Name – Sanika Sanjay Savalajkar  
Batch – 3  
Superset ID - 5299930

**Assignment 4 – Courier Management System**

Task 1 – Database Design -

Created database called couriermanagement -

|  |
| --- |
| create database couriermanagement;  use couriermanagement; |

Created 6 Tables -

|  |
| --- |
| courier  courierservices  employee  location  payment  user |

|  |
| --- |
| create database couriermanagement;  use couriermanagement;    create table User (UserID INT PRIMARY KEY,  Name VARCHAR(255),  Email VARCHAR(255) UNIQUE,Password VARCHAR(255),  ContactNumber VARCHAR(20),  Address TEXT  );    create table courier (CourierID INT PRIMARY KEY,  SenderName VARCHAR(255),  SenderAddress TEXT,  ReceiverName VARCHAR(255),  ReceiverAddress TEXT,  Weight DECIMAL(5, 2),  Status VARCHAR(50),  TrackingNumber VARCHAR(20) UNIQUE,  DeliveryDate DATE);    create table CourierServices (ServiceID INT PRIMARY KEY,  ServiceName VARCHAR(100),  Cost DECIMAL(8, 2));    create table Employee (EmployeeID INT PRIMARY KEY,  Name VARCHAR(255),  Email VARCHAR(255) UNIQUE,  ContactNumber VARCHAR(20),  Role VARCHAR(50),  Salary DECIMAL(10, 2));    create table Location (LocationID INT PRIMARY KEY,  LocationName VARCHAR(100),  Address TEXT );    create table Payment (PaymentID INT PRIMARY KEY,  CourierID INT,  LocationID INT,  Amount DECIMAL(10, 2),  PaymentDate DATE,  FOREIGN KEY (CourierID) REFERENCES Courier(CourierID),  FOREIGN KEY (LocationID) REFERENCES Location(LocationID)); |

Inserted Values into the table -

|  |
| --- |
| INSERT INTO User (UserID, Name, Email, Password, ContactNumber, Address) VALUES  (1, 'Sanika S', 'sanika@gmail.com', 'password123', '1234567890', '123 Prabhat Road, Pune'),  (2, 'John Smith', 'john@gmail.com', 'pass456', '9876543210', '456 Marine Drive, Mumbai');    INSERT INTO Courier  (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate) VALUES  (1, 'Sanika S', '123 Prabhat Road, Pune', 'Madhu K', '789 MG Road, Hyderabad', 5.5, 'In Transit', 'TRK123456', '2025-03-22'),  (2, 'John Smith', '456 Marine Drive, Mumbai', 'Samruddhi T', '321 FC Road, Pune', 2.0, 'Delivered', 'TRK654321', '2025-03-19');    INSERT INTO CourierServices (ServiceID, ServiceName, Cost) VALUES  (1, 'Standard Delivery', 10.00),  (2, 'Express Delivery', 25.00);    INSERT INTO Employee (EmployeeID, Name, Email, ContactNumber, Role, Salary) VALUES  (1, 'John Doe', 'johndoe@gmail.com', '5551234567', 'Courier Driver', 40000.00),  (2, 'Sai Vignesh', 'saivignesh@gmail.com', '5559876543', 'Manager', 60000.00);    INSERT INTO Location (LocationID, LocationName, Address) VALUES  (1, 'Deccan Gymkhana', '123 Prabhat Road, Pune'),  (2, 'Chowpatty', '456 Marine Drive, Mumbai');    INSERT INTO Payment (PaymentID, CourierID, LocationID, Amount, PaymentDate) VALUES  (1, 1, 1, 15.00, '2025-03-20'),  (2, 2, 2, 50.00, '2025-03-18'); |

Courier Table -

|  |
| --- |
|  |

Courierservices table -

|  |
| --- |
|  |

User table -

|  |
| --- |
|  |

Payment table -

|  |
| --- |
|  |

Location table -

|  |
| --- |
|  |

Employee table -

|  |
| --- |
|  |

Task 2 - Select, Where Queries -

1. List all customers

|  |
| --- |
| Select \* from user |

1. List all orders for a specific customer

|  |
| --- |
| select \* from courier where SenderName = 'Sanika S'; |

1. List all couriers

|  |
| --- |
| select \* from Courier; |

1. List all packages for a specific order

|  |
| --- |
| select \* from courier where CourierID = 2; |

1. List all deliveries for a specific courier

|  |
| --- |
| select \* from courier where Status = 'Delivered'; |

1. List all undelivered packages

|  |
| --- |
| select \* from courier where Status != 'Delivered'; |

1. List all packages that are scheduled for delivery today

|  |
| --- |
| SELECT \* FROM Courier WHERE DeliveryDate = CURRENT\_DATE; |

1. List all packages with a specific status

|  |
| --- |
| select \* from courier where Status = 'In Transit'; |

1. Calculate the total number of packages for each courier

|  |
| --- |
| select SenderName, count(\*) as TotalPackages from Courier group by SenderName; |

1. Find the average delivery time for each courier

|  |
| --- |
| SELECT SenderName, AVG(DATEDIFF(DeliveryDate, NOW())) AS AvgDeliveryTime  FROM Courier  GROUP BY SenderName; |

