
        return self.__username

    def get_password(self):
        return self.__password

    def get_role(self):
        return self.__role

    def get_join_date(self):
        return self.__join_date

    def set_first_name(self, first_name):
        self.__first_name = first_name

    def set_last_name(self, last_name):
        self.__last_name = last_name

    def set_email(self, email):
        self.__email = email

    def set_phone_number(self, phone_number):
```

```
        self.__phone_number = phone_number

    def set_username(self, username):
        self.__username = username

    def set_password(self, password):
        self.__password = password

    def set_role(self, role):
        self.__role = role
```

Customer Service:

```
class ICustomerService(ABC):
    @abstractmethod
    def get_customer_by_id(self, customer_id):
        pass

    @abstractmethod
    def get_customer_by_username(self, username):
        pass

    @abstractmethod
    def register_customer(self, customer_data):
        pass

    @abstractmethod
    def update_customer(self, customer_id, updated_data):
        pass

    @abstractmethod
    def delete_customer(self, customer_id):
        pass
```

Vehicle Service:

```
class IVehicleService(ABC):
    @abstractmethod
    def get_vehicle_by_id(self, vehicle_id):
        pass

    @abstractmethod
    def get_available_vehicles(self):
        pass

    @abstractmethod
    def add_vehicle(self, vehicle_data):
        pass

    @abstractmethod
    def update_vehicle(self, vehicle_id, updated_vehicle_data):
        pass

    @abstractmethod
    def remove_vehicle(self, vehicle_id):
        pass
```

Reservation Service:

```
class IReservationService(ABC):
    @abstractmethod
    def get_reservation_by_id(self, reservation_id):
        pass

    @abstractmethod
    def get_reservations_by_customer_id(self, customer_id):
        pass

    @abstractmethod
    def create_reservation(self, reservation_data):
        pass

    @abstractmethod
    def update_reservation(self, reservation_id, updated_reservation_data):
        pass

    @abstractmethod
    def cancel_reservation(self, reservation_id):
        pass
```

Admin Service:

```
class IAdminService(ABC):
    @abstractmethod
    def get_admin_by_id(self, admin_id):
        pass

    @abstractmethod
    def get_admin_by_username(self, username):
        pass

    @abstractmethod
    def register_admin(self, admin_data):
        pass

    @abstractmethod
    def update_admin(self, admin_id, updated_admin_data):
        pass

    @abstractmethod
    def delete_admin(self, admin_id):
        pass
```

Database Context:

```
class DatabaseContext:
    def __init__(self, host, user, password, database):
        self.connection = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )

    def __del__(self):
        if self.connection.is_connected():
            self.connection.close()

    def execute_query(self, query, params=None):
        cursor = self.connection.cursor(dictionary=True)
```

```
        if params:
            cursor.execute(query, params)
        else:
            cursor.execute(query)
        return cursor
```

Interfaces:-

ICustomerService:

```
class CustomerService(ICustomerService):
    def __init__(self, database_context):
        self.database_context = database_context

    def get_customer_by_id(self, customer_id):
        return self.database_context.get_customer_by_id(customer_id)

    def get_customer_by_username(self, username):
        return self.database_context.get_customer_by_username(username)

    def register_customer(self, customer_data):
        return self.database_context.register_customer(customer_data)

    def update_customer(self, customer_id, updated_data):
        return self.database_context.update_customer(customer_id, updated_data)

    def delete_customer(self, customer_id):
        return self.database_context.delete_customer(customer_id)
```

IVehicleService:

```
class VehicleService(IVehicleService):
    def __init__(self, database_context):
        self.database_context = database_context

    def get_vehicle_by_id(self, vehicle_id):
        return self.database_context.get_vehicle_by_id(vehicle_id)

    def get_available_vehicles(self):
        return self.database_context.get_available_vehicles()

    def add_vehicle(self, vehicle_data):
        return self.database_context.add_vehicle(vehicle_data)

    def update_vehicle(self, vehicle_id, updated_vehicle_data):
        return self.database_context.update_vehicle(vehicle_id,
updated_vehicle_data)

    def remove_vehicle(self, vehicle_id):
        return self.database_context.remove_vehicle(vehicle_id)
```

IReservationService:

