Virtual Clinic

Developed By **Major Haroon Ali Baig**

Captain Murtaza Ali Zaidi

Captain Khalid Masood

Captain Khalid Parwaz

Supervised By **Miss Hajra Batool**

Brigadier Dr Riaz A Bashir

Department of Computer Science

Military College of Signals,

Rawalpindi

2007

Abstract

The advent of the Internet has sparked a rapid increase in the *electronification* of all routine activities including life critical services such as diagnosis of diseases. The term "**virtual clinic",** which means a clinic having no physical existence yet providing all the facilities which are available in any real clinic, has captured the imagination of medical entrepreneurs and investors. The idea is to provide medical treatment to people without having them ever set foot in a hospital. Due to its great utility the notion of a virtual clinic has gained immense popularity amongst doctors and patients worldwide. The true value of such systems is evident in remote areas where physical medical services are scarcely available if at all. People in those areas have to travel long distances in order to get adequate treatment and a delay in the provision of the same may prove fatal. A virtual clinic will enable access to instant expert medical advice in any part of the world from any other part.

Our objective is to realize this idea in the form of a generic application which can be deployed in any hospital. The features provided by our **virtual clinic** include online registration of patients and doctors and online diagnosis and prescriptions. The system will also maintain the medical history of every patient ever entering the virtual clinic in order to enable more accurate and detailed analysis by the physicians. A front desk doctor can record basic symptoms such as temperature, blood pressure etc., and refer the case to a specialist, if required, for instant advice.

Declaration

It is declared that we developed this software and this report entirely on the basis of our personal efforts made under the sincere guidance of our project supervisors.

Dedication

The project is dedicated to **Pakistan Army** who offered us the opportunity to become **Software Engineers** and thus enabling us to undertake such an innovative work.

Special thanks to **Military College of Signals** particularly **Computer Science Department** who extended their full support for the accomplishment of the work.

Acknowledgement

First and foremost, we are grateful to Almighty **Allah**, Who gave us the ability to undertake and complete the project successfully.

Very special thanks to our parents whose heart-felt prayers, appreciation, and support has always been a valuable asset and a great source of inspiration for us. They always encouraged us during our academic career.

Special thanks to our supervisors **Miss Hajra Batool Asghar and Brigadier Doctor Riaz A Bashir**, whose valuable suggestions helped us work on this project.

We are indebted to our teachers for their co-operation and encouragement to attain this goal.

We extend our thanks to our course advisor **Major Athar Mohsin Zaidi** for keeping the atmosphere conducive and favourable for studies.

May Almighty **ALLAH** shower his blessings and bounties on all of us.

Project in Brief

Project Title	Virtual Clinic
	MSN Messenger Type
	Client Server Application
Objectives	Patient Diagnoses
	Centralized Database
	Authenticated System
	User Friendly System
	Time Saving
Developed By	Major Haroon Ali Baig
	Captain Murtaza Ali Zaidi
	Captain Khalid Masood
	Captain Khalid Parwaz
Supervised By	Miss Hajra Batool Asghar
	Brigadier Dr Riaz A Bashir
Tools Used	JDK 1.5.0
	Eclipse 3.1.0
	J Creator Pro
	Java Visual Editor
	MySql Database
	Navicat.MySQL.v7.2.2 (GUI)

	MS Project 2003
Operating System	Windows XP Professional
System Used	Pentium IV

Table of Contents

List of Tables

List of Figures / Diagrams

Chapter 1

Introduction

This chapter gives a brief introduction about Virtual Clinic.

1.1 Outline

In this chapter a brief introduction of the project is described along with the purpose of the project.

1.2 Introduction to Virtual Clinic

. Users of the internet are multiplying every day. People from all backgrounds, doing all kinds of jobs had been using internet for information and entertainment ever since its advent. Now with the notion of ubiquitous computing, a range of all other kinds of services are also been put onto the web from where they can be accessed without spatial and temporal restriction. Along with the others, making health business online can drastically improve the acquisition of such services thus can save a lot of lives otherwise lost because of all kinds of difficulties in accessing hospitals. The term virtual clinic is used to describe ventures ranging from the development of web-based courses at bricks and mortar clinics to the creation of entirely new enterprises dedicated solely to the delivery of online distance medical facilities.

Deaths due to curable diseases are very common in remote areas where medical facilities are rare because of inappropriate infrastructure and lack of doctors willing to go there. Reaching to a hospital costs in money and time. Besides there can be many situations where a patient cannot travel without aggravating the illness. All this calls for an efficient and cost-effective mechanism for the provision of medical assistance with minimal worries to the patients. Virtual clinic is an answer to this call.

Hospitals are now trying to provide almost all the services ranging from diagnosis of diseases to analysis and prescriptions to the patients at their desktops. Moreover, research is underway with some success to develop ways and infrastructural support to conduct sophisticated examinations and even surgeries from distance. From looking up hospitals to acquiring services, everything can be done with few mouse clicks and key hits. However no such system existed in Pakistan before we proposed the idea. Although realizing its importance and usefulness some other groups have also now also announced the development of similar systems for their hospitals but none exists in working form yet.

1.3 Problem Definition

To develop indigenous network based application for provision of online medical facilities to the patients particularly in remote areas.

1.4 Project Scope

- The application is network based following the client-server model with a central database. This will enable people on different locations to communicate with each other without any spatial restrictions. Users will assume different roles based on their jobs and perform their duties collaboratively in order to materialize the provision of medical treatment to patients.
- Only authorized access is provided and other security measures such as authentication and validation have been implemented
- The product is generalized which can be implemented in any of the hospitals.

- General symptoms are recorded on every visit and the case is referred to a specialist if required.
- A specialist may prescribe conduct of various laboratory examinations reports of which will be digitized and stored for current and future use
- A central database records all essential information about patients and their visits to virtual clinic. A physician can inquire a person’s medical history for better analysis in addition to the current symptoms

1.5 Aim

To develop a client server network based application with centralised database to help patients in remote areas to get instant online medical advice from specialists with economy and minimal trouble on the patients’ part.

1.6 Objectives

The core objectives of the system “**Virtual Clinic”** is to achieve following:-

- Indigenous development of an application that provide expert medical advise to patients from distance.
- Instant communication between different roles for collaborative completion of tasks
- Complete record of essential information of patients
- Reliable, fast and efficient data transfer and storage.
- Provision of user friendly interface.
- Reliable diagnosis.
- Cost effective

Chapter 2

System Analysis

This chapter describes the detailed system analysis and system's requirements.

2.1 Outline

This chapter reviews the analysis of this project. An extensive analysis of the domain is required before designing phase in order to produce an efficient and accurate design. Study of existing similar systems always gives a clear direction for the establishment of requirements and helps in identifying deficiencies and possible improvements. We studied existing patient management systems to form basic requirements and conducted interviews with domains experts (doctors) to refine and finalize them. Since social and cultural values have a significant impact on the realization of an application, such factors were also considered and thus similar applications built in other regions were analysed carefully before picking up requirements from them. A detailed analysis of the gathered information helped in building design of the application and the database.

2.2 System Analysis

System analysis involved a study and understanding of all the aspects and procedures of the real clinics setup, and a close observation of how things worked in a physical hospital environment. Visits were made to different hospitals especially Combined Military Hospital (CMH) Rawalpindi for the purpose. Interviews with the medical practitioners were also held to get first hand information and feedback about the proposal. Our analysis activities included following.

2.2.1 Data gathering

In order to clearly understand the system we gathered data through different sources which are:-

2.2.1.1 Existing hardcopy documents

Various hardcopy documents and forms exist in hospitals for different purposes such as recording a patient's symptoms, laboratory test prescription and reports, disease diagnosis and prescriptions, etc. These documents give a clear idea of the procedures and what essential information is required for them. The documents were analysed for digitization. During the study it found that information was distributed redundantly in various documents and ill structured. Moreover, some information seemed missing from the documents if they were to be computerized. The discrepancies and deficiencies were noted and catered for during the database design.

2.2.1.2 Interviews

Interviews with domain experts are most useful source of first hand information. Since the documents didn't provide a comprehensive view of the procedures, interaction with the domain experts, medical practitioners in this case, was necessary to resolve missing information and ambiguity issues. Moreover, computerization of such a system which involves deep human interaction is a difficult task , it needs extra care in order to make it friendly and acceptable to the users. Also, since health and treatment are sensitive and critical issues, most accurate information is required. Therefore a number of interviews were made and different people playing different roles in a hospital were contacted and discussions held with them to establish clear and precise requirements. Information given by each group of expected users of the system was recorded and cross-checked against each other to remove inconsistencies.

2.2.2 Data Analysis

Virtual clinic is essentially a data-centric application which involves production, storage and analysis of data to provide the intended services. Dataflow diagram (DFD) is the most suitable model to express processing of data. We have developed a DFD (shown in the design section) for our application.

2.2.3 Component Diagram

Figure 2.1 presents a component diagram for the basic implementation of virtual clinics. There can be multiple GDMO and Physician nodes as required in actual deployment.

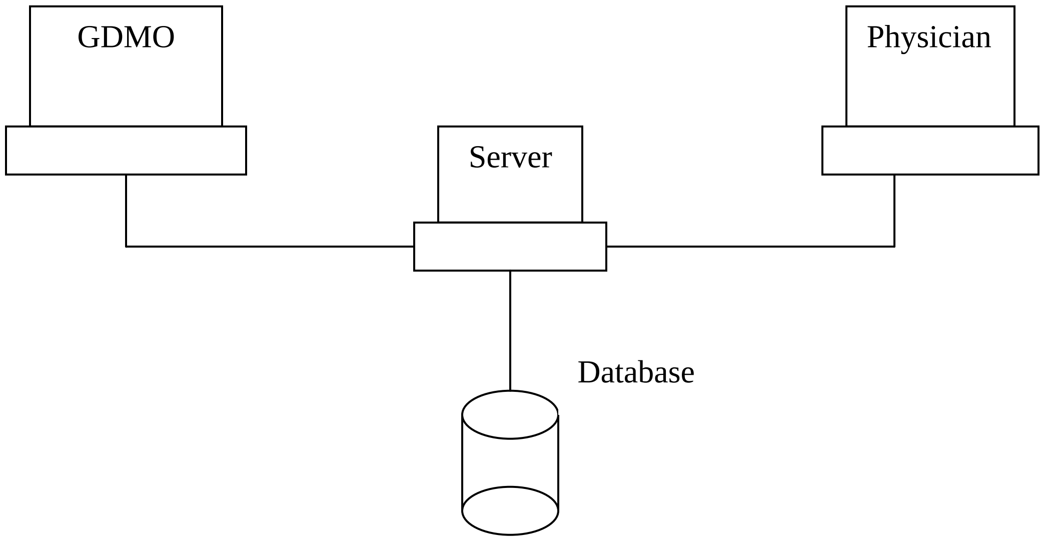


Figure 2.1 System Analysis Diagram

There will be a server running on the network and connected to the database. All the users of the system will be authenticated by this server. Only authorized users can access this system.

2.3 Domain Analysis

- **Virtual Clinic** is a **platform independent** application; therefore, Virtual Clinic can run on any operating system with Java Virtual Machine **(JVM)** environment.
- **Virtual Clinic** assumes that **imaging devices** such as **scanner** and **camera** are attached to the system and the derivers are installed.
- **Detecting imaging devices** whether they are attached and derivers are installed, is not the part of **Virtual Clinic**.
- **Printing device** is attached and configured.

2.3.1 Functional Requirements

The core functional requirement of the project is to develop a system, which provides a complete platform independent automated solution over the network, which can be used to access centralized database. The system will have following functional capabilities.

- User verification
- Prepaid card verification
- Patient’s record entry and retrieval
- Maintaining patient’s history
- Physician’s credit update
- Scanning reports/taking injured area picture
- Sending/receiving files (reports)
- Printing prescription

- Powerful GUI
- Platform Independence

2.3.2 Non-Functional Requirements

2.3.2.1 Efficiency

System should be robust enough and be able to make quick response to user requests.

2.3.2.2 Quality

System should be capable of handling the image in best quality.

2.3.2.3 Reliability

The system should be reliable for efficient functionality.

2.3.2.4 User-friendliness

System should be capable of presenting understandable error messages to user if he/she performed invalid operations, so that user can track the error.

2.3.2.5 Extensibility

Developed product should be extensible to support future enhancements.

2.3.2.6 Portability

The proposed system is easily portable from one environment to another without any major changes.

Chapter 3

Software Requirements Specification

This chapter specifies software requirements.

3.1 Outline

The purpose of this chapter is to define the requirements for the project “Virtual Clinic”. This chapter will provide a general description of our project including product perspective, overview of requirements and general constraints. In addition, it will also provide the specific requirements and functionality needed for the product such as interface, functional and performance requirements and quality attributes.

3.2 Overall Description

3.2.1 Product Perspective

Product i.e. Virtual Clinic will be a new, self-contained product. There will be a server running on the network connected to the database. The users of the system i.e. Administrator, Physician, and GDMO will be authenticated by the server. Only authorized users can access the system.

3.2.2 User Classes and Characteristics

User classes are differentiated based on product functions, technical expertise, security, and educational level. These are:

3.2.2.1 Administrator

- Generation of New Prepaid Card Numbers
- Creation of New Users
- Changing Own Password

3.2.2.2 GDMO

- Verification of prepaid card

- Patient’s record/symptoms
- Scanning Report/x-ray/take picture
- Sending patient report
- Physician credit update
- View physician comments
- Printing prescription
- Changing password
- Signing Out from application

3.2.2.3 Physician

- View patient record/symptoms
- Receiving patient reports
- View patient reports
- View examined patients
- Checking financial assistance
- Changing password
- Signing out from application

3.2.3 Operating Environment

- **Virtual Clinic** is a **platform independent** application; therefore it can run on any operating system with Java Virtual Machine **(JVM)** environment.

3.2.4 Design and Implementation Constraints

- The program is intended to run on a Microsoft operating system only (win 2000, XP).

- Java programming language is chosen for development of the application in order to provide platform independent functionality.
- Open source database i.e. MySql is used.
- The proper list of development tools used is as follows:
    - JDK 1.5.0
    - Eclipse 3.1.0
    - J Creator Pro
    - Java Visual Editor
    - MySql Database
    - Navicat.MySQL.v7.2.2 (GUI)
    - MS Project 2003

3.3 User Documentation

3.3.1 Physician's User Guide

Physician's user guide is described in the following.

3.3.1.1 View Patient Record / Symptoms

This button is used for viewing the referred patient record / symptoms and giving own comments by the specialist after examining all details like reports (if any). First enter the referred patient's ID and press **View** button. Then give own comments and press the **Save** button to insert / update record into the database.

3.3.1.2 Receive Patient Reports

This button is used for downloading the patient’s reports from the File Transfer Server. The downloaded reports will be saved in the Reports directory at user end.

3.3.1.3 View Patient Reports

The down loaded reports can be viewed by pressing this button. Browse the report that is to be viewed by press **Browse** button.

3.3.1.4 View Examined Patients

This button will help the physician to view the list of patients that are examined by him.

3.3.1.5 Check Credit Update

Physician can check his account by pressing this button. This will show the current balance in the account.

3.3.1.6 Change Password

This button will help in changing the password. First enter the old password then new password and confirm the new password and finally save it.

3.3.1.7 Sign Out From Application

To signed-out or exit from the application.

3.4 Assumptions and Dependencies

- Virtual Clinic assumes that imaging devices such as scanner and camera are attached to the system and the derivers are installed.
- It assumes that an electronic scanning machine is available to scan reports/images.

The software will run on all the operating systems which support JVM e.g. Windows 2000, XP etc.

Chapter 4

System Design

This chapter describes the design of Virtual Clinic.

4.1 Outline

We have defined the hardware and software architecture, components, modules, interfaces, and data for the system to satisfy specified requirements. To design the system we have chosen Dataflow diagram (DFD) which is the most suitable model to express processing of data.

4.2 System Design

This phase is divided into two sub-phases: **Preliminary Design**, in which system concepts are established, followed by a **Detailed Design**, in which the exact design specifications are determined.

4.2.1 Preliminary design

- The first task of preliminary design is to review the system's requirements and then consider some of the major aspects of a system
- In the preliminary design stage we have developed a general system design and identified hardware, software and management needs. While working under a preliminary design stage it is also considered that how the input data will be gathered.

