Implementation of a Prototype for Secure Patients Document Retrieval

GRADUATE PROJECT REPORT

Submitted to the Faculty of
The School of Engineering & Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

By

Rahul Potuganti
Spring 2015

Committee Members

Dr. Mario Garcia
Committee Chairperson

Dr. David Thomas
Committee Member
ABSTRACT

Web applications have become very popular nowadays because of its wide range of applications. These applications provide services in the cloud and other service oriented architectures. During the process, a lot of personal data or private data is being exposed which can be a threat. Data privacy for the users has become a very stimulating task. In order to preserve this privacy, the exchange of personal data over the internet should be reduced. To achieve this, the data should be stored securely in the database using the concept of web security with proper authentication.

In this paper, a model has been simulated using a proxy server that acts as an intermediary to store the data securely in the database. The model consists of three web applications, namely the main server, proxy server and the database server. The information is transferred securely to the database via the proxy server. The input to the application will be a registration page where the registration details are taken. The details are encrypted by the main server and are given to the proxy server in the form of an XML string. Both the doctor and patient services are related to each other where a single doctor can have multiple patients. The retrieval mechanism here is based on the user authentication where unauthorized patients are not shown in the doctor. The next module in the project is that the doctor uploads a patient document to the patient and the patient can retrieve the file by downloading it or viewing it with a key that is provided by the doctor. The document is encrypted and decrypted using the key by the patient to view the data.
TABLE OF CONTENTS

Abstract ................................................................................................................................. ii

Table of Contents .................................................................................................................. iii

List of Figures ....................................................................................................................... vi

List of Tables ....................................................................................................................... viii

1. Background and Rationale ............................................................................................. 9
   1.1 Web Services .............................................................................................................. 9
   1.2 Data As a Service ...................................................................................................... 10
   1.3 Restful and Service Oriented Architecture ............................................................. 11
   1.4 Web service composition and Proxy Server ............................................................. 12
   1.5 Health Care Services ............................................................................................... 14
   1.6 Web Security in Health Care Services .................................................................... 15
   1.7 Algorithms Used for Project .................................................................................... 15
      1.7.1 AES/CBC/PKCS5 Padding Algorithm ............................................................... 15
   1.8 Existing System ........................................................................................................ 16

2. Narrative .......................................................................................................................... 19
   2.1 Problem Statement ................................................................................................... 19
   2.2 Motivation ................................................................................................................ 19
   2.3 Project Objective ..................................................................................................... 20
   2.4 Project Scope .......................................................................................................... 20

3. Proposed System ............................................................................................................. 21
   3.1 Methodology and Architecture ................................................................................. 21
3.2 UML Diagrams ........................................................................................................ 24
   3.2.1 Use case diagram .............................................................................................. 24
   3.2.2 Sequence Diagram for Admin Login Page ...................................................... 26
   3.2.3 Class Diagram ................................................................................................. 27
3.3 Environment for the Project .................................................................................. 29
   3.3.1 Netbeans .......................................................................................................... 29
   3.3.2 Oracle Database 12C ....................................................................................... 29
   3.3.3 Java Swing ....................................................................................................... 30
3.4 System Configuration .............................................................................................. 30
3.5 Flowchart Diagram ................................................................................................. 31
   3.5.1 Database Server Flowchart ............................................................................... 31
   3.5.2 Proxy server Flowchart .................................................................................... 32
   3.5.3 Flowchart for Document Uploading ............................................................... 33
   3.5.4 Flowchart for Document Retrieval ................................................................. 34
3.6 Database Design ...................................................................................................... 35
4. Project Implementation ............................................................................................... 39
5. Testing ....................................................................................................................... 46
   5.1 Doctor Test Case ................................................................................................... 46
   5.2 Patient Test Case within Doctor ......................................................................... 47
   5.3 Doctor Login Test Case ....................................................................................... 48
   5.4 Test case for Document Uploading by Doctor ................................................... 50
   5.5 Test Case for Document Retrieval by Patient .................................................... 52
6. Conclusion and Future Work .................................................................................... 54
Bibliography and References .................................................................55
Appendix ..............................................................................................58
# LIST OF FIGURES

Figure 1.1 DaaS General Architecture
Figure 1.2 Restful Web Service
Figure 1.3 Web Service Composition
Figure 1.4 Example of Proxy Server
Figure 3.1 System Architecture
Figure 3.2 Use Case for Database and Proxy
Figure 3.3 Use case for Main Server
Figure 3.4 Sequence Diagram for Database Server
Figure 3.5 Class Diagram
Figure 3.6 Database Server Flowchart
Figure 3.7 Proxy Server Flowchart
Figure 3.8 Flowchart for Document uploading
Figure 3.9 Flowchart for Document Retrieval
Figure 3.10 Flowchart for Document Uploading
Figure 3.11 Flowchart for Document Retrieval
Figure 4.1 Database Server Application Page
Figure 4.2 Proxy Server Home Page
Figure 4.3 Admin Login Page
Figure 4.4 Doctor Login Page
Figure 4.5 Doctor 1 Patients List
Figure 4.6 Patient 1 Details
Figure 4.7 Document Uploading
Figure 4.8 Document Retrieval
Figure 5.1 Doctor Registration Test Case
Figure 5.2 Patient Registration Test Case .............................................................................. 47
Figure 5.3 Doctor 1 Patients List .............................................................................................. 48
Figure 5.4 Patient 2 Details for Doctor 1 .................................................................................. 49
Figure 5.5 Doctor Details for Patient 2 ..................................................................................... 50
Figure 5.6 Document Uploading by Doctor ............................................................................ 51
Figure 5.7 Status of Document Upload ..................................................................................... 51
Figure 5.8 Document Retrieval by Patient ............................................................................... 52
Figure 5.9 File Download Status .............................................................................................. 53
LIST OF TABLES

