GSM NETWORK MONITORING THROUGH MOBILE PHONES

Awais Zia Khan
(2003-NUST-BIT-06)

Irfan Ahmad
(2003-NUST-BIT-11)

A project submitted in partial fulfillment of the requirements for the degree of

Bachelors of Information Technology

NUST Institute of Information Technology
National University of Sciences & Technology
Rawalpindi, Pakistan

(2007)

CERTIFICATE

Certified that the contents and form of this project entitled "**GSM Network Monitoring Through Mobile Phones**" submitted by Mr. Awais Zia Khan and Mr. Irfan Ahmad have been found satisfactory for the requirements of the degree.

Advisor: _____

(Mr. Ali Sajjad)

Co-Advisor: _____

(Mr. Saad Liaquat)

Committee Member: _____

(Dr. Aamir Shafi)

Committee Member: _____

(Mr. Abdul Ghafoor)

DEDICATED TO

In the name of Allah, the Most Gracious, the Most Merciful

To our dear
Family especially to our Parents

ACKNOWLEDGEMENTS

First of all, we are thankful to Almighty Allah for the successful completion of the project. We are also thankful to our family members who supported me not only during this project but also throughout the course of the degree. We can not express my gratitude in words for what they have done for us.

We are especially thankful to my project advisor Mr. Saad Liaquat for his guidance, support and instructive supervision throughout the project. We are particularly grateful to him for giving us time for this project despite the fact that he was extremely busy at the later stages of the project. We also thank him for letting us use his personal mobile phone for testing purposes.

We would also like to thank our project co-advisor Mr. Ali Sajjad who in spite of his busy schedule, always provided support and help at every stage of the project and also provided valuable suggestions for the improvement of the project.

We would like to express our gratitude to Mr.Abdul Ghafoor and Dr. Aamir Shafi for their kind suggestions and advises that helped us a lot during this project.

We are full of gratitude for our friend and class-fellow Hassan Muhammad Khan for allowing us to use his mobile phone for the purpose of testing and final demonstration. We will always cherish all the moments that we spent together with our friends Hassan Mohammad Khan and Sajjad Rizvi at NIIT.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Actual Word
AuC	Authentication Register
BSC	Base Station Controller
BSS	Base Station Subsystem
BTS	Base Transceiver Subsystem
EIR	Equipment Identity Register
HLR	Home Location Register
IMSI	International Mobile Subscriber Identity
MS	Mobile Station
MSC	Mobile Switching Center
NSS	Network Subsystem
OMC	Operations Management Center
SIM	Subscriber Identity Module
VLR	Visitor Location Register

ABSTRACT

The performance of the GSM network is a matter of concern for GSM network operators as well as GSM network subscribers. Both the concerned parties need to know the behavior of GSM network at any given time. This project focuses on developing an application that will monitor and analyze the performance of GSM network by retrieving information from the GSM network on the mobile phones in real time. It will consist of a phone-end application as well a desktop application. The phone-end application will use the GSM radio in the phone to retrieve different information from the GSM network provider that are categorized in different categories such as network information, signal information, battery information, call information and phone information. The important parameters that will analyze the quality of the network include determining signal strength, signal quality, signal-to-noise ratio, visible base stations, inter-cell and intra-cell handover etc during calls as well as in idle state. Logs will be maintained at the phone-end application and these logs will then be transferred to the desktop application. Desktop application will display all the retrieved information. Furthermore, graphs of signal strength, signal-to-noise ratio and signal quality with respect to time will also be maintained on the desktop-side application.

The phone-end application is developed in Symbian OS using C++ while Visual Studio .NET 2005 and Dundas software are used for developing desktop-side application. Log file from the phone-end application is transferred to the desktop-side application using Nokia PC Suite 6.83.

Chapter 1

INTRODUCTION

This chapter gives a brief introduction of the project. The problem statement of the project is defined and the objectives of the project are also outlined. The breakdown of this report is given at the end of this chapter.

1.1. PROBLEM STATEMENT

The problem statement of the project is as follows:

"To develop an application that will monitor and analyze GSM network using mobile phones"

1.2. PROJECT OBJECTIVES

The objective of the project is to develop a system that will enable end-users to determine and monitor the behavior and performance of the GSM network on their mobile phones. The GSM network monitoring system will consist of phone-end application as well as desktop-side application. The phone-end application will allow the interested GSM network subscribers to monitor the GSM network using different quality parameters which are divided into five broad categories of phone information, network information, signal information, call information and battery information. The desktop side application will give GSM network operators option to monitor the performance of their network by maintaining the history and information logs obtained from the mobile side application and analyzing this data in the form of charts and graphs for analyzing the trends in the performance of GSM network as well as identifying current and potential problems. Phone-end application and desktop-side application would be provided an extremely friendly

user interface and would be very easy to use for individual users as well as network operators to use the application.

1.3. DOCUMENT BREAKDOWN

The organization of the remaining chapters is as follows. *Chapter 2* gives a brief overview of the related work that has been done in this field. *Chapter 3* focuses on the literature review of the areas that are of concern for this project. GSM network and Symbian Operating System are mainly covered in this chapter. *Chapter 4* describes the project in detail. The project is divided into two main modules and description of the two modules is given in detail and complete functionality and working of both modules is also provided. *Chapter 5* highlights the major tools and technologies that have been used in the development of the application. *Chapter 6* is related to the architecture of the project. Class diagrams and the database design of the project are also given in this section. *Chapter 7* shows the complete project timeline. *Chapter 8* is the last chapter of the document which describes the conclusion of this project and gives the future direction for enhancements and additional features of GSM network monitoring system. The references for consulted study material are given at the end of this document.

Chapter 2

SURVEY OF RELATED WORK

This chapter of the project report focuses on describing the similar and related work done for analyzing and monitoring GSM network. All the related work has been critically analyzed.

2.1. METHODOLOGY FOR SURVEY

In order to find the related work that has been done so far, functionality driven search was performed. Only those tools or applications were studied that performed almost the same kind of operation as our project does. After thorough research and survey, only one such software was found.

2.2. RELATED WORK

TEMS CellSight is a network performance management software developed by Ericsson. This software manages large volumes of switch data to allow users to monitor network performance and generate statistical reports. TEMS is basically used to collect and monitor valuable network performance indicators, identify current and potential performance problems, investigate problem causes and forecast network growth. This software is however very expensive and is available to the network operators at a discounted rate of $15,000. Therefore, it is not affordable for individual GSM mobile phone users. Another limitation of TEMS is that it is developed by Ericsson and network operators need to use Ericsson phones in order to measure their network's performance. However, since Nokia has a wider market penetration and Nokia phones are widely available, need for developing a

system using Nokia phones has been evolved. The proposed solution in this project is for Nokia mobile phones which can also be used by individual users in addition to network operators.

Chapter 3

BACKGROUND TUTORIAL

In this section, background tutorial of GSM network and Symbian OS is given. This chapter discusses the basic terminologies, gives basic introduction to GSM-Networks, explains GSM-Network architecture and discusses Symbian operating system. Understanding all the above topics is necessary before the development of the application. The above mentioned topics are extremely diverse in nature and only those points are highlighted which are of importance for carrying this project.

3.1. INTRODUCTION TO GSM-NETWORKS

Cellular is one of the fastest growing and most demanding telecommunications applications. The concept of cellular service is the use of variable low-power transmitters where frequencies can be reused within a geographic area called cell. Cells can be sized according to the subscriber density and the demand of a given area. Cells are assigned a group of channels that is completely different from neighboring cells [1]. However, frequencies used in one cell can be re-used in non-neighboring cells in order to carry out more than one conversation at a time without any interruption and interference. After the evolution of cellular telecommunications, a standard with the name of GSM (Global System for Mobile Communication) was formed with the basic purpose of addressing the compatibility issues. The technical options adopted by GSM are digital transmission to ensure increased capacity and security, time division multiplexing for radio channels and encryption of radio channel transmission.

3.2. GSM-NETWORK ARCHITECTURE

As with all the systems in telecommunication area, GSM comes with a hierarchical architecture comprising many entities, interfaces and acronyms. A GSM system consists of following three basic subsystems

- Radio Subsystem (RSS)
- Network Switching Subsystem (NSS)
- Operations Subsystem (OSS)

The architecture of GSM network is shown in Fig 1.

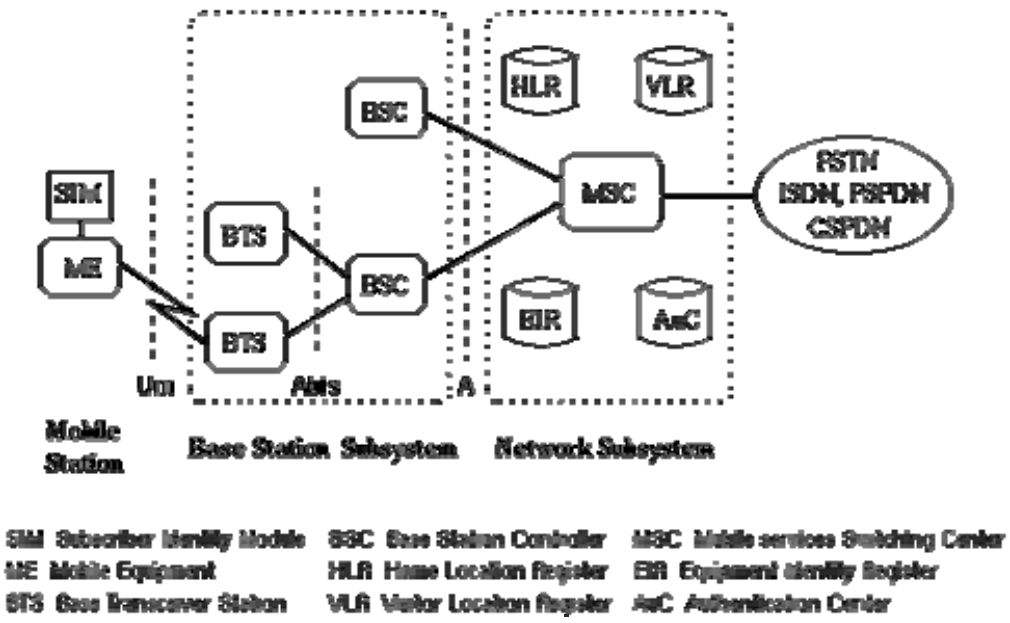


Figure 1: GSM-Network Architecture

3.2.1. Radio Subsystem (RSS)

The Radio Sub System comprises all the radio specific entities, i.e., Mobile Station (MS) or Mobile Host (MH) and Base Station Subsystem (BSS). The interconnection between RSS and NSS is via A-interface.

3.2.1.1. Base station subsystem (BSS)

A GSM network comprises many BSSs, each controlled by a Base Station Controller (BSC). The BSS performs all functions necessary to maintain radio connection to an MH, coding/decoding of voice, and rate adaptation to/from the wireless network part. Besides BSC, the BSS contains several BTSs.

3.2.1.2. Base transceiver station (BTS)

A BTS comprises of all radio equipment, i.e., antennas, signal processing, amplifiers necessary for radio transmission The BTS houses the radio transceivers that define a cell and handles the radio (Um) interface protocols with the mobile station.

3.2.1.3. Base station controller (BSC)

The Base Station Controller (BSC) manages the radio resources for one or more BTSs, across the Abis interface. It manages the radio interface channels (setup, teardown, frequency hopping, etc.) as well as handovers.

