Palestine Polytechnic University

College of Engineering & Technology

Electrical & Computer Engineering Department

Computer Systems Engineering

Graduation Project

Routing Guide

Project Team

Bayan M Khanah **Alaa A Taha**

Project Supervisor

Dr. Amal Dwaik

Hebron- Palestine

June -2011

جامعة بوليتكنيك فلسطين

كلية الهندسة والتكـــنولوجيا

الخليل- فلسطين

اسم المشروع

Routing Guide

اسماء الطلبة

بيان محمد خنة آلاء عبد الفتاح طه

بناء على نظام كلية الهندسة و التكنولوجيا واشراف ومتابعة المشرف المباشر على المشروع وموافقة اعضاء اللجنة الممتحنة تم تقديم هذا المشروع الى دائرة الهندسة الكهربائية وذلك للوفاء بمتطلبات درجة البكالوريوس في هندسة تخصص أنظمة الحاســـوب.

توقيع المشرف

.....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

.....

Abstract

The wide expansion in development of the cities had led to increase in the roads network. Traveling from one place to another within the city can be achieved through multi routes. However, the question is which route is the best? What are the factors that determine this? The project uses the most widely used ones; the cost and the time.

Therefore, the project aims to find the best answer to this question and facilitate the process of travelling across a city through making a simulation of real city, taking Hebron city as a case study.

Using this system; any person in this city can choose the best taxi office to his location, and then the system will find the best path from the customer's location to any destination inside this city.

الملخص

أدي التطور السريع في المدن إلى تعدد وازدياد شبكات الطرق داخل المدينه الواحده ، فالانتقال من مكان الى اخر داخل المدينه يمكن ان يتم من خلال طرق متعدده، اي من هذه الطرق هو الافضل؟ و ما هي العوامل التي تحدد ذلك ، المشروع يستعمل أكثر هذه العوامل استخداما وهي التكلفة و الوقت.

يهدف المشروع "Routing Guide" الى ايجاد الاجابه الافضل لهذا السؤال، و تسهيل عملية السفر والانتقال داخل المدينه الواحده ،من خلال عمل محاكاة لمدينه واقعيه، حيث تم استخدام مدينة الخليل كحاله للدراسه.

بلستخدام هذا النظام يمكن لاي شخص داخل المدينه اختيار أفضل مكتب تكسي بالنسبة إلى موقعه، ثم يقوم النظام بإيجاد الطريق الأفضل من موقع الزبون إلى الموقع الذي يرغب في الذهاب إليه.

Dedication

To our University

To our supervisor

To our parents

To Our Families

To our friends

Acknowledgment

Our deepest gratitude and appreciation goes for our supervisor Dr. Amal Dwaik for her valuable contribution to accomplish this work. Without her supervision these papers wouldn't have been produced…

Thanks and appreciation to our university, which gave us the opportunity to perform this project…

To our families who granted us best condition to carry out this work…

To our friends and colleagues with whom we spent the best time of our life…

List of contents

List of Tables

List of Figures

CHAPTER ONE 1

Introduction

1.1 Project Idea

1.2 Project Importance

1.3 Project Goals

1.4 Previous Systems

1.5 Time Plan

1.6 System Requirements

1.7 Initial Plan

1.8 Road Map

1.1 Project Idea

The main idea of the project is to simulate a real city; in order to build a routing guide system. The routing guide system is a windows-based- application aims to help the normal user in travelling through a given city.

When a customer wants to go anywhere, the system will help him/her to find a suitable Taxi office and a suitable path for the journey.

1.1.1 Example

Figure 1.1 shows a map for Hebron city, if someone stands (for example) at Hebron entrance and want to travel to reach Hebron university, the system will show rent car1 as the best taxi office for the user ,the system gives some information to help the user to call this office and rent a car, then the system will show the best path from Hebron entrance to Hebron University as [Hebron entrance, Pink-Fountain, Ras-Aljora, Al-Haras, Hebron university], in addition the system will calculate the required time or cost for this journey.

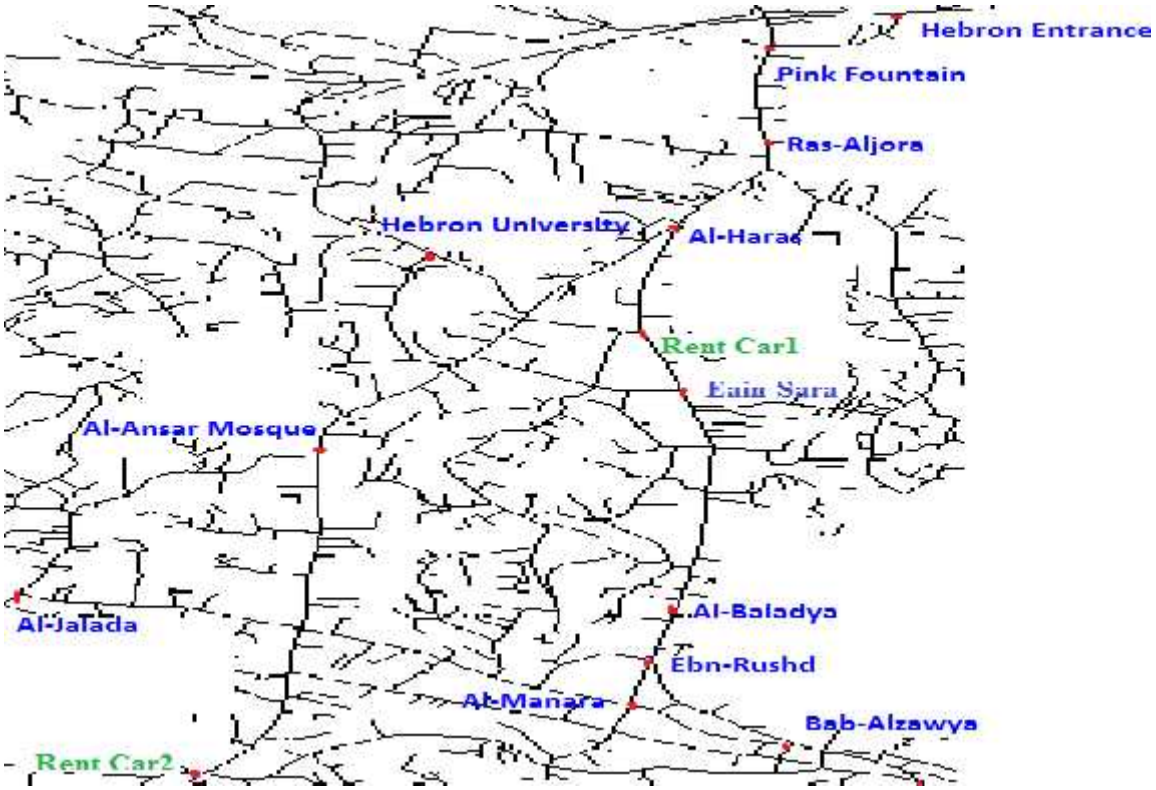

Figure1.1 Hebron map

1.2 Project Importance

Rapid developments of transportation is enormous, the ability to cover vast distances across the country in record times benefits the country's industrial growth, encourage tourism and makes our country more civilized. The wide expansion in development of the cities had led to increase in the roads network. Traveling from one place to another within the city can be achieved through multi routes. Which route is more appropriate? In terms of cost or time, these one of the most frequently asked questions, therefore the system is important because:

1. It gives the visitors of a new city the ability to choose a suitable path without having any worry about knowing the city roads, will save their effort, time and money, and will encourage them to visit it again.

2. The people who live in such a city will get benefit from this system; they will avoid an exploitation of some drivers to get more money; if they will travel to a place that they never visited; again this will save their time and money.

3. The system will help a customer to find the closest rental car office and tell the driver which path to go through; this will remove any confusion amongst the Rental Car Companies and their customers.

1.3. Project Goals

Our project aims to produce a system that will be able to:

1. Find the closest rental car office to the customer location.

2. Obtain the fastest and the cheapest route from one station to another.

3. Calculate the time and cost required to reach the destination.

We may also sought to possibly implement additional features regarding journey information that user may find helpful, such as finding the stations at which the customer may pass through the journey.

1.4. Previous Systems

There were similar ideas made in countries as Jordan, Germany and other European countries. We hope to improve such idea to work in our country and to be implemented in the future by wireless service via mobile application.

There exist web-based routing systems in which the valid routes are described in a document called “National Routing Guide” on the ATOC website

(http://www.atoc.org/index.asp)

The guide is rather complex, so it is not easy for a user to readily work out which routes may be valid.

The following web sites are examples for some applications similar to the project idea:

1. http://www.theaa.com/route-planner/index.jsp

2. http://www.thelivingweb.net/maps.html

3. http://www.esri.com/software/arclogistics/index.html

1.5 Requirements

This section lists the main requirements that must be met in project, in order to set the main services that will be provided.

1.5.1 User requirements

1. The system should provide a Graphical User Interface (GUI) in order to interact with the user in an easy and a simple way.

2. The GUI must be easy and convenient to be understood by user.

3. The GUI should provide the user with the results.

1.5.2 System requirements

1. 5.2.1 Functional requirements

1. The system should apply Dijkstra algorithm on the city map to find the shortest route from one station to another

2. A setup file must be linked to the system; needed to update any constant value that must enter the system to do its work.

3. The system shall be able to be updates under new conditions

4. Give overview about the city, user could see and understand what is happening.

1. 5.2.2 Non functional requirements

1. Usability: the system should be easy to use.

2. Reliability: the system should be reliable under different circumstances. The simulation information must be accurate and reliable.

3. Performance: the system should work with good performance measures. The simulation will provide the user a useful and guide information about the routing process.

1. 6. Time Plan

In this section we will preview the system schedule for the developers, take in Care the time period given for delivery the system, and the dependences between Tasks.

Table 1.1: Tasks description (first semester)

Task ID	Task description
T1.1	Selecting the project.
T1.2	Collecting information.
T1.3	System analysis
T1.4	System design
T1.5	Writing documentation

Table 1.2: Time plan (first semester)

Task/week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T1.1																
T1.2																
T1.3																
T1.4																
T1.5																

Table 1.3: Tasks description (second semester)

Task ID	Task description
T2.1	System design and analysis
T2.2	Component implementation.
T2.3	Component testing
T2.4	Integrity testing
T2.5	Writing documentation

Table 1.4: Time plan (second semester)

Task/week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
T2.1																
T2.2																
T2.3																
T2.4																
T2.5																

1.7 Initial Plan

We aim to automate this process and manipulate the data of a city map to return information useful for the end user.

Thus we devised a method to reliably represent the information on a workable database, so that it represents the information taken from an image for some secondary and main roads in Hebron city; thus building a virtual small city to help in simulating a real one.

1.8. Road Map

The project consists of six chapters, each chapter talks about a specific area of the project. The following explain the content of each chapter.

Chapter One:	"Introduction", this chapter gives an introduction about the system, its importance, lists project objectives, other related systems, and project time plan.
Chapter Two:	"Theoretical Background", this chapter gives a clear picture about the system theoretical background related to the main concepts of Routing.
Chapter Three:	"Design Concepts", this chapter shows system block diagram, design options that can be used in the system ,and a detailed analysis about a virtual city in order to simulate a real on.
Chapter four:	"System Implementation and Testing", this chapter shows system block diagram, Design options that can be used in the system and some testing examples.
Chapter five:	"Conclusion and Future Work", this chapter is simplifying the conclusions and results achieved after implementing the entire project and gives suggestions for the future system developing.

CHAPTER TWO 2

Theoretical Background

2.1 Overview

2.2 Graph Theory

2.3 Working with maps

2.4 Routing

2.5 Important Parameters affect routing

2.6 Time and Cost calculations

2.7 Best Path

2.8 Summary

2.1 Overview

This chapter provides an illustrative theoretical background for the project related topics. The best route determined by the cost required to travel through it and how much time it takes to reach the destination.

2.2 Graph Theory

2.2.1 Definition

Graph theory is the study of points and lines. In particular, it involves the ways in which sets of points, called vertices, can be connected by lines or arcs, called edges. Graphs in this context differ from the more familiar coordinate plots that portray mathematical relations and functions.

Graphs are classified according to their complexity, the number of edges allowed between any two vertices, and whether or not directions are assigned to edges. [1]

In the map each position represents a vertex (Node) and each street segment represents an arc. The map itself represents a directed graph.

2.2.2 Connectivity and block

A vertex cut is a subset of vertices in a graph whose removal from the graph will cause the remaining graph disconnected.

An edge cut is a subset of edges in a graph whose removal from the graph will cause the remaining graph discounted.

A connected graph that has no cut vertices is called block. Every block with at least three vertices is 2-connected. A block of graph is a subgraph that is a block and is maximal with respect to this property. Every graph is the union of its blocks.

A group of paths in a graph is called to be internally-disjoint if no vertex of the graph is an internal vertex of more than one path of the group.

Theorem: A graph containing at least 3 vertices is 2-connected if and only if any two vertices of the graph are connected at least two internally-disjoint paths. [2]

Theorem: If a graph is 2-connected, then any two vertices of the graph are on a common cycle.

Theorem: If a graph is a block containing at least 3 vertices, then any two vertices of the graph are on a common cycle. [2]

2.2.3 Defining networks and graphs

Networks are simple and composed of two fundamental components: edges and junctions. Examples of edges are streets, transmission lines, pipes, and stream reaches. Examples of junctions are street intersections, fuses, switches, service taps, and the confluence of stream reaches. See the following illustration:

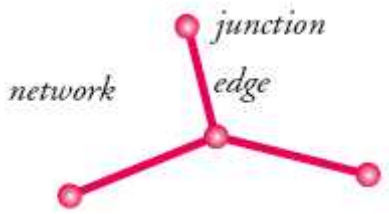

Figure2.1 networks and graphs

Edges connect at junctions, and the flow from one edge—automobiles, electrons, water—is transferred to another edge.

To analyze networks, geographers use a branch of mathematics called graph theory. Graph theory is interesting because it provides rigorous theorems concerning properties of networks and algorithms to solve network traversing problems. For example, one set of related algorithms for solving a problem in graph theory, known as the Chinese Postman Problem, provides optimized solutions for many applications such as postal delivery, street sweeping, bus passenger pickup and drop-off, and garbage collection.

Graph theory is the foundation for understanding networks and topology. A geographic information system (GIS) modeler should be familiar with the concepts and terminology of graph theory because it helps to classify and model connectivity and adjacency relationships among geographic features. [3]

2.3 Working with maps

The system will deal with a map of a given city; here the graph of the map itself cannot be the input to the system. Instead we will analyze this map into numeric data describing each location at this map in a specific manner, by determining all the intersections points positions at the map, and analyze their as we will show later in this chapter.

As we said before all the intersection point must be known, so in this analysis we will describe each intersection point focusing on all parameters which are shown in the previous section. Then describe every location in the map as an adjacent to one or more intersection points.

