VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY

K.R. Road, V.V.Puram, Bangalore-560 004

Department of Computer Science & Engineering

Dissertation on

VIRTUAL HOSPITAL

Submitted By

USN	Name
1BI10CS020	**Boruthalupula Maneesh**
1BI10CS036	**Jyothi Prasad N M**
1BI10CS104	**Srivathsa S Rao**
1BI10CS123	**Vinay Hegde**

for the academic year 2013-2014

Under the Guidance of

Prof. N THANUJA
Asst. Professor

Department of Computer Science & Engineering
Bangalore Institute of Technology
Bangalore-560004

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY

Bangalore-560 004

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that the Project work entitled **"VIRTUAL HOSPITAL"** has been successfully completed by

USN	Name
1BI10CS020	**Boruthalupula Maneesh**
1BI10CS036	**Jyothi Prasad N M**
1BI10CS104	**Srivathsa S Rao**
1BI10CS123	**Vinay Hegde**

students of VIII semester B.E. for the partial fulfillment of the requirements for the Bachelors Degree in Computer Science & Engineering of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY** during the academic year 2013-2014.

Prof. N Thanuja
Internal Guide
Assistant professor
Department of CS&E
Bangalore Institute of Technology
Bangalore - 560004.

Dr. S. NANDAGOPALAN
Professor and Head
Dept. of CS&E
Bangalore Institute of Technology
K. R. Road, V. V. Puram
Bangalore - 560004.

Examiners: **1.**

2.

ACKNOWLEDGEMENT

There have been a lot of people whose support made this project to be completed successfully and it is only appropriate to express our gratitude to them all.

Our first thanks goes to the University. We are grateful to the **Visveswaraya Technological University** for providing us with this opportunity to showcase our skill sets in the form of this project.

We thank **Dr. K. R. Suresh, Principal, BIT,** for providing us with excellent facilities and bringing us all together.

We thank **Dr. S. Nandagopalan, HOD, Department of Computer Science & Engineering,** for his valued co-operation in completion of this project.

We are thankful to our project coordinator, **Prof. S. K. Savitha**, and the entire **Department of Computer Science & Engineering,** for constant support and advice throughout the course of preparation of this project and the report.

Our heartfelt gratitude to our guide, **Prof. N. Thanuja**, **Department of Computer Science & Engineering,** for her constant support, advice and motivation. Thanks for being there for us.

Last, but not the least, we are deeply indebted to our beloved parents who stood by us at all times with their constant motivation and unflinching support. We also thank our friends for their constructive criticism and assistance.

Boruthalupula Maneesh

Jyothi Prasad N M

Srivathsa S Rao

Vinay Hegde

ABSTRACT

In a world with fast growing internet usage, social media and web services, virtualization to a great extent is imminent. Such circumstances encourage more and more services and applications to assist people in their day-to-day needs. Now, there is an application for any need, from searching hotels and restaurants to paying bills and more. This instills the usage of such a service for every basic need. One such basic need is that of medical health care. Here is where this project comes into the picture. Virtual hospital targets at catering for the basic need of healthcare, essentially the preliminary diagnosis for a potential patient.

This project aims at providing medical solutions and all other vital tasks which are done at a hospital online. In this a patient will have to create a profile with his/her choice of username. The patient has to log in using to their system and schedule appointments at their convenience. It also includes a “chat application” where the patient can interact with the corresponding doctor. Tasks like fixing the appointments of patients and advices of doctors can be done easily.

This project also includes a “patient reminder system” where in a patient is informed regarding their ensuing appointments via text messages. They are also informed about important advices or messages from their doctors. The patient reminder system is useful in notifying them about important activities to be carried out such as prescriptions.

CONTENTS

1. INTRODUCTION 1

1.1 Introduction to Virtual Hospital 1

1.2 Problem Statement 1

1.3 Features of Virtual Hospital 1

2. LITERATURE SURVEY 6

2.1 Existing Systems 6

2.1.1 Loop Holes of existing systems 8

2.2 Proposed System 8

2.2.1 Advantages 9

3. SYSTEM REQUIREMENTS 10

3.1 Client-side 10

3.1.1Hardware Requirements 10

3.1.2 Software Requirements 10

3.2 Developer-side 10

3.3 Constraints 11

4. ARCHITECTURE 12

4.1 Web Application 12

4.1.1 HTML5 13

4.1.2 CSS3 13

4.1.3 JavaScript 13

4.1.4 JQuery 13

4.1.5 JSP 14

4.1.6 Servlets 14

4.1.7 JDBC 15

4.1.8 MySQL 15

4.1.9 XML 16

4.2 Chat Application 17
4.2.1 Sockets 18
4.2.2 Multithreading 19
4.2.3 Java Swings 19

5. SYSTEM DESIGN 21
5.1 High Level Design 21
5.1.1 Block Diagram for various modules 21
5.1.2 Use Case Diagram 22
5.2 Low Level Designs 23
5.2.1 Module 1 – Doctor Registration/Login 23
5.2.2 Module 2 – Patient Registration/Login 24
5.2.3 Module 3 - Health Tips 25
5.2.4 Module 4 - Departments 25
5.2.5 Module 5 - Text Alerts Through SMS 26
5.2.6 Module 6 - Consultation 26
5.2.7 Module 7 - Contact Us 28
5.2.8 Module 8 - Prescription Generation 29
5.2.9 Module 9 - Appointments 29

6. IMPLEMENTATION 31
6.1 Overview 31
6.2 Chat Module 31
6.2.1 Code Snippet For SocketClient 32
6.2.2 Code Snippet For Messages 37
6.2.3 Code Snippet For Download 38
6.2.4 Code Snippet For Upload 39
6.2.5 Code Snippet For History 40
6.2.6 Code Snippet For SocketServer 42

6.3 Prescription Module 49
6.4 Appointment Module 61
6.5 SMS Module 70
6.5.1 Code Snippet For Formatting SMS Text 71
6.5.2 Code Snippet For Sending SMS 73
6.6 E-Mail Module 74

7. TESTS AND RESULTS 76

8. CONCLUSION AND FUTURE WORK 78
8.1 Conclusion 78
8.2 Future Work 78

9. BIBLIOGRAPHY 79

APPENDIX A. SCREENSHOTS 80

LIST OF FIGURES

1.1 Increase in internet usage in India 2

1.2 Graph depicting the increase in number of internet users in India 3

4.1 Architectural diagram of the web application 12

4.2 How JSP's are deployed onto the web browsers 14

4.3 Working of Servlets - interaction between JSP and Servlets 15

4.4 Building documents using XML 16

4.5 Client-Server model 17

4.6 Communication between client and server 18

4.7 Interactions in multithreading 19

4.8 Java Foundation Classes 20

5.1 Virtual Hospital – Block diagram 21

5.2 Use Case Diagram for Virtual Hospital 22

5.3 Flowchart for Doctor-Registration/Login 23

5.4 Flowchart for Patient-Registration/Login 24

5.5 Block diagram representing various modules in Health Tips 25

5.6 Block diagram representing various Departments 25

5.7 Working of Text alerts 26

5.8 Sequence Diagram of Chat Application 27

5.9 Flowchart for Contact Us module 28

5.10 Process of prescription generation 29

5.11 Slots provided by doctor for appointments 30

5.12 Slot booked by the patient for appointment with doctor 30

A.1 Home page – Part 1 80

A.2 Home page – Part 2 80

A.3 Health Tips page 81

A.4 Departments Page 81

A.5 Doctor's details in a particular department 82

A.6 About Us page 82

A.7 Contact Us page 83

A.8 Doctor’s registration page 83
A.9 Doctor’s profile page 84
A.10 Doctor providing his appointment slots 84
A.11 Doctor providing the prescription to his patient 85
A.12 Patient’s registration page 85
A.13 Patient’s profile page 86
A.14 Patient logging in to chat 86
A.15 Patient choosing his required appointment 87
A.16 Patient changing his consulting department 87
A.17 Patient’s / Doctor’s complaint page 88
A.18 Patient’s / Doctor’s ‘change password’ page 88

7.1 Test Cases for Virtual Hospital (Table) 76-78

INTRODUCTION

The concept of virtual hospital is well known in many developed countries. This mainly deals with providing constant and effective medical solutions to patients by experienced doctors via the internet without hassle. This will help the patient in saving his/her time and money. This however is not popular in India. The main reason for this being the lack of availability of internet to the common people and the lack of knowledge about computer usage in the rural areas. But these days internet has turned out to be a predominant element in India and many online consultancies are coming into existence in various fields. This is the right time to introduce effective online medical services in India.

1.1 Introduction to Virtual Hospital

Telemedicine is the use of telecommunication and information technologies in order to provide clinical health care at a distance. It helps eliminate distance barriers and can improve access to medical services that would often not be consistently available in distant rural communities. Virtual Hospital uses telemedicine to deliver medical care to the developing world. This project aims at providing medical solutions and all other vital tasks which are done at a hospital online i.e. to deliver the "hospital environment" online.

1.2 Problem Statement

This project aims at reaching healthcare services to distant locations to anyone who has access to an internet connection and also doctors and health facilities to expand their reach, beyond their offices.

1.3 Features of Virtual Hospital

The important features of the virtual hospital are described below:

- ***Online Consulting:*** Online consultations or e-consultations refer to an exchange between government (or a private organization) and citizens using the Internet. They are one form of online deliberation.

As the Internet gains popularity with the public for voicing opinion, citizen participation in policy development through cyberspace is changing the face of democracy. The rise of the Internet has given way to buzzwords such as e-democracy, referring to citizen participation in politics, government issues and policy development through electronic technologies and the Internet. Online consultation is an extension of these concepts. While this definition is framed in the Canadian context, other countries like the UK, Denmark, Scotland, and Australia can also be considered leaders in the field. These and many other countries, like India, are integrating online consultations and engagement using various methods and for a range of purposes.

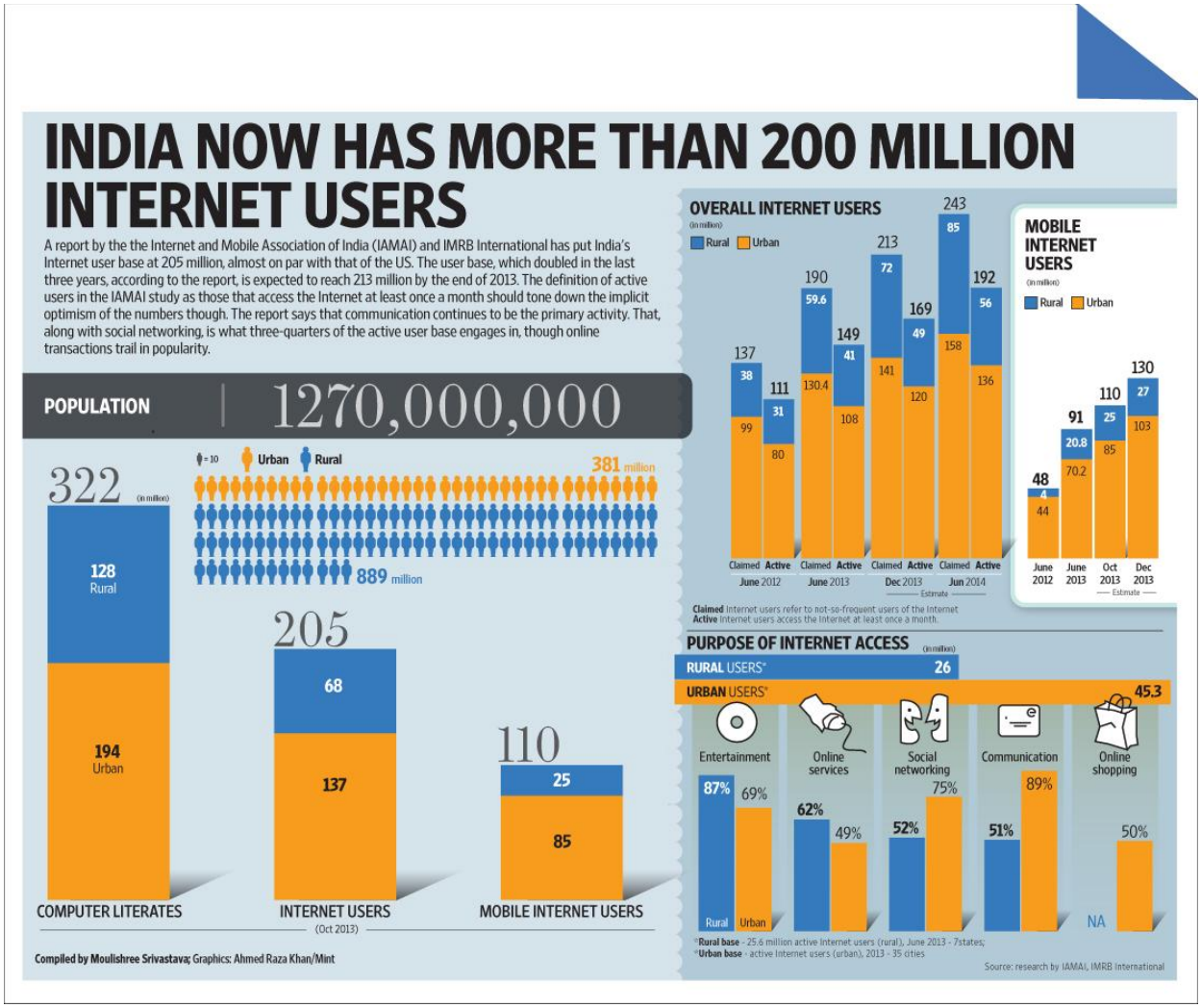

Fig 1.1 Increase in internet usage in India.

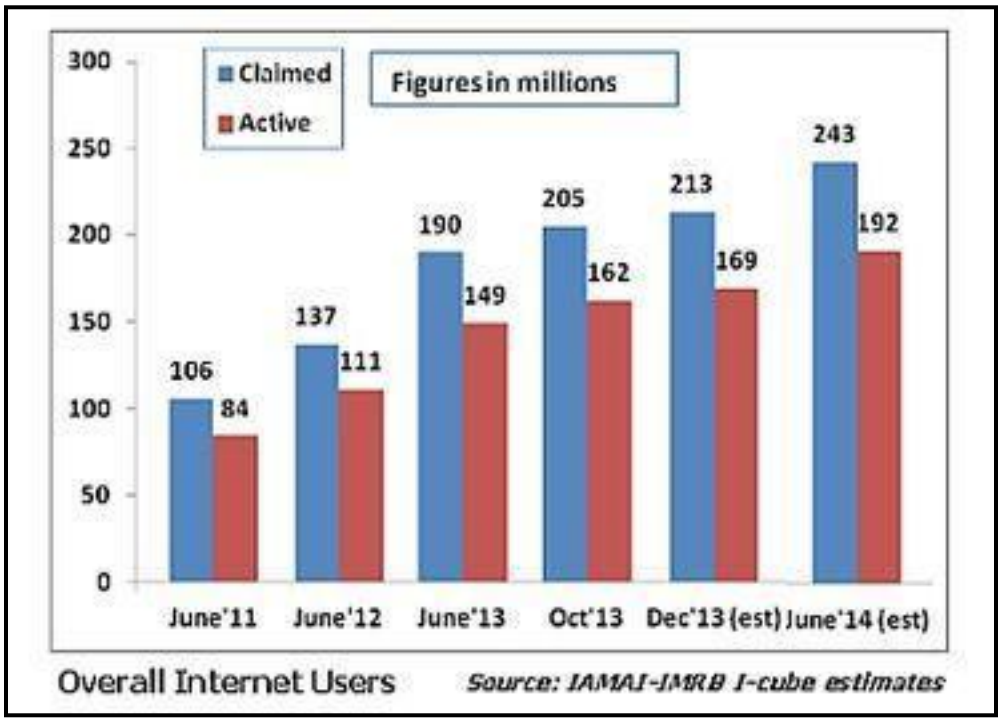

Fig 1.2 Graph depicting the increase in number of internet users in India

The concept of online consultation can be extended to this particular project also. Here, a patient will have to create a profile with a username of his choice that is available. The patient has to log in to their system and schedule appointments at their convenience. The chat application provides the means of communication between the doctor and the patient. The distance barrier between the patients and the doctors is eliminated by the online consultation which happens from home and hence it saves time and money to the patients.

- ***Preliminary Diagnosis:*** While not an official clinical term, the phrase "preliminary diagnosis" is sometimes used informally to refer to the diagnosis that a client receives after an intake interview. Psychological or cardiological disorders can be complex and difficult to accurately diagnose, but many insurance companies require an immediate diagnosis to pay for treatment. The preliminary diagnosis is often correct, but many therapists caution their clients that the diagnosis may change after further sessions. Here, on creating the profile, the patient communicates his symptoms to the doctor; the doctor analyses the

symptoms and diagnoses the patient's health problem and gives him a preliminary diagnosis of the illness that he might be suffering from.

- ***Health Tips:*** Just as any other online health service, our web application provides the patients with helpful information about the general health problems occurring and the ways to prevent them to lead a healthy life. The doctors would periodically post health tips that should be followed by the patients.

- ***Prescriptions:*** A prescription is a health-care programme that governs the plan of care for an individual patient and is implemented by a qualified practitioner. A qualified practitioner might be a physician, dentist, nurse practitioner, pharmacist, psychologist, or other health care providers. Prescriptions may include orders to be performed by a patient, caretaker, nurse, pharmacist, physician, other therapist, or by automated equipment, such as an intravenous infusion pump. Formerly, prescriptions often included detailed instructions regarding compounding of medications but as medications have increasingly become pre-packaged manufactured products, the term "prescription" now usually refers to an order that a pharmacist dispense and that a patient take certain medications. Prescriptions have legal implications, as they may indicate that the prescriber takes responsibility for the clinical care of the patient and in particular for monitoring efficacy and safety. As medical practice has become increasingly complex, the scope of meaning of the term "prescription" has broadened to also include clinical assessments, laboratory tests, and imaging studies relevant to optimizing the safety or efficacy of medical treatment.

  The doctors at Virtual Hospital, after a thorough analysis of the patient's symptoms would generate online prescriptions which would conform to the standard format of the prescriptions. This prescription would be emailed to the patient's instantly. The patient also has the option to download this prescription online after logging in to his account.

- ***Text Alerts:*** Text alerts are automatic messages delivered to your cell phone or other mobile device when you choose. They are designed to remind you of important events, alert you to something, or simply entertain. Text alerts are easy to sign up for and use to your advantage.

  Text alerts exist for all different purposes. Some of the most popular text message alerts are associated with your bank or credit card. They might alert you when a deposit, withdrawal, or payment occurs on your account, or when your credit card is used. If you are concerned about identity theft or credit card theft, mobile alerts can be a great way to keep track of your finances and stay secure.

  Here, the text alerts refer to a patient reminder system wherein, text messages are sent to the patients' cellular phone periodically reminding the patient about the medicines he/she has to take on a regular basis (medication course).

LITERATURE SURVEY

Literature survey is an important phase in the software development life cycle as we acquire the necessary knowledge and skills required to handle or develop a project. Hence, this chapter discusses about the existing system and the drawbacks of the existing system.

The Medical dictionary at www.thefreedictionary.com defines Virtual Hospital as "An evolving format for providing health care in a central location (hub) that relies on components of telemedicine and high-tech communications devices to connect it to a regional hospital, where a greater range of expertise is available." It is a repository of online information which answers patient questions—at the patient's level of sophistication—about diagnostic and therapeutic procedures that take place in the hospital.

The concept of virtual hospital is well known in many developed countries. This mainly deals with providing constant and effective medical solutions to patients by experienced doctors via the internet without hassle. This will help the patient in saving his/her time and money. This, however, is not popular in India and the main reason for this being the lack of availability of internet to the common people and the lack of knowledge about computers. But these days internet has turned out to be a predominant element in India and many online consultancies are coming into existence in various fields. This is the right time to introduce effective online medical services in India.

2.1 Existing Systems

The Virtual Hospital is a 21st century concept that is an evolving format which aims at providing many health-care services through a virtual environment or a web service. As mentioned earlier, this is well known in many developed countries, especially in England and The United States of America. Some of the Virtual Hospitals that are running successfully are introduced below.

The concept of virtual medicine is gaining a following, particularly in rural areas such as Cornwall, West Oxfordshire and in Kent, South East England. And there are projects such as the one that actually works at Kingston Hospital where young people can

go for sexual health checks and get the results texted back to them. That's something that would have been unheard of a few years ago.

There exists an international non-government organization, named Virtual Hospital that operates as part of Virtual Healthcare Limited. Virtual Hospital uses telemedicine to deliver medical care to the developing world. The work of Virtual Hospital is influenced to a great extent by reports published by the World Health Organization (WHO) on Telemedicine developments, American Telemedicine Association and the work of Zaidi and Denis Gilhooly, Principal Adviser in the United Nations.

According to WHO's report on 'Telemedicine - Opportunities and Developments in Member States': "Information and Communication Technologies (ICTs) have great potential to address some of the challenges met both developed and developing countries in providing accessible, cost effective high quality health care services. Telemedicine uses ICTs to overcome geographical barriers and increase access to healthcare services. This is particularly beneficial for rural and underserved communities in developing countries - groups that traditionally suffer from lack of access to healthcare."

The International Virtual e-Hospital (IVeH) Foundation is a non-profit organization, with its headquarters in Austin, Texas. IVeH was created to assist in rebuilding the public healthcare system in developing countries by introducing and implementing telemedicine, telehealth, and virtual educational programs through the concept of the IVeH Network. IVeH is managed solely by its volunteers and governed by an international Board of Directors.

The Virtual Naval Hospital is a digital library of military medicine and naval medicine and humanitarian medicine and disaster medicine. After 9 years of highly regarded service to 9 million users, on January 1, 2006, the US Navy ended its funding of the Virtual Naval Hospital digital library. Much of Virtual Naval Hospital's content remains available online, and as a service to its many patrons it continues to point to the original sources of the textbooks that it contained so that it may continue its mission of delivering expert medical information to providers and patients at the point-of-care in order to help providers take better care of their patients and help patients live healthier lives.

