

RP
00112
c.1

MOBIMEDIC

An Application for Mobile Emergency Services Routing

by

Hassan Abou-Jerji

B.S., Computer Science, American University of Beirut (AUB), 2003

Project submitted in partial fulfillment of the requirements for the Degree of Master of
Science in Computer Science

Department of Computer Science and Mathematics

LEBANESE AMERICAN UNIVERSITY

2009

162922



LEBANESE AMERICAN UNIVERSITY

School of Arts and Sciences - Beirut Campus

Project Approval Form

Student Name: Hassan Abou-Jerji

I.D. #: 200301638

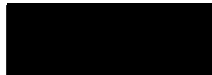
Project Title : MobiMedic: An Application for Mobile Emergency Service
Routing

Program : M.S. in Computer Science

Division/Dept : Computer Science and Mathematics

School : Arts and Sciences - Beirut

Approved by:



Project Advisor: Ramzi A. Haraty



Member : Abdul Nasser Kassar

Date

May 15, 2009

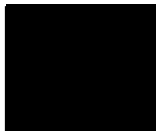
Plagiarism Policy Compliance Statement

I certify that I have read and understood LAU's Plagiarism Policy. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.

This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Hassan Abou-Jerji

Signature:



Date: May 15, 2009

I grant to the LEBANESE AMERICAN UNIVERSITY the right to use this work, irrespective of any copyright, for the University's own purpose without cost to the University or its students and employees. I further agree that the University may reproduce and provide single copies of the work to the public for the cost of reproduction.

To My Dear Parents ... God Bless You

Acknowledgments

I would like to thank my advisor Dr. Ramzi Haraty for his continuous guidance throughout my University days and for his commitment in supporting me with constructive and positive ideas. Also, I would like to thank Dr. Abdul Nasser Kassar for being a member of my project committee.

I would like to express my sincere gratitude to the Lebanese American University whose financial support during my graduate studies made it all possible.

To my Parents and wonderful Sisters I say: Thank you, for your continuous support and encouragement throughout my career and always. A special one goes to my adorable nieces Lara and Reine; you have always put a smile on my face, everyday. I would not have done this without you all.

Moreover, I would like to thank my close friends who worked with me on the shortest distance project during our undergraduate days: Talal Malas, Diana Tabet and Walid El-Horr. And finally, I would like to thank some really special friends for their expert feedback, some for their support and others for both: Tanya for your energy and professional terms, Dani, Heba, Heremi, Sudeep, Zeina and everyone who showed interest in this project and supported my progress up to this level.

I dedicate this project to the Lebanese Red Cross, the hard work I did while serving your noble cause made me the person I am today. Thank you for inspiring me. I will always cherish my Red Cross years.

Abstract

In the busy streets of Beirut, traffic is inevitable. Emergency services struggle through this traffic to respond to calls of various types, some being more urgent than others. Providing safety and medical response to the public in a prompt manner becomes a priority for emergency service providers. Hence, avoiding busy and lengthy roads when possible help them achieve this objective. As traffic is an unpredictable issue, avoiding unnecessary detours could be just the solution they need.

In this project, we develop MobiMedic; a tool based on the map of the city of Beirut that allows emergency dispatch personnel to provide emergency vehicles responding to a call with a shortest journey to take to get to incident site, the condition and the number of cases reported. This will allow the crew of the emergency vehicle responding to the call to be prepared for the case anticipated and get there as fast as possible avoiding lengthy routes. Moreover, our tool will allow for quality assurance audits and reviews as the calls being reported are logged as they occur.

We are hoping that the implementation of MobiMedic at this stage will open doors to enhancements. The functionalities introduced are the corner steps on the way to a state-of-art emergency response solution. We also hope that one day we will be able to make use of this application in real life to serve the Lebanese Red Cross in their noble mission.

Contents

Acknowledgments.....	vi
Abstract	vii
Chapter 1 Introduction	1
1.1 Motivation.....	1
1.2 Overview.....	2
1.3 Environment and Tools	3
Chapter 2 Literature Review	7
Chapter 3 MobiMedic Functions	10
3.1 Emergency Call Details	10
3.2 A Walkthrough on MobiMedic.....	13
Chapter 4 Implementation Details	23
Chapter 5 Conclusion.....	34
Appendix A: MobiMedic Code	36
References	68

List of Figures

Figure 1: MobiMedic Functions	13
Figure 2: Selecting the Street	14
Figure 3: MobiMedic's Response to Street Selection.....	15
Figure 4: Case Details Selection.....	16
Figure 5: Ambulance Selection	17
Figure 6: SMS Text Content.....	17
Figure 7: SMS Sent Confirmation Page.....	18
Figure 8: SMS Received	19
Figure 9: Report Generation.....	19
Figure 10: Help File	20
Figure 11: Empty Choose Street Selection.....	21
Figure 12: Empty Known Condition(s) Selection	21
Figure 13: Empty Number of Casualties Selection	22
Figure 14: Empty Ambulances to Dispatch Selection	22
Figure 15: Map Representations in a Graph.....	23
Figure 16: MFC Design and Development	30

List of Tables

Table 1: Ambulance Numbers per Area.....	11
--	----

Chapter 1 Introduction

1.1 Motivation

In the busy streets of Beirut, transportation of large vehicles can be time consuming. In times of emergency, special emergency vehicles move quickly to respond to calls regarding various types of incidents that might occur and that require quick and efficient transportation to centers where immediate health care can be administered to the incident victims. As a Lebanese Red Cross alumnus, the importance of this issue was obvious and I felt the need to contribute to solve this problem.

Every minute from which the ambulance leaves its station, bound to the accident site, then to the hospital is critical in saving lives. During certain rush hours of the day, it is impossible for ambulances to reach the site in time to administer first aid. Moreover, lengthy roads might increase the risk of delaying the prompt arrival of patients to hospitals. Therefore, an understanding of the locations of ambulance centers and hospitals, in addition to road information is the key to successfully finding the quickest path to the patient; thus, saving more lives. Moreover, feedback to the ambulance is helpful for efficient preparation and response from their side.

In this project, we develop "MobiMedic"; an application that calculates a shortest path route for an emergency vehicle to use as soon as it dispatches. Moreover, MobiMedic will give the first aid paramedics the chance of an earlier preparation for the case ahead of them in terms of crew management and equipment, as they will receive along with the shortest route, the possible condition(s) happening, described in common ER medical terminology and the number of case(s) expected. All this information will be received through a mobile device that can be carried around with ambulances as they dispatch for an ambulance callout, receiving the shortest path to the accident site from their dispatch location and the shortest path to the nearest hospital from the accident site.

MobiMedic, as we hope, will be a very promising tool that has the sky as its limits in terms of functionality. In the remaining parts of this chapter, we give a brief overview of the various attempts we had to build MobiMedic in various ways and discuss the environment used to build it. In chapter 2, we provide a literature review about previous work related to emergency vehicle dispatch and some common routing problems. We discuss the functionalities of our tool in and show a walkthrough in chapter 3 and explain the implementation details in chapter 4. Finally, we highlight the possibilities of enhancements and conclude in chapter 5. With a little funding, we believe that this project can be setting up the ground to the latest techniques in emergency service dispatch systems.

1.2 Overview

Our first approach on this project was an attempt to build a mobile application; one that is loaded on mobile devices to ease the routing. However, this mechanism, in addition to its complexities and the many difficulties we faced while attempting to build it, will make MobiMedic lose many of its functionalities. For example, the ability to get notified about the case the ambulance is heading to pick up, or the condition(s) to be expected at the target site. This made us lose the idea of a mobile application, and focus on an application that ‘serves’ mobile devices rather than run on one; hence, MobiMedic’s current implementation.

We tried a second approach while attempting to build MobiMedic by using Global Positioning Systems (GPS) and the development of MobiMedic as a Geographical Information System (GIS) Solution; GPS is a navigation technology that communicates with a set of satellites to acquire a certain position on earth. The GIS solution uses the GPS technology to analyze and display geographical information [1, 2]. That said, GIS solutions can be developed and customized according to your needs. GIS solutions are commonly available from specialized companies such as ESRI and ThinkGeo, the latter offering GIS components for .NET developers as well. We tried to approach the implementation of MobiMedic this way by building a specialized map containing hospitals, ambulance centers and roads. In GIS development, we can build what is called a ‘layer’ [3]; a layer is a set of objects

that need to be represented on the map. Multiple layers including a layer for hospitals, medical centers and many more objects will be linked to a map of all streets in Beirut.

This information is usually procured from professional companies who register the longitude and latitude positions of every street, hospital and ambulance center that needs to be mapped. Unfortunately, this proved to be, in addition to losing the functionalities that MobiMedic's current functionalities offer, a very expensive option.

This led us to the conclusion that we needed to keep the development simple and controllable, and as a starting point to our work, we referred back to the basic data we collected from a previous project we had worked on as a prototype for an application that offers shortest route calculation in addition to total distances and we had always thought that enhancing this application would be our future step, well, we did it. This enhancement presented in this project is a step closer to achieving modern real time routing for ambulance services. In the next section, we will discuss the background with regards to the environment and tools used to implement MobiMedic.

1.3 Environment and Tools

In this section, we briefly introduce the environment from which MobiMedic was inspired and around which MobiMedic was built. It introduces vehicle routing, mobile environment and the development tools used to build MobiMedic.

1.3.1 Emergency Vehicle Routing

Ambulances and emergency service vehicle drivers suffer a lot in traffic and unnoticed detours specially with a life hanging in the back of the ambulance, the skill and mental strength needed to do the job are not easy, not to forget the unexpected conditions along the way. As a result, a guiding hand can always ease their job and life. Currently, most ambulance drivers use their intuition and personal experience in getting from ambulance centers to destinations, based on just the end destination given by the call center.

State of the art ambulances sometimes are equipped with GPS devices to assist in getting to destinations, especially in large cities. In chapter 2, we discuss some work done on efficient vehicle routing techniques in various domains, and discuss how this can contribute but not entirely solve the problem for emergency medical vehicle routing.

1.3.2 Mobile Environments

Mobile devices are used in our everyday life in various ways; mobile phones specifically are used for phone to phone communication or for short message service (SMS) texting. More advanced mobile phones gives you the option of setting reminders and calendar entries, sending and receiving emails and web browse, even integrating with other devices like PDAs. What is needed to offer mobile phone services are Radio Frequency (RF) fields. Mobile phones communicate in a network, using base stations to receive and transmit calls; the mobile phone connects and transmits to its nearest base station and vice versa [4]. As the mobile receives the signal from the closest base station, each base station will have a coverage area that is call the "cell". The cell defines the maximum range to which that base station's signal can reach. Cells overlap to provide coverage area, hence; when people travel with their mobile device, the mobile moves from one base station's cell to another in a very transparent manner to the user.

The first base station is serving the mobile device, as the mobile device steps out of the base station's coverage area (cell), the next base station picks up and starts transmitting and receiving. In a typical city, and due to the various buildings around base stations, the cell of each base station can vary in coverage area. Modern cities provide base stations in a way to provide service coverage in a continuous manner and without interruptions of signal.

The intermediate party between the sender and receiver, who is situated between the base stations, is best known as 'service provider'. The service provider supports the coverage area by providing base stations. The service provider is responsible for diverting all calls and messages. The service provider can sometimes

provide the service using an existing infrastructure of the telecom provider based on mutual agreements.

1.3.3 Development Environment

1.3.3.1 Microsoft Visual Studio 2005

Microsoft Visual Studio is a programming platform developed by Microsoft Corporation [5]. This enables developers to create, design and build numerous types of applications such as web applications, mobile applications and .NET applications. Visual studio offers a code editor in addition to many built-in tools that allows the design of Graphical User Interface (GUI) and web design.

Visual Studio also supports many built-in programming languages such as C and C++ which is supported via the Visual C++ IDE (Integrated Development Environment) and C# via Visual C#. It also offers support to other languages such as Python via a language service that needs to be installed separately. The graphical user interface and the look-and-feel development for this project were done using MFC (Microsoft Foundation Class).

1.3.3.2 MFC: Microsoft Foundation Class

The Microsoft Foundation Class library is a collection of classes built in C++ that is used in building applications [6]. Used under visual studio, MFC provides a framework with skeleton code that we can fill to build the program. The “main” function will be substituted within the Graphical-User Interface. Moreover, MFC library provide classes to give the GUI elements such as frames, scroll menus, text boxes and hence provide you with the full shell to develop your application and provide GUI at the same time [7, 8]. The details of the MFC implementation will be presented in chapter 4.

1.3.3.3 SMS Broadcasting

An integral part of our application is the SMS sending option. For that, a lot of research was done to investigate how we can utilize this option in our system [9, 10]. As a result, we came upon various 'service providers' who can offer SMS sending; 'Activexperts' is a toolkit that provides SMS and MMS messaging services for software developers via GSM modems. Moreover, another service provider is SMSLibX which is an SMS ActiveX component DLL used for SMS sending (and receiving) from a PC, by using a GSM modem/phone.

The most efficient option which allows for maximum adaptability for our purpose was using a service provider who loads your text message via a URL which triggers the mobile sending service toolkit on the service provider's server. This proved the best option as it gives us the flexibility to manipulate the text we want to send, hence, adding a necessary dynamic feature to the application. Our service provider is Actel; one of the leading providers of telephony services in the Middle East [11]. Details on the SMS sending code used will be explained in the functionality section of chapter 4.

Chapter 2 Literature Review

In this chapter we discuss previous work done in the field of routing optimization for vehicles and other tools available from various vendors for emergency vehicle dispatch, computer aided dispatch and routing, the former being a theoretical approach while the latter is more practical and has been implemented. It is important to introduce these topics being highly related to the ideas and techniques that MobiMedic used. They also serve as study material to further enhance MobiMedic.

In [12] Lee, Epelman, White and Bozer discussed a shortest path approach to the Multiple-Vehicle Routing Problem with Split Pick-Ups. In their paper, the problem of multiple-vehicle routing with split picks was discussed, the problem involves a fleet of trucks with the same capacity but having different supply pick up points and a single assembly plant. The trucks having to move supplies and parts from the supplier to the assembly point, and each supplier may have more than one truck picking up the supplies from there. This approach is an extension to the classical Vehicle Routing Problem, which was defined in [13] as a problem where “a fleet of vehicles with uniform capacity, a common depot, and several consumer demands, find the set of routes with overall minimum route cost which service all the demands”. Another classical problem that addresses routing is the Travelling Salesman Problem [14], where the objective is to find the shortest tour through a set of cities, visiting each city once and finally returning to the starting city. This is another example of trying to find shortest path between set locations from a starting point.

In a more practical idea that was presented in [15], the author introduces Mobile Web Services in emergency systems in the health sector through proposing an application that is based on the usage of a mobile system, based on cellular phones in ambulances. In addition, the doctors will be equipped with mobile devices that are connected through high bandwidth Internet to facilitate an access to patient records from a hospital database at high speeds. In the remaining part of this chapter, we shift to present practical applications and tools developed and implemented from various vendors to serve the important task of emergency vehicle routing and dispatch.

The city of Victoria, Australia and specifically the Metropolitan Ambulance Service [16] were pioneers in using the computer aided dispatch for emergency services. This system was implemented in 1995, further enhanced with Advanced Medical Priority Dispatch and introduced as a computerized version in 1998. This system provides clinical information about the patient being transported to the hospital while the ambulance gets some care information from the hospital to the ambulance. The system is backed by a Motorola data terminal and has an automatic vehicle locator. This system helps the dispatcher in locating the closest ambulance to the incident scene; and hence, managing the resources efficiently.

Another tool that offers other functionalities is the Priority Solutions triage software (PSIAM) which is owned and distributed by Priority Solutions [17]. PSIAM sits on top of an automated clinical content product that is based on integration between Priority Dispatch's automated Medical Priority Dispatch Systems (MPDS) protocols and Clinical Solutions' [24] suite of content products. In this application, callers to dispatch centers using the tool will be directed to trained medical professionals to collect information and use this information to determine the appropriate health services that are required. The tool provides a series of questions to gather initial information from the caller about the incident and as a result, the medical professional receiving the call will be able to direct the call to targeted care providers and emergency dispatchers or provide the necessary first-aid directions to avoid a trip to the emergency room. The information gathered uses a nurse triage protocol; a process of patient prioritization based on the condition and its severity, some nurse triage protocols are used to analyze patient symptoms over phone calls. Callers who provide preliminary information that leads the tool to realize that the caller has a medical emergency, then the MPDS protocols will be triggered to enable emergency dispatch personnel to provide pre-arrival instructions to the caller and initiate the life support protocols needed. For callers whose cases are deemed not urgent, the tool activates Clinical Solutions' suite for the user. The advantages offered by this solution can be summarized by helping eliminate emergency room waiting time or unnecessary visits to emergency rooms.

Emergency response and dispatch tools has been catered to by multiple vendors, the functionalities of each one of them vary and it is only the consumer who can decide on the true worth of the tool depending on the community's needs and their acceptance. Other vendors who provide computer-aided dispatch solutions include Intergraph [18] and Tiburon, Inc. [19].

Chapter 3 MobiMedic Functions

In this chapter, we introduce the functionalities that MobiMedic offer. We depict screen shots from the application and show how the functions are triggered. We also present the practical usage of those functions in a full cycle.

