

CLOUD BASED BIOMETRIC SECURITY FOR ORGANIZATIONS

**A PROJECT REPORT**

***Submitted by***

**ABHINAYA.R [REGISTER NO:211417104002] ABITHA.C [REGISTER NO:211417104004]**

**PRIYANKA.B [REGISTER NO:211417104204]**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report **“CLOUD BASED BIOMETRIC SECURITY FOR ORGANIZATIONS”**is the bonafide work of

**ABHINAYA R (211417104002)**

**ABITHA C (211417104004)**

**PRIYANKA B(211417104204)**

who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr.S.MURUGAVALLI,M.E.,Ph.D.,**  **HEAD OF THE DEPARTMENT** | **SIGNATURE**  **R.SALINI,M.TECH.,**  **ASSISTANT PROFESSOR** |
| DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE,  NAZARTHPETTAI,  POONAMALEE,  CHENNAI – 600 123 | DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE,  NAZARTHPETTAI,  POONAMALEE,  CHENNAI – 600 123 |

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent

**Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic

motivation, which inspired us a lot in completing this project.

We would like express our heartfelt and sincere thanks to our Director **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR, M.E., Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my Project Guide **R. SALINI, M.TECH.,**  and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**ABHINAYA R**

**ABITHA C**

**PRIYANKA B**

# ABSTRACT

The demand for remote data storage and computation services is increasing exponentially in our data-driven society; thus, the need for secure access to such data and services. In this paper, we design a new biometric-based authentication protocol to provide secure access to a remote (cloud) server. In the proposed approach, we consider biometric data of a user as a secret credential. We then derive a unique identity from the user’s biometric data, which is further used to generate the user’s private key. In addition, we propose an efficient approach to generate a session key between two communicating parties using two biometric templates for a secure message transmission. In other words, there is no need to store the user’s private key anywhere and the session key is generated without sharing any prior information. A detailed Real-Or- Random (ROR) model based formal security analysis, informal (non-mathematical) security analysis and also formal security verification using the broadly-accepted Automated Validation of Internet Security Protocols and Applications (AVISPA) tool reveal that the proposed approach can resist several known attacks against (passive/active) adversary. Finally, extensive experiments and a comparative study demonstrate the efficiency and utility of the proposed approach.

|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS** | | |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | **IV** |
|  | **LIST OF FIGURES** |  |
|  | **LIST OF ABBREVATIONS** |  |
| **1** | **INTRODUCTION** |  |
|  | 1.1 Overview |  |
|  | 1.2 Problem definition |  |
| **2** | **LITERATURE SURVEY** |  |
| **3** | **SYSTEM ANALYSIS** |  |
|  | 3.1 Existing system |  |
|  | 3.2 Proposed system |  |
|  | 3.3 Requirement analysis and Specification |  |
|  | 3.3.1 Introduction |  |
|  | 3.3.2 Hardward and Software specification |  |
|  | 3.4 Technology stack |  |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
| **4** | **SYSTEM DESIGN** |  |
|  | 4.1 Er diagram |  |
|  | 4.2 Use case diagram |  |
|  | 4.3 Sequence diagram |  |
|  | 4.4 Activity diagram  4.5 Data Flow Diagram  4.6 Class Diagram |  |
|  |  |  |
| **5** | **SYSTEM ARCHITECTURE** |  |
|  | 5.1 Architecture diagram |  |
|  | 5.2 Modules design specification |  |
|  | 5.2.1 Cloud owner and user registration and key  Generation |  |
|  | 5.2.2 Cloud owner upload file to cloud server |  |
|  | 5.2.3 Decrypt and download file |  |
|  | 5.3 Algorithms used |  |
| **6** | **SYSTEM IMPLEMENTATION** |  |
|  | 6.1 Client side coding |  |
|  | 6.2 Server side coding |  |
| **7** | **SYSTEM TESTING** |  |
|  | 7.1 Coding |  |
|  | 7.2 Coding standards |  |
|  | 7.2.1 Naming conventions |  |
|  | 7.2.2 Value conventions |  |
|  | 7.2.3 Script writing and commenting standard |  |
|  | 7.2.4 Message box format |  |
|  | 7.3 Test procedure |  |
|  | 7.3.1 System testing |  |
|  | 7.4 Test data and output |  |
|  | 7.4.1 Unit testing |  |
|  | 7.4.2 Funcitonal tests |  |
|  | 7.4.3 Performance test |  |
|  | 7.4.4 Stress test |  |
|  | 7.4.5 Structued test |  |
|  | 7.4.6 Integration testing |  |
|  | 7.5 Testing techniques / testing strategies |  |
|  | 7.5.1 Testing |  |
|  | 7.5.2 Software testing strategies |  |
|  | 7.5.2.1 Integration testing |  |
|  | 7.5.2.2 Program testing |  |
|  | 7.5.2.3 Security testing |  |
|  | 7.5.2.4 Validation testing |  |
| **8** | **CONCLUSION** |  |
|  | 8.1 Conclusion and Feature Enhancements |  |
|  | **SAMPLE SCREENS** |  |
|  | **REFERENCES** |  |

**LIST OF FIGURES**

4.1 ER Diagram

4.2 Use Case Diagram

4.3 Sequence Diagram

4.4 Activity Diagram

4.5 Collaboration Diagram

4.6 Data Flow Diagram

4.7 Class Diagram

# LIST OF ABBREVATIONS

# JDK Java Development Toolkit

DEX Dalvik Executables

TCP Transmission Control Protocol

IP Internet Protocol

**HTTP** Hyper Text Transfer Protocol

**ADT** Android Development Tool

**CHAPTER 1**

# INTRODUCTION

## 1.1 Overview:

The demand for remote data storage and computation services is increasing exponentially in our data-driven society; thus, the need for secure access to such data and services. In this paper, we design a new biometric-based authentication protocol to provide secure access to a remote (cloud) server. In the proposed approach, we consider biometric data of a user as a secret credential. We then derive a unique identity from the user’s biometric data, which is further used to generate the user’s private key. In addition, we propose an efﬁcient approach to generate a session key between two communicating parties using two biometric templates for a secure message transmission. In other words, there is no need to store the user’s private key anywhere and the session key is generated without sharing any prior information. A detailed Real-OrRandom (ROR) model based formal security analysis, informal (non-mathematical) security analysis and also formal security veriﬁcation using the broadly-accepted Automated Validation of (AVISPA)toolreveal that the proposed approach h can resist several known attacks against (passive/active) adversary. Finally, extensive experiments and a comparative study demonstrate the efﬁciency and utility of the proposed approach..

## 1.2 Problem Definition:

File sharing among multiple Clients is not secure. To generate a revocable private key directly from an irrevocable ﬁngerprint image. There is no need to store the private key or a direct form of the user’s biometric data anywhere. In the authentication phase, we capture a new biometric ﬁngerprint image of the user, and subsequently generate the private key and encrypt the biometric data as a query. This queried biometric data is then transmitted to the authentication server for matching with the stored data

**CHAPTER 2**

1. **LITERATURE SURVEY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **YEAR** | **AUTHOR NAME** | **PROJECT TITLE** | **MERITS** | **DEMERITS** |
| 1. | Published in journal (2019) | Wencheng Yang, Song Wang,Jiankun Hu, Guanglou Zheng and Craig Valli | Security and Accuracy of Fingerprint-Based Biometrics: A Review | One of the advantages of biometric cryptosystems is that they can bind or directly generate a cryptographic key, which can be used for both authentication and data encryption. The advantages of both biometrics and cryptosystems. Different to a cancelable biometric system, which can only provide a match or non-match report, a biometric cryptosystem can output a key by either. | It is infeasible or computationally difﬁcult to retrieve raw template data. |
| 2. | Published in conference(2018) | Zhihua Xia, Xingming Sun, Neal N. Xiong | A Novel Weber Local Binary Descriptor for Fingerprint Liveness Detection | The method consists of two components: the local binary differential excitation component that extracts intensity-variance features and the local binary gradient orientation component that extracts orientation features. | The calculation model of local binary pattern (LBP) to improve the two components of the original WLD. |
| 3. | Published in conference (2018) | Srinivas Jangirala, Mohammad Wazid | Anonymous Lightweight Chaotic Map-Based Authenticated Key Agreement Protocol for Industrial Internet of Things | A registered user is facilitated to update password & biometrics without the involvement of the GWN. The scheme is very efﬁcient as it uses only cryptographic hash function along with the symmetric encryption/decryption | we present a new three-factor anonymous lightweight authentication protocol suitable for IIoT. A user can access the information of any smart device of monitoring group through the central controller. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4. | Published in Journal (2017) | Mohammad Wazid, Neeraj Kumar, Mauro Conti, Minho Jo | Design of Secure User Authenticated Key Management Protocol for Generic IoT Networks | Different Scanners capture different feature so fingerprints with varying sizes. For higher accuracy, the network must be trained individually for each scanner | Although transfer learning is an efficient approach to machine learning problems with  a limited number of training samples, it has a disadvantage in that the computation time cannot be smaller than the pertained networks. |
| 5. | Published in Journal (2017) | HO YUB JUNG1, YONG SEOK HEO2, SOOCHAHN LEE | Fingerprint liveness detection by a  template-probe convolutional neural network | The ﬁngerprint scanners used in this paper produce grayscale ﬁngerprint images with a white background and a dark foreground.  For higher accuracy, the network must be trained individually for each scanner. | It has a disadvantage in that the computation time cannot be smaller than the pertained networks |

**CHAPTER 3**

**3. SYSTEM ANALYSIS**

## 3.1 Existing system

In existing system uploading and sharing file among multiple client user in cloud environment is very hard to perform and there is no proper authentication among the cloud user and to the cloud server. Hence the file that uploaded on the cloud would not be secure as there is a lot of security problem that related to the cloud storage. In the authentication phase, we capture a new biometric ﬁngerprint image of the user, and subsequently generate the private key and encrypt the biometric data as a query. This queried biometric data is then transmitted to the authentication server for matching with the stored data

