

C# ASSIGNMENT COURIER MANAGEMENT SYSTEM

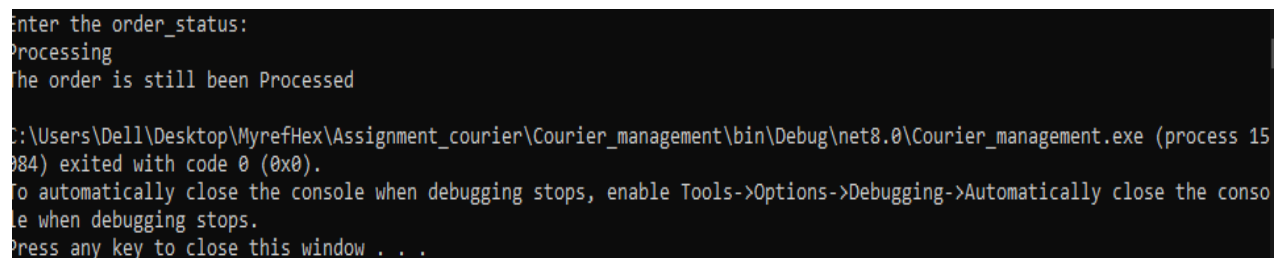
NAME: KARTHICK M

Coding Task 1: Control Flow Statements

1. Write a program that checks whether a given order is delivered or not based on its status (e.g., "Processing," "Delivered," "Cancelled"). Use if-else statements for this.

```
using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.chk_status();
        }

        public void chk_status()    //1
        {
            Console.WriteLine("Enter the order_status: ");
            string order_status = Console.ReadLine();
            if (order_status == "Delivered")
            {
                Console.WriteLine("The order has been Delivered");
            }
            else if (order_status == "Processing")
            {
                Console.WriteLine("The order is still been Processed ");
            }
            else if (order_status == "Cancelled")
            {
                Console.WriteLine("The order has been cancelled");
            }
            else
            {
                Console.WriteLine("Invalid order status. Please check the status
again.");
            }
        }
    }
}
```



```
Enter the order_status:
Processing
The order is still been Processed

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Courier_management.exe (process 15884) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

2. Implement a switch-case statement to categorize parcels based on their weight into "Light," "Medium," or "Heavy."

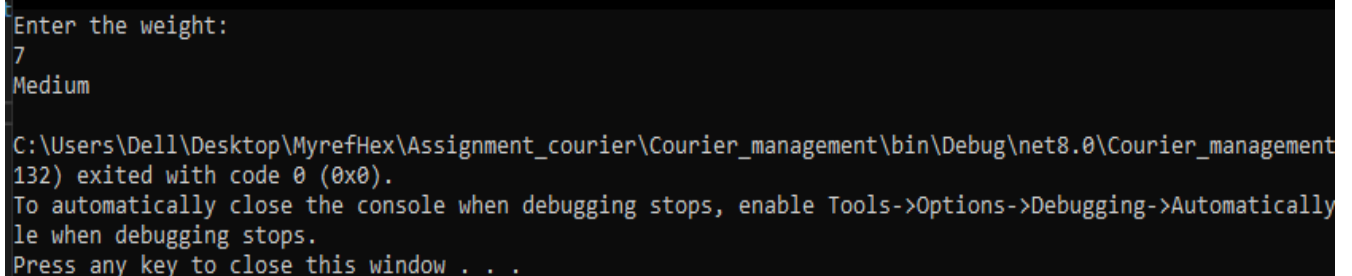
```
using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.category();
        }

        public void category() //2
        {
            Console.WriteLine("Enter the weight:");
            int weight = int.Parse(Console.ReadLine());
            switch (weight)
            {
                case int w when w < 5:
                    Console.WriteLine("Light");
                    break;

                case int w when w >= 5 && w < 10:
                    Console.WriteLine("Medium");
                    break;

                case int w when w >= 10:
                    Console.WriteLine("Heavy");
                    break;

                default:
                    Console.WriteLine("Invalid weight");
                    break;
            }
        }
    }
}
```



```
Enter the weight:
7
Medium

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Courier_management
132) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically
le when debugging stops.
Press any key to close this window . . .
```

3. Implement User Authentication 1. Create a login system for employees and customers using control flow statements.

```

using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.login();
        }
    }
}

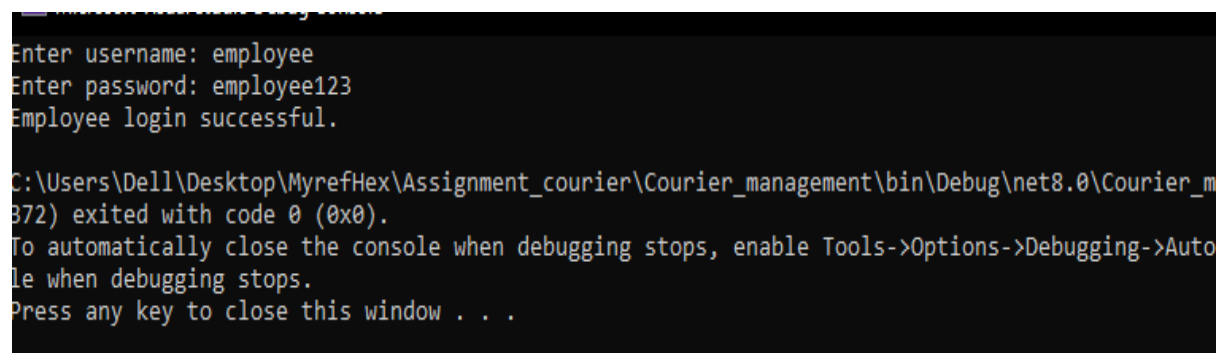
public void login() //3
{
    string employeeUsername = "employee";
    string employeePassword = "employee123";
    string customerUsername = "customer";
    string customerPassword = "customer123";

    Console.WriteLine("Enter username: ");
    string usernameInput = Console.ReadLine();

    Console.WriteLine("Enter password: ");
    string passwordInput = Console.ReadLine();

    if (usernameInput == employeeUsername && passwordInput == employeePassword)
    {
        Console.WriteLine("Employee login successful.");
    }
    else if (usernameInput == customerUsername && passwordInput == customerPassword)
    {
        Console.WriteLine("Customer login successful.");
    }
    else
    {
        Console.WriteLine("Invalid username or password");
    }
}
}

```



```

Enter username: employee
Enter password: employee123
Employee login successful.

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Courier_m
372) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Auto
le when debugging stops.
Press any key to close this window . . .

```

4. Implement Courier Assignment Logic 1. Develop a mechanism to assign couriers to shipments based on predefined criteria (e.g., proximity, load capacity) using loops.