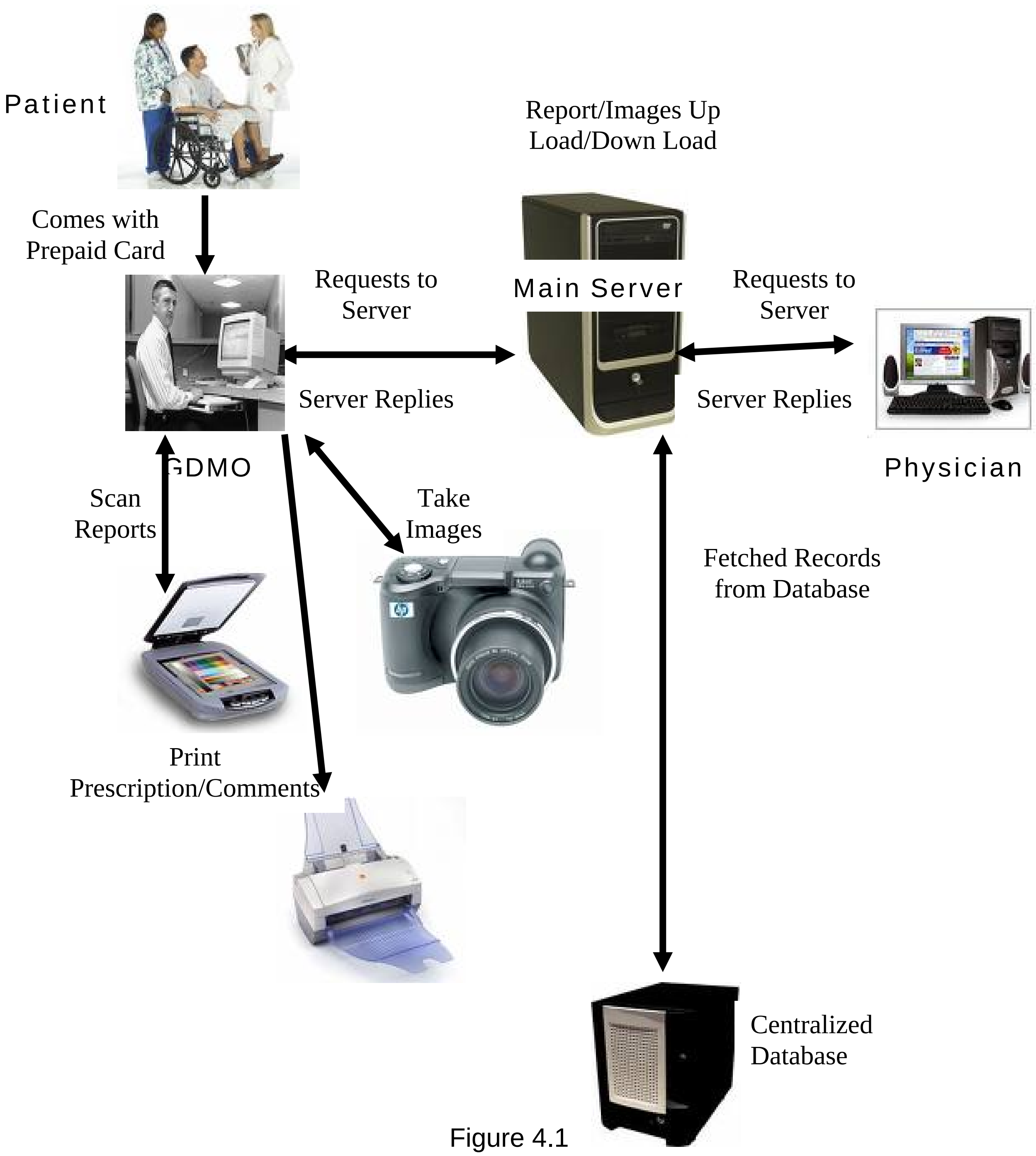

Patient
Report/Images Up Load/Down Load
Comes with Prepaid Card
Requests to Server
Main Server
Requests to Server
Server Replies
Server Replies
GDMO
Physician
Scan Reports
Take Images
Fetched Records from Database
Print Prescription/Comments
Centralized Database

Figure 4.1

4.2.2 Detailed design

The detail design activities are as follows:

- Designing output forms and screens.
- Planning input data forms and procedures.
- Drawing system DFD, planning file access methods and record's formats (Anx 'A').
- Planning database and data communication interfaces
- Designing system security control and considering human factors.

Keeping in mind both issues i.e. **preliminary** and **detail design**, designing process of entire system is started.

4.2.3 Input design

Input design is the main source of interaction between the users and developers. The outcomes of the system depend upon **input design**. More reliable the input design, more efficient and accurate would be the **output design**. To take input or information needed for the system, we have designed the forms. These input forms contain the control buttons and spaces for data entries for each field. Forms are simple and user-friendly. The input forms for the systems are:

- Physician's registration form
- GDMO's registration form
- Patient's general symptoms form
- Sign up form

4.2.4 Output design

The **output design** of the proposed system consists of the queries which are screen oriented. The output design of the system is user friendly and required results will be displayed in an organized and proper manner.

4.2.5 Code design

Object oriented technique is closer to natural thinking process and let analyze multiple individual and reusable components of the system. Therefore, making an extensible product following approaches are considered more suitable.

4.2.5.1 Initial design

For designing **Virtual Clinic**, we followed the strategy of **divide and conquer**. So we divided its processing in few independent processes and tried to create scenarios. For requirements analysis an **initial design** is made. **Virtual Clinic** receives the request from the user and according to that it responds.

4.2.5.2 Detailed design

In detailed design the various modules of the Virtual Clinic are taken into consideration.

The **Virtual Clinic** is divided into following separate independent modules.

4.2.5.2.1 User interface design

The design of **user interface** draws heavily on the experience of the designer. Three categories are suggested:

4.2.5.2.1.1 General interaction

In this system, for **general interaction** with the user, these steps have been designed.

- **Login:** It is the login screen for GDMO, *Physicians* as well as *Administrator*.
- **Main:** It has all the options for users to perform different tasks.
- **Help:** It provides information about system and short cuts help.

4.2.5.2.1.2 Information display

The following guidelines focus on **information display** of the system being designed**:**

- Display currently login user's name
- Display only that information that is relevant to the current context.

4.2.5.2.1.3 Data entry

For data entry these windows have been designed in Virtual Clinic:

- **Sign-In Window**: For user's sign-in to the application
- **User Creation Window**: For creating new users i.e. *GDMO* and *Physicians*.
- **Card Generation Window**: For generating and displaying new generated prepaid card numbers.
- **Main Window**: To display main interface of the application.
- **Display Window**: Various windows for entering and displaying records.
- **Help Window**: To display help about system.

4.3 Database design

A database is a shared collection of inter-related data designed to meet the varied information needs of an organization. Database must be shared which means that all qualified users in the organization have access to the same data for use in a variety of activities. Also there should not be the duplicate copies of the same data; instead all the data must be present at the same place so as to reduce redundancy.

4.3.1 Logical Database Design

To model the conceptual design, we reviewed the existing documents i.e. existing forms and reports, written guidelines, job descriptions, personal narratives, and memoranda of CMH Rawalpindi to determine the data requirements of the database. We carried out interviews of medical professionals and group meetings with doctors and recorded the information gathered. We carried out review of existing automated system and found that there is not any such system implemented.

The main objects/entities we conceived out of above information gathered are:

- Patients
- Doctors
- Administrator
- Prepaid card
- Prescription
- Medicine
- Dosage
- Category

The data of each object will be maintained and there is need to ensure the data integrity of each object. The attributes of each entity have been identified and are given in respective object prototypes i.e. ER Diagram.

The relationship among the objects have been identified and modelled as a prototype in all the data models as a link between the entities.

Logical Data Models (LDMs) are used to explore either the conceptual design of a database or the detailed data architecture of enterprise. LDMs depict the logical data entities, typically referred as data entities, the data attributes describing those entities and the relationships between the entities. (Anx ‘B’).

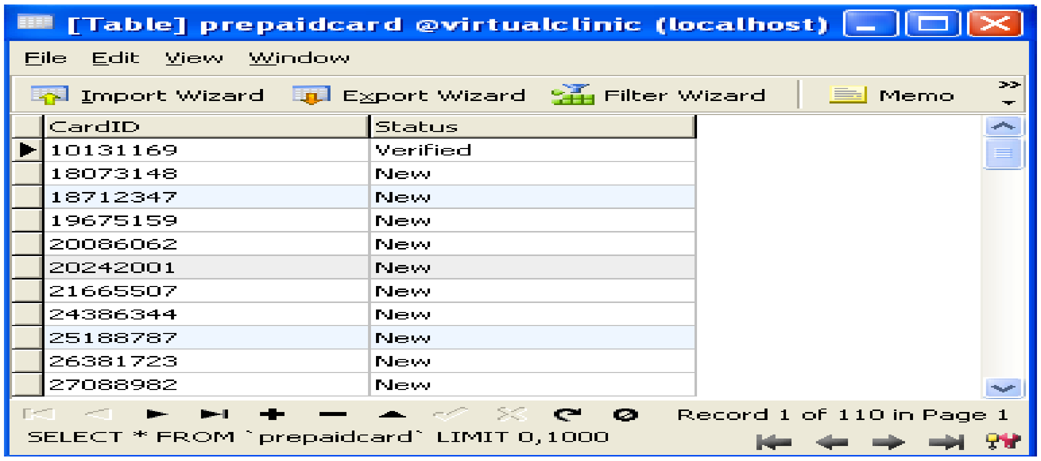
4.3.2 Physical database design

A physical data model is a representation of a data design which takes into account the facilities and constraints of a given database management system. In the lifecycle of a project it is typically derived from a logical data model. A complete physical data model will include all the database artifacts required to create relationships between tables or achieve performance goals, such as indexes, constraint definitions, linking tables, partitioned tables or clusters. The physical data model can usually be used to calculate storage estimates and may include specific storage allocation details for a given database system.

4.3.3 Screen Shots

In proposed **Virtual Clinic** system following are the main database tables that are created for maintaining database records.

- prepaidcard (<u>CardID</u>, Status): To store generated prepaid card numbers

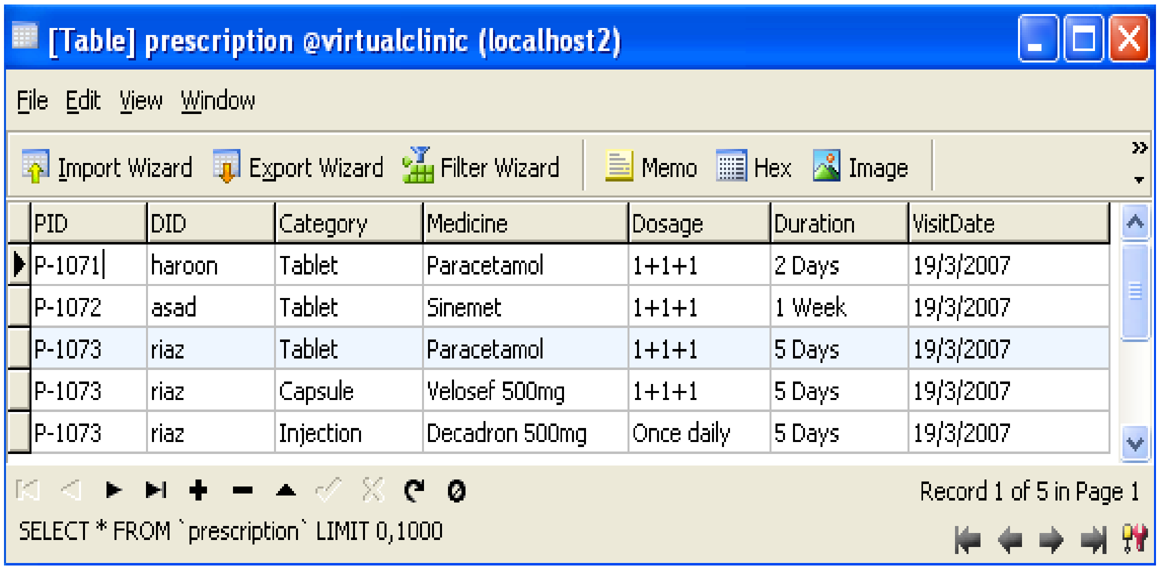

[Table] prepaidcard @virtualclinic (localhost)
File Edit View Window
Import Wizard
Export Wizard
Filter Wizard
Memo
Record 1 of 110 in Page 1

CardID	Status
10131169	Verified
18073148	New
18712347	New
19675159	New
20086062	New
20242001	New
21665507	New
24386344	New
25188787	New
26381723	New
27088982	New

```
SELECT * FROM `prepaidcard` LIMIT 0,1000
```

Figure 4.1

- prescription (<u>PID</u>, DID, Category, Medicine, Dosage, Duration, <u>VisitDate</u>): To store generated prepaid card numbers

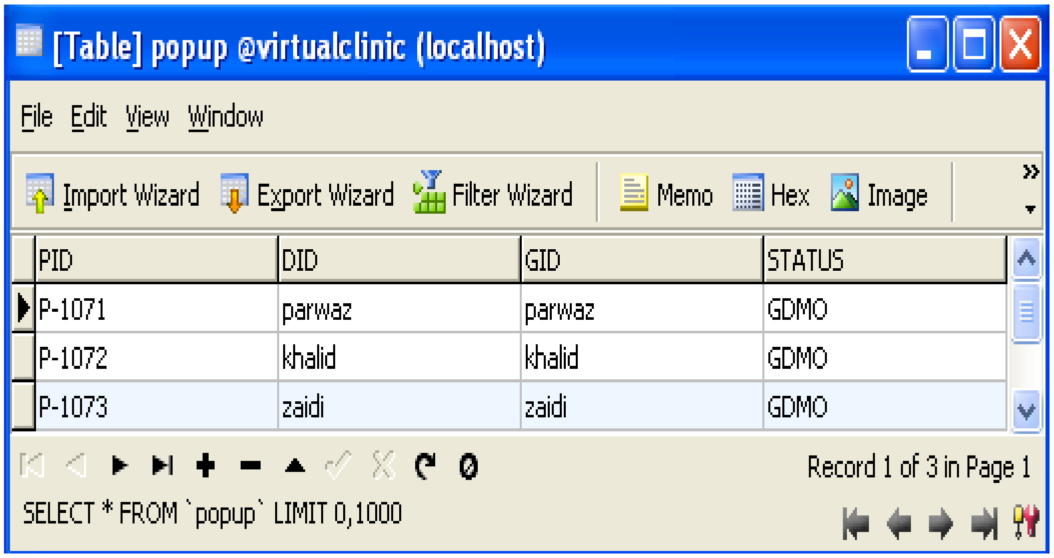


PID	DID	Category	Medicine	Dosage	Duration	VisitDate
P-1071	haroon	Tablet	Paracetamol	1+1+1	2 Days	19/3/2007
P-1072	asad	Tablet	Sinemet	1+1+1	1 Week	19/3/2007
P-1073	riaz	Tablet	Paracetamol	1+1+1	5 Days	19/3/2007
P-1073	riaz	Capsule	Velosef 500mg	1+1+1	5 Days	19/3/2007
P-1073	riaz	Injection	Decadron 500mg	Once daily	5 Days	19/3/2007

```
SELECT * FROM `prescription` LIMIT 0,1000
```

Figure 4.2

- popup (PID, DID, GID, Status)

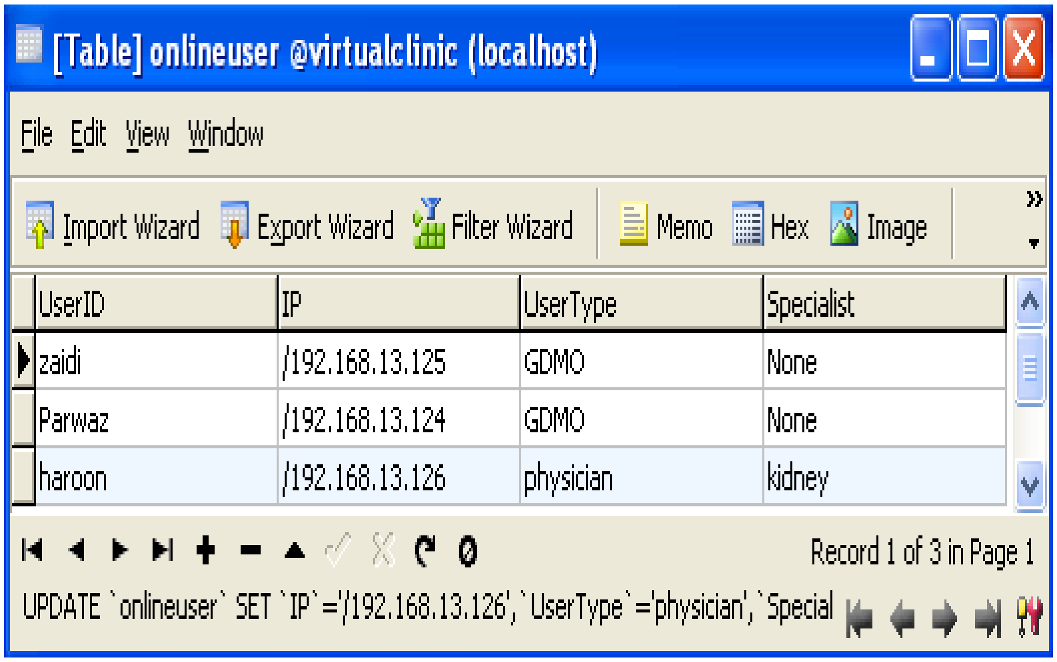


PID	DID	GID	STATUS
P-1071	parwaz	parwaz	GDMO
P-1072	khalid	khalid	GDMO
P-1073	zaidi	zaidi	GDMO

```
SELECT * FROM `popup` LIMIT 0,1000
```

Figure 4.3

- onlineuuser (UserID, IP, UserType, Specialist)

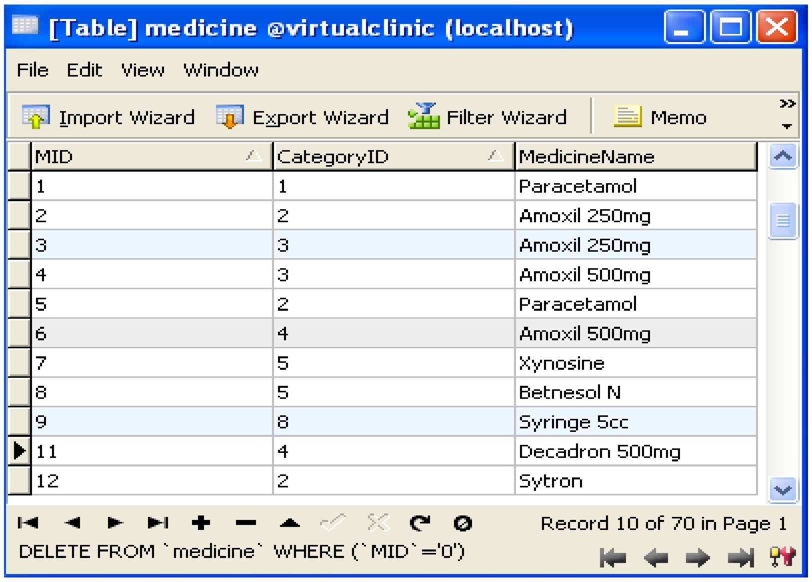


UserID	IP	UserType	Specialist
zaidi	/192.168.13.125	GDMO	None
Parwaz	/192.168.13.124	GDMO	None
haroon	/192.168.13.126	physician	kidney

```
UPDATE `onlineuser` SET `IP`='/192.168.13.126',`UserType`='physician',`Special
```

Figure 4.4

- medicine (MID, CategoryID, MedicineName)


[Table] medicine @virtualclinic (localhost)
File Edit View Window
Import Wizard Export Wizard Filter Wizard Memo