## 3.2 Proposed system

In Proposed system, we introduced secure and efficient ﬁle storage and sharing cloud environment using Remote cloud. Mutual authentication is very important between two target device as the user and to the cloud server as mutual authentication makes the server to trust the cloud storage where cloud owner can store the data and from server end, server will verify the user credential to provide any service to that particular user. Cloud contains the encrypted information along with the parameter related to that file. If any user request to the cloud server with some attribute. Based on the attribute cloud will redirect all the related file to the user in order to decrypt the file user need the key for that file for that user request to RSA and DSE(Key Generation Centre) with the file attribute

**Advantage :**

Biometric has its unique advantages over conventional password and token-based security system, as evidenced by its increased adoption

## 3.3 Requirement analysis and Specifications

### 3.3.1 Introduction :

Cloud services are a norm in our society. However,providing secure access to cloud services is not a trivialtask, and designing robust authentication, authorization and accountingfor access is an ongoing challenge, both operationallyand research-wise. A number of authentication mechanismshave been proposed in the literature, such as those basedon Kerberos , OAuth and OpenID . Generally, these protocols seek to establish a securedelegated access mechanism among two communicating entities connected in a distributed system. These protocols arebased on the underlying assumption that the remote serverresponsible for authentication is a trusted entity in the network.Specifically, a user first registers with a remote server. Thisis needed to ensure the authorization of the owner. When auser wishes to access a server, the remote server authenticates the user and the user also authenticates the server. Onceboth verifications are successfully carried out, the user obtainsto the services from some remote server. One key limitation in existing authentication mechanismsis that the user’s credentials are stored in the authentication server, which can be stolen and misused to gain unauthorizedaccess to various services. Also, to ensure secure and fast communication, existing mechanisms generally use symmetric key cryptography, which requires a number of cryptographic keys to be shared during the authentication process. Thisstrategy results in an overhead to the authentication protocols.

Therefore, in this paper we seek to designa secure and efficient authentication protocol. Specifically, wewill first provide an alternative to conventional password-based authentication mechanism. Then, we demonstrate how one canbuild a secure communication between communicating partiesinvolved in the authentication protocol, without having anysecret pre-loaded (i.e., shared) information.In the proposed approach, we consider a fingerprint imageof a user as a secret credential. From the fingerprint image, generate a private key that is used to enroll the user’scredential secretly in the database of an authentication server.In the authentication phase, we capture a new biometricfingerprint image of the user, and subsequently generate the private key and encrypt the biometric data as a query. Thisbiometric data is then transmitted to the authentication server for matching with the stored data. Once the user isauthenticated successfully, he/she is ready to access his/herservice from the desired server. To obtain secure access tothe service server, mutual authentication between the user andauthentication server, and also between the user and serviceserver have been proposed using a short-term session key.Using two fingerprint data, we present a fast and robustapproach to generate the session key. In addition, a biometricbasedmessage authenticator is also generated for messageauthenticity purpose.

### 3.3.2 Hardware and Software specification

**3.3.2.1 Hardware requirements**

* Hard Disk : 250GB and Above
* RAM : 4GB and Above
* Processor : i3 and Above

**3.3.2.2 Software requirements**

* Windows 10
* JDK 1.8
* My SQL 5.0
* Tomcat 9

## Technology Stack :

* J2EE(JAVA,JSP,JSTL,JSON)
* JavaScript , HTML , CSS
* Spring MVC

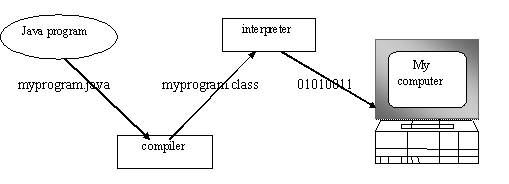
**3.4.1 Java**

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

**3.4.2 Working of Java**

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it   looks for a particular method by the name of **main,** which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv [] of C), and is declared as a static method.

To output text from the program, we execute the **println** method of **System.out,** which is java’s output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user’s program. ****

**3.4.3 Apache tomcat server**

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

**3.4.4 Introduction to Cloud Computing**

Cloud Computing provides us means of accessing the applications as utilities over the Internet. It allows us to create, configure, and customize the applications online.

The term Cloud refers to a Network or Internet**.** In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over public and private networks, i.e., WAN, LAN or VPN.

Applications such as e-mail, web conferencing, customer relationship management (CRM) execute on cloud. Cloud Computing refers to manipulating**,** configuring**,** and accessing the hardware and software resources remotely. It offers online data storage, infrastructure, and application.

**3.4.5 Cloud Computing Technologies**

There are certain technologies working behind the cloud computing platforms making cloud computing flexible, reliable, and usable. These technologies are listed below:

* Virtualization
* Service-Oriented Architecture (SOA)
* Grid Computing
* Utility computing

### 3.4.6 SystemFeatures

#### From a captured user’s ﬁngerprint image, we extract all minutiae points. In order to increase the accuracy in feature extraction, we ﬁrst align the ﬁngerprint image. From this aligned ﬁngerprint image, we select the consistent region. The consistent region can be deﬁned as the ﬁngerprint region, which has a high chance of appearance in any captured ﬁngerprint image. We select this consistent region to extract the minutiae points. To select a set of minutiae points from the consistent region, we propose to use a horizontal segment. Horizontal segment is a small area of the consistent region, which has the highest number of minutiae points. We select these minutiae points to generate a Trellis diagram of the convolution coding and ﬁnally, a codeword Chapter 4

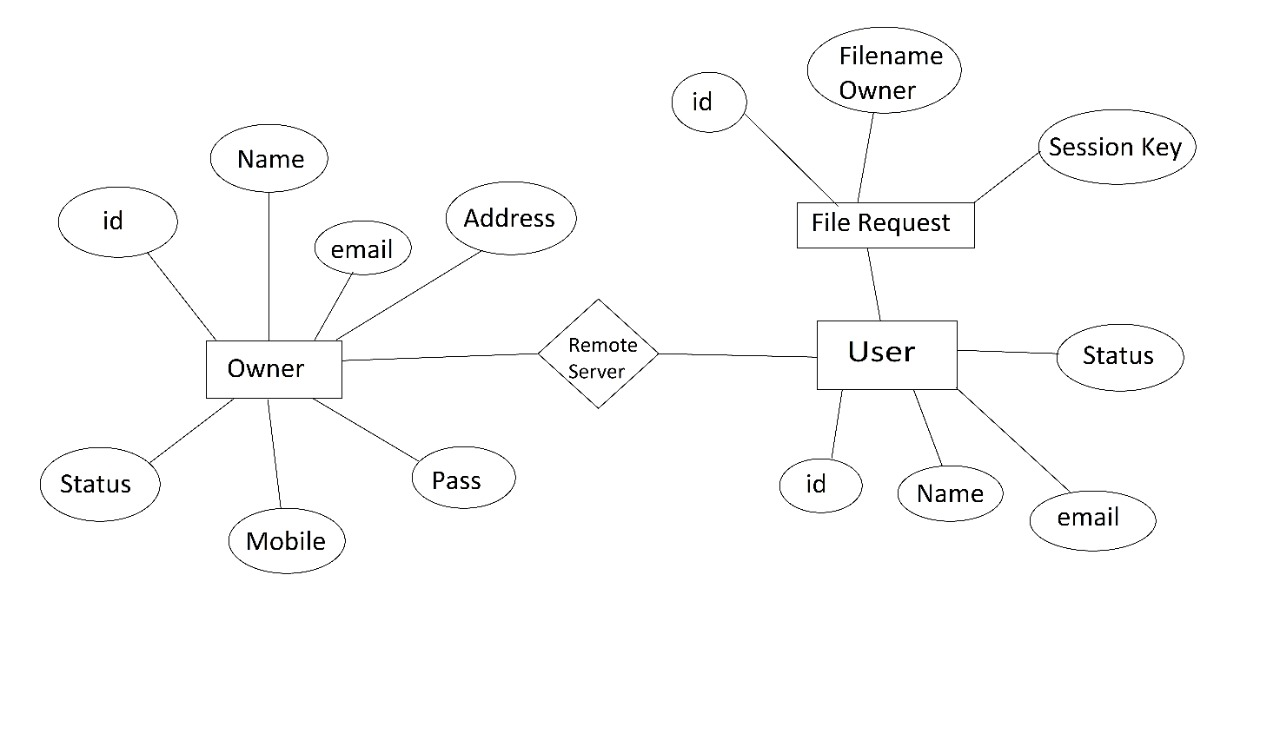
# SYSTEM DESIGN

BioCAP has two key processes, namely: user registration and user authentication. The user registration requires private key generation, whereas user authentication requires generation of the session key and the message authenticator. BioCAP provides a provision to rollover the private key of a user. In addition, BioCAP is secure, computationally less expensive, and overcomes the inherent weaknesses of biometric veriﬁcation. Moreover, BioCAP does not need pre-shared keys, and provides smooth mutual authentication mechanism and demands less number of keys to be managed from application and user point of view.

A systemic approach is required for a coherent and well-running system. Bottom-Up or Top-Down approach is required to take into account all related variables of the system. A designer uses the modelling languages to express the information and knowledge in a structure of system that is defined by a consistent set of rules and definitions. The designs can be defined in graphical or textual modelling languages.

**4.1 ER Diagram**

An **Entity Relationship** (**ER**) **Diagram** is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. They mirror grammatical structure, with entities as nouns and relationships as verbs.