So for every intersection point at the map we must determine the following data:

1. All adjacent points (the adjacent point may be intersection point or any other location on the map). We say tow points are adjacent if they are connected directly by a line segment.
2. The distance between that point and every another adjacent point.
3. Type of the street segment connecting these two points as major, main, secondary….etc.
4. The name or anything defines this street segment itself.

In the next chapter we will describe a map of virtual small city taken from some main and secondary streets in Hebron city, we will describe each point as above.

2.4 Routing

A **route** from an origin (a) to a destination (b) is a continuous and finite collection of nodes and arcs that connects the origin and destination nodes. [4]

In any city in the world there can be more than one direction to reach the destination, routing means to determine which direction should be selected among several options.

Changing the direction may mean to change the road and changing the road will affect several important factors that cannot be ignored.

So routing will occur at the intersection points between these roads, because at these points the decision should be taken to select one of the intersecting roads, and then go throw the selected direction.

Thus if we have a network of streets, and we want to choose a path between a source and a destination points all the intersection points between the streets must be known.

2.5 Important Parameters affect routing

As we said above there is several factors might be changed when changing the road, but these are not the only parameters which affect the decision of choosing a route. Any journey is affected by many factors like the street, the car and the time of the journey.

So we can classify the Parameters affecting the routing decision as follows:

a. Parameters related to the street.

Parameters that changes when move from one road to another, and these are:

1. Street Type: A Street can be Major, Main, Secondary1, Secondary2, or Unpaved Street. And here we will give a brief description about each type:

-Major Streets: The streets with width greater than 20 meters and extend in the city from North to South or from East to West.

-Main Streets: The streets with width greater than or equal 12 meters and connect the major streets.

-Secondary Streets1: The streets with width less than 12 meters and connect Major or Main streets.

-Secondary Streets2: The streets with width less than 12 meters and with dead end or connected to Unpaved Streets.

-Unpaved Streets: The streets with unpaved ground.

*Note: This classification is taken from **Surveying and Geometrics Engineering Department** in Palestine Polytechnic University.*

2. Street length: this affects the distance between the source and the destination and thus the time and the cost for the journey.

3. The street itself: if a street is disabled at any time, then each segment of that street connecting any two points will be disabled, and this affect the routing decision.

b. Parameters related to the car

In Routing decision the car must be considered, as an important factor which determines the following parameters:

1. Average Speed of the car: the average speed of the car with the distance between the source and the destination determine the time required by the equation Time=Distance/Speed.

Car average speed may be changed from one street to another as the street type changed. We will discuss that later.

2. Car Type: this will affect the speed of the car and the amount of the fuel the car consumed.

Here in our project we will focus on the small normal rent cars in Hebron city which there average speed nearly the same.

C. Time Parameters

It is often that any journey will be affected by its time, in this project we classify the time as:

1. Rush hours: there is usually traffic congestion at these hours and thus the car average speed will be affected by this.

The rush hours in Hebron city are usually as follows

- Every day except Holidays from 2:00 to 3:00 at evening and from 8:00 to 9:00 at morning.

- At Ramadan Month before the end of fasting by nearly one hour (Before Adan-Almagreb).

-In the solemnity days we expect that there is traffic congestion all the day. (Solemnity in Islam is Al-fetter and Al-Adha at a specific date in Hijri calendar.

2. Non Rush hours (Normal times): at these hours there is no traffic congestion

We expect that at week end we don't have rush hours so no traffic congestion will occur at any time in the day. (In Hebron the week end is at Friday and Saturday).

In the chapter 4 we will determine the relation between these parameters, taking Hebron city state as the basic of our analysis.

2.6 Time and Cost Calculations

The goal is to find the best path from one source point to a specific destination, but how to decide which route is better?

A normal customer may consider the best route as the route that take the least time, other customer may be interested in the path of the least cost. In this system we will give the customer the ability to choose suitable criteria, then the system will find the best route according to that criteria.

As we said the car speed and the distance between any two points must be known, from these information and by considering all the parameters in section 2.2 the program can calculate the time required to travel between any two adjacent points by the equation Time=Distance/Speed. The total time is equal to the cumulative time from the source to the destination.

We consider the cost between any two adjacent points as the price of the fuel amount which the car consumes when travelling between these two points plus a constant as a Profit margin.

As we said we focus on the state of Hebron and the normal small rent cars and thus cars consume a constant amount of fuel for a given distance, then by knowing the price of one liter fuel, the program will calculate the cost between any two adjacent points by multiplying the fuel amount in liters by the price of one liter, then the total cost is the cumulative cost from the source to the destination plus a profit rate.

Price of the fuel, the amount of fuel consumed in a given distance, the car speed, and every other needed values are put in a setup file, Thus if any value is changed this change occur on the setup file. In the next chapter we will show this file as example on our virtual city.

2.7 Best Path

Again, the goal is to find the best path, either the shortest path with least cost, or the shortest path with least time.

A large network contains so many possibilities. This required finding some method to exclude most of them when applicable. Several algorithms have been developed for finding the shortest (or least-cost) path through a network. Network Analyst uses the most well-known of these, developed by E.W.Dijkstra.

In this section we will give a brief description about the dijkstra and how it works.

2.7.1 What is Dijkstra?

Conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms. [5]

2.7.2 Algorithm process

Given an origin and a destination node, how to find the shortest path? By Executing Dijkstra algorithm we will achieve this.

1. Assign to every node a distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes as unvisited. Set initial node as current.
3. For current node, consider all its unvisited neighbors and calculate their *tentative* distance. For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be 6+2=8. If this distance is less than the previously recorded distance, overwrite the distance.
4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
5. If all nodes have been visited, finish. Otherwise, set the unvisited node with the smallest distance (from the initial node, considering all nodes in graph) as the next "current node" and continue from step 3. [5]

The following diagram shows a network in which the algorithm will find the shortest path from node A to node G.

After applying the Dijkstra algorithm on the following diagram the shortest path from A to G is (A→B→C→G) the cumulative cost through this path is the least and it is equal 15.

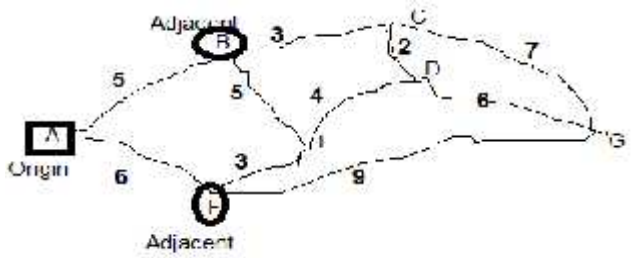

Figure 2.2 Dijkstra algorithm example

2.7.3 Complexity of Dijkstra's Algorithm

With adjacency matrix representation, the running time is O(n2) By using an adjacency list representation and a partially ordered tree data structure for organizing the set vertices, the complexity can be shown to be O(elog n) where e is the number of edges and n is the number of vertices in the digraph. [6]

To improve the performance of dijkstra and makes the search operations faster, a data structure as a sparse matrix could be used.

A sparse matrix is a matrix that allows special techniques to take advantage of the large number of zero elements. This definition helps to define "how many" zeros a matrix needs in order to be "sparse." The answer is that it depends on what the structure of the matrix is, and what you want to do with it. [7]

2.8 Summary

This chapter covered different topics that are related to Routing and factors which affects on it.

CHAPTER THREE 3

Design Concepts

3.1 Overview

3.2 System's Definition

3.3 System components

3.4 Use Cases

6.5 Interface Design

3.6 Database

3.7 User Inputs

3.8 System Outputs

3.9 System Functions

3.10 System Update

3.11 Simulation

3.12 Summary

3.1 Overview

This chapter involves our project objectives in details.

3.2 System's Definition

This software is designed as a guide to help the customer get to his destination safely and easily. We consider this to be the quickest and easiest (but not necessarily the shortest) route, based on average off-peak and on-peak driving conditions

3.3 System components

The main components of the system are:

1. SQL server database.

2. User Inputs: Criteria, Source, Destination.

3. System Outputs: Best path, best taxi office, required time or cost.

3.4 Use Cases

3.4.1 Actors:

- User: any person who use the system.

- Administrator: the person who owns the system, he has the ability to change the database values, and controls the system.

3.4.2 Use Cases Diagram

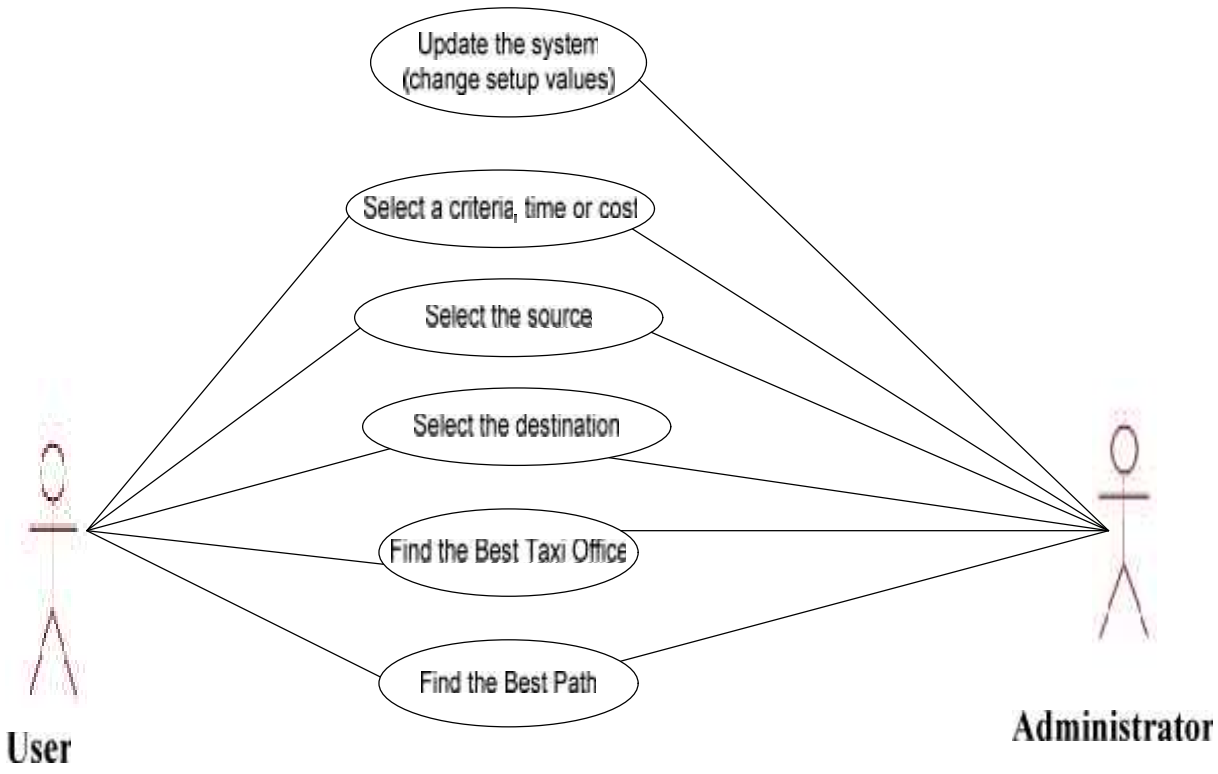

Figure 3.1 Use Cases

3.4.3 Use Cases Description

Use case1: Select the criteria

1. The user selects criteria, either time or cost from the criteria combo box.
2. The system use this input to decide whether to calculate time or cost.
3. The system uses dijkstra in term of cost or time according to the user choice.

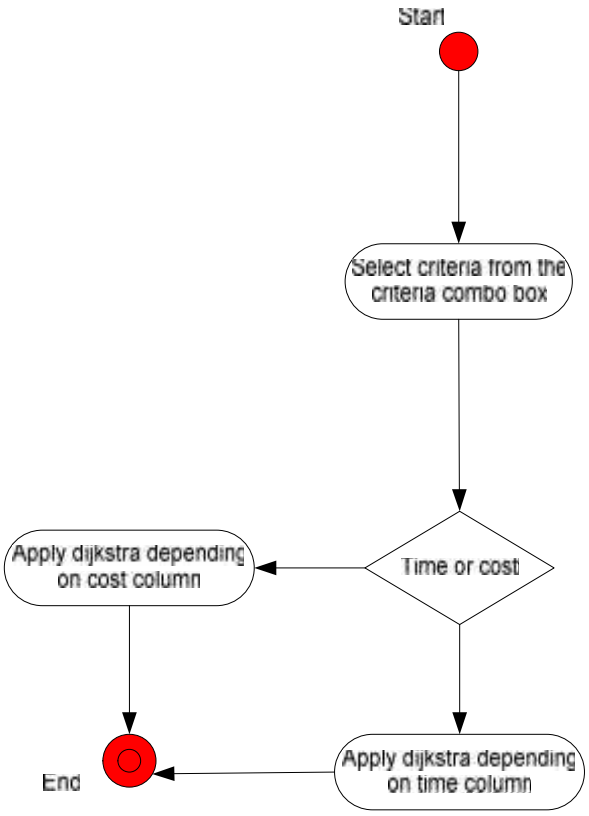

Figure 3.2 Activity diagram use case1

Use case2: Select the source

1. The user selects the source from a list of stations names in the source combo box.
2. The system goes to the source table in the updated database.
3. The system applies dijkstra taking this table as the source node.

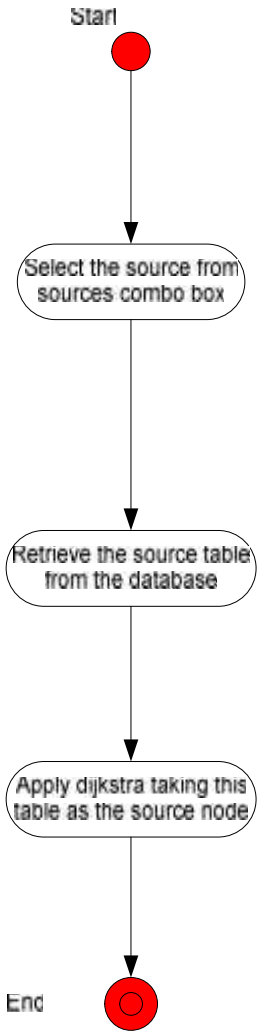

Figure3.3 Activity diagram use case2

Use case3: select the destination

1. The user selects the destination from a list of stations names in the destination combo box.
2. The system goes to the corresponding table in the updated database.
3. The system applies dijkstra considering this table as the destination node.

