Case Study: Transport Management System

Introduction:

The Transport Management System is a console-based application developed using C# and SQL Server, designed to streamline the operations involved in managing vehicles, trips, drivers, and bookings. It adheres to object-oriented principles and follows a architecture with clearly separated layers such as entity, dao, util, exception, and main. The system allows administrators to add, update, or delete vehicles, schedule and cancel trips, manage driver allocation, and handle passenger bookings efficiently.

DB Creation:

```
CREATE DATABASE TransportDB

USE TransportDB

CREATE TABLE Vehicles (
   VehicleID INT IDENTITY(1,1) PRIMARY KEY,
   Model VARCHAR(255),
   Capacity DECIMAL(10, 2),
   Type VARCHAR(50),
   Status VARCHAR(50)
);

CREATE TABLE Routes (
   RouteID INT IDENTITY(1,1) PRIMARY KEY,
   StartDestination VARCHAR(255),
   EndDestination VARCHAR(255),
   Distance DECIMAL(10, 2)
);

CREATE TABLE Trips (
   TripID INT IDENTITY(1,1) PRIMARY KEY,
   VehicleID INT FOREIGN KEY REFERENCES Vehicles(VehicleID),
   RouteID INT FOREIGN KEY REFERENCES Routes(RouteID),
   DepartureDate DATETIME,
   ArrivalDate DATETIME,
   Status VARCHAR(50),
   TripType VARCHAR(50) DEFAULT 'Freight',
   MaxPassengers INT
);

CREATE TABLE Passengers (
   PassengerID INT IDENTITY(1,1) PRIMARY KEY,
   FirstName VARCHAR(255),
   Gender VARCHAR(255),
```

```sql
    Age INT,
    Email VARCHAR(255) UNIQUE,
    PhoneNumber VARCHAR(50)
);

CREATE TABLE Bookings (
    BookingID INT IDENTITY(1,1) PRIMARY KEY,
    TripID INT FOREIGN KEY REFERENCES Trips(TripID),
    PassengerID INT FOREIGN KEY REFERENCES Passengers(PassengerID),
    BookingDate DATETIME,
    Status VARCHAR(50)
);

CREATE TABLE Drivers (
    DriverID INT PRIMARY KEY IDENTITY(1,1),
    Name VARCHAR(255),
    PhoneNumber VARCHAR(50),
    Status VARCHAR(50) -- e.g., Available, Assigned, On Leave
);


-- Insert into Vehicles
INSERT INTO Vehicles (Model, Capacity, Type, Status) VALUES
('Volvo AC', 45.00, 'Bus', 'Available'),
('Tata Ace', 2.50, 'Truck', 'Maintenance'),
('Ashok Leyland', 35.00, 'Bus', 'On Trip'),
('Maruti Omni', 7.00, 'Van', 'Available'),
('Eicher Cargo', 10.00, 'Truck', 'Available');

-- Insert into Routes
INSERT INTO Routes (StartDestination, EndDestination, Distance) VALUES
('Chennai', 'Coimbatore', 505.00),
('Bangalore', 'Mysore', 145.50),
('Hyderabad', 'Vizag', 620.00),
('Delhi', 'Agra', 233.20),
('Mumbai', 'Pune', 150.75);

-- Insert into Trips
INSERT INTO Trips (VehicleID, RouteID, DepartureDate, ArrivalDate, Status, TripType, MaxPassengers)
VALUES
(1, 1, '2025-07-01 08:00', '2025-07-01 15:00', 'Scheduled', 'Passenger', 45),
(2, 2, '2025-07-03 06:00', '2025-07-03 10:00', 'In Progress', 'Freight', 0),
(3, 3, '2025-07-05 09:00', '2025-07-05 17:00', 'Scheduled', 'Passenger', 35),
(4, 4, '2025-07-07 07:30', '2025-07-07 10:30', 'Cancelled', 'Passenger', 7),
(5, 5, '2025-07-09 05:00', '2025-07-09 07:30', 'Completed', 'Freight', 0);

-- Insert into Passengers
INSERT INTO Passengers (FirstName, Gender, Age, Email, PhoneNumber) VALUES
('Ravi Kumar', 'Male', 30, 'ravi.kumar@example.com', '9876543210'),
('Anita Sharma', 'Female', 25, 'anita.sharma@example.com', '9876543211'),
('David Raj', 'Male', 28, 'david.raj@example.com', '9876543212'),
('Priya Verma', 'Female', 32, 'priya.verma@example.com', '9876543213'),
('Naveen Babu', 'Male', 29, 'naveen.babu@example.com', '9876543214');

-- Insert into Bookings
INSERT INTO Bookings (TripID, PassengerID, BookingDate, Status) VALUES
(1, 1, '2025-06-25 10:00', 'Confirmed'),
(1, 2, '2025-06-25 11:00', 'Confirmed'),
(3, 3, '2025-06-26 09:00', 'Cancelled'),
(3, 4, '2025-06-26 10:15', 'Confirmed'),
(4, 5, '2025-06-27 12:00', 'Cancelled');


INSERT INTO Drivers (Name, PhoneNumber, Status) VALUES
('Arun Kumar', '9876543210', 'Available'),
('Meena Raj', '9845123456', 'Available'),
('Ravi Chandran', '9998887776', 'Assigned'),
('Divya S', '9123456789', 'Available'),
('Suresh Babu', '9000011122', 'On Leave');

CREATE TABLE Trips (
   TripID INT IDENTITY(1,1) PRIMARY KEY,
   VehicleID INT FOREIGN KEY REFERENCES Vehicles(VehicleID),
   RouteID INT FOREIGN KEY REFERENCES Routes(RouteID),
   DepartureDate DATETIME,
   ArrivalDate DATETIME,
   Status VARCHAR(50),
   TripType VARCHAR(50) DEFAULT 'Freight',
   MaxPassengers INT
);
CREATE TABLE DriverTrip (
   TripID INT NOT NULL,
   DriverID INT NOT NULL,
   PRIMARY KEY (TripID, DriverID),
   FOREIGN KEY (TripID) REFERENCES Trips(TripID),
   FOREIGN KEY (DriverID) REFERENCES Drivers(DriverID)
);
```

