**CarConnect Case Study**

**SQL Schema Creation:**

Create database CarConnectDB;

Use CarConnectDB;

**Creation of Customer Table**

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL ,

LastName VARCHAR(50) NOT NULL ,

Email VARCHAR(100),

PhoneNumber VARCHAR(18),

Address VARCHAR(255),

Username VARCHAR(50) NOT NULL UNIQUE,

Password VARCHAR(255)NOT NULL ,

RegistrationDate DATE

);

**Creation of Vehicle Table**

CREATE TABLE Vehicle (

VehicleID INT PRIMARY KEY,

Model VARCHAR(100),

Make VARCHAR(100),

Year INT,

Color VARCHAR(50),

RegistrationNumber VARCHAR(20)NOT NULL UNIQUE,

Availability BOOLEAN,

DailyRate DECIMAL(10, 2)

);

**Creation of Reservation Table**

CREATE TABLE Reservation (

ReservationID INT PRIMARY KEY,

CustomerID INT,

VehicleID INT,

StartDate DATETIME,

EndDate DATETIME,

TotalCost DECIMAL(10, 2),

Status ENUM('pending','confirmed', 'completed'),

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID) ON DELETE CASCADE,

FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID) ON DELETE CASCADE

);

**Creation of Admin Table**

CREATE TABLE Admin (

AdminID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

PhoneNumber VARCHAR(18),

Username VARCHAR(50) UNIQUE,

Password VARCHAR(255),

Role ENUM('Super\_admin','fleet manager'),

JoinDate DATE

);

**Inserting Data into Customer Table**

INSERT INTO Customer (CustomerID, FirstName, LastName, Email, PhoneNumber, Address, Username, Password, RegistrationDate)

VALUES

(1, 'Ramesh', 'Patel', 'ramesh@gmail.com', '+91-1234567890', '123 Street, Mumbai', 'ramesh123', 'password123', '2024-04-01'),

(2, 'Priya', 'Sharma', 'priya@gmail.com', '+91-9876543210', '456 Street, Delhi', 'priya456', 'password456', '2024-04-02'),

(3, 'Amit', 'Singh', 'amit@gmail.com', '+91-8765432109', '789 Street, Kolkata', 'amit789', 'password789', '2024-04-03'),

(4, 'Neha', 'Gupta', 'neha@gmail.com', '+91-7654321098', '456 Street, Bangalore', 'neha456', 'password456', '2024-04-04'),

(5, 'Rajesh', 'Kumar', 'rajesh@gmail.com', '+91-6543210987', '789 Street, Hyderabad', 'rajesh789', 'password789', '2024-04-05'),

(6, 'Pooja', 'Verma', 'pooja@gmail.com', '+91-5432109876', '123 Street, Chennai', 'pooja123', 'password123', '2024-04-06'),

(7, 'Sandeep', 'Yadav', 'sandeep@gmail.com', '+91-4321098765', '456 Street, Pune', 'sandeep456', 'password456', '2024-04-07'),

(8, 'Kavita', 'Jain', 'kavita@gmail.com', '+91-3210987654', '789 Street, Ahmedabad', 'kavita789', 'password789', '2024-04-08'),

(9, 'Arun', 'Sharma', 'arun@gmail.com', '+91-2109876543', '123 Street, Jaipur', 'arun123', 'password123', '2024-04-09'),

(10, 'Meena', 'Singh', 'meena@gmail.com', '+91-1098765432', '456 Street, Lucknow', 'meena456', 'password456', '2024-04-10'),

(11, 'Anil', 'Yadav', 'anil@gmail.com', '+91-0987654321', '789 Street, Chandigarh', 'anil789', 'password789', '2024-04-11'),

(12, 'Neeta', 'Patel', 'neeta@gmail.com', '+91-9876543210', '123 Street, Surat', 'neeta123', 'password123', '2024-04-12'),

(13, 'Vijay', 'Kumar', 'vijay@gmail.com', '+91-8765432109', '456 Street, Indore', 'vijay456', 'password456', '2024-04-13'),

(14, 'Sunita', 'Sharma', 'sunita@gmail.com', '+91-7654321098', '789 Street, Bhopal', 'sunita789', 'password789', '2024-04-14'),

(15, 'Sanjay', 'Verma', 'sanjay@gmail.com', '+91-6543210987', '123 Street, Nagpur', 'sanjay123', 'password123', '2024-04-15');

**Inserting Data into Vehicle Table**

INSERT INTO Vehicle (VehicleID, Model, Make, Year, Color, RegistrationNumber, Availability, DailyRate)

