Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications

INTRODUCTION

Recent advances in biosensors, wireless network and embedded systems have assisted the rapid development of a wide range of wearable and implantable sensors in the human body. To collect crucial health data such as blood pressure level, and heart rate, many smart phone based health applications have been developed in the recent past [1], [2]. The data from the sensors is sent to the cloud server, where hospitals have hosted their services for data processing. The data is analyzed to improve the level of healthcare given to the patients. An example of smart cloud based health applications is shown in Fig. 1. Ideally, patients want hospitals to assist them with high efficiency without revealing patients’ identities.The increasing necessity for massive computation and excessive amounts of storage, is driving the healthcare industry to use cloud based servers, because of many advantages they are offering, such as cost saving and scalability.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **LIST OF TABLES** | i  v  vii  xi  xii |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   1.3.1EXISTINGSYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 :PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.2.3 MODULE DIAGRAM  2.2.4GIVEN INPUTAND EXPECTED OUTPUT  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4 : SYSTEM DESIGN**  **4.1 general**  4.1.1 activity diagram  4.1.2 USE CASE DIAGRAM  4.1.3 DATA FLOW DIAGRAM  4.1.4SEQUENCE DIAGRAM  4.1.5 COLLABORATION DIAGRAM  4.1.6CLASS DIAGRAM  4.1.7 SYSTEM ARCHITECTURE  4.1.8 OBJECT DIAGRAM  4.1.9 STATE DIAGRAM  4.1.10 COMPONENT DIAGRAM  4.1.11 E-R DIAGRAM  4.2 DATABASE DESIGN (ALL LEVEL |  |
| 5. | **CHAPTER 5 :SOFTWARE SPECIFICATION**  5.1 general  FRONT END  5.2 features of java  5.2.1 the java framework  5.2.2 objective of java  5.2.3 componets of java frmework  5.3 features of the common language runtime  5.4 j2ee  5.4.1 EVALUTION OF J2EE  5.4.2 THE J2EE DATA ARCHITECTURE  BACK END  5.6 features of SQL SERVER |  |
| 6. | **CHAPTER 6 : IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION  6.3 DATA BASE TABLE STRUCTURE |  |

|  |  |  |
| --- | --- | --- |
| 7. | **CHAPTER 7 : SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 : SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 :**  **APPLICATIONS AND FUTURE ENHANCEMENT**  9.1 GENERAL  9.2 APPLICATIONS  9.3 FUTURE ENHANCEMENTS |  |
| **10** | **CHAPTER 10 :**  10.1CONCLUSION  10.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2 | Activity Diagram |  |
| 4.3 | Use case Diagram |  |
| 4.4 | Data flow diagram |  |
| 4.5 | Sequence diagram |  |
| 4.6 | Collaboration diagram |  |
| 4.7 | Class diagram |  |
| 4.8 | Architecture Diagram |  |
| 4.9 | State Diagram |  |
| 4.1 | Component Diagram |  |
| 4.12 | E-R Diagram |  |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F inal state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Usecase |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |

|  |  |  |  |
| --- | --- | --- | --- |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | DataBase |
| 2. | JVM | Java Virtual Machine |
| 3. | JSP | Java Server Page |
| 4. | PWS | Personalised Web Search |
| 5. | UPS | User Personalised Search |
| 6. | JRE | Java Runtime Environment |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

Many smart healthcare applications are adopting cloud to provide services to patients. However, the sensitive data can be disclosed to the authentication server/service provider. Therefore, security and privacy are crucial to its success and deployment at large scale. Patients don’t want to disclose their identities to the cloud server. One way to protect their identities from cloud server is anonymous authentication.

**1.2. OBJECTIVE**

The performance of our scheme is evaluated by theoretical analysis which demonstrates that it resists various attacks and provides several attractive security features.

* 1. **Existing System:**

In existing system now a days in hospitals lot of patients are coming to hospital the hospital management don’t know weather who is coming to admit in hospital how is going to register they are not identifying. And what is the problem also they do not understand clearly. If any emergency case is there they are not getting operation at correct time to overcome all those problems we are going to implement some methods in proposed system.

**LITERATURE SURVEY**

**TITLE:** The pursuit of citizens’ privacy: a privacy aware smart city is possible

**AUTHOR:** Martinez-Balleste, P. A. Perez-Martinez, and A. Solanas,

**YEAR: 2013**

**DESCRIPTION:**

Cities are growing steadily, and the process of urbanization is a common trend in the world. Although cities are getting bigger, they are not necessarily getting better. With the aim to provide citizens with a better place to live, a new concept of a city was born: the smart city. The real meaning of smart city is not strictly defined, but it has gained much attention, and many cities are taking action in order to be considered 'smart'. These smart cities, founded on the use of information and communication technologies, aim at tackling many local problems, from local economy and transportation to quality of life and e-governance. Although technology helps to solve many of these local problems, their ability to gather unprecedented amounts of information could endanger the privacy of citizens. In this article we identify a number of privacy breaches that can appear within the context of smart cities and their services. We leverage some concepts of previously defined privacy models and define the concept of citizens' privacy as a model with five dimensions: identity privacy, query privacy, location privacy, footprint privacy and owner privacy. By means of several examples of smart city services, we define each privacy dimension and show how existing privacy enhancing technologies could be used to preserve citizens' privacy.

**TITLE:** Real time tele-monitoring of patients with chronic heart-failure using a smartphone: lessons learned

**AUTHOR:** D. Aranki, G. Kurillo, P. Yan, D. M. Liebovitz, and R. Bajcsy

**YEAR:** 2016

**DESCRIPTION:**

We present a smartphone-based system for real-time tele-monitoring of physical activity in patients with chronic heart-failure (CHF). We recently completed a pilot study with 15 subjects to evaluate the feasibility of the proposed monitoring in the real world and examine its requirements, privacy implications, usability, and other challenges encountered by the participants and healthcare providers. Our tele-monitoring system was designed to assess patient activity via minute-by-minute energy expenditure (EE) estimated from accelerometry. In addition, we tracked relative user location via global positioning system (GPS) to track outdoors activity and measure walking distance. The system also administered daily surveys to inquire about vital signs and general cardiovascular symptoms. The collected data were securely transmitted to a central server where they were analyzed in real time and were accessible to the study medical staff to monitor patient health status and provide medical intervention if needed. Although the system was designed for tele-monitoring individuals with CHF, the challenges, privacy considerations, and lessons learned from this pilot study apply to other chronic health conditions, such as diabetes and hypertension, that would benefit from continuous monitoring through mobile-health (mHealth) technologies.

**TITLE:** Security and privacy in cloud computing

**AUTHOR:** Z. Xiao and Y. Xiao,

**YEAR:2013**

**DESCRIPTION:** Data security has consistently been a major issue in information technology. In the cloud computing environment, it becomes particularly serious because the data is located in different places even in all the globe. Data security and privacy protection are the two main factors of user's concerns about the cloud technology. Though many techniques on the topics in cloud computing have been investigated in both academics and industries, data security and privacy protection are becoming more important for the future development of cloud computing technology in government, industry, and business. Data security and privacy protection issues are relevant to both hardware and software in the cloud architecture. This study is to review different security techniques and challenges from both software and hardware aspects for protecting data in the cloud and aims at enhancing the data security and privacy protection for the trustworthy cloud environment. In this paper, we make a comparative research analysis of the existing research work regarding the data security and privacy protection techniques used in the cloud computing.

**TITLE:** A smart health application and its related privacy issues

**AUTHOR:** D. Ding, M. Conti, and A. Solanas,

**YEAR: 2016**

**DESCRIPTION:**

Together with the development of technologies such as those for ubiquitous computing, data mining, Internet of Things (IoT) and wireless sensor networks (WSNs), the concepts of smart cities and mobile health (m-Health) have emerged. Along the same line, the smart health concept (s-Health), understood as a context-aware healthcare paradigm for smart environments, improves the quality of healthcare systems within smart cities. However, s-Health may encounter some privacy and security issues. For example, in order to obtain the current location and health conditions of citizens, these citizens might be continuously monitored, which could be seen as a privacy invasion. In this paper, we describe an application within the s-Health paradigm. In particular, our approach allows to effectively deal with citizens who have respiratory conditions. Our application example suggests low-pollution routes to citizens in order to lessen their respiratory-related problems, and proactively activates water sprays in fountains to reduce the effect of pollution or pollen. Besides the description of the application, the main contribution of the article is the analysis of the emerging privacy issues of the proposed application and the discussion of possible countermeasures.

**TITLE:** Untraceable sensor movement in distributed iot infrastructure

**AUTHOR:** P. Gope and T. Hwang

**YEAR: 2015**

**DESCRIPTION:**

Recent advances in information and communication technologies and embedded systems have given rise to a new disruptive technology, the Internet of Things (IoTs). IoT allows people and objects in the physical world as well as data and virtual environments to interact with each other so as to create smart environments, such as smart transport systems, smart cities, smart health, and so on. However, IoT raises some important questions and also introduces new challenges for the security of systems and processes and the privacy of individuals, such as their location and movements and so on. In this paper, at first, we propose a distributed IoT system architecture. Subsequently, we propose an anonymous authentication scheme, which can ensure some of the notable properties, such as sensor anonymity, sensor untraceability, resistance to replay attacks, cloning attacks, and so on. It is argued that the proposed authentication scheme will be useful in many distributed IoT applications (such as radio-frequency identification-based IoT system, Biosensor-based IoT healthcare system, and so on), where the privacy of the sensor movement is greatly desirable.