1. List all packages with a specific weight range

|  |
| --- |
| Select \* from Courier where Weight between 5 and 10; |

1. Retrieve employees whose names contain 'John'

|  |
| --- |
| select \* from employee where name like '%john%'; |

1. Retrieve all courier records with payments greater than $50.

|  |
| --- |
| select \* from courier c join payment p on c.CourierID = p.CourierID where p.Amount > 50; |

OR

|  |
| --- |
| SELECT \* FROM Courier WHERE CourierID IN (SELECT CourierID FROM Payment WHERE Amount > 50); |

Task 3 - GroupBy, Aggregate Functions, Having, Order By, Where Queries -

1. Find the total number of couriers handled by each employee

|  |
| --- |
| SELECT e.EmployeeID, e.Name, COUNT(c.CourierID) AS TotalCouriers  FROM Employee e  JOIN Courier c ON e.EmployeeID = c.EmployeeID  GROUP BY e.EmployeeID, e.Name  ORDER BY TotalCouriers DESC; |

1. Calculate the total revenue generated by each location

|  |
| --- |
| SELECT l.LocationID, l.LocationName, SUM(p.Amount) AS TotalRevenue  FROM Location l  JOIN Payment p ON l.LocationID = p.LocationID  GROUP BY l.LocationID, l.LocationName  ORDER BY TotalRevenue DESC; |

OR

|  |
| --- |
| Select l.LocationName, Sum(Amount) as TotalRevenue  from Location l join Payment p on p.LocationID = l.LocationID  group by l.LocationName; |

1. Find the total number of couriers delivered to each location

|  |
| --- |
| Select Count(CourierID) as TotalDelivered from Courier where Status = 'Delivered'; |

1. Find the courier with the highest average delivery time

|  |
| --- |
| SELECT c.CourierID, c.SenderName, AVG(DATEDIFF(c.DeliveryDate, CURDATE())) AS AvgDeliveryTime  FROM Courier c  GROUP BY c.CourierID, c.SenderName; |

1. Find Locations with Total Payments Less Than a Certain Amount

|  |
| --- |
| select l.LocationName, Sum(p.Amount) as TotalAmount  from Location l  join Payment p on l.LocationID = p.LocationID  group by l.LocationName  having TotalAmount < 50; |

1. Calculate Total Payments per Location

|  |
| --- |
| select l.LocationName, Sum(p.Amount) as TotalAmount  from Location l  join Payment p on l.LocationID = p.LocationID  group by l.LocationName; |

1. Retrieve couriers who have received payments totaling more than $1000 in a specific location (LocationID = X)

|  |
| --- |
| Select c.CourierID, l.LocationName, Sum(p.Amount) as TotalPayment  from Courier c  inner join Payment p on c.CourierID = p.CourierID  inner join Location l on p.LocationID = l.LocationID  group by c.CourierID, l.LocationName  having TotalPayment > 50; |

1. Retrieve couriers who have received payments totaling more than $1000 after a certain date (PaymentDate > 'YYYY-MM-DD')

|  |
| --- |
| Select c.CourierID, l.LocationName, Sum(p.Amount) as TotalPayment  from Courier c  inner join Payment p on c.CourierID = p.CourierID  inner join Location l on p.LocationID = l.LocationID  where p.PaymentDate > '2025-03-15'  group by c.CourierID, l.LocationName  having TotalPayment > 50; |

1. Retrieve locations where the total amount received is more than $5000 before a certain date (PaymentDate > 'YYYY-MM-DD')

|  |
| --- |
| Select c.CourierID, l.LocationName, Sum(p.Amount) as TotalPayment  from Courier c  inner join Payment p on c.CourierID = p.CourierID  inner join Location l on p.LocationID = l.LocationID  where p.PaymentDate < '2025-03-22'  group by c.CourierID, l.LocationName  having TotalPayment > 50; |

Task 4 - Inner Join, Full Outer Join, Cross Join, Left Outer Join, Right Outer Join -

1. Retrieve Payments with Courier Information

|  |
| --- |
| Select p.PaymentID, p.Amount, p.PaymentDate, c.CourierID, c.SenderName, c.ReceiverName  from Payment p  join Courier c on p.CourierID = c.CourierID; |

1. Retrieve Payments with Location Information

|  |
| --- |
| Select p.PaymentID, p.Amount, p.PaymentDate, l.LocationID, l.LocationName  from Payment p  join Location l on p.LocationID = l.LocationID; |

1. Retrieve Payments with Courier and Location Information

|  |
| --- |
| Select p.PaymentID, p.Amount, p.PaymentDate, c.CourierID, c.SenderName, c.ReceiverName, l.LocationName  from Payment p  join Courier c on p.CourierID = c.CourierID  join Location l on p.LocationID = l.LocationID; |

1. List all payments with courier details

|  |
| --- |
| Select p.\*, c.SenderName, c.ReceiverName  from Payment p  join Courier c on p.CourierID = c.CourierID; |

1. Total payments received for each courier

|  |
| --- |
| Select c.CourierID, c.SenderName, SUM(p.Amount) AS TotalPayments  from Payment p  join Courier c on p.CourierID = c.CourierID  group by c.CourierID; |

1. List payments made on a specific date

|  |
| --- |
| SELECT \* FROM Payment WHERE PaymentDate = '2025-03-20'; |

1. Get Courier Information for Each Payment

|  |
| --- |
| select p.paymentid, c.\*  from payment p  join courier c on p.courierid = c.courierid; |

