VELS
INSTITUTE OF SCIENCE, TECHNOLOGY & ADVANCED STUDIES (VISTAS)
(DEEMED TO BE UNIVERSITY Estd. u/s 3 OF THE UGC ACT, 1956)
NAAC ACCREDITED
PALLAVARAM - CHENNAI - INDIA

SCHOOL OF COMPUTING SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

A FAST NEAREST NEIGHBOUR SEARCH SCHEME OVER OUTSOURCED ENCRYPTED MEDICAL IMAGES

Submitted by

M. MONICA (20139112)

V. SURIYA (20139126)

S. DINESH (20139106)

Under the guidance of

Dr. SOWMYA JAGADEESAN

A PROJECT REPORT

2022-2023

BONAFIDE CERTIFICATE

Certified that this project report "**A FAST NEAREST NEIGHBOUR SEARCH OVER OUTSOURCED ENCRYPTED MEDICAL IMAGES"** is the Bonafide

work of **M. MONICA (20139112), V. SURIYA (20139126), S. DINESH (20139106)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. SOWMYA JAGADEESAN	**Dr. P. SUJATHA**
Senior Faculty	**HEAD OF THE DEPARTMENT**
Department of Computer Application	**Department of Computer Application**
VISTAS, Chennai.	**VISTAS, Chennai.**

ACKNOWLEDGEMENT

We thank the Almighty GOD for the abundant blessings showered on us. We extend our deepest love and gratitude to our dear parents who built up our career and backed us up in life.

We feel thankful to the Head of the Department **Dr. P. Sujatha,** Department of Information Technology, VISTAS, for all her encouragement, which has sustained our efforts.

We express our deepest gratitude to the internal guide **Dr. Sowmya Jagadeesan,** Senior Faculty, Department of Information Technology, VISTAS, for her valuable guidance, ideas and support.

We extend our sincere thanks to the project coordinator **Dr. S.DHANARAJ** for his guidance and support.

We would like to thank all other faculty members of Department of Information Technology for their help and advice throughout our life in this campus.

Finally, we are thankful to all our friends and all others who encouraged us and helped us in doing this project.

TABLE OF CONTENT **Page No.**

ABSTRACT

Medical imaging is crucial for medical diagnosis, and the sensitive nature of medical images necessitates rigorous security and privacy solutions to be in place. In the paper, we propose a secure and efficient scheme to find the exact nearest neighbour over encrypted medical images. Unlike most existing schemes, our scheme can obtain the exact nearest neighbour rather than an approximate result. our proposed scheme obviously meets the security requirements. It protects the secrecy and privacy of data as well as the user's input query while simultaneously hiding data access patterns. Our scheme is designed to ensure the confidentiality of all related medical images. To ensure query privacy, the database and query need to be encrypted before sending to the cloud server. Proposed Homomorphic algorithm is used to encrypt data images. the best method is Advanced Encryption Standard method (AES). There are many types of AES that can be used but the most effective is AES-128. So, the aim of this study is to design image cryptographic application using the AES-128 method. Process of design applications with this method is through several stages, such as process of encryption, decryption, key generation and testing of the methods used. The attacks test is given by cropping, blurring, and enhancing the ciphertext image. To reduce the storage problem in Cloud we have split the image and file into different block and get stored, so storage problem get rectified . The proposed scheme needs to reduce the computation cost on the end-user as much as possible.

CHAPTER -1
SYSTEM STUDY

1.1 Feasibility Study

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

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

1.1.1 Economical Feasibility

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

1.1.2 Technical Feasibility

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

1.1.3 Social Feasibility

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

1.2 System Requirements

1.2.1 Hardware Requirements

System – Pentium IV 2.4 GHz.

Hard disk – 40 GB.

Floppy driver – 1.44 Mb.

Monitor – 15 VGA colour.

Mouse – DELL

RAM – 512 Mb.

1.2.2 Software Requirements

Operating System – Windows XP/7.

Coding Language – Java

Front End – Html/css.

Back End – J2se, J2ee.

Database – Mysql

Tools – NetBeans IDE 7.2.1

CHAPTER-2
SYSTEM ANALYSIS

2.1 Introduction

Cloud computing is becoming a norm in our society and in such a deployment, the data owner can outsource databases and management functionalities to the cloud server. The latter stores the databases and supplies access mechanisms to query and manage the outsourced database. This allows data owners to reduce data management expenses and improve quality of services, hospitals or related medical institutions can store patient medical images in a professional and secure database of a practical electronic healthcare system. Encrypting data by the data owner is a naïve method to ensure privacy, while it ensures the secrecy of the outsourced data from the cloud and unauthorised users. Additionally, to protect query privacy, permitted users should send their requests to the cloud foe evaluation after encryption. However, by analysing the data access patterns, the cloud (or a malicious insider) can derive private information about the real data items even though the data and queries are encrypted. The nearest neighbour search is a vital operation in data mining, machine learning, and information retrieval, and more recently the healthcare industry as well. Recently, due to the emergence of high-dimensional medical AES (Advanced Encryption Standard) is the development of the standard DES (Data Encryption Standard) encryption algorithm of which validity period deemed to be over due to security. The rapid computer speed was considered very dangerous to the DES. This research of encryption and decryption was carried out on an image by adding attack testing. From several attack tests that have been carried out on ciphertext to determine the resistance of the AES method, it was found that the determinant of the success or failure of the decryption process of the image file depends on the pixel value. When the pixel value of the encrypted image is changed, the decryption process have been successful, but it cannot restore the plaintext image.

2.2 Project Description

The "Fast Nearest Neighbour Search Scheme over Outsourced Encrypted Medical Images" project aims to develop a system that can securely and efficiently search for similar medical

images in a large database of encrypted medical images. The system will use AES-128 encryption to protect patient privacy and confidentiality.
The system will consist of several components, including an image cryptographic application, a fast nearest neighbour search algorithm, and a medical image database management system. Medical professionals will be able to access the system through a user interface and search for similar medical images based on their specific requirements.
The image cryptographic application will be responsible for encrypting and decrypting medical images using AES-128 encryption. The fast nearest neighbour search algorithm will efficiently and accurately search for similar medical images in the encrypted database, using advanced techniques such as indexing and data partitioning.
The medical image database management system will be responsible for storing and managing the encrypted medical images in a secure and efficient manner. The system will also include security features such as user authentication and authorization protocols, to ensure that only authorized personnel can access the system and medical image database.

2.3 Modules

2.3.1 Design Goals

Our scheme achieves semantic security under quantifiable leakage functions. In other words, we can formally define the views of the cloud server in stateful leakage functions. Four entities in our scheme, namely: a data owner (e.g., hospital), multiple data users (e.g., doctors in different regions), and two semi-honest cloud servers. Note that data users are authorised by the data owner. The data owner Alice owns a medical image database M, which she would like to outsource to the cloud server. Due to the sensitivity and privacy of medical images, Alice encrypts these images as well as computing and encrypting the mean and standard deviation for each image in the database. In order to process the query, Alice outsources the encrypted calculated results and database to the cloud server. A legitimate data user Bob can upload a medical image to the cloud server and query the exact nearest neighbour. To ensure the privacy of the query, Bob encrypts Q with the Paillier public key pk. Additionally, Bob computes the mean and standard deviation of Q, then he sends his encrypted query and related information after encrypting d.

2.3.2 Security Guarantee

The owner Alice outsources the encrypted image database E_{pk}(M)to C1 and sends the secret key of the Paillier cryptosystem to C2 . The intent of our scheme is to efficiently and securely retrieve the record which is closest to the query in the encrypted database. In brief, now a legitimate user wants to search out the exact nearest neighbour of his query record Q <q_1,…..,q_d>based on the encrypted medical image database E_{pk}(M) stored in C1 . Firstly, the user sends his encrypted query information to C1. After this, C1 and C2 utilize the above three protocols to securely search the nearest neighbour of the input query Q. In the end, only the user will know the exact nearest neighbour to Q. The key steps of our algorithm. user Bob wants to search for an image that is the most similar to his query Q based on E_{pk}(M)in C1. First of all, Bob computes the mean and standard deviation of Q, denoted by μ_q and σ_q. Then he encrypts μ_q and σ_q with the Paillier public key, expressed as Epk(μ_q) and E_{pk} (σ_q). After that, Bob encrypts the query image Q as E_{pk}(Q)and sends Epk(μ_q) , E_{pk} (σ_q). and E_{pk}(Q) to the cloud server C1. After C1 receives the query request and related information from Bob, the cloud servers will execute the query process.

2.3.3 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is a cryptographic algorithm that can be used rightly to secure data. This AES algorithm works on data blocks in the form of 4 x 4 matrix. Symmetrical ciphertext blocks can encrypt (encipher) and decrypt (decipher) information. AES algorithm uses clicking the cryptographic keys 128, 192, and 256 bits to encrypt and decrypt the data. Therefore, this algorithm is known as AES-128, AES-192, and AES-256. This algorithm also has another name that is Rijndael algorithm. It is because this algorithm was made by Rijndael, which is combined from Vincent Rijmen dan John Daemen. AES (Advanced Encryption Standard) is the development of the standard DES (Data Encryption Standard) encryption algorithm of which validity period deemed to be over due to security. the process stages of this algorithm, there are 3 main processes, namely encryption, decryption, and key expansion.

2.3.4 Scalability

When the encrypted database is outsourced to the cloud servers, secure query protocols and algorithms need to preserve the secrecy and privacy of the outsourced database as well as the user's query at all times. At the same time, data access patterns should be hidden from the cloud. In our scheme, because of the encryption of query Q and the semantic security When the encrypted database is outsourced to the cloud servers, secure query protocols and algorithms need to preserve the secrecy and privacy of the outsourced database as well as the user’s query at all times. At the same time, data access patterns should be hidden from the cloud. In our scheme, because of the encryption of query Q and the semantic security. Our proposed scheme has good scalability performance. And the dynamic changes to the database have almost no impact on our algorithm.

2.3.5 Block Storage

Block storage chops data into blocks get it and stores them as separate pieces. Each block of data is given a unique identifier, which allows a storage system to place the smaller pieces of data wherever is most convenient. Block storage is often configured to decouple the data from the user’s environment and spread it across multiple environments that can better serve the data. And then, when data is requested, the underlying storage software reassembles the blocks of data from these environments and presents them back to the user.

2.4 Waterfall Model

The Waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement.

In this article, we will discuss the advantages and disadvantages of the waterfall, should we avoid it? when to use it? and the waterfall model pitfall, and why I see it as the father of the SDLC models.

2.4.1 Waterfall model phases

Waterfall Model contains the main phases similarly to other process models, you can read this article for more information about phases definitions.

When to use Waterfall Model?

Due to the nature of the waterfall model, it is hard to get back to the previous phase once completed. Although, this is can be very rigid in some software projects which need some flexibility, while, this model can be essential or the most suitable model for other software projects' contexts.

