Indian Institute of Information Technology Vadodara

CS 261 Object Oriented Design & Programming Lab

< Course Instructure >

Dr. Ashish Phophalia

Dr. Novarun Deb

Project Report for End Sem (final) Evaluation

< Project Title >

City Delivery Management System

< Submitted by >

Tushar Id 201951163

Table of Contents

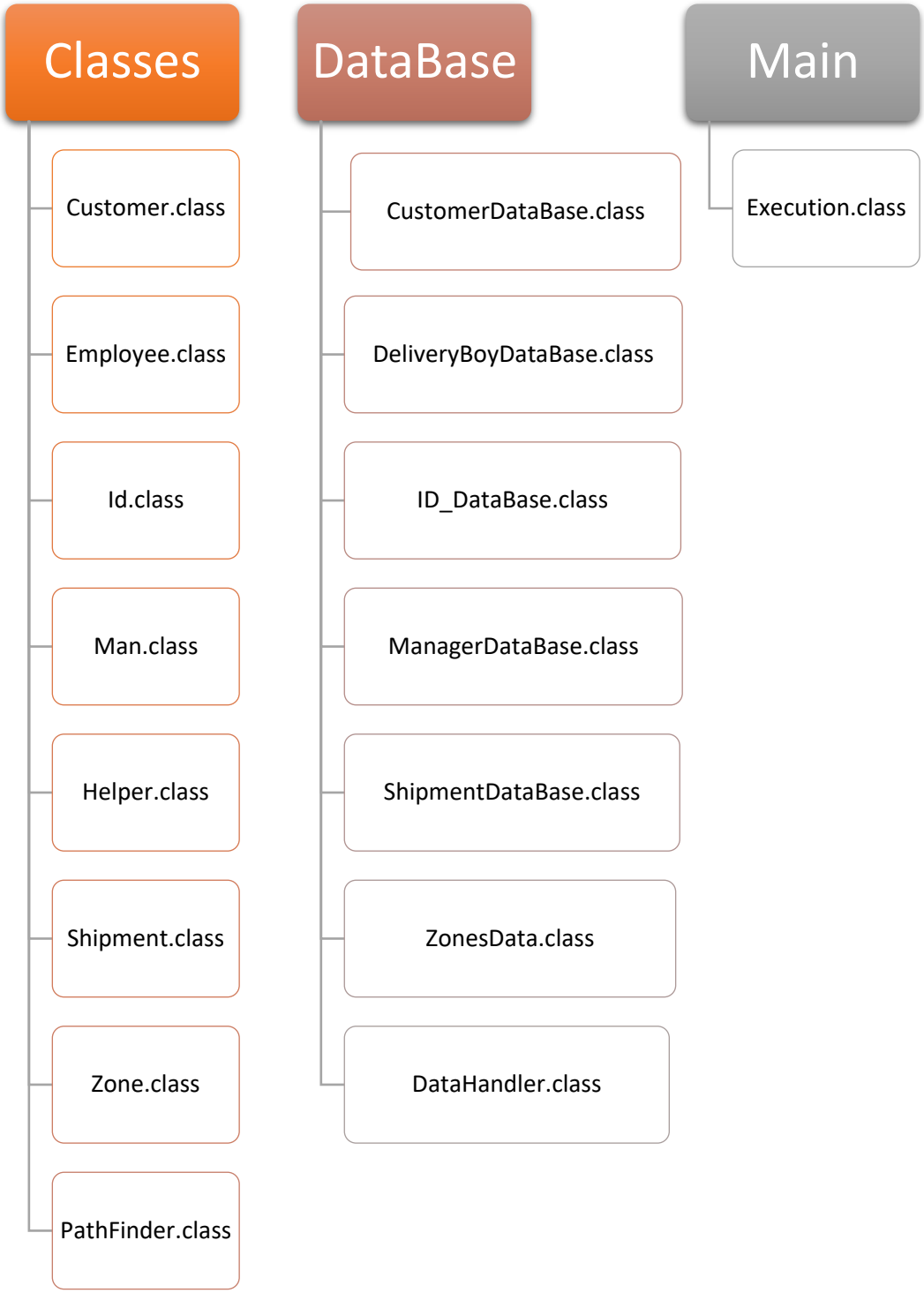
*Click on the name to go to respective section

Problem Statement

Solve the problem of managing all details and data for a company which is working for some delivery system within a city. We have to create a system which will reduce delivery cost and efficiently guide zones (about action they have to perform) and efficiently create a path for the shipment. Following are some major task to be completed

1. Create a system to manage user, employee and shipment related data, login system and creation of new accounts.
2. Create a system which will automatically find nearest zone to sender and receiver. After finding the nearest zone to sender (call it as source zone), Assign the order to delivery boy and inform him with sender's basic details for collecting the shipment.
3. Create a system which will calculate shortest path via zones from source zone to final delivery zone. Human mistakes are possible so create a system which allow to create a new path if somehow the shipment is moved to a wrong zone.
4. Create a system which will show zone manager to which zone they have to move the shipment with a unique id or they have to deliver it. Assign the order to a delivery boy for final delivery.

Project Hierarchy



*Classes, Data Base and Main are packages

Solution

- For managing data related to shipment, customer, employee and zone we have created their respective classes, with the required attributes and functionality. We have created classes named as Id, Shipment, Employee, Man(Abstract), Zone, Customer inside Classes package. There is one abstract Helper class, created for helping in address and date of birth collection. This class has static functions to collect data and return them in required pattern as string. This method checks whether provided date of birth is possible or not. All methods are static because we do not need an instance of it.
- For managing data items operations like reading the data from storage (from File), storing the data to storage, generation of integer id's, adding new element, searching a element by some attributes. We have made data base classes inside DataBase package. All this classes are abstract and all data items and methods are static. There is one class called DataHandler containing two functions which will allow us to read and save data without taking any headache of reading and saving order of different files. This class is also Abstract and methods are static.
- For implementing delivery system:
  We have divided the city in some no of zones with a soft boundary, it means that there is no hard boundary

defining a zone's area. A point will be in a zone if it is the nearest possible zone to that point.

Finding Nearest Zone:

To find nearest zone we will use geo-location coordinates provided by google maps, then we use a simple formula from 2 d geometry to calculate distance between two points. In the method we will pass the coordinates for which we have to find nearest zone. Zones have methods to return x and y coordinate of the zone.

THE DISTANCE FORMULA

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Pseudo Code For finding Nearest Zone

```
Find Nearest Zone (x, y, zones [])
{
        Distance []
        n = zones array length
        for i = 0 to n do
        {
                Distance [ i ] =
                        Calculate distance of x, y from zones [ i ]
        }
        Find min (Distance [])
        //min is the index of minimum Distance in the array
        Return zones [min]
}
```

A method for the above purpose is created within Shipment class called as findNearestZone(). (Click on method name to see java code)

Order Assignment:

Each zone is also divided in four parts and each delivery boy will handle all delivery and parcel collection of that part.

Pseudo Code For assigning order

```
Assigning Order (x, y, zoneX, zoneY)
{
        Coordinate X = x – zoneX
        Coordinate Y = y – zoneY

        If X >= 0 and Y >= 0
                Assign order to delivery boy [1]
        Else if X >= 0 and Y < 0
                Assign order to delivery boy [2]
        Else if X < 0 and Y >= 0
```

```
            Assign order to delivery boy [3]
        Else
            Assign order to delivery boy [4]

}
```

A method for the above purpose is created within Zone class called as assignOrder(). (Click on method name to see java code)

Shortest Path:

We have to find a shortest path from source zone to destination zone. To solve this, we will create a graph with zones as vertices and there is an edge between two vertices if there is a transportation of parcel between these two zones. Then we will use Dijkstra's shortest path algorithm to find shortest path between source zone and destination zone. A class PathFinder is created for managing graph of zones and path finding method. This is abstract class with all attributes and methods static.

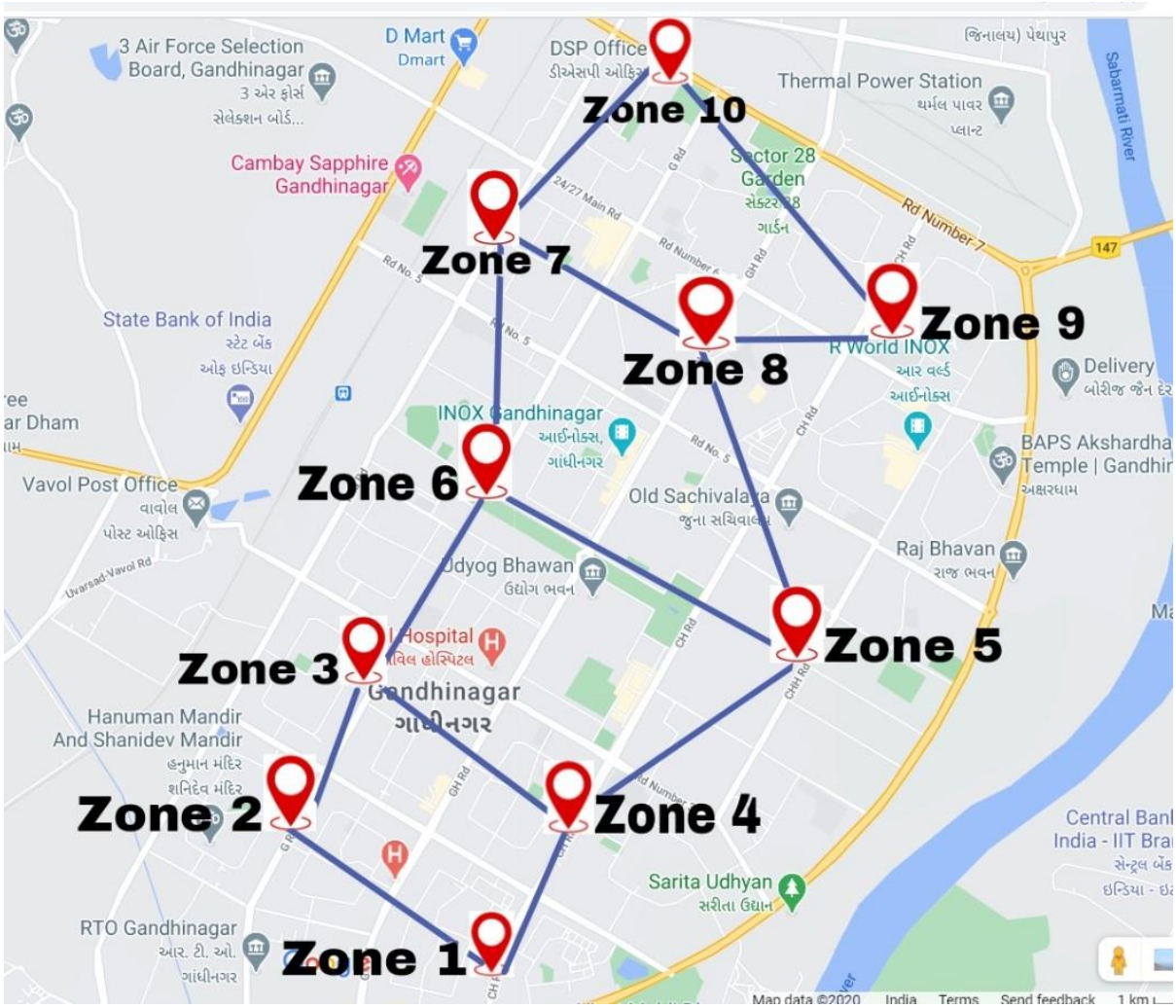
Pseudo Code For finding shortest path

```
getPath(start, end){
        InitializeSingleSource(V,start)
        Visited[no_vertices] = false
        For all vertices
        Do
            Shortest = Pick minimum distance vertex which
                        are not visited
                    For each v adjacent to Shortest
```

```
                    do
                        Relax(Shortest, v , weight)
        path = makeList() // ArrayList  with vertices on path
        return path
}
```