* 1. **Proposed System**

In proposed system to overcome all these type of problems while joining to hospital they have to register first .After registration admin will provide one id to that patient. by using that id they have to login.

**1.4.1 Proposed System Advantage**

* After login of that page where is one menu under that for that patient what the user need they have to send a request to admin for what the requirement they need .
* If admin will receive that request. Immediately .they will respond and they will provide concern person .

**Given Input and Expected output**

User Interface Design

Input : Enter Login name and Password

Output : If valid user name and password then directly open the home page otherwise show error message and redirect to the registration page.

User

Input : login and send request to GM and CSP

Output: get ID and key then use the services provided by the GM / Hospital

Gm

Input : login and view user request and add doctors

Output : Control the services and view doctors.

 CSP

Input : login and view requests of the users

Output : generate key to the user give permission to access services

services

Input : ID and key

Output: view the services and use the services

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Various sharp restorative administrations applications are grasping cloud to offer organizations to patients. In any case, the unstable data can be uncovered to the approval server/pro center. Thusly, security and assurance are fundamental to its thriving and sending wherever scale.

In this project we have the following modules:

1. **User Interface**
2. **Group Manager**
3. **User**
4. **Cloud Service Provider**
5. **Services**

**User Interface:**

This is the first module of our project. The important role for the user is to move login window to user window. This module has created for the security purpose. In this login page we have to enter login user id and password. It will check username and password is match or not (valid user id and valid password). If we enter any invalid username or password we can’t enter into login window to user window it will shows error message. So we are preventing from unauthorized user entering into the login window to user window. It will provide a good security for our project. So server contain user id and password server also check the authentication of the user. It well improves the security and preventing from unauthorized user enters into the network. In our project we are using JSP for creating design. Here we validate the login user and server authentication.

Database

Welcome Page

Registration

Page

Login

Server

Username Yes

Verify

Password

No

**Group Manager**

Group Manager is a first module of this project. This GM is the owner of the hospital. In this module GM login, and then view the all users request and accept the users and generate ID to that users. GM sends accepted user’s details to cloud service provider to provide the hospital services to the users. And GM also add the doctors based on the diseases. He can view the total doctors details.

Login

Verify

Error page

Home Page

Accept Request

Sends User ID to CSP

Generate ID to user

Logout

View Doctors

Invalid

Valid

**User:**

User is the second module of this project. In this module user can register with Group Manager. Send request to get the ID after getting ID user can login to the site, send request to the Cloud Service Provider to access the hospital services. After receiving the key from CSP, user can use the services. If the user is authenticated then only he can use the services otherwise that user is deleted/removed by the CSP.

Register / Login

Verify

Error page

Home Page

Send request to CSP for Key

Send request to GM for ID

Use the services provided by the GM

View First Aid Information

**Cloud Service Provider**

Cloud Service Provider is the third module of the project. CSP login and he can view the patient request , if the patient is authorized then only key will generated to the patient otherwise he can removed without using the services.

Login

Verify

Error page

Home Page

Accept Request

Provide services

Generate key to user

Logout

View ID’s

5. Services

Services are the fourth module of the project. In this module patient can login into the services page and he can view the services provided by the GM/Hospital based on his requirement he can send request to the particular service to access.

Provide Patient ID & Secret Code

Logout

Access the Service

Send request to Doctor

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

Various sharp restorative administrations applications are grasping cloud to offer organizations to patients. In any case, the unstable data can be uncovered to the approval server/pro center. Thusly, security and assurance are fundamental to its thriving and sending wherever scale. Patients would incline toward not to reveal their identities to the cloud server. One way to deal with shield their identities from cloud server is puzzling approval.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shouls what the system do and not how it should be implemented.

**HARDWARE REQUIREMENTS**

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 2GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

**SOFTWARE REQUIREMENTS**

* FRONT END : J2EE (JSP, SERVLET)
* BACK END : MY SQL 5.5
* OPERATING SYSTEM : WINDOWS 7

IDE : ECLIPSE

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**UML Diagrams**

**Use Case Diagram**

****

**EXPLANATION:**

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. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

Class Diagram



**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**State Diagram**

****

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**Activity Diagram**



**EXPLANATION:**

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.

**Sequence Diagram**



**EXPLANATION:**

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.

**Collaboration Diagram**



**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

..

**Object Diagram**



**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security

**Deployment Diagram**



**Component Diagram**



Data Flow Diagram

**DB**

Login

Provide ID

View request

View Doctors

Add Doctors

**DB**

Access services

Send request to CSP

Get ID from GM

Send request to GM

Enter ID &Key

View services

Get key from CSP

Login

Provide services to user

View request

Login

Generate key to user

E-R Diagram

Login

Add doctors

Provide ID to users

View requst

Login

Send request to GM

Get key from CSP

Send request to CSp

Get ID from Gm

login

View request

Generate key to user

Provide services to users

Access services

**EXPLANATION:**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**System Architecture:**

Register / Login

Verify

Error page

Home Page

Send request to CSP for Key

Send request to GM for ID

Use the services provided by the GM

View First Aid Information

Login

Verify

Error page

Home Page

Accept Request

Provide services

Generate key to user

Logout

View ID’s

Provide Patient ID & Secret Code

Logout

Access the Service

Send request to Doctor

**EXPLANATION:**

The approval process usually incorporates disclosing customers' private information for instance, username and mystery key to the approval server. If the patient can be associated or taken after by the affirmation server or then again poisonous adversaries by their sales, their insurance can be broken. Most of the present security sparing therapeutic administrations applications give anonymity from the foes. Regardless, not a lot of them give anonymity from the approval server. In this paper, we have proposed a structure which gives complete security an

**CHAPTER 5**

**DEVELOPMENT TOOLS**

* 1. **GENERAL**

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA. The Primary languages are JAVA,J2EE and J2ME. In this project J2EE is chosen for implementation.

**5.2 FEATURES OF JAVA**

**5.2.1 THE JAVA FRAMEWORK**

**Java** is a programming language originally developed by James Gosling at Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications the java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

**5.2.2 OBJECTIVES OF JAVA**

To see places of Java in Action in our daily life, explore java.com.

## Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

* Write software on one platform and run it on virtually any other platform
* Create programs to run within a Web browser and Web services
* Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
* Combine applications or services using the Java language to create highly customized applications or services
* Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

## Some Ways Software Developers Learn Java

Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

**ObjectOriented**

To be an Object Oriented language, any language must follow at least the four characteristics.

1. Inheritance   :It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse  the existing code and adding addition a features as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the  methods.

4. Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

**5.2.3 JAVA SWING OVERVIEW**

**Abstract Window Toolkit (AWT) is cross-platform**

Swing[[1]](http://en.wikibooks.org/wiki/Java_Programming/Swing#cite_note-0) provides many controls and widgets to build user interfaces with. Swing class names typically begin with a J such as JButton, JList, JFrame. This is mainly to differentiate them from their AWT counterparts and in general is one-to-one replacements. Swing is built on the concept of Lightweight components vs AWT and SWT's concept of Heavyweight components. The difference between the two is that the Lightweight components are rendered (drawn) using purely Java code, such as drawLine and drawImage, whereas Heavyweight components use the native operating system to render the components.

Some components in Swing are actually heavyweight components. The top-level classes and any derived from them are heavyweight as they extend the AWT versions. This is needed because at the root of the UI, the parent windows need to be provided by the OS. These top-level classes include JWindow, JFrame, JDialog and JApplet. All Swing components to be rendered to the screen must be able to trace their way to a root window of one of those classes.

**Note**: It generally it is not a good idea to mix heavyweight components with lightweight components (other than as previously mentioned) as you will encounter layering issues, e.g., a lightweight component that should appear "on top" ends up being obscured by a heavyweight component. The few exceptions to this include using heavyweight components as the root pane and for popup windows. Generally speaking, heavyweight components will render on top of lightweight components and will not be consistent with the look and feel being used in Swing. There are exceptions, but that is an advanced topic. The truly adventurous may want to consider reading this [article](http://java.sun.com/products/jfc/tsc/articles/mixing/) from Sun on mixing heavyweight and lightweight components.

**5.2.4 Evolution of Collection Framework:**

Almost all collections in Java are derived from the [**java.util.Collection**](http://download.oracle.com/javase/7/docs/api/java/util/Collection.html) interface. Collection defines the basic parts of all collections. The interface states the add() and remove() methods for adding to and removing from a collection respectively. Also required is the toArray() method, which converts the collection into a simple array of all the elements in the collection. Finally, the contains() method checks if a specified element is in the collection. The Collection interface is a subinterface of [**java.util.Iterable**](http://download.oracle.com/javase/7/docs/api/java/util/Iterable.html), so the iterator() method is also provided. All collections have an iterator that goes through all of the elements in the collection. Additionally, Collection is a generic. Any collection can be written to store any class. For example, Collection<String> can hold strings, and the elements from the collection can be used as strings without any casting required.

