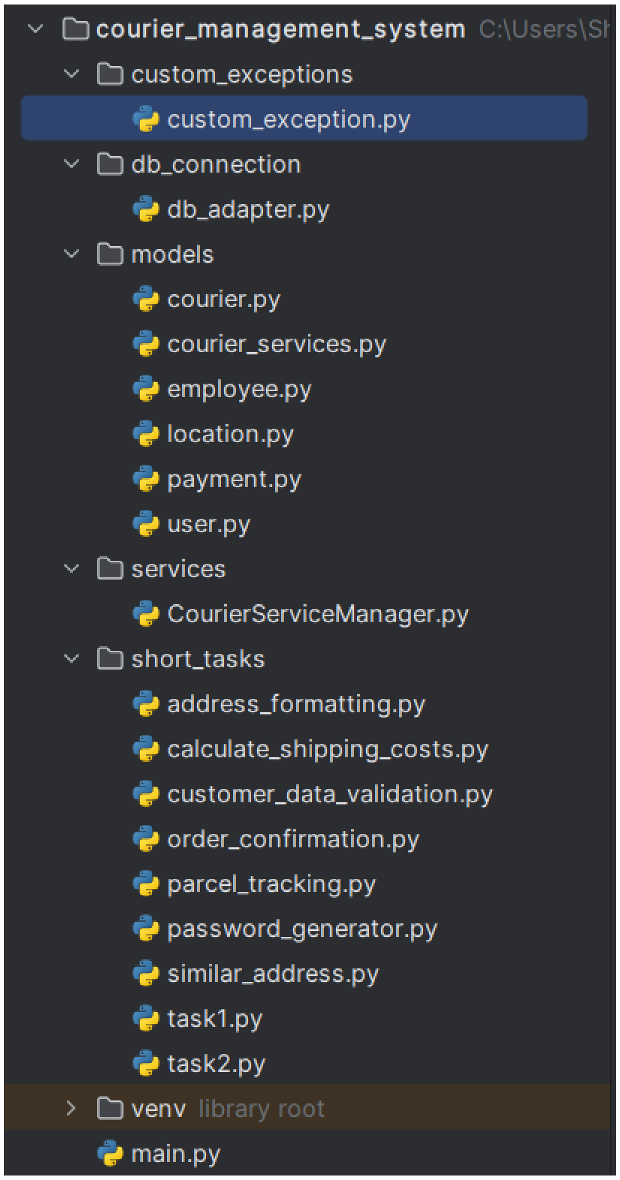
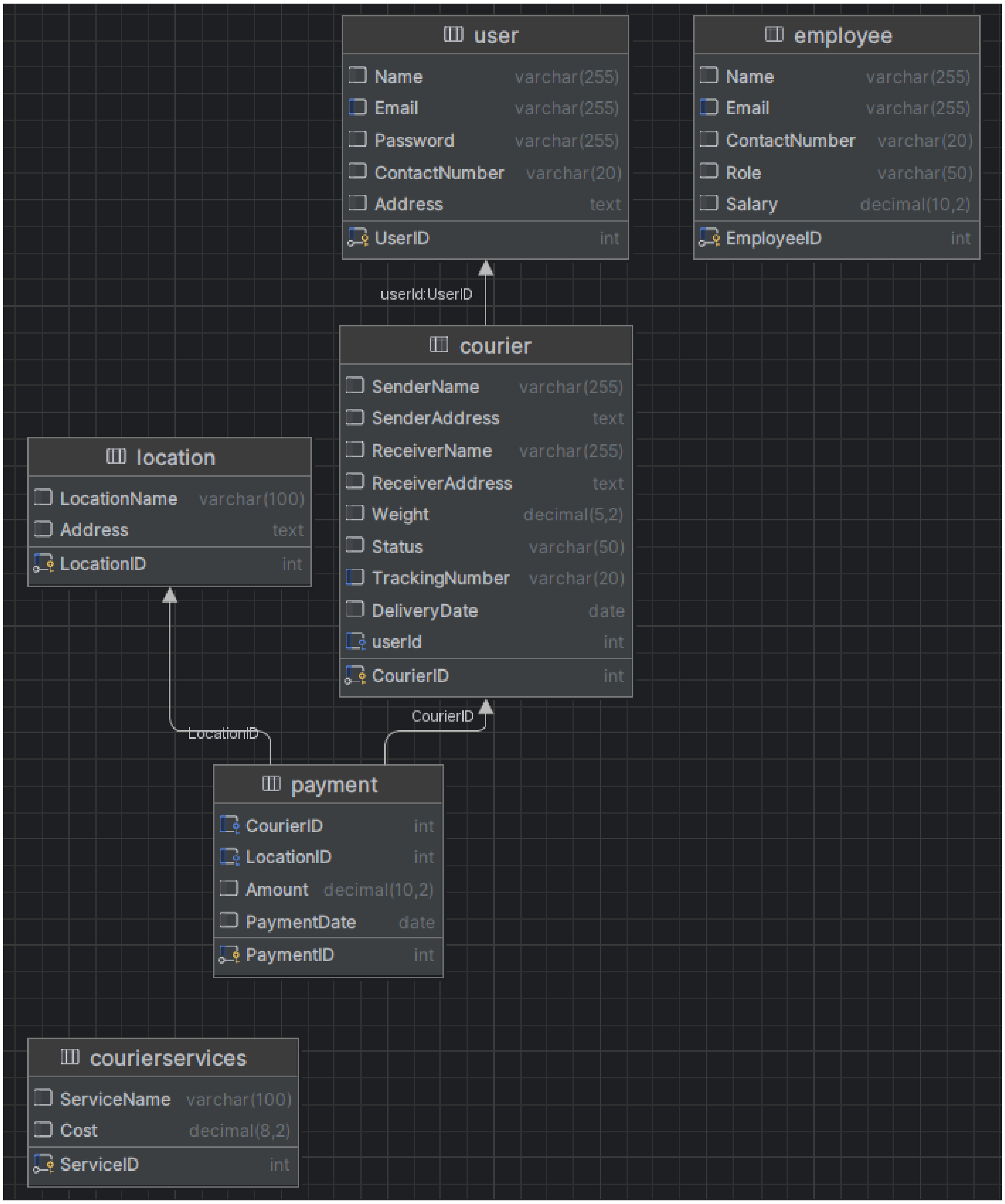
Courier Management System

