
RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

# Healthcare Management System

**Submitted by**

**Iraiyanbu S T (231801061)**

**CS23333-Object oriented Programming using java**

**Department of Artificial Intelligence and Data Science**

**Rajalakshmi Engineering College, Thandalam**

**Nov 2024**

# CHAPTER 1

## INTRODUCTION

Healthcare is a field in which accurate record keeping and communication are critical and yet in which the use of computing and networking technology lags behind other fields. Healthcare professionals and patients are often uncomfortable with computers, and feel that computers are not central to their healthcare mission, even though they agree that accurate record keeping and communication are essential to good healthcare. In current healthcare, information is conveyed from one healthcare professional to another through paper notes or personal communication. For example, in the United States, electronic communication between physicians and pharmacists is not typically employed but, rather, the physician writes a prescription on paper and gives it to the patient. The patient carries the prescription to the pharmacy, waits in line to give it to a pharmacist, and waits for the pharmacist to fill the prescription. To improve this process, the prescriptions could be communicated electronically from the physician to the pharmacist, and the human computer interfaces for the physicians, nurses, pharmacists and other healthcare professionals could be voice enabled.

According to Carmen Catizone of the National Association of Boards of Pharmacy, there are as many as 7,000 deaths from incorrect prescriptions in the United States each year. A Washington Post article indicates that as many as 5% of the 3 billion prescriptions filled each year are incorrect. These numbers indicate that there is an urgent need to reduce the errors in healthcare.

## 1.1 PURPOSE

This paper describes a distributed e-healthcare system that uses the Service- Oriented Architecture as a means of designing, implementing, and managing healthcare services.

## 1.2 PROJECT SCOPE

Effective and timely communication between patients, physicians, nurses, pharmacists, and other healthcare professionals is vital to good healthcare. Current communication mechanisms, based largely on paper records and prescriptions, are old-fashioned, inefficient, and unreliable. When multiple healthcare professionals and facilities are involved in providing healthcare for a patient, the healthcare services provided aren't often coordinated. Typically, a physician writes a prescription on paper and gives it to the patient. The patient carries the prescription to the pharmacy, waits in line to hand the prescription to the pharmacist, and waits for the pharmacist to fill the prescription. The pharmacist might be unable to read the physician's handwriting; the patient could modify or forge the prescription; or the physician might be unaware of medications prescribed by other physicians. These and other problems indicate the need to improve the quality of healthcare.

A distributed electronic healthcare system based on the service-oriented architecture (SOA) can address some of these issues and problems. We developed a distributed e-healthcare system for use by physicians, nurses, pharmacists, and other professionals, as well as by patients and medical devices used to monitor patients. Multimedia input and output—with text, images, and speech—make the system less computer- like and more attractive to users who aren't computer-oriented.

## 1.3 PRODUCT PERSPECTIVE

Our prototype distributed e-healthcare system uses SOA to enforce basic software architecture principles and provide interoperability between different computing platforms and applications that communicate with each other. Although our distributed e-healthcare system provides user-friendly interfaces for busy healthcare professionals and patients, security and privacy are particularly important in this area, so we designed the prototype with security and privacy in mind. The system authenticates users and logs session information and attaches resources to the resource creator, so that only privileged users can view or modify the data.

# CHAPTER 2

## LITERATURE REVIEW

### [1] E-healthcare: an analysis of key themes in research

**Avinandan Mukherjee, John McGinnis –International Journal of Pharmaceutical and Healthcare Marketing 2007.**

Purpose – Healthcare is among the fastest-growing sectors in both developed and emerging economies. E-healthcare is contributing to the explosive growth within this industry by utilizing the internet and all its capabilities to support its stakeholders with information searches and communication processes. The purpose of this paper is to present the state-of-the-art and to identify key themes in research on e-healthcare.

Design/methodology/approach – A review of the literature in the marketing and management of e-healthcare was conducted to determine the major themes pertinent to e-healthcare research as well as the commonalities and differences within these themes.

Findings – Based on the literature review, the five major themes of e-healthcare research identified are: cost savings; virtual networking; electronic medical records; source credibility and privacy concerns; and physician-patient relationships.

Originality/value – Based on these major themes, managerial implications for e-healthcare are formulated. Suggestions are offered to facilitate healthcare service organizations' attempts to further implement and properly utilize e-healthcare in their facilities. These propositions will also help these stakeholders develop and streamline their e-healthcare processes already in use. E-healthcare systems enable firms to

improve efficiency, to reduce costs, and to facilitate the coordination of care across multiple facilities.

**[2] Role Based Access Control Models for E-Healthcare Systems**

**Jarrod Williams Florida A&M University -Department of Computer and Information Sciences**

E-healthcare is the use of web-based systems to share and deliver information across the internet. With this ability, privacy and security must be maintained according to the Health Insurance Portability and Accountability Act (HIPAA) standards [1]. The reasonable approach to developing a system that can meet these requirements is a system that utilizes role-based models. Role-based access control (RBAC) is important because personnel could change but the position and access to the safe information keeps stable. With a role-based model it becomes easier to maintain access control, assign privileges, and personnel to the appropriate role .

Implementation is based off the security policy, which is a critical component of any system because it defines which roles or people have access to what information. An extensible markup language (XML) is used to enforce this policy because it is a web-based technology that is good for data transportation and security. Within this research project, we are able to give an overview for the state-of-art of secure e-healthcare system, and better understand a way of implementing a secure e-healthcare system that meets HIPAA standards and can share information to patients and healthcare facilities via the web service .