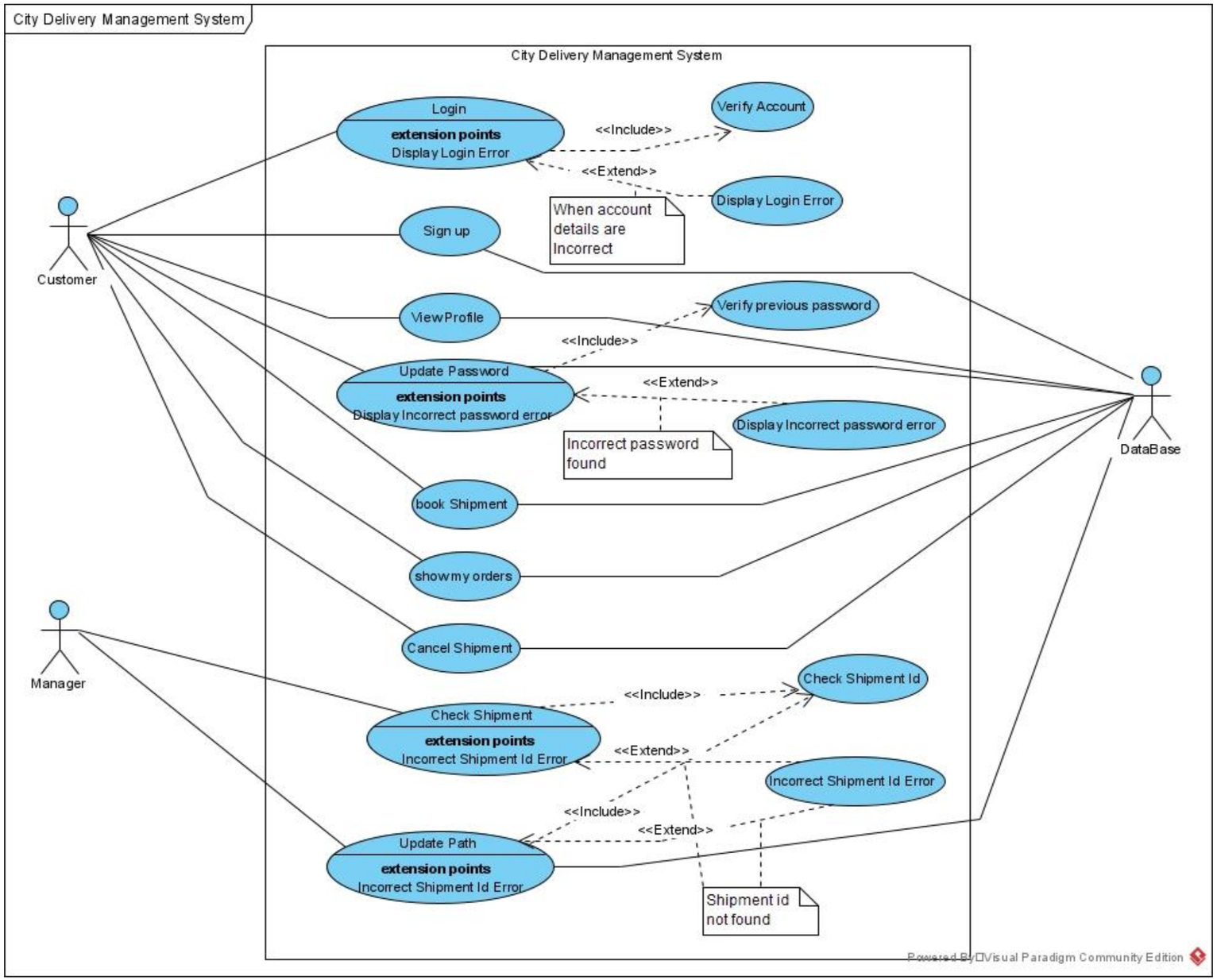
A method for the above is created within PathFinder class called as getPath(). *click to show java code

Example we have taken for Zones:

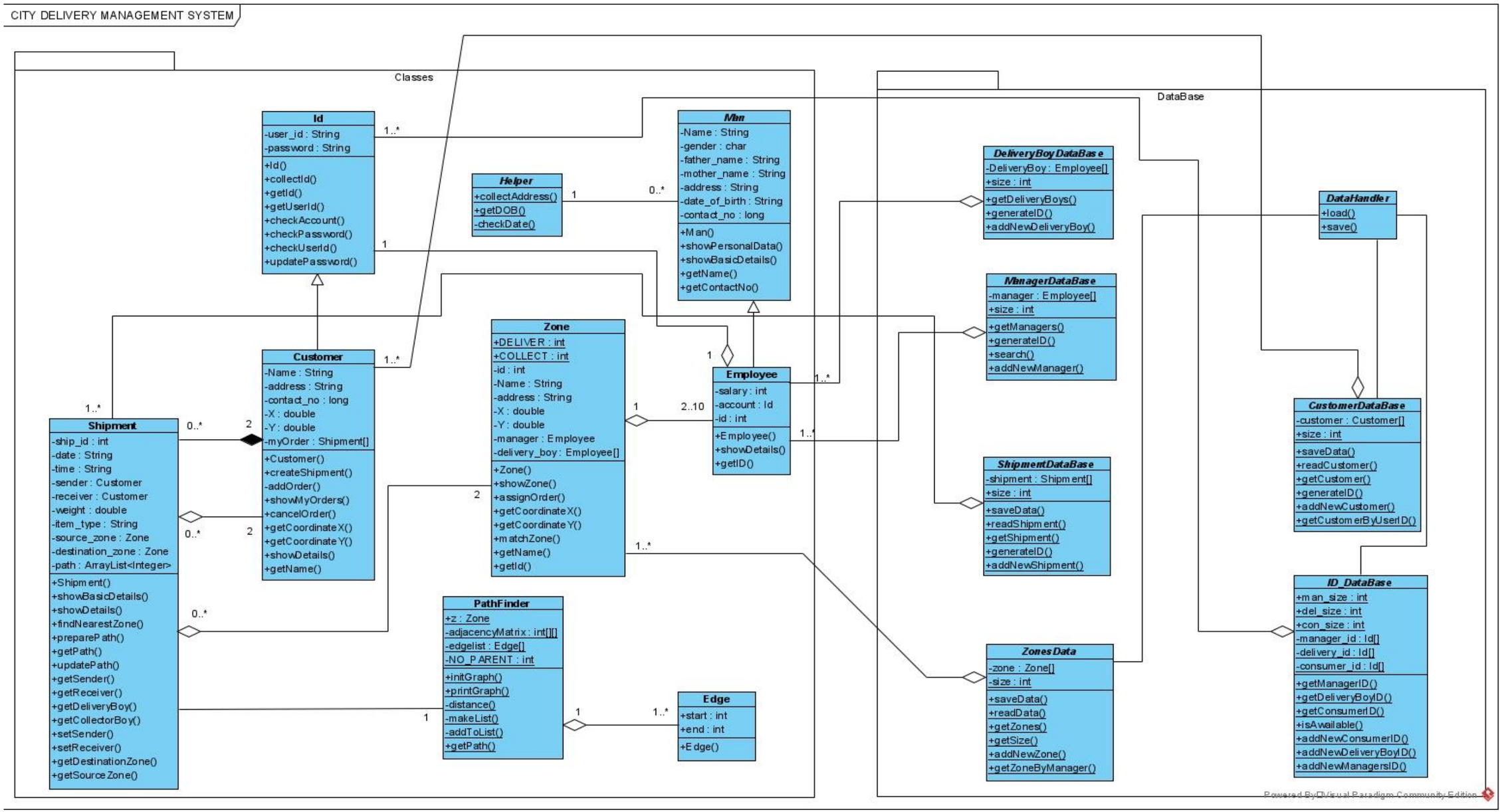


UML Diagrams

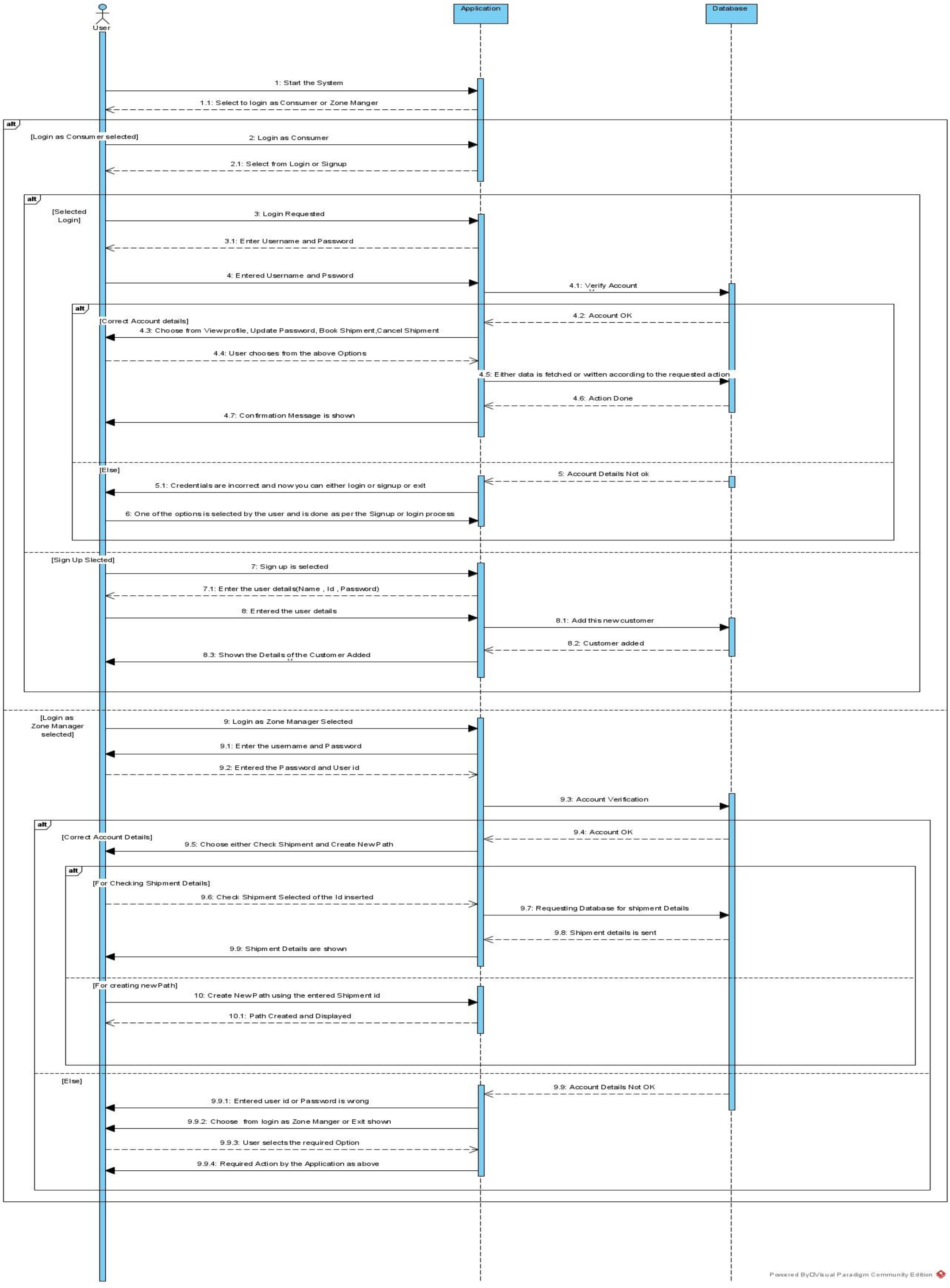
Use Case Diagram



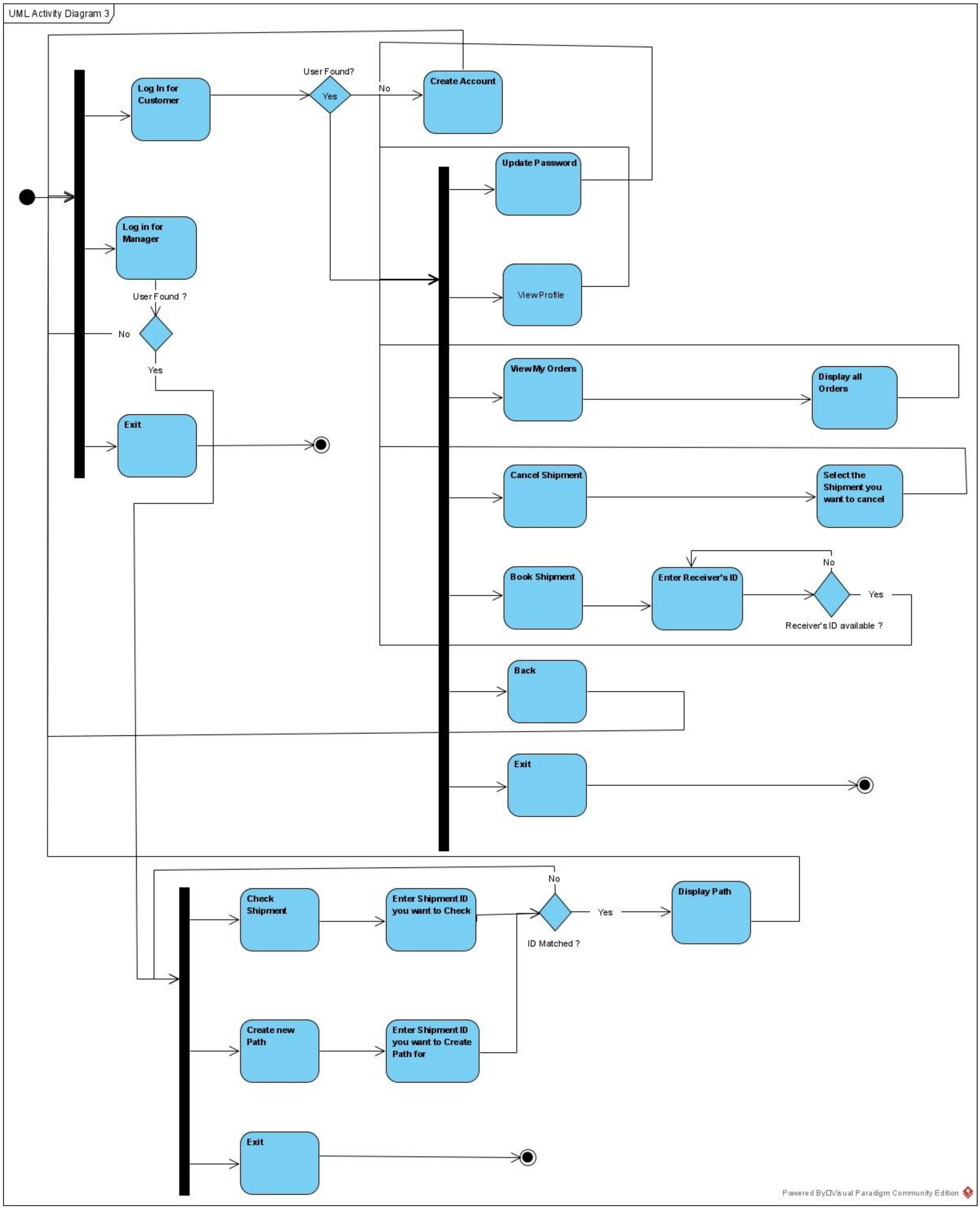
Class Diagram



Sequence Diagram:



Activity Diagram



Java Code of Project:

(Click on the class name to go to Code of that particular class)

1) Classes
    a) Id
    b) Shipment
    c) Man
    d) Customer
    e) Zone
    f) PathFinder
    g) Employee
2) DataBase
    a) ZonesData
    b) ManagerDataBase
    c) DeliveryBoyDataBase
    d) CustomerDataBase
    e) ID_DataBase
    f) ShipmentDataBase
    g) DataHandler
3) Main
    a) Execution
    b) CreateInitData

1. Id class

```java
package Classes;

import java.io.Serializable;
import java.util.Scanner;
import DataBase.ID_DataBase;

public class Id implements Serializable{

    private String user_id;
    private String password;

    public Id() {

    }

    public Id(String user_id, String password) {
        this.user_id = user_id;
        this.password = password;
    }

    //This method collect user id and password

    public void collectId() {
        Scanner sc = new Scanner(System.in);
        boolean notFirst = false;
        do {
            if(notFirst && !ID_DataBase.isAwailable(user_id)) {
                System.out.println("\nUser name not awailable\n");
            }
            notFirst = true;
            System.out.print("Enter user Id    :");

            user_id = sc.nextLine();
        }while(!ID_DataBase.isAwailable(user_id));
        System.out.print("Enter password   :");
        password = sc.nextLine();

    }
    //returns Id object

    public Id getId() {
        return new Id(user_id,password);
    }

    //returns user id

    public String getUserId() {
        return user_id;
    }

    //check user_id and password and return true
    //if it matches

    public boolean checkAccount(String user_id, String password) {
        if(user_id.equals(this.user_id)) {
            return checkPassword(password);
        }
        else {
            return false;
        }
    }
```

```
    //checks password and return true if matches

    public boolean checkPassword(String password)
    {
         return password.equals(this.password) ;
    }

    //checks user id and returns true if matches

    public boolean checkUserID(String user_id)
    {
         return user_id.equals(this.user_id) ;
    }

    //update password

    public void updatePassword()
    {
       Scanner sc = new Scanner(System.in);
       String newpassword;
       System.out.println("Enter your previous password");
       newpassword = sc.next();
       if(checkPassword(newpassword))
       {
            System.out.println("Enter new password");
            password = sc.next();
       }
       else
       {
            System.out.println("Incorrect Password");
       }

    }

}
```

2. Shipment class

```
package Classes;

import java.io.Serializable;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.ArrayList;

import DataBase.ZonesData;

public class Shipment implements Serializable{

    private int ship_id;             //Shipment id
    private String date;             //date of Order placed
    private String time;             //time of Order placed
    private Customer sender;         //Sender details
    private Customer receiver;       //Receiver details
    private double weight;           //weight of the shipment
    private String item_type;       //Type of ordered item for example "book"
    private Zone source_zone;        //Nearest zone to sender
    private Zone destination_zone;  //Nearest zone to receiver
    private ArrayList<Integer> path;

    //Creates a object with provided data items
```

```
    public Shipment(int ship_id, Customer sender, Customer receiver, double weight,
String item_type) {

        LocalTime time = LocalTime.now();
        LocalDate date = LocalDate.now();
        preparePath();
        this.sender = sender;
        this.receiver = receiver;
        this.ship_id = ship_id;
        this.item_type = item_type;
        this.time = time.toString().substring(0,
time.toString().lastIndexOf('.'));
        this.date = date.toString();
        this.weight = weight;
    }

    //Prints some important details

    public void showBasicDetails(int i) {
        System.out.println("-------------------------------------------------");
        System.out.println(i+".  Shipment Details");
        System.out.println("-------------------------------------------------");
        System.out.println("Shipment Id      : "+ship_id);
        System.out.println("Date of booking  : "+date);
        System.out.println("Item type        : "+item_type);
    }

    //Prints complete details of shipment

    public void showDetails() {
        System.out.println("-------------------------------------------------");
        System.out.println("Shipment Details");
        System.out.println("-------------------------------------------------");
        System.out.println("Shipment Id      : "+ship_id);
        System.out.println("Date of booking  : "+date);
        System.out.println("Time of booking  : "+time);
        System.out.println("Weight           : "+weight);
        System.out.println("Item type        : "+item_type);
        sender.showDetails("Sender");
        receiver.showDetails("Receiver");
        System.out.println("-------------------------------------------------");
    }

    //returns zone with minimum distance to coordinates x and y

    public Zone findNearestZone(double x ,double y, Zone zone[])
    {

        int min;
        int n = zone.length;
        double Distance[] = new double[n];

        //Calculating the distance of the coordinate from all the zones
        for(int i=0;i<n;i++)
        {
            Distance[i] = Math.sqrt(Math.pow((x-zone[i].getCoordinateX()),2) +
                    Math.pow((y-zone[i].getCoordinateY()),2));
        }

        //Finding the index of zone with minimum distance
        min=0;

        for(int i=1; i<n; i++)
        {
            if(Distance[i]<Distance[min])
```

```
                {
                    min=i;
                }
            }

            return zone[min];
        }

        public void preparePath() {

            source_zone =
findNearestZone(sender.getCoordinateX(),sender.getCoordinateY(),ZonesData.getZones()
);
            destination_zone =
findNearestZone(receiver.getCoordinateX(),receiver.getCoordinateY(),ZonesData.getZon
es());

            int start = 1+source_zone.getId()%100;
            int end = 1+destination_zone.getId()%100;

            path = PathFinder.getPath(start, end);
        }

        public void getPath(Zone z) {
            System.out.println(path);
            if(z == destination_zone) {
                System.out.println("\nFinal Destination \nDeliver Product...");
                System.out.println("Messege can be send to
"+(destination_zone.assignOrder(this, Zone.DELIVER)).getName());
                return;
            }
            int index = 1 + z.getId()%100;
            for(int i=0;i<path.size();i++) {
                if(index == path.get(i)) {
                    System.out.println("Send it to Zone-"+path.get(i+1));
                    return;
                }
            }
            System.out.println("Moved to wrong zone");
        }

        public void upadtePath(Zone z) {
            int start = 1 + z.getId()%100;
            int end = 1+destination_zone.getId()%100;

            path = PathFinder.getPath(start, end);
            System.out.println("Path Updated Successfully...");
            getPath(z);
        }

        public Customer getSender() {
            return sender;
        }

        public Customer getReceiver() {
            return receiver;
        }

        public void setSender(Customer c) {
            sender = c;
        }

        public void setReceiver(Customer c) {
            receiver = c;
        }
```

```
    public Zone getDestinationZone() {
         return destination_zone;
    }

    public Zone getSourceZone() {
         return source_zone;
    }
}
```

