CLINIC MANAGEMENT SYSTEM
WITH NOTIFICATION USING GSM MODEM

CHAPTER 1

INTRODUCTION

1.1 Introduction

Clinic is an organization that is responsible in providing a health medication and treatment for all types of peoples. Surely**,** everyday there are people that need to use the clinic services. But how can clinic provide a faster and efficient services if they are still using the traditional method on their daily operation? The traditional method means the customers need to fill in their detail in registration form manually and the information will only keep in files. After the registration, the files will be place in the rack and this will cause problems like taking a longer time to retrieve the information, make mistakes during writing or misplaced the files.

As a result, one system called Clinic Management System with Notification using GSM Modem will be develop to resolve all the current problems at clinic. Clinic Management System with Notification using GSM Modem is specially designed to let the clinic staff has a high efficiency management tools, computerized and systematic patients record, and detail of treatment records. This system also provide appointment feature, which allow staffs to view the appointment that already made by doctors and

process it by sending a notification to patients. Patients will receive the notification about their appointment details on their mobile phone.

This new system will replace the current system that is used in clinic and surely this system will improve the clinic services and make their daily operation running smoothly.

1.2 Problem Statements

a) Traditional method, which is the information about patients that is kept on file and back into the rack, has caused problems for the clinic staffsto retrieve the information and this might take a longer time.
b) The traditional method indeed caused too many usage of paper. For that reason, inventing an Eco-friendly product is necessary to save the natural resources.
c) Patient tends to forget their appointment with the doctor, as there is no reminder from the clinic itself.

1.3 Objectives

a) To computerized and centralized all the information in order to reduce the time in retrieving all the data and information.
b) To promote Eco-friendly software that will reduce the usage of paper.
c) To facilitate patients, so that we can notify the patients using SMS notification, so they can be on time for their appointment.

1.4 Scope

a) This system will replace the old system that is currently used in most clinic in Malaysia.
b) This environment of this system is based on Java programming language.
c) This system will divide to three users, which are for the clinic staff, doctors and administrator. Each of these users has their own permitted area in order to access this system.

1.5 Thesis Organization

This thesis consists of six (6) chapters.

Chapter 1 is an introduction of the system. This introduction consists of system overview. Problem statement has been discussed on the problem that faced by the current system. As for the objectives, the reasons of the development of the project are listed. Scope of the project is discussed on project and user limitation.

Chapter 2 is the literature review, which consists of the current system and the technique or software that is used on it.

Chapter 3 is about the system methodology. It will be on the method that is used to develop the system and project planning. On this chapter, there will be an overview of the project planning such as the creating of the software and the device that helps to develop the system.

Chapter 4 is the elaboration of project implementation. This chapter is more or less on the design of the project development.

Chapter 5 will be based on the discussion and result that was received from the data and data analysis, project constrain, and fix and suggestion of the system. Project analysis is the discourse on the project objective, which is the continuation of the project problem.

Chapter 6 is the conclusion of the project. This includes the conclusion of the data that were received, the methodology, and the used research implementation.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter will focus on the literature review. It will cover a period of 1997 to 2011 and will discuss about the concept of the management system, which is the most common problem relating to the system, techniques and basic requirement for this system.

This chapter introduces the definition of computerized system, the GSM modem and also factors that cause the management systems to fail and explain the similar systems.

The developed system is Clinic Management System with Notification Using GSM Modem. These systems are responsible to store patients' data and notify them for their appointment. Before this, most clinics usually used the traditional methods that are quite unsatisfactory. The traditional system used by the staffs is exposed to common mistakes while writing and the probability of having lost document or being misplaced is quite high. Knowing that the document cannot keep as many data as they could about the patient, thus the other important information about them may not be included in that form. Besides that, the patient may have forgotten their appointment with the doctor.

2.2 Definition Management System

Management system actually has been used long time ago. Every organization needs a system that can manage their data and process. Normally they used a conventional method to manage and store their data.

Management system is management documented and tested step-by-step method aimed at smooth functioning through standard practices. Used primarily in franchising industry, management systems generally include detailed information on topics such as ("Businessdictionary.com - online,")

i. organizing an enterprise,
ii. setting and implementing corporate policies,
iii. establishing accounts, monitoring, and quality control procedures,
iv. choosing and training employees,
v. choosing suppliers and getting best value from them, and
vi. marketing and distribution.

2.3 Advantages Computerized System Over The Manual System

A system is an arrangement of elements that when it is put together it becomes an organized and established procedure. A system typically consists of components connected together in order to facilitate the flow of information, matter or energy. A computer system consists of a set of hardware and software, which processes data in a meaningful way.

In every company, keeping record are very important. For the clinic, it is very important to keep the patient record for any reference. There is some method in keeping the record such as using the manual method or the computerized method. The computerized system is better than manual system in keeping record (Egwunyenga, 2009) of the patients. Hence, using the computerized system has so many advantages than the manual system.

One of the advantages using a computerized system is that it is not only easy, but it also saves the time to search the patient record. If they use the manual keeping

record, they have a hard time to find for it. The computerized system will give the opportunity for the companies to do work more effective and efficiently if the company use it (Dalcı & Tanış).

The next advantage is that the staff can update the patient record easily. If the costumers come to the same clinic more than a time, the staffs could find the patient record without any difficulty. If the searching record is easy, the update task is easy as well. The update task is faster and more efficient compared to the manual system.

Another advantage is having this computerized keeping record system, information for a particular period of time can be compiled quickly. With the manual system, it takes time to locate the information from each file and compile it into a report.

Besides that, computerized system can save paper and space. If the clinic is using the manual system, at least a few papers from each file will be used for the keeping record for individual patient. If there are thousands of patients in a clinic, obviously it will need as many papers and files as they could in keeping their record. Doubtlessly, the clinic needs more space to keep the entire file in place. Thus, by using the computerized system, the staffs can store as much details of the patient information in the database given. From this statement, the computerized system also helps in saving cost from buying papers and files for the documentation of patients. By using the computerized system, the target is to create a paperless office, which will turn into reality (Dalcı & Tanış). Obviously the computerized system is better than the manual ones.

2.4 Technique

This section is the review on the current technique on the programming language, in-system programming, database language and methodology.

2.4.1 Programming Language

There are many tools can be used to develop dynamic and interactive system. Java and Visual Basic are the most popular programming tools for graphical user interface (GUI).

a) Java

The Java programming language has been widely accepted as a general purpose language for developing portable applications, toolkits, and applets (Ritchie, 1997).

Java is a programming language originally developed by James Gosling at Sun Microsystems (now part of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java is currently one of the most popular programming languages in use, particularly for client-server web applications (Wikipedia).

Programs written in Java have a reputation for being slower and requiring more memory than those written in C(Dejan). However, Java programs' execution speed has improved significantly with the introduction of Java 2.0 code has approximately half the performance of C code.

b) Visual Basic (VB)

Visual Basic (VB) is the third-generation event-driven programming language and integrated development environment (IDE) from Microsoft for its COM programming model. Visual Basic is designed to be relatively easy to learn and use (Wikipedia).

Microsoft claims that Visual Basic is the quickest and easiest way to create applications for Microsoft Windows [Microsoft 921. Microsoft Windows is one of the fastest selling software packages in history: 3,000,000 copies were sold in the first nine months (Dukovic & Joyce, 1995).

Microsoft Visual Basic is designed for graphical user interface (GUI) programming. It is not a general purpose programming language.

2.4.2 GSM Modem

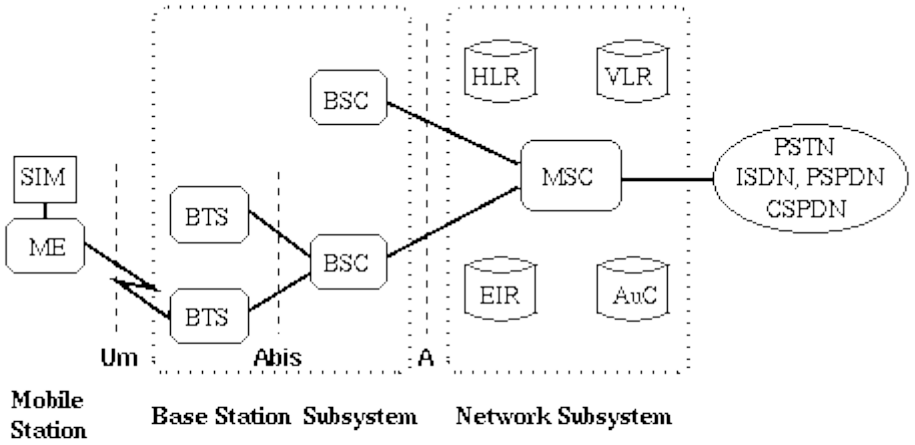
In cellular service there are two main competing network technologies, it's Global System for Mobile Communications (GSM) and Code Division Multiple Access (CDMA) (Constantin, 2011). Since it started in the '80s, GSM telephone system was developed using cell concept for the network topology. Each cell corresponds to a specific antenna (base station), which is placed on towers or tall buildings. The GSM standard has been an advantage to both consumers, who may benefit from the ability to roam and switch carriers without replacing phones, and also to network operators.

GSM also has low-cost implementation of the short message services (SMS), also called astext messaging, which has been supported on other mobile phone standards as well. In view of the fact that there is huge coverage of distance, the GSM infrastructure can be an alternative to transmit or receive data from or to a device like sensor, actuator and complex device near or remotely. Compared to analogue transmission systems, GSM system provides narrowest bandwidth for the channel, through the use of voice compression algorithm; improving the quality of transmission by using detection and correction codes of errors; digital signal encryption to ensure security and protection against unwanted interception.

2.4.2.1 Architecture of the GSM network

A GSM network is composed of several functional entities, whose functions and interfaces are specified. Figure 1 shows the layout of a generic GSM network. The GSM network can be divided into three broad parts. The Mobile Station is carried by the subscriber. The Base Station Subsystem controls the radio link with the Mobile Station. The Network Subsystem, the main part of which is the Mobile services Switching Center (MSC), performs the switching of calls between the mobile users, and between mobile and fixed network users.

The MSC also handles the mobility management operations. Not shown is the Operations and Maintenance Center, which oversees the proper operation and setup of the network. The Mobile Station and the Base Station Subsystem communicate across the Um interface, also known as the air interface or radio link. The Base Station Subsystem communicates with the Mobile services Switching Center across the A interface.



SIM Subscriber Identity Module BSC Base Station Controller MSC Mobile services Switching Center
ME Mobile Equipment HLR Home Location Register EIR Equipment Identity Register
BTS Base Transceiver Station VLR Visitor Location Register AuC Authentication Center

Figure 1.1 General architecture of a GSM network

Source: "Information on mobile"

2.4.3 Similar System

Nowadays, there are many systems that similar to the Clinic Management System with Notification Using GSM Modem.

Based on the research that has done, the developer found the differences and similarities between the three of existing system with the system that will be developing later. Besides the differences and similarities, the research about existing system also was helping the developer to get more idea to develop the system. This table below will explain about all the differences between all the systems.

Description	**Generic Notification System for Internet Information**	**Web based Long-Distance Appointment Registered System**	**Clinic Management System with Notification Using GSM Modem.**
Purpose	To provide a mechanism of delivering a notification messages to a single or multiple recipients based on request by the recipients.	To provide a system for hospital management that allow user to make online appointment.	To computerize and centralized the system of with the addition of appointment elements.
Module	Available to anyone that are browsing the internet.	Consist four layer of module: management view, medical management, patients view and data management view.	Three layer of module: Doctor view, clinic staff view and administrator view.
Implementation	This system by combination of Internet browser, dynamic HTML and GSM modem and XML document.	The implementation of this system by using the ASP programming language with addition of remote registration system.	The implementation of this system by using a Java programming language. Besides that, using the GSM modem.
Database design	The database of this system was	The database of this system based on the	The databases for this system will be

	designed based on generally, which means there use the internet browser database and the user information.	information of the patient, doctor and reservation tables.	designed based on the criteria of patient, appointment and other else.
Advantages		This system allow user to make an appointment without come to hospital, cancel and update their information and appointment.	Can computerize the patient information and user will get the notification about their appointment through mobile phone.
Limitation	The system limitation was found that the environment of this system is still restricted for some types of basic notifications.	Before the user want use this system, they must register manually with hospital and deposit money. If not, they not allowed to using this system.	The limitation might be in this system is the user cannot make the appointment by online.
Availability of user	Large scale of user.	User consists of patient and doctor at the hospital that implement this system.	User consists of staff and doctor at the clinic that implement this system.

Figure 2.1 Comparison between existing systems

2.4.3 Base64 Encoding

This system Base64 content-transfer-encoding or called Base64 encoding, is defined in RFC 2045 .It is a method designed to represent an

arbitrary sequence of octets (8-bit) in a printable text form that allows passing binary data through channels that are designed for flat ASCII text such as SMTP(Postel, 1982). It also allows embedding of binary data in media supporting ASCII text only such as XML files.

Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Character	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Value	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Character	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g	h
Value	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Character	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
Value	51	52	53	54	55	56	57	58	59	60	61	62	63	(pad)			
Character	z	0	1	2	3	4	5	6	7	8	9	+	/	=			

Figure 2.2 Base64 alphabets.

2.4.3.1 Encoding

The process consists in representing groups of 3 octets (24 bits) of input bits as output strings of 4 encoded characters and the input as a linear stream of octets. Arranged from left to right, the input is divided into 24-bit groups, each formed by 3 consecutive octets of the input stream. These 24-bit groups are then treated as groups of 4 concatenated 6-bit groups. Each 6-bit group is a binary number, representing a decimal value between 0 and 63. That value is used as an index into the array of the Base64 alphabet. The corresponding encoded character is placed in the output string.

2.4.3.2 Padding

The input ends with a whole 24-bit group. The output is a multiple of 4 Base64 encoded characters. No special action is needed. The input ends with two octets or a 16-bit group. Two zero bits need to be added to form a whole 3 6-bit group, which translates into 3 Base64 encoded characters. A padding character '=' is needed to make the output a multiple of 4 characters. The input ends with an octet or an 8-bit group. Four zero bits need to be added to have 2 encoded characters. And two padding characters are added.

2.4.3.3 Decoding

The process of decoding works in opposite to the encoding process. That is 24-bit groups of 4 6-bit groups are translated into groups of 3 octets.All line breaks or other characters not in the Base64 alphabet are to be ignored by the

decoding software and also any illegal sequences of characters in the Base64 encoding, such as "====".

2.5 Summary

Clinic Management System was developed using Netbeans that used Java as the main language. This system also used MySQL as their database that store all the information and data.

CHAPTER 3

METHODOLOGY

In this chapter will discuss about the methodology that will be using in the development of Clinic Management System with Notification Using GSM Modem. The fundamental for this project is to develop a management system that can be implemented and integrated in clinics. This project will be conducted based on the Iterative and Incremental Development method. From the beginning, this project will be developed based on the methodology choose.

3.1 Introduction

Iterative and Incremental development is at the liver of a cyclic software development process developed in response to the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between (Wikipedia). Iterative or incremental methodologies provide a cyclical approach to software development, which is especially useful for environments where requirements change often and response need to be quick.

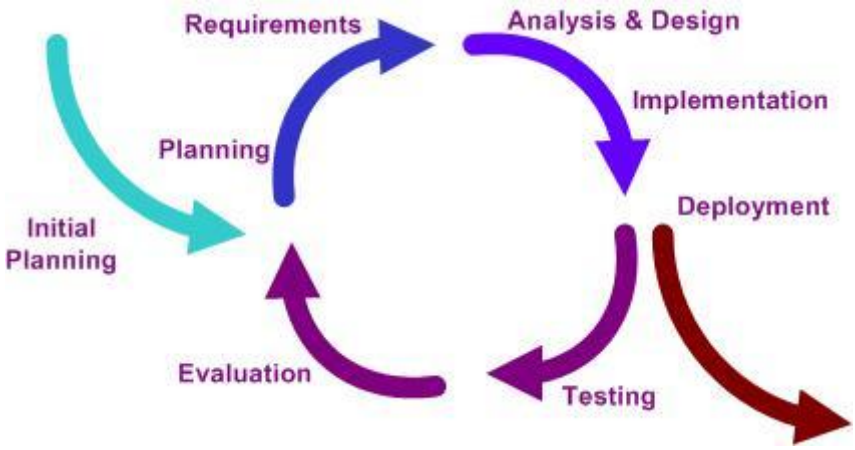


Figure 3.1 Iterative and Incremental Development Method

3.1.1 The Justification of Iterative and Incremental Development Methodology

There are many reasons why we use Iterative and Incremental Development Method for this project. Iterative and Incremental Development Method is very efficient to deal with a project that change often and response need to be quick. It also can deal with a project that the budget didn't cover the costs of the project because change control procedures gobbled up additional funds (Steigjer, 2008). When using Iterative and Incremental Development Method, the development cost of change is more efficient than conventional software development process.

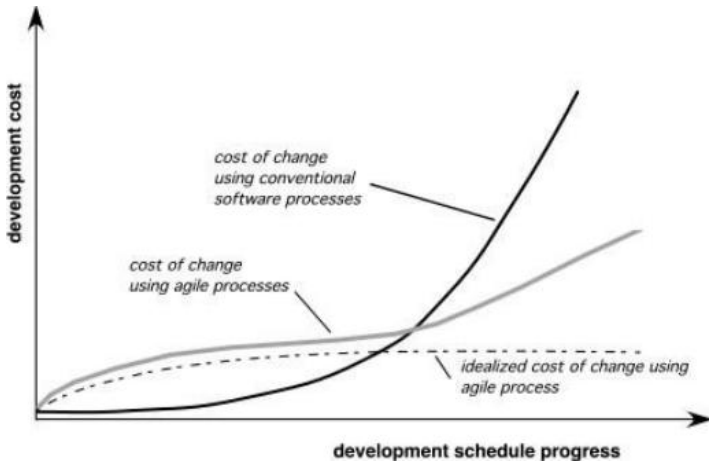


Figure 3.2 Comparison Iterative and Incremental Development Method

Source: Pressman, 2010

3.2 Iterative and Incremental Development Method

The model or approach here are used in developing this project

3.2.1 Planning Stage

In this phase, the most important business function or the goal is identified. Through the process of gathering requirement, it is found out that the system will be develop using NetBeans IDE 6.9.1 technology and interact with My SQL Database Server. The following figure 3.2 is the system's context diagram which shows flow of data between the users and the system.