1. Get Payment Details with Location

|  |
| --- |
| select p.\*, l.locationname, l.address  from payment p  join location l on p.locationid = l.locationid; |

1. Calculating Total Payments for Each Courier

|  |
| --- |
| select courierid, sum(amount) as total\_payment  from payment  group by courierid; |

1. List Payments Within a Date Range

|  |
| --- |
| select \* from payment  where paymentdate between '2025-03-15' and '2025-03-25'; |

1. Retrieve a list of all users and their corresponding courier records, including cases where there are no matches on either side

|  |
| --- |
| select u.*, c.* from user u  left join courier c on u.userid = c.userid  union  select u.*, c.* from user u  right join courier c on u.userid = c.userid; |

1. Retrieve a list of all couriers and their corresponding services, including cases where there are no matches on either side

|  |
| --- |
| select c.*, cs.* from courier c  left join courierservices cs on c.courierid = cs.serviceid  union  select c.*, cs.* from courier c  right join courierservices cs on c.courierid = cs.serviceid; |

1. Retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side

|  |
| --- |
| select e.*, p.* from employee e  left join payment p on e.employeeid = p.courierid  union  select e.*, p.* from employee e  right join payment p on e.employeeid = p.courierid; |

1. List all users and all courier services, showing all possible combinations

|  |
| --- |
| select u.name, cs.servicename from user u  cross join courierservices cs; |

1. List all employees and all locations, showing all possible combinations

|  |
| --- |
| select e.name, l.locationname from employee e  cross join location l; |

1. Retrieve a list of couriers and their corresponding sender information (if available)

|  |
| --- |
| select c.courierid, c.sendername, c.senderaddress  from courier c; |

1. Retrieve a list of couriers and their corresponding receiver information (if available)

|  |
| --- |
| select c.courierid, c.receivername, c.receiveraddress  from courier c; |

1. Retrieve a list of couriers along with the courier service details (if available)

|  |
| --- |
| select c.*, cs.* from courier c join courierservices cs on c.courierid = cs.serviceid; |

1. Retrieve a list of employees and the number of couriers assigned to each employee

|  |
| --- |
| select e.employeeid, e.name, count(c.courierid) as total\_assigned  from employee e  join courier c on e.employeeid = c.userid  group by e.employeeid; |

1. Retrieve a list of locations and the total payment amount received at each location

|  |
| --- |
| select l.locationid, l.locationname, sum(p.amount) as total\_received  from location l  join payment p on l.locationid = p.locationid  group by l.locationid; |

1. Retrieve all couriers sent by the same sender (based on SenderName)

|  |
| --- |
| select \* from courier  where sendername = 'sanika s'; |

1. List all employees who share the same role

|  |
| --- |
| select role, group\_concat(name) as employees  from employee  group by role  having count(\*) > 1; |

1. Retrieve all payments made for couriers sent from the same location

|  |
| --- |
| select p.\* from payment p  join courier c on p.courierid = c.courierid  where c.senderaddress = '123 prabhat road, pune'; |

1. Retrieve all couriers sent from the same location (based on SenderAddress)

|  |
| --- |
| select \* from courier  where senderaddress = '123 prabhat road, pune'; |

1. List employees and the number of couriers they have delivered

|  |
| --- |
| select e.name, count(c.courierid) as total\_delivered  from employee e  join courier c on e.employeeid = c.userid  where c.status = 'delivered'  group by e.name; |

1. Find couriers that were paid an amount greater than the cost of their respective courier services

|  |
| --- |
| select c.\*, p.amount, cs.cost  from courier c  join payment p on c.courierid = p.courierid  join courierservices cs on cs.serviceid = c.courierid  where p.amount > cs.cost; |

Scope - Inner Queries, Non Equi Joins, Equi joins, Exist, Any, All -

1. Find couriers that have a weight greater than the average weight of all couriers

|  |
| --- |
| select \* from courier  where weight > (select avg(weight) from courier); |

1. Find the names of all employees who have a salary greater than the average salary

|  |
| --- |
| select name, salary from employee  where salary > (select avg(salary) from employee); |

1. Find the total cost of all courier services where the cost is less than the maximum cost

|  |
| --- |
| select sum(cost) as total  from courierservices  where cost < (select max(cost) from courierservices); |

1. Find all couriers that have been paid for

|  |
| --- |
| select distinct c.\* from courier c  join payment p on c.courierid = p.courierid; |

1. Find the locations where the maximum payment amount was made

|  |
| --- |
| select locationid, amount from payment  where amount = (select max(amount) from payment); |

1. Find all couriers whose weight is greater than the weight of all couriers sent by a specific sender (e.g., 'SenderName')

|  |
| --- |
| select \* from courier  where weight > all (  select weight from courier where sendername = 'sanika s'); |

Java Implementation -

|  |
| --- |
| Created 6 packages - 1. entity 2. dao  3. exception  4. util  5. main 6. utility  entity package - 1. Courier.java 2. Employee.java 3. Location.java 4. Payment.java 5. User.java 6. CourierCompany.java 7. CourierCompanyCollection.java  dao package - 1. ICourierUserService.java 2. ICourierAdminService.java 3. CourierUserServiceCollectionImpl.java 4. CourierServiceDb.java  exception package - 1. InvalidEmployeeIdException.java 2. TrackingNumberNotFoundException.java  util package - 1. DBConnUtil.java 2. DBPropertyUtil.java  main package - 1. MainModule.java  utility package - 1. CourierUtils.java |

