A PROJECT REPORT

ON

Authenticated Medical Documents Releasing With Privacy Protection And Release Control

Submitted to Jawaharlal Nehru Technological University, Kakinada

In the Partial Fulfillment of Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE

Submitted by

K. Keerthi Priya (21KE1A0548)

Under the guidance of

Dr .G. Ramaswamy, M.Tech,Ph.D

Professor

MALINENI LAKSHMAIAH WOMENS ENGINEERING COLLEGE

(AUTONOMOUS)

Accredited by NBA(CSE&ECE), NAAC A+, Approved by AICTE, NEW DELHI Affiliated to JNTUK

Pulladigunta (V), Vatticherukuru (Md), Guntur (Dt), AP, 522017

2021-2025

MALINENI LAKSHMAIAH WOMENS ENGINEERING COLLEGE

(AUTONOMOUS)

Accredited by NBA(CSE&ECE), NAAC A+,

Approved by AICTE, NEWDELHI, Affiliated to JNTUK
Pulladigunta (V), Vatticherukuru (Md), Guntur (Dt), AP, 522017

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that the project entitled **““Authenticated Medical Documents Releasing With Privacy Protection And Release Control”** is a bonafide work carried out by **K. Keerthi Priya (21KE1A0548)** B.Tech course Affiliated to JNTU Kakinada in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY with the specialization in COMPUTER SCIENCE, during the academic year 2023-2024

Signature of the Guide
(Dr.G.RAMASWAMY, Ph.D)

Head of the Department
(Dr.G.RAMASWAMY,Ph.D)

Viva – voice Held on

External Examiner

DECLARATION

We hereby declare that the project work entitled "Authenticated Medical Documents Releasing With Privacy Protection And Release Control" is entirely our original work carried out under the guidance of Dr.G. Ramaswamy, Professor Of Malineni Lakshmaiah Women's Engineering College, Pulladigunta, Guntur, JNTU Kakinada, A.P, India for the award of the DEGREE OF BACHELOR OF TECHNOLOGY with the specialization in Computer Science. The results carried out in this project report have not been submitted in a part or full for the award of any degree or diploma of this or any other university or institute.

Submitted By

K. Keerthi Priya (21KE1A0548)

ACKNOWLEDGEMENT

All endeavours over a long period can be successful only with the advice and support of many well-wishers. I take this opportunity to express my gratitude and appreciation to all of them.

We wish to express deep sense of gratitude to my beloved and respected guide **Dr.G.Ramaswamy, M.Tech, Ph.D,** Professor of CSE, Malineni Lakshmaiah Women's Engineering College, Guntur, for her valuable guidance, suggestions, and constant encouragement and keen interest enriched throughout the course of project work.

We extend sincere thanks to the **HOD**, **Dr.G. Ramaswamy** for her kind cooperation in completing and making this project a success.

We extend sincere thanks to the Principal, Prof. **Dr.J. APPA RAO** for his kind co-operation in completing and making this project a success.

We would like to thank the Management for their kind co-operation and for providing infrastructure facilities.

We extend thanks to all the Teaching staff of the Department of CSE for their support and encouragement during the course of my project work. I also thank the non-Teaching staff of CSE department for being helpful in many ways in successful completion of my work.

Finally, we thank all those who helped me directly or indirectly in successful completion of this project work.

ABSTRACT

In the context of Information Societies, a tremendous amount of information is daily exchanged or released. Among various information-release cases, medical document release has gained significant attention for its potential in improving healthcare service quality and efficacy. However, integrity and origin authentication of released medical documents is the priority in subsequent applications. Moreover, sensitive nature of much of this information also gives rise to a serious privacy threat when medical documents are uncontrollably made available to untrusted third parties. Redactable signatures allow any party to delete pieces of an authenticated document while guaranteeing the origin and integrity authentication of the resulting (released) subdocument. Nevertheless, most of existing redactable signature schemes (RSSs) are vulnerable to dishonest redactors or illegal redaction detection. To address the above issues, we propose two distinct RSSs with flexible release control (RSSs-FRC). We also analyse the performance of our constructions in terms of security, efficiency and functionality. The analysis results show that the performance of our construction has significant advantages over others, from the aspects of security and efficiency.

INDEX

LIST OF IMAGES

S.NO	FIGURE NO	FIGURE NAME	PAGE NO
1	5.1	System Architecture	13
2	5.2	Dataflow Diagram	15
3	5.3.1	Use Case Diagram	17
4	5.3.2	Class Diagram	18
5	5.3.3	Sequence diagram	19
6	5.3.4	Activity Diagram	20

1. INTRODUCTION

A common phenomenon in healthcare in most Arab countries is the lack of optimal utilization of human and material resources available to provide integrated healthcare to prevent diseases and treat diseases after they occur. Statistics indicate that Arab countries suffer from high rates of health problems, such as diabetes, liver disease, and parasitic diseases, such as histosomiasis and malaria. These health problems could be prevented before they occur or their complications prevented by early detection. This is due to a combination of factors: planning, operational, and technical. If we were able to overcome them, this would lead to significant progress in the level of health care. In addition, there is a weakness and lack of available hospital information systems, which is some of the most advanced software that directly serves all technical and administrative healthcare activities, ensuring that the medical institution has full control over all its activities and resources. The successes of these advanced systems do not depend on the exact selection of equipment and software for storage. Rather, their success depends on their suitability for different users—from healthcare providers, such as doctors, nurses, technicians, and even administrators—where the vision and priorities of each of these categories differ, and their information needs vary, as do the benefits of each of these systems.

The traditional health system (paper) has been replaced by an electronic health information system because the traditional system has been found to be ineffective due to a number of issues, including low storage capacity, high operating and maintenance costs, and system integration [1]. The computerized health system was then replaced by cloud computing because it relies on a more efficient infrastructure, as well as the many benefits of cloud computing in IT, such as cost, scalability, flexibility, and other features [2]. The use of cloud computing in electronic health records reduces costs in the provision of health services, maintenance costs, networks, licensing fees, and infrastructure in general, and this will therefore encourage developers to adopt the cloud in healthcare [2], [3].

2. LITERATURE SURVEY

A Review of Cloud Computing Technology Solution for Healthcare System

AUTHORS: Masrom, Maslin, and Ailar Rahimli

Previously the traditional healthcare information system that used in the healthcare sector was the paper-based and then later it was replaced by the Healthcare Information System (HIS). However the HIS was found not perform effectively because of several issues such as storage capacity, system integration, high operating cost and system maintenance. Cloud computing is a new technology that deliver the software, infrastructure and computational platform as a service over the Internet in any place and any time. This technology has been said can solve many problems of the healthcare system such as increase the storage capacity and add new capability on the existing healthcare system. Cloud computing offers cost effective, increase interoperability and accessibility, optimize resources and integrate the healthcare information systems

Cloud Computing in Healthcare: A Space of Opportunities and Challenges

AUTHORS: HUCÍKOVÁ, Anežka, and Ankica Babic

As the costs of healthcare services rise and healthcare professionals are becoming scarce and hard to find, it is imminent that healthcare organizations consider adopting health information technology (HIT) systems. HIT allows health organizations to streamline many of their processes and provide services in a more efficient and cost-effective manner. The latest technological trends such as Cloud Computing (CC) provide a strong infrastructure and offer a true enabler for HIT services over the Internet. This can be achieved on a pay-as-you-use model of the “e-Health Cloud” to help the healthcare industry cope with current and future demands yet keeping their costs to a minimum. Despite its great potential, HIT as a CC model has not been addressed extensively in the literature. There are no apparent frameworks which clearly encompass all viable schemes and interrelationships between HIT and CC.

3. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

The digital information collected by enterprises, public administrations, and governments has created enormous opportunities for knowledge-based applications. Driven by these benefits, there exists a high demand for the publication and exchange of collected data among numerous parties. However, sensitive information about users is typically contained in the original documents, and the privacy would be violated if such data is released without being processed.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

- Another threat for medical data sharing is that the released data are vulnerable to be tempered with. Relevant to this, yet another important requirement regarding the secondary use of medical data is to provide an authentication mechanism for data users.
- It is quite obvious that medical data is a valuable asset to data holders. In order to guarantee an adequate quality of data, it is crucial to check the origin and integrity of involved data at any time.

4.2 PROPOSED SYSTEM:

Redactable signatures, a straightforward approach, inherently solve the above theoretical incompatibility and practical requirements of privacy information redaction in authenticated medical document releasing. In the definition of redactable signature schemes (RSSs), parts of a signed document are allowed to be removed by any party while preserving the source and integrity verifiability of the remaining subdocument. Another outstanding advantage of the redactable signature is that the reserved subdocument and its signature of the original document do not reveal any content information about deleted parts.

4.2.1 ADVANTAGES OF PROPOSED SYSTEM:

- The healthcare provider forwards the medical documents and the corresponding redactable signatures to another party (redactor) such as patients or hospitals who are the subject or administrator of the signed medical documents.

- The second party is allowed to publicly redact parts of the signed medical documents that they do not want to release to third parties.

4.3 INPUT AND OUTPUT DESIGN:

4.3.1 INPUT DESIGN:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. What data should be given as input?

- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

4.3.2 OBJECTIVE:

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

4.3.3 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

1 Convey information about past activities, current status or projections of the
2 Future.
3 Signal important events, opportunities, problems, or warnings.
4 Trigger an action.
5 Confirm an action.

4.4 SYSTEM REQUIREMENTS:

Software Tools:

➢ System	:	Pentium IV 2.4 GHz.
➢ Hard Disk	:	40 GB.
➢ Floppy Drive	:	1.44 Mb.
➢ Monitor	:	15 VGA Colour.
➢ Mouse	:	Logitech.
➢ Ram	:	512 Mb.

Hardware Tools:

➢ Operating system	:	Windows XP/7.
➢ Coding Language	:	JAVA/J2EE
➢ IDE	:	Netbeans 7.4
➢ Database	:	MYSQL

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

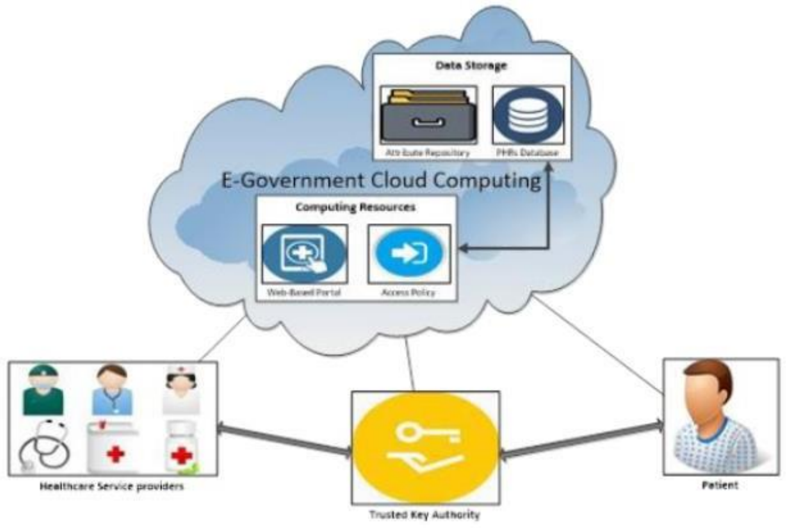


FIG 5.1 System architecture

DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

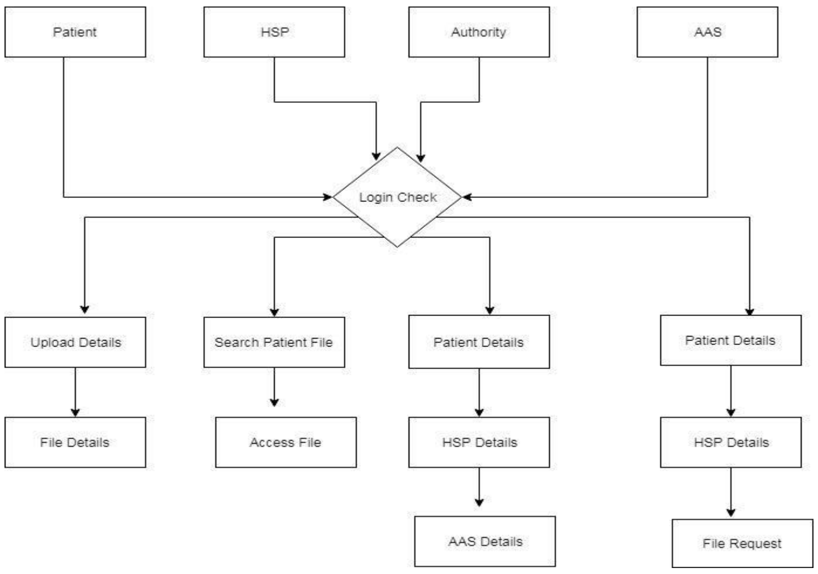


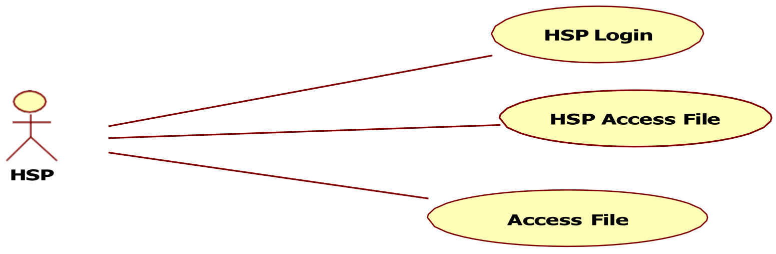
FIG 5.2 Dataflow diagram

5.3 UML DIAGRAM:

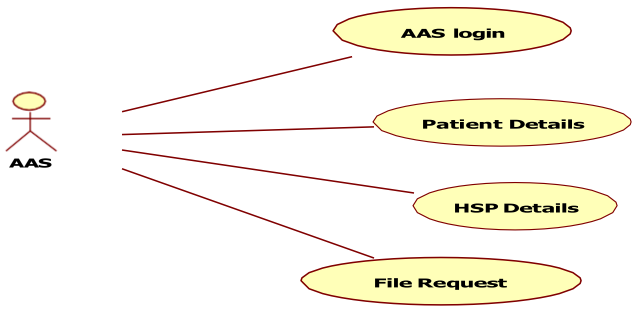
UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