3.1 Emergency Call Details

Function: Select Street

In this function, a list of all streets in Beirut is displayed in the scroll down menu labeled “Choose Street”. The operator selects the street from the scroll menu, and presses the “Select Street” button to enter the street name into the system. This function initiates the search for the area to which the street belongs to. Given the map of Beirut, we divided Beirut into four main zones or areas. Accordingly, the system starts the search for the closest ambulance center and the closest hospital based on the street location within the area. This information is displayed in the Display Box on the main screen, providing the following information to display:

- The area to which the street belongs to (Ras Beirut, Mar Elias, Acharfieh and Riverside),
- The ambulance center from which the ambulance needs to be dispatched from,
- The closest hospital where the ambulance should head to,
- The Route Chosen, and
- The approximated total distance to be travelled by the ambulance.

Function: Known Condition(s)

This function offers a comprehensive list of medical emergency conditions that are usually communicated to emergency medical personnel. This is based on common terms used within paramedic services. The list also offers the option of selecting UNKNOWN when the condition is difficult to explain or cannot be reported.

Function: Number of Casualties

This function allows the operator to select the number of casualties reported or witnessed at the scene. This might be vague in some conditions and as a result, the system will offer the selection of UNKNOWN when the number is not clear. The numbers offered for selection are 1-4 (inclusively), the reasoning of this is that a small number of casualties indicate the need for a limited number of ambulances to be dispatched. Larger scale incidents will be indicated with more than 5 casualties and this usually will require a major ambulance dispatch operation.

Function: Ambulance to Dispatch

This function gives the operator the option to choose which ambulance to dispatch. For the purpose of this project, we assumed a total number of 4 ambulances; 1 ambulance per zone and it is distributed as in the following Table 1:

Table 1: Ambulance Numbers per Area

Area/Zone	Ambulance Callout Number
Ras Beirut	101A
Mar Elias	102B
Achrafieh	103C
Riverside	104D

As a result of this distribution, once the system returns the ambulance center to be contacted as per the area, the operator selects the ambulance number from the list and proceeds to the next function. Once the ambulance is selected, the system refers to the contact number of the ambulance and selects that for processing.

Function: Enter

This function takes the condition(s) specified, the number of casualties and the ambulance number to which the message will be sent for dispatch and prepares the SMS message that will be sent.

Function: Send SMS

This function performs the last step of the dispatch process, which is sending the SMS text to the mobile device on the selected ambulance. The following information will be sent to the selected ambulance:

- The target street,
- The target hospital,
- The route to be taken starting at the ambulance center, passing through the incident scene then finally to the hospital,
- The condition(s) expected, and
- The number of reported cases.

Function: Generate Report

This function allows MobiMedic to log and report in a .txt file, all the cases registered in that working session. The logging happens as soon as the call is dispatched and as a result, at any point during the day, a report can be generated showing all the cases and their details. There are many advantages in those reports as they can help in reviewing call details, quality of service and for collecting statistics.

Other Functions:

A number of functions have been added to MobiMedic that do not serve the main function of MobiMedic, but are necessary in any application:

- Clear Output: removes all data from the display box in the main page of the application.

- Help: Launches a Help document for the end user to refer to.
- Exit: Quits the application.

Figure 1 below is an extract from MobiMedic highlighting the functions explained above.

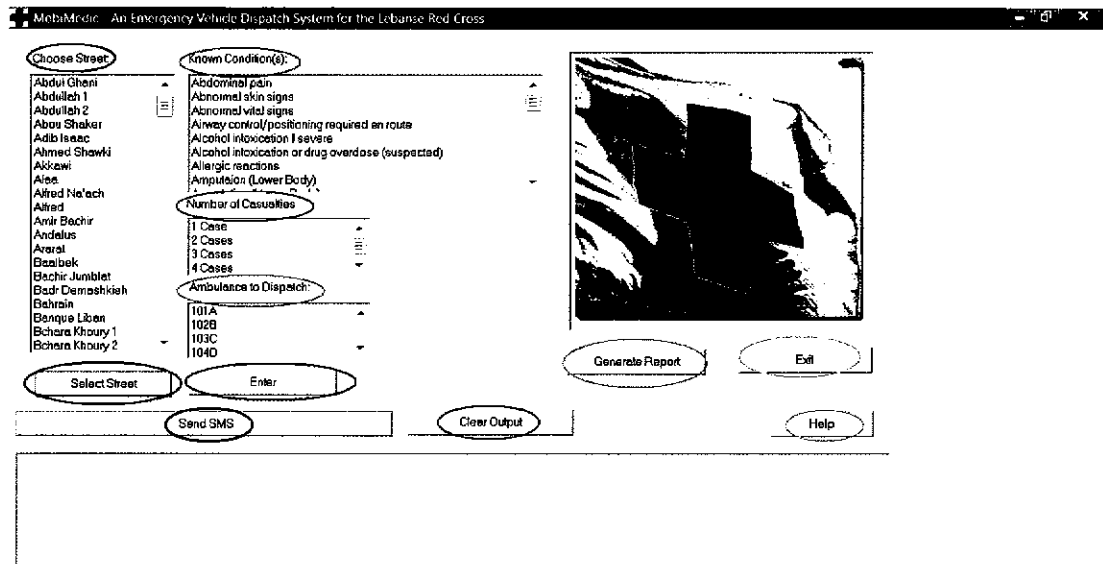


Figure 1: MobiMedic Functions

3.2 A Walkthrough on MobiMedic

This section offers a walkthrough of MobiMedic showing a practical example of usage and its functions' output. All the assumptions, will be stated within each step of the walkthrough. In this example, we will assume that a fainting incident was reported from Hamra Street in Ras Beirut, as a result, the caller reports the case and identifies it as fainting and reports one case being involved.

Street Selection

MobiMedic - An Emergency Vehicle Dispatch System for the Lebanese Red Cross

Choose Street

- Government Palace
- Graham
- Habib Bache 1
- Habib Bache 2
- Habib Shaha 1
- Habib Shaha 2
- Habib Shaha 3
- Habib Shaha 4
- Hamra1**
- Hamra 2
- Henry Donald
- Hsein 1
- Hsein 2
- Ibn Rushd 1
- Ibn Rushd 2
- Ibn Rushd 3
- Ibn Sine
- Ibrahim River
- JC Street
- Industry

Known Condition(s):

- Abdominal pain
- Abnormal skin signs
- Abnormal vital signs
- Airway control/positioning required on route
- Alcohol intoxication I severe
- Alcohol intoxication or drug overdose (suspected)
- Allergic reactions
- Amputation (Lower Body)

Number of Casualties

- 1 Case
- 2 Cases
- 3 Cases
- 4 Cases

Ambulance to Dispatch:

- 101A
- 102B
- 103C
- 104D

Select Street

Enter

Send SMS

Clear Output

Generate Report

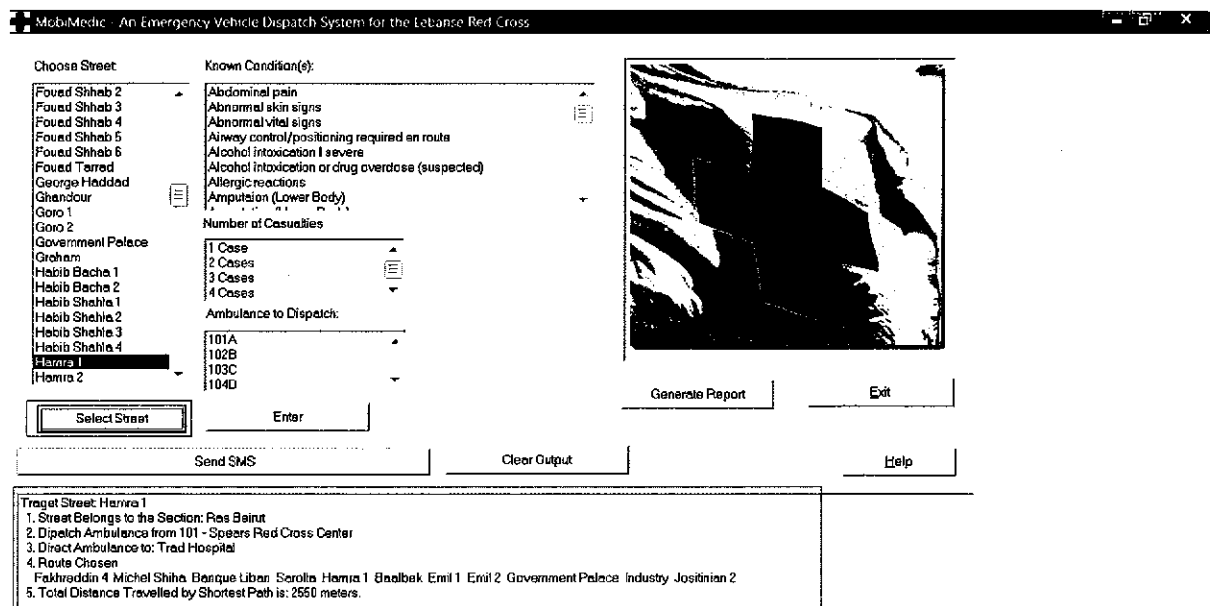
Exit

Help

Figure 2: Selecting the Street

Figure 2 above is a screen shot that shows that the user picked 'Hamra1' in the Choose Street scroll list. Meanwhile, other information can be gathered from the caller. The next step that MobiMedic will do is identify the area where the street belongs to, identify the Red Cross center and the closest hospital from the case scene; and hence, provide the user with a shortest route from the ambulance center in that area to the accident scene and from the accident scene to the nearest hospital. The next screen shot shown in Figure 3, will show the result of this calculation.

Area Identification and Shortest Path Selection



MobiMedic - An Emergency Vehicle Dispatch System for the Lebanese Red Cross

Choose Street

- Foued Shihab 2
- Foued Shihab 3
- Foued Shihab 4
- Foued Shihab 5
- Foued Shihab 6
- Foued Tarrad
- George Haddad
- Ghandour
- Goro 1
- Goro 2
- Government Palace
- Graham
- Habib Bacha 1
- Habib Bacha 2
- Habib Shehla 1
- Habib Shehla 2
- Habib Shehla 3
- Habib Shehla 4
- Hamra 1**
- Hamra 2

Known Condition(s):

- Abdominal pain
- Abnormal skin signs
- Abnormal vital signs
- Airway control/positioning required en route
- Alcohol intoxication / severe
- Alcohol intoxication or drug overdose (suspected)
- Allergic reactions
- Amputation (Lower Body)

Number of Casualties

- 1 Case
- 2 Cases
- 3 Cases
- 4 Cases

Ambulance to Dispatch:

- 101A
- 102B
- 103C
- 104D

Select Street Enter

Generate Report Exit

Send SMS Clear Output Help

Target Street: Hamra 1

1. Street Belongs to the Section: Ras Beirut
2. Dispatch Ambulance from 101 - Spears Red Cross Center
3. Direct Ambulance to: Trad Hospital
4. Route Chosen
Fahkreddine 4 → Micheal Sheha → Banque Liban → Sarolla → *Incident Scene – Hamra 1* → Baalbak → Emil 1 → Emil 2 → Government Palace → Industry → Jositinian 2 → *Destination Hospital – Trad*
5. Total Distance Travelled by Shortest Path is: 2660 meters.

Figure 3: MobiMedic's Response to Street Selection

As soon as the user presses 'Select Street', MobiMedic takes as input 'Hamra1' and identifies the 'Hamra1' street to be in Ras Beirut area. As a result, the closest Red Cross Ambulance Center would be 'Spears' and the closest hospital to be reached would be Trad Hospital. The shortest route to be taken from the ambulance center to the destination is as dictated in the output box labeled in Red in Figure 3: Fahkreddine 4 → Micheal Sheha → Banque Liban → Sarolla → *Incident Scene – Hamra 1* → Baalbak → Emil 1 → Emil 2 → Government Palace → Industry → Jositinian 2 → *Destination Hospital – Trad*.

Case and Condition Reporting

The screenshot displays the 'MobMedic' software interface, titled 'An Emergency Vehicle Dispatch System for the Lebanese Red Cross'. The interface is divided into several sections:

- Choose Street:** A list of streets including Fouad Shihab 2, Fouad Shihab 3, Fouad Shihab 4, Fouad Shihab 5, Fouad Shihab 6, George Tarrad, George Haddad, Chandour, Gora 1, Gora 2, Government Palace, Grahm, Habib Bacha 1, Habib Bacha 2, Habib Shehla 1, Habib Shehla 2, Habib Shehla 3, Habib Shehla 4, Hamra 1, and Hamra 2. 'Hamra 1' is selected.
- Known Condition(s):** A list of conditions including Envenomation, Eye injuries, Fainting, Fall, Hazmat Exposure, Heat exposure, Hemorrhage, Hospital to hospital transport, and others. 'Envenomation' is selected.
- Number of Casualties:** A dropdown menu with options 1 Case, 2 Cases, 3 Cases, and 4 Cases. '1 Case' is selected.
- Ambulance to Dispatch:** A dropdown menu with options 101A, 102R, 103C, and 104D. '101A' is selected.
- Buttons:** 'Select Sheet', 'Enter', 'Generate Report', 'Exit', 'Send SMS', 'Clear Output', and 'Help'.
- Output Area:** A text box at the bottom left showing the following information:
 - Target Street: Hamra 1
 - 1. Street Belongs to the Section: Ras Beirut
 - 2. Dispatch Ambulance from 101 - Spears Red Cross Center
 - 3. Direct Ambulance to: Tred Hospital
 - 4. Route Chosen
 - 5. Total Distance Travelled by Shortest Path is: 2550 meters.

Figure 4: Case Details Selection

Figure 4 above shows the tool at a point where the user has selected the entries required based on the caller's description of the case. The user will manually select ambulance number "101A" for dispatch given that it is known by now, the area is Ras Beirut, hence we are dispatching from Spears ambulance '101A'. The tool is now ready to enter the case details.

Data Entered to the Tool and the Ambulance Number Selection Takes Place

As soon as the user presses the Enter button, the tool takes in the condition from the condition list, the number of cases selected and the ambulance number selected by the user. The system fetches the mobile device number for that ambulance and displays this information as shown in Figure 5. Later, the tool prepares the SMS to be sent and logs the call in the log file.

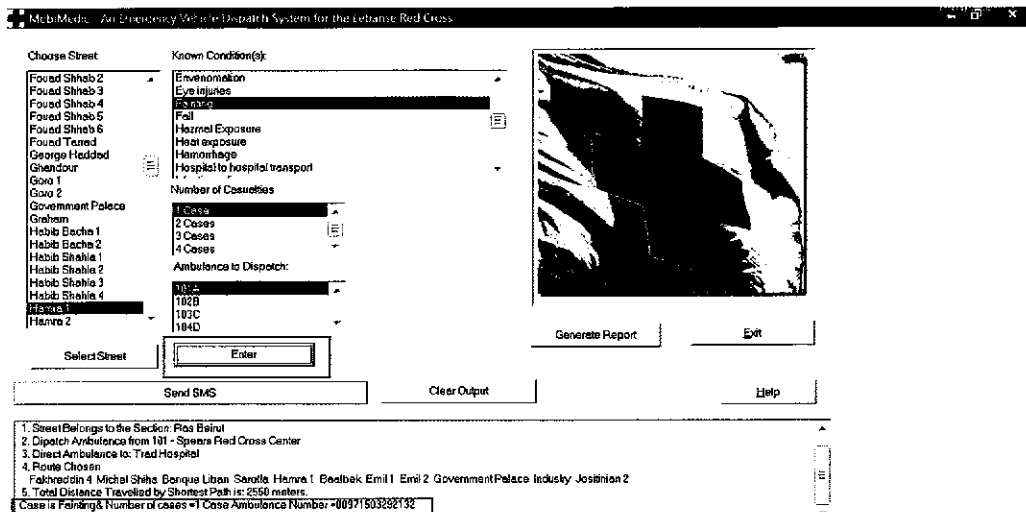


Figure 5: Ambulance Selection

Prepare the SMS for Dispatch:

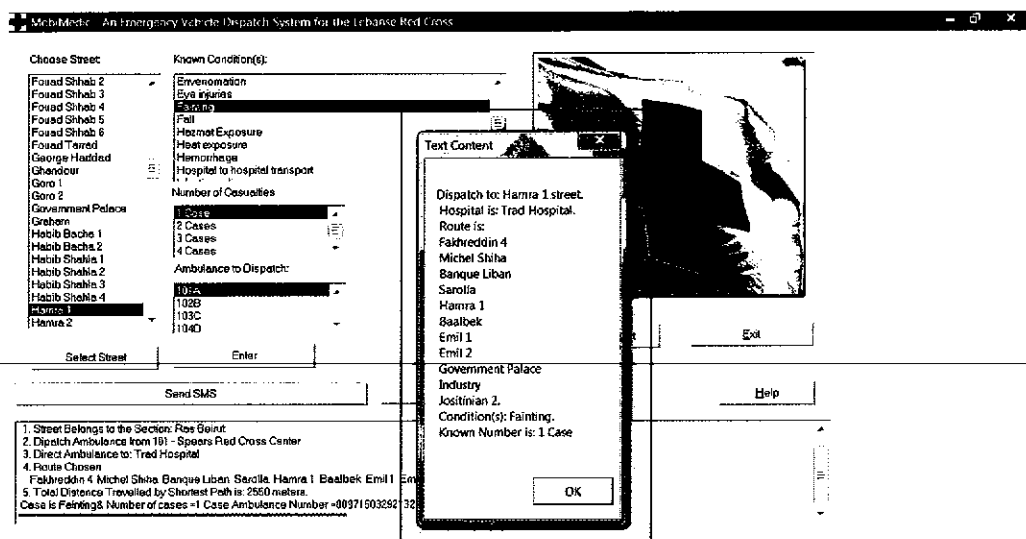


Figure 6: SMS Text Content

The system will show the user in a message box the SMS message content that will be sent as soon as the user presses the Send SMS button. This is shown in Figure 6.