The usage of the waterfall model can fall under the projects which do not focus on changing the requirements, for example:

1. Projects initiated from a request for proposal (RFP), the customer has a very clear documented requirements
2. Mission Critical projects, for example, in a Space shuttle
3. Embedded systems.

We can notice some similarities of these types of projects that they cannot be delivered in iterative, incremental, or agile manner, for example, in embedded systems for the elevator, you cannot deliver an elevator who can go up only without going down, or handling only users requests from inside and ignore outside calls for the elevator.

2.4.2 Validation and Verification Model –V-Model

V-Model is mostly known as the validation and verification software development process model (The Vee Model), and It is one of the most know software development methodology. Although it is considered as an improvement to the waterfall model and it has some similarities as the process also based on sequential steps moving down in a linear way, it differs from the waterfall model as the steps move upwards after the coding phase to form the typical V shape. This V shape demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.

2.4.3 The V-Model

This means that any phase in the development process begins only if the previous phase is complete and has a correspondence related testing phase which is performed against this phase completion. Similar to the Waterfall model, the V-Model does not define the process to go back to the previous phase to handle changes in requirement.

The technical aspect of the project cycle is considered as a V shape starting with the business needs on the upper left and ending with the user acceptance testing on the upper right.

2.4.4 V-Model Model Phases

The primary phases of the V-Model Model are comparable to those of other process models; for more information on SDLC phases definitions, see this article. Furthermore, it breaks down the testing phase into specific processes to guarantee that the validation and verification procedure is carried out properly. As a result, the following testing steps can be included:

1.Unit Testing:

Unit testing is code-level testing that aids in the early detection of faults. The developer is primarily responsible for unit testing his code, albeit not all defects may be identified by unit testing.

2.Functinal Testing:

Functional testing is linked to the low-level design phase, and it guarantees that a group of codes and units are functioning together to provide a new function or service.

3.Integration Testing:

The high-level design phase is linked to integration testing. After any new functionality or changes, integration testing verifies that all system parts are working together.

2.5 Existing System

There are a number of efficient schemes to find the approximate nearest neighbour, designed to improve efficiency in space and time at the cost of accuracy. The scheme based on Locality Sensitive Hashing (LSH) is a well-known and effective method that solves the nearest neighbour query problem in high-dimensional space. The LSH scheme [20] embeds data in low-dimensional subspaces and utilizes hash tables to improve efficiency. e most existing algorithms, this method can obtain the exact nearest neighbour rather than an approximate one. The beauty of this method is simplicity, in the sense of simple pre-processing without involving complex data structures. The g techniques have limitations and are not applicable to the healthcare industry's medical imaging problem. These schemes, which are based on special data structures can reduce the search time cost because of the tree-structured organization of data. However, such data structures require large pre-process time and memory space, and so they are not appropriate for high-dimensional and large-scale data.

2.4.1 Drawbacks of Existing System

- ✓ The need for accuracy in the healthcare industry, these LSH-based schemes are not suitable for solving the medical images problems.
- ✓ The existing techniques have limitations and are not applicable to the healthcare industry's medical imaging problem.
- ✓ Image overloaded problem because all images are stored in same location**.**

2.5 Proposed system

We propose a secure and efficient scheme to find the exact nearest neighbour over encrypted medical images. Instead of calculating the Euclidean distance, we reject candidates by computing the lower bound of Euclidean distance that is related to the mean and standard deviation of data. proposed an efficient scheme for k-nearest neighbour search, which enhances the protection of the decryption key and reduces the burden on data owners. t the algorithm proposed by Ahn et al. can simultaneously ensure both efficiency and accuracy. Based on this algorithm, we present an efficient scheme. our proposed scheme obviously meets the security requirements. It protects the secrecy and privacy of data as well as the user's input query while simultaneously hiding data access patterns. the best method is Advanced Encryption Standard method (AES). There are many types of AES that can be used but the most effective is AES-128. So, the aim of this study is to design image cryptographic application using the AES-128 method. it was found that this method is resistant to cropping attacks, but not resistant to blurring and enhancement attacks. Improve the efficiency if image storage function.

2.5.1 Advantage of Proposed System

- ✓ Our scheme is designed to ensure the confidentiality of all related medical images. To ensure query privacy, the database and query need to be encrypted before sending to the cloud server.

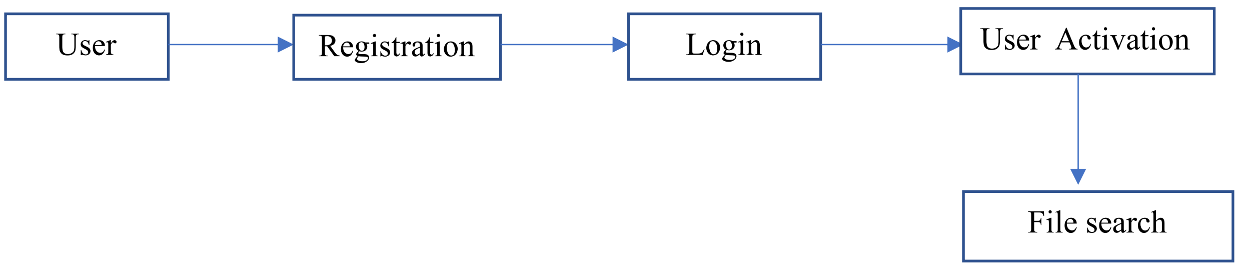
- ✓ Using this algorithm, many data points are eliminated in constant time rather than linear time. When the elements are high-dimensional data, the computational cost reduction will be significant
- ✓ Advanced Encryption Standard (AES) is a cryptographic algorithm that can be used rightly to secure data.
- ✓ AES (Advanced Encryption Standard) is the development of the standard DES (Data Encryption Standard) encryption algorithm of which validity period deemed to be over due to security.