VALUES

(1, 'SUV', 'Toyota', 2022, 'Black', 'MH01AB1234', TRUE, 2500.00),

(2, 'Sedan', 'Honda', 2023, 'Red', 'DL02CD5678', TRUE, 2000.00),

(3, 'Hatchback', 'Maruti', 2021, 'White', 'KA03EF9012', TRUE, 1500.00),

(4, 'SUV', 'Ford', 2022, 'Blue', 'MH04AB5678', TRUE, 2800.00),

(5, 'Sedan', 'Hyundai', 2023, 'Silver', 'DL05CD9012', TRUE, 2200.00),

(6, 'Hatchback', 'Volkswagen', 2021, 'Gray', 'KA06EF3456', TRUE, 1600.00),

(7, 'SUV', 'Chevrolet', 2022, 'Brown', 'MH07AB7890', TRUE, 2600.00),

(8, 'Sedan', 'Kia', 2023, 'Green', 'DL08CD1234', TRUE, 2100.00),

(9, 'Hatchback', 'Nissan', 2021, 'Orange', 'KA09EF5678', TRUE, 1700.00),

(10, 'SUV', 'Jeep', 2022, 'Yellow', 'MH10AB9012', TRUE, 2700.00),

(11, 'Sedan', 'BMW', 2023, 'Purple', 'DL11CD3456', TRUE, 2300.00),

(12, 'Hatchback', 'Audi', 2021, 'Cyan', 'KA12EF7890', TRUE, 1800.00),

(13, 'SUV', 'Mercedes', 2022, 'Magenta', 'MH13AB1234', TRUE, 2900.00),

(14, 'Sedan', 'Tesla', 2023, 'Gold', 'DL14CD5678', TRUE, 2400.00),

(15, 'Hatchback', 'Fiat', 2021, 'Brown', 'KA15EF9012', TRUE, 1900.00);

**Inserting Data into Reservation Table**

INSERT INTO Reservation (ReservationID, CustomerID, VehicleID, StartDate, EndDate, TotalCost, Status)

VALUES

(1, 2, 3, '2024-05-10 06:00:00', '2024-05-15 10:00:00', 12500.00, 'completed'),

(2, 2, 2, '2024-05-12 09:00:00', '2024-05-14 18:00:00', 8000.00, 'Confirmed'),

(3, 4, 3, '2024-05-08 12:00:00', '2024-05-10 23:00:00', 5500.00, 'pending'),

(4, 4, 6, '2024-05-11 10:00:00', '2024-05-17 16:00:00', 13500.00, 'Confirmed'),

(5, 6, 3, '2024-05-14 09:00:00', '2024-05-16 18:00:00', 6700.00, 'completed'),

(6, 6, 9, '2024-05-09 12:00:00', '2024-05-11 19:00:00', 4200.00, 'Confirmed'),

(7, 8, 7, '2024-05-15 05:00:00', '2024-05-20 10:00:00', 15200.00, 'pending'),

(8, 8, 8, '2024-05-18 09:00:00', '2024-05-20 18:00:00', 7300.00, 'pending'),

(9, 10, 9, '2024-05-20 05:00:00', '2024-05-22 12:00:00', 6000.00, 'completed'),

(10, 10, 9, '2024-05-17 10:00:00', '2024-05-23 14:00:00', 14500.00, 'Confirmed'),

(11, 12, 11, '2024-05-22 09:00:00', '2024-05-24 18:00:00', 8800.00, 'Confirmed'),

(12, 12, 6, '2024-05-19 12:00:00', '2024-05-21 17:00:00', 4700.00, 'pending'),

(13, 14, 5, '2024-05-24 10:00:00', '2024-05-29 10:00:00', 16000.00, 'Confirmed'),

(14, 11, 13, '2024-05-28 09:00:00', '2024-05-30 18:00:00', 7200.00, 'pending'),

(15, 15, 15, '2024-05-25 08:00:00', '2024-05-27 12:00:00', 5200.00, 'Confirmed');

**Inserting Data into Admin Table**

INSERT INTO Admin (AdminID, FirstName, LastName, Email, PhoneNumber, Username, Password, Role, JoinDate)

VALUES

(1, 'Aarav', 'Patel', 'aarav@gmail.com', '+91-9876543210', 'aarav', 'adminpassword', 'Super\_admin', '2024-04-01'),

(2, 'Aanya', 'Sharma', 'aanya@gmail.com', '+91-8765432109', 'aanya', 'adminpassword2', 'Super\_admin', '2024-04-02'),

(3, 'Advait', 'Singh', 'advait@gmail.com', '+91-7654321098', 'advait', 'adminpassword3', 'fleet manager', '2024-04-03'),

(4, 'Advika', 'Jain', 'advika@gmail.com', '+91-6543210987', 'advika', 'adminpassword4', 'fleet manager', '2024-04-04'),

(5, 'Ahaan', 'Kumar', 'ahaan@gmail.com', '+91-5432109876', 'ahaan', 'adminpassword5', 'fleet manager', '2024-04-05'),

(6, 'Aisha', 'Verma', 'aisha@gmail.com', '+91-4321098765', 'aisha', 'adminpassword6', 'Super\_admin', '2024-04-06'),

(7, 'Arjun', 'Yadav', 'arjun@gmail.com', '+91-3210987654', 'arjun', 'adminpassword7', 'Super\_admin', '2024-04-07'),

(8, 'Ananya', 'Gupta', 'ananya@gmail.com', '+91-2109876543', 'ananya', 'adminpassword8', 'fleet manager', '2024-04-08'),

(9, 'Aryan', 'Sharma', 'aryan@gmail.com', '+91-1098765432', 'aryan', 'adminpassword9', 'fleet manager', '2024-04-09'),

(10, 'Avni', 'Joshi', 'avni@gmail.com', '+91-0987654321', 'avni', 'adminpassword10', 'Super\_admin', '2024-04-10'),

(11, 'Ayaan', 'Singh', 'ayaan@gmail.com', '+91-9876543210', 'ayaan', 'adminpassword11', 'Super\_admin', '2024-04-11'),

(12, 'Ayushi', 'Pandey', 'ayushi@gmail.com', '+91-8765432109', 'ayushi', 'adminpassword12', 'Super\_admin', '2024-04-12'),

(13, 'Dhruv', 'Yadav', 'dhruv@gmail.com', '+91-7654321098', 'dhruv', 'adminpassword13', 'Super\_admin', '2024-04-13'),

