SICHENG CHEN

MOBILE INFORMATION SYSTEM

Implementation of SMS ticket booking system

Bachelor's Thesis

Information Technology

May 2009

MIKKELIN AMMATTIKORKEAKOULU

Mikkeli University of Applied Sciences

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** May 11, 2009
Author(s) Sicheng Chen	**Degree programme and option** Information Technology

Name of the bachelor's thesis

Mobile Information system — Implementation of SMS ticket booking system

Abstract

Nowadays, more and more enterprises develop and manage their own commercial information systems to keep pace with the E-Business world. People can get the commercial information by surfing the internet. In spite of this, people sometimes still need the 2G mobile services to communicate with information systems for network or economic reasons.

The aim of this study is to develop a commercial information system, in specific, a ticket-booking system, which supports the booking process on mobile phones. For time and device constraints, the system will be simplified but it should be able to complete the data processing between the mobile client and the data server.

Afterwards, I will try to compare the features of the SMS ticket booking system to the common online ticket booking system in some aspects. By comparing the performance in the stability, flexibility, client-oriented and the information security, a final conclusion will be made.

Subject headings, (keywords)

Mobile Business, Information system, GSM, SMS, Serial Port Communication, SMSLib, Transaction Logic, MySQL, Session Control;

Page	Language	URN
68	English	NBN:fi:mamk-opinn2009A5256

Remarks, notes on appendices

The bachelor's thesis is secret until May 30, 2009.

Tutor Matti Koivisto	**Employer of the bachelor's thesis** Mikkeli University of Applied Sciences

TABLE OF CONTENTS

1. Introduction 1
2. GSM network and GSM platform 4
2.1 SMS Network 4
2.1.1 Introduction to SMS 4
2.1.2 The Implementation of Point-to-point SMS in the Network 5
2.2 The Architecture of SMS Platform 7
2.2.1 SMS Gateway 7
2.2.2 Small Business SMS Platform 9
3. GSM modem and AT command 10
3.1 Introduction to GSM modem 10
3.2 SIEMENS MC35 GSM Modem 11
3.2.1 The Main Body 12
3.2.2 Siemens MC35 GSM Modem Status 13
3.2.3 Interfaces and Connected Devices 15
3.3 Control of SIEMENS MC35 GSM Modem 17
3.3.1 Introduction to AT commands 17
3.3.2 Some Basic AT Commands for SMS 18
4. Serial Port Program and SMSLib 21
4.1 Serial Port Programming 21
4.1.1 The Installation of JAVA Communications API 23
4.1.2 Basic Serial Port Programming based on JAVA Communications API 24
4.2 SMSLib 29
4.2.1 Introduction to SMSLib 29
4.2.2 Installation of SMSLib 32
4.4.3 Basic Implementation of SMSLib 33
5. DESIGN AND IMPLEMENTATION OF TRANSACTION 39
5.1 Transaction Model of SMS Ticket Booking 39

5.1.1 Requirements Analysis of SMS Ticket Booking 39
5.1.2 Design of the Transaction Model and Booking System 43
5.2 Implementation of Database 45
5.2.1 Design of Database 45
5.2.2 How to Implement the Database 47
5.3 Control of Sessions 52
5.3.1 Data Structure of Sessions 52
5.3.2 Transaction Logic 54
5.3.3 Implementation of Session Timeout 54
5.4 Demonstration of Transaction Operation 57
6. Conclusion 62
6.1 Advantages and Disadvantages of SMS Ticket Booking System 62
6.2 Future Plan for Amelioration 64
Index of References 66
Books 66
Electronic sources 66

1. INTRODUCTION

Nowadays, more and more enterprises develop and manage their own commercial information systems to keep pace with the E-Business world. People can utilize their personal computers, laptops and mobile phones to get the commercial information from the website and complete their affairs. There are many kinds of E-Business system being used by people today. One of the most popular modes is the on-line shopping system, such as eBay and Amazon. In this thesis, I may concern about another kind of E-business system, the on-line ticket-booking system.

For booking tickets for performances and movies, people intend to have a very flexible and mobile way to realize the ticket-booking business. So using the mobile phone to book tickets from a booking system is a good choice. Moreover, after the booking process and the payment confirmation, the ticket-booking system can send the 2-dimensional bar code to the user's mobile phone directly as electronic tickets. By this process, the clients do not need to print out their electronic tickets or order form.

On the other hand, there is still a possibility that people have to communicate with their service provider through SMS. In respect that in some countries, the 3G communication system is still under construction and many people still hold their 2G instruments. Moreover, the 2G communication system, such as GSM, cannot provide stable and fast network speed, which sometimes causes data transmission lost. The high rent fee of the networks is also a big matter for people to prefer the SMS. Hence, people are inclined to use the commercial information system which is based on the SMS communication.

For SMS communication, there is a solution which is adopted by real business world that the user end sends SMS to the data server, by using the automation SMS server and SMS modem, an end-to-end SMS service is built. For instance, by sending SMS to our SIM card service provider, we can get our service information or review our balance.

As we know, the GSM communication system has many flaws in information security. The A5 algorithm has already been cracked by some groups. However, the commercial information systems which always deal with the personal privacy information demand high level of information security.

The aim of this study is to develop a commercial information system, in specific, a ticket-booking system, which supports the booking process on mobile phones. By simplifying the real business mode which was mentioned before (Figure 1) to a mode that is cheaper and easier for students to implement, I will replace the SMS server and other professional connections with the SMS modem and specific applications. The system is still able to complete the data processing between the mobile client and the data server.

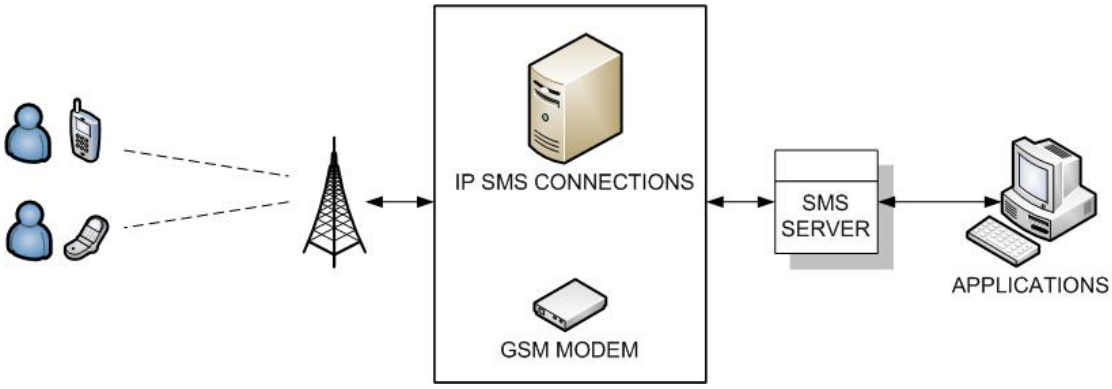


Figure 1.1 The real business mode with SMS server.

Afterwards, I will try to compare the features of the SMS ticket booking system to the common online ticket booking system in some aspects. By comparing the performance in the stability, flexibility, client-oriented and the information security, a final conclusion will be made.

In order to do these I will first consider the network of SMS and the ways of building SMS business platform in Chapter 2. In Chapter 3, I will present an introduction to the MC35 GSM terminal and the design of AT commands to implement the data transmission. Chapter 4 expounds the comparison between two ways of serial port

programming. In the Chapter 5, the transaction model and design of the application will be presented. Finally, a discussion on the advantages and disadvantages of SMS ticket booking system combined with how to enhance the quality of this service mode will be made as the conclusion of this thesis.

2. SMS NETWORK AND SMS PLATFORM

Nowadays, Short Message Service (SMS) has become one of the most popular communication methods in people's daily life, especially for youth. In China, youth who have a fancy for SMS are called the 'thumb tribe'. Regarding the low price and other features of SMS, more and more people intend to use SMS rather than telephone-call to accomplish the daily long-distance conversations. Last year in Lisbon, 2 January 2008 – Vodafone processed 992 million SMS over the festive season (from 21 December to 1 January), an all-time record for the Vodafone network and an increase of 105% on the same period of 2007 (Vodafone, 1/2008 [referred 6.4.2009]). Not only for small talk, but SMS also can be used in many ways, such as SMS newspaper, commonweal (Vodafone, 5/2008 [referred 6.4.2009]), or advertisement. In this thesis, booking tickets by SMS is the main purpose. Thus, in this chapter, I will introduce the implementation of SMS and the construction of the SMS platform.

2.1 SMS Network

2.1.1 Introduction to SMS

The GSM Short Message Service (SMS) is a communication standardized service, utilizing communications protocols which have been standardized, allowing the interchange of short alphanumeric (letters and numbers) (Redl et al. 1998, 211) messages between mobile devices (Wikipedia [referred 5.4.2009]). After receiving, the messages appear on the phone's display and are restored by the mobile phone.

SMS was originally defined as part of the GSM standards in 1985, which means sending a 160-character message to and from GSM mobile devices (Finn Trosby 2004, 5). Moreover, the "store-and-forward" mechanism ensures the delivery of message despite the status of receiver. After the receiver has got the message, the sender will be

notified. SMS also supports other type of mobile telecommunication standards such as CDMA (Redl et al. 1995, 325) and Digital AMPS. Most SMS messages are mobile-to-mobile text messages, and they work world-wide.

2.1.2 The Implementation of Point-to-point SMS in the Network

In general, the GSM distinguishes the short message service into two types, point-to-point SMS and cell broadcast SMS. In this thesis, the on-line booking system is based on the point-to-point SMS, so the implementation of point-to-point SMS will be presented as following.

We can trail the flow of a message to get to know the structure of the relevant GSM network. The short message is generated by the Short Message Entity (SME) which might be a mobile device or a computer located in a fixed network dispatching a message to the Service Center (SC). The SC deals with all the functions related to point-to-point SMS. When receiving a message from a SME, the SC sends it to the target addresses. In some situation, such as the receiver is power-off or out of range, the message could not reach the receiver, then it will be stored by the SC until the receiver can receive it or the message runs out of the maximum time. If requested, the SC sends reports to the generator of messages to notify the failure of delivery (Redl et al. 1998, 213-215).

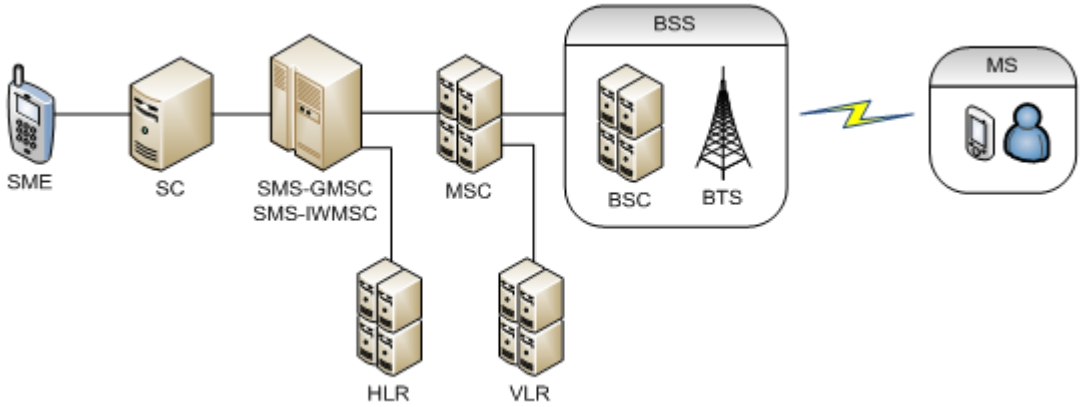


Figure 2.1 SMS network architecture.

The short message service-gateway mobile services switching center (SMS-GMSC) is an isolated function part from the MSC allows it to get the generated short message from the SC. Afterwards, it will send request to the Home Location Register (HLR) to find out where the recipient mobile station (MS) is located. Then the SMS-SMSC will send the message to the MSC and through it to get to the terminal. In Figure 2.1, we can find out the flow (left-to-right) of the short message processing and delivery.

In different with the SMS-GMSC, the short message service-interworking mobile service switching center (SMS-IWMSC) serves as the export switching within the MSC. The responsibility of SMS-IWMSC is dealing with the short messages which are sending from the PLMN or the MS in Figure 2.1. After being received by the MSC and the SMS-IWMSC, the message will be submitted to the recipient's SMS service center (SMserver wireless short message service tutorial 1999 [referred 6.4.2009]).

The Home Location Register (HLR) is a database used for permanent storage and management of the identity and user data of all the subscribers belonging to the local MSC. Upon interrogation by the SMSC, the HLR provides two kinds of information, one is the location information about the recipient mobile station, and the other is the statues information indicating whether the recipient mobile station is capable to receive the short message.

The Mobile Switching Center (MSC) is a complete exchange system which manages the switching functions of the system. With all of its registers, MSC controls calls to and from other data systems, mobile networks and individual mobile stations.

The Base Station System (BSS) provides all radio-related functions, and its primary responsibility is transmitting voice and data traffic between the mobile stations. The BSS consists of Base Station Controllers (BSC) and the Base Transceiver Stations (BTS). The BSC mainly take the responsibility of the intelligence work behind the BTS. It handles the allocation of radio channels and the controlling of handovers between

BTS (Wikipedia 2008b, [referred 6.4.2009]).

The Mobile Station (MS), in the particular case of SMS, is either a sender or a recipient of a short message. The mobile station would also inform the network about abnormal (erroneous) situations, such as memory overflow.