The scopes of project are specified and a schedule has been design as guidance throughout the system development process to make sure the delivery of it on timely mannered. This has been done with the help of Gantt chart produced through Microsoft Project.

3.2.2 System Requirement

One of the most important tasks in the development of software using the Iterative and Incremental Development Method is gathering and defining the requirements for the project. In order to arrange requirement to develop management system, a research and analysis on existing system has been done. Information gathered during the research gives clearer overview on the flow of the process while answering the question on how to achieve the main goal of the system which is to reengineering the current system with the addition of GSM Modem for notification.

Then from the data, I need to analysis it and choose the all of the requirement that I need to include in the software. I need to understand the flow of management system.

The outcomes from requirement phase are:

- A vision document as general vision of the core project's
- Requirements, key features and main constraints.

- A project plan, showing phases, iterations and major milestones.
- An initial use-case model

3.2.3 Hardware Requirements

The hardware requirement here are used in developing this project

I. Workstation

In this system development, a workstation is the most important hardware. Table below explain the minimum requirements

Table 3.1 Workstation requirements

Hardware	**Minimum Specification**
Central Processing Unit (CPU)	Intel Pentium IV or higher
Memory Cache	3MB
Random Access Memory (RAM)	1GB
Hard Disk Space	80GB
Network Transmission Speed	100Mbps

II. GSM Modem

A GSM modem is a specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone. From the mobile operator perspective, a GSM modem looks just like a mobile phone.

When a GSM modem is connected to a computer, this allows the computer to use the GSM modem to communicate over the mobile network. While these GSM modems are most frequently used to provide mobile internet connectivity, many of them can also be used for sending and receiving SMS and MMS messages (NowSMS | SMS Gateway, SMS Server Software, MMS Gateway & MMSC).

A GSM modem can be a dedicated modem device with a serial, USB or Bluetooth connection, or it can be a mobile phone that provides GSM modem capabilities

3.2.4 Software Requirements

The development tool specified here is software to use in developing this project.

a. NetBeans IDE 6.9.1

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Groovy, C, C++, Scala, Clojure, and others.The NetBeans IDE is written in Java and can run anywhere a compatible JVM is installed, including Windows, Mac OS, Linux, and Solaris.

b. Java Runtime Environment

The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language.A Java virtual machine (JVM) is a virtual machine that can execute Java bytecode. It is the code execution component of the Java software platform.

c. My SQL Database System

MySQL nowadays widely use as database platform. There are many reasons why we choose MySQL. This is some of the reason:

- Speed.
  MySQL is fast compared to others.

- Ease of use.
  MySQL is a high-performance but relatively simple database system and is much less complex to set up and administer than larger systems.

- Query language support

It is also understands SQL (Structured Query Language), the standard language of choice for all modern database systems.

- Capability.

  The MySQL server is multi-threaded; so many clients can connect to it at the same time. Each client can use multiple databases simultaneously.

- Connectivity and security

  Besides that, it is fully networked, and databases can be accessed from anywhere on the Internet, so user can share your data with anyone, anywhere. But MySQL has access control so that one person who shouldn't see another's data cannot. To provide additional security, MySQL supports encrypted connections using the Secure Sockets Layer (SSL) protocol. This is will provide to the administrator of this system later.

- Availability and cost.

  MySQL is an Open Source project with dual licensing. First, it is available under the terms of the GNU General Public License (GPL). This means that MySQL is available without cost for most in-house uses. Second, for organizations that prefer or require formal arrangements or that do not want to be bound by the conditions of the GPL, commercial licenses are available.

3.3 Analysis and Design

In this system analysis phase, I need to define the requirement from previous phase. Process analysis can be by observation, interview and many more. For Clinic Management System with Notification Using GSM Modem, I decided to observe and analysis of the similar system. By this I can get most recent and updated problem in existing system. So I can avoid the same problem.

The important thing for make sure my project running smoothly is study for the current process because the system must follow the requirements .The problems and

constraints also defined by me in this phase. The new system must overcome a problem in current system**.**

3.3.1 Flowchart

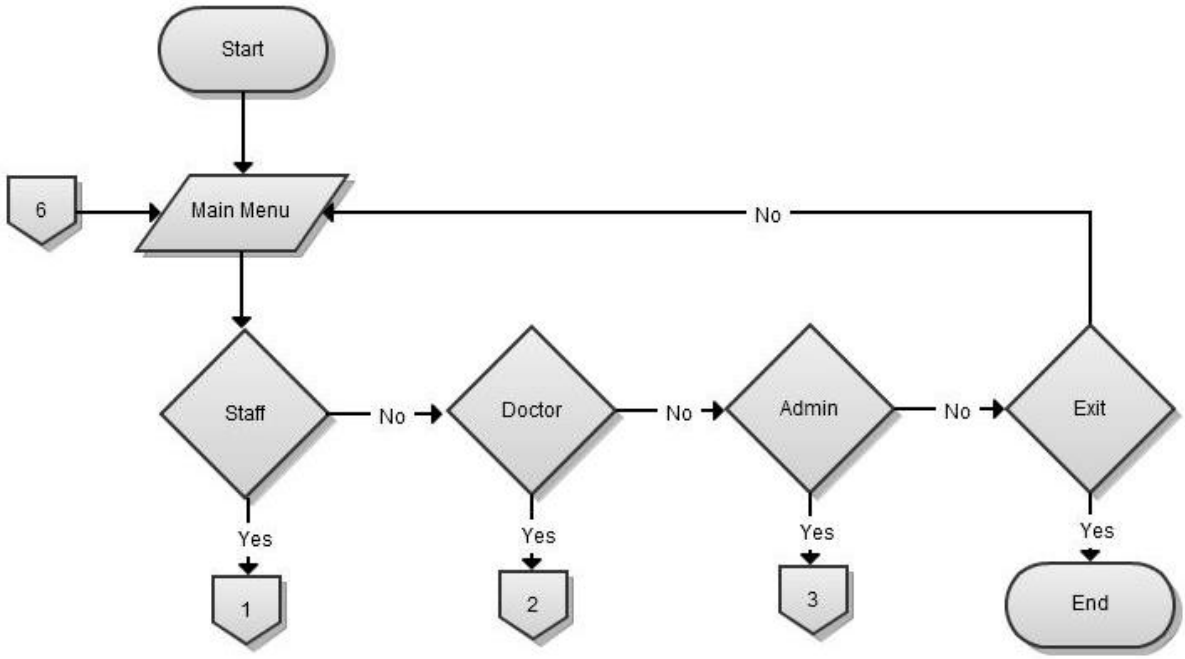


Figure 3.3 Flowchart in the Main Menu

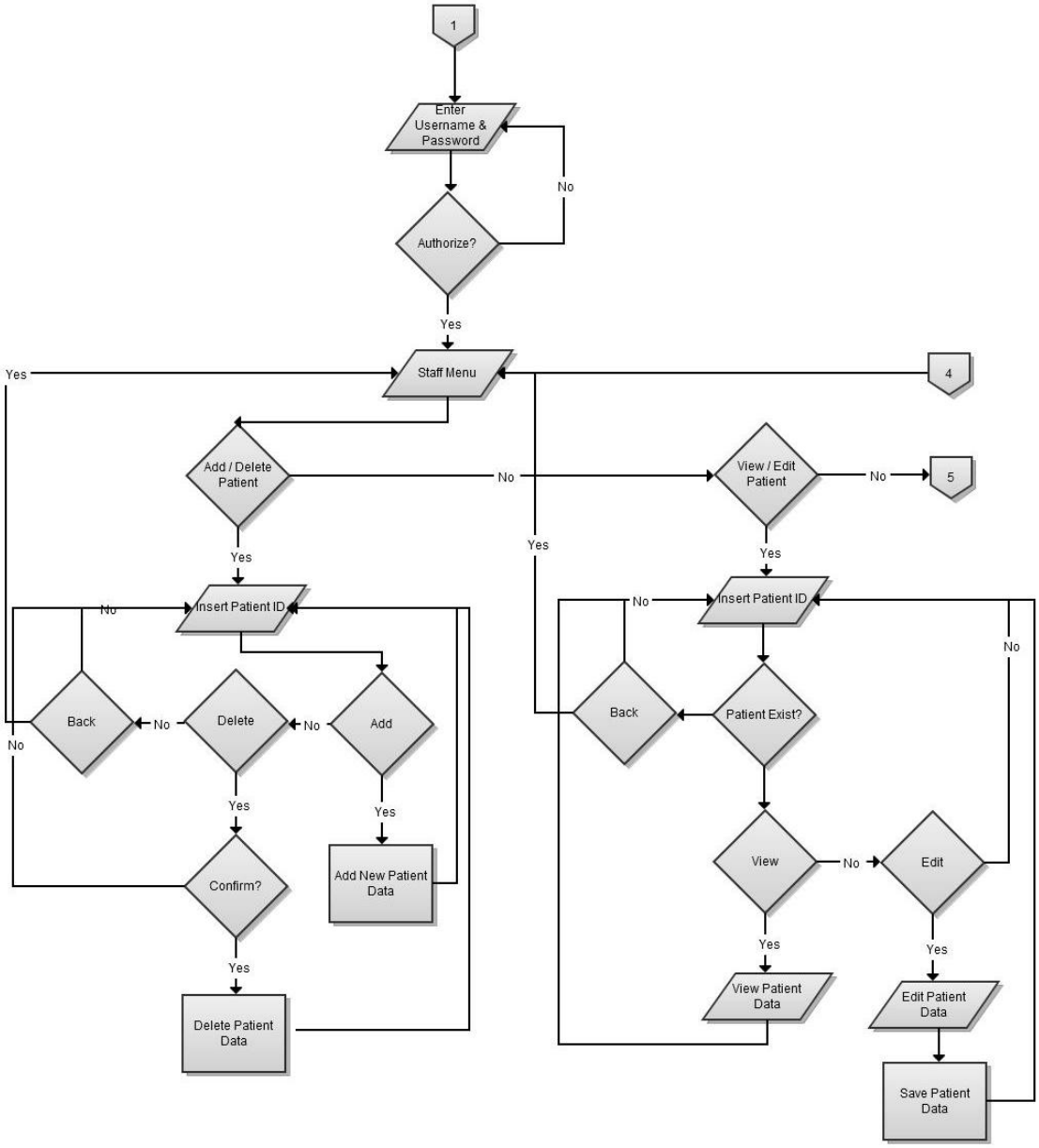


Figure 3.4 Flowchart in the Staff Menu

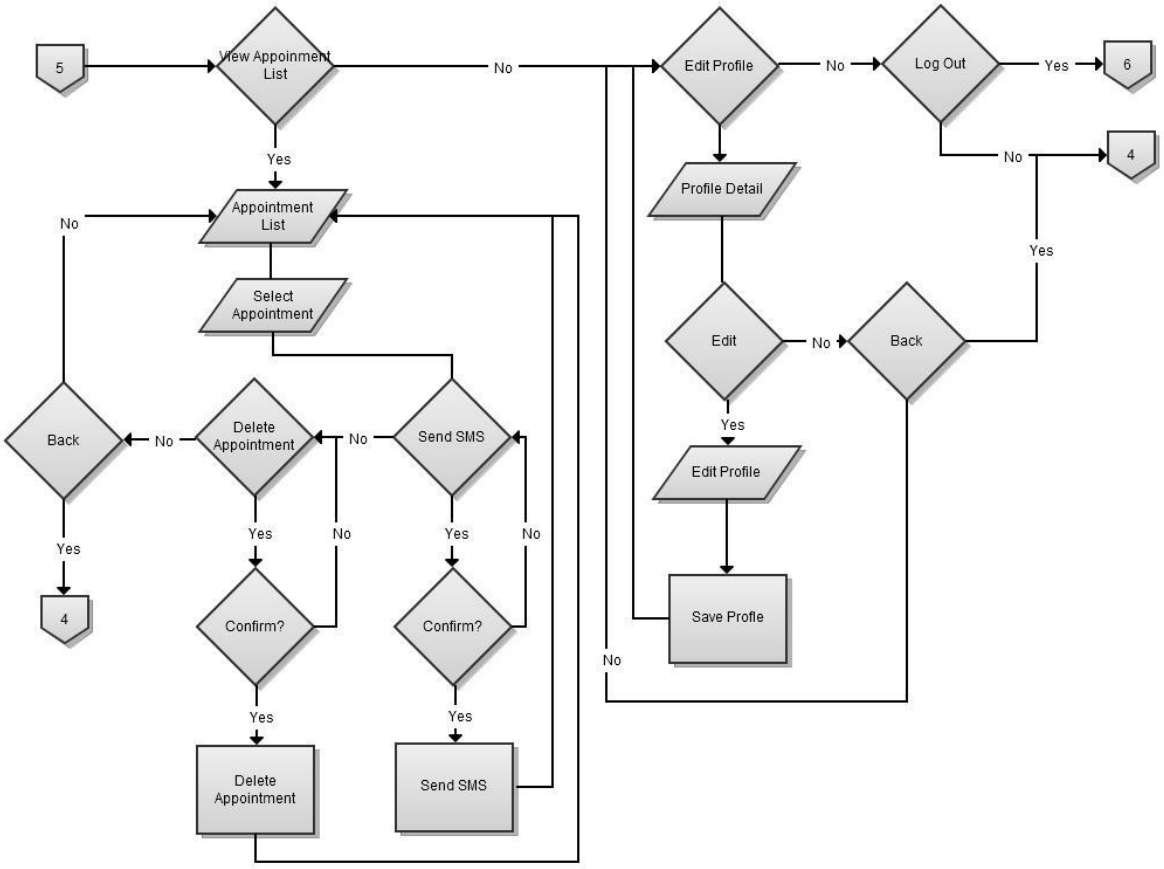


Figure 3.5 Flowchart in the Staff Menu (Continue)

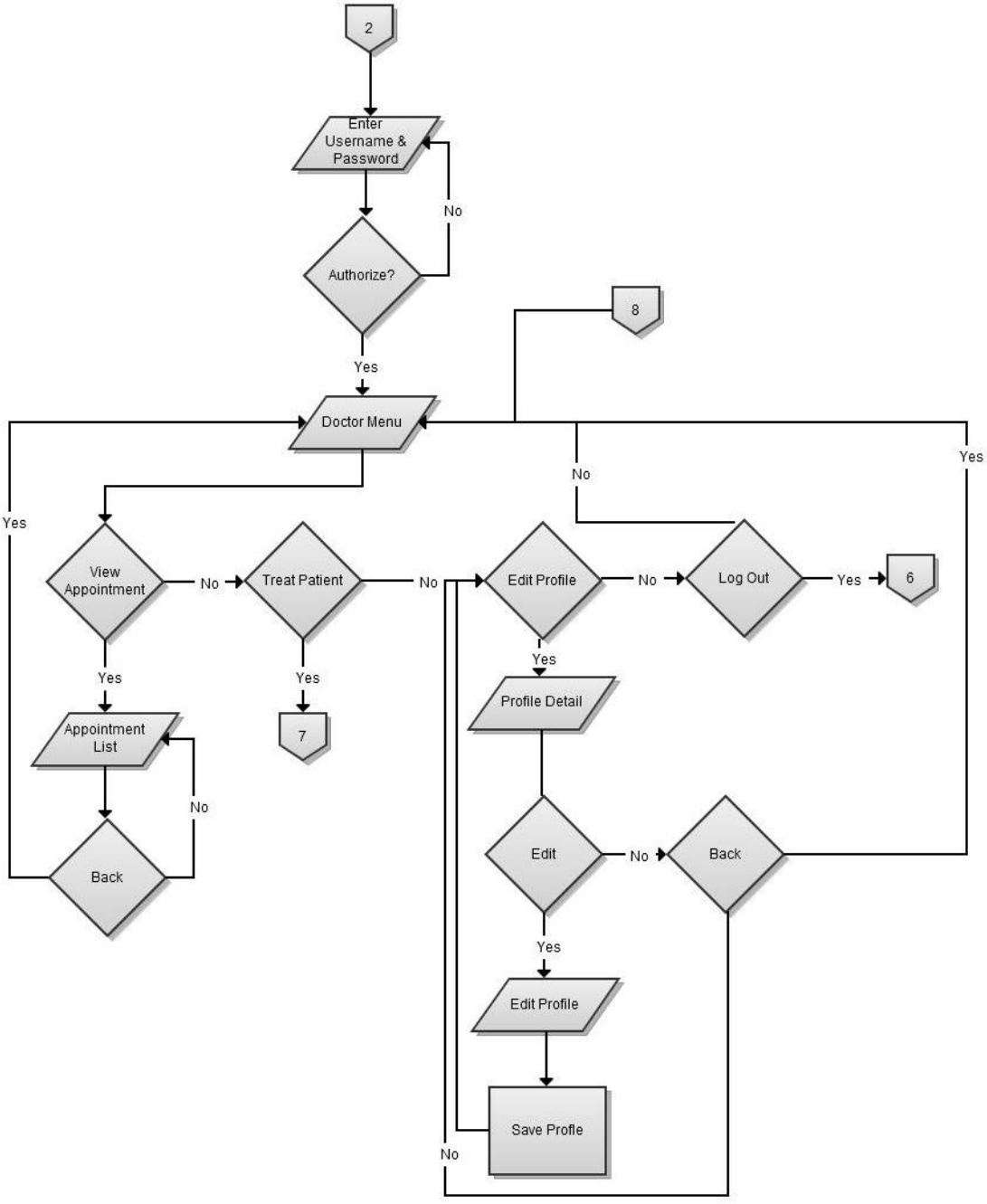


Figure 3.6 Flowchart in the Doctor Menu

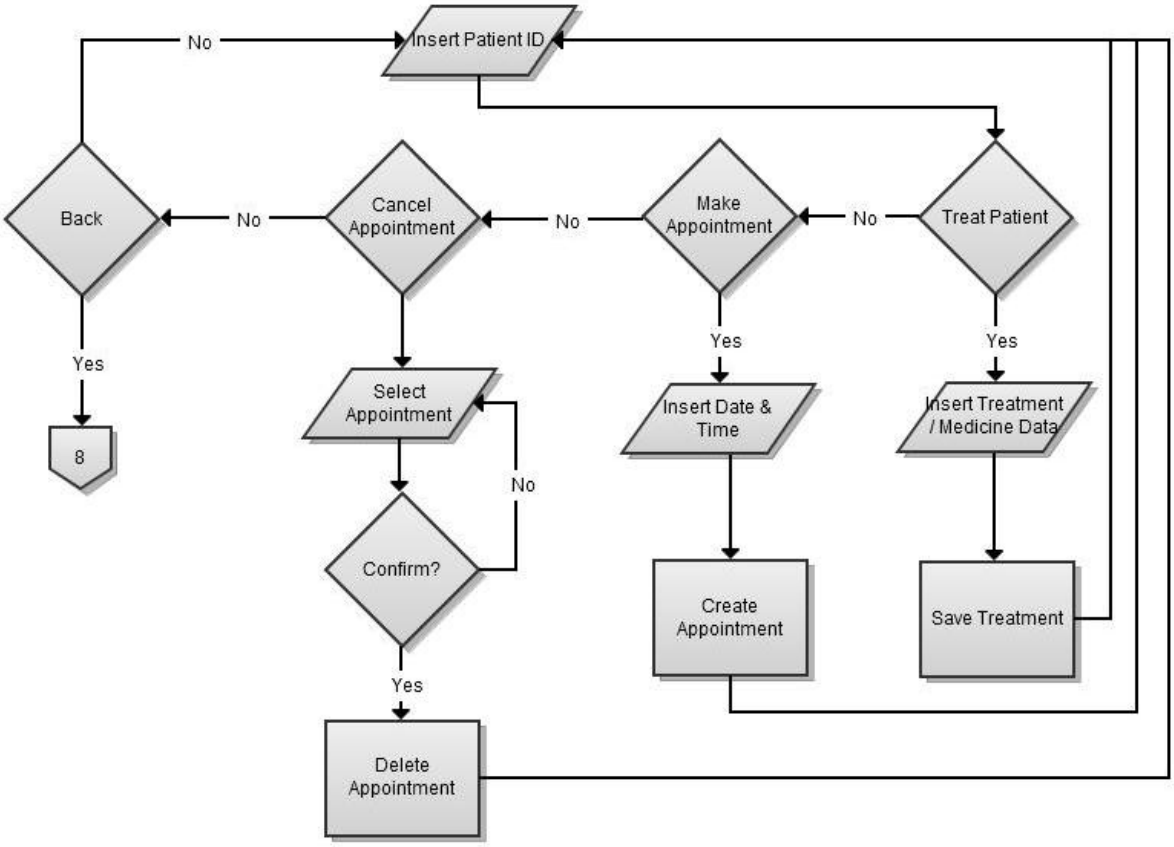


Figure 3.7 Flowchart in the Doctor Menu (Continue)