SMS Sent Successfully

The service provider sends a confirmation message to the browser to confirm that the text SMS was sent correctly. Figure 7 shows how this is displayed via an OK message on the browser, followed by the communication service provider name.

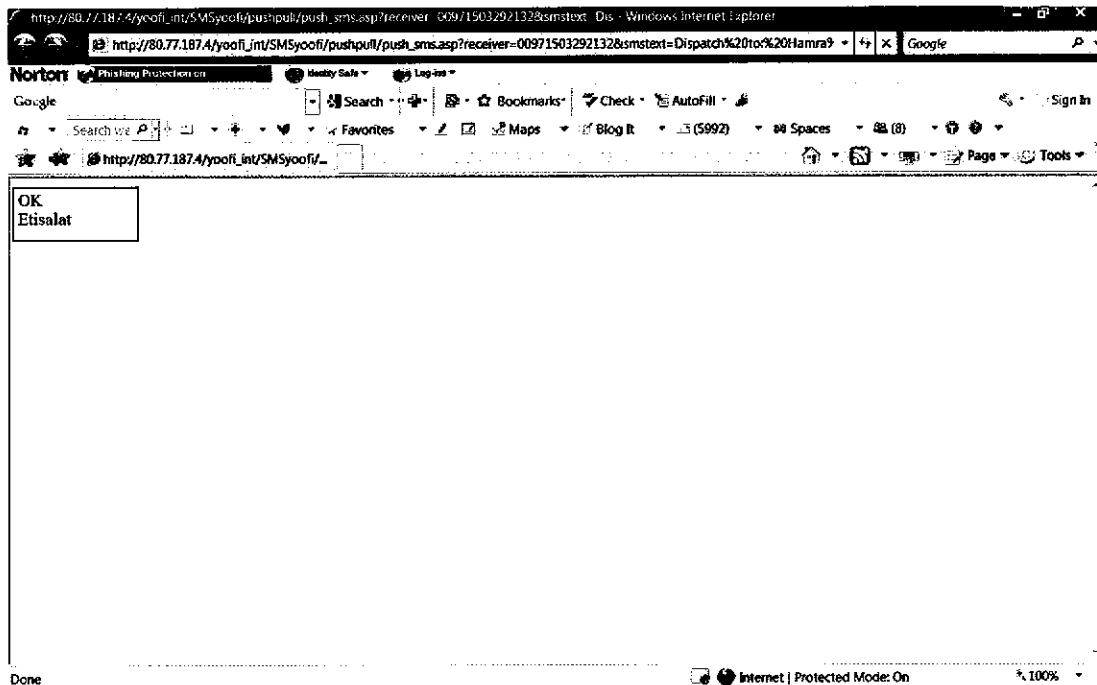


Figure 7: SMS Sent Confirmation Page

SMS Received on the Mobile Device

The SMS reaches the designated ambulance with the needed information to dispatch to the incident scene. For the purpose of demonstration, we used our personal phone as the target phone. This is depicted in Figure 8.

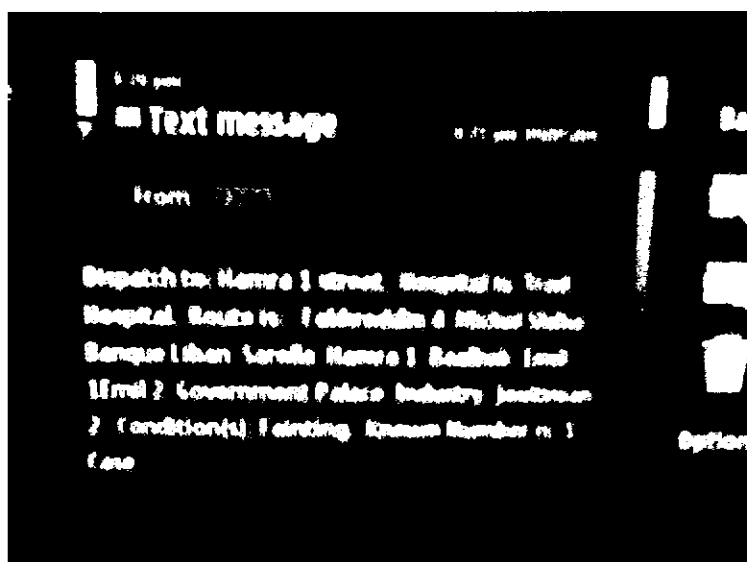


Figure 8: SMS Received

Report Produced Logging the Case:

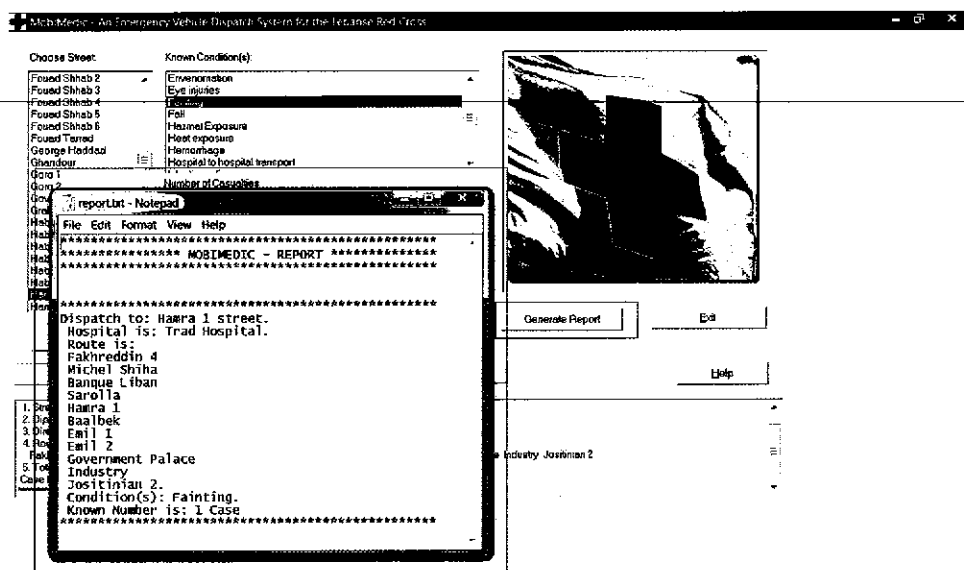


Figure 9: Report Generation

Once a report is needed, the Generate Report button will launch the report.txt showing a log of all incidents that took place and were responded to via MobiMedic, see Figure 9.

Do You Need Help using MobiMedic?

MobiMedic provides a Help button if you need assistance in using the tool. Figure 10 below shows the Help file that is launched when the Help button is pressed.

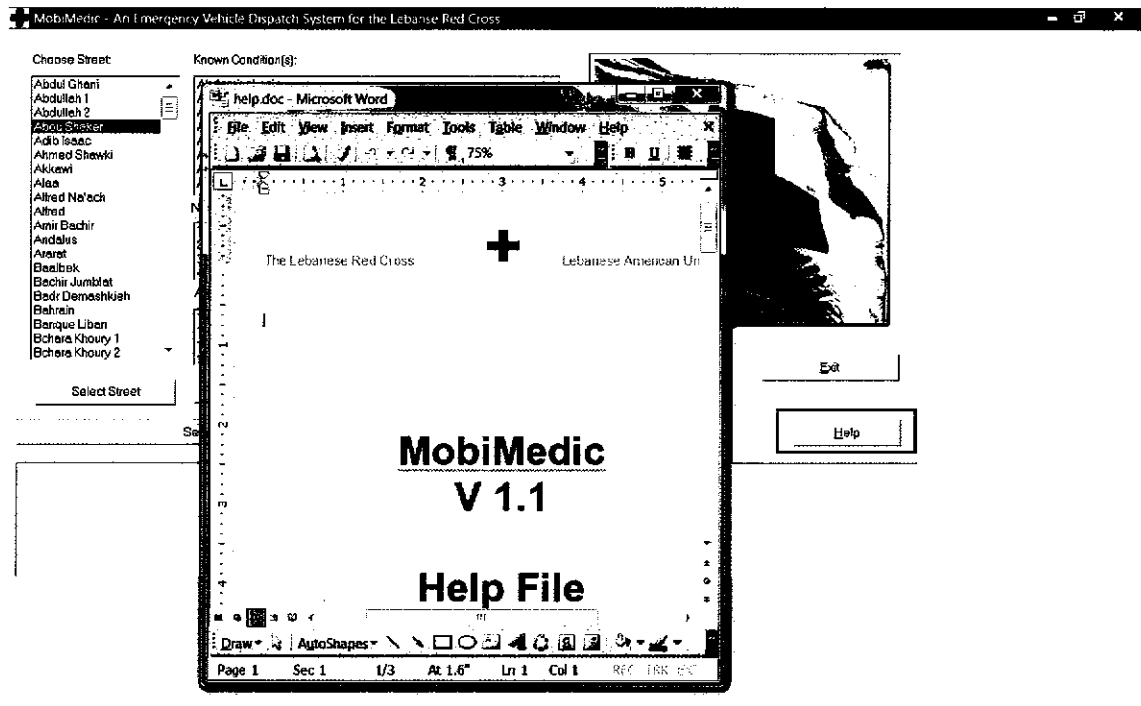


Figure 10: Help File

Street or Case details not entered

Missing details in MobiMedic reduces its efficiency. As a result, a check has been implemented in MobiMedic where an entry must be made in the case details section. If one of the details is unknown, then the user must select the UNKNOWN selection available. If a user tries to input empty details to MobiMedic, the tool will ask the user for an entry as shown in the figures below.

In Figure 11, the user tried to select an empty street. As a result, the tool listeners detected that and issued a prompt message for the user to specifically select a street.

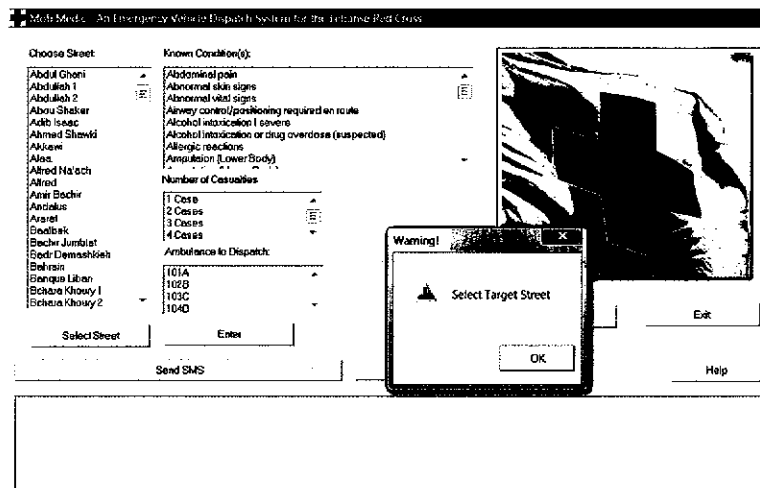


Figure 11: Empty Choose Street Selection

The same case applies in Figure 12 as the user tried to enter the case details without selecting a condition. The prompt message will inform the user to enter a condition.

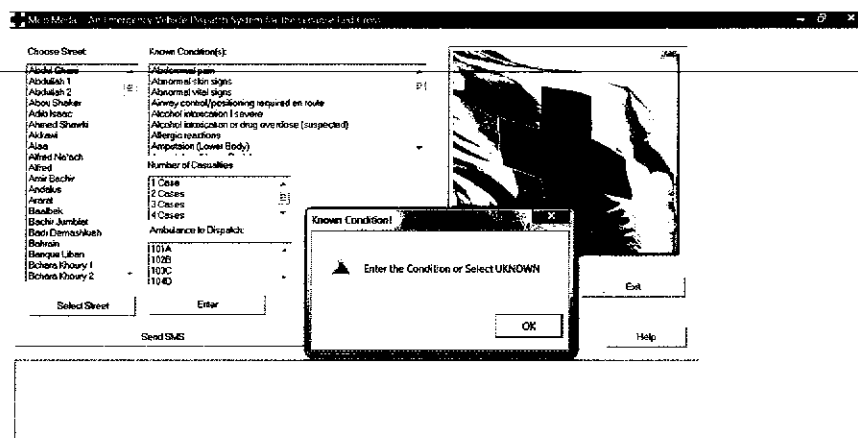


Figure 12: Empty Known Condition(s) Selection

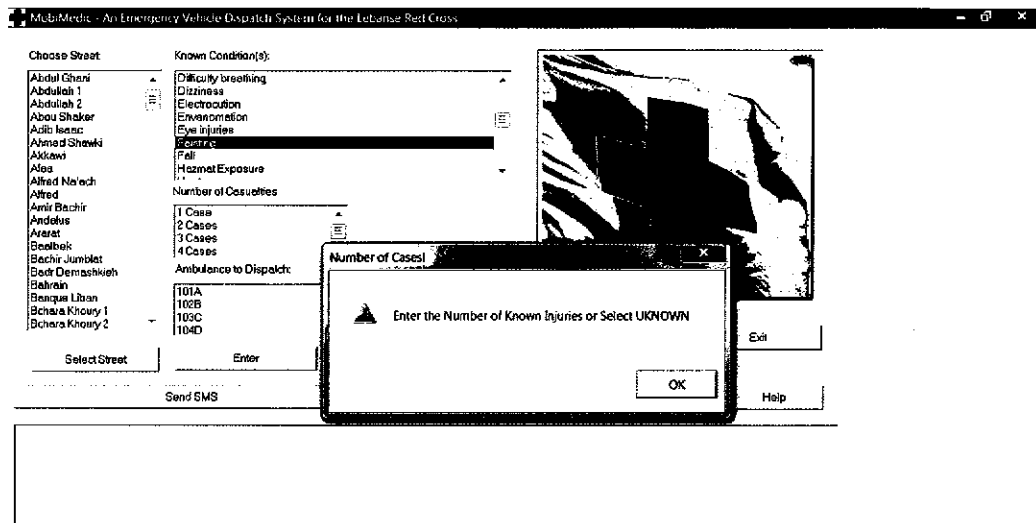


Figure 13: Empty Number of Casualties Selection

In Figure 13, the user tried to enter an empty entry for the number of injured cases, the return prompt message asks the user to enter UNKNOWN if the number is not clear. The same prompt applies if the dispatch user does not select an ambulance number. This is shown in Figure 14.

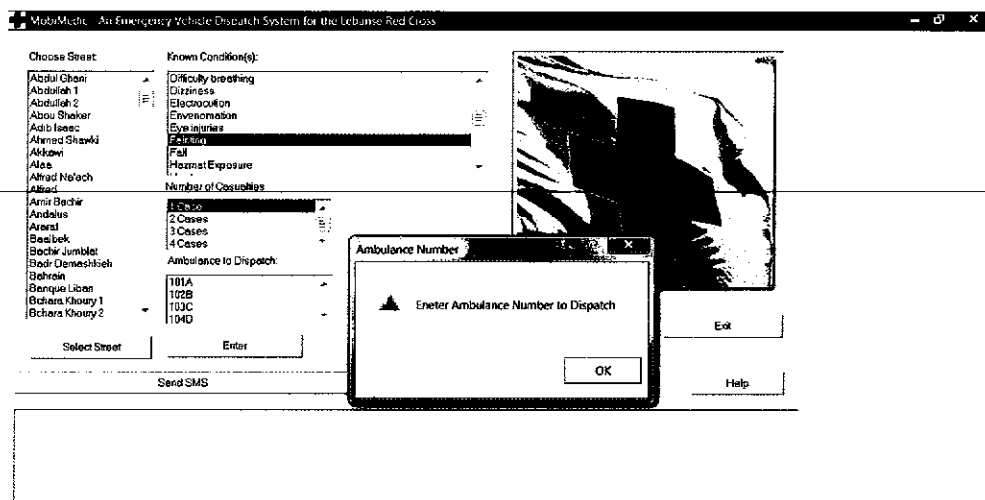


Figure 14: Empty Ambulances to Dispatch Selection

Chapter 4 Implementation Details

In this chapter, we go through the code that lies beneath MobiMedic and clarify how this code was built and the structure used.