The Mobile Station (MS) is the wireless terminal capable of sending and receiving short messages. Besides the telephone call and SMS, the MS also communicates with the base station to transfer some situation information and error reports.

2.2 The Architecture of SMS Platform

2.2.1 SMS Gateway

In order to realize the business service based on SMS, an SMS platform should be implemented. For real-business service, as mentioned in the introduction, there should be a SMS gateway and a SMS server to deal with the complicated SMS world.

As we know, the business service based on the SMS needs to bridge the communication between the server of service and the mobile stations. Thus, in this situation, the server of service which contains some applications for business performs as the SME in the SMS network system. To send and receive SMS text messages on the business-server, the server has to face clients from different places and different wireless carriers which use different types of SCs. It means that the SMS text messaging application which runs on the server may need to support multiple SC-specific protocols (Developers' Home 2/2008a [referred 6.4.2009]). Consequently, the development of application becomes more complex. (Figure 2.2)

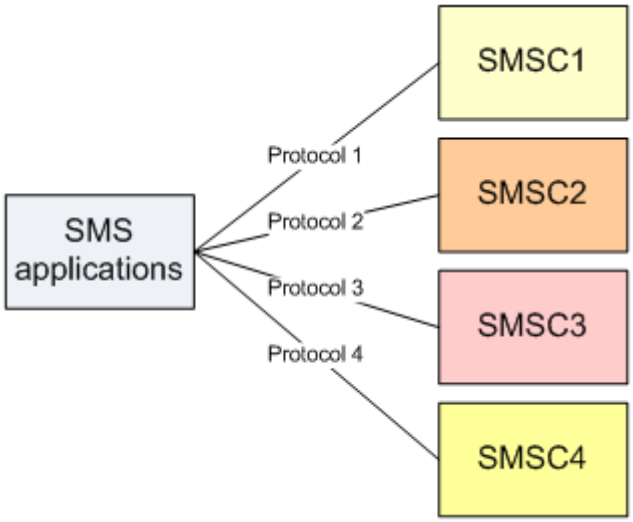


Figure 2.2 SMS accesses to the SMSCs directly.

In this situation, setting up an SMS gateway to cope with different kinds of SC and protocols is a good solution. The SMS text messaging application now does not need to adapt to the SC and protocols which are developed by different vendors. To support more SMSCs, just need to modify the settings of the SMS gateway and do not need to change the source of the application anymore. The use of a SMS gateway can shorten the development of SMS application noticeably. (Figure 2.3)

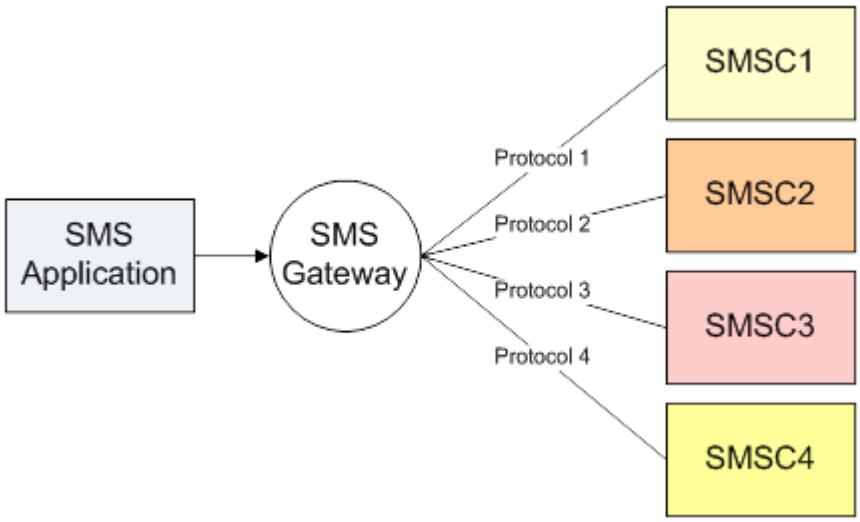


Figure 2.3 SMS accesses to the SMSCs with SMS gateway.

2.2.2 Small Business SMS Platform

Actually, it's difficult for small businesses or individual application developers to afford a professional business solution such as access to the SMS gateway and SMSC, and such devices are also unnecessary because small data flow will be needed. Buying professional SMS devices means the total fee will be very high (Developers' Home 2/2008b [referred 6.4.2009]).

From this point of view, a more convenient and economic way to send SMS messages is utilizing the subscribers supported by telecommunication service providers. Besides direct connections to the SMSC and SMS gateway, using the mobile phone or GSM modem which contains the subscriber identity (SIM card) is a good choice for small business firms or individuals. In this thesis, this solution will be discussed and realized.

As Figure 2.4 shows, the SMS application directly connects to the GSM modem, but not to the SMSC or the SMS gateway. The GSM modem also performs as a mobile station (MS) in the GSM network architecture. Although this kind of SMS platform cannot process bulk data or multiple users at the same time, it can serve for the small business firm well and to be easily implemented in the lab environment. More information about the GSM modem will be presented in the following chapter.

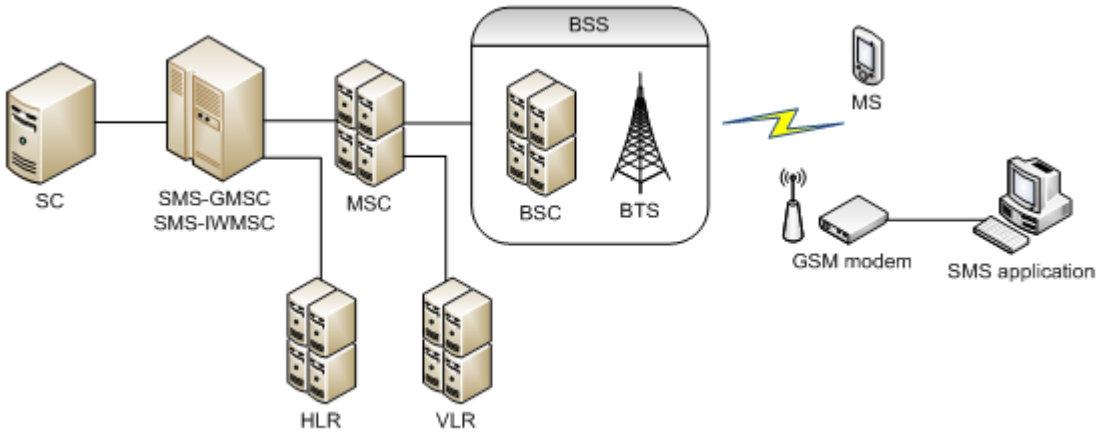


Figure 2.4 SMS platform implements GSM modem.

3. GSM MODEM AND AT COMMAND

In Chapter 2, the GSM modem has been involved in order to accomplish the sending and receiving of SMS. Consequently, some basic concepts and operation of GSM modem will be presented in this Chapter.

3.1 Introduction to GSM modem

GSM modem is a kind of modem which can work under the GSM system. Because most of the clients use the GSM modem for sending and receiving SMS or other kind of short messages, the GSM modem can also be called as SMS modem.

The form of GSM modem can be external devices, such as the SIEMENS MC35, or a PCMCIA Card which can adapted to the laptops. Typically, an external GSM modem always connects to a computer through a serial cable or a USB cable.

The core module of GSM modem is just what we use in the mobile phones actually. As the biggest manufacturers of GSM modems, WAVECOM and SIEMENS are also the mainly module providers of mobile phones. After inserted the SIM card, the GSM modems can startup with its software and get connection to the GSM networks which are provided by the GSM ISPs. The management of GSM modem is generally done by the computer with AT commands, by using the hyper terminal software. Besides sending and receiving SMS, the GSM modems can also be used for voice call or dealing with fax. (Developers' Home 2/2008)

GSM modems can be divided into many types by the difference of interfaces, such as Serial-port GSM modem, USB GSM modem, PCI card GSM modem and Ethernet interface GSM modem. (88soft.cn [referred 11.4.2009])

There are mainly two ways of implementation of GSM modems. The first method is using AT commands which is adopted in this thesis. This is the most substratal way of developing the GSM modem. No matter which kind of modem is being adopted, the mechanism is using Serial-port communication, inputting the AT commands and getting reply from the GSM modem. The second way of GSM modem development is implementing the GSM toolkit or SDK provided by software developers. For GSM or especially the SMS implementation, developers can just employ the APIs of toolkit which are still based on the AT commands to realize the expected functions.

3.2 SIEMENS MC35 GSM Modem

In this project, the role of GSM modem is acted by SIEMENS MC35 terminal, which can work in both GPRS and dual-band GSM channels.

The MC35 Terminal (Figure 3.1) is a compact GSM modem for the transfer of data, voice, SMS and faxes in networks. This terminal contains a robust unit that can be integrated via standard RS-232 interface into the most diverse applications. With the built-in SIM card reader and the plug-in & play features of this terminal, developers can reduce the development and implementation times. Moreover, the product is able to offer some other benefits such as permanent online connection and cost efficient data transmission. (SIEMENS [referred 11.4.2009])

The functionality of the Terminal corresponds to the features of the MC35 module, to which a SIM card reader, an RS232 interface, an analog interface for the handset interface and a greater supply voltage range have been added.

Figure 3.1 SIEMENS MC35 Terminal.

The whole MC35 terminal is constituted by the main body part, a RS232 serial cable, an antenna and the power supply.

3.2.1 The Main Body

The main body of the MC35 terminal is a compact box which integrated all the external interfaces permanently. In addition, the plug-in connections of the main body are standardized and suitable for using under vibration.

As the promotion version of the TC35 terminal, besides the support of GSM 900MHz and GSM 1800MHz system, the MC35 terminal can also operate under the GPRS and relative protocols. Inside the main body box, The Dual-Band GSM/GPRS module MC35 is a super slim and effective communication module for a great variety of applications in telemetry, telematics and telephony.

As Figure 3.2 shows below, there are many interfaces available on the TC35Terminal:

• Connector for the plug-in power supply unit
• Handset connector
• Mini-SIM card holder
• Antenna connector FME
• RS232 interface (V.24/V.28 on the D-Sub socket)

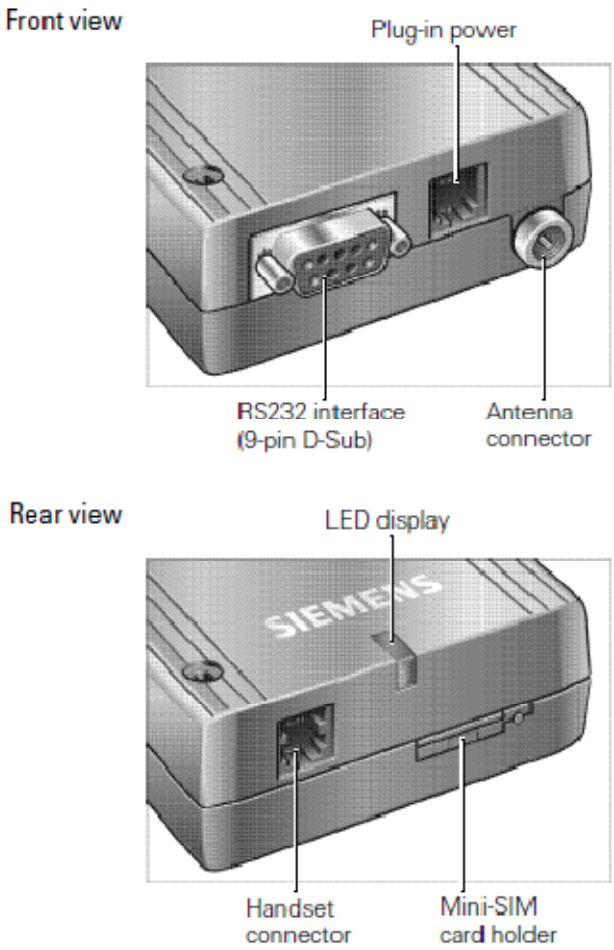


Figure 3.2 The overview of MC35 terminal.

3.2.2 Siemens MC35 GSM Modem Status

After the power supply is connected, the MC35 terminal is turned on automatically. In the middle of the rear view, there is a LED display for users to monitor the working statues of this modem. The operating states are listed in Table 1.

Table 1 The LED displays the following operating states of the MC35 terminal.

Operating State	LED display
After connecting the plug-in power supply unit	On for 2 s
POWER DOWN	Off
Network search No SIM card is inserted No PIN is entered (via PC) No GSM network is available	Flashes rapidly
STANDBY (registered in the network)	Flashes slowly
Connection TALK (Voice Call or Data Call)	On
GPRS data transfer	Flashing

The MC35 terminal is in the POWER DOWN state when the operating voltage for the GSM module is disconnected. In this situation, the software of the GSM module is inactive.

A transition to the POWER DOWN state always occurs in the following circumstances:

• When the Terminal (module) is shut down by means of AT commands (e.g. sleep mode).

• When the external reset line is active. (SIEMENS [referred 11.4.2009])

In the network search state, the MC35 modem searches for the GSM or GPRS network. All the components in the GSM Terminal (module) that are not required are shut down in several stages by the auto-running energy-saving software. Power consumption in this state depends on the current network availability.

This state is reached:

• From the POWER DOWN state: by an active ignition (on) signal at the power supply connection;

• From the TALK or STANDBY state: when the network is lost (out of range).

The MC35 Terminal is ready to send and receive and registered in the network when it is in the STAND BY mode. Paging is performed with the GSM network in order to obtain synchronization with the GSM network.

