ASSIGNMENT 4

NAME : SRINATH N S S

TASK 1 : Control Flow Statements

1.

using System;

class Program

{

static void Main(string[] args)

{

CheckOrderStatus("Delivered");

CheckOrderStatus("Cancelled");

CheckOrderStatus("Shipped"); // Invalid status

}

static void CheckOrderStatus(string status)

{

if (status == "Delivered")

{

Console.WriteLine("The order is delivered.");

}

else if (status == "Processing")

{

Console.WriteLine("The order is still being processed.");

}

else if (status == "Cancelled")

{

Console.WriteLine("The order has been cancelled.");

}

else

{

Console.WriteLine("Invalid order status.");

}

}

}

OUTPUT :

The order is delivered.

The order has been cancelled.

Invalid order status.

2.

class Program

{

static void Main(string[] args)

{

CP(2.5);

CP(10);

CP(50);

}

static void CP(double weight)

{

string category;

switch (weight)

{

case double n when (n <= 5):

category = "Light";

break;

case double n when (n <= 20):

category = "Medium";

break;

default:

category = "Heavy";

break;

}

Console.WriteLine("Parcel weight {0} kg is categorized as {1}",weight,category);

Console.ReadLine();

}

}

OUTPUT :

Parcel weight 2.5 kg is categorized as Light

Parcel weight 10 kg is categorized as Medium

Parcel weight 50 kg is categorized as Heavy

3.

class Program

{

static void Main(string[] args)

{

string correctUsername = "user";

string correctPassword = "pass";

Console.Write("Enter username: ");

string username = Console.ReadLine();

Console.Write("Enter password: ");

string password = Console.ReadLine();

if (username == correctUsername && password == correctPassword)

{

Console.WriteLine("Authentication successful. Welcome, " + username + "!");

}

else

{

Console.WriteLine("Authentication failed. Invalid username or password.");

}

}

}

OUTPUT :

Enter username: user

Enter password: pass

Authentication successful. Welcome, user!

4.

class CProgram

{

static void Main(string[] args)

{

string[] couriers = { "Courier1", "Courier2", "Courier3" };

int[] loadCapacities = { 50, 40, 60 };

int[] shipments = { 20, 30, 40, 70 };

for (int i = 0; i < shipments.Length; i++)

{

string assignedCourier = AssignCourier(couriers, loadCapacities, shipments[i]);

Console.WriteLine($"Shipment with weight {shipments[i]} is assigned to {assignedCourier}.");

}

}

static string AssignCourier(string[] couriers, int[] loadCapacities, int shipmentWeight)

{

string assignedCourier = "No courier available";

for (int i = 0; i < couriers.Length; i++)

{

if (loadCapacities[i] >= shipmentWeight)

{

assignedCourier = couriers[i];

break;

}

}

return assignedCourier;

}

}

OUTPUT :

Shipment with weight 20 is assigned to Courier1.

Shipment with weight 30 is assigned to Courier2.

TASK 2 : LOOPS AND ITERATION

5.

class Program

{

static void Main(string[] args)

{

string[] orders = { "Order1", "Order2", "Order3",};

string customerName = "Robert";

Console.WriteLine("Orders for customer {0}:",customerName);

for (int i = 0; i < orders.Length; i++)

{

Console.WriteLine($"{i + 1}. {orders[i]}");

}

Console.ReadLine();

}

}

Orders for customer Robert:

1. Order1

2. Order2

3. Order3

6.

class Program

{

static void Main(string[] args)

{

string startingLocation = "Warehouse";

string destination = "Destination";

string currentLocation = startingLocation;

Console.WriteLine($"Courier is currently at {currentLocation}.");

while (currentLocation != destination)

{

Console.WriteLine("Courier is en route to the destination...");

currentLocation = destination;

Console.WriteLine("Courier has arrived at {0}.", currentLocation);

}

Console.WriteLine("Courier has reached the destination.");

}

}

OUTPUT :

Courier is currently at Warehouse.

Courier is en route to the destination...

Courier has arrived at Destination.

Courier has reached the destination.