Entity

Vehicle.cs

```
namespace TransportManagementSystem.entity
{
   public class Vehicle
   {
      private int vehicleID;
```

```
        private string model;
        private decimal capacity;
        private string type;
        private string status;

        public Vehicle() { }

        public Vehicle(int vehicleID, string model, decimal capacity, string type, string status)
        {
            this.vehicleID = vehicleID;
            this.model = model;
            this.capacity = capacity;
            this.type = type;
            this.status = status;
        }

        public int VehicleID { get => vehicleID; set => vehicleID = value; }
        public string Model { get => model; set => model = value; }
        public decimal Capacity { get => capacity; set => capacity = value; }
        public string Type { get => type; set => type = value; }
        public string Status { get => status; set => status = value; }
    }
}
```

Route.cs

```
namespace TransportManagementSystem.entity
{
    public class Route
    {
        private int routeID;
        private string startDestination;
        private string endDestination;
        private decimal distance;

        public Route() { }

        public Route(int routeID, string startDestination, string endDestination, decimal distance)
        {
            this.routeID = routeID;
            this.startDestination = startDestination;
            this.endDestination = endDestination;
            this.distance = distance;
        }

        public int RouteID { get => routeID; set => routeID = value; }
        public string StartDestination { get => startDestination; set => startDestination = value; }
        public string EndDestination { get => endDestination; set => endDestination = value; }
        public decimal Distance { get => distance; set => distance = value; }
    }
}
```

Trip.cs

```
using System;

namespace TransportManagementSystem.entity
{
    public class Trip
    {
        private int tripID;
        private int vehicleID;
        private int routeID;
        private DateTime departureDate;
        private DateTime arrivalDate;
        private string status;
        private string tripType;
        private int maxPassengers;

        public Trip() { }

        public Trip(int tripID, int vehicleID, int routeID, DateTime departureDate, DateTime arrivalDate,
string status, string tripType, int maxPassengers)
        {
            this.tripID = tripID;
            this.vehicleID = vehicleID;
            this.routeID = routeID;
            this.departureDate = departureDate;
            this.arrivalDate = arrivalDate;
            this.status = status;
            this.tripType = tripType;
            this.maxPassengers = maxPassengers;
        }

        public int TripID { get => tripID; set => tripID = value; }
        public int VehicleID { get => vehicleID; set => vehicleID = value; }
        public int RouteID { get => routeID; set => routeID = value; }
        public DateTime DepartureDate { get => departureDate; set => departureDate = value; }
        public DateTime ArrivalDate { get => arrivalDate; set => arrivalDate = value; }
        public string Status { get => status; set => status = value; }
        public string TripType { get => tripType; set => tripType = value; }
        public int MaxPassengers { get => maxPassengers; set => maxPassengers = value; }
    }
}
```

Passenger.cs

```
namespace TransportManagementSystem.entity
```

```
{
    public class Passenger
    {
        private int passengerID;
        private string firstName;
        private string gender;
        private int age;
        private string email;
        private string phoneNumber;

        public Passenger() { }

        public Passenger(int passengerID, string firstName, string gender, int age, string email, string
phoneNumber)
        {
            this.passengerID = passengerID;
            this.firstName = firstName;
            this.gender = gender;
            this.age = age;
            this.email = email;
            this.phoneNumber = phoneNumber;
        }

        public int PassengerID { get => passengerID; set => passengerID = value; }
        public string FirstName { get => firstName; set => firstName = value; }
        public string Gender { get => gender; set => gender = value; }
        public int Age { get => age; set => age = value; }
        public string Email { get => email; set => email = value; }
        public string PhoneNumber { get => phoneNumber; set => phoneNumber = value; }
    }
}
```

Driver.cs

```
namespace TransportManagementSystem.entity
{
    public class Driver
    {
        private int driverID;
        private string name;
        private string phoneNumber;
        private string status;
        private string firstName;
        private string licenseNumber;

        public Driver() { }

        public Driver(int driverID, string status, string name, string phoneNumber, string firstName,
string licenseNumber)
        {
            this.driverID = driverID;
            this.status = status;
            this.name = name;
            this.phoneNumber = phoneNumber;
            this.firstName = firstName;
            this.licenseNumber = licenseNumber;
        }

        public int DriverID { get => driverID; set => driverID = value; }
        public string Name { get => name; set => name = value; }
        public string PhoneNumber { get => phoneNumber; set => phoneNumber = value; }
        public string Status { get => status; set => status = value; }

        public string FirstName { get => firstName; set => firstName = value; }        // ✅ property
        public string LicenseNumber { get => licenseNumber; set => licenseNumber = value; } // ✅
property
    }
}
```