3. Zone class

```
package Classes;

import java.io.Serializable;
import java.util.*;

import DataBase.DeliveryBoyDataBase;
import DataBase.ID_DataBase;
import DataBase.ManagerDataBase;
import DataBase.ZonesData;

public class Zone implements Serializable{

    public static final int DELIVER=0;
    public static final int COLLECT=1;
    private int id;
    private String Name;
    private String address;
    private double X;
    private double Y;
    private Employee manager;
    private Employee delivery_boy[] = new Employee[4];

    //Create object with passed data items

    public Zone(String Name, String address, int id ,double X ,double Y,
              Employee manager, Employee db1, Employee db2, Employee db3,
              Employee db4) {
         this.address = address;
         this.manager = manager;
         this.delivery_boy[0] = db1;
         this.delivery_boy[1] = db2;
         this.delivery_boy[2] = db3;
         this.delivery_boy[3] = db4;
         this.id = id;
         this.X = X;
         this.Y = Y;
         this.Name = Name;
    }

    //Collect data from user and create a object

    public Zone() {
         Scanner sc = new Scanner(System.in);
         System.out.println("-----------------------------------------------------");
         System.out.println("Enter Zone Details");
         System.out.println("-----------------------------------------------------");
         System.out.print("Name          : ");
         Name = sc.nextLine();
         System.out.print("Address       : ");
         address = sc.nextLine();
         System.out.print("Coordinates   : ");
```

```
        System.out.print("X   : ");
        X = sc.nextDouble();
        System.out.print("Y   : ");
        Y = sc.nextDouble();
        System.out.println("\nEnter Manager's Details\n");
        manager = new Employee(ManagerDataBase.generateID());
        System.out.println("\nEnter Delivery Boy's Details\n");
        for(int i=0;i<4;i++) {
            delivery_boy[i] = new Employee(DeliveryBoyDataBase.generateID());
        }
        id = 2600 + ZonesData.getSize();
        ManagerDataBase.addNewManager(manager);
        ID_DataBase.addNewManagersID(manager.getID());
    }

    //Prints all details of the Zone

    public void showZone() {
        System.out.println("-------------------------------------------------");
        System.out.println("Zone Details");
        System.out.println("-------------------------------------------------");
        System.out.println("Name          : "+Name);
        System.out.println("Id            : "+id);
        System.out.println("Address       : "+address);
        System.out.println("Manager       : "+manager.getName());
        System.out.println("Delivery Boy  : "+delivery_boy[0].getName());
        for(int i=1;i<4;i++) {
            System.out.println("                "+delivery_boy[i].getName());
        }
    }

    public Employee assignOrder(Shipment s,int action) {
        Customer c;
        Employee e;
        if(action == DELIVER) {
            c = s.getReceiver();
        }
        else {
            c = s.getSender();
        }
        //change origin to zone center
        double x = c.getCoordinateX()-X;
        double y = c.getCoordinateY()-Y;
        //first quadrant
        if(x>0 && y>0) {
            e = delivery_boy[0];
        }
        else if(x<0 && y>0)//second quadrant
        {
            e = delivery_boy[1];
        }
        else if(x<0 && y<0)//third quadrant
        {
            e = delivery_boy[2];
        }
        else {//fourth quadrant
            e = delivery_boy[3];
        }
        return e;
    }

    public double getCoordinateX() {
        return X;
    }
```

```
    public double getCoordinateY() {
        return Y;
    }

    public boolean matchZone(Employee manager) {
        return manager.equals(this.manager);
    }

    public String getName() {
        return Name;
    }

    public int getId() {
        return id;
    }

    public Employee getManager()
    {
        return manager;
    }

    public Employee[] getDeliveryBoy()
    {
        return delivery_boy;
    }
}
```

4. Employee class

```
package Classes;

import java.io.Serializable;
import java.util.Scanner;

public class Employee extends Man implements Serializable{

    private int salary;
    private Id account;
    private int id;

    //collects employee details from user

    public Employee(int id) {
        super();
        Scanner sc = new Scanner(System.in);
        System.out.print("Salary          : ");
        this.salary = sc.nextInt();
        account.collectId();
        this.id=id;
    }

    //creates object with passed data items

    public Employee(int id, Id account, String Name, String father_name,
            String mother_name, char gender, String dob, long contact_no,
            String address, int salary) {

        super(Name, father_name, mother_name, gender, dob, contact_no, address);
        this.id = id;
        this.account = account;
        this.salary = salary;
    }
```

```
    //Prints all employee details

    public void showDetails() {
         System.out.println("-----------------------------------------------------");
         System.out.println("Employee Details");
         System.out.println("-----------------------------------------------------");
         System.out.println("Id              : "+id);
         System.out.println("User Id         : "+account.getUserId());
         showPersonalData();
         System.out.println("Salary          : "+salary+"\n");
    }

    public Id getID() {
         return account;
    }

}
```

5. Customer class

```
package Classes;

import java.io.Serializable;
import java.util.Scanner;
import DataBase.CustomerDataBase;
import DataBase.ShipmentDataBase;

public class Customer extends Id implements Serializable{

    private String Name;
    private String address;
    private long contact_no;
    private double X;
    private double Y;
    private Shipment myOrder[] = new Shipment[0];

    public Customer(String user_id, String password, String Name, String address,
long contact_no, double X, double Y)
    {
         super(user_id,password);
         this.X=X;
        this.Y=Y;
        this.contact_no = contact_no;
        this.address= address;
        this.Name = Name;
    }

    //collect customers data from user

    public Customer() {
         Scanner sc = new Scanner(System.in);
         System.out.println("-----------------------------------------------------");
         System.out.println("\t\tEnter Your Details");
         System.out.println("-----------------------------------------------------");
         collectId();
         System.out.print("Name            : ");
         Name = sc.nextLine();
         System.out.print("Contact No      : ");
         contact_no = sc.nextLong();
         while(contact_no/10000000000l < 6 && contact_no/10000000000l > 9) {
              System.out.println("Not a valid phone no!\nEnter a valid contact no");
              System.out.print("Contact No      : ");
```

```
            contact_no = sc.nextLong();
        }
        System.out.println("\nLocation Coordinates");
        System.out.print("X     : ");
        X = sc.nextDouble();
        System.out.print("Y     : ");
        Y = sc.nextDouble();
        sc.nextLine();
        System.out.println("Enter Address");
        address = Helper.collectAddress();
    }

    //this method will create or book a shipment and
    //Adds that to list

    public void createShipment() throws Exception{
        Scanner sc = new Scanner(System.in);
        int ship_id = 0;
        Customer receiver = null;
        double weight;
        String item_type;
        System.out.println("-------------------------------------------------");
        System.out.println("Enter Shipment Details");
        System.out.println("-------------------------------------------------");
        System.out.print("Receiver's User Id  : ");
        String user_id = sc.next();
        System.out.print("Weight              : ");
        weight = sc.nextDouble();
        System.out.print("Item Type           : ");
        item_type = sc.next();
        receiver = CustomerDataBase.getCustomerByUserID(user_id);
        while(receiver == null) {
            System.out.println("No customer with this user id \nEnter correct user
id");
            System.out.print("Receiver's User Id  : ");
            user_id = sc.next();
            receiver = CustomerDataBase.getCustomerByUserID(user_id);
        }
        ship_id = ShipmentDataBase.generateID();
        Shipment order = new Shipment(ship_id,this,receiver, weight, item_type);
        addOrder(order);
        System.out.println("\nYour Shipment Booked Successfully");
        System.out.println("\nShipment will be collected by"
                +"\n"+order.getSourceZone().assignOrder(order,
Zone.COLLECT).getName()+"\nContact no : "
                +order.getSourceZone().assignOrder(order,
Zone.COLLECT).getContactNo());

    }

    //this method adds order to list

    private void addOrder(Shipment order) {
        Shipment s[] = new Shipment[myOrder.length+1];
        System.arraycopy(myOrder, 0, s, 0, myOrder.length);
        s[myOrder.length] = order;
        myOrder = s;
        ShipmentDataBase.addNewShipment(order);
    }

    //This method will show all orders related to customer

    public void showMyOrders() {
        if(myOrder.length == 0) {
            System.out.println("Did't Found any Order");
```

```
            return;
        }
        for(int i =0;i<myOrder.length;i++) {
            myOrder[i].showBasicDetails(i+1);
        }
    }

    //This method deletes choosen order from the list

    public void cancelOrder() throws Exception{
        Scanner sc = new Scanner(System.in);
        showMyOrders();
        System.out.print("Enter which order you want to cancel : ");
        int ch = sc.nextInt();
        myOrder[ch-1] = myOrder[myOrder.length-1];
        Shipment s[] = new Shipment[myOrder.length-1];
        System.arraycopy(myOrder, 0, s, 0, myOrder.length-1);
        myOrder = s;
    }

    public double getCoordinateX()
    {
        return X;
    }

    public double getCoordinateY()
    {
        return Y;
    }

    //prints customer data

    public void showDetails(String s) {
        System.out.println("\n"+s+" Details      ");
        System.out.println("\nUser Id          : "+getUserId());
        System.out.println("Name             : "+Name);
        System.out.println("Contact No       : "+contact_no);
        System.out.println("Address          : "+address);
    }

    //returns name
    public String getName() {
        return Name;
    }
}
```