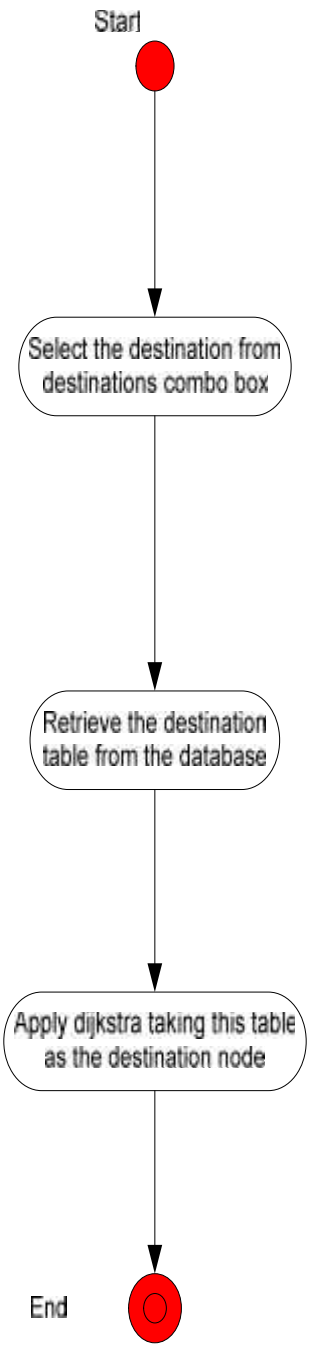

Figure3.4 Activity diagram use case3

Use case4: Find best taxi office

1. User has select criteria either time or cost.

2. User chooses a source from the list of stations names.

3. User presses a button for finding the best taxi office,

4. After the user press this button the system will show the name of the best rental car office to the user's location and some information which helps the user to call this office and rent a car, also the system will show the required time or cost needed for the taxi to reach the user.

Use case5: Find the best path

1. User has select criteria either time or cost.

2. User chooses a source from the list of stations names.

3. User presses a button for finding the best taxi office,

4. The user selects the name of the destination from the list of stations names.

5. The user press the button for finding the best path

6. After the user press this button the system will show the best path from the user's location to the selected destination, also the system will show the required time or cost needed for the taxi to reach the user.

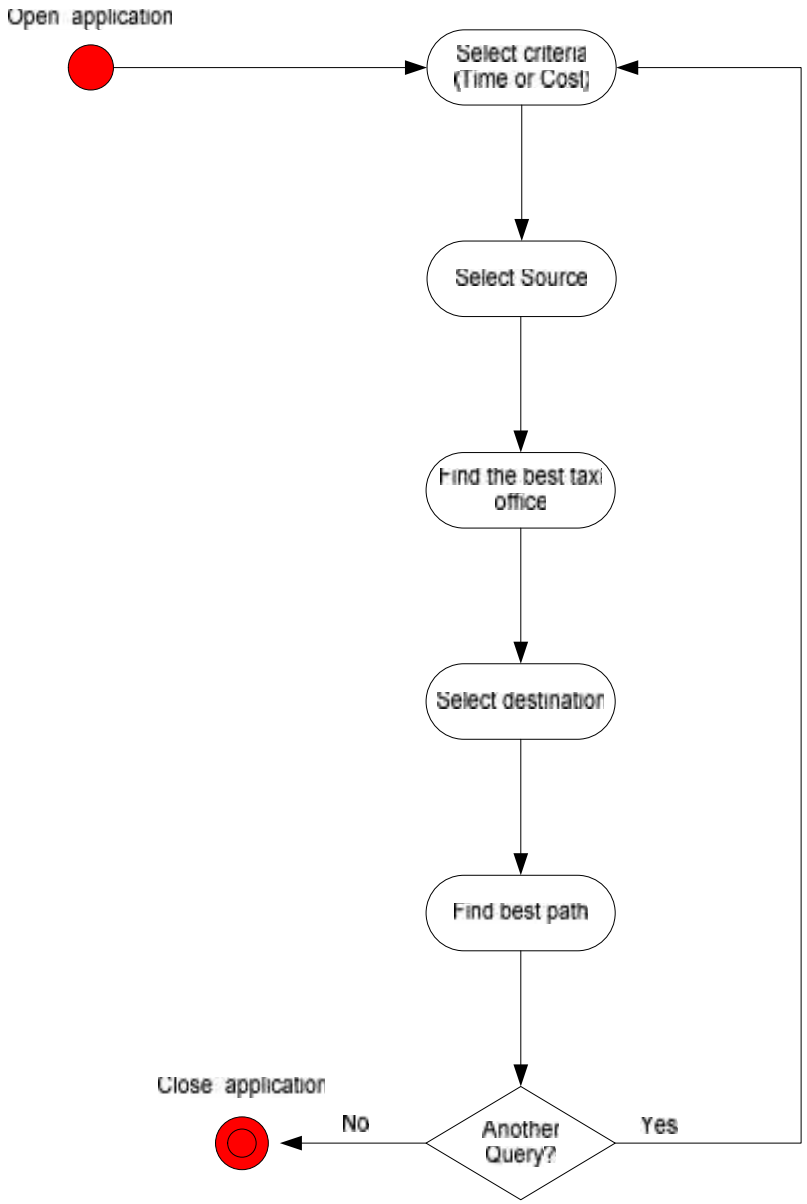

Figure 3.5 Activity Diagram use case 4, and 5

Use case6: Update the system

1. The administrator presses the update button he will be able to change any setup value (for example average speed value or cost value).

2. The system updates this value in the database.

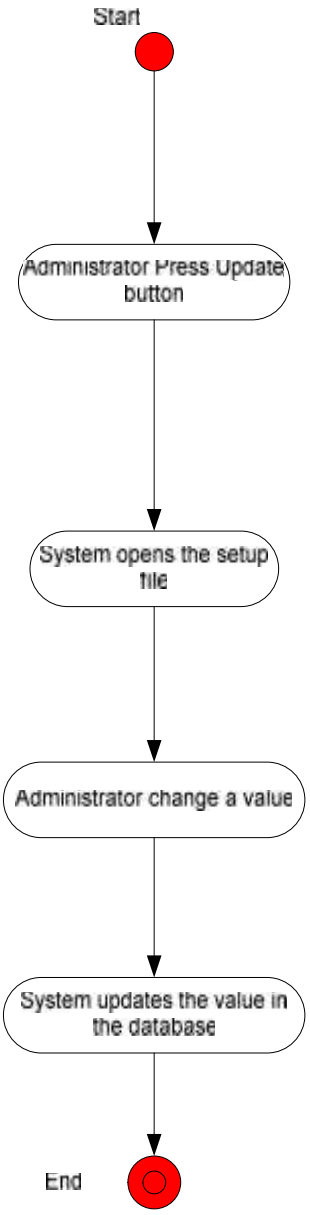

Figure 3.6 Activity Diagram use case 6

3.5 Interface Design

The following figure shows the design of the main screen.

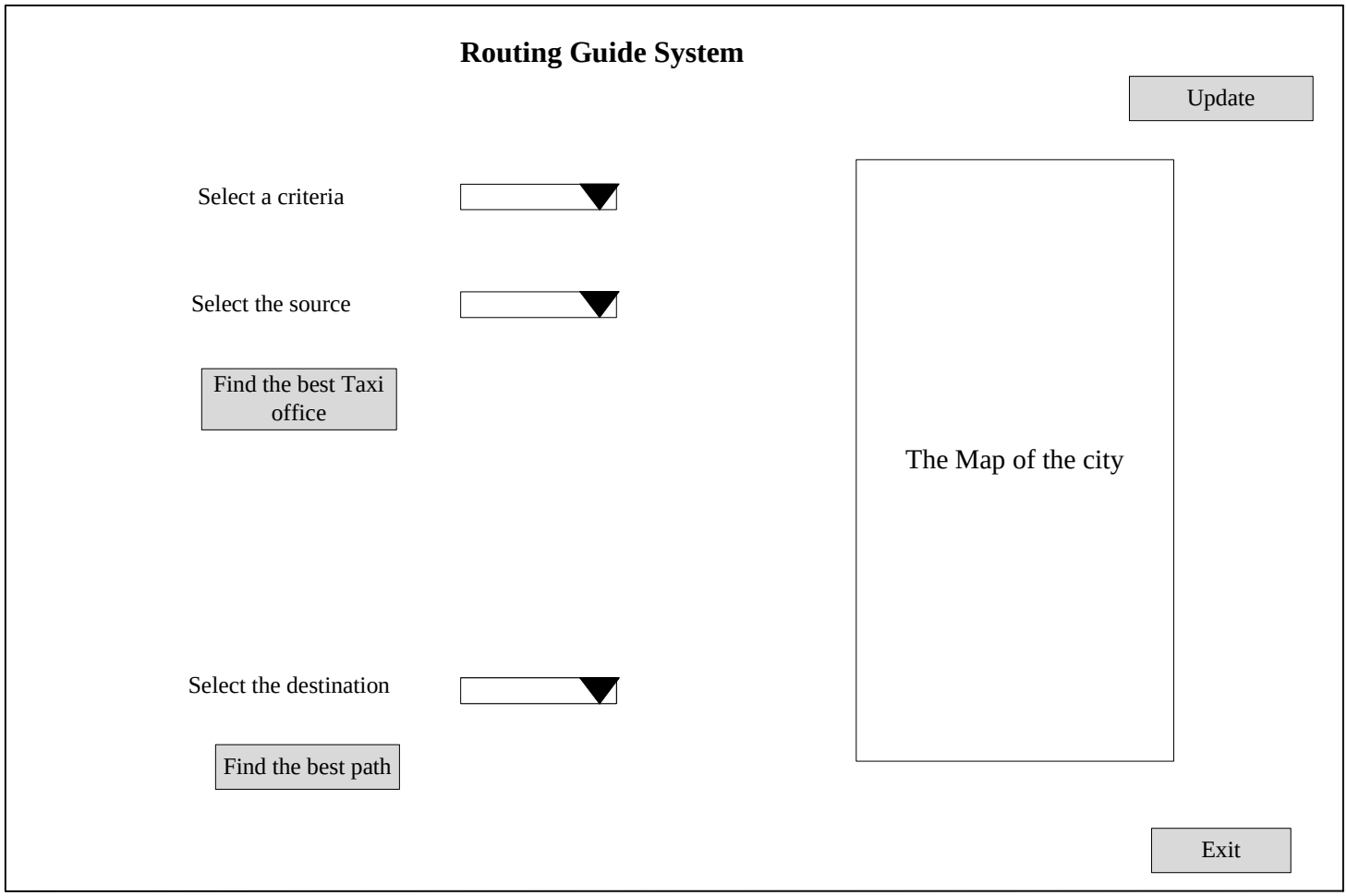

Figure 3.7 Main Screen Design

The update button is for the administrator use, when the administrator press this button the following screen appears.

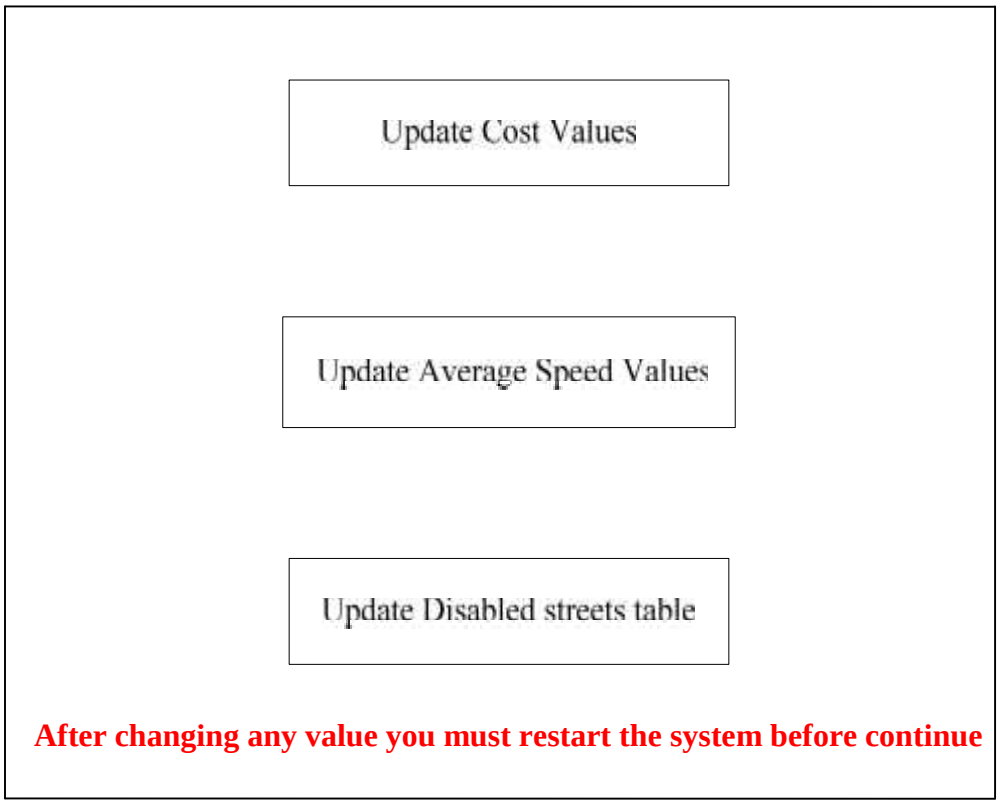

Figure 3.8 Update Screen Design

3.6 Database

After converting the map of the city to a numeric data describing each point, all of this information is implemented as tables in a database which we call the original database.

3.6.1 Database Definition:

This system used SQL Server Management Studio database for storing map information and setup values, also the updated information will be stored to it, it consists from three groups of tables; nodes tables, setup tables and other tables for interface options.

Nodes tables: table for each position in the map, either for an intersection point or for a main station.

Setup tables: tables to store constant values needed for cost and time calculations.

Interface tables: to store positions for each node in the form, and names of known places in order to build a readable and usable interface.

As the program starts execution, it will go through all tables and get the needed information from the nodes and setup tables, then calculate time and cost depending on these information, then storing the results in each node table.

3.6.2 Database Design

First group: Nodes tables

1. Node Table: describe each position in the map.
2. All-Nodes table contains the names of all nodes tables.

Node table	
PK	Node Name
	Adjacency Distance Street ID Street Type Time Cost

Table 3.1: Node table

All-nodes	
PK	Node Name

Table3.2: All-nodes table

Second group: Setup tables

1. Average speed table: contain the average speed values, which are needed in time and cost calculations.
2. Cost table: contain cost values, which are needed in cost calculations.
3. Disabled streets table: contain all streets ID's to determine which street is disabled at a certain moment.
4. Special cases table: specify some special rush hours such as in Ramadan.

Table 3.3: Average speed

Average Speed	
	Street Type Rush Hours Average Speed Distance Per Liter

Table 3.4: Cost

Cost	
PK	Liter Fuel Price
	Profit Margin

Table 3.5: Disabled Streets

Disabled streets	
PK	Street ID
	Disabled

Table 3.6: Special cases

Special cases(Ramadan)	
	Special Rush Hours

Third group: interface tables

1. Places table: include the name for each main station, and its related symbol or node name.
2. Offices table: include the name for each rental car office and its related symbol or node name.
3. Positions table: contain the form x and y positions for each location in the map.
4. Disabled streets Positions table: contain the form x and y positions for the center of each street in the map.