```

using System;
namespace Courier_management
{

```

```

internal class Program
{
    static void Main(string[] args)
    {
        Program program = new Program();
        program.logic() ;
    }
}

public void logic() //4
{
    string[] couriers = { "Anu", "Abi", "Ravi" };
    string[] areas = { "Chennai", "Mumbai", "Delhi" };
    int[] capacities = { 100, 80, 60 };
    int[] loads = { 30, 20, 50 };

    string shipmentArea = "Mumbai";
    int shipmentWeight = 20;

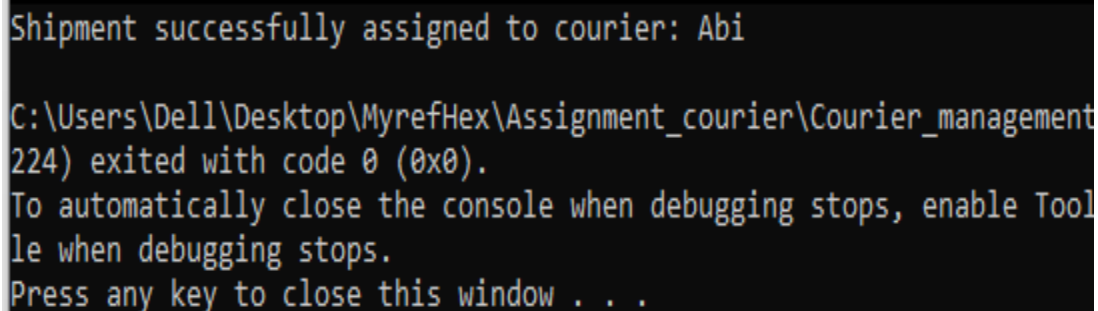
    bool isAssigned = false;

    for (int i = 0; i < couriers.Length; i++)
    {
        bool isSameArea = areas[i] == shipmentArea;
        bool hasCapacity = loads[i] + shipmentWeight <= capacities[i];

        if (isSameArea && hasCapacity)
        {
            Console.WriteLine($"Shipment successfully assigned to courier:
{couriers[i]}");
            loads[i] += shipmentWeight;
            isAssigned = true;
            break;
        }
    }

    if (!isAssigned)
    {
        Console.WriteLine("No suitable courier available for this shipment.");
    }
}
}
}

```



```

Shipment successfully assigned to courier: Abi

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management
224) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tool
le when debugging stops.
Press any key to close this window . . .

```

5. Write a program that uses a for loop to display all the orders for a specific customer.

```
using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.specific_cust();
        }

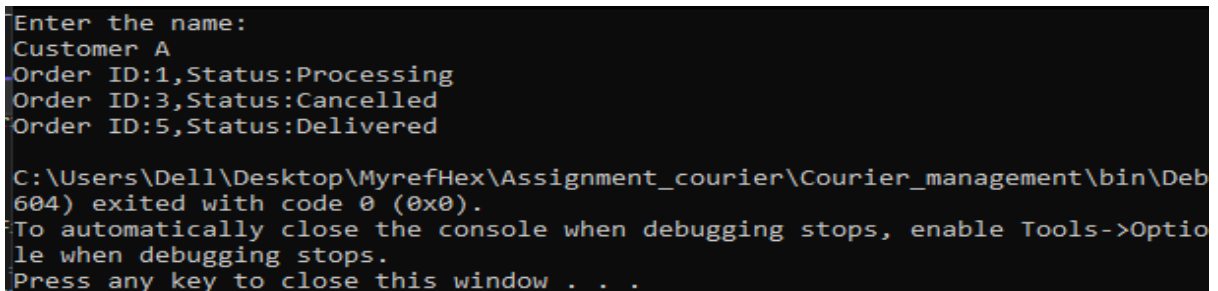
        public void specific_cust() /
        {
            int[] orderid = { 1, 2, 3, 4, 5 };
            string[] customer_name = { "Customer A", "Customer B", "Customer A",
"Customer C", "Customer A" };
            string[] status = { "Processing", "Delivered", "Cancelled", "Processing",
"Delivered" };

            Console.WriteLine("Enter the name: ");
            string name = Console.ReadLine();

            bool orderFound = false;

            for (int index = 0; index < customer_name.Length; index++)
            {
                if (customer_name[index] == name)
                {
                    Console.WriteLine($"OrderID:{orderid[index]},Status:{status[index]}");
                    orderFound = true;
                }
            }

            if (!orderFound)
            {
                Console.WriteLine("No order Found");
            }
        }
    }
}
```



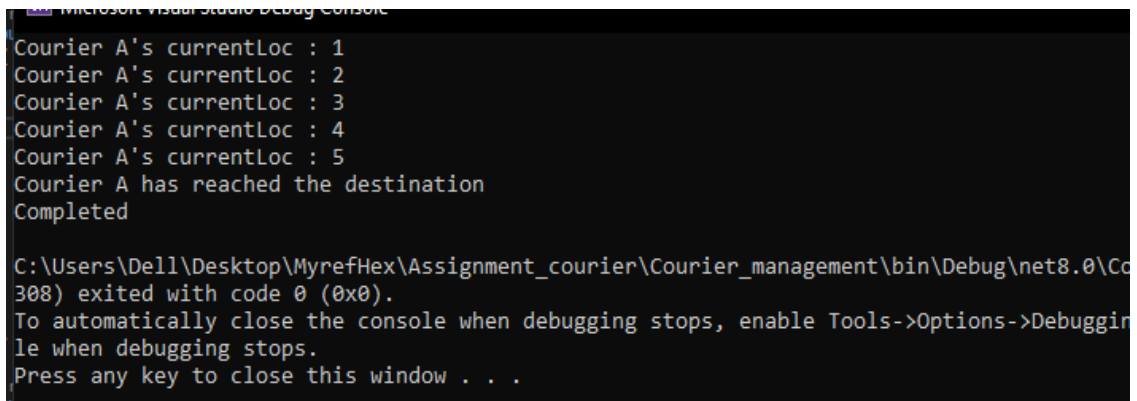
```
Enter the name:
Customer A
Order ID:1,Status:Processing
Order ID:3,Status:Cancelled
Order ID:5,Status:Delivered

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Deb604) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options when debugging stops.
Press any key to close this window . . .
```

6. Implement a while loop to track the real-time location of a courier until it reaches its destination.

```
using System;
using System.Threading;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.track();
        }
    }
}

public void track()
{
    string courierName = "Courier A";
    int currentLoc = 0;
    int destination = 5;
    while (currentLoc < destination)
    {
        currentLoc++;
        Console.WriteLine($"{courierName}'s currentLoc : {currentLoc}");
        Thread.Sleep(1000);
    }
    Console.WriteLine($"{courierName} has reached the destination ");
    Console.WriteLine("Completed");
}
}
```



```
Microsoft Visual Studio Debug Console
Courier A's currentLoc : 1
Courier A's currentLoc : 2
Courier A's currentLoc : 3
Courier A's currentLoc : 4
Courier A's currentLoc : 5
Courier A has reached the destination
Completed

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Co
308) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debuggin
le when debugging stops.
Press any key to close this window . . .
```

7. Create an array to store the tracking history of a parcel, where each entry represents a location update.

```
using System;
using System.Threading;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
```

```

        {
            Program program = new Program();
            program.history();
        }
    public void history()
    {
        string parcelName = "ABC321";
        string[] location = { "Warehouse", "In Transit", "Local Distribution Center", "Delivered" };
        string[,] tracking_hist = new string[location.Length, 2];

        for (int index = 0; index < location.Length; index++)
        {
            string timestamp = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");
            tracking_hist[index, 0] = timestamp;
            tracking_hist[index, 1] = location[index];
            Console.WriteLine($"{timestamp} - Location: {location[index]}");
            Thread.Sleep(1000);
        }
    }
}

```

```

2025-04-07 09:52:21 - Location: Warehouse
2025-04-07 09:52:22 - Location: In Transit
2025-04-07 09:52:23 - Location: Local Distribution Center
2025-04-07 09:52:24 - Location: Delivered

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8
68) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Deb
le when debugging stops.
Press any key to close this window . . .

```

8. Implement a method to find the nearest available courier for a new order using an array of couriers.