One such platform exists in India also. The Yourvirtualhospital.com is a telemedicine platform accessed through a computer with an internet connection that would bring together doctors and patients for medical advice, through online consultation. Headquartered at Calicut, India, Yourvirtualhospital.com is the brainchild of Dr. Suresh Kumar M.D., PhD, and an eminent doctor with more than 25 years of medical experience. They intend to be able to provide the patient an access to a large number of doctors who are experts in their field of specialization. In addition patients would have access to an online pharmacy, an online health store and life style education programs conducted by experts including weight loss programs.

2.1.1 Loop Holes of existing systems

- The Kingston Hospital initiated online health services communicates only the results virtually.
- In the virtual Healthcare limited comprehensive health services are not provided.
- The International Virtual eHospital (IVeH) targets only the underprivileged rural areas to set up the required hardware and personnel and is not available for the general public.
- Virtual Naval Hospital is just a digital library and not real time health consultation is provided.
- Yourvirtualhospital.com had a complicated user interface and is no longer in usage.

2.2 Proposed System

Virtual Hospital is a type of telemedical solution which aims at improving a patient's clinical health status.

Our solution involves a health professional providing consultation with a patient or a specialist assisting the patient to improve the health condition. Doctors and Patients can interact through a chat application after the patient has fixed an appointment with a particular specialist. The patient would also be provided with prescription alerts through text messages to his/her cell phone. Patients can also gather lot of helpful information through the general health tips provided. This solution brings out a clear understanding of

the patient's problem since the profile created by the patient would be stored for a large period of time with updates to the profile whenever an interaction between the doctor and a patient takes place. Virtual Hospital would bring healthcare services to distant locations and also allows doctors and health facilities to expand their reach, beyond their offices.

2.2.1 Advantages

- Online consulting: the patient need not travel to the hospital or where the doctor is available. He can avail the services from wherever he is with just a computer and an internet connection.
- Long-term patient profiles: the patient details are stored for a long period of time so that he can consult again when there is a necessity.
- Prescription alerts: the patient is intimated about his daily medicines to be taken so that his health-care is not neglected.
- Health Tips: elemental tips regarding day-to-day health maintenance are given for any upcoming diseases or any general symptoms.

SYSTEM REQUIREMENTS

Requirement specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. System requirements are the more detailed description of the functionality to be provided. The system requirements states what the system does and not how the system is implemented. The requirement engineering process leads to the production of a requirement document which is the specification for the system.

3.1 Client-side

3.1.1Hardware Requirements

- Processor: Pentium IV and above
- RAM: 512 MB or more
- Memory: at least 40 MB of free Hard Disk space
- Cellular phone to receive text alerts

3.1.2 Software Requirements

- Operating System: Any *[Windows XP/Vista/7/8; Linux; OSX]*
- Web browser:All browsers having Java plug-in enabled and supporting HTML5.*[Google Chrome 11-33,Mozilla Firefox 4-28, Windows IE 10-11]*
- Java Runtime Environment (JRE): Version 7 or higher. JRE is required to run the chat application.
- E-mail account to receive mails of confirmation and prescription

3.2 Developer-side

- Netbeans IDE 7.0 or higher
- Java Runtime Environment(SE) 1.7
- Chrome/Firefox runtime plugin
- MySQL Server 5.6
- GlassFish Server 4.0
- iTextpdfAPI,joda Time API

3.3 Constraints

- The patient has to visit a real world hospital for tests and the test reports.
- The database can store 800-1000 patient profiles due to memory constraints.
- The License numbers of Doctors registering cannot be validated officially due to privacy issues.
- The consumer and the service provider must have access to an internet connection.
- The online payment for service subscription cannot be implemented.
- The doctor should set his working hours for the upcoming week every Saturday.
- The patient would not receive text alerts if DND (Do Not Disturb) is activated in his cellular phone.

ARCHITECTURE

4.1 Web Application

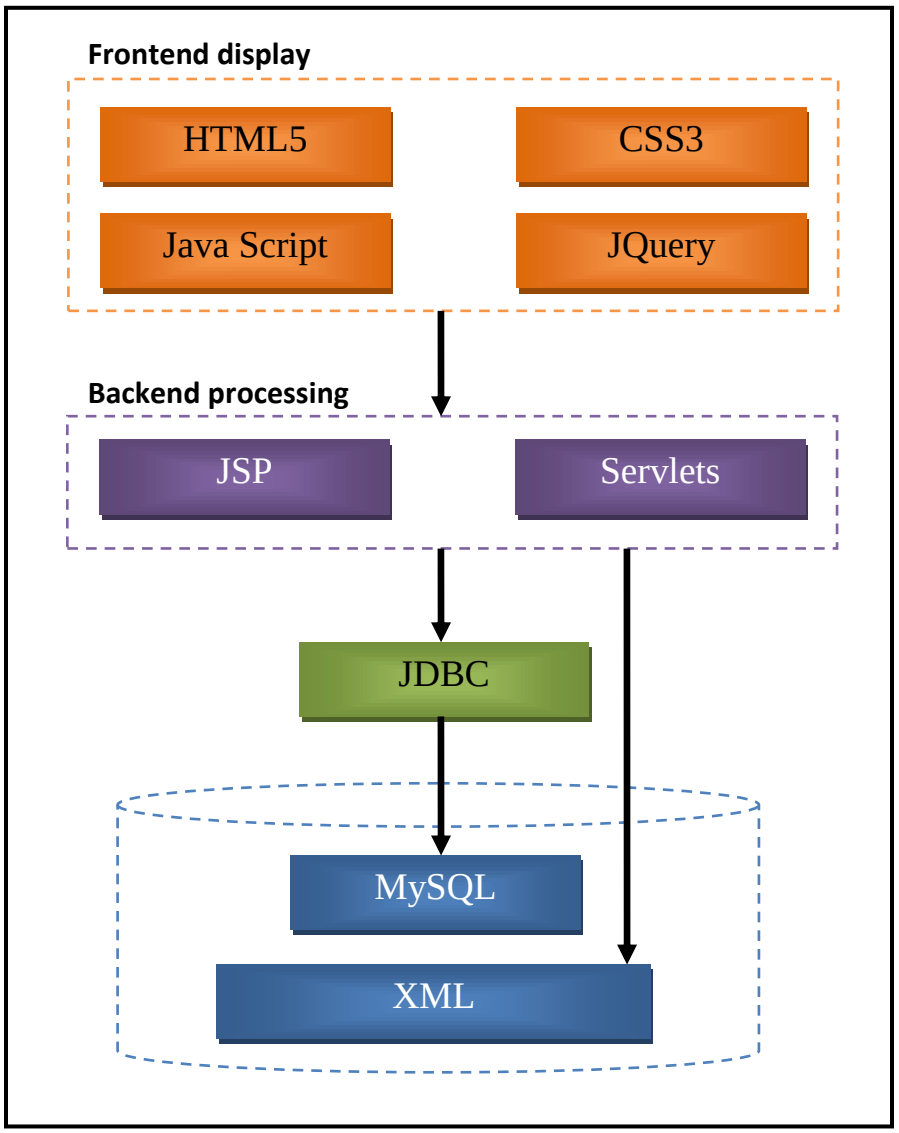

Fig 4.1 Architectural diagram of the web application

4.1.1 HTML5

HTML5 is a markup language used for structuring and presenting content for the World Wide Web and a core technology of the Internet. It is the fifth revision of the HTML standard (created in 1990 and standardized as HTML 4 as of 1997) and, as of December 2012, is a candidate recommendation of the World Wide Web Consortium (W3C). Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices (web browsers, parsers, etc.). HTML5 is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2 HTML.

4.1.2 CSS3

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.

4.1.3 JavaScript

JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side programming, game development and the creation of desktop and mobile applications.

4.1.4 JQuery

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It was released in January 2006 at BarCampNYC by John Resig. Used by over 80% of the 10,000 most visited websites, jQuery is the most popular JavaScript library in use today.

jQuery is free, open source software, licensed under the MIT License. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plugins on top of the JavaScript library.

4.1.5 JSP

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP, but it uses the Java programming language.

To deploy and run JavaServer Pages, a compatible web server with a servlet container, such as Apache Tomcat or Jetty, is required.

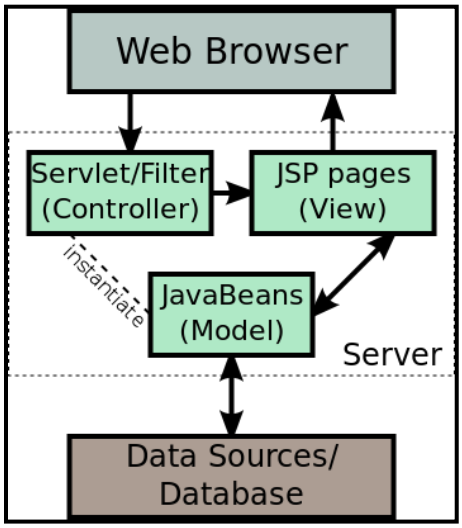

Fig 4.2 How JSP's are deployed onto the web browsers

4.1.6 Servlets

The servlet is a Java programming language class used to extend the capabilities of a server. Although servlets can respond to any types of requests, they are commonly used to extend the applications hosted by web servers, so they can be thought of as Java applets that run on servers instead of in web browsers. These kinds of servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

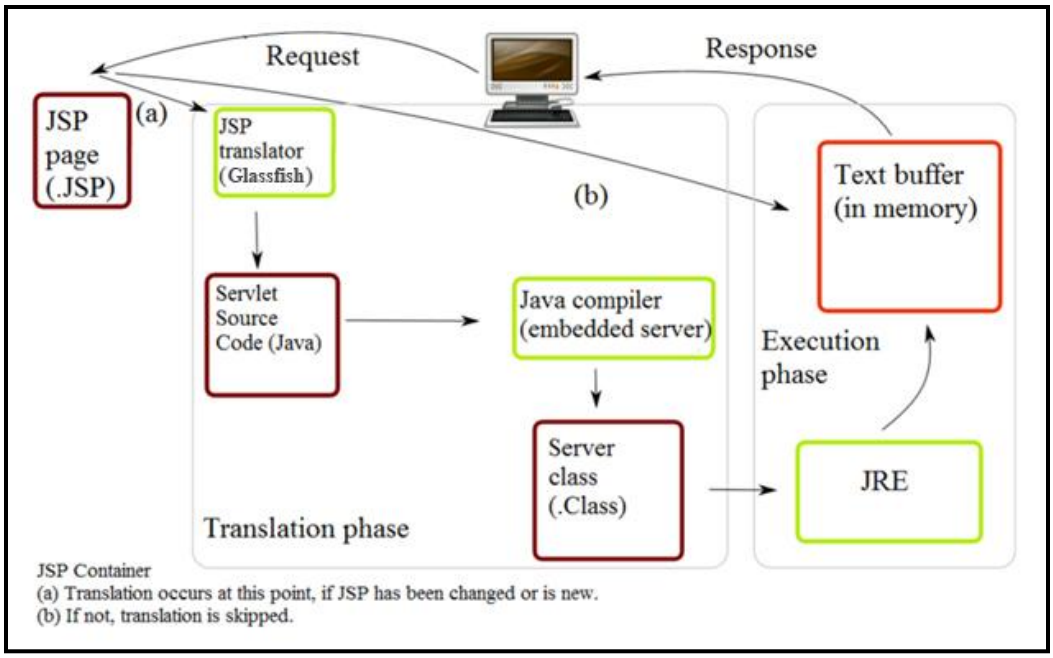

Fig 4.3 Working of Servlets - interaction between JSP and Servlets

4.1.7 JDBC

Java Database Connection is a Java-based data access technology (Java Standard Edition platform) from Oracle Corporation. This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the JVM host environment.

4.1.8 MySQL

MySQL is (as of March 2014) the world's second most widely used open-source relational database management system (RDBMS). It is named after co-founder Michael Widenius's daughter, My. The SQL phrase stands for Structured Query Language.

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation.

4.1.9 XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all free open standards.

The design goals of XML emphasize simplicity, generality, and usability over the Internet.[6] It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. Many application programming interfaces (APIs) have been developed to aid software developers with processing XML data, and several schema systems exist to aid in the definition of XML-based languages.

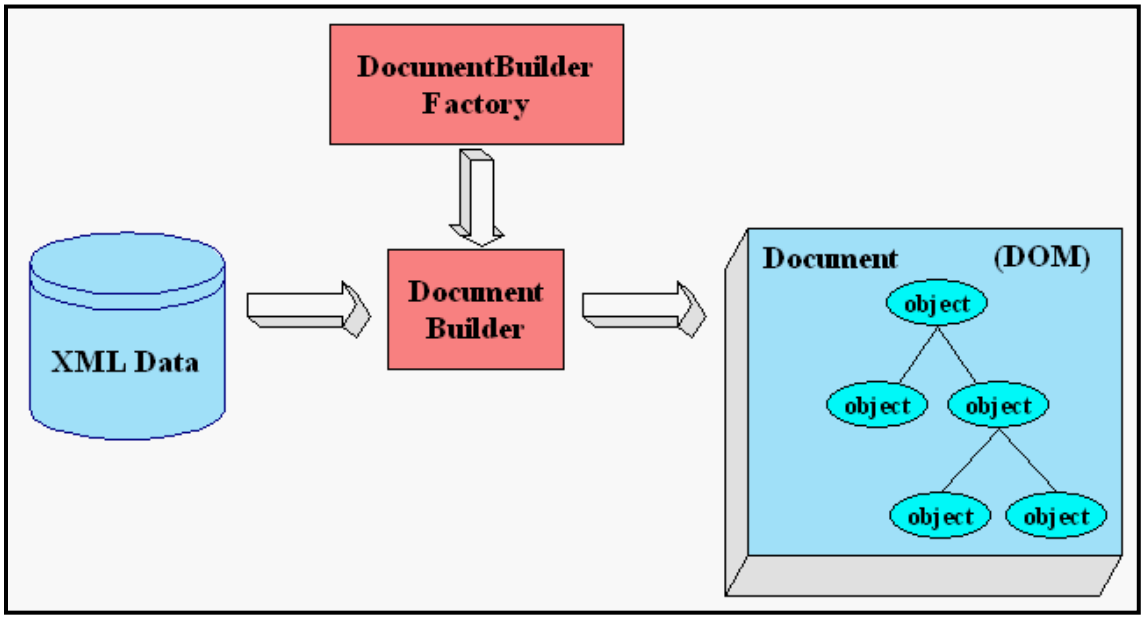

Fig 4.4 Building documents using XML

In this project, XML is mainly used to store information such as patient prescription which is used to generate the prescription pdf and text alerts.

4.2 Chat Application

The chat application is essentially based on the client-server architecture. Networks in which certain computers have special dedicated tasks, providing services to other computers (in the network) are called client-server networks.

The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

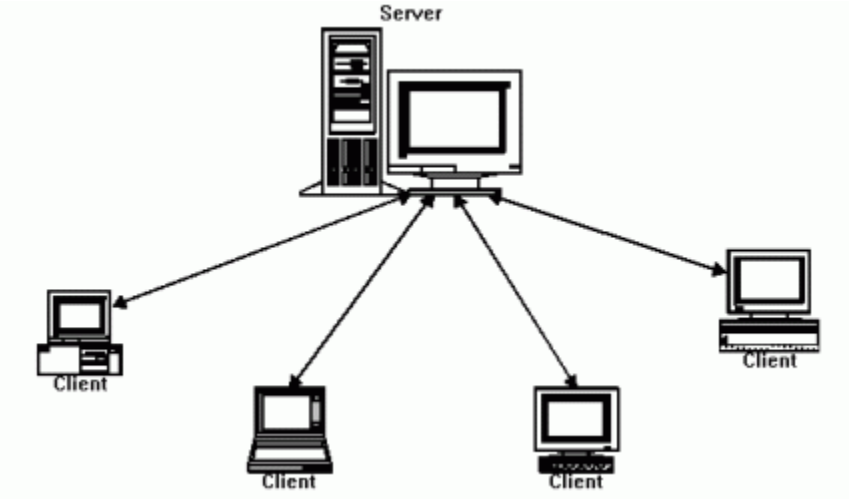

Fig 4.5 Client-Server model

The *client–server* characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services.

Servers are classified by the services they provide. For instance, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a *service*.

Whether a computer is a client, a server, or both, is determined by the nature of the application that requires the service functions. For example, a single computer can run web server and file server software at the same time to serve different data to clients making different kinds of requests. Client software can also communicate with server software within the same computer. Communication between servers, such as synchronizing data, is sometimes called *inter-server* or *server-to-server* communication.

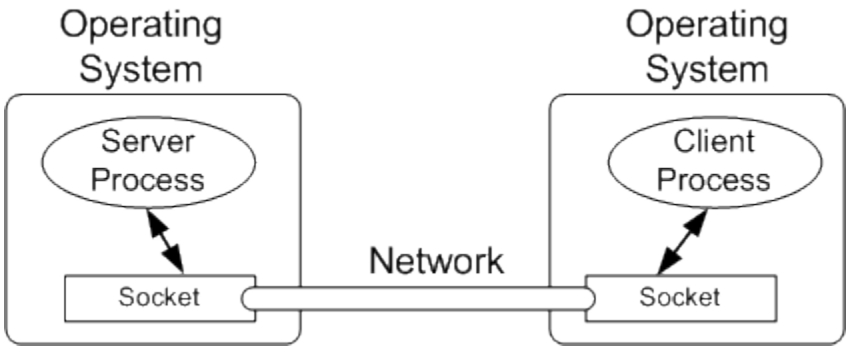

Fig 4.6 Communication between client and server

4.2.1 Sockets

A network socket is an endpoint of an inter-process communication flow across a computer network. Today, most communication between computers is based on the Internet Protocol; therefore most network sockets are Internet sockets. A socket API is an application programming interface (API), usually provided by the operating system, that allows application programs to control and use network sockets. Internet socket APIs are usually based on the Berkeley sockets standard. A socket address is the combination of an IP address and a port number, much like one end of a telephone connection is the combination of a phone number and a particular extension. Based on

this address, internet sockets deliver incoming data packets to the appropriate application process or thread.

4.2.2 Multithreading

Multithreading is the ability of a program or an operating system process to manage its use by more than one user at a time and to even manage multiple requests by the same user without having to have multiple copies of the programming running in the computer. Central processing units have hardware support to efficiently execute multiple threads. These are distinguished from multiprocessing systems (such as multi-core systems) in that the threads have to share the resources of a single core: the computing units, the CPU caches and the translation lookaside buffer (TLB),where multiprocessing systems include multiple complete processing units, multithreading aims to increase utilization of a single core by using thread-level as well as instruction-level parallelism. As the two techniques are complementary, they are sometimes combined in systems with multiple multithreading CPUs and in CPUs with multiple multithreading cores.

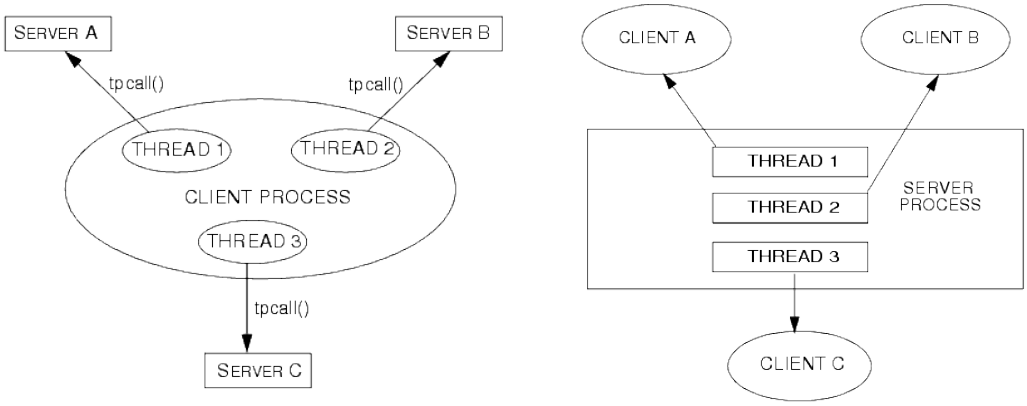

Fig 4.7 Interactions in multithreading

4.2.3 Java Swings

Swing is the primary Java GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs. Thus, the frontend of the chat application is built using Java Swings.

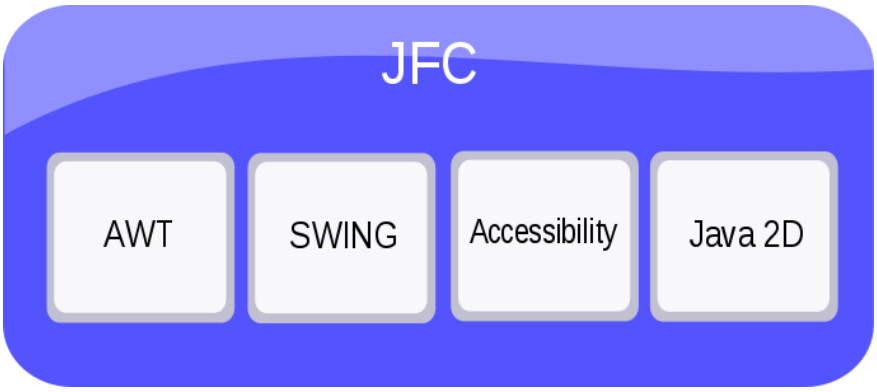

Fig 4.8 Java Foundation Classes

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

SYSTEM DESIGN

Software development usually follows what is known as Software Development Life Cycle. Typically a SDLC has four stages: Requirements, Design, Coding, and Testing. The second stage in SDLC is design stage.

The design stage involves two sub-stages namely:

- High-Level Design
- Low-Level Design

5.1 High Level Design

High-level design provides an overview of an entire system, identifying all its elements at some level of abstraction. This contrasts with Low level Design which exposes the detailed design of each of these elements.

