**Group C**

**Title :- Mini project(*TO GET BEST ROUTE FOR TRAVE***L**)**

Deliverables:

**1.Problem statement:** To design an application for best route to travel using Graph data structure(Dijktras algorithm).

**2. Details about designing:**

* **classes : -** Here we used one class (Dijktras) And functions.
* **Main function:**

**Details of functions:-**

*1. Number of vertices*

*2. Adjancency matrix [][]*

*3. distances []*

*4. Settled*

*5. Unsetteled*

*6. Number of Nodes*

A. dijkstra\_algorithm (**int** adjacency\_matrix[][], **int** source):- In this function we used the Method of dijkstra’s algorithm for finding node of min. distance

B. getNodeWithMinimumDistanceFromUnsettled ():-In this function we find the minimum distance from a node

C. evaluateNeighbours(**int** evaluationNode):-In this function we modify distance according to dijkstras algorithm

D. accept(**int** number\_of\_vertices):-In this function we accept distances between nodes

E. user\_accept(**int** source,**int** destination,**int** number\_of\_vertices):- In this function we find minimum distances between two cities

F. display(**int** n):- In this function we display the cities connected

**3.Block diagram:-**

*Number of vertices*

*Adjancency matrix [][]*

*distances []*

*Settled*

*Unsetteled*

*Number of Nodes*

*Destination*

*Source*

*Counter*

*Pass*

*evaluationNode*

*min*

*node*

*newDistance*

*edgeDistance*

**4. Short note on Data Structure used :-**

***Data structure used here is Graph (Dijktras algorithm)*** .

Dijkstra’s algorithm, published in 1959 and named after its creator Dutch computer scientist Edsger Dijkstra, can be applied on a weighted [graph](https://brilliant.org/wiki/graphs-basic/). The graph can either be directed or undirected.

**Dijkstra's algorithm** is an **algorithm** that is used to solve the **shortest** distance problem. That is, we use it to find the **shortest** distance between two vertices on a graph. Depending on what the graph represents, we can find **shortest** routes, minimum costs, etc. all using this **algorithm**. This Java program,to Implement Dijkstra’s algorithm using Set.Dijkstra’s algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree.

**5. Operations performed**

**1. Logic related to operations:-**

The following operations are performed:

1)Accept():This accept function is for the admin part.

 1.a)Admin enters the total number of cities

.b)A adjacency matrix is created of size of number of cities\*number of cities

C)Admin fills the matrix with the corresponding distances between two cities.

2)User\_accept():

a)User enters the mode of travel either by bus or train

b)User selects the two cities between whose distance he wants to determine.

c)The shortest distance between the two chosen cities is displayed.

d)User has the option to compare and check which mode of travel is more efficient.

3)Display():

a)Displays the cities for the user so he can select the two desired cities to determine the distance

4) dijkstra\_algorithm()

a)Initially distances  between  all the cities are set to maximum value.

b)The cities in the path are added in a hash table unsettled using hashset class and the visited cities are added to settled hashset

5). getNodeWithMinimumDistanceFromUnsettled()

a)Iterator is used to iterate the unsettled hashset

b) The city with minimum distance is found out from the Unsettled hashset

6) evaluateNeighbours()

a)We check if the destination node is present in the settled hash table or not.

b)If not then we discover the neighbours and find the new edge

c)If the new distance is minimum of  distance of destination node then the distance of destination node becomes the new evaluated distance.

d)And we add the destination node to the unsettled hash set

**CODE:-**

**package** dijktras;

/\*Project : TO GET BEST ROUTE FOR TRAVEL

\* Data Structure : GRAPH(Dijkstra's algorithm)

\* Group Members:

\* 2423 Aaditi Patil

\* 2427 Samruddhi Patil

\* 2430 Richa Pawar

\* 2431 Gargi Pol

\*

\* Batch : B2 (SY COMP DIV B)