## 

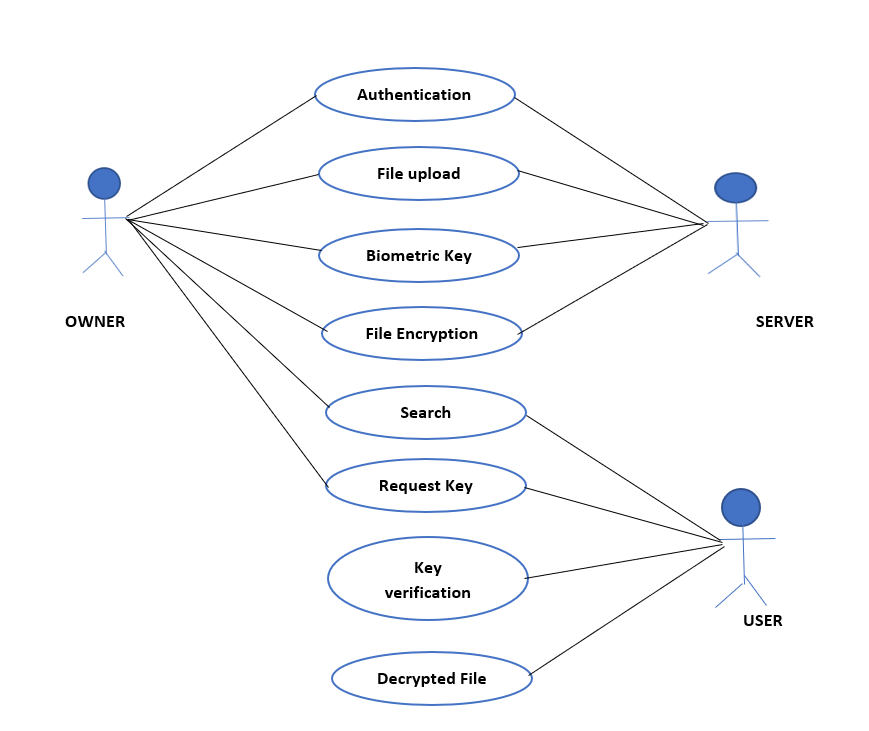
**FIGURE 4.1**

## 4.2 Use Case Diagram:

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

### 4.2.1. Usecase diagram

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.



**FIGURE 4.2**

## 4.3 Sequence Diagram

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

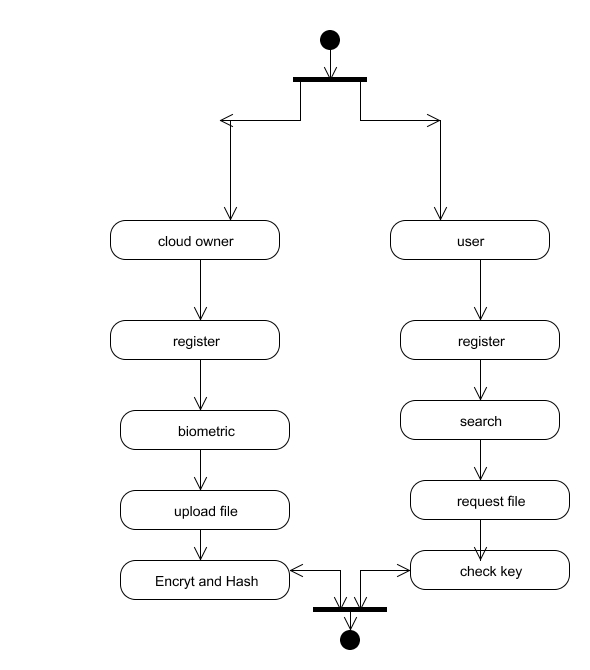
## FIGURE 4.3

## 4.4 Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

* Rounded rectangles represent activities.
* Diamonds represent decisions.
* Bars represent the start or end of concurrent activities.
* A black circle represents the start of the workflow.
* An encircled circle represents the end of the workflow.

****

**FIGURE 4.4**

**FIGURE 4.5**

**4.5 Data Flow Diagram**

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

**LEVEL 0**

Owner/user

register

Biometric

Generate key

server

server

owner

user

authentication

proceed

**LEVEL 1**

**Level 2**

owner

Upload file

Bio key

Encrypt key

Server key verfication

**Level 3**

user

Search

Request file

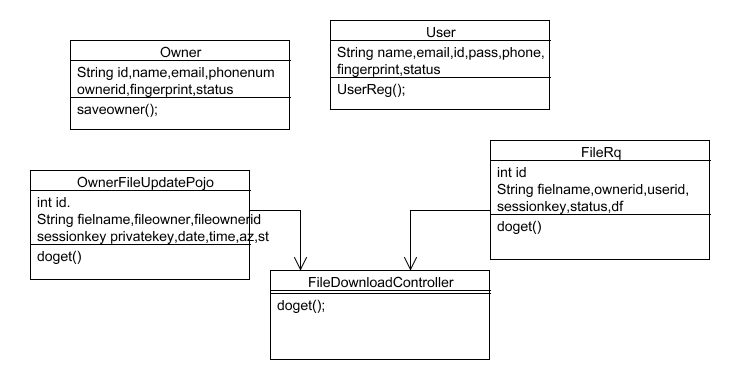
Key sent

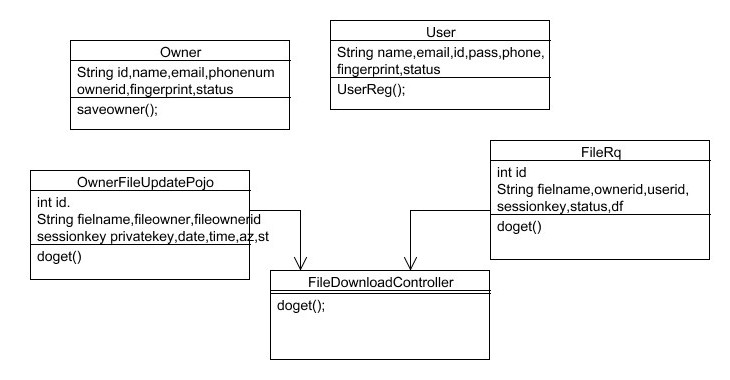
User view file

**FIGURE 4.6**

## 4.6 Class Diagram

A Class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

****

****

**FIGURE 4.7**

**CHAPTER 5**

**SYSTEM ARCHITECTURE**

* 1. **Architecture Diagram**



KGC (Key Generation Centre)

Server Key

Service request

Cloud User as Data Owner

Cloud User as Data Consumer

Owner Registration and Mutual Authentication

User Registration and Mutual Authentication

Key Request to Encrypt File

Upload File

File Download

Fingerprint Biometric

Session key generation

User Fingerprint DataRepository

Access private key

## FIGURE 5.1

## 5.2 MODULES DESIGN SPECIFICATION

* Cloud Owner and User Registration & Key Generation
* Remote Server Authentication between Devices
* Cloud Owner Upload File to Remote Cloud Server
* Cloud User Decrypt and Download file

### 5.2.1 Cloud Owner and User Registration & Key Generation

Initially user need to create a account in the cloud sever by providing all the necessary information related to the user like email id, mobile number, age name, password and Cloud Owner a biometric ﬁngerprint then server will generate a unique identity for that Owner and User. Once Owner And User successfully register then user request to Key Generation centre to provide the keys to user by providing the unique id that was generated by the server. User can access the Cloud data only if he registers their name in key generation centre.

### 5.2.2 Cloud Owner Upload File to Cloud Server

User can able to upload any file in the cloud in order to provide security to the file user need to encrypted the file in the cloud .Cloud contain the only the encrypted format information about the file and corresponding parameter related to the file. In order to maintain security user initially generated the parameter related to the file and along with the mode to upload the data mode as public or private if private then user need to specify the client or mention group of people who can access or download the file then based on these parameter RSA will generate a key and sent to the user based in these key user would encrypt the file and upload to server.

### 5.2.3 Decrypt and Download file:

Then User would search the file based on the parameter if any parameter match in the server, then the server will load the entire related file to that parameter. Before download any file from the sever the user would also need to mutual authentication to the server, if user ﬁngerprint authenticated to the server then he access the file as file is in encrypted format user need to decrypt the file to decrypt the file user request to KGC (Key Generation Centre) for key along with the file parameter and user attribute then user can able to download the file.

## 5.3 ALGORITHMS

* RSA algorithm
* Base 64
* Diffie Hellman algorithm
* **CHAPTER 6**

**SYSTEM IMPLEMENTATION**

* **CLOUD OWNER CONTROLLER:**

package com.gts.controller;

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

import java.util.Map;

import java.util.Random;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.PropertySource;

import org.springframework.core.env.Environment;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.multipart.MultipartFile;

import org.springframework.web.servlet.ModelAndView;

import com.Demo.EncryptorAesGcmPassword;

import com.gts.model.Owner;

import com.gts.model.OwnerFileUpdatePojo;

import com.gts.service.OwnerDaoImp;

import com.gts.service.OwnerFileDao;

@Controller

@PropertySource("classpath:path.properties")

public class CloudOwnerController {

@Autowired

private Environment environment;

@Autowired

private OwnerDaoImp ownerDaoImp;

@Autowired

private OwnerFileDao ownerFileDaoImp;

@RequestMapping(value = "/index")

public ModelAndView test(HttpServletResponse response) throws IOException {

return new ModelAndView("index");

}

//-----------------------------------------------------Register--------------------------------------------------------

@RequestMapping(value = "register")

public ModelAndView testa(HttpServletResponse response) throws IOException {

return new ModelAndView("ownerReg");

}

@RequestMapping(value = "save\_owner", method = RequestMethod.POST)

public ModelAndView OwnerRegister(ModelAndView mv, @RequestParam("name") String name,

@RequestParam("email") String email, @RequestParam("password") String password,

@RequestParam("address") String address, @RequestParam("mb") String mobile, HttpSession session) {

session.setAttribute("sname", name);

session.setAttribute("semail", email);

session.setAttribute("mobile", mobile);

session.setAttribute("spass", password);

session.setAttribute("saddress", address);

// String path = session.getServletContext().getRealPath("/");

// String filaName = file.getOriginalFilename();

// System.out.println(path + " ::------- image file update----:::" + filaName);

// session.setAttribute("simage", path);

// try {

// byte barr[] = file.getBytes();