6. PathFinder class

```
package Classes;

import DataBase.ZonesData;
import java.util.ArrayList;
//==================================class
Edge=======================================//
class Edge{
    int start;
    int end;

    Edge(int s, int e) {
        start = s;
        end = e;
    }
}
```

```
//==================================class
PathFinder=======================================//
public class PathFinder {

    private static Zone z[] = ZonesData.getZones();
    public static int adjacencyMatrix[][];
    private static Edge edgelist[];
    private static final int NO_PARENT = -1;
    public static ArrayList<Integer> str = new ArrayList<Integer>();

    //***************<Constructor>*********************************//
    public static void initGraph(){

        edgelist = new Edge[13];

        edgelist[0] = new Edge(1,2);
        edgelist[1] = new Edge(2,3);
        edgelist[2] = new Edge(3,4);
        edgelist[3] = new Edge(1,4);
        edgelist[4] = new Edge(3,6);
        edgelist[5] = new Edge(4,5);
        edgelist[6] = new Edge(5,6);
        edgelist[7] = new Edge(6,7);
        edgelist[8] = new Edge(5,8);
        edgelist[9] = new Edge(7,8);
        edgelist[10] = new Edge(7,10);
        edgelist[11] = new Edge(8,9);
        edgelist[12] = new Edge(10,9);

        adjacencyMatrix = new int[10][10];

        for(int i=0;i<13;i++) {
            int s = edgelist[i].start;
            int e = edgelist[i].end;

            adjacencyMatrix[s-1][e-1]=adjacencyMatrix[e-1][s-1]=distance(s,e);
        }

    }
    ////////////////////////////////////////////////////////////////////
    public static void printGraph() {
        for(int i=0;i<10;i++) {
            for(int j=0;j<10;j++) {
                System.out.print(adjacencyMatrix[i][j]+"\t");
            }
            System.out.println();
        }
    }
    ////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////////////////
    private static int distance(int s, int e) {

        double x1 = z[s-1].getCoordinateX();
        double x2 = z[e-1].getCoordinateX();

        double y1 = z[s-1].getCoordinateY();
        double y2 = z[e-1].getCoordinateY();

        double a = Math.pow(x1-x2, 2)+Math.pow(y1-y2, 2);
        int dist = (int)(1000*Math.sqrt(a));

        return dist;
    }
```

```
    ////////////////////////////////////////////////////////////////////////////////
////////////////
    public static ArrayList<Integer> getPath(int startVertex,int end)
    {
        startVertex--;
        end--;
        int nVertices = adjacencyMatrix[0].length;
        //shortestDistances[i] will keep the shortest distance from src to i
        int[] shortestDistances = new int[nVertices];
        boolean[] added = new boolean[nVertices];
        //pahle sabko infinity banaya
        for (int vertexIndex = 0; vertexIndex < nVertices; vertexIndex++)
        {
            shortestDistances[vertexIndex] = Integer.MAX_VALUE;
            added[vertexIndex] = false;
        }
        //distance from itself will be zero
        shortestDistances[startVertex] = 0;
        //parent array path store karega
        int[] parents = new int[nVertices];
        // starting vertex ka koi parent nhi h....
        parents[startVertex] = NO_PARENT;
        //finding shortest path for all those
        for (int i = 1; i < nVertices; i++)
        {

            // Pick the minimum distance vertex
            // from the set of vertices not yet
            // processed. nearestVertex is
            // always equal to startNode in
            // first iteration.
            int nearestVertex = -1;
            int shortestDistance = Integer.MAX_VALUE;
            for (int vertexIndex = 0;
                    vertexIndex < nVertices;
                    vertexIndex++)
            {
                if (!added[vertexIndex] &&
                    shortestDistances[vertexIndex] <
                    shortestDistance)
                {
                    nearestVertex = vertexIndex;
                    shortestDistance = shortestDistances[vertexIndex];
                }
            }

            // Mark the picked vertex as
            // processed
            added[nearestVertex] = true;

            // Update dist value of the
            // adjacent vertices of the
            // picked vertex.
            for (int vertexIndex = 0;
                    vertexIndex < nVertices;
                    vertexIndex++)
            {
                int edgeDistance = adjacencyMatrix[nearestVertex][vertexIndex];

                if (edgeDistance > 0
                    && ((shortestDistance + edgeDistance) <
                        shortestDistances[vertexIndex]))
                {
                    parents[vertexIndex] = nearestVertex;
                    shortestDistances[vertexIndex] = shortestDistance +
```

```
                                                                edgeDistance;
                    }
               }
          }

          makeList(startVertex, shortestDistances, parents,end);
          ArrayList<Integer> path = (ArrayList<Integer>) str.clone();
          str.clear();
          return path;
     }
     ////////////////////////////////////////////////////////////////////////////////
/////////
     private static void makeList(int startVertex, int[] distances, int[]
parents,int end)
     {
          int nVertices = distances.length;
          //System.out.print("Zone\t \tDistance\tPath");

          for (int vertexIndex = 0; vertexIndex < nVertices; vertexIndex++)
          {
               if (vertexIndex != startVertex && vertexIndex==end)
               {
                    addToList(vertexIndex, parents);
               }
          }
     }
     ////////////////////////////////////////////////////////////////////////////////
///////////
     private static void addToList(int currentVertex, int[] parents)
     {

          // Base case : Source node has
          // been processed
          if (currentVertex == NO_PARENT)
          {
               return;
          }
          addToList(parents[currentVertex], parents);

          str.add((currentVertex+1));
     }
     ////////////////////////////////////////////////////////////////////////////////
/////////////
}
```

7. Man class

```
package Classes;

import java.io.Serializable;
import java.util.*;

//This class maintain and handle human related data

public abstract class Man implements Serializable{

     private String Name;
     private char gender;
     private String fathers_name;
     private String mothers_name;
     private String address;
     private String date_of_birth;
     private long contact_no;
```

```
    //Collect personal Data and creates a object

    public Man() {
        Scanner sc = new Scanner(System.in);
        System.out.println("-------------------------------------------------");
        System.out.println("Enter your personal details");
        System.out.println("-------------------------------------------------");
        System.out.print("Name            : ");
        Name = sc.nextLine();
        System.out.print("Father's Name   : ");
        fathers_name = sc.nextLine();
        System.out.print("Mother's Name   : ");
        mothers_name = sc.nextLine();
        System.out.print("Gender M/F/T    : ");
        gender = sc.next().charAt(0);
        System.out.println("Date of Birth   : ");
        date_of_birth = Helper.getDOB(sc);
        System.out.print("Contact No      : ");
        contact_no = sc.nextLong();
        System.out.println("Address         : ");
        address = Helper.collectAddress();
    }

    //Crates a object with provided data items

    public Man(String Name, String father_name, String mother_name,
            char gender, String dob, long contact_no, String address)
    {
        this.Name = Name;
        this.fathers_name = father_name;
        this.mothers_name = mother_name;
        this.gender = gender;
        this.date_of_birth = dob;
        this.contact_no = contact_no;
        this.address = address;
    }

    //Shows all personal Data

    public void showPersonalData() {
        System.out.println("Personal Detail");
        System.out.println("Name            : "+Name);
        System.out.println("Father's Name   : "+fathers_name);
        System.out.println("Mother's Name   : "+mothers_name);
        System.out.println("Gender          : "+gender);
        System.out.println("Date of Birth   : "+date_of_birth);
        System.out.println("Contact No      : "+contact_no);
        System.out.println("Address         : "+address);
    }

    //Shows some basic details

    public void showBasicDetails() {
        System.out.println("-------------------------------------------------");
        System.out.println("Name       : "+Name);
        System.out.println("Contact No : "+contact_no);
        System.out.println("-------------------------------------------------");
    }

    //returns Name

    public String getName() {
        return Name;
    }
```

```
      //returns Contact No

      public long getContactNo() {
           return contact_no;
      }
}
```

8. Helper class

```
package Classes;

import java.util.Scanner;

public abstract class Helper {

    //This method will collect all information for address from user
    //convert it to a string and then returns it

    public static String collectAddress() {

         String plot_no;
         String street;
         String city;
         String state;
         String country;

         Scanner scanner = new Scanner(System.in);

         System.out.print("Plot no    : ");
         plot_no = scanner.nextLine();
         System.out.print("Street     : ");
         street = scanner.nextLine();
         System.out.print("City       : ");
         city = scanner.nextLine();
         System.out.print("State      : ");
         state = scanner.nextLine();
         System.out.print("Country    : ");
         country = scanner.nextLine();

         String adrs = plot_no + ", " + street + ", " + city + ", " + state + ", "
+ country;
         return adrs;
    }

    //Collect a date and return as String

     public static String getDOB(Scanner sc) {

         int day;
         int month;
         int year;
         do {
              System.out.print("Day             : ");
              day = sc.nextInt();
              System.out.print("Month (int)     : ");
              month = sc.nextInt();
              System.out.print("Year            : ");
              year = sc.nextInt();
         }while(!checkDate(day,month,year));

         return (day+"/"+month+"/"+year);
     }
```

```
    //Checks weather date is correct or not

    private static boolean checkDate(int day, int month, int year) {

        boolean isLeap = false;
        boolean isCorrect = false;
        //checking leap year
        if(year % 400 == 0 ) {
            isLeap = true;
        }
        else if(year % 100 == 0) {
            isLeap = false;
        }
        else if(year % 4 == 0) {
            isLeap = true;
        }
        //checking date
        if(month > 12 || day > 31) {
            isCorrect = false;
        }
        else if(month == 2) {
            if( isLeap && (day <= 29) ) {
                isCorrect = true;
            }
            else if(!isLeap && (day <= 28)) {
                isCorrect = true;
            }
        }
        else if(month < 8) {
            if(month % 2 == 1 && day <= 31) {
                isCorrect = true;
            }
            else if(day <= 30) {
                isCorrect = true;
            }
        }
        else{
            if(month % 2 == 0 && (day <= 31)) {
                isCorrect = true;
            }
            else if(day <= 30) {
                isCorrect = true;
            }
        }

        if(!isCorrect) {
            System.out.println("-----------------------------------------------
-");
            System.out.println("! Invalid Date \nPlease enter correct date");
            System.out.println("-----------------------------------------------
-");
        }
        return isCorrect;
    }

}
```

- **DataHandler class**

```
package DataBase;

import Classes.PathFinder;
```

```
public abstract class DataHandler
{
  public static void load()
  {
       ZonesData.readData();
       CustomerDataBase.readCustomer();
       PathFinder.initGraph();
       ShipmentDataBase.readShipment();
  }

  public static void save()
  {
       CustomerDataBase.saveData();
       ShipmentDataBase.saveData();
       ZonesData.saveData();
  }

}
```