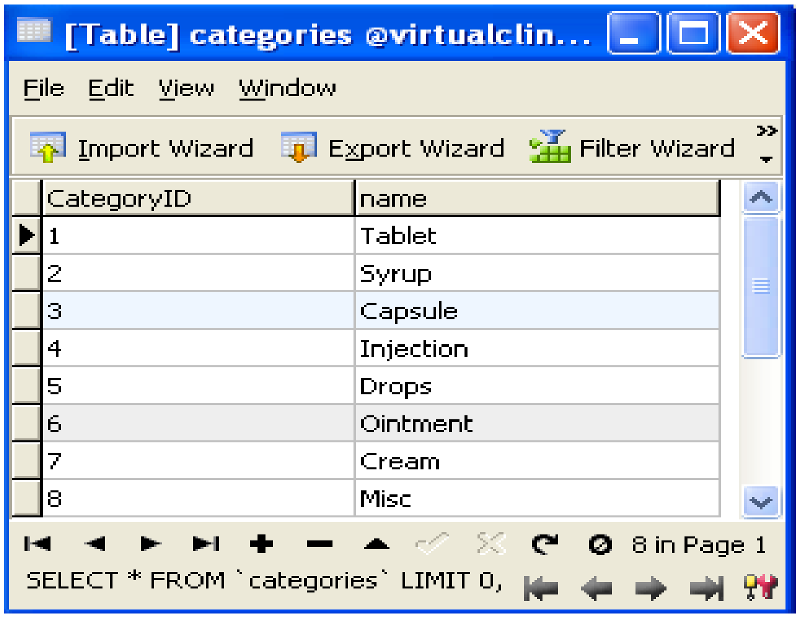
MID	CategoryID	MedicineName
1	1	Paracetamol
2	2	Amoxil 250mg
3	3	Amoxil 250mg
4	3	Amoxil 500mg
5	2	Paracetamol
6	4	Amoxil 500mg
7	5	Xynosine
8	5	Betnesol N
9	8	Syringe 5cc
11	4	Decadron 500mg
12	2	Sytron

Record 10 of 70 in Page 1

```
DELETE FROM `medicine` WHERE (`MID`='0')
```

Figure 4.5

Categories (CategoryID, name)


[Table] categories @virtualclin...
File Edit View Window
Import Wizard Export Wizard Filter Wizard

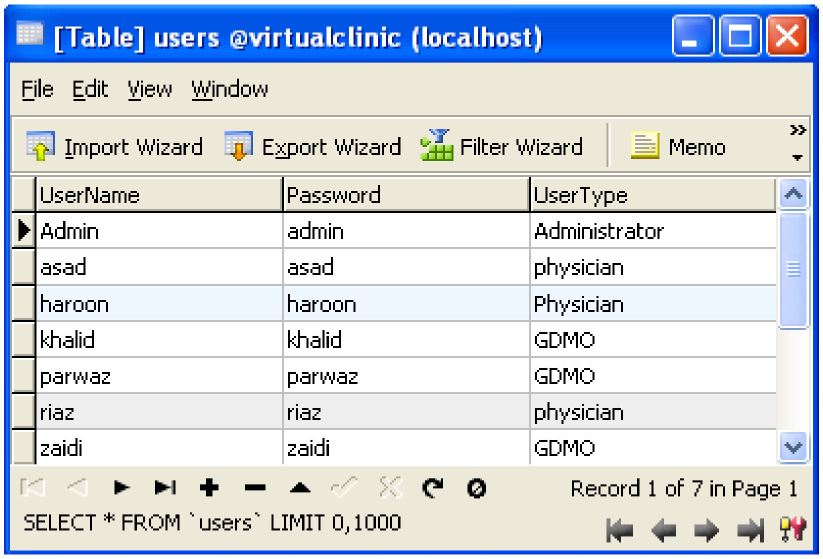
CategoryID	name
1	Tablet
2	Syrup
3	Capsule
4	Injection
5	Drops
6	Ointment
7	Cream
8	Misc

8 in Page 1

```
SELECT * FROM `categories` LIMIT 0,
```

Figure 4.6

- Users (<u>UserName</u>, Password, <u>UserType</u>): To keep users sign-in name and password


[Table] users @virtualclinic (localhost)
File Edit View Window
Import Wizard Export Wizard Filter Wizard Memo

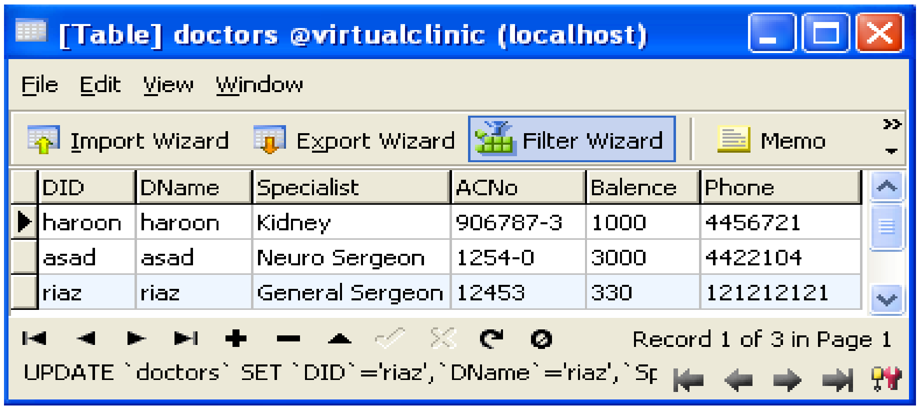
UserName	Password	UserType
Admin	admin	Administrator
asad	asad	physician
haroon	haroon	Physician
khalid	khalid	GDMO
parwaz	parwaz	GDMO
riaz	riaz	physician
zaidi	zaidi	GDMO

Record 1 of 7 in Page 1

```
SELECT * FROM `users` LIMIT 0,1000
```

Figure 4.7

- Doctors (<u>DID</u>, DName, Specialist, ACNo, Balance, Phone): To keep Physician's personal record


[Table] doctors @virtualclinic (localhost)
File Edit View Window
Import Wizard Export Wizard Filter Wizard Memo

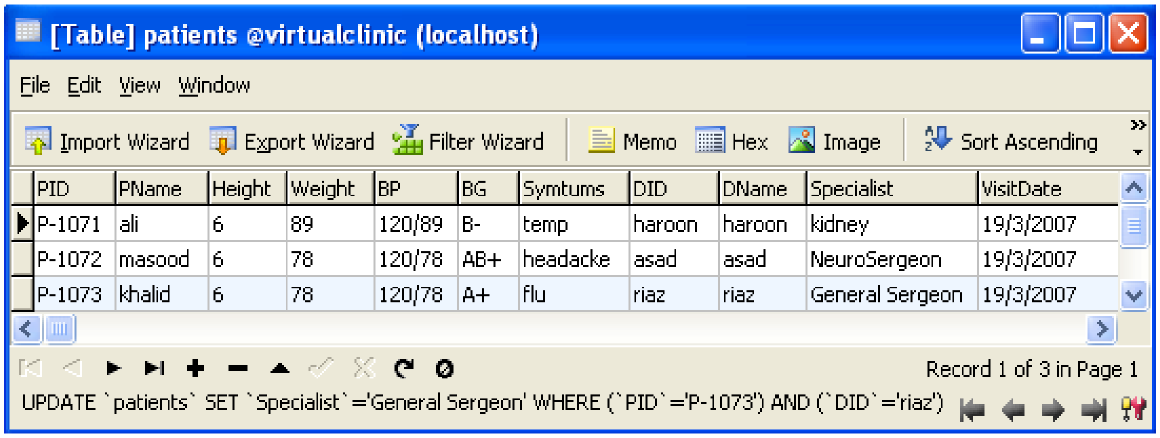
DID	DName	Specialist	ACNo	Balence	Phone
haroon	haroon	Kidney	906787-3	1000	4456721
asad	asad	Neuro Sergeon	1254-0	3000	4422104
riaz	riaz	General Sergeon	12453	330	121212121

Record 1 of 3 in Page 1

```
UPDATE `doctors` SET `DID`='riaz',`DName`='riaz',`Sp
```

Figure 4.8

- Patients (<u>PID</u>, PName, Height, Weight, BP, BG, Symptoms, <u>DID</u>, DName, Specialist, VisitDate): To save and retrieve patient's record



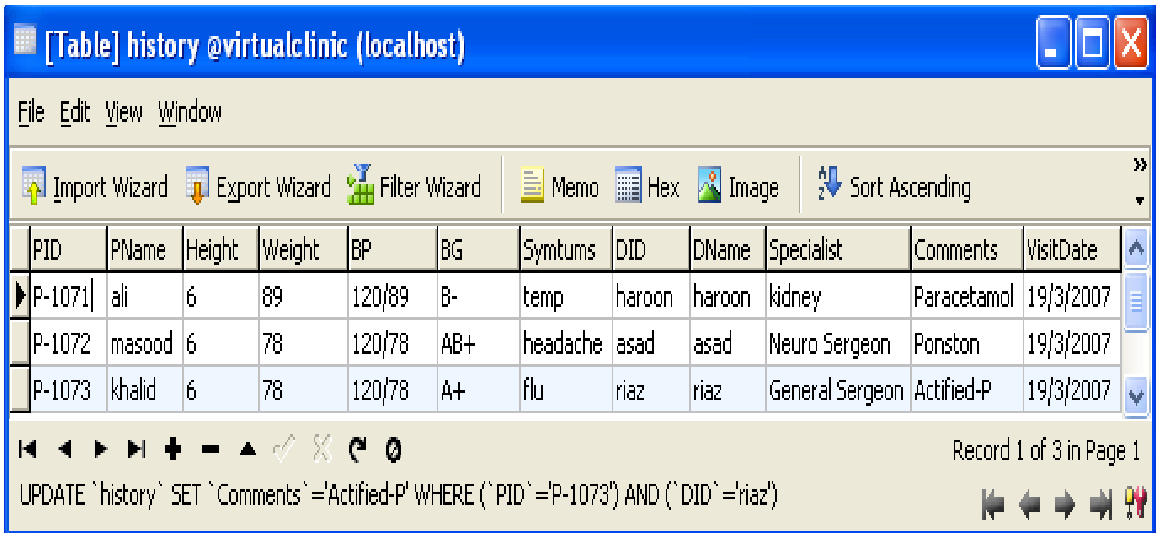
PID	PName	Height	Weight	BP	BG	Symtums	DID	DName	Specialist	VisitDate
P-1071	ali	6	89	120/89	B-	temp	haroon	haroon	kidney	19/3/2007
P-1072	masood	6	78	120/78	AB+	headacke	asad	asad	NeuroSergeon	19/3/2007
P-1073	khalid	6	78	120/78	A+	flu	riaz	riaz	General Sergeon	19/3/2007

Record 1 of 3 in Page 1

```
UPDATE `patients` SET `Specialist`='General Sergeon' WHERE (`PID`='P-1073') AND (`DID`='riaz')
```

Figure 4.9

- History (<u>PID</u>, PName, Height, Weight, BP, BG, Symptoms, <u>DID</u>, DName, Specialist, Comments, <u>VisitDate</u>): To maintain patient's history



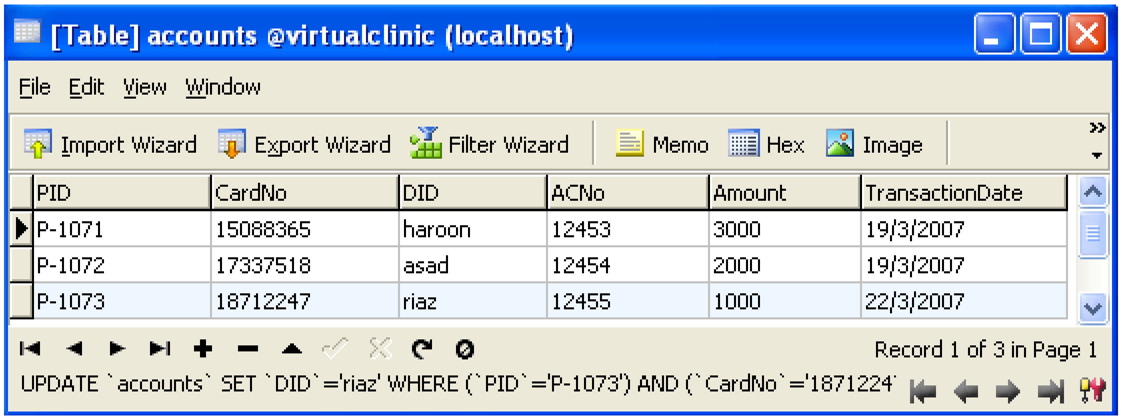
PID	PName	Height	Weight	BP	BG	Symtums	DID	DName	Specialist	Comments	VisitDate
P-1071	ali	6	89	120/89	B-	temp	haroon	haroon	kidney	Paracetamol	19/3/2007
P-1072	masood	6	78	120/78	AB+	headache	asad	asad	Neuro Sergeon	Ponston	19/3/2007
P-1073	khalid	6	78	120/78	A+	flu	riaz	riaz	General Sergeon	Actified-P	19/3/2007

Record 1 of 3 in Page 1

```
UPDATE `history` SET `Comments`='Actified-P' WHERE (`PID`='P-1073') AND (`DID`='riaz')
```

Figure 4.10

- Accounts (<u>PID</u>, CardID, DID, ACNo, Amount, <u>TransactionDate</u>): To save Physician's financial assistance


[Table] accounts @virtualclinic (localhost)
File Edit View Window
Import Wizard Export Wizard Filter Wizard Memo Hex Image
Record 1 of 3 in Page 1

PID	CardNo	DID	ACNo	Amount	TransactionDate
P-1071	15088365	haroon	12453	3000	19/3/2007
P-1072	17337518	asad	12454	2000	19/3/2007
P-1073	18712247	riaz	12455	1000	22/3/2007

```
UPDATE `accounts` SET `DID`='riaz' WHERE (`PID`='P-1073') AND (`CardNo`='1871224
```

Figure 4.11

4.4 Conclusion

This chapter fully describes the design of Virtual Clinic in detail, which includes System design, Input / Output design, Code design, User Interface design, Logical / Physical database design, Keys, Database design, and the Normalization.

Chapter 5

System Implementation

This chapter focuses on the implementation of Virtual Clinic.

5.1 Outline

In this chapter, focus is on how the implementation is actually done. There were a number of decisions necessary to be taken before starting real implementation i.e. **coding according to design**. The **decisions** are:

- Which **operating system** is to be selected?
- Which **programming language** would be **suitable**?
- Which **tool** is to be selected? etc.

5.2 Platform Selection

Virtual Clinic is platform independent application. So platform selection does not matter. Some platforms support multitasking for threads and others don't. **Virtual Clinic** does not depend upon **specific platform** or **operating system. Windows XP Professional** is chosen as platform for development of Virtual Clinic, since it supports thread multitasking.

5.3 Language Selection

According to the requirements of the project Java programming language is chosen for development of the Virtual Clinic application in order to provide platform independent functionality.

5.4 Development Tools

The proper list of development tools which are used is given below.

- JDK 1.5.0
- Eclipse 3.1.0

- Java Visual Editor
- JCreator Pro
- MySql

5.4.1 MySql

My Sql will be used as back end to create database for proposed system due to following features.

5.4.1.1 Single User as well as Multi Users

It supports and provide the facility of single user and as well as the multi user access. It allows the users to share the data.

5.4.1.2 Connectivity

The back end connectivity is very easy and allows different types of computers and operating systems to share information across the networks.

5.4.1.3 Client / Server Environment

It supports the client / server environment.

5.4.1.4 Portability

The database application created or developed can be ported to other operating systems with minor or no modifications.

5.4.1.5 Security and Control

It provides control access to the database and maintains security.

5.4.1.6 Large Amount of Data

It is capable of keeping and handling large amount of data.

5.5 System Implementation

System implementation is the phase in which most of the computer programming occurs. The purpose of system implementation in the life cycle of a project is to build a working system from design specifications prepared during the design phase. The main goal of software implementation is to develop executable software based on the proposed system. The idea of project development is carefully planned here before being implemented on computer. Programming is the combination of science as well as the touch of art is also involved in it. There is however a number of clearly identifiable steps that are always involved in the programming phase and those provide a convenient framework.

To achieve the purpose of implementation phase, following are the objectives, which must be accomplished.

- Construct or Install system components
- Implement Designed System Functions
- Involve end-Users in Pertinent Construction Activities

5.5.1 Construct or Install system components

In this phase, we constructed/install the hardware, software and data storage components of the new system. Actually, construct may not be the best term for what happens during this phase. Construct implies building from scratch. Although many programs are built from scratch, others may actually be modified to fulfill new requirements.

Furthermore, hardware components are not constructed. Now during the implementation phase, the programmers can utilize installed hardware to construct the system. It is worth noting that the programmers may have to learn how to use hardware and software packages before they can develop or modify the system. Still the most time consuming activity is computer programming and maintenance.

5.5.2 Implement Designed System Functions

It is very important that the programmer not stray from the design specifications without the analyst's approval. Why? Because the design specifications were prepared to fulfill end-user requirements. A change in these specifications could throw the system out of synchronization with the end-user's requirements.

5.5.3 Involve End-Users in Pertinent Construction Activities

According to *Daniel Tkach* (1998), "Before starting the implementation of an application it is necessary to establish the requirements. Creating a short statement of system purpose, from the prospective of both the user and the expert, helps focus the designer on solving the problem".

5.6 Conclusion

This chapter focuses on the actual implementation of Virtual Clinic. It describes platform selection, language selection, development tools, back-end database application, features of Microsoft Access and system implementation process

Chapter 6

System Testing & Evaluation

This chapter describes the testing strategies & evaluation of Virtual Clinic.

6.1 Outline

This chapter describes the various testing strategies applied to test and evaluate the system.

6.2 System Testing

System testing is the process of executing the developed system by putting the real values for checking for finding errors i.e. to know where the program fails. Testing of any system has two objectives i.e. **validation** and **verification** of the basic logic of the system, and to ensure that the entire system works properly.

6.3 Validation

These processes involve showing the system to the user and checking whether it fulfills his expectations.

6.4 Verification

This process involves testing the system according to requirement specifications. It is checked that whether implementation is according to specifications. So **Virtual Clinic** is **validated** as well **verified** as per requirement specifications.