Table 3.7: Places

Places	
PK	Symbol
	Place name

Table 3.8: Offices

Offices	
PK	Symbol
	Taxi Office name

Table 3.9: Positions

Positions	
PK	Symbol
	Position X Position Y

Table 3.10: Disables streets positions

Disabled streets positions	
PK	Street ID
	Position X Position Y

3.6.3 Database Relations

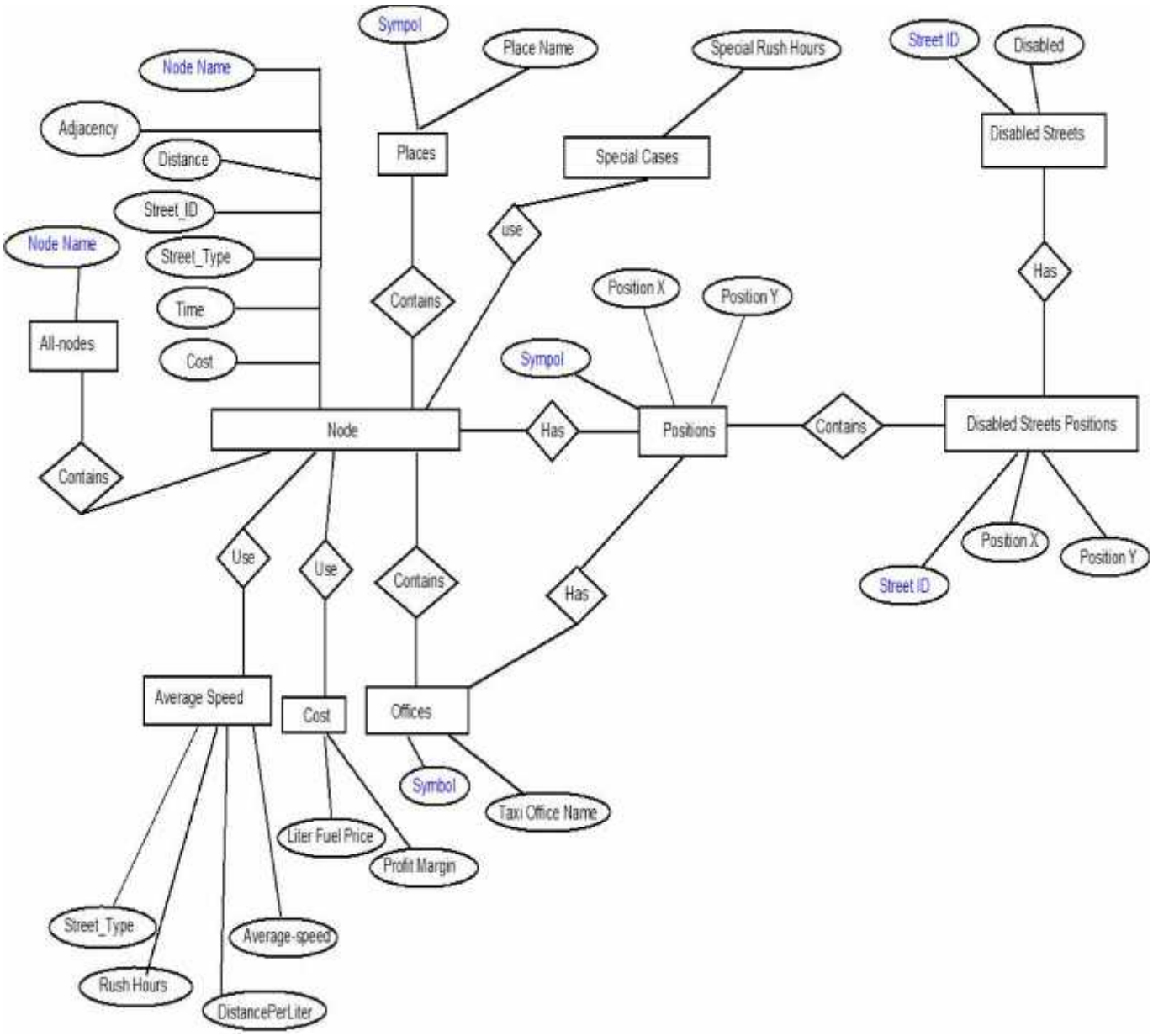

Figure3.9 Database relations

3.7 User Inputs

The system will receive three Inputs from the user:

1. Criteria (time or cost), if the user is interested in time or cost..
2. Source name (current location), the user will choose his current location from a list of stations that located in the given map.
3. Destination name (the desired location), the user will choose the destination name from a list of stations that located in the given map.

3.8 System Outputs

The system has three outputs:

1. Best Rental car office, the system uses two user inputs (criteria, source) to find the best taxi office

2. Best Path, the system uses three user inputs (criteria, source and destination) to find the best path.

3. Total time or Total cost, if the criteria input is time the system will show the required time to reach the detonation and the required time for the taxi to reach the user, else it will show the required cost

3.9 System Functions

These are the main functions of the system:

1. Make a connection with the original database.

2. Update this database filling the time and cost columns.

3. Receive 3 inputs from the user (Source, destination, criteria.)

4. Return three outputs depending on the user inputs (Best Path and Taxi office, Time or cost).

3.9.1 Database connection

To create the connection between SQL server and Netbeans

1. Load sql server driver.
2. Determine local host.
3. Determine database
4. Determine user and password refer Username and Password that used to connect to the SQL Server.

3.9.2 Update database

Update all the node table means to fill time and cost columns with values depending on current time and other options.

The names of all nodes are placed in a table which we call All-Nodes table (table 3.2), get the node table name from the all-nodes table then update that table.

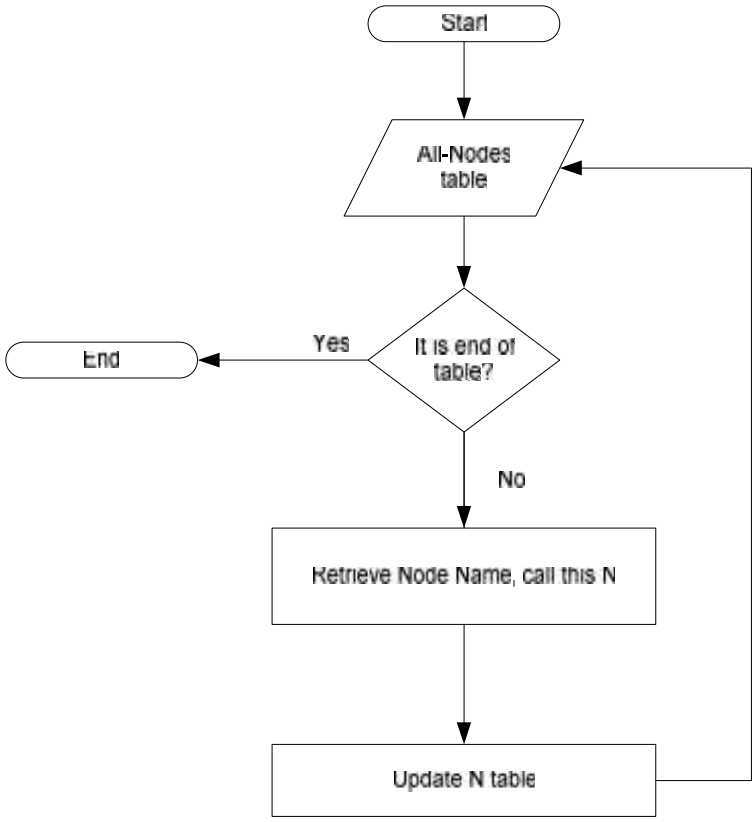

Figure 3.10 Update the database

3.9.3 Finding Best Rental Car Office

In any city there may exist one or many rental car offices, as we said in chapter one, one of the project goals is to tell the customer about the best rental car office.

Again, the best rental car office for any customer is either the closest one which can reach the customer location with the least time, or the cheapest one which can reach the customer in the least cost.

At the moment the user insert his location; the program will determine the locations of all rental car offices in the city as described in the original database.

We assume that when the customer calling the rent office in order to rent a car, the car will be sent from the office location to the customer location. So customer location is the destination node, and the rental car office is the source.

Applying Dijkstra Algorithm each time take one rent car, and compute the total time (or cost). Then compare all results and take the least one. The program will show to the user the rent car office which has the least cumulative time or cost.

All the offices names are stored in a table which we call Offices table (table 3.8).

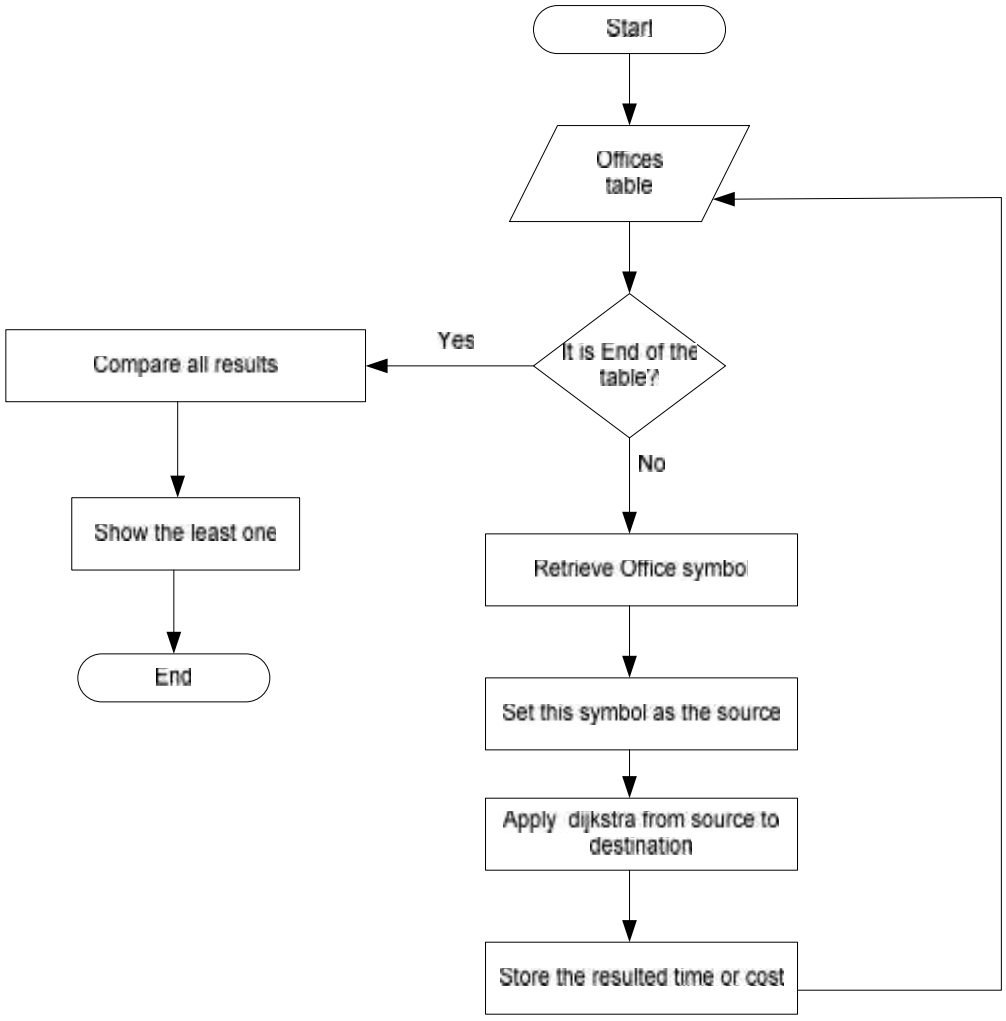

Figure 3.11 Finding the best taxi office

3.9.4 Finding Best Path

To find the best path from a given source node to any destination, we applied dijkstra algorithm from the source node table until reach the destination node table the detailed algorithm is shown in appendix A.

3.10System Update

3.10.1 Sudden events during the journey

The system must treat with any sudden events that may affect the journey, as an accident or a closed street for any reason. As this occurs, the update process will take place.

Here we mean by update is to change some values in the database to treat with the new conditions

In this system we expect to have a disabled street at any time, so the update process will update the disabled streets table (table 3.5).

Real system gets this information from GPS system.

Set value of 1 in the disabled column where street id is for the disabled one.

3.10.2 Update setup values

The update process occurs if one of the constant values in tables 3.3, 3.4, 3.6 changes at any time, the system enables the administrator to change these values from a setup file, the detailed algorithm is shown in appendix A.

3.11 Simulation

As a Simulation of the real life, we build a virtual city from many of Hebron city streets, we analyze this virtual city map in the same way as we described in chapter two.

In real system GPS device can determine the location at any time, in our simulation we start from this point as this point is described in the original database, and from its adjacent point it continues until it reaches the destination.

3.11.1 Streets Description

In the simulation we choose the streets as they have many intersection points, then it could be more than one path between a given source and destination.

As we said in section 2.3 for each street segment between any two nodes, the street length, the type and the street itself must be known.

Here we will show the streets which we choose to build the virtual city, and for each street give an ID and all the needed information. The following graph shows a map consisting of some Hebron streets centerlines. (Center line is a line along the street that divides the street into two equal parts).

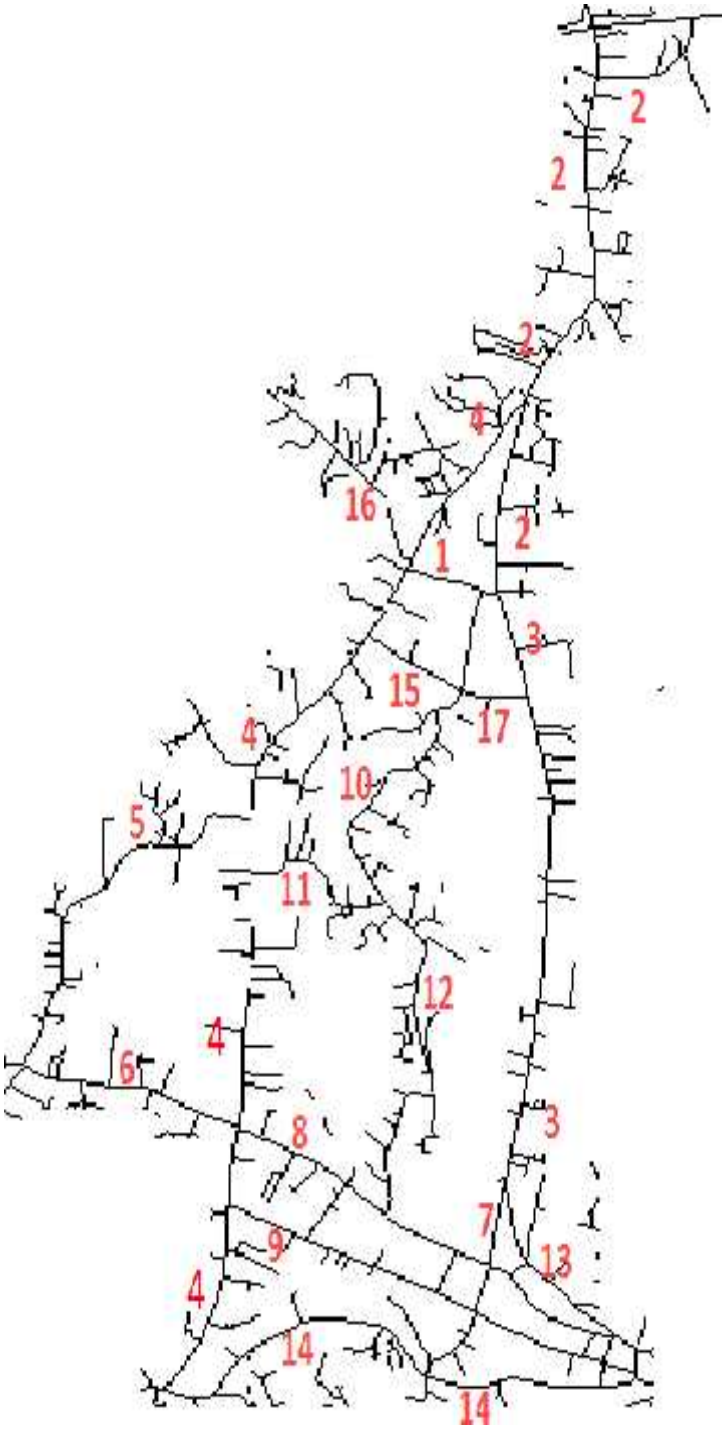

Figure 3.12 Streets Centerlines

The following table describes the map in figure 4.1 in the same way as in chapter two.