TASK 3 : ARRAYS AND DATA STRUCTURES

7.

class Program

{

static void Main(string[] args)

{

string[] trackingHistory = new string[10];

string[] locationUpdates = { "Warehouse", "In transit", "Sorting center", "Out for delivery", "Delivered" };

for (int i = 0; i < locationUpdates.Length && i < 10; i++)

{

trackingHistory[i] = locationUpdates[i];

}

Console.WriteLine("Tracking History:");

for (int i = 0; i < trackingHistory.Length && trackingHistory[i] != null; i++)

{

Console.WriteLine($"Update {i + 1}: {trackingHistory[i]}");

}

Console.ReadLine();

}

}

OUTPUT :

Tracking History:

Update 1: Warehouse

Update 2: In transit

Update 3: Sorting center

Update 4: Out for delivery

Update 5: Delivered

8.

class Program

{

static void Main(string[] args)

{

string[] couriers = { "Courier1", "Courier2"};

int[] distances = { 10, 15,};

int orderLocation = 12;

string nearestCourier = FindNearestCourier(couriers, distances, orderLocation);

Console.WriteLine($"The nearest available courier for the new order is: {nearestCourier}");

}

static string FindNearestCourier(string[] couriers, int[] distances, int orderLocation)

{

string nearestCourier = "No courier available";

int minDistance = int.MaxValue;

for (int i = 0; i < couriers.Length; i++)

{

if (distances[i] < minDistance)

{

minDistance = distances[i];

nearestCourier = couriers[i];

}

}

return nearestCourier;

}

}

OUTPUT :

The nearest available courier for the new order is: Courier1

TASK 4 : STRINGS , 2D ARRAYS ,HASHMAP

9.

class Program

{

static void Main(string[] args)

{

string[,] parcelData = {

{ "S987", "In transit" },

{ "S654", "Out for delivery" },

{ "S123", "Delivered" }

};

Console.Write("Enter Parcel No: ");

string trackingNumber = Console.ReadLine();

bool found = false;

for (int i = 0; i < parcelData.GetLength(0); i++)

{

if (parcelData[i, 0] == trackingNumber)

{

Console.WriteLine($"Status for tracking number {trackingNumber}: {parcelData[i, 1]}");

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Tracking number not found.");

}

}

}

OUTPUT :

Enter Parcel No: S123

Status for tracking number S123: Delivered

10.

class Program

{

static void Main(string[] args)

{

Console.WriteLine(validateCustomerInfo("Ram Kumar", "name")); // true

Console.WriteLine(validateCustomerInfo("Rakk!", "name")); // false

Console.WriteLine(validateCustomerInfo("704-226-5123", "phone")); // true

Console.ReadLine();

}

static bool validateCustomerInfo(string data, string detail)

{

Regex nameRegex = new Regex("^[A-Z][a-zA-Z ]\*$");

Regex phoneRegex = new Regex(@"^\d{3}-?\d{3}-?\d{4}$");

switch (detail.ToLower())

{

case "name":

return nameRegex.IsMatch(data);

case "phone":

return phoneRegex.IsMatch(data);

default:

Console.WriteLine("Invalid detail specified.");

return false;

}

}

}

OUTPUT :

True

False

True

11.

class Program

{

static void Main(string[] args)

{

string street = "31 church street";

string city = "thenni";

string state = "th";

string zipCode = "641012";

string formattedAddress = FormatAddress(street, city, state, zipCode);

Console.WriteLine(formattedAddress);

Console.ReadLine();

}

static string FormatAddress(string street, string city, string state, string zipCode)

{

return $"{char.ToUpper(street[0])}{street.Substring(1)}, {char.ToUpper(city[0])}{city.Substring(1)}, {state} {zipCode}";

}

}

OUTPUT :

31 church street, Thenni, th 641012

12.

class Program

{

static void Main(string[] args)

{

string customerName = "Rajesh Kumar";

int orderNumber = 987654;

string deliveryAddress = "456/7, MG Road, Bangalore, Karnataka, 560001";

DateTime expectedDeliveryDate = DateTime.Today.AddDays(5);

string confirmationEmail = GenerateConfirmationEmail(customerName, orderNumber, deliveryAddress, expectedDeliveryDate);

Console.WriteLine(confirmationEmail);