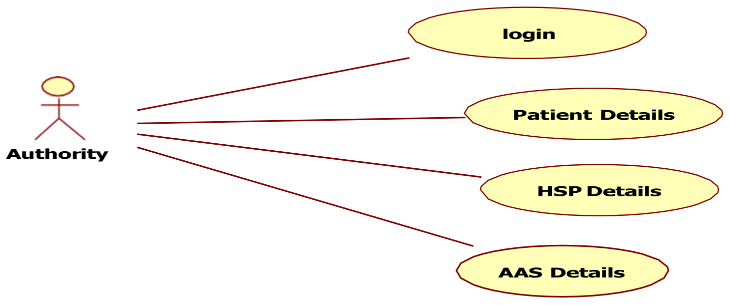
HSP



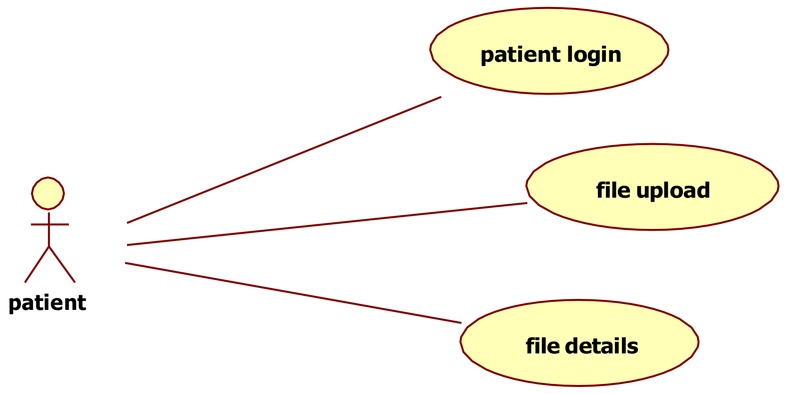
AAS



AUTHORITY



Patient



GOALS: The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

5.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

5.3.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

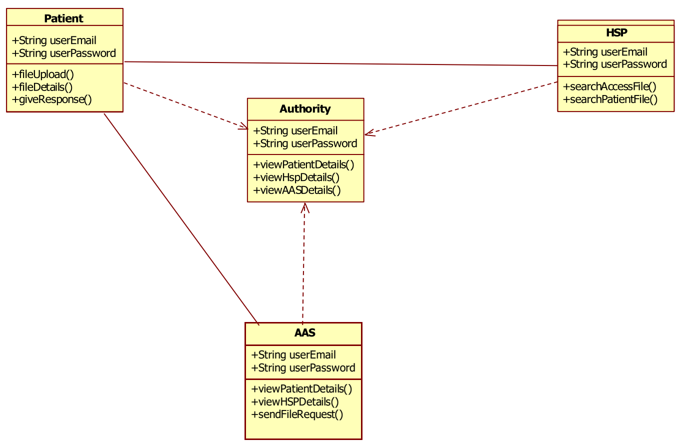


FIG 5.3.2 Class diagram

5.3.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

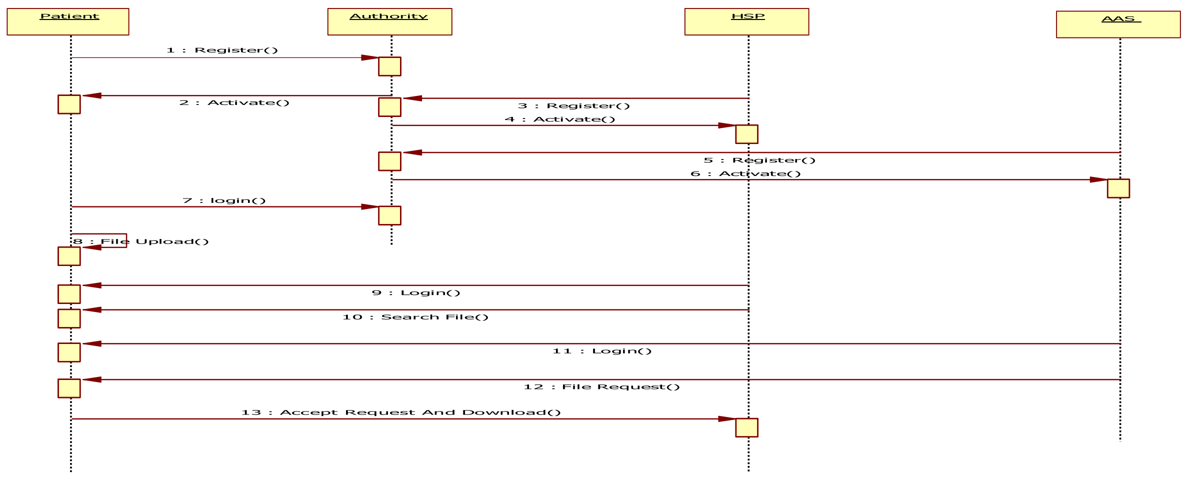


FIG 5.3.3 Sequence diagram

5.3.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

system. An activity diagram shows the overall flow of control.

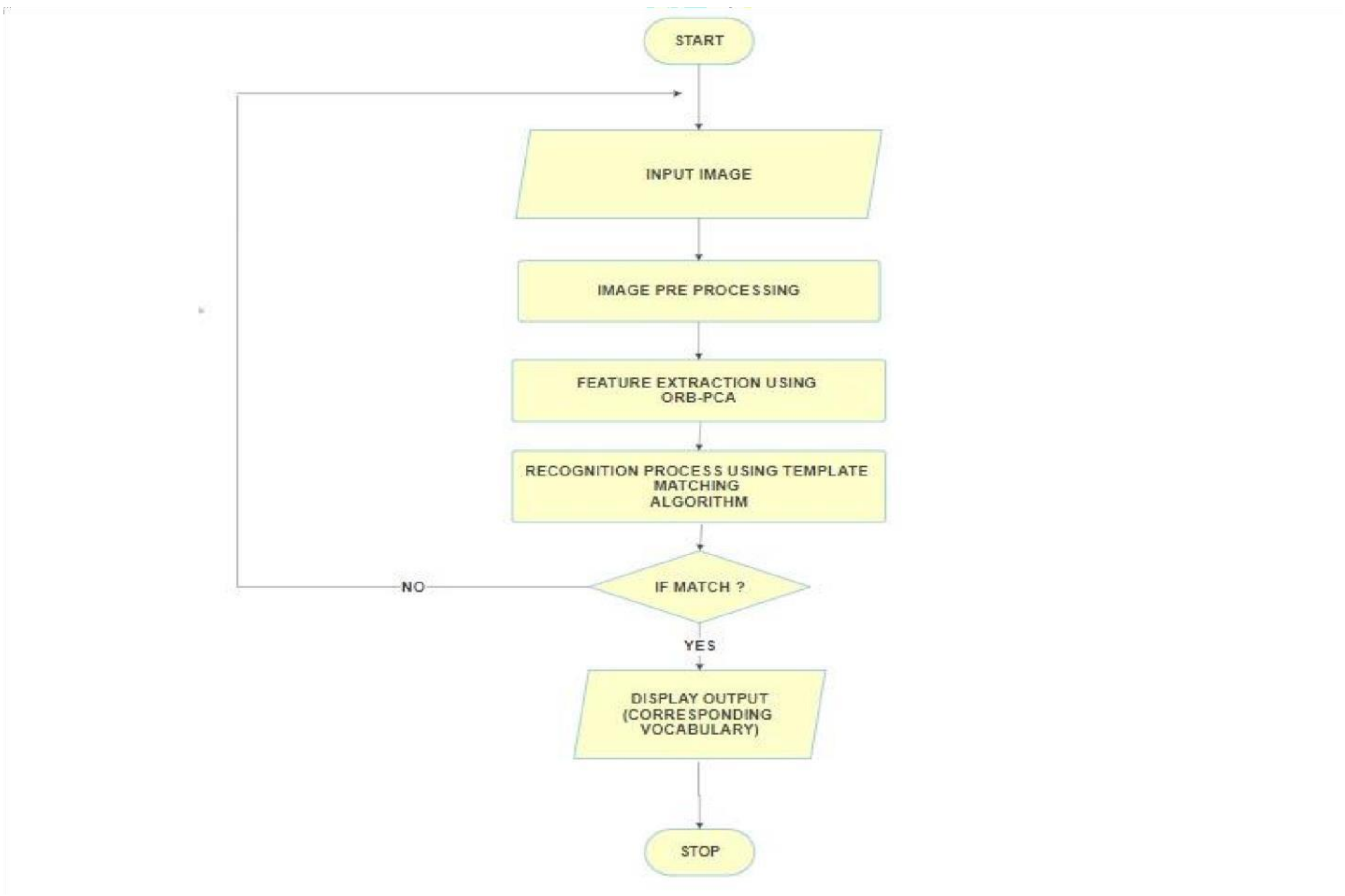

START
INPUT IMAGE
IMAGE PRE PROCESSING
FEATURE EXTRACTION USING ORB-PCA
RECOGNITION PROCESS USING TEMPLATE MATCHING ALGORITHM
NO
IF MATCH ?
YES
DISPLAY OUTPUT (CORRESPONDING VOCABULARY)
STOP

FIG 5.3.4: Activity diagram

6. SOFTWARE ENVIRONMENT

6.1 Java Technology

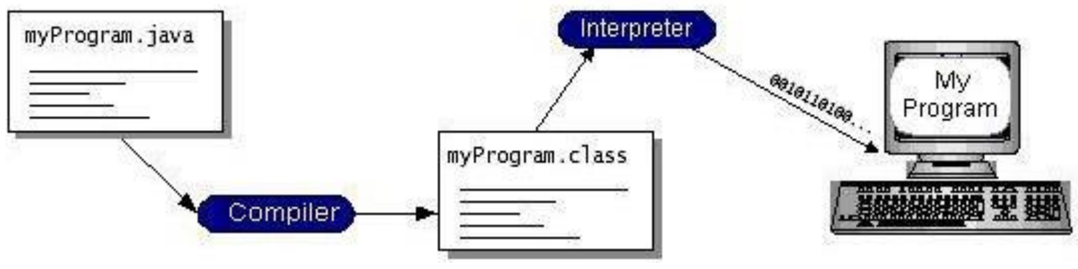
Java technology is both a programming language and a platform.

The Java Programming Language

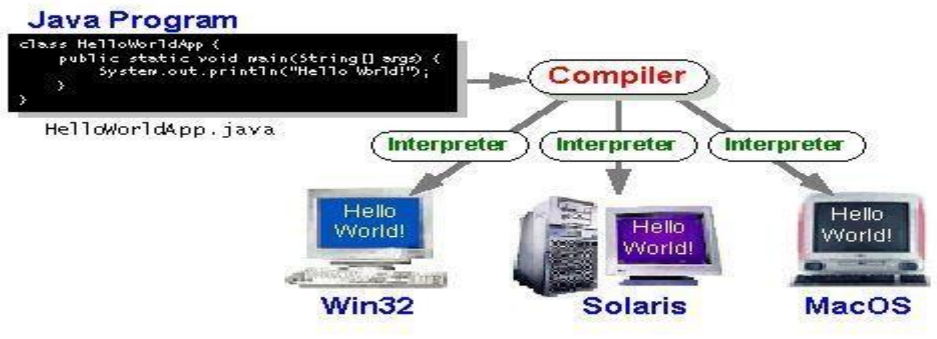
The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler.



The Java Platform

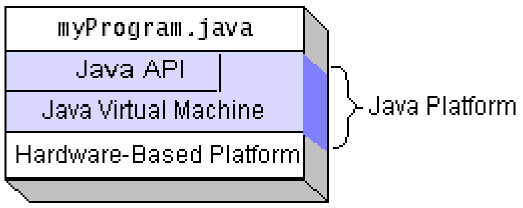
A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans ^{TM}, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC^{TM})**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

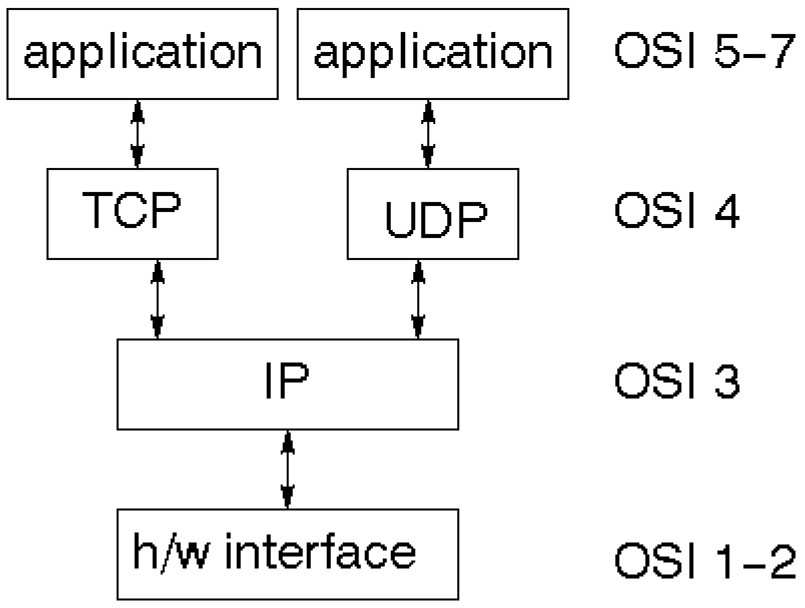
To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.
The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram’s

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address

The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with `Read File` and `Write File` functions.