3.2.3 Interfaces and Connected Devices

As mentioned before, there are many interfaces which can get connection to other devices on the main body of the MC35 terminal. In this section, the interfaces and the plug-in devices will be presented.

The MC35 modem gets the power supply in a wide voltage range (+8 V ... +30 V) via a particular plug-in power supply connectors (Figure 3.3). Two extra control lines are used for switching the Terminal on/off (resetting). A 6-pin Mini-Western connector implements the power supply interface. (Figure 3.4) (SIEMENS [referred 11.4.2009])

Figure 3.3 The power supply unit.

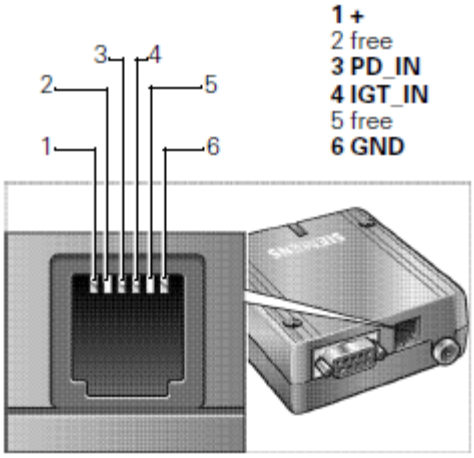


Figure 3.4 Connector for the power supply unit.

As we know, for the GSM modem, we have to insert a GSM card to get GSM services provided by the network service providers. So the SIM card connector is designed for 3V SIM cards in accordance with GSM11.12 phase2 to operate the MC35 GSM modem. Before insert the card, users have to make sure that there is no voltage applied to the MC35 terminal, which means the insert operation of SIM card must be done in the no-power condition.

In order to enhance the transmitting power and the range of the communication signal, an antenna with dual band (GSM 900/1800) can be connected to the RF interface which is implemented as a 50W FME (male) coaxial jack. (Figure 3.5) (SIEMENS [referred 11.4.2009])

Figure 3.5 Antenna used to connect the MC35 terminal.

At last, the mainly part which is related to the transmission with the PC ends is the RS232 Serial cable (Figure 3.6). Through the serial link, the MC35 can receive AT commands sent by PC to get controlled. Besides, when the MC35 modem gets any information or feedback, it can send them back to the PC ends. The RS232 interface is implemented as a 9-pin D-Sub socket with a screw fitting.

Figure 3.6 RS232 serial cable.

3.3 Control of SIEMENS MC35 GSM Modem

The customer application communicates with the MC35 Terminal or the MC35 GSM module by means of AT cellular commands. After get connection to the Serial port of a computer or a virtual COM port when the connection is done by USB connector, the PC ends can use some terminal programs to control the MC35 terminal. Such as Hyper Terminal which is provided by the Windows OS environment or an open source software called 'PUTTY'. In my project, the control of the modem has to be done by the application automatically, so the communication of the serial link must be controlled by Win32com API.

3.3.1 Introduction to AT commands

The birth of AT commands is related to control of modems in the early time. In 1977, in order to send control commands without changing the pins of devices' data ports, Hayes Communications introduced the Hayes AT commands set to help the switch of modem

devices between Data mode and Command mode. The MC35 GSM modem mainly adopts the Hayes AT commands set with some minor distinctions related to the manufactures. (Wikipedia 2009)

In the Hayes AT commands set, “AT” or “at” always precedes a command. The modem needs these two letters to adjust the transmission to the settings of the PC’s COM port. The modem then switches automatically to the defined baud rate and parity of the serial port, until the next command is coming or the modem is turned off. Any commands not started by an AT would not be responded by the modem.

AT commands set plays an important role in communication of GSM modems. Many operations can be done by using AT commands, including GPRS communication, dialing phone number, control of buttons and fax etc.

There are three types of AT commands. The first is called Test Command, by adding “=?” after the command, the modem can reply the whole parameter set which is supported by this command. The second one is Read Command. It consists of an AT command followed by “?”. Users can type Read commands to get the current settings of the modem. Execute Command is the last one, which is followed by “=” and its parameters. Sometimes the Execute Command does not have parameters, so it can be executed directly.

3.3.2 Some Basic AT Commands for SMS

There are many kinds of AT commands designed for different purposes, such as for phone-call control, network services, security, SMS, data transmission, fax and serial port settings. In this research, I will concentrate to the functions related to SMS. Some basic AT commands for sending and receiving of SMS will be presented in the following section.

The MC35 terminal supports two types of SMS, TEXT mode and PDU mode. The TEXT mode is easy for use, and all the SMS discussed in this research is under the TEXT mode. PDU mode asks for coding and decoding, supports more characters. In some languages, such as Chinese or Japanese, the alphabet provided by basic TEXT mode can not fulfill the requirements, so the PDU mode is needed.

By typing the AT command showed below, we can switch the SMS of MC35 modem into TEXT mode:

AT+CMGF=1;

In order to send a short message, four steps have to be done. At first, make sure the MC35 terminal connecting well, by typing the command “AT” and getting an “OK” as reply. After that, enter the PIN code of the SIM card to make it active. An “Error” can be returned if the SIM card has already been activated before. The third step is to set the SMS type to TEXT mode. Finally, users can send messages by input the command “AT+CMGS= xxxx” and press “Enter”. When the sign “>” shows, the content of the message can be typed in. Afterwards, hit the “Enter” and “ctrl+Z” to send the message out. A feedback “+CMGS:” will be returned if the sending succeeds. (Figure 3.7)

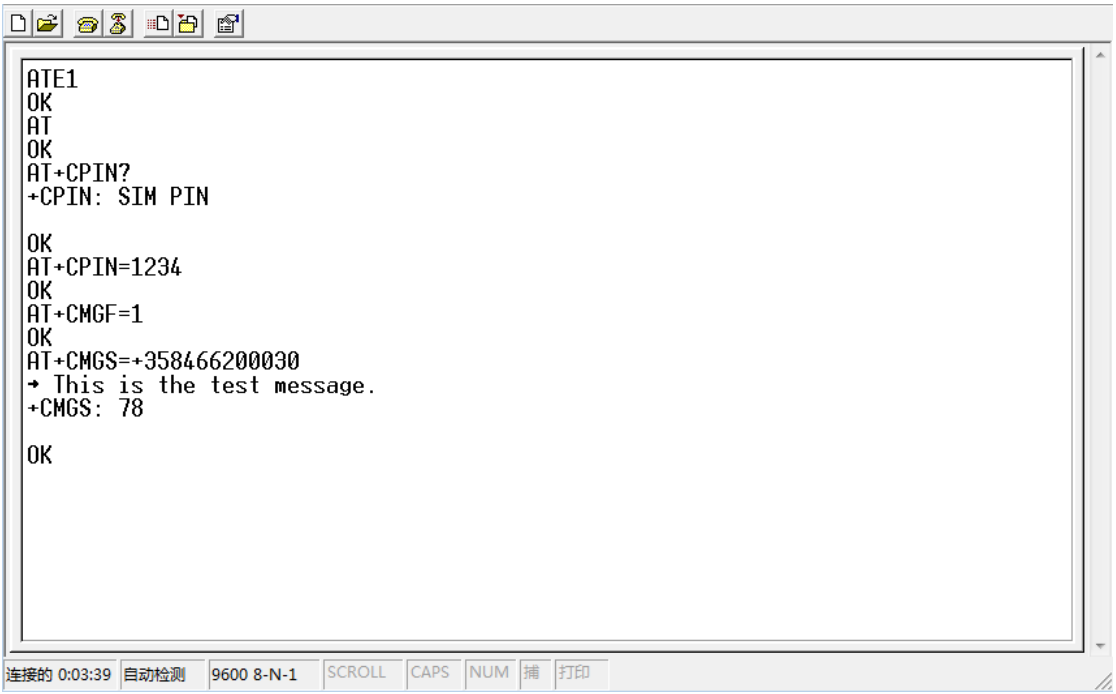


Figure 3.7 Sending a SMS message by using AT commands.

4. SERIAL PORT PROGRAM AND SMSLIB

In Chapter 3, I talked about how to send messages by implementing the AT commands. However, during the business procedure, we can not use the hyper terminal applications to finish the input of AT commands. Thus, such works have to be done by the Serial Port program. In the following chapter, two ways of realizing the Serial Port programming and AT commands programming will be presented. The first method is JAVA serial port programming based on Java Communication API. The other way is to implementing the interfaces supported by SMSLib.

4.1 Serial Port Programming

In this final thesis project, I will use the JAVA JDK 1.6.0 as the development environment. The Java Communications 3.0 API (javax.comm) is a Java extension package that facilitates the development of serial port and parallel port communication applications based on external devices such as modems, Smart Cards, embedded systems, and point-of-sale devices, financial services devices, fax, display terminals, and robotic equipment. (Sun [referred 20.4.2009])

The Java Communications API (denoted as javax.comm) provides applications access to RS-232 hardware (serial ports) and limited access to IEEE-1284 (parallel ports), SPP mode.

Implementations of the API are currently available for Windows 32bit, Solaris x86, and Linux x86. The javax.comm supports interfaces allow administers to specify the locations of ports as well as their visibility, names, and in some cases annotated reference.

There are three levels of classes in the Java communications API:

A. High-level classes like CommPortIdentifier and CommPort manage access and ownership of communication ports.
B. Low-level classes like SerialPort and ParallelPort provide an interface to physical communications ports. The current release of the Java communications API enables access to serial (RS-232) and parallel (IEEE 1284) ports.
C. Driver-level classes provide an interface between the low-level classes and the underlying operating system. Driver-level classes are part of the implementation but not the Java communications API. They should not be used by application programmers.

The javax.comm package provides the following basic services:

A. Enumerate the available ports on the system. The static method CommPortIdentifier.getPortIdentifiers returns an enumeration object that contains a CommPortIdentifier object for each available port. This CommPortIdentifier object is the central mechanism for controlling access to a communications port.
B. Open and claim ownership of communications ports by using the high level methods in their CommPortIdentifier objects.
C. Resolve port ownership contention between multiple Java applications. Events are propagated to notify interested applications of ownership contention and allow the port's owner to relinquish ownership. PortInUseException is thrown when an application fails to open the port.
D. Perform asynchronous and synchronous I/O on communications ports. Low-level classes like SerialPort and ParallelPort have methods for managing I/O on communications ports.
E. Receive events describing communication port state changes. For example, when a serial port has a state change for Carrier Detect, Ring Indicator, DTR, etc. the SerialPort object propagates a SerialPortEvent that describes the state change.

4.1.1 The Installation of JAVA Communications API

Before start the serial port programming in JAVA environment, we have to configure the extension package of javax.comm to the right place to support the library of JAVA environment. Moreover, the com port drivers classes for the operation system (especially the windows 32bit in this project) should be configured right. Otherwise, the compiler of the development environment will block the running procedure.

At first, we have to download the package of JAVA Communication API from the official website supported by Sun. (https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_SMI-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=JAVACOMM-3.0.1-LX-SP-G-F@CDS-CDS_SMI)

There are three files in the package:

File Name	Type
comm	Executable Jar File
javax.comm.properties	Properties
win32com	*.dll

To install it, unzip the downloaded archive package in a temporary place and do the following copies:

- File comm.jar should go under JDKDIR/jre/lib/ext/
- File javax.comm.properties should go under JDKDIR/jre/lib/
- Library files (i.e. win32com.dll for Win32 or the .so Linux library files) should go under JDKDIR/jre/bin/

If the system being used has a separate JRE directory, do the same copies for the JREDIR directory.

For this final thesis project, I used the MyEclipse 6.6 as the development environment. For the compilation process, the configuration of Java build path is needed, because most development environments contain their own development toolkit and have a default Java build path.

For example, I installed the JAVA Communication API to the JDK 1.6.0 directory. Before starting the serial port programming, I have to change the Java build path to this JDK. When starting a new project, we can change the Java build path to a project specific JRE (Figure 4.1). If the Java build path is not set to be the directory which already installed the JAVA Communications API, errors will be warned during the compilation process.

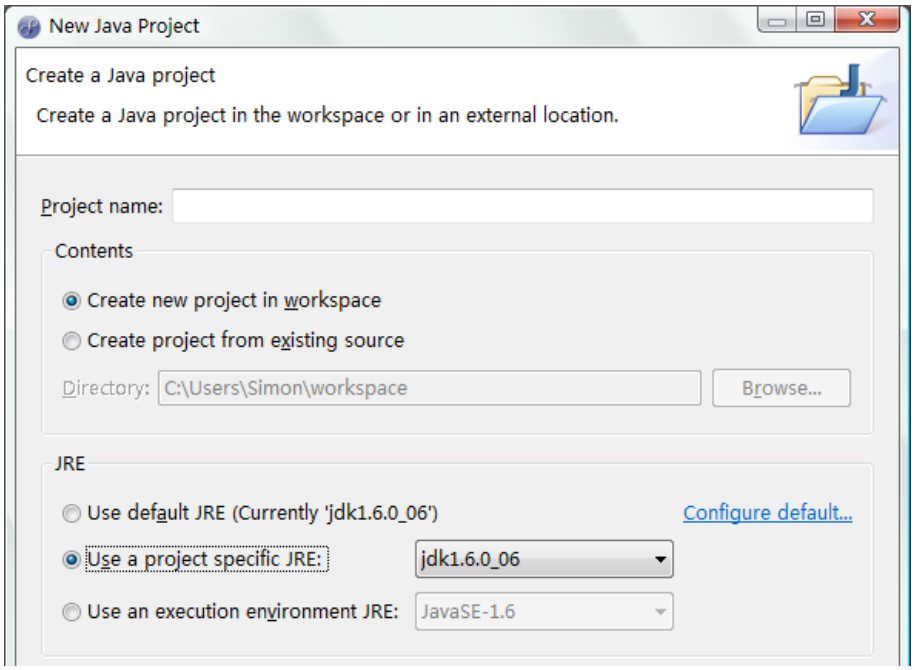


Figure 4.1 Use a project specific JRE.