[3] **A Distributed e-Healthcare System Based on the Service Oriented Architecture Kart, F. Gengxin Miao Moser, L.E. Melliar-Smith, P.M. Dept. of Electr. & Comput. Eng., Univ. of California, Santa Barbara, CA;**

Large-scale distributed systems, such as e-healthcare systems, are difficult to develop due to their complex and decentralized nature. The service oriented architecture facilitates the development of such systems by supporting modular design, application integration and interoperation, and software reuse. With open standards, such as XML, SOAP, WSDL and UDDI, the service oriented architecture supports interoperability between services operating on different platforms and between applications implemented in different programming languages. In this paper we describe a distributed e-healthcare system that uses the service oriented architecture as a basis for designing, implementing, deploying, invoking and managing healthcare services. The e-healthcare system that we have developed provides support for physicians, nurses, pharmacists and other healthcare professionals, as well as for patients and medical devices used to monitor patients. Multi-media input and output, with text, images and speech, make the system more user friendly than existing e-healthcare systems

# CHAPTER 3

## SYSTEM ANALYSIS

### EXISTING SYSTEM

- In existing e-Healthcare systems has focused on record keeping and databases.
- It has also focused on access and security for recording and communicating healthcare information
- Human errors are more by exploiting electronic communication and record keeping.

### PROPOSED SYSTEM

- This system aims to reduce human errors by exploiting electronic communication and record keeping, and by providing user-friendly input and output capabilities.
- Modern technology is use to expose the functionality of our e-healthcare system as Web Services based on the Service Oriented Architecture, so that both humans and applications can use the services provided.
- It provides services that involve patients, physicians, nurses and pharmacists as well as medical monitoring devices, whereas their framework focuses specifically on the use of medical monitoring devices.

# CHAPTER 4

## SOFTWARE REQUIREMENT SPECIFICATION:

### 4.1 OPERATING ENVIRONMENT

#### 4.1.1 HARDWARE ENVIRONMENT:

- Hard Disk: 10GB and Above
- RAM: 512MB and Above
- Processor: Pentium III and Above

#### 4.1.2 SOFTWARE ENVIRONMENT:

- Windows Operating System 2000 and Above
- JDK 1.5
- Web Browser – Internet Explorer
- Apache Tomcat 5.5 Server
- Derby Database
- JSP and Servlets
- Axis Web Service
- Atom/RSS
- J2ME
- XML

**DERBY:**

Apache Derby, an Apache DB Sub Project, is a relational database implemented in Java. Its footprint is so small you can easily embed it in any Java-based solution. In addition to its embedded framework, Derby supports a more familiar client/server framework with the Derby Network Server. This tutorial introduces Derby's basic features and walks you through using both frameworks; first the embedded framework using the Derby Embedded JDBC driver, then the Network Server framework using the Derby Network Client JDBC driver.

**APACHE AXIS:**

**Apache Axis** is an open source, XML based Web Services framework. It consists of a Java and a C++ implementation of the SOAP server, and various utilities and APIs for generating and deploying Web service applications. Using Apache Axis, developers can create interoperable, distributed computing applications. Axis is developed under the auspices of the Apache Software.

**APACHE TOMCAT:**

Apache Tomcat is an implementation of the Java Servlet and Java Server Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the Java Community Process .Apache Tomcat is developed in an open and participatory environment and released under the Apache Software Liscense. Apache Tomcat is intended to be a collaboration of the best-of-breed developers from around the world.

**RSS:**

**RSS** is a family of Web feed formats used to publish frequently updated works—such as blog entries, news headlines, audio, and video—in a standardized format. An RSS document (which is called a "feed", "web feed", or "channel") includes full or summarized text, plus matadata such as publishing dates and authorship. Web feeds benefit publishers by letting them syndicate content automatically. They benefit readers who want to subscribe to timely updates from favored websites or to aggregate feeds from many sites into one place. RSS feeds can be read using software called an "RSS reader", "feed reader", or "aggregator", which can be web based ordesktop based. A standardized XML file format allows the information to be published once and viewed by many different programs. The user subscribes to a feed by entering the feed's URI(often referred to informally as a "URL", although technically, those two terms are not exactly synonumous) into the reader or by clicking an RSS icon in a browser that initiates the subscription process. The RSS reader checks the user's subscribed feeds regularly for new work, downloads any updates that it finds, and provides a user interface to monitor and read the feeds.

**ATOM:**

The name **Atom** applies to a pair of related standards. The *Atom Syndication Format* is an XML language used for Web feeds, while the *Atom Publishing Protocol* (*AtomPub* or *APP*) is a simple HTTP-based protocol for creating and updating web resources.Web feeds allow software programs to check for updates published on a web site. To provide a web feed, a site owner may use specialized software (such as a content management system) that publishes a list (or "feed") of recent articles or content in a standardized, machine-readable format. The feed can then be downloaded by web sites that syndicate content from the feed, or by feed reader programs that allow Internet users to subscribe to feeds and view their content.A feed contains entries, which may be headlines, full-text articles, excerpts, summaries, and/or links to content on a web site, along with various metadata .The Atom format was developed as an alternative to RSS.

**J2ME:**

Java Platform, Micro Edition (Java ME) provides a robust, flexible environment for applications running on mobile and other embedded devices —mobile phones, personal digital assistants (PDAs), TV set-top boxes, and printers. Java ME includes flexible user interfaces, robust security, built-in network protocols, and support for networked and offline applications that can be downloaded dynamically. Applications based on Java ME are portable across many devices, yet leverage each device's native capabilities.

**XML:**

Java Platform, Micro Edition (Java ME) provides a robust, flexible environment for applications running on mobile and other embedded devices —mobile phones, personal digital assistants (PDAs), TV set-top boxes, and printers. Java ME includes flexible user interfaces, robust security, built-in network protocols, and support for networked and offline applications that can be downloaded dynamically. Applications based on Java ME are portable across many devices, yet leverage each device's native capabilities.

## 4.2 DESIGN AND IMPLEMENTATION CONSTRAINTS

### 4.2.1 CONSTRAINTS IN ANALYSIS

- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts
- Constraints as a Separate Concept
- Constraints Implied by the Model Structure

### 4.2.2 CONSTRAINTS IN DESIGN

- Determination of the Involved Classes
- Determination of the Involved Objects
- Determination of the Require Clauses
- Global actions and Constraint Realization

## 4.3 ASSUMPTIONS AND DEPENDENCIES

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation.

## 4.4 SYSTEM FEATURES

A distributed e-healthcare system can help solve this conundrum. Medical monitoring devices worn by the patient, and frequent electronic communication between the patient and a nurse, can ensure that the prescribed treatment is being followed and that the patient is making good progress.