#include <sys/types'>

#include <sys/socket's>

in socket(in family, in type, in protocol);

Here "family" will be `AF_INET` for IP communications, `protocol` will be zero, and `type` will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

JFree Chart

JFree Chart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFree Chart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFree Chart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFree Chart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFree Chart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFree Chart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in JFree Chart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

What is a Java Web Application?

A Java web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content. It is typically comprised of web components such as Java Server Pages (JSP), servlets and JavaBeans to modify and temporarily store data, interact with databases and web services, and render content in response to client requests.

Because many of the tasks involved in web application development can be repetitive or require a surplus of boilerplate code, web frameworks can be applied to alleviate the overhead associated with common activities. For example, many frameworks, such as Java Server Faces, provide libraries for templating pages and session management, and often promote code reuse.

What is Java EE?

Java EE (Enterprise Edition) is a widely used platform containing a set of coordinated technologies that significantly reduce the cost and complexity of developing, deploying, and managing multi-tier, server-centric applications. Java EE builds upon the Java SE platform and provides a set of APIs (application programming interfaces) for developing and running portable, robust, scalable, reliable and secure server-side applications.

Some of the fundamental components of Java EE include:

JavaScript and Ajax Development

JavaScript is an object-oriented scripting language primarily used in client-side interfaces for web applications. Ajax (Asynchronous JavaScript and XML) is a Web 2.0 technique that allows changes to occur in a web page without the need to perform a page refresh. JavaScript toolkits can be leveraged to implement Ajax-enabled components and functionality in web pages.

Web Server and Client

Web Server is a software that can process the client request and send the response back to the client. For example, Apache is one of the most widely used web server. Web Server runs on some physical machine and listens to client request on specific port.

A web client is a software that helps in communicating with the server. Some of the most widely used web clients are Firefox, Google Chrome, Safari etc. When we request something from server (through URL), web client takes care of creating a request and sending it to server and then parsing the server response and present it to the user.

HTML and HTTP

Web Server and Web Client are two separate software's, so there should be some common language for communication. HTML is the common language between server and client and stands for **H**yper **T**ext **M**arkup **L**anguage.

Web server and client needs a common communication protocol, HTTP (**H**yper **T**ext **T**ransfer **P**rotocol) is the communication protocol between server and client. HTTP runs on top of TCP/IP communication protocol.

Some of the important parts of HTTP Request are:

- **HTTP Method** – action to be performed, usually GET, POST, PUT etc.
- **URL** – Page to access
- **Form Parameters** – similar to arguments in a java method, for example user , password details from login page.

Sample HTTP Request:

```
GET /FirstServletProject/jsps/hello.jsp HTTP/1.1
Host: localhost:8080
Cache-Control: no-cache
```

Some of the important parts of HTTP Response are:

- **Status Code** – an integer to indicate whether the request was success or not. Some of the well known status codes are 200 for success, 404 for Not Found and 403 for Access Forbidden.
- **Content Type** – text, html, image, pdf etc. Also known as MIME type
- **Content** – actual data that is rendered by client and shown to user.

MIME Type or Content Type: If you see above sample HTTP response header, it contains tag "Content-Type". It's also called MIME type and server sends it to client to let them know the kind of data it's sending. It helps client in rendering the data for user. Some of the mostly used mime types are text/html, text/xml, application/xml etc.

Understanding URL

URL is acronym of Universal Resource Locator and it's used to locate the server and resource. Every resource on the web has it's own unique address. Let's see parts of URL with an example.
http://localhost:8080/FirstServletProject/jsps/hello.jsp

http:// – This is the first part of URL and provides the communication protocol to be used in server-client communication.

localhost – The unique address of the server, most of the times it's the hostname of the server that maps to unique IP address. Sometimes multiple hostnames point to same IP addresses and web server virtual host takes care of sending request to the particular server instance.

8080 – This is the port on which server is listening, it's optional and if we don't provide it in URL then request goes to the default port of the protocol. Port numbers 0 to 1023 are reserved ports for well known services, for example 80 for HTTP, 443 for HTTPS, 21 for FTP etc.

FirstServletProject/jsps/hello.jsp – Resource requested from server. It can be static html, pdf, JSP, servlets, PHP etc.

Why we need Servlet and JSPs?

Web servers are good for static contents HTML pages but they don't know how to generate dynamic content or how to save data into databases, so we need another tool that we can use to generate dynamic content. There are several programming languages for dynamic content like PHP, Python, Ruby on Rails, Java Servlets and JSPs.

Web Container

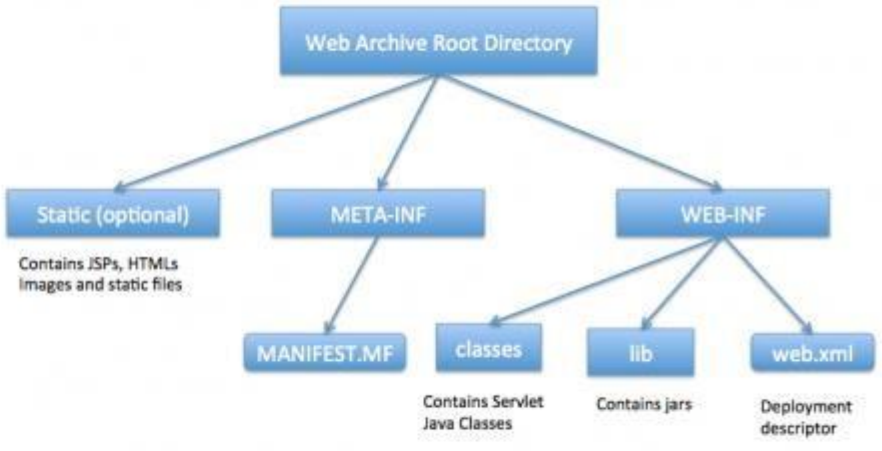
Tomcat is a web container, when a request is made from Client to web server, it passes the request to web container and it's web container job to find the correct resource to handle the request (servlet or JSP) and then use the response from the resource to generate the response and provide it to web server. Then web server sends the response back to the client.

Some of the important work done by web container are:

- **Communication Support** – Container provides easy way of communication between web server and the servlets and JSPs. Because of container, we don't need to build a server socket to listen for any request from web server, parse the request and generate response. All these important and complex tasks are done by container and all we need to focus is on our business logic for our applications.
- **Lifecycle and Resource Management** – Container takes care of managing the life cycle of servlet. Container takes care of loading the servlets into memory, initializing servlets, invoking servlet methods and destroying them. Container also provides utility like JNDI for resource pooling and management.
- **Multithreading Support** – Container creates new thread for every request to the servlet and when it's processed the thread dies. So servlets are not initialized for each request and saves time and memory.
- **JSP Support** – JSPs doesn't look like normal java classes and web container provides support for JSP. Every JSP in the application is compiled by container and converted to Servlet and then container manages them like other servlets.
- **Miscellaneous Task** – Web container manages the resource pool, does memory optimizations, run garbage collector, provides security configurations, support for multiple applications, hot deployment and several other tasks behind the scene that makes our life easier.

Web Application Directory Structure

Java Web Applications are packaged as Web Archive (WAR) and it has a defined structure. You can export above dynamic web project as WAR file and unzip it to check the hierarchy. It will be something


Web Archive Root Directory
Static (optional)
META-INF
WEB-INF
Contains JSPs, HTMLs Images and static files
MANIFEST.MF
classes
lib
web.xml
Contains Servlet Java Classes
Contains jars
Deployment descriptor

like below image.

Deployment Descriptor

web.xml file is the deployment descriptor of the web application and contains mapping for servlets (prior to 3.0), welcome pages, security configurations, session timeout settings etc.
Thats all for the java web application startup tutorial, we will explore Servlets and JSPs more in future posts.

MySQL:

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.
The MySQL Web site (http://www.mysql.com/) provides the latest information about MySQL software.

- **MySQL is a database management system.**
  A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL databases are relational.**
  A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and "pointers" between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.
  SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, "SQL-92" refers to the standard released in

1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

- **My SQL software is Open Source.**

  Open Source means that it is possible for anyone to use and modify the software. Anybody can download the My SQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The My SQL software uses the GPL (GNU General Public License), http://www.fsf.org/licenses/, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed My SQL code into a commercial application, you can buy a commercially licensed version from us. See the My SQL Licensing Overview for more information (http://www.mysql.com/company/legal/licensing/).

- **The My SQL Database Server is very fast, reliable, scalable, and easy to use.**

  If that is what you are looking for, you should give it a try. My SQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to My SQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. My SQL can also scale up to clusters of machines, networked together.

  You can find a performance comparison of My SQL Server with other database managers on our benchmark page.

- **My SQL Server works in client/server or embedded systems.**

  The My SQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

  We also provide My SQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

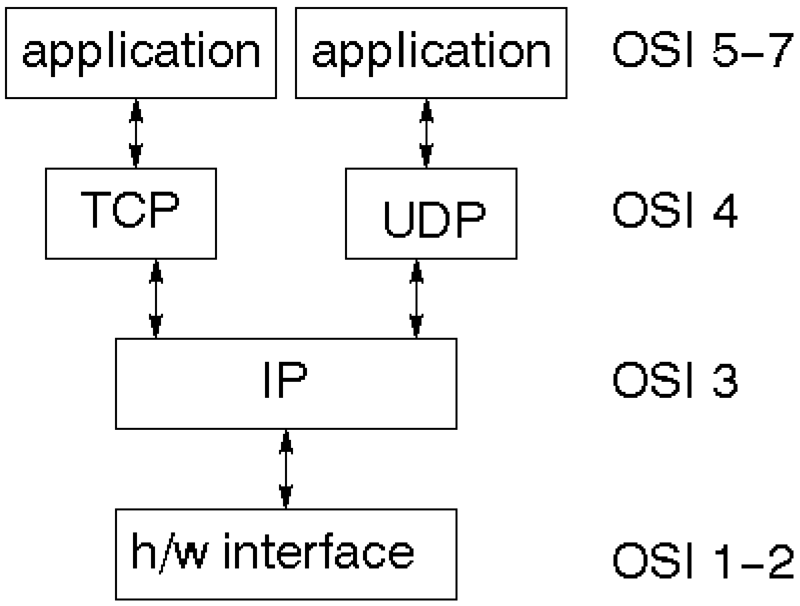
- **A large amount of contributed My SQL software is available.**

  My SQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the My SQL Database Server.

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

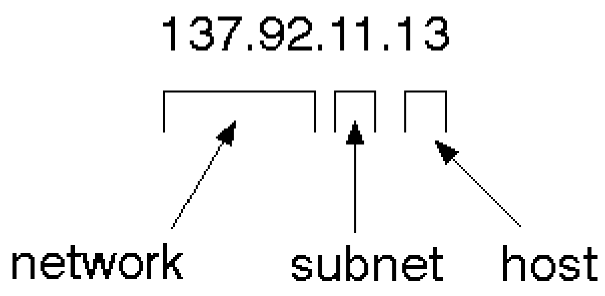
Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address



The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with `Read File` and `Write File` functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be `AF_INET` for IP communications, `protocol` will be zero, and `type` will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

JFree Chart

JFree Chart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFree Chart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFree Chart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

2. Time Series Chart Interactivity

Implement a new (to JFree Chart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFree Chart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in JFree Chart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

What is a Java Web Application?

A Java web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content. It is typically comprised of web components such as Java Server Pages (JSP), servlets and JavaBeans to modify and temporarily store data, interact with databases and web services, and render content in response to client requests.

What is Java EE?

Java EE (Enterprise Edition) is a widely used platform containing a set of coordinated technologies that significantly reduce the cost and complexity of developing, deploying, and managing multi-tier, server-centric applications. Java EE builds upon the Java SE platform and provides a set of APIs (application programming interfaces) for developing and running portable, robust, scalable, reliable and secure server-side applications.

JavaScript and Ajax Development

JavaScript is an object-oriented scripting language primarily used in client-side interfaces for web applications. Ajax (Asynchronous JavaScript and XML) is a Web 2.0 technique that allows changes to occur in a web page without the need to perform a page refresh. JavaScript toolkits can be leveraged to implement Ajax-enabled components and functionality in web pages.

Web Server and Client

Web Server is a software that can process the client request and send the response back to the client. For example, Apache is one of the most widely used web server. Web Server runs on some physical machine and listens to client request on specific port.

A web client is a software that helps in communicating with the server. Some of the most widely used web clients are Firefox, Google Chrome, Safari etc. When we request something from server (through URL), web client takes care of creating a request and sending it to server and then parsing the server response and present it to the user.

HTML and HTTP

Web Server and Web Client are two separate softwares, so there should be some common language for communication. HTML is the common language between server and client and stands for **H**yper **T**ext **M**arkup **L**anguage.

Web server and client needs a common communication protocol, HTTP (**H**yper **T**ext **T**ransfer **P**rotocol) is the communication protocol between server and client. HTTP runs on top of TCP/IP communication protocol.

Some of the important parts of HTTP Request are:

- **HTTP Method** – action to be performed, usually GET, POST, PUT etc.
- **URL** – Page to access
- **Form Parameters** – similar to arguments in a java method, for example user , password details from login page.