Entity package -

Courier.java

|  |
| --- |
| package entity;    import java.util.Date;    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 Date deliveryDate;  private int userId;    public Courier() {}    public Courier(int courierID, String senderName, String senderAddress, String receiverName,  String receiverAddress, double weight, String status, String trackingNumber,  Date 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 getCourierID() { return courierID; }  public void setCourierID(int courierID) { this.courierID = courierID; }    public String getSenderName() { return senderName; }  public void setSenderName(String senderName) { this.senderName = senderName; }    public String getSenderAddress() { return senderAddress; }  public void setSenderAddress(String senderAddress) { this.senderAddress = senderAddress; }    public String getReceiverName() { return receiverName; }  public void setReceiverName(String receiverName) { this.receiverName = receiverName; }    public String getReceiverAddress() { return receiverAddress; }  public void setReceiverAddress(String receiverAddress) { this.receiverAddress = receiverAddress; }    public double getWeight() { return weight; }  public void setWeight(double weight) { this.weight = weight; }    public String getStatus() { return status; }  public void setStatus(String status) { this.status = status; }    public String getTrackingNumber() { return trackingNumber; }  public void setTrackingNumber(String trackingNumber) { this.trackingNumber = trackingNumber; }    public Date getDeliveryDate() { return deliveryDate; }  public void setDeliveryDate(Date deliveryDate) { this.deliveryDate = deliveryDate; }    public int getUserId() { return userId; }  public void setUserId(int userId) { this.userId = userId; }    @Override  public String toString() {  return "Courier [courierID=" + courierID + ", senderName=" + senderName +  ", receiverName=" + receiverName + ", trackingNumber=" + trackingNumber + ", status=" + status + "]";  }  } |

Employee.java

|  |
| --- |
| package 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 getEmployeeID() {  return employeeID;  }    public void setEmployeeID(int employeeID) {  this.employeeID = employeeID;  }    public String getEmployeeName() {  return employeeName;  }    public void setEmployeeName(String employeeName) {  this.employeeName = employeeName;  }    public String getEmail() {  return email;  }    public void setEmail(String email) {  this.email = email;  }    public String getContactNumber() {  return contactNumber;  }    public void setContactNumber(String contactNumber) {  this.contactNumber = contactNumber;  }    public String getRole() {  return role;  }    public void setRole(String role) {  this.role = role;  }    public double getSalary() {  return salary;  }    public void setSalary(double salary) {  this.salary = salary;  }  } |

Location.java

|  |
| --- |
| package entity;    public class Location {  private int locationID;  private String locationName;  private String address;  public Location() {  }    public Location(int locationID, String locationName, String address) {  super();  this.locationID = locationID;  this.locationName = locationName;  this.address = address;  }    public int getLocationID() {  return locationID;  }    public void setLocationID(int locationID) {  this.locationID = locationID;  }    public String getLocationName() {  return locationName;  }    public void setLocationName(String locationName) {  this.locationName = locationName;  }    public String getAddress() {  return address;  }    public void setAddress(String address) {  this.address = address;  }  } |

Payment.java

|  |
| --- |
| package entity;    import java.util.Date;    public class Payment {  private long paymentID;  private long courierID;  private double amount;  private Date paymentDate;  public Payment() {  }    public Payment(long paymentID, long courierID, double amount, Date paymentDate) {  super();  this.paymentID = paymentID;  this.courierID = courierID;  this.amount = amount;  this.paymentDate = paymentDate;  }    public long getPaymentID() {  return paymentID;  }    public void setPaymentID(long paymentID) {  this.paymentID = paymentID;  }    public long getCourierID() {  return courierID;  }    public void setCourierID(long courierID) {  this.courierID = courierID;  }    public double getAmount() {  return amount;  }    public void setAmount(double amount) {  this.amount = amount;  }    public Date getPaymentDate() {  return paymentDate;  }    public void setPaymentDate(Date paymentDate) {  this.paymentDate = paymentDate;  }  } |

User.java

|  |
| --- |
| package 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 getUserID() {  return userID;  }    public void setUserID(int userID) {  this.userID = userID;  }    public String getUserName() {  return userName;  }    public void setUserName(String userName) {  this.userName = userName;  }    public String getEmail() {  return email;  }    public void setEmail(String email) {  this.email = email;  }    public String getPassword() {  return password;  }    public void setPassword(String password) {  this.password = password;  }    public String getContactNumber() {  return contactNumber;  }    public void setContactNumber(String contactNumber) {  this.contactNumber = contactNumber;  }    public String getAddress() {  return address;  }    public void setAddress(String address) {  this.address = address;  }  } |

CourierCompany.java

|  |
| --- |
| package entity;    import java.util.List;    public class CourierCompany {  private String companyName;  private List<Courier> courierDetails;  private List<Employee> employeeDetails;  private List<Location> locationDetails;  public CourierCompany() {  }    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 getCompanyName() {  return companyName;  }    public void setCompanyName(String companyName) {  this.companyName = companyName;  }    public List<Courier> getCourierDetails() {  return courierDetails;  }    public void setCourierDetails(List<Courier> courierDetails) {  this.courierDetails = courierDetails;  }    public List<Employee> getEmployeeDetails() {  return employeeDetails;  }    public void setEmployeeDetails(List<Employee> employeeDetails) {  this.employeeDetails = employeeDetails;  }    public List<Location> getLocationDetails() {  return locationDetails;  }    public void setLocationDetails(List<Location> locationDetails) {  this.locationDetails = locationDetails;  }  } |