```
class ReservationService(IReservationService):
    def __init__(self, database_context):
        self.database_context = database_context

    def get_reservation_by_id(self, reservation_id):
        return self.database_context.get_reservation_by_id(reservation_id)
```

```
    def get_reservations_by_customer_id(self, customer_id):
        return self.database_context.get_reservations_by_customer_id(customer_id)

    def create_reservation(self, reservation_data):
        return self.database_context.create_reservation(reservation_data)

    def update_reservation(self, reservation_id, updated_reservation_data):
        return self.database_context.update_reservation(reservation_id,
updated_reservation_data)

    def cancel_reservation(self, reservation_id):
        return self.database_context.cancel_reservation(reservation_id)
```

IAdminService:

```
class AdminService(IAdminService):
    def __init__(self, database_context):
        self.database_context = database_context

    def get_admin_by_id(self, admin_id):
        return self.database_context.get_admin_by_id(admin_id)

    def get_admin_by_username(self, username):
        return self.database_context.get_admin_by_username(username)

    def register_admin(self, admin_data):
        return self.database_context.register_admin(admin_data)

    def update_admin(self, admin_id, updated_admin_data):
        return self.database_context.update_admin(admin_id, updated_admin_data)

    def delete_admin(self, admin_id):
        return self.database_context.delete_admin(admin_id)
```

Customer Operations

```
def get_customer_by_id(self, customer_id):
    query = "SELECT * FROM Customer WHERE CustomerID = %s"
    cursor = self.execute_query(query, (customer_id,))
    return cursor.fetchone()

def get_customer_by_username(self, username):
    query = "SELECT * FROM Customer WHERE Username = %s"
    cursor = self.execute_query(query, (username,))
    return cursor.fetchone()

def register_customer(self, customer_data):
    query = "INSERT INTO Customer (FirstName, LastName, Email, PhoneNumber,
Address, Username, Password, RegistrationDate) VALUES (%s, %s, %s, %s, %s, %s, %s,
%s)"
    cursor = self.execute_query(query, customer_data)
    self.connection.commit()
    return cursor.lastrowid

def update_customer(self, customer_id, updated_data):
    query = "UPDATE Customer SET FirstName = %s, LastName = %s, Email = %s,
PhoneNumber = %s, Address = %s, Username = %s, Password = %s, RegistrationDate = %s
WHERE CustomerID = %s"
    updated_data += (customer_id,)
```

```
    cursor = self.execute_query(query, updated_data)
    self.connection.commit()
    return cursor.rowcount

def delete_customer(self, customer_id):
    query = "DELETE FROM Customer WHERE CustomerID = %s"
    cursor = self.execute_query(query, (customer_id,))
    self.connection.commit()
    return cursor.rowcount
```

Vehicle operations

```
def get_vehicle_by_id(self, vehicle_id):
    query = "SELECT * FROM Vehicle WHERE VehicleID = %s"
    cursor = self.execute_query(query, (vehicle_id,))
    return cursor.fetchone()

def get_available_vehicles(self):
    query = "SELECT * FROM Vehicle WHERE Availability = 1"
    cursor = self.execute_query(query)
    return cursor.fetchall()

def add_vehicle(self, vehicle_data):
    query = "INSERT INTO Vehicle (Model, Make, Year, Color, RegistrationNumber,
Availability, DailyRate) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    cursor = self.execute_query(query, vehicle_data)
    self.connection.commit()
    return cursor.lastrowid

def update_vehicle(self, vehicle_id, updated_vehicle_data):
    query = "UPDATE Vehicle SET Model = %s, Make = %s, Year = %s, Color = %s,
RegistrationNumber = %s, Availability = %s, DailyRate = %s WHERE VehicleID = %s"
    updated_vehicle_data += (vehicle_id,)
    cursor = self.execute_query(query, updated_vehicle_data)
    self.connection.commit()
    return cursor.rowcount

def remove_vehicle(self, vehicle_id):
    query = "DELETE FROM Vehicle WHERE VehicleID = %s"
    cursor = self.execute_query(query, (vehicle_id,))
    self.connection.commit()
    return cursor.rowcount
```

Reservation operations