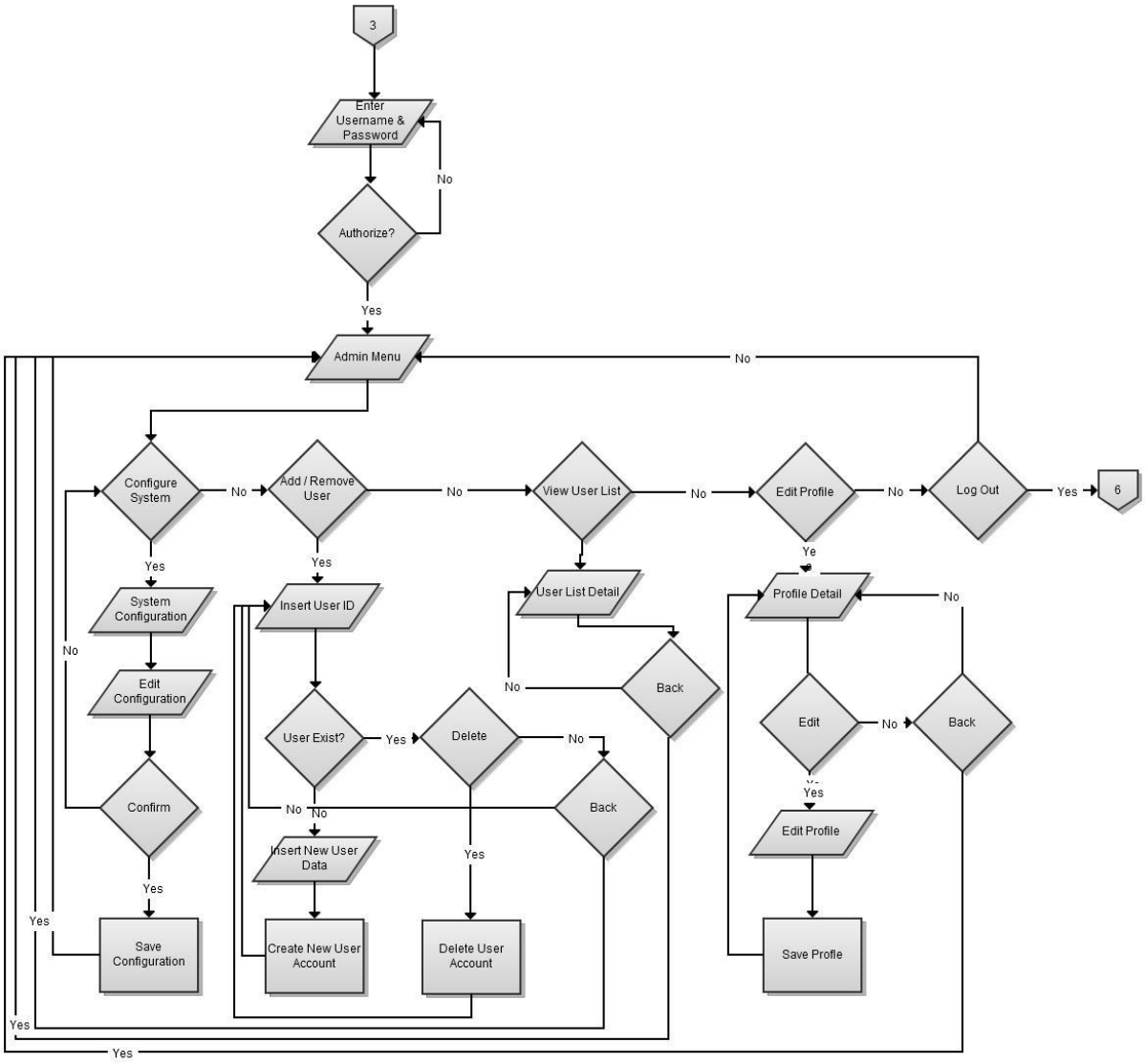


Figure 3.8 Flowchart in the Admin Menu

3.3.2 Entity Relational Diagram Design

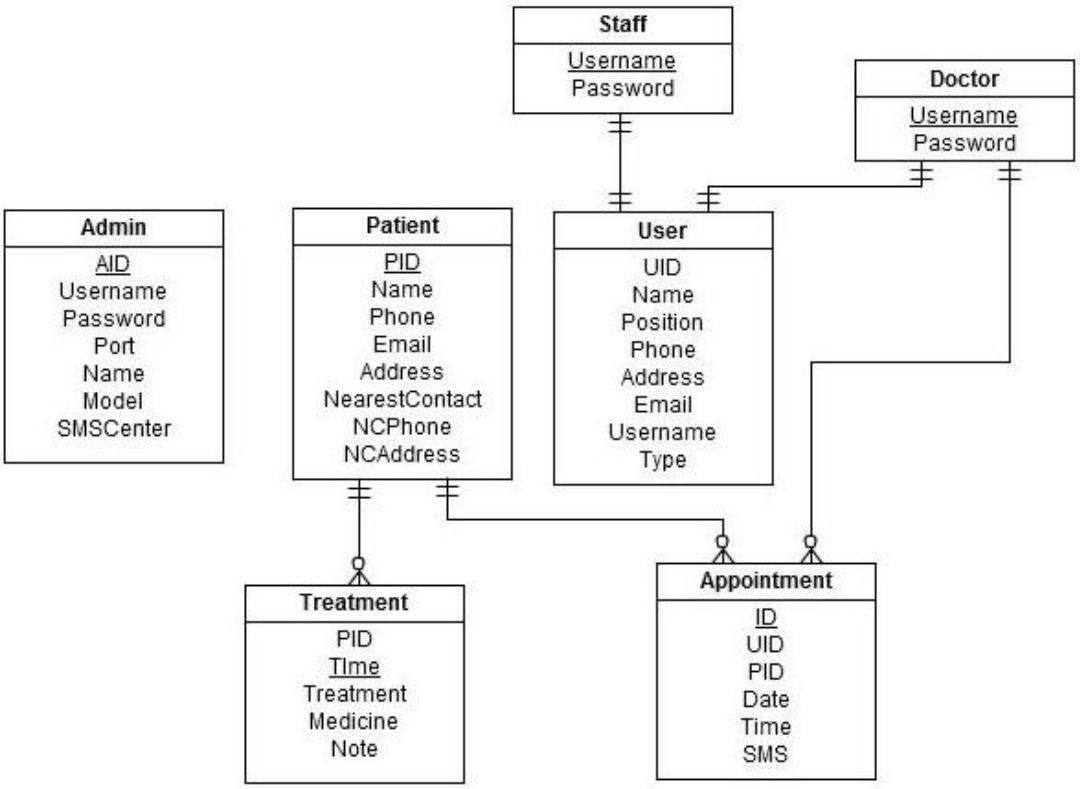


Figure 3.9 Entity Relationship Diagram

Figure 3.8 illustrate on the database flows of the system. There will be seven tables in the database. First table is for Admin which contains AID, Username, Password, and Port for modem communication port, Name for modem's name, and SMSCenter. Admin table will use AID as a primary key. Second table is for Patient. This table contains patient data which are PID, Name, Phone, Email, Address, NearestContact, NCPhone and NCAddress. This table will use PID as a primary key which will use a unique id for connecting between Treatment and Appointment table.

Treatment table will contain treatment data for patient and a patient can have many treatment same with Appointment table. Appointment table contain appointment detail for Patient and Doctor which connected using UID and PID. There also have three others table for system's user. User table will contain all

the detail about user and will connecting with Staff and Doctor table for their login information.

3.3.3 Use Case Diagram Design

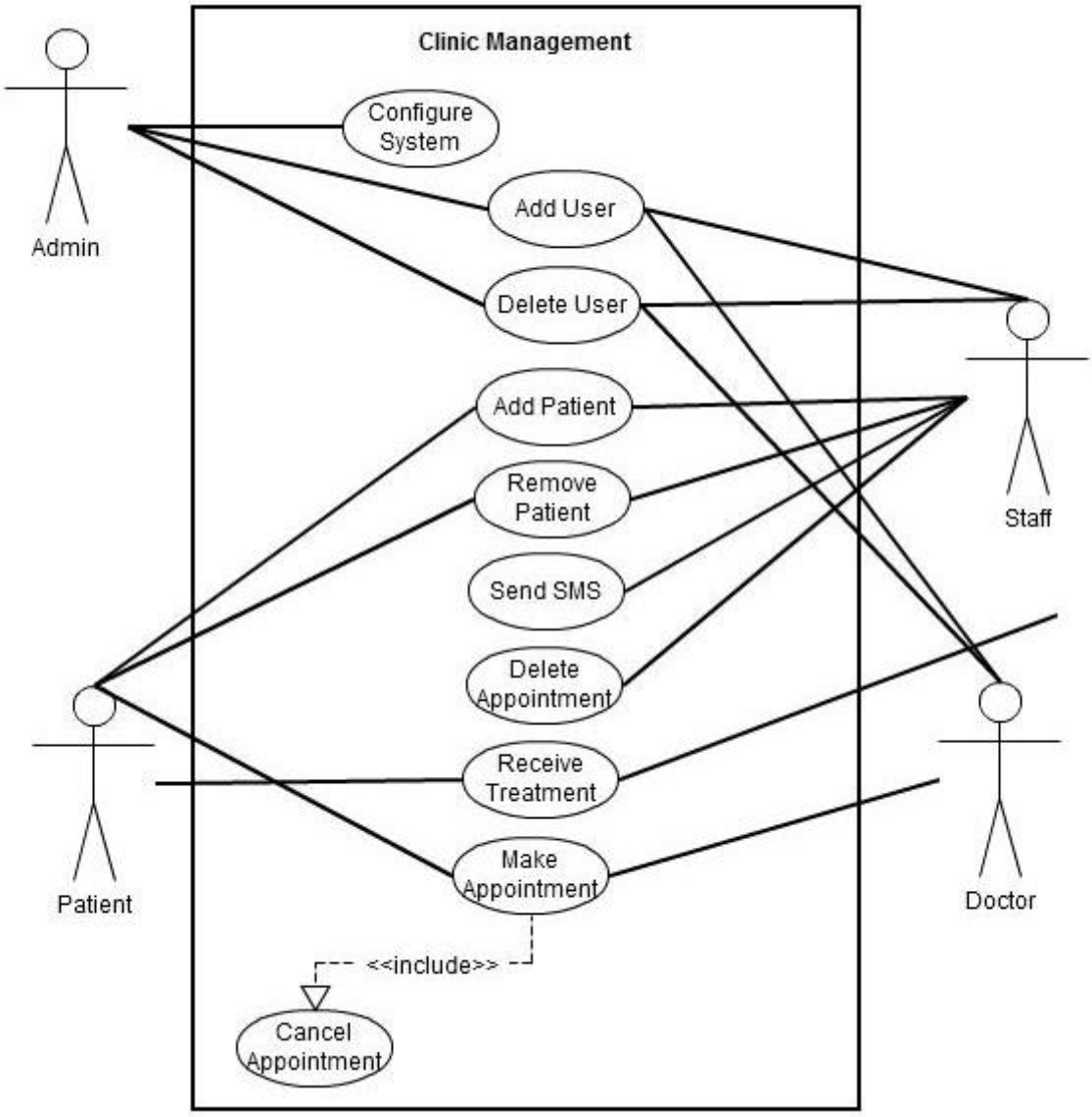


Figure 3.10 Use Case Diagram

Figure 3.9 illustrate on the flows of the system. Admin can configure the system setting, add user and delete user which is doctor or staff. While staff can

add remove new patient. Staff also can view all the created appointment and send the SMS to notify the patient about their appointment and so on delete the expired appointments. Doctor and Patient together will make an appointment and treatment.

3.4 Implementation

During this phase, implement database system with java application are needed. System will connect the database that host at local host. System also need to completely integrated with the GSM modem to make sure the system can send the notification.

3.5 Testing

After the implementation is done, any error and problem which were not discovered in earlier phase will be corrected. Inevitably the system will need maintenance. Besides that, the application is developed to accommodate changes that could happen during the implementation period.

There are many reasons for the changes. Changes could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operation.

3.6 Evaluation and Maintenance

Once the coding for system is generated, the software program testing begins. This is a phase for evaluation and maintenance. GSM Modem can communicate with the application. All the data have been load into the database. Its include test the database performance, integrity, and concurrent access and security constraints. The test conditions conducted by comparing expected outcomes to actual outcomes.

CHAPTER 4

IMPLEMENTATION

System development is including the user interface design (UI), database design and the engine program which is the programming codes. Implementation involves those three criteria so that a system is preformed effectively and efficiently. So, in this chapter the system design and programming codes will be discussed as much as useful for better understanding about Clinic Management System with Notification Using GSM Modem.

The main purpose of this chapter is to deliver brief explanations and shows the important control in this system. Generally, this chapter is about to give details explanations to more about the design that were applied in this project. Basically, this system was developed by using NetBeans as the platform that used programming language in Java together with MySQL as the database platform.

This chapter is divided into two parts. First part will explains about database configuration and implementation for Clinic Management System with Notification Using GSM Modem while another part will explain on system codes/programming and interface for system functionalities/modules.

4.1 Database configuration and Implementation

PhpMyAdmin, MySQL database platform must be connected with the system engine before any programming codes involving database could be run properly.

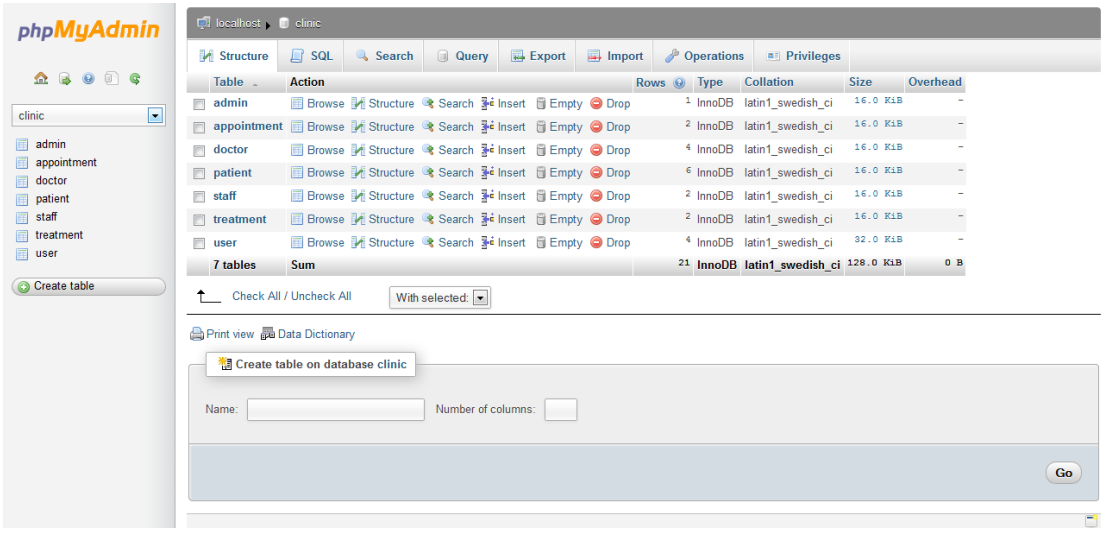


Figure 4.1 Database Properties

Figure 4.1 shows the table properties that have been added into PhpMyAdmin in order to create the database connection with the system engine according to figure 4.2 database design in entity relationship below.