// BufferedOutputStream bout = new BufferedOutputStream(new FileOutputStream(path + "/" + email + ".jpg"));

// Object o = bout;

// bout.write(barr);

// bout.flush();

// bout.close();

//

// } catch (Exception e) {

// e.printStackTrace();

// }

HashSet<String> ha = new HashSet<String>();

List<Owner> lis = ownerDaoImp.getOwnerdate();

for (Owner owner : lis) {

ha.add(owner.getEmail());

}

boolean result = ha.contains(email);

if (result == true) {

mv.setViewName("ownerReg");

} else {

mv.setViewName("AddBio");

}

return mv;

}

@RequestMapping(value = "OwnerLogin")

public ModelAndView login(HttpServletResponse response) throws IOException {

return new ModelAndView("OwnerLogin");

}

@RequestMapping(value = "/loginProcess", method = RequestMethod.POST)

public ModelAndView loginProcess(HttpServletRequest request, HttpServletResponse response,

@ModelAttribute("login") Owner login, HttpSession session) throws IOException {

ModelAndView mav = null;

Owner user = ownerDaoImp.validateUser(login);

if (null != user) {

mav = new ModelAndView("LoginAddBio");

mav.addObject("email", user.getOwnerId());

session.setAttribute("finger", user.getFingerPrint());

session.setAttribute("sesOwnerName", user.getName());

session.setAttribute("sesOwnerId", user.getOwnerId());

session.setAttribute("OwneremailId", user.getEmail());

} else {

mav = new ModelAndView("OwnerLogin");

mav.addObject("msg", "Username or Password is wrong!!");

}

return mav;

}

@RequestMapping(value = "/OwnerReg", method = RequestMethod.GET)

public ModelAndView RegisterProcess(ModelAndView mv, HttpServletRequest request, HttpServletResponse response,

@RequestParam("txttemplate") String fig, HttpSession session) throws IOException {

Random rand = new Random();

int rand\_int1 = rand.nextInt(1000);

String status = "pending";

String ownerId = "owid" + Integer.toString(rand\_int1);

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

String name = (String) session.getAttribute("sname");

String email = (String) session.getAttribute("semail");

String phone = (String) session.getAttribute("mobile");

String password = (String) session.getAttribute("spass");

String address = (String) session.getAttribute("saddress");

Owner owner = new Owner();

owner.setName(name);

owner.setEmail(email);

owner.setPhone(phone);

owner.setPass(password);

owner.setAddres(address);

owner.setOwnerId(ownerId);

owner.setStatus(status);

owner.setFingerPrint(fig);

List<Owner> lis = ownerDaoImp.getOwnerdate();

String msg = "";

boolean count = ownerDaoImp.saveOwner(owner);

msg = "Registration Successfull";

mv.addObject("msg", msg);

mv.setViewName("OwnerLogin");

return mv;

}

@RequestMapping(value = "/logout", method = RequestMethod.GET)

public ModelAndView Logout(ModelAndView mv, HttpServletRequest request, HttpServletResponse servletResponse,

HttpSession session) throws IOException {

session = request.getSession(false);

session.invalidate();

mv.setViewName("index");

return mv;

}

@RequestMapping(value = "/LoginSucess", method = RequestMethod.POST)

public ModelAndView loginsucess(HttpServletRequest request, HttpServletResponse response, HttpSession session,

ModelAndView mv) throws IOException {

String oemail = (String) session.getAttribute("OwneremailId");

System.out.println("================>>>>>>" + oemail);

List<Owner> lis = ownerDaoImp.getOwnerdate();

Map<String, String> map = new HashMap<String, String>();

for (Owner owner : lis) {

map.put(owner.getEmail(), owner.getStatus());

}

String status = map.get(oemail);

if (status.equalsIgnoreCase("pending")) {

System.out.println("Contain Value is Match");

mv.addObject("msg", "Process Is Pending Pleace Wait");

mv = new ModelAndView("OwnerLogin");

} else {

System.out.println("Contain Value is not Match");

mv = new ModelAndView("ownerupdatefile");

}

return mv;

}

@RequestMapping(value = "/uploadFile", method = RequestMethod.POST)

public ModelAndView uploadFileHandler(@RequestParam("fname") String name, @RequestParam("tb") String privatekey,

@RequestParam("file") MultipartFile file, ModelAndView mv, HttpServletRequest request,

HttpServletResponse response) throws IOException {

String msg = null;

if (!file.isEmpty()) {

try {

byte[] bytes = file.getBytes();

// Creating the directory to store file

// String rootPath = System.getProperty("catalina.home");

// File dir = new File(rootPath + File.separator + "tmpFiles");

File dir = new File(environment.getRequiredProperty("locationpath"));

if (!dir.exists())

dir.mkdirs();

// Create the file on server

File serverFile = new File(dir.getAbsolutePath() + File.separator + name + ".txt");

BufferedOutputStream stream = new BufferedOutputStream(new FileOutputStream(serverFile));

stream.write(bytes);

stream.close();

System.out.println(name);

File file1 = new File(environment.getRequiredProperty("locationpath") + name + ".txt"); // creates a new

// file

// instance

FileReader fr = new FileReader(file1); // reads the file

BufferedReader br = new BufferedReader(fr); // creates a buffering character input stream

StringBuffer sb = new StringBuffer(); // constructs a string buffer with no characters

String line;

while ((line = br.readLine()) != null) {

sb.append(line); // appends line to string buffer

sb.append("\n"); // line feed

}

fr.close(); // closes the stream and release the resources

System.out.println("Contents of File: ");

System.out.println(sb.toString()); // returns a string that textually represents the object

String fileAsString = sb.toString();

EncryptorAesGcmPassword ss = new EncryptorAesGcmPassword();

String aa = ss.setEcy(fileAsString, privatekey);

// byte[] ENCname = Base64.getEncoder().encode(fileAsString.getBytes());

// byte[] decodedString = Base64.getDecoder().decode(new

// String(name).getBytes("UTF-8"));

System.out.println(aa);

// System.out.println(new String(decodedString));

//---------------------ECFILE write----------

FileWriter fw = new FileWriter(environment.getRequiredProperty("locationreport") + name);

fw.write(aa);

fw.close();

request.setAttribute("file", aa);

request.setAttribute("filename", name);

request.setAttribute("privatekey", privatekey);

// request.setAttribute("msg", msg);

msg = "File upload Sucess";

mv.setViewName("ownerupdatefile");

return mv;

} catch (Exception e) {

msg = "You failed to upload " + name + " => " + e.getMessage();

request.setAttribute("msg", msg);

mv.setViewName("ownerupdatefile");

request.setAttribute("filename", name);

return mv;

}

} else {

msg = "You failed to upload " + name + " because the file was empty";

request.setAttribute("msg", msg);

request.setAttribute("filename", name);

mv.setViewName("ownerupdatefile");

return mv;

}

}

@RequestMapping(value = "/deleteowner", method = RequestMethod.GET)

public ModelAndView UserDalete(ModelAndView mv, HttpServletRequest request, HttpServletResponse response,

Model model, @RequestParam("customerId") String customerId) throws IOException {

int result = ownerDaoImp.onDelete(customerId);

model.addAttribute("ownerlist", ownerDaoImp.getOwnerdate());

mv.setViewName("ShowInfo");

return mv;

}

@RequestMapping(value = "/ownerfileupdate", method = RequestMethod.POST)

public ModelAndView OwnerFileUpdate(ModelAndView mv, HttpServletRequest request, HttpServletResponse response,

Model model, @RequestParam("fileOwnerId") String ownerid,

@RequestParam("fileOwnerName") String fileOwnerName, @RequestParam("filename") String filename,

@RequestParam("eyfile") String eyfile, @RequestParam("pry") String pry) throws IOException {

System.out.println(ownerid + "" + fileOwnerName + "" + filename + "------'''''" + pry);

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");

LocalDateTime now = LocalDateTime.now();

String date = dtf.format(now).toString();

OwnerFileUpdatePojo ofdi = new OwnerFileUpdatePojo();

ofdi.setDate\_time(date);

ofdi.setFileName(filename);

ofdi.setFileOwnerName(fileOwnerName);

ofdi.setPrivateKey(pry);

ofdi.setFileOwnerId(ownerid);

ofdi.setStatus("Pending");

ofdi.setSessionKey(eyfile);

ofdi.setSt("upload");

ofdi.setAz("no");

boolean result = ownerFileDaoImp.saveOwner(ofdi);

request.setAttribute("msg", "Uplode File In Cloud");

mv.setViewName("ownerupdatefile");

return mv;

}

@RequestMapping(value = "/OwnerShowUpdateFile", method = RequestMethod.GET)

public ModelAndView OwnerShowUpdateFile(ModelAndView mv, HttpServletRequest request, HttpServletResponse response,

Model model, HttpSession session) throws IOException {

String ownerId = (String) session.getAttribute("sesOwnerId");

System.out.println("-------------->>>>>>" + ownerId);

// List<OwnerFileUpdatePojo> list = ownerFileDaoImp.getFileDetails();

model.addAttribute("userfileinfo", ownerFileDaoImp.getFileDetails());

model.addAttribute("userfilerequest", ownerFileDaoImp.getRqFiles());

request.setAttribute("msg", "Suecss");

mv.setViewName("ownerviewRq");

return mv;

}

@PostMapping("/KeySharing")

public ModelAndView KeySharing(ModelAndView mv, HttpServletRequest request, HttpServletResponse response,

Model model, HttpSession session, @RequestParam("Id") String id, @RequestParam("key") String key)

throws IOException {

System.out.println(id + "=-------------------->>>" + key);

ownerDaoImp.onkeyShareing(key, id);

model.addAttribute("userfileinfo", ownerFileDaoImp.getFileDetails());

model.addAttribute("userfilerequest", ownerFileDaoImp.getRqFiles());

request.setAttribute("msg", "Suecss");

mv.setViewName("ownerviewRq");

return mv;

}

}

**CHAPTER 7**

# 7. SYSTEM TESTING

## 7.1 Coding

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

## 7.2 Coding Standards

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

* Program should be simple, clear and easy to understand.
* Naming conventions
* Value conventions
* Script and comment procedure
* Message box format
* Exception and error handling

### 7.2.1 Naming Conventions

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

**Class names :**

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

**Member Function and Data Member name :**

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

### 7.2.2 Value Conventions

Value conventions ensure values for variable at any point of time. This involves the following:

* Proper default values for the variables.
* Proper validation of values in the field.
* Proper documentation of flag values.