Console.ReadLine();

}

static string GenerateConfirmationEmail(string customerName, int orderNumber, string deliveryAddress, DateTime expectedDeliveryDate)

{

string emailMessage = $"Dear {customerName},\n\n";

emailMessage += "Happy Morning , Thank you for your order! Your order details are as follows:\n";

emailMessage += $"Order Number: {orderNumber}\n";

emailMessage += $"Delivery Address: {deliveryAddress}\n";

emailMessage += $"Expected Delivery Date: {expectedDeliveryDate:dd/MM/yyyy}\n\n";

emailMessage += "Please feel free to contact us.\n\n";

emailMessage += "Best regards,\nThe Marketting Team";

return emailMessage;

}

}

OUTPUT :

Dear Rajesh Kumar,

Happy Morning , Thank you for your order! Your order details are as follows:

Order Number: 987654

Delivery Address: 456/7, MG Road, Bangalore, Karnataka, 560001

Expected Delivery Date: 11/05/2024

Please feel free to contact us.

Best regards,

The Marketting Team

13.

using System;

class Program

{

static void Main(string[] args)

{

string sourceAddress = "123, MG Road, Bangalore, Karnataka";

string destinationAddress = "456, Jayanagar, Bangalore, Karnataka";

double distance = 300;

double parcelWeight = 5;

double shippingCost = CalculateShippingCost(distance, parcelWeight);

Console.WriteLine($"Shipping cost from {sourceAddress} to {destinationAddress}: ₹{shippingCost:F2}");

Console.ReadLine();

}

static double CalculateShippingCost(double distance, double parcelWeight)

{

const double ratePerKilometer = 2.5;

const double ratePerKilogram = 20;

double distanceCost = distance \* ratePerKilometer;

double weightCost = parcelWeight \* ratePerKilogram;

double totalShippingCost = distanceCost + weightCost;

return totalShippingCost;

}

}

OUTPUT :

Shipping cost from 123, MG Road, Bangalore, Karnataka to 456, Jayanagar, Bangalore, Karnataka: ?850.00

14.

class Program

{

static void Main(string[] args)

{

string password = GeneratePassword(12);

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

Console.ReadLine();

}

static string GeneratePassword(int length)

{

const string validChars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&\*()\_+-=[]{}|;:,.<>?";

Random random = new Random();

StringBuilder passwordBuilder = new StringBuilder();

for (int i = 0; i < length; i++)

{

passwordBuilder.Append(validChars[random.Next(validChars.Length)]);

}

return passwordBuilder.ToString();

}

}

OUTPUT :

Generated Password: w.)j)lEi.PQz

15.

class Program

{

static void Main(string[] args)

{

List<string> addresses = new List<string>

{

"123, MG Road, Bangalore, Karnataka",

"124, MG Road, Bangalore, Karnataka",

"456, Gandhipuram, Coimbatore, Tamil Nadu",

"457, Gandhipuram, Coimbatore, Tamil Nadu",

"789, Bharathi Street, Madurai, Tamil Nadu"

};

Dictionary<string, List<string>> similarAddresses = FindSimilarAddresses(addresses);

foreach (var kvp in similarAddresses)

{

Console.WriteLine($"Similar addresses for {kvp.Key}:");

foreach (var address in kvp.Value)

{

Console.WriteLine(address);

}

Console.WriteLine();

}

}

static Dictionary<string, List<string>> FindSimilarAddresses(List<string> addresses)

{

Dictionary<string, List<string>> similarAddresses = new Dictionary<string, List<string>>();

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

{

string address1 = addresses[i];

List<string> similar = new List<string>();

for (int j = i + 1; j < addresses.Count; j++)

{

string address2 = addresses[j];

if (AreAddressesSimilar(address1, address2))

{

similar.Add(address2);

}

}

if (similar.Count > 0)

{

similarAddresses[address1] = similar;

}

}

return similarAddresses;

}

static bool AreAddressesSimilar(string address1, string address2)

{

address1 = address1.ToLower();

address2 = address2.ToLower();

return address1.Contains(address2) || address2.Contains(address1);

}

}

OUTPUT :

Similar addresses for 123, MG Road, Bangalore, Karnataka:

124, MG Road, Bangalore, Karnataka

Similar addresses for 456, Gandhipuram, Coimbatore, Tamil Nadu:

457, Gandhipuram, Coimbatore, Tamil Nadu

TASK 5 : Object Oriented Programming

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.entity

{

public class Courier

{

private int courierID;

private string senderName;

private string senderAddress;

private string receiverName;

private string receiverAddress;

private double weight;

private string status;

private string trackingNumber;

private DateTime deliveryDate;

private int userId;

public Courier()

{

}

public Courier(int courierID, string senderName, string senderAddress, string receiverName, string receiverAddress,

double weight, string status, string trackingNumber, DateTime deliveryDate, int userId)

{

this.courierID = courierID;

this.senderName = senderName;

this.senderAddress = senderAddress;

this.receiverName = receiverName;

this.receiverAddress = receiverAddress;

this.weight = weight;

this.status = status;

this.trackingNumber = trackingNumber;

this.deliveryDate = deliveryDate;

this.userId = userId;

}

public int CourierID { get => courierID; set => courierID = value; }

public string SenderName { get => senderName; set => senderName = value; }

public string SenderAddress { get => senderAddress; set => senderAddress = value; }

public string ReceiverName { get => receiverName; set => receiverName = value; }

public string ReceiverAddress { get => receiverAddress; set => receiverAddress = value; }

public double Weight { get => weight; set => weight = value; }

public string Status { get => status; set => status = value; }

public string TrackingNumber { get => trackingNumber; set => trackingNumber = value; }

public DateTime DeliveryDate { get => deliveryDate; set => deliveryDate = value; }

public int UserId { get => userId; set => userId = value; }

public override string ToString()

{

return $"Courier ID: {CourierID}, Sender Name: {SenderName}, Receiver Name: {ReceiverName}, Weight: {Weight}, Status: {Status}, Tracking Number: {TrackingNumber}, Delivery Date: {DeliveryDate}, User ID: {UserId}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.entity

{

public class CourierCompany

{

private string companyName;

private List<Courier> courierDetails;

private List<Employee> employeeDetails;

private List<Location> locationDetails;

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)

{

this.companyName = companyName;

this.courierDetails = courierDetails;

this.employeeDetails = employeeDetails;

this.locationDetails = locationDetails;

}

public string CompanyName { get => companyName; set => companyName = value; }

public List<Courier> CourierDetails { get => courierDetails; set => courierDetails = value; }

public List<Employee> EmployeeDetails { get => employeeDetails; set => employeeDetails = value; }

public List<Location> LocationDetails { get => locationDetails; set => locationDetails = value; }

public override string ToString()

{

return $"CompanyName: {CompanyName}, CourierDetails Count: {CourierDetails.Count}, EmployeeDetails Count: {EmployeeDetails.Count}, LocationDetails Count: {LocationDetails.Count}";

}

public CourierCompanyCollection CourierCompanyCollection { get; set; }

public CourierCompany(string companyName)

{

CourierCompanyCollection = new CourierCompanyCollection(companyName);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.entity

{

public class CourierCompanyCollection

{

public string CompanyName { get; set; }

public List<Courier> CourierDetails { get; set; }

public List<Employee> EmployeeDetails { get; set; }

public List<Location> LocationDetails { get; set; }

public CourierCompanyCollection(string companyName)

{

CompanyName = companyName;

CourierDetails = new List<Courier>();

EmployeeDetails = new List<Employee>();

LocationDetails = new List<Location>();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.entity

{

public class Employee

{

private int employeeID;

private string employeeName;

private string email;

private string contactNumber;

private string role;

private double salary;

public Employee()

{

}

public Employee(int employeeID, string employeeName, string email, string contactNumber, string role, double salary)

{

this.employeeID = employeeID;

this.employeeName = employeeName;

this.email = email;

this.contactNumber = contactNumber;

this.role = role;

this.salary = salary;

}

public int EmployeeID { get => employeeID; set => employeeID = value; }

public string EmployeeName { get => employeeName; set => employeeName = value; }

public string Email { get => email; set => email = value; }

public string ContactNumber { get => contactNumber; set => contactNumber = value; }

public string Role { get => role; set => role = value; }

public double Salary { get => salary; set => salary = value; }

public override string ToString()