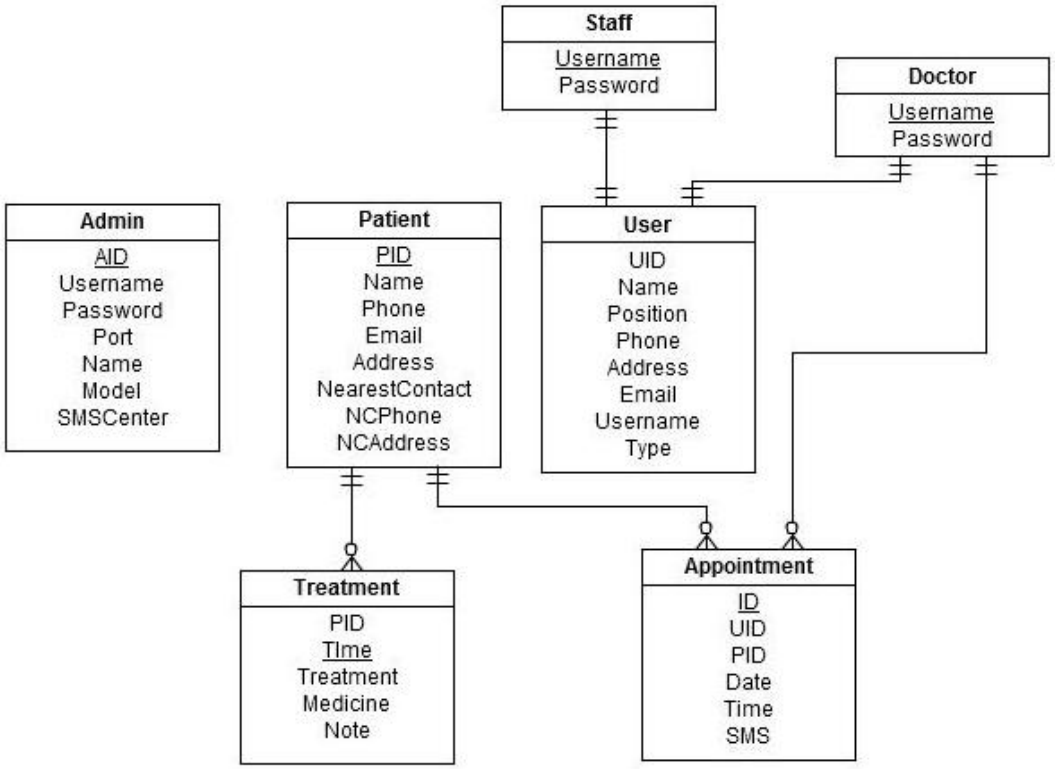


Figure 4.2 Entity Relationship

First table is for Admin which contains AID, Username, Password, Port for modem communication port, Name for modem's name, and SMSCenter. Admin table will use AID as a primary key.

Second table is for Patient that contains patient data which are PID, Name, Phone, Email, Address, NearestContact, NCPhone and NCAddress. This table will use PID as a primary key which will use a unique id for connecting between Treatment and Appointment table. Treatment table will contain treatment data for patient and a patient can have many treatment same with Appointment table. Appointment table contain appointment detail for Patient and Doctor which connected using UID and PID.

There also have three others table for system's user. User table will contain all the detail about user and will connecting with Staff and Doctor table for their login information.

4.2 System User Interfaces

4.2.1 Main Menu

Figure 4.3 Main Menu

In Figure 4.3 above shows the main form for this system. This form will appear when the system is executed.

```
/*
 * MainMenu.java
 *
 * Created on May 13, 2012, 5:23:38 PM
 */

package clinicmanagementsystem;

import java.sql.Connection;
import java.sql.Statement;
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
import javax.swing.JOptionPane;
/**
 *
 * @author areMaL
 */
public class MainMenu extends javax.swing.JFrame {
    /** Creates new form MainMenu */
    public MainMenu() {
        initComponents();
    }
     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new LogDoctor().setVisible(true);
        this.setVisible(false);
    }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        new LogStaff().setVisible(true);
        this.setVisible(false);
    }
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        new LogAdmin().setVisible(true);
        this.setVisible(false);
    }
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
    }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
    }
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            String StatusS = null;
            Statement stmt2 = connection.createStatement();
```

```
        String queryString2 = "SELECT * FROM admin" ;
        ResultSet rset2 = stmt2.executeQuery(queryString2);
        if(rset2.next()){
           StatusS = rset2.getString("STATUS");
         }
         if(StatusS.equals("1")){
            SystemStatusLabel1.setText("System is Online");
            SystemStatusLabel.setEnabled(true);
         }else{
            SystemStatusLabel1.setText("System is Offline");
            SystemStatusLabel.setEnabled(false);
         }
    }
        catch(Exception ex)
       {
        JOptionPane.showMessageDialog(null,ex.getMessage());
    }
}
/**
* @param args the command line arguments
*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new MainMenu().setVisible(true);
        }
    });
}}
```

4.2.2 Login

Figure 4.4 Doctor Log In Form

In Figure 4.4 above shows the login form for this system. This form will appear when the users (Admin, Staff and Doctor) want to use the system.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        try{
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
          Statement stmt = connection.createStatement();
          String username = UsernameField.getText();
          String password = PassField.getText();
          String queryString = "SELECT * FROM doctor WHERE USERNAME = '" + username + "'" ;
          ResultSet rset = stmt.executeQuery(queryString);
          String user = "";
          String pass = "";
          if(rset.next()){
             user = rset.getString("USERNAME");
             pass = rset.getString("PASSWORD");
          }
          String passwordDec = Crypt.decrypt(pass);
          if(username.equals(user) && password.equals(passwordDec)){
             new DoctorMenu(username).setVisible(true);
```

```
            setVisible(false);
            dispose();
        }
        else{
            JOptionPane.showMessageDialog(null,"Please Try Again","Authetication
Failed",JOptionPane.WARNING_MESSAGE);
    }    }
    catch(Exception exception){
    }  }
```

4.2.3 Doctor Menu

Figure 4.5 Doctor Menu

In Figure 4.5 above shows the Doctor Menu for this system. This form will appear when doctor successfully log in to the system. Doctor can access the Treat Patient Menu or View Appointment Menu.

```
public class DoctorMenu extends javax.swing.JFrame {
    String username2;
    String DID;
    /** Creates new form DoctorMenu */
    public DoctorMenu(String username) {
        initComponents();
        UsernameLabel.setText(username);
        username2 = username;
    }
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        new MainMenu().setVisible(true);
}
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        System.exit(0);
}
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        new ProfileMenu(username2).setVisible(true);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
    }
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt2 = connection.createStatement();
            String queryString2 = "SELECT * FROM user WHERE USERNAME = '" + username2 + "'" ;
            ResultSet rset2 = stmt2.executeQuery(queryString2);
            if(rset2.next()){
            IDLabel.setText(rset2.getString("UID"));
            NameLabel.setText(rset2.getString("NAME"));
           }}
            catch(Exception ex)
           {
            JOptionPane.showMessageDialog(null,ex.getMessage());
        }   }
    private void TreatButtonActionPerformed(java.awt.event.ActionEvent evt) {
        new TreatPatientS(username2).setVisible(true);
    }
    private void ViewApButtonActionPerformed(java.awt.event.ActionEvent evt) {
         DID = IDLabel.getText();
         new ViewAp(DID).setVisible(true);
    }
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            private String username;
```

```
public void run() {
    new DoctorMenu(username).setVisible(true);
} });    }
```

4.2.4 Edit Profile

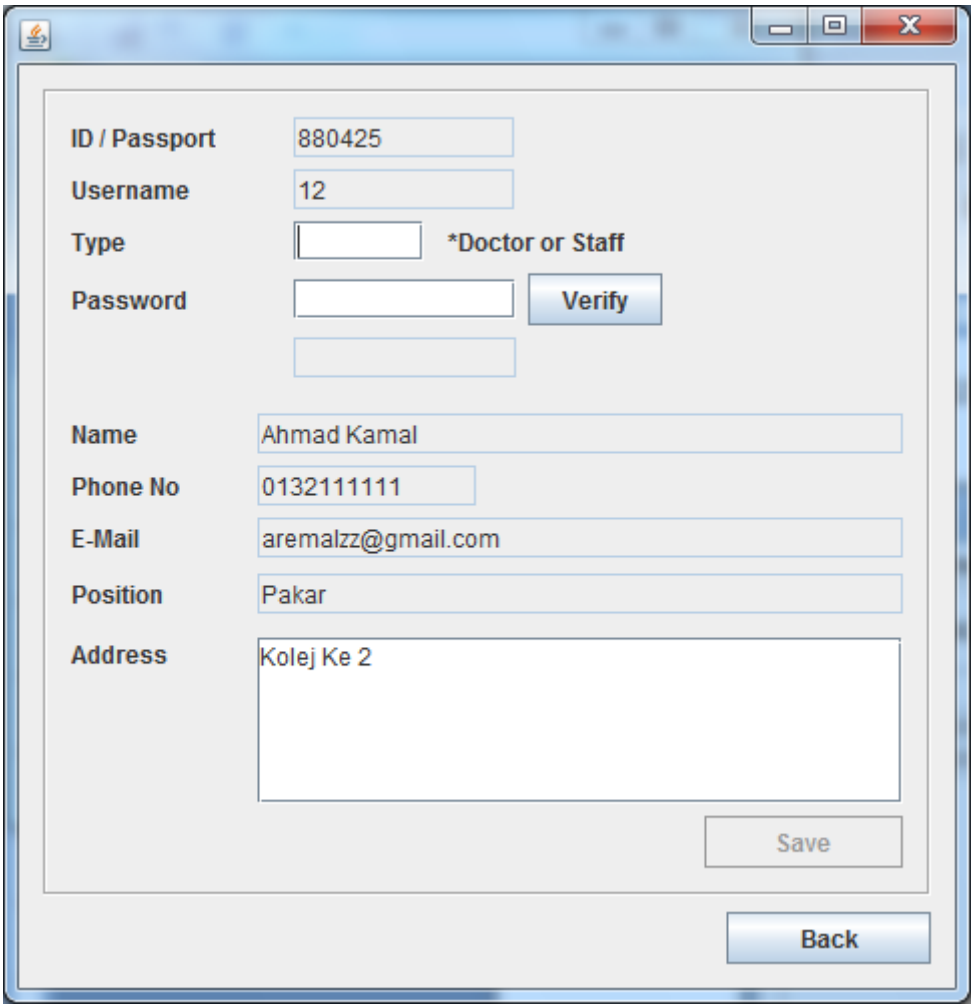


Figure 4.6 Edit Profile Form

In Figure 4.6 above shows the Edit Profile form. The users can edit their own profile by accessing this form from the menu bar.

```
public class ProfileMenu extends javax.swing.JFrame {
    private int Edit = 0;
    /** Creates new form ProfileMenu */
    public ProfileMenu(String username) {
        initComponents();
        UsernameField.setText(username);
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       dispose();
    }
```

```
    private void ConfirmButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (TypeField.getText().equalsIgnoreCase("doctor")){
       try{
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
          Statement stmt = connection.createStatement();
          String username = UsernameField.getText();
          String password = PassField.getText();
          String queryString = "SELECT * FROM doctor WHERE USERNAME = '" + username + "'" ;
          ResultSet rset = stmt.executeQuery(queryString);
          String user=null,pass=null;
          if(rset.next()){
             user = rset.getString("USERNAME");
             pass = rset.getString("PASSWORD");
          }
          String passwordDec = Crypt.decrypt(pass);
          if(username.equals(user) && password.equals(passwordDec)){
             if (Edit == 0) {
          SaveButton.setEnabled(true);
          ConfirmButton.setEnabled(false);
          NameField.setEditable(true);
          PhoneField.setEditable(true);
          EmailField.setEditable(true);
          AddressArea.setEditable(true);
          PassField.setEditable(true);
          Pass2Field.setEditable(true);
          PositionField.setEditable(true);
          TypeField.setEditable(false);
          Edit = 1;
          }
          Edit = 0;
          }
       }
       catch(Exception ex)
         {
          JOptionPane.showMessageDialog(null,ex.getMessage());
       }}
     else if (TypeField.getText().equalsIgnoreCase("staff")){
          try{
```

```
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            String username = UsernameField.getText();
            String password = PassField.getText();
            String queryString = "SELECT * FROM staff WHERE USERNAME = '" + username + "'";
            ResultSet rset = stmt.executeQuery(queryString);
            String user=null,pass=null;
            if(rset.next()){
               user = rset.getString("USERNAME");
               pass = rset.getString("PASSWORD");
            }
            String passwordDec = Crypt.decrypt(pass);
            if(username.equals(user) && password.equals(passwordDec)){
               if (Edit == 0) {
            SaveButton.setEnabled(true);
            ConfirmButton.setEnabled(false);
            NameField.setEditable(true);
            PhoneField.setEditable(true);
            EmailField.setEditable(true);
            AddressArea.setEditable(true);
            PassField.setEditable(true);
            Pass2Field.setEditable(true);
            PositionField.setEditable(true);
            TypeField.setEditable(false);
            Edit = 1;
            }
            Edit = 0;
            }
         }
         catch(Exception ex)
           {
            JOptionPane.showMessageDialog(null,ex.getMessage());
         }}
       else{
            JOptionPane.showMessageDialog(null,"Please Try Again","Authetication
Failed",JOptionPane.WARNING_MESSAGE);
      }   }
      private void SaveButtonActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        String pass1;
    String pass2;
    pass1 = PassField.getText();
    pass2 = Pass2Field.getText();
    if(!(pass1.equalsIgnoreCase(pass2))){
     JOptionPane.showMessageDialog(null,"Password does not
match","Error",JOptionPane.WARNING_MESSAGE);
    }else{
    if (! (TypeField.getText().equalsIgnoreCase("doctor")) &
!((TypeField.getText().equalsIgnoreCase("staff")))){
        JOptionPane.showMessageDialog(null,"Please fill the Type Field corectly (Doctor /
Staff)","Error",JOptionPane.WARNING_MESSAGE);
    }else{
          try{
             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
             Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
             Statement stmt = connection.createStatement();
             Statement stmt1 = connection.createStatement();
             String Name = NameField.getText();
             String Phone = PhoneField.getText();
             String Address = AddressArea.getText();
             String Email = EmailField.getText();
             String Position = PositionField.getText();
             String Username = UsernameField.getText();
             String Password = PassField.getText();
             String passwordEnc = Crypt.encrypt(Password);
             String Type = TypeField.getText();
             if (Name.equals("")| Phone.equals("")| Address.equals("")| Email.equals("")|
Position.equals("")){
             JOptionPane.showMessageDialog(null,"Please fill all the
field","Error",JOptionPane.WARNING_MESSAGE);
             }
             else{
             if (Type.equalsIgnoreCase("Doctor")){
             stmt1.executeUpdate("UPDATE doctor SET PASSWORD = '" + passwordEnc + "'  WHERE
USERNAME = '" + Username + "'") ;
             stmt.executeUpdate("UPDATE user SET NAME = '" + Name + "' , PHONE = '" + Phone + "' ,
ADDRESS = '" + Address + "' , EMAIL = '" + Email + "' , POSITION = '" + Position +"'  WHERE
USERNAME = '" + Username + "');
```

```
            }
            else if (Type.equalsIgnoreCase("Staff")){
            stmt1.executeUpdate("UPDATE staff SET PASSWORD = '" + passwordEnc + "'   WHERE
USERNAME = '" + Username + "'") ;
            stmt.executeUpdate("UPDATE  user SET Name = '" + Name + "' , PHONE = '" + Phone + "'
,ADDRESS = '" + Address + "' ,EMAIL = '" + Email + "' ,  POSITION = '" + Position +"'   WHERE
USERNAME = '" + Username + "'");
            }
             JOptionPane.showMessageDialog(null,"User "+Username+" have been successfully
saved","Success",JOptionPane.INFORMATION_MESSAGE);
            }   }
          catch(Exception ex)
          {
          JOptionPane.showMessageDialog(null,ex.getMessage());
          NameField.setEditable(false);
          PhoneField.setEditable(false);
          EmailField.setEditable(false);
          AddressArea.setEditable(false);
          PositionField.setEditable(false);
          UsernameField.setEditable(false);
          PassField.setEditable(false);
          Pass2Field.setEditable(false);
          TypeField.setEditable(false);
          Edit = 0;
       }  }   }   }
   private void formWindowIconified(java.awt.event.WindowEvent evt) {
      // TODO add your handling code here:
   }
   private void formWindowOpened(java.awt.event.WindowEvent evt) {
      try{
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
          String username = UsernameField.getText();
          Statement stmt2 = connection.createStatement();
          String queryString2 = "SELECT * FROM user WHERE USERNAME = '" + username + "' ;
          ResultSet rset2 = stmt2.executeQuery(queryString2);
          if(rset2.next()){
          IDField.setText(rset2.getString("UID"));
          NameField.setText(rset2.getString("NAME"));
```

```
        PhoneField.setText(rset2.getString("PHONE"));
        EmailField.setText(rset2.getString("EMAIL"));
        AddressArea.setText(rset2.getString("ADDRESS"));
        PositionField.setText(rset2.getString("POSITION"));
           }}
        catch(Exception ex)
       {
        JOptionPane.showMessageDialog(null,ex.getMessage());
    }   }
/**
* @param args the command line arguments
*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        private String username;
        public void run() {
            new ProfileMenu(username).setVisible(true);
        }     });    }
```

4.2.5 Treatment Form

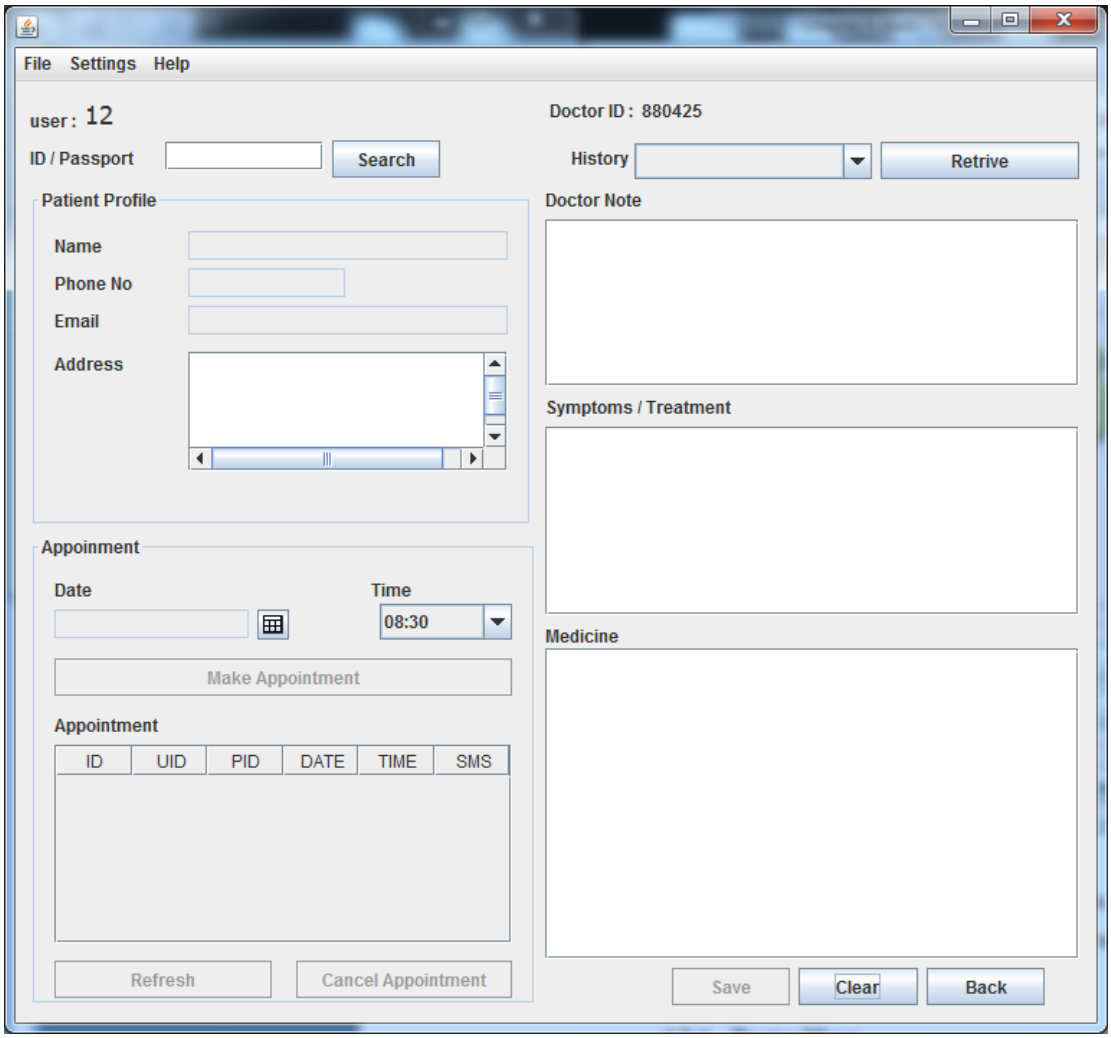


Figure 4.7 Treatment Form

In Figure 4.7 shows the Treat Patient Menu for this system which only can be access by doctor. This form will appear when doctors want to serve their patient.