CourierCompanyCollection.java

|  |
| --- |
| package entity;    import java.util.ArrayList;  import java.util.List;    public class CourierCompanyCollection {    private String companyName;  private List<Courier> courierDetails;  private List<Employee> employeeDetails;  private List<Location> locationDetails;    public CourierCompanyCollection() {  courierDetails = new ArrayList<>();  employeeDetails = new ArrayList<>();  locationDetails = new ArrayList<>();  }    public CourierCompanyCollection(String companyName) {  this();  this.companyName = companyName;  }    public String getCompanyName() { return companyName; }  public void setCompanyName(String companyName) { this.companyName = companyName; }    public List<Courier> getCourierDetails() { return courierDetails; }  public void setCourierDetails(List<Courier> courierDetails) { this.courierDetails = courierDetails; }    public List<Employee> getEmployeeDetails() { return employeeDetails; }  public void setEmployeeDetails(List<Employee> employeeDetails) { this.employeeDetails = employeeDetails; }    public List<Location> getLocationDetails() { return locationDetails; }  public void setLocationDetails(List<Location> locationDetails) { this.locationDetails = locationDetails; }    @Override  public String toString() {  return "CourierCompanyCollection [companyName=" + companyName + ", totalCouriers=" + courierDetails.size() + "]";  }  } |

Dao package -

ICourierUserService.java

|  |
| --- |
| package dao;    import java.util.List;  import entity.Courier;    public interface ICourierUserService {  String placeOrder(Courier courierObj);  String getOrderStatus(String trackingNumber);  boolean cancelOrder(String trackingNumber);  List<Courier> getAssignedOrder(int courierStaffId);  } |

ICourierAdminService.java

|  |
| --- |
| package dao;    import entity.Employee;    public interface ICourierAdminService {  int addCourierStaff(Employee obj);  } |

CourierUserServiceCollectionImpl.java

|  |
| --- |
| package dao;    import java.util.ArrayList;  import java.util.List;  import entity.Courier;  import entity.CourierCompanyCollection;    public class CourierUserServiceCollectionImpl implements ICourierUserService {    private CourierCompanyCollection companyObj;    public CourierUserServiceCollectionImpl(CourierCompanyCollection companyObj) {  this.companyObj = companyObj;  }    @Override  public String placeOrder(Courier courierObj) {  companyObj.getCourierDetails().add(courierObj);  return courierObj.getTrackingNumber();  }    @Override  public String getOrderStatus(String trackingNumber) {  for (Courier c : companyObj.getCourierDetails()) {  if (c.getTrackingNumber().equalsIgnoreCase(trackingNumber)) {  return c.getStatus();  }  }  return "Tracking number not found.";  }    @Override  public boolean cancelOrder(String trackingNumber) {  for (Courier c : companyObj.getCourierDetails()) {  if (c.getTrackingNumber().equalsIgnoreCase(trackingNumber)) {  if (!c.getStatus().equalsIgnoreCase("Delivered")) {  c.setStatus("Cancelled");  return true;  }  }  }  return false;  }    @Override  public List<Courier> getAssignedOrder(int courierStaffId) {  // Placeholder: requires mapping of courier to staff  return new ArrayList<>();  }  } |

CourierServiceDb.java

|  |
| --- |
| package dao;    import java.sql.\*;  import java.util.ArrayList;  import java.util.List;  import entity.Courier;  import util.DBPropertyUtil;    public class CourierServiceDb {  private static Connection connection;    public CourierServiceDb() {  connection = DBPropertyUtil.getDbConnection();  }    public boolean insertCourier(Courier courier) {  boolean result = false;  try {  String query = "INSERT INTO Courier (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";  PreparedStatement pstmt = connection.prepareStatement(query);  pstmt.setInt(1, courier.getCourierID());  pstmt.setString(2, courier.getSenderName());  pstmt.setString(3, courier.getSenderAddress());  pstmt.setString(4, courier.getReceiverName());  pstmt.setString(5, courier.getReceiverAddress());  pstmt.setDouble(6, courier.getWeight());  pstmt.setString(7, courier.getStatus());  pstmt.setString(8, courier.getTrackingNumber());  pstmt.setDate(9, new java.sql.Date(courier.getDeliveryDate().getTime()));    int count = pstmt.executeUpdate();  result = count > 0;  } catch (SQLException e) {  System.out.println("Insert failed: " + e.getMessage());  }  return result;  }    public boolean updateStatus(String trackingNumber, String status) {  try {  String query = "UPDATE Courier SET Status = ? WHERE TrackingNumber = ?";  PreparedStatement pstmt = connection.prepareStatement(query);  pstmt.setString(1, status);  pstmt.setString(2, trackingNumber);  int rows = pstmt.executeUpdate();  return rows > 0;  } catch (SQLException e) {  System.out.println("Error updating status: " + e.getMessage());  }  return false;  }    public Courier getCourierByTrackingNumber(String trackingNumber) {  Courier courier = null;  try {  String query = "SELECT \* FROM Courier WHERE TrackingNumber = ?";  PreparedStatement pstmt = connection.prepareStatement(query);  pstmt.setString(1, trackingNumber);  ResultSet rs = pstmt.executeQuery();    if (rs.next()) {  courier = new Courier(  rs.getInt("CourierID"),  rs.getString("SenderName"),  rs.getString("SenderAddress"),  rs.getString("ReceiverName"),  rs.getString("ReceiverAddress"),  rs.getDouble("Weight"),  rs.getString("Status"),  rs.getString("TrackingNumber"),  rs.getDate("DeliveryDate"),  0 // userId is not in DB schema but present in Courier POJO  );  }  } catch (SQLException e) {  System.out.println("Error fetching courier: " + e.getMessage());  }  return courier;  }    public List<Courier> getDeliveryHistory() {  List<Courier> list = new ArrayList<>();  try {  Statement stmt = connection.createStatement();  ResultSet rs = stmt.executeQuery("SELECT \* FROM Courier");    while (rs.next()) {  Courier courier = new Courier(  rs.getInt("CourierID"),  rs.getString("SenderName"),  rs.getString("SenderAddress"),  rs.getString("ReceiverName"),  rs.getString("ReceiverAddress"),  rs.getDouble("Weight"),  rs.getString("Status"),  rs.getString("TrackingNumber"),  rs.getDate("DeliveryDate"),  0  );  list.add(courier);  }  } catch (SQLException e) {  System.out.println("Error retrieving history: " + e.getMessage());  }  return list;  }  } |