{

return $"EmployeeID: {EmployeeID}, EmployeeName: {EmployeeName}, Email: {Email}, ContactNumber: {ContactNumber}, Role: {Role}, Salary: {Salary}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.entity

{

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}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.entity

{

public class Payment

{

private long paymentID;

private long courierID;

private double amount;

private DateTime paymentDate;

public Payment()

{

}

public Payment(long paymentID, long courierID, double amount, DateTime paymentDate)

{

this.paymentID = paymentID;

this.courierID = courierID;

this.amount = amount;

this.paymentDate = paymentDate;

}

public long PaymentID { get => paymentID; set => paymentID = value; }

public long CourierID { get => courierID; set => courierID = value; }

public double Amount { get => amount; set => amount = value; }

public DateTime PaymentDate { get => paymentDate; set => paymentDate = value; }

public override string ToString()

{

return $"PaymentID: {PaymentID}, CourierID: {CourierID}, Amount: {Amount}, PaymentDate: {PaymentDate}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.entity

{

public class User

{

private int userID;

private string userName;

private string email;

private string password;

private string contactNumber;

private string address;

public User()

{

}

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

{

this.userID = userID;

this.userName = userName;

this.email = email;

this.password = password;

this.contactNumber = contactNumber;

this.address = address;

}

public int UserID { get => userID; set => userID = value; }

public string UserName { get => userName; set => userName = value; }

public string Email { get => email; set => email = value; }

public string Password { get => password; set => password = value; }

public string ContactNumber { get => contactNumber; set => contactNumber = value; }

public string Address { get => address; set => address = value; }

public override string ToString()

{

return $"User ID: {UserID}, Name: {UserName}, Email: {Email}, Password: {Password}, Contact Number: {ContactNumber}, Address: {Address}";

}

}

}

public void AddCourier(Courier courier)

{

courierDetails.Add(courier);

}

public void AddEmployee(Employee employee)

{

employeeDetails.Add(employee);

}

public void AddLocation(Location location)

{

locationDetails.Add(location);

}

public override string ToString()

{

string courierInfo = "Courier Details:\n";

foreach (var courier in courierDetails)

{

courierInfo += $"{courier}\n";

}

string employeeInfo = "Employee Details:\n";

foreach (var employee in employeeDetails)

{

employeeInfo += $"{employee}\n";

}

string locationInfo = "Location Details:\n";

foreach (var location in locationDetails)

{

locationInfo += $"{location}\n";

}

return $"Company Name: {companyName}\n{courierInfo}{employeeInfo}{locationInfo}";

}

}

TASK 6 : INTERFACE/ABSTRACT CLASS

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.services

{

public interface ICourierAdminService

{

int AddCourierStaff(string name, string contactNumber);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Assignment4\_DB.entity;

namespace Assignment4\_DB.services

{

public interface ICourierUserService

{

string PlaceOrder(Courier courierObj);

string GetOrderStatus(string trackingNumber);

bool CancelOrder(string trackingNumber);

List<Courier> GetAssignedOrders(int courierStaffId);

}

}

TASK 7 - //EXCEPTION HANDLING

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.exception

{

public class InvalidEmployeeIdException : Exception

{

public InvalidEmployeeIdException(string message) : base(message)

{

LogErrorMessage(message);

}

private void LogErrorMessage(string errorMessage)

{

Console.WriteLine($"Error occurred: {errorMessage}");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment4\_DB.exception

{

public class TrackingNumberNotFoundException : Exception

{

public TrackingNumberNotFoundException(string message) : base(message)

{

LogErrorMessage(message);

}

private void LogErrorMessage(string errorMessage)

{

Console.WriteLine($"Error occurred: {errorMessage}");

}

}

}

TASK 8 - COLLECTIONS AND SERVICE IMPLEMENTATION

using System;

using System.Collections.Generic;

using Entities;

namespace Services

{

public class CourierUserServiceCollectionImpl : ICourierUserService

{

private CourierCompanyCollection companyObj;

public CourierUserServiceCollectionImpl()