```
public class TreatPatientS extends javax.swing.JFrame {
    int AddUser ;
    /** Creates new form TreatPatientS */
    public TreatPatientS(String username2) {
        initComponents();
        UserLabel.setText(username2);
    }
    private void RetriveButtonActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        try{
            String Value = HComboBox.getSelectedItem().toString();
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            String queryString = "SELECT * FROM treatment WHERE TIME = '" + Value + "'" ;
            ResultSet rset = stmt.executeQuery(queryString);
            if(rset.next()){
                TArea.setText(rset.getString("TREATMENT"));
                MArea.setText(rset.getString("MEDICINE"));
                NoteArea.setText(rset.getString("NOTE"));
            }
        } catch(Exception e){
        }
}
    private void SearchButtonActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            String ID = IDField.getText();
            String username = UserLabel.getText();
            Statement stmt2 = connection.createStatement();
            String queryString2 = "SELECT * FROM user WHERE USERNAME = '" + username + "'" ;
            ResultSet rset2 = stmt2.executeQuery(queryString2);
            if(rset2.next()){
                DoctorIDLabel.setText(rset2.getString("UID"));
            }
            String queryString = "SELECT * FROM patient WHERE PID = '" + ID + "'" ;
            ResultSet rset = stmt.executeQuery(queryString);
            if(rset.next()){
                NameField.setText(rset.getString("NAME"));
                PhoneField.setText(rset.getString("PHONE"));
                EmailField.setText(rset.getString("EMAIL"));
                AddressArea.setText(rset.getString("ADDRESS"));

                SearchButton.setEnabled(false);
                SaveButton.setEnabled(true);
```

```
            MakeApButton.setEnabled(true);
            RefreshApButton.setEnabled(true);
            CancelApButton.setEnabled(true);
            IDField.setEditable(false);
            AddUser = 0;
            Statement st=connection.createStatement();
            ResultSet rs=st.executeQuery("SELECT * from treatment WHERE PID = ('" + ID + "')");
            while(rs.next()){
             HComboBox.addItem(rs.getTimestamp("TIME"));
            }
            ResultSet rset3 = null;
            PreparedStatement pst = null;
            String sql = "SELECT * FROM appointment WHERE PID = '" + ID + "'";
            pst = connection.prepareStatement(sql);
            rset3 = pst.executeQuery();
            jTable1.setModel(DbUtils.resultSetToTableModel(rset3));
            JOptionPane.showMessageDialog(null,"User
Found","Done",JOptionPane.INFORMATION_MESSAGE);
          }
       } catch(Exception ex)
       {
          JOptionPane.showMessageDialog(null,ex.getMessage());
       }
}
   private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
      dispose();
}
   private void SaveButtonActionPerformed(java.awt.event.ActionEvent evt) {
      try{
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
          Statement stmt = connection.createStatement();
          String Medicine = MArea.getText();
          String Treatment = TArea.getText();
          String Note = NoteArea.getText();
          String ID = IDField.getText();
          stmt.executeUpdate("INSERT INTO treatment (TREATMENT,MEDICINE,NOTE,PID)
VALUES ('" + Treatment + "' , '" + Medicine + "', '" + Note + "', '" + ID + "')");
```

```
            JOptionPane.showMessageDialog(null,"Treatment record have been successfully
saved","Success",JOptionPane.INFORMATION_MESSAGE);
        }
        catch(Exception ex)
        {         {
                JOptionPane.showMessageDialog(null,ex.getMessage());
            }       }}
    private void RefreshApButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String ID = IDField.getText();
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            ResultSet rset3 = null;
            PreparedStatement pst = null;
            String sql = "SELECT * FROM appointment WHERE PID = '" + ID + "'";
            pst = connection.prepareStatement(sql);
            rset3 = pst.executeQuery();
            jTable1.setModel(DbUtils.resultSetToTableModel(rset3));
        } catch(Exception exception){
        }}
    private void CancelApButtonActionPerformed(java.awt.event.ActionEvent evt) {
        int option = JOptionPane.showConfirmDialog((Component)
                null, "Are you sure want to cancel the appointment", "Alert",
JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION ) {
            String ApID="";
            int row = jTable1.getSelectedRow();
            ApID = jTable1.getModel().getValueAt(row, 0).toString();
            if(ApID.equals("")){
                JOptionPane.showMessageDialog(null,"Please select the appointment first");
            }else{
                try{
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
                    Statement stmt = connection.createStatement();
                    String queryString = "DELETE FROM appointment WHERE ID = '" + ApID + "'" ;
                    stmt.executeUpdate(queryString);
```

```
                ResultSet rset3 = null;
                PreparedStatement pst = null;
                String sql = "SELECT * FROM appointment WHERE PID = '" + ApID + "'";
                pst = connection.prepareStatement(sql);
                rset3 = pst.executeQuery();
                jTable1.setModel(DbUtils.resultSetToTableModel(rset3));
                JOptionPane.showMessageDialog(null,"Appointment data have been
deleted.","Success",JOptionPane.INFORMATION_MESSAGE);
            } catch(Exception exception){
            }          }       } else if (option == JOptionPane.CANCEL_OPTION) {
        }   }
   private void ClearButtonActionPerformed(java.awt.event.ActionEvent evt) {
      IDField.setText("");
      NameField.setText("");
      PhoneField.setText("");
      EmailField.setText("");
      AddressArea.setText("");
      NoteArea.setText("");
      MArea.setText("");
      TArea.setText("");
      DateTextField.setText("");
      HComboBox.removeAllItems();
      IDField.setEditable(true);
      SearchButton.setEnabled(true);
      MakeApButton.setEnabled(false);
      RefreshApButton.setEnabled(false);
      CancelApButton.setEnabled(false);
      SaveButton.setEnabled(false);
      AddUser = 0;
   }
   private void MakeApButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if(DateTextField.getText().equalsIgnoreCase("")){
           JOptionPane.showMessageDialog(null,"Please select the appointment date first");
      }else{
      try{
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
         Statement stmt = connection.createStatement();
         String uid = DoctorIDLabel.getText();
```

```
        String pid = IDField.getText();
        String date =  DateTextField.getText();
        String time = TimeComboBox.getSelectedItem().toString();
        String SMS = "Draft";
        stmt.executeUpdate("INSERT INTO appointment (UID,PID,DATE,TIME,SMS) VALUES ('" +
uid + "' , '" + pid + "', '" + date + "', '" + time + "', '" + SMS + "')");
        JOptionPane.showMessageDialog(null,"Appointment has been booked
successfully","Success",JOptionPane.INFORMATION_MESSAGE);
      }
      catch(Exception ex)
      {          {              JOptionPane.showMessageDialog(null,ex.getMessage());
        }       }       }}
   private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
      System.exit(0);
}                                   private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt)
{
      new Console().setVisible(true);
}
   private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
      new About().setVisible(true);
}
   private void dateFocusLost(java.awt.event.FocusEvent evt) {
      String date = DateTextField.getText();
      setDate(date);
   }
      private void dateOnlyPopupChanged(java.beans.PropertyChangeEvent evt) {
      if (evt.getNewValue() instanceof Date)
        setDate((Date)evt.getNewValue());
   }
      public void setDate(String dateString)
   {
                    Date date = null;
                    try        {
        if ((dateString != null) && (dateString.length() > 0))
          date = dateFormat.parse(dateString);
                    } catch (Exception e)       {
        date = null;
                    }
      this.setDate(date);
   }
```

```
    public void setDate(Date date)
{
    String dateString = "";
    if (date != null)
                    dateString = dateFormat.format(date);
    DateTextField.setText(dateString);
    jCalendarButton1.setTargetDate(date);
}
/**
* @param args the command line arguments
*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            String username2=null;
            new TreatPatientS(username2).setVisible(true);
        }
    });
}
```

4.2.6 Doctor's Appointment View

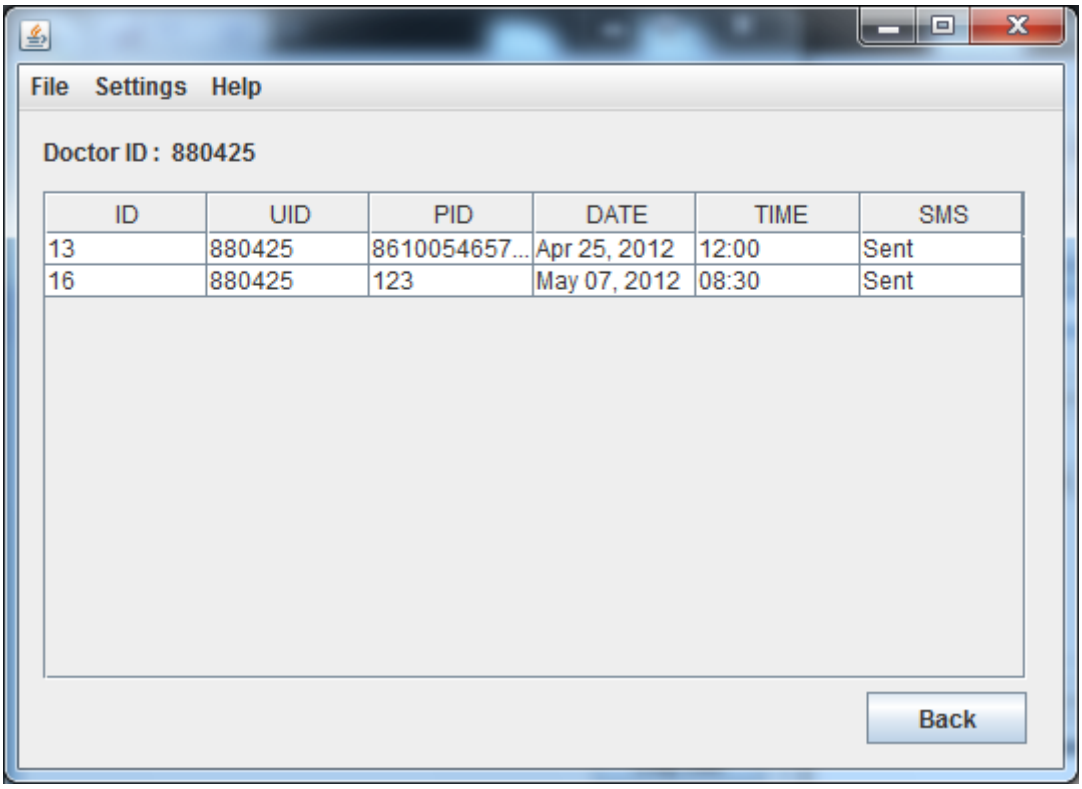


Figure 4.8 Appointment View for Doctor

In Figure 4.8 above shows the view of Doctor. The doctor will see only his appointment only.

```
public class ViewAp extends javax.swing.JFrame {
    /** Creates new form ViewAp */
    public ViewAp(String DID) {
        initComponents();
        DoctorIDLabel.setText(DID);
    }
private void formWindowOpened(java.awt.event.WindowEvent evt) {
        try{
            String DID = DoctorIDLabel.getText();
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            ResultSet rset = null;
            PreparedStatement pst = null;
            String sql = "SELECT * FROM appointment WHERE UID = '" + DID + "'";
```

```
            pst = connection.prepareStatement(sql);
            rset = pst.executeQuery();
            jTable1.setModel(DbUtils.resultSetToTableModel(rset));
            }
             catch(Exception ex){
             JOptionPane.showMessageDialog(null,ex.getMessage());
        }
    }
    private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
    }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
}
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                String DID = null;
                new ViewAp(DID).setVisible(true);
            }
        });
    }
```

4.2.7 Staff Menu

Figure 4.9 Staff Menu

In Figure 4.9 above shows the view of Staff Menu. This form will appear when staff successfully log in to the system.

```
public class StaffMenu extends javax.swing.JFrame {
    String username2;
    /** Creates new form DoctorMenu */
    public StaffMenu(String username) {
        initComponents();
        UsernameLabel.setText(username);
        username2 = username;
    } private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        new MainMenu().setVisible(true);
}
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
```

```
}
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        new ProfileMenu(username2).setVisible(true);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
    }
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt2 = connection.createStatement();
            String queryString2 = "SELECT * FROM user WHERE USERNAME = '" + username2 + "'" ;
            ResultSet rset2 = stmt2.executeQuery(queryString2);
            if(rset2.next()){
            IDLabel.setText(rset2.getString("UID"));
            NameLabel.setText(rset2.getString("NAME"));
           }}
            catch(Exception ex)
           {        JOptionPane.showMessageDialog(null,ex.getMessage());
        }
    }
    private void SearchPatientButtonActionPerformed(java.awt.event.ActionEvent evt) {
        new EditPatient().setVisible(true);
    }
    private void AddPatientButtonActionPerformed(java.awt.event.ActionEvent evt) {
        new AddPatient().setVisible(true);
    }
    private void ViewApButtonActionPerformed(java.awt.event.ActionEvent evt) {
        new SMSAP().setVisible(true);
    }
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
    private String username;
    public void run() {
        new StaffMenu(username).setVisible(true);
```

4.2.8 Add New Patient

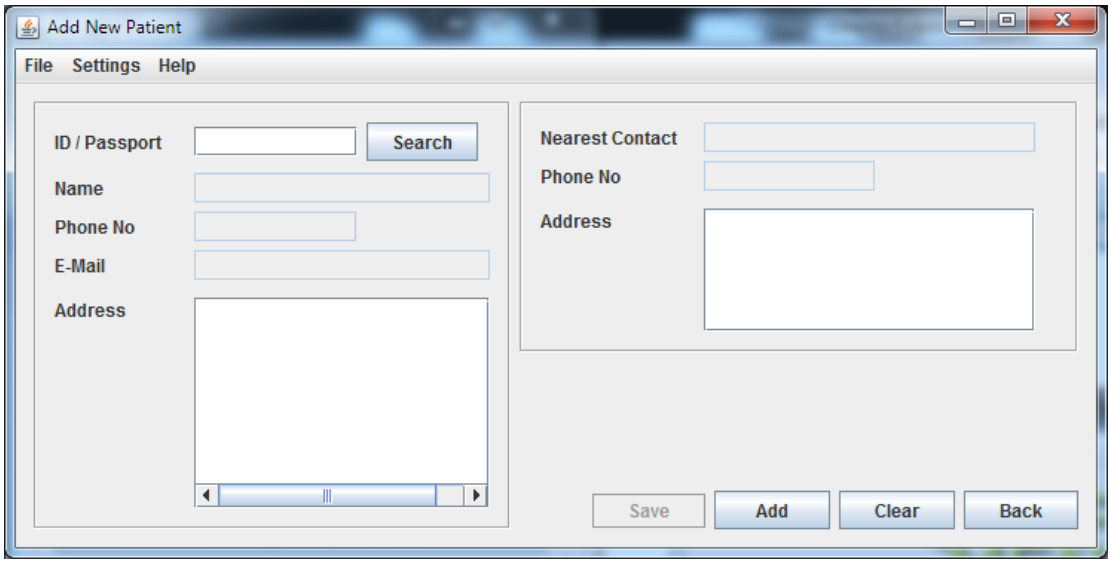


Figure 4.10 Add New Patient Form

In Figure 4.10 above shows the Add New Patient. This form will appear when staffs want to register or add new patient to the system.