CHAPTER - 3

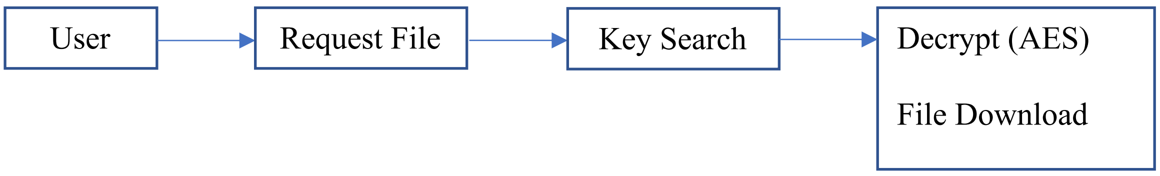
SYSTEM DESIGN

3.1 Data flow Diagram

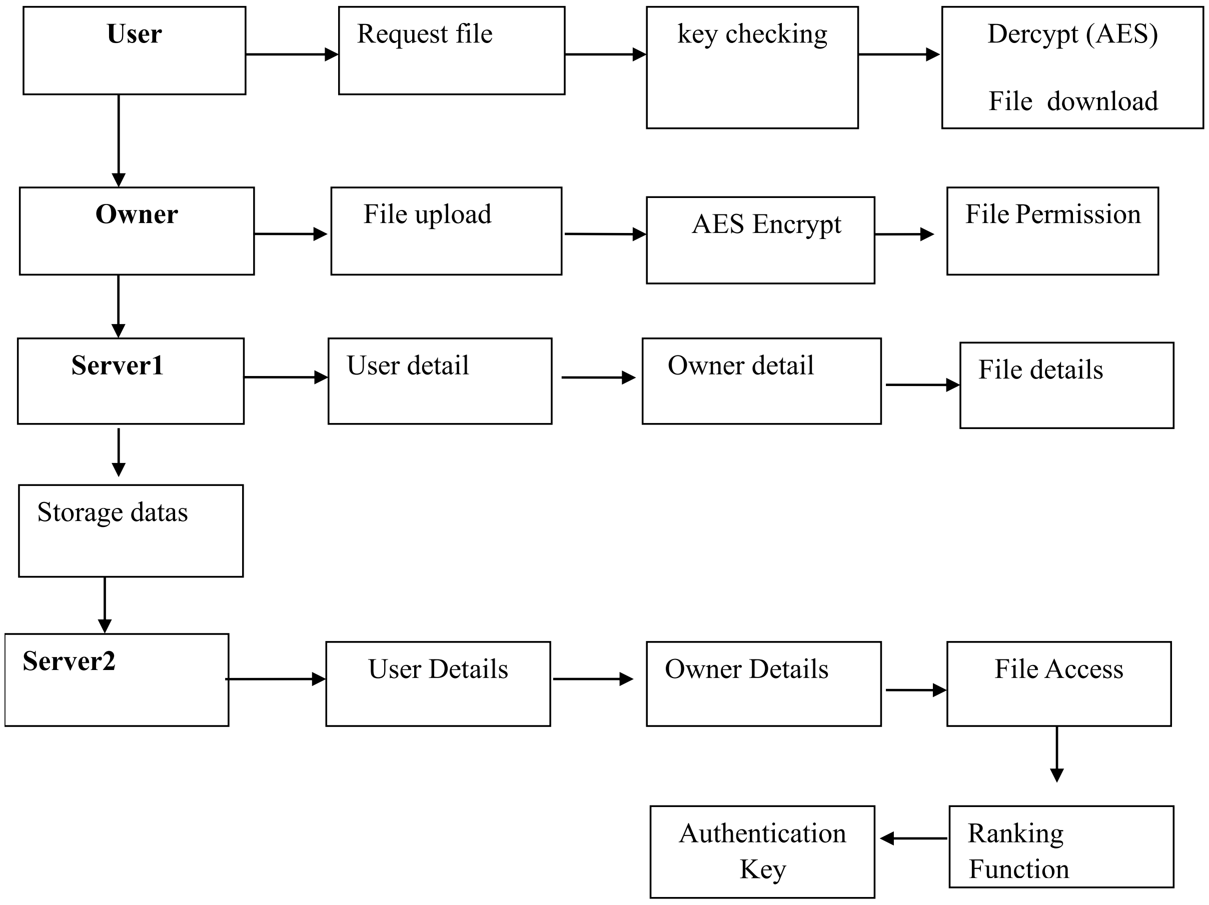
LEVEL-0



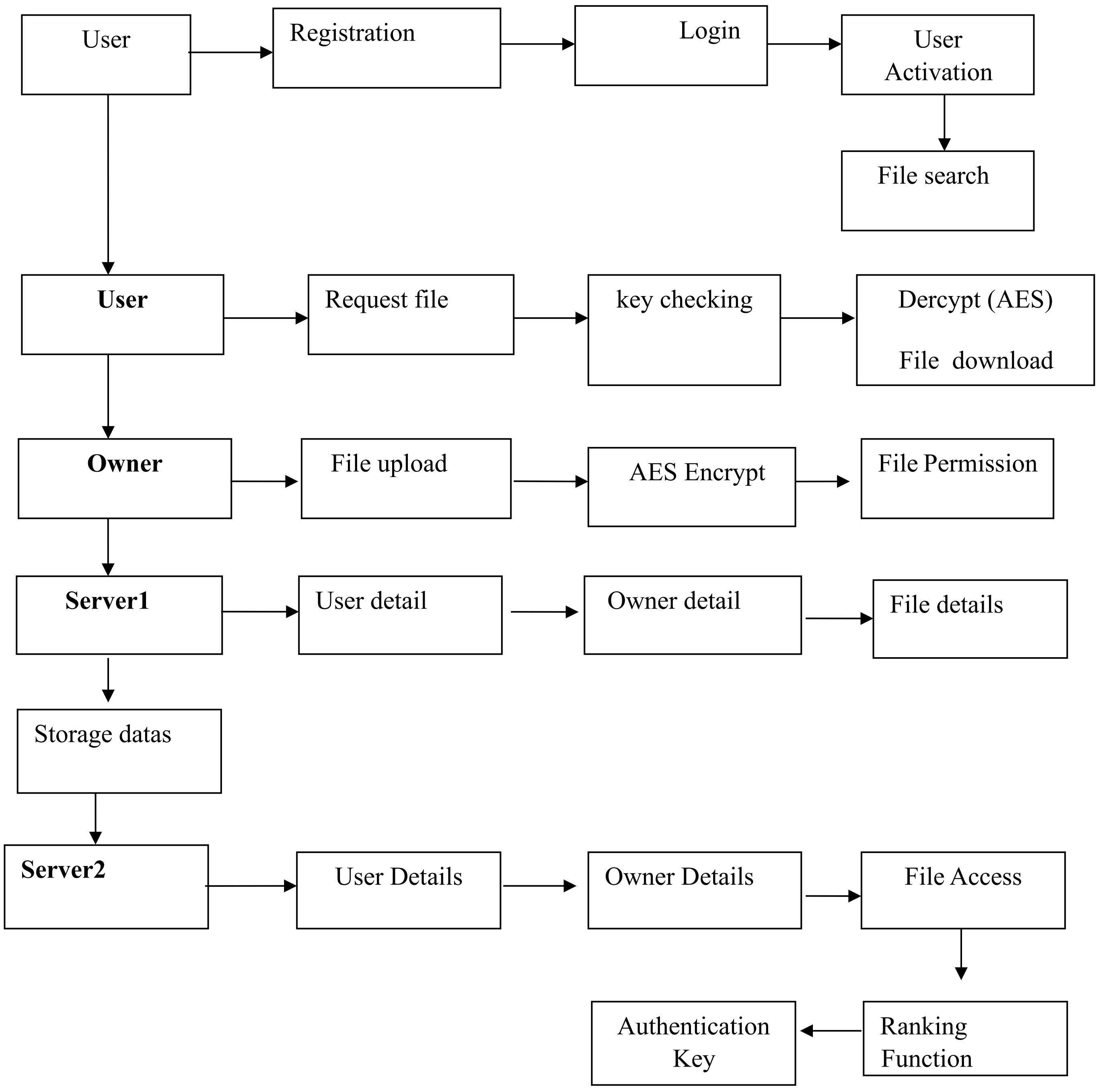
LEVEL-1



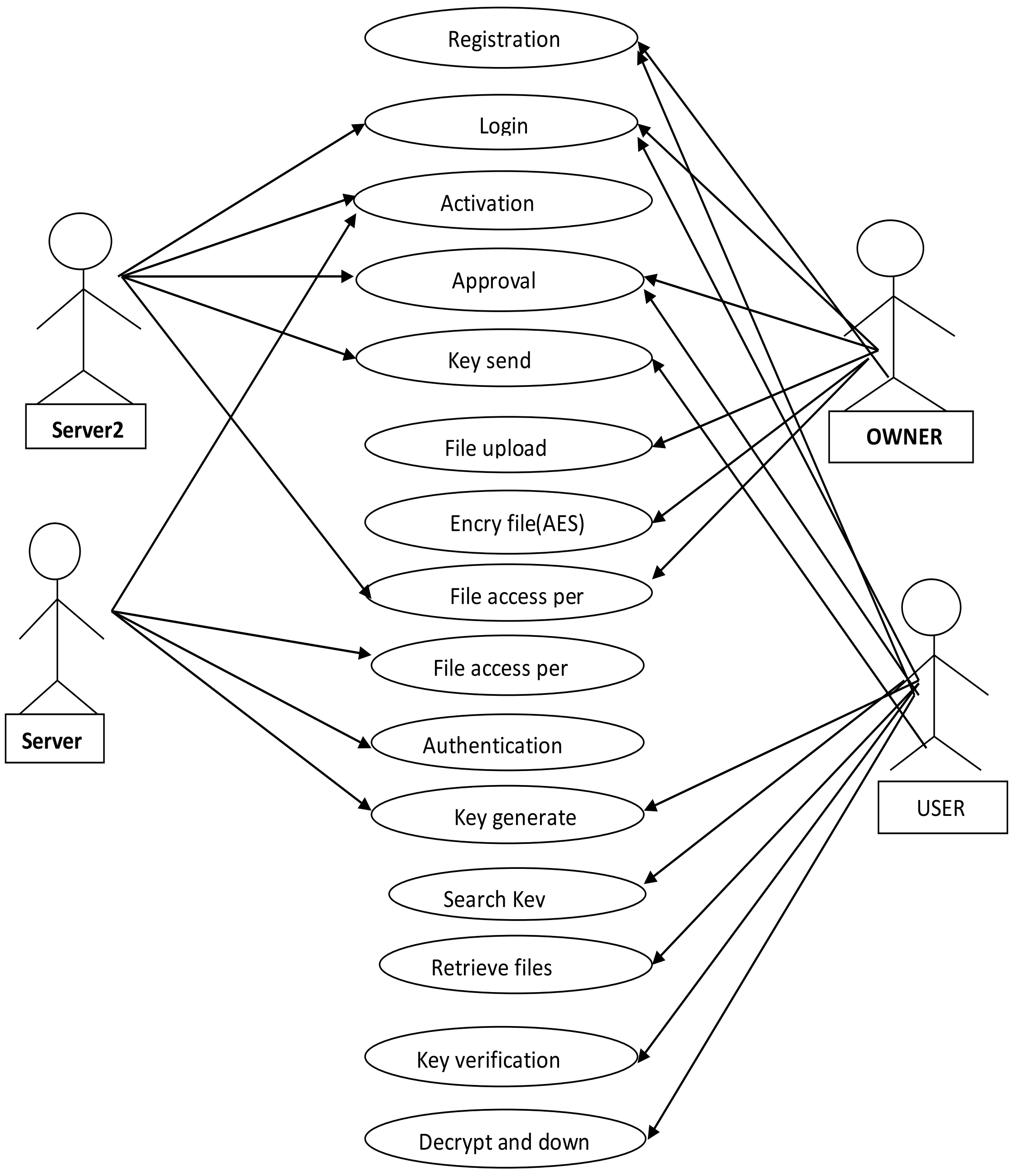
LEVEL-2


User
Request file
key checking
Dercypt (AES)
File download
Owner
File upload
AES Encrypt
File Permission
Server1
User detail
Owner detail
File details
Storage datas
Server2
User Details
Owner Details
File Access
Ranking Function
Authentication Key

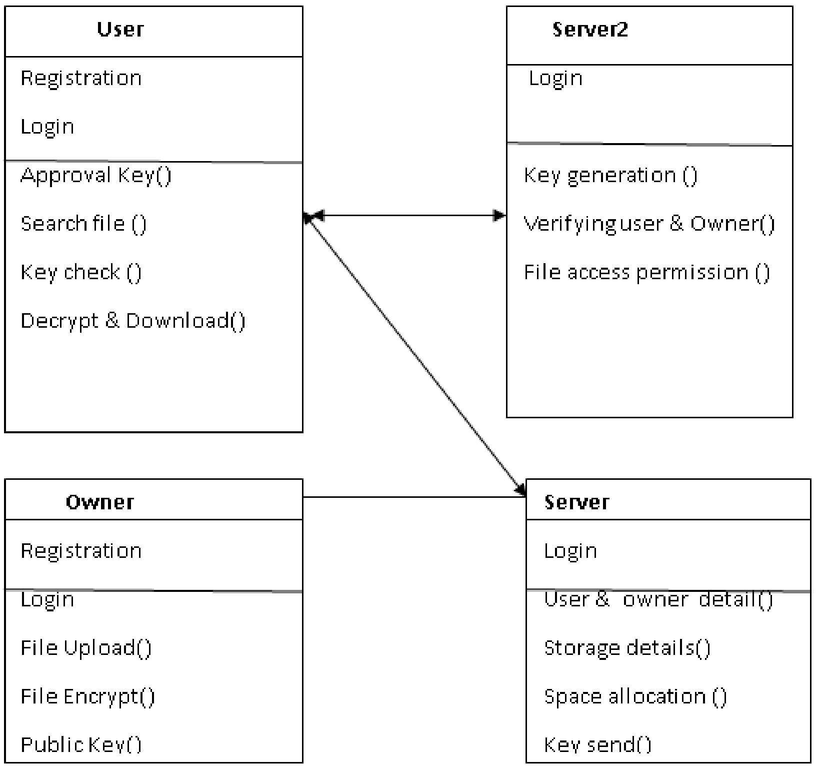
3.2 OVERALL DIAGRAM



3.3 Case Diagram



3.4 Class Diagram



3.5 Architecture Diagram

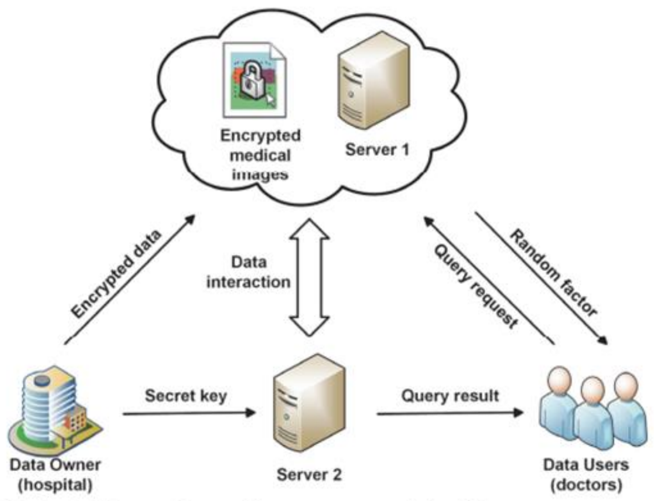


Fig. 1. Architecture for searching over encrypted cloud data

3.6 Source Code

CODING:

OWNERREG.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package DB;
import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.security.SecureRandom;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;
 /**
 *
 * @author welcome
 */
@WebServlet("/owneereg")
@MultipartConfig(maxFileSize = 16177215)    // upload file's size up to 16MB
public class ownereg extends HttpServlet {
   // database connection settings
  protected void doPost(HttpServletRequest request,
           HttpServletResponse response) throws ServletException, IOException {
   try {
      //String id = request.getParameter("id");
   String usernamee = request.getParameter("username");
   String passs = request.getParameter("pass");
   String emaild = request.getParameter("email");
   String gender = request.getParameter("gen");
   String phone = request.getParameter("phone");

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

InputStream inputStream = null;


Part filePart = request.getPart("profile");

if (filePart != null) {


   System.out.println(filePart.getName());

   System.out.println(filePart.getSize());

   System.out.println(filePart.getContentType());


   inputStream = filePart.getInputStream();

}


Connection con = Dbconnection.getConnection();

   Statement st3 = con.createStatement();

try {


   int i =3;

if (i>=1){


      String sql = "Insert into ownreg(usernmae, password, mailid, mobilenumber, gender,
location, status,keyss,uimage) values (?,?,?,?,?,?,?,?,?)";

   PreparedStatement statement = con.prepareStatement(sql);

   statement.setString(1, usernamee);
```

```
    statement.setString(2, passs);
    statement.setString(3, emaild);
    statement.setString(4, phone);
    statement.setString(5,gender);
    statement.setString(6, location);
    statement.setString(7, "waiting");
    statement.setString(8, "Nogenerated");
    if (inputStream != null) {
        statement.setBlob(9, inputStream);
    }
    int row = statement.executeUpdate();
    if (row > 0) {
         response.sendRedirect("Owner_reg.jsp?umssuc=userregister");
    }
    else{
      response.sendRedirect("Owner_reg.jsp?umsfail=failed");
    }
  }
  } catch (SQLException ex) {
    ex.printStackTrace();
  }
  } catch (SQLException ex) {
            Logger.getLogger(ownereg.class.getName()).log(Level.SEVERE, null, ex);
    }
    }

}
```

USERREG.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package DB;
import java.io.IOException;
import java.io.InputStream;
import java.net.InetAddress;
import java.security.SecureRandom;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;

/**
 *
 * @author welcome
 */
@WebServlet("/userreg")
@MultipartConfig(maxFileSize = 16177215)    // upload file's size up to 16MB
public class userreg extends HttpServlet {
   // database connection settings
   protected void doPost(HttpServletRequest request,
          HttpServletResponse response) throws ServletException, IOException {
   try {
   //String id = request.getParameter("id");
   String usernamee = request.getParameter("username");
   String passs = request.getParameter("pass");
   String emaild = request.getParameter("email");
   String gender = request.getParameter("gen");

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

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

InputStream inputStream = null;

Part filePart = request.getPart("profile");

if (filePart != null) {

   System.out.println(filePart.getName());

   System.out.println(filePart.getSize());

   System.out.println(filePart.getContentType());

   inputStream = filePart.getInputStream();

}

Connection con = Dbconnection.getConnection();

   Statement st3 = con.createStatement();

try {

    int i =3;

if (i>=1){

    String sql = "Insert into userreg(usernmae, password, mailid, mobilenumber, gender,
location, status,keyss,uimage) values (?,?,?,?,?,?,?,?,?)";

   PreparedStatement statement = con.prepareStatement(sql);

   statement.setString(1, usernamee);

   statement.setString(2, passs);

   statement.setString(3, emaild);

   statement.setString(4, phone);

   statement.setString(5,gender);

   statement.setString(6, location);
   statement.setString(7, "waiting");

   statement.setString(8, "Nogenerated");

   if (inputStream != null) {

      statement.setBlob(9, inputStream);

   }

   int row = statement.executeUpdate();

   if (row > 0) {

       response.sendRedirect("User_reg.jsp?umsgg=userregister");

   }

   else{

     response.sendRedirect("User_reg.jsp?umsfail=failed");

   }

}

} catch (SQLException ex) {

   ex.printStackTrace();

}

} catch (SQLException ex) {

         Logger.getLogger(userreg.class.getName()).log(Level.SEVERE, null, ex);

   }

   }

}
```

MAIL.JAVA:

```
/*

 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package DB;
/**
 *
 * @author DELL
 */
/*
 * To change this template, choose Tools T | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author welcome
 */
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
/**
```

```
 *
 * @author java4
 */
public class mail1 {
    public static boolean secretMail(String msg, String subb, String email) {
        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.ssl.trust", "smtp.gmail.com");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class",
                "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");
        props.put("mail.java.net.preferIPv4Stack", "true");
        // Assuming you are sending email from localhost
        Session session = Session.getDefaultInstance(props,
                new javax.mail.Authenticator() {
                    protected PasswordAuthentication getPasswordAuthentication() {
                        return new
PasswordAuthentication("stockmanagements635@gmail.com", "stock1994");
                    }
                })
        System.out.println("Message   " + msg);
        try {
            Message message = new MimeMessage(session);
```

```
          message.setFrom(new InternetAddress(subb));

          message.setRecipients(Message.RecipientType.TO,

                InternetAddress.parse(email));

          message.setSubject(subb);

         // message.setText(msg,"text/html; charset=utf-8");

          message.setContent(msg, "text/html; charset=utf-8");

          Transport.send(message);

          System.out.println("Done");

          return true;

       } catch (MessagingException e) {

          System.out.println(e);

          e.printStackTrace();

          return false;

          // throw new RuntimeException(e);

       }

    }

}
```

FTPCON.JAVA:

```
/*

 * To change this template, choose Tools | Templates

 * and open the template in the editor.

 */

package DB;

/**
```

```
 *
 * @author welcome
 */
import java.io.File;
import java.io.FileInputStream;
import org.apache.commons.net.ftp.FTPClient;
public class Ftpcon {
   FTPClient client = new FTPClient();
   FileInputStream fis = null;
   boolean status;
   public boolean upload(File file,String filetyp ,String foldername) {
      try {
         String nnn="PPPPPPPPPPPPPP"+filetyp;
         String mmm="LLLLLLLLLLLLL"+foldername;
         System.out.println(nnn);
         System.out.println(mmm);
         client.connect("ftp.drivehq.com");
         client.login("cloudclouds24", "cloudcloud1994");
         client.enterLocalPassiveMode();
         fis = new FileInputStream(file);
         status = client.storeFile(" /"+filetyp+"/"+foldername+"/" + file.getName(), fis);
         //status = client.storeFile(" /textfile/server3/" + file.getName(), fis);
         client.logout();
         fis.close();
      } catch (Exception e) {
            System.out.println(e);
        }
        if (status) {
            System.out.println("success");
            return true;
        } else {
            System.out.println("failed");
            return false;
        }
    }
}
```

FILEUPLOAD.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package DB;
import com.oreilly.servlet.MultipartRequest;
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

import java.io.PrintWriter;

import java.security.SecureRandom;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.Statement;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.Random;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/**

 *

 * @author java4

 */

public class fileupload extends HttpServlet {
  File file;

  final String filepath = "E:/encfiles/";

PreparedStatement pstm = null;

  protected void processRequest(HttpServletRequest request, HttpServletResponse
response)

        throws ServletException, IOException {

      response.setContentType("text/html;charset=UTF-8");

      PrintWriter out = response.getWriter();

      try {

        MultipartRequest m = new MultipartRequest(request, filepath);

            HttpSession session = request.getSession(true);

        File file = m.getFile("file");

        String filename = file.getName().toLowerCase();

       //String test = "http://www.example.com/abc?page=6";



        Connection con = Dbconnection.getConnection();

                  BufferedReader br = new BufferedReader(new FileReader(filepath +
filename));

        StringBuffer sb = new StringBuffer();

        String temp = null;



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

          sb.append(temp);

        }

        String str = sb.toString();
```

```
System.out.println("File in SB :" + sb.toString());
int size = (int) file.length();
System.out.println("File Size :" + size);
 String ownernmae = (String) request.getSession().getAttribute("ownername");
 String oidd = (String) request.getSession().getAttribute("ownerid");


String fname=m.getParameter("fname");
String descrip=m.getParameter("descrip");
String akey1=m.getParameter("akey1");
String akey2=m.getParameter("akey2");
String akey3=m.getParameter("akey3");
String filetypes = m.getParameter("ftype");
Calendar cal = Calendar.getInstance();
SimpleDateFormat format = new SimpleDateFormat("HH:mm dd/MM/yyyy");
          String report= format.format(cal.getTime());
System.out.println("File Size :" + size);
     FileInputStream fiss = null;
     fiss = new FileInputStream(file);
                KeyGenerator keyGen = KeyGenerator.getInstance("AES");
                keyGen.init(128);
                SecretKey secretKey = keyGen.generateKey();
                System.out.println("secret key:" + secretKey);
                //converting secretkey to String
                byte[] be = secretKey.getEncoded();//encoding secretkey
```

```
                    String skey = Base64.encode(be);

                    System.out.println("converted secretkey to string:" + skey);

                    String encstr = new BlowfishAlgorithm().encrypt(str);

                    System.out.println(str);

//                   String decry = new BlowfishAlgorithm().decrypt(str);

//                    System.out.println("decc Text :" + decry);




        DateFormat dateFormat = new SimpleDateFormat("yyyy.MM.dd G 'at'
HH:mm:ss ");

        Date date = new Date();

        String time = dateFormat.format(date);

        System.out.println("current Date " + time);

        Random ra = new Random();

            int range=ra.nextInt(3)+1;

            Random skeyy = new SecureRandom();

          int skey2 = 25;

          String subauthkey1 =
"1010010101001100111010010100101001001001010010";

          String subautk1 = "";

          String subautk2 = "";

          for (int i = 0; i < skey2; i++) {

            int index = (int) (skeyy.nextDouble() * subauthkey1.length());

            subautk1 += subauthkey1.substring(index, index + 1);

          }
```

```
                subautk2 = new BlowfishAlgorithm().encrypt(subautk1);

//int range = 0 - 3 + 1;

String folder;

System.out.println("***************"+range);

if(range==1){

   folder="server1";

}

else if(range==2){

   folder="server2";

}else if(range==3)

{

   folder="server3";

}else{

   folder="server2";

}

   System.out.println("***************"+folder);

            Integer n=5 ;

           if (n>3) {

                  con = Dbconnection.getConnection();

                         pstm = con.prepareStatement("insert into ownerfilee
(ownername,ownerid,filename,descrip,akey1,akey2,akey3,files,encryptf,privatekey,publ
ickey,udate,status,filetypes,folders)values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

               Statement st = con.createStatement();