Table 3.1 Software Requirements .................................................................31
Table 3.2 Sample Doctors Database Table ...................................................36
Table 3.3 Sample Patients Database Table ...................................................37
Table 3.4 Sample Doctorskey Database Table ............................................38
Table 3.5 Sample Patientkeys Database Table .............................................38
Table 3.5 Sample Filetable for Document Retrieval ....................................38
1. BACKGROUND AND RATIONALE

Web services have become very important nowadays because of its functionalities. It is a combination of various web based applications using architectures like the Service Oriented or Restful [12]. Each web service has a database as a backend and a web page as a front end as its application user interface.

Service composition means the collaboration of various web services that are used to program a particular business job. While achieving this goal, there might be many privacy issues that may arise and also pose a significant threat. With this problem, the data while communicating with different web services may be lost [12].

1.1 Web Services

With the improvement of technology, the use of web services over the internet through a service provider has become very easy and convenient. Web service as a single server must be able to give reliable and secure validations for the policies. Web services security is an important factor in the distributed platforms like e-commerce, health care, etc [10]. Based on these factors, the data as a service also became a very popular one that is used for the aggregation of different services to give a possible outcome. The client-server construction modeling has bad marks over the web administration structure modeling due to the absence of reusability [9].

The web service can be a piece of code that is used for accessing the web using XML message passing system. The code here is used for encoding all the communications that occur in the web service. When a customer invokes a web service with a message containing the code, then the related response is generated.
1.2 Data as a Service

Data as a service is defined as providing the data to the users based on their request. The data can be of any form and dimensions irrespective of the user and the client. It can be public data or private data like the geospatial data, weather data, reports, etc. Service oriented architecture is the main reason that made the data as a service very popular. In the business scenario, it can act as a mediator that tries to process the data and give it the end user. Data as a service composition is considered as a consolidation of various smaller services for a business peer to peer network [7]. Figure 1.1 explains the general architecture for the data as a service.

Figure 1.1 DaaS General Architecture
1.3 Restful and Service Oriented Architecture

Restful is an architecture that is used to create web services. It is a substitute to other architectures in the web like the service oriented architecture or web service description language. The Rest architecture is used more often because of its functionalities and advantages in the real time projects. There are some characteristics of Restful architecture that mainly focus on the resources that are used for the web service. These resources here are stored in the XML format. It is also a client-server architecture where both of them are separated using an interface [21]. It is stateless, layered and cacheable architecture with uniform interface and identification of resources.

Rest architecture is transferred and communicated over the HTTP and uses some methods for its process. The methods that are used in the architecture are GET and POST that has its functionality. It supports media types like the hypertext markup language, Javascript object notation (JSON) with some HTTP URI paths and response codes. The HTTP requests are passed to the server from the client where the location of the resource is identified. Once the resource type is located, then it is passed with the help of Uniform Resource Locator that contains the name, protocol and the query [22]. Figure 1.2 shows the architecture of the REST services where all the resources in the web are accessed based on methods like GET, PUT, POST and DELETE.

The transfer of data from one service to another service using a port address and a URL is a part of the Rest architecture. The data is transfers to any ip address that s specified.
Software oriented architecture is defined as a design in which one application provides services to another one through a network. It is described as the collaboration of data, software and platform services. In general the language that is used for the data is XML. Web services description language is used for services while the simple object access protocol is used for the communication between the services.

1.4 Web Service Composition and Proxy Server

In web service terminology, composition refers to the creation of a new service based on the collaboration of various small services. For example, if there is a problem and if it cannot be fulfilled by an individual service, then a set of smaller services can be combined to become larger service to automate a particular problem. For instance, the Figure 1.3 shows the aggregation of various smaller services like the car rental service, hotel service and the airline service to make a bigger service that achieves a particular
business task. The service composition here can be between different smaller services related to larger service. The user consolidates formalization with run-time screen and utilizes their separate qualities [8].

A proxy server in the web services is defined as a mediator between the user and the client. In case of data retrieval, when the data is sent from the service to the user, the proxy server takes the data for further processing and keeps a temporary record of the data and then sends it to the user as per the request [18]. Figure 1.4 shows the proxy in the web applications that is used to store and process the data.

**Figure 1.3 Web Service Composition [29]**

proxy server takes the data for further processing and keeps a temporary record of the data and then sends it to the user as per the request [18]. Figure 1.4 shows the proxy in the web applications that is used to store and process the data.
1.5 Health Care Services

Health care is referred to the treatment of diseases for humans. Healthcare services are provided in various fields like the medicine, dentistry, pharmacy or other health related professions.

Healthcare servers contain the database that is related to a particular domain and data. The database is also used for mass storages and transfer of health-related data.

Server virtualization in health care is a new technology that is also used to store health care related data and secure them. This technology uses the concept of one server that makes the software and hardware requirements independent [13].

Some of the health care databases that are in the real time scenario are public health, hospitals, clinical research etc.
1.6 Web Security in Healthcare services

There are some recent mobile based applications that are used to access the patient’s records with the help of the REST API. With this, the data is securely stored and accessed with some protocols. Some frameworks are also being implemented for the health care records, accessing in data mashup [11].