6.5 Testing Strategies

Virtual Clinic application is developed using **object-oriented design**, so **bottom-up strategy** is selected for testing. Small components are first tested and then overall system is tested after integrating its subsystems.

Each class is tested according to its specification of design and according to transition behavior. The system testing was performed, errors were reported and fixed. Following are the four steps followed:

6.5.1 Unit Testing

Initially tests focus on each module individually, assuring that it functioning properly as a unit. The advantage of **unit testing** is that errors, if any, which are module, focused can be dealt at this level individually. This type of testing has been conducted for multiple modules of the **Virtual Clinic** in parallel.

6.5.2 Integration Testing

The modules must be assembled or integrated to form the complete software package. **Integration testing** addresses the issue associated with the dual problem of verification and program construction.

It also encompasses the testing of the software design. Using **MySql** and **JDBC:ODBC**, windows were developed and tested separately first and then integrated into a proper system, following the input design and the output design. Then the integration testing took place.

6.5.3 Validation Testing

Validation testing verifies that all elements are working properly and that the overall system function and performance is achieved. At this stage the interfacing errors were uncovered and corrected and this testing proved to be the final series of software tests and thus regarded as validation tests. This is where the CMH Surgical Department took an active part and reported bugs that were encountered during its use.

6.5.4 Code Testing

The code testing of the of the whole system is done by testing each module independently, which assures that code is correct and working properly as per defined specifications and desired required output.

The coding of the system is done in a way that is readable, understandable and comments were given properly. The coding errors were detected and corrected during compilation and execution of the project.

6.6 Project Testing Report

The project testing report reveals that the whole system is fully tested and functioning properly. All the independent modules are working individually and integrated successfully.

For the “Virtual Clinic” unit testing, integration testing and validation testing were performed and it was found that system fulfils the requirements. Therefore, from the various testing strategies applied for the testing of the system it is proved that proposed system Virtual Clinic is fully tested and working properly under the specified problem statement and covers defined project scope.

6.7 User Training

The users of the system were trained on this system. They found the system user friendly and ease in use. They interact with the system by performing various tasks and getting the desired and required output as per their expectations.

6.8 System Evaluation

Software evaluation involves whether objectives, functional/non-functional requirements, of user are fulfilled or not. Virtual Clinic evaluated according to its functional/non-functional requirements.

Following are main features of the Virtual Clinic.

- Only authorized users can access and use the system
- Automated generation of prepaid card numbers
- Prepaid card number verification
- Efficient patient’s record insertion and retrieval.
- Fast record searching
- Reliable patient’s diagnoses
- Scanning the patient’s reports and saved in JPEG/JPG format
- Taking patient’s injured area picture and saved in JPEG/JPG format
- Sending and receiving patient’s reports
- Viewing patient’s reports
- Maintaining patient’s history

- Physician’s credit update
- Changing user’s password
- Provide instant help while using the software.
- Provide an user friendly GUI (Graphical User Interface)
- Provides Windows XP look and feel on any Operating System

From the above merits / features the evaluation of the proposed system is very much clear and insures that the system is fully functional.

6.9 Conclusion

This chapter concludes that the proposed Virtual Clinic system undergoes the various testing strategies like unit testing, integration testing, validation testing and code testing which assure the functionality of the system is up to the requirements.

Chapter 7

Future Extensions / Enhancements

This chapter describes the future extensions and enhancements of proposed system.

7.1 Outline

This chapter is about the future extension / enhancement of the proposed system that can be implantable in this system.

7.2 Future Extensions / Enhancements

The proposed **Virtual Clinic** system can be extended as per future requirements. The possible future enhancements are given and illustrated in following.

7.2.1 Online Internet Based System

This system can be implemented as an online system by placing it on the web. There are only minor changes are required for online implementation of the system. With this extension the patient can examine himself while sitting at home using internet.

7.2.2 Online Physician's Selection

Another enhancement can be made as the online physician's selection by the patient. The physician's name, field of specialization and other details would be placed on the web page for easy selection.

7.2.3 Online Patient's Registration

This system can be extended in order to make online patient's registration. This process is used for new patients in order to give patient's ID for future use and reference.

7.2.4 Intelligent System

Make this system intelligent so that the system can analyze the patient's symptoms and display the list of suitable and available physicians for the specified symptoms, and patient can select the physician of own choice.

7.2.5 Video Conferencing

This system can be extended by implementing a module which provides the facility of live video conferencing with the physician. This will help the physician in better treatment of the patient.

7.2.6 Online Physicians and GDMO Registration

System can be extended for online registration of Physicians and GDMO, so that an organization can hire their staff online.

Chapter 8

User Guide

This chapter is about the user’s guide for the proposed system.

8.1. Outline

This chapter gives an idea to the users of the system (Physicians and GDMOs) i.e. how to interact with system.

8.2. Administrator's User Guide

The administrator has to sign-in by entering administrator user name and password before using and accessing the system. Administrator can perform following tasks.

8.2.1. Generation of new prepaid card numbers

In order to generate new prepaid card numbers, the administrator first select choice of **Generate Prepaid Card Numbers** and then press **Generate** button. Then the list of newly generated prepaid card numbers will displayed. By default 50 prepaid card numbers are generated by the system.

8.2.2. Creation of new users

For creating new user of the system i.e. GDMO or Physician then first administrator select the choice of **Create New User** and then enter the new user's name and password. To create GDMO select GDMO **radio button** and for creating Physician select **Physician radio button**.

If the user is **GDMO** then **Save** button becomes enable and if the **Physician radio button** is selected then other fields becomes enabled and by entering physician's phone number **Save** button becomes enable. Finally press the **Save** button to save information into the database.

8.2.3. Changing own password

If the administrator wants to change own password then press the **Change Password** button. A screen will be displayed which helps in changing the password.

8.2.4 Entering new record of medicine

When ever new medicine is introduced in the market and recommended by the doctors enters it in the database.

8.3. GDMO's User Guide

The user's guide for GDMO is described in the following.

8.3.1. Verify Prepaid Card

Press this button to verify the prepaid card given by patient from the database. The server will response either the card is valid or invalid.

8.3.2. Patient's Record / Symptoms

This button is used to enter patient record / symptoms into the database. There will be two options i.e. **View Existing Patient Record** or **New Case**. To view the existing patient's record, enter Patient's ID and press the **View** button. The **New Case** button will ask for two choices i.e. **New Patient** or **Existing Patient**. Select the appropriate choice to continue. In case of **New Patient** entirely new patient record / symptoms are inserted where as in case of **Existing Patient** the record / symptoms of the existing patient will be updated in the database.

8.3.3. Scan Report / X-Ray / Take Picture

This button is used to scan patient's reports if any like **X-Rays, Blood reports, Urine reports, Sugar Reports** etc or to **take picture** of patient's injured area. These reports or picture will be saved in the system for sending.

8.3.4. Send Patient Report

This button helps in sending the scanned reports / taken pictures to the physician. GDMO will upload the patient's reports to the server. Either a single file can be uploaded or a folder containing various reports.

8.3.5. View Physician Comments

This button is used to view the physician's comments for the referred patient by entering the patient's ID.

8.3.6. Print Prescription

This button is used to print the patient's prescription in the end and will help in future use.

8.3.7. Change Password

This button will help in changing the password. First enter the old password then new password and confirm the new password and finally save it.

8.3.8. Sign Out From Application

To signed-out or exit from the application.

8.4. Physician's User Guide

Physician's user guide is described in the following.

8.4.1. View patient record / symptoms

This button is used for viewing the referred patient record / symptoms and giving own comments by examining all details like reports (if any) and listening voice. First enter the referred patient's ID and press **View** button. Then give own comments and press the **Save** button to insert / update record into the database.

8.4.2. Receive patient reports

This button is used for downloading the patient's reports from the FTP Server. The down loaded reports will be saved in the current system.

8.4.3. View patient reports

The down loaded reports can be viewed by pressing this button. Browse the report that is to be viewed by press **Browse** button.

8.4.4. View examined patients

This button will help the physician to view the list of patients that are examined by him.

8.4.5 Check credit update

Physician can check his account by pressing this button. This will show the current balance in the account.

8.4.6 Change password

This button will help in changing the password. First enter the old password then new password and confirm the new password and finally save it.

8.4.7 Sign out from application

To signed-out or exit from the application.

8.5. Conclusion

This chapter concludes the user’s guide of system users i.e. how **Administrator**, **GDMO** and **Physician** will interact with the system?

Annex

Anx 'A'

1. Data Flow Diagram

Data flow diagrams show the flow of data through the system and the work or processing performed by that system. Types of DFDs are as following

1.1 Logical DFDs.

The **Logical DFDs** show only the data and processes

1.2 Physical DFDs.

The **Physical DFDs** show the sequence of process, changes to data and files, and reports or other outputs.

Data Flow Diagrams can be constructed at several levels to show the different amounts of details about the systems. Types of data flow diagrams are:

1.3 Context Diagram

In **context diagram** or **level zero**, the emphasis is given on the relationship between the system and its environment. The system as a whole is represented by bubble or circle and external entities are shown from which input flows and to which output is directed.

Context Level DFD

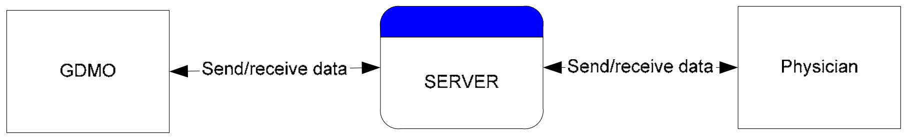


Figure 1

Zero Level DFD

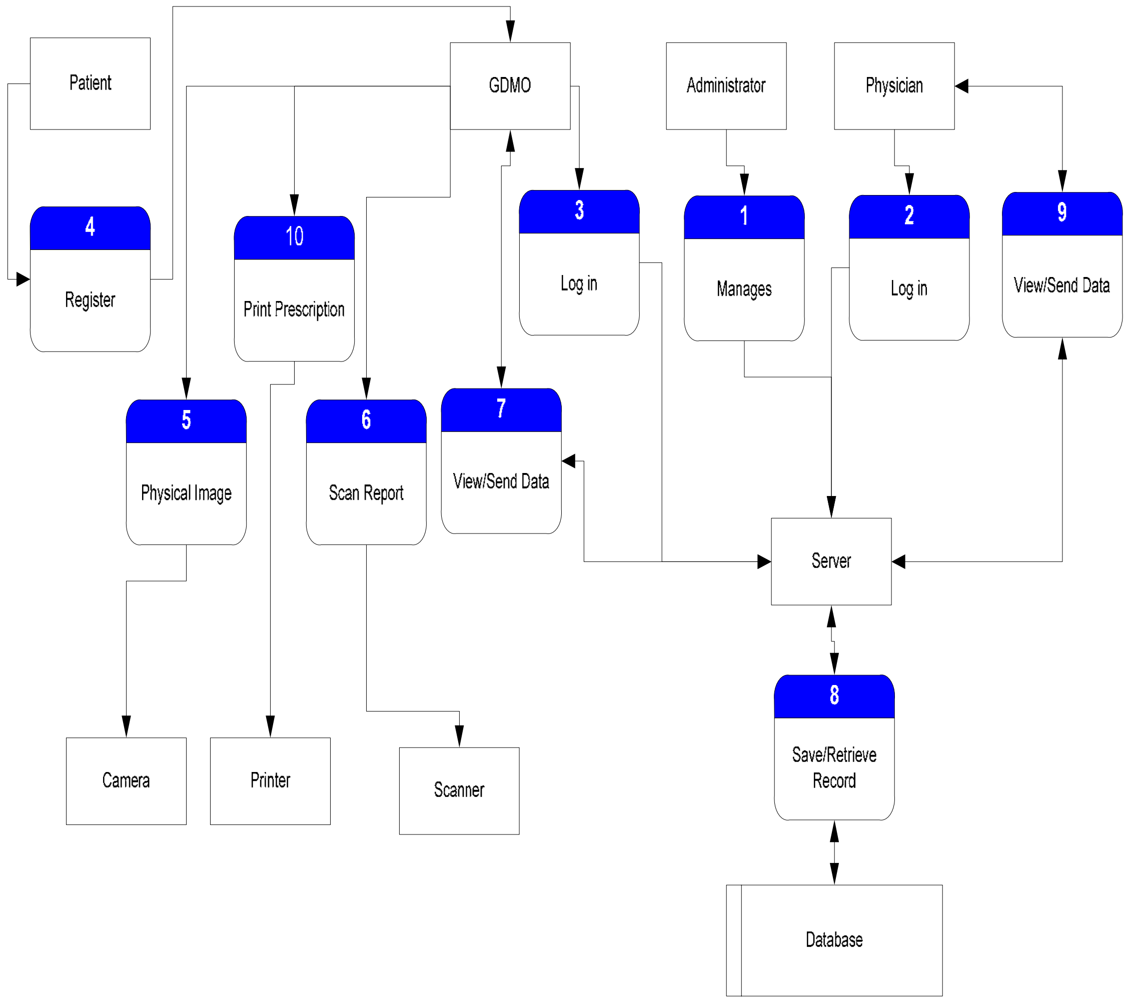


Figure 2

1.4 Level one

Level One is the diagram showing the system itself. It pictures the major process along with the external entities, data stores and data flow. It is a single, top-level diagram of system and does not show each process in detail. The parts of this diagram can be expanded using detailed diagrams.

Level 1 DFD

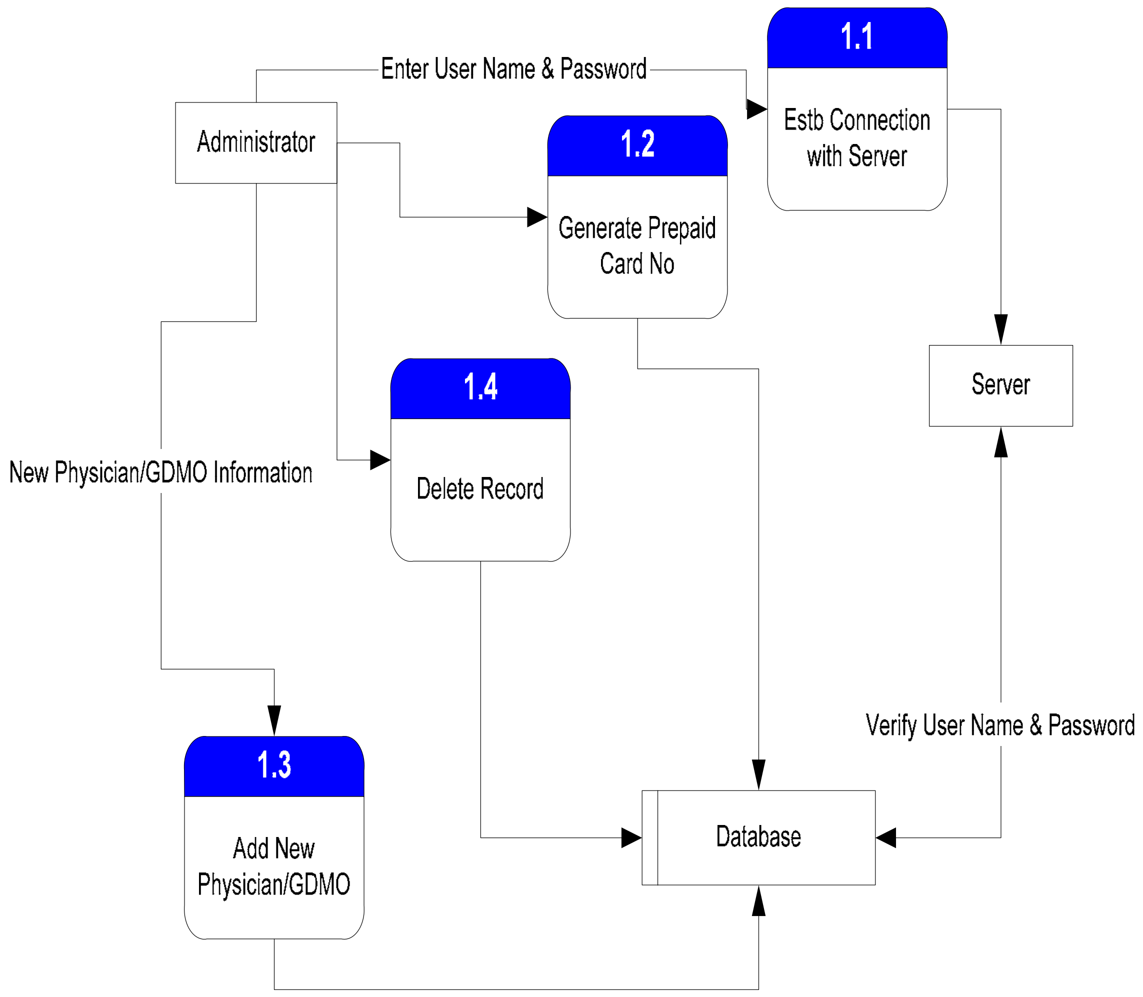

1.1
Estb Connection with Server
Enter User Name & Password
Administrator
1.2
Generate Prepaid Card No
Server
1.4
Delete Record
New Physician/GDMO Information
Verify User Name & Password
1.3
Add New Physician/GDMO
Database