Sample HTTP Request:

```
1    GET /FirstServletProject/jsps/hello.jsp HTTP/1.1
2    Host: localhost:8080
3    Cache-Control: no-cache
```

Some of the important parts of HTTP Response are:

- **Status Code** – an integer to indicate whether the request was success or not. Some of the well known status codes are 200 for success, 404 for Not Found and 403 for Access Forbidden.
- **Content Type** – text, html, image, pdf etc. Also known as MIME type
- **Content** – actual data that is rendered by client and shown to user.

MIME Type or Content Type: If you see above sample HTTP response header, it contains tag "Content-Type". It's also called MIME type and server sends it to client to let them know the kind of data it's sending. It helps client in rendering the data for user. Some of the mostly used mime types are text/html, text/xml, application/xml etc.

Understanding URL

URL is acronym of Universal Resource Locator and it's used to locate the server and resource. Every resource on the web has it's own unique address. Let's see parts of URL with an example.
http://localhost:8080/FirstServletProject/jsps/hello.jsp

http:// – This is the first part of URL and provides the communication protocol to be used in server-client communication.

localhost – The unique address of the server, most of the times it's the hostname of the server that maps to unique IP address. Sometimes multiple hostnames point to same IP addresses and web server virtual host takes care of sending request to the particular server instance.

8080 – This is the port on which server is listening, it's optional and if we don't provide it in URL then request goes to the default port of the protocol. Port numbers 0 to 1023 are reserved ports for well known services, for example 80 for HTTP, 443 for HTTPS, 21 for FTP etc.

FirstServletProject/jsps/hello.jsp – Resource requested from server. It can be static html, pdf, JSP, servlets, PHP etc.

Why we need Servlet and JSPs?

Web servers are good for static contents HTML pages but they don't know how to generate dynamic content or how to save data into databases, so we need another tool that we can use to generate dynamic content. There are several programming languages for dynamic content like PHP, Python, Ruby on Rails, Java Servlets and JSPs.
Java Servlet and JSPs are server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.

Web Container

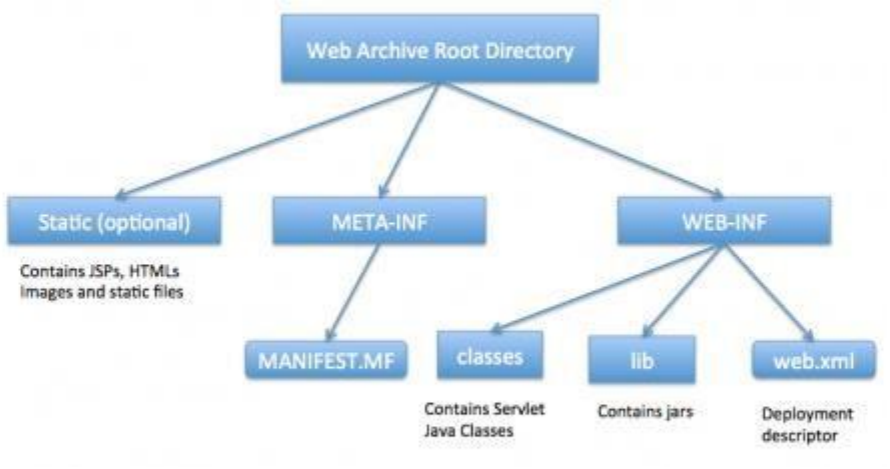
Tomcat is a web container, when a request is made from Client to web server, it passes the request to web container and it's web container job to find the correct resource to handle the request (servlet or JSP) and then use the response from the resource to generate the response and provide it to web server. Then web server sends the response back to the client.
When web container gets the request and if it's for servlet then container creates two Objects HTTPServletRequest and HTTPServletResponse. Then it finds the correct servlet based on the URL and creates a thread for the request. Then it invokes the servlet service() method and based on the HTTP method service() method invokes doGet() or doPost() methods. Servlet methods generate the dynamic page and write it to response. Once servlet thread is complete, container converts the response to HTTP response and send it back to client.
Some of the important work done by web container are:

- **Communication Support** – Container provides easy way of communication between web server and the servlets and JSPs. Because of container, we don't need to build a server socket to listen for any request from web server, parse the request and generate response. All these important and complex tasks are done by container and all we need to focus is on our business logic for our applications.

- **Lifecycle and Resource Management** – Container takes care of managing the life cycle of servlet. Container takes care of loading the servlets into memory, initializing servlets, invoking servlet methods and destroying them. Container also provides utility like JNDI for resource pooling and management.
- **Multithreading Support** – Container creates new thread for every request to the servlet and when it's processed the thread dies. So servlets are not initialized for each request and saves time and memory.
- **JSP Support** – JSPs doesn't look like normal java classes and web container provides support for JSP. Every JSP in the application is compiled by container and converted to Servlet and then container manages them like other servlets.
- **Miscellaneous Task** – Web container manages the resource pool, does memory optimizations, run garbage collector, provides security configurations, support for multiple applications, hot deployment and several other tasks behind the scene that makes our life easier.

Web Application Directory Structure

Java Web Applications are packaged as Web Archive (WAR) and it has a defined structure. You can export above dynamic web project as WAR file and unzip it to check the hierarchy. It will be something like below image.


Web Archive Root Directory
Static (optional)
Contains JSPs, HTMLs Images and static files
META-INF
WEB-INF
MANIFEST.MF
classes
Contains Servlet Java Classes
lib
Contains jars
web.xml
Deployment descriptor

Deployment Descriptor

web.xml file is the deployment descriptor of the web application and contains mapping for servlets (prior to 3.0), welcome pages, security configurations, session timeout settings etc.

Thats all for the java web application startup tutorial, we will explore Servlets and JSPs more in future posts.

My SQL:

My SQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.
The My SQL Web site (http://www.mysql.com/) provides the latest information about My SQL software.

- **My SQL is a database management system.**
  A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as My SQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.
- **My SQL software is Open Source.**
  Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), http://www.fsf.org/licenses/, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information (http://www.mysql.com/company/legal/licensing/).

- **The My SQL Database Server is very fast, reliable, scalable, and easy to use.**
  If that is what you are looking for, you should give it a try. My SQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to My SQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. My SQL can also scale up to clusters of machines, networked together.

You can find a performance comparison of My SQL Server with other database managers on our benchmark page.

- **My SQL Server works in client/server or embedded systems.**
  The My SQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).
  We also provide My SQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

- **A large amount of contributed My SQL software is available.**
  My SQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the My SQL Database Server.

The official way to pronounce “My SQL” is “My Ess Que Ell” (not “my sequel”), but we do not mind if you pronounce it as “my sequel” or in some other localized way.

7. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

MODULE DESCRIPTION:

Number of Modules

After careful analysis the system has been identified to have the following modules:

1. **Authority**
2. **Patient**
3. **Healthcare Service Provider**
4. **Authority Attribute**

MODULES DESCRIPTION:

Authority

The principle of least authority states that each component of the system should be given authority to access only the information and resources that it needs for its operation. This principle is fundamental to the secure design of software systems, as it helps to limit an application's attack surface and to isolate vulnerabilities and faults. Unfortunately, current programming languages do not provide adequate help in controlling the authority of application modules, an issue that is particularly acute in the case of entrusted third-party extensions. In this paper, we present a language design that facilitates controlling the authority granted to each application module. The key technical novelty of our approach is that modules are first-class, statically typed capabilities. First-class modules are essentially objects, and so we formalize our module system by translation into an object calculus and prove that the core calculus is type-safe and authority-safe. Unlike prior formalizations, our work defines authority non-transitively, allowing engineers to reason about software designs that use wrappers to provide an attenuated version of a more powerful capability.

Patient

This module registers patient details based on general and demographic information. Patients are allocated a Unique Health Identification Number (UHID) and discount cards at the time of registration. Detailed information of patients, Mandatory fields for crucial patient information as per JCI Standards, Alerts in place to prevent erroneous data entry, Generates Smart Card with Unique Health Identification Number (UHID), Advanced multi-criteria search for registered patients, FID wrist band generation and Provision for recording sponsor, insurance and medical tourism details.

Healthcare Service Provider:

The entire health care system is weaved with each other by the single body that is hospital or provider (doctor).

While the other entities include-

- **Patient/Consumers:** Patient Enrolled
- **Regulatory Authority:** HIPAA, OASIS assessment, HCFA 1500 and UB92, etc.
- **Health-care and Life-Science solution Vendors**

Basic Terminology of Health Care System

- **Provider**: A health care professional (doctor), medical group, clinic, lab, hospital, etc. licensed by health care services
- **Medicare:** A federal health insurance program for senior citizen and permanently disabled people
- **Medicaid:** A joint and state program that helps low-income families and individuals pay for the cost associated with medical care
- **CPT code**: A current procedural terminology code is a medical code set to describe medical, surgical and diagnostic services
- **HIPAA**: It is a set of rules and regulations which doctors, hospitals, healthcare providers and health plan must follow in order to provide their services

Authority Attribute

The traditional role-based access control model (RBAC) can not meet the requirements of Service Oriented Architectures (SOA) on the distribution and openness, Attribute-Based

Access Control (ABAC), which is more fine-grained in access control, is more fit into the SOA open environment. This paper presents an ABAC-based cross-domain access control system, together with the security domain as a attribute with the subject, object, authority, environment attributes as the basis for access to the decision-making, eliminating integration constraints for the SOA framework based on the RBAC, somehow improves the scalability and alterability of the system, solved the problem of cross-domain access control.

Along these lines, display the Annoy Control-F, which ultimately keeps the character spillage and accomplish the full secrecy. Our security assessment demonstrates that both Annoy Control and Annoy Control-F are secure under the decisional bilinear Diffie-Hellman presumption, and our execution assessment shows the attainability of our plans.

can be made more robust and effective across a broader range of applications.

8. SOURCE CODE

DBConnection.java:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.gcloud.db;

import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author Poojitha
 */
public class DBConnection {
    public static Connection con = null;
    public static Connection getDBConnection(){
            return con;
            }else{
                System.out.println("Connection object is Null");
            }
        }
            try {
                //con.close();
            } catch (Exception e) {
            }
            return con;
        }
    }
}
```

AASLoginCheck.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.gcloud.actions;

import com.gcloud.db.DBConnection;
import java.io.IOException;
import javax.servlet.http.HttpSession;

/**
 *
 * @author Poojitha
 */
public class AASLoginCheck extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String Email = request.getParameter("email");
        String pswd = request.getParameter("pswd");

aasregister where email = ? and pswd = ?";
            ps = con.prepareStatement(sqlQury);
            ps.setString(1, Email);
            ps.setString(2, pswd);
            rs = ps.executeQuery();
            if(rs.next()){
            aaname = rs.getString("patientname");
            hs.setAttribute("aaName", aaname);
            hs.setAttribute("aaEmail", Email);
            status = rs.getString("status");

        ex.printStackTrace();
        response.sendRedirect("AAS.jsp?msg=failed");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

AASRegisterAction.java

```
/*
.gcloud.actions;


/**

     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String accessKey = "NotGenerated";
        out.println("i know Your Password " + ppswd);
        out.println("i know Your Password " + mobile);

        Connection con = null;
        PreparedStatement ps = null;
        try {
            con = DBConnection.getDBConnection();
            String sqlQuery = "insert into
            ps.setString(5, dob);
            ps.setString(6, gender);
            ps.setString(7, state);


        } catch (Exception e) {
            e.printStackTrace();
            response.sendRedirect("AASRegister.jsp?msg=exist");
        }finally{
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
            }
        }

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
```

AuthorityLoginCheck.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.gcloud.actions;

import java.io.IOException;
;

/**
 *
 * @author Poojitha
 */
public class AuthirityLoginCheck extends HttpServlet {



    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
           throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
       PrintWriter out = response.getWriter();


   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
   /**
    * Handles the HTTP <code>GET</code> method.



   /**
    * Handles the HTTP <code>POST</code> method.
    *
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    */


    *
    * @return a String containing servlet description
    */
   @Override
```

HSPLoginCheck.java

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.gcloud.actions;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
 * @author Poojitha
 */
        PrintWriter out = response.getWriter();
        String email = request.getParameter("email");
        String pswd = request.getParameter("pswd");

        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        String status = null;

            if(rs.next()){
            hspname = rs.getString("patientname");
            hs.setAttribute("hspName", hspname);
            hs.setAttribute("hspemail", email);
            hs.setAttribute("hsprole", rs.getString("role"));
            status = rs.getString("status");
            if(status.equalsIgnoreCase("Activated")){
            response.sendRedirect("HSPHome.jsp?msg=success");
            }else if(status.equalsIgnoreCase("waiting")){
             response.sendRedirect("HSP.jsp?msg=notactivated")
            }else{
            response.sendRedirect("HSP.jsp?msg=failed");
            }
            /*if(rs.next()){
            status = rs.getString(1);
            }else{

        ex.printStackTrace();
        response.sendRedirect("HSP.jsp?msg=notexist");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
tion";
    }// </editor-fold>
}
```

HSPRegisterAction.java

```
package com.gcloud.actions;


import com.gcloud.db.DBConnection;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
public class HSPRegisterAction extends HttpServlet {


   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
            con = DBConnection.getDBConnection();
            String sqlQuery = "insert into
hspregister(patientname,email,pswd,mobile,dob,gender,state,country,status,accesskey,r
ole) values(?,?,?,?,?,?,?,?,?,?,?)";
            ps = con.prepareStatement(sqlQuery);
            ps.setString(1, patientName);
            ps.setString(2, pemail);
            ps.setString(3, ppswd);
            ps.setString(4, mobile);
            ps.setString(5, dob);
                response.sendRedirect("HSPRegister.jsp?msg=failed");
            }