Some of the recent improvements in the security related to health care are Health Level 7 that developed fast Health Interoperable Resources for clinical organizations. It is developed with the REST architecture where the messages are in the form of documents [14].

The next improvement that is implemented is the Restful health exchange that is used for data transfer in the health sector securely. The project here is open source that is simple and fast.

1.7 Algorithms Used for the Architecture

1.7.1 AES/CBC/PKCS5Padding Algorithm

The AES/CBC/PKCS5Padding algorithm is an inbuilt library available in Java that is used for the purpose of encryption and decryption. It is a part of Java Cryptography Extension (JCE). It has other functionalities like the key generation, message authentication, digital signatures etc. The algorithm has many functions, objects and methods to perform a set of operations. The algorithm is used to encrypt the input given to a cipher text that is encrypted. The data here is taken in the form of blocks with the help of padding.
1.8 Existing Systems

P3P is a project that gives an opportunity for the user to take it as a reference and model their architecture based on the privacy issues. The existing model is an example of this project. In the existing system, the data as a service is implemented based on the privacy policies. In general the data suppliers give an idea about how the service must be utilized while the others take into account regarding the personalized data in that service.

There are two important factors that increase the privacy issues in the data as a service. First is the service related to the database collect an enormous amount of personal information. The second factor is that this process, share the information with other components. Because of these factors the privacy of data is not preserved.

In the existing system, there are also some challenges involved. Firstly is the privacy specification where the privacy should be specified properly in the service and is understood correctly by the user according to the problem. Secondly, is the privacy issues within the service or a web composition that communicate among themselves during the process [16]. Lastly is how to deal with some of the privacy policies that are not appropriate or match with the existing systems. Some of the previous work that are related to this project are explained briefly.

Salah-Eddine Tbahriti [1] discussed an approach that has a combination of various Data related service collaborations. There are many privacy issues that are related to the web services that are responsible for various frauds and crimes. The author here tries to define some set of privacy policies that meet the requirement of the data privacy. The privacy can be raised by the input data or the methodology. The method also uses
some protocols based on the cost to verify the matching of the rules and the data for privacy issues.

Benjamin C. M. Fung [2] used the concept of privacy in a mashup that is a technology used to add and give excellent services to the customers. When a user tries to request some data based on a query, then this service that is used to collect the data from all the services and gives it to the end user. There might be many challenges involved in the process like the privacy of data or data loss. In order to overcome these issues, the author has proposed a method that analyzes the privacy issue in any real time application based on some architecture that is used to help in protecting the data privacy.

Brahim Medjahed [3] proposed an architecture developed for the integrity of data privacy for the web services. Privacy can be a major problem in the case of data that is given as input to the model or the outcome of the model. The architecture defines a procedure that is used to match various privacy compositions for the services. The author also uses a negotiation method for the privacy of data if there are some problems in the matching of privacy policies.

Michael Mrissa [4] proposed a similar model that mainly deals with privacy restrictions in the data as a service is implemented. The old web services that were running the internet services had a lot of privacy issues related to different policies and regulations. When the user requests a query for the Data as a service, then how the privacy content can be preserved is being proposed in this paper. Based on the study of data the author has also proposed a new model that will combine both the policies and various web service formats.
Mahmoud Barhamgi [5] has proposed a hybrid approach that deals with ontologies and questions. Web services can be invoked using a query from the user's side. Due to this the retrieval of data becomes a difficult task if the questions are very complex. To overcome this problem the author had defined resource description framework as views for answering these questions in the web service composition. Several algorithms were also proposed to these questions for the web services.

Junpei Kawamoto and Masatoshi Yoshikawa [6] have proposed an attack model that is used to answer various questions in the social networking domain. They are described as logs files in the data as service architecture that are queried to obtain public information. This method also defends the system from various attacks of the hackers. The model also deals with two important factors that are the cost analysis and the safety measures.
2. NARRATIVE

2.1 Problem Statement

Data privacy has been a very major problem in this present world. Similarly, web services are used in all of the industries. Data privacy in the web services is also a major issue nowadays. So if the user wants to store the data securely in the data so that it is not hacked by anyone then the architecture must be secured. There are some approaches to keep the data secure in the web, but no specific one to check based on privacy policies and proxy server. So a user using this system should be able to conserve privacy of the data while information retrieval for the web services.

2.2 Motivation

Privacy of data is more important and is more important when the data is related to any personal information like the address, social security number, credit card details, etc. If this information is misused than the privacy of personal data is not preserved. For example, if the credit card details for a person are misused then there could be a lot of financial issues. Based on these problems it makes the user more alert and motivated enough to develop a system or model so as to preserve the privacy of data. This methodology here makes the system more secure keeping in mind about the private data. With the distributed platform also the security issues have increased [15].
2.3 Project Objective

From the previous method and results, there must be some architecture that is intended to preserve the privacy of data at the data level. This design is used to protect the privacy of data at the service level. The objective here is to store the data securely in the data as a service database based on a proxy server architecture using restful services. When the user gives the details, it gets stored on the proxy server in XML format in the proxy [17]. In the proxy, the data is processed based on service integration for web service composition and data registry to store the data in the database securely. The above goals can be achieved by this proposed system that is intended to store the data securely in the web database also showing the time required for secure transfer of data.