```
def get_reservation_by_id(self, reservation_id):
    query = "SELECT * FROM Reservation WHERE ReservationID = %s"
    cursor = self.execute_query(query, (reservation_id,))
    return cursor.fetchone()

def get_reservations_by_customer_id(self, customer_id):
    query = "SELECT * FROM Reservation WHERE CustomerID = %s"
    cursor = self.execute_query(query, (customer_id,))
    return cursor.fetchall()

def create_reservation(self, reservation_data):
    query = "INSERT INTO Reservation (CustomerID, VehicleID, StartDate, EndDate,
TotalCost, Status) VALUES (%s, %s, %s, %s, %s, %s)"
    cursor = self.execute_query(query, reservation_data)
    self.connection.commit()
    return cursor.lastrowid
```

```
def update_reservation(self, reservation_id, updated_reservation_data):
    query = "UPDATE Reservation SET CustomerID = %s, VehicleID = %s, StartDate =
%s, EndDate = %s, TotalCost = %s, Status = %s WHERE ReservationID = %s"
    updated_reservation_data += (reservation_id,)
    cursor = self.execute_query(query, updated_reservation_data)
    self.connection.commit()
    return cursor.rowcount

def cancel_reservation(self, reservation_id):
    query = "DELETE FROM Reservation WHERE ReservationID = %s"
    cursor = self.execute_query(query, (reservation_id,))
    self.connection.commit()
    return cursor.rowcount
```

Admin operations

```
def get_admin_by_id(self, admin_id):
    query = "SELECT * FROM Admin WHERE AdminID = %s"
    cursor = self.execute_query(query, (admin_id,))
    return cursor.fetchone()

def get_admin_by_username(self, username):
    query = "SELECT * FROM Admin WHERE Username = %s"
    cursor = self.execute_query(query, (username,))
    return cursor.fetchone()

def register_admin(self, admin_data):
    query = "INSERT INTO Admin (FirstName, LastName, Email, PhoneNumber, Username,
Password, Role, JoinDate) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
    cursor = self.execute_query(query, admin_data)
    self.connection.commit()
    return cursor.lastrowid

def update_admin(self, admin_id, updated_admin_data):
    query = "UPDATE Admin SET FirstName = %s, LastName = %s, Email = %s,
PhoneNumber = %s, Username = %s, Password = %s, Role = %s, JoinDate = %s WHERE
AdminID = %s"
    updated_admin_data += (admin_id,)
    cursor = self.execute_query(query, updated_admin_data)
    self.connection.commit()
    return cursor.rowcount

def delete_admin(self, admin_id):
    query = "DELETE FROM Admin WHERE AdminID = %s"
    cursor = self.execute_query(query, (admin_id,))
    self.connection.commit()
    return cursor.rowcount
```

Custom Exceptions:-

```
class AuthenticationException(Exception):

    def __init__(self, message="Authentication failed. Incorrect username or
password."):
        self.message = message
        super().__init__(self.message)


class ReservationException(Exception):

    def __init__(self, message="Reservation failed."):
        self.message = message
        super().__init__(self.message)


class VehicleNotFoundException(Exception):

    def __init__(self, message="Vehicle not found."):
        self.message = message
        super().__init__(self.message)


class AdminNotFoundException(Exception):

    def __init__(self, message="Admin not found."):
        self.message = message
        super().__init__(self.message)


class InvalidInputException(Exception):

    def __init__(self, message="Invalid input data."):
        self.message = message
        super().__init__(self.message)


class DatabaseConnectionException(Exception):

    def __init__(self, message="Database connection failed."):
        self.message = message
        super().__init__(self.message)

def login(username, password):
    if not is_valid_credentials(username, password):
        raise AuthenticationException()

def make_reservation(vehicle_id):
    if not is_vehicle_available(vehicle_id):
        raise ReservationException()

def get_vehicle_details(vehicle_id):
    vehicle = find_vehicle(vehicle_id)
    if not vehicle:
        raise VehicleNotFoundException()

def get_admin_details(admin_id):
    admin = find_admin(admin_id)
    if not admin:
        raise AdminNotFoundException()

def validate_input(data):
    if not is_valid(data):
```

```
        raise InvalidInputException()

def connect_to_database():
    if not is_database_connected():
        raise DatabaseConnectionException()
```

Main.py:-