Booking.cs

```
using System;

namespace TransportManagementSystem.entity
{

    public class Booking
    {
        private int bookingID;
        private int tripID;
        private int passengerID;
        private DateTime bookingDate;
        private string status;

        public Booking() { }

        public Booking(int bookingID, int tripID, int passengerID, DateTime bookingDate, string status)
        {
            this.bookingID = bookingID;
            this.tripID = tripID;
            this.passengerID = passengerID;
            this.bookingDate = bookingDate;
            this.status = status;
        }

        public int BookingID { get => bookingID; set => bookingID = value; }
        public int TripID { get => tripID; set => tripID = value; }
        public int PassengerID { get => passengerID; set => passengerID = value; }
        public DateTime BookingDate { get => bookingDate; set => bookingDate = value; }
        public string Status { get => status; set => status = value; }
    }
}
```

Dao

ITransportService.cs

```
using System.Collections.Generic;
using TransportManagementSystem.entity;

namespace TransportManagementSystem.dao
{
    public interface ITransportManagementService
    {
        bool AddVehicle(Vehicle vehicle);
        bool UpdateVehicle(Vehicle vehicle);
        bool DeleteVehicle(int vehicleId);
        bool ScheduleTrip(int vehicleId, int routeId, string departureDate, string arrivalDate);
        bool CancelTrip(int tripId);
        bool BookTrip(int tripId, int passengerId, string bookingDate);
        bool CancelBooking(int bookingId);
        bool AllocateDriver(int tripId, int driverId);
        bool DeallocateDriver(int tripId);
        List<Booking> GetBookingsByPassenger(int passengerId);
        List<Booking> GetBookingsByTrip(int tripId);
        List<Driver> GetAvailableDrivers();
    }
}
```

ServiceProcess.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using TransportManagementSystem.entity;
using TransportManagementSystem.exception;
using TransportManagementSystem.util;

namespace TransportManagementSystem.dao
{
    public class TransportManagementServiceImpl : ITransportManagementService
    {
        public bool AddVehicle(Vehicle vehicle)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "INSERT INTO Vehicles (Model, Capacity, Type, Status) VALUES (@Model,
@Capacity, @Type, @Status)";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@Model", vehicle.Model);
                cmd.Parameters.AddWithValue("@Capacity", vehicle.Capacity);
                cmd.Parameters.AddWithValue("@Type", vehicle.Type);
                cmd.Parameters.AddWithValue("@Status", vehicle.Status);
                return cmd.ExecuteNonQuery() > 0;
            }
        }

        public bool UpdateVehicle(Vehicle vehicle)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "UPDATE Vehicles SET Model=@Model, Capacity=@Capacity, Type=@Type,
Status=@Status WHERE VehicleID=@VehicleID";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@Model", vehicle.Model);
                cmd.Parameters.AddWithValue("@Capacity", vehicle.Capacity);
                cmd.Parameters.AddWithValue("@Type", vehicle.Type);
                cmd.Parameters.AddWithValue("@Status", vehicle.Status);
                cmd.Parameters.AddWithValue("@VehicleID", vehicle.VehicleID);
                int rows = cmd.ExecuteNonQuery();
                if (rows == 0)
                    throw new VehicleNotFoundException("Vehicle ID not found.");
                return true;
            }
        }

        public bool DeleteVehicle(int vehicleId)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "DELETE FROM Vehicles WHERE VehicleID=@VehicleID";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@VehicleID", vehicleId);
                int rows = cmd.ExecuteNonQuery();
                if (rows == 0)
                    throw new VehicleNotFoundException("Vehicle ID not found.");
                return true;
            }
        }

        public bool ScheduleTrip(int vehicleId, int routeId, string departureDate, string arrivalDate)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = @"INSERT INTO Trips (VehicleID, RouteID, DepartureDate, ArrivalDate, Status,
TripType, MaxPassengers)
                    VALUES (@VehicleID, @RouteID, @DepartureDate, @ArrivalDate, 'Scheduled',
'Passenger', 20)";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@VehicleID", vehicleId);
                cmd.Parameters.AddWithValue("@RouteID", routeId);
                cmd.Parameters.AddWithValue("@DepartureDate", DateTime.Parse(departureDate));
                cmd.Parameters.AddWithValue("@ArrivalDate", DateTime.Parse(arrivalDate));
                return cmd.ExecuteNonQuery() > 0;
            }
        }

        public bool CancelTrip(int tripId)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "UPDATE Trips SET Status='Cancelled' WHERE TripID=@TripID";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@TripID", tripId);
                return cmd.ExecuteNonQuery() > 0;
            }
        }