## 4.5.EXTERNAL INTERFACE REQUIREMENTS

### 4.5.1User Interfaces

The User Interface is Web Browsers like Internet Explorer, Mozilla Firefox, Opera, etc.

### 4.5.2 SERVER SIDE

The web application will be hosted on one of the department's Linux servers and connecting to one of the school Oracle Database server. The web server is listening on the web standard port, port 80.

### 4.5.3 CLIENT SIDE

The system is a web based application; clients are requiring using a modern web browser such as Mozilla Firebox 1.5, Internet Explorer 6 and Enable Cookies. The computer must have an Internet connection in order to be able to access the system.

## 4.6 SOFTWARE INTERFACES

### 4.6.1 SERVER SIDE

The system already has the required software to host a Java web application. An Apache Web server will accept all requests from the client and forward specific requests to Tomcat 5.5 Servlet Container with J2EE 5.0

### 4.6.2 CLIENT SIDE

*An OS is capable of running a modern web browser which supports HTML version 3.2 or higher.*

## 4.7 COMMUNICATION INTERFACES

The HTTP protocol will be used to facilitate communications between the client and server.

## 4.8 OTHER NON FUNCTIONAL REQUIREMENTS

### 4.8.1 PERFORMANCE REQUIREMENTS

We expect that most pages be returned in 1 second or less to users for whom bandwidth isn't an issue. We will attempt to keep pages to a reasonable size, however, in order to ensure that users with slow connections (e.g. 56K) do not experience a delay of more than 3 seconds for most pages. One way to do this is to limit queries so they do not return more than 20 events per page. For larger pages, or pages that return search results from the database, we expect page returns of 3 seconds for users with fast connections and 5 seconds for users with slower connections to be reasonable.

### 4.8.2 SAFETY REQUIREMENTS

The software may be safety-critical. If so, there are issues associated with its integrity level. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level. Systems with different requirements for safety levels must be separated. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

### 4.8.3 SECURITY REQUIREMENTS

Do not block the some available ports through the windows firewall

Two machines should be connected with LAN setting.

### 4.8.4 SOFTWARE QUALITY ATTRIBUTES

**Functionality**: are the required functions available, including interoperability and security.

**Reliability:** maturity, fault tolerance and recoverability.

**Usability:** how easy it is to understand, learn and operate the software system.

**Efficiency:** performance and resource behavior.

**Maintainability:** how easy is it to modify the software.

**Portability:** can the software easily be transferred to another environment, including install ability.

## 4.9 PROCESS DESCRIPTION

**Clinical Module:**

**Input:**

- Physician Login Page.
- Enter the username and password.

**Process:**

- **Clinical staff services___**
- Add patient data / notes
- Forwards the prescription to the patient and pharmacy.

- Make referrals to other physician
- **Patient services:**
- Make appointments
- Deliver notification

**Output:**

- Select the menu bar.
- Patient Diagnosis data.
- Patient appointment

**Pharmacy module:**

**Input:**

Pharmacy Login page.

**Process:**

**Pharmacy Services:**

Receive / view / fill prescriptions

**Patient Services**

- Enable renewals of prescriptions
- Provide notification

**Output:**

Menu bar (delivery date, today delivery, medicine details). Select any one from menu bar.

**Atom/RSS module:**

**Input:**

This module will run only in the Exception case and input is not needed (Web services not running).

**Process:**

The contents are converted as XML format and stored in local disk.

**Output:**

Show the Patient diagnosis data for this module.