                         pstm.setString(1, ownernmae);
```

```
                        pstm.setString(2, oidd);
                        pstm.setString(3, fname);
                        pstm.setString(4, descrip);
                        pstm.setString(5, akey1);
                        pstm.setString(6, akey2);
                        pstm.setString(7, akey3);
                        pstm.setBinaryStream(8,fiss);
                        pstm.setString(9, encstr);
pstm.setString(10, subautk1);
                        pstm.setString(11, subautk2);
                        pstm.setString(12, report);
                        pstm.setString(13, "waiting");
                        pstm.setString(14, filetypes);
                        pstm.setString(15, folder);
             int i = pstm.executeUpdate();
             //session.setAttribute("ofilen", fiename);
             //session.setAttribute("trapp", skey);
             //session.setAttribute("oencc", encstr);
             //cloud storing encrypted file
          FileWriter fw = new FileWriter(file);
          fw.write(encstr);
          fw.close();
          FileInputStream fis = null;
          fis = new FileInputStream(file);
```

```
            boolean status = new Ftpcon().upload(file,filetypes,folder);
                        if (status) {
                            response.sendRedirect("Owner_fileup1.jsp?umssuc=success");
                        } else {
                            response.sendRedirect("Owner_fileup1.jsp?umsfail=failed");
                        }
            } else {
                out.println("Error in FTP Connection");
            }
        } catch (Exception e) {
            out.println(e);
        } finally {
            out.close();
        }
    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
     * Handles the HTTP
     * <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
```

```
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}
/**
 * Handles the HTTP
 * <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
```

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

DOWNLOAD.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package DB;


import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * A servlet that retrieves a file from MySQL database and lets the client
 * downloads the file.
 * @author www.codejava.net
 */
@WebServlet("/download")
public class downnload extends HttpServlet {

   // size of byte buffer to send file
   private static final int BUFFER_SIZE = 4096;

   // database connection settings
```

```
protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    // get upload id from URL's parameters\
     HttpSession session = request.getSession();
     String fid = session.getAttribute("dowfileid").toString();
     System.out.println("RRRRRRRRRR"+fid);
//        String usernmae = session.getAttribute("unames").toString();
//         String uidd = session.getAttribute("uidd").toString();
     Integer id=Integer.parseInt(fid);


    Connection conn = null; // connection to the database
     Statement st = null;
    try {
        // connects to the database


        conn =  Dbconnection.getConnection();;
                st = conn.createStatement();
        // queries the database
        String sql = "SELECT * FROM ownerfilee WHERE id = ?";
        PreparedStatement statement = conn.prepareStatement(sql);
        statement.setInt(1, id);
        String fileName =null;
        String encry =null;
```

```
            ResultSet result = statement.executeQuery();
            if (result.next()) {
                // gets file name and file blob data
                fileName = result.getString("filename");
                encry = result.getString("encryptf");
                Blob blob = result.getBlob("files");
                InputStream inputStream = blob.getBinaryStream();
                int fileLength = inputStream.available();

                System.out.println("fileLength = " + fileLength);
                    String decstr = new BlowfishAlgorithm().decrypt(encry);
                    System.out.println("*****"+decstr);
                ServletContext context = getServletContext();

                // sets MIME type for the file download
                String mimeType = context.getMimeType(fileName);
                if (mimeType == null) {
                    mimeType = "application/octet-stream";
                } int i=3;
//              if(i>=1){
//                  Calendar cal = Calendar.getInstance();
//          SimpleDateFormat format = new SimpleDateFormat("HH:mm dd/MM/yyyy");
//                    String report= format.format(cal.getTime());
//              int j = st.executeUpdate("insert into
downloaduser(usreid,username,filename,date)values('"+uidd+"','"+usernmae+"','"+fil
eName+"','"+report+"')");

//          }



                // set content properties and header attributes for the response

                response.setContentType(mimeType);

                response.setContentLength(fileLength);

               response.setHeader("Content-Disposition", "attachment;filename=\"" +
fileName + ".txt\"");




                // writes the file to the client

                OutputStream outStream = response.getOutputStream();



                byte[] buffer = new byte[BUFFER_SIZE];

                int bytesRead = -1;



                while ((bytesRead = inputStream.read(buffer)) != -1) {

                    outStream.write(buffer, 0, bytesRead);

                }



                inputStream.close();

                outStream.close();

            } else {

                // no file found
```

```
                response.getWriter().print("File not found for the id: " + id);
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            response.getWriter().print("SQL Error: " + ex.getMessage());
        } catch (IOException ex) {
            ex.printStackTrace();
            response.getWriter().print("IO Error: " + ex.getMessage());
        } finally {
            if (conn != null) {
                // closes the database connection
                try {
                    conn.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        }
    }
}
```

DBCONNECTION.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package DB;


/**

 *

 * @author welcome

 */

import java.sql.Connection;

import java.sql.DriverManager;


/**

 *

 * @author java2

 */

public class Dbconnection {


    public static Connection getConnection() {

        Connection con = null;

        try {

            Class.forName("com.mysql.jdbc.Driver");

            con = 
DriverManager.getConnection("jdbc:mysql://localhost:3306/image_enrcyption", 
"root", "");

        } catch (Exception ex) {

            ex.printStackTrace();
        }
        return con;
    }
}
```

BLOWFISHALGORITH.JAVA:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package DB;


/**
 *
 * @author welcome
 */
import com.sun.org.apache.xml.internal.security.exceptions.Base64DecodingException;
import com.sun.org.apache.xml.internal.security.utils.Base64;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;


/**
 *
 * @author dhanoopbhaskar
 */
public class BlowfishAlgorithm {


    KeyGenerator keyGenerator = null;
    SecretKey secretKey = null;
    Cipher cipher = null;


    public BlowfishAlgorithm() {
        try {
            /**
             * Create a Blowfish key
             */
            keyGenerator = KeyGenerator.getInstance("Blowfish");
            secretKey = keyGenerator.generateKey();


            /**
             * Create an instance of cipher mentioning the name of algorithm
             *     - Blowfish
             */
            cipher = Cipher.getInstance("Blowfish");
        } catch (NoSuchPaddingException ex) {
            System.out.println(ex);
        } catch (NoSuchAlgorithmException ex) {
            System.out.println(ex);
        }

    }

    /**
     *
     * @param plainText
     * @return cipherBytes
     */
    public byte[] encryptText(String plainText) {
        byte[] cipherBytes = null;
        try {
            /**
             * Initialize the cipher for encryption
             */
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            /**
             * Convert the text string to byte format
             */
            byte[] plainBytes = plainText.getBytes();
            /**
             * Perform encryption with method doFinal()
             */
            cipherBytes = cipher.doFinal(plainBytes);
        } catch (IllegalBlockSizeException ex) {
            System.out.println(ex);
        } catch (BadPaddingException ex) {
            System.out.println(ex);
        } catch (InvalidKeyException ex) {
            System.out.println(ex);
        }

        return cipherBytes;
    }

    /**
     *
     * @param cipherBytes
     * @return plainText
     */
    public String decryptText(byte[] cipherBytes) {
        String plainText = null;
        try {
            /**
             * Initialize the cipher for decryption
             */
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            /**
             * Perform decryption with method doFinal()
             */
            byte[] plainBytes = cipher.doFinal(cipherBytes);
            /**
             * Convert encrypted text to string format
             */
            plainText = new String(plainBytes);
        } catch (IllegalBlockSizeException ex) {
            System.out.println(ex);
        } catch (BadPaddingException ex) {
            System.out.println(ex);
        } catch (InvalidKeyException ex) {
            System.out.println(ex);
        }

        return plainText;
    }

    /**
```

```
     *
     * @param plainText
     * @return cipherText
     */
    public String encrypt(String plainText) {
        String cipherText = null;
        byte[] cipherBytes = encryptText(plainText);
        cipherText = bytesToString(cipherBytes);
        return cipherText;
    }

    /**
     *
     * @param cipherText
     * @return plainText
     */
    public String decrypt(String cipherText) {
        String plainText = null;
        byte[] cipherBytes = stringToBytes(cipherText);
        plainText = decryptText(cipherBytes);
        return plainText;
    }

//    public static void main(String[] args) {
```

```
//        BlowfishAlgorithm blowfishAlgorithm = new BlowfishAlgorithm();

//        String textToEncrypt = "Blowfish Algorithm";

//        System.out.println("Text before Encryption: " + textToEncrypt);

//        String cipherText = blowfishAlgorithm.encrypt(textToEncrypt);

//        System.out.println("Cipher Text: " + cipherText);

//        System.out.println("Text after Decryption: " +
blowfishAlgorithm.decrypt(cipherText));

//    }


    /**

     *

     * @param rawText

     * @return plainText

     *

     * Perform Base64 encoding

     */

    private String bytesToString(byte[] rawText) {

        String plainText = null;

        plainText = Base64.encode(rawText);

        return plainText;

    }


    /**

     *

     * @param plainText
     * @return rawText
     *
     * Perform Base64 decoding
     */
    private byte[] stringToBytes(String plainText) {
        byte[] rawText = null;
        try {
            rawText = Base64.decode(plainText);
        } catch (Base64DecodingException ex) {
            System.out.println(ex);
        }
        return rawText;
    }
}
```

CHAPTER – 4

TESTING

4.1 Unit Testing

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

4.2 Integration Testing

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

4.3 Functional Testing

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

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

4.4 System Testing

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

4.5 White Box Testing

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

4.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

CHAPTER - 5
SYSTEM TESTING & IMPLEMENTATION

5.1 Introduction

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

5.2 Unit Testing

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

5.2.1 Test strategy and approach

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

5.2.2 Test Objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

5.2.3 Features to be Tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

5.3 Integration Testing

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

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

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

5.4 Acceptance Testing

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

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

5.5 Test Case

Table 5.5.1

Test Case ID	Test Case Action	Expected Result	Actual Result	Test Case Status
1	Import Dataset	After importing Data set details on file entered the page of files should be displayed	As excepted	Pass
		If the importing Data set details on file not entered page of files should be displayed should not be displayed	Something went wrong	Fail

Table 5.5.2

Test Case ID	Test Case Action	Expected Result	Actual Result	Test Case Status
1	Keystroke Analysis	To Predict the Keystroke value added	As expected	Pass

Table 5.5.3

Test Case ID	Test Case Action	Expected Result	Actual Result	Test Case Status
1	Classification Technique	After Detecting Keystroke files New tenders should be displayed	As excepted	Pass
		After Detecting which files not Detecting Keystroke should be displayed not be displayed	Something went wrong	Fail

Table 5.5.4

Test Case ID	Test Case Action	Expected Result	Actual Result	Test Case Status
1	Product upload	After entering Product on file upload page of files should be displayed	As excepted	Pass
		If the entering Product on file upload page of files should be displayed should not be displayed	Something went wrong	Fail

Table 5.5.5

Test Case ID	Test Case Action	Expected Result	Actual Result	Test Case Status
1	Fire wall agents	To Predict the user has to Received the files.	As expected	Pass

Table 5.5.6

Test Case ID	Test Case Action	Expected Result	Actual Result	Test Case Status
1	Performance comparison	After Detecting which files not performance comparison should be displayed	As excepted	Pass
		After Detecting which files not performance comparison the user location should be displayed not be displayed	Something went wrong	Fail

5.6 Software Environment

Java Technology

Java technology is both a programming language and a platform.