Table 3.11 Streets Description

ID	Street Name	Name In Arabic	Street Length	Street Type
1	-	-	280	Major
2	Al-Quds Street	شارع القدس	1907	Major
3	Eain-Sara Street	شارع عين ساره	1818	Major
4	Al-Salam Street	شارع السلام	3272	Major
5	Khalet-Almagareba	شارع خلة المغاربه	1160	Main
6	AL- Maskoubia	شارع المسكوبية	670	Major
7	Al-Manara	شارع المنارة	410	Major
8	King Housein	شارع الملك حسين	851	Major
9	Al-Adel Street	شارع العدل	1298	Major
10	Abu-Ganam	شارع أبو غنام	1237	Main
11	Al-Yarza	شارع اليرزة	494	Main
12	Ebn-Khaldoon	شارع ابن خلدون	1050	Main
13	King Feisal	شارع الملك فيصل	750	Major
14	Beer-Alsaba	شارع بئر السبع	1583	Major
15	-		321	Main
16	-		270	Secondary1
17	-		200	Main

Note: Figure 37, table 3.11 Street names, lengths, and types are taken from a shape file of Hebron streets centerlines. We obtain this shape file from the ***Surveying and Geometrics Engineering Department*** *in Palestine Polytechnic University.*

A Shapefile is a digital vector (non-topological) storage format for storing geometric location and associated attribute information. The Shapefile format is created by ArcView and can be used by ArcView, ARC/INFO, ArcGIS and other widely used GIS software. [8]

3.11.2 Intersection points

As we said in chapter two the routing occur at the intersection points between the streets. Figure 4.1 shows a map of the virtual city that we have with all of its streets. Here we will determine the position of every intersection point, then describe it with all needed information, each point forming a table of the original database.

The following graph shows the positions of all intersection points between any two intersecting streets, giving a name to each point.

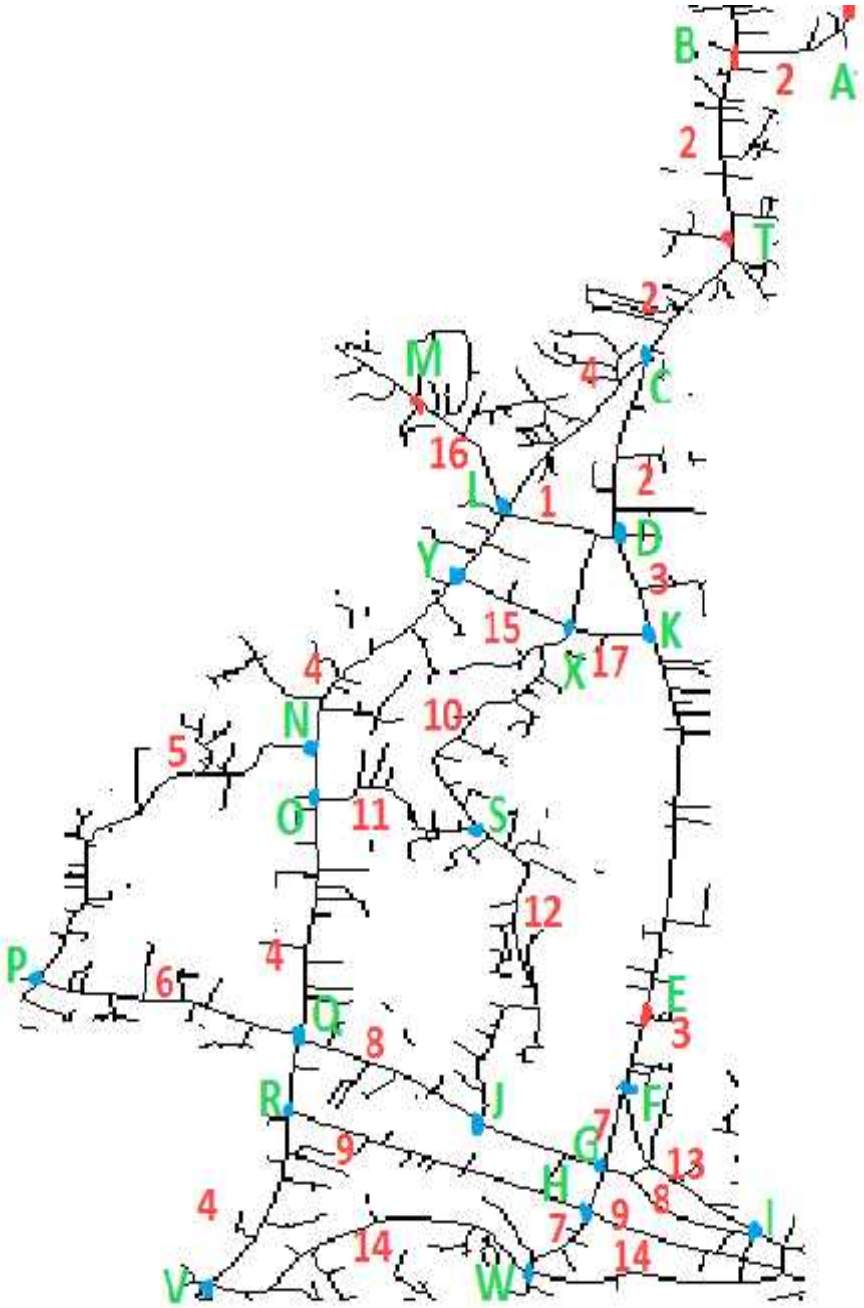

Figure 3.13 Intersection points

3.11.3 The Original Database

In the Original database each intersection point is implemented by a table, with number of rows equal to the number of all points in the map, and with seven columns as follows

Column1: Node; contain all the nodes in the map

Column2: Adjacency; if the node in column one is adjacent to the table node; the adjacency is equal to 1 else it is equal to 0.

Column3: Distance; at each row if the adjacency is equal to 1 the distance between the two adjacent nodes is placed in this column. We obtain these values from Google Earth and from the Hebron shape file. (To read more about Google Earth and Shape file, read section 4.2 in chapter four).

Column4: Street ID; we give each street a unique ID, place this in the database is useful because there might be a problem during the journey, and one of the streets may be disabled for any reason (for example a sudden accident) So the system must be able find a path without passing through the disabled street.

In real application a GPS system will determine all streets which are disabled at a given time. In our simulation to a real system we include a table in a setup file which contains all streets with value 1 in the disabled column. As shown in table 3.2

Table 3.12 Disabled Streets

Street ID	Disabled
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0

At the moment a GPS system determine a disabled street, the value of the column disabled will be 1 at that street ID, and then the system will ignore those streets when finding a path.

Column5: Street Type; as we said in chapter two we have 5 types of street, (for simplicity in our database we give each type a number).

Major: 1

Main: 2

Secondary1:3

Secondary2: 4

Unpaved streets: 5

Column 6: Time; the system will calculate time required between any two adjacent points by using the equation Time=Distance/Speed but how to determine the speed? In the next section we will illustrate that.

Column 7: Cost; like the time the system will calculate the cost using the equation Cost=Amount of fuel consumed to travel between these two points multiplied by the price of a liter of fuel. Again this will be illustrated in the next section.

Finally each node in the original database must have a table.

Table3.13 Node Table in the original database

Node Name	Adjacency(0 or 1)	Distance(m)	Street ID (1 to # of streets)	Street Type(1,2,3,4 or 5)	Time	Cost
Node1						
Node2						
. . .						
Node n						

Take Node S as example:

Table 3.14 Table of S in the Original Database

Node	Adjacency	Distance	Street ID	Street Type	Time	Cost
A	0					
B	0					
C	0					
D	0					
E	0					
F	0					
G	0					
H	0					
I	0					
J	1	1050	12	2		
K	0					
L	0					
M	0					
N	0					
O	1	575	11	2		
P	0					
Q	0					
R	0					
S	0					
T	0					
V	0					
W	0					
X	1	900	10	2		
Y	0					

Here S is adjacent to three point J, O; X as shown in figure 4.2, and thus adjacency is equal to 1 at these three points.

3.11.4 Update the Original Database

Original database is consisting of a number of tables; each describes one point, in our case the original database is consisting of 24 tables like table 3.3.

At the moment the program is executed it will update the original database by filling the two columns of time and cost.

3.11.4.1 Time

Again; to fill the time column we need to know the car speed, the car speed is determined by the street type and by the time of the journey.

Table 3.15 Average Speed

Street Type	Rental Car Average Speed(Km/h)	
	Rush Hours Speed	Daily Speed
Major Streets	30	60
Main Streets	30	60
Secondary Streets1	40	50
Secondary Streets2	40	40
Unpaved Streets	20	20

*We obtain these values from the **Surveying and Geometrics Engineering Department** in Palestine polytechnic University.*

Again all these values are placed in a table in a setup file, thus it could be changed any time a change is occur.

The system will go through each table; for each table if find 1 in adjacency column it calculates the time, taking the speed determined by the value in column street type and the time at which the program is executed, this time is determined by the hour, day, and date, these values are taken from the device on which the program is being executed.

Then simply the system will calculate the time for each row the adjacency is equal 1.by dividing the distance over the average speed.

3.11.4.2 Cost

To fill the cost column the system needs to know the amount of fuel consumed when travelling between two nodes with a given speed.

We know that a car consumes fuel depending on the distance travelled and its speed; table 4.5 shows these values

Table 3.16 Cost

Car Speed (Km/hour)	Distance to consume 1 liter(km)
60-100	15
20-50	10

These values depend on the car itself; here we focus on the small rent cars that are usually work in Hebron; and by making questioners asking some of the drivers we get these values as the average of driver's answers.

Then the system will calculate the cost each time it finds adjacency equal one by:

1. Find the amount of fuel consumed dividing the distance column over 15 or 10; it depends on the speed which depends on the street type and the time as we said.
2. Multiplying the value from step one by the price of one liter fuel.
3. Multiply the result from two by a profit margin.
4. Add the result from 2 to the result from 3

Again; values in table 3.5 and the fuel price are input to the system via a setup file; and thus it can be changed any time.

3.11.5 Main Stations at the Virtual City

Through all the streets in figure 3.2, we have some of main stations in Hebron; and these stations could be a source or a destination for a journey. Each location at the map may or may not be an intersection point. Figure 3.3 shows the main places in our simulation.

Each station has a table as each intersection point has; looking at figure 3.2 we obtain this table:

Table 3.17 Main Stations

Symbol	Station
A	Hebron Entrance
B	Pink Fountain
C	Al Haras
E	Al Baladya
F	Ebn Rushd
G	Al Manara
I	Bab Alzawya
K	Eain Sara
M	Hebron University
N	Al Ansar Mosque
P	Al Jalada
T	Ras Al Jora
X	Nimra
D	Rent Car1
V	Rent Car2

Table 4.6 is placed in the database, the system internally terminate with the intersection points, and then output known places to the user as in column two in table 4.7.

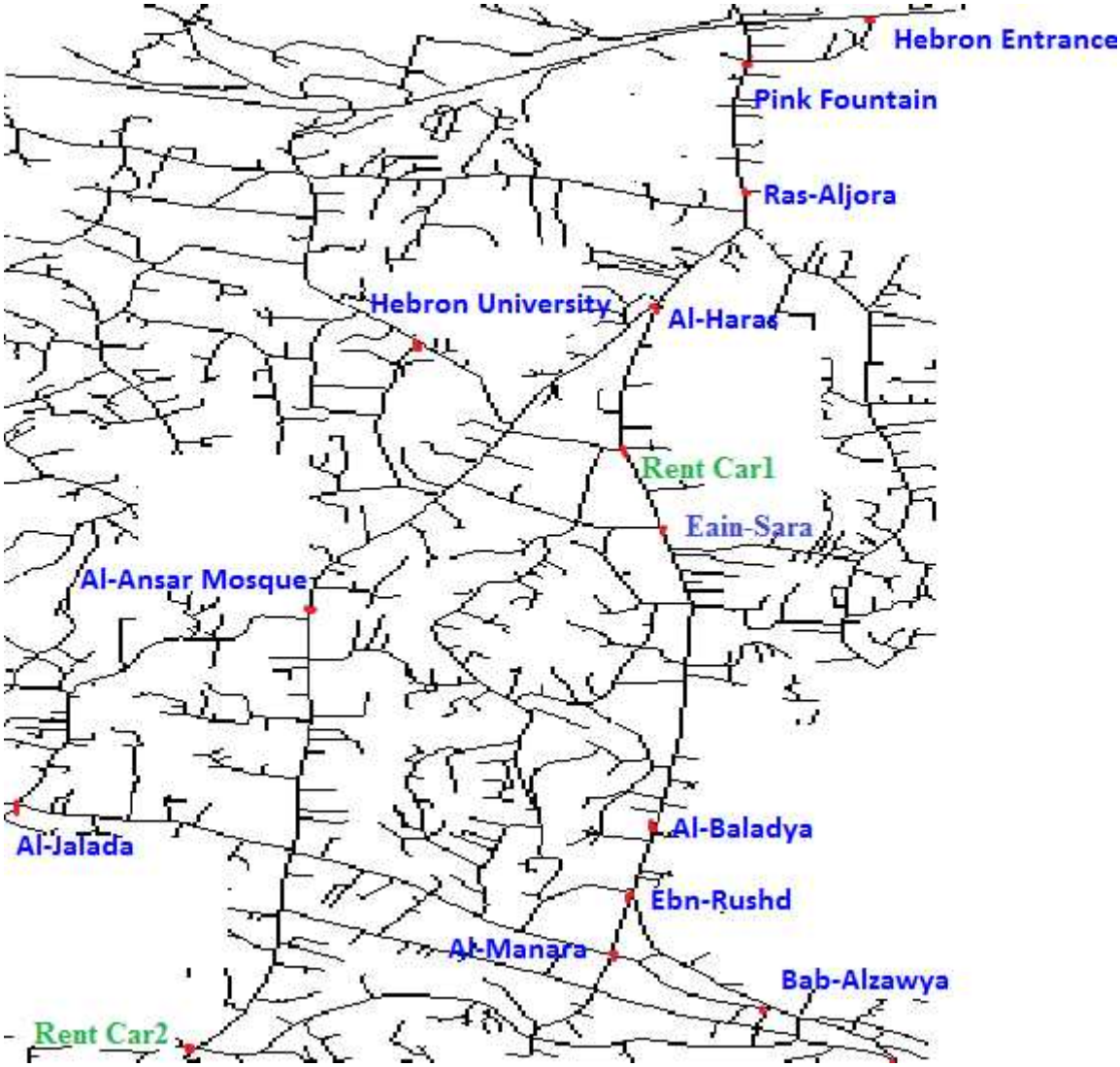

Figure 3.14 Main Stations

As we see in figure 3.9 we assume two rent offices at two different positions; in order to make the system choose the best one, their positions are just an assumption to simulate the reality in which a city may have many than one rent offices

3.12 Summary

In this chapter we exposed the system design options in more details, we talked about the GUI design and other design options, and also we give a detailed description of the simulation for a real city by building a virtual one from some real streets in Hebron.