**Speech module:**

**Input:**Text to the diagnosis data

**Output:**Text to speech only.

# CHAPTER 5

## SYSTEM DESIGN SPECIFICATION

### 5.1 SYSTEM ARCHITECTURE

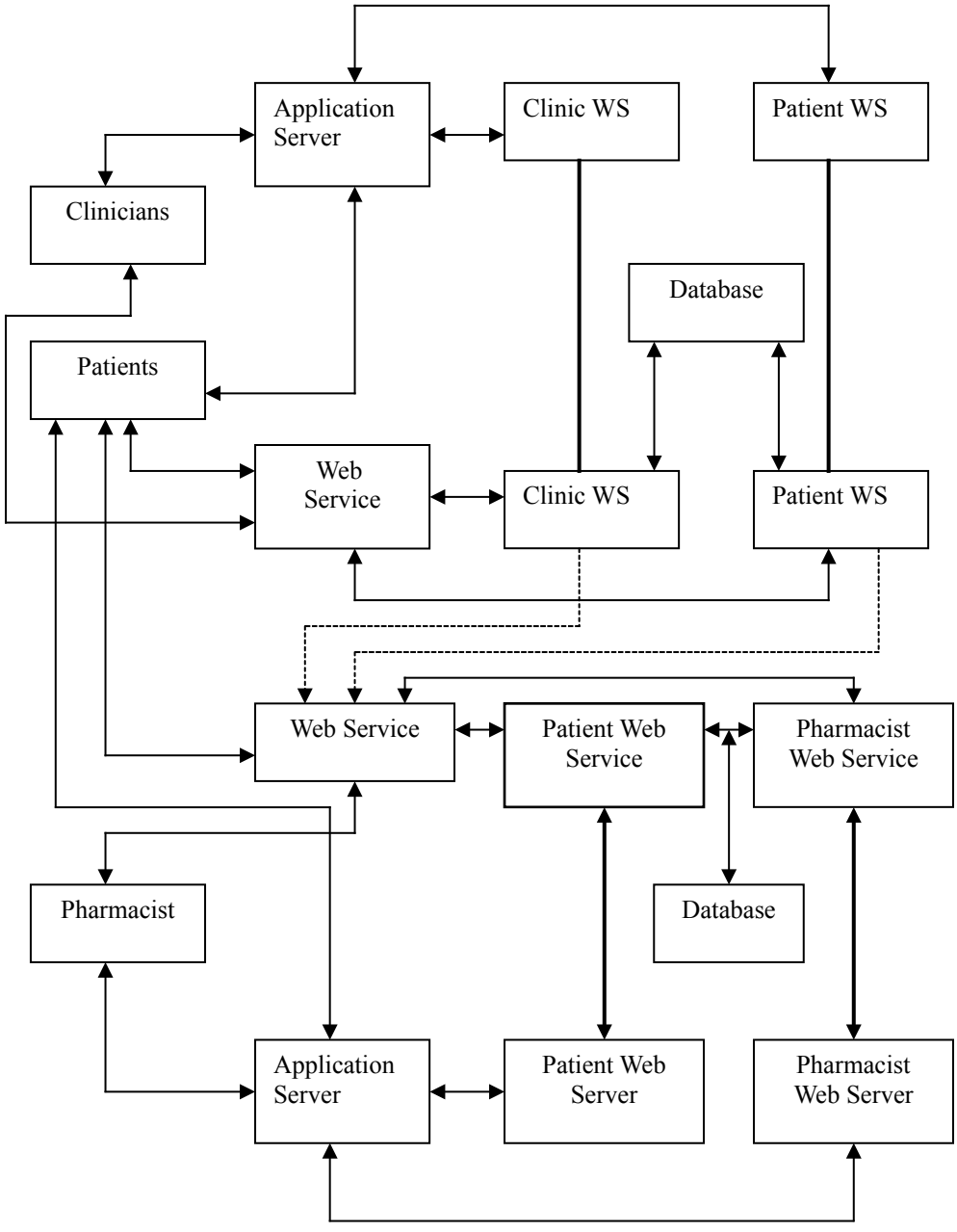


**Fig 5.1 COMPLETE ARCHITECTURE OF SOA**

## 5.2. MODULES

### CLINIC MODULE

The Clinic module exposes two interfaces, a Web Server and a Web Service, for the clinic staff, the patients and the medical monitoring devices. The Web Server interface is intended for users who prefer to use a Web browser to access the healthcare services. Humans or devices to communicate with the e-healthcare system can use the Web Service interface. The Web Server uses the Web Services to access the data. The Clinic module provides support for routine activities of the physician. It maintains information, such as the physician's appointments for a specific day/week, the patients that s/he has examined, notes related to the patients, etc. The Clinic module sends prescriptions from the physician to the desired pharmacies using the Web Service provided.

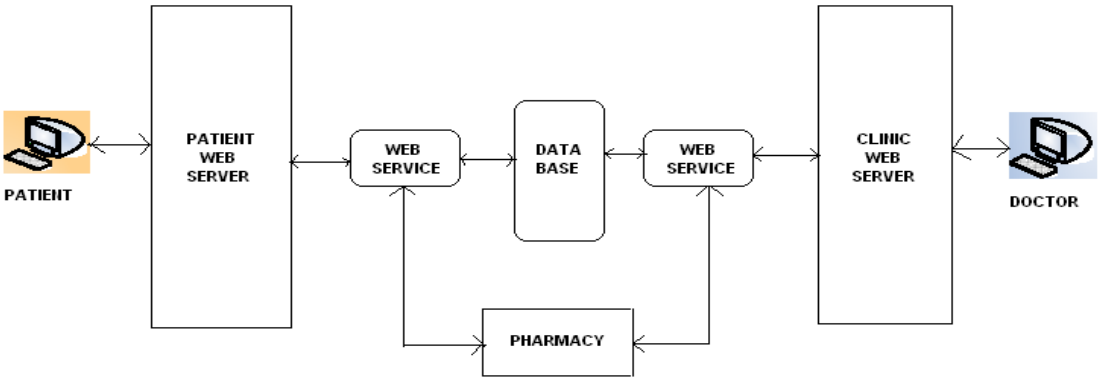


**Fig 5.2.1 CLINIC MODULE**

| PATIENTID | PASSWORD | NAME | SEX | DOB | ADDRESS | MOBILE | LANDLINE |
|---|---|---|---|---|---|---|---|
| PAT4654 | ASDF | MURALI | MALE | 15/08/87 | FGDHJ | 984567321 | 4567221 |
| PAT4655 | ABCD | MAHESH | MALE | 13/01/87 | GHDFG | 9884108999 | 26562445 |

**Table 5.2.1 PATIENT DETAILS**

| PHYSICIANID | APPOINTMENTDATE | TIME | PATIENTID | STATUS | NOON |
|---|---|---|---|---|---|
| PHY4655 | 13/032008 | 9 | PAT4655 | OPEN | AM |
| PHY4663 | 14/03/2008 | 4 | PAT4657 | OPEN | PM |

**Table 5.2.2PHYSICIAN APPOINTMENTS**

| PATIENTID | NO | SYMPTOMS | PREVIOUS MEDICINES | MEDICAL DIAGNOSIS | PHYSICIAN ID | DIAGNOSIS DATE | STATUS |
|---|---|---|---|---|---|---|---|
| PAT4654 | 1 | FEVER | CROCIN | ANACIN | PHY4655 | 13/03/2008 | CLOSED |
| PAT4658 | 2 | HEADACHE | CROCIN | SARIDON | PHY4658 | 14/03/2008 | CLOSED |

**Table 5.2.3 MEDICAL DIAGNOSIS**

| ID | PASSWORD | NAME | SEX | DOB | ADDRESS | PHONE | QUALIFICATON |
|---|---|---|---|---|---|---|---|
| PHY4655 | ASDF | MURALI | MALE | 15/08/1987 | GHFGH | 76767 | JGJHHJG |
| PHY4656 | GFFS | MAHESH | MALE | 13/01/1987 | HJKHJK | 65675 | HGJG |

**Table 5.2.4 PHYSICIAN DETAILS**

## PHARMACY MODULE

The Pharmacy module exposes Web Server and Web Service interfaces. The Web Server interface allows the users to access the e-healthcare system at the pharmacy using a browser. The Web Service interface provides access for applications deployed at the pharmacy and can also be used by humans and devices. The Pharmacy module provides services to the pharmacist, patients and devices used at the pharmacy. The Pharmacy module keeps a record of the patient's prescriptions for the pharmacist's and the patient's reference. When the physician submits a new prescription to the pharmacy, the Clinic module at the physician's office communicates with the Pharmacy module at the pharmacy. The pharmacist can view the outstanding prescriptions for the patients, as they are received from the physicians. The Pharmacy module updates the status of the prescriptions as the pharmacist fills them. The patient can determine, via the Web Server or Web Service, whether a prescription has been filled and is ready for pick up or delivery.