File Structure



```
courier_management_system C:\Users\Sh
  custom_exceptions
    custom_exception.py
  db_connection
    db_adapter.py
  models
    courier.py
    courier_services.py
    employee.py
    location.py
    payment.py
    user.py
  services
    CourierServiceManager.py
  short_tasks
    address_formatting.py
    calculate_shipping_costs.py
    customer_data_validation.py
    order_confirmation.py
    parcel_tracking.py
    password_generator.py
    similar_address.py
    task1.py
    task2.py
  venv library root
  main.py
```

Database Schema



Custom_exception.py

```python
class CourierNotFound(Exception):
   def __init__(self, value):
       self.value = value
   def __str__(self):
       return "CourierNotFound :" + self.value


class TrackingNumberNotFoundException(Exception):
   def __init__(self, message="Tracking number not found"):
       self.message = message
       super().__init__(self.message)


class InvalidEmployeeIdException(Exception):
   def __init__(self, message="Invalid employee ID"):
       self.message = message
       super().__init__(self.message)
```

Db_adapter.py

```
import mysql.connector


def get_db_connection():
   # Replace the following values with your MySQL server credentials
   config = {
       'user': 'root',
       'password': 'prakhar123',
       'host': 'localhost',
       'database': 'cmsdb'
   }

   try:
       connection = mysql.connector.connect(**config)
       # print("Connected to the database")
       # print('hello World')
       return connection
   except mysql.connector.Error as err:
       print(f"Error: {err}")
       return None


def get_ids(table_name, id_column_name):
   mydb = get_db_connection()
   my_cursor = mydb.cursor()
   sql = 'SELECT ' + id_column_name + ' FROM ' + table_name + ' ORDER BY ' +
id_column_name + ' DESC LIMIT 1'
   # print(sql)
   my_cursor.execute(sql)
   x = list(my_cursor.fetchone())[0]
```

```
    return int(x) + 1


def get_cnts(table_name, id_column_name, column_id):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT count(*) as count FROM ' + table_name + ' WHERE ' +
id_column_name + '=' + column_id
    # print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    return int(x)

def get_counts(table_name, id_column_name1, id_column_name2, column_id1,
column_id2):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT count(*) as count FROM ' + table_name + ' WHERE ' +
id_column_name1 + '=' + column_id1 + ' AND ' + id_column_name2 + '=' + '"' +
str(column_id2) + '"'
    # print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    # print(x)
    return int(x)
```