3.2.1.4. Mobile host (MH)

The MH comprises all user equipment and software needed for communication with a GSM network. An MH consists of independent hardware and software and one of Subscriber Identity Module (SIM), which stores the user specific data, while an MH can be identified via the International Mobile Equipment Identity (IMEI).

The SIM card contains many identifiers in tables such as

- Personal Identity Number (PIN)
- PIN Unblocking Key (PUK)
- Authentication Key
- International Mobile Subscriber Identity (IMSI)

3.2.2. Network and Switching Subsystem (NSS)

The NSS connects the wireless network with standard public networks, performs handovers between different BSSs, comprises function for worldwide localization of users and supports charging, accounting, and roaming of users between different providers in different countries. The NSS consists of following switches and databases.

3.2.2.1. Mobile switching center (MSC)

The central component of the Network Subsystem is the Mobile services Switching Center (MSC). It acts like a normal switching node of the PSTN or ISDN, and in addition provides all the functionality needed to handle a mobile subscriber, including registration, authentication, location updating, inter-MSC handovers, and call routing to a roaming subscriber.

3.2.2.2. Home location register (HLR)

The Home Location Register (HLR) contains all the administrative information of each subscriber registered in the corresponding GSM network, along with the current location of the subscriber.

3.2.2.3. Visitor location register (VLR)

The Visitor Location Register contains selected administrative information from the HLR, necessary for call control and provision of the subscribed services, for each mobile currently located in the geographical area controlled by the VLR.

3.2.3. Operations Subsystem (OSS)

The third part of GSM system, the Operations Subsystem (OSS), contains all functions necessary of network operations and maintenance.

3.2.3.1. Operations and maintenance center (OMC)

The OMC monitors and controls all network entities via O-interface (SS7 with X.25). Typical OMC management functions are traffic monitoring, status reports of network entities, subscriber and security management. OMCs use the concept of Telecommunication Management Network (TMN) as standardize by (ITU-T).

3.2.3.2. Authentication center (AuC)

The Authentication Center (AuC) is a protected database that stores a copy of the secret key stored in each subscriber's SIM card, used for authentication and ciphering on the radio channel.

3.2.3.3. Equipment identity register (EIR)

The Equipment Identity Register (EIR) is a database that contains a list of all valid mobile equipment on the network, where the mobile equipment is identified by its International Mobile Equipment Identity (IMEI).

3.3. SYMBIAN OPERATING SYSTEM

Symbian OS is an operating system, designed for mobile devices, with associated libraries, user interface frameworks and reference implementations of

common tools. The fact that Symbian OS is built for handheld devices that have limited resources, strong emphasis is laid on keeping memory usage low. Symbian OS has a microkernel architecture, which means that the minimum necessary is within the kernel. It contains a scheduler and memory management, but no networking or file system support. These things are provided by user-side servers. Nokia provides different SDKs for programming on Symbian OS. Each version of the SDK depends on the selected Symbian platform and the selected device on which the programming needs to be done.

3.3.1. Application Structure of Symbian OS GUI Applications

In this project, graphical user interface (GUI) needs to be developed using Symbian OS on the mobile phone. So, in order to understand the application structure of Symbian OS GUI application, the hierarchy of classes given in Fig 2 needs to be considered.

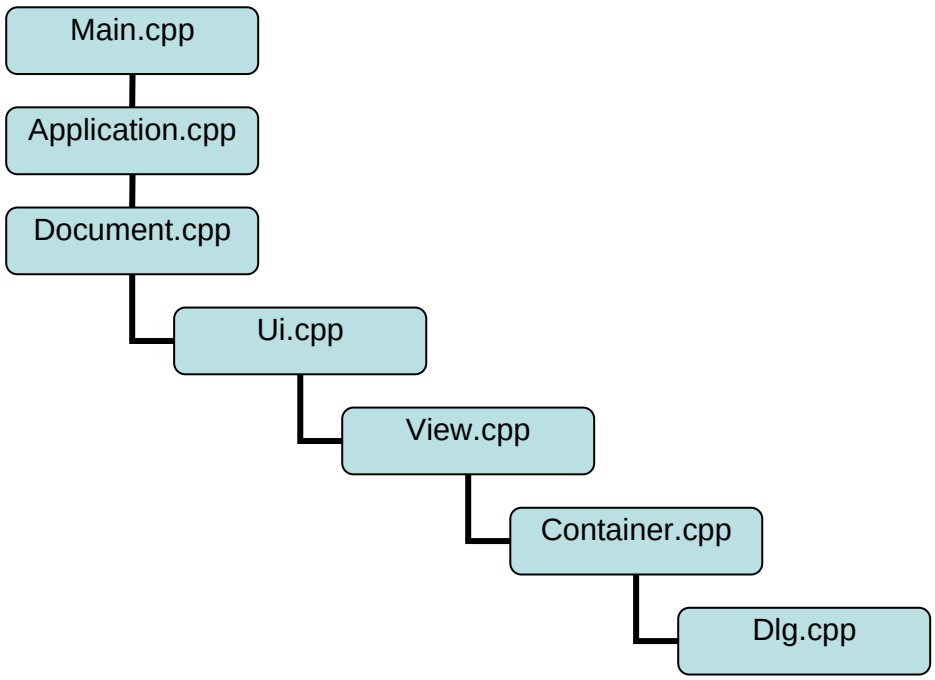


Figure 2: Application Structure of Symbian OS GUI Application

3.3.1.1. Application class

Application class is used to define the properties of the application and at the same time creates a new blank document.

3.3.1.2. Document class

Document class is responsible for storing and restoring the application's data and also creates an application user interface.

3.3.1.3. UI class

An application UI is entirely invisible. This class handles drawing and screen-based interactions.

3.3.1.4. View class

View class actually allows displaying data on the screen and interaction with it.

3.3.1.5. Container class

Container class allows creating a container to insert data in it. It could be a dialog, panel etc.

3.3.1.6. Dlg class

Dlg class creates a dialog in which forms are appended. It is actually in Dlg class that different components like textbox, textfieds etc are added.

Chapter 4

PROJECT DESCRIPTION

In order to analyze the GSM network using mobile phones, two applications are needed. One application is developed on the mobile phone for the purpose of retrievg information from the GSM network while the second application is developed for desktop side for analyzing the trends in the performance of GSM network. The project architecture is given below:

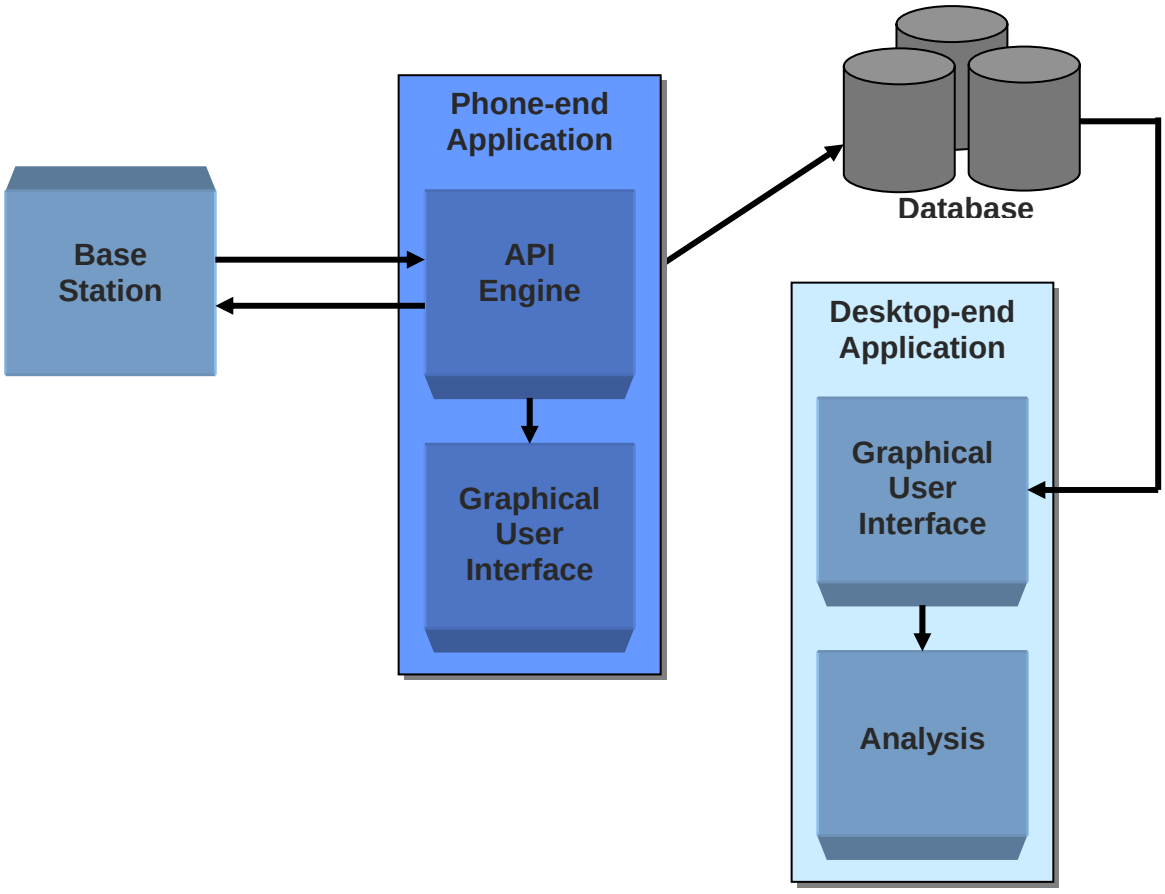


Figure 3: Project Architecture

4.1. PHONE-END APPLICATION

The phone-end application will allow the interested GSM network subscribers to monitor the GSM network using different quality parameters which are divided into five broad categories of phone information, network information, signal information, call information and battery information. Phone-end application can be used by individual GSM users as well as GSM network operators. The application will use the GSM radio in the phone to access the GSM network and retrieve the information. All the information will be displayed on a user friendly graphical user interface, with a different tab page for different categories of information. The user of this application will select the “Fetch” option to fetch values from the network. Once this application is started, the values in the tab pages will be automatically updated in the fields if there is any change in the value of any category. This retrieved information will also be stored in a log file on the mobile phone and it will be actually this log file that will be transferred to the desktop-side information. The log file will be transferred to the desktop side automatically after every 10 seconds using Nokia PC Suite if data cable is attached with the phone. All the categories of information are discussed in detail below:

4.1.1. Phone Information

In phone information category, three parameters will be retrieved, namely IMIE number, phone manufacturer name and phone model.

IMIE number is unique for every mobile hand-set. IMIE is fetched to uniquely identify the mobile on which the application will run. In our application, IMIE number will not change if the mobile hand-set remains same.

Phone manufacturer name returns the name of manufacturer of the phone and this information will also not change if the phone of same manufacturer is used.

Phone model returns the model number of the phone and this information will also not change if the application is run on one phone.

4.1.2. Network Information

In network information category, general information regarding network is retrieved which includes registration status, network mode, mobile country code, cell id, location area code, connected tower and network operator name.

Registration status indicates the service that the current network provides to the phone. It indicates statuses such as No Service, Emergency Only, Busy, Roaming etc.

Network mode describes the technology that the network uses. This includes GSM, CDMA 2000, and WCDMA etc.

Mobile Country Code (MCC) is a 3-digit code that indicates the country that the network is in. It is returned in a descriptor. Since this application will be run only in Pakistan, this value will remain same.