```

using System;
using System.Threading;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.nearest();
        }
    public void nearest()
    {
        string[] courier = { "Courier A", "Courier B", "Courier C" };
        int[] distance = { 10, 5, 15 };
        int[] capacity = { 20, 10, 25 };

        Console.WriteLine("Enter the order : ");
        int order = int.Parse(Console.ReadLine());
    }
}

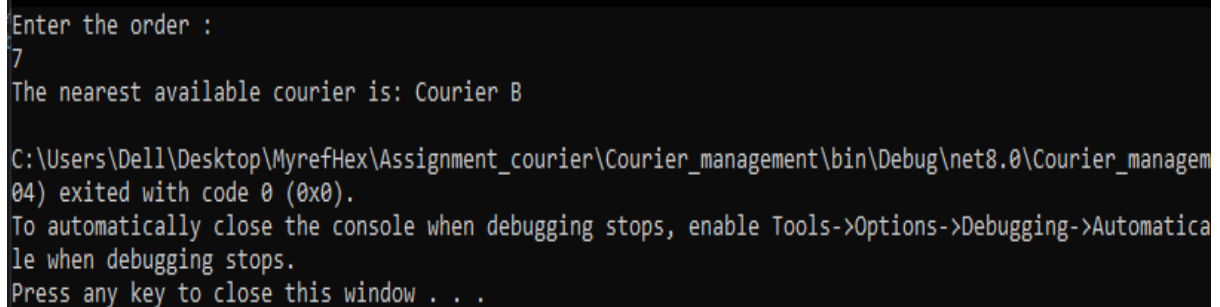
```

```

int nearest_index = -1;
int shortest_distance = int.MaxValue;

for (int index = 0; index < courier.Length; index++)
{
    if (capacity[index] >= order)
    {
        if (distance[index] < shortest_distance)
        {
            shortest_distance = distance[index];
            nearest_index = index;
        }
    }
}
if (nearest_index != -1)
{
    Console.WriteLine($"The nearest available courier is:
{courier[nearest_index]}");
}
else
{
    Console.WriteLine("No available couriers.");
}
}
}

```



```

Enter the order :
7
The nearest available courier is: Courier B

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Courier_management.exe
04) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically
close when debugging stops.
Press any key to close this window . . .

```

9. Parcel Tracking: Create a program that allows users to input a parcel tracking number. Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel delivered" based on the tracking number's status.

```

using System;
using System.Threading;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.parcel_tracking();
        }
        public void parcel_tracking() //9
    }
}

```



```

{
    string[,] parceldata = { { "ABC123", "Parcel in transit" },
    { "DEF456", "Parcel out for delivery" },
    { "GHI789", "Parcel delivered" },
    { "JKL012", "Parcel in transit" } };

    Console.WriteLine("Enter the tracking_number : ");
    string tracking_number = Console.ReadLine();
    bool isFound = false;
    for (int index = 0; index < parceldata.Length; index++)
    {
        if (parceldata[index, 0] == tracking_number)
        {
            Console.WriteLine($"Status for {tracking_number} :
{parceldata[index, 1]} ");
            isFound = true;
            break;
        }
    }

    if (!isFound)
    {
        Console.WriteLine("Not found");
    }
}
}
}

```

```

Enter the tracking_number :
GHI789
Status for GHI789 : Parcel delivered

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\
244) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugg
le when debugging stops.
Press any key to close this window . . .

```

10. Customer Data Validation: Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name address or phone number. Validate customer information based on following criteria. Ensure that names contain only letters and are properly capitalized, addresses do not contain special characters, and phone numbers follow a specific format (e.g., ###-###-####).

```

using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();

            program.data_validation();
        }
    }
    public void data_validation() //10
    {

```

```

Console.WriteLine("Enter the name: ");
String name = Console.ReadLine();
if (Regex.IsMatch(name, @"^[a-zA-Z\s]+$"))
{
    Console.WriteLine("Valid Name");
}
else
{
    Console.WriteLine("Invalid name");
}

Console.WriteLine("Enter the Address: ");
String address = Console.ReadLine();
if (Regex.IsMatch(address, @"^[a-zA-Z0-9\s,.-]+$"))
{
    Console.WriteLine("Valid Address");
}
else
{
    Console.WriteLine("Invalid Address");
}

Console.WriteLine("Enter the Phone_number: ");
String Phone_number = Console.ReadLine();
if (Regex.IsMatch(Phone_number, @"^\d{3}-\d{3}-\d{4}$"))
{
    Console.WriteLine("Valid Phone Number");
}
else
{
    Console.WriteLine("Invalid number");
}
}
}
}

```

```

Enter the name:
Karthick
Valid Name
Enter the Address:
123 main road
Valid Address
Enter the Phone_number:
222-456-7890
Valid Phone Number

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug
972) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Option
le when debugging stops.
Press any key to close this window . . .

```

11. Address Formatting: Develop a function that takes an address as input (street, city, state, zip code) and formats it correctly, including capitalizing the first letter of each word and properly formatting the zip code.

```

using System;
namespace Courier_management

```

```

{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.address_formatting()
        }
    }
}

public void address_formatting() //11
{
    Console.WriteLine("Enter street address:");
    string street = Console.ReadLine();

    Console.WriteLine("Enter city:");
    string city = Console.ReadLine();

    Console.WriteLine("Enter state:");
    string state = Console.ReadLine();

    Console.WriteLine("Enter zip code:");
    string zipCode = Console.ReadLine();

    string formattedStreet = string.Join(" ", Array.ConvertAll(street.Split(' '),
word => char.ToUpper(word[0]) + word.Substring(1).ToLower()));

    string formattedCity = char.ToUpper(city[0]) + city.Substring(1).ToLower();

    string formattedState = state.ToUpper();

    string formattedZipCode = zipCode.Length == 9
        ? zipCode.Substring(0, 5) + "-" + zipCode.Substring(5)
        : zipCode;

    string formattedAddress = $"{formattedStreet}, {formattedCity},
{formattedState} {formattedZipCode}";

    Console.WriteLine("Formatted Address:");
    Console.WriteLine(formattedAddress);
}
}
}

```

```

Enter street address:
Nehru street
Enter city:
Chennai
Enter state:
Tamil nadu
Enter zip code:
98999
Formatted Address:
Nehru Street, Chennai, TAMIL NADU 98999

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Courier
596) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Au
le when debugging stops.
Press any key to close this window . . .

```

12. Order Confirmation Email: Create a program that generates an order confirmation email. The email should include details such as the customer's name, order number, delivery address, and expected delivery date.

```
using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();

            program.order_Cnfrmation()
        }
    }
}

public void order_Cnfrmation()    //12
{
    Console.WriteLine("Enter customer's name:");
    string customerName = Console.ReadLine();

    Console.WriteLine("Enter order number:");
    string orderNumber = Console.ReadLine();

    Console.WriteLine("Enter delivery address:");
    string deliveryAddress = Console.ReadLine();

    DateTime expectedDeliveryDate = DateTime.Today.AddDays(2);

    string emailContent = $"
Subject: Order Confirmation - Order #{orderNumber}

Dear {customerName},

Thank you for placing an order with us! Your order #{orderNumber} has been
confirmed.

Order Details:
- Order Number: {orderNumber}
- Delivery Address: {deliveryAddress}
- Expected Delivery Date: {expectedDeliveryDate:yyyy-MM-dd}

If you have any questions or concerns, please feel free to contact our customer
support.

Thank you for choosing our service!

Best regards,
Your Company Name
";
    Console.WriteLine("\nOrder Confirmation Email:");
    Console.WriteLine(emailContent);
}
}
}
```

```

Enter customer's name:
Karthick
Enter order number:
1234
Enter delivery address:
123 main road,Chennai

Order Confirmation Email:

Subject: Order Confirmation - Order #1234

Dear Karthick,

Thank you for placing an order with us! Your order #1234 has been confirmed.

Order Details:
- Order Number: 1234
- Delivery Address: 123 main road,Chennai
- Expected Delivery Date: 2025-04-09

If you have any questions or concerns, please feel free to contact our customer support.

Thank you for choosing our service!

Best regards,
Your Company Name

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Courier_management.exe (process
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the co
Press any key to close this window . . .

```

13. Calculate Shipping Costs: Develop a function that calculates the shipping cost based on the distance between two locations and the weight of the parcel. You can use string inputs for the source and destination addresses.