(14, 'Diya', 'Sharma', 'diya@gmail.com', '+91-6543210987', 'diya', 'adminpassword14', 'Super\_admin', '2024-04-14'),

(15, 'Ishaan', 'Patel', 'ishaan@gmail.com', '+91-5432109876', 'ishaan', 'adminpassword15', 'fleet manager', '2024-04-15');

**Customer.cs**

using System;

class Customer

{

private int CustomerID;

private string FirstName;

private string LastName;

private string Email;

private string PhoneNumber;

private string Address;

private string Username;

private string Password;

private string RegistrationDate;

public Customer()

{

CustomerID=0;

FirstName="None";

LastName="None";

Email="None";

PhoneNumber="None";

Address="None";

Username="None";

Password="None";

RegistrationDate="None";

}

public Customer(int cid, string f, string l, string e, string p, string a, string u, string pass, string rdate)

{

CustomerID=cid;

FirstName=f;

LastName=l;

Email=e;

PhoneNumber=p;

Address=a;

Username=u;

Password=pass;

RegistrationDate=rdate;

}

public int customerID

{

get { return CustomerID; }

set { CustomerID=value ; }

}

public string firstName

{

get { return FirstName; }

set { FirstName=value ; }

}

public string lastName

{

get { return LastName; }

set { LastName=value ; }

}

public string email

{

get { return Email; }

set { Email=value ; }

}

public string phoneNumber

{

get { return PhoneNumber; }

set { PhoneNumber=value ; }

}

public string address

{

get { return Address; }

set { Address=value ; }

}

public string username

{

get { return Username; }

set { Username=value ; }

}

public string password

{

get { return Password; }

set { Password=value ; }

}

public string registrationDate

{

get { return RegistrationDate; }

set { RegistrationDate=value ; }

}

public void Authenticate(string pass)

{

if (password==pass)

{

Console.WriteLine("The user is Authenticated");

}

}

public void PrintInfo()

{

Console.WriteLine("CustomerID:"+ customerID);

Console.WriteLine("First Name:"+ firstName);

Console.WriteLine("Last Name:"+ lastName);

Console.WriteLine("Email:"+ email);

Console.WriteLine("Phone Number:"+ phoneNumber);

Console.WriteLine("Address:"+ address);

Console.WriteLine("Username:"+ username);

Console.WriteLine("Password:"+ password);

Console.WriteLine("Registration Date:"+ registrationDate);

}

}

**Vehicle.cs**

using System;

class Vehicle

{

private int VehicleID;

private string Model;

private string Make;

private int Year;

private string Color;

private string RegistrationNumber;

private bool Availability;

private double DailyRate;

public Vehicle()

{

VehicleID = 0;

Model = "None";

Make = "None";

Year = 0;

Color = "None";

RegistrationNumber = "None";

Availability = false;

DailyRate = 0.0;

}

public Vehicle(int vid, string mo, string ma, int y, string c, string r, bool a, double ra)

{

VehicleID = vid;

Model = mo;

Make = ma;

Year = y;

Color = c;

RegistrationNumber = r;

Availability = a;

DailyRate = ra;

}

public int vehicleID

{

get { return VehicleID; }

set { VehicleID = value; }

}

public string model

{

get { return Model; }

set { Model = value; }

}

public string make

{

get { return Make; }

set { Make = value; }

}

public int year

{

get { return Year; }

set { Year = value; }

}

public string color

{

get { return Color; }

set { Color = value; }

}

public string registrationNumber

{

get { return RegistrationNumber; }

set { RegistrationNumber = value; }

}

public bool availability

{

get { return Availability; }

set { Availability = value; }

}

public double dailyRate

{

get { return DailyRate; }

set { DailyRate = value; }

}

public void PrintInfo()

{

Console.WriteLine("VehicleID: " + vehicleID);

Console.WriteLine("Model: " + model);

Console.WriteLine("Make: " + make);

Console.WriteLine("Year: " + year);

Console.WriteLine("Color: " + color);

Console.WriteLine("Registration Number: " + registrationNumber);

Console.WriteLine("Availability: " + availability);

Console.WriteLine("Daily Rate: " + dailyRate);

}

}

**Reservation.cs**

using System;

class Reservation

{

private int ReservationID;

private int CustomerID;

private int VehicleID;

private string StartDate;

private string EndDate;

private double TotalCost;

private string Status;

public Reservation()

{

ReservationID = 0;

CustomerID = 0;

VehicleID = 0;

StartDate = "None";

EndDate = "None";

TotalCost = 0.0;

Status = "None";

}

public Reservation(int rid, int cid, int vid, string s, string end, double c, string stat)

{

ReservationID = rid;

CustomerID = cid;

VehicleID = vid;

StartDate = s;

EndDate = end;

TotalCost = c;

Status = stat;

}

public int reservationID

{

get { return ReservationID; }

set { ReservationID = value; }

}

public int customerID

{

get { return CustomerID; }

set { CustomerID = value; }

}

public int vehicleID

{

get { return VehicleID; }

set { VehicleID = value; }

}

public string startDate

{

get { return StartDate; }

set { StartDate = value; }

}

public string endDate

{

get { return EndDate; }

set { EndDate = value; }

}

public double totalCost

{

get { return TotalCost; }

set { TotalCost = value; }

}

public string status

{

get { return Status; }

set { Status = value; }

}

public void CalculateTotalCost()

{

Console.WriteLine("The total cost for reservation is:"+ totalCost);

}

public void PrintInfo()