2.4 Project Scope

The project has nice applications and uses. The goal here is to maintain the privacy level for both the public and private data. So this architecture has some applications in the field of e-commerce, medical services, travel agency services, etc. In these applications, there is also collaboration with various services to achieve a particular business task. The functionality can also be extended based on the requirement of the user.
3. PROPOSED SYSTEM

3.1 Methodology and Architecture

This project describes a simulation model, enhancing data privacy while it is stored and transferred from the user interface to the local database. The privacy is maintained not only at the data level, but also in the service level. The primary process in the architecture is in the proxy server where the data is processed and encrypted to store it in the data as a service database. It is not only taking care of a single service, but also multiple services to integrate it with data from a service. The privacy of data is preserved at both the web servers.

The user here uses different web services that are built using the Restful service architecture. The services are related to the health care with patient and doctor information. All these services are integrated to store the data in the database. The doctor and the patient have a many to many relationship where authentication is done based on keys.

In this architecture, the main web server has a login registration page where the login is being admin and doctor. The admin login is used for adding a new doctor or patient to the database. The patient page has an option to select a doctor so that the data is reflected back in the database. Once the user fills in the details, then the data is encrypted and is sent to the middle tier server. Once the data is reached the middle tier server then the role of proxy comes into the picture wherein the security region the data is captured and reencrypted.
Later in the service integration block the entire web services are integrated based on the user request that is made by the Restful service architecture. During this methodology, the service generates two keys where one is private, and the other one is public. The private key here is used by the proxy where the data is received from the main server. The data use the public key to a service server to store and retrieve the request queries. In this way, the data is stored securely in the database based on the architecture. During this process, the user also tries to show the time that is taken for the secure data transfer between the main server and the DaaS server.

When the doctor is logged in with the credentials then the get patient details button is used to show all the patients that are stored in the database. If the patient is registered within the doctor then the details are shown else displays a message not authorized.

Figure 3.1 shows the architecture and design for the project. The end users here are the login registration pages that are created using the JSP and Java programming. After that, the data is redirected to the hospital application server where the data is encrypted and stored. The methodology here is used for the faster transfer of data from the main server to the application server and is used for the vast amount of data transfers securely.

The next module is used to retrieve the doctor document securely by the patient using a key. First the doctor uploads a file for a particular patient under him. After that the patient has to retrieve the data based on the key with which the file was encrypted. In this way there is secure patients document retrieval.
The key here is shared between the doctor and the patient to encrypt and decrypt the data in the file. Once the data is decrypted then the document is retrieved by the patient that is uploaded by the doctor.

Figure 3.1 System Architecture
There are three applications that are running in the project. The applications have its functionality and use to store the data securely in the database and protect the doctor and patient data from hacking procedures.

3.2 UML Diagrams

3.2.1 Use case Diagram

The functionalities that are shown in Figure 3.2 give a clear picture of the methods or the processes that take place in the complete architecture. The use case diagram here consists of use cases that have the methods and the actors that are responsible for the processes to take place. Basically, there are three servers in the project where one is the main server, the second one is the DaaS server and the last one is the proxy server that acts as an intermediary between the main and DaaS server.

![Use Case Diagram for Database and Proxy](image)

**Figure 3.2 Use Case for Database and Proxy**
Coming to the actions that take place are sent and receive encrypted data which in turn validates the service. After that, the data is received back to the main server from the proxy server in the form of the XML file that is encrypted and is now visible to the user.

In the main server application the data is received from the login details. The other tasks for the application are to update the doctor and patient details. Figure 3.3 shows the main application server use case with the methods.

![Figure 3.3 Use Case for Main Server](image-url)
3.2.2 Sequence Diagram for Admin login page

The sequence diagram gives an idea of the interaction of processes within the project. It consists of blocks and arrows for the flow of data. Figure 3.4 shows the methods that are used for the storage of data in the database server. In the entire application server, the login is done based on the validation.

Figure 3.4 Sequence Diagram for Database Server
3.2.3 Class Diagram

Class diagram in a project represents different classes, attributes, methods and their relationships. They are also used in data modeling and conceptual modeling. Figure 3.5 represents the class diagram for the complete with the classes, attributes and their relationships.

Figure 3.5 Sample Class Diagram
The class diagram have different classes namely databaseserverinterface, database server, DBClass, Security, Manframe, patientregister, ShowPatientdetails, main frame, proxy server, doctor register and log.

To upload and download the files there are some other classes related to the application server like the uploadfile, uploadfile1 and downloadpage1.

In this the database server interface acts as an interface class that is used to connect between the applications. The database server class is related to the DbClass which is used to store the data in the server database. The Manframe class is used to show the user interface for the proxy. Doctor register and patient register classes are respectively used to register the doctors and patients in the database. Showpatientdetails is used to retrieve the details from the database and show the results in the web page. The log class is responsible for the login of the admin and the doctor.

The classes here have different methods associated to perform some function. For example the DBClass have several functions like the storeDoctor, storePatient etc that are used to perform some task. All the classes also have attributes like the variable names, username, password etc.

The relationship between the classes plays a major role for the flow of the project. The databaseserverinterface class is related to databaseserver which in turn is related to DBClass. The DBClass is connected to security class that is used to encrypt and decrypt the data. The databaseserverinterface acts as an interface and is connected again to the proxyserver class which is related to the proxy server. The proxyserver is again
connected to the doctor register and log classes in the main server. In this way all the classes, its methods and attributes are related to each other.