CHAPTER FOUR 4

System Implementation and Testing

4.1 Overview

4.2 Development Environment overview

4.3 Implementation phases:

4.4 The system final windows

4.5 Real System Testing

4.6 Summary

4.1 Overview

In this chapter we will talk about the most important tasks in our system to work as needed, which is system implementation

4.2 Development Environment overview

The lower case tools which we used in developing the system can be explained as the following:

4.2.1 Setting up the hardware and operating system

The hardware equipments needed for the operation of this system are:

Personal computer and the operating systems that must be installed on PC's are one of the following OS, Microsoft Windows 7 Ultimate, Microsoft Windows 7 Home Edition, Microsoft Windows XP Professional, or Microsoft Windows Vista Home Basic.

4.2.2 Setting up the Software for the System:

The software packages that this system is built on are:

- NetBeans IDE 6.9.1

We found it flexible and easy to use. It was used for both developing the GUI of the application and for writing the source code.

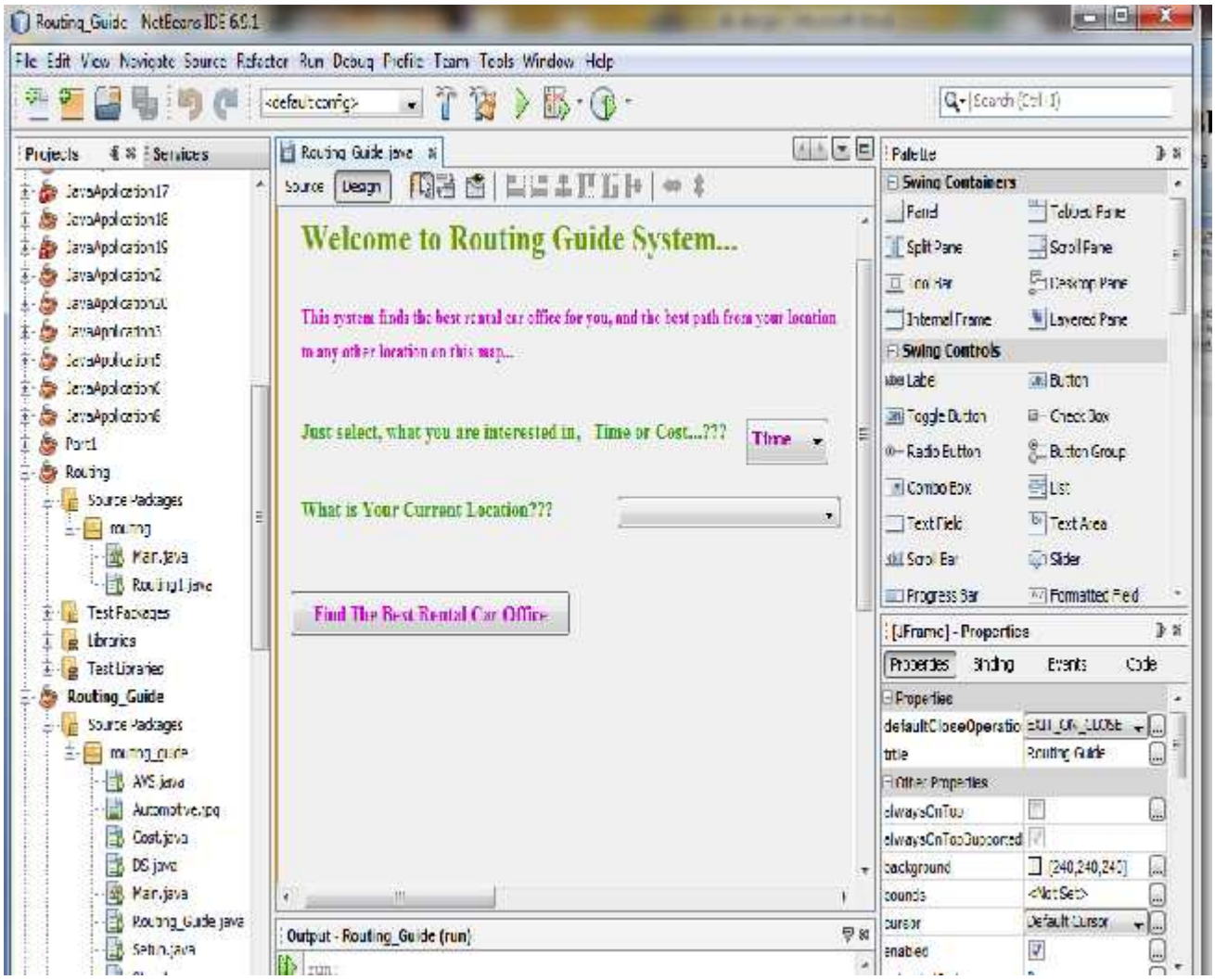

Figure 4.1: NetBeans 6.9.1 IDE

- Microsoft SQL server 2005 Management Studio: to implement the database.

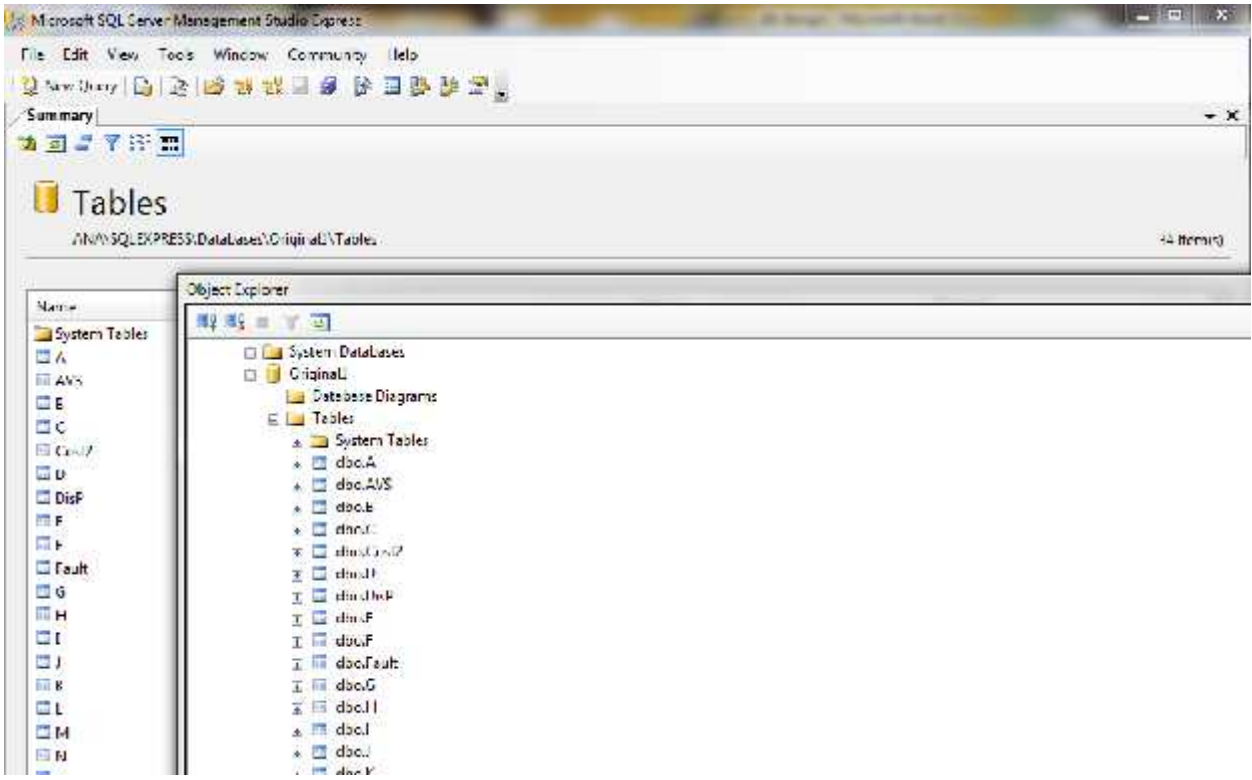

Figure 4.2: SQL Server Management Studio

- Dot NET Framework: needed for the sql server to run at the machine

- Java development kit (jdk) to compile Netbeans source code into machine code to be executable fine and in order to provide us with the libraries that we need.

.

4.3 Implementation phases:

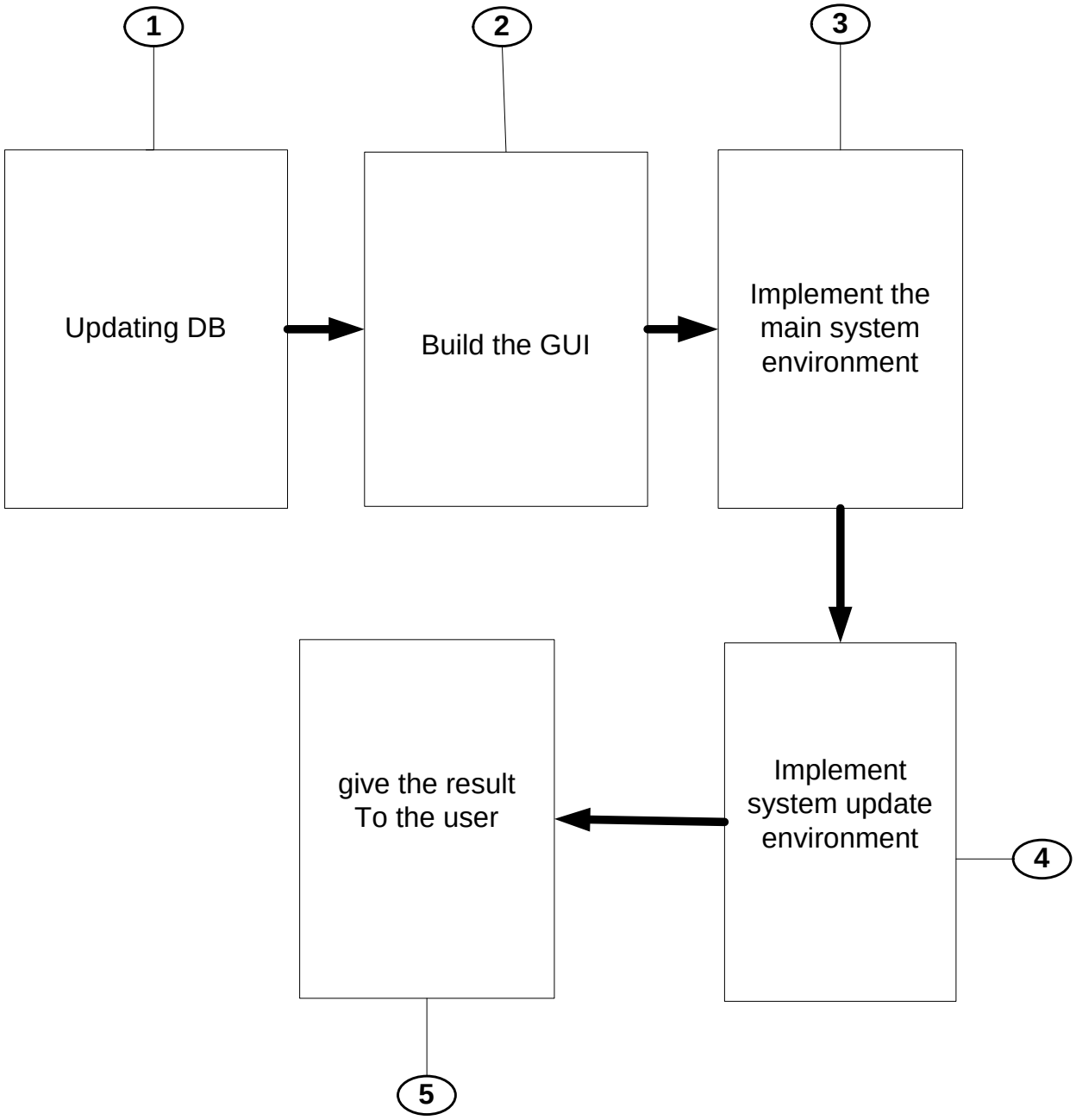

Figure 4.3: implementation phases

4.4 The system final windows

Build the GUI

The implementation of the forms in this project as following:

- **Main screen form:**

This form consists of:

1. Three combo boxes one for choosing the criteria, second for the sources lists enables the user to select one s and the third is for the destinations list enables the user to select one.