5.1.1 Block Diagram for various modules

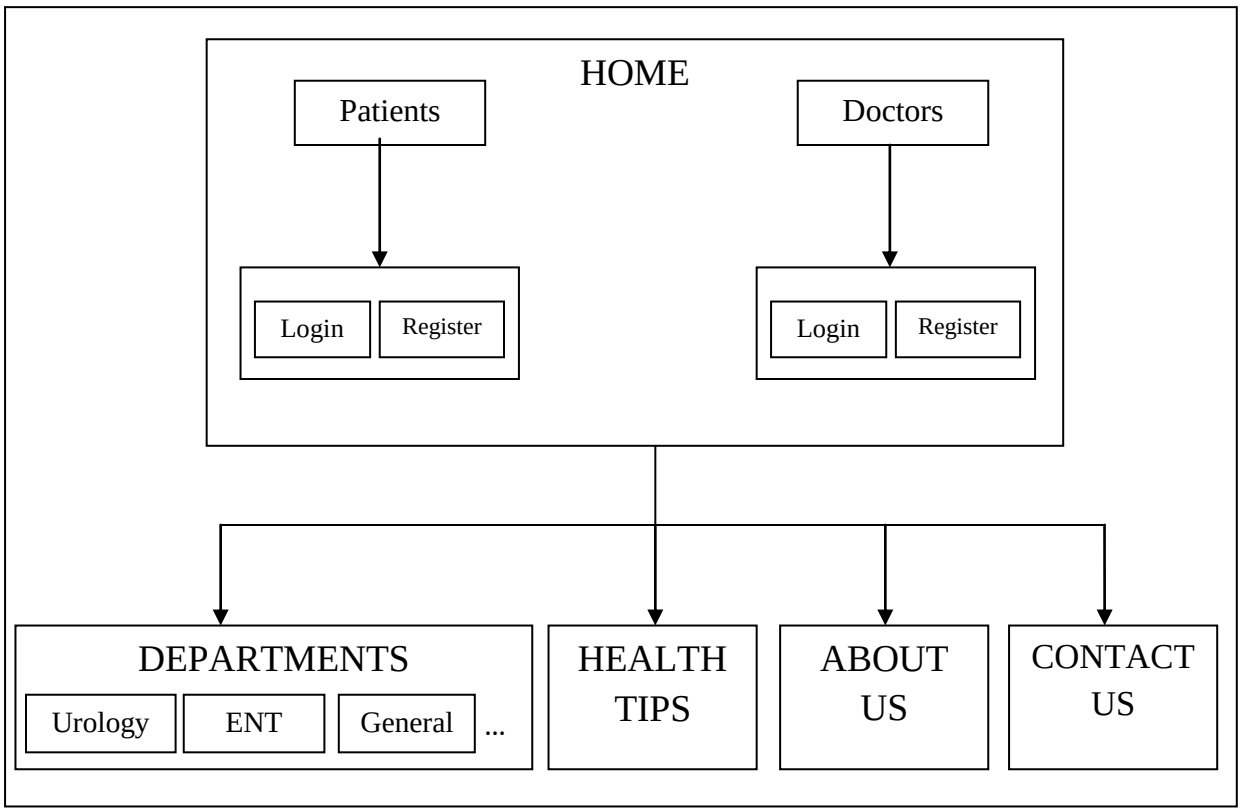

Fig 5.1 Virtual Hospital – Block diagram

The Virtual Hospital web-service has a home screen which has various modules namely Departments, Contact Us, Health Tips, About Us. It also has tabs for the doctor and patient login/registration.

5.1.2 Use Case Diagram

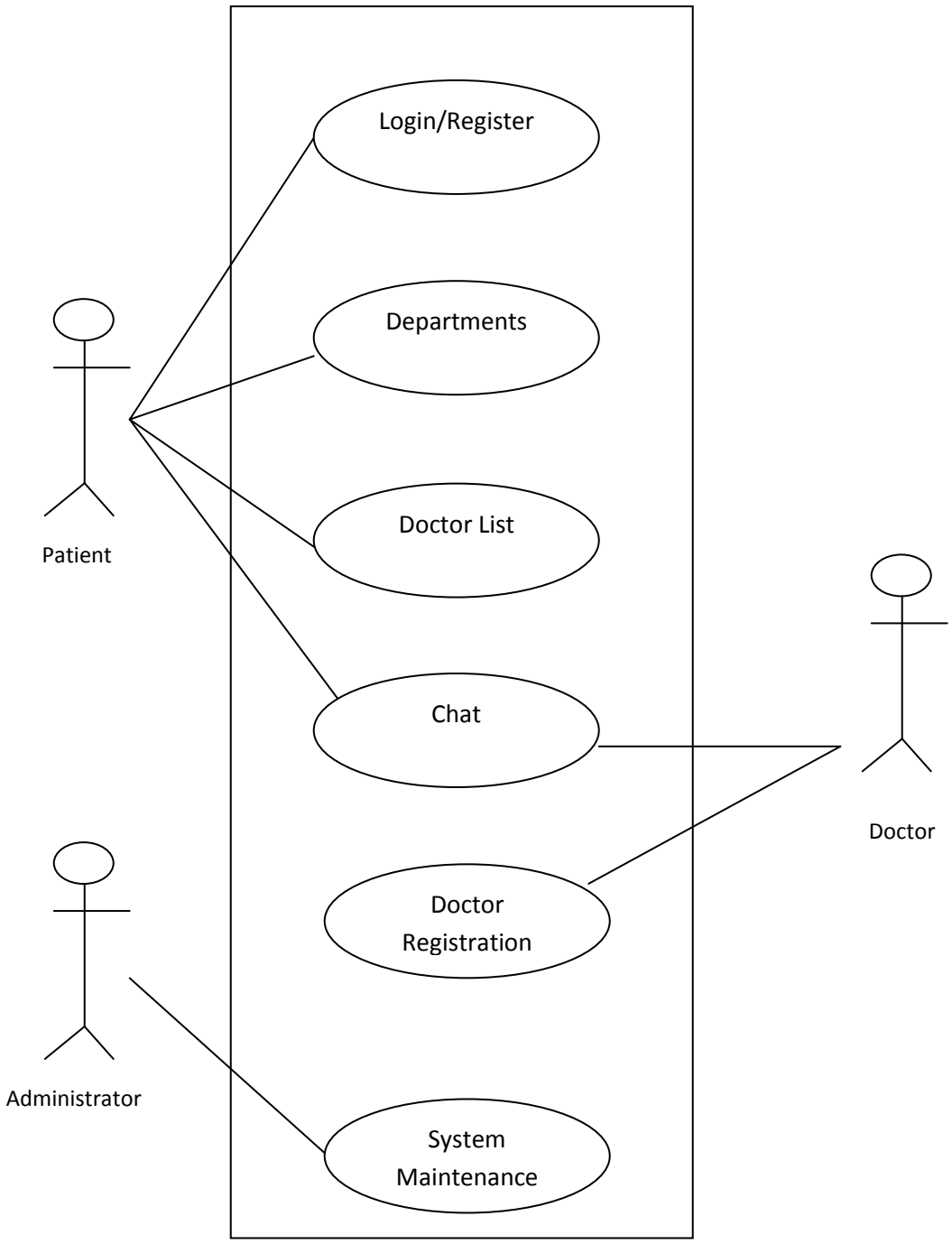

Fig 5.2Use Case Diagram for Virtual Hospital

5.2 Low Level Designs

During the detailed phase, the view of the application developed during the high level design is broken down into modules and programs.

5.2.1 Module 1 – Doctor Registration/Login

The Doctor has to upload his documents and profile details in order to register to be in the doctor's list that the patient approaches. His license number is checked for credibility and a temporarily generated password is sent to the Doctor's email to login for the first time. He can then change his password and continue. After logging in he will be redirected to a personalized page with the details of patients he is currently treating.

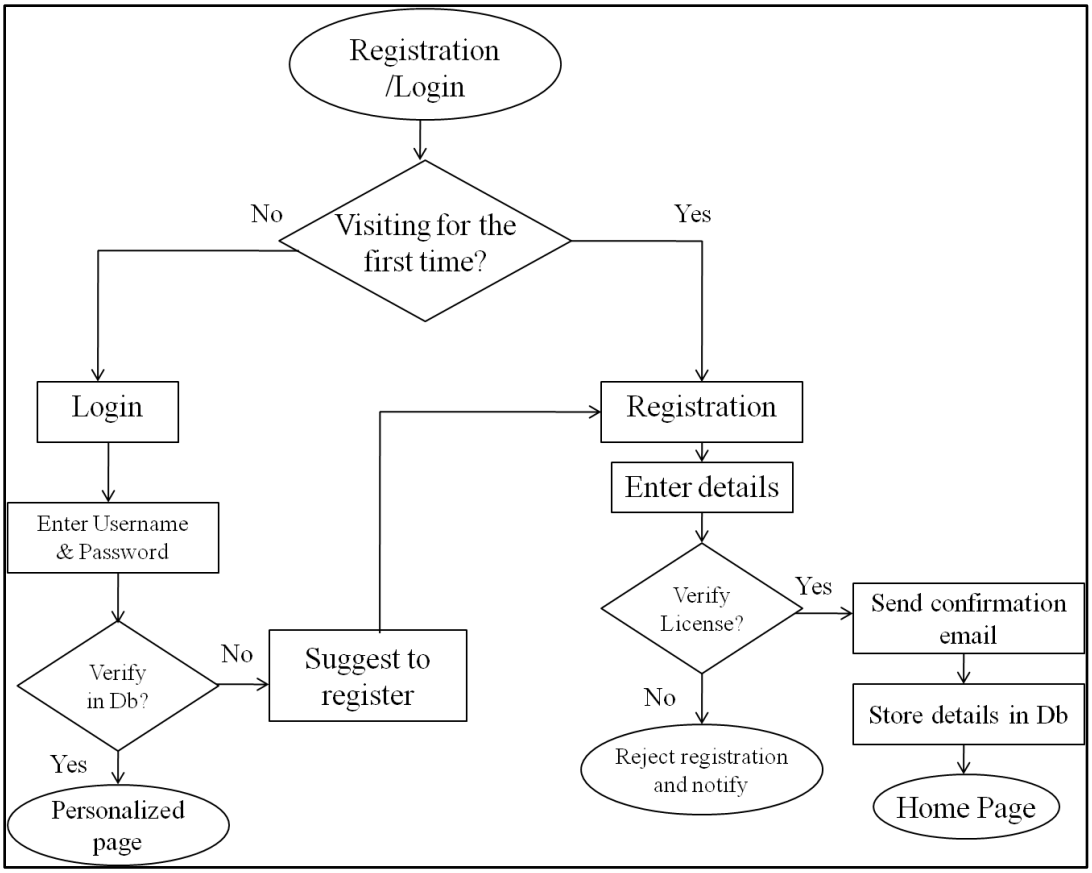

Fig 5.3Flowchart for Doctor-Registration/Login

5.2.2 Module 2 – Patient Registration/Login

The Patient can login to avail the services. Once the patient logs-in, he is redirected to a personalized page where he can contact the concerned doctor for consultation (through a chat application) and selecting appointments. If it is the first time then the patient has to register and create his profile.

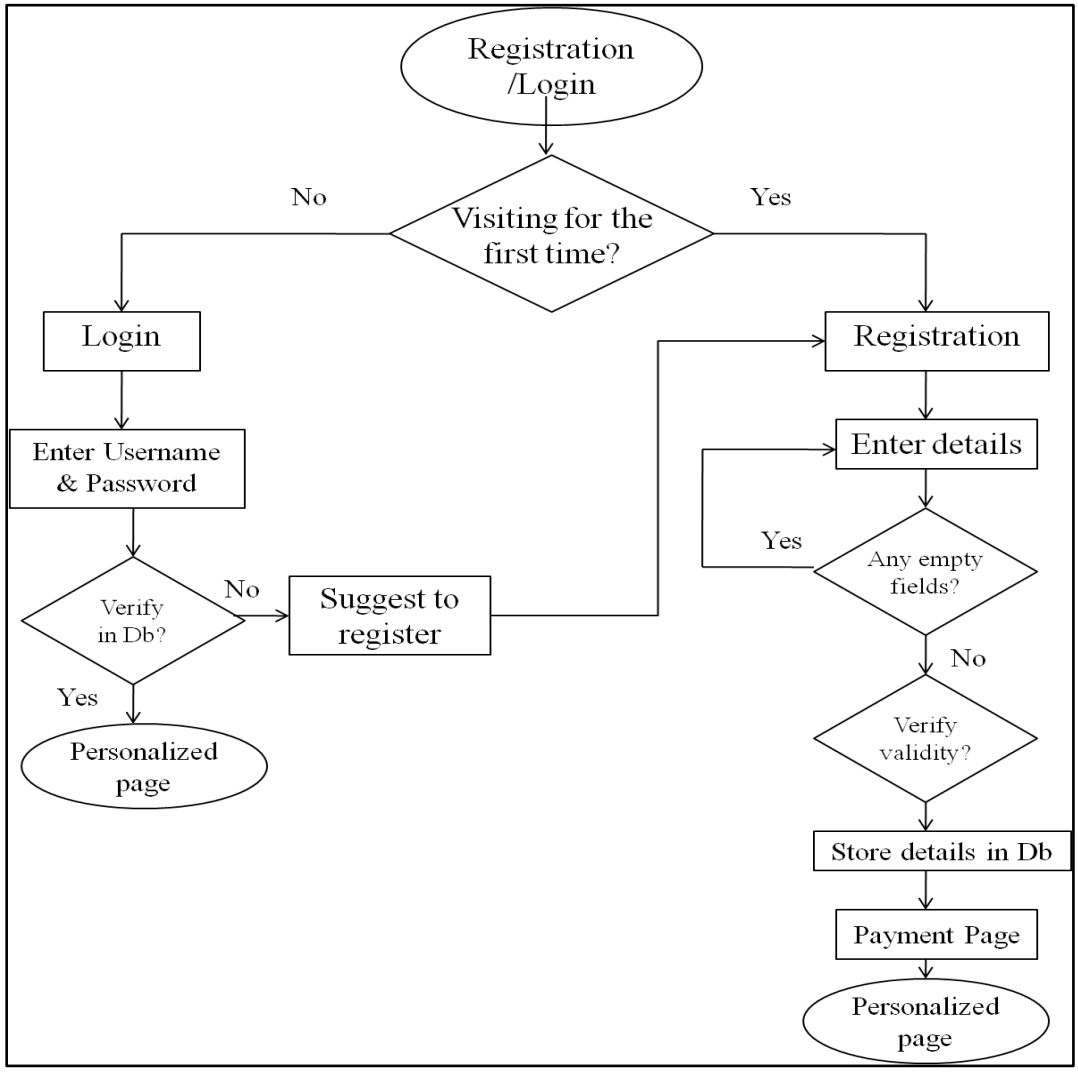

Fig 5.4Flowchart for Patient-Registration/Login

5.2.3 Module 3 - Health Tips

This module provides the patients with general health tips- at regular intervals- to build a better and healthier life. It is categorized into various modules so that the patient can browse through the content which he/she is interested in.

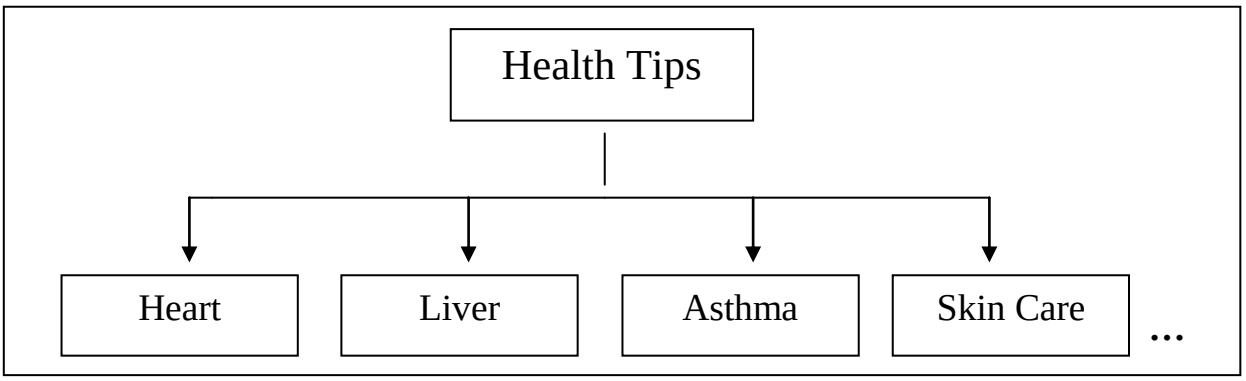

Fig 5.5 Block diagram representing various modules in Health Tips

5.2.4 Module 4 - Departments

This module provides the details about the various departments present in the virtual hospital. Each department contains the details about the doctors, their name, qualification and contact details.

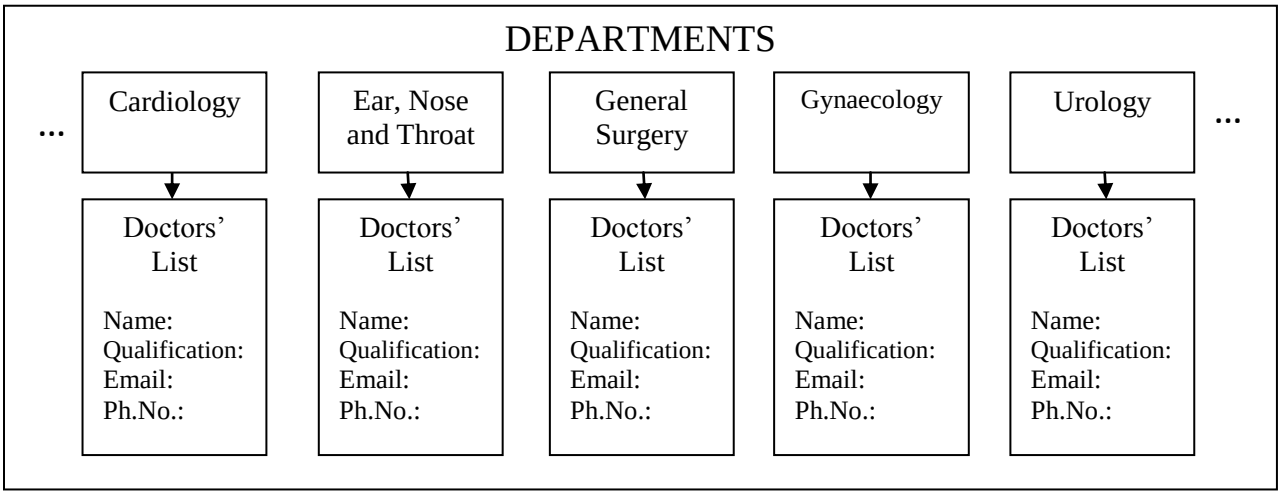

Fig 5.6 Block diagram representing various Departments

5.2.5 Module 5 - Text Alerts Through SMS

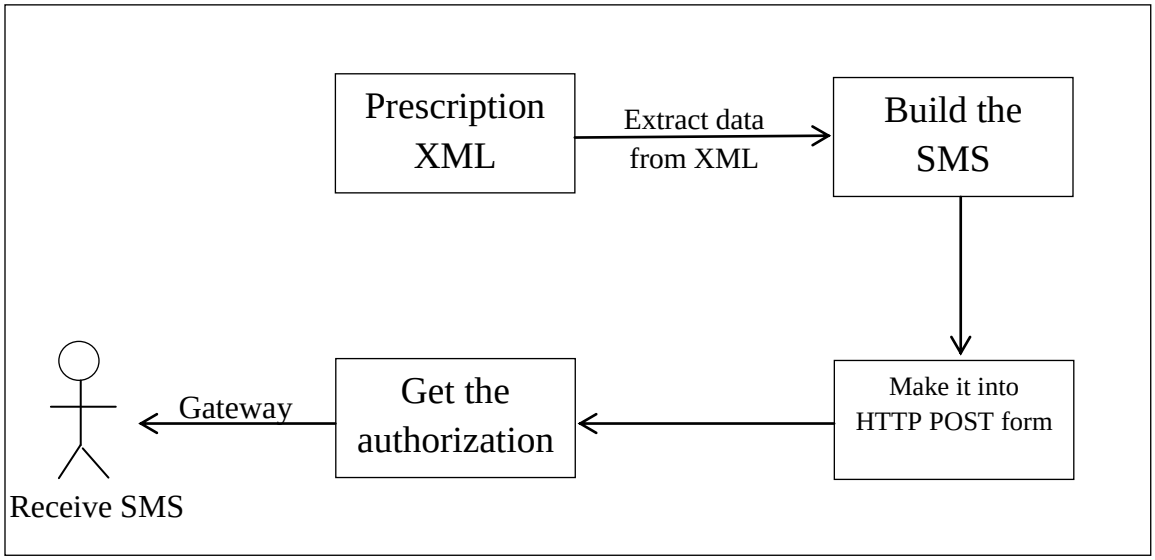

Fig 5.7 Working of Text alerts

As the previous module stated, the prescription details are stored in an xml file. This particular module accesses data from the xml file and the format of the text message is built based on the medicines to be taken that day at that hour. Once the message body and the recipient details are set, this information is built into a HTTP Post request using the url() function provided by java. The authorization for the sender is verified and the text message is sent through the SMS gateway (here, site2sms) to the concerned patient(s).

5.2.6 Module 6 - Consultation

This module provides means for the patient to interact with the doctor and avail the services. This interaction between the patient and the doctor is achieved through a chat application. The patient can upload any of his previous reports which will be available for the doctor to download and the doctor can do the same for the reports and prescriptions.

The patient's profile is updated with the reports and prescriptions that the concerned doctor suggests.