### 3.3 Environment for the project

#### 3.3.1 Netbeans

Netbeans is used to develop, edit and execute Java code or files containing it. Netbeans is also user-friendly where it helps the programmer with the spaces, functions, brackets, source code and also helps in matching the words while programming. Netbeans is an integrated development environment that offers various tools for enterprises, desktop, mobile application and Java web development [20]. It also helps the user with some coding suggestions and generators. The IDE editor supports other programming languages also like C, PHP, C++ and HTML.

Netbeans is used to execute many languages and can run on multiple operating systems such as Windows, Linux, Solaris and other platforms that are compatible with Java virtual machine (JVM). The project here is done in Netbeans IDE 7.0 and is the latest version that provides many features of Java [20]. The Netbeans has various features such as framework wizards which accept dialogs, and many other features.

#### 3.3.2 Oracle Database 11G

Oracle databases have been the most widely used one for the storage and manipulation of the data. It is used to automate the database services and is more
effective in this version. Oracle 12C is also popular to provide advanced features like resource sharing, cost savings, and management flexibility.

The database connectivity in this is flexible with high density and is done without changing the requirements of already built applications. It also supports some advanced features like advance row, Network compression. It gives the database a good performance with the query. It also deals with automatic data optimization with security constraints having 500 new features [25].

3.3.3 Java Swing

Java swing is used to design the front end for any web application. It has many graphical user interface components that are helpful for building these applications [19]. It is an extension of the abstract windows toolkit interface with extra features and components. It also uses model view controller architecture to support the multiple look and usability [26].

3.4 System configuration

The system configuration for the project is shown in table 3.2. The operating system can be windows 7 or 8. The programming languages used for the project are Java, Java servlet pages and Java swings. To run the project the minimum requirement is JDK 1.7 with NetBeans and Oracle 11G as the backend database for the storage of the information.
Table 3.2 Software Requirements

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Windows 7 or 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Language</td>
<td>Java, JSP and Java Swings</td>
</tr>
<tr>
<td>Java Version</td>
<td>JDK 1.7</td>
</tr>
<tr>
<td>IDE</td>
<td>Netbeans</td>
</tr>
<tr>
<td>Database</td>
<td>Oracle 11G</td>
</tr>
</tbody>
</table>

3.5 Flowchart Diagram

Flowchart diagram is described as the flow of execution for any code or a process. It has a start and stop processes with other processes and decisions. All the components in the flowchart are specified with a shape that tells its functionality.

3.5.1 Database server Flowchart

Figure 3.6 describes the flow chart for the database server. In this application, the data transfer from the proxy server to database server is shown.
The data flow diagram gives the view about the operations that are going on in the database server. In this, the information is sent to the database server from the proxy server using 1522 port. After that, data is transferred then it is encrypted or decrypted and is stored in the database on the database server. In this way, the data is stored securely. The line in the figure 3.6 is the separation between the applications.

3.5.2 Proxy Server Flowchart

Figure 3.7 represents the proxy server flowchart that describes the information transfer from the main server to the proxy server using the ports.
In this, the data is transferred to the patient, hospital and doctor services from the ports. Before it is done, the data is encrypted using an XML parser for security reasons and is transferred using the 1800 and 1900 ports.

### 3.5.3 Flowchart for Document Uploading

For the uploading of the file which can be a text document the doctor has to login with his credentials. After the doctor is logged in, there is another option called upload file for patient that is used to upload a file for a patient. The doctor can upload a text file for any patient under him using a key so that the file is encrypted and stored in database. Figure 3.8 shows the flow of process for the uploading of file for the doctor.
3.5.3 Flowchart for Document Retrieval

Once the doctor has uploaded the file now the patient will be able to retrieve the file only with the key that was used to encrypt the file which should be shared. Once the document retrieval button is clicked then the patient has to enter the key to decrypt the file and download it. Figure 3.9 shows the flow for the document retrieval mechanism.
3.6 Database Design

The implemented project uses the Oracle 11G database for the storage of encrypted data. In the olden days the data was stored in the memory but due to some disadvantages like paging that was ruled out. The queries in the 11G are similar to those of the SQL and are also used to execute complex queries.
The data in a table here is stored in rows and columns. The row is defined in the table using a table name, datatype, column name and width. Once the query is executed then the data is stored in the database in the form of rows and columns. The Oracle database also has a concept of database compression that is used to compress the repeated values and are replaced by a short reference for a symbol table [27].

The database used here has five tables namely the doctors, patients, doctorskey, patientkeys and filetable. In all the tables the data that is given in the registration are stored securely in the database tables in their respective fields.

For example the doctors table shown in Table 3.2 consists of fields like username, password, fullname, phone, email and address. After the data is captured using an XML string then it is stored in these fields in the database. After that the retrieval of data is done in the next phase of the project where the data is decrypted and are retrieved from the field in the database.

<table>
<thead>
<tr>
<th><strong>Username</strong></th>
<th><strong>Doc1</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Password</strong></td>
<td><strong>doc1</strong></td>
</tr>
<tr>
<td><strong>Fullname</strong></td>
<td><strong>Doctor1</strong></td>
</tr>
<tr>
<td><strong>Phone</strong></td>
<td><strong>3618776565</strong></td>
</tr>
<tr>
<td><strong>Email</strong></td>
<td><strong><a href="mailto:Doc1@gmail.com">Doc1@gmail.com</a></strong></td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td><strong>Corpus Christi</strong></td>
</tr>
</tbody>
</table>
Similarly the other table that is used in the project is the patients table that is used to store the patient’s information from the patient registration page. The patient tables shown in Table 3.3 have fields like fullname, phone, email, address, doctor and disease. Once the data is entered in the patient registration, then the data is stored in their respective fields securely in the database.