- **CustomerDataBase class**

```
package DataBase;

import Classes.Customer;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.time.LocalDate;

public abstract class CustomerDataBase {

    public static int size = 0;
    private static Customer customer[] = new Customer[0];


    public static void saveData()
    {
        try
        {
           FileOutputStream fileOut = new FileOutputStream("fileCustomer.txt");

           ObjectOutputStream objOut = new ObjectOutputStream(fileOut);

           for(int k=0; k<size; k++)
           {objOut.writeObject(customer[k]);}

           objOut.close();
        }
        catch (Exception e)
         {
           e.getStackTrace();
        }
    }

    public static void readCustomer()
    {
         FileInputStream fis=null;
         ObjectInputStream dis=null;

         try
```

```
        {
             fis = new FileInputStream("fileCustomer.txt");
             dis = new ObjectInputStream(fis);

             for(int i=0; i<Integer.MAX_VALUE; i++)
             {
                  addNewCustomer((Customer)dis.readObject());
             }
        }
        catch ( ClassNotFoundException  e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
             //error occur when no more elements in file
        }
        finally
        {
             try
             {
                  dis.close();
                  fis.close();
             }
             catch (IOException e)
             {
                  e.printStackTrace();
             }
        }

        for(int k=0; k<size; k++)
        {
             ID_DataBase.addNewConsumerID(customer[k].getId());
        }

    }


    public static Customer[] getCustomer()
    {
         return customer;
    }

    public static int generateID()
    {
         LocalDate d = LocalDate.now();
         int id = 10000*(Integer.parseInt(d.toString().substring(0,4)));
         return id + customer.length + 5000;
    }

    public static void addNewCustomer(Customer newcustomer)
    {
         Customer c[] = new Customer[size+1];
         System.arraycopy(customer, 0, c, 0, size);
         c[size] = newcustomer;
         customer = c;

         size++;

    }

    public static Customer getCustomerByUserID(String user_id) {

         for(int i=0;i<customer.length;i++) {
              if(customer[i].checkUserID(user_id)) {
```

```
                    return customer[i];
               }
          }

          return null;
     }

}
```

- **ManagerDataBase class**

```
package DataBase;

import java.time.LocalDate;
import Classes.Employee;

public abstract class ManagerDataBase {

     public static int size = 0;
     private static Employee manager[] = new Employee[0];

     public static Employee[] getManagers() {
          return manager;
     }

     public static int generateID() {
          LocalDate d = LocalDate.now();
          int id = 10000*(Integer.parseInt(d.toString().substring(0,4)));
          return id + manager.length;
     }

     //search manager by user id

     public static Employee search(String user_id, String password) {
          for(int i=0;i<manager.length;i++) {
               if(manager[i].getID().checkAccount(user_id, password)) {
                    return manager[i];
               }
          }
          return null;
     }

     public static void addNewManager(Employee n) {
          Employee z[] = new Employee[size+1];
          System.arraycopy(manager, 0, z, 0, size);
          z[size] = n;
          manager = z;

          size++;
     }

}
```

- **DeliveryBoyDataBase class**

```
package DataBase;

import java.time.LocalDate;
import Classes.Employee;
```

```
public abstract class DeliveryBoyDataBase{

  public static int size = 0;
  private static Employee DeliveryBoy[] = new Employee[0];

  public static Employee[] getDeliveryBoys ()
  {
    return DeliveryBoy;
  }

  public static int generateID ()
  {
    LocalDate d = LocalDate.now ();
    int id = 10000 * (Integer.parseInt (d.toString ().substring (0, 4)));
    return id + DeliveryBoy.length + 1000;
  }

  public static void addNewDeliveryBoy (Employee n)
  {

    Employee z[] = new Employee[size + 1];
    System.arraycopy (DeliveryBoy, 0, z, 0, size);
    z[size] = n;
    DeliveryBoy = z;

    size++;

  }

}
```

- **ID_DataBase class**

```
package DataBase;

import Classes.Id;

public abstract class ID_DataBase {

    public static int man_size = 0;
    public static int con_size = 0;
    public static int del_size = 0;
    private static Id managers_id[] = new Id[0];
    private static Id deliveryboy_id[] = new Id[0];
    private static Id consumer_id[] = new Id[0];


    public static Id[] getManagersID() {
        return managers_id;
    }

    public static Id[] getDeliveryBoyID() {
        return deliveryboy_id;
    }

    public static Id[] getConsumerID() {
        return consumer_id;
    }

    public static void addNewManagersID(Id manager) {
        Id id[] = new Id[man_size+1];
        System.arraycopy(managers_id, 0, id, 0, man_size);
```

```
            id[man_size] = manager;
            managers_id = id;

            man_size++;
        }

        public static void addNewDeliveryBoyID(Id delivery_boy) {
            Id id[] = new Id[del_size+1];
            System.arraycopy(deliveryboy_id, 0, id, 0, del_size);
            id[del_size] = delivery_boy;
            deliveryboy_id = id;

            del_size++;
        }

        public static void addNewConsumerID(Id consumer) {
            Id id[] = new Id[con_size+1];
            System.arraycopy(consumer_id, 0, id, 0, con_size);
            id[con_size] = consumer;
            consumer_id = id;

            con_size++;
        }

        //checks availability of user id
        //returns true if it is available

        public static boolean isAwailable(String user_id) {

            for(int i=0 ; i<consumer_id.length ; i++) {
                if(consumer_id[i].checkUserID(user_id)) {
                    return false;
                }
            }

            for(int i=0 ; i<deliveryboy_id.length ; i++) {
                if(deliveryboy_id[i].checkUserID(user_id)) {
                    return false;
                }
            }

            for(int i=0 ; i<managers_id.length ; i++) {
                if(managers_id[i].checkUserID(user_id)) {
                    return false;
                }
            }

            return true;
        }

}
```

- **ShipmentDataBase class**

```
package DataBase;

import java.time.LocalDate;

import Classes.Shipment;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
```

```
import java.io.ObjectOutputStream;

public abstract class ShipmentDataBase {

    public static int size = 0;

    private static Shipment shipment[] = new Shipment[0];



    public static void saveData()
    {
    try
    {
        FileOutputStream fileOut = new FileOutputStream("fileShipment.txt");

        // Creates an ObjectOutputStream
        ObjectOutputStream objOut = new ObjectOutputStream(fileOut);

        // Writes objects to the output stream

        for(int k=0; k<size; k++)
        {objOut.writeObject(shipment[k]);}

        objOut.close();

    }
    catch (Exception e)
     {
        e.getStackTrace();
    }
     }

     public static void readShipment()
     {
          FileInputStream fis=null;
          ObjectInputStream dis=null;

          try
          {
               fis = new FileInputStream("fileShipment.txt");
               dis = new ObjectInputStream(fis);

               for(int i=0;i<Integer.MAX_VALUE;i++)
               {
                    addNewShipment((Shipment)dis.readObject());
               }
          }
          catch ( ClassNotFoundException  e)
          {
              e.printStackTrace();
          }
          catch (IOException e)
          {
               //error occur when no more elements in file
          }
          finally
          {
               try
               {
                    dis.close();
                    fis.close();
               }
               catch (IOException e)
               {
```

```
                    e.printStackTrace();
               }
          }

          for(int i=0; i<size; i++)
          {

     shipment[i].setSender(CustomerDataBase.getCustomerByUserID(shipment[i].getSende
r().getUserId()));

     shipment[i].setReceiver(CustomerDataBase.getCustomerByUserID(shipment[i].getRec
eiver().getUserId()));
          shipment[i].preparePath();
          }
     }

     public static Shipment[] getShipment()
     {
          return shipment;
     }

     public static Shipment getShipment(int id)
     {
          if((id/1000)%10 == 3)
          {
               if(id%1000 < shipment.length)
               {
                    return shipment[(id%1000) - 1];
               }
          }
          return null;
     }

     public static int generateID()
     {
          LocalDate d = LocalDate.now();
          int id = 10000*(Integer.parseInt(d.toString().substring(0,4)));
          return id + shipment.length + 3000;
     }

     public static void addNewShipment(Shipment newshipment)
     {
          Shipment s[] = new Shipment[size+1];
          System.arraycopy(shipment, 0, s, 0, size);
          s[size] = newshipment;
          shipment = s;

          size++;
     }

}
```

- **ZonesData class**

```
package DataBase;

import Classes.Zone;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

```
import Classes.Employee;

public abstract class ZonesData {

    private static int size = 0;
    private static Zone zone[] = new Zone[0];

    public static void saveData()
    {
        try
        {
           FileOutputStream fileOut = new FileOutputStream("fileZone.txt");

           ObjectOutputStream objOut = new ObjectOutputStream(fileOut);

           for(int k=0; k<size; k++)
           {objOut.writeObject(zone[k]);}

           objOut.close();
        }
        catch (Exception e)
         {
           e.getStackTrace();
        }
    }

    public static void readData()
    {
        FileInputStream fis=null;
        ObjectInputStream dis=null;

        try
        {
            fis = new FileInputStream("fileZone.txt");
            dis = new ObjectInputStream(fis);

            for(int i=0;i<Integer.MAX_VALUE;i++)
            {
                addNewZone((Zone)dis.readObject());
                ManagerDataBase.addNewManager(zone[i].getManager());

    DeliveryBoyDataBase.addNewDeliveryBoy(zone[i].getDeliveryBoy()[0]);
            }
        }

        catch ( ClassNotFoundException  e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
            //error occur when no more elements in file
        }
        finally
        {

            try
            {
                dis.close();
                fis.close();
            }
            catch (IOException e)
            {
                e.printStackTrace();
```

```
                }
            }

            for(int k=0; k<size; k++ )
            {

        ID_DataBase.addNewManagersID(ManagerDataBase.getManagers()[k].getID());

        ID_DataBase.addNewDeliveryBoyID(DeliveryBoyDataBase.getDeliveryBoys()[k].getID(
));
            }
        }

        public static Zone [] getZones()
        {
            return zone;
        }

        public static int getSize()
        {
            return size;
        }

        public static void addNewZone(Zone newzone)
        {
            Zone z[] = new Zone[size+1];
            System.arraycopy(zone, 0, z, 0, size);
            z[size] = newzone;
            zone = z;
            size++;

        }
        //return zone where provided manager is posted
        public static Zone getZoneByManager(Employee manager) {
            for(int i=0;i<zone.length;i++) {
                if(zone[i].matchZone(manager)) {
                    return zone[i];
                }
            }
            return null;
        }
}
```

- **Execution class**

```
package Main;

import DataBase.*;
import Classes.*;
import java.util.Scanner;

public class Execution {

    static Scanner sc = new Scanner(System.in);

    public static void main(String[] args)
    {
        DataHandler.load();

        System.out.println("\n\t\tWellcome to our company\n");
        MainUI();
    }
```