```

using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.calulating_cost()
        }
    }
    public void calulating_cost() ///13
    {
        Console.WriteLine("Enter source address:");
        string sourceAddress = Console.ReadLine();

        Console.WriteLine("Enter destination address:");
        string destinationAddress = Console.ReadLine();

        Console.WriteLine("Enter parcel weight in kilograms:");
        double parcelWeight = Convert.ToDouble(Console.ReadLine());

        const double BASE_COST = 5.0;
        const double DISTANCE_COST_FACTOR = 0.1;
        const double WEIGHT_COST_FACTOR = 0.2;

        double distanceKm = 50.0;

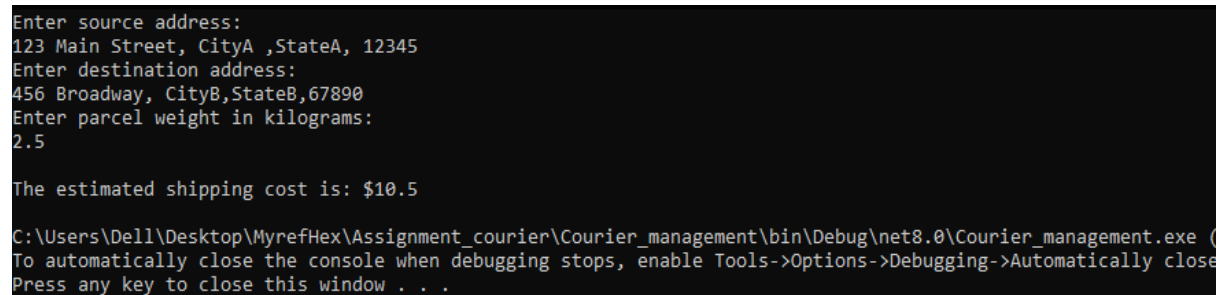
        double distanceCost = distanceKm * DISTANCE_COST_FACTOR;
        double weightCost = parcelWeight * WEIGHT_COST_FACTOR;
        double totalCost = BASE_COST + distanceCost + weightCost;
    }
}

```

```

        Console.WriteLine($"The estimated shipping cost is: ${totalCost}");
    }
}

```



```

Enter source address:
123 Main Street, CityA ,StateA, 12345
Enter destination address:
456 Broadway, CityB,StateB,67890
Enter parcel weight in kilograms:
2.5

The estimated shipping cost is: $10.5

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\net8.0\Courier_management.exe (
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close
Press any key to close this window . . .

```

14. Password Generator: Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.

```

using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();

            program.genertor();
        }
        public void genertor()
        {
            Console.Write("Enter desired password length: ");
            int length = Convert.ToInt32(Console.ReadLine());

            string digits = "0123456789";
            string special = "!@#$%^&*";
            string upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
            string lower = "abcdefghijklmnopqrstuvwxyz";

            string allCharacters = upper + lower + digits + special;
            Random rand = new Random();
            string password = "";

            for (int i = 0; i < length; i++)
            {
                int index = rand.Next(allCharacters.Length);
                password += allCharacters[index];
            }

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

```

```

Enter desired password length: 12
Generated Password: r3IBnH0qt^A5

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_management\bin\Debug\
712) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options
le when debugging stops.
Press any key to close this window . . .

```

15. Find Similar Addresses: Implement a function that finds similar addresses in the system. This can be useful for identifying duplicate customer entries or optimizing delivery routes. Use string functions to implement this.

```

using System;
namespace Courier_management
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Program program = new Program();
            program.similar_address()
        }

        public void similar_address()
        {
            List<string> addresses = new List<string>
            {
                "101 Vivekananda Street",
                "101 Vivekananda St",
                "200 Tagore Road",
                "200 Tagore Rd",
                "305 Bose Nagar",
                "305 Bose Ngr",
                "400 Netaji Avenue",
                "401 Netaji Avenue"
            };

            //Console.WriteLine("Similar addresses found:\n");

            for (int i = 0; i < addresses.Count; i++)
            {
                for (int j = i + 1; j < addresses.Count; j++)
                {
                    string addr1 = addresses[i].ToLower();
                    string addr2 = addresses[j].ToLower();

                    if (addr1.Contains(addr2) || addr2.Contains(addr1))
                    {
                        Console.WriteLine($"Similar addresses are :\n\n
                        \"{addresses[i]}\" \nand\n \"{addresses[j]}\" \n\n");
                    }
                }
            }
        }
    }
}

```

```

Similar addresses are :

    "101 Vivekananda Street"
and
    "101 Vivekananda St"

C:\Users\Dell\Desktop\MyrefHex\Assignment_courier\Courier_m
468) exited with code 0 (0x0).
To automatically close the console when debugging stops, en
le when debugging stops.
Press any key to close this window . . .

```

Task 5: Object Oriented Programming Scope : Entity classes/Models/POJO, Abstraction/Encapsulation Create the following model/entity classes within package entities with variables declared private, constructors(default and parametrized, getters, setters and toString())

1. User Class: Variables: userID , userName , email , password , contactNumber , address
2. Courier Class Variables: courierID , senderName , senderAddress , receiverName , receiverAddress , weight , status, trackingNumber , deliveryDate ,userId
3. Employee Class: Variables employeeID , employeeName , email , contactNumber , role String, salary
4. Location Class Variables LocationID , LocationName , Address
5. CourierCompany Class Variables companyName , courierDetails -collection of Courier Objects, employeeDetails -collection of Employee Objects, locationDetails - collection of Location Objects.
6. Payment Class: Variables PaymentID long, CourierID long, Amount double, PaymentDate Date

User class

```

namespace sample_project.model
{
    public class User
    {
        public int UserID { get; set; }
        public string UserName { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
        public string ContactNumber { get; set; }
        public string Address { get; set; }
        public User() { }

        public User(int userID, string userName, string email, string password,
string contactNumber, string address)
        {

```



```

        UserID = userID;
        UserName = userName;
        Email = email;
        Password = password;
        ContactNumber = contactNumber;
        Address = address;
    }

    public override string ToString()
    {
        return $"User ID
: {UserID} \tUserName: {UserName} \tEmail: {Email} \tPassword: {Password} \tContactNumber
: {ContactNumber}" +
            $" \tAddress {Address}";
    }
}

```

Employee class:

```

namespace sample_project.model
{
    public class Employee
    {
        public int EmployeeID { get; set; }
        public string EmployeeName { get; set; }
        public string Email { get; set; }
        public string ContactNumber { get; set; }
        public string Role { get; set; }
        public double Salary { get; set; }

        public Employee() { }
        public Employee(int employeeID, string employeeName, string email, string
contactNumber, string role, double salary)
        {
            EmployeeID = employeeID;
            EmployeeName = employeeName;
            Email = email;
            ContactNumber = contactNumber;
            Role = role;
            Salary = salary;
        }
        public override string ToString()
        {
            return $"Employee ID: {EmployeeID}, Name: {EmployeeName}, Role:
{Role}, Salary: {Salary}";
        }
    }
}

```

Courier class:

```

namespace sample_project.model
{
    public class Courier
    {
        public int CourierID { get; set; }
    }
}

```