Cell id returns the id of the cell in which the user is currently in.

Network operator name returns the name of the GSM operator. This value will also remain same if the same SIM card is used on the phone.

4.1.3. Signal Information

The parameters selected for signal information include signal detectable, signal strength, number of signal bars, signal quality and signal-to-noise ratio.

Signal detectable is used to check if signals from the GSM network are detectable or not. This value can change depending on whether signal is appearing on mobile phone or not.

Signal strength returns the strength of the signal in dBm. This value can vary from time to time.

Number of signal bars represents the total number of signal bars that appear on the phone at anytime. This value can also vary from time to time, depending on the performance of the network.

Signal quality returns quality of signals on the phone at any time. Higher percentage of signal quality represents higher signal quality. It is measured as:

Signal Quality = (No. of Signal Bars*80)/7 + (Signal Strength * 20)/110

Signal-to-noise ratio compares the level of a desired signal to the level of background noise. The higher the ratio, the less obtrusive the background noise is. The formula for calculating signal-to-noise ratio is:

Signal-to-Noise Ratio = 10 log (Signal Strength/Noise)

4.1.4. Call Information

In call information category, general information related to call status is retrieved in the application. This includes call-in-progress indicator, call status, date of call, current time, direction of the call and ciphering status.

Call-in-progress indicator determines whether a call is in progress or not.

Call status returns the status of line which could be idle, dialing, ringing, answering, connecting, connected, reconnect pending, disconnecting, hold or transferring.

Date of call returns the date on which the call is made or received.

Direction of the call returns whether the call is incoming or outgoing.

Ciphering status could be GSM, WCDMA or none.

4.1.5. Battery Information

Battery status and battery level are the two parameters that are retrieved in battery information category for this application.

Battery status represents the current status of battery. It could be unknown status, battery active, external power, no battery or power fault.

Battery level returns the current percentage of remaining battery power.

4.2. DESKTOP-SIDE APPLICATION

The desktop side application will give GSM network operators option to monitor the performance of their network by maintaining the history and information logs obtained from the mobile side application and analyzing this data in the form of graphs for analyzing the trends in the performance of GSM network as well as identifying current and potential problems.

Since the desktop-side application is meant for network operators, only authorized users can access this application; so the user will have to log in when this application is launched. The desktop-side application has a user friendly graphical user interface which consists of different sections to display different categories of information. When the data cable that is connected to the mobile phone which is running phone-end application, is attached with the desktop computer, the log file containing retrieved GSM information automatically transfers to the desktop side. The log file is automatically transferred to the desktop side after every 10 seconds. On the desktop side, this information is sorted and

directly stored in the tables of a database. The database consists of different tables for holding data of different categories of information. The desktop-side application fetches data from this database and displays on the graphical user interface. The desktop-side application displays updated information whenever the information of any category changes on the mobile phone. Logs of all the categories of information are maintained on the desktop side and can be accessed from the tool bar of desktop-side information.

Special space is reserved for displaying graphs on the desktop-side application. Graphs of signal strength with respect to time, signal quality with respect to time and signal-to-noise ratio with respect to time are created. These graphs are dynamically updated whenever there is a change in signal strength, signal quality or signal-to-noise ratio. These graphs give a clear idea of how the signal strength or signal quality of the network is varying with respect to time and hence one can deduce whether the performance of the network is satisfactory or unsatisfactory.

4.3. DEVELOPMENT CONSTRAINTS

All the information that has been discussed in the above section has been retrieved in real time with the exception of network information. Symbian has put a restriction on accessing information related registration status, network mode, mobile country code (MCC), cell id, location area code, name of connected tower and network operator name; which fall under the category of network information.

Symbian Ltd. grants access to information to the users according to the different level of capabilities that it has defined. Most of the information that needs to be retrieved comes under ReadUserData and WriteUserData capabilities. These

capabilities can be accessed without any restriction. However, information related to GSM network comes under ReadDeviceData and WriteDeviceData capabilities. These two capabilities are restricted. To access information from these two capabilities, a VeriSign ACS Publisher ID and Symbian Signed account is needed. For acquiring VeriSign ACS Publisher ID and Symbian Signed account, payment of $350 needs to be made to Symbain Ltd., after which a key will be delivered to the user. This key will generate a certificate which will allow testing of the application and hence retrieval of network values. Since this key can not be purchased owing to its high price, the values for network information in the application are hard-coded. However, complete source code is provided for retrieving network information which can be tested if the key is purchased from Symbain Ltd.

Chapter 5

TOOLS AND TECHNOLOGIES

This chapter focuses on different tools and technologies that have been used in the development of GSM information monitoring system. It provides a brief overview of each tool and also highlight why that tool is being used.

5.1. SYMBIAN OPERATING SYSTEM

Symbian OS is an advanced, open operating system licensed by the world's leading mobile phone manufacturers for mobile devices with associated libraries, user interface frameworks and reference implementations of common tools. It is designed for the specific requirements of advanced 2.5G and 3G mobile phones. Symbian OS combines the power of an integrated applications environment with mobile telephony, bringing advanced data services to the mass market. The fact that Symbian OS is built for handheld devices that have limited resources, strong emphasis is laid on keeping memory usage low. Symbian OS has a microkernel architecture, which means that the minimum necessary is within the kernel. It contains a scheduler and memory management, but no networking or file system support. These things are provided by user-side servers.

Different versions of Symbian OS have been introduced in the market with extras features added in every latest version. The choice of Symbian OS version in this project was of immense importance. The fact that different versions of Symbian OS have different capabilities and functionalities meant that we had to study the APIs of all the latest versions to decide that which version provides the

functionalities that are needed in this project and at the same time also consider the mobile phones that were available for us.

5.1.1. Symbian OS v7.0

Amongst the different versions of Symbian OS that we studied, Symbian OS v7.0 was the first. Third Party Telephony API was the only API that is of our concern for this project. This version provided high level control of telephone data calls but it did not provide the necessary features for retrieving information by accessing the network.

5.1.2. Symbian OS v8.1a and v8.1b

Symbian OS v8.1a and v8.1b allowed the access to network for the purpose of information retrieval but still it did not provide all the values that would be needed to efficiently monitor the GSM network. In addition to this, mobile device compatible to these Symbian OS versions were not available.

5.1.3. Symbian OS v9.1

Symbian OS v9.1 is one of the latest versions released by Symbian. This version provided high level control of telephone data calls and it addition provided support for retrieving information regarding the network, signals as well as calls. The features that it provides are needed for this project and it was our popular choice for the project. Fortunately, mobile device compatible to this version of Symbian OS i.e. Nokia N80 was also available. Therefore, Symbian OS v9.1 was finally selected for this project.

5.2. NOKIA PC SUITE

Nokia PC Suite is used to edit, store, and synchronize Nokia mobile phone data with a Microsoft Windows based PC system. Therefore, in this project, Nokia PC Suite 6.83 is used for the purpose of transferring data retrieved on the phone-end application to the desktop-side application. The source code for transferring data from mobile desktop is written using Nokia PC Suite API 1.1.

5.3. VISUAL STUDIO .NET 2005

Visual Studio .NET 2005 has been as the technology for developing the desktop-side application. Graphical user interface will be developed using C# programming language in Visual Studio .NET 2005. Database queries for reading values from log file and putting into database tables and database queries for reading values from database and displaying in the GUI are written in ADO.NET.

5.4. DUNDAS

Dundas Chart for Windows Form Professional Edition is used for creating dynamic charts on the desktop-side application. The control in this software has been created using C#, which means that it is a fully-managed .NET component, and it has been specifically designed for use with Microsoft's Visual Studio .NET.

Chapter 6

DETAILED DESIGN

The detailed design specifications are explained in this chapter. Class diagrams of phone-end application, file transfer application and desktop-side application are given in the first section of this chapter. In the next section, use case diagrams are provided. The last section of this chapter discusses the database design of desktop application.

6.1. CLASS DIAGRAMS

Class diagrams of desktop-side application, file transfer application and phone-end application are given below:

6.1.1. Class Diagram of Desktop-Side Application

The class diagram of Desktop-Side application is given in Figure 4.