Courier.py

```
import random

from custom_exceptions.custom_exception import CourierNotFound
from db_connection.db_adapter import *


class Courier:


    def __init__(self, courier_id, sender_name, sender_address, receiver_name,
receiver_address, weight, status,
                 tracking_number, delivery_date, user_id):
        self.connection = get_db_connection()
        self.__courier_id = courier_id
        self.__sender_name = sender_name
        self.__sender_address = sender_address
        self.__receiver_name = receiver_name
        self.__receiver_address = receiver_address
        self.__weight = weight
        self.__status = status
        self.__tracking_number = tracking_number
        self.__delivery_date = delivery_date
        self.__user_id = user_id

    def get_courier_id(self):
        return self.__courier_id

    def get_sender_name(self):
        return self.__sender_name

    def get_sender_address(self):
        return self.__sender_address

    def get_receiver_name(self):
        return self.__receiver_name

    def get_receiver_address(self):
        return self.__receiver_address

    def get_weight(self):
        return self.__weight

    def get_status(self):
        return self.__status

    def get_tracking_number(self):
        return self.__tracking_number

    def get_delivery_date(self):
        return self.__delivery_date

    def update_courier_details(self, sender_name=None, sender_address=None,
receiver_name=None, receiver_address=None,
                              weight=None, status=None, tracking_number=None,
delivery_date=None):
        my_cursor = self.connection.cursor()

        if sender_name:
            sql = 'UPDATE courier SET SenderName = %s WHERE CourierID = %s'
            para = (sender_name, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('SenderName Updated successfully')

        if sender_address:
            sql = 'UPDATE courier SET SenderAddress = %s WHERE CourierID = %s'
            para = (sender_address, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('SenderAddress Updated successfully')

        if receiver_name:
            sql = 'UPDATE courier SET ReceiverName = %s WHERE CourierID = %s'
            para = (receiver_name, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('ReceiverName Updated successfully')

        if receiver_address:
            sql = 'UPDATE courier SET ReceiverAddress = %s WHERE CourierID = %s'
            para = (receiver_address, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('ReceiverAddress Updated successfully')

        if weight:
            sql = 'UPDATE courier SET Weight = %s WHERE CourierID = %s'
            para = (weight, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Weight Updated successfully')

        if status:
            sql = 'UPDATE courier SET Status = %s WHERE CourierID = %s'
            para = (status, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Status Updated successfully')

        if tracking_number:
            sql = 'UPDATE courier SET TrackingNumber = %s WHERE CourierID = %s'
            para = (tracking_number, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('TrackingNumber Updated successfully')

        if delivery_date:
            sql = 'UPDATE courier SET DeliveryDate = %s WHERE CourierID = %s'
            para = (delivery_date, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('DeliveryDate Updated successfully')

        print('Courier details updated successfully')

    def info(self):
        print(f"Courier ID: {self.get_courier_id()}")
        print(f"Sender Name: {self.get_sender_name()}")
        print(f"Sender Address: {self.get_sender_address()}")
        print(f"Receiver Name: {self.get_receiver_name()}")
        print(f"Receiver Address: {self.get_receiver_address()}")
        print(f"Weight: {self.get_weight()}")
        print(f"Status: {self.get_status()}")
        print(f"Tracking Number: {self.get_tracking_number()}")
        print(f"Delivery Date: {self.get_delivery_date()}")
        print("---------------")

    @staticmethod
    def place_courier(self, sender_name, sender_address, receiver_name,
receiver_address, weight, status,
                     tracking_number, delivery_date):
        connection = get_db_connection()
        my_cursor = self.connection.cursor()

        try:
            # Assuming that the CourierID is auto-incremented in the database
            sql = '''
                   INSERT INTO courier (CourierID, SenderName, SenderAddress,
ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate)
                   VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
               '''
            para = (
                get_ids('courier', 'CourierID'), sender_name, sender_address,
receiver_name, receiver_address, weight,
                status, tracking_number,
                delivery_date)
            my_cursor.execute(sql, para)
            connection.commit()
            print('Courier placed successfully.')
        except Exception as e:
            print(f'Error placing courier: {e}')
        finally:
            connection.close()




    @classmethod
    def generate_tracking_number(cls):
        random_number = random.randint(100000, 999999)
        tracking_number = f"TN{random_number}"
        return tracking_number
```