        public bool BookTrip(int tripId, int passengerId, string bookingDate)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = @"INSERT INTO Bookings (TripID, PassengerID, BookingDate, Status)
                    VALUES (@TripID, @PassengerID, @BookingDate, 'Confirmed')";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@TripID", tripId);
                cmd.Parameters.AddWithValue("@PassengerID", passengerId);
                cmd.Parameters.AddWithValue("@BookingDate", DateTime.Parse(bookingDate));
                return cmd.ExecuteNonQuery() > 0;
            }
        }

        public bool CancelBooking(int bookingId)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "UPDATE Bookings SET Status='Cancelled' WHERE BookingID=@BookingID";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@BookingID", bookingId);
                return cmd.ExecuteNonQuery() > 0;
            }
        }

        public bool AllocateDriver(int tripId, int driverId)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = @"INSERT INTO DriverTrip (TripID, DriverID) VALUES (@TripID, @DriverID)";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@TripID", tripId);
                cmd.Parameters.AddWithValue("@DriverID", driverId);
                return cmd.ExecuteNonQuery() > 0;
            }
        }

        public bool DeallocateDriver(int tripId)
        {
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "DELETE FROM DriverTrip WHERE TripID=@TripID";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@TripID", tripId);
                return cmd.ExecuteNonQuery() > 0;
            }
        }

        public List<Booking> GetBookingsByPassenger(int passengerId)
        {
            List<Booking> bookings = new List<Booking>();
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "SELECT * FROM Bookings WHERE PassengerID=@PassengerID";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@PassengerID", passengerId);
                SqlDataReader reader = cmd.ExecuteReader();
                while (reader.Read())
                {
                    bookings.Add(new Booking
                    {
                        BookingID = Convert.ToInt32(reader["BookingID"]),
                        TripID = Convert.ToInt32(reader["TripID"]),
                        PassengerID = Convert.ToInt32(reader["PassengerID"]),
                        BookingDate = Convert.ToDateTime(reader["BookingDate"]),
                        Status = reader["Status"].ToString()
                    });
                }
            }
            if (bookings.Count == 0)
                throw new BookingNotFoundException("No bookings found for the given passenger ID.");

            return bookings;
        }

        public List<Booking> GetBookingsByTrip(int tripId)
        {
            List<Booking> bookings = new List<Booking>();
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = "SELECT * FROM Bookings WHERE TripID=@TripID";
                SqlCommand cmd = new SqlCommand(query, conn);
                cmd.Parameters.AddWithValue("@TripID", tripId);
                SqlDataReader reader = cmd.ExecuteReader();
                while (reader.Read())
                {
                    bookings.Add(new Booking
                    {
                        BookingID = Convert.ToInt32(reader["BookingID"]),
                        TripID = Convert.ToInt32(reader["TripID"]),
                        PassengerID = Convert.ToInt32(reader["PassengerID"]),
                        BookingDate = Convert.ToDateTime(reader["BookingDate"]),
                        Status = reader["Status"].ToString()
                    });
                }
            }

            if (bookings.Count == 0)
                throw new BookingNotFoundException("No bookings found for the given trip ID.");

            return bookings;
        }

        public List<Driver> GetAvailableDrivers()
        {
            List<Driver> drivers = new List<Driver>();
            using (SqlConnection conn = DBUtil.GetConnection())
            {
                conn.Open();
                string query = @"
                SELECT * FROM Drivers
                WHERE DriverID NOT IN (SELECT DriverID FROM DriverTrip)";
                SqlCommand cmd = new SqlCommand(query, conn);
                SqlDataReader reader = cmd.ExecuteReader();
                while (reader.Read())
                {
                    drivers.Add(new Driver
                    {
                        DriverID = Convert.ToInt32(reader["DriverID"]),
                        FirstName = reader["FirstName"].ToString(),
                        LicenseNumber = reader["LicenseNumber"].ToString(),
                        PhoneNumber = reader["PhoneNumber"].ToString()
                    });
                }
            }
            return drivers;
        }
    }
}
```

Exception

Transportexceptions.cs

```
using System;

namespace TransportManagementSystem.exception
{
    public class VehicleException : Exception
    {
        public VehicleException(string msg) : base(msg) { }
    }

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

    public class TripException : Exception
    {
        public TripException(string msg) : base(msg) { }
    }

    public class TripNotFoundException : Exception
    {
        public TripNotFoundException(string msg) : base(msg) { }
    }

    public class BookingException : Exception
    {
        public BookingException(string msg) : base(msg) { }
    }
}
public class BookingNotFoundException : Exception
{
    public BookingNotFoundException(string msg) : base(msg) { }
}
    public class VehicleNotFoundException : Exception
    {
        public VehicleNotFoundException(string message) : base(message) { }
    }
```

Util

DBUtil.cs

```
using System;
using System.Data.SqlClient;
using System.IO;
using System.Linq;

namespace TransportManagementSystem.util
{
    public class DBUtil
    {

        public static string GetConnectionString()
        {
            return "Server=(localdb)\\MSSQLLocalDB;Database=TransportDB;Trusted_Connection=True;";
        }

        public static string GetConnectionStringFromFile(string fileName)
        {
            var config = File.ReadAllLines(fileName)
                .ToDictionary(line => line.Split('=')[0].Trim(), line => line.Split('=')[1].Trim());

            return config["ConnectionString"];
        }

        public static SqlConnection GetConnection()
        {
            string connStr = GetConnectionString();
            return new SqlConnection(connStr);
        }

        public static SqlConnection GetConnection(string fileName)
        {
            string connStr = GetConnectionStringFromFile(fileName);
            return new SqlConnection(connStr);
        }
    }
}
```

Main

MainModule.cs