\*/

**import** java.util.\*;

**public** **class** Dijkstras

{

**public** Dijkstras()//constructor

{

}

**int** adjacency\_matrix[][];//matrix for graph

**int** number\_of\_vertices;

**private** **int** distances[];

**private** Set<Integer> settled;//visited nodes

**private** Set<Integer> unsettled;//non visited nodes

**private** **int** number\_of\_nodes;

**private** **int** adjacencyMatrix[][];

**public** Dijkstras(**int** number\_of\_nodes)//parametric constructor

{

**this**.number\_of\_nodes = number\_of\_nodes;

distances = **new** **int**[number\_of\_nodes + 1];

settled = **new** HashSet<Integer>();

unsettled = **new** HashSet<Integer>();

adjacencyMatrix = **new** **int**[number\_of\_nodes + 1][number\_of\_nodes + 1];

}

**public** **void** dijkstra\_algorithm(**int** adjacency\_matrix[][], **int** source)//Method of dijkstras algorithm

{

**int** evaluationNode;

**for** (**int** i = 1; i <= number\_of\_nodes; i++)

**for** (**int** j = 1; j <= number\_of\_nodes; j++)

adjacencyMatrix[i][j] = adjacency\_matrix[i][j];

**for** (**int** i = 1; i <= number\_of\_nodes; i++)

{

distances[i] = Integer.***MAX\_VALUE***;

}

unsettled.add(source);

distances[source] = 0;

**while** (!unsettled.isEmpty())

{

evaluationNode = getNodeWithMinimumDistanceFromUnsettled();//to find node of minimum distance

unsettled.remove(evaluationNode);

settled.add(evaluationNode);

evaluateNeighbours(evaluationNode);

}

}

**private** **int** getNodeWithMinimumDistanceFromUnsettled()//method to find the minimum distance from a node

{

**int** min;

**int** node = 0;

Iterator<Integer> iterator = unsettled.iterator();

node = iterator.next();

min = distances[node];

//iterate to find minimum distance

**for** (**int** i = 1; i <= distances.length; i++)

{

**if** (unsettled.contains(i))

{

**if** (distances[i] <= min)

{

min = distances[i];

node = i;

}

}

}

**return** node;

}

**private** **void** evaluateNeighbours(**int** evaluationNode)//to modify distance according to dijkstras algorithm

{

**int** edgeDistance = -1;

**int** newDistance = -1;

**for** (**int** destinationNode = 1; destinationNode <= number\_of\_nodes; destinationNode++)

{

**if** (!settled.contains(destinationNode))

{

**if** (adjacencyMatrix[evaluationNode][destinationNode] != Integer.***MAX\_VALUE***)

{

edgeDistance = adjacencyMatrix[evaluationNode][destinationNode];

newDistance = distances[evaluationNode] + edgeDistance;

//new distance is minimum of edge distance and new distance

**if** (newDistance < distances[destinationNode])

{

distances[destinationNode] = newDistance;

}

unsettled.add(destinationNode);

}

}

}

}

**public** **void** accept(**int** number\_of\_vertices)//accept distances between nodes

{

String istring = **null**;

String jstring=**null**;

Scanner scan = **new** Scanner(System.***in***);

//System.out.println("Enter the number of vertices");

//number\_of\_vertices = scan.nextInt();

adjacency\_matrix = **new** **int**[number\_of\_vertices + 1][number\_of\_vertices + 1];

**for** (**int** i = 1; i <= number\_of\_vertices; i++)

{

**if**(i==1)

{

istring="Mumbai";

}

**else** **if**(i==2)

{

istring="Pune";

}

**else** **if**(i==3)

{

istring="Nashik";

}

**else** **if**(i==4)

{

istring="Satara";

}

**else** **if**(i==5)

{

istring="Wai";

}

**else** **if**(i==6)

{

istring="Solapur";

}

**else** **if**(i==7)

{

istring="Nagpur";

}

**else** **if**(i==8)

{

istring="Kolhapur";

}

**else** **if**(i==9)

{

istring="Sangli";

}

**else** **if**(i==10)

{

istring="Sangamner";

}

**for** (**int** j = 1; j <= number\_of\_vertices; j++)

{

**if**(j==1)

{

jstring="Mumbai";

}

**else** **if**(j==2)

{

jstring="Pune";

}

**else** **if**(j==3)

{

jstring="Nashik";

}

**else** **if**(j==4)

{

jstring="Satara";

}

**else** **if**(j==5)

{

jstring="Wai";