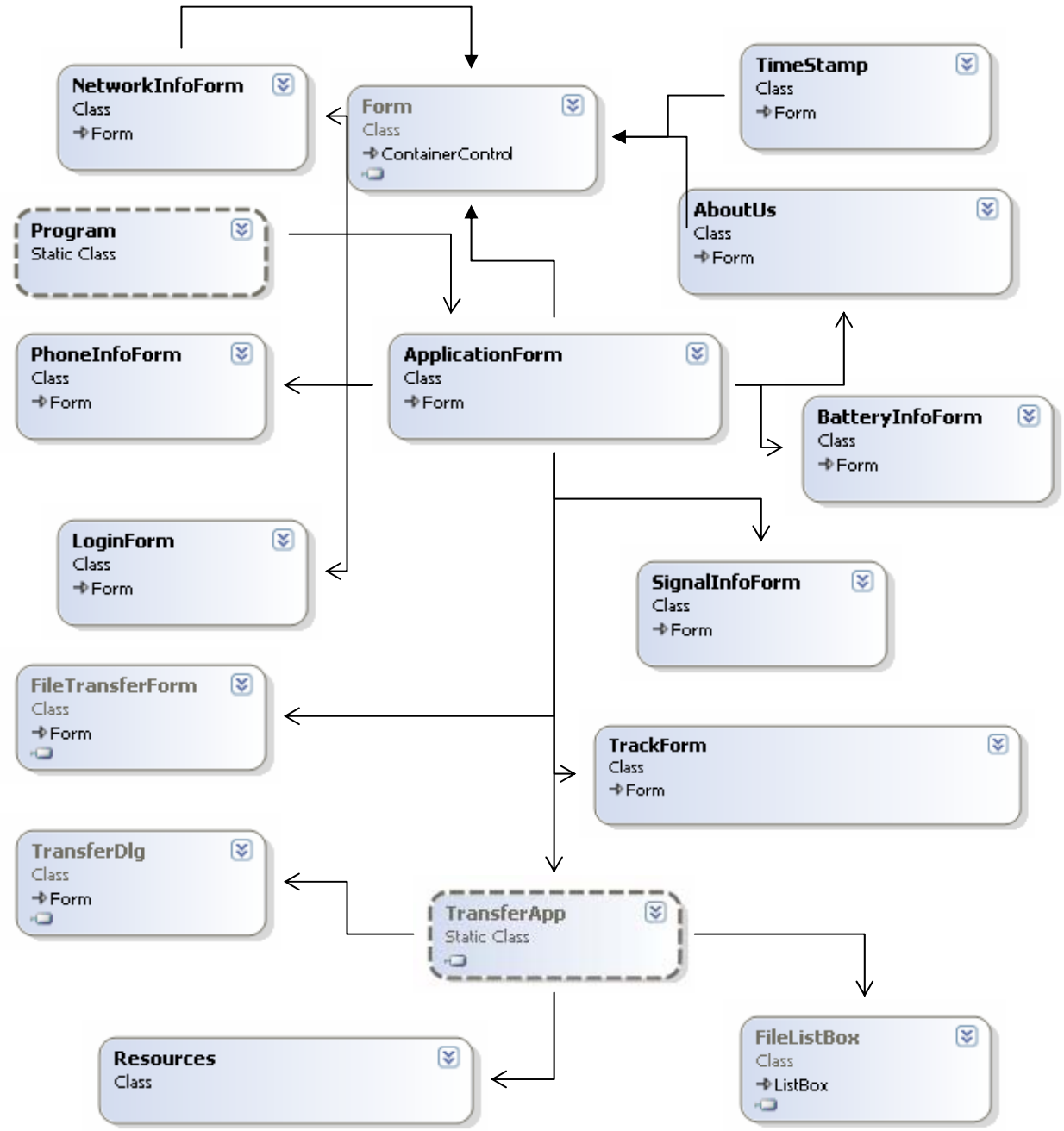
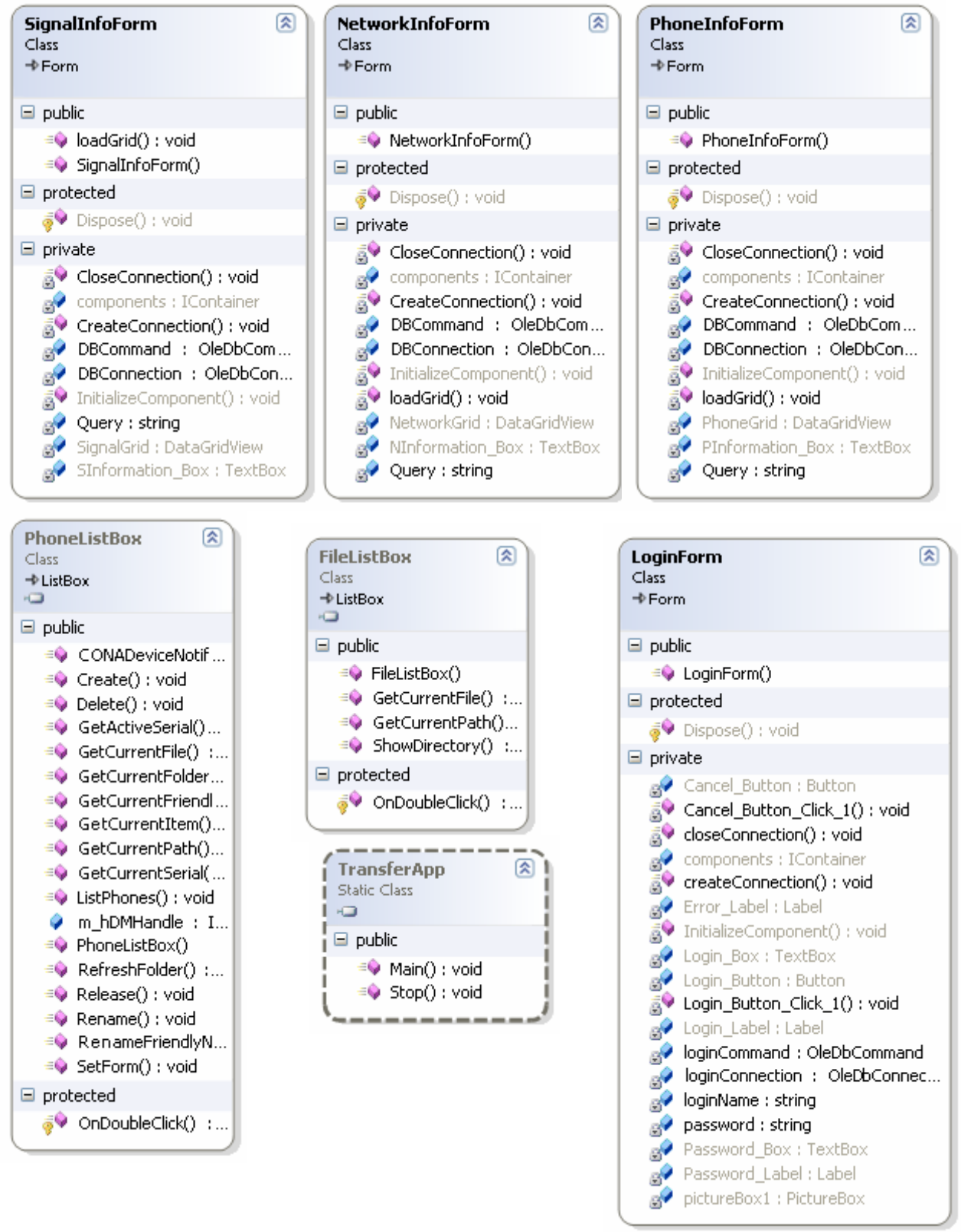


Figure 4: Class Diagram of Desktop-Side Application

The expanded view of the classes is given in Figure 5:



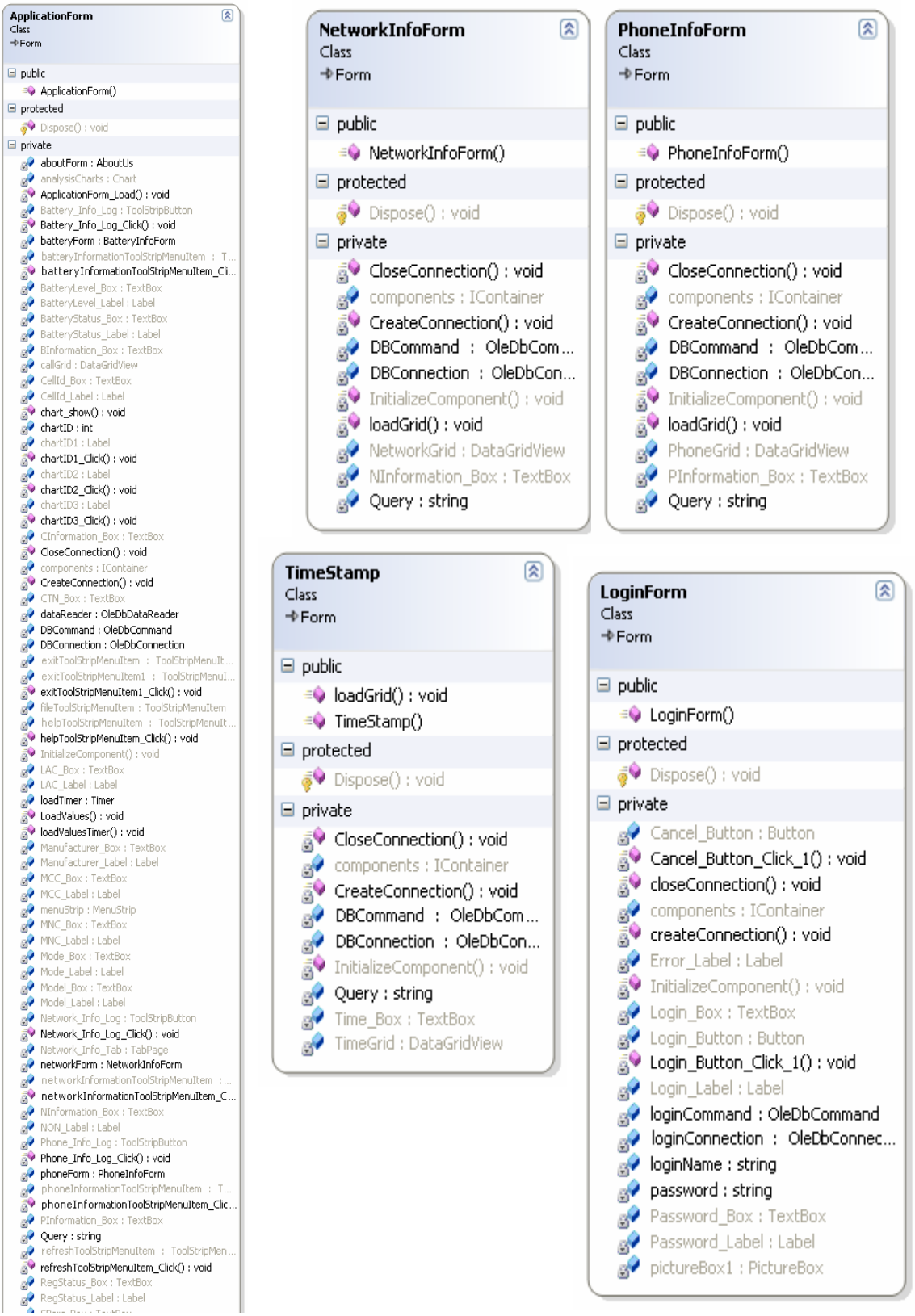


Figure 5: Expanded View of Class Diagram of Desktop-Side Application

6.1.2. Class Diagram of Phone-End Application

Figure 6: Class Diagram of File Transfer Application

6.2. USE CASES

6.2.1. Use Case: 1

Use case name: FetchValues

Participating Actors:

- Individual User
- Network Operator

Entry Condition

- SIM card is inserted in the mobile handset.

Flow of Events

- Individual user or network operator launches application.
- Individual user or network operator selects “Fetch” option.

Exit Condition

- The use case terminates when the individual user or network operator chooses “exit” option.

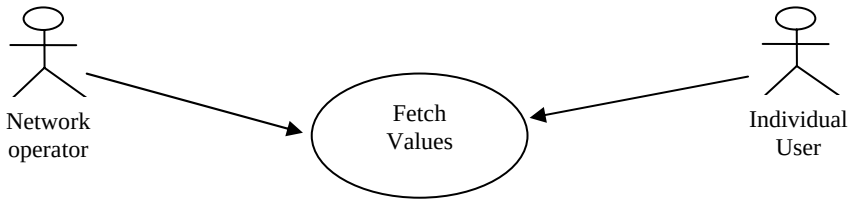


Figure 7: Use Case 1 FetchValues

6.2.2. Use Case: 2

Use case name: TransferData

Participating Actors:

- Network Operator

Entry Condition

- SIM card is inserted in the mobile handset.
- Phone-end application is launched.

Flow of Events

- Follow Use Case:1
- Network operator selects the mobile phone and the file to transfer from the mobile phone.
- Network operator selects the location on the desktop to save file.

Exit Condition

- The use case terminates when the log file is successfully transferred.

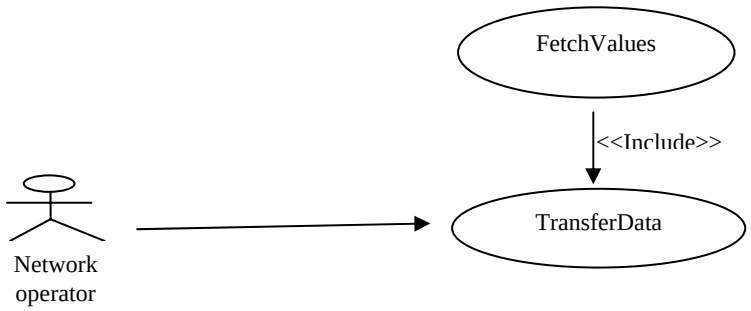


Figure 8: Use Case 2 TransferData

6.2.3. Use Case: 3

Use case name: AnalyzeValues

Participating Actors:

- Network Operator

Entry Condition

- Phone-end application is launched.
- File-transfer application is running.

Flow of Events

- Follow Use Case:2
- Values from transferred file on the desktop are stored in the database.
- Values from database are displayed on the GUI of desktop-side application.

Exit Condition

- The use case terminates when the values are displayed in charts.