courier_Services.py

```python
import random

from custom_exceptions.custom_exception import CourierNotFound
from db_connection.db_adapter import *


class Courier:


   def __init__(self, courier_id, sender_name, sender_address, receiver_name,
receiver_address, weight, status,
                tracking_number, delivery_date, user_id):
       self.connection = get_db_connection()
       self.__courier_id = courier_id
       self.__sender_name = sender_name
       self.__sender_address = sender_address
       self.__receiver_name = receiver_name
       self.__receiver_address = receiver_address
       self.__weight = weight
       self.__status = status
       self.__tracking_number = tracking_number
       self.__delivery_date = delivery_date
       self.__user_id = user_id

   def get_courier_id(self):
       return self.__courier_id

   def get_sender_name(self):
       return self.__sender_name

   def get_sender_address(self):
       return self.__sender_address

   def get_receiver_name(self):
       return self.__receiver_name

   def get_receiver_address(self):
       return self.__receiver_address

   def get_weight(self):
       return self.__weight

   def get_status(self):
       return self.__status

   def get_tracking_number(self):
       return self.__tracking_number
```

```
    def get_delivery_date(self):
        return self.__delivery_date

    def update_courier_details(self, sender_name=None, sender_address=None,
receiver_name=None, receiver_address=None,
                              weight=None, status=None, tracking_number=None,
delivery_date=None):
        my_cursor = self.connection.cursor()

        if sender_name:
            sql = 'UPDATE courier SET SenderName = %s WHERE CourierID = %s'
            para = (sender_name, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('SenderName Updated successfully')

        if sender_address:
            sql = 'UPDATE courier SET SenderAddress = %s WHERE CourierID = %s'
            para = (sender_address, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('SenderAddress Updated successfully')

        if receiver_name:
            sql = 'UPDATE courier SET ReceiverName = %s WHERE CourierID = %s'
            para = (receiver_name, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('ReceiverName Updated successfully')

        if receiver_address:
            sql = 'UPDATE courier SET ReceiverAddress = %s WHERE CourierID = %s'
            para = (receiver_address, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('ReceiverAddress Updated successfully')

        if weight:
            sql = 'UPDATE courier SET Weight = %s WHERE CourierID = %s'
            para = (weight, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Weight Updated successfully')

        if status:
            sql = 'UPDATE courier SET Status = %s WHERE CourierID = %s'
            para = (status, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Status Updated successfully')

        if tracking_number:
            sql = 'UPDATE courier SET TrackingNumber = %s WHERE CourierID = %s'
            para = (tracking_number, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('TrackingNumber Updated successfully')

        if delivery_date:
            sql = 'UPDATE courier SET DeliveryDate = %s WHERE CourierID = %s'
            para = (delivery_date, self.__courier_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('DeliveryDate Updated successfully')

        print('Courier details updated successfully')

    def info(self):
        print(f"Courier ID: {self.get_courier_id()}")
        print(f"Sender Name: {self.get_sender_name()}")
        print(f"Sender Address: {self.get_sender_address()}")
        print(f"Receiver Name: {self.get_receiver_name()}")
        print(f"Receiver Address: {self.get_receiver_address()}")
        print(f"Weight: {self.get_weight()}")
        print(f"Status: {self.get_status()}")
        print(f"Tracking Number: {self.get_tracking_number()}")
        print(f"Delivery Date: {self.get_delivery_date()}")
        print("---------------")

    @staticmethod
    def place_courier(self, sender_name, sender_address, receiver_name,
receiver_address, weight, status,
                     tracking_number, delivery_date):
        connection = get_db_connection()
        my_cursor = self.connection.cursor()

        try:
            # Assuming that the CourierID is auto-incremented in the database
            sql = '''
                   INSERT INTO courier (CourierID, SenderName, SenderAddress,
ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate)
                   VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
               '''
            para = (
                get_ids('courier', 'CourierID'), sender_name, sender_address,
receiver_name, receiver_address, weight,
                status, tracking_number,
                delivery_date)
            my_cursor.execute(sql, para)
            connection.commit()
            print('Courier placed successfully.')
        except Exception as e:
            print(f'Error placing courier: {e}')
        finally:
            connection.close()




    @classmethod
    def generate_tracking_number(cls):
        random_number = random.randint(100000, 999999)
        tracking_number = f"TN{random_number}"
        return tracking_number
```