### 7.2.3 Script Writing and Commenting Standard

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

### 7.2.4 Message Box Format

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

* X – User has performed illegal operation.
* ! – Information to the user.

## 7.3 Test Procedure

7.3.1 System testing:

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

* Static analysis is used to investigate the structural properties of the Source code.
* Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

## 7.4 Test data and Output

### 7.4.1 Unit Testing :

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

### 7.4.2 Functional Tests

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

* Performance Test
* Stress Test
* Structure Test

**7.4.3 Performance Test**

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

**7.4.4 Stress Test**

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

### 7.4.5 Structured Test

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

* Exercise all logical decisions on their true or false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structures to assure their validity.
* Checking attributes for their correctness.
* Handling end of file condition, I/O errors, buffer problems and textual errors in output information

### 7.4.6 Integration Testing

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

## 7.5 Testing Techniques / Testing Stratergies

### 7.5.1 Testing :

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

**7.5.1.1 White box testing**

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* Flow graph notation
* Cyclometric complexity
* Deriving test cases

**7.5.1.2 Black box testing**

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Comparison testing

### 7.5.2 Software Testing Strategies:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

**7.5.2.1 Integration Testing:**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

**7.5.2.2 Program Testing:**

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

**7.5.2.3 Security Testing:**

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

**7.5.2.4 Validation Testing :**

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

\* The function or performance characteristics confirm to specifications and are accepted.

\* A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

**7.5.2.5 User acceptance testing**

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

* Input screen design.
* Output screen design.