Map Representation: Class Graph

The map of Beirut was implemented as a weighted directed graph. This facilitated the proposed division of the map into zones or areas as there will be four zones and hence four graphs. Our implementation started by defining class Graph. In a typical graph implementation, there are two main building blocks to form a graph: Nodes and Edges. As a result, the map of Beirut was studied thoroughly, and the roads of Beirut were represented as edges, while the hospitals, ambulance centers and intersections were represented as nodes. Edges connect nodes and as a result, our streets “edges” connect intersections, hospitals and ambulance centers “nodes”. To ease this implementation, we mapped a real life map [21] with references from Google maps [24] into an abstract image to ease up the graph “image” we had in mind. Figure 15 explains the mapping.

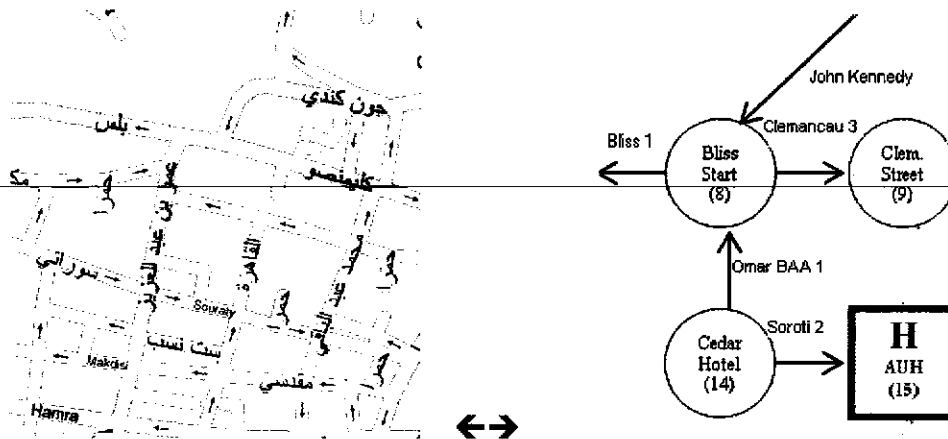


Figure 15: Map Representations in a Graph

In the map above, the intersections represented as “Bliss Start (8)” is the intersection between Bliss Street, Omar BAA (Omar Bin AbdulAziz) street and Clemancau Street, the directions shown on the abstract map is a representation of street directions in real life.

Streets Creation Class: Weight

As discussed earlier, streets are represented by edges; these edges are represented by the list "roadlist[]", and are represented in the graph as edges by using the Weight class; the class weight has the following attributes:

```
class Weight
{
    public: int distance, color;
           string graph, street;
```

String graph is the name of the zone to which the street belongs and the color is the type of the node to which the street leads to - Intersection, Hospital and Ambulance Center. The distance is the approximated length of the road and the street is the street name, which already exists in the roadlist[] list.

The method addWeight was used to link the streets, with their attributes, to the intersections; hence, forming up the graph. Method addWeight below takes the following parameters:

```
void addWeight(string gr, int v1, int v2, int dist,int col, string st)
string gr = graph name (zone name)
int v1 = node number from where the street starts
int v2 = node number to where the street ends or hits another intersection.
int dist = Approximate length of the road in meters.
int col = type of the node the edge ends at; example: Hospital.
string st = street name.
```

For example, the following line of code adds the street "Soroti 2" to the graph in zone "Ras Beirut" linking an intersection to node 15 (Hospital) .

```
rasbeirut.addWeight("Ras Beirut",14,15,600,HOSPITAL,"Soroti 2");
```

When the user enters the name of the street on which the case is occurring, the following methods take the job of getting the head and tail of the edge. This is the way to connect the edge to the node. The methods `getHead()` and `getTail()` are defined as follows and they return the head or the tail respectively:

```
int getHead(string s)
{
    for(int i=0;i<numVert;i++)
        for(int j=0;j<numVert;j++)

            if((matrix[i][j].getStreet().compare(s))==0)
                return i;
}
int getTail(string s)
{
    for(int i=0;i<numVert;i++)
        for(int j=0;j<numVert;j++)
            if((matrix[i][j].getStreet().compare(s))==0)
                return j;
}
```

Given we have a connected directed graph and we have methods to retrieve the heads and tails of edges, we need to run a shortest path algorithm from the Red Cross center that we retrieved and we need to run it twice. Once from the Red Cross center to the street where the incident is reported, and again from the street to the closest hospital.

Finding the Zone

This is the first step which is needed, to find the name of the zone to which the street belongs to. This is achieved through calling the method `getGraphName()`, which searches for the selected street name in all four zones and returns the graph name:

```
string getGraphName(string s)
{
    for(int i=0;i<numVert;i++)
        for(int j=0;j<numVert;j++)
            if ( (matrix[i][j].getStreet().compare(s))==0 )
                return matrix[i][j].getGraph();
    .....
}
```

Finding the Shortest Paths: Dijkstra's Algorithm

For the use of this project, we used and implemented a shortest path discovery algorithm: Dijkstra's algorithm [23]. The reason behind using Dijkstra's algorithm is because we are using a weighted, directed graph to represent the map. As a result, Dijkstra's was the more obvious choice and due to its simplicity in searching a weighted directed graph, it suited our needs.

Dijkstra's algorithm finds the shortest paths from a single source node to all other nodes in a weighted, directed graph. It starts at a single node, checks the weight on all edges leading to all its neighbors and chooses the one with the smallest weight. Moreover, it jumps to the neighbor node and marks the starting node as visited so that it will not be checked on the next search. This builds up a shortest route from the initiating node to the final node.

In our implementation, since we have a street name with a tail and a head, with the head being the previous intersection and the tail being the next intersection, we could start a search from the Red Cross center to the head of the incident street,

and from the tail of the incident location, run another Dijkstra's search to the nearest hospital. The following is an extract of the main lines of Dijkstra's algorithm that was implemented:

```
int Dijkstra(Graph G, int start,int end)
{
    .....
    while(!G.empty(G))
    {
        u = G.returnMin(G);
        for( w=G.first_neighbor(u); w<G.vertex_num(); w=G.next(u,w) )
            Relax(G,u,w);
    }
    .....
}
```

Shortest Path Search

The first step in this process, is to retrieve the tail (the intersection where the road starts) of the road by running method `getTail(string street)`. The next step is an easier step; since we have identified the tail, we run Dijkstra's algorithm between the ambulance center in that zone where the street is located and the accident site, as explained earlier, we have located an ambulance zone in each of the four areas. The same step is applied again; however, this time running Dijkstra's from the street to all hospitals in that zone and then getting the closest one. The Ambulance center name, the destination hospital and the road list will be concatenated into a string for display purposes.

SMS Code

Now that the SMS text is ready to be sent, sending the SMS was done by utilizing the following URL:

http://IPADDRESSMASKED/yoofi_int/SMSyoofi/pushpull/push_sms.asp?receiver=receiver_number&smstext=text_to_send&operator=etisalat&country=uae&smstype=text&senderID=2206&username=username&password=password

This URL triggers the SMS server at the SMS service provider and sends the text under the variable `text_to_send` to the receiver whom we provide its number under the variable `receiver_number`, my communication service provider is Etisalat; my local provider in Abu Dhabi, United Arab Emirates.

To trigger the URL to launch from within MobiMedic, we needed to make the URL variable able to accommodate changes to the text that will be sent. At the beginning, the string was static, we were not able to change the content of the SMS to be sent according to the different cases reported, so we used C++'s power to convert our string to char using the function: `std::string.c_str()`;

```
std::string str = "some string";  
const char* psz = str.c_str();
```

Hence, we create a string of our choice and convert it into a static char string and use that as a file name to execute the URL and launch it in a browser using `ShellExecute`. The following code shows the whole operation:

```
string s="";  
std::string str =  
http://IPADDRESSMASKED/yoofi_int/SMSyoofi/pushpull/push_sms.asp?receiver=  
+""ambnumber""&smstext="+""finalSMS""&operator=etisalat&country=uae&smstype  
=text&senderID=2206&username=username&password=password";  
char* psz = strdup(str.c_str());  
file= psz;  
ShellExecute(NULL, NULL, file, NULL, NULL, NULL);
```

To confirm the SMS sending from the service provider, the web browser will return an OK message with the name of the service provider.

Launching the Help File

Launching the help file is simply executed by running a ShellExecute command on the file Help.exe once the button Help is pressed.

```
void CEmergencySystemsDlg::OnHelpButton() // Help file opened
{

ShellExecute(m_hWnd, "open", "help.doc", NULL, workingdir,
SW_SHOWNORMAL);

}
```

Graphical User Interface and MFC's Methods

Finally, we will explain what made MobiMedic have a body and an interface; we will explain MFC's methods and the Graphical User Interface developed to make MobiMedic user friendly and practical. Figure 16 shows an image of the typical MFC environment under Visual Studio. Note that use of Visual Studio gives you the beauty of designing your look and feel of the GUI and it automatically maps it to code.

We used the GUI and MFC to build up what represents the 'main' class; one that contains the initiations and declarations of variables and function definitions. All declarations of integers, strings, arrays and character arrays were defined in the declarations section. As an example, the conditions array was defined as follows:

```
//Conditions list
char *CEmergencySystemsDlg::conditions[] = {"Abdominal pain",
"Abnormal skin signs", "Abnormal vital signs", .....
```

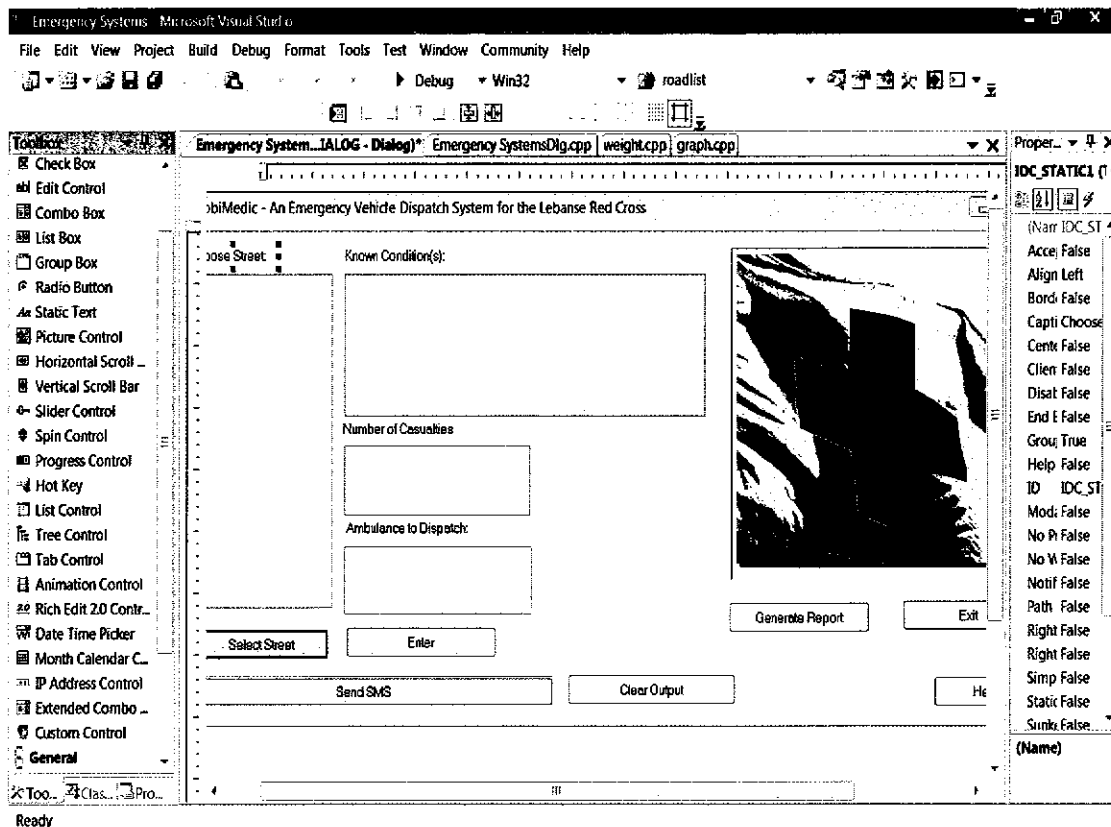


Figure 16: MFC Design and Development

The main functions used were built from class `CDialog()`: an MFC base class used to display Dialog Boxes [7], member functions from within `CDialog()` were essential to build the functions that MobiMedic provides, from ScrollList Boxes to Output Boxes and from Buttons to Images. The main dialog code was created under:

```

BOOL CEmergencySystemsDlg::OnInitDialog()
{
    CDialog::OnInitDialog()
    .....

```

This member function is called in response to `WM_INITDIALOG` message; the `WM_INITDIALOG` message is sent to the dialog box before the dialog box appears which uses this message to initialize the appearance of the dialog box [20]. This function specifies whether the application set a focus on one of the controls in the Dialog Box. If the Bool return is non-zero, Windows chooses the input focus of the first control in the dialog box.

Within that function, we defined all the list boxes that appears on the main screen and wrote the code that fills it up with the required display data. CListBox member function creates a list box that displays the conditions from the conditions array defined in the declarations section of the code. A sample of the code for the cases list box creation is shown below.

```
CListBox *Condition;  
Condition= (CListBox *) GetDlgItem( IDC_CONDITION);  
for ( int i=0; conditions[i] !=NULL; i++)  
    Condition->AddString(conditions[i]);
```

Every button we define in MFC creates an equivalent listener skeleton in the code for us to define an action to be preformed when that button is pressed. We will use the Select Road and Enter buttons as examples to cover up a bigger scope of the code. When the Select Road button is pressed, method OnSelected() will be called that takes the following actions:

The four graphs we intend to create will be established and filled up here:

```
void CEmergencySystemsDlg::OnSelected() // "Select Road" button  
{  
  
    // Create 4 graphs as independent blocks  
    Graph rasbeirut(#ofnodes);  
    Graph achrafieh(#ofnodes);  
    ...  
}
```

Moreover, adding the streets to the map to connect the nodes takes place here. As the following piece of code shows, we add the street Charles De Gaul 1 to zone Ras Beirut, connecting nodes 0 and 1. The street has approximate length of 750 m using the following line of code:

```
rasbeirut.addWeight("Ras Beirut",0,1,750,INTERSECTION,"Charles De Gaul 1");
```

As another example, the following piece of code adds Makased Hospital Street to the Mar Elias zone, connecting node 20, the Makasid Hospital, with node 19, the street has approximate length of 400 m:

```
mar_elias.addWeight("Mar Elias",19,20,400,HOSPITAL,"Makased Hospital Street");
```

On a similar note, when the Enter button is selected, the code that collects the condition, the number of cases and the desired ambulance number to where the SMS text should be sent will be executed here. First, we need to retrieve our selection, as a result, method GetCurSel() was used to choose what the user selected. We use the code for condition selection as an example:

```
Condition = ( CListBox * ) GetDlgItem( IDC_CONDITION );
// pointer (Condition) to listbox (IDC_CONDITION)
int listindex = Condition->GetCurSel();
// gets index of selected listbox item
if ( listindex == LB_ERR )
// if no item is selected --> exit function
{
    MessageBox( "Enter the Condition or Select UNKNOWN","Known
    Condition!",MB_ICONWARNING );
    return;
}
```

The above code creates a temporary integer that stores the numeric value of the selection location and if no selection is made, a prompt message will highlight this and asks for a selection. Once a selection is made, method GetText on the listindex number will be run to retrieve the textual value of the entry and the returned value will be stored in a temp character array which will be casted into a string. Note that this implementation was preferred over the textual insertion by the user to remove the possibilities of typing errors and misspells. This selection code is as follows:

```
Condition-> GetText(listindex, temp);
cond = (string) temp;.....
```

Reporting

All data entered will be streamed into a .txt file before being sent via SMS. This directly serves our reporting requirements. This is achieved using the <fstream> and <iostream> libraries that opens the file and streams text into the file. The code below explains this simple and very useful reporting function:

```
ofstream myfile; //define a file for i/o
myfile.open("report.txt");
.....
myfile<< "\n" + SMScontent + "\n";
myfile<< "*****\n";
.....
```

to the GPS which can provide a pictorial demonstration to the ambulance driver in addition to dynamically navigating through route, giving directions and calculating changes in route. Moreover, MobiMedic can include a re-routing option based on priority of calls. If an ambulance is dispatched to a call and the dispatcher gets a higher priority call, then the dispatcher should be able to re-route the ambulance to the higher priority incident location.

The ideas mentioned in this chapter serve as a starter to where we can take MobiMedics' functionalities. We believe that with a little sponsorship, effort and time, we can extend MobiMedic to become the main course for modern day emergency vehicle response and dispatch tools.