Figure 3

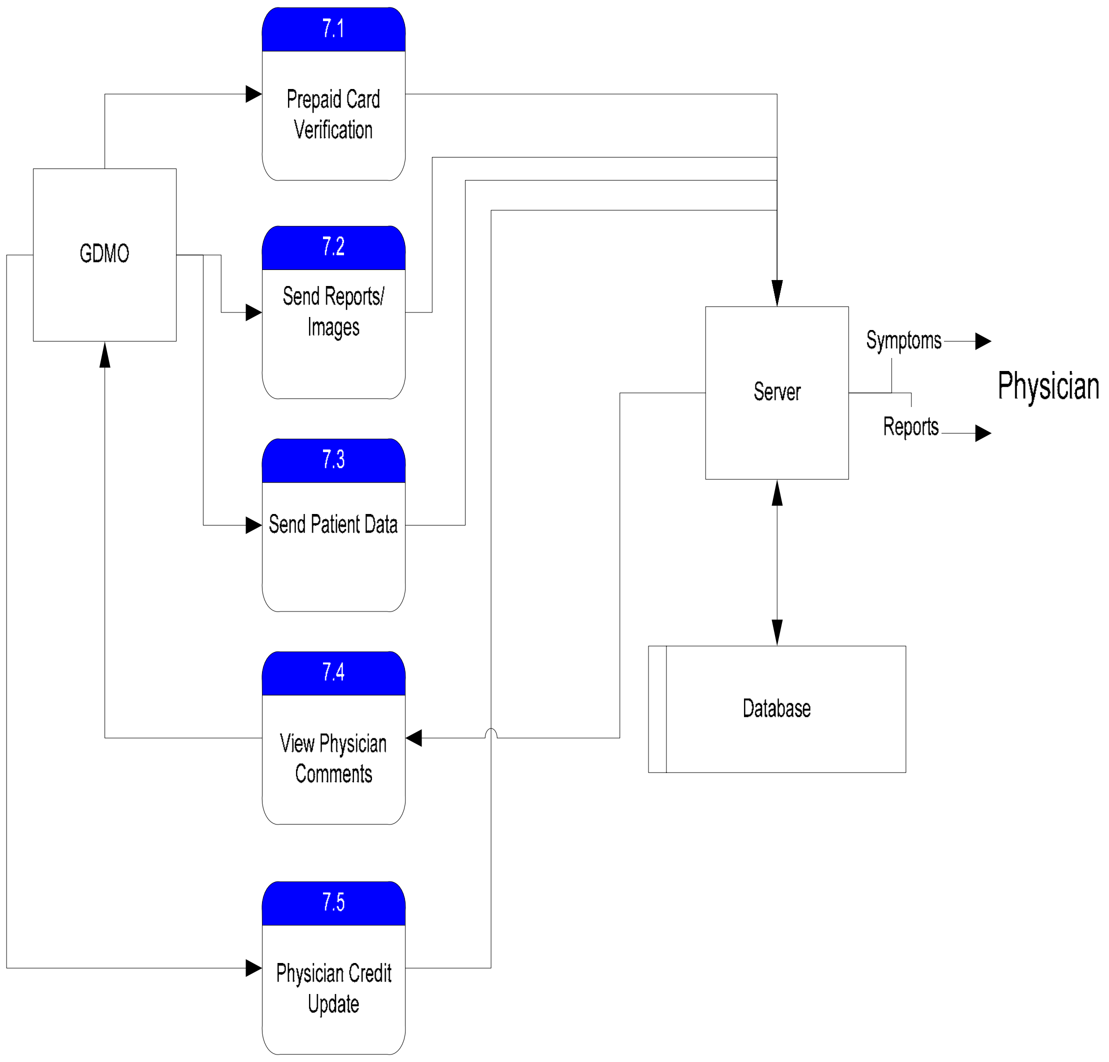

7.1
Prepaid Card Verification
7.2
Send Reports/ Images
7.3
Send Patient Data
7.4
View Physician Comments
7.5
Physician Credit Update
GDMO
Server
Symptoms
Reports
Physician
Database

Figure 4

E-R Model

E-R Model diagram of Virtual Clinic database is depicted below.

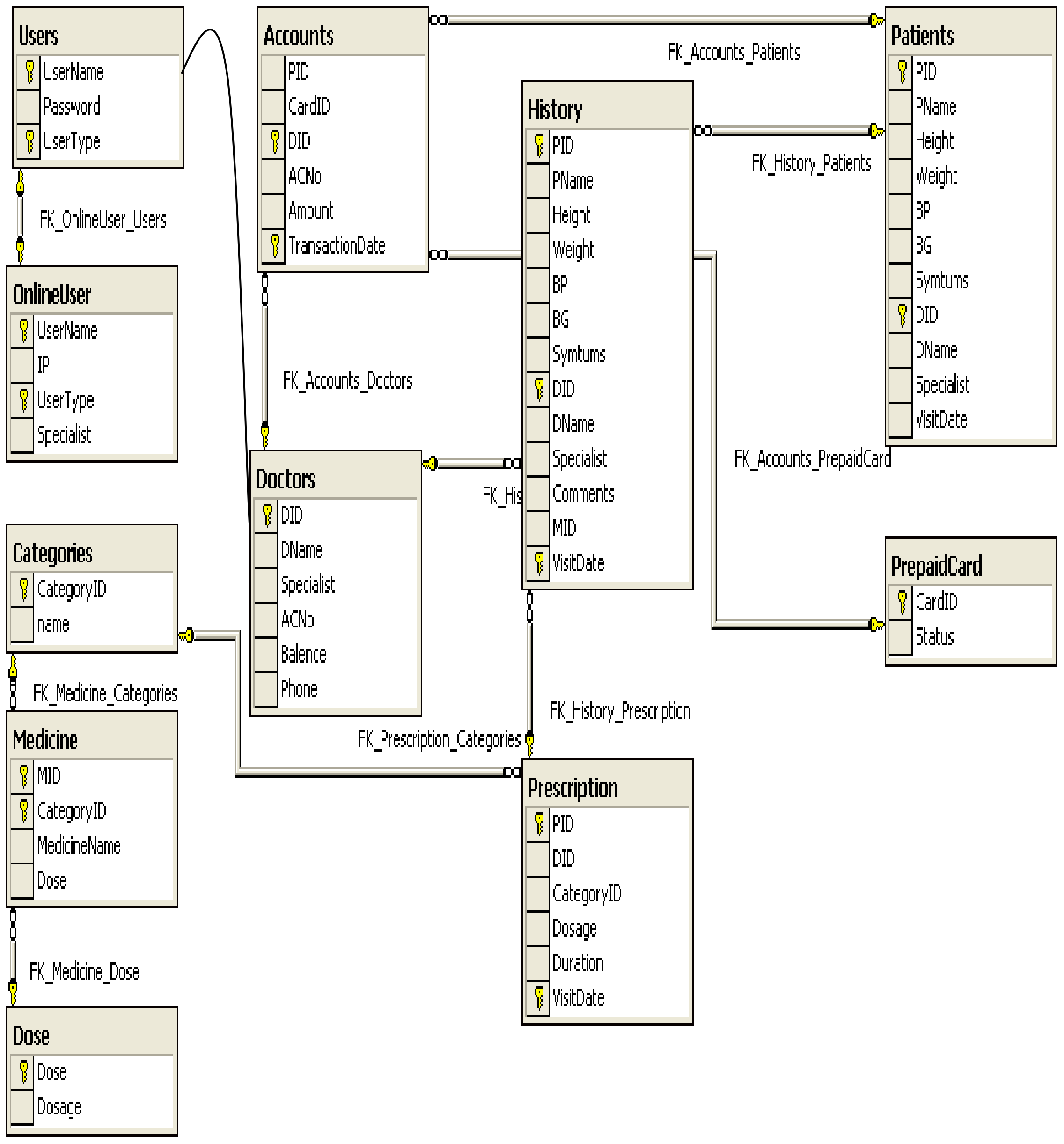
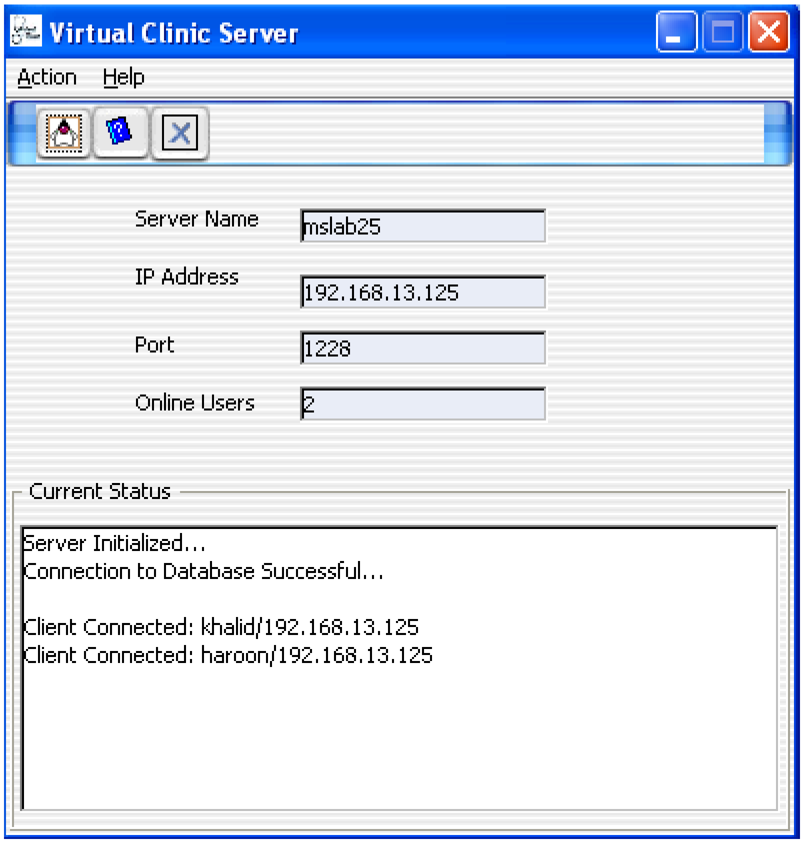


Figure 1

Anx 'C'

GUI's

Main Server Interface



Server Name: mslab25

IP Address: 192.168.13.125

Port: 1228

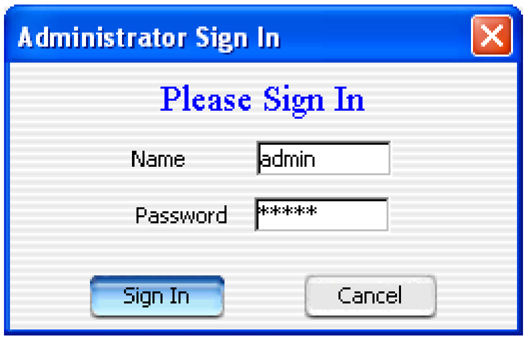
Online Users: 2

```
Server Initialized...
Connection to Database Successful...

Client Connected: khalid/192.168.13.125
Client Connected: haroon/192.168.13.125
```

Figure 1

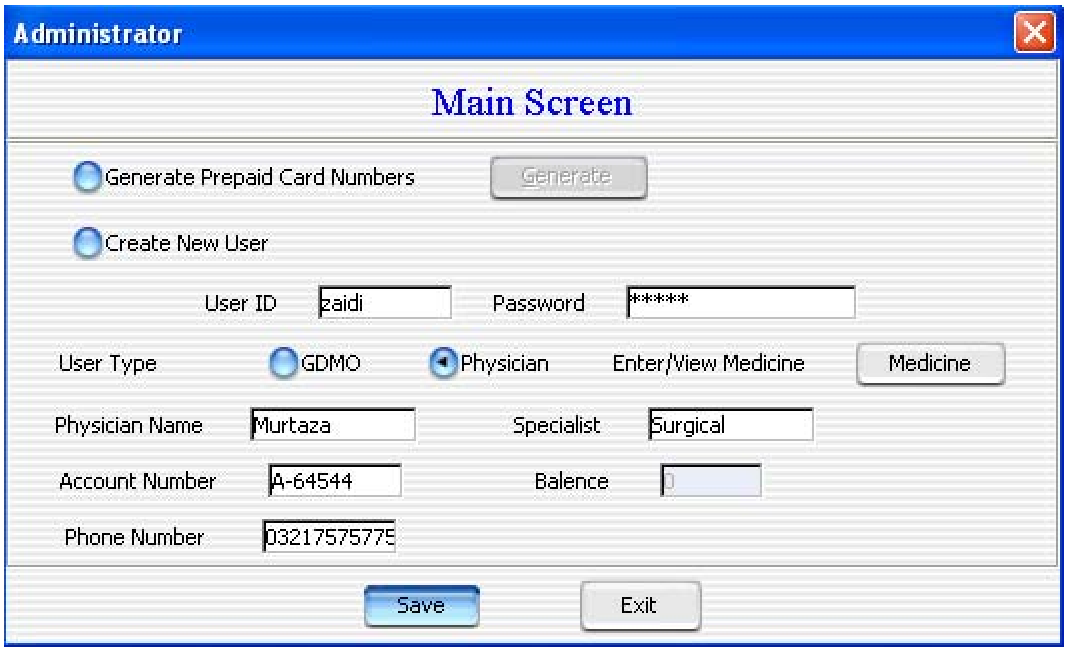
Administrator Signin Interface



Name: admin
Password: *****

Figure 2

Administrator User Creation Interface

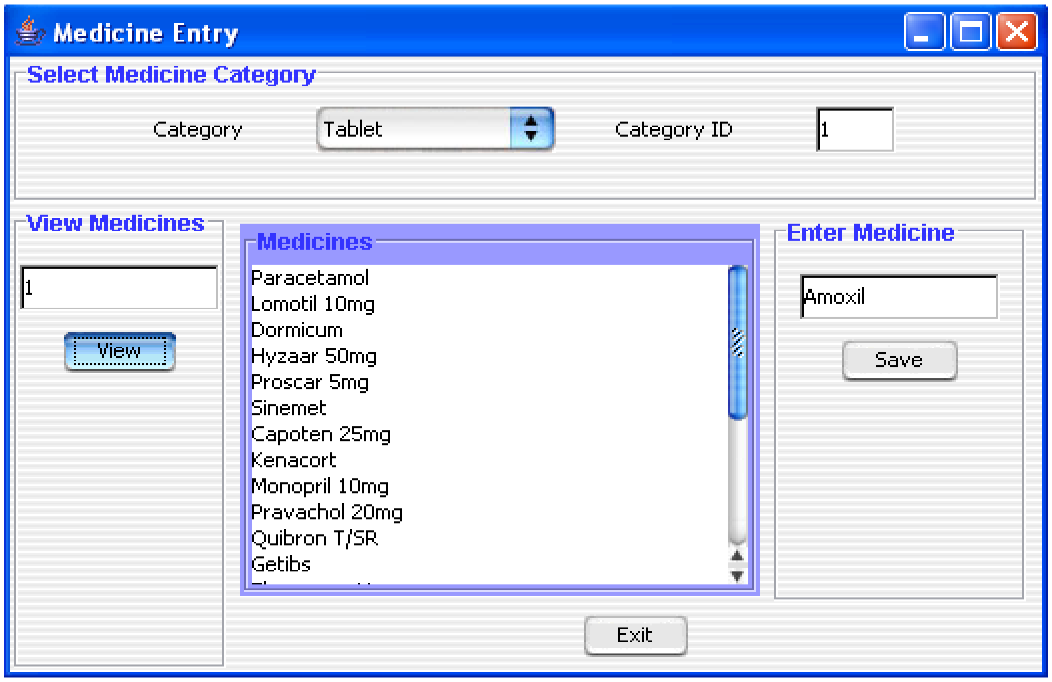


User ID: zaidi
Password: *****
User Type: GDMO [✓ Physician]
Enter/View Medicine

Physician Name: Murtaza
Specialist: Surgical
Account Number: A-64544
Balence: 0
Phone Number: 03217575775

Figure 3

Medicine Entry Interface



Category: Tablet

Category ID: 1

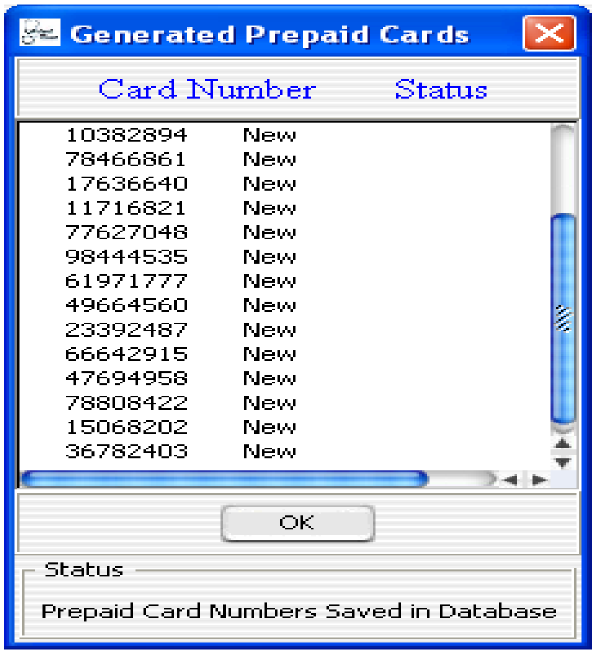
1

Paracetamol
Lomotil 10mg
Dormicum
Hyzaar 50mg
Proscar 5mg
Sinemet
Capoten 25mg
Kenacort
Monopril 10mg
Pravachol 20mg
Quibron T/SR
Getibs

Amoxil

Figure 4

Generate Prepaid Card Interface

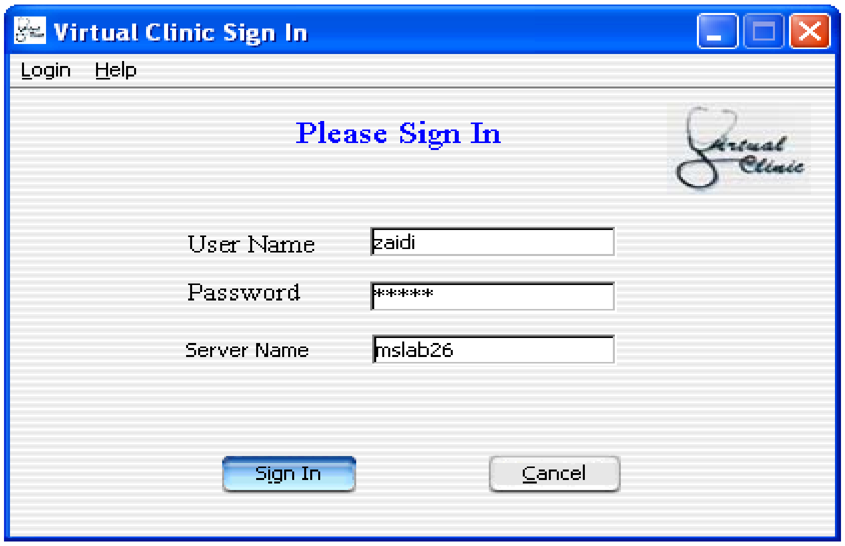


Card Number	Status
10382894	New
78466861	New
17636640	New
11716821	New
77627048	New
98444535	New
61971777	New
49664560	New
23392487	New
66642915	New
47694958	New
78808422	New
15068202	New
36782403	New