        } catch (Exception e) {
            e.printStackTrace();
            response.sendRedirect("HSPRegister.jsp?msg=exist");
        }finally{
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
            }
        }

    }



    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">


if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }


    }



    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

PatientFileUpload.java

```
package com.gcloud.actions;


import com.gcloud.db.DBConnection;
import java.io.IOException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

extends HttpServlet {



   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {


      System.out.println("File Name "+filename);
      System.out.println(hsptl+"<->"+dctr+"<->"+srgen+"<->"+insurance);
      StringBuffer accessPermission = new StringBuffer();
      if (hsptl!=null){
      accessPermission.append(hsptl+" ");
      }
      if(dctr!=null){
      accessPermission.append(dctr+" ");
      }
      if(srgen!=null){
      accessPermission.append(srgen+" ");
      }

().getTime());
      try {

         con = DBConnection.getDBConnection();
         String sqlQuery = "insert into
patientfiles(pemail,filename,accessibility,cdate,cfile) values(?,?,?,?,?)";
         ps = con.prepareStatement(sqlQuery);
```

```
            System.out.println(" Error at "+ex.getMessage());
            response.sendRedirect("fileupload.jsp?msg=failed");
        }finally{
            try {
                ps.close();
                con.close();
            } catch (Exception e) {
            }
        }
    }

>
}
```

PatientLoginCheck.java

```
package com.gcloud.actions;

import com.gcloud.db.DBConnection;
import java.io.IOException;
*/
public class PatientLoginCheck extends HttpServlet {



    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String patientEmail = request.getParameter("email");
        String pswd = request.getParameter("pswd");
        HttpSession hs = request.getSession();
        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        String status = null;
        try {
            con = DBConnection.getDBConnection();
            String sqlQury = "select status from patientregister where email = ? and pswd =
?";



            response.sendRedirect("Patient.jsp?msg=failed");
            }



        }catch(Exception ex){

        ex.printStackTrace();
        response.sendRedirect("Patient.jsp?msg=notexist");
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

PatientRegisterAction.java

```
;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Poojitha
 */
public class PatientRegisterAction extends HttpServlet {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String patientName = request.getParameter("pname");
        String pemail = request.getParameter("pemail");
        String ppswd = request.getParameter("ppswd");
        String mobile = request.getParameter("mobile");

        out.println("i know Your Password " + mobile);

        Connection con = null;
        PreparedStatement ps = null;
        try {
            con = DBConnection.getDBConnection();
            if (count > 0) {
                response.sendRedirect("PatientRegister.jsp?msg=success");
            } else {
                 response.sendRedirect("PatientRegister.jsp?msg=failed");
            }
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

AccessKeyGenerations.java

```
package com.gcloud.utility;

import java.security.SecureRandom;
import java.util.Random;

/**
 *
 * @author Poojitha
 */
public class AccessKeyGenerations {
  static final private String ALPHABET =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz"
;

    public static void main(String[] args) {
        AccessKeyGenerations gsk = new AccessKeyGenerations();
        char spacerChar = 'D';
        String key = gsk.randomUUID(15, 0, spacerChar);
        System.out.println("Key "+key+" And its Length is "+key.length());

    }
}
```

SendingMails.java

```
package com.gcloud.utility;

import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;



        Session session = Session.getDefaultInstance(props,
        new javax.mail.Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication() {
        return                                                          new
PasswordAuthentication("datapointprojects13@gmail.com","lx160cm@1");
        }
        });

        //System.out.println("Message   " + msg);
        try {
            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress(userid));


    }
}
```

9. SYSTEM TESTING

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.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs 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.

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 test

Functional tests provide 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 must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

System Test

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.

6.1 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.
-

6.2 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.

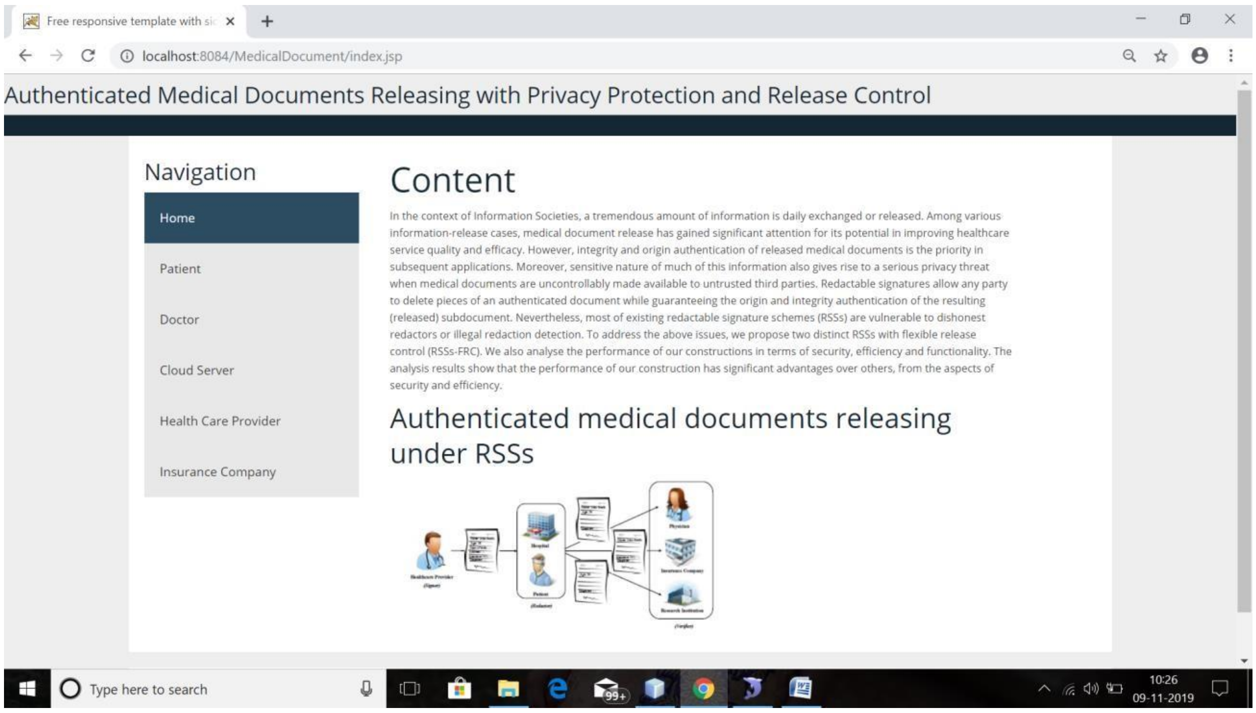
6.3 Acceptance Testing

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.

10. SCREENSHOTS

Home Page



localhost:8084/MedicalDocument/index.jsp

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Navigation

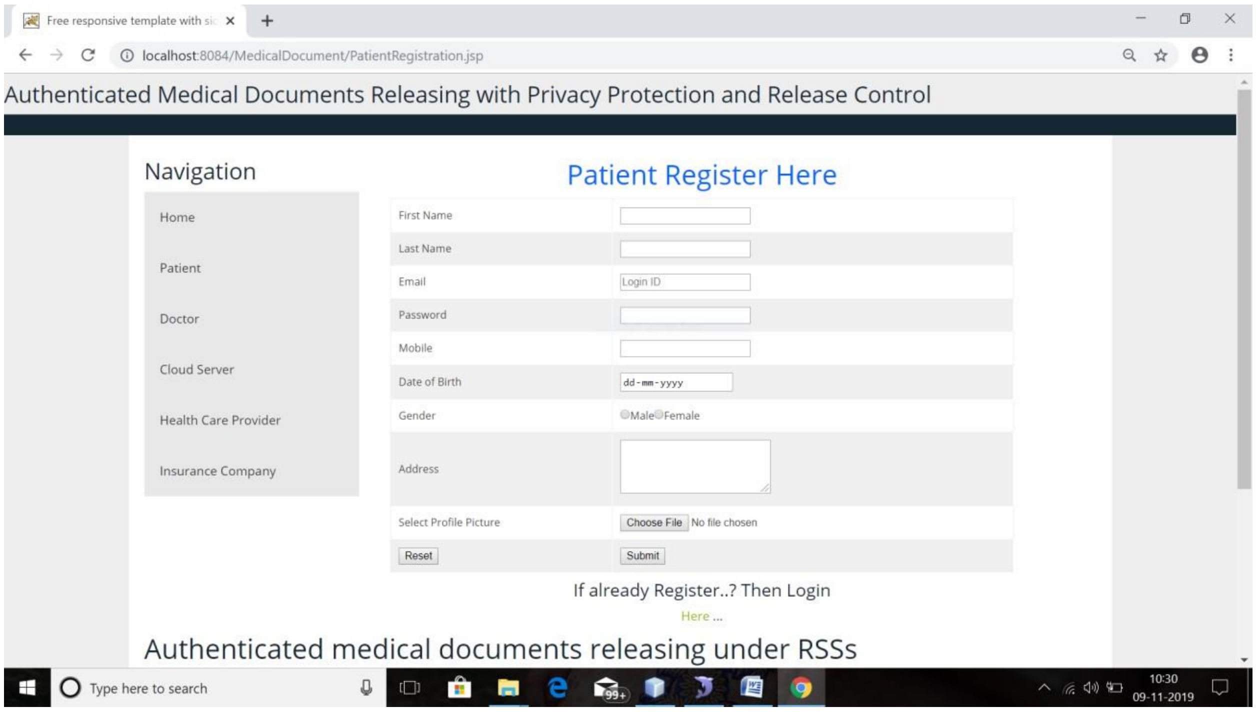
- Home
- Patient
- Doctor
- Cloud Server
- Health Care Provider
- Insurance Company

Content

In the context of Information Societies, a tremendous amount of information is daily exchanged or released. Among various information-release cases, medical document release has gained significant attention for its potential in improving healthcare service quality and efficacy. However, integrity and origin authentication of released medical documents is the priority in subsequent applications. Moreover, sensitive nature of much of this information also gives rise to a serious privacy threat when medical documents are uncontrollably made available to untrusted third parties. Redactable signatures allow any party to delete pieces of an authenticated document while guaranteeing the origin and integrity authentication of the resulting (released) subdocument. Nevertheless, most of existing redactable signature schemes (RSSs) are vulnerable to dishonest redactors or illegal redaction detection. To address the above issues, we propose two distinct RSSs with flexible release control (RSSs-FRC). We also analyse the performance of our constructions in terms of security, efficiency and functionality. The analysis results show that the performance of our construction has significant advantages over others, from the aspects of security and efficiency.

Authenticated medical documents releasing under RSSs

Patient Registration



localhost:8084/MedicalDocument/PatientRegistration.jsp

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Navigation

- Home
- Patient
- Doctor
- Cloud Server
- Health Care Provider
- Insurance Company

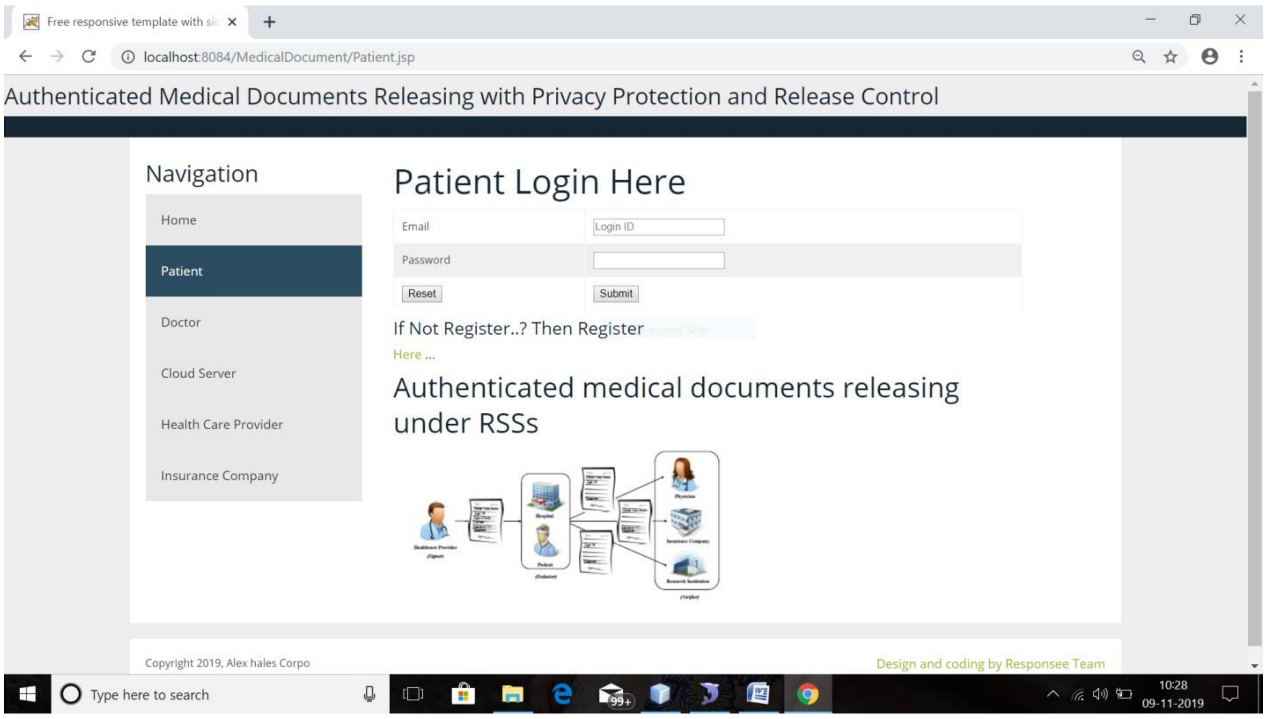
Patient Register Here

First Name	
Last Name	
Email	Login ID
Password	
Mobile	
Date of Birth	dd-mm-yyyy
Gender	Male Female
Address	
Select Profile Picture	Choose File No file chosen
Reset	Submit

If already Register..? Then Login Here ...