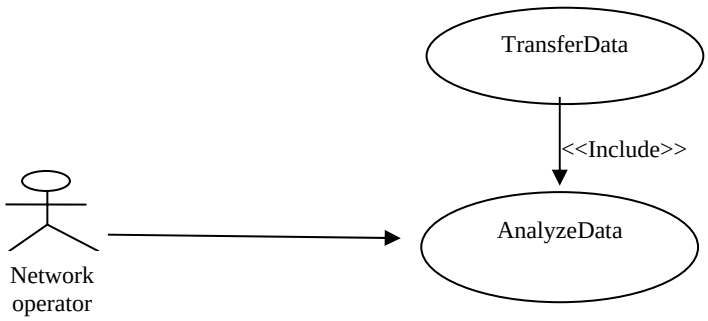


Figure 9: Use Case 3 AnalyzeValues

6.2.4. Use Case Diagram

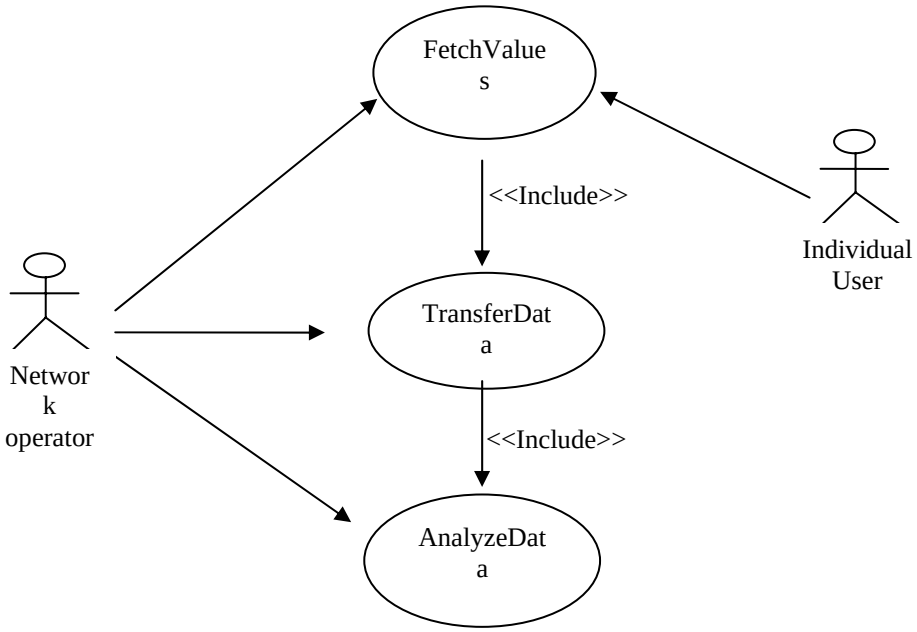


Figure 10: Consolidated Use Case Diagram

6.3. SEQUENCE DIAGRAM

The sequence diagram of the project is given below:

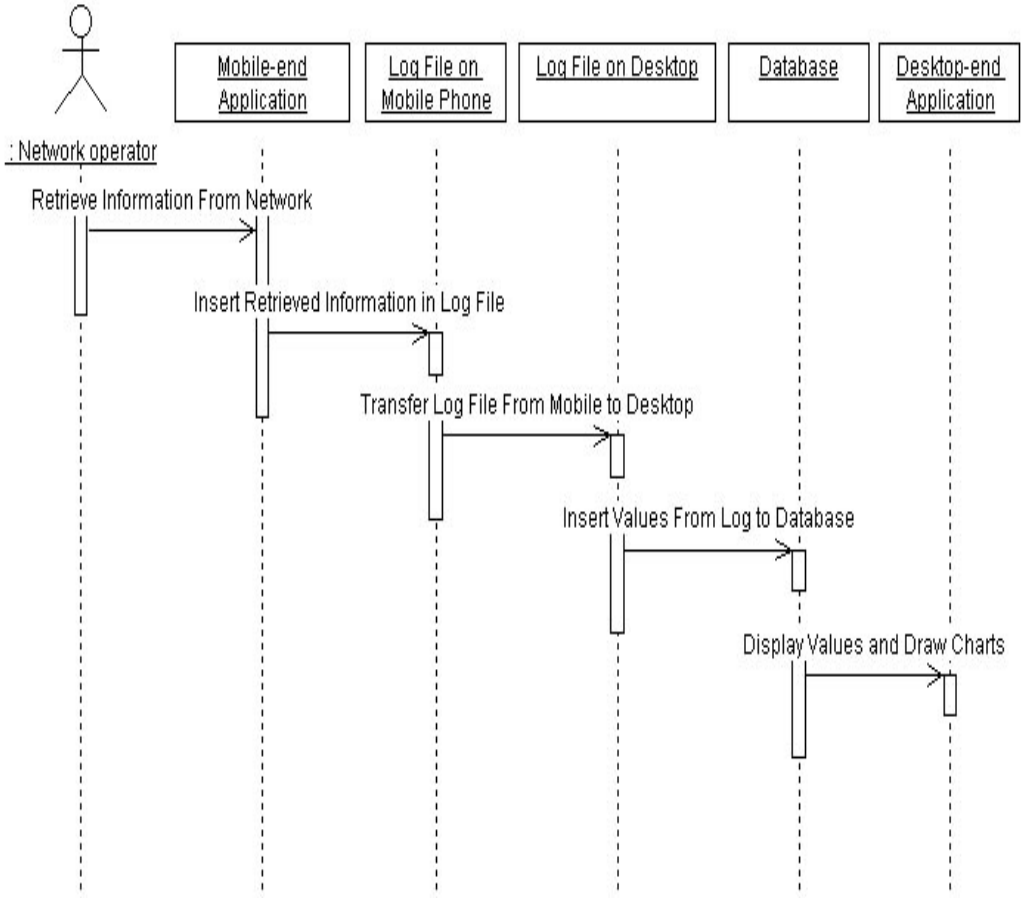


Figure 11: Class Diagram of File Transfer Application

6.4. DESIGN OF BACK-END DATABASE

6.4.1. Overall design of database

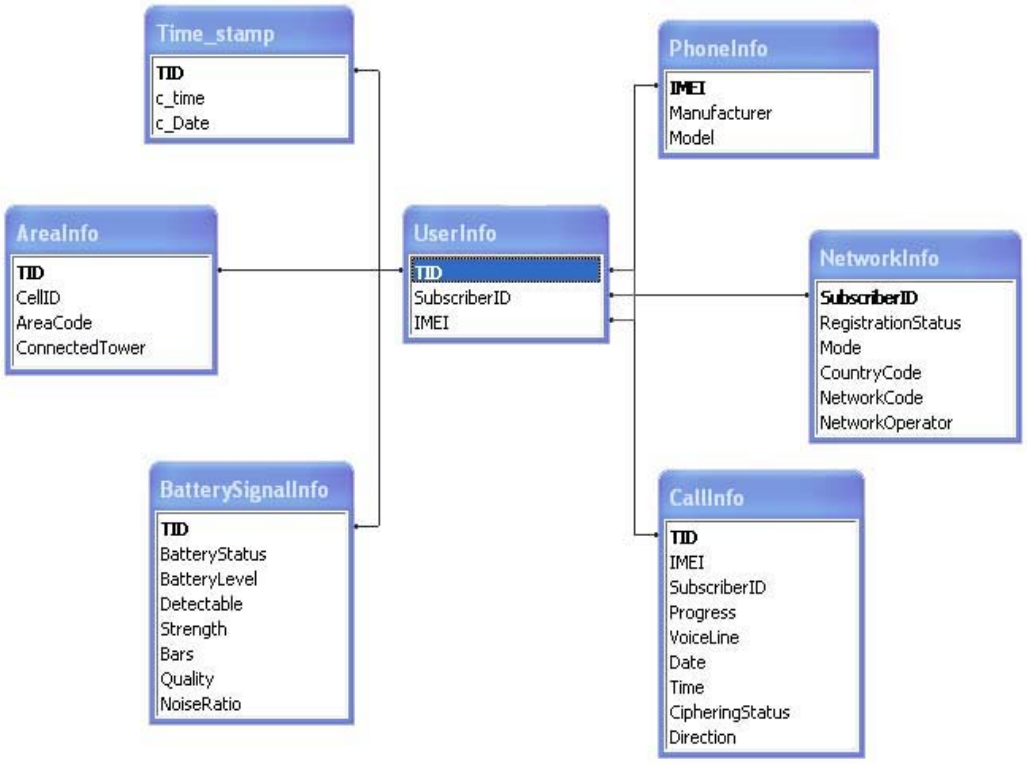


Figure 12: Database design

6.4.2. Database tables and their attributes

6.4.2.1. Area Info

Table 1: Area Information

Area Info		
Field	**Data type**	**Constraint**
TID	Number	FK
Cell ID	Number	PK
Area Code	Number	
Connected Tower	Text	

6.4.2.2. Battery Signal Info

Table 2: Battery and Signal Information

Battery Signal Info		
Field	**Data type**	**Constraint**
TID	Number	FK
Battery Status	Text	
Battery Level	Number	
Detectable	Text	
Strength	Number	
Bars	Number	
Quality	Number	
Noise Ratio	Number	

6.4.2.3. Call Info

Table 3: Call Information

Call Info		
Field	**Data type**	**Constraint**
TID	Number	FK
IMEI	Text	
Subscriber ID	Text	PK
Progress	Text	
Voice Line	Text	

Date	Text	
Time	Text	
Ciphering Status	Text	
Direction	Text	

6.4.2.4. Network Info

Table 4: Network Information

Network Info		
Field	**Data type**	**Constraint**
Subscriber ID	Text	FK
Registration Status	Text	
Mode	Text	
Country Code	Number	
Network Code	Number	
Network Operator	Text	

6.4.2.5. Phone Info

Table 5: Phone Information

Phone Info		
Field	**Data type**	**Constraint**
IMEI	Text	FK
Manufacturer	Text	PK
Model	Text	

6.4.2.6. Time Stamp

Table 6: Time Stamp

Time Stamp		
Field	**Data type**	**Constraint**
TID	Number	FK
C_time	Text	
C_Date	Text	

6.4.2.7. User Info

Table 7: User Information

User Info		
Field	**Data type**	**Constraint**
TID	Number	PK
Subscriber ID	Text	
IMEI	Text	

6.4.2.8. Login Info

Table 8: Login Information

Login Info		
Field	**Data type**	**Constraint**
Name	Text	PK
Password	Text	

Chapter 7

CONCLUSION AND FUTURE WORK

7.1. CONCLUSION

The project has been successfully tested. The phone-end application has been successfully deployed on Nokia 3250 and Nokia N80. All the information on the mobile phone has been successfully retrieved with the exception of network information because of the reason stated in section 4.3. A log file is created on the mobile phone which contains all the retrieved information. This log file has been successfully transferred to the desktop-side using Nokia PC Suite. All the retrieved information is stored in a database on the desktop-side and successfully displayed on the GUI of desktop-side application. Graphs of signal strength, signal quality and signal-to-noise ratio with respect to time have also been created.

This application can be used by individual GSM users as well as GSM network operators. Individual users can use this application on their mobile phones to check different network and signals related information and decide about the performance of their GSM network provider. Network operators can use this application to monitor the performance of their network. They can store the logs of information to analyze the general trend of the network. Graphical displays will also be available to them monitor the behavior of GSM network and hence identify current and potential problems.

7.2. FUTURE WORK

The functionality of the GSM network monitoring system could be enhanced by retrieving more values on the GSM network. Furthermore, Bluetooth or GRPS could be used for the purpose of transferring data from the phone-end application to the desktop-side application in order to make the application more robust.