{

Console.WriteLine("Reservation ID: " + reservationID);

Console.WriteLine("Customer ID: " + customerID);

Console.WriteLine("Vehicle ID: " + vehicleID);

Console.WriteLine("Start Date: " + startDate);

Console.WriteLine("End Date: " + endDate);

Console.WriteLine("Total Cost: " + totalCost);

Console.WriteLine("Status: " + status);

}

}

**Admin.cs**

using System;

class Admin

{

private int AdminID;

private string FirstName;

private string LastName;

private string Email;

private string PhoneNumber;

private string Username;

private string Password;

private string Role;

private string JoinDate;

public Admin()

{

AdminID = 0;

FirstName = "None";

LastName = "None";

Email = "None";

PhoneNumber = "None";

Username = "None";

Password = "None";

Role = "None";

JoinDate = "None";

}

public Admin(int aid, string f, string l, string e, string p, string u, string pass, string r, string jdate)

{

AdminID = aid;

FirstName = f;

LastName = l;

Email = e;

PhoneNumber = p;

Username = u;

Password = pass;

Role = r;

JoinDate = jdate;

}

public int adminID

{

get { return AdminID; }

set { AdminID = value; }

}

public string firstName

{

get { return FirstName; }

set { FirstName = value; }

}

public string lastName

{

get { return LastName; }

set { LastName = value; }

}

public string email

{

get { return Email; }

set { Email = value; }

}

public string phoneNumber

{

get { return PhoneNumber; }

set { PhoneNumber = value; }

}

public string username

{

get { return Username; }

set { Username = value; }

}

public string password

{

get { return Password; }

set { Password = value; }

}

public string role

{

get { return Role; }

set { Role = value; }

}

public string joinDate

{

get { return JoinDate; }

set { JoinDate = value; }

}

public void Authenticate(string pass)

{

if (Password == pass)

{

Console.WriteLine("Admin is authenticated.");

}

}

public void PrintInfo()

{

Console.WriteLine("Admin ID: " + adminID);

Console.WriteLine("First Name: " + firstName);

Console.WriteLine("Last Name: " + lastName);

Console.WriteLine("Email: " + email);

Console.WriteLine("Phone Number: " + phoneNumber);

Console.WriteLine("Username: " + username);

Console.WriteLine("Role: " + role);

Console.WriteLine("Join Date: " + joinDate);

}

}

**Interface.cs**

using System;

public interface ICustomerService

{

void GetCustomerById(int customerId);

void GetCustomerByUsername(string username);

void RegisterCustomer(Customer customerData);

void UpdateCustomer(Customer customerData);

void DeleteCustomer(int customerId);

}

public interface IVehicleService

{

void GetVehicleById(int vehicleId);

void GetAvailableVehicles();

void AddVehicle(Vehicle vehicleData);

void UpdateVehicle(Vehicle vehicleData);

void RemoveVehicle(int vehicleId);

}

public interface IReservationService

{

void GetReservationById(int reservationId);

void GetReservationsByCustomerId(int customerId);

void CreateReservation(Reservation reservationData);

void UpdateReservation(Reservation reservationData);

void CancelReservation(int reservationId);

}

public interface IAdminService

{

void GetAdminById(int adminId);

void GetAdminByUsername(string username);

void RegisterAdmin(Admin adminData);

void UpdateAdmin(Admin adminData);

void DeleteAdmin(int adminId);

}

**CustomerService.cs**

using System;

using System.Collections.Generic;

class CustomerService : ICustomerService

{

private List<Customer> customers;

public CustomerService(List<Customer> customerList)

{

customers = customerList;

}

public void GetCustomerById(int custId)

{

bool found = false;

foreach (Customer cust in customers)

{

if (cust.customerID == custId)

{

cust.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Customer does not exist");

}

}

public void GetCustomerByUsername(string uname)

{

bool found = false;

foreach (Customer cust in customers)

{

if (cust.username == uname)

{

cust.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("username does not exist");

}

}

public void RegisterCustomer(Customer cust)

{

foreach (Customer ecust in customers)

{

if (ecust.customerID == cust.customerID || ecust.username == cust.username)

{

Console.WriteLine("Customer with this ID or Username already exists.");

return;

}

}

customers.Add(cust);

Console.WriteLine("Customer registered.");

}

public void UpdateCustomer(Customer cust)

{

bool found = false;

for (int i = 0; i < customers.Count; i++)

{

if (customers[i].customerID == cust.customerID)

{

customers[i] = cust;

found = true;

Console.WriteLine("Customer Info updated.");

break;

}

}

if (!found)

{

Console.WriteLine("Customer does not exist");

}

}

public void DeleteCustomer(int custId)

{

bool found = false;

for (int i = 0; i < customers.Count; i++)

{

if (customers[i].customerID == custId)

{

customers.RemoveAt(i);

found = true;

Console.WriteLine("Customer deleted");

break;

}

}

if (!found)

{

Console.WriteLine("Customer does not exist");

}

}

}

**VehicleService.cs**

using System;

using System.Collections.Generic;

class VehicleService : IVehicleService

{

private List<Vehicle> vehicles;

public VehicleService(List<Vehicle> vehicleList)

{

vehicles = vehicleList;

}

public void GetVehicleById(int vehicleId)

{

bool found = false;

foreach (Vehicle vehicle in vehicles)

{

if (vehicle.vehicleID == vehicleId)

{

vehicle.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Vehicle does not exist.");

}

}

public void GetAvailableVehicles()