Authenticated medical documents releasing under RSSs

Patient Login


Free responsive template with sic

localhost:8084/MedicalDocument/Patient.jsp

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Navigation

- Home
- Patient
- Doctor
- Cloud Server
- Health Care Provider
- Insurance Company

Patient Login Here

Email	Login ID
Password	
Reset	Submit

If Not Register..? Then Register Here ...

Authenticated medical documents releasing under RSSs

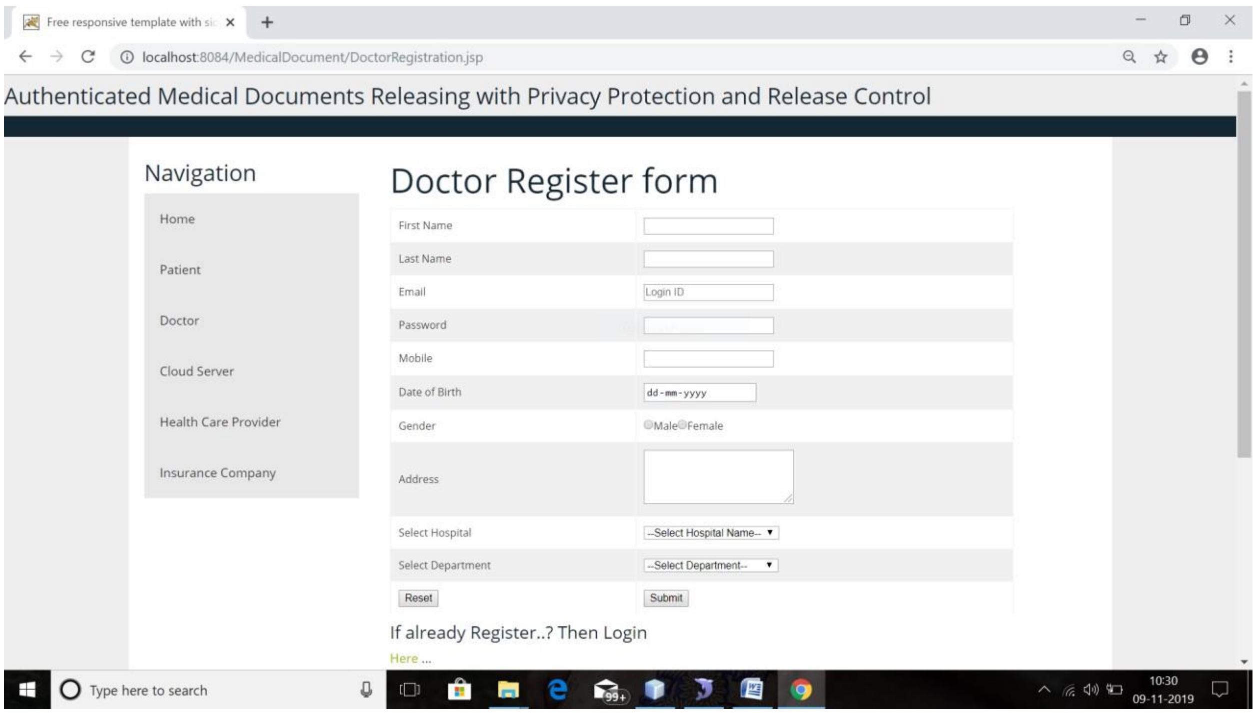
Copyright 2019, Alex hales Corpo

Design and coding by Responsee Team

Type here to search

10:28 09-11-2019

Doctor Registration


Free responsive template with sic

localhost:8084/MedicalDocument/DoctorRegistration.jsp

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Navigation

- Home
- Patient
- Doctor
- Cloud Server
- Health Care Provider
- Insurance Company

Doctor Register form

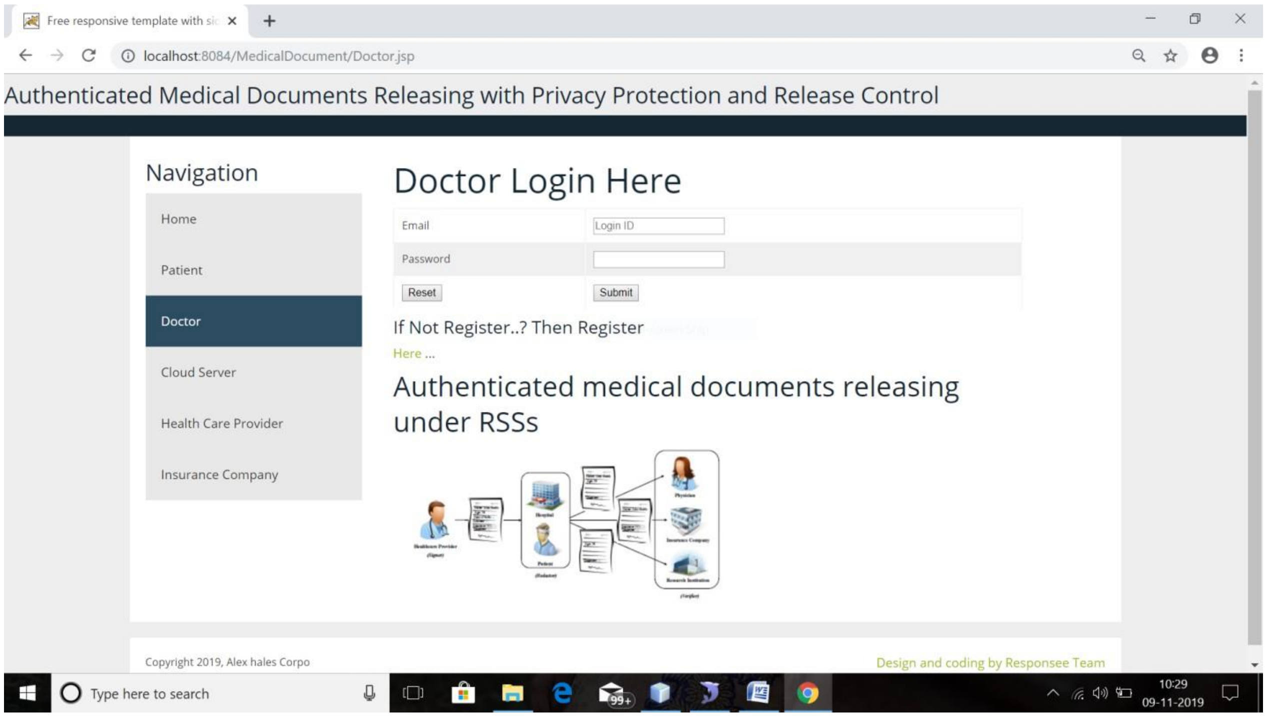
First Name	
Last Name	
Email	Login ID
Password	
Mobile	
Date of Birth	dd-mm-yyyy
Gender	Male Female
Address	
Select Hospital	--Select Hospital Name--
Select Department	--Select Department--
Reset	Submit

If already Register..? Then Login Here ...

Type here to search

10:30 09-11-2019

Doctor Login



Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Doctor Login Here

Email	Login ID
Password	

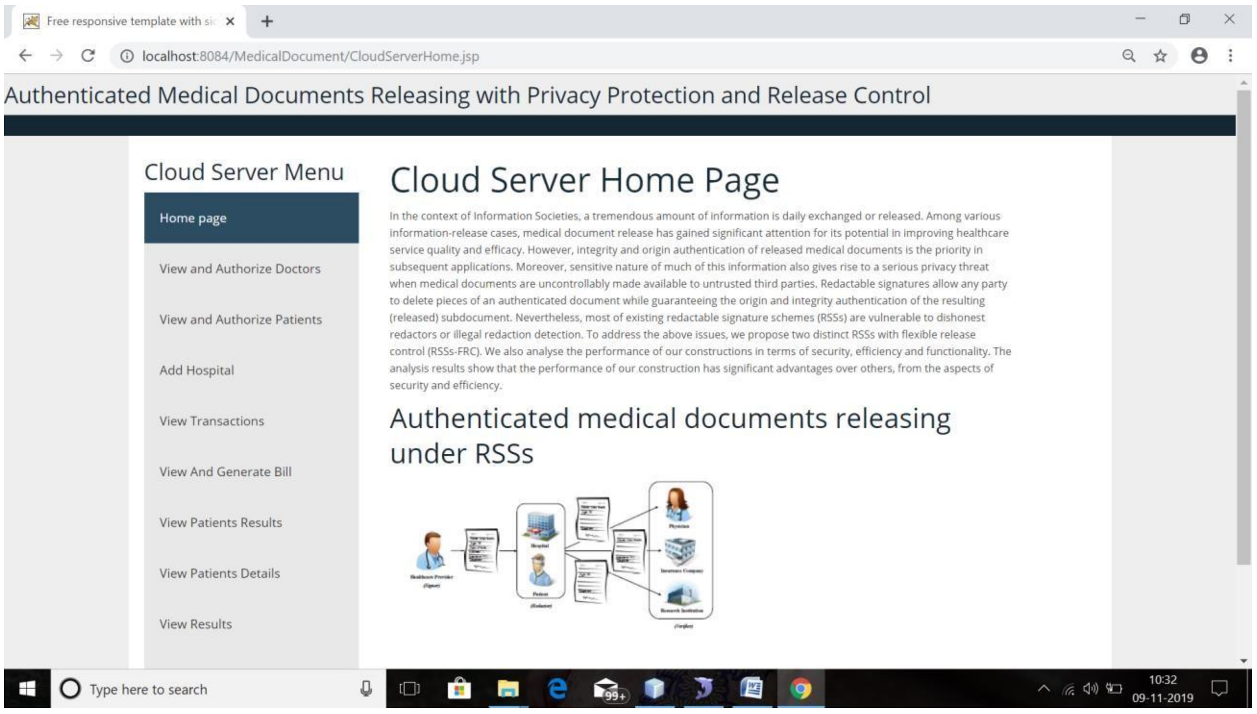
If Not Register..? Then Register Here ...

Authenticated medical documents releasing under RSSs

Copyright 2019, Alex hales Corpo

Design and coding by Responsee Team

Cloud Server Home Page



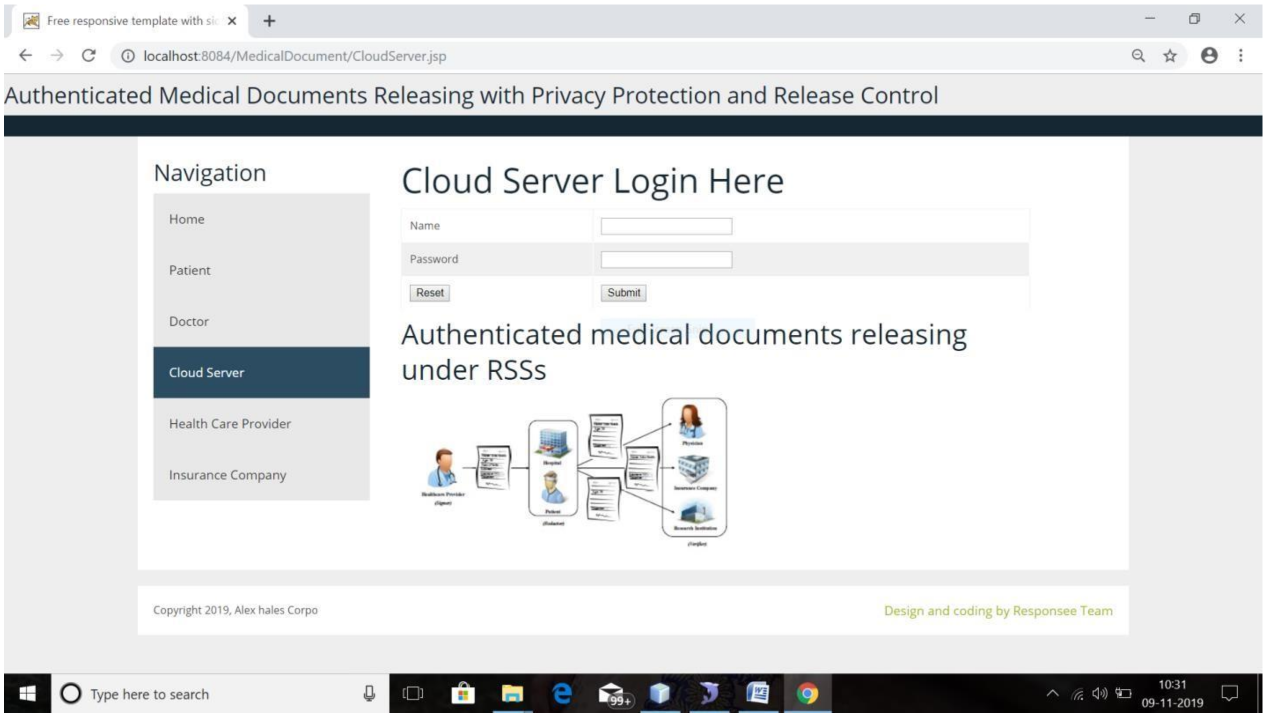
Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Cloud Server Home Page

In the context of Information Societies, a tremendous amount of information is daily exchanged or released. Among various information-release cases, medical document release has gained significant attention for its potential in improving healthcare service quality and efficacy. However, integrity and origin authentication of released medical documents is the priority in subsequent applications. Moreover, sensitive nature of much of this information also gives rise to a serious privacy threat when medical documents are uncontrollably made available to untrusted third parties. Redactable signatures allow any party to delete pieces of an authenticated document while guaranteeing the origin and integrity authentication of the resulting (released) subdocument. Nevertheless, most of existing redactable signature schemes (RSSs) are vulnerable to dishonest redactors or illegal redaction detection. To address the above issues, we propose two distinct RSSs with flexible release control (RSSs-FRC). We also analyse the performance of our constructions in terms of security, efficiency and functionality. The analysis results show that the performance of our construction has significant advantages over others, from the aspects of security and efficiency.