REFERENCES

[1] Tisal, J. (2000). The GSM Network. GPRS Evolution: One step towards UMTS, John Wiley, New York.

[2] Nokia forum (15/05/2007). <http://forum.nokia.com>

[3] Symbian OS API. (22/04/2007) <http://developer.symbian.com/main/oslibrary>

[4] Stichbury, J. (2000). Symbian OS Explained - Effective C++ Programming for Smart Phones. John Wiley, New York.

[5] Babin, S. (2001). Developing Software for Symbian OS - An introduction to Creating Smart Phone Applications in C++. John Wiley, New York.

[6] Harrison R. (2003) Symbian OS C++ for Mobile Phones. John Wiley, New York.

Appendix A – Sample Source Code

****

To retrieve signal information on mobile phone:
****

```
void CGIMSAppDlg::GetSignalApiCompleteL(TInt aStatus,
CTelephony::TSignalStrengthV1 aSignalStrengthV1, TDesC16& aErrMsg)
   {
        if (aStatus != KErrNone)
     {
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdSD, &aErrMsg);
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdMS, &aErrMsg);
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdSB, &aErrMsg);
                }
  else
     {
        // Update Nw Regstrn status to UI
                    _LIT( KDash, "-");
            _LIT( KdBm, "dBm");

            TInt32 SignalStrength = aSignalStrengthV1.iSignalStrength;
            iEdwinSignalStrength.Copy(iSpace);
            iEdwinSignalStrength.AppendNum(SignalStrength);

            iEdwinSignalStrengthPer.Copy(KDash);
            iEdwinSignalStrengthPer.Append(iEdwinSignalStrength);
            iEdwinSignalStrengthPer.Append(KdBm);

            TInt8 SignalBar = aSignalStrengthV1.iBar;
            iEdwinBar.Copy(iSpace);
            iEdwinBar.AppendNum(SignalBar);

            TInt64 SQuality = (((SignalBar*80)/7)+ ((SignalStrength*20)/110));
                    iEdwinQuality.Copy(iSpace);
                    iEdwinQuality.AppendNum(SQuality);

                    iEdwinQualityPer.Copy(iEdwinQuality);
                    iEdwinQualityPer.Append(KPer);

                    TRealFormat ValFormat=
TRealFormat::TRealFormat(10,2);

                    TReal SRatio;
                    Math::Log(SRatio,(((SQuality)/(100-SQuality+1))+2));
                    iEdwinRatio.Copy(iSpace);
                    iEdwinRatio.AppendNum(SRatio,ValFormat);
```

```
                    SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdMS,
&iEdwinSignalStrengthPer);
                    SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdSB,
&iEdwinBar);
                    SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdSQ,
&iEdwinQualityPer);
                    SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdSR,
&iEdwinRatio);
                }
    }
```

****

To display phone information in GUI of phone-end application

****

```
void CGIMSAppDlg::GetPhoneIdApiCompleteL(TInt aStatus,
CTelephony::TPhoneIdV1 aPhoneId, TDesC16& aErrMsg)
   {
   if (aStatus != KErrNone)
      {
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdPhoneMfr, &aErrMsg);
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdPhoneModel, &aErrMsg);
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdPhoneSerNum,
&aErrMsg);
      }
   else
      {
            //Update Phone Mfr to UI
            iEdwinPhoneMfr.Copy(aPhoneId.iManufacturer);
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdPhoneMfr,
&iEdwinPhoneMfr);

            //Update Phone Model to UI
            iEdwinPhoneModel.Copy(aPhoneId.iModel);
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdPhoneModel,
&iEdwinPhoneModel);

            //Update Phone Ser num to UI
            iEdwinPhoneSerNum.Copy(aPhoneId.iSerialNumber);
            SetEdwinTextL((TInt)EGIMSAppDlgCtrlIdPhoneSerNum,
&iEdwinPhoneSerNum);
      }
   }
```

****

To transfer log file from phone-end application to desktop-side application

****

```
public unsafe void DoTransfer()
{
        uint dwMedia = CONAPI_CONSTANTS.CONAPI_MEDIA_ALL;
        uint dwDeviceID = 0;
        uint dwResult = 0;

        fixed (IntPtr* pp = &m_hFSHandle)
        {
                // Open FS
        dwResult=ConnAPI.CONAOpenFS(m_strSerial,&dwMedia,pp,&dwDeviceI
D);
        if ( dwResult != CONAPI_ERRORS.CONA_OK )
        {
                CONAPI_ERRORS.ShowError("CONAOpenFS failed!",dwResult);
                return;
        }

        // Register callback function
        dwResult =
        ConnAPI.CONARegisterFSNotifyCallback(m_hFSHandle,CONAPI_CONS
        TANTS.CONAPI_REGISTER,pfnFSCallBack);

        if ( dwResult != CONAPI_ERRORS.CONA_OK )
        {

        CONAPI_ERRORS.ShowError("CONARegisterFSNotifyCallBack
failed!",dwResult);
        ConnAPI.CONACloseFS(m_hFSHandle);
        return;
        }

        // If copy is enabled
        if ( m_bCopy )
        {
                dwResult = ConnAPI.CONACopyFile(m_hFSHandle,
                m_iMode | CONAPI_CONSTANTS.CONA_RENAME,
                m_strFile,m_strSource,m_strTarget);

                // If failed or user cancelled
                if ( dwResult != CONAPI_ERRORS.CONA_OK )
                {

        CONAPI_ERRORS.ShowError("CONACopyFile failed!",dwResult);
```

```
                }
        }

        // Unregister FS callback function
        dwResult =
        ConnAPI.CONARegisterFSNotifyCallback(m_hFSHandle,CONAPI_CONS
        TANTS.CONAPI_UNREGISTER,pfnFSCallBack);
        if ( dwResult != CONAPI_ERRORS.CONA_OK )
        {
                CONAPI_ERRORS.ShowError("CONARegisterFSNotifyCallback
unregister failed!",dwResult);
        }

        // Close FS
        dwResult = ConnAPI.CONACloseFS(m_hFSHandle);
        if ( dwResult != CONAPI_ERRORS.CONA_OK )
        {
                CONAPI_ERRORS.ShowError("CONACloseFS failed!",dwResult);
        }
        return;
        }
}
```

****

To load data from file

****

```
public string[] importData()
      {
        string text = "";

        try
        {
           using (StreamReader sr = File.OpenText(filePath))
           {
              string str;
              while ((str = sr.ReadLine()) != null)
              {
                 text += str;
              }
           }
           File.Delete(filePath);
        }
        catch (Exception exp)
        {
```

```
            MessageBox.Show("File Reading Failed: error "+exp);
        }

        data = text.Split(new char[] { ',' }, 27);

        for (int valCount = 0; valCount < 27; valCount++)
        {
            data[valCount] = (data[valCount].Substring(1, (data[valCount].Length -
2)));
        }
        return data;
    }
```

****

To display values on the deskop-side

****

```
private void LoadValues()
    {
        Query = "SELECT * FROM NetworkInfo WHERE SubscriberID=(select
LAST(SubscriberID) from NetworkInfo)";

        DBConnection.Open();
        DBCommand = new OleDbCommand(Query, DBConnection);

        dataReader = DBCommand.ExecuteReader();

        if (dataReader != null)
        {
            dataReader.Read();
            Subscriber_Box.Text = dataReader[0].ToString();
            RegStatus_Box.Text = dataReader[1].ToString();
            Mode_Box.Text = dataReader[2].ToString();
            MCC_Box.Text = dataReader[3].ToString();
            MNC_Box.Text = dataReader[4].ToString();
            NON_Box.Text = dataReader[5].ToString();
          }
         DBCommand.Dispose();
         dataReader.Close();
         CloseConnection();
        ////////////////////////////////////////////////////////////////
         Query = "SELECT * FROM AreaInfo WHERE TID=(select MAX(TID)
from AreaInfo)";

         DBConnection.Open();
```

```
            DBCommand = new OleDbCommand(Query, DBConnection);

            dataReader = DBCommand.ExecuteReader();

            if (dataReader != null)
            {
               dataReader.Read();
               CellId_Box.Text = dataReader[1].ToString();
               LAC_Box.Text = dataReader[2].ToString();
               SPN_Box.Text = dataReader[3].ToString();
            }
            DBCommand.Dispose();
            dataReader.Close();
            CloseConnection();

           ////////////////////////////////////////////////////////////////////
            Query = "SELECT * FROM PhoneInfo WHERE IMEI=(select LAST(IMEI)
from PhoneInfo)";

            DBConnection.Open();
            DBCommand = new OleDbCommand(Query, DBConnection);

            dataReader = DBCommand.ExecuteReader();

            if (dataReader != null)
            {
               dataReader.Read();
               Manufacturer_Box.Text = dataReader[1].ToString();
               Model_Box.Text = dataReader[2].ToString();
               SerialNumber_Box.Text = dataReader[0].ToString();
            }
            DBCommand.Dispose();
            dataReader.Close();
            CloseConnection();
           ////////////////////////////////////////////////////////////////////
            Query = "SELECT * FROM BatterySignalInfo WHERE TID=(select
MAX(TID) from BatterySignalInfo)";

            DBConnection.Open();
            DBCommand = new OleDbCommand(Query, DBConnection);

            dataReader = DBCommand.ExecuteReader();

            if (dataReader != null)
            {
               dataReader.Read();
               BatteryStatus_Box.Text= dataReader[1].ToString();
               BatteryLevel_Box.Text = dataReader[2].ToString()+" %";
```

```
            SDetectable_Box.Text = dataReader[3].ToString();
            SQuality_Box.Text = dataReader[6].ToString()+" %";
            SStrength_Box.Text = "-" + dataReader[4].ToString()+" mDb";
            SBars_Box.Text = dataReader[5].ToString();
            SRatio_Box.Text = dataReader[7].ToString();
        }
        DBCommand.Dispose();
        dataReader.Close();
        CloseConnection();

        ////////////////////////////////////////////////////////////////////
        Query = "SELECT TID, Progress, VoiceLine, Date, Time, CipheringStatus,
Direction FROM CallInfo";

        DBConnection.Open();
        DBCommand = new OleDbCommand(Query, DBConnection);

        OleDbDataAdapter DAddapter = new OleDbDataAdapter(DBCommand);
        DataSet dSet = new DataSet();

        DAddapter.Fill(dSet);

        callGrid.DataSource = dSet.Tables[0];


        DBCommand.Dispose();
        dSet.Dispose();
        CloseConnection();
    }
**********************************************************************
```