**Table 3.3 Sample Patients Database Table**

<table>
<thead>
<tr>
<th>Fullname</th>
<th>Patient1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>3565243543</td>
</tr>
<tr>
<td>Email</td>
<td><a href="mailto:Patient1@gmail.com">Patient1@gmail.com</a></td>
</tr>
<tr>
<td>Address</td>
<td>Austin</td>
</tr>
<tr>
<td>Doctor</td>
<td>Doc1</td>
</tr>
<tr>
<td>Disease</td>
<td>Cancer</td>
</tr>
</tbody>
</table>

The next two tables in the database are used to store the keys that are given during the registration in patient and doctor. The fields in this table are fullname or username, key1 and key2. The table 3.4 and 3.5 respectively show the sample doctorskey and patientkeys tables in the database with its fields. The keys used are 16 bit keys for both the tables.
Table 3.4 Sample Doctorskey Database Table

<table>
<thead>
<tr>
<th>Username</th>
<th>Doc1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key1</td>
<td>Thisisasecurekey</td>
</tr>
<tr>
<td>Key2</td>
<td>Thisisasecurekey</td>
</tr>
</tbody>
</table>

Table 3.5 Sample Patientkeys Database Table

<table>
<thead>
<tr>
<th>Fullname</th>
<th>Patient1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key1</td>
<td>qwertyuiopasdfgh</td>
</tr>
<tr>
<td>Key2</td>
<td>qwertyuiopasdfgh</td>
</tr>
</tbody>
</table>

The table 3.6 is used to store the doctor name, patient name for the uploading of the file. It also has the file name along with the secret key that is shared by the doctor and patient to retrieve the data.

Table 3.6 Sample Filetable for Document Retrieval

<table>
<thead>
<tr>
<th>Docname</th>
<th>Doc1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient</td>
<td>Pat1</td>
</tr>
<tr>
<td>File name</td>
<td>abc.txt</td>
</tr>
<tr>
<td>Key</td>
<td>thisisasecretkey</td>
</tr>
</tbody>
</table>
4. PROJECT IMPLEMENTATION

The project consists of three applications running namely the main server, proxy server and the database server. In the main application there are three modules. One is the doctor registration, next is the patient registration and the last thing is the doctor login to view patient details.

Figure 4.1 below is the home page for one of the three application servers. The page is for the database server in the project. First the database server is started using the port 1800.

![Database Server Application Page](image)

**Figure 4.1 Database Server Application Page**

Below is the Figure 4.2 that gives the star up page for the proxy server that acts as an intermediary between the main server and the database server. In this page, it has one component to start the proxy server so as to store the data and transfer it securely. Now the proxy server is started using the port 1900.
Figure 4.2 Proxy Server Home Page

Figure 4.3 shows the index page for the main server. Once the proxy server is been started, then it invokes some code to start services. Once the process is started, then the doctor and patient services will be called to load the data from the user end.

In the index page there are two types of login; one is for the admin that is used to store the data for the doctors and the patients. Once the admin login is done, then the login page has two options. First one is the new doctor that is used to store the doctor related information. Second one is the new patient that is used to store the patient related information. After the registration the data is encrypted and is stored as an XML string and is sent to the proxy server. The procedure can be seen in the proxy server console that runs in the background. After the data reaches the proxy server, then it is again encrypted and is sent to the database server. Finally the data is encrypted again and is stored securely in the database server tables.
Figure 4.3 Admin Login Page

Figure 4.4 is the login page for the doctor. Once the doctor login is done the get patient details that are used to retrieve back the data from the database.

Figure 4.4 Doctor Login Page
If the get patient details button is clicked then all the patients that are registered are shown in tabular format. If the patient is registered under the current doctor then the patient details are shown else the patients details are not shown as it is an unauthorized access. Figure 4.5 shows the list of patients that are registered. In this scenario as the pat1 is registered under the doctor 1 so the details are shown for it. The pat2 is not an authorized one so it displays a message stating not an authorized person. To get the pat1 detail the database tables are searched with the username and is retrieved back decrypting it in the database server and is sent to the proxy server. In the proxy the data is decrypted again using the keys already hardcoded in the proxy and is sent to the main server. Finally the data is decrypted using the keys stored in the patientkeys table and the results are shown.

![Figure 4.5 Doctor 1 Patients List](image-url)
After the patient details are shown it also has a doctor button that is used to display the doctor details that are related to that patient. Figure 4.6 shows the patient 2 details displayed with the doctor button. If the doctor button is clicked then it shows the doctor details related to patient 1.

![Figure 4.6 Patient 1 Details](image)

In this way the data is stored securely in the database with multi-level encryption at application level. It also has an authorization mechanism that is used to prevent unauthorized access to the details which maintains data privacy at all the levels for the doctors and the patients. The tables and the authentication are also done in a real time scenario with multiple doctors and patients.
The next module in the project is to upload a file for the patient by the doctor and retrieve it back by the patient. The mechanism is achieved using a key that is shared both by the doctor and patient for encrypting and decrypting the file and retrieve it. Figure 4.7 shows the uploading of file by the doctor in his login.