```

        public string SenderName { get; set; }
        public string SenderAddress { get; set; }
        public string ReceiverName { get; set; }
        public string ReceiverAddress { get; set; }
        public double Weight { get; set; }
        public string Status { get; set; }
        public string TrackingNumber { get; set; }
        public DateTime DeliveryDate { get; set; }
        public int LocationID { get; set; }
        public int EmployeeID { get; set; }
        public int ServiceID { get; set; }

        public Courier() { }
        public Courier(int courierID, string senderName, string senderAddress,
            weight, string receiverName, string receiverAddress, double
            deliveryDate, string status, string trackingNumber, DateTime
            int locationID, int employeeID, int serviceID)
        {
            CourierID = courierID;
            SenderName = senderName;
            SenderAddress = senderAddress;
            ReceiverName = receiverName;
            ReceiverAddress = receiverAddress;
            Weight = weight;
            Status = status;
            TrackingNumber = trackingNumber;
            DeliveryDate = deliveryDate;
            LocationID = locationID;
            EmployeeID = employeeID;
            ServiceID = serviceID;
        }

        public override string ToString()
        {
            return $"CourierID: {CourierID}, Sender: {SenderName}, Receiver:
{ReceiverName}, " +
                $"Weight: {Weight}kg, Status: {Status}, Tracking:
{TrackingNumber}, " +
                $"Delivery: {DeliveryDate.ToShortDateString()}, LocationID:
{LocationID}, " +
                $"EmployeeID: {EmployeeID}, ServiceID: {ServiceID}";
        }
    }
}

```

Location class:

```

namespace sample_project.model
{
    public class Location
    {
        private int locationID;
        private string locationName;
        private string address;

        public Location() { }
        public Location(int locationID, string locationName, string address)

```

```

    {
        this.locationID = locationID;
        this.locationName = locationName;
        this.address = address;
    }

    public int LocationID { get => locationID; set => locationID = value; }
    public string LocationName { get => locationName; set => locationName =
value; }
    public string Address { get => address; set => address = value; }

    public override string ToString()
    {
        return $"LocationID: {locationID}, LocationName: {locationName},
Address: {address}";
    }
}

```

Payment class:

```

namespace sample_project.model
{
    public class Payment
    {
        public int PaymentID { get; set; }
        public int CourierID { get; set; }
        public int LocationID { get; set; }
        public decimal Amount { get; set; }
        public DateTime PaymentDate { get; set; }
        public int EmployeeID { get; set; }
        public Payment () { }

        public Payment(int paymentID, int courierID, int locationID, decimal
amount, DateTime paymentDate, int employeeID)
        {
            PaymentID = paymentID;
            CourierID = courierID;
            LocationID = locationID;
            Amount = amount;
            PaymentDate = paymentDate;
            EmployeeID = employeeID;
        }

        public override string ToString()
        {
            return $"PaymentID: {PaymentID}, CourierID: {CourierID}, LocationID:
{LocationID}, " +
                $"Amount: {Amount}, PaymentDate:
{PaymentDate.ToShortDateString()}, " +
                $"EmployeeID: {EmployeeID}";
        }
    }
}

```

CourierCompany class:

```

namespace sample_project.model

```

```

{
    public class CourierCompany
    {
        public string CompanyName { get; set; }
        public List<Courier> CourierDetails { get; set; }
        public List<Employee> EmployeeDetails { get; set; }
        public List<Location> LocationDetails { get; set; }

        public CourierCompany()
        {
            CourierDetails = new List<Courier>();
            EmployeeDetails = new List<Employee>();
            LocationDetails = new List<Location>();
        }

        public CourierCompany(string companyName, List<Courier> courierDetails,
                                List<Employee> employeeDetails, List<Location>
locationDetails)
        {
            CompanyName = companyName;
            CourierDetails = courierDetails;
            EmployeeDetails = employeeDetails;
            LocationDetails = locationDetails;
        }
    }
}

```

Task 6: Service Provider Interface /Abstract class

Create 2 Interface /Abstract class ICourierUserService and ICourierAdminService

interface ICourierUserService { // Customer-related functions

placeOrder()

/ Place a new courier order. * @param courierObj Courier object created using values entered by users * @return The unique tracking number for the courier order . Use a static variable to generate unique tracking number. Initialize the static variable in Courier class with some random value. Increment the static variable each time in the constructor to generate next values.**

getOrderStatus();

/Get the status of a courier order. *@param trackingNumber The tracking number of the courier order. * @return The status of the courier order (e.g., yetToTransit, In Transit, Delivered). */**

cancelOrder()

/ Cancel a courier order. * @param trackingNumber The tracking number of the courier order to be canceled. * @return True if the order was successfully canceled, false otherwise.*/**

getAssignedOrder(); / Get a list of orders assigned to a specific courier staff member * @param courierStaffId The ID of the courier staff member. * @return A list of courier orders assigned to the staff member.*/**

```

namespace sample_project.dao
{
    public interface ICourierUserService
    {
        bool placeOrder(Courier newCourier);
        string getOrderStatus(string trackingNumber);
        bool cancelOrder(string trackingNumber);
        List<Courier> getAssignedOrder(int courierStaffId);
        List<Courier> RetrieveDeliveryHistory(string trackingNumber);
        decimal GenerateRevenueReport();
    }
}

```

```

namespace sample_project.dao
{
    public interface ICourierAdminService
    {
        int addCourierStaff(Employee obj);
    }
}

```

Task 7: Exception Handling

(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch ,finally,throw & throws keyword usage)

Define the following custom exceptions and throw them in methods whenever needed .
Handle all the exceptions in main method,

1. **TrackingNumberNotFoundException** :throw this exception when user try to withdraw amount or transfer amount to another acco
2. **InvalidEmployeeIdException** throw this exception when id entered for the employee not existing in the system

```

namespace sample_project.Exceptions
{
    public class TrackingNumberNotFoundException : Exception
    {
        public TrackingNumberNotFoundException() : base("Tracking number not found in the system.")
        {
        }

        public TrackingNumberNotFoundException(string message) : base(message)
        {
        }
    }
}

```

```

namespace sample_project.Exceptions
{
    public class InvalidEmployeeIdException: Exception
    {
        public InvalidEmployeeIdException() : base("Employee Not Found/Employee
Has not been allotted any assets") { }

        public InvalidEmployeeIdException(string message) : base(message) { }
    }
}

```

Task 8: Service implementation

1. Create CourierUserServiceImpl class which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompany.

This variable can be used to access the Object Arrays to access data relevant in method implementations.

2. Create CourierAdminService Impl class which inherits from CourierUserServiceImpl and implements ICourierAdminService interface.

3. Create CourierAdminServiceCollectionImpl class which inherits from CourierUserServiceCollectionImpl and implements ICourierAdminService interface.

```

namespace sample_project.dao
{
    public interface ICourierUserService
    {
        bool placeOrder(Courier newCourier);
        string getOrderStatus(string trackingNumber);
        bool cancelOrder(string trackingNumber);
        List<Courier> getAssignedOrder(int courierStaffId);

        List<Courier> RetrieveDeliveryHistory(string trackingNumber);
        decimal GenerateRevenueReport();
    }
}

namespace sample_project.dao
{
    public class CourierUserServiceImpl : ICourierUserService
    {
        private readonly string _connectionString;

        public CourierUserServiceImpl()
        {
            _connectionString = DBConnUtil.GetConnString();
        }

        public List<Courier> GetAllCouriers()
        {
            List<Courier> couriers = new List<Courier>();
            try
            {

```

```

        using (SqlConnection connection = new
SqlConnection(_connectionString))
        {
            string query = "SELECT * FROM Courier";
            SqlCommand command = new SqlCommand(query, connection);

            connection.Open();
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                couriers.Add(new Courier(
                    Convert.ToInt32(reader["CourierID"]),
                    reader["SenderName"].ToString(),
                    reader["SenderAddress"].ToString(),
                    reader["ReceiverName"].ToString(),
                    reader["ReceiverAddress"].ToString(),
                    Convert.ToDouble(reader["Weight"]),
                    reader["Status"].ToString(),
                    reader["TrackingNumber"].ToString(),
                    Convert.ToDateTime(reader["DeliveryDate"]),
                    Convert.ToInt32(reader["LocationID"]),
                    Convert.ToInt32(reader["EmployeeID"]),
                    Convert.ToInt32(reader["ServiceID"])
                ));
            }
        }
    }
    catch (SqlException ex)
    {
        Console.WriteLine($"Database error: {ex.Message}");
        throw;
    }
    return couriers;
}

```