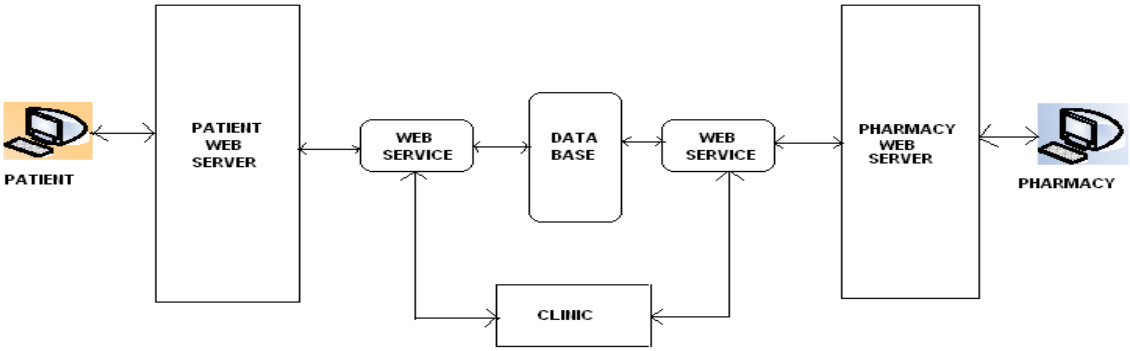



**Fig 5.2.2 PHARMACY MODULE**

| PATIENTID | PHYSICIANID | PHARMACYNAME | MEDICINE | DOSAGE | TIMESOFDAY | DATE |
|---|---|---|---|---|---|---|
| PAT4655 | PHY4655 | ABC | CROCIN | 6 | 2 | 13/03/07 |
| PAT4654 | PHY4662 | XYZ | ANACIN | 6 | 2 | 24/03/07 |

**Table 5.2.5 PHYSICIAN PRESCRIPTION**

## ATOM/RSS MODULE

Atom/RSS are syndication technologies, based on XML, that enable the sharing and communication of information between heterogeneous platforms by making the information self-describing. They allow a publisher to make information available to consumers on the Web, which retrieve that information subsequently. The information is delivered from the publisher to the consumer as an XML file, called an Atom/RSS feed.

This project develops a Consistent Data Replication (CDR) and Reliable Data Distribution (RDD) infrastructure that replicates information from one

computer to another using Atom/RSS feeds. It uses this infrastructure to synchronize information on the physician's desktop or server computer with that on his/her PDA, allowing the physician to view that information when it is offline. At the start of the day, our software on the PDA retrieves the necessary updates from the Clinic Web Service on the desktop or server computer via a wired or wireless network. Any modifications to the information on the PDA are stored locally on the PDA. At the end of the day, our software on the PDA generates an update feed for the Clinic Web Service on the desktop or server computer to read.

## SPEECH MODULE

Natural Voices provides a simple and efficient way to produce natural sounding device-to-human voice interaction. It can accurately and naturally pronounce words, and speak in sentences that are clear and easy-to understand, without the feeling that a computer is talking to the human. Natural Voices supports many languages, male and female voices, and the SAPI, Voice XML and JSAPI interface standards. Using Natural Voices, created text to- speech software for the prototype device, which runs in the background and accepts messages in Voice XML format.