Prepaid Card Numbers Saved in Database

Figure 5

Sign In Interface



User Name: zaidi
Password: ******
Server Name: mslab26

Figure 6

GDMO’s Interface

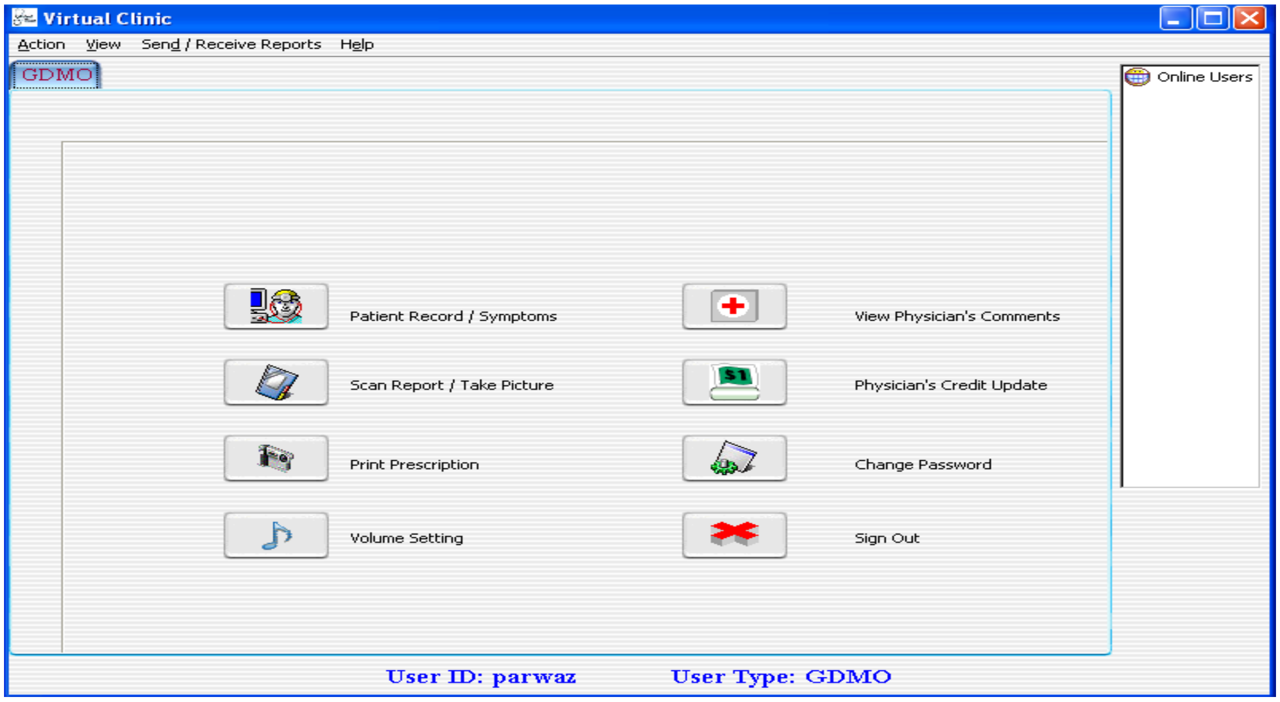

Virtual Clinic
Action View Send / Receive Reports Help
GDMO
Online Users
Patient Record / Symptoms
View Physician's Comments
Scan Report / Take Picture
Physician's Credit Update
Print Prescription
Change Password
Volume Setting
Sign Out
User ID: parwaz
User Type: GDMO

Figure 7

New Patient Entry Interface

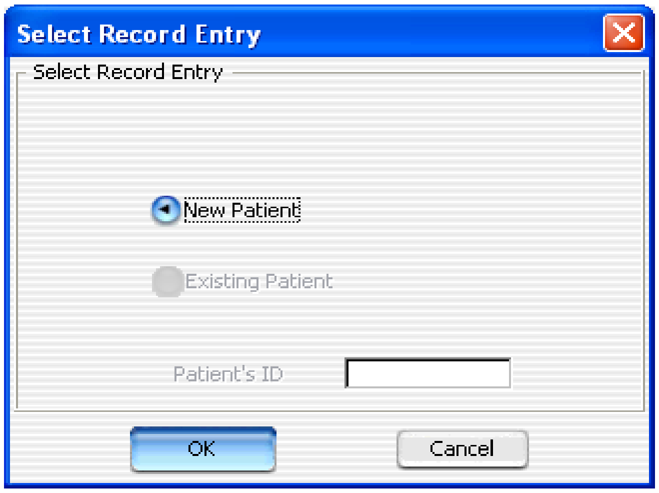
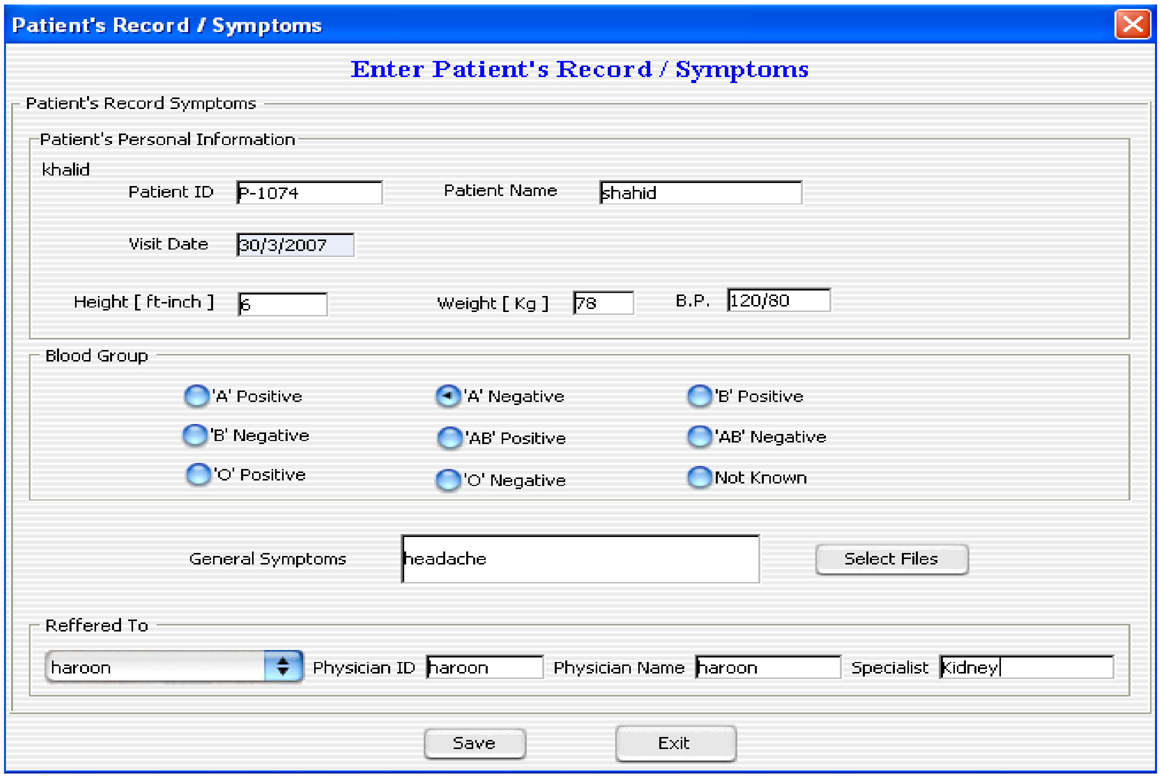


Figure 8

Enter Patient’s Record / Symptoms Interface



Patient's Record / Symptoms

Enter Patient's Record / Symptoms

Patient's Record Symptoms

Patient's Personal Information

khalid

Patient ID: P-1074   Patient Name: shahid

Visit Date: 30/3/2007

Height [ft-inch]: 6   Weight [Kg]: 78   B.P.: 120/80

Blood Group

'A' Positive [✓ 'A' Negative] 'B' Positive

'B' Negative 'AB' Positive 'AB' Negative

'O' Positive 'O' Negative Not Known

General Symptoms: headache   Select Files

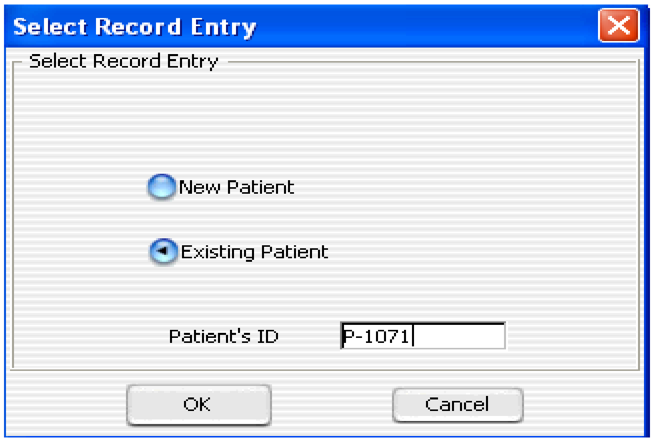
Reffered To

haroon   Physician ID: haroon   Physician Name: haroon   Specialist: Kidney

Save   Exit

Figure 9

Select Record Entry Interface



Select Record Entry

Select Record Entry

New Patient

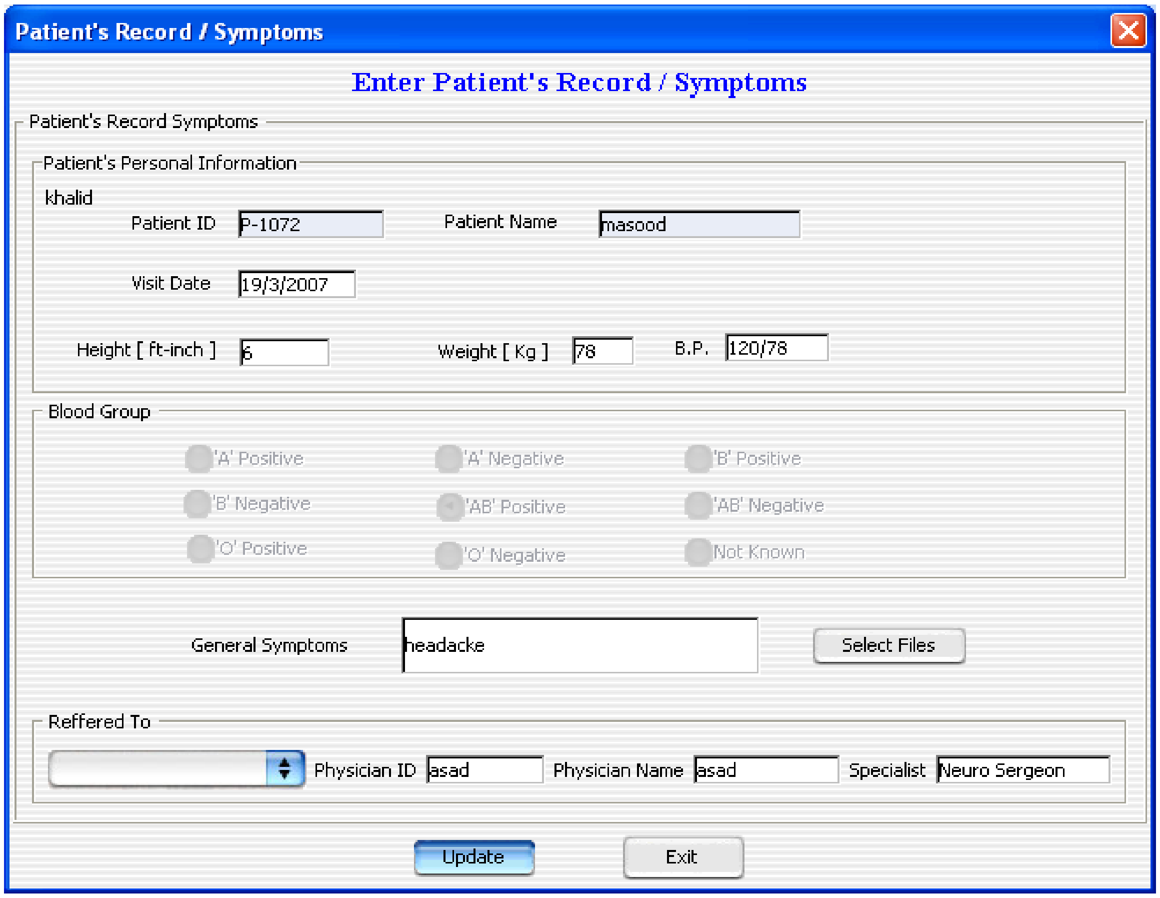
[✓ Existing Patient]

Patient's ID: P-1071

OK   Cancel

Figure 10

Record Update Interface



khalid

Patient ID P-1072

Patient Name masood

Visit Date 19/3/2007

Height [ft-inch] 6

Weight [Kg] 78

B.P. 120/78

General Symptoms headacke

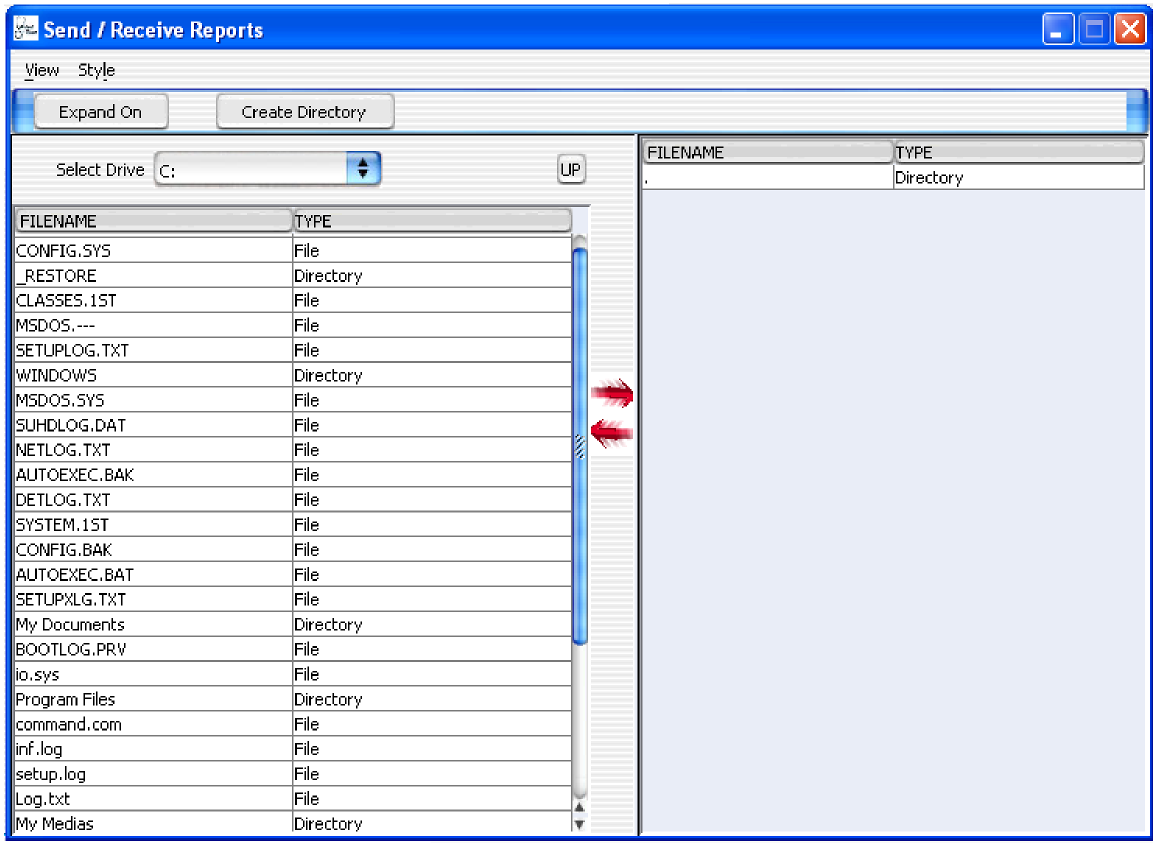
Physician ID asad

Physician Name asad

Specialist Neuro Sergeon

Figure 11

Send Patient's File(s) / Report(s) Interface

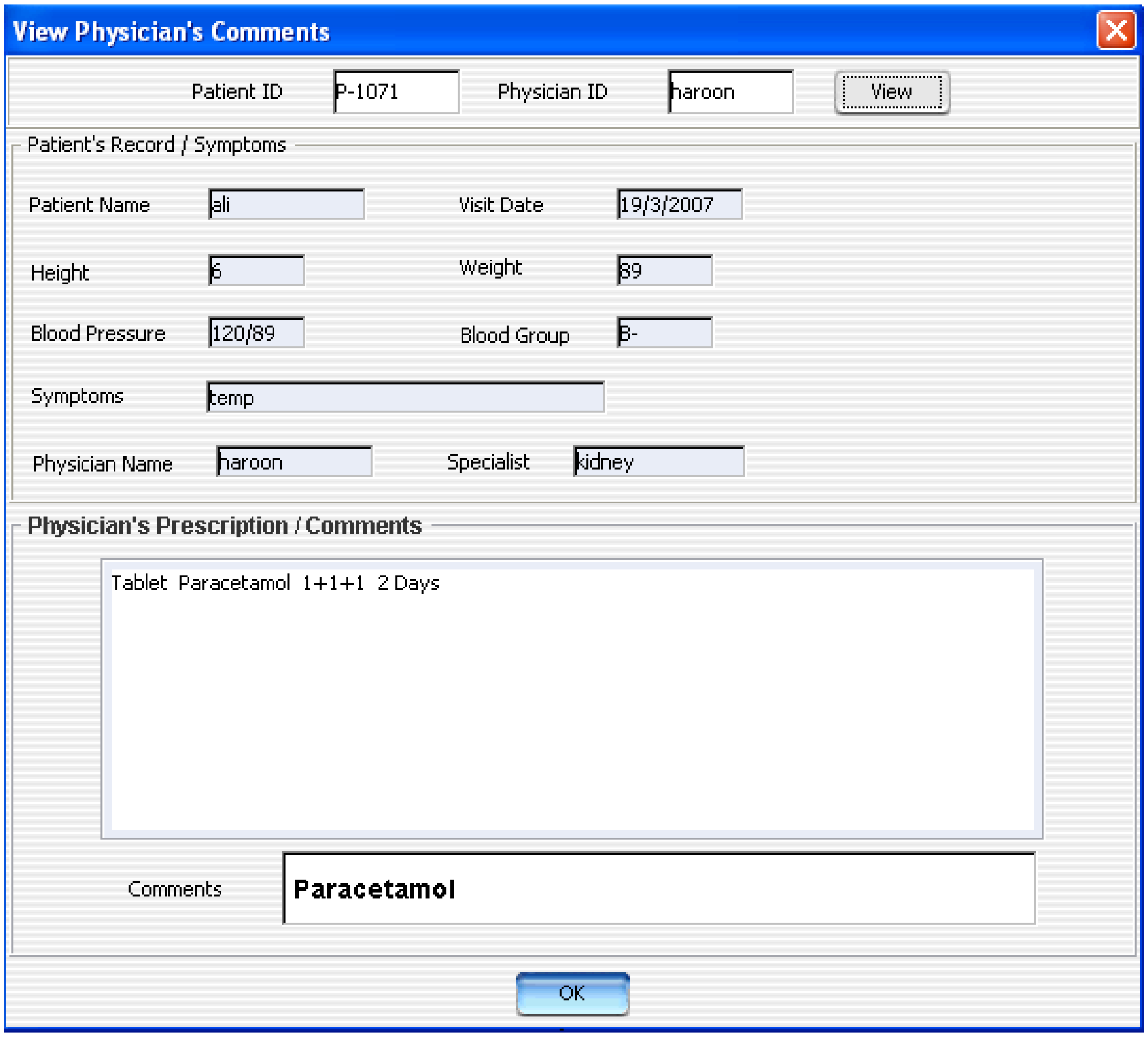


FILENAME	TYPE
CONFIG.SYS	File
_RESTORE	Directory
CLASSES.1ST	File
MSDOS.---	File
SETUPLOG.TXT	File
WINDOWS	Directory
MSDOS.SYS	File
SUHDLOG.DAT	File
NETLOG.TXT	File
AUTOEXEC.BAK	File
DETLOG.TXT	File
SYSTEM.1ST	File
CONFIG.BAK	File
AUTOEXEC.BAT	File
SETUPXLG.TXT	File
My Documents	Directory
BOOTLOG.PRV	File
io.sys	File
Program Files	Directory
command.com	File
inf.log	File
setup.log	File
Log.txt	File
My Medias	Directory