4.1.2 Basic Serial Port Programming based on JAVA Communications API

In the package of Java Communication API, there are some samples for developers to learn how to use the javax.comm. In this section, I will talk about the basic idea and procedure of serial port programming.

When using the JAVA Communications API to control the SIEMENS MC35 terminal, there are three levels of work need to be done (Figure 4.2). The basic layer is Serial port driver, which is already integrated by the Javax.comm package. This layer takes the responsibility of the initialization of serial port, configuring the parameters, such as port number, bit rates and parity. The upper layer is the Serial port programming, developers have to write programs to control the PC to read from or write data to the serial port, even the external devices. Some basic methods are provided by the Javax.comm to write and read data in bits, in addition, the API also offer a serial port listener class for monitoring. The layer staying on the top is AT commands programming. In the communication between PC and GSM terminals, AT commands must be input sentence by sentence, in a logical way, and a reasonable interval is necessary. Consequently, the developer has to design a procedure of AT commands input.

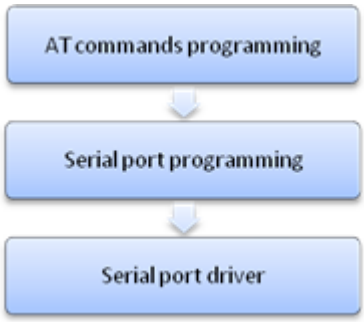


Figure 4.2 Three levels of work in Serial port communication.

There are three necessary steps in the serial port communication, namely initialization of serial port, reading data from the serial port and writing in bits. To initialize a serial port communication which is related to the serial port driver talked above, the following codes can realize this function.

```
    try{
      portId = CommPortIdentifier.getPortIdentifier(PortName);
      //return the identifier which is used to mark the port.
      try{
        serialPort = (SerialPort)
        portId.open("Serial_Communication", 2000);
      //2 seconds of waiting time;
      } catch (PortInUseException e){return InitFail;}
      //Use InputStream in to read from the serial port, and OutputStream
      //out to write to the serial port.
      try {
        in  = serialPort.getInputStream();
        //return the InputStream object that can be used to read from the port
        out = serialPort.getOutputStream();
        //return the outputStream;
      } catch (IOException e){return InitFail;}
      try {
          serialPort.addEventListener(this);
      } catch (TooManyListenersException e) {}
      serialPort.notifyOnDataAvailable(true);
      //Initialise the communication parameters to 9600, 8, 1, none.
      try{
         serialPort.setSerialPortParams(9600,
              SerialPort.DATABITS_8,
              SerialPort.STOPBITS_1,
              SerialPort.PARITY_NONE);
      } catch (UnsupportedCommOperationException e){return InitFail;}
    } catch (NoSuchPortException e){return InitFail;}
```

A port identifier is needed when building a serial port session. After that, we can use the method “open” to try to start the session. There is an integer type of parameter as the time limitation of building the session. If the time of opening exceeds the number in milliseconds, an exception is thrown out.

In the following, the input stream and output stream should be defined to support the reading and writing actions. Besides, the parameters of the serial port communication have to be decided to regulate the data form sent between the PC and the external devices.

The writing action can be done by the method “write(char c)” provided by OutputStream class. In the codes shown below, the “write()” method can only output the data to the serial port in bits. Thus, when writing a token, developer will need a cycle to execute the writing.

```
public void WritePort(String Msg)
{
  try
  {
    for (int i = 0; i < Msg.length(); i++)
       out.write(Msg.charAt(i));
      //write one character each time
  } catch (IOException e)  {}
}
```

The realization of reading is more complicated, there are many ways to implement this function. The first way to do this is adding a serial port listener to catch the incoming data notification.

```
  try {
      serialPort.addEventListener(this);
  } catch (TooManyListenersException e) {}
  serialPort.notifyOnDataAvailable(true);
```

In the initialization part, add such codes to the structure to add a port event listener to the opening serial port, and turn on the data notification.

```
public void serialEvent(SerialPortEvent event) {
        switch(event.getEventType()) {
        case SerialPortEvent.DATA_AVAILABLE:
            byte[] readBuffer = new byte[20];
            try {
                while (inputStream.available() > 0) {
                    int numBytes = inputStream.read(readBuffer);
                }
                System.out.print(new String(readBuffer));
            } catch (IOException e) {}
            break;
        }
    }
```

By implementing the method of serialEvent, developer can catch the data available notification and control the reading.

The reading of the Serial Port can also be done by a thread class. The thread takes the response of fetching data in bits from the serial port and put the bits into a buffer. The main program can read this buffer in two ways. One is setting a fixed length of bits, when the length of the buffer reaches the limit, the data will be transmitted as a String. The other way is setting a default interval. For example, the interval of reading data is 1000 millisecond, which means the main program will try to get the bits read by the thread per second.

All the methods have their flaw in some aspects. The way of implementing serial port listener is depends on the serial port driver. During the development, I found that for different kinds of USB/RS232 connector, the times of receiving data incoming notification are unstable. Moreover, the available() method of InputStream class can not return a creditable value.

The fixed length reading is not compatible with the SMS receiving, because the length of the incoming short messages is unpredictable. The method of setting interval may waste some time.

4.2 SMSLib

In the previous section, I talked about the basic serial port programming based on JAVA Communication API. In this final thesis project, I intend to involve a compact JAVA library called "SMSLib" to implement all the functions related to SMS sending and receiving.

4.2.1 Introduction to SMSLib

SMSLib is a Java library developed by Thanasis Delenikas, which allows developers to send and receive SMS messages via a compatible GSM modem or GSM phone. SMSLib also supports some bulk SMS operators (for outbound messaging only). SMSLib is an open source JAVA library project founded in Google Code Community, and the official website is http://code.google.com/p/smslib/.

SMSLib provides a lot of interfaces to assist the developer to control the GSM modem. There are some interfaces focusing on device initialization, and some devoting to control the sending and receiving SMS messages. As mentioned in last section, the work of serial port communication with GSM terminal can be divided into 3 levels (Figure 4.3). The SMSLib combines all the three levels of work into it, and provides interfaces for each level. It means developers do not need to worry about the operation of AT commands and the reading of serial port.

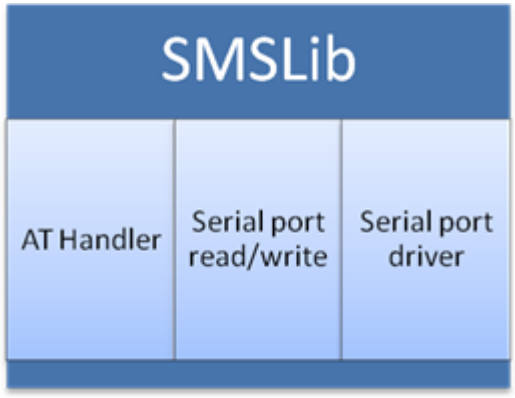


Figure 4.3 The SMSLib combines all the three levels of work

SMSLib does not only provide interfaces to take charge of sending and receiving SMS messages, but also possesses some features which are relevant to SMS, such as bulk message server connector, message encryption and phonebook. Some main features of SMSLib are shown as following:

- Supports GSM phones and GSM modems connected via serial port interfaces or IP interfaces.
- Works with PDU/TEXT protocols.
- Supports Inbound & Outbound simple text messages.
- Works with 7bit, 8bit and UCS2 (Unicode) message encodings.
- Supports Inbound & Outbound big (multipart) messages.
- Flash messaging.
- Outbound messages with port information / addressing.
- Outbound WAP PUSH SI messages.
- Status (Delivery) Report messages.
- Basic GSM information available: Modem, Manufacturer, S/W revision, Signal level, etc.
- Sending priorities.
- SMSLib gives you the option of defining callback methods which it will call when certain events occur
- Supports bulk SMS operators.
- SMSLib can send/receive encrypted messages. The current encryption engine uses the JDK AES strong key (128bit) implementation.

For more information, the wiki page of SMSLib may help a lot. (http://code.google.com/p/smslib/wiki/)

As we know, for different kinds of GSM terminal, the AT commands system may have some minor differences. Consequently, the SMSLib is not available on every GSM terminal. The following models are reported to be compatible:

- Billionton: PCMCIA PCGPRSQ-B.

- EagleTec: GSM modems.
- Fargo Maestro 20.
- ITengo: 3000, WM1080A.
- Janus: GSM864Q.
- Nokia: 6070, 6100, 6210, 6310, 6310i, 6230, 6230i, 6681, 8250, 8310, 6610, 6800, 7210, 6810, 7250i, 6103, 6020, 3220, 6822, 5140, 5140i, 30 (terminal).
- Multitech: Multimodem GPRS (SMSLib for Java can also drive the Multitech Multimodem via its IP port).
- Sharp: GX30, GX32.
- Siemens: MC35i, M35, M50, M65, C45, TC35i, C65, M55, TC65t.
- SIMCOM Ltd: SIMCOM_SIM100S.
- Sony Ericsson: K300i, SE K800i, K700i, K750i, SE W850i, W880i, GC89, Z550a, W800, W580i, W810, i320, GT48.
- Ubinetics: GDC201.
- Wavecom: M1206B, M1306B, WMOD2 Wismo, Fastrack Supreme 10, WISMOQCDMA CDMA.
- Huawei: E220 (may require the forced setting of SMSC address).
- Motorola: V3.
- Teltonika: ModemUSB.
- Motorola: V3, L6.
- Samsung: D520.
- Samba: 55-SET GSM/GPRS USB modem.

The following models are reported to be incompatible:

- Nokia: All Symbian models (they partially work or don't work at all).
- Nokia: 6131.
- Blackberry 8700g.
- Nokia 5610 XpressMusic.
- Motorola G18 GSM Modem.

4.2.2 Installation of SMSLib

Although the SMSLib is quite convenient for SMS developers, but the installation procedure is not easy to finish. The SMSLib contains a variety of functions, and also need many libraries and APIs to support it.

To use SMSLib, the developer has to install the following components to the developing environment:

- Sun JDK 1.5 or newer.
- Java Communications API.
- Simple logging facade for Java (slf4j).
- Apache Jakarta Commons - NET.

Starting from SMSLib v3.4, the installation procedure has changed a bit. Some SMSServer components that required extra dependencies have been pulled out of the main source tree. This was done in order to reduce the required third party libraries and dependency files for those wanting to use the core library.

In last section, I have already talked how to install the JAVA Communication API, so the process of installation will be omitted in this part.

Simple logging facade for Java (slf4j)

Java logging facade (slf4j) acts as the middle layer between an application and the various logging systems. slf4j can be downloaded for free from http://www.slf4j.org/download.html. There are lots of jar file in the slf4j package, and the slf4j-api-1.5.6.jar file (1.5.6 was the current version at the time of this writing) is needed. Add this file to somewhere in the classpath or in Java's lib/ext directories, together with one of the following files, depending on which logging system is in use:

Logging System	Required .jar
JCL	slf4j-jcl-1.5.6.jar
Apache log4j	slf4j-log4j12-1.5.6.jar
JDK logging	slf4j-jdk14-1.5.6.jar

Apache Jakarta Commons - NET

Developer can download the Apache Jakarta Commons/NET from http://commons.apache.org/downloads/download_net.cgi. The developer need to add the commons-net-2.0.jar file (2.0 was the current version at the time of this writing) to somewhere in the classpath or in Java's lib/ext directories.

For example, I am using the JDK 1.6.0 as the JAVA builder, the JAVA build path is C:\Program Files\Java\jdk1.6.0_06\. Thus the jar files mentioned above should be added to C:\Program Files\Java\jdk1.6.0_06\jre\lib\ext.

4.4.3 Basic Implementation of SMSLib

There are two examples ReadMessages.java and SendMessage.java located in the package folder (src/java/examples/modem) to assist developers to find out the basics of how to read / send messages via the GSM terminals.

Before dealing with the programs, there are some concepts need to be defined. In SMSLib version 3.4, SMSLib has introduced the concept of the Gateway, which is an interface to a device or service that can send and receive SMS messages. A gateway could be a GSM modem or a supported bulk SMS provider. SMSLib can handle multiple gateways at the same time. Accordingly, when using an external GSM modem, we have to define a gateway in our program to set the parameters of it.

In the SMSLib, Service is also a very important concept. If we can regard the gateway as the virtual model of our GSM device, then the Service can be defined as the operation logics of the device. The Service takes charge of the sending and receiving of SMS messages, the controlling of gateway, the handling of callback methods and deciding the start or stop of the service.