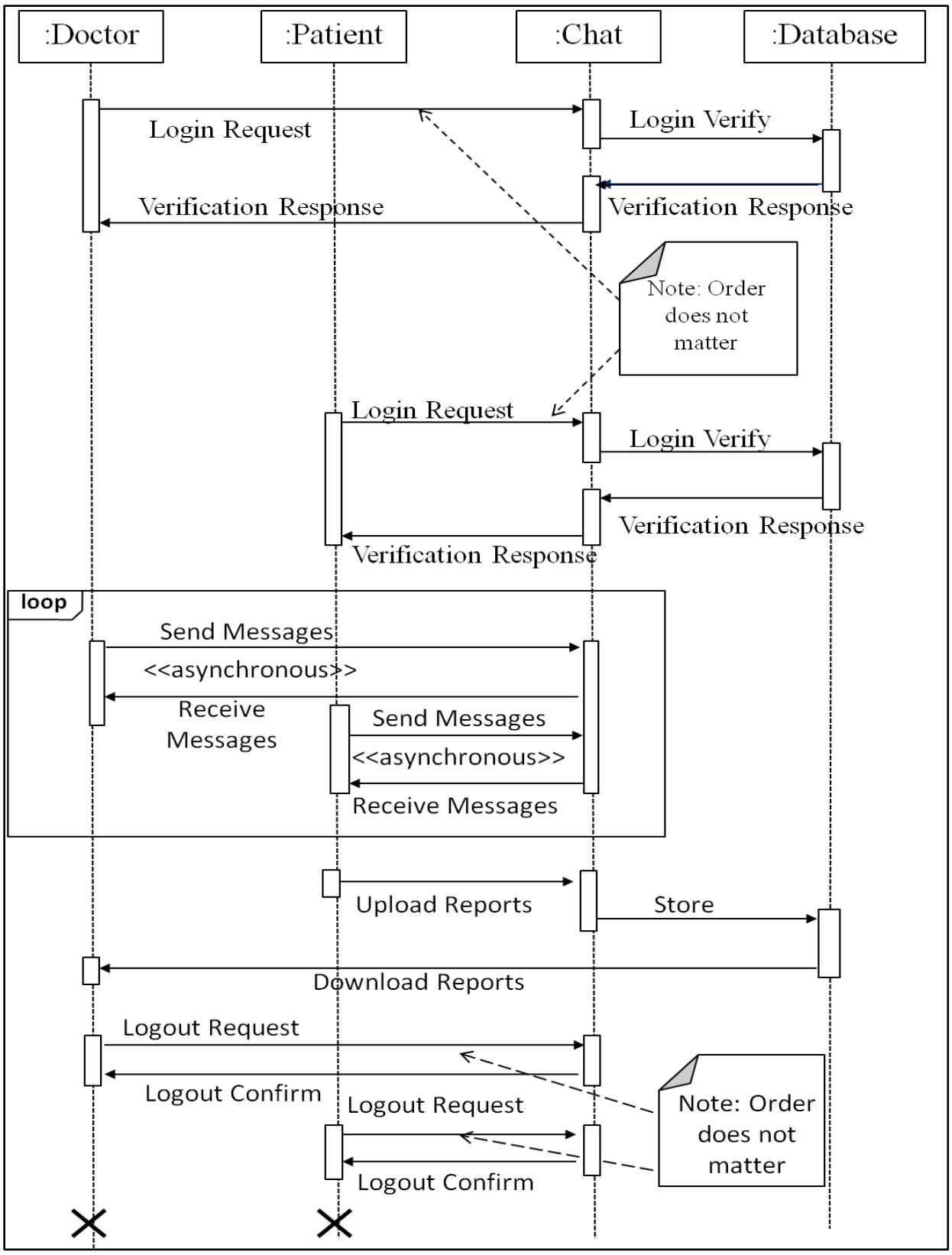

Fig 5.8 Sequence Diagram of Chat Application

5.2.7 Module 7 - Contact Us

This module provides contact details of the administrator. It also lets the user contact the administrator through a form element where he can fill in his/her queries/messages.

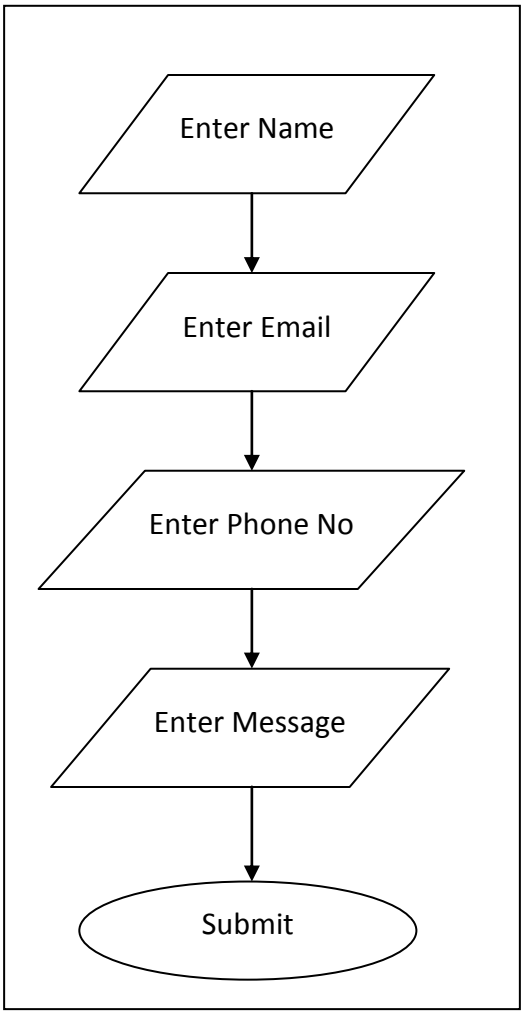

Fig 5.9 Flowchart for Contact Us module

5.2.8 Module 8 - Prescription Generation

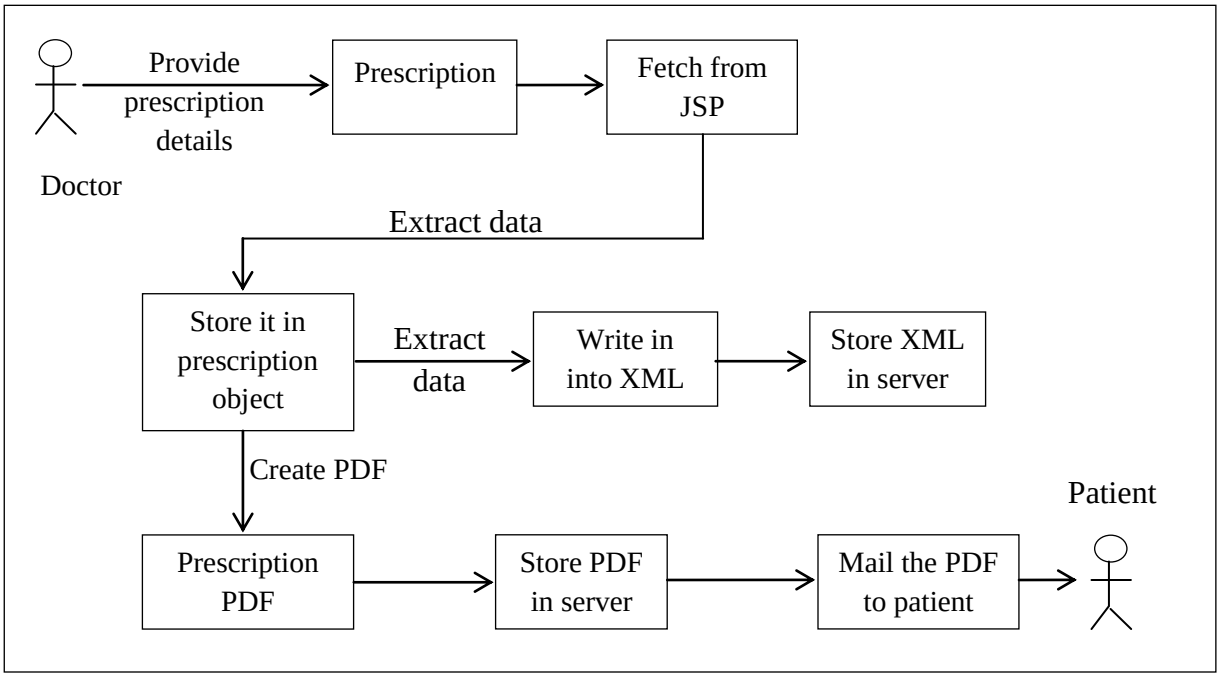

Fig 5.10 Process of prescription generation

The Fig 5.10 illustrates the processes involved in the generation of the prescription pdf file. The Doctor enters the prescription details through a form in a jsp page. The form values are later retrieved from the jsp file and stored in a Map data structure, named prescription. The flow splits into two actions here on viz., store in xml file for future reference and create a pdf file to be mailed to the patient.

5.2.9 Module 9 - Appointments

This module provides details about the appointments provided by the doctor through a period of one week, and also the slots booked by the patients to get an appointment with a particular doctor.

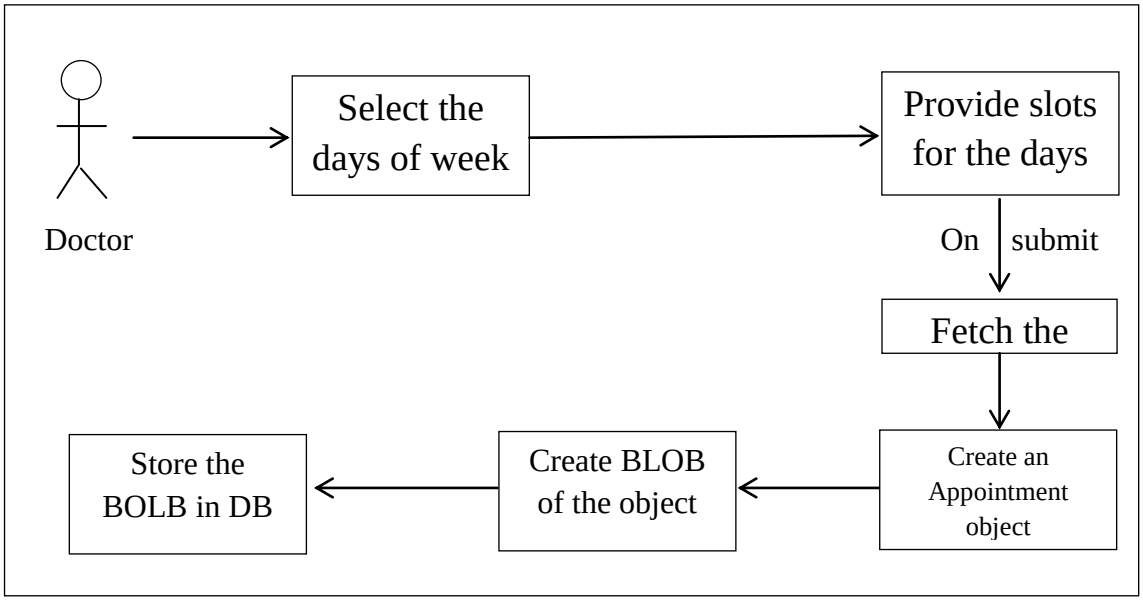

Fig 5.11 Slots provided by doctor for appointments

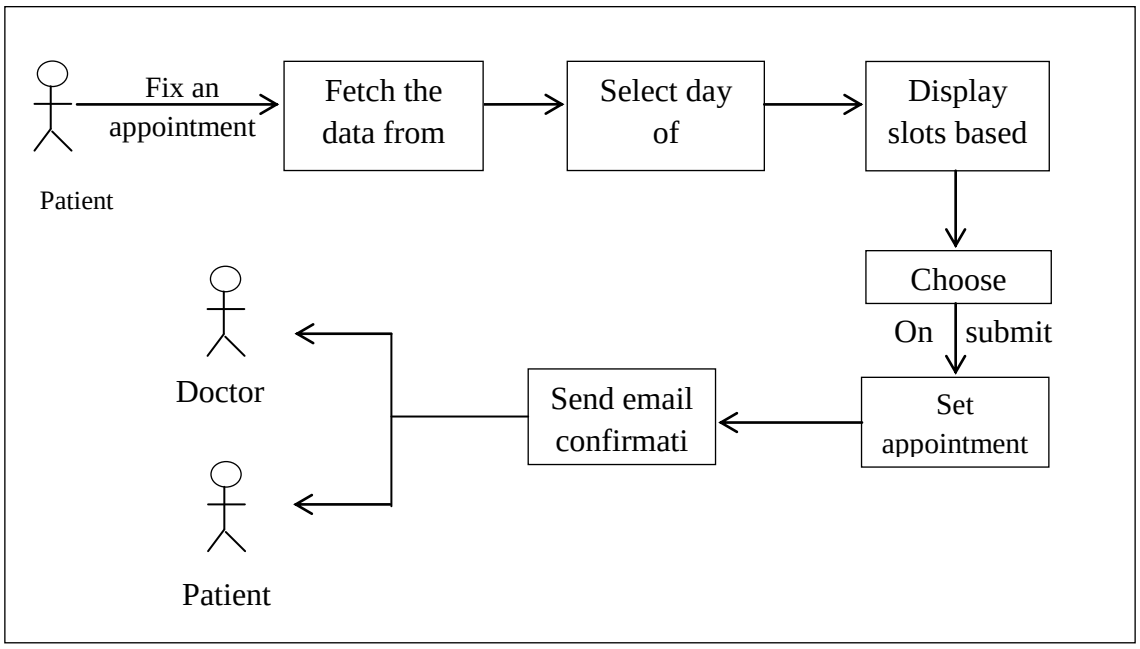

Fig 5.12 Slot booked by the patient for appointment with doctor

IMPLEMENTATION

6.1 Overview

The third stage in the SDLC is the Implementation stage. Implementation encompasses all the processes in getting new software or hardware operating properly in its environment, including installation, running, testing and the necessary changes. This chapter gives details of the implementation and the code snippets of various modules of the Virtual Hospital.

6.2 Chat Module

Chat is built on the client – server architecture where every doctor/patient when needs to start a chat will be given an instance of client which is called JMessenger. JMessenger is connected to the server i.e JServer through sockets and exchange of messages between 2 JMessengers happens with the help of JServer.

JMessenger has 5 Major Modules. They are

- SocketClient
- Message
- Download
- Upload
- History

It also includes a simple UI which is made using Swings and the events triggered by the components of this UI call the appropriate above module to chat with the appropriate doctor/patient.

6.2.1 Code Snippet For SocketClient

```
public class SocketClient implements Runnable{
        public SocketClient(ChatFrame frame) throws IOException{
                ui = frame; this.serverAddr = ui.serverAddr; this.port = ui.port;
                socket = new Socket(InetAddress.getByName(serverAddr), port);
                Out = new ObjectOutputStream(socket.getOutputStream());
                Out.flush();
                In = new ObjectInputStream(socket.getInputStream());
                hist = ui.hist;
        }
        @Override
        public void run() {
                boolean keepRunning = true;
                while(keepRunning){
                  try {
                        Message msg = (Message) In.readObject();
                        if(msg.type.equals("message")){
                          if(msg.recipient.equals(ui.username)){
                ui.jTextArea1.append("["+msg.sender +" > Me] : " + msg.content + "\n");
                          }
                          else{
                ui.jTextArea1.append("["+ msg.sender +" > "+ msg.recipient +"] : " +
        msg.content + "\n");
                }
               if(!msg.content.equals(".bye") && !msg.sender.equals(ui.username)){
                  String msgTime = (new Date()).toString();
                  try{
                                hist.addMessage(msg, msgTime);
                                DefaultTableModel table = (DefaultTableModel)
                        ui.historyFrame.jTable1.getModel();
```

```
                        table.addRow(new Object[]{msg.sender, msg.content,
                "Me", msgTime});
              }
              catch(Exception ex){}
            }
        }
        else if(msg.type.equals("login")){
            if(msg.content.equals("TRUE")){

              ui.jButton2.setEnabled(false);
              ui.jButton3.setEnabled(false);
              ui.jButton4.setEnabled(true); ui.jButton5.setEnabled(true);
              ui.jTextArea1.append("[SERVER > Me] : Login Successful\n");
              ui.jTextField3.setEnabled(false); ui.jPasswordField1.setEnabled(false);
            }
            else{
              ui.jTextArea1.append("[SERVER > Me] : Login Failed\n");
            }
        }
        else if(msg.type.equals("test")){
          ui.jButton2.setEnabled(true);
          ui.jTextField3.setEnabled(true); ui.jPasswordField1.setEnabled(true);
          ui.jButton7.setEnabled(true);
        }
        else if(msg.type.equals("newuser")){
            if(!msg.content.equals(ui.username)){
              boolean exists = false;
              for(int i = 0; i < ui.model.getSize(); i++){
                    if(ui.model.getElementAt(i).equals(msg.content)){
                      exists = true; break;
                    }
```

```
            }
              if(!exists){ ui.model.addElement(msg.content);
            }
      }
    }
    else if(msg.type.equals("signup")){
            if(msg.content.equals("TRUE")){
              ui.jButton2.setEnabled(false);
              ui.jButton4.setEnabled(true); ui.jButton5.setEnabled(true);
              ui.jTextArea1.append("[SERVER > Me] : Singup Successful\n");
            }
    else{
      ui.jTextArea1.append("[SERVER > Me] : Signup Failed\n");
      }
    }
    else if(msg.type.equals("signout")){
            if(msg.content.equals(ui.username)){
              ui.jTextArea1.append("["+ msg.sender +" > Me] : Bye\n");
              ui.jButton4.setEnabled(false);
              for(int i = 1; i < ui.model.size(); i++){
                    ui.model.removeElementAt(i);
              }
            ui.clientThread.stop();
            }
            else{
              ui.model.removeElement(msg.content);
              ui.jTextArea1.append("["+ msg.sender +" > All] : "+ msg.content +" has
    signed out\n");
            }
    }
    else if(msg.type.equals("upload_req")){
```

```
   if(JOptionPane.showConfirmDialog(ui, ("Accept "'+msg.content+'" from
"+msg.sender+" ?")) == 0){
   JFileChooser jf = new JFileChooser();
   jf.setSelectedFile(new File(msg.content));
   int returnVal = jf.showSaveDialog(ui);
   String saveTo = jf.getSelectedFile().getPath();
   if(saveTo != null && returnVal == JFileChooser.APPROVE_OPTION){
   Download dwn = new Download(saveTo, ui);
   Thread t = new Thread(dwn);
   t.start();
   send(new Message("upload_res", ui.username, (""+dwn.port), msg.sender));
}
else{
   send(new Message("upload_res", ui.username, "NO", msg.sender));
   }
}
else{
   send(new Message("upload_res", ui.username, "NO", msg.sender));
   }
}
else if(msg.type.equals("upload_res")){
          if(!msg.content.equals("NO")){
             int port  = Integer.parseInt(msg.content);
             String addr = msg.sender;
             ui.jButton5.setEnabled(false); ui.jButton6.setEnabled(false);
             Upload upl = new Upload(addr, port, ui.file, ui);
             Thread t = new Thread(upl);
             t.start();
          }
          else{
```

```
                    ui.jTextArea1.append("[SERVER > Me] : "+msg.sender+" rejected file
        request\n");
                }
        }
        else{
          ui.jTextArea1.append("[SERVER > Me] : Unknown message type\n");
          }
        }
        catch(Exception ex) {
          keepRunning = false;
          ui.jTextArea1.append("[Application > Me] : Connection Failure\n");
          ui.jTextField2.setEditable(true);
          ui.jButton4.setEnabled(false);
          ui.jButton5.setEnabled(false);
          ui.jButton5.setEnabled(false);
          for(int i = 1; i < ui.model.size(); i++){
                ui.model.removeElementAt(i);
          }
        ui.clientThread.stop();
        System.out.println("Exception SocketClient run()");
        ex.printStackTrace();
        }
  }
}
public void send(Message msg){
        try {
          Out.writeObject(msg);
          Out.flush();
          System.out.println("Outgoing : "+msg.toString());
          if(msg.type.equals("message") && !msg.content.equals(".bye")){
                String msgTime = (new Date()).toString();
```

```
                try{
                  hist.addMessage(msg, msgTime);
                  DefaultTableModel table = (DefaultTableModel)
        ui.historyFrame.jTable1.getModel();
                  table.addRow(new Object[]{"Me", msg.content, msg.recipient,
        msgTime});
                }
                catch(Exception ex){}
            }
        }
        catch (IOException ex) {
          System.out.println("Exception SocketClient send()");
        }
    }
public void closeThread(Thread t){
        t = null;
   }
}
```

6.2.2 Code Snippet For Messages

```
public class Message implements Serializable{
        private static final long serialVersionUID = 1L;
        public String type, sender, content, recipient;
        public Message(String type, String sender, String content, String recipient){
          this.type = type; this.sender = sender; this.content = content; this.recipient =
recipient;
        }
        @Override
        public String toString(){
          return "{type='"+type+"', sender='"+sender+"', content='"+content+"',
recipient='"+recipient+"'}";
        }       }
```

6.2.3 Code Snippet For Download

```
public class Download implements Runnable{
        public Download(String saveTo, ChatFrame ui){
          try {
                server = new ServerSocket(0);//creates a server socket bound to port 0.
                port = server.getLocalPort();
                this.saveTo = saveTo;
                this.ui = ui;
          }
          catch (IOException ex) {
                System.out.println("Exception [Download : Download(...)]");
          }
        }
        @Override
        public void run() {
          try {
                socket = server.accept();
                System.out.println("Download : "+socket.getRemoteSocketAddress());
                In = socket.getInputStream();
                Out = new FileOutputStream(saveTo);
                byte[] buffer = new byte[1024];
                int count;
                while((count = In.read(buffer)) >= 0){
                  Out.write(buffer, 0, count);}
                Out.flush();
                ui.jTextArea1.append("[Application > Me] : Download complete\n");
                if(Out != null){ Out.close(); }
                if(In != null){ In.close(); }
                if(socket != null){ socket.close(); }
                }
        }       }
```

6.2.4 Code Snippet For Upload

```
public Upload(String addr, int port, File filepath, ChatFrame frame){
        super();
        try {
          file = filepath; ui = frame;
          socket = new Socket(InetAddress.getByName(addr), port);
          Out = socket.getOutputStream();
          In = new FileInputStream(filepath);
        }
        catch (Exception ex) {
          System.out.println("Exception [Upload : Upload(...)]");
        }
}
@Override
public void run() {
        try {
          byte[] buffer = new byte[1024];
          int count;
          while((count = In.read(buffer)) >= 0){
                Out.write(buffer, 0, count);
          }
        Out.flush();
        ui.jTextArea1.append("[Applcation > Me] : File upload complete\n");
        ui.jButton5.setEnabled(true); ui.jButton6.setEnabled(true);
        ui.jTextField5.setVisible(true);
        if(In != null){ In.close(); }
        if(Out != null){ Out.close(); }
        if(socket != null){ socket.close(); }
}
```

6.2.5 Code Snippet For History

```
public class History {
        public History(String filePath){
          this.filePath = filePath;
        }
        public void addMessage(Message msg, String time){
        try {
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(filePath);
        Node data = doc.getFirstChild(
        Element message = doc.createElement("message");
        Element _sender = doc.createElement("sender");
        sender.setTextContent(msg.sender);
        Element _content = doc.createElement("content");
        content.setTextContent(msg.content);
        Element _recipient = doc.createElement("recipient");
        recipient.setTextContent(msg.recipient);
        Element _time = doc.createElement("time");
        time.setTextContent(time);
        message.appendChild(_sender);
        message.appendChild(_content);
        message.appendChild(_recipient);
        message.appendChild(_time);
        data.appendChild(message);
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(filePath));
        transformer.transform(source, result);
}
```

```
public void FillTable(HistoryFrame frame){
DefaultTableModel model = (DefaultTableModel) frame.jTable1.getModel();
        try{
        File fXmlFile = new File(filePath);
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.parse(fXmlFile);
        doc.getDocumentElement().normalize();
        NodeList nList = doc.getElementsByTagName("message");
        for (int temp = 0; temp < nList.getLength(); temp++) {
        Node nNode = nList.item(temp);
        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
        Element eElement = (Element) nNode;
              model.addRow(new Object[]{getTagValue("sender", eElement),
getTagValue("content", eElement), getTagValue("recipient", eElement),
getTagValue("time", eElement)});
            }
        }
      }
      catch(Exception ex){
        System.out.println("Filling Exception");
      }
   }

public static String getTagValue(String sTag, Element eElement) {
        NodeList nlList =
eElement.getElementsByTagName(sTag).item(0).getChildNodes();
        Node nValue = (Node) nlList.item(0);
        return nValue.getNodeValue();
  }
}
```

JServer has 3 Major Modules. They are

- SocketServer
- Database
- Message

SocketServer provides each client with an instance of Server thread which handles the request from that particular client. The Server thread contains the logic to send responses to the client.