There are three main types of collections:

* Lists: always ordered, may contain duplicates and can be handled the same way as usual arrays
* Sets: cannot contain duplicates and provide random access to their elements
* Maps: connect unique keys with values, provide random access to its keys and may host duplicate values

**LIST**

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two concrete classes implement List. The first is java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it. The other implementation is java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.

SET:

Java's [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet,java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a [java.util.HashMap](http://download.oracle.com/javase/7/docs/api/java/util/HashMap.html) to store the hashes and elements and to prevent duplicates. Java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. [java.util.TreeSet](http://download.oracle.com/javase/7/docs/api/java/util/TreeSet.html) uses a red-black tree implemented by a [java.util.TreeMap](http://download.oracle.com/javase/7/docs/api/java/util/TreeMap.html). The red-black tree makes sure that there are no duplicates. Additionally, it allows Tree Set to implement java.util.SortedSet.

The [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method, or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet

[java.util.SortedSet](http://download.oracle.com/javase/7/docs/api/java/util/SortedSet.html) is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.

**MAP:**

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

**Thread:**

Simply put, a thread is a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time.

## Creating threads

Java's creators have graciously designed two ways of creating threads: implementing an interface and extending a class. Extending a class is the way Java inherits methods and variables from a parent class. In this case, one can only extend or inherit from a single parent class. This limitation within Java can be overcome by implementing interfaces, which is the most common way to create threads. (Note that the act of inheriting merely allows the class to be run as a thread. It is up to the class to start() execution, etc.)

Interfaces provide a way for programmers to lay the groundwork of a class. They are used to design the requirements for a set of classes to implement. The interface sets everything up, and the class or classes that implement the interface do all the work. The different set of classes that implement the interface have to follow the same rules.

**5.5 Conclusion**

Swing's high level of flexibility is reflected in its inherent ability to override the native host [operating system](http://en.wikipedia.org/wiki/Operating_system) (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. The Java thread scheduler is very simple. All threads have a priority value which can be changed dynamically by calls to the threads setPriority() method . Implementing the above concepts in our project to do the efficient work among the Server.

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding:**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>login page</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

<style type=*"text/css"*>

*.login*{

display: *inline-block*;

height: *400px*;

}

/\* Style The Dropdown Button \*/

*.dropbtn* {

background-color: *#4CAF50*;

color: *white*;

padding: *8px*;

font-size: *20px*;

border: *none*;

cursor: *pointer*;

}

*.dropdown* {

position: *relative*;

display: *inline-block*;

}

*.dropdown-content* {

display: *none*;

position: *absolute*;

background-color: *#f9f9f9*;

min-width: *160px*;

box-shadow: *0px 8px 16px 0px rgba(0,0,0,0.2)*;

z-index: *1*;

}

*.dropdown-content* **a** {

color: *black*;

padding: *12px 16px*;

text-decoration: *none*;

display: *block*;

}

*.dropdown-content* **a***:hover* {background-color: *#f1f1f1*}

*.dropdown:hover* *.dropdown-content* {

display: *block*;

}

*.dropdown:hover* *.dropbtn* {

}

</style>

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1><br><br>

<center>

<div class=*" w3-striped w3-green"* style="font-size: *20px*;">

<div class=*"w3-button "* style="width: *220px*;">

<a href=*"index.jsp"*>Home</a>

</div>

<div class=*"w3-button "* style="width: *220px*;">

<a href=*"Dlogin.jsp"*>GroupManager Login</a>

</div>

<div class=*"w3-button "* style="width: *220px*;">

<a href=*"login.jsp"*>User Login</a>

</div>

<!--<div class="w3-button" style="width: 220px;">

<a href="CSplogin.jsp">Cloud Provider Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="services.jsp">Services</a>

</div> -->

<div class=*"dropdown"*>

<button class=*"dropbtn"*><u>Cloud Service Provider</u></button>

<div class=*"dropdown-content "* style="width: *220px*;">

<a href=*"CSplogin.jsp"*>Cloud Provider Login</a>

<a href=*"services.jsp"*>Services</a>

</div>

</div>

<div class=*"dropdown"*>

<button class=*"dropbtn"*><u>First Aid</u></button>

<div class=*"dropdown-content"* class=*"w3-button"* style="width: *220px*;">

<a href=*"hattack.html"*>Heart Attack</a>

<a href=*"sugar.html"*>Diabetes</a>

<a href=*"elec.html"*>Electric shock</a>

<a href=*"knife.html"*>Knife cutting</a>

<a href=*"loose.html"*>Loose Motion</a>

<a href=*"nose.html"*>Nose bleeding</a>

<a href=*"pois.html"*>Poisoning</a>

<a href=*"vomit.html"*>Vomiting</a>

<a href=*"dog.html"*>Dog bite</a>

</div>

</div>

</div>

</div>

<br><br><br><br><br><br><br><br><br><br><br>

<p style="text-align: *justify*;font-size: *20px*;font-style: *italic*;color: *white*;">

Abstract-Many smart healthcare applications are adopting

cloud to provide services to patients. However, the sensitive data

can be disclosed to the authentication server/service provider.

Therefore, security and privacy are crucial to its success and deployment

at large scale. Patients don't want to disclose their identities

to the cloud server. One way to protect their identities from

cloud server is anonymous authentication. The authentication

process normally involves disclosing users private information

such as username and password to the authentication server. If

the patient can be linked or tracked by the authentication server

or malicious adversaries by their requests, their privacy can be

breached. Most of the existing privacy preserving health care

applications provide anonymity from the adversaries. However,

very few of them provide anonymity from the authentication

server. In this paper, we have proposed a system which provides

complete privacy and anonymity to the users of health care

applications from adversaries and the authentication server. In

our proposed authentication scheme, we have utilized rotating

group signature scheme based on Elliptic curve cryptography

(ECC) to provide anonymity to the patients. To add an extra layer

of protection, we have used The Onion Router (TOR) to provide

privacy at the network layer. The performance of our scheme

is evaluated by theoretical analysis which demonstrates that it

resists various attacks and provides several attractive security

features.</p></center>

</body>

</html>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>login page</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

<style type=*"text/css"*>

*.login*{

display: *inline-block*;

height: *400px*;

}

/\* Style The Dropdown Button \*/

*.dropbtn* {

background-color: *#4CAF50*;

color: *white*;

padding: *16px*;

font-size: *16px*;

border: *none*;

cursor: *pointer*;

}

*.dropdown* {

position: *relative*;

display: *inline-block*;

}

*.dropdown-content* {

display: *none*;

position: *absolute*;

background-color: *#f9f9f9*;

min-width: *160px*;

box-shadow: *0px 8px 16px 0px rgba(0,0,0,0.2)*;

z-index: *1*;

}

*.dropdown-content* **a** {

color: *black*;

padding: *12px 16px*;

text-decoration: *none*;

display: *block*;

}

*.dropdown-content* **a***:hover* {background-color: *#f1f1f1*}

*.dropdown:hover* *.dropdown-content* {

display: *block*;

}

*.dropdown:hover* *.dropbtn* {

background-color: *#3e8e41*;

}

</style>

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1><center>

<div class=*" w3-striped w3-green"* style="font-size: *20px*;">

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"index.jsp"*>Home</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"Dlogin.jsp"*>GroupManager Login</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"login.jsp"*>User Login</a>

</div>

<!--<div class="w3-button" style="width: 220px;">

<a href="CSplogin.jsp">Cloud Provider Login</a>

</div>

<!-- <div class="w3-button" style="width: 220px;">

<a href="services.jsp">Services</a>

</div> -->

<div class=*"dropdown"*>

<button class=*"dropbtn"* style="font-size: *20px*;"><u>Cloud Service Provider</u></button>

<div class=*"dropdown-content"* class=*"w3-button"* style="width: *220px*;">

<a href=*"CSplogin.jsp"*>Cloud Provider Login</a>

<a href=*"services.jsp"*>Services</a>

</div>

</div>

<div class=*"dropdown"*>

<button class=*"dropbtn"* style="font-size: *20px*;"><u>First Aid</u></button>

<div class=*"dropdown-content"* class=*"w3-button"* style="width: *220px*;">

<a href=*"hattack.html"*>Heart Attack</a>

<a href=*"sugar.html"*>Diabetes</a>

<a href=*"elec.html"*>Electric shock</a>

<a href=*"knife.html"*>Knife cutting</a>

<a href=*"loose.html"*>Loose Motion</a>

<a href=*"nose.html"*>Nose bleeding</a>

<a href=*"pois.html"*>Poisoning</a>

<a href=*"vomit.html"*>Vomiting</a>

<a href=*"dog.html"*>Dog bite</a>

</div>

</div>

</div>

</div>

<br><br><br><br>

<div class=*"login w3-card-3 w3-orange w3-padding"* style="height: *100%*;">

<h3 class=*"w3-indigo"* style="font-size: *20px*;">User login</h3>

<form action=*"Ulogin"* method=*"post"*>

<table class=*"w3-table w3-orange"*>

<tr><td><strong class=*"w3-text-white"*>Email</strong></td><td><input type=*"text"* name=*"email"* class=*"w3-input"*></td></tr>