```

public bool placeOrder(Courier courier)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(_connectionString))
        {
            string query = @"INSERT INTO Courier
(CourierID, SenderName, SenderAddress, ReceiverName,
ReceiverAddress, Weight, Status,
TrackingNumber, DeliveryDate,
LocationID, EmployeeID, ServiceID)
VALUES
(@CourierID, @SenderName, @SenderAddress,
@ReceiverName,
@ReceiverAddress, @Weight, @Status,
@TrackingNumber, @DeliveryDate,
@LocationID, @EmployeeID, @ServiceID)";

            SqlCommand command = new SqlCommand(query, connection);

            command.Parameters.AddWithValue("@CourierID", courier.CourierID);
            command.Parameters.AddWithValue("@SenderName", courier.SenderName);

```

```

        command.Parameters.AddWithValue("@SenderAddress",
courier.SenderAddress);
        command.Parameters.AddWithValue("@ReceiverName",
courier.ReceiverName);
        command.Parameters.AddWithValue("@ReceiverAddress",
courier.ReceiverAddress);
        command.Parameters.AddWithValue("@Weight", courier.Weight);
        command.Parameters.AddWithValue("@Status", courier.Status ??
"Processing");
        command.Parameters.AddWithValue("@TrackingNumber",
courier.TrackingNumber);
        command.Parameters.AddWithValue("@DeliveryDate",
courier.DeliveryDate);
        command.Parameters.AddWithValue("@LocationID", courier.LocationID);
        command.Parameters.AddWithValue("@EmployeeID", courier.EmployeeID);
        command.Parameters.AddWithValue("@ServiceID", courier.ServiceID);

        connection.Open();
        int rowsAffected = command.ExecuteNonQuery();

        return rowsAffected > 0;
    }
}
catch (SqlException ex)
{
    Console.WriteLine($"Database error: {ex.Message}");
    throw;
}
}

```

```

public string getOrderStatus(string trackingNumber)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(_connectionString))
        {
            string query = "SELECT Status FROM Courier WHERE TrackingNumber =
@TrackingNumber";
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@TrackingNumber", trackingNumber);

            connection.Open();
            object result = command.ExecuteScalar();

            if (result == null)
            {
                throw new TrackingNumberNotFoundException($"Tracking number
{trackingNumber} not found");
            }

            return result.ToString();
        }
    }
    catch (SqlException ex)
    {
        Console.WriteLine($"Database error: {ex.Message}");
        throw;
    }
}

```



```

public bool cancelOrder(string trackingNumber)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(_connectionString))
        {
            string query = "UPDATE Courier SET Status = 'Cancelled' WHERE
TrackingNumber = @TrackingNumber";
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@TrackingNumber", trackingNumber);

            connection.Open();
            int rowsAffected = command.ExecuteNonQuery();

            if (rowsAffected == 0)
            {
                throw new TrackingNumberNotFoundException($"Tracking number
{trackingNumber} not found");
            }

            return true;
        }
    }
    catch (SqlException ex)
    {
        Console.WriteLine($"Database error: {ex.Message}");
        throw;
    }
}

```

```

public List<Courier> getAssignedOrder(int employeeID)
{
    List<Courier> assignedOrders = new List<Courier>();

    try
    {
        using (SqlConnection connection = new SqlConnection(_connectionString))
        {
            string query = "SELECT * FROM Courier WHERE EmployeeID =
@EmployeeID";
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@EmployeeID", employeeID);

            connection.Open();
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                assignedOrders.Add(new Courier(
                    Convert.ToInt32(reader["CourierID"]),
                    reader["SenderName"].ToString(),
                    reader["SenderAddress"].ToString(),
                    reader["ReceiverName"].ToString(),
                    reader["ReceiverAddress"].ToString(),
                    Convert.ToDouble(reader["Weight"]),
                    reader["Status"].ToString(),
                    reader["TrackingNumber"].ToString(),
                    Convert.ToDateTime(reader["DeliveryDate"]),
                    Convert.ToInt32(reader["LocationID"]),
                    Convert.ToInt32(reader["EmployeeID"]),
                    Convert.ToInt32(reader["ServiceID"])
                ));
            }
        }
    }
}

```

```

        ));
    }
}
if (assignedOrders.Count == 0)
{
    throw new InvalidEmployeeIdException($"No assigned orders found.
Employee ID {employeeID} might be invalid.");
}
}
catch (SqlException ex)
{
    Console.WriteLine($"Database error: {ex.Message}");
    throw;
}

return assignedOrders;
}

public List<Courier> RetrieveDeliveryHistory(string trackingNumber)
{
    List<Courier> history = new List<Courier>();
    try
    {
        using (SqlConnection connection = new SqlConnection(_connectionString))
        {
            string query = "SELECT * FROM Courier WHERE TrackingNumber =
@TrackingNumber";
            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@TrackingNumber",
trackingNumber);
                connection.Open();
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        history.Add(new Courier
                        {
                            CourierID = Convert.ToInt32(reader["CourierID"]),
                            SenderName = reader["SenderName"].ToString(),
                            SenderAddress = reader["SenderAddress"].ToString(),
                            ReceiverName = reader["ReceiverName"].ToString(),
                            ReceiverAddress =
reader["ReceiverAddress"].ToString(),
                            Weight = Convert.ToDouble(reader["Weight"]),
                            Status = reader["Status"].ToString(),
                            TrackingNumber =
reader["TrackingNumber"].ToString(),
                            DeliveryDate =
Convert.ToDateTime(reader["DeliveryDate"]),
                            LocationID = Convert.ToInt32(reader["LocationID"]),
                            EmployeeID = Convert.ToInt32(reader["EmployeeID"]),
                            ServiceID = Convert.ToInt32(reader["ServiceID"])
                        });
                    }
                }
            }
        }
    }
    catch (Exception ex)
    {

```

```

        Console.WriteLine($"Error retrieving delivery history: {ex.Message}");
        return history;
    }
}

public decimal GenerateRevenueReport()
{
    try
    {
        using (SqlConnection connection = new
SqlConnection(_connectionString))
        {
            string query = "SELECT SUM(Amount) as TotalRevenue FROM
Payment";
            using (SqlCommand command = new SqlCommand(query,
connection))
            {
                connection.Open();
                object result = command.ExecuteScalar();
                return result != DBNull.Value ? Convert.ToDecimal(result)
: 0;
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error generating revenue report:
{ex.Message}");
        return 0;
    }
}
}
}

```

```

namespace sample_project.dao
{
    public interface ICourierAdminService
    {
        int addCourierStaff(Employee obj);
    }
}

```

```

namespace sample_project.dao
{
    public class CourierAdminServiceImpl : CourierUserServiceImpl,
ICourierAdminService
    {
        public int addCourierStaff(Employee employee)
        {
            using (SqlConnection connection = new
SqlConnection(DBConnUtil.GetConnString()))
            {
                using (SqlCommand cmd = new SqlCommand())
                {
                    cmd.CommandText = @"INSERT INTO Employees
Salary)
                                (EmployeeName, Email, ContactNumber, Role,
                                VALUES

```

```

        (@EmployeeName, @Email, @ContactNumber,
        @Role, @Salary);