Authenticated medical documents releasing under RSSs

Cloud Server Login


Free responsive template with si

localhost:8084/MedicalDocument/CloudServer.jsp

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Navigation

- Home
- Patient
- Doctor
- Cloud Server
- Health Care Provider
- Insurance Company

Cloud Server Login Here

Name	
Password	
Reset	Submit

Authenticated medical documents releasing under RSSs

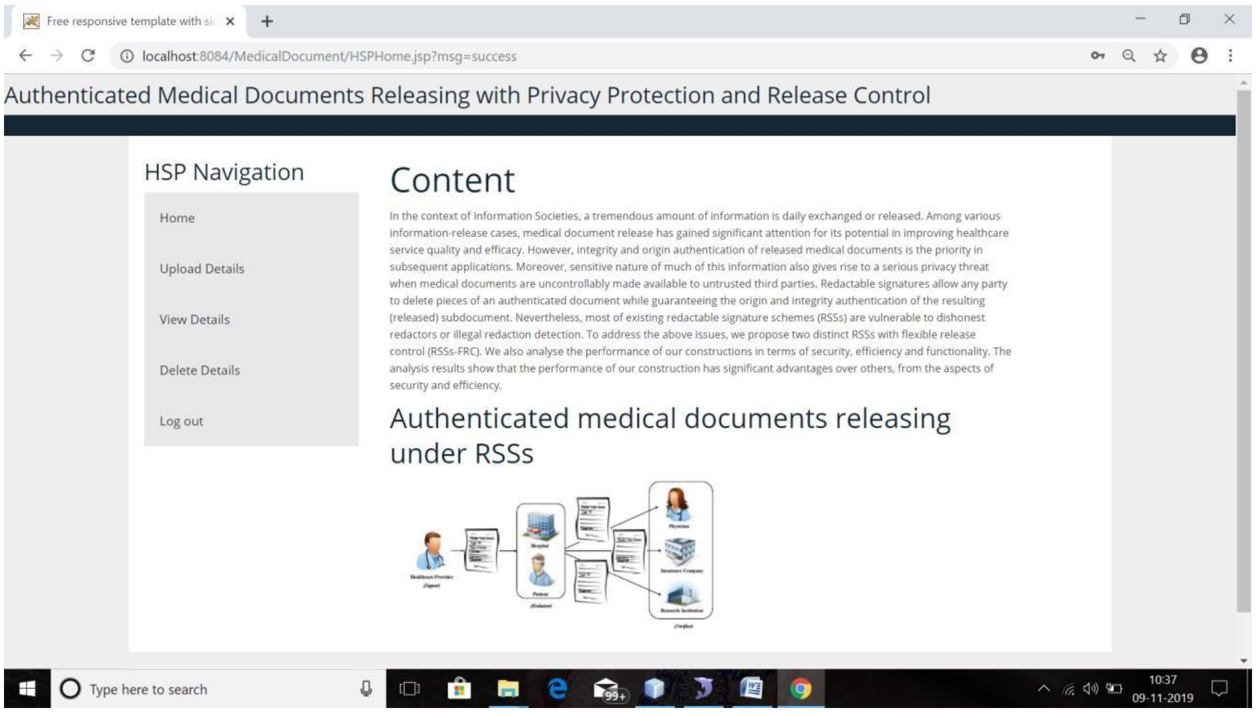
Copyright 2019, Alex hales Corpo

Design and coding by Responsee Team

Type here to search

10:31
09-11-2019

HSP Home Page


Free responsive template with si

localhost:8084/MedicalDocument/HSPHome.jsp?msg=success

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

HSP Navigation

- Home
- Upload Details
- View Details
- Delete Details
- Log out

Content

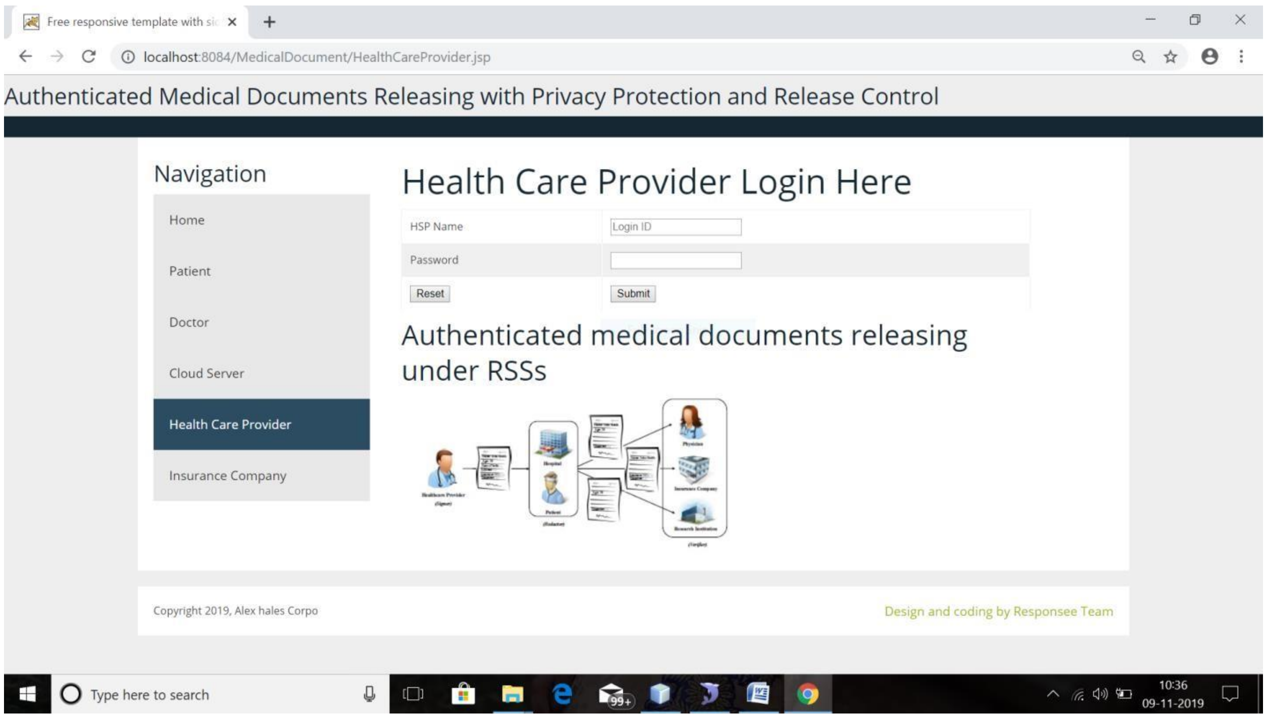
In the context of Information Societies, a tremendous amount of information is daily exchanged or released. Among various information-release cases, medical document release has gained significant attention for its potential in improving healthcare service quality and efficacy. However, integrity and origin authentication of released medical documents is the priority in subsequent applications. Moreover, sensitive nature of much of this information also gives rise to a serious privacy threat when medical documents are uncontrollably made available to untrusted third parties. Redactable signatures allow any party to delete pieces of an authenticated document while guaranteeing the origin and integrity authentication of the resulting (released) subdocument. Nevertheless, most of existing redactable signature schemes (RSSs) are vulnerable to dishonest redactors or illegal redaction detection. To address the above issues, we propose two distinct RSSs with flexible release control (RSSs-FRC). We also analyse the performance of our constructions in terms of security, efficiency and functionality. The analysis results show that the performance of our construction has significant advantages over others, from the aspects of security and efficiency.

Authenticated medical documents releasing under RSSs

Type here to search

10:37
09-11-2019

HSP Login


Free responsive template with si

localhost:8084/MedicalDocument/HealthCareProvider.jsp

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Navigation

- Home
- Patient
- Doctor
- Cloud Server
- Health Care Provider
- Insurance Company

Health Care Provider Login Here

HSP Name	Login ID
Password	
Reset	Submit

Authenticated medical documents releasing under RSSs

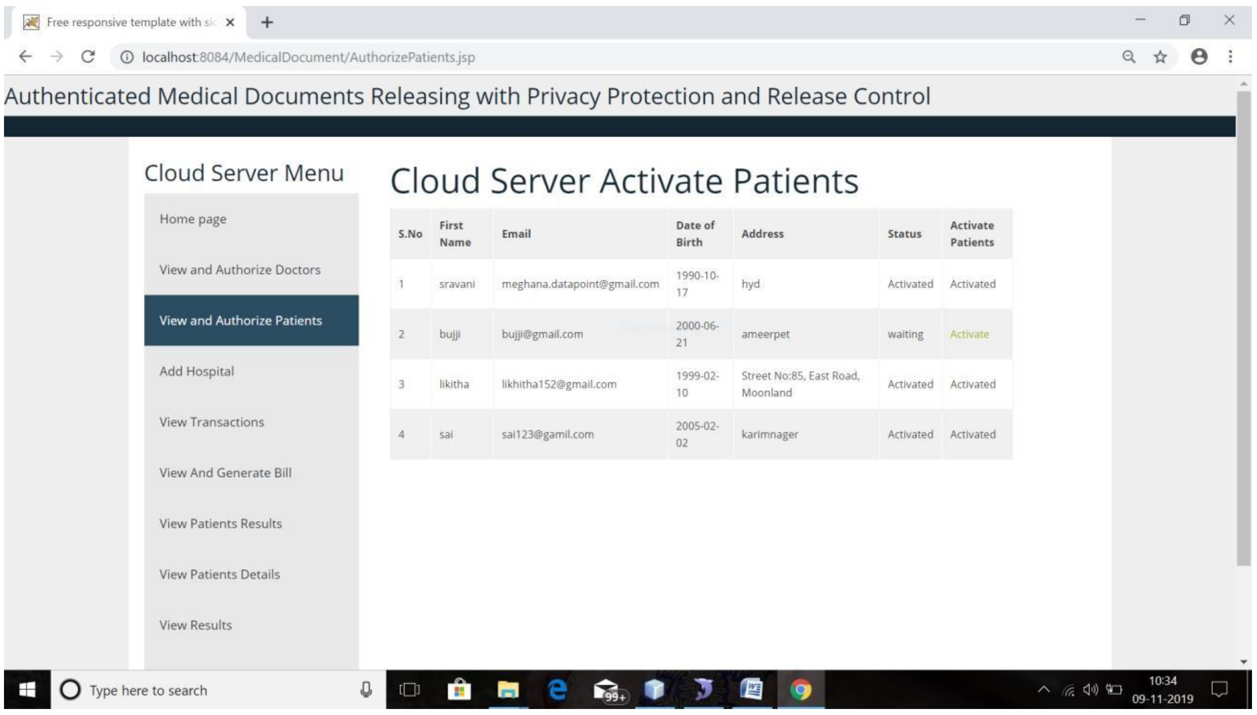
Copyright 2019, Alex hales Corpo

Design and coding by Responsee Team

Type here to search

10:36 09-11-2019

View And Authorize Patients


Free responsive template with si

localhost:8084/MedicalDocument/AuthorizePatients.jsp

Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Cloud Server Menu

- Home page
- View and Authorize Doctors
- View and Authorize Patients
- Add Hospital
- View Transactions
- View And Generate Bill
- View Patients Results
- View Patients Details
- View Results

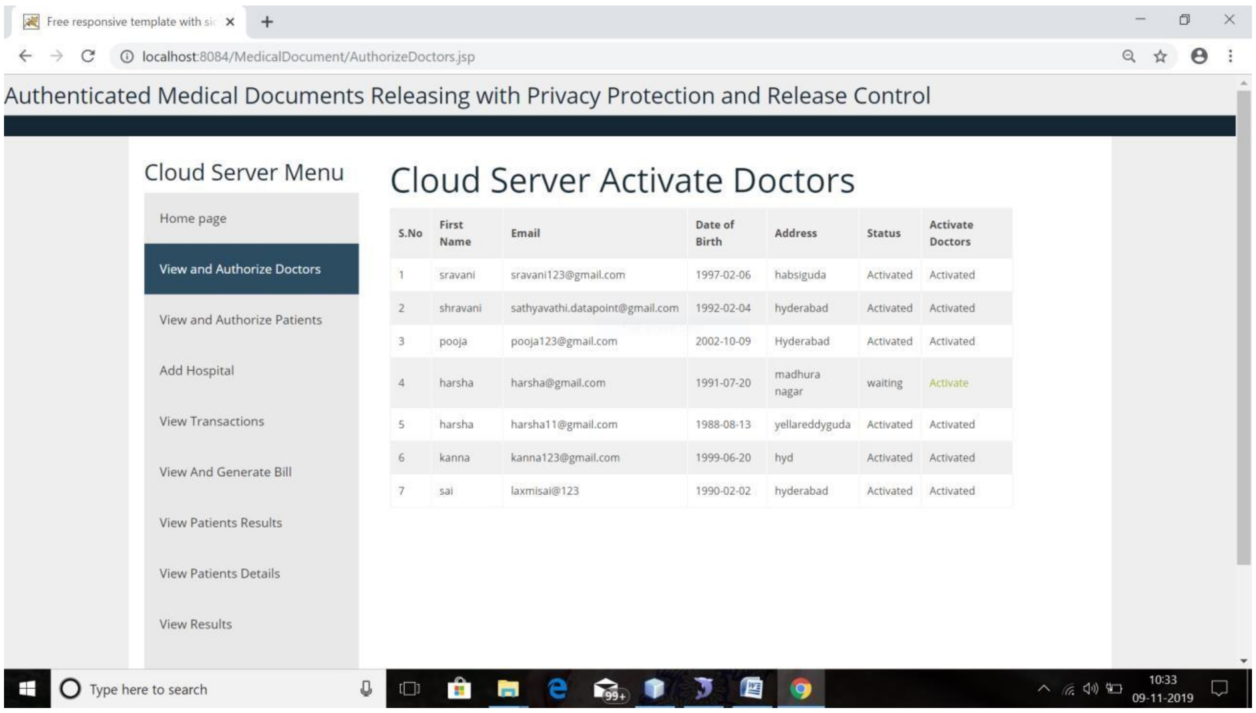
Cloud Server Activate Patients

S.No	First Name	Email	Date of Birth	Address	Status	Activate Patients
1	sravani	meghana.datapoint@gmail.com	1990-10-17	hyd	Activated	Activated
2	bujji	bujji@gmail.com	2000-06-21	ameerpet	waiting	Activate
3	likitha	likhitha152@gmail.com	1999-02-10	Street No:85, East Road, Moonland	Activated	Activated
4	sai	sai123@gamil.com	2005-02-02	karimnager	Activated	Activated