Employee.py

```
from db_connection.db_adapter import *

class Employee:
   def __init__(self, employee_id, name, email, contact_number, role, salary):
       self.connection = get_db_connection()
       self.__employee_id = employee_id
       self.__name = name
       self.__email = email
       self.__contact_number = contact_number
       self.__role = role
       self.__salary = salary

   def get_employee_id(self):
       return self.__employee_id

   def get_employee_name(self):
       return self.__name

   def get_employee_email(self):
       return self.__email

   def get_employee_contact_number(self):
       return self.__contact_number

   def get_employee_role(self):
       return self.__role

   def get_employee_salary(self):
       return self.__salary

   def update_employee_details(self, name=None, email=None,
contact_number=None, role=None, salary=None):
       my_cursor = self.connection.cursor()

       if name:
           sql = 'UPDATE employee SET Name = %s WHERE EmployeeID = %s'
           para = (name, self.__employee_id)
           my_cursor.execute(sql, para)
           self.connection.commit()
           print('Name Updated successfully')

       if email:
           sql = 'UPDATE employee SET Email = %s WHERE EmployeeID = %s'
           para = (email, self.__employee_id)
           my_cursor.execute(sql, para)
           self.connection.commit()
           print('Email Updated successfully')

       if contact_number:
           sql = 'UPDATE employee SET ContactNumber = %s WHERE EmployeeID = %s'
           para = (contact_number, self.__employee_id)
           my_cursor.execute(sql, para)
           self.connection.commit()
           print('ContactNumber Updated successfully')

       if role:
           sql = 'UPDATE employee SET Role = %s WHERE EmployeeID = %s'
           para = (role, self.__employee_id)
           my_cursor.execute(sql, para)
           self.connection.commit()
           print('Role Updated successfully')

       if salary:
           sql = 'UPDATE employee SET Salary = %s WHERE EmployeeID = %s'
           para = (salary, self.__employee_id)
           my_cursor.execute(sql, para)
           self.connection.commit()
           print('Salary Updated successfully')

       print('Employee details updated successfully')

   def info(self):
       print(f"Employee ID: {self.get_employee_id()}")
       print(f"Name: {self.get_employee_name()}")
       print(f"Email: {self.get_employee_email()}")
       print(f"Contact Number: {self.get_employee_contact_number()}")
       print(f"Role: {self.get_employee_role()}")
       print(f"Salary: {self.get_employee_salary()}")
       print("---------------")
```

Location.py

```
from db_connection.db_adapter import *


class Location:
    def __init__(self, location_id, location_name, address):
        self.connection = get_db_connection()
        self.__location_id = location_id
        self.__location_name = location_name
        self.__address = address

    def get_location_id(self):
        return self.__location_id

    def get_location_name(self):
        return self.__location_name

    def get_address(self):
        return self.__address

    def update_location_details(self, location_name=None, address=None):
        my_cursor = self.connection.cursor()

        if location_name:
            sql = 'UPDATE location SET LocationName = %s WHERE LocationID = %s'
            para = (location_name, self.__location_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('LocationName Updated successfully')

        if address:
            sql = 'UPDATE location SET Address = %s WHERE LocationID = %s'
            para = (address, self.__location_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Address Updated successfully')

        print('Location details updated successfully')

    def info(self):
        print(f"Location ID: {self.get_location_id()}")
        print(f"Location Name: {self.get_location_name()}")
        print(f"Address: {self.get_address()}")
        print("----------------")
```

Payment.py

```
from db_connection.db_adapter import *

class Payment:
   def __init__(self, payment_id, courier_id, location_id, amount,
payment_date):
        self.connection = get_db_connection()
        self.__payment_id = payment_id
        self.__courier_id = courier_id
        self.__location_id = location_id
        self.__amount = amount
        self.__payment_date = payment_date

   def get_payment_id(self):
        return self.__payment_id

   def get_courier_id(self):
        return self.__courier_id

   def get_location_id(self):
        return self.__location_id

   def get_amount(self):
        return self.__amount

   def get_payment_date(self):
        return self.__payment_date

   def update_payment_details(self, courier_id=None, location_id=None,
amount=None, payment_date=None):
        my_cursor = self.connection.cursor()

        if courier_id:
            sql = 'UPDATE payment SET CourierID = %s WHERE PaymentID = %s'
            para = (courier_id, self.__payment_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('CourierID Updated successfully')

        if location_id:
            sql = 'UPDATE payment SET LocationID = %s WHERE PaymentID = %s'
            para = (location_id, self.__payment_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('LocationID Updated successfully')

        if amount:
            sql = 'UPDATE payment SET Amount = %s WHERE PaymentID = %s'
            para = (amount, self.__payment_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Amount Updated successfully')

        if payment_date:
            sql = 'UPDATE payment SET PaymentDate = %s WHERE PaymentID = %s'
            para = (payment_date, self.__payment_id)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('PaymentDate Updated successfully')

        print('Payment details updated successfully')

    def info(self):
        print(f"Payment ID: {self.get_payment_id()}")
        print(f"Courier ID: {self.get_courier_id()}")
        print(f"Location ID: {self.get_location_id()}")
        print(f"Amount: {self.get_amount()}")
        print(f"Payment Date: {self.get_payment_date()}")
        print("----------------")
```