Figure 4.7 Document Uploading

After the document is uploaded by the doctor for the patient using the key ten the file is encrypted and stored in database. Now the doctor can logout and in the home page the retrieval mechanism is possible.
In the home page the document retrieval button is used to retrieve the document for the patient. Once the key is entered then the files is downloaded and view the data. Figure 4.8 shows the retrieval mechanism for the patient. Once the get file button is clicked then the file is downloaded and can be viewed.

**Figure 4.8 Document Retrieval**
5. TESTING

The testing for this project can be done with various user login and registration requests from different services. The test cases for the project can be in five modules.

5.1 Doctor Test Case

First one is the doctor test case that is used to store the doctor details in the database table. The data is stored mainly with the username for the doctor that is doc1. First the new doctor link is clicked, which gives the registration page. Once the doctor details are filled with doc1 username and the register button is clicked then the data is updates in the database in the database server securely. Figure 5.1 shows the doctor registration page with its fields. After the details are filled, they are securely stored in the database server.

Figure 5.1 Doctor Registration Test Case
5.2 Patient Test Case within Doctor

Third test case is the patient login page where the registration for the patients is done. It is a new patient with full name pat1. After the admin login the new patient link is used to get to the patient registration page. Figure 5.2 shows the registration page where the details are filled. In the page there is one more field that is used to select the doctor under which the patient must register.

![Figure 5.2 Patient Registration Test Case](image)

Once the patient is registered, the pat1 is registered under the doc2 username doctor. So the authentication for the patient 1 is done only by the doctor 1 and with no other doctor.
5.3 Doctor Login Test Case

In the home page now the login can be done as a doctor instead of admin. So once the doctor 1 is logged in with the credentials then a new page appears with a button to show the patient details. If the button is clicked then all the patients that are registered are shown as a list. Figure 5.3 shows the list of all the patients within doctor 1.

![Figure 5.3 Doctor 1 Patients List](image)

The login is for the doctor 1 who has only one patient under it that is patient 2. So if the get details are clicked for patient 1 then it displays a message stating not an
authorized user. Similarly if the get details button is clicked for the patient 2, then the details are shown in figure 5.4.

![Figure 5.4 Patient 2 Details for Doctor 1](image)

After the details are shown, the doc1 button is clicked to view the details of the doctor 1 that is related to the patient,. In this the data is again retrieved form the database with decrypting the data and is shown in Figure 5.5.
Figure 5.5 Doctor Details for Patient 2

In the similar way the doctor 2 can be logged in to view only the details of patient 1 as it is registered under the doctor. If the doctor 2 wants to see the patient 2 details then the authorization id blocked displaying a message.

5.4 Test Case for Document Uploading by Doctor

The next test case is for the uploading a file for the patient by the doctor. The doctor can now login and then upload a medical file for a patient. The file is encrypted and uploaded for the patient. Figure 5.6 shows the uploading for the document.
Figure 5.6 Document Uploading by Doctor

When the go to upload button is clicked then the file is uploaded for that particular doctor. Figure 5.7 shows the status for file uploading.

Figure 5.7 Status of Document Upload
5.5 Test Case for Document Retrieval by Patient

Once the doctor has uploaded the document to the patient, the patient can retrieve the file using the key provided by the doctor. The file is decrypted and is downloaded. Figure 5.8 shows the retrieval mechanism for the patient.

The patient under the doctor can download the file using the key. Once the get file button is clicked then the file is downloaded successfully and can be viewed. Figure 5.9 shows the status for the file download.
Figure 5.9 File download Status

The file is downloaded and can be viewed that has the data sent by the doctor to the patient.
6. CONCLUSION AND FUTURE WORK

This project proposes a clear idea of how to preserve the data from being exposed to others. The project focuses on the data that is related to someone’s personal information that must be kept private to avoid misuse. The architecture that is used here is based on the composition two services that is doctor and patient services, where the user gives information in the form of a web page from where the data is processed using the main server. After the processing of data, it is sent to the proxy server where further processing of data is done based on the functionalities in the architecture. After that, the data is securely stored in the data as a service database and retrieve the information.

The next module is to upload and download a file securely by the doctor and patient with a shared key so that the data is encrypted and decrypted. The doctor uploads the file for the patient using a key and then the patient can download the file and view the data with the shared key. In this way there is a secure patient’s document retrieval.

Future work for the project is based on the scope of the project. The project can be further implemented in other domains or fields and can also be added to the cloud computing field with furthermore access to other technologies.


[20] https://netbeans.org/features/


[27] http://docs.oracle.com/cd/B28359_01/server.111/b28318/schema.htm#CNCPT1132


[29] https://diuf.unifr.ch/drupal/softeng/sites/diuf.unifr.ch.drupal.softeng/files/file/publications/internal/WP09-03.pdf
APPENDIX

The following code is the security file for the project that performs AES encryption.