Database refers to the login XML file which has the details about the user log in Id's and password. This will be used whenever an user starts a chat by entering his username and password.

Message here has the same format as that of the JMessenger and it implements Serializable class.

6.2.6 Code Snippet For SocketServer

```
public ServerThread(SocketServer _server, Socket _socket){
        super();
        server = _server;
        socket = _socket;
        ID     = socket.getPort();
        ui = _server.ui;
   }
public void send(Message msg){
        try {
          streamOut.writeObject(msg);
          streamOut.flush();
}       }
public void run(){
        ui.jTextArea1.append("\nServer Thread " + ID + " running.");
        while (true){
          try{
                Message msg = (Message) streamIn.readObject();
```

```
                server.handle(ID, msg);
            }
        }
}
public void open() throws IOException {
        streamOut = new ObjectOutputStream(socket.getOutputStream());
        streamOut.flush();
        streamIn = new ObjectInputStream(socket.getInputStream());
}
public void close() throws IOException {
        if (socket != null)    socket.close();
        if (streamIn != null)  streamIn.close();
        if (streamOut != null) streamOut.close();
        }
}

public class SocketServer implements Runnable {
        public SocketServer(ServerFrame frame){
        clients = new ServerThread[50];
        ui = frame;
        db = new Database(ui.filePath);
        try{
          server = new ServerSocket(port);
          port = server.getLocalPort();
          ui.jTextArea1.append("Server startet. IP : " + InetAddress.getLocalHost() + ",
Port : " + server.getLocalPort());
          start();
        }
}
```

```
public SocketServer(ServerFrame frame, int Port){
        clients = new ServerThread[50];
        ui = frame;
        port = Port;
        db = new Database(ui.filePath);
        try{
                server = new ServerSocket(port);
                port = server.getLocalPort();
                ui.jTextArea1.append("Server startet. IP : " + InetAddress.getLocalHost()
        + ", Port : " + server.getLocalPort());
                start();
        }
}
public void run(){
        while (thread != null){
          try{
                ui.jTextArea1.append("\nWaiting for a client ...");
                addThread(server.accept());
          }
        }
}
public void start(){
        if (thread == null){
                thread = new Thread(this);
                thread.start();
        }
}
```

```
public void stop(){
        if (thread != null){
                thread.stop();
                thread = null;
        }
}
private int findClient(int ID){
        for (int i = 0; i < clientCount; i++){
                if (clients[i].getID() == ID){
                        return i;
                }
        }
        return -1;
}

public synchronized void handle(int ID, Message msg){
        if (msg.content.equals(".bye")){
        Announce("signout", "SERVER", msg.sender);
        remove(ID);
        }

        else{
          if(msg.type.equals("login")){
                if(findUserThread(msg.sender) == null){
                  if(db.checkLogin(msg.sender, msg.content)){
                        clients[findClient(ID)].username = msg.sender;
                        clients[findClient(ID)].send(new Message("login", "SERVER",
                "TRUE", msg.sender));
                        Announce("newuser", "SERVER", msg.sender);
                        SendUserList(msg.sender);
                  }
```

```
                else{
                        clients[findClient(ID)].send(new Message("login", "SERVER",
                "FALSE", msg.sender));
                }
                }
                else{
                clients[findClient(ID)].send(new Message("login", "SERVER",
        "FALSE", msg.sender));
                }
          }
          else if(msg.type.equals("message")){
                if(msg.recipient.equals("All")){
                Announce("message", msg.sender, msg.content);
                }
             else{
                findUserThread(msg.recipient).send(new Message(msg.type,
msg.sender, msg.content, msg.recipient));
                clients[findClient(ID)].send(new Message(msg.type, msg.sender,
        msg.content, msg.recipient));
                }
          }
          else if(msg.type.equals("test")){
                clients[findClient(ID)].send(new Message("test", "SERVER", "OK",
        msg.sender));
          }
          else if(msg.type.equals("signup")){
             if(findUserThread(msg.sender) == null){
                if(!db.userExists(msg.sender)){
                        db.addUser(msg.sender, msg.content);
                        clients[findClient(ID)].username = msg.sender;
```

```
                    clients[findClient(ID)].send(new Message("signup", "SERVER",
            "TRUE", msg.sender));
                    clients[findClient(ID)].send(new Message("login", "SERVER",
            "TRUE", msg.sender));
                    Announce("newuser", "SERVER", msg.sender);
                    SendUserList(msg.sender);
            }
            else{
                    clients[findClient(ID)].send(new Message("signup", "SERVER",
            "FALSE", msg.sender));
            }
        }
        else{
          clients[findClient(ID)].send(new Message("signup", "SERVER", "FALSE",
msg.sender));
          }
        }
        else if(msg.type.equals("upload_req")){
          if(msg.recipient.equals("All")){
                clients[findClient(ID)].send(new Message("message", "SERVER",
        "Uploading to 'All' forbidden", msg.sender));
          }
          else{
                findUserThread(msg.recipient).send(new Message("upload_req",
        msg.sender, msg.content, msg.recipient));
          }
        }
        else if(msg.type.equals("upload_res")){
          if(!msg.content.equals("NO")){
                String IP =
        findUserThread(msg.sender).socket.getInetAddress().getHostAddress();
```

```
                findUserThread(msg.recipient).send(new Message("upload_res", IP,
        msg.content, msg.recipient));
          }
          else{
                findUserThread(msg.recipient).send(new Message("upload_res",
        msg.sender, msg.content, msg.recipient));
          }
        }
}
public void Announce(String type, String sender, String content){
        Message msg = new Message(type, sender, content, "All");
        for(int i = 0; i < clientCount; i++){
                clients[i].send(msg);
        }
}
public void SendUserList(String toWhom){
        for(int i = 0; i < clientCount; i++){
                findUserThread(toWhom).send(new Message("newuser", "SERVER",
        clients[i].username, toWhom));
        }
}
public ServerThread findUserThread(String usr){
        for(int i = 0; i < clientCount; i++){
          if(clients[i].username.equals(usr)){
                return clients[i];
          }
        }
        return null;
}
```

```
public synchronized void remove(int ID){
        int pos = findClient(ID);
        if (pos >= 0){
          ServerThread toTerminate = clients[pos];
          ui.jTextArea1.append("\nRemoving client thread " + ID + " at " + pos);
          if (pos < clientCount-1){
            for (int i = pos+1; i < clientCount; i++){
                clients[i-1] = clients[i];
            }
          }
        clientCount--;
        try{
                toTerminate.close();
        }
        toTerminate.stop();
  }
}
```

6.3 Prescription Module

This module retrieves the data from the form that the Doctor fills for a particular patient. All the required information is stored in a data structure named Dosage and is saved for further use to write the prescription pdf and send SMS alerts.

```
//tablets
public class tablets {
        public tablets() {       dose = new Dosage[3];    }
        public String getName() {       return name;    }
        public void setName(String name) {       this.name =  name;    }
        public void setQuantiy(String qnt) {       this.quantity = qnt;    }
        public String getQuantity() {       return this.quantity;    }
}
```

```
//presccription
public class presccription {
   public HashMap<String ,tablets>prescriptionMap;
   public presccription() {        this.prescriptionMap = new HashMap<>();    }
   public void setAge(int n){       this.age = n;    }
   public void setSex(String sex){       this.sex = sex;    }
   public int getAge(){       return this.age;    }
   public String getSex(){       return this.sex;    }
   public void setDoctorName(String name){       this.doctorname = name;    }
   public void setPatientName(String name){       this.patientname = name;    }
   public void setDays(int n){       this.numberOfDays = n;    }
   public void setTablets(int n){       this.numberOfTablets = n;    }
   public String getDoctor(){       return doctorname;     }
   public String getPatient(){       return patientname;     }
   public int getNumberOfDays(){       return numberOfDays;     }
   public int getNumberOfTablets(){       return numberOfTablets;     }
   public void enterTablet(String tabletName , tablets t){
         this.prescriptionMap.put(tabletName, t);
    }
}

//prescriptionStore
public class prescriptionstore extends HttpServlet {
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
         backend.presccription p = new backend.presccription();
         File file = new File("D:\\webApplication\\prescription.xml");
         boolean fetch = fetchFromJSP(p , request , response);
         File prescription = null;
         try {           prescription = writeToPdf(p);       }
```

```
        try {           boolean db = storeInDB(p,prescription);       }
        try {           boolean xml  = writeToXml(p,file);        }
        try {          boolean storeInServer = writeToServer(p,prescription,request);       }
        request.getRequestDispatcher("/doctorpersonal.jsp").forward(request, response);
}

private File writeToPdf(presccription p) throws BadElementException, IOException {
        fileNamePdf = p.getPatient()+year+month+date1;
        File file = new File("D://#JP//"+fileNamePdf+ ".pdf");
        try {
                Document document = new Document();
                PdfWriter.getInstance(document, new FileOutputStream(file));
                document.open();
                addMetaData(document,p);
                addTitlePage(document,p);
                addContent(document,p);
                addDeclaration(document,p);
                document.close();
        }sendmail(file, p);
        return file;
}

public void addMetaData(Document document,presccription p) {
   document.addTitle("Virtual Hospital");
   document.addSubject("Prescription");
   document.addKeywords("precription");
   document.addAuthor(p.getDoctor());
}
```

```
public void  addTitlePage(Document document,presccription p)
    throws DocumentException, BadElementException, IOException {

        try {
                //Establish connection with database
                //Query to avoid sql injection
                rs = st.executeQuery();
                if( rs.first()){
                        name = rs.getString(1);
                        docName = name;
                        qualification = rs.getString(2);
                }
        }
        Paragraph preface = new Paragraph();
        addEmptyLine(preface, 1);
Image logo =
Image.getInstance("D:\\#JP\\WebApplication1\\web\\images\\letterHead.gif");
        PdfPTable letterHead = new PdfPTable(2);
        PdfPCell c1 = new PdfPCell();
        c1.addElement(logo);
        c1.setBorder(2);
        letterHead.addCell((logo));
        PdfPCell c2 = new PdfPCell(new
Phrase(name.toUpperCase()+"\n\t"+qualification.toUpperCase()+"\n*add other info
here*\n"));
        c2.setHorizontalAlignment(Element.ALIGN_RIGHT);
        c2.setBorder(2);
        letterHead.addCell(c2);
        preface.add(letterHead);
        document.add(preface);
        Paragraph patDetail = new Paragraph();
```

```
        addEmptyLine(patDetail, 2);
        Paragraph p3 = new Paragraph();
        Chunk ch1 = new Chunk("Patient Name:",blueFont);
        ch1.setUnderline(0.5f, -1.5f);
        Chunk ch2 = new Chunk(p.getPatient().toUpperCase(),smallBold);
        Chunk ch3 = new Chunk("\nAge:",blueFont);
        ch3.setUnderline(0.5f, -1.5f);
        String pAge = String.valueOf(p.getAge());;
        Chunk ch4 = new Chunk(pAge,smallBold);
        Chunk ch5 = new Chunk("\nSex:",blueFont);
        ch5.setUnderline(0.5f, -1.5f);
        Chunk ch6 = new Chunk(p.getSex()+"\n",smallBold);
        p3.add(ch1);
        p3.add(ch2);
        p3.add(ch3);
        p3.add(ch4);
        p3.add(ch5);
        p3.add(ch6);
        p3.setAlignment(Element.ALIGN_RIGHT);
        patDetail.add(p3);
        document.add(patDetail);
}

public void addContent(Document document,presccription p) throws
DocumentException {

        Paragraph subPart = new Paragraph();
        addEmptyLine(subPart,2);
        createPrescription(subPart,p);
        document.add(subPart);
}
```

```
public void createPrescription(Paragraph subPart, presccription p){
        Paragraph p01 = new Paragraph("List of Medicines:\n\n", subFont);
        subPart.add(p01);
        Set<Map.Entry<String,tablets>>set = p.prescriptionMap.entrySet();
        int serialNo=1;
        for(Map.Entry<String,tablets>temp:set){
                tablets t = temp.getValue();
                String tname = null;
                String dos = " ";
                String quant = t.getQuantity();
                int noOfDays = p.getNumberOfDays();
                tname = t.getName();

                for(int i=0 ; i<3 ;i++){
                        if(t.dose[i]!=null){
                                Dosage d = t.dose[i];
                                dos = dos+"-  1 ";
                                if(d.getFood())
                                        dos = dos+" (bf)  ";
                                else if(!d.getFood())
                                        dos = dos+" (af)  ";

                        }
                        else{   dos = dos+"-  0  ";      }
                n=noOfDays;
                }
                String finaldos = serialNo+")
        "+tname.toUpperCase()+"\t\t\t"+dos+"\t\t\t[Quantity : "+quant+"]\n";
                Paragraph p00 = new Paragraph(finaldos, smallBold);
                serialNo++;
                subPart.add(p00);
```

```
        }
        subPart.add(new Paragraph("\nPrescription duration : "+n+" days", smallBold));
}

public void addDeclaration(Document document, presccription p) throws
DocumentException{
        Paragraph declaration = new Paragraph();
        addEmptyLine(declaration, 2);
declaration.add(new Paragraph("*Prescription generated by:
"+p.getDoctor().toUpperCase() +" , On " + new Date() + "\n", smallBold));
        declaration.add(new Paragraph("**This prescription is provided by a valid doctor
and authorised by Virtual Hospital**", redFont));
        document.add(declaration);
}

public void addEmptyLine(Paragraph paragraph, int number) {
        for (int i = 0; i < number; i++) {
                paragraph.add(new Paragraph(" "));
        }
}

private boolean storeInDB(presccription p, File prescription) throws SQLException,
FileNotFoundException {

        Connection connect = null;
        PreparedStatement statement = null;
        try {           Class.forName("com.mysql.jdbc.Driver");        }
    //Establish connection with database
        statement = connect.prepareStatement("INSERT INTO prescription
VALUES(?,?)");
        statement.setString(1,p.getPatient());
```

```
        FileInputStream fis;
        fis = new FileInputStream(prescription);
        statement.setBinaryStream(2, fis);
        boolean execute = statement.execute();
        return execute;
}

private boolean writeToXml(presccription p, File xmlFile1) throws
TransformerException, SAXException, IOException, ClassNotFoundException,
InstantiationException, SQLException, IllegalAccessException {
        try {
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        org.w3c.dom.Document doc;
        doc = docBuilder.parse(xmlFile1);
        doc.getDocumentElement().normalize();
        org.w3c.dom.Element rootElement = doc.getDocumentElement();
  // patient elements
        org.w3c.dom.Element patient = doc.createElement("patient");
        rootElement.appendChild(patient);
  // name elements
        org.w3c.dom.Element name = doc.createElement("name");
        name.appendChild(doc.createTextNode(p.getPatient()));
        patient.appendChild(name);

        org.w3c.dom.Element age = doc.createElement("age");
        age.appendChild(doc.createTextNode(Integer.toString(p.getAge())));
        patient.appendChild(age);
        org.w3c.dom.Element sex = doc.createElement("sex");
        sex.appendChild(doc.createTextNode(p.getSex()));
        patient.appendChild(sex);
```

```
        org.w3c.dom.Element dateofprescription =
doc.createElement("dateofprescription");
        Date dt = new Date();
        dateofprescription.appendChild(doc.createTextNode(dt.toString()));
        patient.appendChild(dateofprescription);
        org.w3c.dom.Element numberOfDays = doc.createElement("numberOfDays");

numberOfDays.appendChild(doc.createTextNode(Integer.toString(p.getNumberOfDays()
)));
        patient.appendChild(numberOfDays);
        ResultSet rs;
  //Establish connection with database
  //Query to avoid sql injection
      rs = st.executeQuery();
      String mobileNumber = null;
      if(rs.first())
         mobileNumber = rs.getString(1);
      org.w3c.dom.Element mobileno = doc.createElement("MobileNumber");
      mobileno.appendChild(doc.createTextNode(mobileNumber));
      patient.appendChild(mobileno);
      Set<Map.Entry<String , tablets>>map = p.prescriptionMap.entrySet();
      for(Map.Entry<String , tablets>temp:map){
         org.w3c.dom.Element tablet = doc.createElement("tablet");
         (doc.createTextNode(temp.getKey()));
         tablet.setAttribute("quantity", temp.getValue().getQuantity());
         org.w3c.dom.Element tname = doc.createElement("name");
         tname.appendChild(doc.createTextNode(temp.getKey()));
         tablet.appendChild(tname);

         tablets t = temp.getValue();
         for(int i=0 ; i< 3 ;i++){
```

```
        Dosage d = t.dose[i];
        if(d!=null){
                String time = d.getTime().toString();
                org.w3c.dom.Element dosage = doc.createElement(time);
                if(d.getFood())
                dosage.appendChild(doc.createTextNode("Beforefood"));}
        else{
                dosage.appendChild(doc.createTextNode("Afterfood"));
                tablet.appendChild(dosage);}
        }
        patient.appendChild(tablet);
}
// write the content into xml file
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
DOMSource source = new DOMSource(doc);
StreamResult result = new StreamResult(xmlFile1);
transformer.transform(source, result);
System.out.println("File saved!");
 }
return true;
}

@SuppressWarnings("empty-statement")
private boolean fetchFromJSP(presccription p,HttpServletRequest request,
HttpServletResponse response) {
      doctorname = request.getParameter("doctorname");
      p.setDoctorName(doctorname);
      patientname = request.getParameter("patientname");
      p.setPatientName(patientname);
      sex = request.getParameter("sex");
```

```
p.setSex(sex);
age = Integer.parseInt(request.getParameter("age"));
p.setAge(age);

//Retrieve details of tablet-1
tablet = request.getParameter("tablet1");
if(tablet!=null && !tablet.isEmpty())
{
   tablets t = new tablets();
   t.setName(tablet);

            //Set the quantity
   quantity = request.getParameter("quantity1");
   t.setQuantiy(quantity);

   //Get the values of Morning session
String[] temp = request.getParameterValues("Dosage1m");

  //Initialize a temporary array to store details of that session
  boolean[] temp1 = new boolean[3];
  for(int i=0; i<3; i++){
     temp1[i] = false;
  }
  //Check if it is "Before Food" or "After Food" and store the same
  if(temp != null ){
      if(temp[0]!=null)
         temp1[0] = true;
      if(temp[1].equals("bf"))
         temp1[1] = true;
      else if(temp[1].equals("af"))
         temp1[2] = true;
```

```
        //Update all of the retrieved information into the Dosage data structure
        if(temp1[0] == true){
            Dosage d = new Dosage();
            d.setTime(Dosage.Time.MORNING);
            if(temp1[1] == true)
                d.setFood(1);
            else if(temp1[2] == true)
                d.setFood(2);
            t.dose[0] = d;
        }
}

/*Get the values of Afternoon session
 *Initialize a temporary array to store details of that session
 *Check if it is "Before Food" or "After Food" and store the same
 *Update all of the retrieved information into the Dosage data structure
 */

/*Get the values of Evening session
 *Initialize a temporary array to store details of that session
 *Check if it is "Before Food" or "After Food" and store the same
 *Update all of the retrieved information into the Dosage data structure
 */
//Repeat the above steps for the other 4 tablets and update the tablets data structure
      numberOfDays = Integer.parseInt(request.getParameter("numberofdays"));
      p.setDays(numberOfDays);
      return true;
}
```

```
private boolean writeToServer(presccription p, File prescription ,HttpServletRequest
request) throws FileUploadException, Exception {
      try {
      InputStream is = new FileInputStream(prescription);
      String filename = p.getPatient()+fileNamePdf;
      String outputfile = this.getServletContext().getRealPath(filename+".pdf");
      FileOutputStream os = new FileOutputStream (outputfile);
      System.out.println(outputfile);
      // write bytes taken from uploaded file to target file
      int ch = is.read();
      while (ch != -1) {
                  os.write(ch);
                  ch = is.read();
      }
      os.close();
      System.out.println("prescription saved");
   }
   return true;
  }
}
```