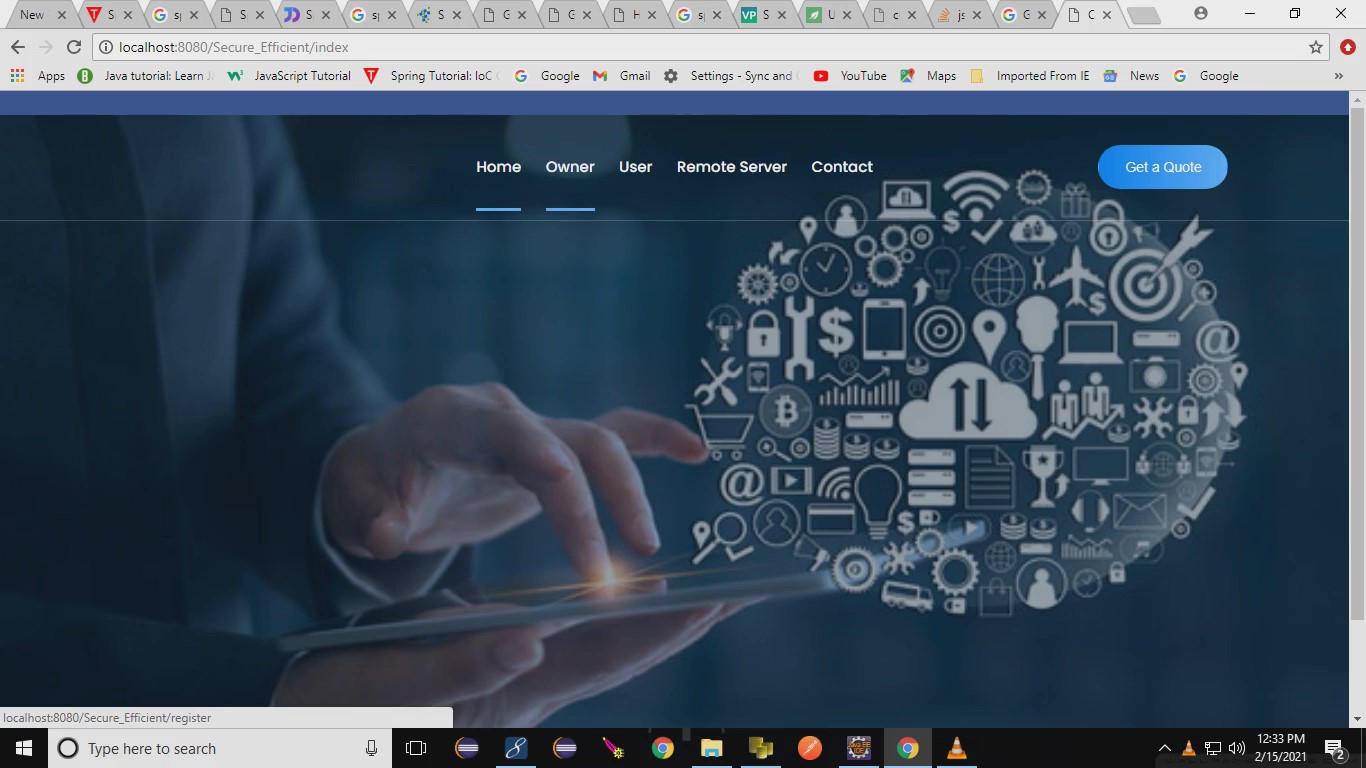
**CHAPTER 8**

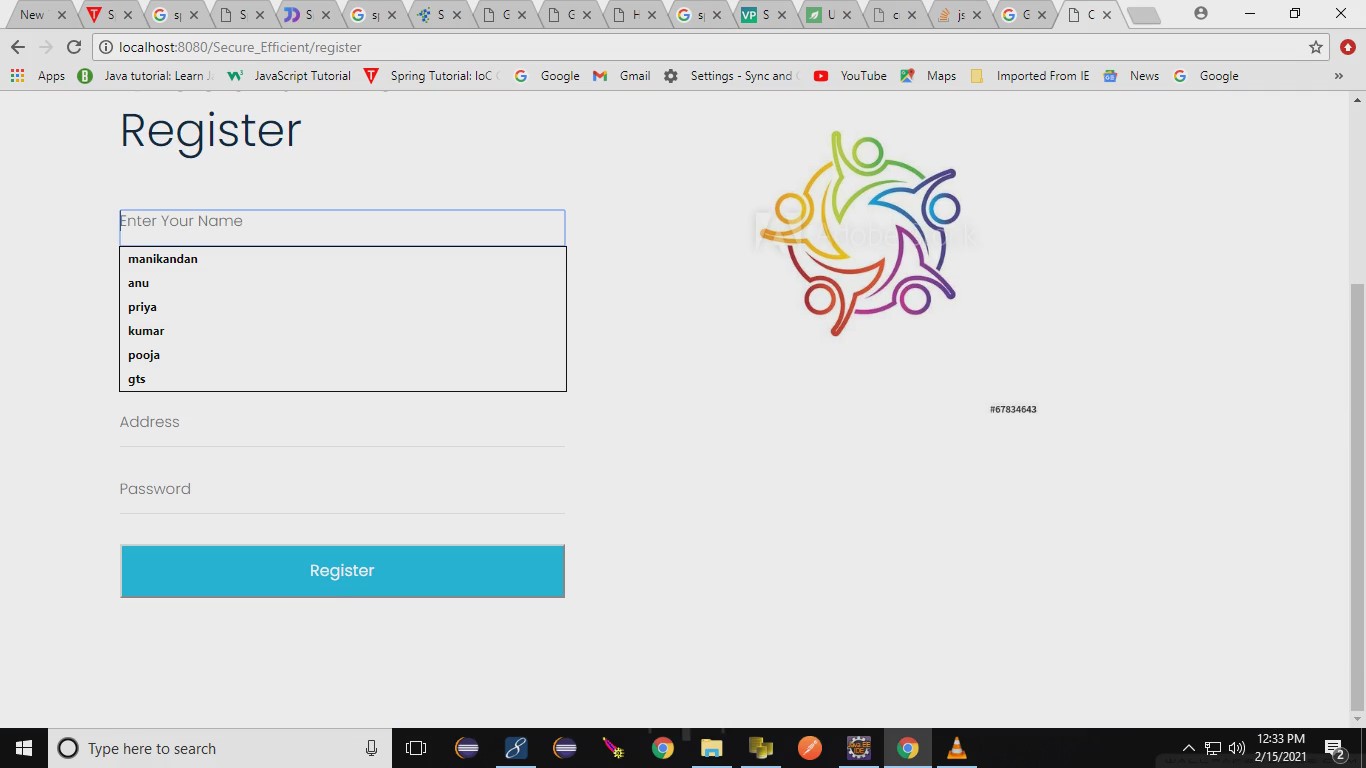
**CONCLUSION**

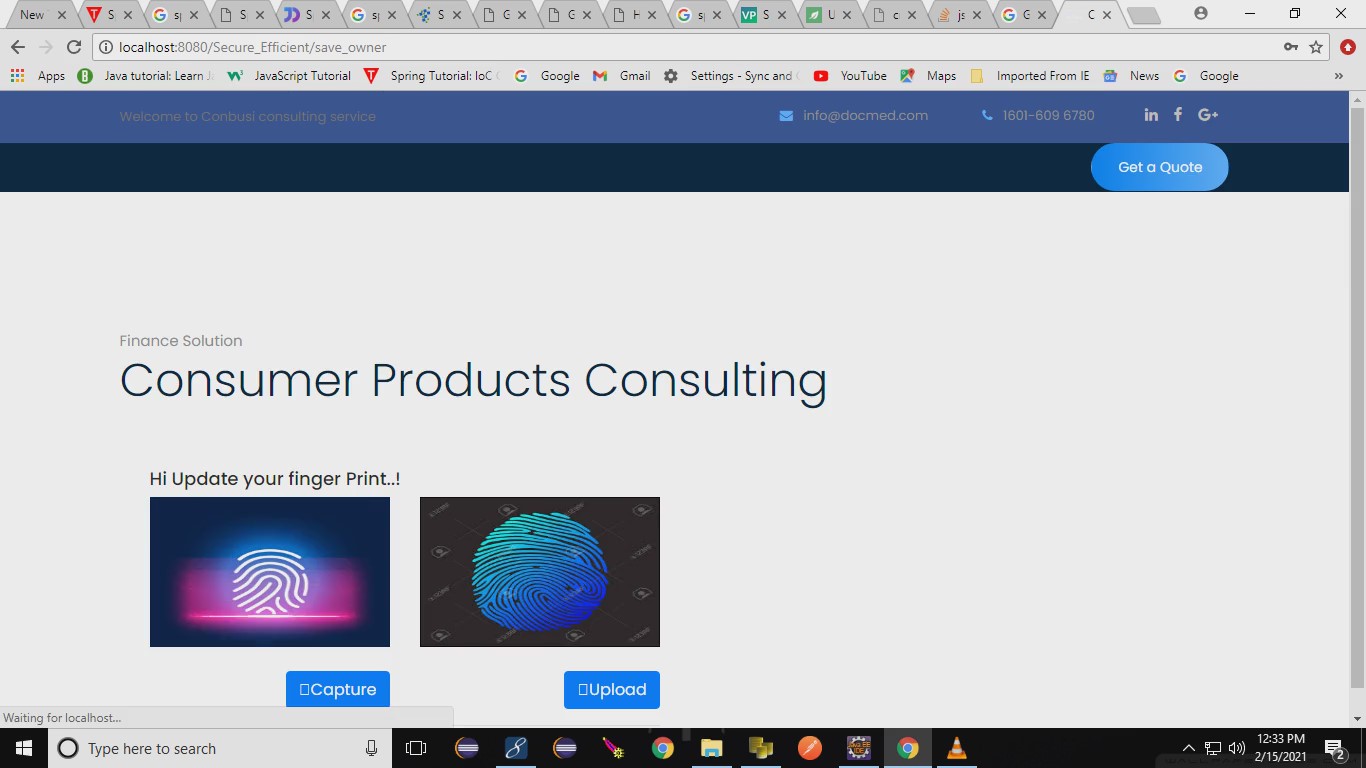
* 1. **Conclusion and Feature enhancement**

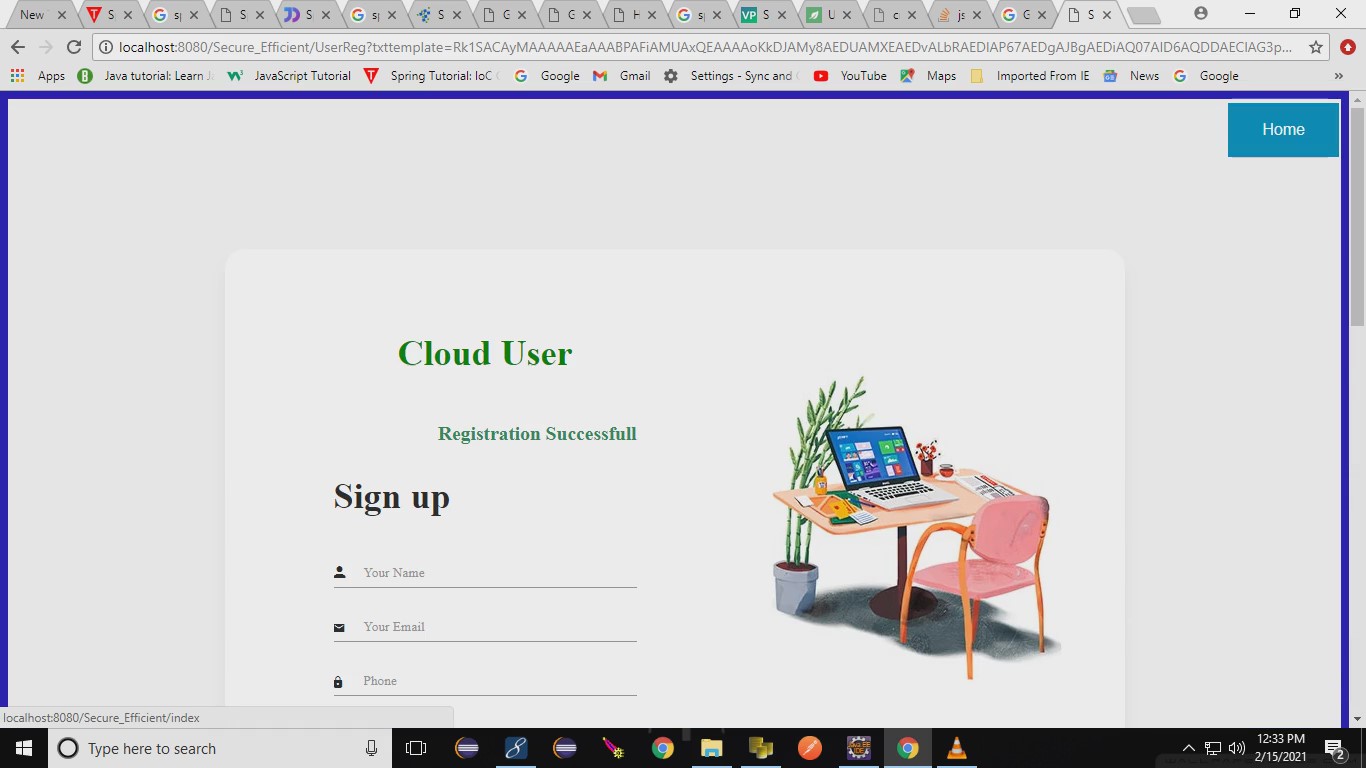
In this paper, we design a new biometric-based authentication protocol to provide secure access to a remote (cloud) server. In addition, we propose an efficient approach to generate a session key between two communicating parties using two biometric templates for a secure message transmission. In other words, there is no need to store the user’s private key anywhere and the session key is generated without sharing any prior information. The main aim of the project is used to derive a unique identity from the users biometric data which is further used to generate the user’s private key.