User.py

```
from db_connection.db_adapter import *

class User:
    def __init__(self, user_id, name, email, password, contact_number, address):
        self.connection = get_db_connection()
        self.__user_id = user_id
        self.__name = name
        self.__email = email
        self.__password = password
        self.__contact_number = contact_number
        self.__address = address

    def get_user_id(self): return self.__user_id
    def get_user_name(self): return self.__name
    def get_user_email(self): return self.__email
    def get_user_password(self): return self.__password
    def get_user_contact_number(self): return self.__contact_number
    def get_user_address(self): return self.__address
    def update_user_details(self, name=None, email=None, password=None,
contact_number=None, address=None):
        my_cursor = self.connection.cursor()

        if name:
            sql = 'UPDATE user SET name = %s WHERE userID = %s'
            para = (name, self.__user_id)
            my_cursor.execute(sql, para)
            self.connection.connect()
            print('Name Updated successfully')

        if email:
            sql = 'UPDATE user SET email = %s WHERE userID = %s'
            para = (email, self.__user_id)
            my_cursor.execute(sql, para)
            self.connection.connect()
            print('email Updated successfully')

        if password:
            sql = 'UPDATE user SET password = %s WHERE userID = %s'
            para = (password, self.__user_id)
            my_cursor.execute(sql, para)
            self.connection.connect()
            print('password Updated successfully')

        if contact_number:
            sql = 'UPDATE user SET ContactNumber = %s WHERE userID = %s'
            para = (contact_number, self.__user_id)
            my_cursor.execute(sql, para)
            self.connection.connect()
            print('contact_number Updated successfully')

        if address:
            sql = 'UPDATE user SET address = %s WHERE userID = %s'
            para = (address, self.__user_id)
            my_cursor.execute(sql, para)
            self.connection.connect()
            print('address Updated successfully')

        print('User details updated successfully')

    def info(self):
        print(f"User ID: {self.get_user_id()}")
        print(f"Name: {self.get_user_name()}")
        print(f"Email: {self.get_user_email()}")
        print(f"Contact Number: {self.get_user_contact_number()}")
        print(f"Address: {self.get_user_address()}")
        print("---------------")
```

Courier_service_manage.py

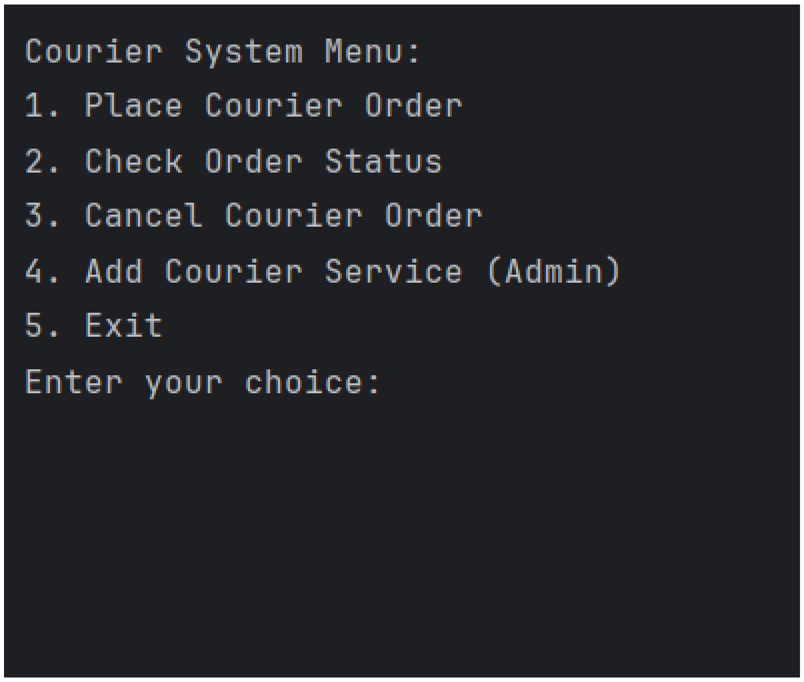
```python
from custom_exceptions.custom_exception import CourierNotFound,
TrackingNumberNotFoundException
from db_connection.db_adapter import *
from models.courier import Courier


class CourierService:

    def __init__(self):
        self.connection = get_db_connection()

    @staticmethod
    def add_courier_services(service_id, service_name, cost, employee_id):
        connection = get_db_connection()
        my_cursor = connection.cursor()
        #print(service_id, service_name, cost, employee_id)
        try:
            if not CourierService.is_employee_admin(employee_id):
                raise PermissionError("Only admins can add courier services.")
            sql = '''
                INSERT INTO courierservices (ServiceID, ServiceName, Cost)
                VALUES (%s, %s, %s)
            '''
            para = (service_id, service_name, cost)
            #print(sql, para)
            my_cursor.execute(sql, para)
            connection.commit()
            print('Courier service added successfully.')
        except PermissionError as e:
            print(f'Error: {e}')
        except Exception as e:
            print(f'Error adding courier service: {e}')
        finally:
            connection.close()

    @staticmethod
    def is_employee_admin(employee_id):
        admin_counts = get_counts('employee', 'employeeID', 'role', employee_id,
'Admin')
        print(admin_counts)
        return admin_counts > 0

    @classmethod
    def get_courier_by_tracking_no(cls, track_id):
        try:
            mydb = get_db_connection()
            my_cursor = mydb.cursor()
            sql = '''
                SELECT * FROM Courier WHERE TrackingNumber = %s
                '''
            para = (track_id,)
            # print(sql, para)
            my_cursor.execute(sql, para)
            x = my_cursor.fetchone()
            if x is None:
                raise TrackingNumberNotFoundException('Invalid Tracking Number')
            else:
                return Courier(*x)
        except TrackingNumberNotFoundException as tnfe:
            print('An error occurred: ', tnfe)
        except Exception as e:
            print('An error occurred: ', e)

    @classmethod
    def cancel_courier(cls, courierID):
        mydb = get_db_connection()
        my_cursor = mydb.cursor()
        courier_exists = get_cnts('courier', 'CourierID', courierID)

        try:
            if courier_exists > 0:
                sql = '''DELETE FROM Courier WHERE CourierID = %s'''
                para = (courier_exists,)
                my_cursor.execute(sql, para)
                self.connection.commit()
                print('Courier Order deleted successfully')
            else:
                raise CourierNotFound('Invalid Courier ID')
        except CourierNotFound as c:
            print(c)
        except Exception as e:
            print('An error occurred', e)
```