<tr><td><strong class=*"w3-text-white"*>Password</strong></td><td><input type=*"password"* name=*"pass"* class=*"w3-input"*></td></tr>

<tr><td><a href=*reg.jsp* class=*"w3-text-teal w3-hover-text-pink"* >Click here<br> to register</a></td>

<td><input type=*submit* value=*LOGIN* class=*"w3-button w3-pink"* style="width: *100%*"></td></tr>

</table></form>

</div>

</center>

</body>

</html>

**package com.servlets;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import javax.servlet.http.HttpSession;**

**import com.controler.DBConnect;**

**/\*\***

**\* Servlet implementation class Ulogin**

**\*/**

**@WebServlet("/Ulogin")**

**public class Ulogin extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public Ulogin() {**

**super();**

**// TODO Auto-generated constructor stub**

**}**

**/\*\***

**\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)**

**\*/**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

**// TODO Auto-generated method stub**

**PrintWriter out=response.getWriter();**

**String email=request.getParameter("email");**

**String pass=request.getParameter("pass");**

**String id=request.getParameter("id");**

**String sql1 = "select \* from ureg where email='"+email+"' and pass='"+pass+"'";**

**boolean b = DBConnect.getData(sql1);**

**if(b == true){**

**HttpSession session = request.getSession();**

**session.setAttribute("email", email);**

**session.setAttribute("id", id);**

**response.sendRedirect("userhome.jsp");**

**}else{**

**out.println("<script type=\"text/javascript\">");**

**out.println("alert('failed to login');");**

**out.println("window.location='login.jsp'</script>");**

**}**

**}**

**}**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>registration page</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

<style type=*"text/css"*>

*.login*{

display: *inline-block*;

height: *400px*;

}

/\* Style The Dropdown Button \*/

*.dropbtn* {

background-color: *#4CAF50*;

color: *white*;

padding: *16px*;

font-size: *16px*;

border: *none*;

cursor: *pointer*;

}

*.dropdown* {

position: *relative*;

display: *inline-block*;

}

*.dropdown-content* {

display: *none*;

position: *absolute*;

background-color: *#f9f9f9*;

min-width: *160px*;

box-shadow: *0px 8px 16px 0px rgba(0,0,0,0.2)*;

z-index: *1*;

}

*.dropdown-content* **a** {

color: *black*;

padding: *12px 16px*;

text-decoration: *none*;

display: *block*;

}

*.dropdown-content* **a***:hover* {background-color: *#f1f1f1*}

*.dropdown:hover* *.dropdown-content* {

display: *block*;

}

*.dropdown:hover* *.dropbtn* {

background-color: *#3e8e41*;

}

</style>

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1><center>

<div class=*" w3-striped w3-green"* style="font-size: *20px*;">

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"index.jsp"*>Home</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"Dlogin.jsp"*>GroupManager Login</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"login.jsp"*>User Login</a>

</div>

<!--<div class="w3-button" style="width: 220px;">

<a href="CSplogin.jsp">Cloud Provider Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="services.jsp">Services</a>

</div> -->

<div class=*"dropdown"*>

<button class=*"dropbtn"* style="font-size: *20px*;"><u>Cloud Service Provider</u></button>

<div class=*"dropdown-content"* class=*"w3-button"* style="width: *220px*;">

<a href=*"CSplogin.jsp"*>Cloud Provider Login</a>

<a href=*"services.jsp"*>Services</a>

</div>

</div>

<div class=*"dropdown"*>

<button class=*"dropbtn"* style="font-size: *20px*;"><u>First Aid</u></button>

<div class=*"dropdown-content"* class=*"w3-button"* style="width: *220px*;">

<a href=*"hattack.html"*>Heart Attack</a>

<a href=*"sugar.html"*>Diabetes</a>

<a href=*"elec.html"*>Electric shock</a>

<a href=*"knife.html"*>Knife cutting</a>

<a href=*"loose.html"*>Loose Motion</a>

<a href=*"nose.html"*>Nose bleeding</a>

<a href=*"pois.html"*>Poisoning</a>

<a href=*"vomit.html"*>Vomiting</a>

<a href=*"dog.html"*>Dog bite</a>

</div>

</div>

</div>

</div>

<form action=*"Register"* method=*"post"*>

<h3 class=*"w3-center"*>REGISTER HERE</h3>

<table class=*"w3-table w3-bordered w3-orange"* style="width: *400px*;" >

<tr><td>Enter Name</td> <td><input type=*"text"* name=*name* ></td></tr>

<tr><td>Enter Email</td> <td><input type=*"text"* name=*email* ></td></tr>

<tr><td>Enter Mobile No.</td> <td><input type=*"text"* name=*mob* ></td></tr>

<tr><td>Enter Age</td> <td><input type=*"text"* name=*age* ></td></tr>

<tr><td>Select Gender</td> <td><input type=*"text"* name=*gen* ></td></tr>

<tr><td>Decease</td> <td><input type=*"text"* name=*des* ></td></tr>

<tr><td>Enter Password</td> <td><input type=*"password"* name=*pass* ></td></tr>

<tr><td>Confirm Password</td> <td><input type=*"password"* name=*cpass* ></td></tr>

<tr><td><input type=*"submit"* class=*"w3-green w3-button"* style="width: *100%*"></td>

<td><a href=*"login.jsp"* class=*"w3-text-black w3-hover-text-green"*>click here to login</a></td></tr>

</table>

</form>

</center>

</body>

</html>

**package com.servlets;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import java.sql.SQLException;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**import com.beans.UserBean;**

**import com.controler.DBConnect;**

**/\*\***

**\* Servlet implementation class Register**

**\*/**

**@WebServlet("/Register")**

**public class Register extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**/\*\***

**\* @see HttpServlet#HttpServlet()**

**\*/**

**public Register() {**

**super();**

**// TODO Auto-generated constructor stub**

**}**

**/\*\***

**\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)**

**\*/**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**}**

**/\*\***

**\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)**

**\*/**

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

**// TODO Auto-generated method stub**

**PrintWriter out=response.getWriter();**

**String name=request.getParameter("name");**

**String email=request.getParameter("email");**

**String mobile=request.getParameter("mob");**

**String age=request.getParameter("age");**

**String gender=request.getParameter("gen");**

**String des=request.getParameter("des");**

**String pass=request.getParameter("pass");**

**UserBean u=new UserBean();**

**u.setAge(age);**

**u.setUname(name);**

**u.setEmail(email);**

**u.setMobile(mobile);**

**u.setPass(pass);**

**u.setGender(gender);**

**u.setDes(des);**

**int i;**

**try {**

**i = DBConnect.regUser1(u);**

**if(i>0)**

**{**

**out.println("<script type=\"text/javascript\">");**

**out.println("alert('Registration successfull');");**

**out.println("window.location='login.jsp'</script>");**

**}else{**

**out.println("<script type=\"text/javascript\">");**

**out.println("alert('invalid username or password');");**

**out.println("window.location='reg.jsp'</script>");**

**}**

**} catch (SQLException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}}**

**package** com.servlets;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.sql.SQLException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** com.beans.UserBean;

**import** com.controler.DBConnect;

/\*\*

\* Servlet implementation class Req

\*/

@WebServlet("/Req")

**public** **class** Req **extends** HttpServlet {

**private** **static** **final** **long** serialVersionUID = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** Req() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** HttpServlet#doGet(HttpServletRequest request, HttpServletResponse

\* response)

\*/

**protected** **void** doGet(HttpServletRequest request,

HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

PrintWriter out = response.getWriter();

String id = request.getParameter("id");

String name = request.getParameter("name");

String email = request.getParameter("email");

String mobile = request.getParameter("mob");

String age = request.getParameter("age");

String gender = request.getParameter("gen");

String des = request.getParameter("des");

UserBean u = **new** UserBean();

u.setId(id);

u.setUname(name);

u.setEmail(email);

u.setMobile(mobile);

u.setAge(age);

u.setGender(gender);

u.setDes(des);

String sql1 = "select \* from user where email='" + email + "'";

**try** {

**boolean** b = DBConnect.getData(sql1);

**if** (b == **true**) {

out.println("<script type=\"text/javascript\">");

out.println("alert('Request already Sended');");

out.println("window.location='Ureq.jsp'</script>");

} **else** {

**int** i;

i = DBConnect.regUser2(u);

**if** (i > 0) {

out.println("<script type=\"text/javascript\">");

out.println("alert('Request Sended successfully');");

out.println("window.location='Ureq.jsp'</script>");

} **else** {

out.println("<script type=\"text/javascript\">");

out.println("alert('invalid details');");

out.println("window.location='error.jsp'</script>");