2. Three buttons:

1. To find best rental car office and required time or cost.
2. To find best path and required time or cost
3. Update button for the system administrator
4. Exit button

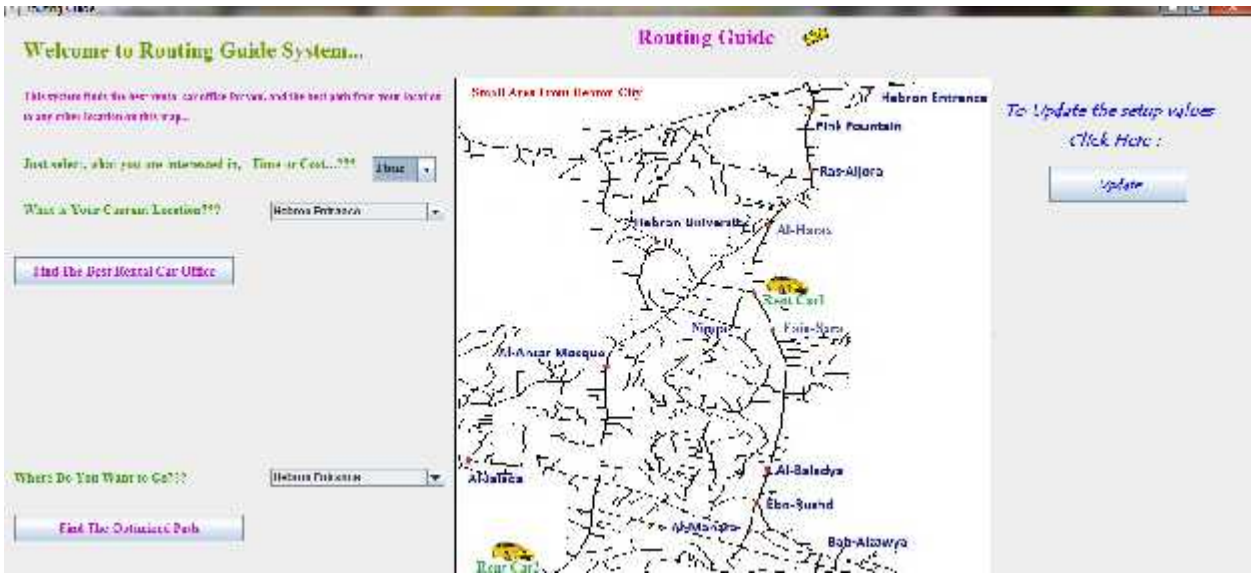

Figure 4.4 Main window

1. Button1: To find best rental car office and required time or cost.

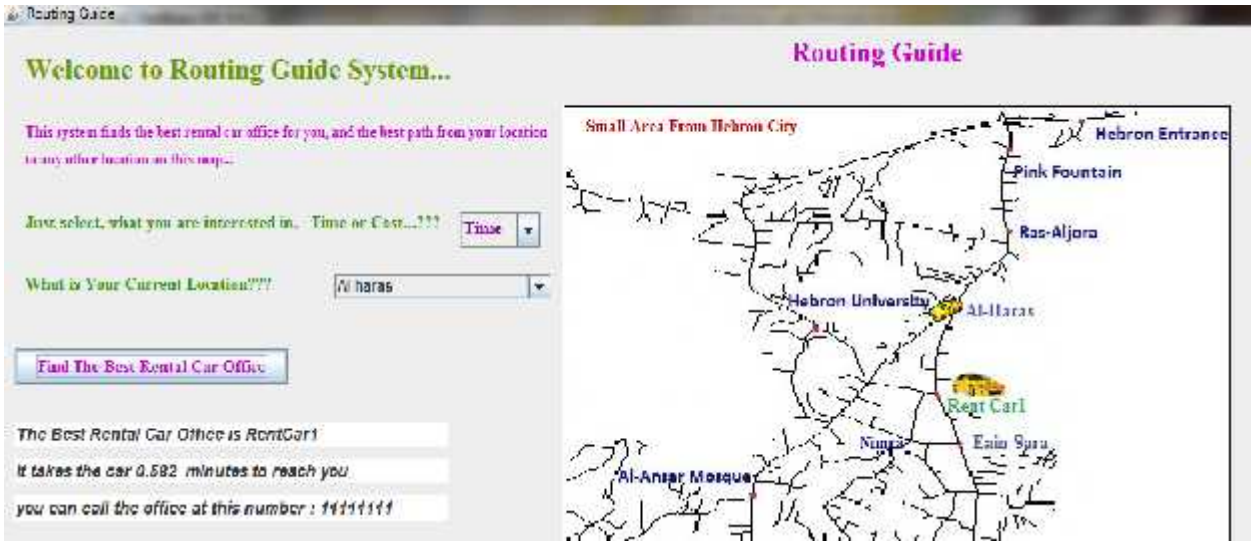

Figure 4.5 Button 1 result

2. Button2: To find best path and required time or cost

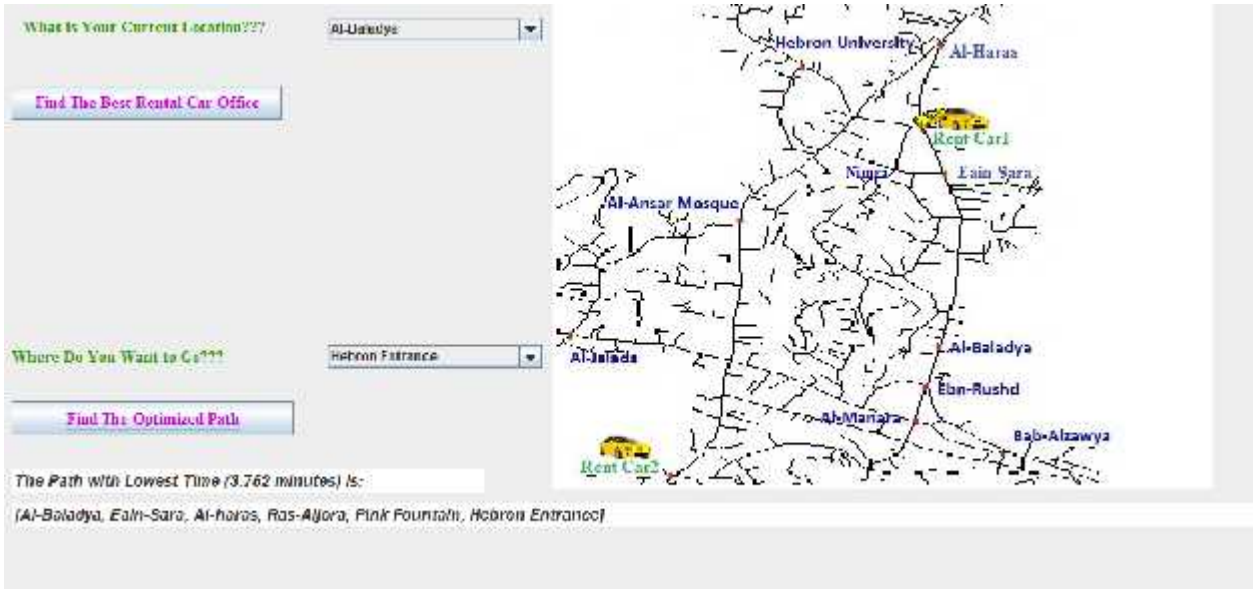

Figure 4.6 Button 2 result

3. Update Button

- System Update form: Consists of three buttons

    1. Update average speed values
    2. Update cost values
    3. Update disabled streets

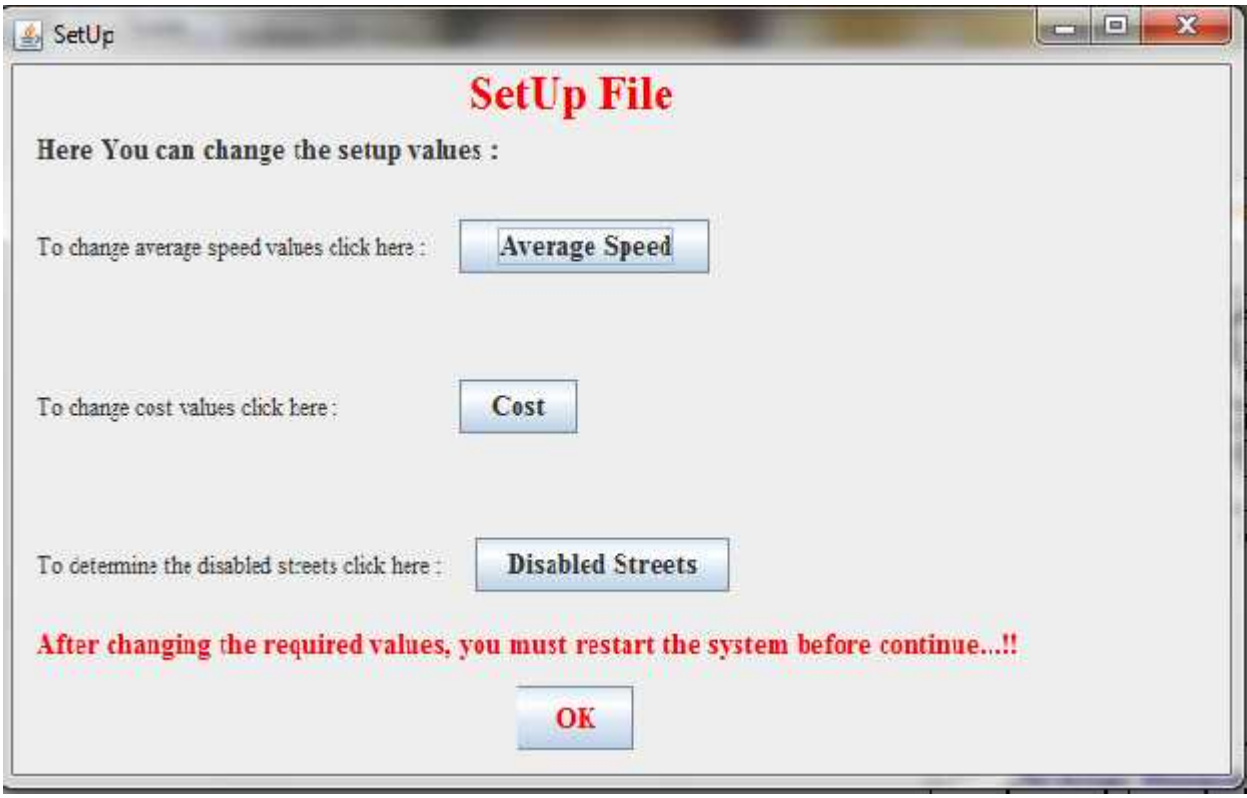

Figure 4.7 Update window

1. Average speed update

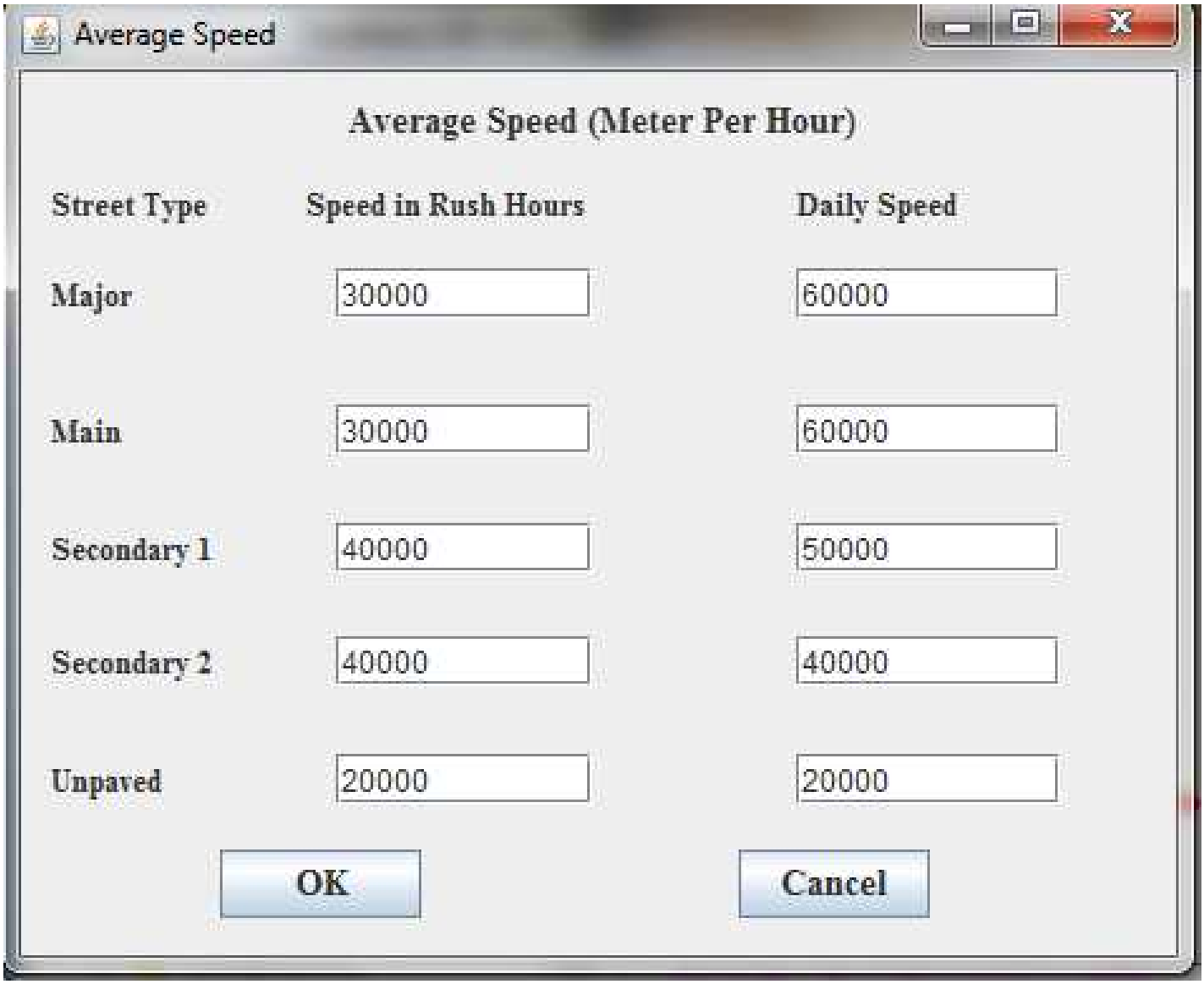

Figure 4.8 Update average speed window

2. Cost values update

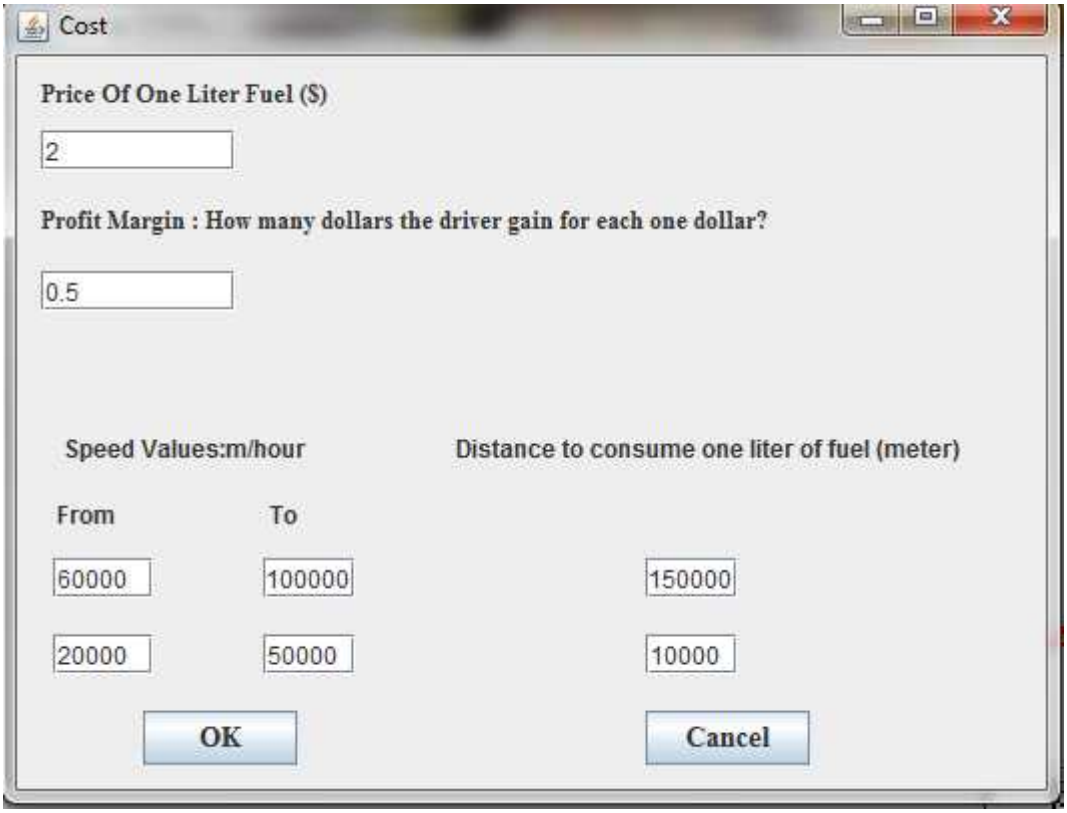

Figure 4.9 Update cost values window

3. Update disabled streets

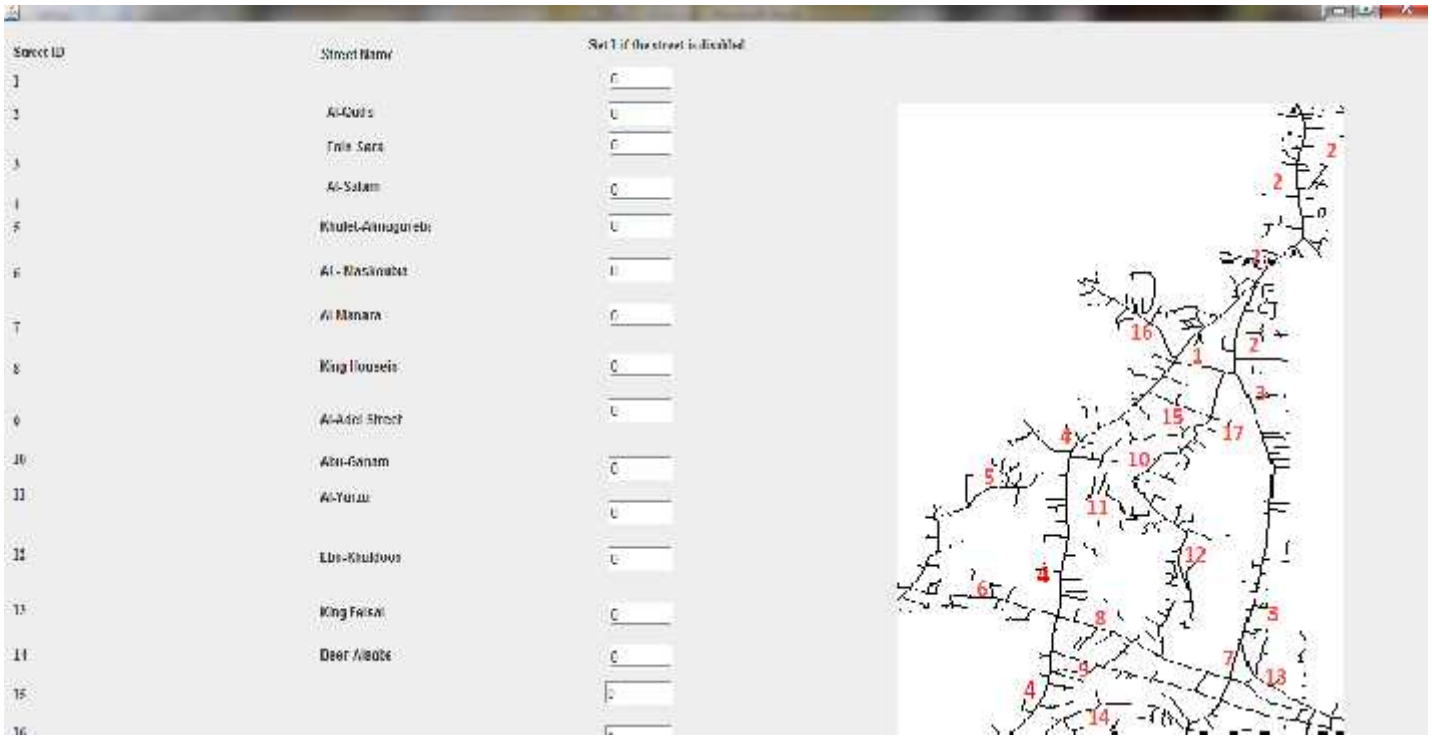

Figure 4.10 Update disabled streets window

4.5 Real System Testing

Examples:

1. Finding the best Route from Hebron entrance to the Hebron University at normal time, setting the criteria as time the result will be as follow:

Figure 4.11: testing example1

But if the system is running at a rush hour (for example 8:30 at the morning) the result will be as follow:

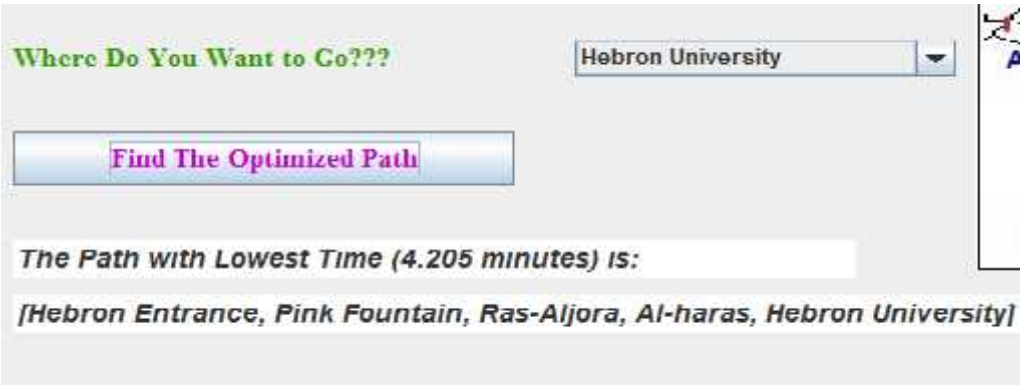

Figure 4.12: testing example1

Note that the path remains the same but the time increased.

2. Finding the best Route from Bab-Alzawya at Al-haras, setting the criteria as time the result will be as follow:

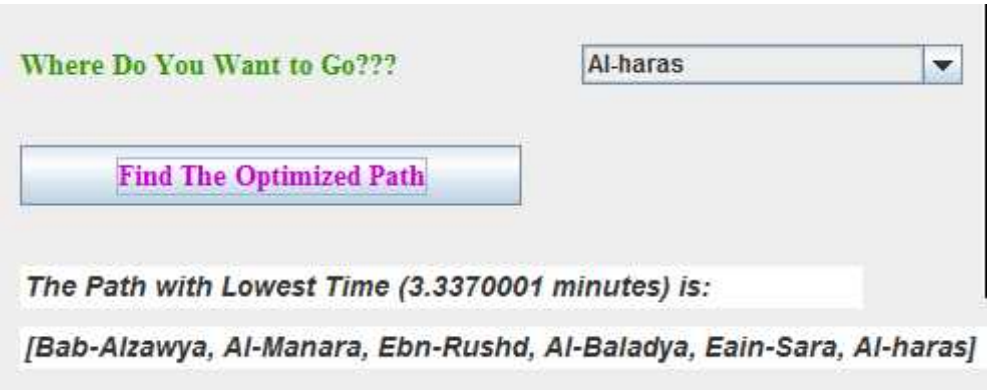

Figure 4.13: testing example2

But if Eain-Sara Street is disabled the result will be as follow

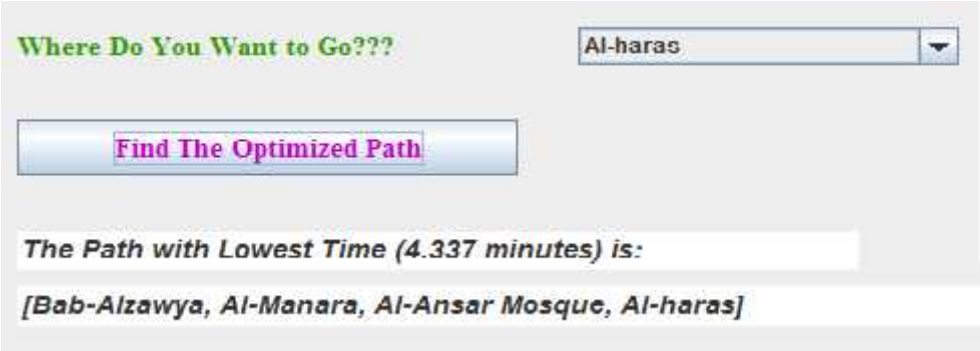

Figure 4.14 testing example2

Note that the path is changed, to pass through Al-Salam Street, and it takes more time.

4.6 Summary

In this chapter we show the implementation of the main system functions and windows as the user can easily treat with it. We also show some testing examples and compare the results.

CHAPTER FIVE 5

Future Work and Conclusion

5.1 Overview

5.2 Conclusion

5.3 Future Work

5.1 Overview

In this chapter we will talk about some of the modifications that can be done over our system in the future, and the conclusion that we derived from it.

5.2 Conclusion

- With this software tool, the user can determine optimized options for travelling anywhere, and any time in a given city.
- Each user can know at what time he/she will arrive to the desired location, and how much money the journey will cost.
- We can apply this software on any real city if the required information is available.
- This software is easy to use by any customer, because all the required information is shown to him as he run the application. For example he will see the city with the locations of all places and the car movement from any source to any destination.
- Finding the best rental car office is a good property for this software because it removes any ambiguous question the user may ask about how he could rent a car.
- After ending the system implementation it improves our performance in java programming and SQL server 2005.

5.3 Future Work

Our system can be enhanced in many ways to achieve more robust system, some of the enhancements are:

- Remove some restrictions in the system, such as extracting the original database to cover the whole Hebron city.

- To connect the system with other systems for other cities, to make a full system covering whole Palestine.

- Connect it with the GPS System in order to determine the current location, and then the system will continue until reach the destination.

- To apply the system via mobile application, the mobile could have a wireless connection with the SQL server in order to get the required information from the database. Applying the system on mobile make it easier to e used at any time, and reachable for larger number of users.

- Use the sparse matrix and graph theory to improve the system performance and design.

References

Web Sites and papers

1. What is Com 6/8/2011 http://whatis.techtarget.com/definition/0,,sid9_gci934747,00.html