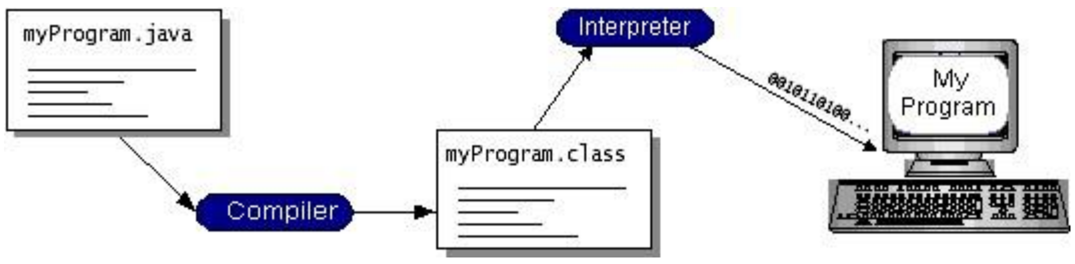
The Java Programming Language

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

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted

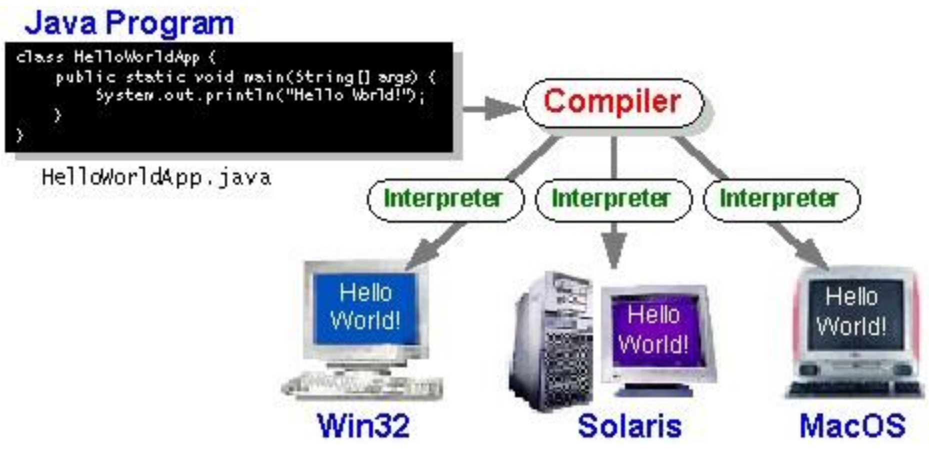
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



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

Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The Java Platform

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

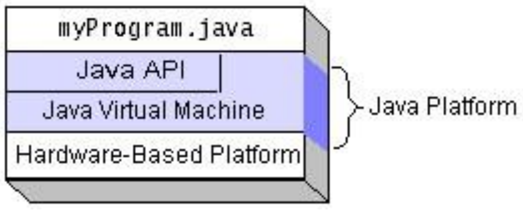
The Java platform has two components:

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

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

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

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



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

What Can Java Technology Do?

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

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

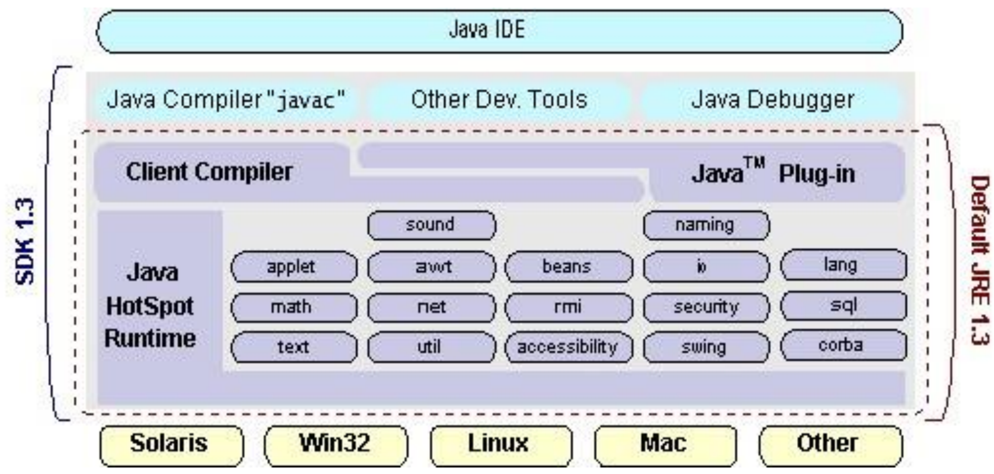
An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java
Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

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

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.

- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans^{TM}, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC^{TM})**: Provides uniform access to a wide range of relational databases.

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



How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get Started Quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.

- **Write Less Code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write Better Code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop Programs More Quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid Platform Dependencies With 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java^{TM} Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write Once, Run Anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute Software More Easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

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

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access

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

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

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

JDBC was announced in March of 1996. It was released for a 90 day public review that ended
June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC.
That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. SQL Level API

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

2. SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle nonstandard functionality in a manner that is suitable for its users.

3. **JDBC must be implemental on top of common database interfaces**

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. Provide a Java interface that is consistent with the rest of the Java system

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. Keep it simple

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. Use strong, static typing wherever possible

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

7. Keep the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

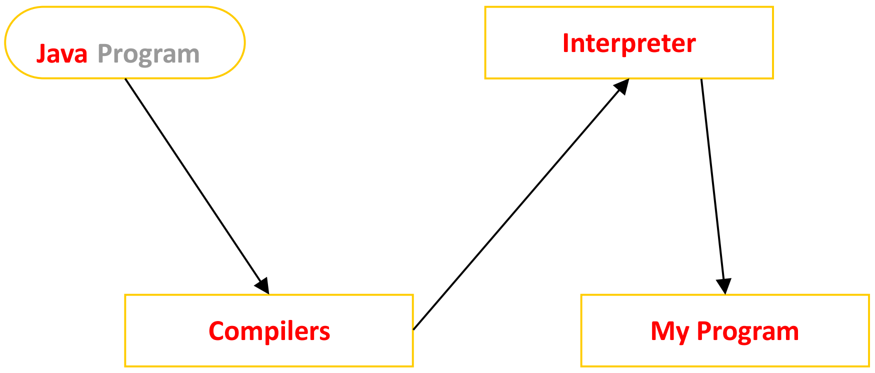
- Finally we decided to precede the implementation using Java Networking.
- And for dynamically updating the cache table we go for MS Access database.
- Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

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

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.



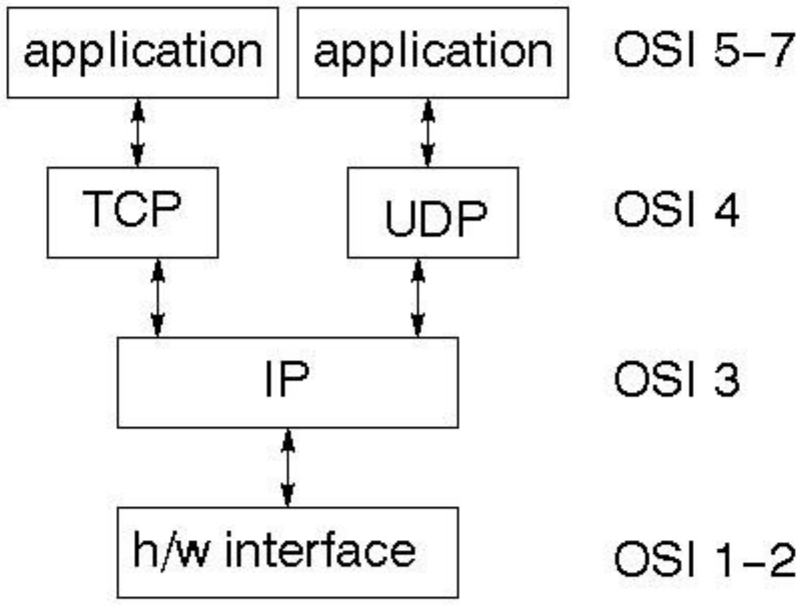
You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

Networking:

TCP/IP stack

The TCP/IP Stack Is Shorter Than the OSI One



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

IP Datagram's

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

UDP

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

TCP

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

Internet Addresses

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

Network Address

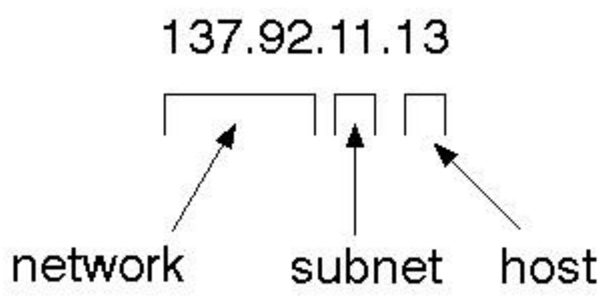
Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32. **Subnet Address**

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

Host Address

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

Total Address



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

Port Addresses

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

Sockets

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

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

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

JFree Chart

JFree Chart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFree Chart's extensive feature set includes: A consistent and well-documented API, supporting a wide range of chart types; A flexible design that is easy to extend, and targets both server-side and client-side applications; Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG); JFree Chart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public License (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include: Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas); Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart; Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

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

3. Dashboards

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

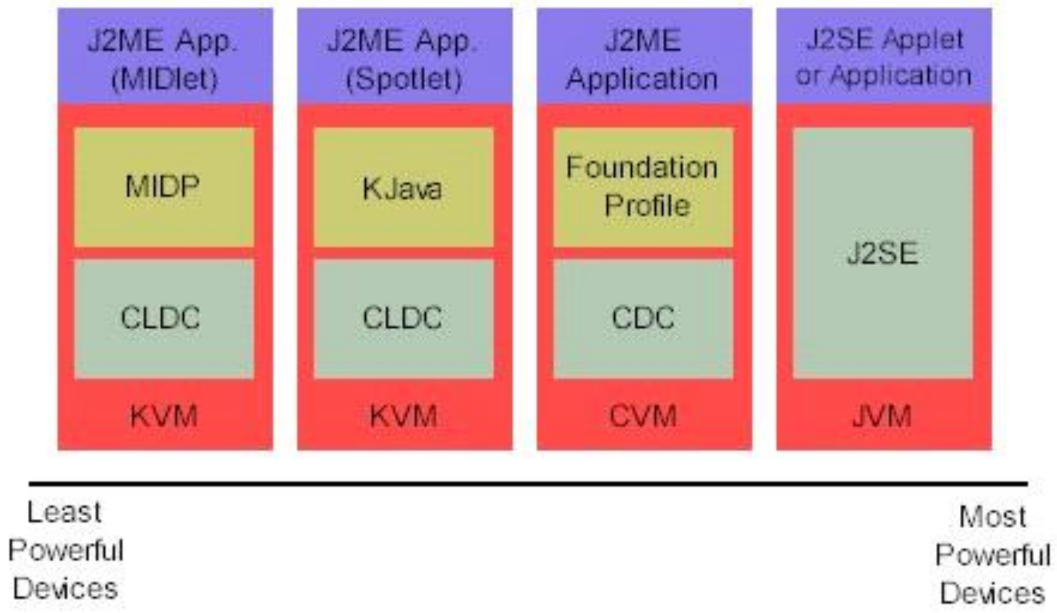
4. Property Editors

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

J2ME (Java 2 Micro Edition) :-

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1.General J2ME Architechture:



J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role pre verification plays in this process.