```
public class AddPatient extends javax.swing.JFrame {
    private int AddUser;
    /** Creates new form AddPatient */
    public AddPatient() {
        initComponents();
    } private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
    }
    private void ClearButtonActionPerformed(java.awt.event.ActionEvent evt) {
        IDField.setText("");
        NameField.setText("");
        PhoneField.setText("");
        EmailField.setText("");
        AddressArea.setText("");
        NContactField.setText("");
        NCPhoneField.setText("");
        NCAddressArea.setText("");
        IDField.setEnabled(true);
```

```
        SearchButton.setEnabled(true);
        SaveButton.setEnabled(false);
        AddButton.setEnabled(true);
       AddUser = 0;
    }
    private void SearchButtonActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            String ID = IDField.getText();
            String queryString = "SELECT * FROM patient WHERE PID = '" + ID + "'" ;
            ResultSet rset = stmt.executeQuery(queryString);
            if(rset.next()){
                NameField.setText(rset.getString("NAME"));
                PhoneField.setText(rset.getString("PHONE"));
                EmailField.setText(rset.getString("EMAIL"));
                AddressArea.setText(rset.getString("ADDRESS"));
                NContactField.setText(rset.getString("NEARESTCONTACT"));
                NCPhoneField.setText(rset.getString("NCPHONE"));
                NCAddressArea.setText(rset.getString("NCADDRESS"));
                JOptionPane.showMessageDialog(null,"Patient
Found","Done",JOptionPane.INFORMATION_MESSAGE);
                SaveButton.setEnabled(false);
                AddButton.setEnabled(false);
            }
        }
        catch(Exception ex)
        {
            JOptionPane.showMessageDialog(null,ex.getMessage());
           }
    }
    private void AddButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (IDField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(null,"Please fill the ID");
        }else{
        IDField.setEnabled(false);
        NameField.setEditable(true);
        PhoneField.setEditable(true);
```

```
        EmailField.setEditable(true);
        AddressArea.setEditable(true);
        NContactField.setEditable(true);
        NCPhoneField.setEditable(true);
        NCAddressArea.setEditable(true);
        IDField.setEditable(true);
        SearchButton.setEnabled(false);
        SaveButton.setEnabled(true);
        AddButton.setEnabled(true);
        AddUser = 1;
              }
    }

    private void SaveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            Statement stmt1 = connection.createStatement();
            String ID = IDField.getText();
            String Name = NameField.getText();
            String Phone = PhoneField.getText();
            String Address = AddressArea.getText();
            String Email = EmailField.getText();
            String NContact = NContactField.getText();
            String NCPhone = NCPhoneField.getText();
            String NCAddress = NCAddressArea.getText();
            if (ID.equals("") | Name.equals("")| Phone.equals("")| Address.equals("")| Email.equals("")|
NContact.equals("")|NCPhone.equals("")| NCAddress.equals("")){
                JOptionPane.showMessageDialog(null,"Please fill the data
corectly","Error",JOptionPane.WARNING_MESSAGE);
            }
            else{
            stmt.executeUpdate("INSERT INTO patient
(PID,NAME,EMAIL,PHONE,ADDRESS,NEARESTCONTACT,NCPHONE,NCADDRESS) VALUES
('" + ID + "' , '" + Name + "', '" + Email + "', '" + Phone + "', '" + Address + "', '" + NContact + "', '" +
NCPhone + "' , '" + NCAddress + "')");
                IDField.setText("");
                NameField.setText("");
```

```
            PhoneField.setText("");
            EmailField.setText("");
            AddressArea.setText("");
            NContactField.setText("");
            NCPhoneField.setText("");
            NCAddressArea.setText("");
            IDField.setEnabled(true);
            SearchButton.setEnabled(true);
            SaveButton.setEnabled(false);
            AddUser = 0;
            JOptionPane.showMessageDialog(null,"Patient "+Name+" have been added
successfully","Success",JOptionPane.INFORMATION_MESSAGE);
          }
          }
       catch(Exception ex)
       {
          JOptionPane.showMessageDialog(null,ex.getMessage());
       }
    }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
       System.exit(0);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
       new Console().setVisible(true);
}
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
       new About().setVisible(true);
}
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
       java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
             new AddPatient().setVisible(true);
          }
       });    }
```

4.2.9 View Patient Detail

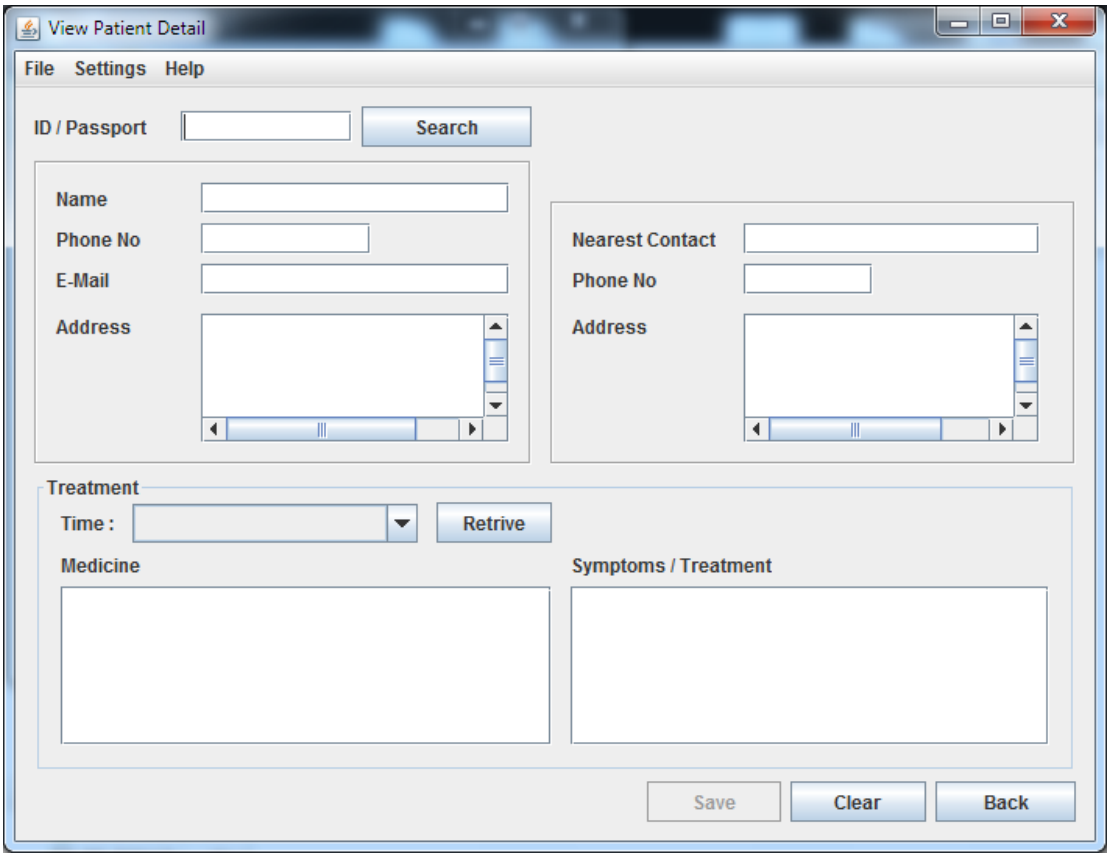


Figure 4.11 View Patient Detail Form

In Figure 4.11 shows the View Patient Detail form which only can be access by staff. Staff can view the detail of patient information including the treatment that has been made by doctor.

```
public class EditPatient extends javax.swing.JFrame {
    int AddUser ;
   /** Creates new form EditPatient */
   public EditPatient() {
       initComponents();
   } private void RetriveButtonActionPerformed(java.awt.event.ActionEvent evt) {
       try{
      String Value = HComboBox2.getSelectedItem().toString();
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
       Statement stmt = connection.createStatement();
          String queryString = "SELECT * FROM treatment WHERE TIME = '" + Value + "'" ;
          ResultSet rset = stmt.executeQuery(queryString);
```

```
            if(rset.next()){
                TArea.setText(rset.getString("TREATMENT"));
                MArea.setText(rset.getString("MEDICINE"));
            }       }
        catch(Exception e){}
    }
    private void SearchButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (IDField.getText().equalsIgnoreCase("")){
            JOptionPane.showMessageDialog(null,"Please fill the ID");
        }else{
            try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            String ID = IDField.getText();
            String queryString = "SELECT * FROM patient WHERE PID = '" + ID + "'" ;
            ResultSet rset = stmt.executeQuery(queryString);
            if(rset.next()){
                NameField.setText(rset.getString("NAME"));
                PhoneField.setText(rset.getString("PHONE"));
                EmailField.setText(rset.getString("EMAIL"));
                AddressArea.setText(rset.getString("ADDRESS"));
                NearContactField.setText(rset.getString("NEARESTCONTACT"));
                NCPhoneField.setText(rset.getString("NCPHONE"));
                NCAddressArea.setText(rset.getString("NCADDRESS"));
                Statement st=connection.createStatement();
                ResultSet rs=st.executeQuery("SELECT * from treatment WHERE PID = ('" + ID + "')");
                while(rs.next()){
                HComboBox2.addItem(rs.getTimestamp("TIME"));}
                SearchButton.setEnabled(false);
                SaveButton.setEnabled(true);
                IDField.setEditable(false);
                JOptionPane.showMessageDialog(null,"User
Found","Done",JOptionPane.INFORMATION_MESSAGE);
            }

        }
        catch(Exception ex)
          {
```

```
            JOptionPane.showMessageDialog(null,ex.getMessage());
        }       }    }
    private void SaveButtonActionPerformed(java.awt.event.ActionEvent evt) {
      try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            String ID = IDField.getText();
            String Name = NameField.getText();
            String Phone = PhoneField.getText();
            String Address = AddressArea.getText();
            String Email = EmailField.getText();
            String NContact = NearContactField.getText();
            String NCPhone = NCPhoneField.getText();
            String NCAddress = NCAddressArea.getText();
            stmt.executeUpdate("UPDATE patient SET NAME = '" + Name + "' , PHONE = '" + Phone + "'
, ADDRESS = '" + Address + "' , EMAIL = '" + Email + "' , NEARESTCONTACT = '" + NContact +"' ,
NCPHONE = '" + NCPhone +"', NCADDRESS = '" + NCAddress +"'  WHERE PID = '" + ID + "'");
             JOptionPane.showMessageDialog(null,"Patient "+Name+" data have been successfully
saved","Success",JOptionPane.INFORMATION_MESSAGE);
            }
          catch(Exception ex)
          {{
          JOptionPane.showMessageDialog(null,ex.getMessage());
        }    }    }
    private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
       dispose();
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       IDField.setText("");
       NameField.setText("");
       PhoneField.setText("");
       EmailField.setText("");
       AddressArea.setText("");
       MArea.setText("");
       TArea.setText("");
       NearContactField.setText("");
       NCPhoneField.setText("");
       NCAddressArea.setText("");
```

```
        HComboBox2.removeAllItems();
        IDField.setEditable(true);
        SearchButton.setEnabled(true);
        SaveButton.setEnabled(false);
        AddUser = 0;
    }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
}
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new EditPatient().setVisible(true);
            }
        });
```

4.2.10 View Appointment

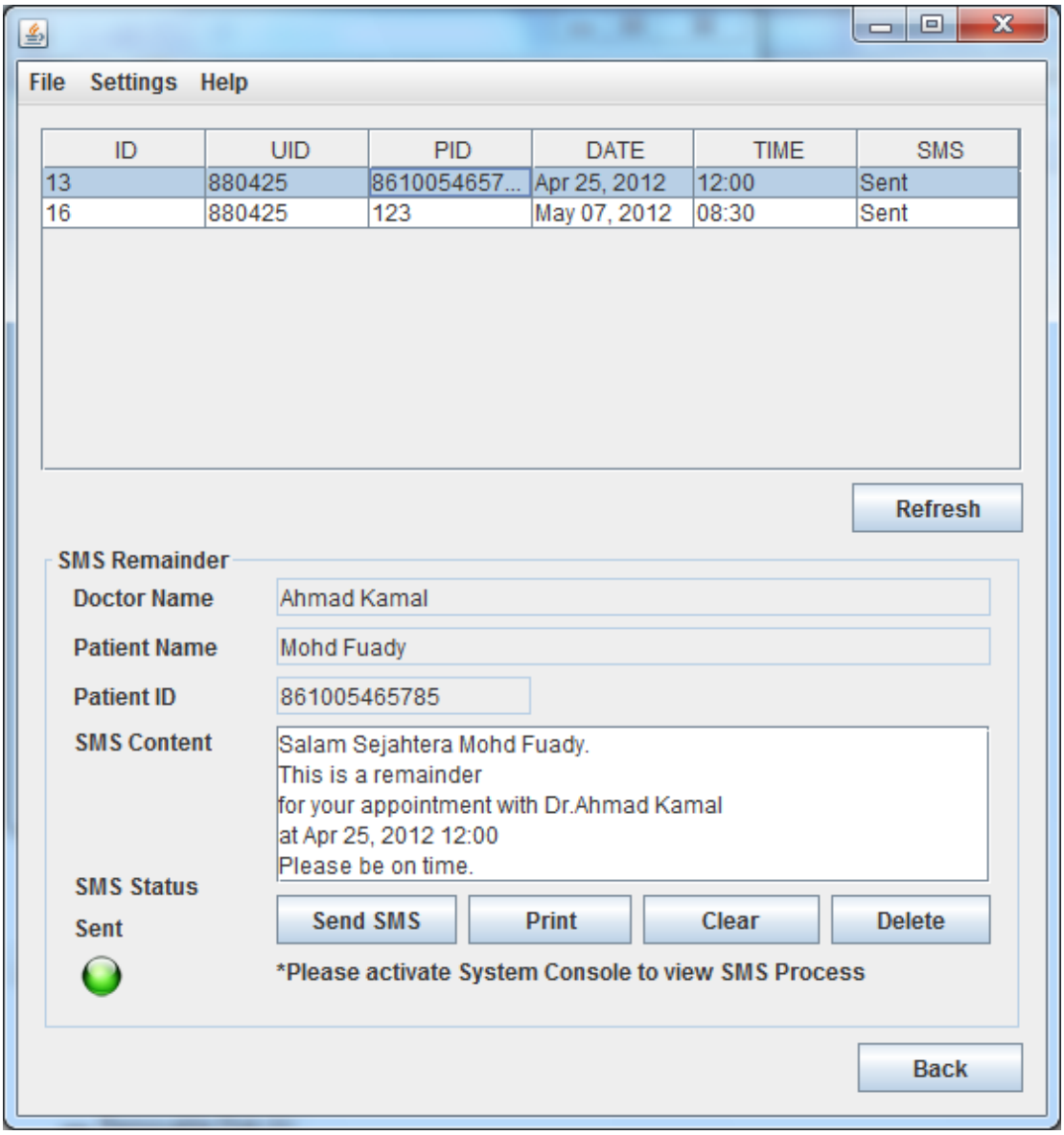


Figure 4.12 View Appointment Form

In Figure 4.12 shows the View Appointment form, which only can be access by staff. Staff can view the detail of every appointment that has been made by every registered doctor in the system. Staff also can notify the patient for the appointment by accessing a send SMS button in this form.

```
public class SMSAP extends javax.swing.JFrame {
    String Phone;
    String SMS;
    String SMSID;
    String MName;
    String MModel;
    String MPort;
    String SMSNo;
    String PrintDetail = "test print";
    private static Font fnt = new Font("Tahoma",Font.PLAIN,15);
    /** Creates new form SMSAP */
    public SMSAP() {
        initComponents();
    } private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
        int row = jTable1.getSelectedRow();
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            Statement stmt2 = connection.createStatement();
            String queryString2 = "SELECT * FROM user WHERE UID = '" +
jTable1.getModel().getValueAt(row, 1).toString() + "'" ;
            ResultSet rset2 = stmt2.executeQuery(queryString2);
            if(rset2.next()){
                DNameField.setText(rset2.getString("NAME"));
            }
            String queryString = "SELECT * FROM patient WHERE PID = '" +
jTable1.getModel().getValueAt(row, 2).toString() + "'" ;
            ResultSet rset = stmt.executeQuery(queryString);
            if(rset.next()){
                PNameField.setText(rset.getString("NAME"));
                Phone = rset.getString("PHONE");
            }
            String queryString3 = "SELECT * FROM appointment WHERE PID = '" +
jTable1.getModel().getValueAt(row, 2).toString() + "'" ;
            ResultSet rset3 = stmt.executeQuery(queryString3);
            if(rset3.next()){
                SMSID = rset3.getString("SMS");
            }       }
```

```
        catch(Exception ex)

        {
            JOptionPane.showMessageDialog(null,ex.getMessage());
        }
        PIDField.setText(jTable1.getModel().getValueAt(row, 2).toString());
        SMS = jTable1.getModel().getValueAt(row, 5).toString();
        SMSArea.setText("Salam Sejahtera " + PNameField.getText()+ ".\nThis is a remainder \nfor your
appointment with Dr."
                + DNameField.getText() +"\nat " +jTable1.getModel().getValueAt(row, 3).toString()+" "+
jTable1.getModel().getValueAt(row, 4).toString()+
                "\nPlease be on time.");
        if(SMSID.equals("Sent")){
            SendLabel.setText("Sent");
            SendGLabel2.setEnabled(true);
        }else if(SMSID.equals("Draft")){
            SendLabel.setText("Draft");
            SendGLabel2.setEnabled(false);
        }    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        UpdateTable();
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt2 = connection.createStatement();
            String queryString2 = "SELECT * FROM admin" ;
            ResultSet rset2 = stmt2.executeQuery(queryString2);
            if(rset2.next()){
            MName = (rset2.getString("NAME"));
            MModel = (rset2.getString("MODEL"));
            MPort = (rset2.getString("PORT"));
            SMSNo = (rset2.getString("SMSCENTER"));
             }}
            catch(Exception ex)
            {
            JOptionPane.showMessageDialog(null,ex.getMessage());
        }
    }
```

```
   private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
      dispose();
   }
   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      if(PIDField.getText().equals("")){
      JOptionPane.showMessageDialog (null,"Please select the appointment first");
      }else{
      int option = JOptionPane.showConfirmDialog((Component)
      null, "Are you sure want to send a SMS using this setting :"
            + "\nModem Name : " +MName+ "   Modem Port  :" + MPort+ "   Center No : " + SMSNo+
"\n"
            + "\n        SMS Process will take some time to be done. Please wait patiently.          ", "Alert",
JOptionPane.OK_CANCEL_OPTION);

       if (option == JOptionPane.OK_OPTION ) {
            try
                  {                     doIt();              }
                  catch (Exception e)
                  {}      }
      else if (option == JOptionPane.CANCEL_OPTION) {
       JOptionPane.showMessageDialog (null,"The operation has been cancelled");
      }       }    }
   private void ClearButtonActionPerformed(java.awt.event.ActionEvent evt) {
         PIDField.setText("");
         DNameField.setText("");
         PNameField.setText("");
         SMSArea.setText("");
         SendGLabel2.setEnabled(false);
   }
   private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      int row = jTable1.getSelectedRow();
     String ApID = jTable1.getModel().getValueAt(row, 0).toString();
      if(PIDField.getText().equals("")){
         JOptionPane.showMessageDialog (null,"Please select the appointment first");
      }else{
      int option = JOptionPane.showConfirmDialog((Component)
            null, "Are you sure want to deletel the appointment", "Alert",
JOptionPane.OK_CANCEL_OPTION);
      if (option == JOptionPane.YES_OPTION ) {
```