{

bool found = false;

foreach (Vehicle vehicle in vehicles)

{

if (vehicle.availability)

{

vehicle.PrintInfo();

found = true;

}

}

if (!found)

{

Console.WriteLine("No available vehicles.");

}

}

public void AddVehicle(Vehicle vehicleData)

{

foreach (Vehicle eVehicle in vehicles)

{

if (eVehicle.vehicleID == vehicleData.vehicleID || e.registrationNumber == vehicleData.registrationNumber)

{

Console.WriteLine("Vehicle already exists.");

return;

}

}

vehicles.Add(vehicleData);

Console.WriteLine("Vehicle added.");

}

public void UpdateVehicle(Vehicle vehicleData)

{

bool found = false;

for (int i = 0; i < vehicles.Count; i++)

{

if (vehicles[i].vehicleID == vehicleData.vehicleID)

{

vehicles[i] = vehicleData;

found = true;

Console.WriteLine("Vehicle Info updated.");

break;

}

}

if (!found)

{

Console.WriteLine("Vehicle does not exist.");

}

}

public void RemoveVehicle(int vehicleId)

{

bool found = false;

for (int i = 0; i < vehicles.Count; i++)

{

if (vehicles[i].vehicleID == vehicleId)

{

vehicles.RemoveAt(i);

found = true;

Console.WriteLine("Vehicle removed successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Vehicle does not exist.");

}

}

}

**ReservationService.cs**

using System;

using System.Collections.Generic;

class ReservationService : IReservationService

{

private List<Reservation> reservations;

public ReservationService(List<Reservation> reservationList)

{

reservations = reservationList;

}

public void GetReservationById(int reservationId)

{

bool found = false;

foreach (Reservation r in reservations)

{

if (r.reservationID == reservationId)

{

r.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Reservation does not exist.");

}

}

public void GetReservationsByCustomerId(int customerId)

{

bool found = false;

foreach (Reservation r in reservations)

{

if (r.customerID == customerId)

{

r.PrintInfo();

found = true;

}

}

if (!found)

{

Console.WriteLine("No reservations found for the customer.");

}

}

public void CreateReservation(Reservation reservationData)

{

foreach (Reservation er in reservations)

{

if (er.reservationID == reservationData.reservationID)

{

Console.WriteLine("Reservation with this ID already exists.");

return;

}

}

reservations.Add(reservationData);

Console.WriteLine("Reservation created successfully.");

}

public void UpdateReservation(Reservation reservationData)

{

bool found = false;

for (int i = 0; i < reservations.Count; i++)

{

if (reservations[i].reservationID == reservationData.reservationID)

{

reservations[i] = reservationData;

found = true;

Console.WriteLine("Reservation updated successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Reservation does not exist.");

}

}

public void CancelReservation(int reservationId)

{

bool found = false;

for (int i = 0; i < reservations.Count; i++)

{

if (reservations[i].reservationID == reservationId)

{

reservations.RemoveAt(i);

found = true;

Console.WriteLine("Reservation cancelled successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Reservation does not exist.");

}

}

}

**AdminService.cs**

using System;

using System.Collections.Generic;

class AdminService : IAdminService

{

private List<Admin> admins;

public AdminService(List<Admin> adminList)

{

admins = adminList;

}

public void GetAdminById(int adminId)

{

bool found = false;

foreach (Admin a in admins)

{

if (a.adminID == adminId)

{

a.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Admin does not exist.");

}

}

public void GetAdminByUsername(string username)

{

bool found = false;

foreach (Admin a in admins)

{

if (a.username == username)

{

a.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Admin does not exist.");

}

}

public void RegisterAdmin(Admin adminData)

{

foreach (Admin ea in admins)

{

if (ea.username == adminData.username)

{

Console.WriteLine("Username exists.");

return;

}

}

admins.Add(adminData);

Console.WriteLine("Admin registered.");

}

public void UpdateAdmin(Admin adminData)

{

bool found = false;

for (int i = 0; i < admins.Count; i++)

{

if (admins[i].adminID == adminData.adminID)

{

admins[i] = adminData;

found = true;

Console.WriteLine("Admin information updated successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Admin with this ID does not exist.");

}

}

public void DeleteAdmin(int adminId)

{

bool found = false;

for (int i = 0; i < admins.Count; i++)

{

if (admins[i].adminID == adminId)

{

admins.RemoveAt(i);

found = true;

Console.WriteLine("Admin deleted successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Admin with this ID does not exist.");

}

}

}

**SQLCommand.cs**

using System;

using System.Data;

using System.Data.SqlClient;

using System.Text;

namespace CarRentalSystem

{

class Program

{

static SqlConnection con = null;

static SqlCommand cmd;

static SqlDataReader dr;

static void Main(string[] args)

{

Console.WriteLine("Welcome to Car Rental System!");

Console.WriteLine("1. Insert Data to Customer Table");

Console.WriteLine("2. Insert Data to Vehicle Table");

Console.WriteLine("3. Insert Data to Reservation Table");

Console.WriteLine("4. Insert Data to Admin Table");

Console.WriteLine("Choose one of the above options for Insertion:");

op=Convert.ToInt32(Console.ReadLine());

switch(op)

{

case 1:

InsertCustomerData();

break;

case 2:

InsertVehicleData();

break;

case 3:

InsertReservationData();

break;

case 4:

InsertAdminData();

break;

}

}

public static SqlConnection GetConnection()

{

con = new SqlConnection("data source =SQLEXPRESS; initial catalog = CarConnectDB; Integrated security = true;");

con.Open();

return con;