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

* Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

4. Configurations overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5. J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

* java.lang

* java.io

* java.util

* javax.microedition.io

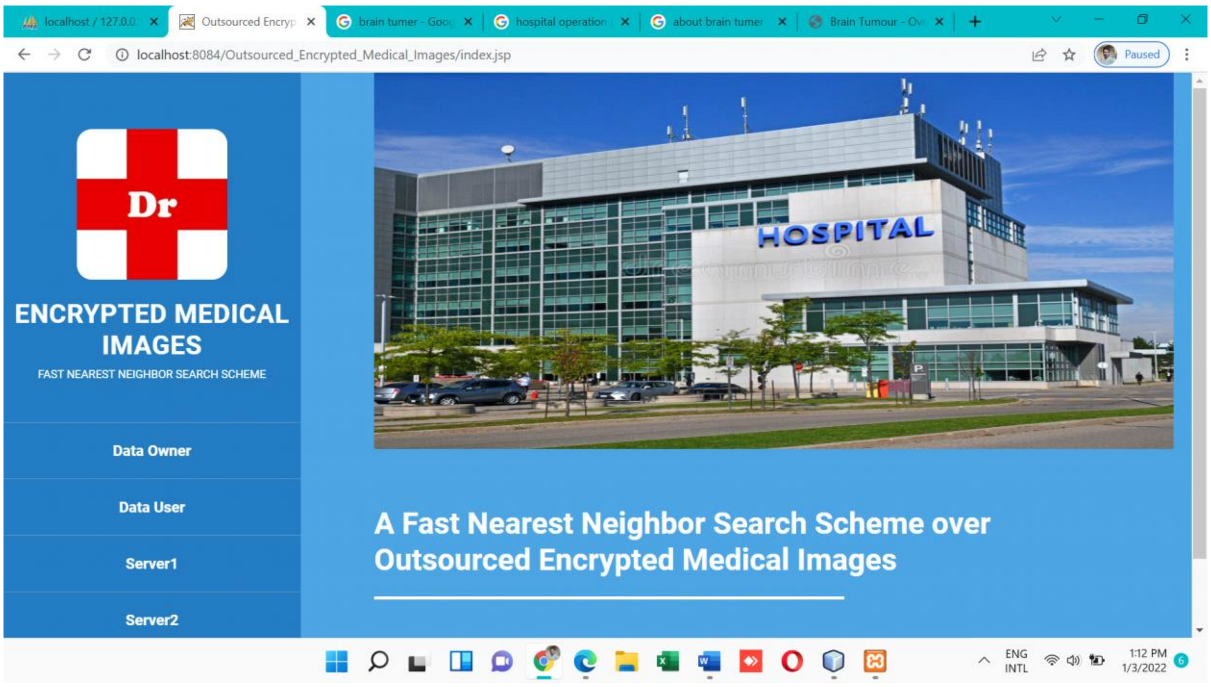
* javax.microedition.lcdui

* javax.microedition.midlet

* javax.microedition.rms

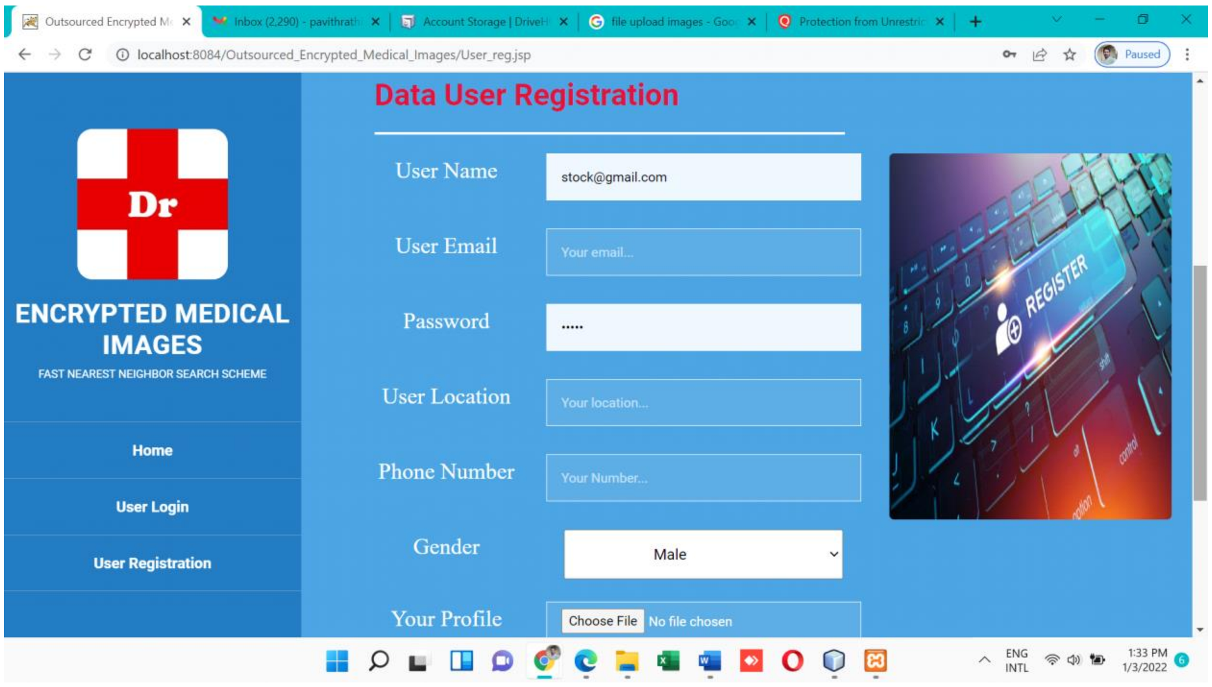
CHAPTER – 6
RESULTS & CONCLUSION

6.1 Results



localhost:8084/Outsourced_Encrypted_Medical_Images/index.jsp
Dr
ENCRYPTED MEDICAL IMAGES
FAST NEAREST NEIGHBOR SEARCH SCHEME
Data Owner
Data User
Server1
Server2
HOSPITAL

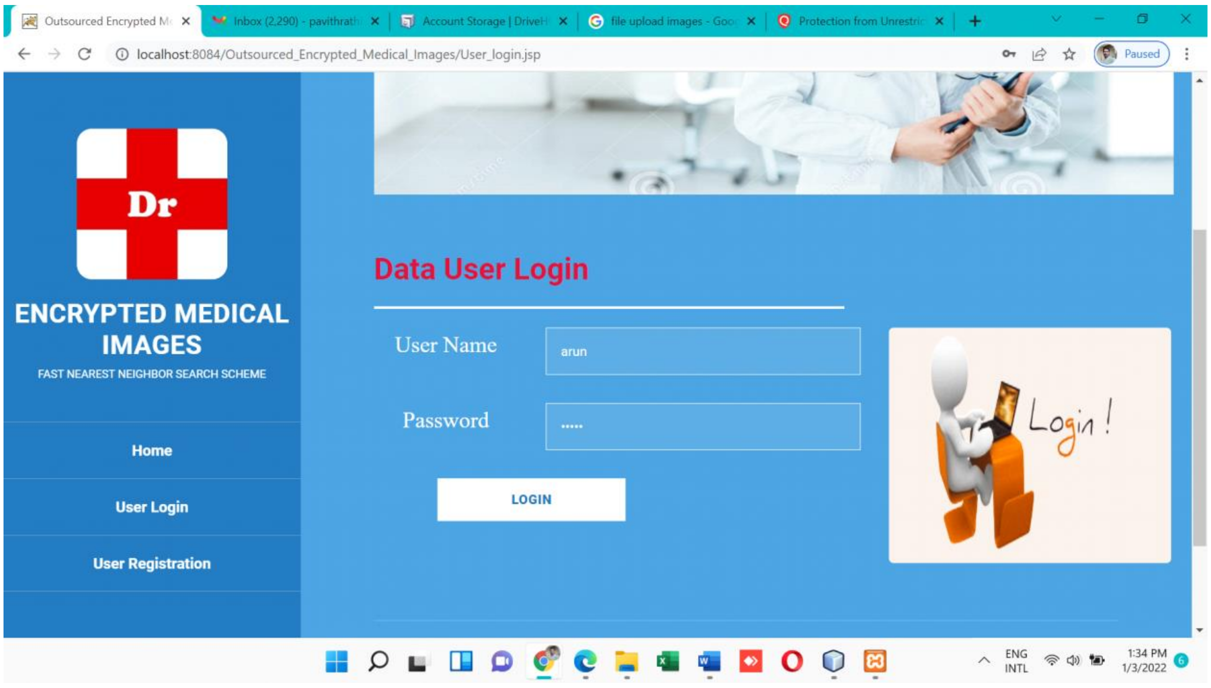
A Fast Nearest Neighbor Search Scheme over Outsourced Encrypted Medical Images



localhost:8084/Outsourced_Encrypted_Medical_Images/User_reg.jsp
Dr
ENCRYPTED MEDICAL IMAGES
FAST NEAREST NEIGHBOR SEARCH SCHEME
Home
User Login
User Registration
REGISTER

Data User Registration

User Name	stock@gmail.com
User Email	Your email...
Password	.....
User Location	Your location...
Phone Number	Your Number...
Gender	Male
Your Profile	Choose File No file chosen