Appendix A: MobiMedic Code

EmergencySystemsDlg.cpp

```
#include "stdafx.h"
#include "Emergency Systems.h"
#include "Emergency SystemsDlg.h"
#include "Graph.cpp"
#include <iostream> //for file streaming
#include <fstream> //for file streaming
#include <sstream> //for the url variable

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();
    enum { IDD = IDD_ABOUTBOX };

protected:
    virtual void DoDataExchange(CDataExchange* pDX);

DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD) {}
void CAboutDlg::DoDataExchange(CDataExchange* pDX) {
    CDialog::DoDataExchange(pDX); }

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

CEmergencySystemsDlg::CEmergencySystemsDlg(CWnd* pParent
/*=NULL*/)
    : CDialog(CEmergencySystemsDlg::IDD, pParent)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME); }

void CEmergencySystemsDlg::DoDataExchange(CDataExchange*
pDX)
{
    CDialog::DoDataExchange(pDX); }

BEGIN_MESSAGE_MAP(CEmergencySystemsDlg, CDialog)
ON_BN_CLICKED( IDC_SELECTBUTTON, OnSelected )
ON_BN_CLICKED( IDC_IMAGEBUTTON, OnView )
```

```

ON_BN_CLICKED( IDC_ABOUTBUTTON, OnAbout )
ON_BN_CLICKED( IDC_CLEARBUTTON, OnClear )
ON_BN_CLICKED( ID_HELPBUTTON, OnHelpButton ) //Help File
ON_BN_CLICKED( IDC_REPORT, OnReportButton ) //reporting
ON_WM_SYSCOMMAND()
ON_WM_DESTROY()
ON_WM_PAINT()
ON_WM_QUERYDRAGICON()
ON_STN_CLICKED( SMALL_MAP,
&CEmergencySystemsDlg::OnStnClickedMap)
END_MESSAGE_MAP()

BOOL CEmergencySystemsDlg::OnInitDialog()
{
CDialog::OnInitDialog();
CListBox *StreetListBox;
StreetListBox = ( CListBox * ) GetDlgItem(
IDC_STREETLISTBOX );

for ( int i = 0; roadlist[ i ] != NULL; i++ )
StreetListBox->AddString( roadlist[ i ] );

//filling the conditions array
CListBox *Condition;
Condition= (CListBox *) GetDlgItem( IDC_CONDITION);
for ( int i=0; conditions[i] !=NULL; i++)
Condition->AddString(conditions[i]);

//filling the numbers array
CListBox *Numb;
Numb= (CListBox *) GetDlgItem( IDC_NUMBERS);
for ( int i=0; numbers[i] !=NULL; i++)
Numb->AddString(numbers[i]);

//Different ambulances ready to dispatch
CListBox *Ambulances;
Ambulances= (CListBox *) GetDlgItem( IDC_AMBULANCES);
for (int i=0; ambulancevalues[i] != NULL; i++)
Ambulances->AddString(ambulancevalues[i]);

ASSERT((IDM_ABOUTBOX & 0xFFFF0) == IDM_ABOUTBOX);
ASSERT(IDM_ABOUTBOX < 0xF000);

CMenu* pSysMenu = GetSystemMenu(FALSE);
if (pSysMenu != NULL)
{
CString strAboutMenu;
strAboutMenu.LoadString(IDS_ABOUTBOX);
if (!strAboutMenu.IsEmpty())
{
pSysMenu->AppendMenu(MF_SEPARATOR);

```

```

        pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
                               strAboutMenu);
    }
}

SetIcon(m_hIcon, TRUE);        // Set big icon
SetIcon(m_hIcon, FALSE);      // Set small icon

return TRUE;
}

void CEmergencySystemsDlg::OnSysCommand(UINT nID, LPARAM
lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}

void CEmergencySystemsDlg::OnDestroy()
{
    WinHelp(0L, HELP_QUIT);
    CDialog::OnDestroy();
}

void CEmergencySystemsDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM)
dc.GetSafeHdc(), 0);

        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else

```

```

        {
            CDialog::OnPaint();
        }
    }

HCURSOR CEmergencySystemsDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon; }

//*****

char *file = "";
char *workingdir = "";

//declare string targetstreet so we can send with SMS
string targetstreet="";

//declare string hospital so we can send it with SMS
string destination_hospital="";

//declare string route to take the final route with SMS
string route="";

//declare string cond so we can send it with SMS
string cond="";

//declare string num so we can send number with SMS
string num="";

//declare the ambulance number to dispatch
string ambulance="";

//declare string for reporting to help in reporting
string forreporting="";

string finalSMS="";

//actual phone number of the ambulance
string ambnumber="";

// an array of the street names
char *CEmergencySystemsDlg::roadlist[] =
{"Abdul Baki" ,"Abdul Ghani","Abdullah 1","Abdullah
2","Abou Shaker","Adib Isaac","Ahmed Shawki",
"Akkawi","Alaa","Alfred
Na'ach","Alfred","AmirBachir","Andalus","Ararat","Baalbek
","Bachir Jumblat","Badr Demashkieh","Bahrain","Banque
Liban","Bchara Khoury 1",
"Bchara Khoury 2","Bchara Khoury 3","Bchara
Khoury4","Berlin","Betty","Bheim","Bliss 1",
"Bliss 2","Brasil","Cairo","Cedars","Charles De Gaul
1","Charles De Gaul 2","Charles De Gaul 3",

```


"Charles Helou", "Charles Street", "Cheikh
 Gabi", "Clemancau", "Clemancau 1", "Clemancau 2", "Clemancau
 3", "Corniche Jamil 1", "Corniche Jamil 2", "Corniche
 Jamil3", "Dabbas", "Damascus 1", "Damascus 2", "Damascus
 3", "Damascus 4", "Elias Sarkis 1", "Elias Sarkis 2", "Emil
 1", "Emil 2", "Fadel Haraty", "Fakhreddin 1", "Fakhreddin
 2", "Fakhreddin 3", "Fakhreddin 4", "Fard Naash 1", "Farid
 Trad 2", "Fouad Shhab 1", "Fouad Shhab 2", "Fouad Shhab
 3", "Fouad Shhab 4", "Fouad Shhab 5", "Fouad Shhab 6", "Fouad
 Tarrad", "George Haddad", "Ghandour", "Goro 1", "Goro 2",
 "Government Palace", "Graham", "Habib Bacha 1", "Habib
 Bacha2", "Habib Shahla 1", "Habib Shahla 2", "Habib Shahla
 3", "Habib Shahla 4", "Hamra 1", "Hamra 2", "Henry Donald",
 "Hsein 1", "Hsein 2", "Ibn Rushd 1", "Ibn Rushd 2", "Ibn
 Rushd 3", "Ibn Sina", "Ibrahim River", "IC
 Street", "Industry", "Istiklal 1", "Istiklal 2", "Istiklal
 3", "Istiklal 4", "Itani 1", "Itani 2", "Jacob", "Jean D'arc
 1", "Jean D'arc 2", "JohnKennedy", "Jositinian1", "Jositinian
 2", "Kabiyat1", "Kabiyat2", "KhalilMoutran", "Khalil", "Kuwait
 ", "Limby", "Madame Curie 1", "Madame
 Curie2", "MadhatPasha", "Madrid", "Makased Hospital
 Street", "Mar Elias 1", "Mar Elias 2", "Mar Elias
 3", "Martyr", "Mary", "Matthaf", "Maurice", "Mazraa 1", "Mazraa
 2", "Mazraa 3", "Mbarak", "Mheyddin", "Michel Abou
 Shahla", "Michel Shiha", "Middle East Road",
 "Monot Street", "Mousa Nammour", "Msaytbeh 1", "Msaytbeh
 2", "Muwawiya", "Najib Irdaty", "Nwayli", "Omar BAA 1", "Omar
 BAA 2", "Omar Bayham", "Omar Daouk", "Ouzay", "Paris",
 "Peter Street", "Picadilly", "Qulaylat", "Rachid
 Karameh", "Rachid", "Rachideen", "Rafik Hariri", "Rafik
 Hariri Plaza", "Ras Elnabe", "Rayes 1", "Rayes 2", "Rene
 Muawad 1", "Rene Muawad 2", "Rene Muawad 3",
 "Riad Solh 1", "Riad Solh 2", "Riad Solh 3", "Riverway
 1", "Riverway 2", "RML", "Roma 1", "Roma 2",
 "Royal Plaza", "Saddat 1", "Saddat 2", "Saeb Salam 1", "Saeb
 Salam 2", "Saeb Salam 3", "Saeb Salam 4", "Saeb Salam
 5", "Saeb Salam 6", "Salaheddin", "Salim Salam", "Salim
 Slim", "Sarolla", "Sea 1", "Sea 2", "Sea 3", "Sea 4", "Sea
 5", "Sea 6", "Sea 7", "Shatila", "Sidani 1", "Sidani
 2", "Sleiman Boustani", "Soroti
 1", "Soroti2", "Spears", "Tabaris", "Takieddine Solh 1",
 "TakieddineSolh2", "TawficSalem", "Transit", "Unesco", "Verdu
 n", "Vienna", "Wadi Sabra", "Wagen", "Wehdeh", "Zahret Ihsan",
 NULL };

//Conditions list array

```

char *CEmergencySystemsDlg::conditions[] = {"Abdominal
pain",
"Abnormal skin signs",
"Abnormal vital signs",
"Airway control/positioning required en route",

```

"Alcohol intoxication - severe",
"Alcohol intoxication or drug overdose (suspected)",
"Allergic reactions",
"Amputation (Lower Body)",
"Amputation (Upper Body)",
"Animal bites",
"Auto pedestrian/ bike",
"Back pain-non-traumatic",
"Blood Glucose",
"Bone fractures",
"Burns Major",
"Burns Minor",
"Cardiac symptoms other than chest pain",
"Cardiac/hemodynamic monitoring required en route",
"Cardiac Arrest",
"Chocking episode",
"Cold exposure",
"Combination of trauma and burns",
"Convulsions",
"Difficulty breathing",
"Dizziness",
"Electrocution",
"Envenomation",
"Eye injuries",
"Fainting",
"Fall",
"Hazmat Exposure",
"Heat exposure",
"Hemorrhage",
"Hospital to hospital transport",
"Infectious diseases",
"Lightning Strike",
"Major Trauma",
"Medical Device Failure",
"Motorcycle accident",
"Natural disasters",
"Near Drowning",
"Near syncope",
"Neurologic Distress",
"Skull fracture",
"Trauma",
"Pain (Acute)",
"Paralysis",
"Pedestrian thrown/run over",
"Pelvic fracture",
"Penetrating injuries",
"Poisons",
"Post-operative procedure complications",
"Pregnancy",
"Psychiatric",
"Stroke",

```

"Recent Fracture",
"Respiratory arrest",
"Seizures",
"Severe dehydration",
"Sick Person - Fever",
"Special handling en route - isolation",
"Special handling en route to reduce pain - orthopedic
device,"
"Stings",
"Suction required en route",
"Unconscious",
"Unconscious-Found",
"Weakness",
"UNKNOWN", NULL};

//Number of Injuries defined
char *CEmergencySystemsDlg::numbers[] = {"1 Case","2
Cases","3 Cases","4 Cases","5 Cases","5 Cases
+","UNKNOWN", NULL};

//Ambulances available
char *CEmergencySystemsDlg::ambulancevalues[] = {"101A",
"102B", "103C", "104D", NULL};

//*****

// when the "Select Road" button is clicked

void CEmergencySystemsDlg::OnSelected()
{

// Create 4 graphs as independent blocks
Graph rasbeirut(51);
Graph achrafieh(35);
Graph mar_elias(23);
Graph riverside(18);

//Creating the roads of the graph and linking the nodes
// 1st Graph (Ras Beirut)
rasbeirut.addWeight("Ras
Beirut",0,1,750,INTERSECTION,"Charles De Gaul 1");
rasbeirut.addWeight("Ras
Beirut",0,6,600,INTERSECTION,"Najib Irdaty");
rasbeirut.addWeight("Ras
Beirut",0,22,300,INTERSECTION,"Bahrain");
rasbeirut.addWeight("Ras
Beirut",0,29,700,INTERSECTION,"Charles De Gaul 2");
rasbeirut.addWeight("Ras
Beirut",1,0,750,INTERSECTION,"Charles De Gaul 1");
rasbeirut.addWeight("Ras
Beirut",1,2,1500,INTERSECTION,"Paris");

```

```

rasbeirut.addWeight("Ras
Beirut",2,1,1500,INTERSECTION,"Paris");
rasbeirut.addWeight("Ras
Beirut",2,3,500,INTERSECTION,"Ibn Sina");
rasbeirut.addWeight("Ras
Beirut",2,4,250,INTERSECTION,"Graham");
rasbeirut.addWeight("Ras
Beirut",3,2,800,INTERSECTION,"Rafik Hariri Plaza");
rasbeirut.addWeight("Ras
Beirut",3,5,250,INTERSECTION,"Fakhreddin 1");
rasbeirut.addWeight("Ras
Beirut",4,8,400,INTERSECTION,"John Kennedy");
rasbeirut.addWeight("Ras
Beirut",5,3,250,INTERSECTION,"Fakhreddin 1");
rasbeirut.addWeight("Ras
Beirut",5,4,600,INTERSECTION,"Omar Daouk");
rasbeirut.addWeight("Ras
Beirut",5,11,100,INTERSECTION,"Fakhreddin 2");
rasbeirut.addWeight("Ras
Beirut",6,0,600,INTERSECTION,"Najib Irdaty");
rasbeirut.addWeight("Ras
Beirut",6,1,300,INTERSECTION,"IC Street");
rasbeirut.addWeight("Ras
Beirut",6,12,250,HOSPITAL,"Sidani 1");
rasbeirut.addWeight("Ras
Beirut",6,17,250,INTERSECTION,"Saddat 1");
rasbeirut.addWeight("Ras
Beirut",7,6,350,INTERSECTION,"Bliss 2");
rasbeirut.addWeight("Ras
Beirut",7,13,200,INTERSECTION,"Jean D'arc 1");
rasbeirut.addWeight("Ras
Beirut",8,9,150,INTERSECTION,"Clemancau 3");
rasbeirut.addWeight("Ras
Beirut",8,7,250,INTERSECTION,"Bliss
1");rasbeirut.addWeight("Ras
Beirut",9,10,150,INTERSECTION,"Clemancau 2");
rasbeirut.addWeight("Ras Beirut",9,15,250,HOSPITAL,"Abdul
Baki");
rasbeirut.addWeight("Ras
Beirut",10,49,400,INTERSECTION,"Clemancau
1rasbeirut.addWeight("Ras
Beirut",10,20,350,INTERSECTION,"Roma 2");
rasbeirut.addWeight("Ras
Beirut",11,5,100,INTERSECTION,"Fakhreddin 2");
rasbeirut.addWeight("Ras
Beirut",11,50,600,INTERSECTION,"Fakhreddin 3");
rasbeirut.addWeight("Ras
Beirut",11,28,1300,REDCROSS,"Fakhreddin 3");
rasbeirut.addWeight("Ras
Beirut",12,13,10,INTERSECTION,"Sidani 2");

```

```

rasbeirut.addWeight("Ras
Beirut",13,14,250,INTERSECTION,"Soroti 1");
rasbeirut.addWeight("Ras
Beirut",13,18,250,INTERSECTION,"Jean D'arc 2");
rasbeirut.addWeight("Ras
Beirut",14,8,250,INTERSECTION,"Omar BAA 1");
rasbeirut.addWeight("Ras
Beirut",14,15,600,HOSPITAL,"Soroti 2");
rasbeirut.addWeight("Ras
Beirut",16,49,50,INTERSECTION,"Jositinian 1");
rasbeirut.addWeight("Ras
Beirut",16,21,100,INTERSECTION,"Jositinian 2");
rasbeirut.addWeight("Ras
Beirut",17,22,400,INTERSECTION,"Kuwait");
rasbeirut.addWeight("Ras
Beirut",17,23,250,INTERSECTION,"Saddat 2");
rasbeirut.addWeight("Ras
Beirut",18,17,300,INTERSECTION,"Hamra 2");
rasbeirut.addWeight("Ras
Beirut",18,24,200,INTERSECTION,"Baalbek");
rasbeirut.addWeight("Ras
Beirut",19,14,250,INTERSECTION,"Omar BAA 2");
rasbeirut.addWeight("Ras
Beirut",19,18,250,INTERSECTION,"Hamra 1");
rasbeirut.addWeight("Ras
Beirut",20,15,100,HOSPITAL,"Cairo");
rasbeirut.addWeight("Ras
Beirut",20,19,200,INTERSECTION,"Sarolla");
rasbeirut.addWeight("Ras
Beirut",20,26,100,INTERSECTION,"RML");
rasbeirut.addWeight("Ras
Beirut",21,16,100,HOSPITAL,"Jositinian 2");
rasbeirut.addWeight("Ras
Beirut",21,20,300,INTERSECTION,"Banque Liban");
rasbeirut.addWeight("Ras
Beirut",21,27,50,INTERSECTION,"Industry");
rasbeirut.addWeight("Ras
Beirut",22,0,300,INTERSECTION,"Bahrain");
rasbeirut.addWeight("Ras
Beirut",22,23,350,INTERSECTION,"Alaa");
rasbeirut.addWeight("Ras
Beirut",22,29,300,INTERSECTION,"Salaheddin");
rasbeirut.addWeight("Ras
Beirut",23,22,350,INTERSECTION,"Alaa");
rasbeirut.addWeight("Ras
Beirut",23,24,400,INTERSECTION,"Hsein 1");
rasbeirut.addWeight("Ras
Beirut",23,30,350,INTERSECTION,"Badr Demashkieh");
rasbeirut.addWeight("Ras
Beirut",23,31,500,INTERSECTION,"Hsein 2");