}

public static void InsertCustomerData()

{

con = GetConnection();

Console.WriteLine("Enter Details for Customer Table:");

Console.WriteLine("Enter First Name:");

string f = Console.ReadLine();

Console.WriteLine("Enter Last Name:");

string l = Console.ReadLine();

Console.WriteLine("Enter Email:");

string e = Console.ReadLine();

Console.WriteLine("Enter Phone Number:");

string p = Console.ReadLine();

Console.WriteLine("Enter Address:");

string a = Console.ReadLine();

Console.WriteLine("Enter Username:");

string u = Console.ReadLine();

Console.WriteLine("Enter Password (use a secure method in production):");

string pass = Console.ReadLine();

DateTime rd = DateTime.Now;

cmd = new SqlCommand("INSERT INTO Customer VALUES (@firstName, @lastName, @email, @phoneNumber, @address, @username, @password, @registrationDate)", con);

cmd.Parameters.AddWithValue("@FirstName", f;

cmd.Parameters.AddWithValue("@LastName", l);

cmd.Parameters.AddWithValue("@Email", e);

cmd.Parameters.AddWithValue("@PhoneNumber", p);

cmd.Parameters.AddWithValue("@Address", a);

cmd.Parameters.AddWithValue("@Username", u);

cmd.Parameters.AddWithValue("@Password", pass);

cmd.Parameters.AddWithValue("@RegistrationDate", rd);

int rows = cmd.ExecuteNonQuery();

if (rows>0)

{

Console.WriteLine("Record added successfully.");

}

else

{

Console.WriteLine("Unable to add the record");

}

}

public static void InsertVehicleData()

{

con = GetConnection();

Console.WriteLine("Enter Details for Vehicle Table");

Console.WriteLine("Enter Model:");

string mo = Console.ReadLine();

Console.WriteLine("Enter Make:");

string ma = Console.ReadLine();

Console.WriteLine("Enter Year:");

int y = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Color:");

string c = Console.ReadLine();

Console.WriteLine("Enter Registration Number:");

string rn = Console.ReadLine();

Console.WriteLine("Is the vehicle available for rent (true/false):");

bool a = Convert.ToBoolean(Console.ReadLine());

Console.WriteLine("Enter Daily Rate:");

decimal dr = Convert.ToDecimal(Console.ReadLine());

cmd = new SqlCommand("INSERT INTO Vehicle VALUES (@model, @make, @year, @color, @registrationNumber, @availability, @dailyRate)", con);

cmd.Parameters.AddWithValue("@Model", mo);

cmd.Parameters.AddWithValue("@Make", ma);

cmd.Parameters.AddWithValue("@Year", y);

cmd.Parameters.AddWithValue("@Color", c);

cmd.Parameters.AddWithValue("@RegistrationNumber", rn);

cmd.Parameters.AddWithValue("@Availability", a);

cmd.Parameters.AddWithValue("@DailyRate", dr);

int rows = cmd.ExecuteNonQuery();

if (rows>0)

{

Console.WriteLine("Record added successfully.");

}

else

{

Console.WriteLine("Unable to add the record");

}

}

public static void InsertReservationData()

{

con = GetConnection();

Console.WriteLine("Enter Details for the Reservation Table");

Console.WriteLine("Enter Customer ID:");

int cid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Vehicle ID:");

int vid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Reservation Start Date (yyyy-mm-dd):");

DateTime sd = Convert.ToDateTime(Console.ReadLine());

Console.WriteLine("Enter Reservation End Date (yyyy-mm-dd):");

DateTime ed = Convert.ToDateTime(Console.ReadLine());

Console.WriteLine("Enter Total Cost:");

decimal tc = Convert.ToDecimal(Console.ReadLine());

Console.WriteLine("Enter Reservation Status (e.g., Pending, Confirmed):");

string stat = Console.ReadLine();

cmd = new SqlCommand("INSERT INTO Reservation VALUES (@customerID, @vehicleID, @startDate, @endDate, @totalCost, @status)", con);

cmd.Parameters.AddWithValue("@CustomerID", cid);

cmd.Parameters.AddWithValue("@VehicleID", vid);

cmd.Parameters.AddWithValue("@StartDate", sd);

cmd.Parameters.AddWithValue("@EndDate", ed);

cmd.Parameters.AddWithValue("@TotalCost", tc);

cmd.Parameters.AddWithValue("@Status", stat);

int rows = cmd.ExecuteNonQuery();

if (rows>0)

{

Console.WriteLine("Record added successfully.");

}

else

{

Console.WriteLine("Unable to add the record");

}

}

public static void InsertAdminData()

{

con = GetConnection();

Console.WriteLine("Enter Details for the Admin Table");

Console.WriteLine("Enter First Name:");

string fn = Console.ReadLine();

Console.WriteLine("Enter Last Name:");

string ln = Console.ReadLine();

Console.WriteLine("Enter Email:");

string em = Console.ReadLine();

Console.WriteLine("Enter Phone Number:");

string pn = Console.ReadLine();

Console.WriteLine("Enter Username:");

string un = Console.ReadLine();

Console.WriteLine("Enter Password (use a secure method in production):");

string pas = Console.ReadLine();

Console.WriteLine("Enter Role (e.g., Super Admin, Fleet Manager):");

string rol = Console.ReadLine();