Exception package -

InvalidEmployeeIdException.java

|  |
| --- |
| package exception;    public class InvalidEmployeeIdException extends Exception {  public InvalidEmployeeIdException(String message) {  super(message);  }  } |

TrackingNumberNotFoundException.java

|  |
| --- |
| package exception;    public class TrackingNumberNotFoundException extends Exception {  public TrackingNumberNotFoundException(String message) {  super(message);  }  } |

Util package -

db.properties File -

|  |
| --- |
| user=root  password=Sanika@12345  protocol=jdbc:mysql:  system=localhost  database=couriermanagement  port=3306 |

DBConnUtil.java

|  |
| --- |
| package util;    import java.io.FileInputStream;  import java.io.IOException;  import java.util.Properties;    public class DBConnUtil {    public static String getConnectionString(String fileName) throws IOException {  Properties props = new Properties();  FileInputStream fis = new FileInputStream(fileName);  props.load(fis);    String user = props.getProperty("user");  String password = props.getProperty("password");  String protocol = props.getProperty("protocol");  String system = props.getProperty("system");  String port = props.getProperty("port");  String database = props.getProperty("database");    return protocol + "//" + system + ":" + port + "/" + database + "?user=" + user + "&password=" + password;  }  } |

DBPropertyUtil.java

|  |
| --- |
| package util;    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;    public class DBPropertyUtil {    private static final String fileName = "db.properties";    public static Connection getDbConnection() {  Connection con = null;  try {  String connStr = DBConnUtil.getConnectionString(fileName);  con = DriverManager.getConnection(connStr);  } catch (Exception e) {  System.out.println("Error establishing DB connection.");  e.printStackTrace();  }  return con;  }  } |