Now, I will present some explanation on the sending and receiving programs, and such programs play essential roles in my final thesis project. The following codes are a part from the sample called “SendMessage.java”.

```
Service srv;
OutboundMessage msg;
OutboundNotification outboundNotification = new
OutboundNotification();
srv = new Service();
SerialModemGateway gateway = new SerialModemGateway("modem.com1",
"COM5", 9600, "SIEMENS", "MC35");
gateway.setInbound(true);
gateway.setOutbound(true);
gateway.setSimPin("1234");
srv.setOutboundNotification(outboundNotification);
srv.addGateway(gateway);
srv.startService();
System.out.println("Modem Information:");
System.out.println("  Manufacturer: " + gateway.getManufacturer());
System.out.println("  Model: " + gateway.getModel());
System.out.println("  Serial No: " + gateway.getSerialNo());
System.out.println("  SIM IMSI: " + gateway.getImsi());
System.out.println("  Signal Level: " + gateway.getSignalLevel() +
"%");
System.out.println("  Battery Level: " + gateway.getBatteryLevel() +
"%");
```

We can find that at the beginning, developers have to define a Service and initialize a gateway. When initializing the gateway, the ID of the gateway, the serial port and the bitrate must be decided as parameters.

Afterwards, the Inbound() and Outbound() of the gateway must be enabled to achieve the sending and receiving. The method “setSimPin()” is used to enable the SIM card if there is a PIN code authentication.

The gateway must be added to the service to support its work. In addition, by calling some methods such as getManufacturer(), the developer can get some information about the device in use.

By using the method shown above, I implemented a connector of the MC35 GSM terminal for my application. After setting the parameters of the connection (Figure 4.4), we can connect to the device and the information of the device will be shown (Figure 4.5).

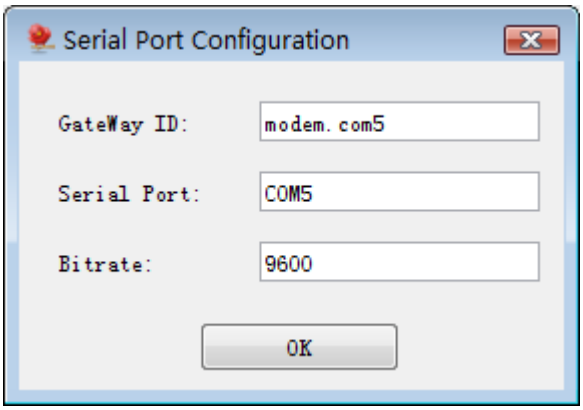


Figure 4.4 Connection settings.

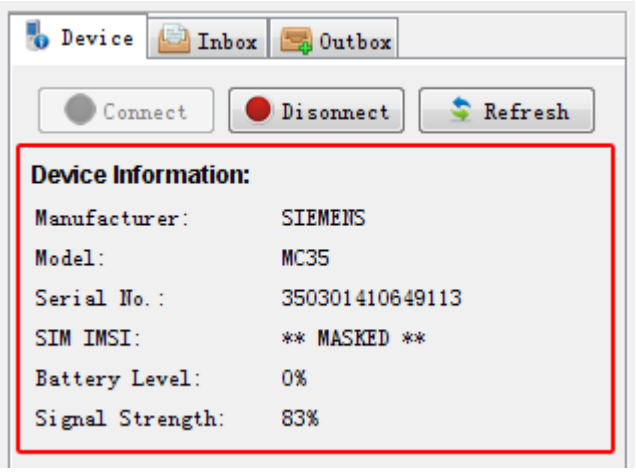


Figure 4.5 The information of device connected.

SMSLib gives you two options for sending messages. Sending a message synchronously means when you call SMSLib to send a message, your thread blocks until the message is sent or the sending fails. The asynchronous way means when you call SMSLib to send a message, your thread does not block and SMSLib returns immediately. SMSLib queues your message(s) and sends them in the background. You may choose to get notified about the fate of your messages.

In the codes shown below, we can find that after initializing the SMS message, the action of sending message is done directly by the service, so it is a synchronous way.

```
// Send a message synchronously.
msg = new OutboundMessage("+358466200030", "Hello from SMSLib!");
srv.sendMessage(msg);
```

If the developer wants to send message automatically and in the background, then using the asynchronous way is a good choice. After the initialization of the messages, the message is added to a queue of outbound messages. Moreover, if the priority is set, the sequence of sending can be changed. The message with higher priority will be sent first.

```
// You can also queue some asynchronous messages to see how the callbacks are
called...
msg = new OutboundMessage("+309999999999", "Wrong number!");
msg.setPriority(OutboundMessage.Priorities.LOW);
srv.queueMessage(msg, gateway.getGatewayId());
msg = new OutboundMessage("+308888888888", "Wrong number!");
msg.setPriority(OutboundMessage.Priorities.HIGH);
```

In my project, I used the asynchronous sending as the method of sending message. After a message is composed by administrator (Figure 4.6) or auto-generated by the system, the message will be added to the queue of sending. The message will be shown in a out-box monitor if the sending succeed (Figure 4.7).

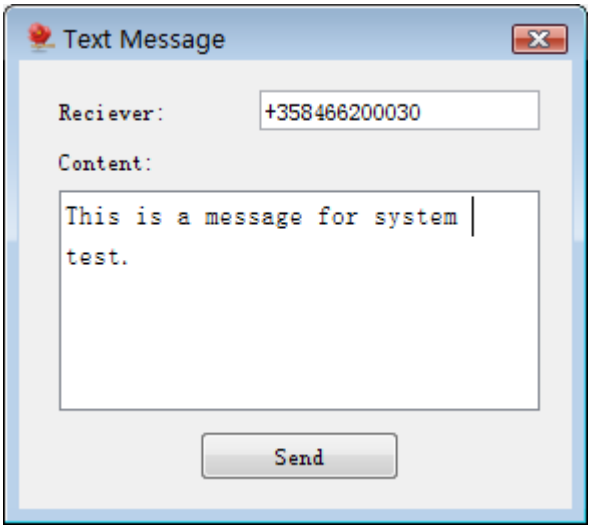


Figure 4.6 The dialog for composing a text message.

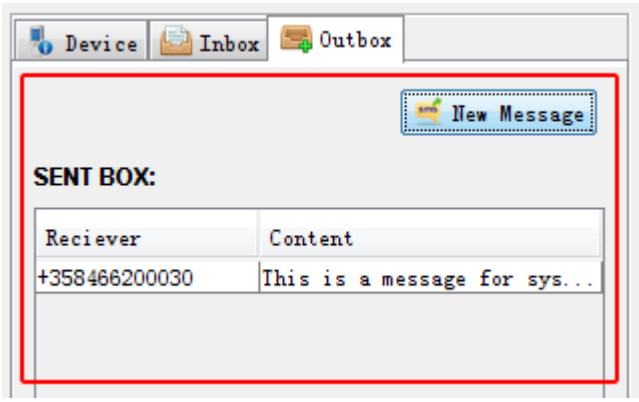


Figure 4.7 Outbound messages will be shown after sent.

The reading of the messages can be realized by the following codes. The method readMessage() can get messages to a list and then the system can print them out.

```
//Read Messages. The reading is done via the Service object and affects all Gateway objects defined. This can also be more directed to a specific Gateway
msgList = new ArrayList<InboundMessage>();
this.srv.readMessages(msgList, MessageClasses.ALL);
for (InboundMessage msg : msgList)
    System.out.println(msg);
```

If the InboundNotification have been added to the service, then the developer can get informed by the notification if there is upcoming message. By implementing the process() method, the developer can decide how to handle the message, such as print it.

I extract two parameters from the Inbound Message notification, namely the text and the originator of messages. The information will be shown in the in-box monitor when a message is received. (Figure 4.8)

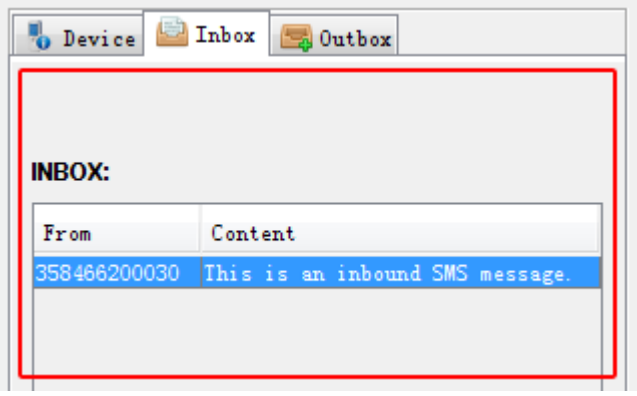


Figure 4.8 The inbound message will be shown in the inbox monitor.

5. DESIGN AND IMPLEMENTATION OF TRANSACTION

As we know, a mature e-business service or software should experience many steps in the design and development, such as requirements analysis, high-level design, detailed design and technical design (Carter, 315-318). Moreover, after the implementation of the system, an integrated testing is needed to verify and validate the functions of the system. In this chapter, the transaction model of a basic SMS ticket booking system will be discussed. In addition, I will present some details of implementation of the transaction. Finally, a demonstration of the realization of the transaction flow will be shown.

5.1 Transaction Model of SMS Ticket Booking

The successful design of an e-business takes several steps. Before the design of the transaction model of our SMS ticket booking system, we have to make requirements analysis of the SMS ticket booking, in order to know what is needed by the customers and which function is useless or not easy to be implemented. Afterwards, we need to do the transition from analysis to design, take one step forward to the implementation of the system.

5.1.1 Requirements Analysis of SMS Ticket Booking

In this section, I will derive the requirements of the Ticket-Booking System (denoted as TBS). For that, we first review the user requirements, application developer requirements and requirements of other module depending on the TBS. From these, we derive our own set of requirements.

Requirements include:

- High-Level Description of Functionality
- Data Constraints
- Data Operations
- Interface and Reporting Requirements
- Nonfunctional Requirements

The User Requirements and Application Developer Requirements as listed are hold generally for the construction of the Ticket-booking system project. They have been derived from reference of other types of booking system or e-commerce system, and from discussions with my supervisor.

5.1.1.1 Ticket-Booking System Requirements and User Scenarios

This section reviews the user scenarios, and extracts all information relevant to data management and visualization.

High-Level Description of Functionality

HDF1: There is one type of clients, unregistered user. Unregistered user can book tickets from the system. The transaction history of unregistered user will also be logged.
HDF2: The administrator of the system can change data records as described in "Data Operations".
HDF3: All new data must be validated before they are added to the repository or database.
HDF4: The TBS sends the transaction confirmation to the users by SMS. After the payment has been confirmed, the electronic tickets are sent.

Data Constraints

DC1: A single client can book more than one tickets.

DC2: A single client can book only one kind of tickets at once, the booking information will be saved in the sessions.

DC3: Every client has to book at least one ticket to built a successful transaction.

DC4: There are two types of ticket information, Movie and Performance.

DC5: There are two types of transaction, “In Process” and “Complete”. Before the confirmation of payment, the transactions which are under payment will be signed as “In Process”.

DC6: Each transaction will have a transaction number which is created by the system automatically for the user to trail.

Data Operations

DO1: The administrator should be able to review all the transaction information in the repository.

DO2: After a booking process, a transaction is added to the record of transactions automatically.

DO3: The administrator can delete the transaction or edit its status if the payment process failed.

DO4: The administrator can ask the system to sort, search or organize the transaction information by certain criteria: transaction number, user account, type of tickets, time and status of transactions.

DO5: The administrator can create, edit, modify and delete the Movie and Performance information.

DO6: The administrator can ask the system to sort, search or organize the event information by certain criteria: type of event, name, and the date of released. Other detail information of event can also be the criteria, such as director, cast, places and date of event.

DO7: Clients can get the event information directly by sending SMS messages.

DO8: Clients can a can send SMS messages to ask the system to sort, search or organize the event information by certain criteria: type of event, name, and the date of released. Other detail information of event can also be the criteria, such as director, cast, places and date of event.

DO9: After getting the detail information sent by TBS, clients should be able to booking the tickets of this event by sending a confirmation.

DO10: Clients can choose the way to deliver their tickets, by mail, electronic tickets or 2 dimensional barcode for mobile phone users.

DO11: Clients also can send SMS messages to review their transaction history.

5.1.1.2 Requirements to Other Part of the Ticket-Booking System

To meet the requirements identified in the previous section, the work of Ticket-booking system has to be able to utilize other services. Certain functionality and properties are required from these services in order to achieve a successful outcome of the system. These requirements are listed below. I distinguish two requirement categories: compulsory requirements which have to be done and optional requirements which would be nice to have and which would ease the work significantly.

Compulsory External Requirements

CER1: The mobile booking system should be contemporized with the TBS database.

CER2: The TBS should be able to utilize the SMS service to transmit the ticket information and electronic tickets.

Optional External Requirements

OER1: If impossible, the electronic tickets would be sent in the form of 2D barcode.

OER2: During the transmission, it is better to utilize the SMS with encryption.

5.1.2 Design of the Transaction Model and Booking System

The requirements analysis is concerned with identifying requirements. It should identify all potential requirements in an ideal way. Moreover, additional requirements may be identified throughout the life cycle in practice. Although analysis should evaluate the feasibility of satisfying each of requirements, the specific decisions about how the requirements will be met in the future system should be avoided in the analysis.

Design involves creating plans for an application program to meet some of the requirements derived in the analysis. Each type of model focuses on some aspects of the real world and ignores others. Like analysis, the design process can be subdivided for our consideration into various sub processes, such as high-level design, detailed design and technical design.

For time constraints, I simplified the requirements of the SMS ticket booking system to make it easier to be implemented. The part of database query has been eliminated, in order to short the transaction flow and simplify the implementation of database programming. In a real world ticket booking system, there should be a user-query service helping clients to find the tickets which they want. In this final thesis, the user-queried information will be replaced by predigested event information which contains multiple choices.