Main.py (entry point)

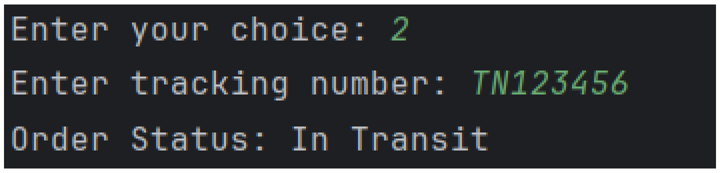
```
from models.courier import Courier
from services.CourierServiceManager import CourierService
from datetime import datetime

class CourierSystemMenu:
    def __init__(self):
       self.courier_service = CourierService()

   def display_menu(self):
       while True:
           print("\nCourier System Menu:")
           print("1. Place Courier Order")
           print("2. Check Order Status")
           print("3. Cancel Courier Order")
           print("4. Add Courier Service (Admin)")
           print("5. Exit")

           choice = input("Enter your choice: ")

           if choice == "1":
               self.place_courier_order()
           elif choice == "2":
               self.check_order_status()
           elif choice == "3":
               self.cancel_courier_order()
           elif choice == "4":
               self.add_courier_service()
           elif choice == "5":
               print("Exiting Courier System. Goodbye!")
               break
           else:
               print("Invalid choice. Please enter a valid option.")

   def place_courier_order(self):

       sender_name = input("Enter sender name: ")
       sender_address = input("Enter sender address: ")
       receiver_name = input("Enter receiver name: ")
       receiver_address = input("Enter receiver address: ")
       weight = float(input("Enter weight: "))
       status = "Yet to Transit"
       tracking_number = Courier.generate_tracking_number()
       delivery_date = input("Enter delivery date (YYYY-MM-DD): ")

       Courier.place_courier(sender_name, sender_address, receiver_name,
receiver_address, weight, status, tracking_number,
datetime.strptime(delivery_date, "%Y-%m-%d").date())
       print(f"Courier order placed successfully. Tracking Number:
{tracking_number}")

   def check_order_status(self):
       tracking_number = input("Enter tracking number: ")
       temp_courier =
CourierService.get_courier_by_tracking_no(track_id=tracking_number)
       print(f"Order Status: {temp_courier.get_status()}")
```

```
   def cancel_courier_order(self):
       courier_id = input("Enter Courier ID: ")
       CourierService.cancel_courier(courier_id)

   def add_courier_service(self):
       employee_id = input("Enter admin employee ID: ")
       if self.courier_service.is_employee_admin(employee_id):
           employee_id2 = (input("Enter Employee ID: "))
           service_id = (input("Enter service ID: "))
           service_name = input("Enter service name: ")
           cost = (input("Enter cost: "))
           self.courier_service.add_courier_services(service_id, service_name,
cost, employee_id2)
       else:
           print("Error: Only admins can add courier services.")

if __name__ == "__main__":
   menu = CourierSystemMenu()
   menu.display_menu()
```