{

companyObj = new CourierCompanyCollection();

}

public string PlaceOrder(Courier courierObj)

{

Random random = new Random();

string trackingNumber = random.Next(100000, 999999).ToString();

courierObj.TrackingNumber = trackingNumber;

companyObj.CourierDetails.Add(courierObj);

return trackingNumber;

}

public string GetOrderStatus(string trackingNumber)

{

foreach (var courier in companyObj.CourierDetails)

{

if (courier.TrackingNumber == trackingNumber)

{

return courier.Status;

}

}

throw new TrackingNumberNotFoundException($"Tracking number '{trackingNumber}' not found.");

}

public bool CancelOrder(string trackingNumber)

{

foreach (var courier in companyObj.CourierDetails)

{

if (courier.TrackingNumber == trackingNumber)

{

companyObj.CourierDetails.Remove(courier);

return true; // Successfully canceled the order

}

}

throw new TrackingNumberNotFoundException($"Tracking number '{trackingNumber}' not found.");

}

public List<Courier> GetAssignedOrders(int courierStaffId)

{

List<Courier> assignedOrders = new List<Courier>();

foreach (var courier in companyObj.CourierDetails)

{

if (courier.UserId == courierStaffId)

{

assignedOrders.Add(courier);

}

}

return assignedOrders;

}

}

}

using System;

using System.Collections.Generic;

using Entities;

namespace Services

{

public class CourierUserServiceImpl : ICourierUserService

{

protected CourierCompanyCollection companyObj;

public CourierUserServiceImpl()

{

companyObj = new CourierCompanyCollection();

}

public string PlaceOrder(Courier courierObj)

{

Random random = new Random();

string trackingNumber = random.Next(100000, 999999).ToString();

courierObj.TrackingNumber = trackingNumber;

companyObj.CourierDetails.Add(courierObj);

return trackingNumber;

}

public string GetOrderStatus(string trackingNumber)

{

foreach (var courier in companyObj.CourierDetails)

{

if (courier.TrackingNumber == trackingNumber)

{

return courier.Status;

}

}

throw new TrackingNumberNotFoundException($"Tracking number '{trackingNumber}' not found.");

}

public bool CancelOrder(string trackingNumber)

{

foreach (var courier in companyObj.CourierDetails)

{

if (courier.TrackingNumber == trackingNumber)

{

companyObj.CourierDetails.Remove(courier);

return true; // Successfully canceled the order

}

}

throw new TrackingNumberNotFoundException($"Tracking number '{trackingNumber}' not found.");

}

public List<Courier> GetAssignedOrders(int courierStaffId)

{

List<Courier> assignedOrders = new List<Courier>();

foreach (var courier in companyObj.CourierDetails)

{

if (courier.UserId == courierStaffId)

{

assignedOrders.Add(courier);

}

}

return assignedOrders;

}

}

}

using System;

using Entities;

namespace Services

{

public class CourierAdminServiceImpl : CourierUserServiceImpl, ICourierAdminService

{

public int AddCourierStaff(string name, string contactNumber)

{

Random random = new Random();

int employeeId = random.Next(1000, 9999);

Employee newEmployee = new Employee

{

EmployeeID = employeeId,

EmployeeName = name,

ContactNumber = contactNumber

};

companyObj.EmployeeDetails.Add(newEmployee);

return employeeId; // Placeholder ID

}

}

}

using System;

using Entities;

namespace Service

{

public class CourierAdminServiceCollectionImpl : CourierUserServiceCollectionImpl, ICourierAdminService

{

public int AddCourierStaff(string name, string contactNumber)

{

Random random = new Random();

int employeeId = random.Next(1000, 9999);

Employee newEmployee = new Employee

{

EmployeeID = employeeId,

EmployeeName = name,

ContactNumber = contactNumber

};

companyObj.EmployeeDetails.Add(newEmployee);

return employeeId; // Placeholder ID

}

}

}

//TASK 9 - DATABASE INTERACTION

using System;

using System.Data;

using System.Data.SqlClient;

namespace util

{

public class DBConnection

{

private static string connectionString = "Server=LAGLOP\\SQLEXPRESS;Database=CASE\_STUDY;Trusted\_Connection=True;TrustServerCertificate=True;";

public static SqlConnection GetConnection()