6.4 Appointment Module

This module implements the appointment generation of the web application. The provideSlots code block sets the appointment slots available for a doctor on certain days. The takeAppoinment code block helps the patient to select an appointment. Every slot provided by the doctor is stored as an Appoinment data structure that contains objects of type Days to specify the time. This Appointments data steucture is stored as a blob (binary large objects) on the database for future access.

```
//Appointments
public class Appointment  implements Serializable{
   private static final long serialVersionUID = 091734954340005586L;
```

```
    int numberOfDays;
    public int numberOfSlots;
    Day day;
    public HashMap<Day,HashMap<LocalTime,String>>appointments;
    HashMap<LocalTime,String>temp;

    public Appointment() {
        this.appointments = new  HashMap<Day,HashMap<LocalTime,String>>();
    }
    public void storeInMAp(Day d1 , LocalTime start ,LocalTime end){
         DateTimeFormatter formatter;
        formatter = DateTimeFormat.forPattern("HH:mm");
         Days d = new Days();
        this.day = d1;
         d.setStartTime(start);
        d.setEndTime(end);
        d.generateSlots();
         this.numberOfSlots = d.returnSlots()+1;
        temp = d.map2;
         int count = numberOfSlots;
        LocalTime time = start;
        while(count!=0){
            temp.put(time, null);
            time = time.plusMinutes(60);
            count--;
        }
        appointments.put(day, temp);
    }
public void setNumberOfDays(int n){
        this.numberOfDays = n;
    }
```

```
public void updateMap(String time1 , Day d, String s){
        DateTimeFormatter formatter;
        formatter = DateTimeFormat.forPattern("HH:mm");
        int trim = time1.lastIndexOf(":");
        String time2 = time1.substring(0, trim);
        System.out.println(time1);
        LocalTime time = LocalTime.parse(time2,formatter);
        System.out.println(time);
        HashMap<LocalTime,String>innermap = appointments.get(d);
        innermap.put(time,s);
        appointments.put(d,innermap);
   }
}

//Days
public class Days {
    HashMap<LocalTime,String>map2;
    public Days(){
        map2 = new HashMap<>();
    }
    public enum Day {
        SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
        THURSDAY, FRIDAY, SATURDAY
    }
    public void setDay(Day d){
        this.day = d;
    }
     public void setStartTime(LocalTime t){
        this.startTime = t;
    }
```

```
    public void setEndTime(LocalTime t){
        this.endTime = t;
    }
    public void generateSlots(){
        int totalTime;
        int minutes = Minutes.minutesBetween(startTime, endTime).getMinutes();
        totalTime = minutes/60;
        this.numberOfSlots = totalTime * 2;
    }
    public int returnSlots(){
        return  this.numberOfSlots;
    }
    public Day fetchDay(){
        return this.day;
    }
 }

//slotProvider
public class slotprovider extends HttpServlet {
   private boolean fetchFromJsp(Appointment ap, HttpServletRequest request) throws
SQLException {
        String license = request.getParameter("license");
        String days[] = request.getParameterValues("Days");
        ap.setNumberOfDays(days.length);
        DateTimeFormatter formatter;
        formatter = DateTimeFormat.forPattern("HH:mm");
        Day d;
        String from;
        String to;
```

```
      if(days != null){
          for (String day : days) {
             switch(day){
                case "Mon":
                    d = Day.MONDAY;
                    from = request.getParameter("monfrom");
                    to = request.getParameter("monto");
                    ap.storeInMAp(d, LocalTime.parse(from, formatter),
LocalTime.parse(to,formatter));
                   break;
                case "Tue":
                    //Do the same for Tuesday's entry
                   break;
                case "Wed":
                    //Do the same for Wednesday's entry
                   break;
                case "Thur":
                    //Do the same for Thursday's entry
                   break;
                case "Fri":
                    //Do the same for Friday's entry
                   break;
                case "Sat":
                    //Do the same for Saturday's entry
                   break;
                case "Sun":
                    //Do the same for Sunday's entry
                   break;
}    }     }
       storeInDb(ap,license);
       return true;    }
```

```
    private void storeInDb(Appointment ap , String license) throws SQLException {
         Connection connect = null;
        PreparedStatement statement = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(prescriptionstore.class.getName()).log(Level.SEVERE, null,
ex);
        }
      connect = DriverManager.getConnection
("jdbc:mysql://localhost:3306/mynewdatabase?zeroDateTimeBehavior=convertToNull","
root","virtualhospital@0");
      statement = connect.prepareStatement("Delete from appointments where doctor_id
=?");
      statement.setString(1, license);
      int execute = statement.executeUpdate();
      System.out.println(license+" delete executed : "+execute);
     statement = connect.prepareStatement("INSERT INTO
appointments(doctor_id,serialized_object) VALUES(?,?)");
     statement.setString(1 , license);
     statement.setObject(2, ap);
     execute = statement.executeUpdate();
     System.out.println("db done");
   }
}

//takeAppointment
public class TakeAppointments extends HttpServlet {
```

```
    private boolean fetchFromJsp(HttpServletRequest request) throws IOException,
ClassNotFoundException, IllegalAccessException, InstantiationException,
SQLException {
  String patientName = request.getParameter("user");
  String previousConsultation = request.getParameter("group");
  String reasonForAppoint = request.getParameter("reason");
  String dayOfAppoint = request.getParameter("day");
  String AppointmentTime = request.getParameter(dayOfAppoint);
  String did = fetchdoctor(patientName);
  Appointment a = fetchfromdb(did);
  a.updateMap(AppointmentTime, Days.Day.valueOf(dayOfAppoint), patientName);
  updateInDb(a,did);
  sendMail(did , patientName ,  dayOfAppoint , AppointmentTime ,reasonForAppoint ,
previousConsultation);
  int index = AppointmentTime.lastIndexOf(":");
  String at2 = AppointmentTime.substring(0, index);
  storeInDb(patientName, at2 , dayOfAppoint , did);
  return true;
 }
public String fetchdoctor(String patient) {
        ResultSet rs;
        String did = null;
    try {
        //Establish connection with database
        //Write code to avoid sql injection
        if(rs.first())
            did = rs.getString(1);


    } catch (InstantiationException | IllegalAccessException | ClassNotFoundException |
SQLException e) {
```

```
    }
    return did;
  }

public Appointment fetchfromdb(String did) throws IOException {
        ResultSet rs;
        Appointment a = null;
    try {
       //Establish connection with database
      //Write code to avoid sql injection
       rs.next();
        byte[] buf = rs.getBytes(1);
       ObjectInputStream objectIn = null;
       if (buf != null)
            objectIn = new ObjectInputStream(new ByteArrayInputStream(buf));
       Object appointmentObject = objectIn.readObject();
        a = (Appointment)appointmentObject;
       rs.close();
    }
       return a;
 }

private void updateInDb(Appointment a, String did) throws ClassNotFoundException,
InstantiationException, IllegalAccessException, SQLException {
        ResultSet rs;
       //Establish connection with database
       //Write code to avoid sql injection
          st.setObject(1, a);
          st.setString(2, did);
          int count =  st.executeUpdate();
    }
```

```
private void sendMail(String did, String patientName, String dayOfAppoint, String
AppointmentTime, String reasonForAppoint, String previousConsultation) throws
ClassNotFoundException, InstantiationException, IllegalAccessException,
SQLException {
        //Code to send e-mail confirmation
}
private void storeInDb(String patientName, String AppointmentTime, String
appointmentday , String did) throws SQLException {
      try {

         Date d = new Date();


         try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
         } catch (    InstantiationException | IllegalAccessException ex) {
            Logger.getLogger(TakeAppointments.class.getName()).log(Level.SEVERE,
null, ex);
         }
      } catch (ClassNotFoundException ex) {
         Logger.getLogger(TakeAppointments.class.getName()).log(Level.SEVERE, null,
ex);
      }
         Connection con = null;
         try {
            //Establish connection with database
         } catch (SQLException ex) {
            Logger.getLogger(patientregister.class.getName()).log(Level.SEVERE, null,
ex);
         }
         PreparedStatement pst = null;
```

```
        try {
            pst = (PreparedStatement) con.prepareStatement("insert into patient_appointment
values(?,?,?,?,?)");
        } catch (SQLException ex) {
            Logger.getLogger(TakeAppointments.class.getName()).log(Level.SEVERE, null,
ex);
        }
        try {
            pst.setString(1,patientName);
        } catch (SQLException ex) {
            Logger.getLogger(TakeAppointments.class.getName()).log(Level.SEVERE, null,
ex);
        }
              pst.setString(2,AppointmentTime);
              pst.setString(3,appointmentday );
              pst.setString(4, did);
              pst.setString(5,null);
             int i = pst.executeUpdate();

    }
}
```

6.5 SMS Module

The necessary prescription data that are needed to send reminders about the medicines to patients are stored in a prescription XML file. A scheduler runs every hour and is made to perform its assigned task(to send SMS) at the scheduled time. The module uses a HTTP POST request to the SMS Gateway with the required data built into an URL.

6.5.1 Code Snippet For Formatting SMS Text

This module implements fetching of required data and delegates it to the SMS module

```
public class SmsMain {
  public static void main(String[] args) {
        final SmsMain sm = new SmsMain();
        Timer timer = new Timer();
        TimerTask tt = new TimerTask(){
          public void run(){
                Calendar cal = Calendar.getInstance();
                int hour = cal.get(Calendar.HOUR_OF_DAY);
                if(hour == 7){   sm.sendsms("MORNING","Beforefood");  }
                if(hour == 9){   sm.sendsms("MORNING","Afterfood");    }
                if(hour == 12){   sm.sendsms("AFTERNOON","Beforefood");      }
                if(hour == 14){   sm.sendsms("AFTERNOON","Afterfood");       }
                if(hour == 18){   sm.sendsms("EVENING","Beforefood");  }
                if(hour == 20){   sm.sendsms("EVENING","Afterfood");    }
        };
        timer.schedule(tt, 1000, 1000*60*60);
}
private void sendsms(String time, String food) {
        sms s = new sms();
        File file = new File("D:\\webApplication\\prescription.xml");
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        doc = docBuilder.parse(file);
        doc.getDocumentElement().normalize();
        NodeList patients = doc.getElementsByTagName("patient");
        for(int i=0 ; i < patients.getLength();i++){
          Node patient = patients.item(i);
```

```
            Node name = patient.getFirstChild();
            String ptntname = name.getTextContent();
            String tabletname = "%20";
            String mobno = null;
            NodeList tablets = patient.getChildNodes();
            for(int j = 0 ;j < tablets.getLength() ; j++){
                  Node tablet = tablets.item(j);
                  switch (tablet.getNodeName().toString()) {
                    case "tablet":
                          String temp = tablet.getFirstChild().getTextContent();
                          NodeList child = tablet.getChildNodes();
                          for(int k = 0; k < child.getLength();k++){
                                Node mae = child.item(k);
                                if(mae.getNodeName().equals(time) &&
                          mae.getTextContent().equals(food))
                                tabletname = tabletname+"%20,"+temp;
                          }
                          break;
                    case "MobileNumber":
                          mobno = tablet.getTextContent();
                          break;
                  }
            }
            if(tabletname!=null && !tabletname.equals("%20"))
                  s.delegate(mobno , ptntname , food , tabletname );
            }
}
```

6.5.2 Code Snippet For Sending SMS

This module takes the data from the previous module, formats the message body, establishes a connection with the third party SMS gateway and sends the text message through the registered account.

```
public class sms {
    public void delegate(String number , String patientname , String food , String medicine) {
    String msg1 = "%0AHello"+"%20"+patientname;
    String msg2 = "%0AThis%20is%20a%20reminder%20to%20take%20the%20following%20medicines%20"+food+":%0A";
    String message = msg1+msg2+medicine;
    String Url = "https://site2sms.p.mashape.com/index.php?";
    String text1 = "uid="+"9886071109";
    String text2 = "&pwd="+"srivathsa";
    String text3 = "&phone="+number;
    String text4 = "&msg="+message;
    String finalUrl = Url+text1+text2+text3+text4;
    URL url;
    url = new URL(finalUrl);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("X-Mashape-Authorization", "zxIeiJEdrJ4sjNb6MgRZTyCC9V1c7rOa");
    conn.setDoOutput(true);
    int responseCode = conn.getResponseCode()
    BufferedReader in = new BufferedReader( new InputStreamReader(conn.getInputStream()));
    String inputLine;
    StringBuffer response = new StringBuffer();
```

```
        while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
        }
        in.close();
        }
}
```

6.6 E-Mail Module

This module implements the sending of electronic mails to Doctors (for registration confirmation with password and appointment reminders) and Patients (for appointment reminders and prescription)

```
boolean sendmail(File file , presccription p) {
       Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");
        Session session = Session.getInstance(props,
        new javax.mail.Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication() {
                return new
PasswordAuthentication("virtualhospitalteam@gmail.com","finalyearproject");
                }
        });
       String mailid = getMail(p);
        try {
                Message message = new MimeMessage(session);
                message.setFrom(new
InternetAddress("virtualhospitalteam@gmail.com"));
                message.setRecipients(Message.RecipientType.TO,
                InternetAddress.parse(mailid));
```

```
                message.setSubject("Virtual Hospital Confirmation Email");

                 // Create the message part
                BodyPart messageBodyPart = new MimeBodyPart();
                String text1 = "Hello,\n" ;
                String text2 = " Here is your prescription!\n";
                String text3 = "We will remind you on the registered phone";
                String text6 = "\n\n Thank You,\n";
                String text7 = "Virtual Hospital Team";
                String finaltext = text1+text2+text3+text6+text7;
                messageBodyPart.setText(finaltext);
                Multipart multipart = new MimeMultipart();
                // Set text message part
                multipart.addBodyPart(messageBodyPart);
        // Part two is attachment
        messageBodyPart = new MimeBodyPart();
        String filename = fileNamePdf+".pdf";
        DataSource source = new FileDataSource(file);
        messageBodyPart.setDataHandler(new DataHandler(source));
        messageBodyPart.setFileName(filename);
        multipart.addBodyPart(messageBodyPart);
        // Send the complete message parts
        message.setContent(multipart );
        Transport.send(message);
             System.out.println("Done");
             return true;
          }
}
```

TESTS AND RESULTS

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs (errors or other defects).

- White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

- Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

Sl No.	Action	Expected Result	Actual Result	Status
1	After opening the application the patient should register by entering required details like name, age, sex, symptoms, email ID etc.	The entered information should be stored in the respective database and if a required column is left blank then it should ask the patient to enter the details.	As Expected	OK
2	The doctor similar to the patient has to initially register by entering the details like UPIN, email ID, mobile number etc.	The entered information should be stored in the respective database and if a required column is left blank then it should ask the doctor to enter the details. Also, username and	As Expected	OK

		password will be generated automatically which will be mailed to the doctor.		
3	In order to book an appointment, consult a doctor etc., the patient must first login by entering his username and password.	The systemwill verify the login details of the patient and must either allow or deny based on the details in the database.	As Expected	OK
4	In order to provide slots for the patient, give prescription, enter health tips the doctor must first login to his account.	The system will verify the login details of the patient and must either allow or deny based on the details in the database.	As Expected	OK
5	After logging-in the doctor must provide slots for the patients to consult him/her. This is done by the doctor by fixing a particular set of slots in a week.	The slots provided by the doctor are stored in a class which in-turn is stored as an object on the database. These slots must be displayed in the corresponding patients' profiles.	As Expected	OK
6	The patient, after logging-in can book slots in a particular time of the week based on the slots provided by the corresponding doctor.	If a slot is already booked then it must be blocked and when a patient is book a slot no other patient must be able to get the same slot. Also the patient must be restricted from booking the slots which are already taken by someone else.	As Expected	OK
7	After the patient books a particular slot he must be able to consult the corresponding doctor using the chat application.	The chat application will develop a connection between the client and server and allows the doctor and patient to chat and upload files.	As Expected	OK
8	The doctor must be able to fill the prescription details for a particular patient with the respective dosages and submit it.	The submitted PDF must be automatically converted to PDF format and mailed to the respective patient. Also this must be stored in an XML file for sending text alerts.	As Expected	OK
9	The patient should have entered his	The patient will automatically get prescription alerts via SMS	As Expected	OK

	mobile number and should have a prescription provided by a doctor.	about the dosages and also at what point of time in the day the patient is supposed to take the medication.		
10	The patient must book a slot in order to consult a doctor.	After booking the slot the patient must get an automatically generated mail to the email-id which he has provided while registering.	As Expected	OK
11	The patient/doctor can upload an image which he prefers to be the profile picture for his account.	The picture must be displayed in the patient/doctor's personalized page.	As Expected	OK

Table 7.1 Test Cases for Virtual Hospital

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

Virtual Hospital brings the "hospital environment" online and eliminates the distance barrier existing between the doctors and patients. It also helps in expanding the health facilities beyond their offices.

The doctor analyzes the patient's symptoms and interacts with the patient through a chat application. This is called the preliminary diagnosis, which helps patients to prevent a particular health problem occurring in the near future. The patient is free to share files like previous prescriptions provided by a different doctor and test reports, in the chat application. Prescriptions are generated in the standard format which is mailed to the patient and the patient also has an option to download the prescription through his profile which facilitates easier access to the prescriptions.

Patients are reminded about their medications and dosages at regular intervals through text alerts to their cellular phone. There has been multitude of attempts, some of them successful, in producing online medical solutions and some having their own drawbacks; we have tried to overcome the drawbacks of the existing solutions by creating a medical web application which is more comprehensive and easy to use.

8.2 FUTURE WORK

This web application, on link up with a bank, can be extended to include the fee payment option online. Also, if the registered doctors agree to share their license number officially through the medical council of India, then the license numbers can be used in our web application. If the medical electronic industry comes up with more tele-medical devices to conduct different medical tests from home, these devices can be used in our web application to send the results of the test conducted to the doctor for more thorough analysis.

BIBLIOGRAPHY

[1] http://docs.oracle.com/javase/7/docs/api/

[2] http://en.wikipedia.org/wiki/XML

[3] http://www.w3.org/TR/REC-xml/

[4] http://www.w3.org/TR/xml11/#charsets

[5] http://www.w3.org/TR/2008/REC-xml-20081126/#charsets

[6] http://xml.coverpages.org/xmlApplications.html

[7] http://www.codeproject.com/Articles/524120/A-Java-Chat-Application

[8] https://netbeans.org/kb/docs/webclient/html5-gettingstarted.html

[9] http://www.mkyong.com/java

[10] http://www.stat.berkeley.edu/~statcur/Workshop2/Presentations/XML.pdf

[11] http://www.w3.org/TR/2012/CR-html5-20121217/

[12] http://www.w3schools.com/html/html5_intro.asp

[13] http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html

[14] Stephanie Bodoff et al: The J2EE Tutorial, 2nd Edition, Pearson Education, 2004.