}

}} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

/\*\*

\* **@see** HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

\* response)

\*/

**protected** **void** doPost(HttpServletRequest request,

HttpServletResponse response) **throws** ServletException, IOException {

// **TODO** Auto-generated method stub

}

}

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>login page</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

<style type=*"text/css"*>

*.login*{

display: *inline-block*;

height: *400px*;

}

/\* Style The Dropdown Button \*/

*.dropbtn* {

background-color: *#4CAF50*;

color: *white*;

padding: *16px*;

font-size: *16px*;

border: *none*;

cursor: *pointer*;

}

*.dropdown* {

position: *relative*;

display: *inline-block*;

}

*.dropdown-content* {

display: *none*;

position: *absolute*;

background-color: *#f9f9f9*;

min-width: *160px*;

box-shadow: *0px 8px 16px 0px rgba(0,0,0,0.2)*;

z-index: *1*;

}

*.dropdown-content* **a** {

color: *black*;

padding: *12px 16px*;

text-decoration: *none*;

display: *block*;

}

*.dropdown-content* **a***:hover* {background-color: *#f1f1f1*}

*.dropdown:hover* *.dropdown-content* {

display: *block*;

}

*.dropdown:hover* *.dropbtn* {

background-color: *#3e8e41*;

}

</style>

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1><center>

<div class=*" w3-striped w3-green"* style="font-size: *20px*;">

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"index.jsp"*>Home</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"Dlogin.jsp"*>GroupManager Login</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"login.jsp"*>User Login</a>

</div>

<!--<div class="w3-button" style="width: 220px;">

<a href="CSplogin.jsp">Cloud Provider Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="services.jsp">Services</a>

</div> -->

<div class=*"dropdown"*>

<button class=*"dropbtn"* style="font-size: *20px*;"><u>Cloud Service Provider</u></button>

<div class=*"dropdown-content"* class=*"w3-button"* style="width: *220px*;">

<a href=*"CSplogin.jsp"*>Cloud Provider Login</a>

<a href=*"services.jsp"*>Services</a>

</div>

</div>

<div class=*"dropdown"*>

<button class=*"dropbtn"* style="font-size: *20px*;"><u>First Aid</u></button>

<div class=*"dropdown-content"* class=*"w3-button"* style="width: *220px*;">

<a href=*"hattack.html"*>Heart Attack</a>

<a href=*"sugar.html"*>Diabetes</a>

<a href=*"elec.html"*>Electric shock</a>

<a href=*"knife.html"*>Knife cutting</a>

<a href=*"loose.html"*>Loose Motion</a>

<a href=*"nose.html"*>Nose bleeding</a>

<a href=*"pois.html"*>Poisoning</a>

<a href=*"vomit.html"*>Vomiting</a>

<a href=*"dog.html"*>Dog bite</a>

</div>

</div>

</div>

</div>

<br><br><br><br>

<!--.................................................................................................. -->

<div class=*"login w3-card-3 w3-orange w3-padding"* style="height: *100%*;">

<h3 class=*"w3-indigo"*>CSP Login</h3>

<form action=*"CSPlogin"* method=*"post"*>

<table class=*"w3-table w3-orange"*>

<tr><td><strong class=*"w3-text-indigo"*>Email</strong></td><td><input type=*"text"* name=*email* class=*"w3-input"*></td></tr>

<tr><td><strong class=*"w3-text-indigo"*>Password</strong></td><td><input type=*"password"* name=*pass* class=*"w3-input"*></td></tr>

<tr><td></td><td><input type=*submit* value=*LOGIN* class=*"w3-button w3-blue"* style="width: *100%*"></td></tr>

</table>

</form>

</div></center>

</body>

</html>

<%@page import=*"java.sql.SQLException"*%>

<%@page import=*"java.sql.ResultSet"*%>

<%@page import=*"com.controler.DBConnect"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1>

<center>

<div class=*" w3-striped w3-orange"*>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"cspHome.jsp"*>Home</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"logout.jsp"*>Logout</a>

</div></center>

<!--<div class="w3-button" style="width: 220px;">

<a href="#">GroupManager Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="#">User Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="#">Cloud Provider Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="services.jsp">Services</a>

</div> -->

</div>

<br><br>

<center>

<table class=*"w3-table w3-border w3-orange"* style="width:*100%*;height: *50%*;">

<tr class=*"w3-pink"*>

<th >PATIENT ID</th><th>PATIENT KEY</th><th>STATUS</th>

</tr>

<%

//String id=request.getParameter("id");

//String key1=request.getParameter("key1");

**try**{

ResultSet r=DBConnect.getUser3();

**while**(r.next())

{

%>

<tr><td><%=r.getString(1) %></td><td><%=r.getString(2) %></td>

<td><a href=*"CkeckKey?id=*<%=r.getString(1) %>*&&key1=*<%=r.getString(2) %>*"*>Request</a></td>

</tr>

<%

}

}**catch**(SQLException e){

e.printStackTrace();

}

%>

<tr><td></td></tr></table></center>

</body>

</html>

<%@page import=*"java.sql.SQLException"*%>

<%@page import=*"java.sql.ResultSet"*%>

<%@page import=*"com.controler.DBConnect"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1>

<center>

<div class=*" w3-striped w3-orange"*>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"cspHome.jsp"*>Home</a>

</div>

<!--<div class="w3-button" style="width: 220px;">

<a href="#">GroupManager Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="#">User Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="#">Cloud Provider Login</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href="services.jsp">Services</a>

</div> -->

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"logout.jsp"*>Logout</a>

</div>

</div></center>

<br><br>

<table class=*"w3-table w3-border w3-blue"*>

<tr class=*"w3-orange"* style="width: *500px*;;height: *50%*;">

<th >PATIENT ID</th><th>PATIENT KEY</th><th>ASSIGN KEY</th>

</tr>

<%

//String id=request.getParameter("id");

//String key1=request.getParameter("key1");

**try**{

String cc=request.getParameter("id");

System.out.println(cc);

session=request.getSession(**false**);

ResultSet r=DBConnect.getUser4();

**while**(r.next())

{

%>

<tr><td><%=r.getString(1) %></td><td><%=r.getString(2) %></td>

<td><a href=*"CkeckKey1?id=*<%=r.getString(1) %>*&&key1=*<%=r.getString(2) %>*"*>Assign Key</a></td>

</tr>

<%

}

}**catch**(SQLException e){

e.printStackTrace();

}

%>

<tr><td></td></tr></table>

</body>

</html>

<%@page import=*"java.sql.Statement"*%>

<%@page import=*"java.sql.Connection"*%>

<%@page import=*"java.sql.ResultSet"*%>

<%@page import=*"com.controler.DBConnect"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1>

<center>

<table class=*"w3-table w3-bordered w3-teal"*>

<tr class=*"w3-card-4 w3-pink"*>

<th>DOCTOR ID</th>

<th>NAME</th>

<th>EMAIL</th>

<th>MOBILE NO.</th>

<th>SPECIALIST</th>

<th>EXPERIENCE</th>

<th>GENDER</th><th></th>

</tr>

<%

Connection con=DBConnect.connect();

Statement s=con.createStatement();

session=request.getSession(**false**);

String id=(String)session.getAttribute("id");

String hs=request.getParameter("hs");

System.out.println(id);

ResultSet r=s.executeQuery("select \* from doctor where spe='"+hs+"'");

//ResultSet r =DBConnect.getUser5(id);

**try**{

**if**(r.next()){

%>

<form action=*"Urequest"* method=*"post"*>

<tr class=*"w3-hover-white w3-card-3"* style="width: *500px*;;">

<tr>

<td><%= r.getString(1) %></td>

<td><%= r.getString(2) %></td>

<td><%= r.getString(3) %></td>

<td><%= r.getString(4) %></td>

<td><%= r.getString(5) %></td>

<td><%= r.getString(6) %></td>

<td><%= r.getString(8) %></td>

<td><input type=*"submit"* value=*"REQUEST"* class=*"w3-pink w3-round w3-button"*></td>

</tr>

<%

}}**catch**(Exception e)

{

out.println("error msg: "+e);

System.out.println(e);

}

%>

</body>

</html>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1>

<div class=*" w3-striped w3-orange"* style="font-size: *20px*;">

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"gmhome.jsp"* class=*"w3-text-white w3-hover-text-blue"*>GM Home</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"urequests.jsp"* class=*"w3-text-white w3-hover-text-blue"*>Patient requests</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"adddoctors.jsp"* class=*"w3-text-white w3-hover-text-blue"* >Add Doctor</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*doctorAdmin.jsp* class=*"w3-text-white w3-hover-text-blue"*>Doctors List</a>

</div>

<!--<div class="w3-button" style="width: 220px;">

<a href=fpsAtD.jsp class="w3-text-white w3-hover-text-blue">Patients Daily Activities</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href=umsg.jsp class="w3-text-white w3-hover-text-blue" >Messages</a>