        SELECT SCOPE_IDENTITY();"
        cmd.Parameters.AddWithValue("@EmployeeName",
employee.EmployeeName);
        cmd.Parameters.AddWithValue("@Email", employee.Email);
        cmd.Parameters.AddWithValue("@ContactNumber",
employee.ContactNumber);
        cmd.Parameters.AddWithValue("@Role", employee.Role);
        cmd.Parameters.AddWithValue("@Salary", employee.Salary);
        cmd.Connection = connection;
        connection.Open();
        int newId = Convert.ToInt32(cmd.ExecuteScalar());
        return newId;
    }
}
}
}
}
}

```

Task 9: Database Interaction

Connect your application to the SQL database for the Courier Management System

1. Write code to establish a connection to your SQL database.

Create a class DBConnection in a package connectionutil with a static variable connection of Type Connection and a static method getConnection() which returns connection. Connection properties supplied in the connection string should be read from a property file.

2. Create a Service class CourierServiceDb in dao with a static variable named connection of type Connection which can be assigned in the constructor by invoking the method in DBConnection Class.

3. Include methods to insert, update, and retrieve data from the database (e.g., inserting a new order, updating courier status).

4. Implement a feature to retrieve and display the delivery history of a specific parcel by querying the database. 1. Generate and display reports using data retrieved from the database (e.g., shipment status report, revenue report).

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration;

namespace sample_project.util
{
    public static class DBConnUtil
    {
        public static string GetConnString()
        {

```

```

        return
        ConfigurationManager.ConnectionStrings["CourierDB"].ConnectionString;
    }
}

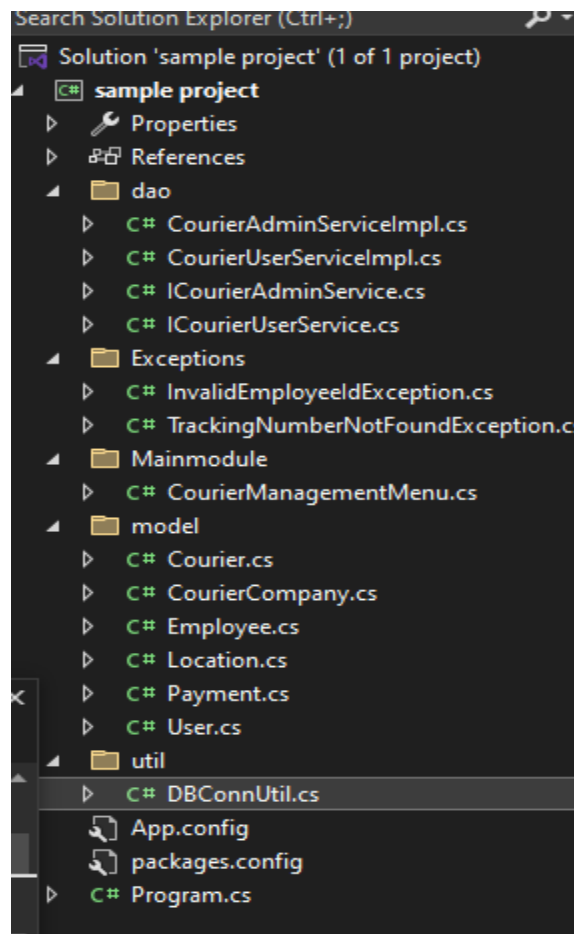
```

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <connectionStrings>
        <add name="PayrollDB"
            connectionString="Server=DESKTOP-
JHVN0CR\SQLEXPRESS;Database=Payroll;Integrated Security=True;Encrypt=False;"
            providerName="System.Data.SqlClient"/>
    </connectionStrings>
</configuration>

```

FILE STRUCTURE



```

using sample_project.dao;
using sample_project.MainModule;

namespace sample_project

```

```

{
    class Program
    {
        static void Main(string[] args)
        {
            // Initialize services
            ICourierUserService courierService = new CourierUserServiceImpl();
            ICourierAdminService adminService = new CourierAdminServiceImpl();

            // Create and display menu
            CourierManagementMenu menu = new
CourierManagementMenu(courierService, adminService);
            menu.DisplayMenu();
        }
    }
}

```

MAIN

```

namespace sample_project.MainModule
{
    public class CourierManagementMenu
    {
        private readonly ICourierUserService _courierService;
        private readonly ICourierAdminService _adminService;

        public CourierManagementMenu(ICourierUserService courierService,
ICourierAdminService adminService)
        {
            _courierService = courierService;
            _adminService = adminService;
        }

        public void DisplayMenu()
        {
            while (true)
            {
                Console.WriteLine("\nCourier Service Menu:");
                Console.WriteLine("1. Place an order");
                Console.WriteLine("2. Get order status");
                Console.WriteLine("3. Cancel an order");
                Console.WriteLine("4. Get assigned orders");
                Console.WriteLine("5. View delivery history");
                Console.WriteLine("7. Generate revenue report");
                Console.WriteLine("8. Add courier staff (Admin)");
                Console.WriteLine("9. Exit");

                Console.Write("Enter your choice: ");
                string choice = Console.ReadLine();

                try
                {
                    switch (choice)
                    {
                        case "1":
                            PlaceOrder();
                            break;
                        case "2":
                            GetOrderStatus();
                            break;
                        case "3":

```

```

        CancelOrder();
        break;
    case "4":
        GetAssignedOrders();
        break;
    case "5":
        ViewDeliveryHistory();
        break;
    case "6":
        GenerateRevenueReport();
        break;
    case "7":
        AddCourierStaff();
        break;
    case "8":
        Console.WriteLine("Exiting program...");
        return;
    default:
        Console.WriteLine("Invalid choice. Please enter a
number between 1 and 9.");
        break;
    }
}
catch (Exception ex)
{
    Console.WriteLine($"An error occurred: {ex.Message}");
}
}
}

```

```

private void PlaceOrder()
{
    Console.WriteLine("\n--- Place New Order ---");

    Courier courier = new Courier();

    Console.Write("Enter Courier ID: ");
    courier.CourierID = int.Parse(Console.ReadLine());

    Console.Write("Enter Sender Name: ");
    courier.SenderName = Console.ReadLine();

    Console.Write("Enter Sender Address: ");
    courier.SenderAddress = Console.ReadLine();

    Console.Write("Enter Receiver Name: ");
    courier.ReceiverName = Console.ReadLine();

    Console.Write("Enter Receiver Address: ");
    courier.ReceiverAddress = Console.ReadLine();

    Console.Write("Enter Weight (kg): ");
    courier.Weight = double.Parse(Console.ReadLine());

    Console.Write("Enter Tracking Number: ");
    courier.TrackingNumber = Console.ReadLine();

    Console.Write("Enter Delivery Date (YYYY-MM-DD): ");
    courier.DeliveryDate = DateTime.Parse(Console.ReadLine());

    Console.Write("Enter Location ID: ");
    courier.LocationID = int.Parse(Console.ReadLine());
}

```

```

        Console.WriteLine("Enter Employee ID: ");
        courier.EmployeeID = int.Parse(Console.ReadLine());

        Console.WriteLine("Enter Service ID: ");
        courier.ServiceID = int.Parse(Console.ReadLine());

        bool success = _courierService.placeOrder(courier);
        Console.WriteLine(success ? "Order placed successfully!" : "Failed to
place order.");
    }

    private void GetOrderStatus()
    {
        Console.WriteLine("\n--- Check Order Status ---");
        Console.WriteLine("Enter Tracking Number: ");
        string trackingNumber = Console.ReadLine();

        string status = _courierService.getOrderStatus(trackingNumber);
        Console.WriteLine($"Current Status: {status}");
    }

    private void CancelOrder()
    {
        Console.WriteLine("\n--- Cancel Order ---");
        Console.WriteLine("Enter Tracking Number: ");
        string trackingNumber = Console.ReadLine();

        bool success = _courierService.cancelOrder(trackingNumber);
        Console.WriteLine(success ? "Order cancelled successfully!" : "Failed
to cancel order.");
    }

    private void GetAssignedOrders()
    {
        Console.WriteLine("\n--- View Assigned Orders ---");
        Console.WriteLine("Enter Employee ID: ");
        int employeeId = int.Parse(Console.ReadLine());

        var orders = _courierService.getAssignedOrder(employeeId);

        if (orders.Count == 0)
        {
            Console.WriteLine("No orders assigned to this employee.");
            return;
        }