Security.java

package vv.we;

import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;

public class Security
{
    public String encrypt(String data, String KEY1, String KEY2)
    {
        byte[] ENCRYPTED=new byte[2];
        try {
            IvParameterSpec IVPARAMETERSPEC = new IvParameterSpec(KEY2.getBytes("UTF-8"));

            SecretKeySpec SECRETKEYSPEC = new SecretKeySpec(KEY1.getBytes("UTF-8"), "AES");
            Cipher CIPHER = Cipher.getInstance("AES/CBC/PKCS5PADDING");
            CIPHER.init(Cipher.ENCRYPT_MODE, SECRETKEYSPEC, IVPARAMETERSPEC);
            ENCRYPTED = CIPHER.doFinal(data.getBytes());
        }
    }
}
System.out.println("ENCRYPTED string:" +
Base64.encodeBase64String(ENCRYPTED));

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

return Base64.encodeBase64String(ENCRYPTED);

public String decrypt(String enc_data, String KEY1, String KEY2)
{
    byte[] original=new byte[2];
    try {
        IvParameterSpec IVPARAMETERSPEC = new IvParameterSpec(KEY2.getBytes("UTF-8"));

        SecretKeySpec SECRETKEYSPEC = new SecretKeySpec(KEY1.getBytes("UTF-8"), "AES");

        Cipher CIPHER = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        CIPHER.init(Cipher.DECRYPT_MODE, SECRETKEYSPEC, IVPARAMETERSPEC);
        original= CIPHER.doFinal(Base64.decodeBase64(enc_data));
    } catch (Exception ex) {
        ex.printStackTrace();
    }

    return new String(original);
}
The following code is for the uploading the document by the doctor to the patient.

**Uploadfile.java**

```java
package vv.we;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.util.List;
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.*;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.RequestContext;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

@WebServlet(name = "UploadFile2", urlPatterns = {"/UploadFile2"})
public class UploadFile2 extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
```
try {

    HttpSession session = request.getSession();

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

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

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

    List<FileItem> formItems = null;

    InputStream fileContent = null;

    String fileName = "";

    String patientUserName = "";

    String hashCode = "";

    try {

        formItems = new ServletFileUpload(new
        DiskFileItemFactory()).parseRequest(request);

    }catch(Exception e){

        e.printStackTrace();

    }

    for (FileItem item : formItems) {

        if (item.isFormField()) {

        }

    }

    // Process form file field (input type="file").

    String fieldName = item.getFieldName();

    fileName = item.getName();

    System.out.println("fieldName:"+fieldName);

    System.out.println("FileNAme:"+fileName);

    fileContent = item.getInputStream();

}
DBClass dbc = new DBClass();

// InputStream inputStream = null; // input stream of the upload file

// obtains the upload file part in this multipart request

String text = "";
int i = 0;
while((i=fileContent.read())!=-1)
{
    text = text+(char)i;
}
System.out.println(text);
System.out.println(key);
boolean b = dbc.uploadFile(user, patient, key,fileName, text);
if(b)
{
    out.println("File Uploaded Successfully.");
    out.println("<a href='DoctorHome.jsp'>Back</a>");
}
else
{
    out.println("File not uploaded...");
    out.println("<a href='UploadFile.jsp'>Try again</a>");
}
The following code is for the interface class for the applications.

**Databaseclassinterface.java**

```java
import java.rmi.*;
import java.util.*;
public interface DatabaseServerInterface extends Remote
{
    public String storeDoctor(String username, String password, String fullname, String phone, String email, String address, String key1, String key2) throws RemoteException;
    public String storePatient(String fullname, String phone, String email, String address, String doctor, String disease, String key1, String key2) throws RemoteException;
    public List<String> getDoctorNames() throws RemoteException;
    public boolean loginDoctor(String username, String password) throws RemoteException;
    public List<String> getPatientNames() throws RemoteException;
    public String getPatientDetails(String username, String docname) throws RemoteException;
    public String getDoctorDetails(String user) throws RemoteException;
}
```

The following code is a part of the DBclass for storing the data.

**DBClass.java**

```java
public class DBClass
{
    private String USERNAME = "system";
    private String PASSWORD = "Admin123";
```
private String URL = "jdbc:oracle:thin:@localhost:1521:orcl";
private String DRIVER = "oracle.jdbc.OracleDriver";
private Connection con = getConnection();
private Statement st = null;
private ResultSet rs = null;
private PreparedStatement ps = null;
private Security sec = new Security();

public Connection getConnection()
{
    Connection con = null;
    try {
        Class.forName(DRIVER);
        con = DriverManager.getConnection(URL, USERNAME, PASSWORD);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return con;
}

public String storeDoctor(String username, String password, String fullname, String phone, String email, String address, String key1, String key2)
{
    String msg = "";
    try {
        String QUERY1 = "INSERT INTO DOCTORSKEY VALUES(?,?,?)";
        ps = con.prepareStatement(QUERY1);
        ps.setString(1, username);
        ps.setString(2, password);
        ps.setString(3, fullname);
        ps.executeUpdate();
        ps.close();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return msg;
}
ps.setString(2, key1);
ps.setString(3, key2);
int j = ps.executeUpdate();
ps.close();

String QUERY = "insert into doctors values(?,?,?,?)";
ps = con.prepareStatement(QUERY);
ps.setString(1, username);
ps.setString(2, sec.encrypt(password, key1, key2));
ps.setString(3, sec.encrypt(fullname, key1, key2));
ps.setString(4, sec.encrypt(phone, key1, key2));
ps.setString(5, sec.encrypt(email, key1, key2));
ps.setString(6, sec.encrypt(address, key1, key2));
int i = ps.executeUpdate();
if(i>0)
{
    msg = "Doctor Data Updated in Database Server....";
}
else
{
    msg = "Doctor Data Not updated properly...";
}
ps.close();
}
catch(Exception e)
{
    e.printStackTrace();
}
return msg;