```
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
                String queryString = "DELETE FROM appointment WHERE ID = '" + ApID + "'" ;
                stmt.executeUpdate(queryString);
                ResultSet rset3 = null;
                PreparedStatement pst = null;
                String sql = "SELECT * FROM appointment WHERE PID = '" + ApID + "'";
                pst = connection.prepareStatement(sql);
                rset3 = pst.executeQuery();
                jTable1.setModel(DbUtils.resultSetToTableModel(rset3));
        JOptionPane.showMessageDialog(null,"Appointment data have been
deleted.","Success",JOptionPane.INFORMATION_MESSAGE);
        }       catch(Exception exception){
        }       }       else if (option == JOptionPane.CANCEL_OPTION) {
        }       }   }
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
                ResultSet rset3 = null;
                PreparedStatement pst = null;
                String sql = "SELECT * FROM appointment";
                pst = connection.prepareStatement(sql);
                rset3 = pst.executeQuery();
                jTable1.setModel(DbUtils.resultSetToTableModel(rset3));
                SendGLabel2.setEnabled(false);
        }
        catch(Exception exception){
        }   }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
}
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
      new About().setVisible(true); public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new SMSAP().setVisible(true);
         }       });    }
   private void UpdateTable(){
            try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            ResultSet rset = null;
            PreparedStatement pst = null;
            String sql = "SELECT * FROM appointment";
            pst = connection.prepareStatement(sql);
            rset = pst.executeQuery();
            jTable1.setModel(DbUtils.resultSetToTableModel(rset));
            }
             catch(Exception ex)
         {{
         JOptionPane.showMessageDialog(null,ex.getMessage());
       }    }    }
    public void doIt() throws Exception
          {
            OutboundNotification outboundNotification = new OutboundNotification();
                     System.out.println("Sending message from a serial gsm modem.");
                     SerialModemGateway gateway = new SerialModemGateway("MODEM", MPort ,
115200,MName,MModel);
                     gateway.setInbound(true);
                     gateway.setOutbound(true);
                     gateway.setSimPin("0000");
                     // Explicit SMSC address set is required for some modems.
                     // Below is for VODAFONE GREECE - be sure to set your own!
                     gateway.setSmscNumber(SMSNo);
                     Service.getInstance().setOutboundMessageNotification(outboundNotification);
                     Service.getInstance().addGateway(gateway);
                     Service.getInstance().startService();
            JOptionPane.showMessageDialog (null, "Modem Information:\n"
             +"Manufacturer: " + gateway.getManufacturer()+"\n"
             +"Model: " + gateway.getModel()+"\n"
             +"Serial No: " + gateway.getSerialNo()+"\n"
```

```
            //+"SIM IMSI: " + gateway.getImsi()+"\n"
            +"Signal Level: " + gateway.getSignalLevel() + " dBm\n"
            + "\n        PLEASE WAIT UNTIL A MESSAGE STATUS DIALOG BOX APPEAR.        "
            );
                    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() + " dBm");
                    System.out.println("  Battery Level: " + gateway.getBatteryLevel() + "%");
                    System.out.println();
                    // Send a message synchronously.
                    OutboundMessage msg = new OutboundMessage(Phone, SMSArea.getText());
                    Service.getInstance().sendMessage(msg);
            JOptionPane.showMessageDialog (null,"SMS to :" + Phone
                  +"\nStatus Report : " + msg.getMessageStatus()
                  +"\nError Message : " + msg.getErrorMessage()
                  + "\nRetry :" + msg.getRetryCount()
                  );
            if (msg.getMessageStatus().toString().equalsIgnoreCase("SENT")){
               SMS = "Sent";
               Service.getInstance().stopService();
            }else if (msg.getMessageStatus().toString().equalsIgnoreCase("FAILED")){
               SMS = "Draft";
               Service.getInstance().stopService();
            }
        try{
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
          Statement stmt = connection.createStatement();
          stmt.executeUpdate("UPDATE appointment SET SMS = '" + SMS + "'  WHERE PID = '" +
PIDField.getText() + "'");
        }
          catch(Exception ex)
         {
          JOptionPane.showMessageDialog(null,ex.getMessage());
        }
throw new RuntimeException("Compiled Code");
```

```
}
public class OutboundNotification implements IOutboundMessageNotification
        {                       public void process(AGateway gateway, OutboundMessage msg)
                        {
                                JOptionPane.showMessageDialog (null,"Outbound handler called from
Gateway: " + gateway.getGatewayId());
                        //System.out.println("Outbound handler called from Gateway: " +
gateway.getGatewayId());
                                //System.out.println(msg);
                        }       }
```

4.2.11 Admin Menu

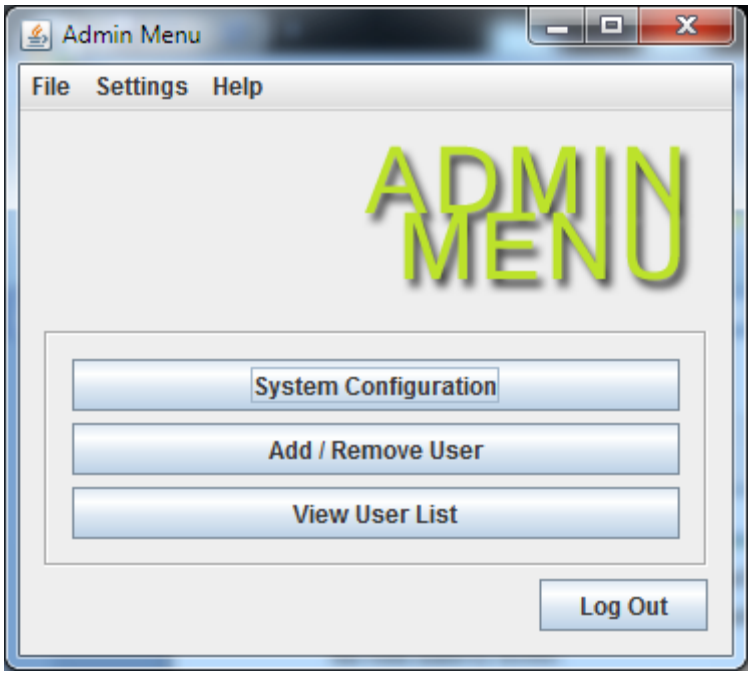


Figure 4.13 Admin Menu

In Figure 4.13 shows the Admin Menu which appear when administrator successfully log in to the system. Administrator can have three options in this menu.

```
public class AdminMenu extends javax.swing.JFrame {

    /** Creates new form AdminMenu */
    public AdminMenu() {
        initComponents();
    } private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
        new MainMenu().setVisible(true);
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        new AddUser().setVisible(true);
        this.setVisible(false);
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
      new UsList().setVisible(true);

    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        new SystemConf().setVisible(true);
        this.setVisible(false);
    }

    private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
}

    private void jMenuItem7ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}

    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new AdminMenu().setVisible(true);
            }
        });    }
```

4.2.12 System Configuration

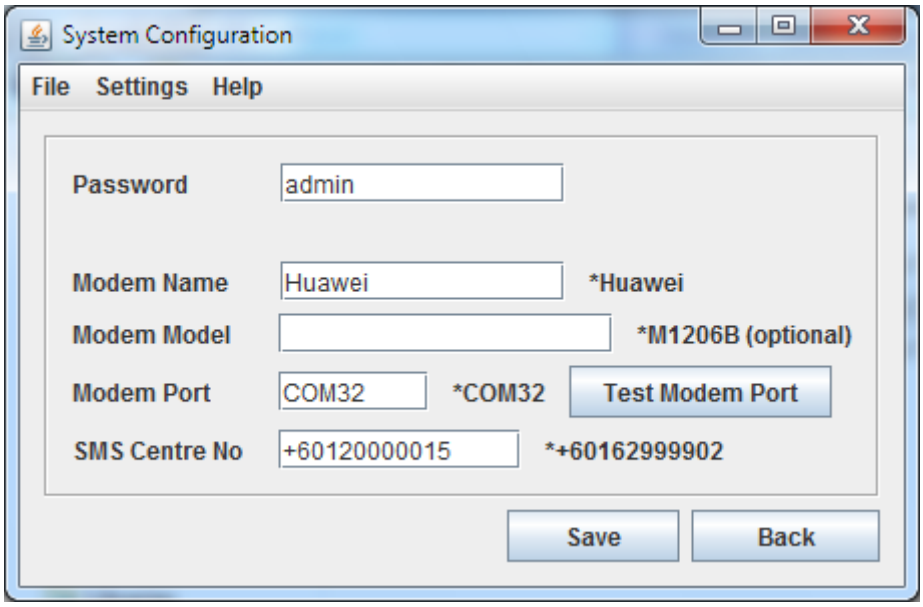


Figure 4.14 System Configuration Form

In Figure 4.14 shows the System Configuration form. Administrator can edit the configuration setting in this form.

```
public class SystemConf extends javax.swing.JFrame {
    /** Creates new form SystemConf */
    public SystemConf() {
        initComponents();
    } private void formWindowOpened(java.awt.event.WindowEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt2 = connection.createStatement();
            String queryString2 = "SELECT * FROM admin" ;
            ResultSet rset2 = stmt2.executeQuery(queryString2);
            if(rset2.next()){
            AdminPassField.setText(rset2.getString("PASSWORD"));
            MNameField.setText(rset2.getString("NAME"));
            MModelField.setText(rset2.getString("MODEL"));
            MPortField.setText(rset2.getString("PORT"));
            SMSNoField.setText(rset2.getString("SMSCENTER"));
             }}
```

```
          catch(Exception ex)
          {
          JOptionPane.showMessageDialog(null,ex.getMessage());
        }
    }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      int option = JOptionPane.showConfirmDialog((Component)
       null, "Are you sure want to change the settings", "Alert", JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION ) {
          try{
             Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
             Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
             Statement stmt = connection.createStatement();
             String Pass = AdminPassField.getText();
             String MName = MNameField.getText();
             String MModel = MModelField.getText();
             String MPort = MPortField.getText();
             String SMSNo = SMSNoField.getText();
             if (MPort.equals("")){
             JOptionPane.showMessageDialog(null,"Please fill all the Modem Port
field","Error",JOptionPane.WARNING_MESSAGE);
             }
             else{
             stmt.executeUpdate("UPDATE  admin SET Password = '" + Pass + "' , Name = '" + MName +
"' ,Model = '" + MModel + "' ,Port = '" + MPort + "' ,SMSCENTER = '" + SMSNo + "' ");
             }
              JOptionPane.showMessageDialog(null,"System Setting have been successfully
saved","Success",JOptionPane.INFORMATION_MESSAGE);
              new AdminMenu().setVisible(true);
              dispose();
          }
          catch(Exception ex){
          JOptionPane.showMessageDialog(null,ex.getMessage());
        }
    }else if (option == JOptionPane.CANCEL_OPTION) {
       }
    }
    private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
       dispose();
```

```
        new AdminMenu().setVisible(true);
    }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
}
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}



    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new SystemConf().setVisible(true);
            }
        });
    }
```

4.2.13 Add Remove User

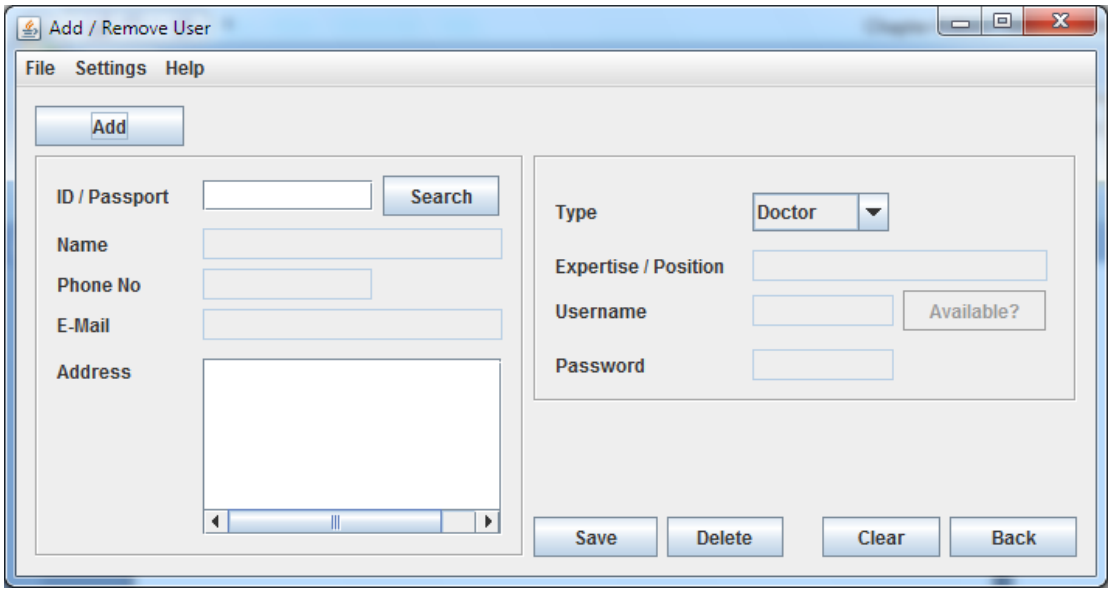


Figure 4.15 Add / Remove User Form

In Figure 4.15 shows the Add/Remove User form which only can be access by Administrator.

```
public class AddUser extends javax.swing.JFrame {


   private int AddUser;
   private int Add = 0;
   /** Creates new form AddUser */
   public AddUser() {
      initComponents();
   } private void SearchButtonActionPerformed(java.awt.event.ActionEvent evt) {
      try{
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
         Statement stmt = connection.createStatement();
         String ID = IDField.getText();
         String queryString = "SELECT * FROM user WHERE UID = '" + ID + "'" ;
         ResultSet rset = stmt.executeQuery(queryString);
         if(rset.next()){
            NameField.setText(rset.getString("NAME"));
            PhoneField.setText(rset.getString("PHONE"));
            EmailField.setText(rset.getString("EMAIL"));
            AddressArea.setText(rset.getString("ADDRESS"));
```

```
            PositionField.setText(rset.getString("POSITION"));
            UsernameField.setText(rset.getString("USERNAME"));
            TComboBox.setSelectedItem(rset.getString("TYPE"));
            JOptionPane.showMessageDialog(null,"User
Found","Done",JOptionPane.INFORMATION_MESSAGE);
            DeleteButton.setEnabled(true);
            SaveButton.setEnabled(false);
         }
      } catch(Exception ex)
      {
         JOptionPane.showMessageDialog(null,ex.getMessage());
      }
}
   private void UsernameFieldActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
}
   private void UsernameButtonActionPerformed(java.awt.event.ActionEvent evt) {
      if( AddUser == 0) {
         NameField.setEditable(true);
         PhoneField.setEditable(true);
         EmailField.setEditable(true);
         PositionField.setEditable(true);
         AddressArea.setEditable(true);
         PasswordField.setEditable(true);
         AddUser = 1;
      }
      try{
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
         Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
         Statement stmt = connection.createStatement();
         String Username = UsernameField.getText();
         ResultSet rset = stmt.executeQuery("SELECT USERNAME FROM user WHERE USERNAME
= '" + Username + "'");
         if(rset.next()){
            String name = (rset.getString("USERNAME"));
            if(Username.equalsIgnoreCase(name)){
               JOptionPane.showMessageDialog(null,"Username already
used","Done",JOptionPane.INFORMATION_MESSAGE);
               NameField.setEditable(false);
```

```
                PhoneField.setEditable(false);
                EmailField.setEditable(false);
                PositionField.setEditable(false);
                AddressArea.setEditable(false);
                PasswordField.setEditable(false);
                AddUser = 0;
              }
            }
        } catch(Exception ex) {
            JOptionPane.showMessageDialog(null,ex.getMessage());
        }
    }
    private void AddButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if (Add == 0) {
            UsernameField.setEditable(true);
            UsernameButton.setEnabled(true);
            Add = 1;
        }
        Add = 0;
        AddUser = 0;
}
    private void SaveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
            Statement stmt = connection.createStatement();
            Statement stmt1 = connection.createStatement();
            String ID = IDField.getText();
            String Name = NameField.getText();
            String Phone = PhoneField.getText();
            String Address = AddressArea.getText();
            String Email = EmailField.getText();
            String Position = PositionField.getText();
            String Username = UsernameField.getText();
            String Password = PasswordField.getText();
            String passwordEnc = Crypt.encrypt(Password);
            String Type = TComboBox.getSelectedItem().toString();
            if (ID.equals("") | Name.equals("")| Phone.equals("")| Address.equals("")| Email.equals("")|
Position.equals("")|Password.equals("")){
```

```
            JOptionPane.showMessageDialog(null,"Please fill the data
corectly","Error",JOptionPane.WARNING_MESSAGE);
        } else{
            if (Type.equalsIgnoreCase("Doctor")){
                stmt1.executeUpdate("INSERT INTO doctor (USERNAME,PASSWORD) VALUES ('" +
Username + "' , '" + passwordEnc + "')");
                stmt.executeUpdate("INSERT INTO user
(UID,NAME,PHONE,ADDRESS,EMAIL,POSITION,USERNAME,TYPE) VALUES ('" + ID + "' , '" +
Name + "', '" + Phone + "', '" + Address + "', '" + Email + "', '" + Position + "', '" + Username + "' , '" +
Type + "')");
            } else if (Type.equalsIgnoreCase("Staff")){
                stmt1.executeUpdate("INSERT INTO staff VALUES ('" + Username + "' , '" + passwordEnc
+ "')");
                stmt.executeUpdate("INSERT INTO user
(UID,NAME,PHONE,ADDRESS,EMAIL,POSITION,USERNAME,TYPE) VALUES ('" + ID + "' , '" +
Name + "', '" + Phone + "', '" + Address + "', '" + Email + "', '" + Position + "', '" + Username + "', '" +
Type + "')");
            }
            NameField.setEditable(false);
            PhoneField.setEditable(false);
            EmailField.setEditable(false);
            PositionField.setEditable(false);
            AddressArea.setEditable(false);
            PasswordField.setEditable(false);
            UsernameButton.setEnabled(false);
            SaveButton.setEnabled(false);
            AddUser = 0;
            JOptionPane.showMessageDialog(null,"User "+Username+" have been added
successfully","Success",JOptionPane.INFORMATION_MESSAGE);
        }
      } catch(Exception ex)
      {
        JOptionPane.showMessageDialog(null,ex.getMessage());
        NameField.setEditable(false);
        PhoneField.setEditable(false);
        EmailField.setEditable(false);
        AddressArea.setEditable(false);
        PositionField.setEditable(false);
        UsernameField.setEditable(false);
        PasswordField.setEditable(false);
```