## DATA FLOW DIAGRAM

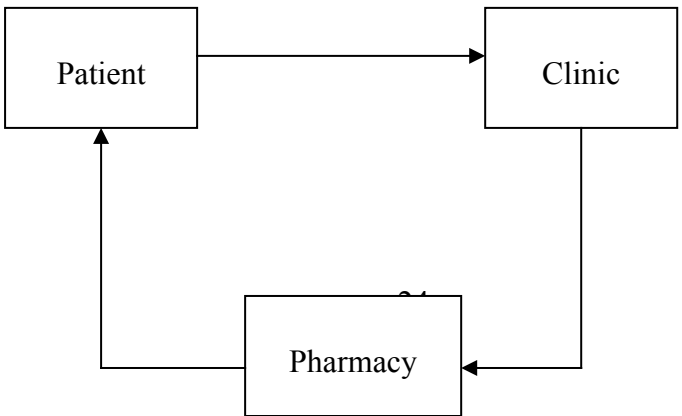


**Fig 5.2.1- LEVEL 0:**

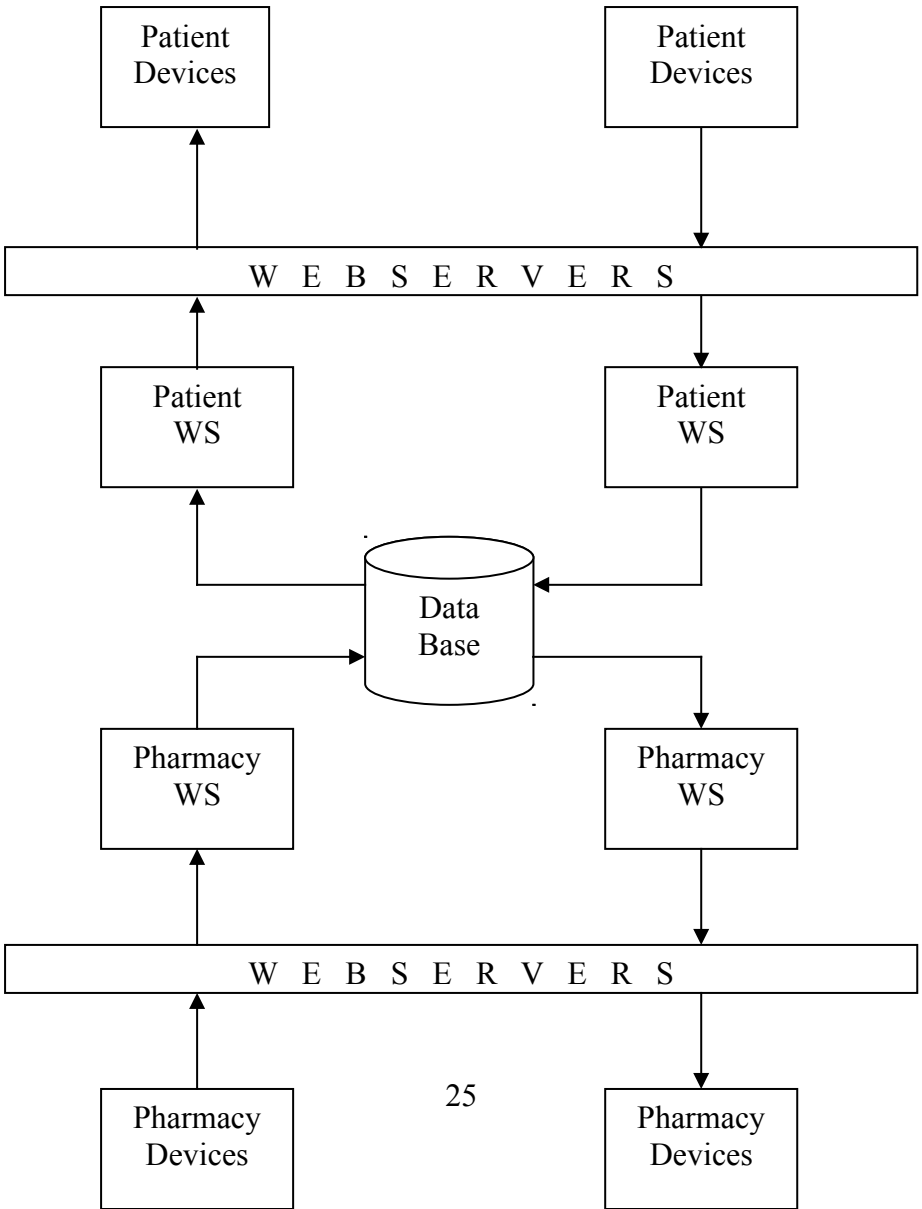


**Fig 5.2.2 DATA FLOW DIAGRAM-LEVEL1:**

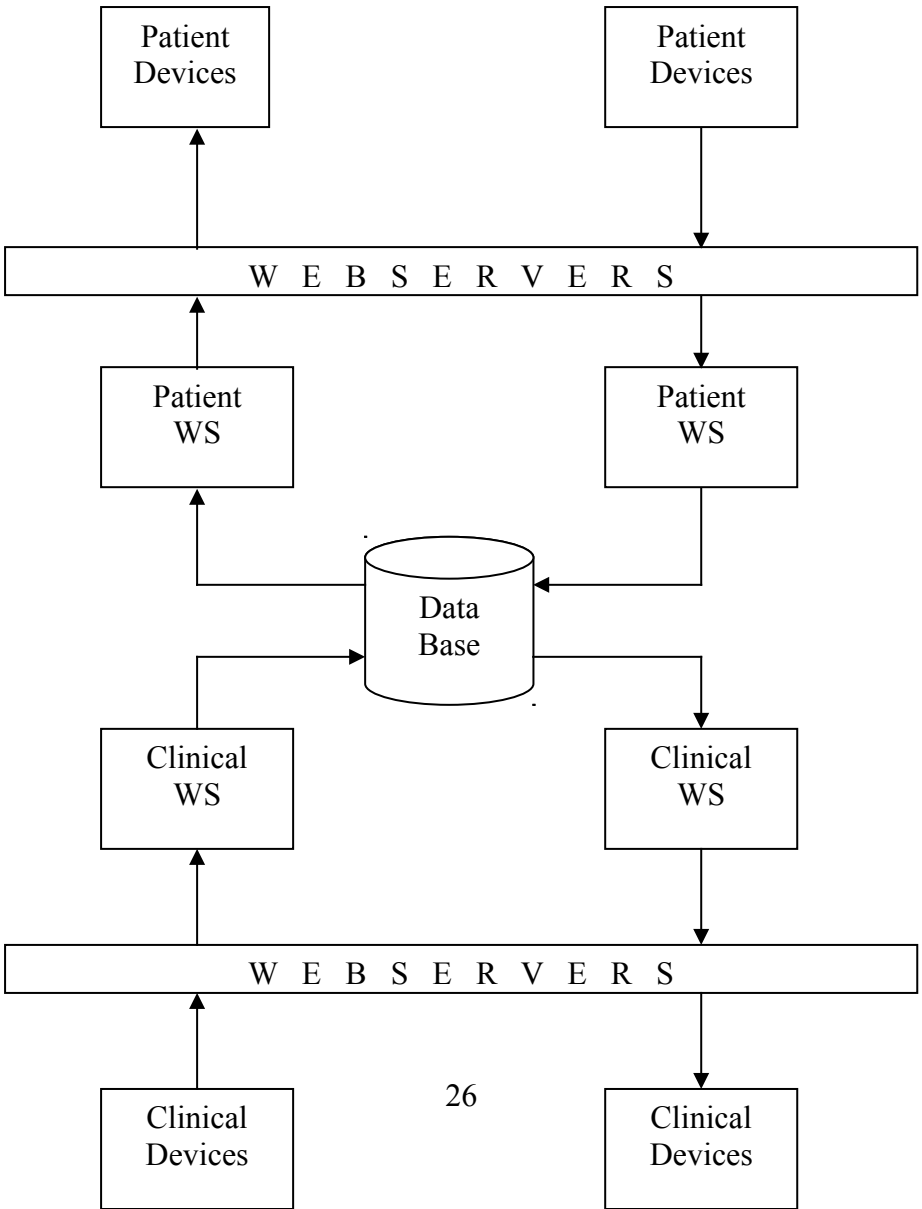


**Fig 5.2.3 DATA FLOW DIAGRAM-LEVEL 2**

# CHAPTER 6

## SYSTEM IMPLEMENTATION

System implementation is the important stage of project when the theoretical design is tuned into practical system. The main stages in the implementation are as follows:

- Planning
- Training
- System testing and
- Changeover Planning

Planning is the first task in the system implementation. Planning means deciding on the method and the time scale to be adopted. At the time of implementation of any system people from different departments and system analysis involve. They are confirmed to practical problem of

controlling various activities of people outside their own data processing departments. The line managers controlled through an implementation coordinating committee. The committee considers ideas, problems and complaints of user department, it must also consider:

- The implication of system environment;
- Self selection and allocation for implementation tasks;
- Consultation with unions and resources available;
- Standby facilities and channels of communication;

# CHAPTER 7

## SYSTEM TESTING

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn't work. Testing is the process of executing a program with the intent of finding errors.