}

**else** **if**(j==6)

{

jstring="Solapur";

}

**else** **if**(j==7)

{

jstring="Nagpur";

}

**else** **if**(j==8)

{

jstring="Kolhapur";

}

**else** **if**(j==9)

{

jstring="Sangli";

}

**else** **if**(j==10)

{

jstring="Sangamner";

}

//entering distance between cities

System.***out***.println("ENTER DISTANCE IN KMS FOR PATH BETWEEN: "+istring+" And "+jstring);

adjacency\_matrix[i][j] = scan.nextInt();

**if** (i == j)

{

adjacency\_matrix[i][j] = 0;

**continue**;

}

**if** (adjacency\_matrix[i][j] == 0)

{

adjacency\_matrix[i][j] = Integer.***MAX\_VALUE***;

}

}

}

}

**public** **int** user\_accept(**int** source,**int** destination,**int** number\_of\_vertices)//to find minimum distances between two cities

{

**int** dist = 0;

Dijkstras obj=**new** Dijkstras();

Scanner scan1 = **new** Scanner(System.***in***);

Dijkstras obj2 = **new** Dijkstras(number\_of\_vertices);

obj2.dijkstra\_algorithm(adjacency\_matrix, source);

//to find shortest distance

System.***out***.println("THE SHORTEST DISTANCE IS:");

**int** i;

**for** ( i = 1; i <= obj2.distances.length - 1; i++)

{

**if** (i == destination)

{

System.***out***.print(obj2.distances[i]);

dist=obj2.distances[i];// shortest path

**break**;

}

}

**return** dist;

}

**public** **void** display(**int** n)//display the cities connected

{

System.***out***.println("\n-------------------------");

System.***out***.println(" WELCOME");

System.***out***.println("-------------------------");

System.***out***.println("THE FOLLOWING CITIES ARE CONSIDERED:");

System.***out***.println("----------------------- ");

System.***out***.println("CITY\t\tCITY NO:");

System.***out***.println("----------------------- ");

**switch**(n)

{

**case** 1://if route for 1 city created

System.***out***.println("MUMBAI\t\t1");

**break**;

**case** 2:// if route between 2 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

**break**;

**case** 3:// if route between 3 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

**break**;

**case** 4://if route between 4 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

System.***out***.println("SATARA\t\t4");

**break**;

**case** 5://if route between 5 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

System.***out***.println("SATARA\t\t4");

System.***out***.println("WAI\t\t5");

**break**;

**case** 6://if route between 6 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

System.***out***.println("SATARA\t\t4");

System.***out***.println("WAI\t\t5");

System.***out***.println("SOLAPUR\t\t6");

**break**;

**case** 7://if route between 7 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

System.***out***.println("SATARA\t\t4");

System.***out***.println("WAI\t\t5");

System.***out***.println("SOLAPUR\t\t6");

System.***out***.println("NAGPUR\t\t7");

**break**;

**case** 8://if route between 8 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

System.***out***.println("SATARA\t\t4");

System.***out***.println("WAI\t\t5");

System.***out***.println("SOLAPUR\t\t6");

System.***out***.println("NAGPUR\t\t7");

System.***out***.println("KOLHAPUR\t\t8");

**break**;

**case** 9://if route between 9 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

System.***out***.println("SATARA\t\t4");

System.***out***.println("WAI\t\t5");

System.***out***.println("SOLAPUR\t\t6");

System.***out***.println("NAGPUR\t\t7");

System.***out***.println("KOLHAPUR\t\t8");

System.***out***.println("SANGLI\t\t9");

**break**;

**case** 10://if route between 10 cities available

System.***out***.println("MUMBAI\t\t1");

System.***out***.println("PUNE\t\t2");

System.***out***.println("NASHIK\t\t3");

System.***out***.println("SATARA\t\t4");

System.***out***.println("WAI\t\t5");

System.***out***.println("SOLAPUR\t\t6");

System.***out***.println("NAGPUR\t\t7");

System.***out***.println("KOLHAPUR\t\t8");

System.***out***.println("SANGLI\t\t9");

System.***out***.println("SANGAMNER\t\t10");