Type here to search

10:34 09-11-2019

View And Authorize Doctors

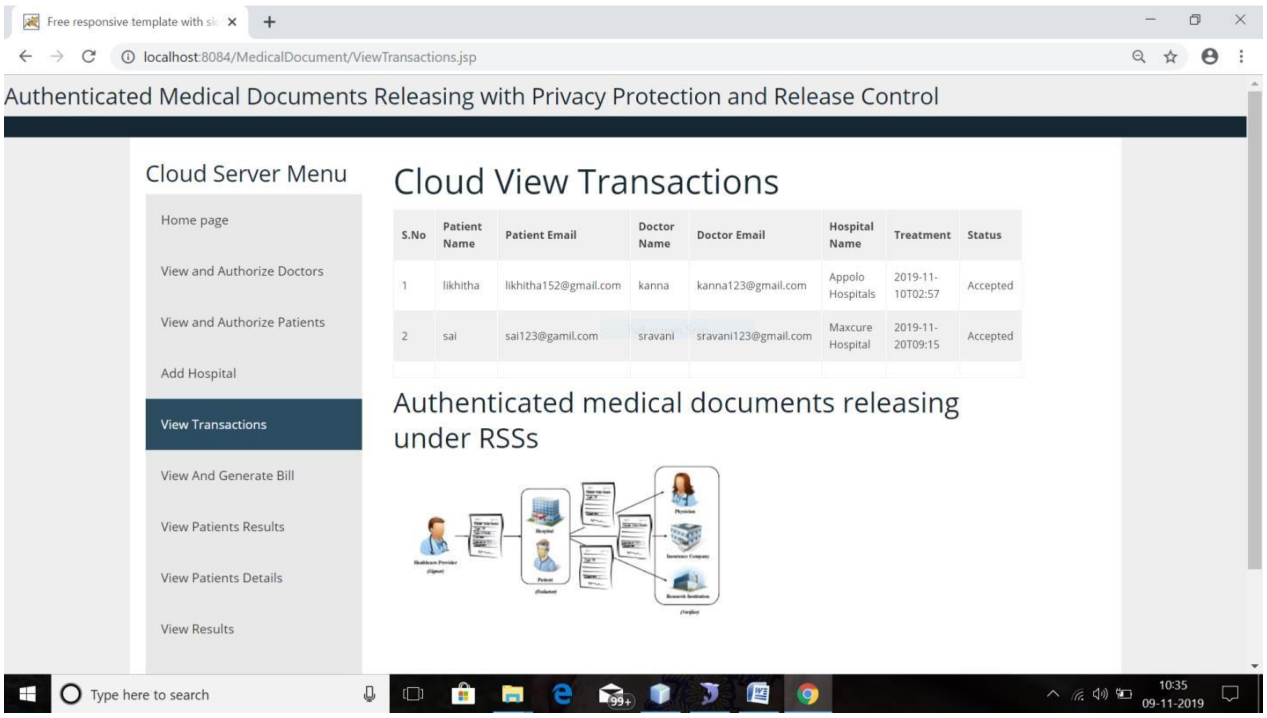


Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Cloud Server Activate Doctors

S.No	First Name	Email	Date of Birth	Address	Status	Activate Doctors
1	sravani	sravani123@gmail.com	1997-02-06	habsiguda	Activated	Activated
2	shravani	sathyavathi.datapoint@gmail.com	1992-02-04	hyderabad	Activated	Activated
3	pooja	pooja123@gmail.com	2002-10-09	Hyderabad	Activated	Activated
4	harsha	harsha@gmail.com	1991-07-20	madhura nagar	waiting	Activate
5	harsha	harsha11@gmail.com	1988-08-13	yellareddyguda	Activated	Activated
6	kanna	kanna123@gmail.com	1999-06-20	hyd	Activated	Activated
7	sai	laxmisai@123	1990-02-02	hyderabad	Activated	Activated

View Transactions



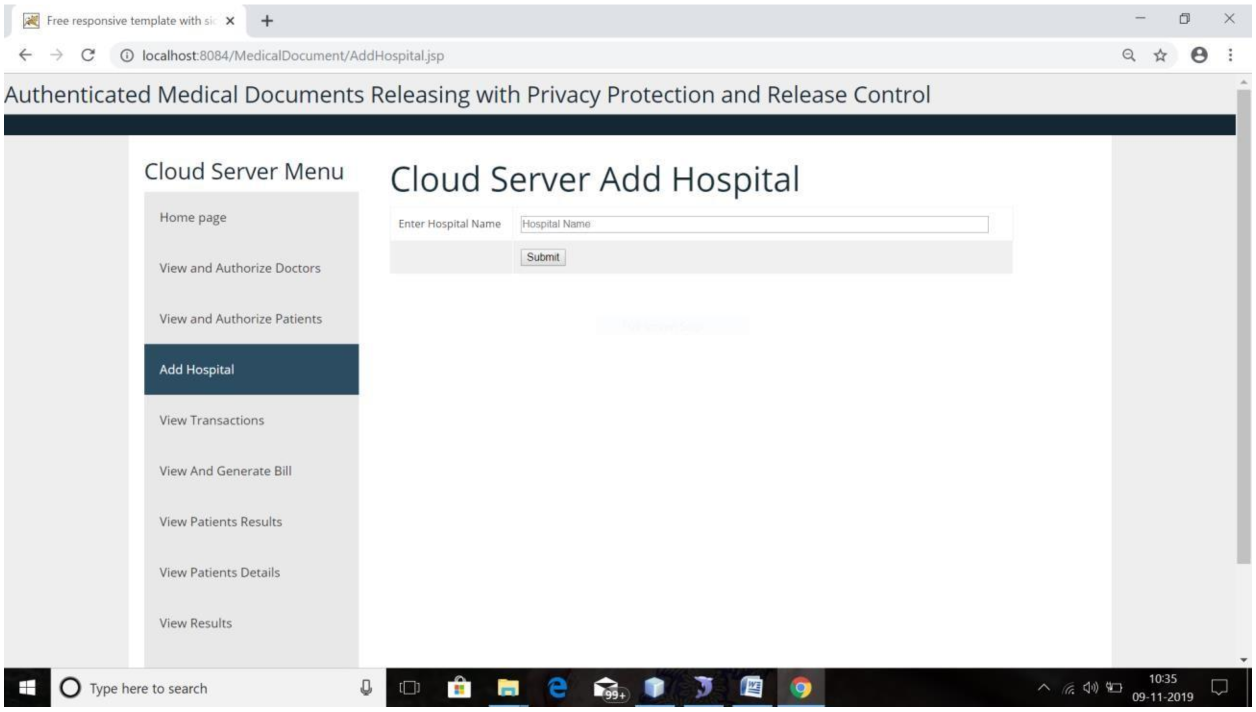
Authenticated Medical Documents Releasing with Privacy Protection and Release Control

Cloud View Transactions

S.No	Patient Name	Patient Email	Doctor Name	Doctor Email	Hospital Name	Treatment	Status
1	likhitha	likhitha152@gmail.com	kanna	kanna123@gmail.com	Appolo Hospitals	2019-11-10T02:57	Accepted
2	sai	sai123@gamil.com	sravani	sravani123@gmail.com	Maxcure Hospital	2019-11-20T09:15	Accepted

Authenticated medical documents releasing under RSSs

Add Hospital


Free responsive template with si
localhost:8084/MedicalDocument/AddHospital.jsp
Authenticated Medical Documents Releasing with Privacy Protection and Release Control
Cloud Server Menu
Home page
View and Authorize Doctors
View and Authorize Patients
Add Hospital
View Transactions
View And Generate Bill
View Patients Results
View Patients Details
View Results
Cloud Server Add Hospital
Enter Hospital Name
Hospital Name
Submit
Type here to search
10:35
09-11-2019

Upload Patient Details

11. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a secure cloud-based EHR framework that guarantees the security and privacy of medical data stored in the cloud, relying on hierarchical multi-authority CP-ABE to enforce access control policies. The proposed framework provides a high level of integration, interoperability, and sharing of EHRs among healthcare providers, patients, and practitioners. In the framework, the attribute domain authority manages a different attribute domain and operates independently. In addition, no computational overhead is completed by the government authority, and multi-factor applicant authentication have been identified and proofed. The proposed scheme can be adopted by any government that has a cloud computing infrastructure and provides treatment services to the majority of citizen patients. Future work includes implementing and evaluating the proposed scheme in a real-world environment.

12. REFERENCES

[1] Masrom, Maslin, and Ailar Rahimli. "A Review of Cloud Computing Technology Solution for Healthcare System." Research Journal of Applied Sciences, Engineering and Technology 8, no. 20 (2014): 2150–2155.

[2] HUCÍKOVÁ, Anežka, and Ankica Babic. "Cloud Computing in Healthcare: A Space of Opportunities and Challenges." Transforming Healthcare with the Internet of Things (2016): 122.

[3] Yang, Haibo, and Mary Tate. "A descriptive literature review and classification of cloud computing research." CAIS 31 (2012): 2.

[4] Zissis, Dimitrios, and Dimitrios Lekkas. "Addressing cloud computing security issues." Future Generation computer systems 28, no. 3 (2012): 583–592.

[5] Nigam, Vaibhav Kamal, and Shubham Bhatia. "Impact of Cloud Computing on Health Care." (2016). [6] ―How to Improve Healthcare with Cloud Computing‖, By Hitachi Data Systems, white paper, (2012). [7] Mehraeen, Esmaeil, Marjan Ghazisaeedi, Jebraeil Farzi, and Saghar Mirshekari. "Security Challenges in Healthcare Cloud Computing: A Systematic Review." Global Journal of Health Science 9, no. 3 (2016): 157.

[8] Sun, Dawei, Guiran Chang, Lina Sun, and Xingwei Wang. "Surveying and analyzing security, privacy and trust issues in cloud computing environments." Procedia Engineering 15 (2011): 2852–2856.

[9] Khan, Nabeel, and Adil Al-Yasiri. "Identifying cloud security threats to strengthen cloud computing adoption framework." Procedia Computer Science 94 (2016): 485–490.

[12] Reddy, B. Eswara, TV Suresh Kumar, and Gandikota Ramu. "An efficient cloud framework for health care monitoring system." In Cloud and Services Computing (ISCOS), 2012 International Symposium on, pp. 113-117. IEEE, 2012.

[13] Parekh, Maulik, and B. Saleena. "Designing a cloud based framework for healthcare system and applying clustering techniques for region wise diagnosis." Procedia Computer Science 50 (2015): 537–542.

[16] Zhiwei Yu, Chaokun Wang, Clark Thomborson, Jianmin Wang, Shiguo Lian and Athanasios V. Vasilakos, A novel watermarking method for software protection in the cloud, SOFTWARE – PRACTICE AND EXPERIENCE, 42:409–430, 2012.

[17] https://www.yesser.gov.sa/en/Pages/default.aspx

[18] Huang, Jie, Mohamed Sharaf, and Chin-Tser Huang. "A hierarchical framework for secure and scalable ehr sharing and access control in multi-cloud." In Parallel Processing Workshops (ICPPW), 2012 41st International Conference on, pp. 279–287. IEEE, 2012.

[19] Huang, Qinlong, Yixian Yang, and Mansuo Shen. "Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing." Future Generation Computer Systems72 (2017): 239–249.

[22] Li, Qinyi, Hu Xiong, Fengli Zhang, and Shengke Zeng. "An expressive decentralizing kp-abe scheme with constant-size ciphertext." IJ Network Security 15, no. 3 (2013): 161–170.

[23] Li, Qi, Jianfeng Ma, Rui Li, Ximeng Liu, Jinbo Xiong, and Danwei Chen. "Secure, efficient and revocable multi-authority access control system in cloud storage." Computers & Security 59 (2016): 45–59.

[27] Boneh, Dan, and Matt Franklin. "Identity-based encryption from the Weil pairing." In Annual international cryptology conference, pp. 213-229. Springer, Berlin, Heidelberg, 2001.

[28] Sahai, Amit, and Brent Waters. "Fuzzy identity-based encryption." In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 457-473. Springer, Berlin, Heidelberg, 2005.

[29] Li, Ming, et al. "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption." IEEE transactions on parallel and distributed systems 24.1 (2013): 131-143.

[30] Wang, Guojun, Qin Liu, and Jie Wu. "Hierarchical attributebased encryption for fine-grained access control in cloud storage services." In Proceedings of the 17th ACM conference on Computer and communications security, pp. 735-737. ACM, 2