```
public static void MainUI()
{

     int ch=0;
     //String user_id, password;
     System.out.println("================================================");
     System.out.println("1. Login for Consumer");
     System.out.println("2. Login for Zone Manager");
     System.out.println("3. Exit");
     System.out.print("Enter your choice : ");
     try {
          ch = sc.nextInt();
     }catch(Exception e) {
          System.out.println("Please Enter a integer no !");
          MainUI();
     }
     switch(ch) {
     case 1:
          LoginCustomerUI();
          MainUI();
          break;
     case 2:
          LoginForZoneManager();
          MainUI();
          break;
     case 3:
          DataHandler.save();
          System.exit(0);
          break;
     default :
          System.out.println("Invalid Choice");
          MainUI();
     }
}

public static void LoginCustomerUI() {
     int ch=0;
     String user_id, password;
     System.out.println("================================================");
     System.out.println("1. Login");
     System.out.println("2. Sign up");
     System.out.println("3. Back");
     System.out.println("4. Exit");
     System.out.print("Enter your choice : ");
     try {
          ch = sc.nextInt();
     }catch(Exception e) {
          System.out.println("Please enter a integer no !");
          LoginCustomerUI();
     }
     System.out.println("------------------------------------------------");

     switch(ch) {
     case 1:
          System.out.print("\nEnter your User id   : ");
          user_id = sc.next();
          System.out.print("Enter your Password  : ");
          password = sc.next();
          Customer c = CustomerDataBase.getCustomerByUserID(user_id);
          if(c == null) {
               System.out.println("\nInvalid user id");
               hold();
               LoginCustomerUI();
          }
```

```
            if(c.checkAccount(user_id, password)) {
                CustomerUI(c);
            }else {
                System.out.println("Incorrect user id or password");
                LoginCustomerUI();
            }

            hold();

            LoginCustomerUI();
            break;
        case 2:
            try {
                Customer n = new Customer();
                CustomerDataBase.addNewCustomer(n);
            }catch(Exception e) {
                System.out.println("Incorrect Data format!");
            }

            hold();

            LoginCustomerUI();
            break;
        case 3:
            MainUI();
            break;
        case 4:
            DataHandler.save();
            System.exit(0);
            break;
        default :
            System.out.println("Invalid Choice");
            hold();
            LoginCustomerUI();
        }
    }

    public static void CustomerUI(Customer c) {
        int ch=0;
        System.out.println("================================================");
        System.out.println("\t\tHello Mr/Mrs "+c.getName());
        System.out.println("1. Profile");
        System.out.println("2. Update Password");
        System.out.println("3. Book Shipment");
        System.out.println("4. Cancel Shipment");
        System.out.println("5. Show my orders");
        System.out.println("6. Back");
        System.out.println("7. Exit");
        System.out.print("Enter your choice : ");
        try {
            ch = sc.nextInt();
        }catch(Exception e) {
            System.out.println("Please Enter a integer no !");
            CustomerUI(c);
        }
        System.out.println("------------------------------------------------");

        switch(ch) {
        case 1:
            c.showDetails("My");
            hold();
            CustomerUI(c);
            break;
        case 2:
            c.updatePassword();
```

```
                hold();
                CustomerUI(c);
                break;
           case 3:
                try {
                     c.createShipment();
                } catch (Exception e) {
                     System.out.println(e);
                }
                hold();
                CustomerUI(c);
                break;
           case 4:
                try {
                     c.cancelOrder();
                } catch (Exception e) {
                     System.out.println("No order with this no!");
                }
                hold();
                CustomerUI(c);
                break;
           case 5:
                System.out.println("\t\tYour Orders");
                c.showMyOrders();
                hold();
                CustomerUI(c);
                break;
           case 6:
                LoginCustomerUI();
                break;
           case 7:
                DataHandler.save();
                System.exit(0);
                break;
           default:
                System.out.println("Invalid Choice");
                hold();
                CustomerUI(c);
           }
      }

      public static void LoginForZoneManager() {
           System.out.println("================================================");
           System.out.print("User Id          : ");
           sc.nextLine();
           String user_id = sc.nextLine();
           System.out.print("Password         : ");
           String password = sc.nextLine();
           Employee manager = ManagerDataBase.search(user_id, password);
           if(manager == null) {
                System.out.println("\nInvalid user id or password\n");
                MainUI();
           }else {
                ZoneUI(manager);
           }
      }

      public static void ZoneUI(Employee manager) {

           Zone z = ZonesData.getZoneByManager(manager);
           System.out.println("================================================");
           System.out.println("\t\t"+z.getName());
           System.out.println("================================================");
           System.out.println("1. Check Shipment");
           System.out.println("2. Create New Path");
```

```
            System.out.println("3. Back");
            System.out.println("4. Exit");
            System.out.print("Enter your choice  : ");
            int ch = sc.nextInt();
            int ship_id;
            Shipment s;
            switch(ch) {
            case 1:
                 System.out.print("Shipment Id  : ");
                 ship_id = sc.nextInt();
                 s = ShipmentDataBase.getShipment(ship_id);
                 if(s == null) {
                      System.out.println("\nShipment not found\n");
                 }else {
                      s.getPath(z);
                 }
                 hold();
                 ZoneUI(manager);
                 break;
            case 2:
                 System.out.print("Shipment Id  : ");
                 ship_id = sc.nextInt();
                 s = ShipmentDataBase.getShipment(ship_id);
                 if(s == null) {
                      System.out.println("\nShipment not found\n");
                 }else {
                      s.upadtePath(z);
                 }
                 hold();
                 ZoneUI(manager);
                 break;
            case 3:
                 MainUI();
                 break;
            case 4:
                 DataHandler.save();
                 System.exit(0);
                 break;
            default:
                 System.out.println("Invalid choice");
                 hold();
                 ZoneUI(manager);
            }
       }

       public static void hold() {
            System.out.println("\nPress Enter to continue");
            sc.nextLine();
            sc.nextLine();
       }