```
        Add = 0;
        AddUser = 0;
    }}
  private void DeleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    int option = JOptionPane.showConfirmDialog((Component)
        null, "Are you sure want to delete the user", "Alert", JOptionPane.OK_CANCEL_OPTION);
    if (option == JOptionPane.OK_OPTION ) {
      try{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
        Statement stmt = connection.createStatement();
        String ID = IDField.getText();
        String Position = TComboBox.getSelectedItem().toString();
        String Username = UsernameField.getText();
        if (Position.equalsIgnoreCase("doctor")){
          String queryString = "DELETE FROM user WHERE UID = '" + ID + "'" ;
          String queryString2 = "DELETE FROM doctor WHERE USERNAME = '" + Username + "'"
;          stmt.executeUpdate(queryString);
          stmt.executeUpdate(queryString2);
        }else if (Position.equalsIgnoreCase("staff")){
          String queryString = "DELETE FROM user WHERE UID = '" + ID + "'" ;
          String queryString2 = "DELETE FROM doctor WHERE USERNAME = '" + Username + "'"
;          stmt.executeUpdate(queryString);
          stmt.executeUpdate(queryString2);
        }
        IDField.setText("");
        NameField.setText("");
        PhoneField.setText("");
        EmailField.setText("");
        AddressArea.setText("");
        PositionField.setText("");
        UsernameField.setText("");
        PasswordField.setText("");
        JOptionPane.showMessageDialog(null,"User data have been
deleted.","Success",JOptionPane.INFORMATION_MESSAGE);
      } catch(Exception exception){
      }
    } else if (option == JOptionPane.CANCEL_OPTION) {
    }}
```

```
    private void ClearButtonActionPerformed(java.awt.event.ActionEvent evt) {
        IDField.setText("");
        NameField.setText("");
        PhoneField.setText("");
        EmailField.setText("");
        PositionField.setText("");
        AddressArea.setText("");
        UsernameField.setText("");
        PasswordField.setText("");
        NameField.setEditable(false);
        PhoneField.setEditable(false);
        EmailField.setEditable(false);
        PositionField.setEditable(false);
        AddressArea.setEditable(false);
        PasswordField.setEditable(false);
        UsernameButton.setEnabled(false);
        SaveButton.setEnabled(true);
        AddUser = 0;
}
    private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
         dispose();
        new AdminMenu().setVisible(true);
}
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
}
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}




    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new AddUser().setVisible(true);
    }
```

4.2.14 User List

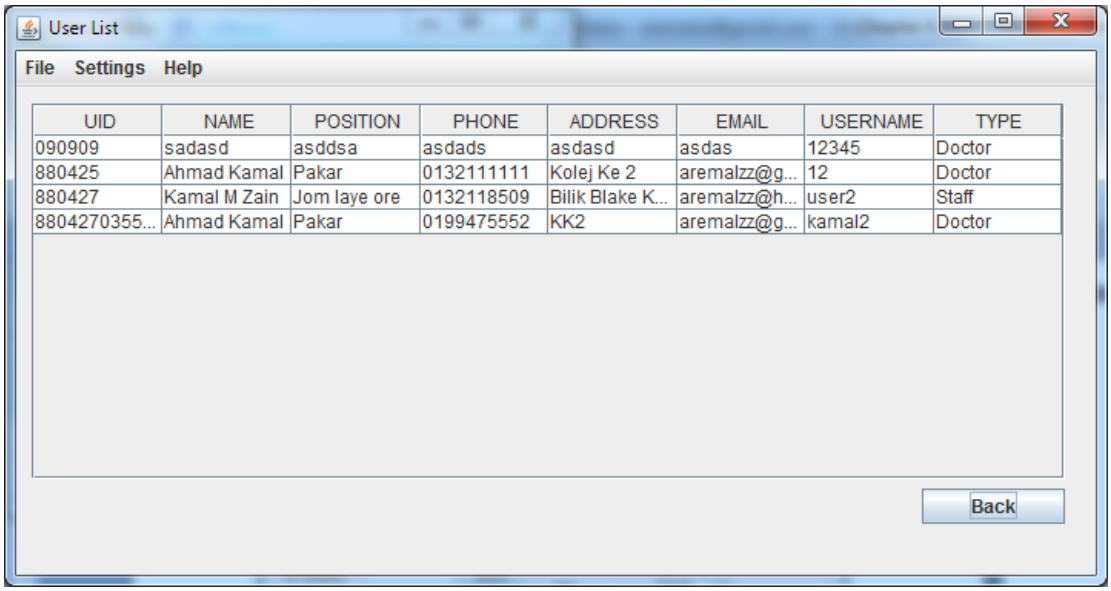


Figure 4.16 User List View for Admin

In Figure 4.16 shows the User List form which only can be access by Administrator.

```
public class UsList extends javax.swing.JFrame {
    /** Creates new form UsList */
    public UsList() {
        initComponents();
    } private void BackButtonActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
}
    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        UpdateTable();
    }
    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
}
    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        new Console().setVisible(true);
}
    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
        new About().setVisible(true);
}


    /**
```

```
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new UsList().setVisible(true);
            }
        });
    }

    private void UpdateTable(){

                try{

                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/clinic","kamal","kamal123");
                ResultSet rset = null;
                PreparedStatement pst = null;
                String sql = "SELECT * FROM user";
                pst = connection.prepareStatement(sql);
                rset = pst.executeQuery();
                jTable1.setModel(DbUtils.resultSetToTableModel(rset));

                }
                 catch(Exception ex)

            {{
             JOptionPane.showMessageDialog(null,ex.getMessage());

           }

    }
    }
```

CHAPTER 5

RESULT AND DISCUSSION

5.1 Introduction

This chapter discuss about the result after the testing, the development of Clinic Management System with Notification using GSM Modem, advantages and disadvantages and the assumption for future development of the system. Hopefully, the discussion would bring out some benefits and ideas for the future developer to upgrade and enhance the performance, functionality of this system and the user interface (UI) design so that it will be more interactive and user friendly.

5.2 Result and Discussion

With the successful development of this system prototype, it has met its objectives stated in Chapter 1 which are:

i. To computerized and centralized all the information in order to reduce the time in retrieving all the data and information.

ii. To promote Eco-friendly software that will reduce the usage of paper.

iii. To facilitate patients, so that we can notify the patients using SMS notification, so they can be on time for their appointment.

5.3 Lesson Learnt

i. Skill

All related skills are needed in order to success in developing Clinic Management System with Notification using GSM Modem. This is proven when doing research that requires interaction with other people.

ii. Project Planning

Planning is a critical part, so this part must give more attention in the beginning of the project. So this project takes much time to develop and not a perfect one. Therefore, debugging is required while running. In the future, this project should have more details in a proper project planning before started any project.

5.4 Advantages and Disadvantages

In any system developed, there are some advantages and disadvantages in them. So, in this section is a brief explanation to the advantages and disadvantages of Clinic Management System with Notification using GSM Modem.

5.4.1 Advantages

Clinic Management System with Notification using GSM Modem has many advantages than the disadvantages. With this application, business activities would run smoothly and efficiently. Plus the history of patient can be trigger easily. To enable this function is not as simplistic as thought because the data has to be manipulated properly and tested gradually so that the output is satisfy. The advantages of PRMS are as below:

i. The patient record can be updated in real-time uses. The patient has its own unique identical ID so that there would not have redundancies. By this way a search to the menu could be attempt smoothly and precisely.

ii. Patient Record Management System are managed safely and interactively which the flow of the system navigates the user properly.

iii. Clinic Management System with Notification using GSM Modem is a secure enough from any unauthorized access. The entire doctors and staffs password is encrypted using Base64 algorithm.

iv. Doctor can make their appointment with patient base on their free time.

v. Staff can send a SMS to patients to notify their appointment.

vi. The error handling has been included to minimize user mistake while key in the data information so this will make the data information become more trustworthy.

5.4.2 Disadvantages

Although this system has fulfilled the user requirement but it also has some disadvantages. The disadvantages of this system are:

i. Patients cannot make their appointment without going to the clinic. Patients must have seen the doctor first before make any appointment.

5.5 Conclusion

The main purpose of developing Clinic Management System with Notification using GSM Modem is to help any clinic or hospital to manage patient record in the premise. This system is able to manage patients' data effectively and easily. In the other words, this system helps the staff to work easily when they want to record about patient information and details.

This system is developed based in the problems and situations (scenarios) that occur in that clinic premise. Clinic Management System with Notification using GSM Modem is developed based on interactive methodology. The system development are Netbeans and PhpmyAdmin. It used the Java as the programming language and MySQL as the database platform and data connection between the system and the database.

There are three distinct types of users. The three different types of user that are admin, staffs and doctors have their own functionalities and each of them have also have own specialties in this Clinic Management System with Notification using GSM Modem.

6.0 Reference

Ritchie, S., (1997). *Systems programming in java*. *17*(3), 30-32.

Dejan, J. (n.d.). *Why java will always be slower than c* . Retrieved November 22, 2011, from http://www.jelovic.com/articles/why_java_is_slow.htm

Wikipedia. (n.d.). Retrieved October 5, 2011, from Visual Basic: http://en.wikipedia.org/wiki/Visual_Basic

Wikipedia. (n.d.). Retrieved October 5, 2011, from Java (programming language): http://en.wikipedia.org/wiki/Java_(programming_language)

Dukovic, J. M., & Joyce, D. T., (1995). An evaluation of object-based programming with visual basic.

Businessdictionary.com - online business dictionary. (n.d.). Retrieved from http://www.businessdictionary.com/definition/management-system.html

Egwunyenga, E. J., (2009). A Record Keeping in Universities: Associated Problems and Management Options in South West Geo-Political Zone of Nigeria. *Int J Edu Sci,* 1(2), 109-113.

Dalcı, I. & Tanış, V. N., (n.d.). Benefits of Computerized AccountingInformation Systems on the JIT Production Systems. *Review of Social, Economic & Business Studies,* 2, 45-64

Fredrick, K., (2009). A Web-Based blood donor management information system for the Red Cross Society, Uganda (WBBDMI).

Shigeta. A, Suto. & Nosu. K., (2008). Development of Management System for Electric Referral Documents and Healthcare Information Exchanging Based on Standardization Protocols. 227.

Constantin, D. O., (2011). GSM infrastructure used for data transimission.IEEE publishing.

Wikipedia. (n.d.). Retrieved December 4, 2011, from Iterative and incremental development: http://en.wikipedia.org/wiki/Iterative_and_incremental_development

NowSMS | SMS Gateway, SMS Server Software, MMS Gateway & MMSC (n.d.). Retrieved December 10, 2011, from What is a GSM Modem? | NowSMS: http://www.nowsms.com/faq/what-is-a-gsm-mode

Steigjer, M. (2008). *Traditional versus iterative development method, and when to use which*. Retrieved from http://www.silvercrestconsulting.com/gui/pdf/1237375021.pdf

Pressman, R. S. (2010). *Software engineering, a practitioner's approach*. (7th ed. ed.). McGraw-Hill Science/Engineering/Math.

Information on mobile industry!. (n.d.). Retrieved from http://www.mobileisgood.com/ArchitectureOfTheGSMNetwork.php

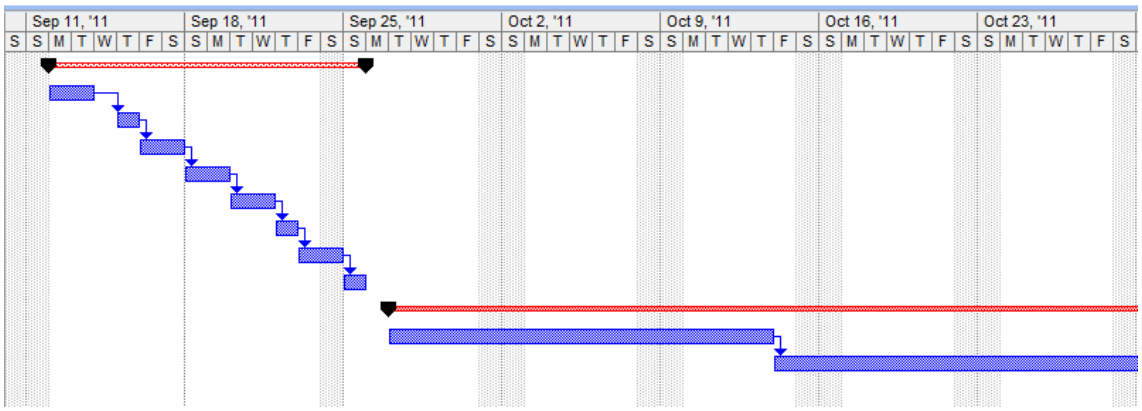
Postel, J. (1982). Simple mail transfer protocol. Retrieved from http://www.faqs.org/rfcs/rfc821.html

APPENDICES

APPENDIX A

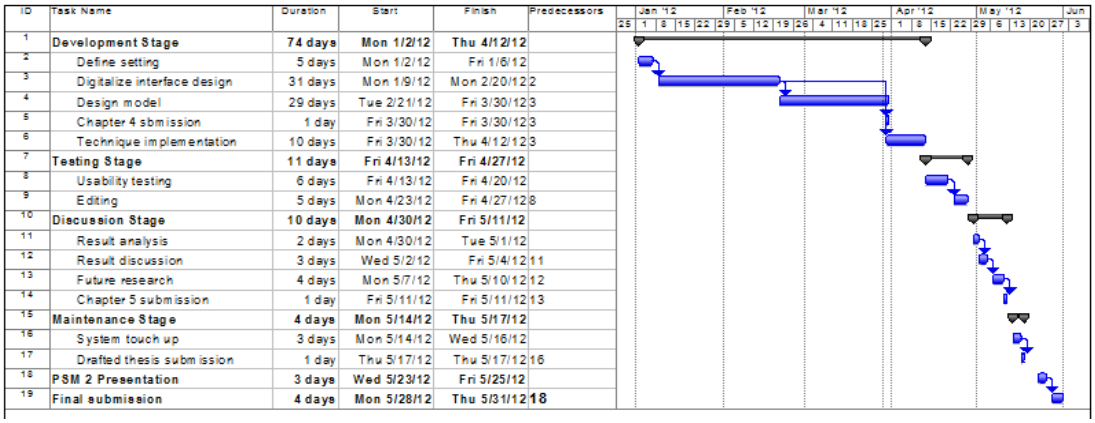
Gant chart PSM I

1		**Project Requirement Planning**	13 days	Mon 9/12/11	Sun 9/25/11	
2		Identify the title of project	2 days	Mon 9/12/11	Tue 9/13/11	
3		Identfy the problem statement of project	1 day	Thu 9/15/11	Thu 9/15/11	2
4		Identify the objective of project	2 days	Fri 9/16/11	Sat 9/17/11	3
5		Identify the scope of project	2 days	Sun 9/18/11	Mon 9/19/11	4
6		Identify task	2 days	Tue 9/20/11	Wed 9/21/11	5
7		Estimate task duration	1 day	Thu 9/22/11	Thu 9/22/11	6
8		Develop analysis flow	1 day	Fri 9/23/11	Sat 9/24/11	7
9		Analysis software and hardware tools	1 day	Sun 9/25/11	Sun 9/25/11	8
10		**User design system**	**60 days**	**Tue 9/27/11**	**Fri 12/16/11**	
11		Chapter 1 : Introduction	13 days	Tue 9/27/11	Thu 10/13/11	
12		Chapter 2: Literature review	26 days	Fri 10/14/11	Fri 11/18/11	11
13		Chapter 3: Methodology	15 days	Sat 11/19/11	Thu 12/8/11	12
14		Chapter 4: Expected Result and Discussion	3 days	Fri 12/9/11	Tue 12/13/11	13
15		Chapter 5: Conclusion	3 days	Wed 12/14/11	Fri 12/16/11	14
16		Submission of draft thesis and PSM report to SV	5 days	Mon 12/26/11	Fri 12/30/11	
17		Submission of overview and log book to coordinat	3 days	Wed 12/28/11	Fri 12/30/11	
18		PSM presentation	5 days	Mon 1/16/12	Fri 1/20/12	
19		Submission of PSM1 report	2 days	Fri 1/20/12	Sun 1/22/12	



APPENDIX B

Gant chart PSM II



ID	Task Name	Duration	Start	Finish	Predecessors
1	**Development Stage**	**74 days**	**Mon 1/2/12**	**Thu 4/12/12**	
2	Define setting	5 days	Mon 1/2/12	Fri 1/6/12	
3	Digitalize interface design	31 days	Mon 1/9/12	Mon 2/20/12	2
4	Design model	29 days	Tue 2/21/12	Fri 3/30/12	3
5	Chapter 4 sbmission	1 day	Fri 3/30/12	Fri 3/30/12	3
6	Technique implementation	10 days	Fri 3/30/12	Thu 4/12/12	3
7	**Testing Stage**	**11 days**	**Fri 4/13/12**	**Fri 4/27/12**	
8	Usability testing	6 days	Fri 4/13/12	Fri 4/20/12	
9	Editing	5 days	Mon 4/23/12	Fri 4/27/12	8
10	**Discussion Stage**	**10 days**	**Mon 4/30/12**	**Fri 5/11/12**	
11	Result analysis	2 days	Mon 4/30/12	Tue 5/1/12	
12	Result discussion	3 days	Wed 5/2/12	Fri 5/4/12	11
13	Future research	4 days	Mon 5/7/12	Thu 5/10/12	12
14	Chapter 5 submission	1 day	Fri 5/11/12	Fri 5/11/12	13
15	**Maintenance Stage**	**4 days**	**Mon 5/14/12**	**Thu 5/17/12**	
16	System touch up	3 days	Mon 5/14/12	Wed 5/16/12	
17	Drafted thesis submission	1 day	Thu 5/17/12	Thu 5/17/12	16
18	**PSM 2 Presentation**	**3 days**	**Wed 5/23/12**	**Fri 5/25/12**	
19	**Final submission**	**4 days**	**Mon 5/28/12**	**Thu 5/31/12**	**18**