SNAPSHOTS

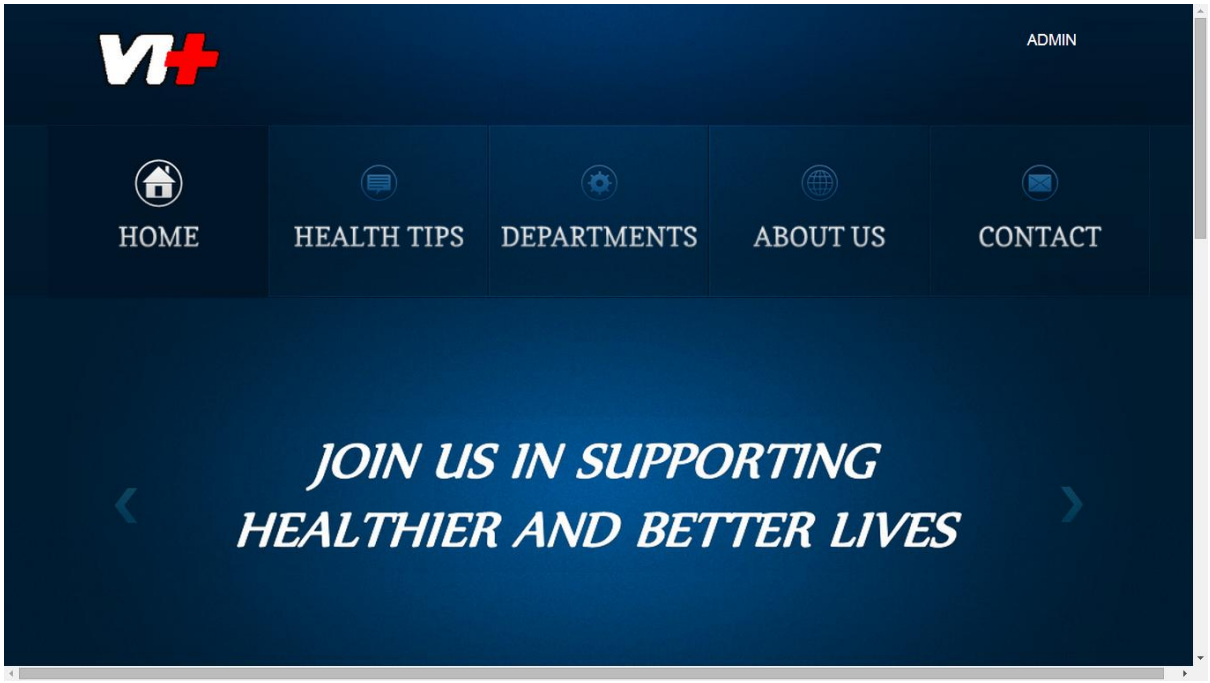

Fig A.1 Home page – Part 1

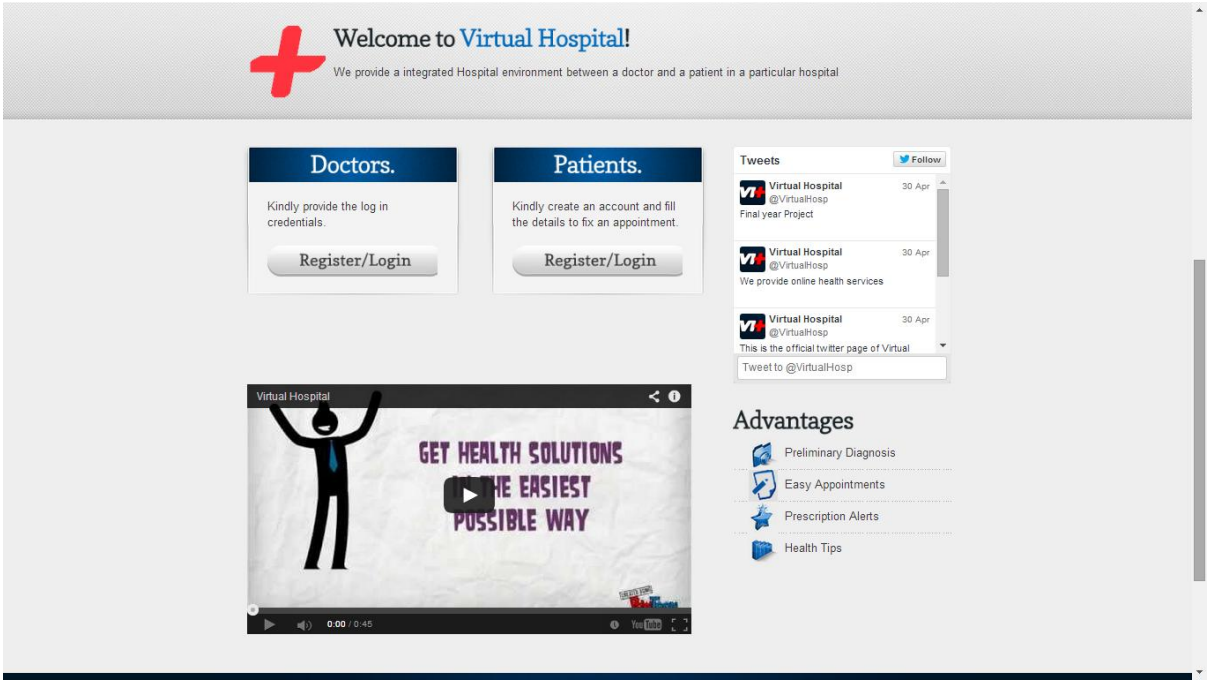

Fig A.2 Home page – Part 2

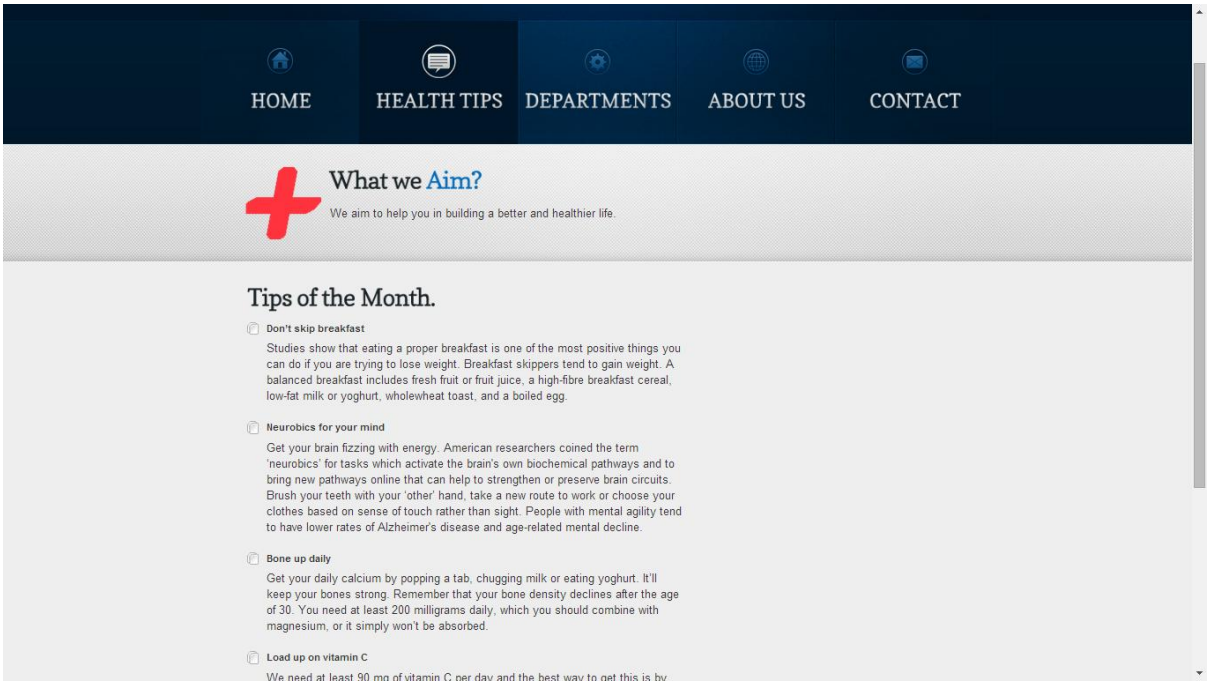

Fig A.3 Health Tips page

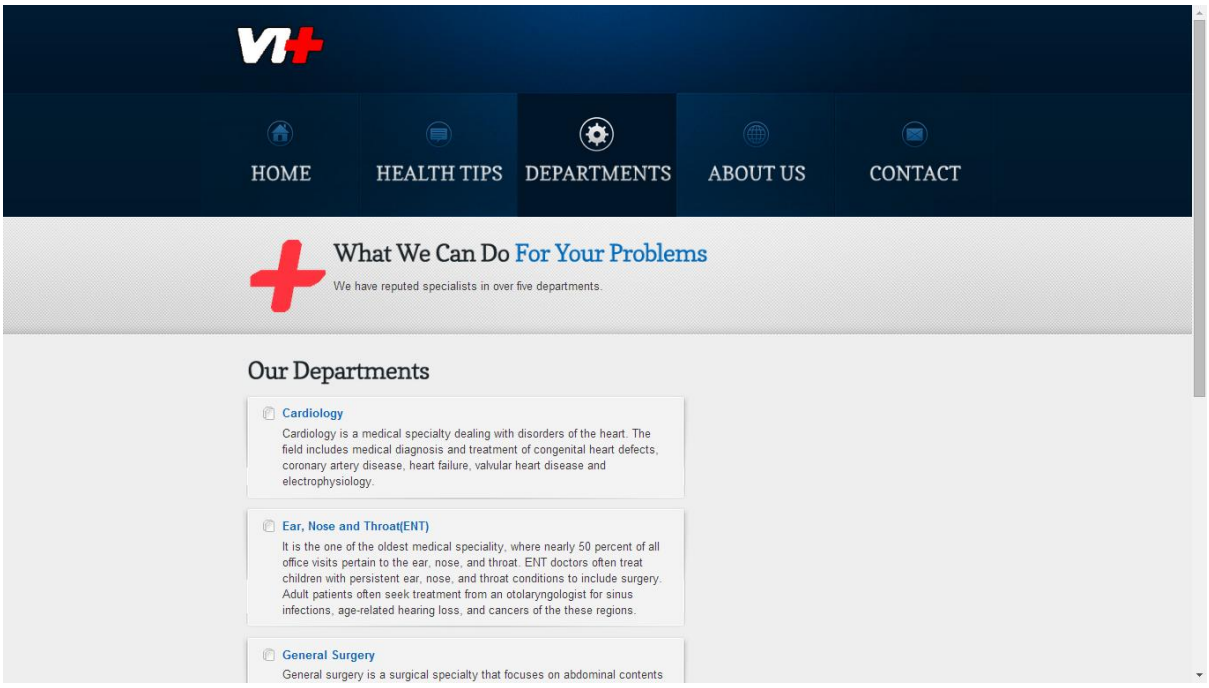

Fig A.4 Departments Page

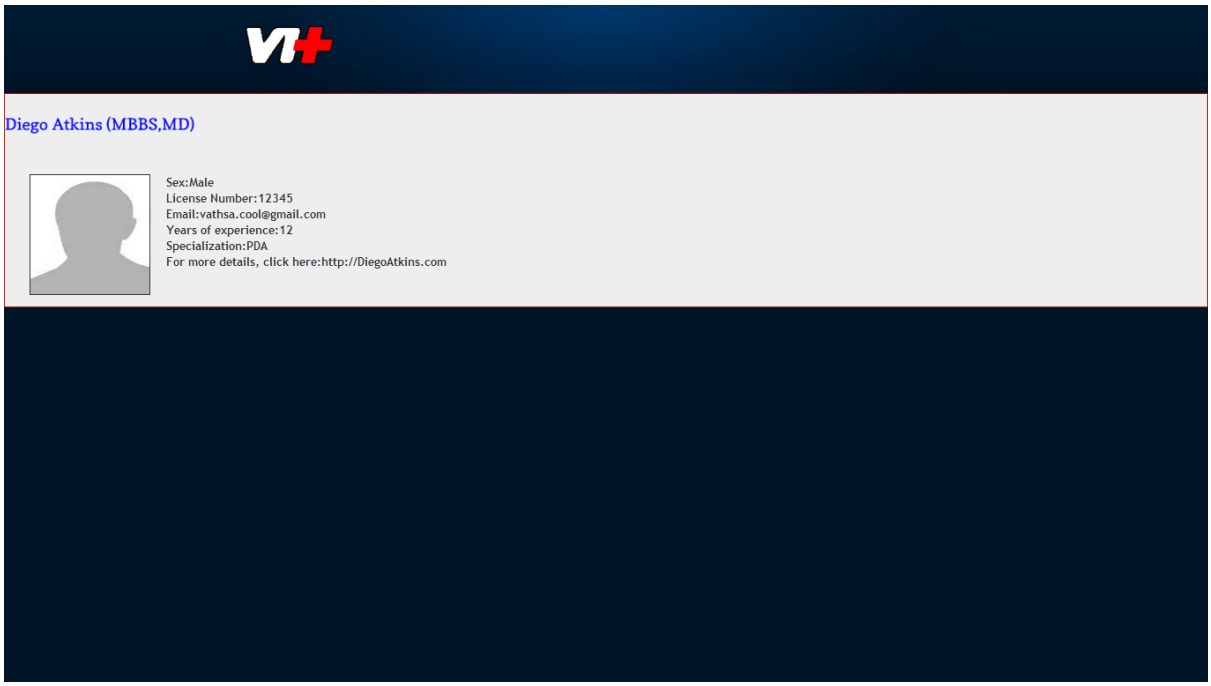

Fig A.5 Doctor's details in a particular department

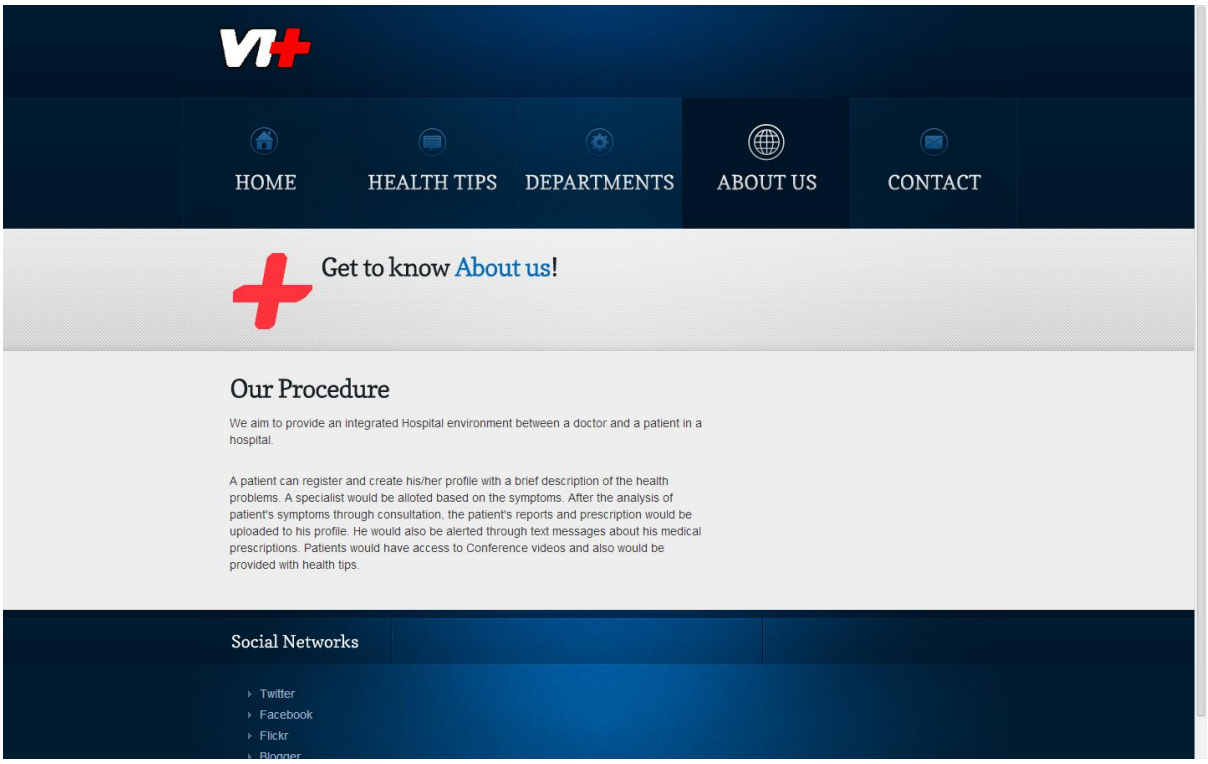

Fig A.6 About Us page

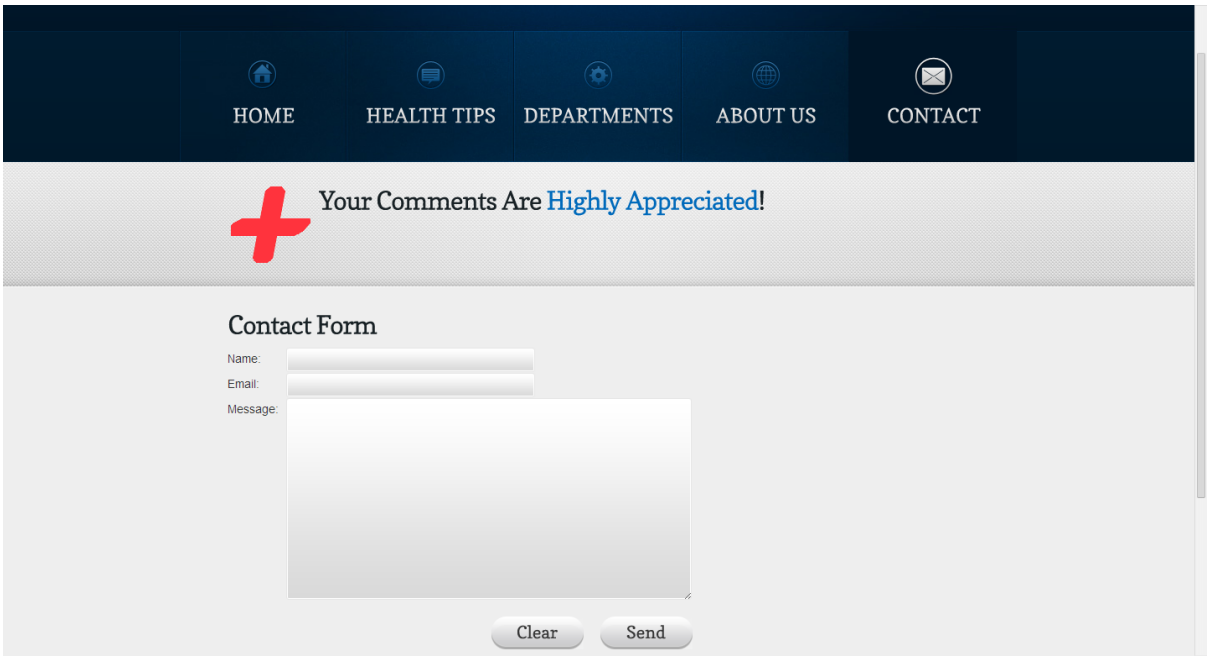

Fig A.7 Contact Us page

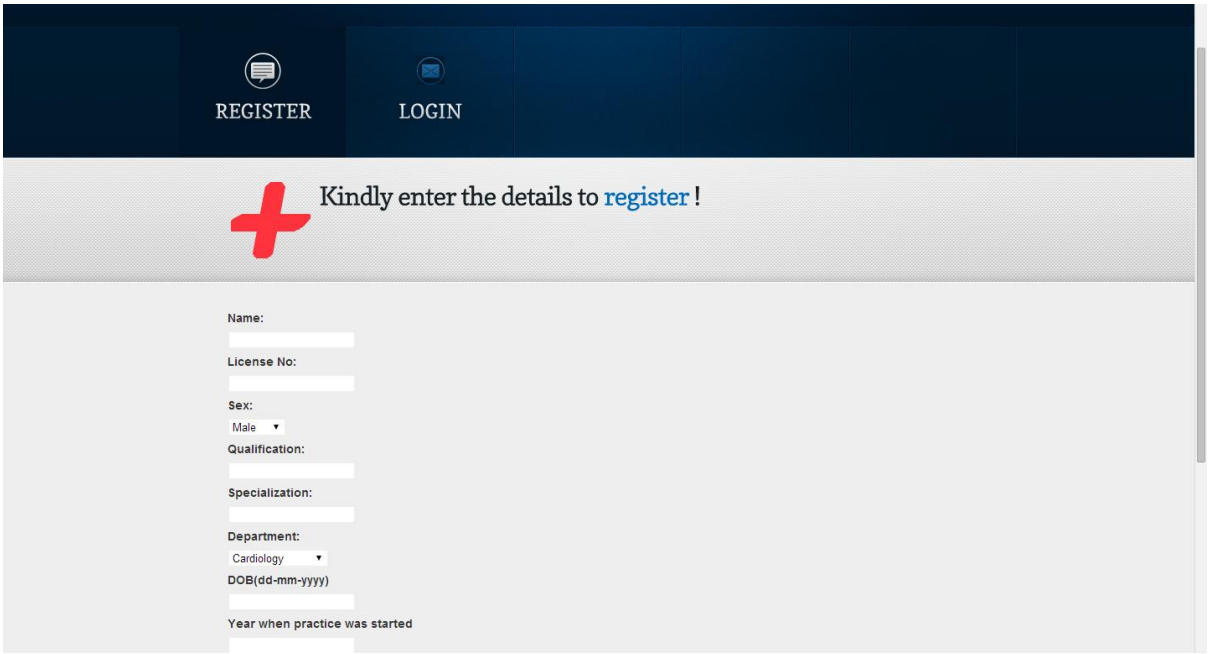

Fig A.8 Doctor's registration page

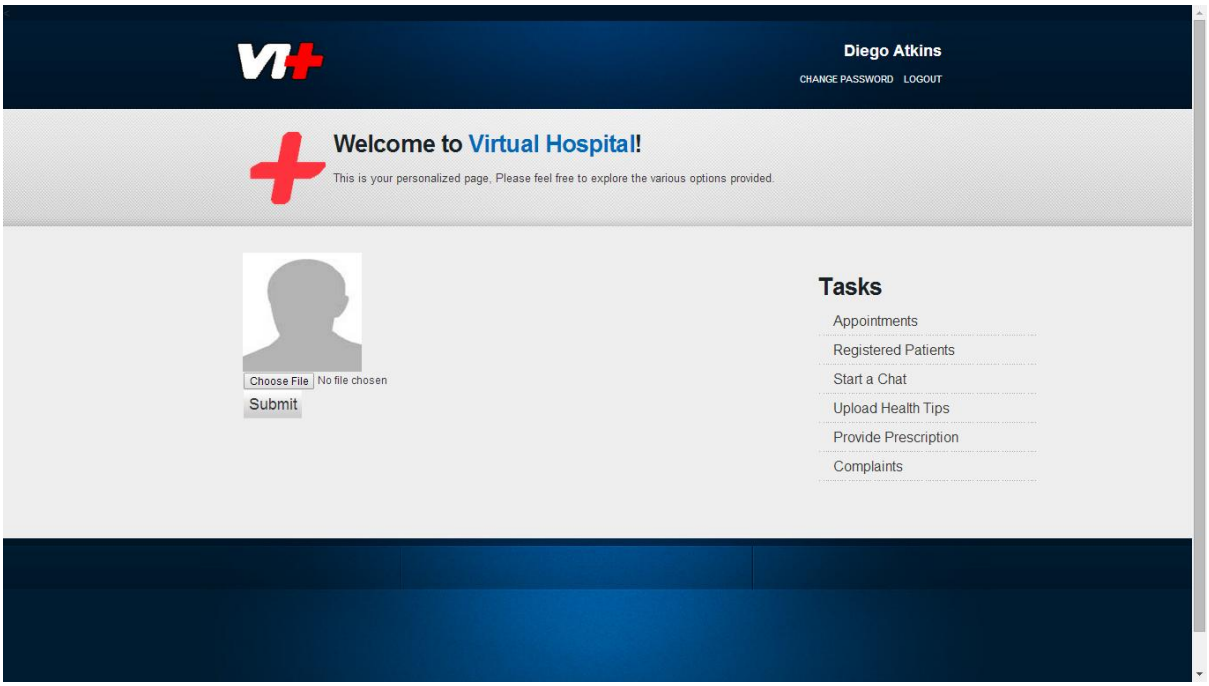

Fig A.9 Doctor's profile page

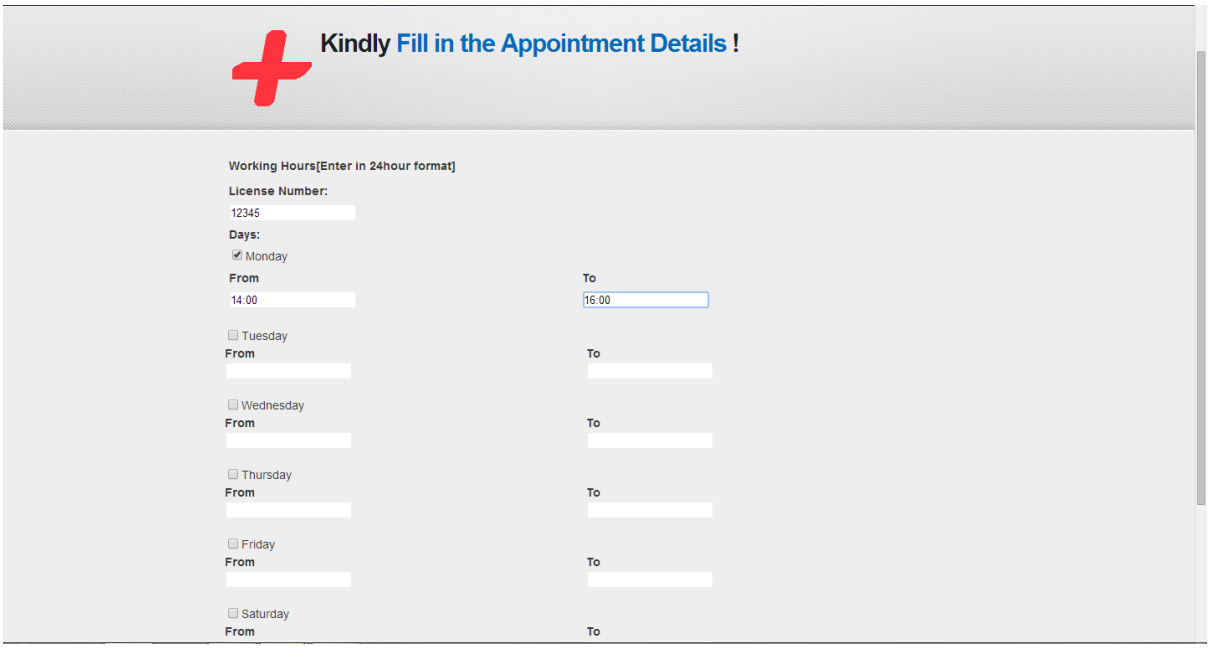