**break**;

}

}

}

public class Main\_Project {

public static void main(String[] args) {

int ch=0;

int choice=0;

int choice1=0;

int counter=0;

int flag1=0;

int flag2=0;

int source=0;

int destination=0;

int source1 = 0;

int destination1=0;

Dijkstras obj1=new Dijkstras();//object of Dijkstras class

Dijkstras obj2=new Dijkstras();

String pass;

int number\_of\_vertices=0;

Scanner sc = new Scanner(System.in);

String comp = "admin123" ;//default password

do//menu for home page

{

System.out.println("----------------------- ");

System.out.println("\tWELCOME");

System.out.println("----------------------- ");

System.out.println("LOGIN IN AS:\n1.ADMIN\n2.CUSTOMER");

int bus = 0;

int train = 0;

ch=sc.nextInt();

switch(ch)

{

case 1://case for Admin login

int cnt=0;

if(counter==1)

{

System.out.println("YOU HAVE ALREADY FILLED IN THE DETAILS!!");

}

else//to enter route details

{

do

{

System.out.println("ENTER YOUR PASSWORD");

pass=sc.next();

if(pass.compareTo(comp)==0 )

{

System.out.println("----------------------- ");

System.out.println(" HOLA,ADMIN!!");

System.out.println("----------------------- ");

System.out.println("FILL IN THE INFORMATION BELOW:");

System.out.println("\nENTER NUMBER OF CITIES FOR WHICH ARE FILLING THE INFORMATION:");

number\_of\_vertices = sc.nextInt();

System.out.println("----------------------- ");

System.out.println("ENTER DETAILS FOR BUS:");

System.out.println("----------------------- ");

obj1.accept(number\_of\_vertices);

System.out.println("----------------------- ");

System.out.println("ENTER DETAILS FOR TRAIN:");

System.out.println("----------------------- ");

obj2.accept(number\_of\_vertices);

flag1=1;

counter=1;

break;

}

else//if wrong password entered

{

System.out.println("YOU ENTERED WRONG PASSWORD!!");

cnt++;

System.out.println("YOU HAVE ONLY "+(3-cnt)+" ATTEMPTS LEFT");

}

}while(cnt<3 && cnt>0);

}

break;

case 2://case for customer services

if(flag1==1)//travel routes are ready

{

do

{

//Menu for customer services

obj1.display(number\_of\_vertices);

System.out.println("\nHOW DO YOU WANT TO TRAVEL?");

System.out.println("----------------------");

System.out.println("1.BY BUS\n2.BY TRAIN\n3.COMPARE COST EFFICIENCY\n4.EXIT");

System.out.println("----------------------- ");

System.out.println("PLEASE ENTER YOUR CHOICE:");

choice=sc.nextInt();

switch(choice)

{

case 1://case for customer to travel by bus

System.out.println("ENTER THE STARTING CITY NO.:");

source = sc.nextInt();

System.out.println("ENTER THE DESTINATION CITY NO.");

destination = sc.nextInt();

bus=obj1.user\_accept(source,destination,number\_of\_vertices);

break;

case 2://case for customer to travel by train

System.out.println("ENTER THE STARTING CITY NO.:");

source1 = sc.nextInt();

System.out.println("ENTER THE DESTINATION CITY NO.");

destination1 = sc.nextInt();

train=obj2.user\_accept(source1,destination1,number\_of\_vertices);

break;

case 3://case for customer to to compare travel by bus and train

if(source==source1 && destination==destination1)

{

flag2=1;//travel routes for bus and train must be same

}

if(flag2==1)

{

if(bus>train)//if traveling distance by train is less

{

System.out.println("BUS REQUIRES MORE DISTANCE TO BE TRAVELLED THAN TRAIN");

System.out.println("HENCE,TRAIN IS MORE EFFICIENT FOR THIS ROUTE!!");

}

else if(train>bus)//if traveling distance by bus is less

{

System.out.println("TRAIN REQUIRES MORE DISTANCE TO BE TRAVELLED THAN BUS");

System.out.println("HENCE,BUS IS MORE EFFIECIENT FOR THIS ROUTE!!");

}

else//if traveling distance by bus and train are equal

{

System.out.println("TRAIN AND BUS BOTH ARE EQUALLY EFFICEIENT FOR THIS ROUTE!! ");