</div>-->

<div class=*"w3-button"* style="width: *220px*;">

<a href=*Dlogout* class=*"w3-text-white w3-hover-text-blue"* >logout</a>

</div>

</div>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"StyleSheet"* href=*"w3.css"* >

</head>

<body style="background-image: *url('logos/john.jpg')*;">

<h1 style="font-size: *40px*;" class=*"w3-text-white"*>

Anonymous Authentication Scheme for Smart

Cloud Based Healthcare Applications</h1>

<div class=*" w3-striped w3-orange"* style="font-size: *20px*;">

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"gmhome.jsp"* class=*"w3-text-white w3-hover-text-blue"*>GM Home</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"urequests.jsp"* class=*"w3-text-white w3-hover-text-blue"*>Patient requests</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*"adddoctors.jsp"* class=*"w3-text-white w3-hover-text-blue"* >Add Doctor</a>

</div>

<div class=*"w3-button"* style="width: *220px*;">

<a href=*doctorAdmin.jsp* class=*"w3-text-white w3-hover-text-blue"*>Doctors List</a>

</div>

<!--<div class="w3-button" style="width: 220px;">

<a href=fpsAtD.jsp class="w3-text-white w3-hover-text-blue">Patients Daily Activities</a>

</div>

<div class="w3-button" style="width: 220px;">

<a href=umsg.jsp class="w3-text-white w3-hover-text-blue" >Messages</a>

</div>-->

<div class=*"w3-button"* style="width: *220px*;">

<a href=*Dlogout* class=*"w3-text-white w3-hover-text-blue"* >logout</a>

</div>

</div>

package com.servlets;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.catalina.Session;

import com.controler.DBConnect;

/\*\*

\* Servlet implementation class PatientKey

\*/

@WebServlet("/PatientKey")

public class PatientKey extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public PatientKey() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

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

// TODO Auto-generated method stub

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

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

// TODO Auto-generated method stub

PrintWriter out=response.getWriter();

String cc=request.getParameter("id");

String dd=request.getParameter("key2");

System.out.println(cc+ " "+dd);

HttpSession session = request.getSession();

session.setAttribute("key2", dd);

session.setAttribute("id", cc);

session =request.getSession(false);

String sql="select id,key2 from cloud where id='"+session.getAttribute("id")+"' and key2='"+session.getAttribute("key2")+"'";

boolean b = DBConnect.getData(sql);

System.out.println(sql);

//boolean b = DBConnect.getData(sql);

try{

if(b==true){

response.sendRedirect("services1.jsp");

}else{

out.println("<script type=\"text/javascript\">");

out.println("alert('failed to Access Sevices');");

out.println("window.location='services.jsp'</script>");

}

}catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

System.out.println(e);

}

}

}

**CHAPTER 7**

**SNAPSHOTS**

**General:**

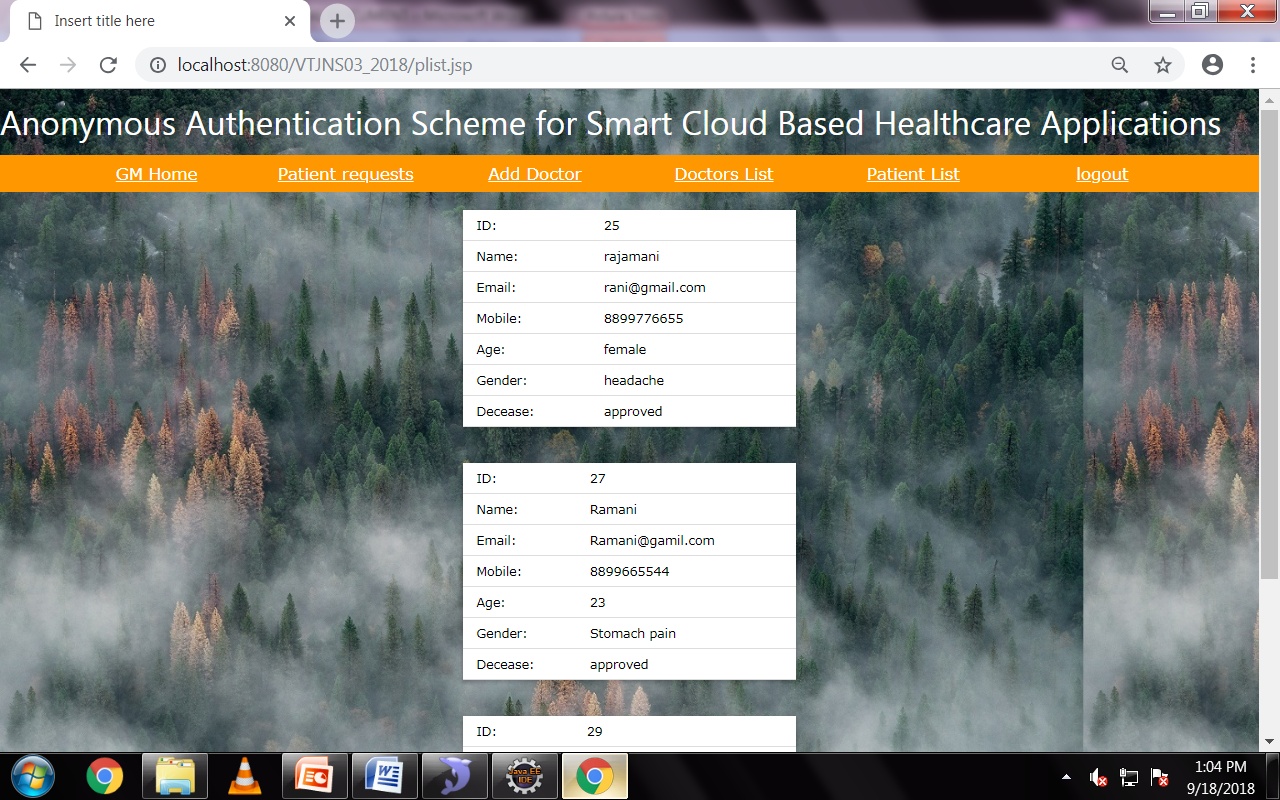
This project is implements like web application using COREJAVA and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