```
using System;
using TransportManagementSystem.dao;
using TransportManagementSystem.entity;
using TransportManagementSystem.exception;
using System.Collections.Generic;

namespace TransportManagementSystem.main
{
    public class MainModule
    {
        static ITransportManagementService transportService = new
TransportManagementServiceImpl();

        public static void Main(string[] args)
        {
            bool exit = false;
            while (!exit)
            {
                Console.WriteLine("\n--- Transport Management System Menu ---");
                Console.WriteLine("1. Add Vehicle");
                Console.WriteLine("2. Update Vehicle");
                Console.WriteLine("3. Delete Vehicle");
                Console.WriteLine("4. Schedule Trip");
                Console.WriteLine("5. Cancel Trip");
                Console.WriteLine("6. Book Trip");
                Console.WriteLine("7. Cancel Booking");
                Console.WriteLine("8. View Bookings by Passenger");
                Console.WriteLine("9. View Bookings by Trip");
                Console.WriteLine("10. Allocate Driver");
                Console.WriteLine("11. Deallocate Driver");
                Console.WriteLine("12. View Available Drivers");
                Console.WriteLine("0. Exit");
                Console.Write("Enter your choice: ");

                string choice = Console.ReadLine();

                try
                {
                    switch (choice)
                    {
                        case "1": AddVehicle(); break;
                        case "2": UpdateVehicle(); break;
                        case "3": DeleteVehicle(); break;
                        case "4": ScheduleTrip(); break;
                        case "5": CancelTrip(); break;
                        case "6": BookTrip(); break;
                        case "7": CancelBooking(); break;
                        case "8": ViewBookingsByPassenger(); break;
                        case "9": ViewBookingsByTrip(); break;
                        case "10": AllocateDriver(); break;
                        case "11": DeallocateDriver(); break;
                        case "12": ViewAvailableDrivers(); break;
                        case "0":
                            exit = true;
                            Console.WriteLine("Exiting application. Goodbye!");
                            break;
                        default:
                            Console.WriteLine("Invalid choice! Please enter a valid option.");
                            break;
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Error: " + ex.Message);
                }
            }
        }

        static void AddVehicle()
        {
            Console.WriteLine("\n--- Add Vehicle ---");
            Vehicle v = new Vehicle();

            Console.Write("Model: ");
            v.Model = Console.ReadLine();

            Console.Write("Capacity (decimal): ");
            v.Capacity = decimal.Parse(Console.ReadLine());

            Console.Write("Type (Truck/Van/Bus): ");
            v.Type = Console.ReadLine();

            Console.Write("Status (Available/On Trip/Maintenance): ");
            v.Status = Console.ReadLine();

            transportService.AddVehicle(v);
            Console.WriteLine("Vehicle added successfully.");
        }

        static void UpdateVehicle()
        {
            Console.WriteLine("\n--- Update Vehicle ---");
            Vehicle v = new Vehicle();

            Console.Write("VehicleID to update: ");
            v.VehicleID = int.Parse(Console.ReadLine());

            Console.Write("New Model: ");
            v.Model = Console.ReadLine();
            Console.Write("New Capacity (decimal): ");
            v.Capacity = decimal.Parse(Console.ReadLine());

            Console.Write("New Type (Truck/Van/Bus): ");
            v.Type = Console.ReadLine();

            Console.Write("New Status (Available/On Trip/Maintenance): ");
            v.Status = Console.ReadLine();

            transportService.UpdateVehicle(v);
            Console.WriteLine("Vehicle updated successfully.");
        }

        static void DeleteVehicle()
        {
            Console.WriteLine("\n--- Delete Vehicle ---");
            Console.Write("VehicleID to delete: ");
            int id = int.Parse(Console.ReadLine());

            transportService.DeleteVehicle(id);
            Console.WriteLine("Vehicle deleted successfully.");
        }

        static void ScheduleTrip()
        {
            Console.WriteLine("\n--- Schedule Trip ---");

            Console.Write("VehicleID: ");
            int vehicleId = int.Parse(Console.ReadLine());

            Console.Write("RouteID: ");
            int routeId = int.Parse(Console.ReadLine());

            Console.Write("Departure Date (yyyy-MM-dd HH:mm): ");
            string departureDate = DateTime.Parse(Console.ReadLine()).ToString("yyyy-MM-dd HH:mm");

            Console.Write("Arrival Date (yyyy-MM-dd HH:mm): ");
            string arrivalDate = DateTime.Parse(Console.ReadLine()).ToString("yyyy-MM-dd HH:mm");

            bool success = transportService.ScheduleTrip(vehicleId, routeId, departureDate, arrivalDate);

            Console.WriteLine(success ? "Trip scheduled successfully." : "Failed to schedule trip.");
        }

        static void CancelTrip()
        {
            Console.WriteLine("\n--- Cancel Trip ---");
            Console.Write("TripID to cancel: ");
            int id = int.Parse(Console.ReadLine());
```