To display log file of network information on the desktop-side application

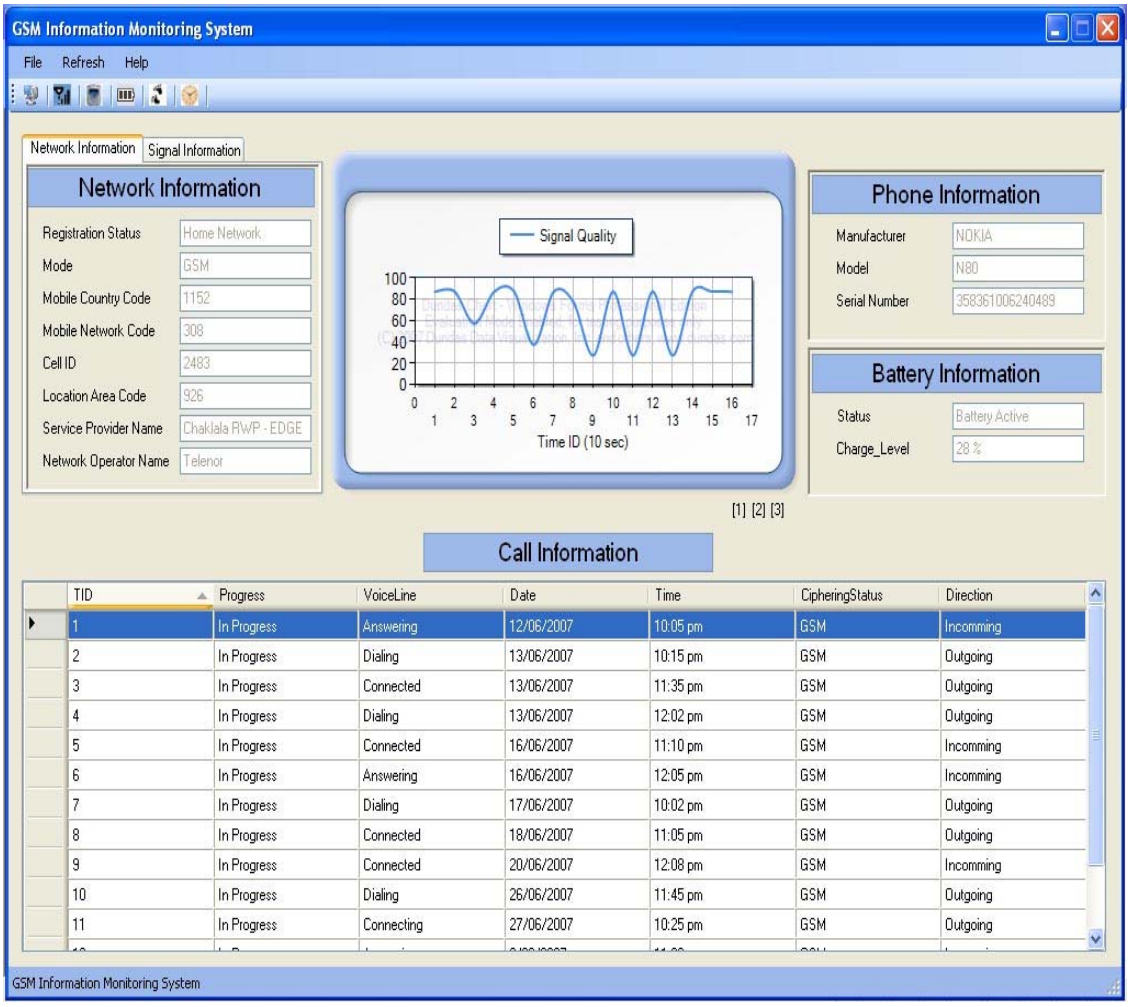
****

```
public partial class NetworkInfoForm : Form
  {
    private OleDbConnection DBConnection;
    private OleDbCommand DBCommand;
    private string Query;

    public NetworkInfoForm()
    {
      InitializeComponent();
      CreateConnection();
      loadGrid();
    }
```

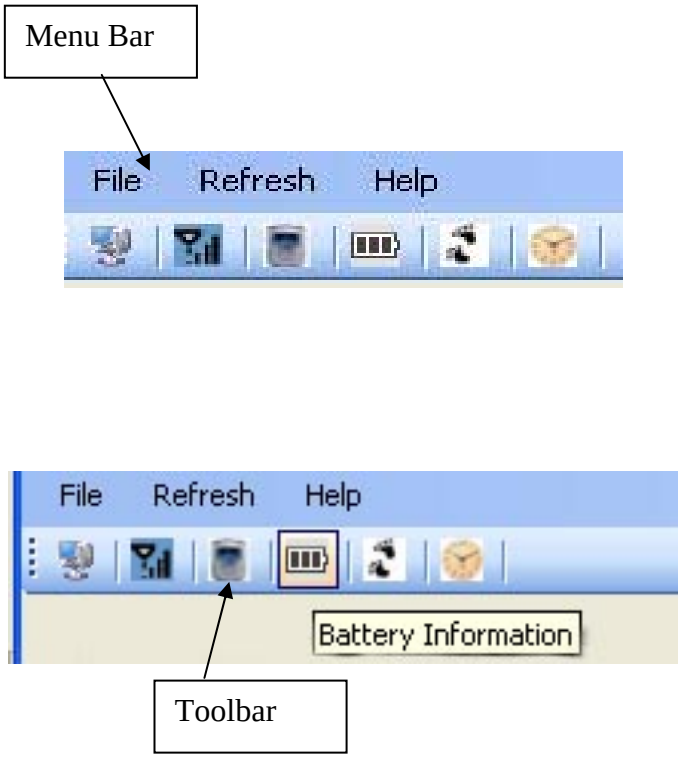
```
        private void loadGrid()
        {

            Query = "SELECT * FROM NetworkInfo";

            DBConnection.Open();
            DBCommand = new OleDbCommand(Query, DBConnection);

            OleDbDataAdapter DAddapter = new OleDbDataAdapter(DBCommand);
            DataSet dSet = new DataSet();

            DAddapter.Fill(dSet);

            NetworkGrid.DataSource = dSet.Tables[0];

            DBCommand.Dispose();
            dSet.Dispose();
            CloseConnection();
        }

        private void CreateConnection()
        {
            DBConnection = new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=Database/GIMS_Database.mdb");
        }

        private void CloseConnection()
        {
            DBConnection.Close();
        }
    }
```

Appendix B – Screen shots

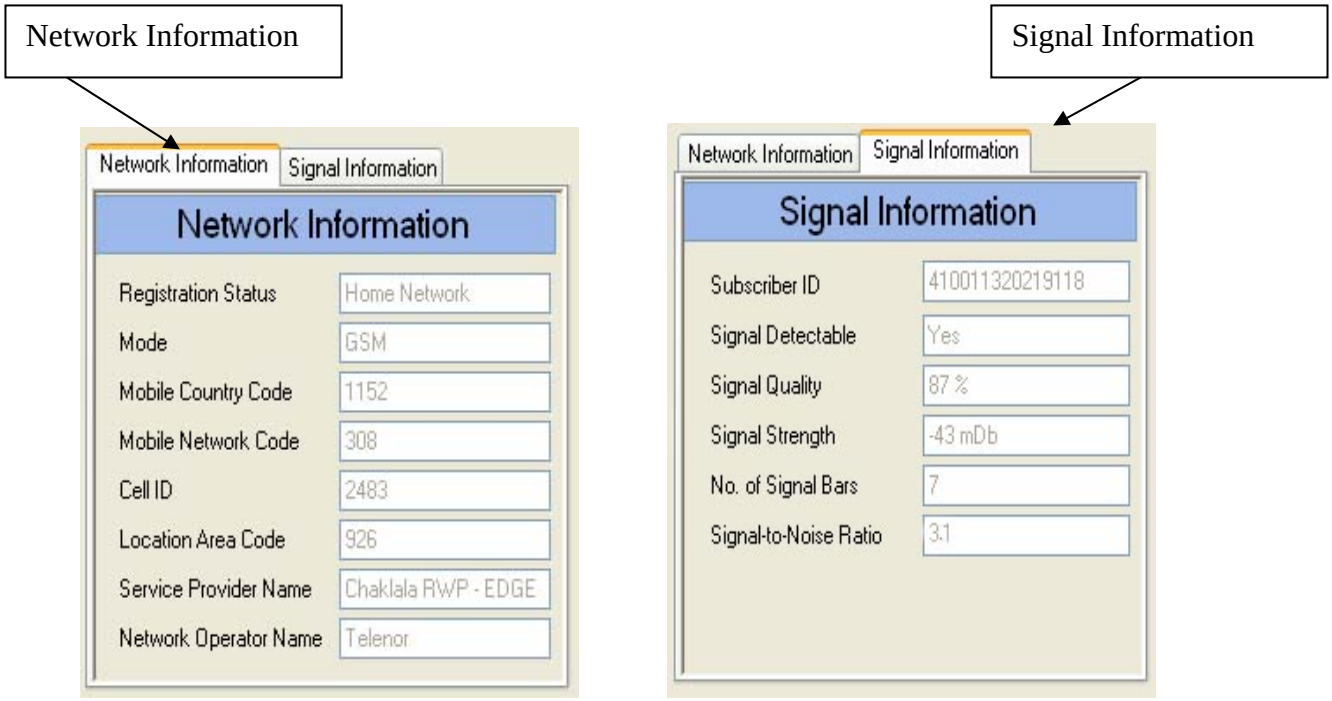
Overall View of Desktop-Side Application



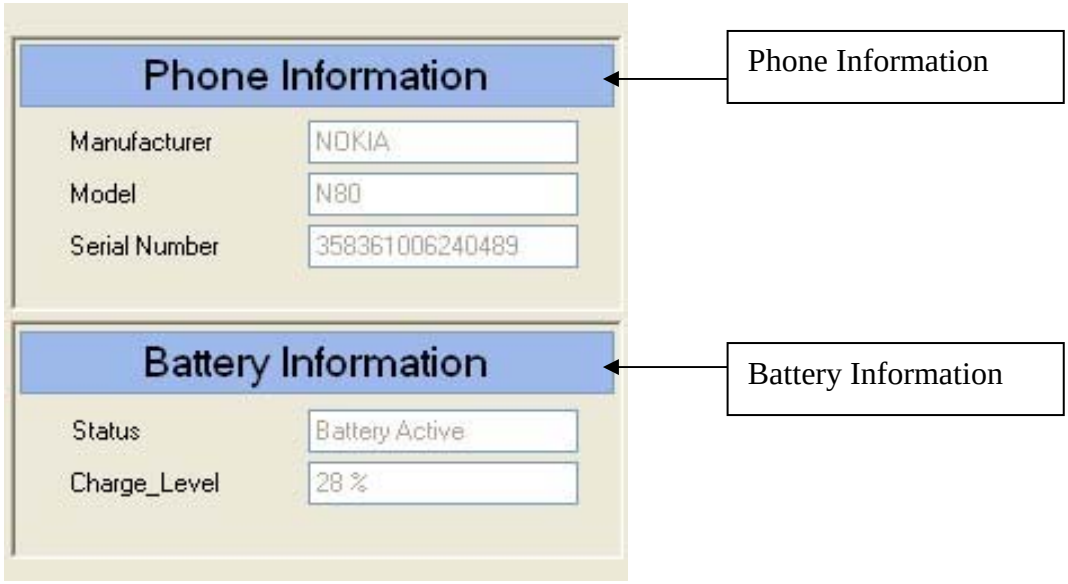
Menu and Toolbar Interaction



Dialog for Displaying Signal and Network Information



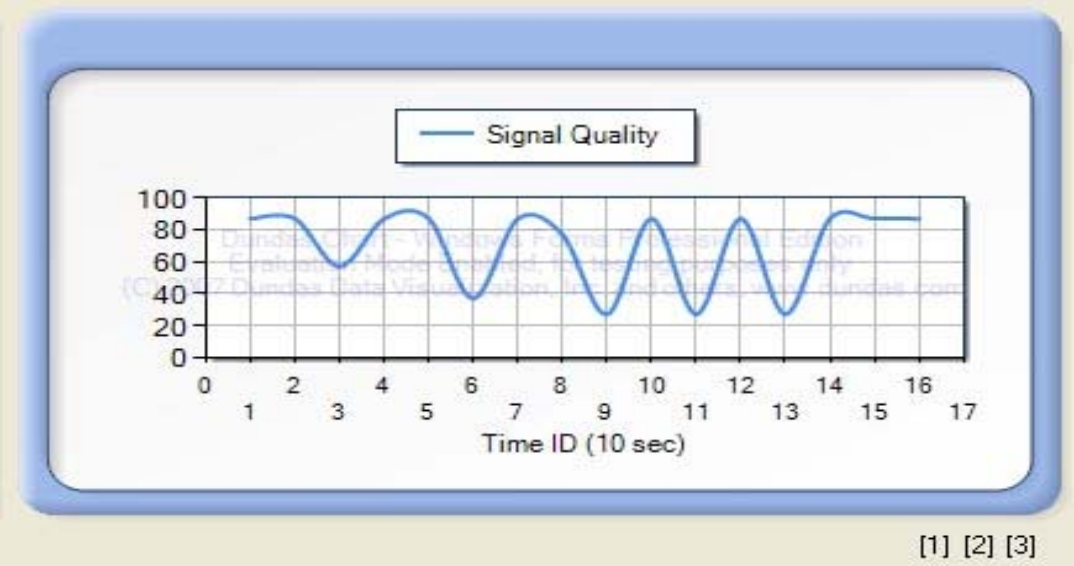
Dialog for Displaying Phone and Battery Information