Fig A.10 Doctor providing his appointment slots

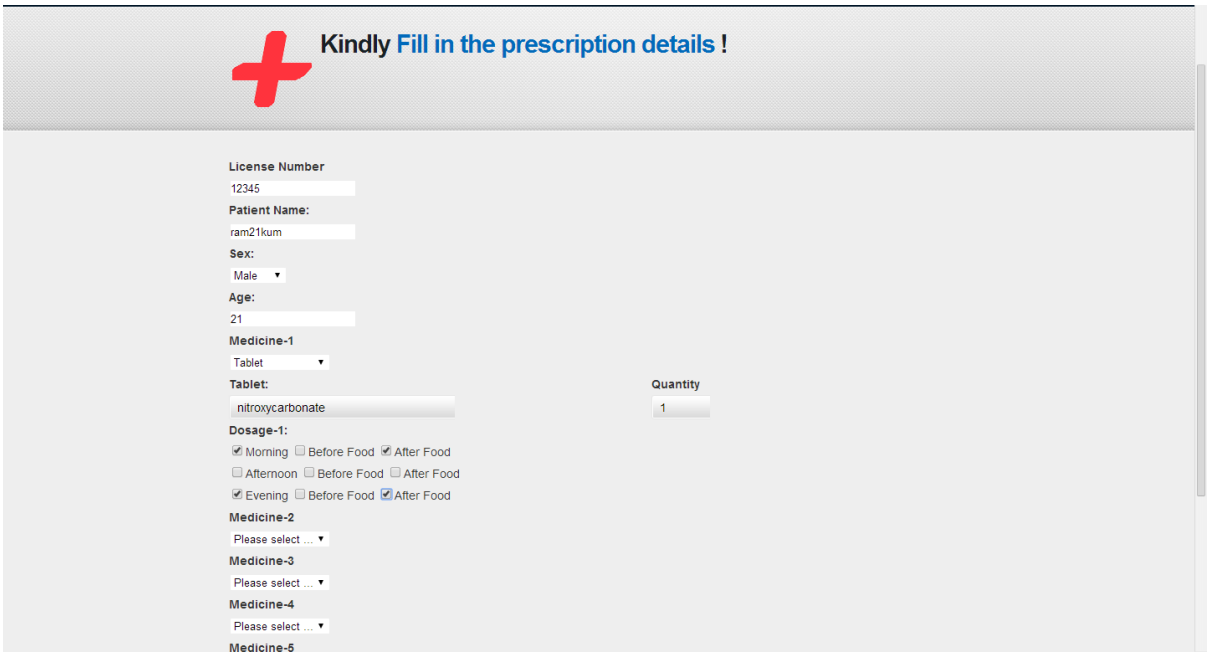

Fig A.11 Doctor providing the prescription to his patient

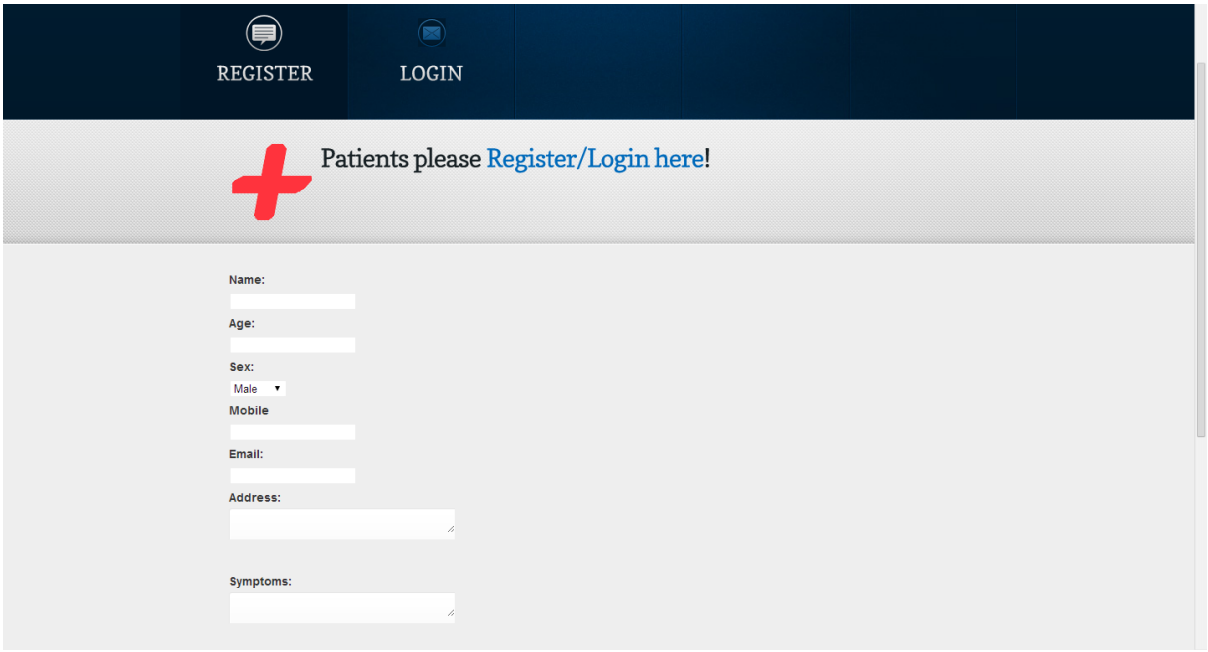

Fig A.12 Patient's registration page

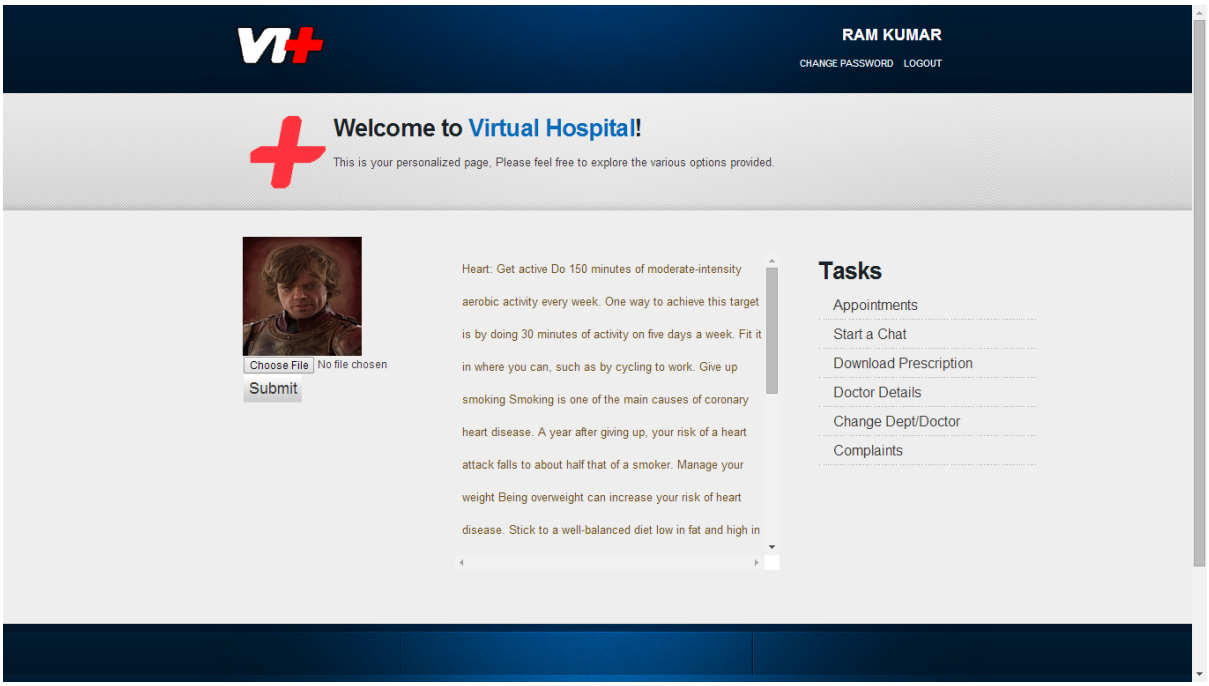

Fig A.13 Patient's profile page

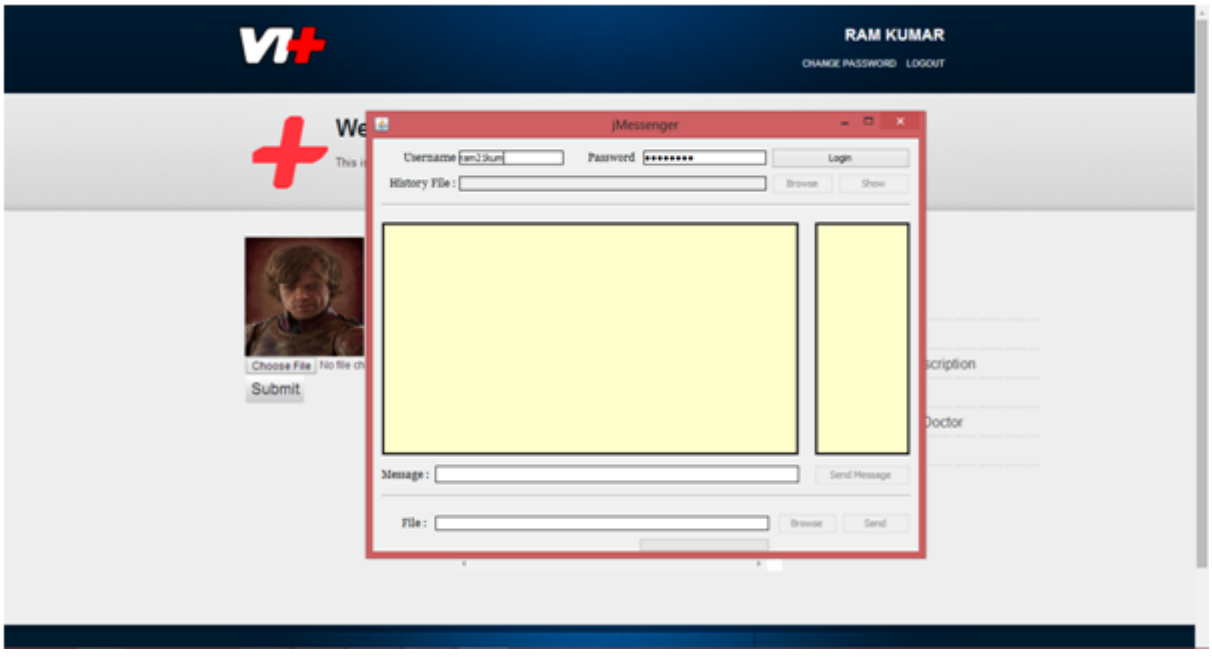

Fig A.14 Patient logging in to chat

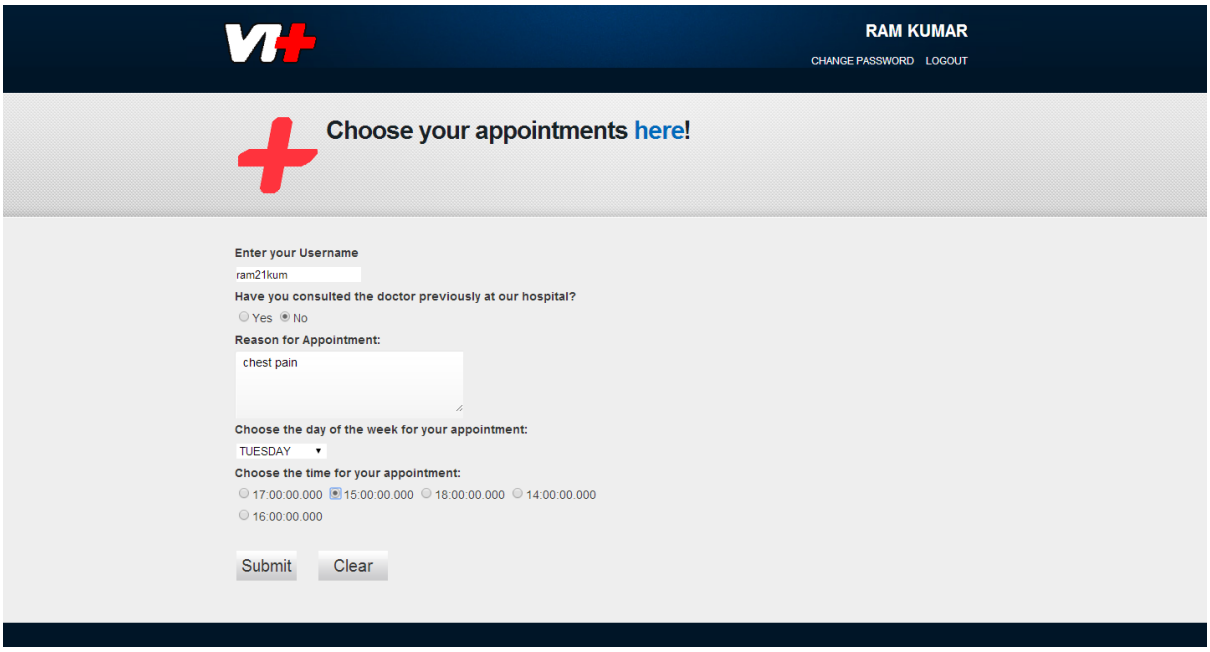

Fig A.15 Patient choosing his required appointment

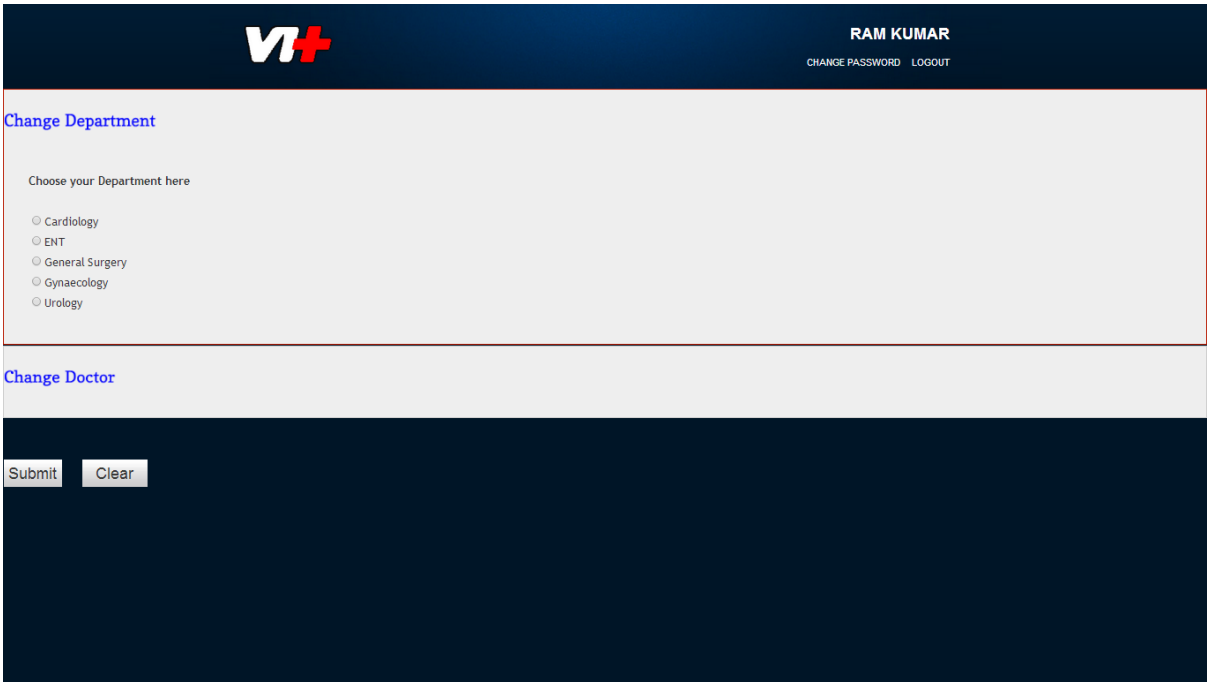

Fig A.16 Patient changing his consulting department

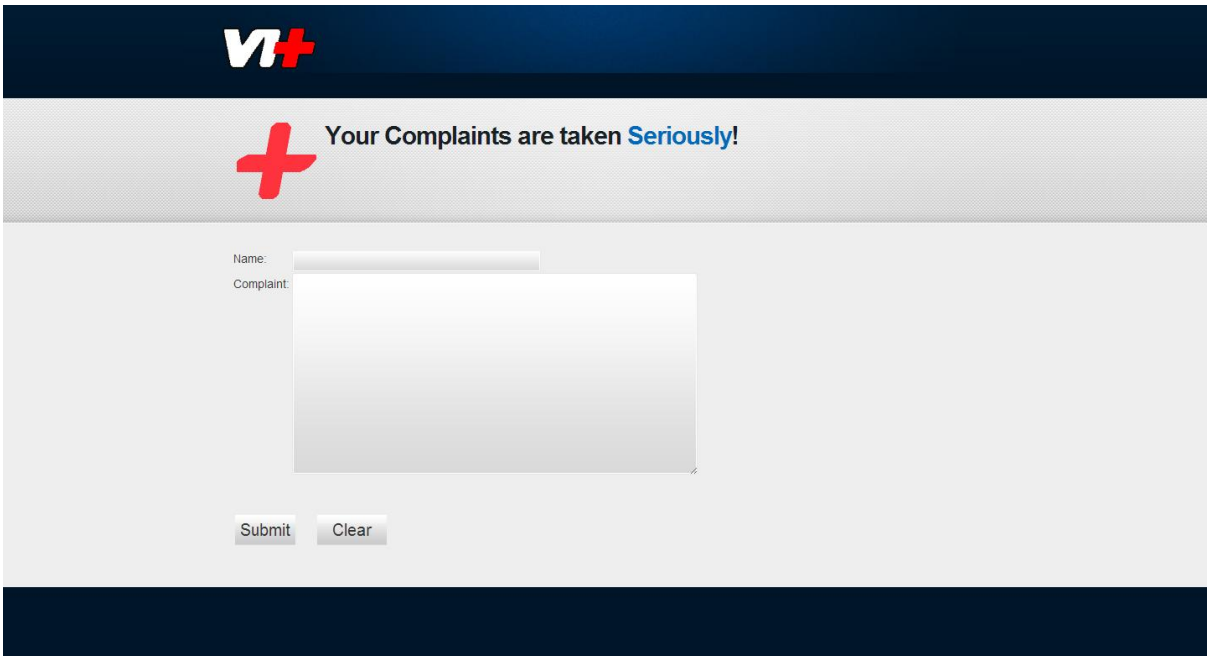

Fig A.17 Patient's / Doctor's complaint page

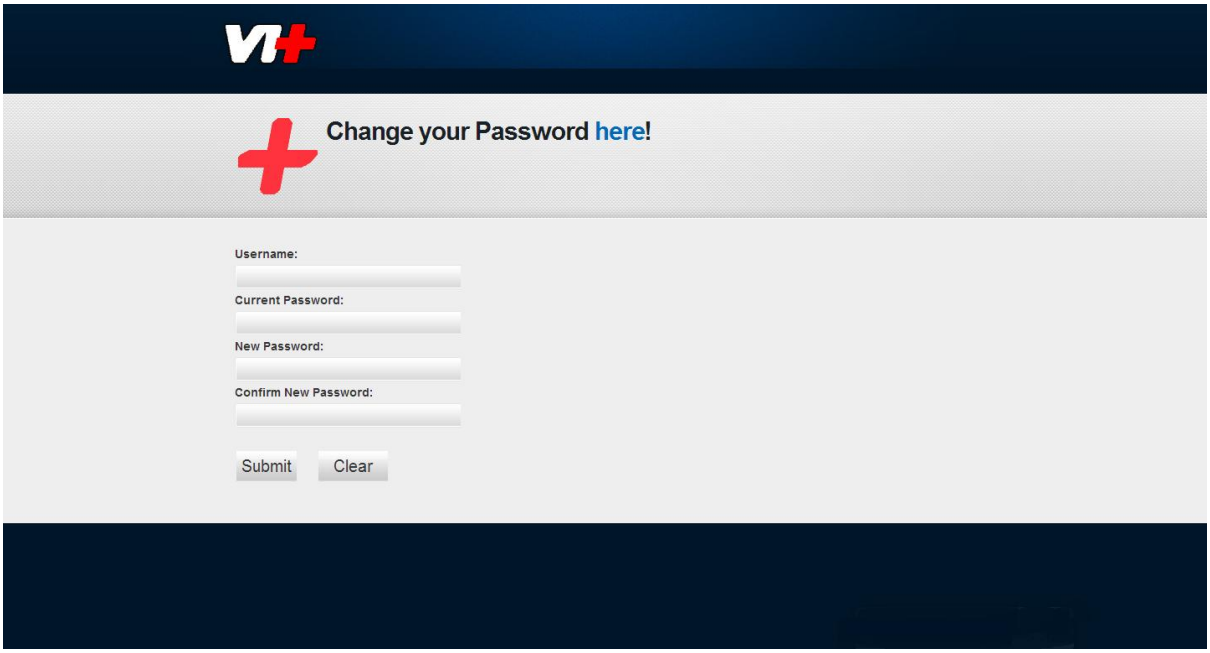

Fig A.18 Patient's / Doctor's 'change password' page