```
from dao import CustomerService, VehicleService, ReservationService, AdminService
from exception import AuthenticationException, ReservationException,
VehicleNotFoundException, AdminNotFoundException, InvalidInputException,
DatabaseConnectionException


class CarConnect:
    def __init__(self):
        self.db_context = DatabaseContext("localhost", "root", "HARSHA1@singh",
"carrental")

    def display_menu(self):
        print("\nWelcome to CarConnect Management System")
        print("1. Customer Management")
        print("2. Vehicle Management")
        print("3. Reservation Management")
        print("4. Admin Management")
        print("5. Exit")

    def run(self):
        while True:
            self.display_menu()
            choice = input("Enter your choice: ")

            if choice == "1":
                self.customer_management()
            elif choice == "2":
                self.vehicle_management()
            elif choice == "3":
                self.reservation_management()
            elif choice == "4":
                self.admin_management()
            elif choice == "5":
                print("Exiting CarConnect Management System. Goodbye!")
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    def customer_management(self):
        customer_service = CustomerService(self.db_context)
        while True:
            print("\nCustomer Management:")
            print("1. Register Customer")
            print("2. Update Customer")
            print("3. Delete Customer")
            print("4. Show customer by id:")
            print("5. Go back")
            choice = input("Enter your choice: ")

            if choice == "1":
                first_name = input("Enter first name: ")
                last_name = input("Enter last name: ")
                email = input("Enter email: ")
                phone_number = input("Enter phone number: ")
```

```
                address = input("Enter address: ")
                username = input("Enter username: ")
                password = input("Enter password: ")
                registration_date = input("Enter registration date (YYYY-MM-DD): ")
                customer_data = (first_name, last_name, email, phone_number,
address, username, password, registration_date)
                customer_service.register_customer(customer_data)
                print("Customer registered successfully")
            elif choice == "2":
                customer_id = input("Enter customer ID: ")
                first_name = input("Enter first name: ")
                last_name = input("Enter last name: ")
                email = input("Enter email: ")
                phone_number = input("Enter phone number: ")
                address = input("Enter address: ")
                username = input("Enter username: ")
                password = input("Enter password: ")
                customer_data = (first_name, last_name, email, phone_number,
address, username, password, customer_id)
                customer_service.update_customer(customer_data)
            elif choice == "3":
                customer_id = input("Enter customer ID: ")
                customer_service.delete_customer(customer_id)
            elif choice == "4":
                customer_id = input("Enter Customer ID: ")
                customer_data = customer_service.get_customer_by_id(customer_id)
                if customer_data:
                    print("Customer Details:")
                    print(customer_data)
                else:
                    print("Customer not found.")


            elif choice == "5":
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    def vehicle_management(self):
        vehicle_service = VehicleService(self.db_context)
        while True:
            print("\nVehicle Management:")
            print("1. Add Vehicle")
            print("2. Update Vehicle")
            print("3. Remove Vehicle")
            print("4. Get vehicle by ID")
            print("5. Go back")
            choice = input("Enter your choice: ")

            if choice == "1":
                model = input("Enter vehicle model: ")
                make = input("Enter vehicle make: ")
                year = input("Enter manufacturing year: ")
                color = input("Enter vehicle color: ")
                registration_number = input("Enter registration number: ")
                availability = input("Is the vehicle available? (True/False): ")
                daily_rate = input("Enter daily rental rate: ")
                vehicle_data = (model, make, year, color, registration_number,
availability, daily_rate)
                vehicle_service.add_vehicle(vehicle_data)

            elif choice == "2":
                vehicle_id = input("Enter vehicle ID: ")
                model = input("Enter vehicle model: ")
```