```

```

rasbeirut.addWeight("Ras
Beirut",24,25,300,INTERSECTION,"Emil 1");
rasbeirut.addWeight("Ras
Beirut",24,31,250,INTERSECTION,"Wadi Sabra");
rasbeirut.addWeight("Ras
Beirut",25,19,250,INTERSECTION,"Picadilly");
rasbeirut.addWeight("Ras
Beirut",25,26,400,INTERSECTION,"Emil 2");
rasbeirut.addWeight("Ras
Beirut",26,27,250,INTERSECTION,"Government Palace");
rasbeirut.addWeight("Ras
Beirut",26,33,250,INTERSECTION,"Rene Muawad 1");
rasbeirut.addWeight("Ras
Beirut",27,21,50,INTERSECTION,"Industry");
rasbeirut.addWeight("Ras
Beirut",27,28,500,REDCROSS,"Spears");
rasbeirut.addWeight("Ras
Beirut",28,50,100,INTERSECTION,"Fakhreddin 4");
rasbeirut.addWeight("Ras
Beirut",50,28,100,INTERSECTION,"Fakhreddin 4");
rasbeirut.addWeight("Ras
Beirut",50,21,400,INTERSECTION,"Michel Shiha");
rasbeirut.addWeight("Ras
Beirut",28,48,250,INTERSECTION,"Bheim");
rasbeirut.addWeight("Ras
Beirut",29,0,700,INTERSECTION,"Charles De Gaul 2");
rasbeirut.addWeight("Ras
Beirut",29,22,300,INTERSECTION,"Salaheddin");
rasbeirut.addWeight("Ras
Beirut",29,30,250,INTERSECTION,"Shatila");
rasbeirut.addWeight("Ras
Beirut",29,36,700,INTERSECTION,"Charles De Gaul 3");
rasbeirut.addWeight("Ras
Beirut",30,23,350,INTERSECTION,"Badr Demashkieh");
rasbeirut.addWeight("Ras
Beirut",30,29,250,INTERSECTION,"Shatila");
rasbeirut.addWeight("Ras
Beirut",30,31,650,INTERSECTION,"Madame Curie 1");
rasbeirut.addWeight("Ras
Beirut",30,37,250,INTERSECTION,"Takieddine Solh 1");
rasbeirut.addWeight("Ras
Beirut",31,32,125,INTERSECTION,"Madame Curie 2");
rasbeirut.addWeight("Ras
Beirut",32,25,200,INTERSECTION,"Alfred");
rasbeirut.addWeight("Ras
Beirut",32,33,400,INTERSECTION,"Henry Donald");
rasbeirut.addWeight("Ras
Beirut",32,44,250,INTERSECTION,"Rachid Karamah");
rasbeirut.addWeight("Ras
Beirut",33,34,500,HOSPITAL,"Mary");

```

```

rasbeirut.addWeight("Ras
Beirut", 33, 39, 200, INTERSECTION, "Rene Muawad 2");
rasbeirut.addWeight("Ras
Beirut", 34, 33, 500, INTERSECTION, "Mary");
rasbeirut.addWeight("Ras
Beirut", 34, 35, 250, INTERSECTION, "Rachid");
rasbeirut.addWeight("Ras
Beirut", 34, 40, 250, INTERSECTION, "Jacob");
rasbeirut.addWeight("Ras
Beirut", 35, 34, 250, HOSPITAL, "Rachid");
rasbeirut.addWeight("Ras
Beirut", 35, 41, 400, INTERSECTION, "Mar Elias 1");
rasbeirut.addWeight("Ras
Beirut", 35, 48, 300, INTERSECTION, "Maurice");
rasbeirut.addWeight("Ras
Beirut", 36, 29, 700, INTERSECTION, "Charles De Gaul 3");
rasbeirut.addWeight("Ras
Beirut", 36, 37, 600, INTERSECTION, "Andalus");
rasbeirut.addWeight("Ras
Beirut", 36, 42, 250, INTERSECTION, "Royal Plaza");
rasbeirut.addWeight("Ras
Beirut", 37, 30, 250, INTERSECTION, "Takieddine Solh 1");
rasbeirut.addWeight("Ras
Beirut", 37, 36, 600, INTERSECTION, "Andalus");
rasbeirut.addWeight("Ras
Beirut", 37, 38, 500, INTERSECTION, "Takieddine Solh 2");
rasbeirut.addWeight("Ras
Beirut", 38, 31, 100, INTERSECTION, "Itani 1");
rasbeirut.addWeight("Ras
Beirut", 38, 37, 500, INTERSECTION, "Takieddine Solh 2");
rasbeirut.addWeight("Ras
Beirut", 38, 44, 250, INTERSECTION, "Mbarak");
rasbeirut.addWeight("Ras
Beirut", 39, 33, 200, INTERSECTION, "Rene Muawad 2");
rasbeirut.addWeight("Ras
Beirut", 39, 40, 300, INTERSECTION, "Istiklal 1");
rasbeirut.addWeight("Ras
Beirut", 39, 44, 250, INTERSECTION, "Betty");
rasbeirut.addWeight("Ras
Beirut", 39, 45, 100, INTERSECTION, "Rene Muawad 3");
rasbeirut.addWeight("Ras
Beirut", 40, 41, 300, INTERSECTION, "Istiklal 2");
rasbeirut.addWeight("Ras
Beirut", 40, 46, 250, INTERSECTION, "Madhat Pasha");
rasbeirut.addWeight("Ras
Beirut", 41, 35, 400, INTERSECTION, "Mar Elias 1");
rasbeirut.addWeight("Ras
Beirut", 41, 47, 300, INTERSECTION, "Mar Elias 2");
rasbeirut.addWeight("Ras
Beirut", 42, 36, 250, INTERSECTION, "Royal Plaza");

```

```

rasbeirut.addWeight("Ras
Beirut",43,38,250,INTERSECTION,"Itani 2");
rasbeirut.addWeight("Ras
Beirut",43,42,350,INTERSECTION,"Berlin");
rasbeirut.addWeight("Ras
Beirut",43,44,450,INTERSECTION,"Wehdeh");
rasbeirut.addWeight("Ras
Beirut",44,32,250,INTERSECTION,"Rachid Karameh");
rasbeirut.addWeight("Ras
Beirut",44,38,250,INTERSECTION,"Mbarak");
rasbeirut.addWeight("Ras
Beirut",44,39,250,INTERSECTION,"Betty");
rasbeirut.addWeight("Ras
Beirut",44,43,450,INTERSECTION,"Wehdeh");
rasbeirut.addWeight("Ras
Beirut",44,45,250,INTERSECTION,"Ibn Rushd 1");
rasbeirut.addWeight("Ras
Beirut",45,39,100,INTERSECTION,"Rene Muawad 3");
rasbeirut.addWeight("Ras
Beirut",45,44,250,INTERSECTION,"Ibn Rushd 1");
rasbeirut.addWeight("Ras
Beirut",45,46,250,INTERSECTION,"Ibn Rushd 2");
rasbeirut.addWeight("Ras
Beirut",46,47,300,INTERSECTION,"Ibn Rushd 3");
rasbeirut.addWeight("Ras
Beirut",47,41,300,INTERSECTION,"Mar Elias 2");
rasbeirut.addWeight("Ras
Beirut",48,28,250,REDCROSS,"Shatila");
rasbeirut.addWeight("Ras
Beirut",49,11,700,INTERSECTION,"Clemancau");

```

```

// 2nd Graph (Achrafieh)
achrafieh.addWeight("Achrafieh",0,1,350,INTERSECTION,"Sea
2");
achrafieh.addWeight("Achrafieh",0,2,500,INTERSECTION,"Sea
1");
achrafieh.addWeight("Achrafieh",0,3,400,INTERSECTION,"Sea
5");
achrafieh.addWeight("Achrafieh",1,0,350,INTERSECTION,"Sea
2");
achrafieh.addWeight("Achrafieh",1,4,450,INTERSECTION,"Sea
6");
achrafieh.addWeight("Achrafieh",1,5,900,INTERSECTION,"Sea
3");
achrafieh.addWeight("Achrafieh",2,0,500,INTERSECTION,"Sea
1");
achrafieh.addWeight("Achrafieh",2,3,300,INTERSECTION,"Sea
4");
achrafieh.addWeight("Achrafieh",3,0,400,INTERSECTION,"Sea
5");

```



```

achrafieh.addWeight("Achrafieh",3,2,300,INTERSECTION,"Sea
4");
achrafieh.addWeight("Achrafieh",3,4,350,INTERSECTION,"Sea
7");
achrafieh.addWeight("Achrafieh",3,6,400,INTERSECTION,"Ahm
ed Shawki");
achrafieh.addWeight("Achrafieh",4,1,450,INTERSECTION,"Sea
6");
achrafieh.addWeight("Achrafieh",4,3,350,INTERSECTION,"Sea
7");
achrafieh.addWeight("Achrafieh",4,5,300,INTERSECTION,"Tra
nsit");
achrafieh.addWeight("Achrafieh",4,7,500,INTERSECTION,"Lim
by");
achrafieh.addWeight("Achrafieh",5,1,900,INTERSECTION,"Sea
3");
achrafieh.addWeight("Achrafieh",5,4,300,INTERSECTION,"Tra
nsit");
achrafieh.addWeight("Achrafieh",5,8,300,INTERSECTION,"Mar
tyr");
achrafieh.addWeight("Achrafieh",5,11,700,INTERSECTION,"Ge
orge Haddad");
achrafieh.addWeight("Achrafieh",6,3,400,INTERSECTION,"Ahm
ed Shawki");
achrafieh.addWeight("Achrafieh",6,7,400,INTERSECTION,"Wag
en");
achrafieh.addWeight("Achrafieh",7,4,500,INTERSECTION,"Lim
by");
achrafieh.addWeight("Achrafieh",7,8,300,INTERSECTION,"Ced
ars");
achrafieh.addWeight("Achrafieh",7,9,350,INTERSECTION,"Ria
d Solh 1");
achrafieh.addWeight("Achrafieh",8,5,300,INTERSECTION,"Mar
tyr");
achrafieh.addWeight("Achrafieh",8,10,300,INTERSECTION,"Ib
rahim River");
achrafieh.addWeight("Achrafieh",9,10,350,INTERSECTION,"Am
ir Bachir");
achrafieh.addWeight("Achrafieh",9,12,300,REDCROSS,"Riad
Solh 2");
achrafieh.addWeight("Achrafieh",10,7,250,INTERSECTION,"Ri
ad Solh 3");
achrafieh.addWeight("Achrafieh",10,11,200,INTERSECTION,"G
oro 1");
achrafieh.addWeight("Achrafieh",10,13,300,INTERSECTION,"D
abbas");
achrafieh.addWeight("Achrafieh",11,5,700,INTERSECTION,"Ge
orge Haddad");
achrafieh.addWeight("Achrafieh",11,14,400,INTERSECTION,"C
harles Street");

```

```

achrafieh.addWeight("Achrafieh",12,9,300,INTERSECTION,"Ri
ad Solh 2");
achrafieh.addWeight("Achrafieh",12,13,500,INTERSECTION,"F
ouad Shhab 1");
achrafieh.addWeight("Achrafieh",12,22,900,INTERSECTION,"S
alim Slim");
achrafieh.addWeight("Achrafieh",13,10,300,INTERSECTION,"D
abbas");
achrafieh.addWeight("Achrafieh",13,12,500,INTERSECTION,"F
ouad Shhab 1");
achrafieh.addWeight("Achrafieh",13,14,300,INTERSECTION,"T
abaris");
achrafieh.addWeight("Achrafieh",13,17,500,INTERSECTION,"B
chara Khoury 1");
achrafieh.addWeight("Achrafieh",14,11,400,INTERSECTION,"C
harles Street");
achrafieh.addWeight("Achrafieh",14,13,300,INTERSECTION,"T
abaris");
achrafieh.addWeight("Achrafieh",14,15,250,INTERSECTION,"G
handour");
achrafieh.addWeight("Achrafieh",15,14,250,INTERSECTION,"G
handour");
achrafieh.addWeight("Achrafieh",15,18,600,INTERSECTION,"M
onot Street");
achrafieh.addWeight("Achrafieh",16,17,900,INTERSECTION,"I
stiklal 3");
achrafieh.addWeight("Achrafieh",17,13,500,INTERSECTION,"B
chara Khoury 1");
achrafieh.addWeight("Achrafieh",17,16,900,INTERSECTION,"M
onot Street");
achrafieh.addWeight("Achrafieh",17,18,300,INTERSECTION,"I
stiklal 4");
achrafieh.addWeight("Achrafieh",17,24,500,INTERSECTION,"B
chara Khoury 2");
achrafieh.addWeight("Achrafieh",17,25,700,INTERSECTION,"D
amascus 1");
achrafieh.addWeight("Achrafieh",18,15,600,INTERSECTION,"M
onot Street");
achrafieh.addWeight("Achrafieh",18,17,300,INTERSECTION,"I
stiklal 4");
achrafieh.addWeight("Achrafieh",18,19,900,INTERSECTION,"E
lias Sarkis 1");
achrafieh.addWeight("Achrafieh",18,29,1000,HOSPITAL,"Habi
b Bacha 1");
achrafieh.addWeight("Achrafieh",19,18,900,INTERSECTION,"E
lias Sarkis 1");
achrafieh.addWeight("Achrafieh",19,20,200,INTERSECTION,"E
lias Sarkis 2");
achrafieh.addWeight("Achrafieh",19,29,700,HOSPITAL,"Adib
Isaac");

```

```

achrafieh.addWeight("Achrafieh",20,19,200,INTERSECTION,"E
lias Sarkis 2");
achrafieh.addWeight("Achrafieh",20,29,1200,HOSPITAL,"Alfr
ed Na'ach");
achrafieh.addWeight("Achrafieh",21,16,400,INTERSECTION,"M
saytbeh 1");
achrafieh.addWeight("Achrafieh",22,12,900,REDCROSS,"Salim
Slim");
achrafieh.addWeight("Achrafieh",22,21,200,INTERSECTION,"M
ichel Abou Shahla");
achrafieh.addWeight("Achrafieh",23,12,1000,REDCROSS,"Mhey
ddin");
achrafieh.addWeight("Achrafieh",23,22,800,INTERSECTION,"A
bdul Ghani");
achrafieh.addWeight("Achrafieh",23,26,300,INTERSECTION,"N
wayli");
achrafieh.addWeight("Achrafieh",24,17,500,INTERSECTION,"B
chara Khoury 2");
achrafieh.addWeight("Achrafieh",24,23,250,INTERSECTION,"R
as Elnabe");
achrafieh.addWeight("Achrafieh",24,27,250,INTERSECTION,"B
chara Khoury 3");
achrafieh.addWeight("Achrafieh",25,17,700,INTERSECTION,"D
amascus 1");
achrafieh.addWeight("Achrafieh",25,24,500,INTERSECTION,"T
awfic Salem");
achrafieh.addWeight("Achrafieh",25,28,400,INTERSECTION,"D
amascus 2");
achrafieh.addWeight("Achrafieh",25,32,750,INTERSECTION,"D
amascus 3");
achrafieh.addWeight("Achrafieh",26,23,300,INTERSECTION,"N
wayli");
achrafieh.addWeight("Achrafieh",26,27,250,INTERSECTION,"F
adel Haraty");
achrafieh.addWeight("Achrafieh",26,30,250,HOSPITAL,"Ouzay
");
achrafieh.addWeight("Achrafieh",27,24,250,INTERSECTION,"B
chara Khoury 3");
achrafieh.addWeight("Achrafieh",27,28,500,INTERSECTION,"Q
ulaylat");
achrafieh.addWeight("Achrafieh",27,31,200,INTERSECTION,"B
chara Khoury 4");
achrafieh.addWeight("Achrafieh",28,25,400,INTERSECTION,"D
amascus 2");
achrafieh.addWeight("Achrafieh",28,29,600,HOSPITAL,"Habib
Bacha 2");
achrafieh.addWeight("Achrafieh",28,32,300,INTERSECTION,"D
amascus 4");
achrafieh.addWeight("Achrafieh",29,18,1000,INTERSECTION,"
Habib Bacha 1");