```
            bool success = transportService.CancelTrip(id);
            Console.WriteLine(success ? "Trip cancelled successfully." : "Failed to cancel trip.");
        }

        static void BookTrip()
        {
            Console.WriteLine("\n--- Book Trip ---");
            Console.Write("TripID: ");
            int tripId = int.Parse(Console.ReadLine());

            Console.Write("PassengerID: ");
            int passengerId = int.Parse(Console.ReadLine());

            string bookingDate = DateTime.Now.ToString("yyyy-MM-dd HH:mm");

            bool success = transportService.BookTrip(tripId, passengerId, bookingDate);

            Console.WriteLine(success ? "Booking successful." : "Booking failed.");
        }

        static void CancelBooking()
        {
            Console.WriteLine("\n--- Cancel Booking ---");
            Console.Write("BookingID to cancel: ");
            int id = int.Parse(Console.ReadLine());

            bool success = transportService.CancelBooking(id);
            Console.WriteLine(success ? "Booking cancelled successfully." : "Failed to cancel booking.");
        }

        static void ViewBookingsByPassenger()
        {
            Console.WriteLine("\n--- View Bookings by Passenger ---");
            Console.Write("PassengerID: ");
            int id = int.Parse(Console.ReadLine());

            List<Booking> bookings = transportService.GetBookingsByPassenger(id);

            if (bookings.Count == 0)
            {
                Console.WriteLine("No bookings found.");
                return;
            }

            foreach (var b in bookings)
            {
                Console.WriteLine($"BookingID: {b.BookingID}, TripID: {b.TripID}, Date: {b.BookingDate},
Status: {b.Status}");
            }
        }
        static void ViewBookingsByTrip()
        {
            Console.WriteLine("\n--- View Bookings by Trip ---");
            Console.Write("TripID: ");
            int id = int.Parse(Console.ReadLine());

            List<Booking> bookings = transportService.GetBookingsByTrip(id);

            if (bookings.Count == 0)
            {
                Console.WriteLine("No bookings found.");
                return;
            }

            foreach (var b in bookings)
            {
                Console.WriteLine($"BookingID: {b.BookingID}, PassengerID: {b.PassengerID}, Date:
{b.BookingDate}, Status: {b.Status}");
            }
        }

        static void AllocateDriver()
        {
            Console.WriteLine("\n--- Allocate Driver to Trip ---");
            Console.Write("TripID: ");
            int tripId = int.Parse(Console.ReadLine());
            Console.Write("DriverID: ");
            int driverId = int.Parse(Console.ReadLine());

            bool result = transportService.AllocateDriver(tripId, driverId);
            Console.WriteLine(result ? "Driver allocated successfully." : "Allocation failed.");
        }

        static void DeallocateDriver()
        {
            Console.WriteLine("\n--- Deallocate Driver from Trip ---");
            Console.Write("TripID: ");
            int tripId = int.Parse(Console.ReadLine());

            bool result = transportService.DeallocateDriver(tripId);
            Console.WriteLine(result ? "Driver deallocated successfully." : "Deallocation failed.");
        }

        static void ViewAvailableDrivers()
        {
            Console.WriteLine("\n--- Available Drivers ---");
            var drivers = transportService.GetAvailableDrivers();

            if (drivers.Count == 0)
            {
                Console.WriteLine("No available drivers.");
                return;
            }