DateTime jd = DateTime.Now;

cmd = new SqlCommand("INSERT INTO Admin VALUES (@firstName, @lastName, @email, @phoneNumber, @username, @password, @role, @joinDate)", con);

cmd.Parameters.AddWithValue("@FirstName", fn);

cmd.Parameters.AddWithValue("@LastName", ln);

cmd.Parameters.AddWithValue("@Email", em);

cmd.Parameters.AddWithValue("@PhoneNumber", pn);

cmd.Parameters.AddWithValue("@Username", un);

cmd.Parameters.AddWithValue("@Password", pas);

cmd.Parameters.AddWithValue("@Role", rol);

cmd.Parameters.AddWithValue("@JoinDate", jd);

int rows = cmd.ExecuteNonQuery();

if (rows>0)

{

Console.WriteLine("Record added successfully.");

}

else

{

Console.WriteLine("Unable to add the record");

}

}

}

}

**CustomExceptions.cs**

using System;

namespace CustomExceptions

{

public class AuthenticationException : Exception

{

public AuthenticationException(string msg) : base(msg) { }

}

public class ReservationException : Exception

{

public ReservationException(string msg) : base(msg) { }

}

public class VehicleNotFoundException : Exception

{

public VehicleNotFoundException(string msg) : base(msg) { }

}

public class AdminNotFoundException : Exception

{

public AdminNotFoundException(string msg) : base(msg) { }

}

public class InvalidInputException : Exception

{

public InvalidInputException(string msg) : base(msg) { }

}

public class DatabaseConnectionException : Exception

{

public DatabaseConnectionException(string msg) : base(msg) { }

}

}

**AuthenticationService.cs**

using System;

using System.Collections.Generic;

public class AuthenticationService

{

private List<Customer> customers;

private List<Admin> admins;

public AuthenticationService(List<Customer> customers, List<Admin> admins)

{

this.customers = customers;

this.admins = admins;

}

public void AuthenticateCustomer(string uname, string pass)

{

foreach (Customer customer in customers)

{

if (customer.username == uname && customer.password == pass)

{

Console.WriteLine("Customer is Autheticated");

}

}

throw new AuthenticationException("Authentication Failure");

}

public void AuthenticateAdmin(string uname, string pass)

{

foreach (Admin admin in admins)

{

if (admin.username == uname && admin.password == pass)

{

Console.WriteLine("Admin is authenticated");

}

}

throw new AuthenticationException("Admin Authentication Failure");

}

}

**ReportGenerator.cs**

using System;

using System.Collections.Generic;

public class ReportGenerator

{

private List<Vehicle> vehicles;

private List<Reservation> reservations;

public ReportGenerator(List<Vehicle> vehicles, List<Reservation> reservations)

{

this.vehicles = vehicles;

this.reservations = reservations;

}

public void VehicleInfo(int id)

{

foreach (Vehicle v in vehicles)

{

if (v.vehicleID == id)

{

v.PrintInfo();

return;

}

}

throw new VehicleNotFoundException("Vehicle Not Found");

}

public void ReservationInfo(int customerId)

{

bool found = false;

foreach (Reservation r in reservations)

{

if (r.customerID == customerId)

{

found = true;

r.PrintInfo();

}

}

if (!found)

{

throw new ReservationException("No Reservations Found");

}

}

}

**TestCases.cs**

using NUnit.Framework;

using NUnit.Framework.Legacy;

using System;

using System.Collections.Generic;

using CarRentalSystem;

namespace CarRentalSystem.Tests

{

[TestFixture]

public class CarRentalSystemUnitTests

{

private AuthenticationService authService;

private ICustomerService customerService;

private IVehicleService vehicleService;

private List<Customer> customers;

private List<Vehicle> vehicles;

[SetUp]

public void Setup()

{

customers = new List<Customer>

{

new Customer

{

customerID = 1,

firstName = "Karan",

lastName = "Das",

username = "karandass23",

password = "xyzabc",

email = "karandas23@gmail.com",

phoneNumber = "9876543210",

address = "Chennai",

registrationDate = DateTime.Now

}

};

vehicles = new List<Vehicle>

{

new Vehicle

{

vehicleID = 1,

make = "Land Cruiser",

model = "Prado",

year = 2020,

color = "White",

registrationNumber = "TN25RN7389",

availability = true,

dailyRate = 75.00M

}

};

authService = new AuthenticationService(customers, new List<Admin>());

customerService = new CustomerService(customers);

vehicleService = new VehicleService(vehicles);

}

[Test]

public void TestCustomerAuthentication()

{

try

{

authService.AuthenticateCustomer("karandas23", "xyzabc");

ClassicAssert.Fail("Test Fail");

}

catch (AuthenticationException ex)

{

ClassicAssert.AreEqual("Authentication failed", ex.Message);

}

}

[Test]

public void TestUpdateCustomerInformation()

{

Customer customer = customerService.GetCustomerById(1);

customer.email = "kiaramilan@gmail.com";

customerService.UpdateCustomer(customer);

Customer uc = customerService.GetCustomerById(1);

ClassicAssert.AreEqual("kiaramilan@gmail.com", uc.email);

}

[Test]

public void TestAddNewVehicle()

{

Vehicle newVehicle = new Vehicle

{

VehicleID = 2,

Make = "Hyundai",

Model = "Elantra",

Year = 2025,

Color = "White",

RegistrationNumber = "TN26YZ6789",

Availability = true,

DailyRate = 80.00M

};

vehicleService.AddVehicle(newVehicle);

Vehicle v=vehicleService.GetVehicleById(2);

ClassicAssert.AreEqual("Hyundai", addedVehicle.Make);