In this ticket booking system, client is the starter of transactions. To begin a transaction session, the client can send any text to the SMS terminal adopted by the TBS. After receiving the request of booking, the TBS will reply the predigested event information. There are three options provided by the system, the clients may send A, B or C to choose one of the suggested events. The text sent other than these three letters will be ignored and the system will reply a warning message to ask the client to reply in time. If the client makes a choice, the TBS will call the database to get more information about the chosen performance or movie, and send the information to our client. The client is

asked to reply “Y” to confirm that he really wants to book this event. When receiving the confirmation, the TBS will record this order and produce a Ticket ID and send it back as an electronic ticket. Afterwards, the session will be ended. (Figure 5.1)

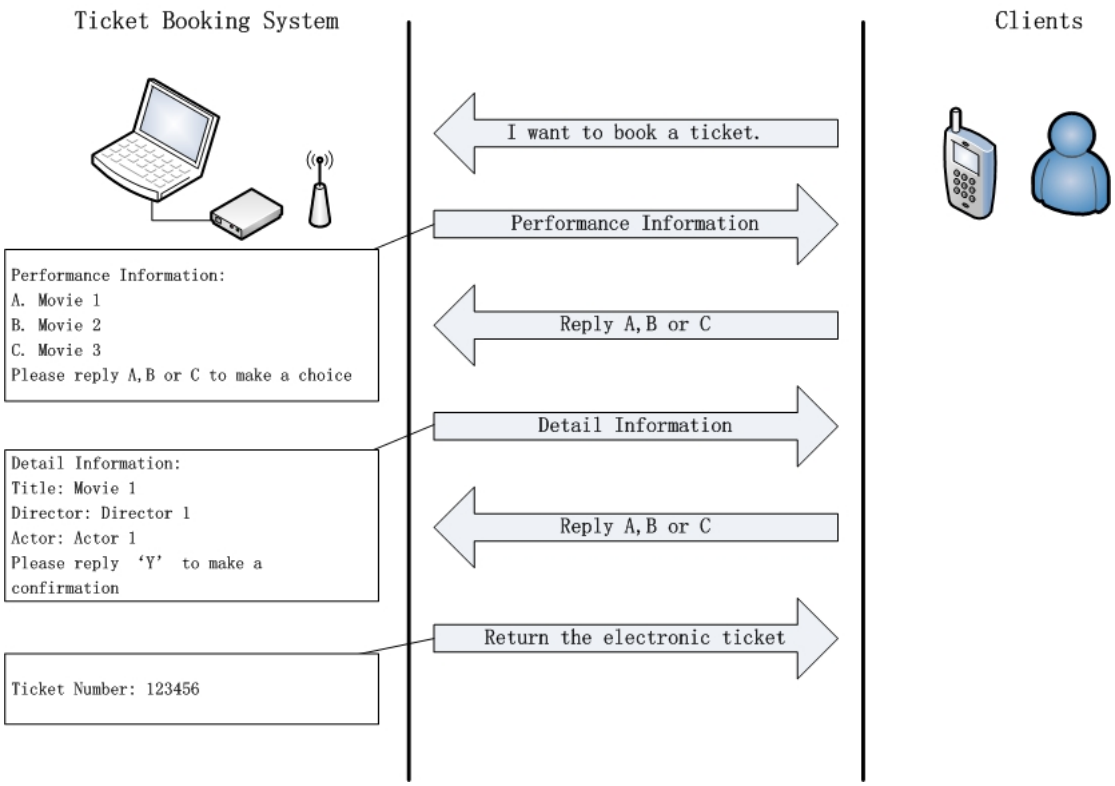


Figure 5.1 The transaction flow of SMS ticket booking system.

In the technical design, there are two main points needed to be concerned about. First is the implementation of database, especially the edit, update and query of database. The second one is the control of session. How to add a session and control the status of sessions is the essential. These two points will be discussed in following sections.

5.2 Implementation of Database

As known, the database is a significant component of information systems. The database provides multiple functions, such as access, edit and update rather than only store information. The design of database plays an important role in the implementation of database, which is aimed at enhancing the speed of execution and eliminating redundancy. Consequently, I will present the design of the database which is adopted by my TBS. Moreover, the implementation method will also be mentioned as following.

5.2.1 Design of Database

As mentioned above, the functions and contents of the database have been simplified to make the whole system works in a straightforward way. In a real world SMS ticket booking system, the database should provide the service of event search. The search of events is related to the query of multi-tables, which takes a lot of workloads of system. Moreover, in order to achieve the search of event, clients have to send more SMS messages, so the transaction flow will be extended. In the final thesis project, the design of the database only reserves the data items which are necessary in a normal ticket booking system.

The transaction of the TBS decides which data is useful or useless. In the transaction flow, the system has to reply event information which contains multiple choices. The event information is only consisted by titles of events, so the title is necessary. Afterwards, the TBS will send the detail information with names of director, actor and actress to the clients when the option is made. Consequently, the names of director, actor and actress should be stored by the database in the same item with the title. For the purpose of query or other operations, event ID is also needed in this table to enhance the searching of data and identifying the uniqueness of data. (Figure 5.2) The event ID is generated automatically by the database, so it should be set as an INTEGER and auto increased.

Table Name: movie_info Database: test Comment: InnoDB free: 4096 kB

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
no	INTEGER	✓	✓	☑ UNSIGNED ☐ ZEROFILL	NULL	
name	VARCHAR(30)	✓		☐ BINARY	NULL	
director	VARCHAR(30)	✓		☐ BINARY	NULL	
actor	VARCHAR(30)	✓		☐ BINARY	NULL	
actress	VARCHAR(30)	✓		☐ BINARY	NULL	

Figure 5.2 Columns defined in the table of event information.

After received the confirmation sent by clients, the TBS will record information of the transaction, and reply a ticket ID automatically. Thus, the table of transaction information should contain the ID of ticket, the event ID and the phone numbers of clients. In addition, I add the name of event into this table for monitoring the insert operation of database. (Figure 5.3)

Table Name: order_info Database: test Comment: InnoDB free: 4096 kB

Columns and Indices | Table Options | Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id	VARCHAR(20)	✓		☐ BINARY	0	
no	VARCHAR(45)	✓		☐ BINARY	0	
name	VARCHAR(45)	✓		☐ BINARY	0	
phone_num	VARCHAR(45)	✓		☐ BINARY	0	

Figure 5.3 Columns defined in the table of transaction information.

5.2.2 How to Implement the Database

In my final thesis project, I utilize MyEclipse as the development environment. To make the database compatible with the IDE, I chose the MySQL 5.0 as the database server of the project. The MySQL database has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. It's used on every continent, by individual Web developers as well as many of the world's largest and fastest-growing organizations to save time and money powering their high-volume Web sites, business-critical systems and packaged software. (MySQL [referred 5.5.2009])

Three steps are needed when setup the database server and development environment. At first, we need to install the MySQL 5.0 on the computer. As we know, the MySQL is open source database software, so it can be downloaded from the official website of MySQL (URL: http://dev.mysql.com/downloads/). The second step is the installation of GUI tools for MySQL. The GUI tools are not compulsory, developers can access to the MySQL database server by using the commands line under the MS-DOS. However, the GUI tools make the access, edit and other operations much easier. In addition, with GUI tools, developer can check the data in a visual form. Finally, in order to implement a database under the JAVA environment, we need to setup the JDBC development environment. The current version of JDK installed on the OS already provides developers with the JDBC package. JDBC includes the java.sql and javax.sql packages, which have the necessary interfaces and classes needed for developing JDBC applications. Besides the JDBC, we also need a specific driver for the connection between JDBC and the database server. Most of the drivers are provided by the providers of database software. For MySQL, developers can download the mysql-connector-java drivers in four versions for free on the official website of MySQL. (URL: http://dev.mysql.com/downloads/connector/j/5.1.html)

5.2.2.1 Introduction to JDBC

JDBC is a JAVA API extended package that used to execute the SQL tokens. It consisted of classes and interfaces that implemented by JAVA languages. Thus, developers can use only JAVA to develop their programs rather than involve other kinds of language.

With JDBC, the use of SQL becomes easier. In other words, developers do not need to write specific programs for the access of different kinds of databases. They can access to database by just the same codes with different kinds of connector drivers.

JDBC extends the ability of JAVA in many scopes. For example, by using JAVA and JDBC, enterprises can derive a website with applets which access to a long-range database. They can also use JDBC to connect asynchronous distributed system through intranet or internet. Along with the popularity of JAVA, more convenient methods of accessing the database are requested by developers and users.
By using the JDBC, developers can achieve three steps of tasks, such as connection to the database, calling the SQL tokens and disposing the query results. (Figure 5.4)

Figure 5.4 JDBC can finish three steps of tasks.

JDBC is a kind of “low-level” interfaces, which means that it can be used to call the SQL tokens directly. Developers can implement higher level interfaces or tools based on the JDBC API. In next section, I will present how to do so.

5.2.2.2 JAVA Database Programming

As mentioned above, the implementation of JAVA database requests a specific driver which is oriented to the JAVA development environment. For that reason, we have to download the mysql-connector-java extended package as the driver for MyEclipse IDE. The current version of mysql-connector-java is 5.1, but for the stability of the system and the reuse of existing codes, I adopt the mysql-connector-java-3.1.14 as the connector driver.

Unzip the package of mysql-connector-java-3.1.14-bin, we can get a executable Jar file. The setup process is quite easy to be finished, just copy the jar file to the JDKDIR/jre/lib/ext. After set the exact build path, then we can start the database programming.

```
Class.forName("com.mysql.jdbc.Driver");
//Load the driver for MySQL.
System.out.println("OK,DownLoad the driver");
Connection con=DriverManager.getConnection
("jdbc:mysql://localhost:3306/test","root","root");
/*  "test" is the database built in MySQL,
    username is root, passwords is root. */
```

The connecting process can be done by the codes shown above. The first sentence is for loading the MySQL-JAVA connector driver. After defining a connection, we can get a MySQL-JAVA connection, which means the program can access the MySQL database directly. The first parameter of the connection is the path of the database. In this example, the database called “test” is located at the port 3306 of localhost server. The following parameters represent the username and passwords.

```
con.setCatalog("test");
Statement statement = con.createStatement();
//define the operation of database.
String sql=new String();
sql=sql.concat("INSERT "+table_name+" VALUES(null");
for(int j=0;j<i;j++){
    sql=sql.concat(",'"+data[j]+"'");
}
sql=sql.concat(")");
System.out.println("sql:"+sql);
statement.execute(sql);
//statement.execute
("insert"+table_name+"values(null,'Movie1','D1','A1','A1')");
```

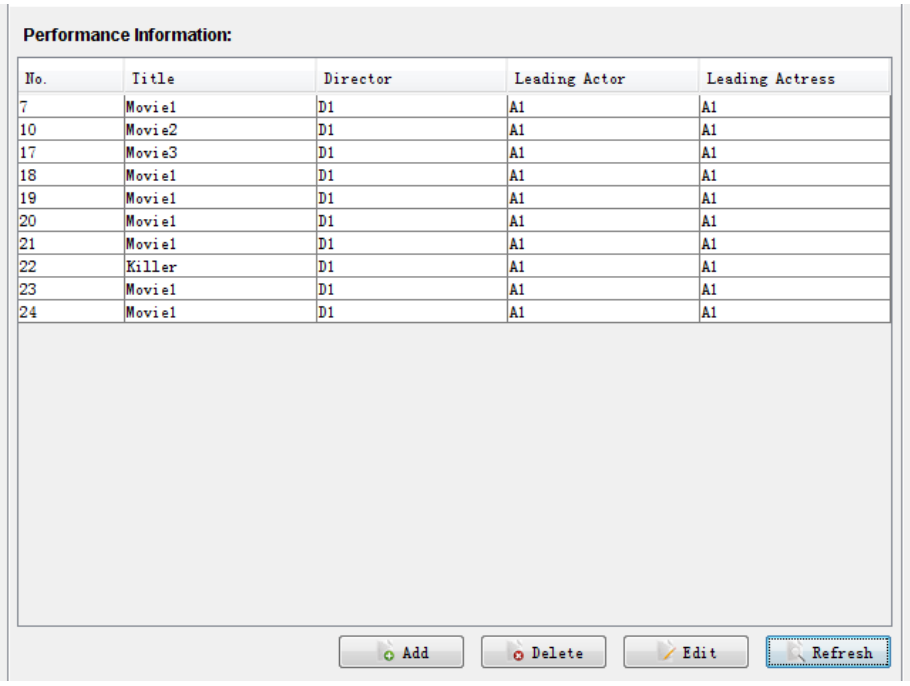
The codes shown above take the response of inserting data into a table. By providing a data set “data[ ]”, we can realize the inserting. For example, in my project, the data for the table of event information could be (null,'Movie1','D1','A1','A1'), the parameter “null” means this data is auto generated by the database. The method execute of a statement can realize the call of SQL tokens.

```
statement.execute
("DELETE FROM "+table_name+" WHERE "+column_name+" = '"+num+"'");
```

For the same rules, other kind of SQL tokens can be executed by the method execute(), such as DELETE and UPDATE.

```
String sql="SELECT * FROM "+table_name;
ResultSet rs = statement.executeQuery(sql);
//get the set of query result
if(!rs.next()){
    System.out.println("Reach the end of the table.");
}
else {
    vector.clear();
    do{
        Vector newdata=new Vector(i);
        for(int j=0;j<i;j++){
        String result=rs.getString(column_name[j]);
        result=new String(result.getBytes("ISO-8859-1"),"GB2312");
        newdata.add(result);
        }
        vector.add(newdata);
    }while(rs.next());
}
rs.close();
con.close();
```

We can realize the query of a table by using the codes shown above. The point is the definition of query result set, which stores the result of query. In the codes, the results are encapsulated into a vector which can be used by the JTable model (Figure 5.5).