Utility package -

CourierUtils.java

|  |
| --- |
| package utility;  import entity.Courier;  import java.security.SecureRandom; import java.text.SimpleDateFormat; import java.util.\*;  public class CourierUtils {  // === Task 1 ===  public static void checkStatus(String status) { if (status.equalsIgnoreCase("Delivered")) {  System.out.println("Order has been delivered.");  } else if (status.equalsIgnoreCase("Processing")) {  System.out.println("Order is still being processed.");  } else if (status.equalsIgnoreCase("Cancelled")) {  System.out.println("Order was cancelled.");  } else {  System.out.println("Unknown status.");  } }  public static void categorizeWeight(double weight) {  String category;  if (weight < 5) category = "Light";  else if (weight < 15) category = "Medium";  else category = "Heavy";   switch (category) {  case "Light" -> System.out.println("This parcel is Light.");  case "Medium" -> System.out.println("This parcel is Medium.");  case "Heavy" -> System.out.println("This parcel is Heavy.");  } }  public static void login() {  Scanner scanner = new Scanner(System.in);  String storedUsername = "admin";  String storedPassword = "pass123";   System.out.print("Enter username: ");  String inputUser = scanner.nextLine();  System.out.print("Enter password: ");  String inputPass = scanner.nextLine();   if (inputUser.equals(storedUsername) && inputPass.equals(storedPassword)) {  System.out.println("Login successful.");  } else {  System.out.println("Invalid credentials.");  } }  public static void assignCourier(int[] loads) {  int min = loads[0], index = 0;  for (int i = 1; i < loads.length; i++) {  if (loads[i] < min) {  min = loads[i];  index = i;  }  }  System.out.println("Assign parcel to Courier ID: " + index + " (Load: " + min + ")"); }  // === Task 2 ===  public static void displayOrdersByUser(List<Courier> orders, int userId) {  for (Courier c : orders) {  if (c.getUserId() == userId) {  System.out.println(c);  }  } }  public static void trackCourierLive(String[] path) throws InterruptedException {  int i = 0;  while (i < path.length) {  System.out.println("Location: " + path[i]);  Thread.sleep(1000);  i++;  }  System.out.println("Courier delivered."); }  // === Task 3 ===  public static void displayTrackingHistory(String[] locations) {  System.out.println("Tracking History:");  for (String loc : locations) {  System.out.println("→ " + loc);  } }  public static int findNearestCourier(int[] distances) {  int min = distances[0], index = 0;  for (int i = 1; i < distances.length; i++) {  if (distances[i] < min) {  min = distances[i];  index = i;  }  }  System.out.println("Nearest Courier ID: " + index + " | Distance: " + min + "km");  return index; }  // === Task 4 ===  public static void trackParcel2D() {  String[][] parcels = {  {"TRK123", "In Transit"},  {"TRK456", "Out for Delivery"},  {"TRK789", "Delivered"}  };   Scanner sc = new Scanner(System.in);  System.out.print("Enter tracking number: ");  String track = sc.nextLine();   boolean found = false;  for (String[] p : parcels) {  if (p[0].equalsIgnoreCase(track)) {  System.out.println("Status: " + p[1]);  found = true;  break;  }  }  if (!found) System.out.println("Tracking number not found."); }  public static boolean validateCustomerData(String data, String type) {  return switch (type.toLowerCase()) {  case "name" -> data.matches("[A-Z][a-z]+( [A-Z][a-z]+)\*");  case "address" -> data.matches("[A-Za-z0-9 ,]+");  case "phone" -> data.matches("\\d{3}-\\d{3}-\\d{4}");  default -> false;  }; }  public static String formatAddress(String street, String city, String state, String zip) {  return capitalize(street) + ", " + capitalize(city) + ", " + capitalize(state) + " - " + zip.replaceAll("\\D", ""); }  private static String capitalize(String str) {  String[] words = str.toLowerCase().split(" ");  StringBuilder sb = new StringBuilder();  for (String w : words) {  sb.append(Character.toUpperCase(w.charAt(0))).append(w.substring(1)).append(" ");  }  return sb.toString().trim(); }  public static String generateEmail(String name, String order, String address, Date date) {  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  return "Dear " + name + ",\nYour order #" + order + " will be delivered to:\n" + address +  "\nExpected Delivery: " + sdf.format(date) + "\nThank you for choosing us!"; }  public static double calculateShippingCost(String src, String dest, double weight) {  int dist = Math.abs(src.length() - dest.length()) \* 10;  return dist \* (weight / 5.0) \* 10; }  public static String generatePassword(int length) {  String upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ", lower = "abcdefghijklmnopqrstuvwxyz",  digits = "0123456789", special = "!@#$%&\*", all = upper + lower + digits + special;  SecureRandom rand = new SecureRandom();  StringBuilder sb = new StringBuilder();  sb.append(upper.charAt(rand.nextInt(upper.length())));  sb.append(lower.charAt(rand.nextInt(lower.length())));  sb.append(digits.charAt(rand.nextInt(digits.length())));  sb.append(special.charAt(rand.nextInt(special.length())));  for (int i = 4; i < length; i++) {  sb.append(all.charAt(rand.nextInt(all.length())));  }  return sb.toString(); }  public static List<String> findSimilarAddresses(String[] db, String keyword) {  List<String> results = new ArrayList<>();  for (String addr : db) {  if (addr.toLowerCase().contains(keyword.toLowerCase())) {  results.add(addr);  }  }  return results; }  } |