}

}

else

{

System.out.println("YOU HAVE CHOSEN TWO DIFFERENT ROUTES!!");

System.out.println("PLEASE SELECT SAME ROUTES TO CALCULATE EFFICIENCY!");

}

}

System.out.println("\nENTER '1' TO CONTINUE '0' TO EXIT");

choice1=sc.nextInt();

}while(choice1!=0);

}

else//travel routes are yet to be filled by admin

{

System.out.println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\t\t\t\tTHE WEBSITE IS STILL UNDER CONSTRUCTION!!");

System.out.println("\t\t\t\tWE APOLOGIZE FOR THE INCONVENIENCE");

System.out.println("\t\t\t\tVISIT AGAIN!!");

System.out.println("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

break;

}

}while(ch!=0);

}

}

**OUTPUT:-**

**-----------------------**

**WELCOME**

**-----------------------**

**LOGIN IN AS:**

**1.ADMIN**

**2.CUSTOMER**

**2**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**THE WEBSITE IS STILL UNDER CONSTRUCTION!!**

**WE APOLOGIZE FOR THE INCONVENIENCE**

**VISIT AGAIN!!**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**-----------------------**

**WELCOME**

**-----------------------**

**LOGIN IN AS:**

**1.ADMIN**

**2.CUSTOMER**

**1**

**ENTER YOUR PASSWORD**

**asas**

**YOU ENTERED WRONG PASSWORD!!**

**YOU HAVE ONLY 2ATTEMPTS LEFT**

**ENTER YOUR PASSWORD**

**fdfd**

**YOU ENTERED WRONG PASSWORD!!**

**YOU HAVE ONLY 1ATTEMPTS LEFT**

**ENTER YOUR PASSWORD**

**admin123**

**-----------------------**

**HOLA,ADMIN!!**

**-----------------------**

**FILL IN THE INFORMATION BELOW:**

**ENTER NUMBER OF CITIES FOR WHICH ARE FILLING THE INFORMATION:**

**5**

**-----------------------**

**ENTER DETAILS FOR BUS:**

**-----------------------**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Mumbai**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Pune**

**220**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Nashik**

**360**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Satara**

**240**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Wai**

**210**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Mumbai**

**220**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Pune**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Nashik**

**335**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Satara**

**120**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Wai**

**110**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Mumbai**

**360**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Pune**

**335**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Nashik**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Satara**

**385**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Wai**

**350**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Mumbai**

**240**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Pune**

**120**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Nashik**

**385**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Satara**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Wai**

**40**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Mumbai**

**210**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Pune**

**110**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Nashik**

**350**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Satara**

**40**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Wai**

**0**

**-----------------------**

**ENTER DETAILS FOR TRAIN:**

**-----------------------**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Mumbai**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Pune**

**150**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Nashik**

**320**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Satara**

**270**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Mumbai And Wai**

**240**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Mumbai**

**150**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Pune**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Nashik**

**310**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Satara**

**140**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Pune And Wai**

**112**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Mumbai**

**320**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Pune**

**310**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Nashik**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Satara**