{

SqlConnection connection = new SqlConnection(connectionString);

try

{

connection.Open();

}

catch (Exception ex)

{

Console.WriteLine("Error connecting to database: " + ex.Message);

}

return connection;

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using ConnectionUtil;

using Entities;

namespace Services

{

public class CourierServiceDb

{

private static SqlConnection connection = DBConnection.GetConnection();

public static void InsertCourierOrder(Courier courier)

{

try

{

string query = "INSERT INTO CourierOrders (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, UserId) VALUES (@SenderName, @SenderAddress, @ReceiverName, @ReceiverAddress, @Weight, @Status, @TrackingNumber, @DeliveryDate, @UserId)";

SqlCommand command = new SqlCommand(query, connection);

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);

command.Parameters.AddWithValue("@TrackingNumber", courier.TrackingNumber);

command.Parameters.AddWithValue("@DeliveryDate", courier.DeliveryDate);

command.Parameters.AddWithValue("@UserId", courier.UserId);

command.ExecuteNonQuery();

Console.WriteLine("Courier order inserted successfully.");

}

catch (Exception ex)

{

Console.WriteLine("Error inserting courier order: " + ex.Message);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using ConnectionUtil;

using Entities;

namespace Services

{

public class CourierServiceDb

{

private static SqlConnection connection = DBConnection.GetConnection();

public static void InsertCourierOrder(Courier courier)

{

try

{

string query = "INSERT INTO CourierOrders (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, UserId) VALUES (@SenderName, @SenderAddress, @ReceiverName, @ReceiverAddress, @Weight, @Status, @TrackingNumber, @DeliveryDate, @UserId)";

SqlCommand command = new SqlCommand(query, connection);

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);

command.Parameters.AddWithValue("@TrackingNumber", courier.TrackingNumber);

command.Parameters.AddWithValue("@DeliveryDate", courier.DeliveryDate);

command.Parameters.AddWithValue("@UserId", courier.UserId);

command.ExecuteNonQuery();

Console.WriteLine("Courier order inserted successfully.");

}

catch (Exception ex)

{

Console.WriteLine("Error inserting courier order: " + ex.Message);

}

}

public static void UpdateCourierStatus(string trackingNumber, string newStatus)

{

try

{

string query = "UPDATE CourierOrders SET Status = @Status WHERE TrackingNumber = @TrackingNumber";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Status", newStatus);

command.Parameters.AddWithValue("@TrackingNumber", trackingNumber);

command.ExecuteNonQuery();

Console.WriteLine("Courier status updated successfully.");

}

catch (Exception ex)

{

Console.WriteLine("Error updating courier status: " + ex.Message);

}

}

public static List<string> GetDeliveryHistory(string trackingNumber)

{

List<string> deliveryHistory = new List<string>();

try

{

string query = "SELECT Status, DeliveryDate FROM CourierOrders WHERE TrackingNumber = @TrackingNumber";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@TrackingNumber", trackingNumber);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

string status = reader["Status"].ToString();

DateTime deliveryDate = Convert.ToDateTime(reader["DeliveryDate"]);

deliveryHistory.Add($"Status: {status}, Delivery Date: {deliveryDate}");

}

reader.Close();

}

catch (Exception ex)

{

Console.WriteLine("Error retrieving delivery history: " + ex.Message);

}

return deliveryHistory;

}

}

}

OUTPUT :

--- Inserting a New Order ---

Enter tracking number: TRK123456

Enter status: In Transit

New order inserted successfully.

--- Updating Courier Status ---

Enter tracking number: TRK123456

Enter new status: Delivered

Order status updated successfully.

--- Retrieving Delivery History ---

Enter tracking number: TRK123456

Delivery History for Parcel TRK123456:

1. Date: 2022-05-01, Status: In Transit

2. Date: 2022-05-03, Status: Out for Delivery

3. Date: 2022-05-05, Status: Delivered

Shipment Status

TRK123456 - Delivered

TRK789012 - In Transit

TRK345678 - Delivered

Revenue Report - Total Revenue: 5000