2. Graph Theory 6/9/2011: http://www.graph-theory.net/

3. Esri Developer Network 6/11/2011: http://edndoc.esri.com/arcobjects/9.2/NET/e084da94-d4f7-4da7-86ed-7df684ff2144.htm

4. Route 6/11/2011: http://en.mimi.hu/gis/route.html

5. Wikipedia 6/11/2011:http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

6. E.W. Dijkstra. “A note on two problems in connection with graphs”. 1959.

7. WalfarmMathWorld, Sparse Matrix, 6/15/2011 http://mathworld.wolfram.com/SparseMatrix.html

8. USGS CMG "Shapefile" Definition 6/13/2011

http://walrus.wr.usgs.gov/infobank/programs/html/definition/shapefile.html

Appendix A Algorithms

1. Database connection

Begin:

5. *Load sql server driver ("com.microsoft.sqlserver.jdbc.SQLServerDriver");*
6. *Determine localhost: 1433 refers to connect to SQL Server at local host port 1433 (default port).*
7. *Determine database Name: Originall refers to the database name we want to connect to.*
8. *Determine user and password refer Username and Password that used to connect to the SQL Server.*

End

2. Update database algorithm

Input: database tables with empty time, cost columns

Output: database tables with filled time, cost columns

Update process:

Begin

1. Select node name from all-nodes table (table 3.2), call this N

2. Update N table

3. While not end of all-nodes table Repeat from 1 to 2

End

3. Update node table algorithm

Input: database with empty time, cost columns

Output: updated database with filled time, cost columns

Process:

If (system time in rush hours) {

For each node table {

If (adjacency=1) {

Check street type, call this x

Get average speed from average speed table (table 3.3) where street type is x and rush hours is 1

Update time column set time=Distance/ average speed

Get Distance per liter from average speed table where street type is x and rush hours is 1

Get Price of 1 Liter from cost table (table3.4)

*Update cost column set cost= (Distance/distance per liter)*price of 1liter*

}

}

}

Else if (system time is not in rush hours){

For each table {

If (adjacency=1) {

Check street type, call this x

Get average speed from average speed table (table 3.3) where street type is x and rush hours is 0

Update time column set time=Distance/ average speed

Get Distance per liter from average speed table where street type is x and rush hours is 0

Get price of 1 liter from cost table (table 3.4)

*Update cost column set cost= (Distance/distance per liter)*price of 1liter} } }*

4. Find best rental car office algorithm

Input: Criteria

Source

Output: the best rental car office name

The required time or cost

Process:

Get symbol from Places table where place name equal to source, call this D

If (criteria is Time){

For each symbol in offices table {

Set symbol equal to S

If (criteria is Time){

Apply dijkstra from S to D, to find the least time path

Store the least time

}

Compare the results, get the least one

Show S which has the least result

}

Else If (criteria is Cost) {

For each symbol in offices table {

Set symbol equal to S

Apply dijkstra from S to D, to find the least cost path

Store the least cost

}

Compare the results, get the least one

Show S which has the least result

Get profit margin from cost table call this m

*Show required cost equal to total cost*m+ total cost*

}

5. Finding Best Path

Input: Criteria

Source

Destination

Output: best path

Total time or cost

Process:

Get symbol from Places table where place name equal to source, call this S

Get symbol from Places table where place name equal to destination, call this D

If (criteria is Time) {

Apply dijkstra from S to D, to find the least time path

Show the result

Show the least time

}

Else If (criteria is Cost){

Apply dijkstra from S to D, to find the least cost path

Get profit margin from cost table call this m

*Total cost= resulted cost from dijkstre *m+ resulted cost from dijkstre*

Show best path resulted from dijkstra

Show total cost

}

6. Update setup values

Input: new value for average speed or cost

Output: store the new value in the database

Process:

If (the new value related to speed){

Go to average table

Update the specified value

}

Else if (the new value related to cost){

Go to cost table

Update the specified value

}