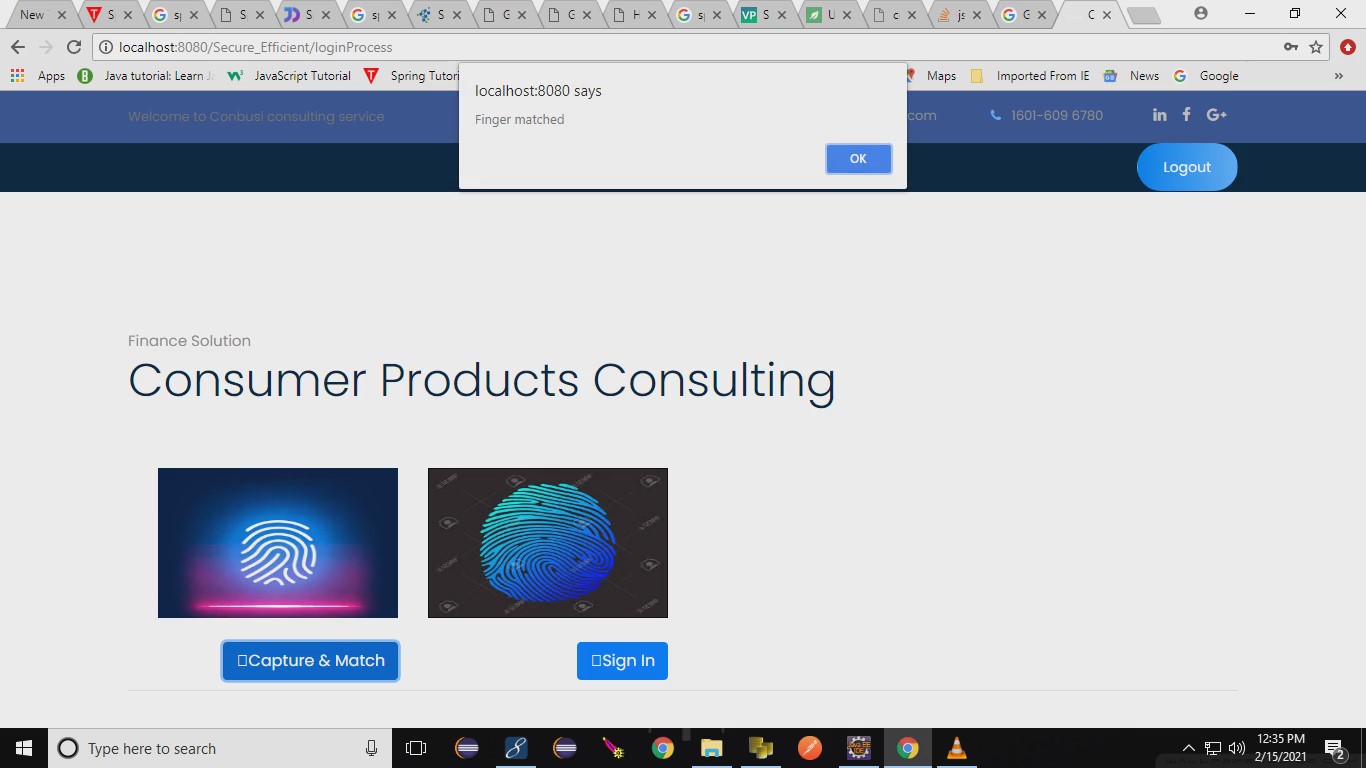
**SAMPLE SCREENS**

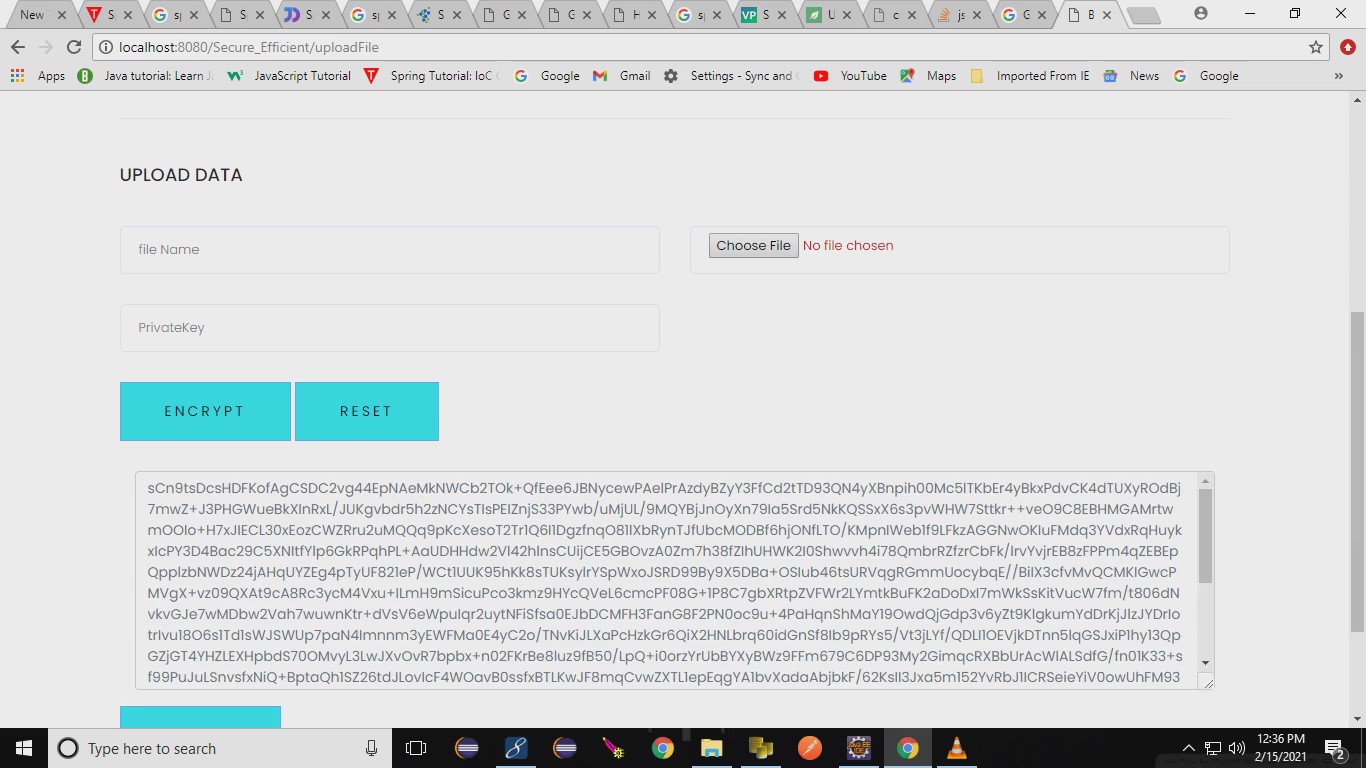


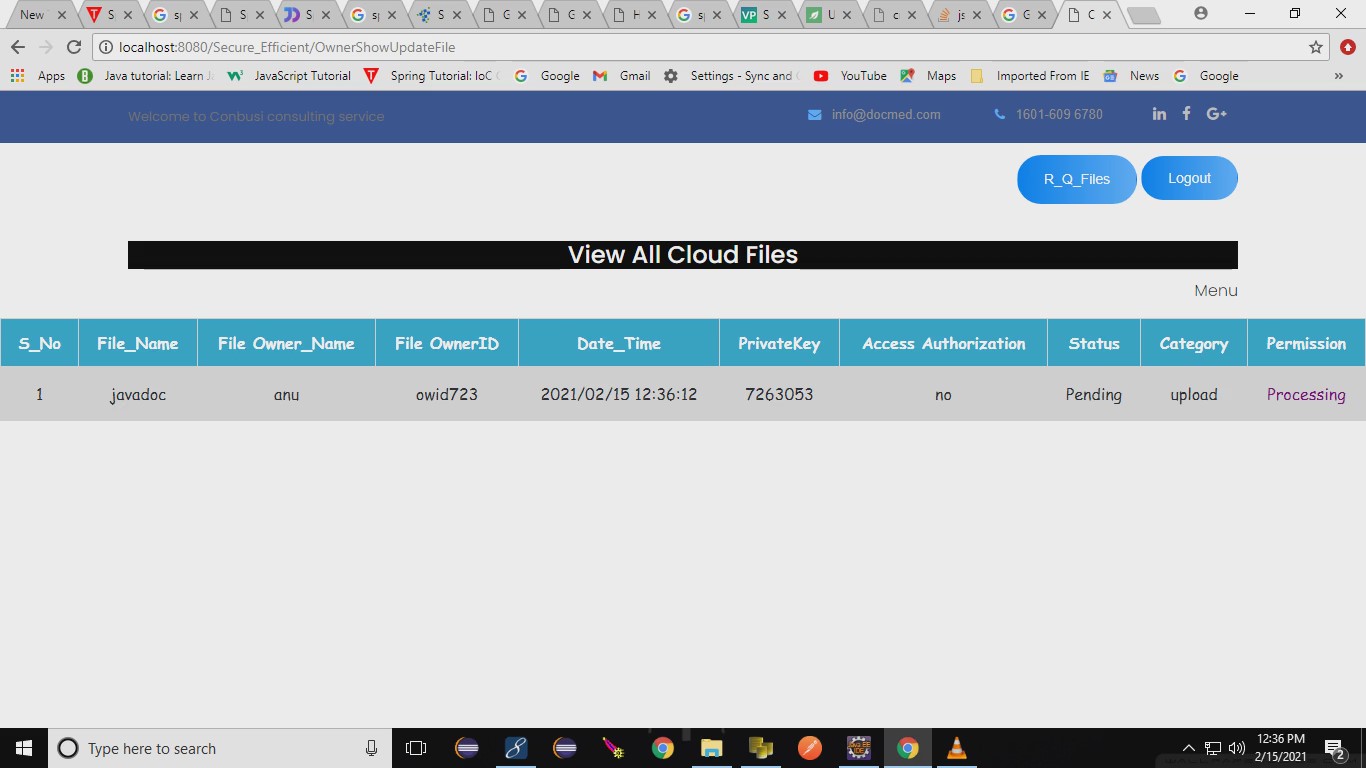


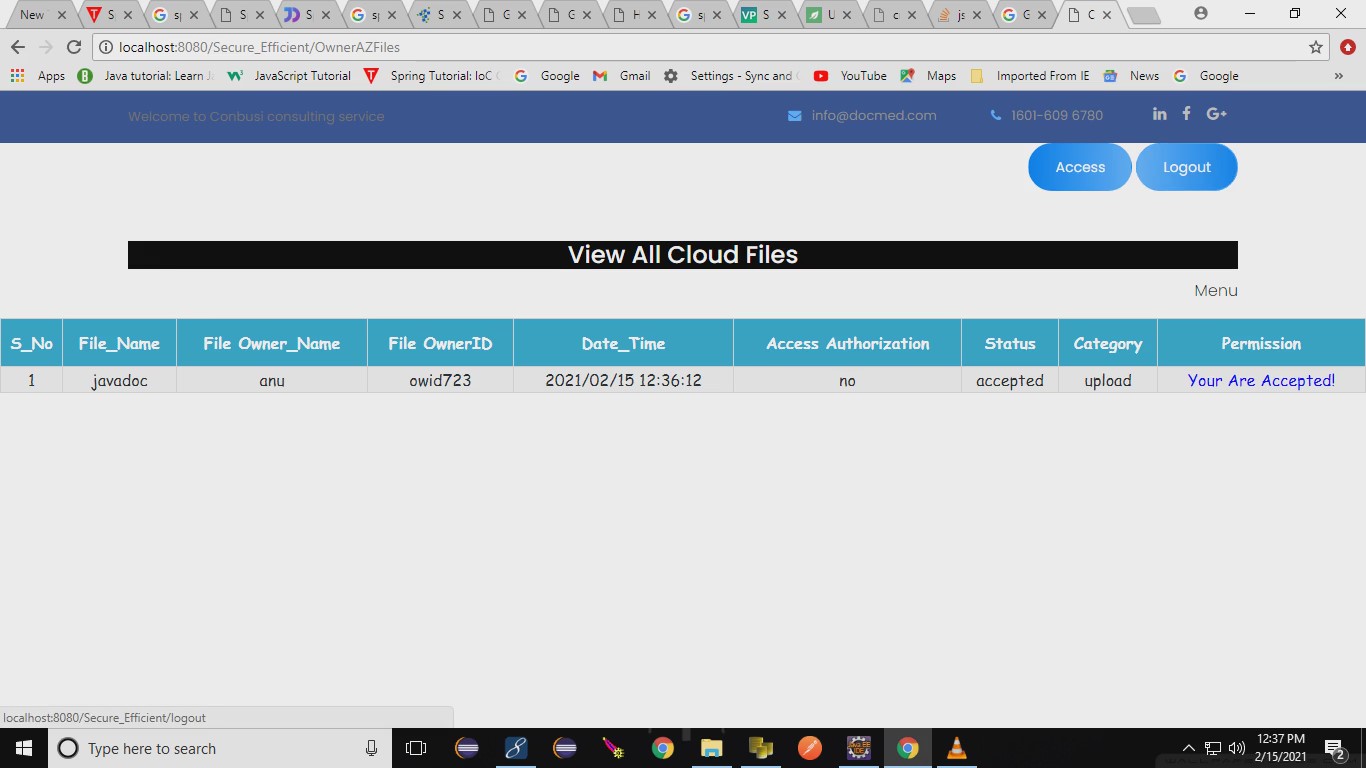


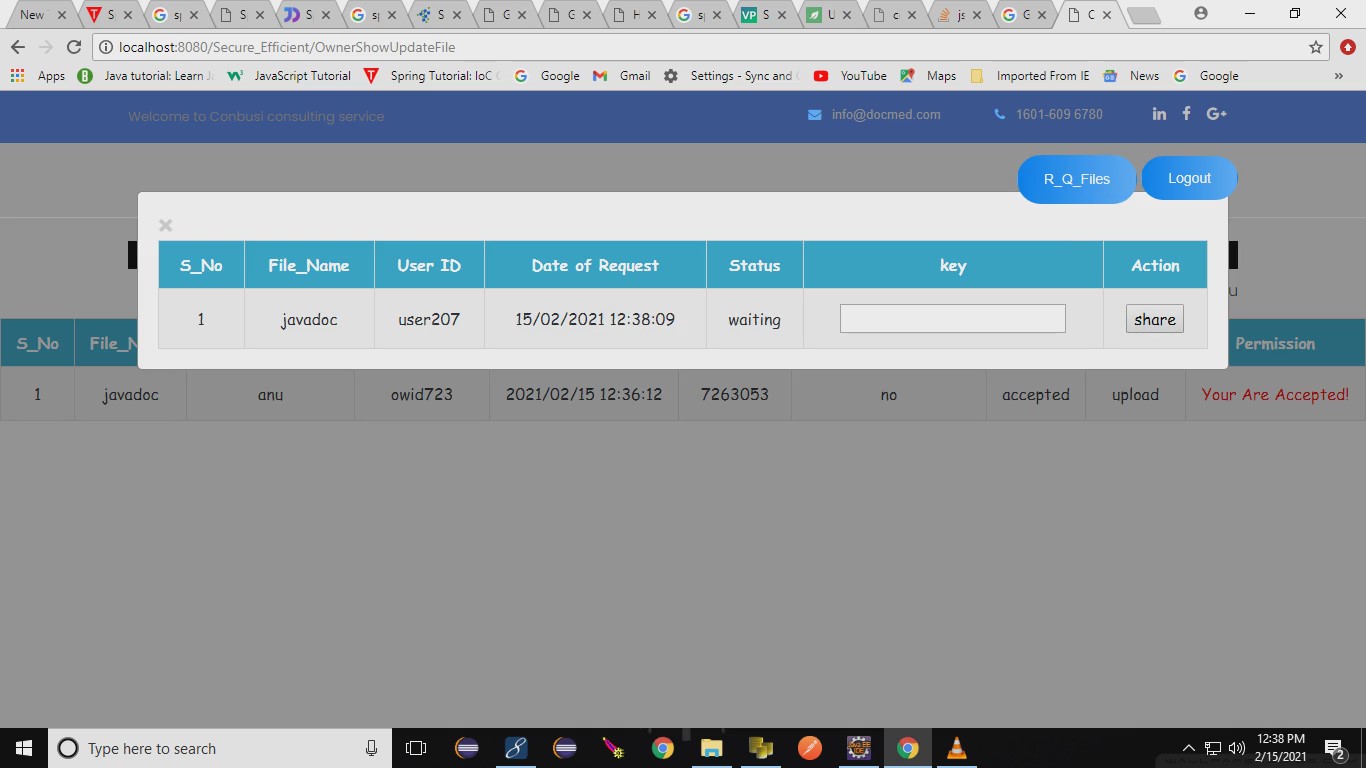


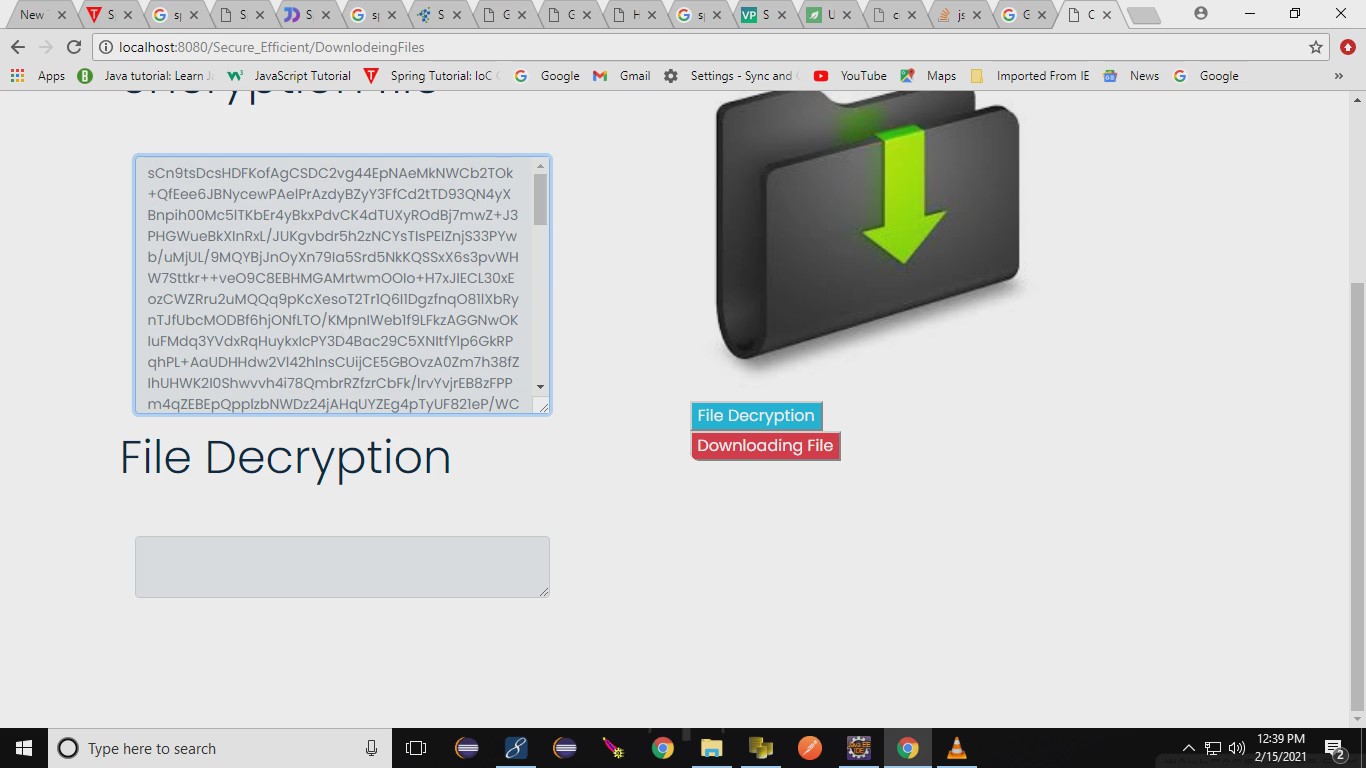












**REFERENCES**

1. Wenchang Yang, Song Wang,Jiankun Hu, Guanglou Zheng and Craig Valli ,”Security and Accuracy of Fingerprint-Based Biometrics: A Review” Received: 2 December 2018; Accepted: 23 January 2019; Published: 28 January 2019

2. Zhihua Xia, Xingming Sun, Neal N. Xiong ,”A Novel Weber Local Binary Descriptor for Fingerprint Liveness Detection” IEEE Transactions on Systems, Man, and Cybernetics: Systems · January 2018

3.Srinivas Jangirala, Mohammad Wazid,”Anonymous Lightweight Chaotic Map-Based Authenticated Key Agreement Protocol for Industrial Internet of Things” IEEE Transactions on Dependable and Secure Computing · July 2018

4. C. Yuan, X. Sun, and Q. M. J. Wu, “Difference co-occurrence matrix using BP neural network for fingerprint liveness detection,” Soft Computing, vol. 23, no. 13, pp. 5157–5169, 2019.

5.S. Roy, S. Chatterjee, A. K. Das, S. Chattopadhyay, S. Kumari, and M. Jo, “Chaotic Map-Based Anonymous User Authentication Scheme With User Biometrics and Fuzzy Extractor for Crowdsourcing Internet of Things,” IEEE Internet of Things Journal, vol. 5, no. 4, pp. 2884– 2895, Aug 2018**.**