            foreach (var d in drivers)
            {
                Console.WriteLine($"DriverID: {d.DriverID}, Name: {d.Name}, Phone: {d.PhoneNumber},
Status: {d.Status}");
            }
        }
      }
}
```

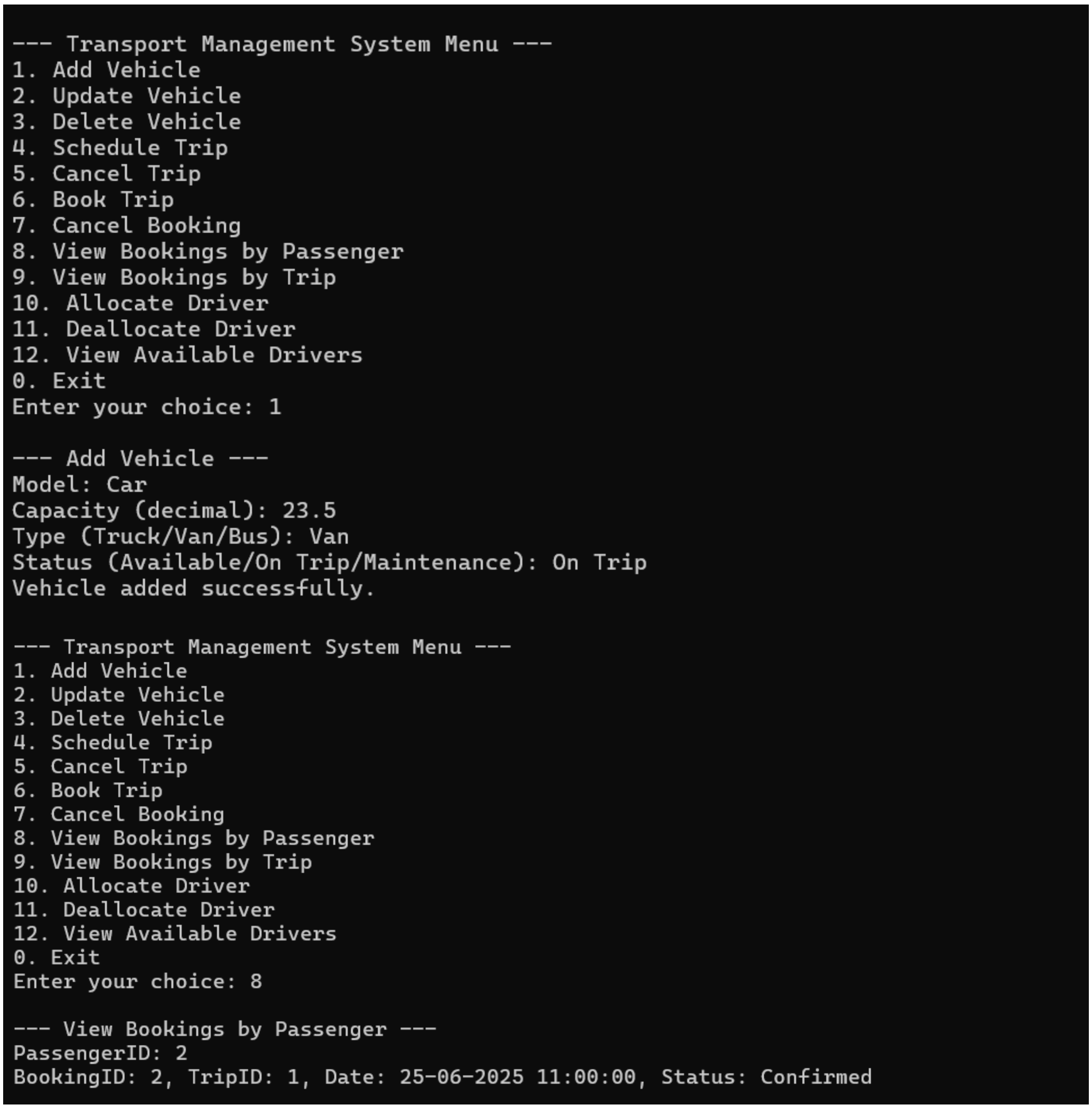
Test

TransportTest.cs

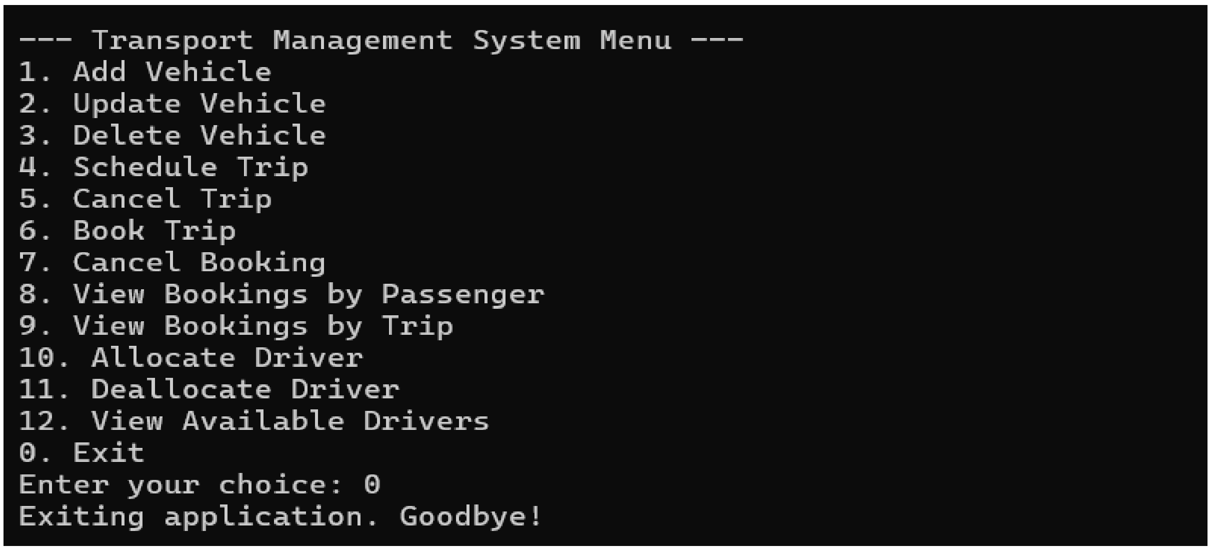
```
using NUnit.Framework;
using TransportManagementSystem.dao;
using TransportManagementSystem.entity;
using TransportManagementSystem.myexceptions;
using System;

namespace TransportManagementSystem.Tests
{
  [TestFixture]
  public class TransportServiceTests
  {
    private ITransportManagementService _service;

    [SetUp]
    public void Setup()
    {
      _service = new TransportManagementServiceImpl();
    }


    [Test]
    public void AddVehicle_ShouldAddSuccessfully()
    {
      var vehicle = new Vehicle(0, "Unit Test Van", 7.5m, "Van", "Available");
      bool result = _service.AddVehicle(vehicle);
      Assert.IsTrue(result);
    }