No.	Title	Director	Leading Actor	Leading Actress
7	Movie1	D1	A1	A1
10	Movie2	D1	A1	A1
17	Movie3	D1	A1	A1
18	Movie1	D1	A1	A1
19	Movie1	D1	A1	A1
20	Movie1	D1	A1	A1
21	Movie1	D1	A1	A1
22	Killer	D1	A1	A1
23	Movie1	D1	A1	A1
24	Movie1	D1	A1	A1

Figure 1.5 The database is shown by JTable in the application.

5.3 Control of Sessions

In the real world SMS ticket booking system, there should be more than one client connected to the system and dealing with the transactions. In the transaction flow, we can find that the system has to record the status of each transaction and the clients' choices. For that reason, I intend to involve a concept of "session" into the TBS.

The session is an object that represents a transaction of a client. For each transaction, when it begins, the system will allocate a space to form a session which contains the client's phone number and other information such as status and choices.

5.3.1 Data Structure of Sessions

As mentioned in the transaction flow, the session should contain at least four parts of data. At first, the phone number of the client is necessary to identify the owner of the session. Moreover, the phone number can be used to reply the SMS message. Secondly, the status of the session is needed to mark the process of transaction flow. Thirdly, the choice made by client is recorded by the session in order to generate electronic tickets automatically. Finally, the domain of session time is aimed at the control of session-timeout and recycle of resource. In addition, the session time can also prevent DoS attack and transaction jam. (Figure 5.6)

Figure 5.6 Data stored by a session.

In my final thesis project, I use the LinkedList class as the carrier of sessions. The LinkedList is not a data structure which has fast operation speed, but the features of LinkedList are quite suitable for the session control. At first, in the session control process, sessions need to be initialized and inserted into the session set at any-time. Moreover, when a transaction is finished or time-out, the system has to delete this session. In JAVA environment, the program can add the nodes to the list or delete nodes from it at anywhere, anytime. Thus, the system spending is minor when using the LinkedList. Besides, the features of LinkedList also offer flexibility of data operations.

```
LinkedList sessions=new LinkedList();
Iterator it=sessions.iterator();
//new a iterator for traveling the list
boolean num_exist=false;
String number=new String("+"+originator);
while(it.hasNext()){                        //traveling
    SessionNode session=(SessionNode)it.next();
    if(session.phone_num.equalsIgnoreCase(originator)){
        num_exist=true;
    }
}
if(!num_exist){
        SessionNode new_session=new SessionNode(originator);
        sessions.addLast(new_session);
}
```

The codes shown above defined the basic operation of the data structure of sessions. The variable called “originator” represents a new client that just access to the TBS. While receiving the SMS message request, the system will search this number in its set of sessions. If the number already existed in the list, then the system can call the transaction logic to process the session. Otherwise, the system will initialize a new session node and add it to the end of the list.

5.3.2 Transaction Logic

The transaction logic is the essential part of the TBS, it asks the clients to reply the right message at the right status. In the description of transaction flow, I have already discussed the transaction model of TBS. However, the transaction model is still not an ideal model for developers to implement the logical functions of the transactions.

By using the state machine, the transaction logic of the SMS ticket booking system can be described more directly, and it is much easier for developers to implement the codes.

In the transaction logic, the status of session can be represented by four states. The original state (status 0) represents the initialization of the session. If clients send any text messages to the TBS, then the state comes to a mid-state. There are two mid-states between the original state and the terminal state. Status 1 represents the stage of choosing a event, after sending a right choice, it becomes status 2, and otherwise it will stay. The status 2 takes the response of checking confirmation. If a letter "Y" is got, which means the client send a confirmation, and then the status 2 can come to the terminal state. The terminal state (status 3) represents the end of the session, if it is reached, then the session will be terminated. (Figure 5.7)

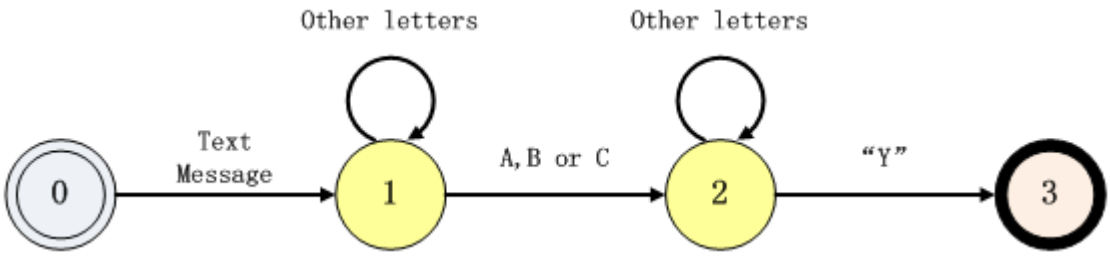


Figure 5.7 The state machine of transaction.

5.3.3 Implementation of Session Timeout

Try to imagine a scene that a client is booking tickets from a system or website by SMS service, what will happen? In the real world, the situation is quite complicated. If the

client is located in a domain with low GSM signal strength, or in where too many subscribers exist, then the SMS message may be delayed or even could not reach the terminal. In that situation, some clients may give up the booking procedure, and the sessions exist in TBS could never be finished. Moreover, we have to prefigure there will be someone try to attack the system by this kind of flaws.

In the experiment environment, it is not a big deal, because there are not many people can begin sessions. However, in the real world, such situation could be a disaster to the TBS. The "dead" sessions can occupy the space of the system, and there is no more space can be allocated to the users who really want to book tickets.

For that reason, the mechanism of session timeout is necessary. That means if a session stays at the same state for a period of time, then it will be judged as a "dead" session and terminated. In the section called "data structure of sessions", I mentioned there is a domain which represents the session time exists in the session node. When the status of a session is being changed, then TBS will refresh the session time (in my project, the time will be set to 0). At other times, I implemented a thread called "timer" to add and check the session time of each session.

As the following shows, I define a class called timer which extends the class of thread. By using the thread, the object of the timer can run automatically under the background. When initialize the object of this class, we can set the interval time (in milliseconds) and the time limit (duration). Any sessions whose session time is over the time limit will be judged as "dead" session and terminated. The interval time sets the interval between two times of session time checking. For example, if the interval time is 10000 and the time limit equals 10, that means the timer will travel the list of sessions and add 1 to the session time of each session. If the session time of a session reaches 10, then the session node will be deleted from the list. After the process, the thread of timer object will sleep for 10 seconds.

```java
public class Timer extends Thread{
LinkedList list;
int duration;
int timeInterval;
boolean enable;
public Timer(LinkedList list,int dura,int interval){
    this.list=list;
    this.duration=dura;
    this.timeInterval=interval;
    this.enable=false;
}
public void setEnabled(boolean e){
    this.enable=e;
}
public void run(){
    while(enable){
        try{
            Thread.sleep(timeInterval);
            changeTime();
        }catch(Exception e){e.printStackTrace();}
    }
}
public void changeTime(){
    Iterator it=list.iterator();
    SessionNode deleteOne=null;
    while(it.hasNext()){                        //traveling
        SessionNode session=(SessionNode)it.next();
        if(session.wait_time<duration){
            session.addTime(1);
        }
        else{
            deleteOne=session;
            System.out.println("Session timeout");
        }
    }
    if(deleteOne!=null){list.remove(deleteOne);}
}
}
```

5.4 Demonstration of Transaction Operation

At the end of this chapter, I want to show the process of transaction operations based on the software of my project.

When a client intends to book a ticket, she or he has to send a request in SMS form to the server of SMS booking system (Figure 5.8).

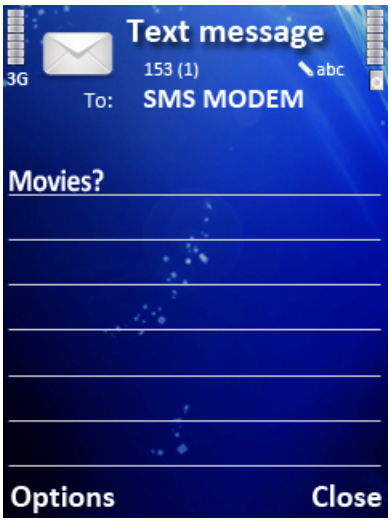


Figure 5.8 The request sent by clients.

At the server end, after getting the request sent by the client, the server queries the database table of event information (Figure 5.9), and then sends a feedback message. (Figure 5.10)

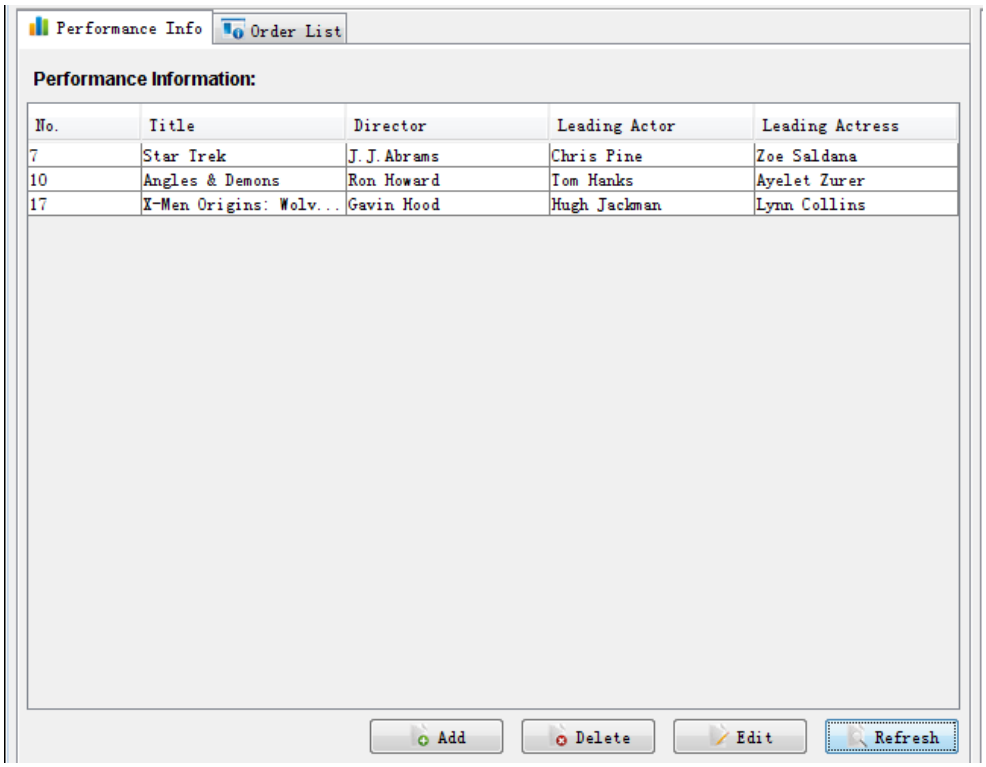


Figure 5.9 The server reads the database.

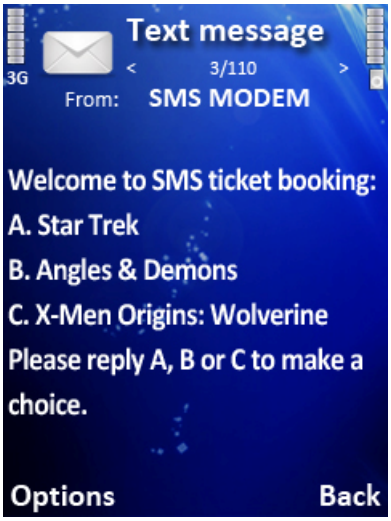


Figure 5.10 The feedback message sent by the server.

The client has to send a message which contains the letter ‘A’, ‘B’ or ‘C’ to make a choice. For example, the client chooses ‘C’ as the choice. (Figure 5.11)

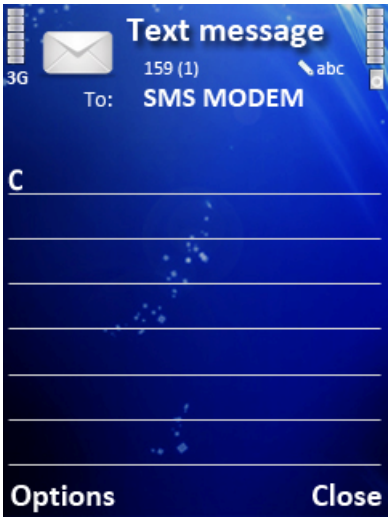


Figure 5.11 The client sends 'C' as the choice.

When the server receives this reply, it will access to the database and fetch more details for the client, and then send the details to the clients in SMS. (Figure 5.12)

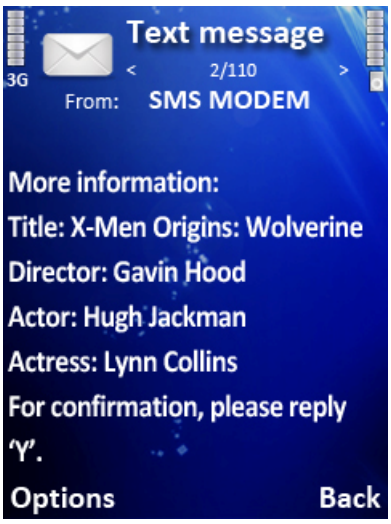


Figure 5.12 The details of choice 'C'.