}
```

- **CreateInitData**

```
package Main;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

```
import Classes.*;
import DataBase.*;

public class CreateInitData {

    private static Zone zone[] = new Zone[10];
    private static Employee manager[] = new Employee[10];
    private static Employee DeliveryBoy[] = new Employee[10];
    private static Id id[] = new Id[10];
    private static Id id1[] = new Id[10];
    private static Customer customer[] = new Customer[10];
    private static Shipment shipment[] = new Shipment[10];

    public static void main(String[] args) {
        id[0] = new Id("Shyam","Shyam123");
        id[1] = new Id("Aziz","Aziz123");
        id[2] = new Id("Dashrath","Dashrath123");
        id[3] = new Id("Ajay","Ajay123");
        id[4] = new Id("Nayan","Nayan123");
        id[5] = new Id("Hardik","Hardik123");
        id[6] = new Id("Aman","Aman123");
        id[7] = new Id("Axar","Axar123");
        id[8] = new Id("Rahul","Rahul123");
        id[9] = new Id("Himesh","Himesh123");

        id1[0] = new Id("Ankush","Ankush123");
        id1[1] = new Id("Jasabhai","Jasabhai123");
        id1[2] = new Id("Fatehsingrao","Fatehsingrao123");
        id1[3] = new Id("Ranjit","Ranjit123");
        id1[4] = new Id("Anna","Anna123");
        id1[5] = new Id("Susan","Susan123");
        id1[6] = new Id("Khandubhai","Khandubhai123");
        id1[7] = new Id("Raghav","Raghav123");
        id1[8] = new Id("Mahendra","Mahendra123");
        id1[9] = new Id("Vallabh","Vallabh123");

        DeliveryBoy[0] =
                  new Employee (20200000, id1[0], "Ankush Kumar", "Ram Kumar",
                        "Ramvati Sinha", 'M', "25/02/2002",9128985987l,
"Gandhinagar",
                        10000);

                DeliveryBoy[1] =
                  new Employee (20200001, id1[1], "Jasabhai Barad", "Prakash
Brahmbhatt",
                        "Ramya AKA Divya", 'M', "1/12/1990", 9412384672l,
"Gandhinagar",
                        10000);

                DeliveryBoy[2] =
                  new Employee (20200002, id1[2], "Fatehsinghrao Gaekwad",
"Jaisukh lal Hathi",
                        "Mimi Chakraborty", 'M', "1/12/1996", 932741182681,
"Gandhinagar", 10000);

                DeliveryBoy[3] =
                  new Employee (20200003, id1[3], "Ranjitsinh Pratapsinh Gaekwad",
"Somjibhai Damor",
                        "Nusrat Jahan", 'M', "5/1/1996", 97813546721,
"Gandhinagar", 10000);

                DeliveryBoy[4] =
                  new Employee (20200004, id1[4], "Anna Akhmatova", "Anna
Andreyenva",
```

```
                        "Shruti Choudhary", 'M', "6/12/1996", 9451673257l,
"Gandhinagar", 10000);

                DeliveryBoy[5] =
                  new Employee (20200005, id1[5], "Susan Ashton", "Susan Rae
Hill",
                        "Jyoti Mirdha", 'M', "6/11/1999", 9412736578l,
"Gandhinagar", 10000);

                DeliveryBoy[6] =
                  new Employee (20200006, id1[6], "Khandubhai Kasanji Desai",
"Kanaiyalal Maneklal Munshi",
                        "Priyanka Gandhi", 'M', "8/9/1996", 9412358464l,
"Gandhinagar", 10000);

                DeliveryBoy[7] =
                  new Employee (20200007, id1[7], "Raghavji Patel", "Parsottam
Ukabhai Sabariya",
                        "Nagma", 'M', "5/5/1859", 94123784211, "Gandhinagar",
                        100);

                DeliveryBoy[8] =
                  new Employee (20200008, id1[8], "Mahendrasinh Vaghela",
"Harisinh Pratapsinh Chavda",
                        "Priya Dutt", 'M', "6/11/1836",9412378465l,
"Gandhinagar", 100);

                DeliveryBoy[9] =
                  new Employee (20200009, id1[9], "Vallabh Dharaviya", "Violet
Alva",
                        "Chhavi Rajawat", 'M', "5/1/1896",9874125745l,
"Gandhinagar",
                        10000);

        manager[0] = new Employee(20200000,id[0],"Shyam Mehta","Dorab Mehta",
                "Leela Mehta",'M',"date of birth",7014316242l,
                "Aadarsh Nagar SocietySector 24, Gandhinagar",10000);
        manager[1] = new Employee(20200001,id[1],"Aziz Gulzarilal
Nanda","Gulzarilal Nanda","Ramlata Nanda",
                'M',"10/5/1980",6457643591l,"Parth society House no. 47-
48Pethapur, Gandhinagar",10000);
        manager[2] = new Employee(20200002,id[2],"Dashrath Patel","Hetbhai
Patel","Meenaben Patel",'M',
                "20/6/1980",6457622591l,"Sector 3B Sector 3, Gandhinagar",10000);
        manager[3] = new Employee(20200003,id[3],"Ajay Jadeja","Ravindra
Jadeja","Heeraben Jadeja",'M',
                "8/6/1985",6498643591l,"sector 1c prime pgSector 1,
Gandhinagar",10000);
        manager[4] = new Employee(20200004,id[4],"Nayan Mongia","Ramlal
Mongia","Jeevlata Mongia",'M',
                "12/6/1989",4737643591l,"Sector 7CSector 7/B, Sector 7,
Gandhinagar",10000);
        manager[5] = new Employee(20200005,id[5],"Hardik Pandya","Krunal
Pandya","Daya Bhumik Pandya",'M',
                "19/8/1978",9087643591l,"Gujarat Knowledge Society Sector 10B,
Sector 10, Gandhinagar",10000);
        manager[6] = new Employee(20200006,id[6],"Aman Parekh","Vinod
Parekh","Asha Parekh",'M',
                "23/6/1980",8097643591l,"Panchshil Park Sarvodaynagar Society,
Sector 21, Gandhinagar",10000);
        manager[7] = new Employee(20200007,id[7],"Axar Patel","Ravindra
Patel","Ratna Patel",'M',
                "10/2/1978",9927643591l,"C Sector 12B, Gandhinagar",10000);
        manager[8] = new Employee(20200008,id[8],"Rahul Pathak","Narayan
Pathak","Supriya Pathak",'M',
```

```
                    "10/4/1978",8807643591l,"Sector 7/B Sector 7,
Gandhinagar",10000);
          manager[9] = new Employee(20200009,id[9],"Himesh Pujara","Cheteshwar
Pujara","Reema Pujara",'M',
                    "19/8/1979",92176435911,"GIDC Electronic Estate Sector 25,
Gandhinagar",10000);

          Employee delivery_boy[] = DeliveryBoy;

          zone[0] = new Zone("Zone-1","Sector 2,
Gandhinagar",2600,23.201553,72.640793,manager[0],delivery_boy[0],
                    delivery_boy[1],delivery_boy[2],delivery_boy[3]);
          zone[1] = new Zone("Zone-2","Sector 3,
Gandhinagar",2601,23.207973,72.627358,manager[1],delivery_boy[1],
                    delivery_boy[4],delivery_boy[2],delivery_boy[3]);
          zone[2] = new Zone("Zone-3","Sector 6,
Gandhinagar",2602,23.215843,72.632045,manager[2],delivery_boy[2],
                    delivery_boy[3],delivery_boy[4],delivery_boy[5]);
          zone[3] = new Zone("Zone-4","Sector 7,
Gandhinagar",2603,23.208969,72.644979,manager[3],delivery_boy[3],
                    delivery_boy[4],delivery_boy[5],delivery_boy[6]);
          zone[4] = new Zone("Zone-5","Sector 10,
Gandhinagar",2604,23.218279,72.659388,manager[4],delivery_boy[4],
                    delivery_boy[5],delivery_boy[6],delivery_boy[7]);
          zone[5] = new Zone("Zone-6","Sector 12,
Gandhinagar",2605,23.227749,72.639539,manager[5],delivery_boy[5],
                    delivery_boy[6],delivery_boy[7],delivery_boy[8]);
          zone[6] = new Zone("Zone-7","Sector 24,
Gandhinagar",2606,23.244824,72.641600,manager[6],delivery_boy[6],
                    delivery_boy[7],delivery_boy[8],delivery_boy[9]);
          zone[7] = new Zone("Zone-8","Sector 22,
Gandhinagar",2607,23.236829,72.654318,manager[7],delivery_boy[7],
                    delivery_boy[8],delivery_boy[9],delivery_boy[1]);
          zone[8] = new Zone("Zone-9","Sector 29,
Gandhinagar",2608,23.239700,72.663066,manager[8],delivery_boy[8],
                    delivery_boy[1],delivery_boy[2],delivery_boy[9]);
          zone[9] = new Zone("Zone-10","Sector 27,
Gandhinagar",2609,23.251131,72.645717,manager[9],delivery_boy[9],
                    delivery_boy[1],delivery_boy[2],delivery_boy[3]);


          try
          {

             FileOutputStream fileOut = new FileOutputStream("fileZone.txt");

             ObjectOutputStream objOut = new ObjectOutputStream(fileOut);

             //System.out.println("Success");

             for(int k=0; k<10; k++)
             {
               //System.out.println("Success");
               objOut.writeObject(zone[k]);
             }

             System.out.println("Zone Success");

             objOut.close();
          }
          catch (Exception e)
           {
             e.getStackTrace();
          }
```

```
        customer[0] = new Customer("Arnav","Arnav123","Arnav Singh", "Rd Number
4B, Sector 18, Gandhinagar, Gujarat 382018", 92658586781, 23.217780, 72.667277);
        customer[1] = new Customer("Pranav","Pranav123","Pranav Singh Rajput",
"Green City, Sector 26, Gandhinagar, Gujarat 382027", 92658586131, 23.255147,
72.638517);
        customer[2] = new Customer("Suresh","Suresh123","Suresh Pal", "K Rd, Green
City, Sector 26, Gandhinagar, Gujarat 382026", 92658589961, 23.259896, 72.641488);
        customer[3] = new Customer("Hitesh","Hitesh123","Hitesh Bhai Patel",
"Sector 28 GIDC, Sector 28, Gandhinagar, Gujarat 382041", 92600586861, 23.256569,
72.658944);
        customer[4] = new Customer("Ruchi","Ruchi123","Ruchi Gada", "Krishi
Bhavan, 2nd Floor, B-Wing, Sector - 10 A, CH Rd, Gandhinagar, Gujarat 382010",
92658581161, 23.214849, 72.650591);
        customer[5] = new Customer("Jignesh","Jignesh123","Jignesh Modi", "Service
Rd, Infocity, Gandhinagar, Gujarat 382421", 92652226861, 23.194350, 72.638759);
        customer[6] = new Customer("Rajesh","Rajesh123","Rajesh Gupta", "7J3H+94
Gandhinagar, Gujarat", 92658386861, 23.253417, 72.627792);
        customer[7] = new Customer("Suvendra","Suvendra123","Suvendra Shah",
"Sector 10A, Sector 10, Gandhinagar, Gujarat 382010", 92658512361, 23.214078,
72.653869);
        customer[8] = new Customer("Bhadresh","Bhadresh123","Bhadresh mehta",
"6PGJ+CC Chiloda, Gujarat", 92658999861, 23.226040, 72.731065);
        customer[9] = new Customer("Krishna","Krishna123","Krishna Jain",
"Randheja, Gujarat 382620", 92658544461, 23.292676, 72.638558);

        try
        {

           FileOutputStream fileOut = new FileOutputStream("fileCustomer.txt");

           ObjectOutputStream objOut = new ObjectOutputStream(fileOut);

           //System.out.println("Success");

           for(int k=0; k<10; k++)
           {
             //System.out.println("Success");
             objOut.writeObject(customer[k]);
           }

           System.out.println("Customer Success");

           objOut.close();
        }
        catch (Exception e)
         {
            e.getStackTrace();
        }

         ZonesData.readData();
         CustomerDataBase.readCustomer();
         PathFinder.initGraph();

         shipment[0] = new Shipment(20203001,customer[0],customer[2], 22, "books");
         shipment[1] = new Shipment(20203002,customer[1],customer[2], 32, "books");
         shipment[2] = new Shipment(20203003,customer[0],customer[4], 29, "other");
         shipment[3] = new Shipment(20203004,customer[4],customer[8], 12, "kitchen
items");
         shipment[4] = new Shipment(20203005,customer[6],customer[2], 10, "food
items");
         shipment[5] = new Shipment(20203006,customer[7],customer[5], 17,
"stationary items");
         shipment[6] = new Shipment(20203007,customer[3],customer[6],  8, "books");
         shipment[7] = new Shipment(20203008,customer[0],customer[9], 27,
"electronics");
```

```
        shipment[8] = new Shipment(20203009,customer[7],customer[2], 25, "books");
        shipment[9] = new Shipment(20203010,customer[3],customer[9], 35,
"medicines");
        try
        {
           FileOutputStream fileOut = new FileOutputStream("fileShipment.txt");

           // Creates an ObjectOutputStream
           ObjectOutputStream objOut = new ObjectOutputStream(fileOut);

           // Writes objects to the output stream

           for(int k=0; k<10; k++)
           {objOut.writeObject(shipment[k]);}

           System.out.print("Shipment Success");
           objOut.close();

        }
        catch (Exception e)
        {
           e.getStackTrace();
        }

        FileInputStream fis=null;
        ObjectInputStream dis=null;

        try
        {
             fis = new FileInputStream("fileShipment.txt");
             dis = new ObjectInputStream(fis);

             for(int i=0;i<Integer.MAX_VALUE;i++)
             {
                  Shipment z = (Shipment)dis.readObject();
                  z.showDetails();


    System.out.println("##########################################################
###################");
             }
        }

        catch ( ClassNotFoundException  e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
             //error occur when no more elements in file
        }
        finally
        {

             try
             {
                  dis.close();
                  fis.close();
             }
             catch (IOException e)
             {
                  e.printStackTrace();
             }
        }
```

```
        try
        {
             fis = new FileInputStream("fileZone.txt");
             dis = new ObjectInputStream(fis);

             for(int i=0;i<Integer.MAX_VALUE;i++)
             {
                  Zone z = (Zone)dis.readObject();
                  z.showZone();
                  z.getManager().showBasicDetails();
                  z.getDeliveryBoy()[0].showBasicDetails();


    System.out.println("##########################################################
##################");
             }
        }

        catch ( ClassNotFoundException  e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
             //error occur when no more elements in file
        }
        finally
        {

             try
             {
                  dis.close();
                  fis.close();
             }
             catch (IOException e)
             {
                  e.printStackTrace();
             }
        }

        try
        {
             fis = new FileInputStream("fileCustomer.txt");
             dis = new ObjectInputStream(fis);

             for(int i=0;i<Integer.MAX_VALUE;i++)
             {
                  Customer z = (Customer)dis.readObject();
                  z.showDetails("My");


    System.out.println("##########################################################
##################");
             }
        }

        catch ( ClassNotFoundException  e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
             //error occur when no more elements in file
        }
        finally
```

```
        {

            try
            {
                dis.close();
                fis.close();
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }
        }

    }
```

Key Features

i. All data items related to an entity and all actions it can perform is placed at one place to encapsulate them.
ii. Finding nearest zone is not depending on the arrangement of zones. It will work fine in any kind of arrangement of zones. It needs only location coordinates of all zones to find the nearest zone.
iii. Since all kind of delivery and collection of parcel is handled by nearest zones it minimizes time and cost.
iv. Zone is also divided in sub divisions reduces time and cost and also provide an efficient way to assign order.
v. It is automatic system which will assign orders to delivery boy without any human interaction.
vi. Dijkstra's algorithm of finding shortest path will calculate shortest path in minimum time and efficiently.