FILENAME	TYPE
.	Directory

Figure 12

View Physician’s Comments Interface



Patient ID P-1071 Physician ID haroon

Patient's Record / Symptoms

Patient Name	ali	Visit Date	19/3/2007
Height	6	Weight	89
Blood Pressure	120/89	Blood Group	B-
Symptoms	temp		
Physician Name	haroon	Specialist	kidney

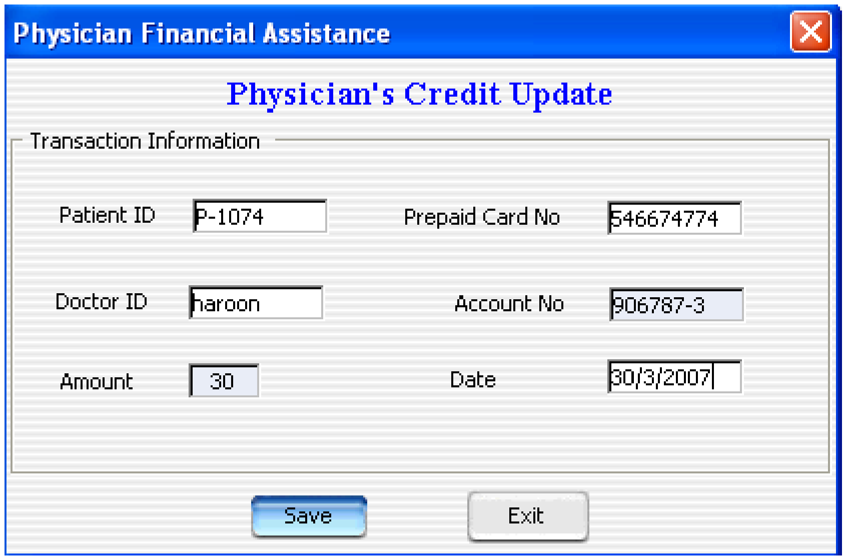
Physician's Prescription / Comments

Tablet Paracetamol 1+1+1 2 Days

Comments **Paracetamol**

Figure 13

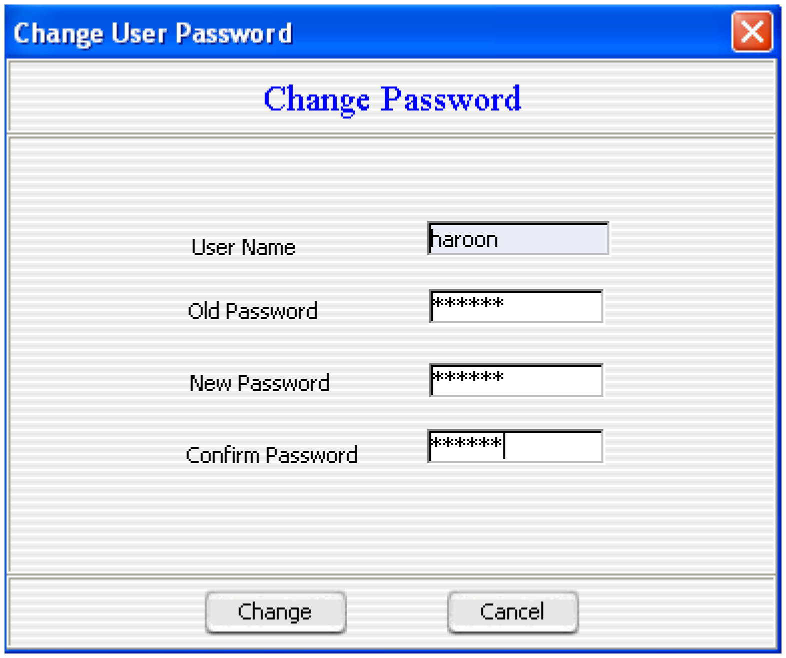
Physician's Credit Update Interface



Patient ID	P-1074	Prepaid Card No	546674774
Doctor ID	haroon	Account No	906787-3
Amount	30	Date	30/3/2007

Figure 14

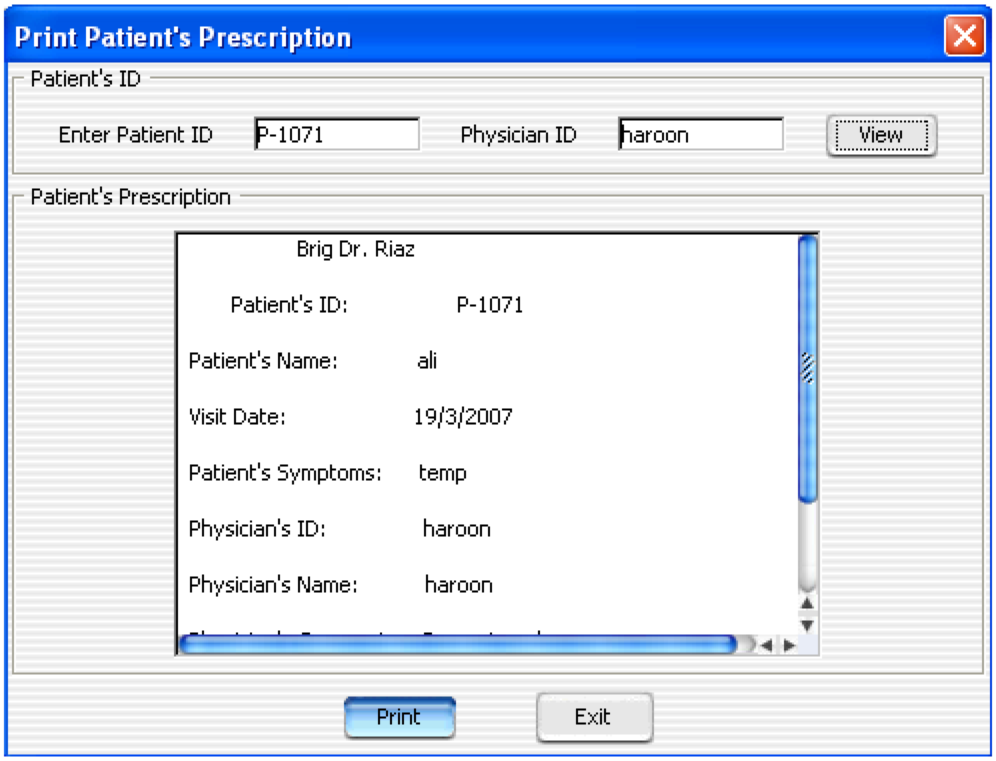
Change Password Interface



User Name	haroon
Old Password	******
New Password	******
Confirm Password	******

Figure 15

Print Patient’s Prescription



Enter Patient ID P-1071 Physician ID haroon

Brig Dr. Riaz

Patient's ID: P-1071

Patient's Name: ali

Visit Date: 19/3/2007

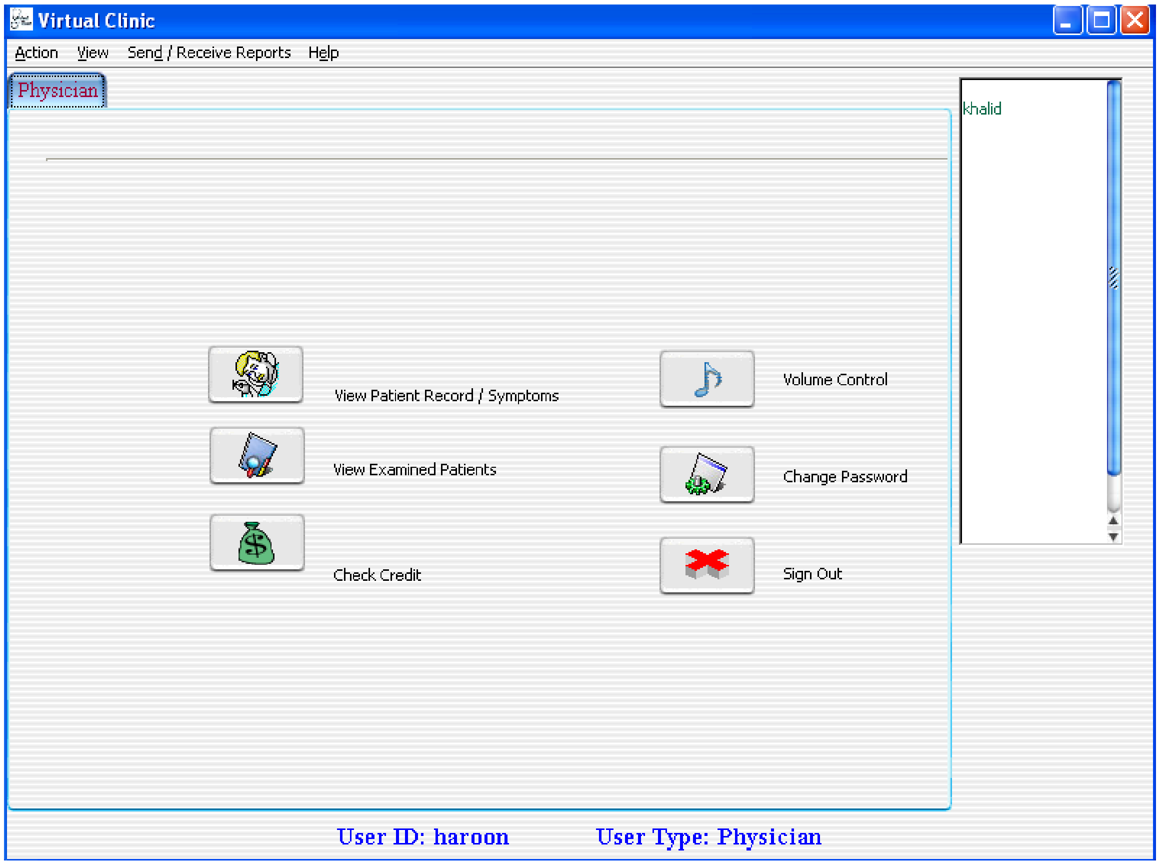
Patient's Symptoms: temp

Physician's ID: haroon

Physician's Name: haroon

Figure 16

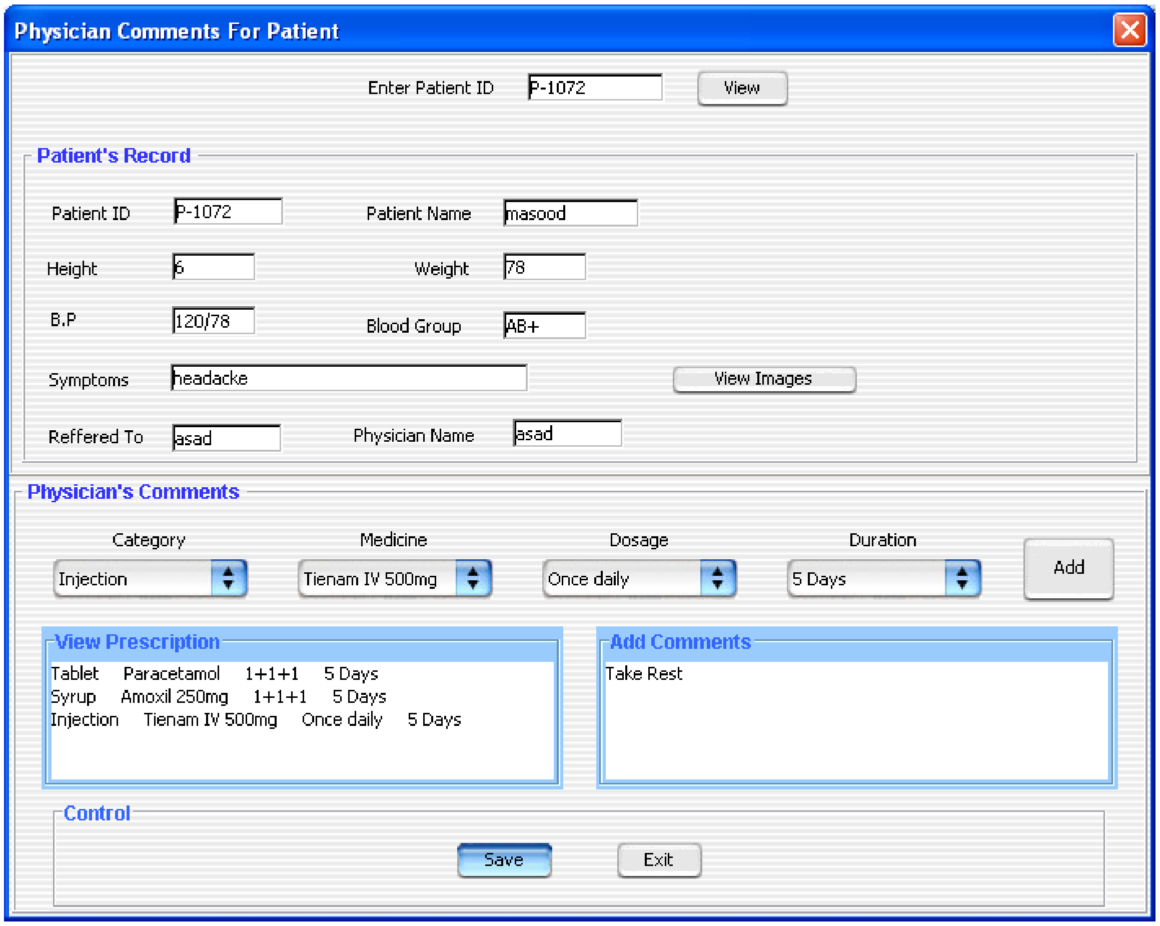
Physician’s Interface



User ID: haroon User Type: Physician

Figure 17

Physician's Comments for Patient Interface



Physician Comments For Patient

Enter Patient ID: P-1072

Patient's Record

Patient ID: P-1072
Patient Name: masood
Height: 6
Weight: 78
B.P: 120/78
Blood Group: AB+
Symptoms: headacke
Reffered To: asad
Physician Name: asad

Physician's Comments

Category: Injection
Medicine: Tienam IV 500mg
Dosage: Once daily
Duration: 5 Days