To make a confirmation for this time of booking, the client may have to reply the letter 'Y' (Figure 5.13). After getting this confirmation, the server will auto-generate an electronic ticket which will be sent to the client (Figure 5.14). The information of this ticket will be stored in the database. (Figure 5.15)

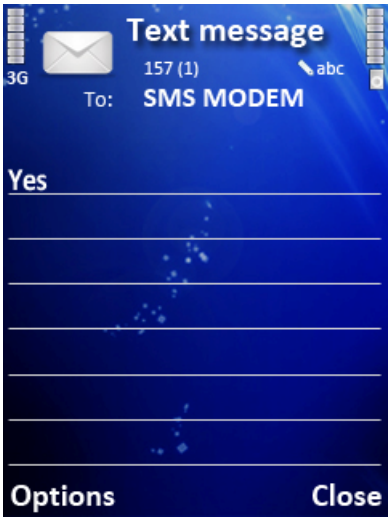


Figure 5.13 The confirmation sent by the client.

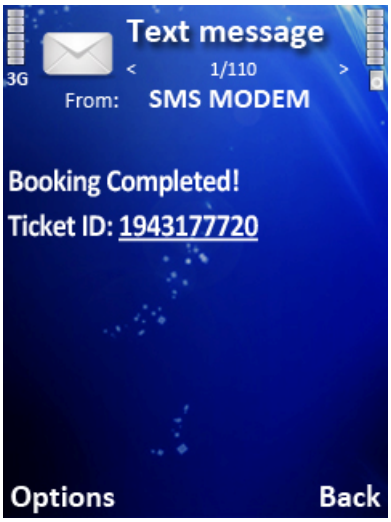


Figure 5.14 The electronic ticket auto-generated by the server.

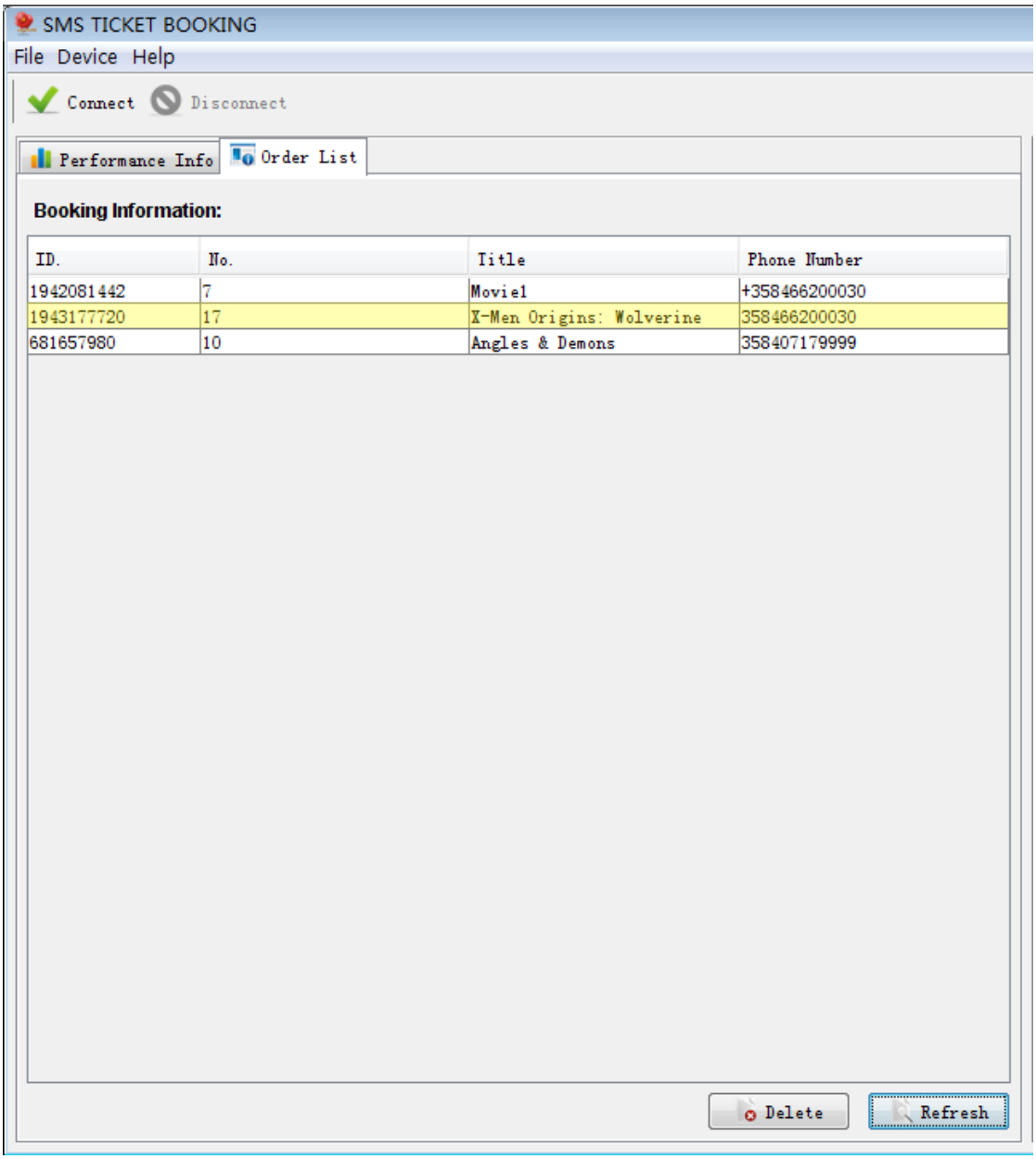


Figure 5.15 The ticket information is stored in the database.

6. CONCLUSION

At the end of my final thesis work, I will present a short analysis of the advantages and disadvantages of the SMS ticket booking system when compared to other type of ticket booking system. After that, the future plan which could enhance the quality of my work will be presented.

6.1 Advantages and Disadvantages of SMS Ticket Booking System

As the coin has two sides, for everything, the advantages come with disadvantages. For the ticket booking systems or other type of e-commerce information systems, they should have some basic features, such as stability, flexibility, client-oriented and information-security.

In the aspect of stability, the SMS ticket booking system can perform well. Because the SMS service is using the plain-text during the data transmission, and the feedback data from clients is really simple, the possibility of transmission failures is very low. Moreover, the SMS service is a common telecommunication service which does not base on platforms or service providers. SMS service only needs very little system spending of clients' mobile device. Nearly every GSM terminal could send or receive SMS messages, that means everyone can enjoy the service of SMS ticket booking without worrying about the network protocols or the GSM device is not compatible with the service. As mentioned in the previous chapter, by using the mechanism of session control, the SMS ticket booking system can also avoid the impacts made by GSM network problems or even hostile attacks.

In the aspect of flexibility, the SMS ticket booking system has very strong superiority. Clients can book tickets just by sending and receiving SMS messages in anywhere, at anytime, which means there is no need to care that the device is too heavy to carry, or no

network service exists. Somebody may think that the wireless internet service provided by 3G can also have such feature of flexibility. However, the problem is that not everyone is using the 3G mobile device. In addition, in some countries, the 3G network is still under construction, so the 2G service is a good choice.

The problem related to client-oriented is a little bit complicated. Concerning with the low price of SMS service and the low requirements on device, the SMS ticket booking system is a very good client-oriented e-commerce information system. On the other hand, the clients always have the requests of searching some ticket information, which is more suitable for the place, or time. From that point, the on-line ticket booking websites always perform very well, because clients can input many conditions to search which ticket is more suitable. Whereas for the SMS ticket booking, it is not easy for service providers that implement syntax or tokens to realize the search. Moreover, such manner will not be welcomed by clients. The character limit of SMS message also counterworks the client-oriented of the service. When we book some ticket on websites, we can view a lot of items on one page. However, clients may only get 3 to 4 items in one SMS message. It is not very convenient for choosing a suitable ticket.

SMS service is not a very safe way of doing transactions, because the SMS message adopts plain-text during transmission. Although the current version of SIM cards is quite strong in authentication, hackers still have chances to do wire-tapping on former version SIM cards. The GSM system adopts SIM ID checking to secure the information privacy, but some of the former checking technologies have already been cracked, so it is dangerous if the client is using a former version of SIM card. In contrast, the ticket booking systems based on websites have more information security features, which could encrypt the data being transferred.

For the device used in my final thesis project, it seems impossible to prevent the DoS attacks, even if the system has the session timeout mechanism. In the experiment, I found that the GSM messages always jam when the device is trying to receive multiple

messages at the same time. Fortunately, the device of real-commercial use is quite different. The real-commercial structure always adopts special GSM gateway which can avoid the traffic jam problems. Enterprises also can get the SMS information directly from the GSM network providers.

6.2 Future Plan for Amelioration

For time constraints, the functions of my final thesis project are still not very integrated. Consequently, it is possible to make it better in many ways. As mentioned in the previous section, a usable e-commerce information system should have features in the aspect of stability, flexibility, client-oriented and information security. From these features, some future plan can be made to ameliorate the SMS ticket booking system.

Long SMS message

As mentioned before, in the aspect of client-orientation, the character limit of SMS message holds down the ability of the SMS ticket booking system. For that reason, I can adopt the long SMS message to improve it. As known, the SMS message only supports 160 characters in one message. It may be enough in daily life, but not when offering ticket information. Nowadays, most of mobile phones support long SMS messages, so more ticket information can be shown by using long SMS messages.

Ticket information query

In Chapter 5, I discussed why I simplified the transaction flow in this final thesis project. Thus, the function of ticket information query can be realized in the future. By adding the time table and place information of event, clients can search ticket information by sending SMS messages. The problem is how to create syntax for search tokens. There should be a formula to regulate the tokens of search, so the system can execute the multi-table query automatically, and otherwise the messages for search will be dropped

as in-compatible messages. It is not easy for clients that send search messages in a right form. Thus, an instruction may need to be sent to client before the launch the search.

AES encryption

The information security feature can be enhanced by implementing the AES encryption. Advanced Encryption Standard (AES) is an encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide (Wikipedia [referred 7.5.2009]). Moreover, as the core of SMS service in my project, the SMSLib API also support the message encryption by using AES algorithm, so the system can send encrypted messages to the clients. The problem is that the clients can get the encrypted messages but it is unable to read it. For that reason, in the future, the research of embedding the AES into client software which will be running on mobile phones will be a topic. By implementing this, clients can send encrypted SMS messages to the system and receive encrypted replies.

INDEX OF REFERENCES

Books:

Carter, Jim A. Developing e-commerce systems. Prentice Hall.

Redl, Siegmund M 1995. An introduction to GSM. Artech House.

Redl, Siegmund M 1998. GSM and personal communications handbook. Artech House.

Electronic sources:

BSS. Wikipedia, Updated 2008b [referred 6.4.2009]
Available in www-format:
<URL: http://en.wikipedia.org/wiki/Base_Station_Subsystem>

Christmas and New Year: Vodafone processed 992 million SMS. Vodafone, Updated 1/2008 [referred 6.4.2009]
Available in www-format:
<URL: http://www.vodafone.com/start/media_relations/news/local_press_releases/portugal/portugal_press_release/christmas_and_new.html>

Finn Trosby. SMS, the strange duckling of GSM. Telektronikk vol. 3 2004, page 5. [referred 6.4.2009]

Hayes AT Command. Wikipedia, 2009. Updated 2.4.2009 [referred 12.4.2009].
Available in www-format:
<URL: http://en.wikipedia.org/wiki/AT_command>

Introduction to GSM modem. 88soft.cn, [referred 11.4.2009]
Available in www-format:
<URL: http://www.88soft.cn/gsm-modem/gsm-modem-intro.html>

NewNet SMserver wireless short message service tutorial. ADC Telecommunications, 1999. [referred 6.4.2009]
Available in pdf-format:
<URL: http://www.mobilein.com/SMS_tutorial.pdf>

Short Message Service / SMS Tutorial: How to Send SMS Messages from a Computer / PC. Developers' Home, Updated 2/2008 [referred 6.4.2009]
Available in www-format:
<URL: http://www.developershome.com/sms/howToSendSMSFromPC.asp>

Short Message Service / SMS Tutorial: What is an SMS gateway. Developers' Home, Updated 2/2008 [referred 6.4.2009]
Available in www-format:
<URL: http://www.developershome.com/sms/sms_tutorial.asp?page=smsGateway>

SIEMENS MC35 data sheet. SIEMENS AG, [referred 11.4.2009]
Available in pdf-format:
<URL: http://harvestelectronics.com/harvest/pdf/mc35_data_sheet.pdf>

SIEMENS TC35 user guide. SIEMENS AG, [referred 12.4.2009]
Available in pdf-format:
< URL:http://harvestelectronics.com/harvest/pdf/tc35_t_ug.pdf >

SMS. Wikipedia, 2008a [referred 6.4.2009]

Available in www-format:
<URL: http://en.wikipedia.org/wiki/Short_message_service>

Sun JAVA Communication API documents. java.sun.com [referred 20.4.2009].
Available in www-format:
< URL: http://java.sun.com/products/javacomm/reference/docs/index.html>

Thanks to SMS, hospitals received thousands of liters of blood. Vodafone, Updated 5/2008**.** [referred 6.4.2009]
Available in www-format:
<URL:http://www.vodafone.com/start/media_relations/news/local_press_releases/czech_republic/czech_republic_press/thanks_to_sms__hospitals.html>

Why MySQL? mysql.com [referred 5.5.2009]
Available in www-format:
< URL: http://www.mysql.com/why-mysql/>