        Console.WriteLine($"Assigned Orders ({orders.Count}):");
        foreach (var order in orders)
        {
            Console.WriteLine($"- {order.TrackingNumber}: {order.Status}
(Due: {order.DeliveryDate.ToShortDateString()})");
        }
    }

    private void ViewDeliveryHistory()
    {
        Console.WriteLine("\n--- View Delivery History ---");
    }

```



```

        Console.WriteLine("Enter Tracking Number: ");
        string trackingNumber = Console.ReadLine();

        var history =
_courierService.RetrieveDeliveryHistory(trackingNumber);

        if (history.Count == 0)
        {
            Console.WriteLine("No delivery history found for this tracking
number.");
            return;
        }

        Console.WriteLine($"
Delivery History ({history.Count} records):");
        foreach (var record in history)
        {
            Console.WriteLine($"- ID: {record.CourierID}, Status:
{record.Status}, Delivery Date: {record.DeliveryDate.ToShortDateString()}");
            Console.WriteLine($"  From: {record.SenderName}
({record.SenderAddress})");
            Console.WriteLine($"  To: {record.ReceiverName}
({record.ReceiverAddress})");
            Console.WriteLine($"  Weight: {record.Weight}kg, Service ID:
{record.ServiceID}
");
        }
    }

    private void GenerateRevenueReport()
    {
        Console.WriteLine("
--- Revenue Report ---");

        decimal totalRevenue = _courierService.GenerateRevenueReport();
        Console.WriteLine($"Total Revenue: {totalRevenue:C}");
    }

    private void AddCourierStaff()
    {
        Console.WriteLine("
--- Add New Courier Staff ---");
        Employee employee = new Employee();

        Console.WriteLine("Enter Staff Name: ");
        employee.EmployeeName = Console.ReadLine();

        Console.WriteLine("Enter Email: ");
        employee.Email = Console.ReadLine();

        Console.WriteLine("Enter Contact Number: ");
        employee.ContactNumber = Console.ReadLine();

        Console.WriteLine("Enter Role: ");
        employee.Role = Console.ReadLine();

        Console.WriteLine("Enter Salary: ");
        employee.Salary = double.Parse(Console.ReadLine());

        int staffId = _adminService.addCourierStaff(employee);
        Console.WriteLine($"New staff member added with ID: {staffId}");
    }
}

```

```

Enter your choice: 1

--- Place New Order ---
Enter Courier ID: 14
Enter Sender Name: Amal davis
Enter Sender Address: jagan street,chennai
Enter Receiver Name: prakash
Enter Receiver Address: jawahar street,chennai
Enter Weight (kg): 9.8
Enter Tracking Number: TN779931
Enter Delivery Date (YYYY-MM-DD): 2024-10-23
Enter Location ID: 1
Enter Employee ID: 2
Enter Service ID: 1
Order placed successfully!

```

```
SELECT * FROM Courier WHERE CourierID = 14;
```

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
14	Amal davis	jagan street,chennai	prakash	jawahar street,chennai	9.80	Processing	TN779931	2024-10-23	1	2	1

```

Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. View delivery history
7. Generate revenue report
8. Add courier staff (Admin)
9. Exit
Enter your choice: 2

```

```

--- Check Order Status ---
Enter Tracking Number: TN123456
Current Status: In Transit

```

```
SELECT Status
FROM Courier
WHERE TrackingNumber = 'TN123456';
```

Status
In Transit

```

Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. View delivery history
7. Generate revenue report
8. Add courier staff (Admin)
9. Exit
Enter your choice: 3

--- Cancel Order ---
Enter Tracking Number: TN678901
Order cancelled successfully!

```

```
SELECT Status FROM Courier WHERE TrackingNumber = 'TN678901';
```

%	
Results	Messages
Status	
Cancelled	

```

Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. View delivery history
7. Generate revenue report
8. Add courier staff (Admin)
9. Exit
Enter your choice: 4

--- View Assigned Orders ---
Enter Employee ID: 2

Assigned Orders (9):
- TN123456: In Transit (Due: 01-03-2025)
- TN789012: Delivered (Due: 02-03-2025)
- TN234567: In Transit (Due: 04-03-2025)
- TN456789: In
Transit (Due: 06-03-2025)
- TN567890: In Transit (Due: 07-03-2025)
- Tn5876: Processing (Due: 15-12-2023)
- TN87632: Processing (Due: 27-12-2025)
- TN877665: Processing (Due: 17-12-2024)
- TN779931: Processing (Due: 23-10-2024)

```

SELECT * FROM Courier WHERE EmployeeID = 2												
10 %												
Results Messages												
	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2025-03-01	1	2	1
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2025-03-02	1	2	2
3	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2025-03-04	1	2	2
4	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2025-03-06	3	2	2
5	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2025-03-07	3	2	1
6	10	kathi	123 main	krish	468 main	4.60	Processing	Tn5876	2023-12-15	1	2	1
7	11	akshara	987 road	gokul	567 main road	4.60	Processing	TN87632	2025-12-27	1	2	1
8	13	Smith	nehru street	raina	nethaji street	5.50	Processing	TN877665	2024-12-17	1	2	1
9	14	Amal davis	jagan street, chennai	prakash	jawahar street, chennai	9.80	Processing	TN779931	2024-10-23	1	2	1

```

1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. View delivery history
6. Generate revenue report
7. Add courier staff (Admin)
8. Exit
Enter your choice: 5

--- View Delivery History ---
Enter Tracking Number: TN789012

Delivery History (1 records):
- ID: 2, Status: Delivered, Delivery Date: 02-03-2025
  From: Priya Sharma (34 Kaveri Street, Bangalore)
  To: Amit Patel (56 Krishna Lane, Hyderabad)
  Weight: 1.8kg, Service ID: 2

```

SELECT												
CourierID, SenderName, SenderAddress,												
ReceiverName, ReceiverAddress,												
Weight, Status, TrackingNumber,												
DeliveryDate, ServiceID												
FROM												
Courier												
WHERE												
TrackingNumber = 'TN789012';												
6												
Results Messages												
	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	ServiceID		
	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2025-03-02	2		

```

Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. View delivery history
6. Generate revenue report
7. Add courier staff (Admin)
8. Exit
Enter your choice: 6

--- Revenue Report ---
Total Revenue: ₹ 44,601.50

```

SELECT SUM(Amount) as TotalRevenue FROM Payment;

100 %

Results Messages

	TotalRevenue
1	44601.50

```

--- Add New Courier Staff ---
Enter EmpID: 8
Enter Staff Name: Maha
Enter Email: maha@gmail.com
Enter Contact Number: 3445456657
Enter Role: Clerk
Enter Salary: 34568

```

Results Messages

	EmployeeId	Name	Email	ContactNumber	Role	Salary
1	1	amav	amav@email.com	1112223333	Manager	50000.00
2	2	kaushik	kaushik@email.com	4445556666	Delivery Person	30000.00
3	3	karthick	karthick@email.com	5556667777	Delivery Person	30000.00
4	4	Johncena	john.cena@email.com	1234567890	Manager	60000.00
5	5	Johnson	johnson@email.com	9876543210	Clerk	45000.00
6	6	Akila	akila@gmail.com	7876676556	Delivery Person	32000.00
7	7	Manoj	manoj@gmail.com	3454566776	Clerk	47000.00
8	8	Maha	maha@gmail.com	3445456657	Clerk	34568.00