View Prescription

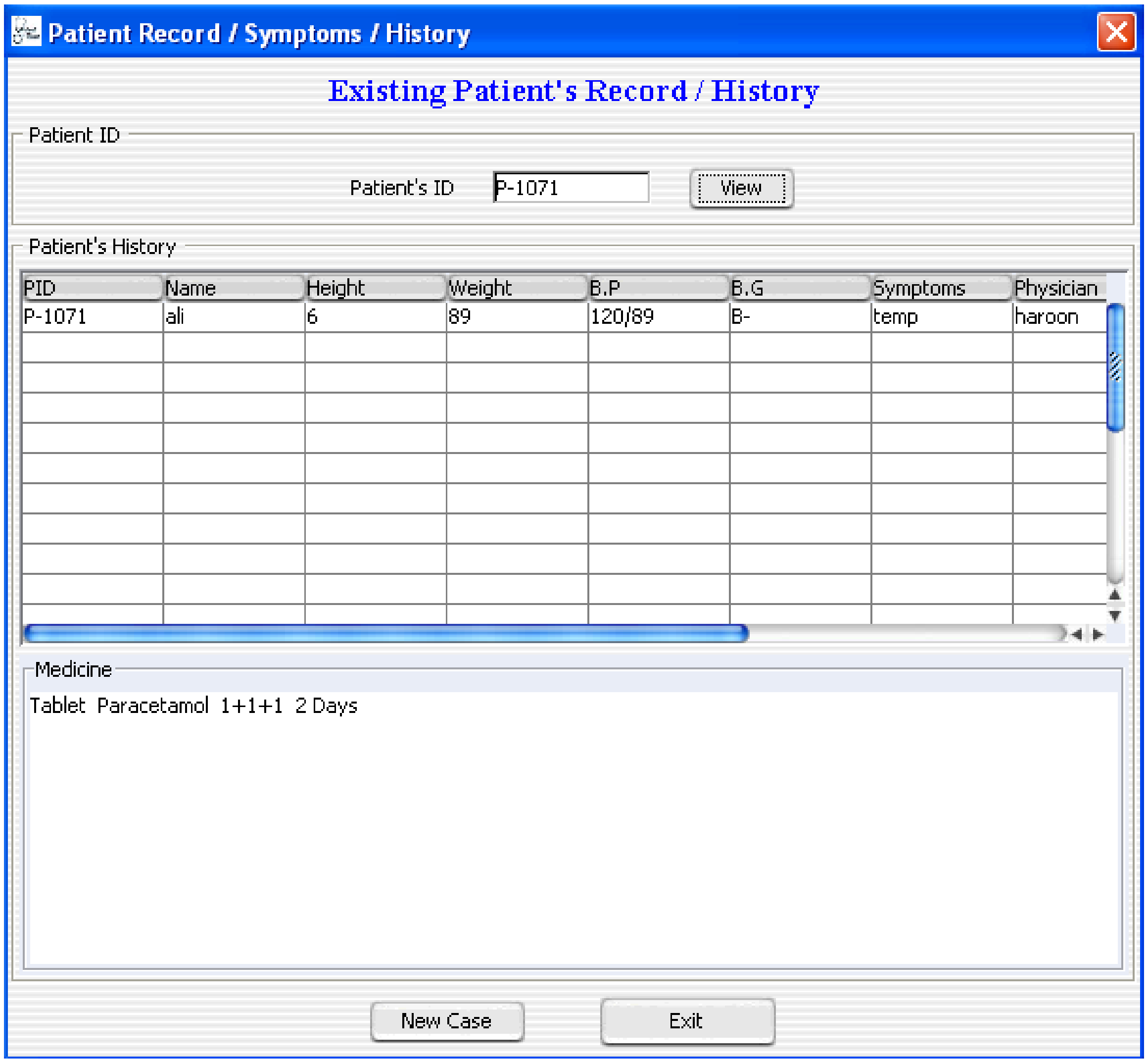
```
Tablet     Paracetamol     1+1+1     5 Days
Syrup      Amoxil 250mg     1+1+1     5 Days
Injection     Tienam IV 500mg     Once daily     5 Days
```

Add Comments

Take Rest

Figure 18

Patient’s Record / Symptoms / History Interface



Patient Record / Symptoms / History

Existing Patient's Record / History

Patient ID

Patient's ID: P-1071

Patient's History

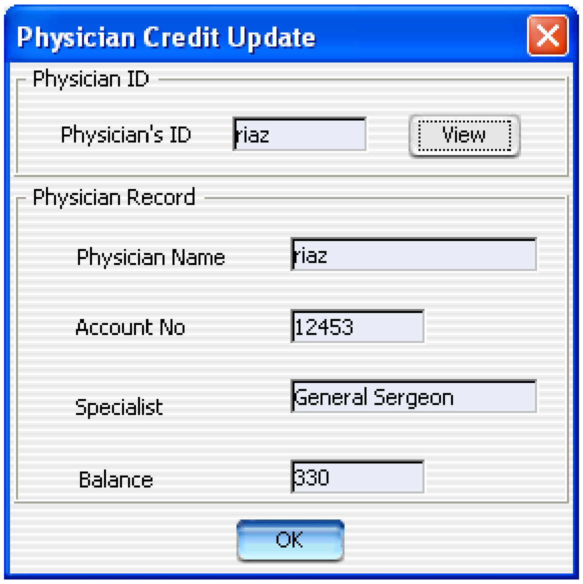
PID	Name	Height	Weight	B.P	B.G	Symptoms	Physician
P-1071	ali	6	89	120/89	B-	temp	haroon

Medicine

Tablet Paracetamol 1+1+1 2 Days

Figure 19

Physician’s Financial Assistance Interface



Physician's ID	riaz
Physician Name	riaz
Account No	12453
Specialist	General Sergeon
Balance	330

Figure20

Anx 'D'

1. Java 2 Platform, Standard Edition 1.5 (J2SE)

The Java 2 Platform, Standard Edition (J2SE) 1.5 has introduced several new features and enhancements for Object Serialization, Remote Method Invocation (RMI), Common Object Request Broker Architecture (CORBA), and networking in general. The new features and enhancements either address some changes in standard specifications such as CORBA, or improve the performance of existing features. This article gives an overview of the new features and enhancements.

2. Overview of New Features and Enhancements

New features and enhancements have been introduced for RMI, serialization, and CORBA as discussed in the following sections.

2.1 RMI

The RMI enhancements in J2SE 1.5 are:

- Server-side Stack Traces Retained in Remote Exceptions
- The RMI runtime implementation will now preserve the server-side stack trace information of an exception that is thrown from a remote call. In other words, when such an exception becomes accessible to client code, its stack trace will now contain all of its original server-side trace data followed by the client-side trace. Certain RMI server applications, however, may wish to prevent any server-side stack trace data

from accompanying an exception to be marshalled as a result of a remote call, perhaps for reasons of performance or confidentiality.

- Service Provider Interface for `RMIClassLoader`
- Certain static methods of `java.rmi.server.RMICLassLoader` can now delegate their behavior to an instance of a new service provider interface `java.rmi.server.RMIClassLoaderSpi`.

2.2 Dynamic Server Host Name

- The property `java.rmi.server.hostname` is now dynamically updated to indicate that future exports should use a new host name.
- RMI Implementation Logging
- The new implementation of RMI in J2SE 1.5 uses the new **Java Logging APIs** to provide implementation-specific logging output. Please see **RMI Implementation Logging** for more information.

2.3 Serializing Primitive Class Objects

In earlier releases, a `ClassNotFoundException` would be thrown when passing a `Class` object to an RMI class. Now, if an object that contains `Class` objects for primitive types is passed as a remote method parameter or return value, it is stored in an instance of `java.rmi.MarshalledObject`.

2.4 Serialization

There are several enhancements and bug fixes to object serialization in J2SE 1.5. The enhancements are:

- De-Serialization of unshared objects

In previous releases, security-conscious programmers had to clone private internal objects after de-serializing them to guard against the possibility that outside parties with access to the serialization stream could append spurious back handles to the sensitive objects, which results in extra references to them during de-serialization. This solution slows performance and wastes memory. In J2SE 1.4, a more efficient solution is provided by allowing receivers to invalidate *unshared* objects as they are read in, which makes it impossible for third parties to dereference back-handles to those objects occurring in the stream.

3. New Features and Enhancements in Networking

3.1 IPv6

Internet Protocol version 6 support for TCP and UDP application (including multicast) is now included in J2SE 1.4.

3.2 HTTP Digest Authentication

The HTTP digest authentication implementation has been updated to support proxies and origin servers.

3.3 Unconnected/Unbound Socket

This allows more flexible socket creation, binding, and connection. It enables manipulation of socket options before establishing or accepting connections. In addition a timeout can be specified when establishing a connection. A new class `javax.net.ssl.SSLSocket` which is a subclass of `java.net.Socket` has been added to provide security for data sent through sockets through encryption.

3.4 Connected UDP Socket

The UDP protocol is a connectionless protocol; however, the `DatagramSocket.connect` method now establishes the address association at native level. Where supported, this allows an application to have visibility of ICMP port unreachable messages as an indication that the remote application is unavailable.

3.5 Uniform Resource Identifier (URI)

The `java.net.URI` is new a class that allows URI construction and parsing without the presence of a protocol handler, which is not possible with the `URL` class.

3.6 JNDI DNS Service Provider in InetAddress

This enhancement in class `java.net.InetAddress` enables applications to configure a pure Java name service provider by using a DNS name service provider through JNDI.

3.7 URLEncoder and URLDecoder

These have been added to enable applications to use other character encoding/decoding schemes.

3.8 TCP Out-of-Band Data

New methods in class `java.net.Socket` have been added to provide limited support for TCP urgent data for support certain legacy applications. Urgent data may be sent on any TCP socket. However, only partial support for receiving urgent data is provided.

3.9 Sockets

Full V5 and V4 TCP support with auto negotiation with the proxy of which version to use.

Anx 'E'

1. Java I/O Streams

Most programs use data in one form or another, whether it is as input, output, or both. The sources of input and output can vary between a local file, a socket on the network, a database, variables in memory, or another program. Even the type of data can vary between objects, characters, multimedia, and others.

The **Java Development Kit** (JDK) provides APIs for reading and writing streams of data. These APIs have been part of the core JDK since version 1.0, but are often overshadowed by the more well-known APIs, such as **JavaBeans**, **JFC**, **RMI**, **JDBC**, and so on. However, input and output streams are the backbone of the JDK APIs, and understanding them is not only crucial, but can also make programming with them a lot of fun.

This article covers the fundamentals of Java streams by reviewing the differences between byte and character streams, peruses the various stream classes available in the `java.io` package, and looks at the concept of *stream chaining*.

2. Overview

To bring data into a program, a Java program opens a stream to a data source, such as a file or remote socket, and reads the information serially. On the flip side, a program can open a stream to a data source and write to it in a serial fashion. Whether you are reading from a file or from a socket, the concept of serially reading from, and writing to different data sources is the same. For that very reason, once you understand the top level classes (`java.io.Reader`, `java.io.Writer`), the remaining classes are straightforward to work with.

3. Character Streams versus Byte Streams

Prior to JDK 1.1, the input and output classes (mostly found in the `java.io` package) only supported 8-bit *byte* streams. The concept of 16-bit Unicode *character* streams was introduced in JDK 1.1. While byte streams were supported via the `java.io.InputStream` and `java.io.OutputStream` classes and their subclasses, character streams are implemented by the `java.io.Reader` and `java.io.Writer` classes and their subclasses.

Most of the functionality available for byte streams is also provided for character streams. The methods for character streams generally accept parameters of data type *char* parameters; while *byte* streams, you guessed it, work with *byte* data types. The names of the methods in both sets of classes are almost identical except for the suffix, that is, character-stream classes end with the suffix `Reader` or `Writer` and byte-stream classes end with the suffix `InputStream` and `OutputStream`. For example, to read files using character streams, we will use the `java.io.FileReader` class; for reading it using byte streams you would use `java.io.FileInputStream`.

Unless we are working with binary data, such as image and sound files, we should use readers and writers (character streams) to read and write information for the following reasons:

They can handle any character in the Unicode character set (while the byte streams are limited to ISO-Latin-1 8-bit bytes).

They are easier to internationalize because they are not dependent upon a specific character encoding.

They use buffering techniques internally and are therefore potentially much more efficient than byte streams.

4. Bridging the Gap between Byte and Character Streams

To bridge the gap between the byte and character stream classes, JDK 1.1 and JDK 1.2 provide the `java.io.InputStreamReader` and `java.io.OutputStreamWriter` classes. The only purpose of these classes is to convert byte data into character-based data according to a specified (or the platform default) encoding. For example, the static data member "`in`" in the "`System`" class is essentially a handle to the `Standard Input` (stdin) device. If we want to *wrap* this inside the `java.io.BufferedReader` class that works with character-streams, you use `InputStreamReader` class as follows:

```
BufferedReader in = new BufferedReader (new
InputStreamReader (System.in));
```

5. Various Stream Classes

As you might have guessed, the `java.io` package contains the Java I/O stream classes. These classes are either the *top level* abstract classes or the *specialized descendant* implementation classes, both types are described below.

6. Top Level Classes: `java.io.Reader` and `java.io.Writer`

`Reader` and **`Writer`** are the abstract parent classes for character-stream based classes in the **`java.io`** package. As discussed above, `Reader` classes are used to read 16-bit character streams and `Writer` classes are used to write to 16-bit character streams.

7. Other Notable Methods

Some other notable methods in the top-level classes include `skip(int)`, `mark(int)`, `reset()`, `available()`, `ready()` and `flush()`, these are described below.

- **`skip()`** as the name implies, allows you to skip over characters.
- **`mark()`** and **`reset()`** provide a book-marking feature that allows you to read ahead in a stream to inspect the upcoming data but not necessarily process it. Not all streams support "marking". To determine if a stream supports marking, use the `markSupported()` method.
- **`InputStream.available()`** tells you how many bytes are available to be read before the next `read()` will block. **`Reader.ready()`** is similar to the `available()` method, except it does not indicate how many characters are available.
- The **`flush()`** method simply writes out any buffered characters (or bytes) to the destination (for example, file, or socket).

Specialized Descendent Stream Classes

There are several specialized stream classes that subclass from the Reader and Writer classes to provide additional functionality. For example, the **`BufferedReader`** not only provides buffered reading for efficiency but also provides methods such as "readLine()" to read a line of input.

The following class hierarchy shows a few of the specialized classes found in the `java.io` package:

- Reader
- BufferedReader
- LineNumberReader
- FilterReader
- PushbackReader
- InputStreamReader
- FileReader
- StringReader

The above hierarchy simply demonstrates how stream classes extend their parent classes (for example, `LineNumberReader`) to add more specialized functionality.

The following three tables provide a more comprehensive list of the various descendent classes found in the `java.io` and other packages, along with a brief description for each class. These descendent classes are divided into two categories: those that read from, or write to *data sinks*, and those that perform some sort of processing on the data—this distinction is merely to group the classes into two logical sections, you do not have to know one way or the other when using them.

CharArrayReader and CharArrayWriter	For reading from or writing to character buffers in memory
FileReader and FileWriter	For reading from or writing to files

PipedReader and PipedWriter	Used to forward the output of one thread as the input to another thread
StringReader and StringWriter	For reading from or writing to strings in memory

Table 1. Data Sink Streams

Table 2. Processing Streams	
BufferedReader and BufferedWriter	For buffered reading/writing to reduce disk/network access for more efficiency
InputStreamReader and OutputStreamWriter	Provide a bridge between byte and character streams.
SequenceInputStream	Concatenates multiple input streams.
ObjectInputStream and ObjectOutputStream	Use for object serialization.
DataInputStream and DataOutputStream	For reading/writing Java native data types.
LineNumberReader	For reading while keep tracking of the line number.
PushbackReader	Allows to "peek" ahead in a stream by one character.

Table 2. Processing Streams

Table 3. Miscellaneous Streams (java.util.zip package)	
CheckedInputStream and CheckedOutputStream	For reading/writing and maintaining a checksum for verifying the integrity of the data.
GZIPInputStream and GZIPOutputStream	For reading/writing data using GZIP compression/decompression scheme.
ZipInputStream and ZipOutputStream	For reading/writing ZIP archive files.

Table 3. Miscellaneous Streams (java.util.zip package)

Anx 'F'

1. Java Database Connectivity [JDBC]

"**JDBC** is the most useful tool in the **JDK**, and the least understood". The **JDBC** is a simple set of classes for accessing relational databases from Java with a class library that mirrors the **ODBC** from Microsoft, has been ignored because database access is not new when compared to RMI and the new event model. It is also understood that many vendors, rather than waiting for JDBC to be an official part of the 1.1 release, have come up with JDBC-compliant APIs.

Keep in mind that JDBC database access is not limited to some of the larger vendors of database servers, such as Informix, Oracle, Sybase, and Microsoft. Database access from JDBC equally applies to smaller desktop database systems, such as **xBase files**, **FoxPro**, **MS Access**, and **SQL**. JDBC, through ODBC, even works with text files, and Excel spreadsheets.

Within JDBC there are four particularly important classes: **DriverManager**, **Connection**, **PreparedStatement**, and **ResultSet**. Each class corresponds to an indispensable phase of database access:

The **DriverManager** loads and configures your database driver on your client.

The **Connection** class performs connection and authentication to your database server using URLs.

PreparedStatement moves SQL to the database engine for preprocessing and eventually execution.

The **ResultSet** class allows for the inspection of results from "select" statements. Each class relies on its previous class for instance creation. For example, an instance of Connection cannot exist without being created from the DriverManager.

JDBC can work with a great number of drivers, there's a class called the DriverManager to load drivers into your environment. Generally, you will only need the driver manager once, in the beginning of the main or init section of your application or applet, respectively.

There's only one method that you really need to know for the DriverManager, and that's **getConnection**. Connection objects are created from the class DriverManager. **Connection** objects are driven by **URLs**.

The **DriverManager** is very politically correct about how it finds the right driver for the URL specified in **getConnection**. It's not dictatorial but rather operates by voluntary delegation, where JDBC visits each driver registered with the DriverManager and asks, "Will you connect to this URL?" If the answer is no, the next driver is asked and the process continues, otherwise the driver attempts to contact the URL.

Retrieving the information is a fairly straightforward process of fetching the row and retrieving the row values column by column:

Retrieving values out of the database is simply shorthand for a typed accessor method such as **getString**, **getDouble**, or **getInteger**. Second, there are two forms of those accessor methods. One method retrieves values by column index starting at 1. The other method retrieves values by column name. The first method is much more efficient, more open to producing erroneous code.

PreparedStatement objects can also take in-parameters, which are parameters that can be passed into a SQL statement after the statement has been prepared on the database server and before the statement is executed.

Anx ‘G’

References

[1]. Java Network Programming, Second Edition

[2]. By Merlin Hughes, Michael Shoffner, Derek Hamner

[3]. Manning Publications Company; ISBN 188477749X

[4]. [JMF 2.1.1e] Java Media Frame Work

[5]. By sun Micro System

[6]. www.java.sun.com

[7]. http://java.sun.com/products/java-media/jmf/

[8]. [OODJ 1998] Bill McCarty, Stephen Gilbet

[9]. Object Oriented Design in Java

[10]. http://www.google.com

[11]. http://www.skinlf.l2fprod.com/

[12]. http://www.twain.org

[13]. http://www.mysql.com/products/connector/odbc/

[14]. http://www.mysql.com/

[15]. http://dev.mysql.com/downloads/connector/

[16]. http://www.navicat.com/

[17]. http://nixbit.com/cat/programming/widgets/skinlf/