```

```

achrafieh.addWeight("Achrafieh",29,19,700,INTERSECTION,"A
dib Isaac");
achrafieh.addWeight("Achrafieh",29,20,1200,INTERSECTION,"
Alfred Na'ach");
achrafieh.addWeight("Achrafieh",29,33,250,INTERSECTION,"K
halil");
achrafieh.addWeight("Achrafieh",30,26,250,INTERSECTION,"O
uzay");
achrafieh.addWeight("Achrafieh",31,27,200,INTERSECTION,"B
chara Khoury 4");
achrafieh.addWeight("Achrafieh",31,32,700,INTERSECTION,"A
bdullah 1");
achrafieh.addWeight("Achrafieh",31,34,600,HOSPITAL,"Omar
Bayham");
achrafieh.addWeight("Achrafieh",32,25,750,INTERSECTION,"D
amascus 3");
achrafieh.addWeight("Achrafieh",32,28,300,INTERSECTION,"D
amascus 4");
achrafieh.addWeight("Achrafieh",32,31,700,INTERSECTION,"A
bdullah 1");
achrafieh.addWeight("Achrafieh",32,33,500,INTERSECTION,"A
bdullah 2");
achrafieh.addWeight("Achrafieh",32,34,500,HOSPITAL,"Matth
af");
achrafieh.addWeight("Achrafieh",33,29,250,HOSPITAL,"Khali
l");
achrafieh.addWeight("Achrafieh",33,32,500,INTERSECTION,"A
bdullah 2");
achrafieh.addWeight("Achrafieh",34,31,600,INTERSECTION,"O
mar Bayham");

// 3rd Graph (Mar Elias)
mar_elias.addWeight("Mar
Elias",0,1,500,INTERSECTION,"Vienna");
mar_elias.addWeight("Mar
Elias",0,6,400,INTERSECTION,"Saeb Salam 1");
mar_elias.addWeight("Mar
Elias",0,12,1300,INTERSECTION,"Rafik Hariri");
mar_elias.addWeight("Mar
Elias",1,2,300,INTERSECTION,"Unesco");
mar_elias.addWeight("Mar
Elias",1,6,400,INTERSECTION,"Verdun");
mar_elias.addWeight("Mar
Elias",2,1,300,INTERSECTION,"Unesco");
mar_elias.addWeight("Mar
Elias",2,3,300,INTERSECTION,"Mazraa 1");
mar_elias.addWeight("Mar
Elias",2,7,250,INTERSECTION,"Rachideen");
mar_elias.addWeight("Mar
Elias",3,2,300,INTERSECTION,"Mazraa 1");

```

```

mar_elias.addWeight("Mar
Elias",3,4,400,INTERSECTION,"Mazraa 2");
mar_elias.addWeight("Mar
Elias",4,3,400,INTERSECTION,"Mazraa 2");
mar_elias.addWeight("Mar
Elias",4,5,250,INTERSECTION,"Mazraa 3");
mar_elias.addWeight("Mar Elias",4,8,300,INTERSECTION,"Mar
Elias 3");
mar_elias.addWeight("Mar
Elias",5,4,250,INTERSECTION,"Mazraa 3");
mar_elias.addWeight("Mar
Elias",6,0,400,INTERSECTION,"Saeb Salam
1");mar_elias.addWeight("Mar
Elias",6,1,400,INTERSECTION,"Verdun");
mar_elias.addWeight("Mar
Elias",6,7,300,INTERSECTION,"Saeb Salam 2");
mar_elias.addWeight("Mar
Elias",6,14,400,INTERSECTION,"Fouad Tarrad");
mar_elias.addWeight("Mar
Elias",7,2,250,INTERSECTION,"Rachideen");
mar_elias.addWeight("Mar
Elias",7,6,300,INTERSECTION,"Saeb Salam 2");
mar_elias.addWeight("Mar
Elias",7,8,400,INTERSECTION,"Saeb Salam 3");
mar_elias.addWeight("Mar
Elias",7,15,400,INTERSECTION,"Habib Shahla 1");
mar_elias.addWeight("Mar Elias",8,4,300,INTERSECTION,"Mar
Elias 3");
mar_elias.addWeight("Mar
Elias",8,7,400,INTERSECTION,"Saeb Salam 3");
mar_elias.addWeight("Mar
Elias",8,9,300,INTERSECTION,"Saeb Salam 4");
mar_elias.addWeight("Mar
Elias",8,16,300,INTERSECTION,"Muwawiya");
mar_elias.addWeight("Mar
Elias",9,5,250,INTERSECTION,"Bachir Jumblat");
mar_elias.addWeight("Mar
Elias",9,8,300,INTERSECTION,"Saeb Salam 4");
mar_elias.addWeight("Mar
Elias",9,10,100,INTERSECTION,"Saeb Salam 5");
mar_elias.addWeight("Mar
Elias",10,9,100,INTERSECTION,"Saeb Salam 5");
mar_elias.addWeight("Mar
Elias",10,11,450,INTERSECTION,"Saeb Salam 6");
mar_elias.addWeight("Mar
Elias",10,18,400,INTERSECTION,"Salim Salam");
mar_elias.addWeight("Mar
Elias",11,10,450,INTERSECTION,"Saeb Salam 6");
mar_elias.addWeight("Mar
Elias",12,0,1300,INTERSECTION,"Rafik Hariri");

```

```

mar_elias.addWeight("Mar
Elias",12,13,300,HOSPITAL,"Middle East Road");
mar_elias.addWeight("Mar
Elias",13,12,300,INTERSECTION,"Middle East Road");
mar_elias.addWeight("Mar
Elias",13,14,250,INTERSECTION,"Farid Trad 2");
mar_elias.addWeight("Mar
Elias",14,13,250,INTERSECTION,"Farid Trad 2");
mar_elias.addWeight("Mar
Elias",14,15,300,INTERSECTION,"Peter Street");
mar_elias.addWeight("Mar
Elias",15,7,400,INTERSECTION,"Habib Shahla 1");
mar_elias.addWeight("Mar
Elias",15,16,250,INTERSECTION,"Habib Shahla 2");
mar_elias.addWeight("Mar
Elias",16,15,250,INTERSECTION,"Habib Shahla 2");
mar_elias.addWeight("Mar
Elias",16,17,250,INTERSECTION,"Habib Shahal 3");
mar_elias.addWeight("Mar
Elias",17,9,300,INTERSECTION,"Msaytbeh 2");
mar_elias.addWeight("Mar
Elias",17,16,250,INTERSECTION,"Habib Shahla 3");
mar_elias.addWeight("Mar
Elias",17,18,200,INTERSECTION,"Habib Shahla 4");
mar_elias.addWeight("Mar
Elias",18,10,400,INTERSECTION,"Salim Salam");
mar_elias.addWeight("Mar
Elias",18,17,200,INTERSECTION,"Habib Shahla 4");
mar_elias.addWeight("Mar
Elias",18,19,400,INTERSECTION,"Sleiman Boustani");
mar_elias.addWeight("Mar
Elias",18,21,500,REDCROSS,"Khalil Moutra");
mar_elias.addWeight("Mar Elias",18,22,500,HOSPITAL,"Mousa
Nammour");
mar_elias.addWeight("Mar
Elias",19,11,300,INTERSECTION,"Abou
Shaker");mar_elias.addWeight("Mar
Elias",19,20,400,HOSPITAL,"Makased Hospital Street");
mar_elias.addWeight("Mar
Elias",20,19,400,INTERSECTION,"Makased Hospital Street");
mar_elias.addWeight("Mar
Elias",21,18,500,HOSPITAL,"Khalil Moutran");
mar_elias.addWeight("Mar
Elias",22,18,500,INTERSECTION,"Mousa Nammour");

// 4th Graph (River Side)
riverside.addWeight("River
Side",0,1,500,HOSPITAL,"Brasil");
riverside.addWeight("River
Side",0,2,1000,INTERSECTION,"Charles Helou");

```

```

riverside.addWeight("River
Side",0,7,350,INTERSECTION,"Madrid");
riverside.addWeight("River
Side",1,0,500,INTERSECTION,"Brasil");
riverside.addWeight("River
Side",2,0,1000,INTERSECTION,"Charles Helou");
riverside.addWeight("River
Side",2,5,200,INTERSECTION,"Corniche Jamil 1");
riverside.addWeight("River
Side",3,7,350,INTERSECTION,"Goro 2");
riverside.addWeight("River
Side",3,10,400,INTERSECTION,"Akkawi");
riverside.addWeight("River
Side",4,5,650,INTERSECTION,"Riverway 1");
riverside.addWeight("River
Side",4,7,600,INTERSECTION,"Riverway 2");
riverside.addWeight("River
Side",4,9,400,HOSPITAL,"Kabiyat
1");riverside.addWeight("River
Side",5,2,200,INTERSECTION,"Corniche Jamil 1");
riverside.addWeight("River
Side",5,4,650,INTERSECTION,"Riverway 1");
riverside.addWeight("River
Side",5,14,600,INTERSECTION,"Corniche Jamil 2");
riverside.addWeight("River
Side",6,10,250,INTERSECTION,"Fouad Shhab 2");
riverside.addWeight("River
Side",7,0,350,INTERSECTION,"Madrid");
riverside.addWeight("River
Side",7,4,600,INTERSECTION,"Riverway 2");
riverside.addWeight("River Side",7,8,250,HOSPITAL,"Rayes
1");
riverside.addWeight("River
Side",7,11,500,INTERSECTION,"Fard Naash 1");
riverside.addWeight("River
Side",8,7,250,INTERSECTION,"Rayes 1");
riverside.addWeight("River
Side",8,12,300,INTERSECTION,"Rayes 2");
riverside.addWeight("River
Side",9,13,500,INTERSECTION,"Kabiyat 2");
riverside.addWeight("River
Side",10,3,400,INTERSECTION,"Akkawi");
riverside.addWeight("River Side",10,6,250,REDCROSS,"Fouad
Shhab 2");
riverside.addWeight("River
Side",10,11,250,INTERSECTION,"Fouad Shhab 3");
riverside.addWeight("River
Side",10,15,300,INTERSECTION,"Zahret Ihsan");
riverside.addWeight("River
Side",11,7,500,INTERSECTION,"Fard Naash 1");

```

```

riverside.addWeight("River
Side",11,10,250,INTERSECTION,"Fouad Shhab 3");
riverside.addWeight("River
Side",11,12,250,INTERSECTION,"Fouad Shhab 4");
riverside.addWeight("River Side",12,8,300,HOSPITAL,"Rayes
2");
riverside.addWeight("River
Side",12,11,250,INTERSECTION,"Fouad Shhab 4");
riverside.addWeight("River
Side",12,13,600,INTERSECTION,"Fouad Shhab 5");
riverside.addWeight("River
Side",13,9,500,HOSPITAL,"Kabiyat 2");
riverside.addWeight("River
Side",13,12,600,INTERSECTION,"Fouad Shhab 5");
riverside.addWeight("River
Side",13,14,250,INTERSECTION,"Fouad Shhab 6");
riverside.addWeight("River
Side",13,16,400,INTERSECTION,"Ararat");
riverside.addWeight("River
Side",14,5,600,INTERSECTION,"Corniche Jamil 2");
riverside.addWeight("River
Side",14,13,250,INTERSECTION,"Fouad Shhab 6");
riverside.addWeight("River
Side",14,17,750,INTERSECTION,"Corniche Jamil 3");
riverside.addWeight("River
Side",15,10,300,INTERSECTION,"Zahret Ihsan");
riverside.addWeight("River
Side",16,13,400,INTERSECTION,"Ararat");
riverside.addWeight("River
Side",16,17,850,INTERSECTION,"Cheikh Gabi");
riverside.addWeight("River
Side",17,14,750,INTERSECTION,"Corniche Jamil 3");
riverside.addWeight("River
Side",17,16,850,INTERSECTION,"Cheikh Gabi");

```

```

string display = "";

```

```

CListBox *StreetListBox;                                //
create a new list box
    StreetListBox = ( CListBox * ) GetDlgItem(
IDC_STREETLISTBOX );    // pointer (StreetListBox) to
listbox (IDC_STREETLIST)
    int listindex = StreetListBox->GetCurSel();
    // gets index of selected listbox item
    if ( listindex == LB_ERR )
        // if no item is selected --> exit function
    {
        MessageBox( "Select Target
Street", "Warning!", MB_ICONWARNING );
        return;
    }

```



```

    }

    CListBox *OutputListBox;
    // create a new list box
    OutputListBox = ( CListBox * ) GetDlgItem(
IDC_OUTPUTLISTBOX );    // pointer (StreetsList) to
IDC_OUTPUTLIST listbox
    char temp[ 32 ];
    // temp array to store the string selected
    // in the IDC_STREETSLIST listbox
    StreetListBox->GetText( listindex, temp );
    // calls CListBox member function 'GetText' to get
the string at 'iCurSel' index from the CListBox object
pointed to by 'StreetsList' and store the string in
'temp' targetstreet=(string) temp; prepare string target
street for finalSMS display = "Traget Street: " +
targetstreet; OutputListBox->AddString( display.data() );
// add string 'temp' to listbox pointed to by
IDC_OUTPUTLIST
    display = "";

//Find Which Graph the Street is in *****

int site = 0, // accident site vertex index head
tail=0,      // tail of the accident site street
distance1 = 0, // distance from ambulance --> accident
site
distance2 = 0, // distance from crashsite --> hospital
d2a = 0,     // temporary distance trials
d2b = 0,     //
d2c = 0,     //
d2d = 0,     //
totaldistance; //
totaldistance = distance1 + distance 2

string s1 = rasbeirut.getGraphName( roadlist[ listindex
]);
string s2 = achrafieh.getGraphName( roadlist[ listindex
]);
string s3 = mar_elias.getGraphName( roadlist[ listindex
]);
string s4 = riverside.getGraphName( roadlist[ listindex
]);

string x1,x2,x2a,x2b,x2c,x2d, hospital, center; //string
declarations

if( (s1.compare("notexist")) != 0 )
{
center = "101 - Spears Red Cross Center";
display = " 1. Street Belongs to the Section: " + s1;

```

```

OutputListBox->AddString( display.data() );
display = "";

// Dijkstra between ambulance center and accident site
// Algorithm is run between the ambulance node and the
head of the street's node
site = rasbeirut.getHead( roadlist[ listindex ] );
//edited
distance1 = rasbeirut.Dijkstra(rasbeirut, 28, site);
x1 = rasbeirut.returnRoads(rasbeirut, site);

//Get the tail of the street
tail = rasbeirut.getTail( roadlist[listindex]);
// Dijkstra between accident site & all hospitals in that
block
d2a = rasbeirut.Dijkstra(rasbeirut, tail, 12); x2a =
rasbeirut.returnRoads(rasbeirut, 12);
d2b = rasbeirut.Dijkstra(rasbeirut, tail, 15); x2b =
rasbeirut.returnRoads(rasbeirut, 15);
d2c = rasbeirut.Dijkstra(rasbeirut, tail, 16); x2c =
rasbeirut.returnRoads(rasbeirut, 16);
d2d = rasbeirut.Dijkstra(rasbeirut, tail, 34); x2d =
rasbeirut.returnRoads(rasbeirut, 34);

distance2 = d2a; x2 = x2a; hospital = "Khalidy Hospital";
if (d2b < distance2){ distance2 = d2b; x2 = x2b; hospital
= "AUH Hospital"; }
if (d2c < distance2){ distance2 = d2c; x2 = x2c; hospital
= "Trad Hospital"; }
if (d2d < distance2){ distance2 = d2d; x2 = x2d; hospital
= "Atibba Hospital"; }
}
else if ( (s2.compare("notexist")) != 0 )
{
center = "103 Civil Defense Red Cross Center";
display = " 1. Street Belongs to the Section: " + s2;
OutputListBox->AddString( display.data() );
display = "";

// Dijkstra between ambulance center and accident site
site = achrafieh.getHead( roadlist[ listindex ] );
distance1 = achrafieh.Dijkstra(achrafieh, 12, site);
x1 = achrafieh.returnRoads(achrafieh, site);
tail = rasbeirut.getTail( roadlist[listindex]);
// Dijkstra between accident site & all hospitals in that
block
d2a = achrafieh.Dijkstra(achrafieh, tail, 29); x2a =
achrafieh.returnRoads(achrafieh, 29);
d2b = achrafieh.Dijkstra(achrafieh, tail, 30); x2b =
achrafieh.returnRoads(achrafieh, 30);

```

```

d2c = achrafieh.Dijkstra(achrafieh, tail, 34); x2c =
achrafieh.returnRoads(achrafieh, 34);
distance2 = d2a; x2 = x2a; hospital = "Hotel Dieu
Hospital";
if (d2b < distance2){ distance2 = d2b; x2 = x2b; hospital
= "Barbir Hospital"; }
if (d2c < distance2){ distance2 = d2c; x2 = x2c; hospital
= "Military Hospital"; }
}
else if ( (s3.compare("notexist")) != 0 )
{
center = "102 Dah'ieh Red Cross Center";
display = " 1. Street Belongs to the Section: " + s3;
OutputListBox->AddString( display.data() );
display = "";

// Dijkstra between ambulance center and accident site
site = mar_elias.getHead( roadlist[ listindex ] );
distance1 = mar_elias.Dijkstra(mar_elias, 21, site);
x1 = mar_elias.returnRoads(mar_elias, site);
tail = rasbeirut.getTail( roadlist[listindex] );
// Dijkstra between accident site & all hospitals in that
block
d2a = mar_elias.Dijkstra(mar_elias, tail, 13); x2a =
mar_elias.returnRoads(mar_elias, 13);
d2b = mar_elias.Dijkstra(mar_elias, tail, 20); x2b =
mar_elias.returnRoads(mar_elias, 20);
d2c = mar_elias.Dijkstra(mar_elias, tail, 22); x2c =
mar_elias.returnRoads(mar_elias, 22);
distance2 = d2a; x2 = x2a; hospital = "Middle East
Hospital";
if (d2b < distance2){ distance2 = d2b; x2 = x2b; hospital
= "Makassed Hospital"; }
if (d2c < distance2){ distance2 = d2c; x2 = x2c; hospital
= "Beirut Hospital"; }
}
else if ( (s4.compare("notexist")) != 0 )
{
center = "104 Tabarees Red Cross Center";
display = " 1. Street Belongs to the Section: " + s4;
OutputListBox->AddString( display.data() );
display = "";

// Dijkstra between ambulance center and accident site
site = riverside.getHead( roadlist[ listindex ] );
distance1 = riverside.Dijkstra(riverside, 6, site);
x1 = riverside.returnRoads(riverside, site);
tail = rasbeirut.getTail( roadlist[listindex] );
// Dijkstra between accident site & all hospitals in that
block