Main package -

|  |
| --- |
| package main;    import dao.CourierServiceDb;  import entity.Courier;  import exception.TrackingNumberNotFoundException;  import utility.CourierUtils;    import java.text.SimpleDateFormat;  import java.util.Date;  import java.util.List;  import java.util.Scanner;    public class MainModule {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  CourierServiceDb service = new CourierServiceDb();  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");    boolean running = true;    while (running) {  System.out.println("\n=== Courier Management System ===");  System.out.println("1. Place New Order");  System.out.println("2. Check Order Status");  System.out.println("3. Cancel Order");  System.out.println("4. Show All Orders");  System.out.println("5. Validate Customer Info");  System.out.println("6. Format Address");  System.out.println("7. Generate Order Confirmation Email");  System.out.println("8. Calculate Shipping Cost");  System.out.println("9. Generate Secure Password");  System.out.println("10. Find Similar Addresses");  System.out.println("11. Parcel Tracking (2D Array)");  System.out.println("12. Real-Time Courier Tracking");  System.out.println("13. Show Orders by User ID");  System.out.println("14. Assign Nearest Courier");  System.out.println("15. Weight Category");  System.out.println("16. Login System");  System.out.println("17. Exit");  System.out.print("Choose option: ");    int choice = scanner.nextInt();  scanner.nextLine(); // consume newline    try {  switch (choice) {  case 1 -> {  System.out.print("Courier ID: ");  int id = scanner.nextInt(); scanner.nextLine();  System.out.print("Sender Name: ");  String sender = scanner.nextLine();  System.out.print("Sender Address: ");  String sAddr = scanner.nextLine();  System.out.print("Receiver Name: ");  String receiver = scanner.nextLine();  System.out.print("Receiver Address: ");  String rAddr = scanner.nextLine();  System.out.print("Weight (kg): ");  double weight = scanner.nextDouble(); scanner.nextLine();  System.out.print("Status: ");  String status = scanner.nextLine();  System.out.print("Tracking Number: ");  String trackingNumber = scanner.nextLine();  System.out.print("Delivery Date (yyyy-MM-dd): ");  Date date = sdf.parse(scanner.nextLine());    Courier c = new Courier(id, sender, sAddr, receiver, rAddr, weight, status, trackingNumber, date, 1);  if (service.insertCourier(c))  System.out.println("Order placed successfully!");  }    case 2 -> {  System.out.print("Enter tracking number: ");  String track = scanner.nextLine();  Courier c = service.getCourierByTrackingNumber(track);  if (c != null) {  CourierUtils.checkStatus(c.getStatus());  } else {  throw new TrackingNumberNotFoundException("Tracking number not found.");  }  }    case 3 -> {  System.out.print("Enter tracking number to cancel: ");  String track = scanner.nextLine();  if (service.updateStatus(track, "Cancelled")) {  System.out.println("Order cancelled.");  } else {  throw new TrackingNumberNotFoundException("Order not found.");  }  }    case 4 -> {  List<Courier> all = service.getDeliveryHistory();  for (Courier c : all) System.out.println(c);  }    case 5 -> {  System.out.print("Enter detail type (name/address/phone): ");  String type = scanner.nextLine();  System.out.print("Enter value: ");  String value = scanner.nextLine();  boolean valid = CourierUtils.validateCustomerData(value, type);  System.out.println(valid ? "Valid" : "Invalid");  }    case 6 -> {  System.out.print("Street: ");  String street = scanner.nextLine();  System.out.print("City: ");  String city = scanner.nextLine();  System.out.print("State: ");  String state = scanner.nextLine();  System.out.print("ZIP: ");  String zip = scanner.nextLine();  String formatted = CourierUtils.formatAddress(street, city, state, zip);  System.out.println("Formatted Address:\n" + formatted);  }    case 7 -> {  System.out.print("Customer Name: ");  String name = scanner.nextLine();  System.out.print("Order Number: ");  String order = scanner.nextLine();  System.out.print("Delivery Address: ");  String addr = scanner.nextLine();  System.out.print("Expected Delivery (yyyy-MM-dd): ");  Date date = sdf.parse(scanner.nextLine());  System.out.println(CourierUtils.generateEmail(name, order, addr, date));  }    case 8 -> {  System.out.print("Source Address: ");  String src = scanner.nextLine();  System.out.print("Destination Address: ");  String dst = scanner.nextLine();  System.out.print("Weight (kg): ");  double wt = scanner.nextDouble();  double cost = CourierUtils.calculateShippingCost(src, dst, wt);  System.out.println("Shipping cost: ₹" + cost);  }    case 9 -> {  System.out.print("Password length: ");  int len = scanner.nextInt(); scanner.nextLine();  String password = CourierUtils.generatePassword(len);  System.out.println("Generated Password: " + password);  }    case 10 -> {  String[] db = {  "123 Prabhat Road, Pune",  "456 Marine Drive, Mumbai",  "321 FC Road, Pune"  };  System.out.print("Search for: ");  String keyword = scanner.nextLine();  List<String> matches = CourierUtils.findSimilarAddresses(db, keyword);  matches.forEach(System.out::println);  }    case 11 -> {  CourierUtils.trackParcel2D();  }    case 12 -> {  String[] path = {"Warehouse", "Sorting Center", "On Route", "Out for Delivery", "Delivered"};  CourierUtils.trackCourierLive(path);  }    case 13 -> {  System.out.print("Enter user ID to view orders: ");  int uid = scanner.nextInt(); scanner.nextLine();  List<Courier> all = service.getDeliveryHistory();  CourierUtils.displayOrdersByUser(all, uid);  }    case 14 -> {  int[] distances = {3, 5, 2, 8};  CourierUtils.findNearestCourier(distances);  }    case 15 -> {  System.out.print("Enter weight (kg): ");  double weight = scanner.nextDouble(); scanner.nextLine();  CourierUtils.categorizeWeight(weight);  }    case 16 -> {  CourierUtils.login();  }    case 17 -> {  running = false;  System.out.println("Thank you for using Courier Management System.");  }    default -> System.out.println("Invalid option.");  }  } catch (TrackingNumberNotFoundException e) {  System.out.println("Error: " + e.getMessage());  } catch (Exception e) {  System.out.println("An unexpected error occurred: " + e.getMessage());  e.printStackTrace();  }  }    scanner.close();  }  } |

Output -

|  |
| --- |
|  |

1. Place New Order

|  |
| --- |
|  |

1. Check Order Status

|  |
| --- |
|  |

1. Cancel Order

|  |
| --- |
|  |

1. Show All Orders

|  |
| --- |
|  |

1. Validate Customer Info

|  |
| --- |
|  |

1. Format Address

|  |
| --- |
|  |

1. Generate Order Confirmation Email

|  |
| --- |
|  |

1. Calculate Shipping Cost

|  |
| --- |
|  |

1. Generate Secure Password

|  |
| --- |
|  |

1. Find Similar Addresses

|  |
| --- |
|  |

1. Parcel Tracking (2D Array)

|  |
| --- |
|  |

1. Real-Time Courier Tracking

|  |
| --- |
|  |

1. .
2. Assign Nearest Courier

|  |
| --- |
|  |

1. Weight Category

|  |
| --- |
|  |

1. Login System

|  |
| --- |
|  |

1. Exit

|  |
| --- |
|  |