### 7.1TESTING OBJECTIVES

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

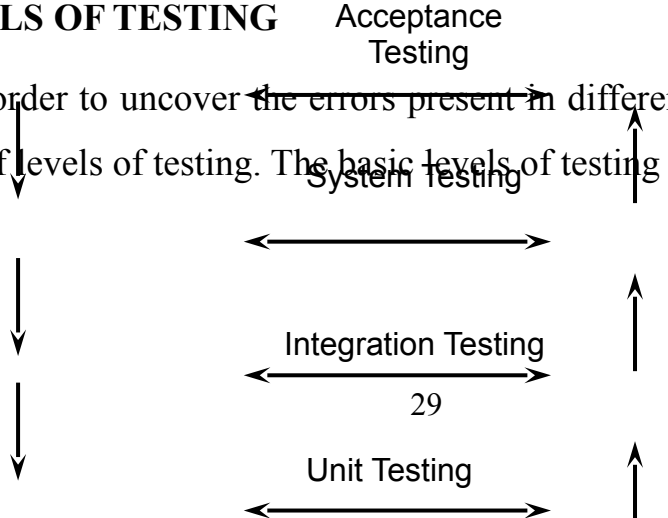
- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.

**PROCESS:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement

## 7.2LEVELS OF TESTING

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are as shown below…



Client Needs

Requirements

Design

Code

## 7.3TESTING STRATERGIES:

### UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**INTEGRATION TESTING:**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**FUNCTIONAL TESTING:**

Functional tests provide a systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output :identified classes of application outputs.
Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify

Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEM TESTING:**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**WHITE BOX TESTING:**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level .

**BLACK BOX TESTING:**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested . Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see”

into it. The test provides inputs and responds to outputs without considering how the software works.

**UNIT TESTING:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**TEST STRATEGY AND APPROACH**

Field testing will be performed manually and functional tests will be written in detail.

**TEST OBJECTIVES**

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.
- Features to be tested.
- Verify that the entries are of the correct format.
- No duplicate entries should be allowed.
- All links should take the user to the correct page.

**INTEGRATION TESTING:**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

**ACCEPTANCE TESTING:**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

# CHAPTER 8

## CONCLUSION & FUTURE ENHANCEMENT

Healthcare is a field where information has to be maintained properly. This field needs to create a user-friendly system, which guides users at all steps they need to perform in it. The information provided by the users must be kept secured, as the healthcare information is very much confidential. The prescriptions for a certain patient are forwarded electronically to the pharmacy. This avoids the unnecessary time taken by the patient to carry the prescription to the pharmacy.

Thus our healthcare is much secured providing authentication to the user. Our project guides the user to the action they need to perform. Our project is user-friendlier than all other e-healthcare systems with voice messages and blue tooth enhancement.

Our e-healthcare system currently focuses on the relationships between patients, physicians and pharmacists. We plan to extend the system to other healthcare facilities and professionals, such as laboratory technicians who perform and report tests and analyses requested by physicians. We also plan to investigate whether our Clinic and Pharmacy modules can be interfaced to applications supplied by pharmaceutical companies that provide information on medications and dosages and warn of interactions between medications. In addition, we plan to investigate drug delivery devices, such as e-pillboxes, that prompt and monitor the regular and timely consumption of medications.

# CHAPTER 9

## APPENDICES

### APPENDIX 1

### 9.1 SOURCE CODE

**PHYSICIAN SIGN UP PROGRAM:**

```
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.util.Vector;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.rpc.ParameterMode;
import javax.xml.rpc.ServiceException;
import javax.xml.namespace.QName;
import java.io.FileInputStream;
import java.util.Properties;
import java.util.Vector;
import java.util.Enumeration;
import org.apache.derby.jdbc.EmbeddedDriver;
import org.apache.derby.jdbc.ClientDriver;

public class SignUp
{
        public String check(Vector details)
        {

                String patientId = null;
                String str =null;
```

```
String str1 =null;
String str2 =null;
System.out.println("vector:"+details);
String password=details.get(0).toString();
String patientname=details.get(1).toString();
String sex=details.get(2).toString();
String dateofbirth=details.get(3).toString();
String address=details.get(4).toString();
String street=details.get(5).toString();
String state=details.get(6).toString();
String pincode=details.get(7).toString();
String mobile=details.get(8).toString();
String landline=details.get(9).toString();
String occupation=details.get(10).toString();
String email=details.get(11).toString();
System.out.println("password:"+password + "patientname"
+patientname);

try
{


Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
String databaseURL =
"jdbc:derby://murali1:1234/EHealth";
Properties properties = new Properties();
properties.put("user","app");
properties.put("password","app");
```

```
properties.put("retreiveMessagesFromServerOnGetMessage", "true");
                Connection con = DriverManager.getConnection(databaseURL, properties);
                Statement st=con.createStatement();
                ResultSet rs = st.executeQuery("select physicianId from physician_personaldetails");
                ResultSetMetaData rsmd = rs.getMetaData();
                int numberOfColumns = rsmd.getColumnCount();
                System.out.println("The Query is "+numberOfColumns);
                while(rs.next()){
                str= rs.getString(1);

                }
                str2 = str.substring(0,3);
                int i = Integer.parseInt(str.substring(3));
                i++;
                str1 = Integer.toString(i);
                patientId = str2.concat(str1);
                System.out.println("The patientId is "+patientId);
                String query = "insert into physician_personaldetails values("+"'"+patientId+"'"+","+"'"+password+"'"+","+"'"+patientname+"'"+","+"'"+sex+"'"+","+"'"+dateofbirth+"'"+","+"'"+address+"'"+","+"'"+street+"'"+","+"'"+state+"'"+","+"'"+pincode+"'"+","+"'"+mobile+"'"+","+"'"+landline+"'"+","+"'"+occupation+"'"+","+"'"+email+"'"+")";
                System.out.println("The Query is "+query);
                st.executeUpdate(query);
                rs.close();
```

```
                st.close();
                con.close();

            }
            catch(Exception e)
            {
                e.printStackTrace();
            }

            return patientId;
        }
}
```