```

```

d2a = riverside.Dijkstra(riverside, tail, 1); x2a =
riverside.returnRoads(riverside, 1);
d2b = riverside.Dijkstra(riverside, tail, 8); x2b =
riverside.returnRoads(riverside, 8);
d2c = riverside.Dijkstra(riverside, tail, 9); x2c =
riverside.returnRoads(riverside, 9);

distance2 = d2a; x2 = x2a; hospital = "Karantina
Hospital";
if (d2b < distance2){ distance2 = d2b; x2 = x2b; hospital
= "Orthodox Hospital"; }
if (d2c < distance2){ distance2 = d2c; x2 = x2c; hospital
= "J'itaouy Hospital"; }
    }

// Add both distances to get the total distance travelled
by the ambulance
totaldistance = distance1 + distance2;
char output[ 32 ];
itoa(totaldistance, output, 10);    // cast from int to
string

display = "  2. Dipatch Ambulance from " + center;
OutputListBox->AddString( display.data() );
destination_hospital = hospital; // fill dest hospital in
detination_hotpial to be sent via sms
display = "  3. Direct Ambulance to: " + hospital;
OutputListBox->AddString( display.data() );
display = "  4. Route Chosen ";
OutputListBox->AddString( display.data() );
display = "    " + x2;
route=x2; //fill the route here
OutputListBox->AddString( display.data() );
display = "  5. Total Distance Travelled by Shortest Path
is: " + (string)output + " meters.";
OutputListBox->AddString( display.data() );
//display =
"*****
*****";
//OutputListBox->AddString( display.data() );
display = "";
}
void CEmergencySystemsDlg::OnClear()
{
    CListBox *OutputListBox;
    OutputListBox = ( CListBox * ) GetDlgItem(
IDC_OUTPUTLISTBOX );
    OutputListBox->ResetContent();
}

```

```

void CEmergencySystemsDlg::OnView() // When the ENTER
button is pressed
{
    string condandnum = "";
    string display="";
    char temp[50]; //condition
    char temp1[10]; //number of casualties
    char temp2[10] //ambulance number
    CListBox *Condition; // create a new list box
    Condition = ( CListBox * ) GetDlgItem( IDC_CONDITION
); // pointer (Condition) to listbox
(IDC_CONDITION)
    int listindex = Condition->GetCurSel();
    // gets index of selected listbox item
    if ( listindex == LB_ERR )
// if no item is selected --> exit function
    {
        MessageBox( "Enter the Condition or Select
UNKNOWN", "Known Condition!", MB_ICONWARNING );
        return;
    }

    CListBox *Numb;
    // create a new list box
    Numb = ( CListBox * ) GetDlgItem( IDC_NUMBERS );
    // pointer (Numbers) to listbox (IDC_NUMBER)
    int listindexnum = Numb->GetCurSel();
    // gets index of selected listbox item
    if ( listindexnum == LB_ERR )
    // if no item is selected --> exit function
    {
        MessageBox( "Enter the Number of Known Injuries
or Select UNKNOWN", "Number of Cases!", MB_ICONWARNING );
        return;
    }

    CListBox *Ambulances;
    Ambulances=( CListBox *) GetDlgItem(
IDC_AMBULANCES);
    int listindexnum1 = Ambulances->GetCurSel();
    if (listindexnum1 ==LB_ERR)
    {
        MessageBox("Eneter Ambulance Number to
Dispatch", "Ambulance Number", MB_ICONWARNING);
        return;
    }

    Condition-> GetText(listindex, temp);
    cond = (string) temp;
    //String cond to be sent via Text!
    //collect the number of injured

```

```

Numb-> GetText(listindexnum, temp1);
num =(string) temp1;//String Num to be sent via txt
Ambulances-> GetText(listindexnum1,temp2);
ambulance=(string) temp2;

if (ambulance=="101A")
    ambnumber="00971503292132";
else if(ambulance=="102B")
    ambnumber="03Amulance2";
else if(ambulance=="103C")
    ambnumber="03Ambulance3";
else if(ambulance=="104D")
    ambnumber="03Ambulance4";

condandnum = "\n Case is " + cond + "\n& Number of cases
=" + num + "\n Ambulance Number =" +ambnumber;//for
testing

CListBox *OutputListBox;
OutputListBox=(CListBox *) GetDlgItem(IDC_OUTPUTLISTBOX);
OutputListBox->AddString(condandnum.data());
display =
"*****";
OutputListBox->AddString( display.data() );
}

void CEmergencySystemsDlg::OnAbout() //Labelled as Send
SMS
{

string finalSMS="";
string street = targetstreet; //gets the targetstreet
from OnSelect()
string temp = destination_hospital; //gets the hospital
name from OnSelect()
string temp2 = route; //gets the route from OnSelect()
string temp3 =cond; //gets the condition status from
onView()
string temp4 =num; //gets the number of injured from
onView()

// Prepare the SMS content
finalSMS = "Dispatch to: "+street+ " street. \n Hospital
is: " + temp + ". \n Route is: " + temp2 + ". \n
Condition(s): " + temp3 + ". \n Known Number is: " +
temp4;
MessageBox (finalSMS.data(),"Text Content");

```

```

//*****REPORTING
CODE*****//
forreporting = forreporting + "\n" +
"*****\n" +
finalSMS; //Collect Data for Reporting
ofstream myfile; //define a file for i/o
myfile.open("report.txt");
myfile<<
"*****\n";
myfile<< "***** MOBIMEDIC - REPORT
*****\n";
myfile<<
"*****\n";
myfile<< "\n" + forreporting + "\n";
myfile<<
"*****\n";
/*****
*****
***** SMS sending code: Contacts my local
provider *****
***** and sends my text dynamically to the
mobile *****
***** I select
*****/
string s="";
std::string str =
"http://80.77.187.4/yoofi_int/SMSyoofi/pushpull/push_sms.
asp?receiver="+ambnumber+"&smstext="+finalSMS+"&operator=
etisalat&country=uae&smstype=text&senderID=2202&username=
earnstyoung&password=earnstyoungpass";
char* psz = strdup(str.c_str());
file= psz;
ShellExecute (NULL, NULL, file, NULL, NULL, NULL);

//*****
}
void CEmergencySystemsDlg::OnReportButton() //Report
Opened for Viewing
{
ShellExecute(m_hWnd, "open", "report.txt", NULL,
workingdir, SW_SHOWNORMAL);

}
void CEmergencySystemsDlg::OnHelpButton() // Help file
opened
{

ShellExecute(m_hWnd, "open", "help.doc", NULL,
workingdir, SW_SHOWNORMAL);

}

```

```

//*****
*****

```

```

void CEmergencySystemsDlg::OnStnClickedMap()
{
    ShellExecute(m_hWnd, "open", "Help.txt", NULL,
        workingdir, SW_SHOWNORMAL);
}

```

Graph.cpp

```

#include "Weight.cpp"

```

```

const int    INTERSECTION= -1,
              REDCROSS    = -2,
              HOSPITAL    = -3,
              VISITED      = -4,
              UNVISITED    = -5,
              INFINITY     = 100000;

```

```

class Graph: public Weight    // class Graph inherits from super
class Weight

```

```

{
    private: int numVert, numEdge;
             int *v, *d, *parent, *shortest;
             string *roads;
             Weight w;
             Weight **matrix;

    public:   // Constructor
             Graph(int numVertex)
             {
                 int i, j;
                 numVert = numVertex;
                 numEdge = 0;
                 v= new int[numVert];
                 //initializing vertex array
                 d= new int[numVert];
                 parent= new int[numVert];
                 shortest= new int[numVert];
                 roads = new string[numVert];

                 for(i=0; i<numVert; i++)
                 {
                     v[i] = 0;           //unvisited
                     roads[i] = "";
                 }

                 matrix = (Weight**)new Weight*[numVert];
                 for(i=0; i<numVert; i++)
                     matrix[i] = new Weight[numVert];
                 for(i=0; i<numVert; i++)
                     for(j=0; j<numVert; j++)
                     {
                         w.graph = "";
                         w.distance = 0;

```



```

        w.color = 0;
        w.street = "";
        matrix[i][j] = w;
    }
    //initial weight is 0
}

// Destructor
Graph()
{
    delete []v;
    for(int i=0; i<numVert; i++)
        delete [] matrix[i];
    delete [] matrix;
}

int vertex_num(){ return numVert;}
int edge_num()   { return numEdge;}
int first_neighbor(int vertex)
// returns vetex's first neighbor
{
    int neighbor;
    for(neighbor=0;neighbor<numVert;neighbor++)

    if(matrix[vertex][neighbor].getDistance()!=0)
        return neighbor;
    return neighbor;
}

int next(int v1, int v2)
//gets v1's neighbor after v2
{
    int i;
    for(i=v2+1;i<numVert;i++)
        if(matrix[v1][i].getDistance()!=0)
            return i;
    return i; // return -1 if there are no neighbors
}

void addWeight(string gr, int v1, int v2, int dist,int col, string
st)
{
    if (matrix[v1][v2].getDistance()==0)
        numEdge++;
    matrix[v1][v2].setGraph(gr);
    matrix[v1][v2].setDistance(dist);
    matrix[v1][v2].setColor(col);
    matrix[v1][v2].setStreet(st);
}

void delEdge(int v1, int v2)
{
    if (matrix[v1][v2].getDistance()!=0)
        numEdge--;
    matrix[v1][v2].setDistance(0);
}

Weight getWeight(int v1, int v2){return matrix[v1][v2];}

string getCol(int v1,int v2)
{

```

```

switch(((matrix[v1][v2].getColor())*(-1))-1)
{
    case 0: return "INTERSECTION";
    case 1: return "REDCROSS";
    case 2: return "HOSPITAL";

}
}

string getGraphName(string s) //Returns the Name of the Graph -- i.e
The zone
{
    for(int i=0;i<numVert;i++)
        for(int j=0;j<numVert;j++)
            if (
(matrix[i][j].getStreet().compare(s))==0 )
                return matrix[i][j].getGraph();
            return "notexist";
}

int getMark(int vertex) {return v[vertex];}
void setMark(int vertex, int val){v[vertex]=val;}
int getHead(string s)
{
    for(int i=0;i<numVert;i++)
        for(int j=0;j<numVert;j++)
            if((matrix[i][j].getStreet().compare(s))==0)
                return i;
}
int getTail(string s)
{
    for(int i=0;i<numVert;i++)
        for(int j=0;j<numVert;j++)
            if((matrix[i][j].getStreet().compare(s))==0)
                return j;
    return 0;
}

string returnStreet(int v1, int v2) { return
(matrix[v1][v2].getStreet()); }

int getDist(int v1,int v2){ return
(matrix[v1][v2].getDistance()); }

bool empty(Graph G)
{
    int limit=G.vertex_num();
    int count=0;

    for(int i=0;i<limit;i++)
        if(G.d[i]==-10)
            count++;
    if(count==limit)
        return true;
    return false;
}

int returnMin(Graph G)
{
    int limit= G.vertex_num();
    int min=100000;
    int i;

```

```

        for(int i=0;i<limit;i++)
            if(G.d[i]<min && G.d[i]>=0)
            {
                d[i]=-10;
                return i;
                break;
            }
        return i;
    }

void Initialize_Single_Source(Graph G, int start)
{
    int v;
    int limit=G.vertex_num();

    for(v=0; v<limit;v++)
    {
        G.d[v] = INFINITY;
        G.shortest[v] = INFINITY;
        G.parent[v] = NULL;
    }
    G.d[start] = 0;
    G.shortest[start] = 0;
}

void Relax(Graph G, int u, int v)
{
    if(G.d[v]> (G.shortest[u]+ G.getDist(u,v)))
    {
        G.d[v]= G.shortest[u]+ G.getDist(u,v);
        G.parent[v]=u;
        G.roads[v] = G.roads[u] + " \n " +
        G.returnStreet(u,v);
        G.shortest[v]=G.d[v];
    }
}

int Dijkstra(Graph G, int start,int end)
{
    int u,w;
    int limit=G.vertex_num();
    Initialize_Single_Source(G, start);
    while(!G.empty(G))
    {
        u = G.returnMin(G);
        for( w=G.first_neighbor(u); w<G.vertex_num();
        w=G.next(u,w) );
        Relax(G,u,w);
    }
    return shortest[end];
}

string returnRoads(Graph G, int end) { return G.roads[end]; }
};

```

Weight.cpp

```

#include <string>
#include <cstring>
#include <stdlib.h>
#include <stdio.h>

using namespace std;
using std::string;

class Weight
{
    public:      int distance, color;
                string graph, street;

    public:      // Default Constructor
                Weight()
                {
                    graph = "";
                    distance = 0;
                    color = 0;
                    street = "";
                }

                // User-defined Constructor
                Weight(string g, int d, int c, string s)
                {
                    graph = g;
                    distance = d;
                    color = c;
                    street = s;
                }

                // Reader Functions
                string getGraph()           { return graph; }
                int getDistance()           { return distance; }
                int getColor()              { return color; }

                string getStreet()          { return
street.data(); }

                // Writer Functions
                void setGraph(string g)     { graph = g; }
                void setDistance(int d)     { distance = d; }
                void setColor(int c)        { color = c; }
                void setStreet(string s)    { street = s; }
};

```

References

- [1] ESRI. (2009). The GIS Software Leader from:
<http://www.esri.com/>
- [2] ThinkGeo. (2009). GIS Components for .NET Developers, GPS Tracking Software and GIS Services from <http://thinkgeo.com/>
- [3] MapInfo Professional User Guide (Version 9.5). (2008). NewYork.
- [4] About Vodafone. (2009). How do mobiles work from
http://www.vodafone.com/start/about_vodafone/what_we_do/technology/how_do_mobiles_work.html
- [5] Visual Studio 2005. (2009). Visual Studio Developer Center from
<http://msdn.microsoft.com/en-us/library/ms950416.aspx>
- [6] Microsoft Foundation Class Tutorial – Step by Step Guide. (2006). From
http://depts.washington.edu/cmmr/biga/chapter_tutorials/mfc_tutorial/index.html
- [7] MFC Library Reference, CDialog Class. (2009). from
[http://msdn.microsoft.com/en-us/library/132s802t\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/132s802t(VS.71).aspx)
-
- [8] Deitel, H.M., Deitel, P.J., Nieto, T., & Strassberger, E.,(2000). Getting Started with Microsoft Visual C++ 6 with an Introduction to MFC (2nd ed). NewJersey: Prentice-Hall
- [9] ActiveXperts Software B.V. (2009). ActiveXperts SMS Messaging Server from:
<http://www.activexperts.com/xmstoolkit/>
- [10] SMS&Co. (2008). SMSLibX - SMS ActiveX to send SMS and receive SMS from PC from: <http://www.smsco.it/tomcat/en/sms/smslibx.jsp>

- [11] ACTEL. (2007). from <http://actelme.com>
- [12] Lee, C., Epelman, M., Whiteiii, C., & Bozer, Y. (2006). A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research Part B: Methodological* , 40 (4), 265-284.
- [13] Machado, P., Tavares, J., Pereira, F. B., & Costa, E. (2002). Vehicle routing problem: Doing it the evolutionary way. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference . New York: Morgan Kaufmann Publishers.
- [14] Travelling Salesman Problem. (2008). The Travelling Salesman Problem from: <http://www.tsp.gatech.edu/>
- [15] S. El-Masri. (2005). Mobile comprehensive emergency system using mobile web services. In the Second International Conference on Innovations in Information Technology (IIT'05).
- [16] Metrooplitan Ambulance Service. (2002). The Metropolitan Ambulance Service from <http://www.ambulancevic.com.au/opservices/communications.html>
- [17] Priority Solutions. (2007). PSIAM Software, Clinical Triage & Emergency Dispatch Software from http://www.prioritysolutionsinc.com/psiam_software
- [18] Intergraph. (2009). Intergraph SG&I Products from <http://www.intergraph.com/sgi/products/product-family.aspx?family=3>
- [19] Tiburon. (2009). Tiburon, Inc. – Emergency Dispatch Software” from <http://www.tibinc.com/emergency-dispatch-software.html>
- [20] MSDN: Microsoft Developer Network. (2009). Visual C++ Tutorials, Library, and More on MSDN from <http://msdn.microsoft.com/en-us/visualc/default.aspx>
- [21] Map of Lebanon. (1998). GEOprojects, United Kingdom.

[22] Red Cross Flag for the main application from:
<http://img.wonkette.com/politics/Red%20Cross%20flag.jpg>

[23] Deitel, H.M., & Deitel, P.J., (1998). C++ How to Program (2nd ed.). NewJersey: Prentice-Hall.

[24] Clinical Solutions. (2008). Healthcare solutions, clinical software and IT from www.csdss.com/solutions/index.aspx