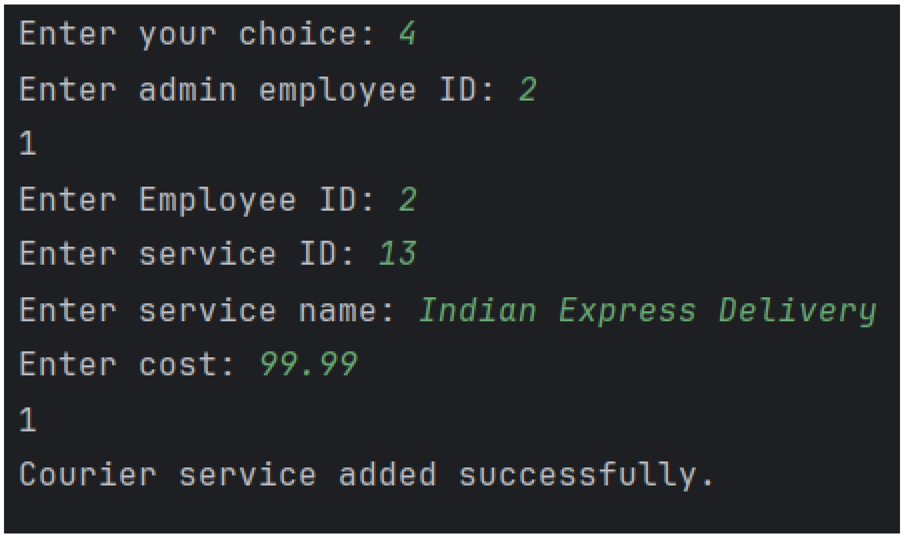
Output



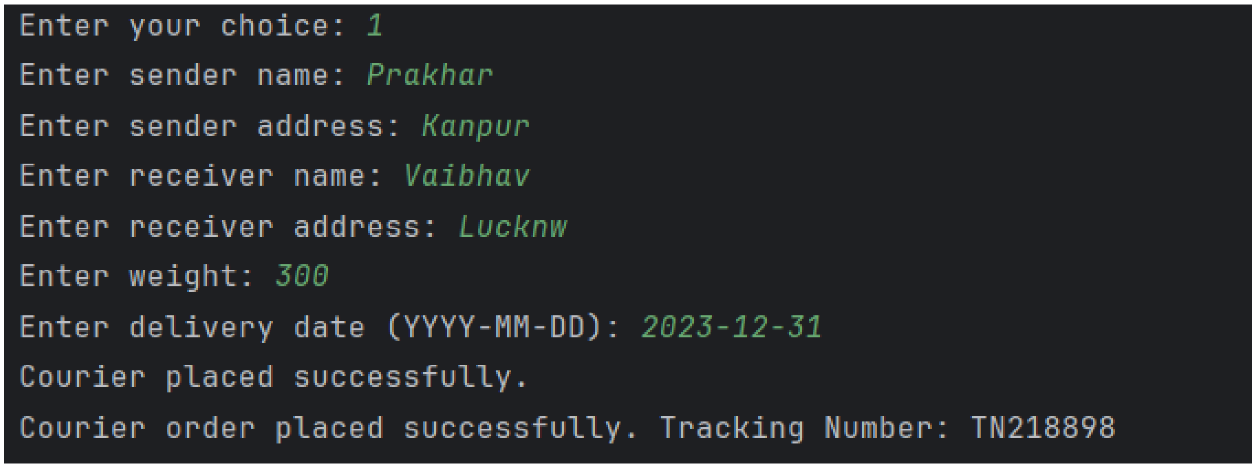
```
Courier System Menu:
1. Place Courier Order
2. Check Order Status
3. Cancel Courier Order
4. Add Courier Service (Admin)
5. Exit
Enter your choice:
```



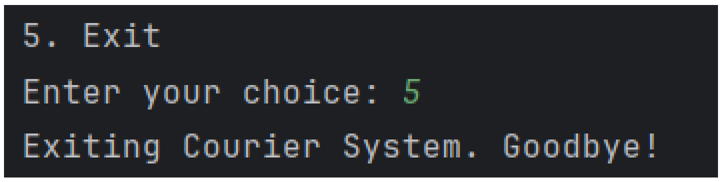
```
Enter your choice: 2
Enter tracking number: TN123456
Order Status: In Transit
```



```
Enter your choice: 4
Enter admin employee ID: 2
1
Enter Employee ID: 2
Enter service ID: 13
Enter service name: Indian Express Delivery
Enter cost: 99.99
1
Courier service added successfully.
```



```
Enter your choice: 1
Enter sender name: Prakhar
Enter sender address: Kanpur
Enter receiver name: Vaibhav
Enter receiver address: Lucknw
Enter weight: 300
Enter delivery date (YYYY-MM-DD): 2023-12-31
Courier placed successfully.
Courier order placed successfully. Tracking Number: TN218898
```



```
5. Exit
Enter your choice: 5
Exiting Courier System. Goodbye!
```