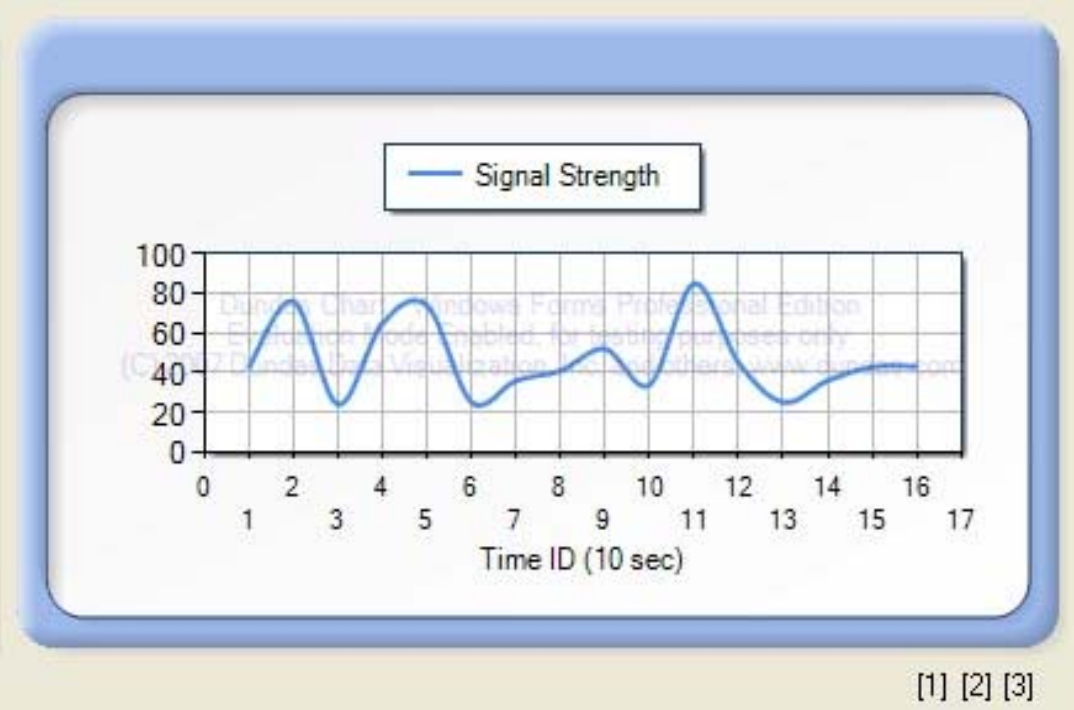
Dialog for Displaying Call Information

TID	Type	VoiceLine	Status	Date	Time	Duration	Number	CipheringStatus	Direction
1	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
2	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
3	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
4	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
5	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
6	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
7	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
8	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
9	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
10	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
11	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
12	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing
13	Voice	Ringing	Busy	22-05-07	12:05 am	90	22251	Enabled	Outgoing

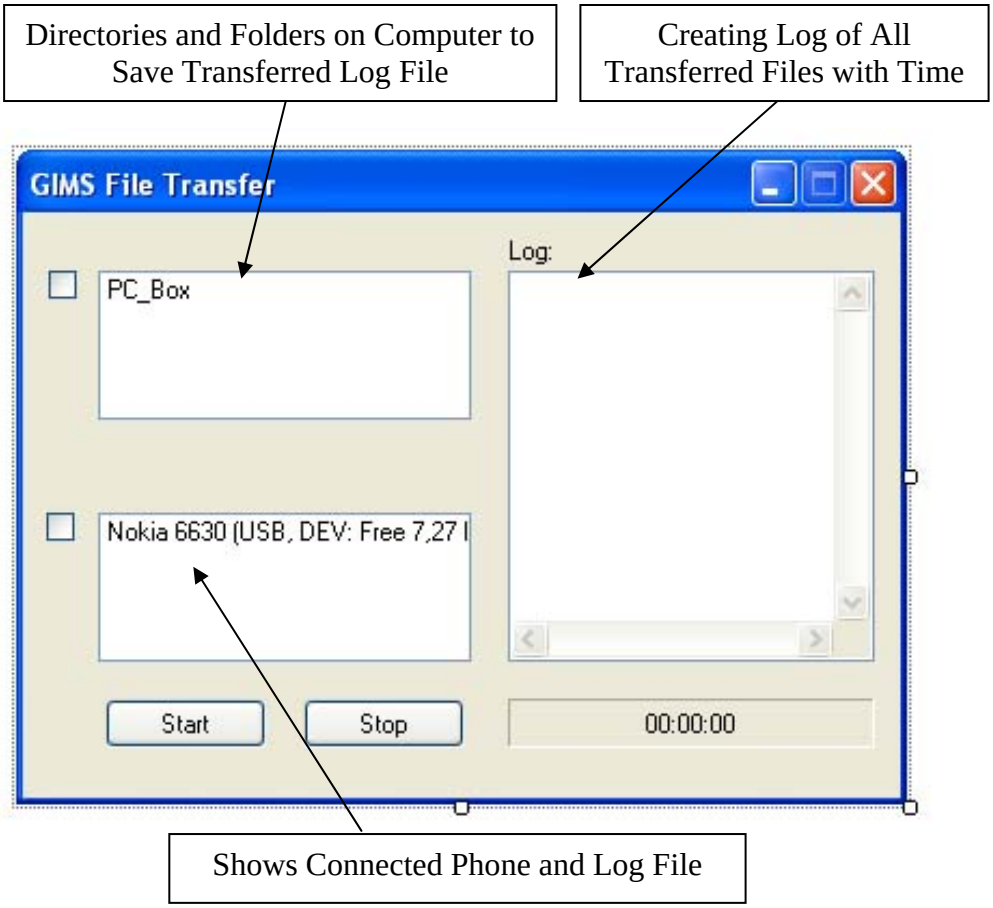
Dynamic Chart Showing Varying Signal Quality With Respect To Time



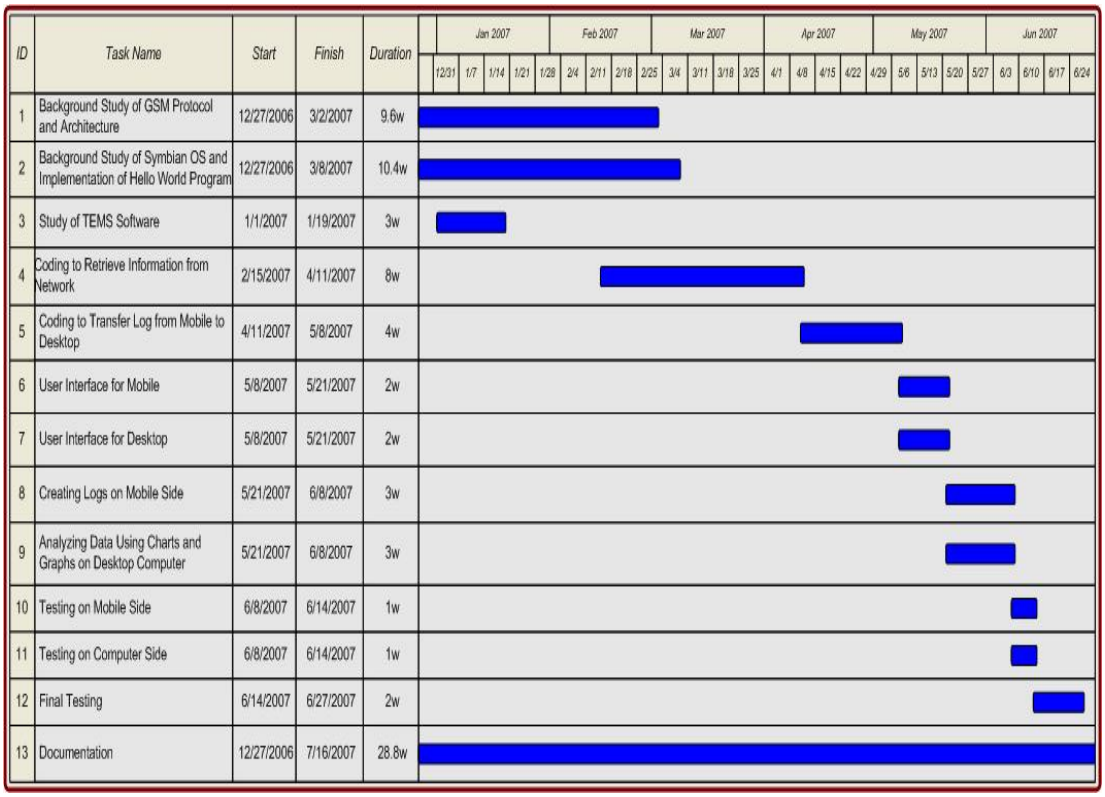
Dynamic Chart Showing Varying Signal Strength With Respect To Time



File Transfer Dialog Box



Appendix C – Project Timeline



ID	Task Name	Start	Finish	Duration
1	Background Study of GSM Protocol and Architecture	12/27/2006	3/2/2007	9.6w
2	Background Study of Symbian OS and Implementation of Hello World Program	12/27/2006	3/8/2007	10.4w
3	Study of TEMS Software	1/1/2007	1/19/2007	3w
4	Coding to Retrieve Information from Network	2/15/2007	4/11/2007	8w
5	Coding to Transfer Log from Mobile to Desktop	4/11/2007	5/8/2007	4w
6	User Interface for Mobile	5/8/2007	5/21/2007	2w
7	User Interface for Desktop	5/8/2007	5/21/2007	2w
8	Creating Logs on Mobile Side	5/21/2007	6/8/2007	3w
9	Analyzing Data Using Charts and Graphs on Desktop Computer	5/21/2007	6/8/2007	3w
10	Testing on Mobile Side	6/8/2007	6/14/2007	1w
11	Testing on Computer Side	6/8/2007	6/14/2007	1w
12	Final Testing	6/14/2007	6/27/2007	2w
13	Documentation	12/27/2006	7/16/2007	28.8w

Project start: 15th November 2006

Development start: 1st February 2007

Project completion: 16th July 2007