}

[Test]

public void TestUpdateVehicleDetails()

{

Vehicle vehicle = vehicleService.GetVehicleById(1);

vehicle.DailyRate = 90.00M;

vehicleService.UpdateVehicle(vehicle);

Vehicle uv = vehicleService.GetVehicleById(1);

ClassicAssert.AreEqual(90.00M, uv.DailyRate);

}

[Test]

public void TestGetAvailableVehicles()

{

vehicleService.GetAvailableVehicles();

Vehicle allVehicles = vehicleService.GetInternalVehicleList();

foreach (Vehicle vehicle in allVehicles)

{

ClassicAssert.IsTrue(vehicle.Availability);

}

}

[Test]

public void TestGetAllVehicles()

{

Vehicle allVehicles = vehicleService.GetAllVehicles();

ClassicAssert.AreEqual(vehicles.Count, allVehicles.Count);

}

}

}

**Main Program:**

using System;

using System.Collections.Generic;

class MainProgram

{

static void Main(string[] args)

{

List<Customer> customers = new List<Customer>();

List<Vehicle> vehicles = new List<Vehicle>();

List<Reservation> reservations = new List<Reservation>();

List<Admin> admins = new List<Admin>();

ICustomerService customerService = new CustomerService(customers);

IVehicleService vehicleService = new VehicleService(vehicles);

IReservationService reservationService = new ReservationService(reservations);

IAdminService adminService = new AdminService(admins);

bool run = true;

while (run)

{

Console.WriteLine("CAR CONNECT MENU:");

Console.WriteLine("1. Register Customer");

Console.WriteLine("2. View Customer by ID");

Console.WriteLine("3. Add Vehicle");

Console.WriteLine("4. View Vehicles");

Console.WriteLine("5. Create Reservation");

Console.WriteLine("6. View Reservation");

Console.WriteLine("7. Register Admin");

Console.WriteLine("8. View Admin");

Console.WriteLine("9. Exit");

int choice = Convert.ToInt32(Console.ReadLine());

Console.WriteLine();

switch (choice)

{

case 1:

Console.WriteLine("Customer ID: ");

int cid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("First Name: ");

string fname = Console.ReadLine();

Console.WriteLine("Last Name: ");

string lname = Console.ReadLine();

Console.WriteLine("Email: ");

string email = Console.ReadLine();

Console.WriteLine("Phone Number: ");

string phone = Console.ReadLine();

Console.WriteLine("Address: ");

string address = Console.ReadLine();

Console.WriteLine("Username: ");

string uname = Console.ReadLine();

Console.WriteLine("Password: ");

string pwd = Console.ReadLine();

Console.WriteLine("Registration Date (yyyy-MM-dd): ");

string rdate = Console.ReadLine();

customerService.RegisterCustomer(new Customer(cid, fname, lname, email, phone, address, uname, pwd, rdate));

break;

case 2:

Console.WriteLine("Enter Customer ID: ");

int cid2=Convert.ToInt32(Console.ReadLine())

customerService.GetCustomerById(cid2);

break;

case 3:

Console.WriteLine("Vehicle ID: ");

int vid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Model: ");

string model = Console.ReadLine();

Console.WriteLine("Make: ");

string make = Console.ReadLine();

Console.WriteLine("Year: ");

int year = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Color: ");

string color = Console.ReadLine();

Console.WriteLine("Registration Number: ");

string regNo = Console.ReadLine();

Console.WriteLine("Is Available?");

bool avail = Convert.ToBoolean(Console.ReadLine());

Console.WriteLine("Daily Rate: ");

double rate = Convert.ToDouble(Console.ReadLine());

vehicleService.AddVehicle(new Vehicle(vid, model, make, year, color, regNo, avail, rate));

break;

case 4:

vehicleService.GetAvailableVehicles();

break;

case 5:

Console.WriteLine("Reservation ID: ");

int rid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Customer ID: ");

int rcid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Vehicle ID: ");

int rvid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Start Date: ");

string sdate = Console.ReadLine();

Console.WriteLine("End Date: ");

string edate = Console.ReadLine();

Console.WriteLine("Total Cost: ");

double cost = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Status: ");

string status = Console.ReadLine();

reservationService.CreateReservation(new Reservation(rid, rcid, rvid, sdate, edate, cost, status));

break;

case 6:

Console.WriteLine("Enter Reservation ID: ");

int rid6=Convert.ToInt32(Console.ReadLine());

reservationService.GetReservationById();

break;

case 7:

Console.WriteLine("Admin ID: ");

int aid = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("First Name: ");

string afirst = Console.ReadLine();

Console.WriteLine("Last Name: ");

string alast = Console.ReadLine();

Console.WriteLine("Email: ");

string aemail = Console.ReadLine();

Console.WriteLine("Phone Number: ");

string aphone = Console.ReadLine();

Console.WriteLine("Username: ");

string auser = Console.ReadLine();

Console.WriteLine("Password: ");

string apass = Console.ReadLine();

Console.WriteLine("Role: ");

string arole = Console.ReadLine();

Console.WriteLine("Join Date: ");

string adate = Console.ReadLine();

adminService.RegisterAdmin(new Admin(aid, afirst, alast, aemail, aphone, auser, apass, arole, adate));

break;

case 8:

Console.WriteLine("Enter Admin Username: ");

adminService.GetAdminByUsername(Console.ReadLine());

break;

case 9:

run = false;

Console.WriteLine("exit");

break;

default:

Console.WriteLine("Invalid choice.");

break;

}

}

}

}