```
                make = input("Enter vehicle make: ")
                year = input("Enter manufacturing year: ")
                color = input("Enter vehicle color: ")
                registration_number = input("Enter registration number: ")
                availability = input("Is the vehicle available? (True/False): ")
                daily_rate = input("Enter daily rental rate: ")
                vehicle_data = (model, make, year, color, registration_number,
availability, daily_rate, vehicle_id)
                vehicle_service.update_vehicle(vehicle_data)
            elif choice == "3":
                vehicle_id = input("Enter vehicle ID: ")
                vehicle_service.remove_vehicle(vehicle_id)
            elif choice == "4":
                vehicle_id = input("Enter Vehicle ID: ")
                vehicle_data = vehicle_service.get_vehicle_by_id(vehicle_id)
                if vehicle_data:
                    print("Vehicle Details:")
                    print(vehicle_data)
                else:
                    print("Vehicle not found.")
            elif choice == "5":
                break
            else:
                print("Invalid choice. Please enter a valid option.")

    def reservation_management(self):
        reservation_service = ReservationService(self.db_context)
        while True:
            print("\nReservation Management:")
            print("1. Create Reservation")
            print("2. Update Reservation")
            print("3. Cancel Reservation")
            print("4. Go back")
            choice = input("Enter your choice: ")

            if choice == "1":
                customer_id = input("Enter customer ID: ")
                vehicle_id = input("Enter vehicle ID: ")
                start_date = input("Enter start date (YYYY-MM-DD): ")
                end_date = input("Enter end date (YYYY-MM-DD): ")
                total_cost = input("Enter total cost: ")
                status = input("Enter status(A/N): ")
                reservation_data = (customer_id, vehicle_id, start_date, end_date,
total_cost, status)
                reservation_service.create_reservation(reservation_data)
                print("Reservation Created" if status == "A" else "Car not
available")
            elif choice == "2":
                reservation_id = input("Enter reservation ID: ")
                customer_id = input("Enter customer ID: ")
                vehicle_id = input("Enter vehicle ID: ")
                start_date = input("Enter start date (YYYY-MM-DD): ")
                end_date = input("Enter end date (YYYY-MM-DD): ")
                total_cost = input("Enter total cost: ")
                status = input("Enter status: ")
                reservation_data = (customer_id, vehicle_id, start_date, end_date,
total_cost, status, reservation_id)
                reservation_service.update_reservation(reservation_data)
            elif choice == "3":
                reservation_id = input("Enter reservation ID: ")
                reservation_service.cancel_reservation(reservation_id)
            elif choice == "4":
                break
            else:
```

```
                print("Invalid choice. Please enter a valid option.")

    def admin_management(self):
        admin_service = AdminService(self.db_context)
        while True:
            print("\nAdmin Management:")
            print("1. Register Admin")
            print("2. Update Admin")
            print("3. Delete Admin")
            print("4. Go back")
            choice = input("Enter your choice: ")

            if choice == "1":
                first_name = input("Enter first name: ")
                last_name = input("Enter last name: ")
                email = input("Enter email: ")
                phone_number = input("Enter phone number: ")
                username = input("Enter username: ")
                password = input("Enter password: ")
                role = input("Enter role: ")
                join_date = input("Enter join date (YYYY-MM-DD): ")
                admin_data = (first_name, last_name, email, phone_number, username,
password, role, join_date)
                admin_service.register_admin(admin_data)
            elif choice == "2":
                admin_id = input("Enter admin ID: ")
                first_name = input("Enter first name: ")
                last_name = input("Enter last name: ")
                email = input("Enter email: ")
                phone_number = input("Enter phone number: ")
                username = input("Enter username: ")
                password = input("Enter password: ")
                role = input("Enter role: ")
                admin_data = (first_name, last_name, email, phone_number, username,
password, role, admin_id)
                admin_service.update_admin(admin_data)
            elif choice == "3":
                admin_id = input("Enter admin ID: ")
                admin_service.delete_admin(admin_id)
            elif choice == "4":
                break
            else:
                print("Invalid choice. Please enter a valid option.")

if __name__ == "__main__":
    car_connect = CarConnect()
    car_connect.run()
```