6.Jain, A.K.; Flynn, P.; Ross, A.A. “Handbook of Biometrics”; Springer: New York, NY,

7.Riaz, N.; Riaz, N.; Riaz, A.; Riaz, A.; Khan, S.A.; Khan, S.A. “Biometric template security: An overview”. Sensor Rev. 2017, 38, 120–127.

1. Prabhakar, S.; Pankanti, S.; Jain, A.K. “Biometric recognition: Security and privacy concerns. IEEE Secure”. Priv. 2003, 1, 33–42.
2. Awad, A.I.; Hassanien, A.E. Impact of Some Biometric Modalities on Forensic Science. In “Computational Intelligence in Digital Forensics: Forensic Investigation and Applications”; Springer: Berlin, Germany, 2014; pp. 47–62.
3. Zheng, G.; Shankaran, R.; Orgun, M.A.; Qiao, L.; Saleem, K. “Ideas and challenges for securing wireless implantable medical devices:” A review. IEEE Sens. J. 2016, 17, 562–576.
4. Zheng, G.; Fang, G.; Shankaran, R.; Orgun, M.A.; Zhou, J.; Qiao, L.; Saleem, K.” Multiple ECG fiducial points-based random binary sequence generation for securing wireless body area networks.“IEEE J. Biomed. Health Inf. 2017, 21, 655–663.
5. T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng, “A Privacy Preserving Communication Protocol for IoT Applications in Smart Homes,” IEEE Internet of Things Journal, 2017, DOI: 10.1109/JIOT.2017.2707489.
6. S. Challa, M. Wazid, A. K. Das, N. Kumar, A. G. Reddy, E. J. Yoon, and K. Y. Yoo, “Secure Signature-Based Authenticated Key Establishment Scheme for Future IoT Applications,” IEEE Access, vol. 5, pp. 3028– 3043, 2017