    [Test]
    public void BookTrip_ShouldCreateBooking()
    {
      bool result = _service.BookTrip(1, 1, DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"));
      Assert.IsTrue(result);
```

```
        }



        [Test]
        public void GetBooking_InvalidBooking_ShouldThrowException()
        {
            Assert.Throws<BookingNotFoundException>(() => _service.GetBookingsByPassenger(-100));
        }
    }
}
```

OUTPUT

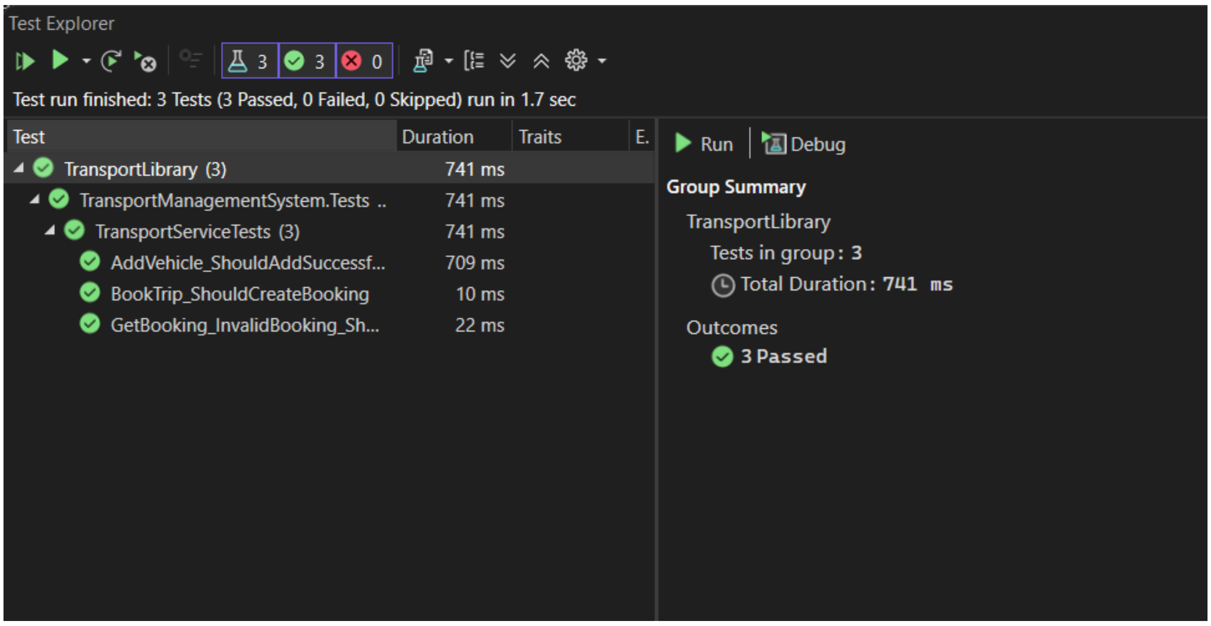


```
--- Transport Management System Menu ---
1. Add Vehicle
2. Update Vehicle
3. Delete Vehicle
4. Schedule Trip
5. Cancel Trip
6. Book Trip
7. Cancel Booking
8. View Bookings by Passenger
9. View Bookings by Trip
10. Allocate Driver
11. Deallocate Driver
12. View Available Drivers
0. Exit
Enter your choice: 1

--- Add Vehicle ---
Model: Car
Capacity (decimal): 23.5
Type (Truck/Van/Bus): Van
Status (Available/On Trip/Maintenance): On Trip
Vehicle added successfully.

--- Transport Management System Menu ---
1. Add Vehicle
2. Update Vehicle
3. Delete Vehicle
4. Schedule Trip
5. Cancel Trip
6. Book Trip
7. Cancel Booking
8. View Bookings by Passenger
9. View Bookings by Trip
10. Allocate Driver
11. Deallocate Driver
12. View Available Drivers
0. Exit
Enter your choice: 8

--- View Bookings by Passenger ---
PassengerID: 2
BookingID: 2, TripID: 1, Date: 25-06-2025 11:00:00, Status: Confirmed
```



```
--- Transport Management System Menu ---
1. Add Vehicle
2. Update Vehicle
3. Delete Vehicle
4. Schedule Trip
5. Cancel Trip
6. Book Trip
7. Cancel Booking
8. View Bookings by Passenger
9. View Bookings by Trip
10. Allocate Driver
11. Deallocate Driver
12. View Available Drivers
0. Exit
Enter your choice: 0
Exiting application. Goodbye!
```

TESTING



Test run finished: 3 Tests (3 Passed, 0 Failed, 0 Skipped) run in 1.7 sec

Test	Duration	Traits	E.
TransportLibrary (3)	741 ms		
TransportManagementSystem.Tests ..	741 ms		
TransportServiceTests (3)	741 ms		
AddVehicle_ShouldAddSuccessf...	709 ms		
BookTrip_ShouldCreateBooking	10 ms		
GetBooking_InvalidBooking_Sh...	22 ms		

Group Summary
TransportLibrary
Tests in group: 3
Total Duration: 741 ms

Outcomes
3 Passed