**370**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Nashik And Wai**

**350**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Mumbai**

**270**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Pune**

**140**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Nashik**

**370**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Satara**

**0**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Satara And Wai**

**60**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Mumbai**

**240**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Pune**

**112**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Nashik**

**350**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Satara**

**60**

**ENTER DISTANCE IN KMS FOR PATH BETWEEN: Wai And Wai**

**0**

**-----------------------**

**WELCOME**

**-----------------------**

**LOGIN IN AS:**

**1.ADMIN**

**2.CUSTOMER**

**2**

**-------------------------**

**WELCOME**

**-------------------------**

**THE FOLLOWING CITIES ARE CONSIDERED:**

**-----------------------**

**CITY CITY NO:**

**-----------------------**

**MUMBAI 1**

**PUNE 2**

**NASHIK 3**

**SATARA 4**

**WAI 5**

**HOW DO YOU WANT TO TRAVEL?**

**----------------------**

**1.BY BUS**

**2.BY TRAIN**

**3.COMPARE COST EFFICIENCY**

**4.EXIT**

**-----------------------**

**PLEASE ENTER YOUR CHOICE:**

**1**

**ENTER THE STARTING CITY NO.:**

**1**

**ENTER THE DESTINATION CITY NO.**

**5**

**THE SHORTEST DISTANCE IS:**

**210 KMS**

**ENTER &#39;1&#39; TO CONTINUE &#39;0&#39; TO EXIT**

**1**

**-------------------------**

**WELCOME**

**-------------------------**

**THE FOLLOWING CITIES ARE CONSIDERED:**

**-----------------------**

**CITY CITY NO:**

**-----------------------**

**MUMBAI 1**

**PUNE 2**

**NASHIK 3**

**SATARA 4**

**WAI 5**

**HOW DO YOU WANT TO TRAVEL?**

**----------------------**

**1.BY BUS**

**2.BY TRAIN**

**3.COMPARE COST EFFICIENCY**

**4.EXIT**

**-----------------------**

**PLEASE ENTER YOUR CHOICE:**

**2**

**ENTER THE STARTING CITY NO.:**

**2**

**ENTER THE DESTINATION CITY NO.**

**3**

**THE SHORTEST DISTANCE IS:**

**310 KMS**

**ENTER &#39;1&#39; TO CONTINUE &#39;0&#39; TO EXIT**

**1**

**-------------------------**

**WELCOME**

**-------------------------**

**THE FOLLOWING CITIES ARE CONSIDERED:**

**-----------------------**

**CITY CITY NO:**

**-----------------------**

**MUMBAI 1**

**PUNE 2**

**NASHIK 3**

**SATARA 4**

**WAI 5**

**HOW DO YOU WANT TO TRAVEL?**

**----------------------**

**1.BY BUS**

**2.BY TRAIN**

**3.COMPARE COST EFFICIENCY**

**4.EXIT**

**-----------------------**

**PLEASE ENTER YOUR CHOICE:**

**3**

**YOU HAVE CHOSEN TWO DIFFERENT ROUTES!!**

**PLEASE SELECT SAME ROUTES TO CALCULATE EFFICIENCY!**

**ENTER &#39;1&#39; TO CONTINUE &#39;0&#39; TO EXIT**

**1**

**-------------------------**

**WELCOME**

**-------------------------**

**THE FOLLOWING CITIES ARE CONSIDERED:**

**-----------------------**

**CITY CITY NO:**

**-----------------------**

**MUMBAI 1**

**PUNE 2**

**NASHIK 3**

**SATARA 4**

**WAI 5**

**HOW DO YOU WANT TO TRAVEL?**

**----------------------**

**1.BY BUS**

**2.BY TRAIN**

**3.COMPARE COST EFFICIENCY**

**4.EXIT**

**-----------------------**

**PLEASE ENTER YOUR CHOICE:**

**1**

**ENTER THE STARTING CITY NO.:**

**2**

**ENTER THE DESTINATION CITY NO.**

**3**

**THE SHORTEST DISTANCE IS:**

**335 KMS**

**ENTER &#39;1&#39; TO CONTINUE &#39;0&#39; TO EXIT**

**1**

**-------------------------**

**WELCOME**

**-------------------------**

**THE FOLLOWING CITIES ARE CONSIDERED:**

**-----------------------**

**CITY CITY NO:**

**-----------------------**

**MUMBAI 1**

**PUNE 2**

**NASHIK 3**

**SATARA 4**

**WAI 5**

**HOW DO YOU WANT TO TRAVEL?**

**----------------------**

**1.BY BUS**

**2.BY TRAIN**

**3.COMPARE COST EFFICIENCY**

**4.EXIT**

**-----------------------**

**PLEASE ENTER YOUR CHOICE:**

**3**

**BUS REQUIRES MORE DISTANCE TO BE TRAVELLED THAN TRAIN**

**HENCE,TRAIN IS MORE EFFICIENT FOR THIS ROUTE!!**

**ENTER &#39;1&#39; TO CONTINUE &#39;0&#39; TO EXIT**

**0**

**-----------------------**

**WELCOME**

**-----------------------**

**LOGIN IN AS:**

**1.ADMIN**

**2.CUSTOMER**