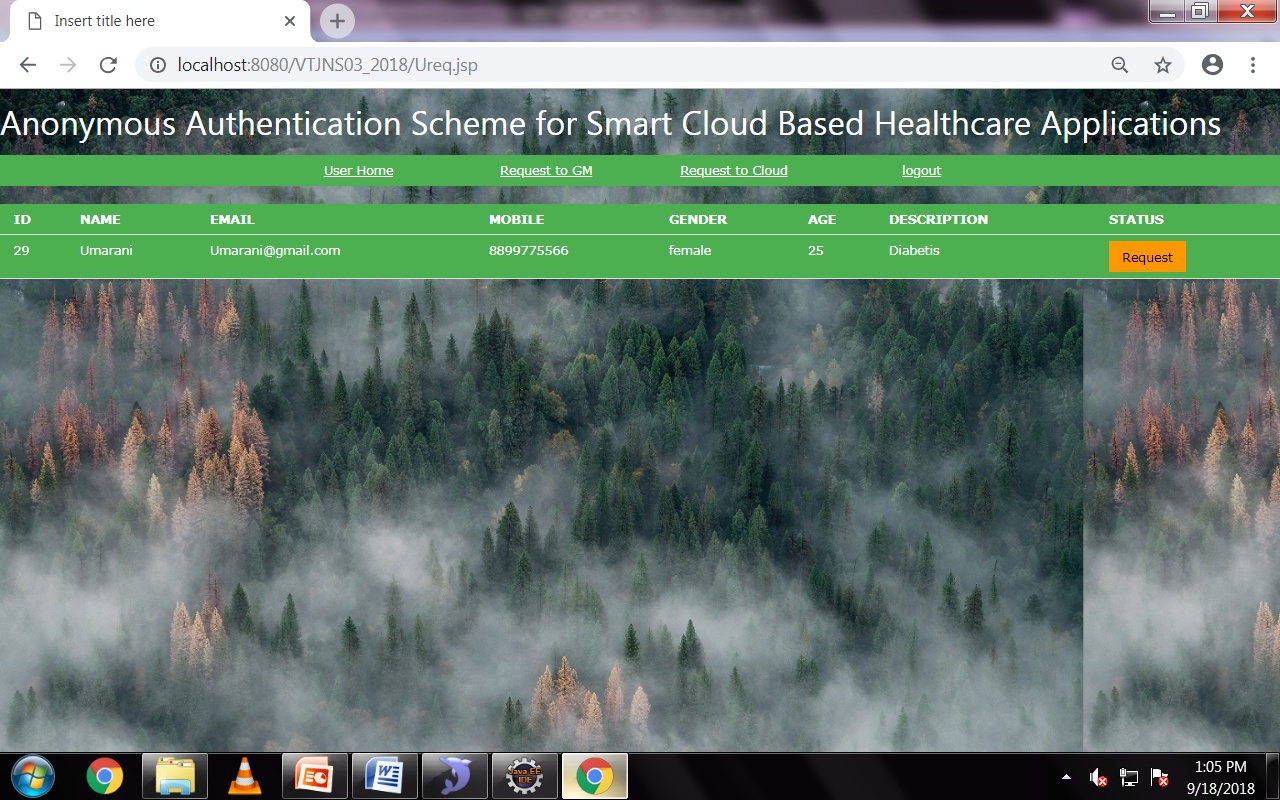
**SNAPSHOTS**

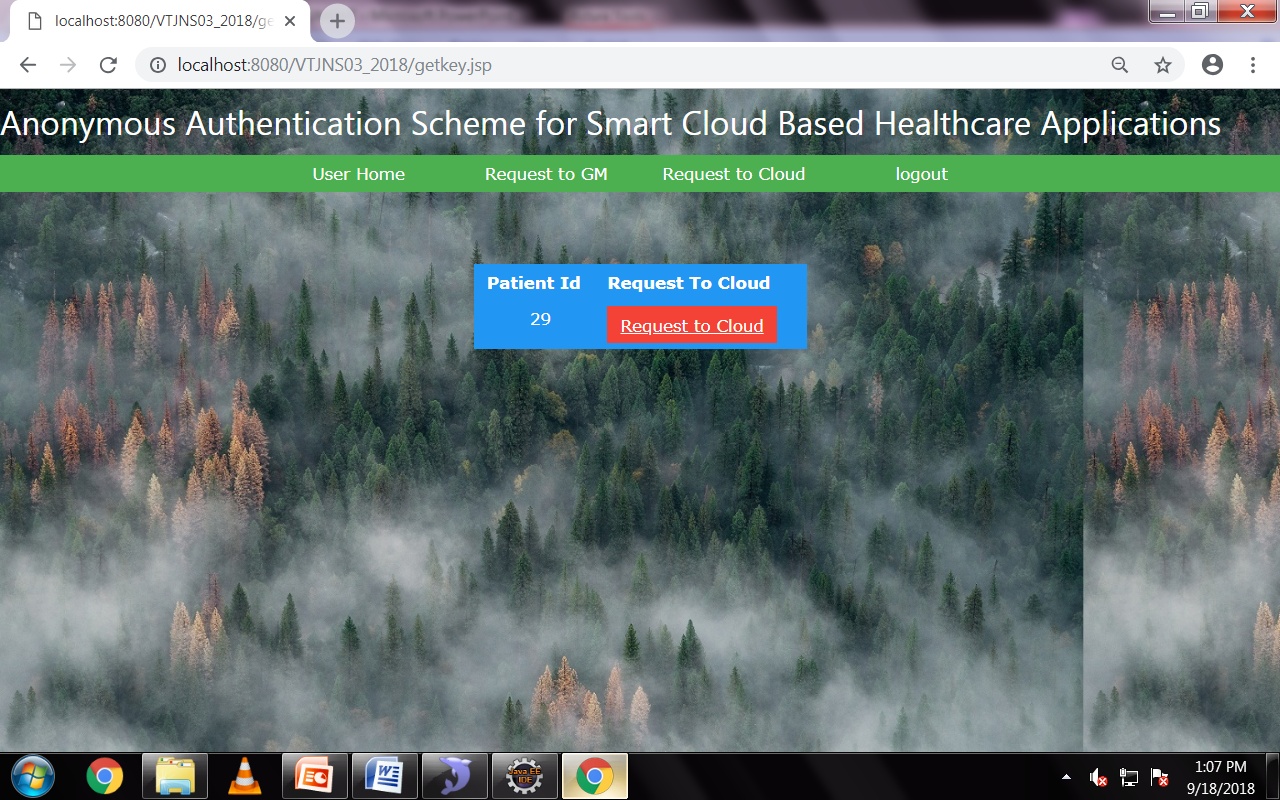
****

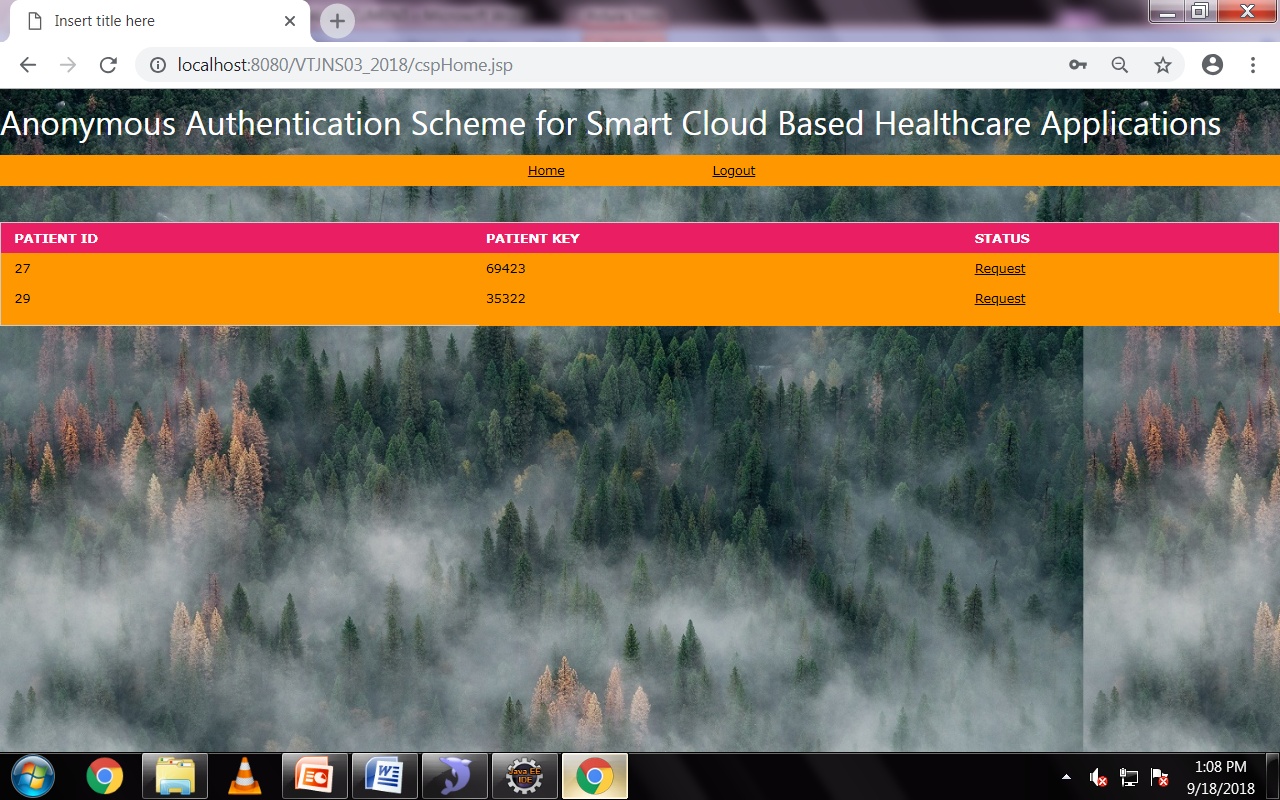
****

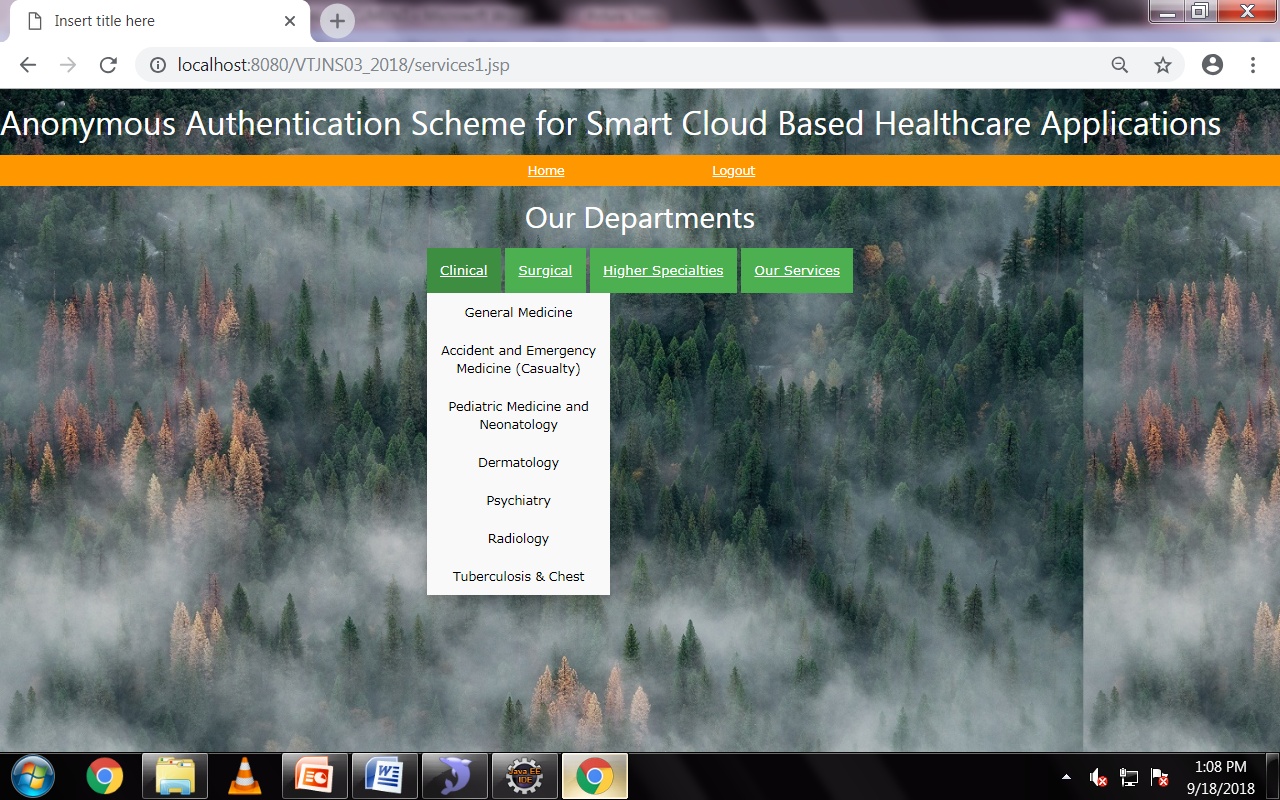
****

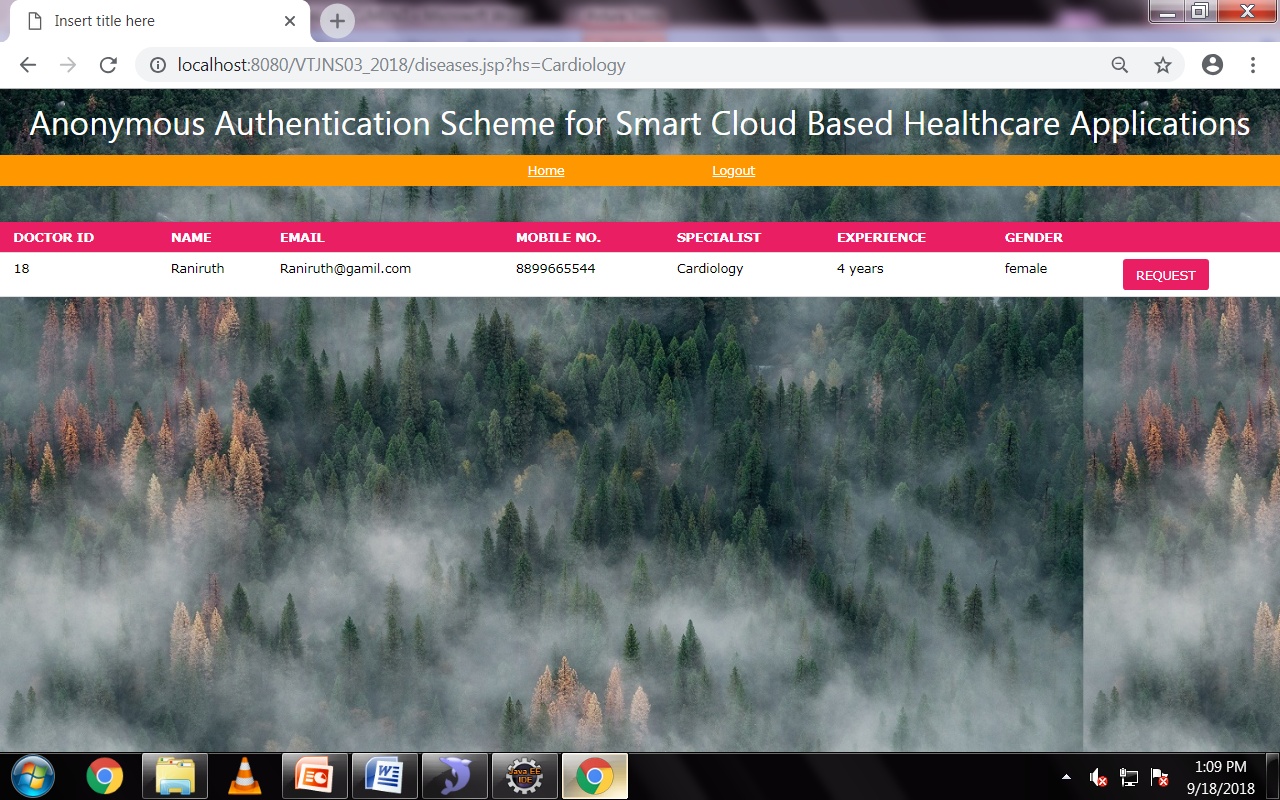
****

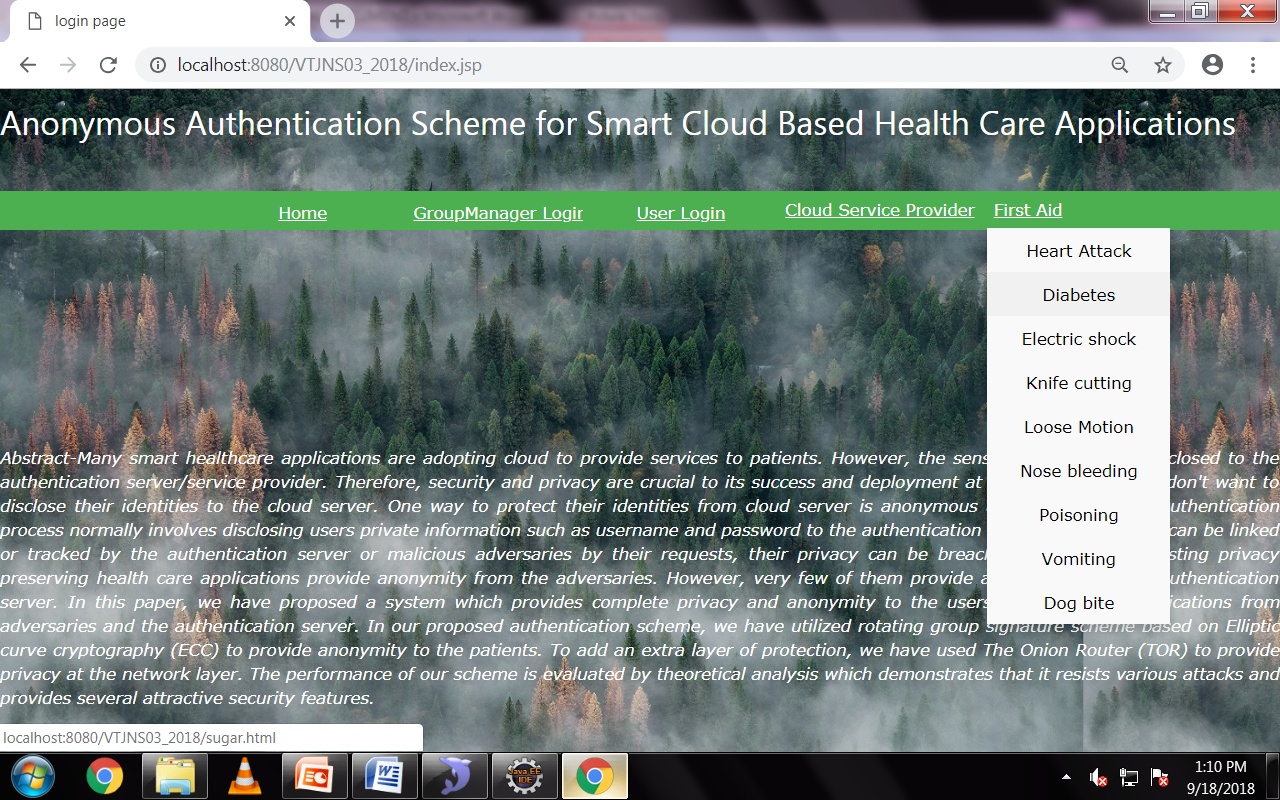
****

****

****

****

****

****

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

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.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3Types of Tests**

**8.3.1 Unit testing**

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

**8.3.2 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.

**8.3.3 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.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

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

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

**8.3.6 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.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**CONCLUSION & REFERENCE**

Protecting the privacy of patients is crucial to the success of smart cloud based healthcare applications. In this paper, we have presented the anonymous authentication scheme for smart cloud based healthcare applications. The proposed scheme preserves the privacy of patients when they access the services hosted on the cloud. The scheme utilizes rotating group signature scheme based on ECC. Due to smaller key sizes used in ECC, the security of the system can be easily scaled up by increasing the key size without affecting the computational complexity. The scheme adds an extra layer of protection against traffic analysis attacks by an eavesdropper by providing anonymity at the network layer by employing TOR. The scheme protects patients’ sensitive data from an eavesdropper and untrusted cloud servers. One salient feature of our scheme is that the medical application or service providers cannot reveal the identity of the patient hence protecting the privacy. In this paper, we have designed a practical system which is

secure and efficient. The proposed authentication scheme ensures that the patients can consume services without revealing their identity at the time of consumption or retrospectively.