**DOCTOR DIAGNOSIS PROGRAM**

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.util.Vector;
import java.text.SimpleDateFormat;
import java.util.Date;
```

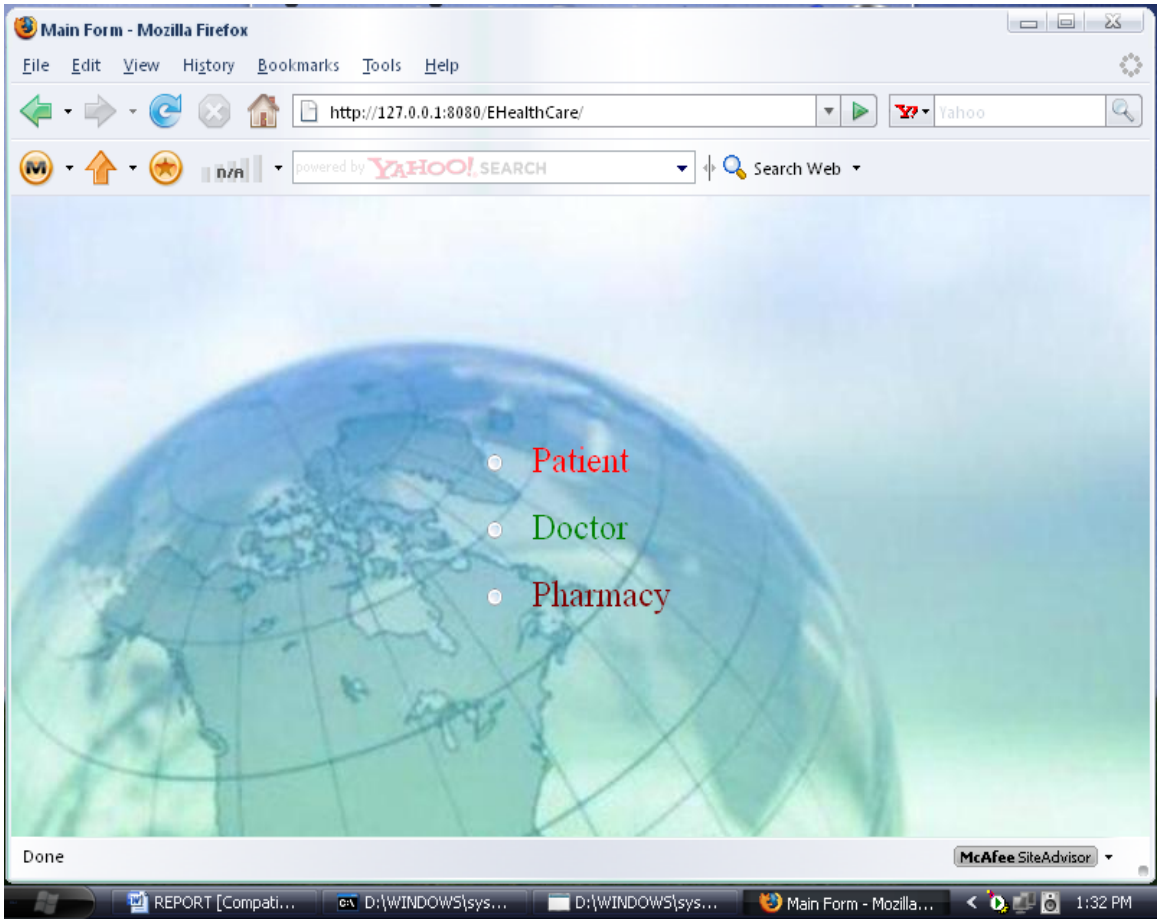
```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.rpc.ParameterMode;
import javax.xml.rpc.ServiceException;
import javax.xml.namespace.QName;
import java.io.FileInputStream;
import java.util.Properties;
import java.util.Vector;
import java.util.Enumeration;
import org.apache.derby.jdbc.EmbeddedDriver;
import org.apache.derby.jdbc.ClientDriver;

public class Ddiagnosis
{
        public boolean check(Vector details)
{
                boolean flag = false;
                System.out.println("vector:"+details);
                String Pdiagnosis = details.get(0).toString();
                //String diagnosisdate = details.get(1).toString();
                String diagnosis = details.get(1).toString();
                String patientId = details.get(2).toString();
                System.out.println("Pdiagnosis:"+diagnosis);
                String todate = null;
                try
                {
```

```
Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
                String databaseURL =
"jdbc:derby://murali1:1234/EHealth";
                Properties properties = new Properties();
                properties.put("user","app");
                properties.put("password","app");

        properties.put("retreiveMessagesFromServerOnGetMessage", "true");
                Connection con =
DriverManager.getConnection(databaseURL, properties);
                Statement st=con.createStatement();
                String status = "open";
                SimpleDateFormat format = new
SimpleDateFormat("yyyy-MM-dd");
                Date ddate = new Date();
                todate =  format.format(ddate);
                System.out.println("todate::"+todate);

                String query = "select * from patient_medicaldiagnosis
where patientId="'+patientId+"' and status="'+status+"'";

                System.out.println("The Query is "+query);
                ResultSet rs;
                rs = st.executeQuery(query);
                while(rs.next()){
                status="closed";
```

```
                String query1="update patient_medicaldiagnosis set
medicaldiagnosis='"+diagnosis+"',diagnosisdate='"+todate+"',status='"+statu
s+"' where patientId='"+patientId+"'";
                st.executeUpdate(query1);
                flag=true;
                }

                rs.close();
                st.close();
                con.close();
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            return flag;
        }
}
```

**APPENDIX 2**

**9.2 SCREENSHOTS:**




**A.2.1 MAIN PAGE FOR HEALTHCARE**