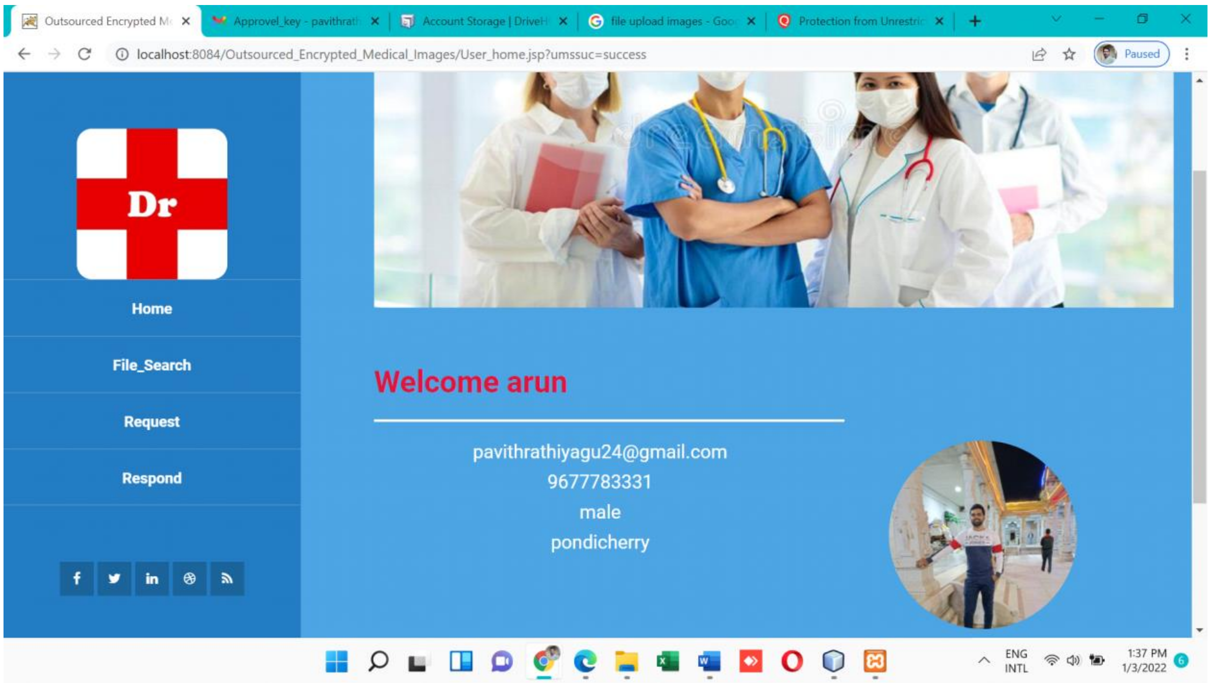

localhost:8084/Outsourced_Encrypted_Medical_Images/User_login.jsp

ENCRYPTED MEDICAL IMAGES

FAST NEAREST NEIGHBOR SEARCH SCHEME

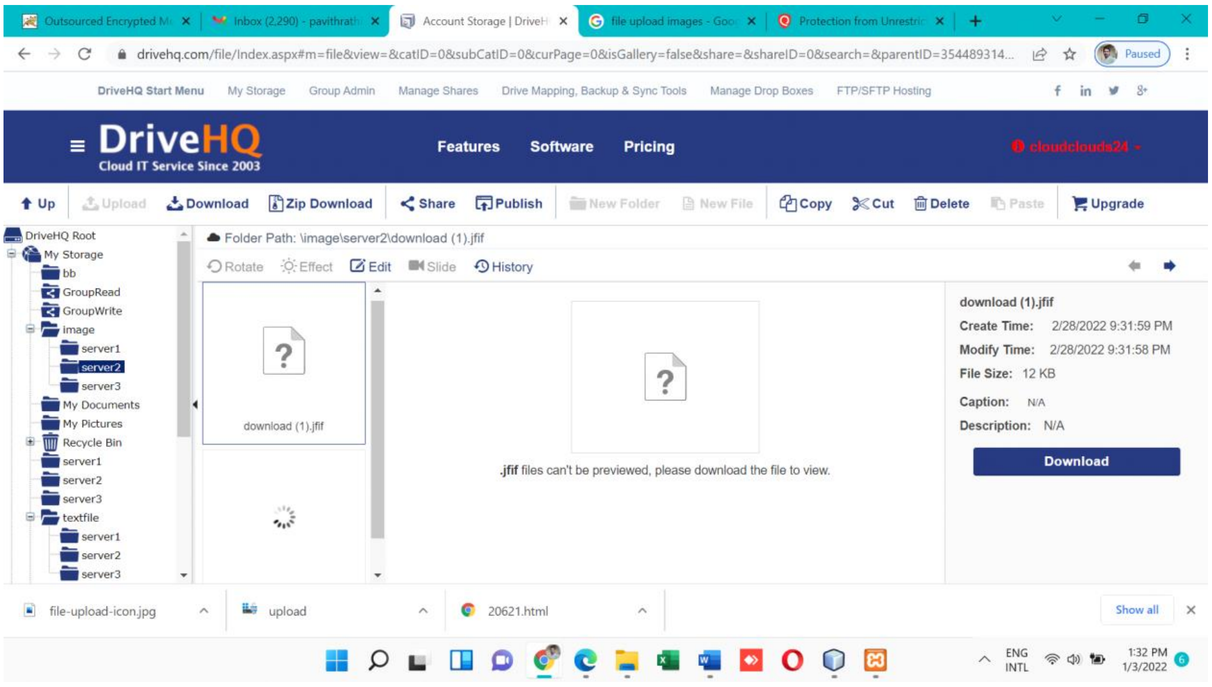
Data User Login

User Name	arun
Password	.....


localhost:8084/Outsourced_Encrypted_Medical_Images/User_home.jsp?umssuc=success

Welcome arun

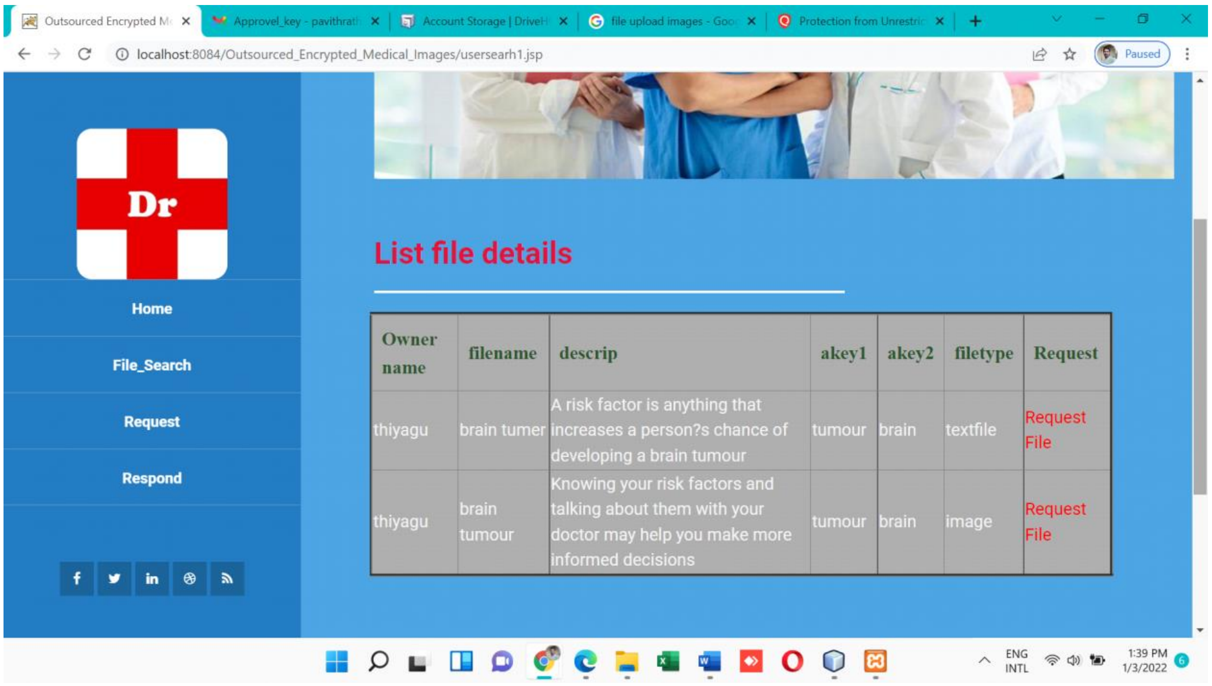
pavithrathiyagu24@gmail.com
9677783331
male
pondicherry



Folder Path: \image\server2\download (1).jfif

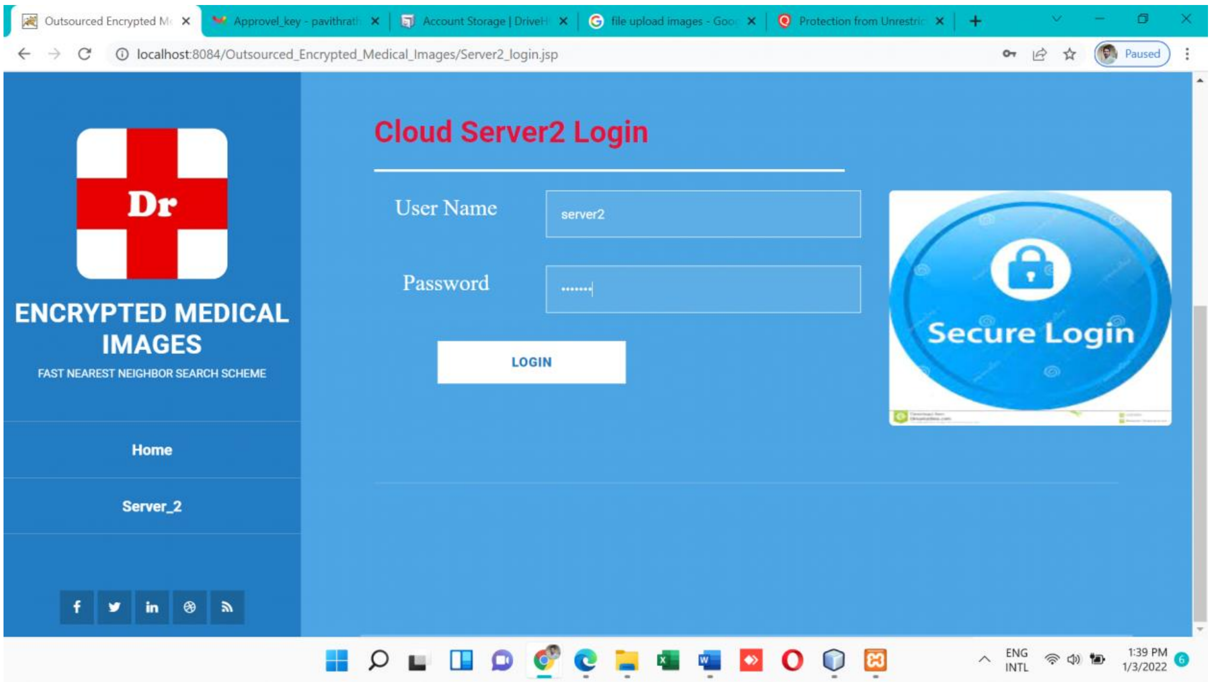
.jfif files can't be previewed, please download the file to view.

download (1).jfif
Create Time: 2/28/2022 9:31:59 PM
Modify Time: 2/28/2022 9:31:58 PM
File Size: 12 KB
Caption: N/A
Description: N/A



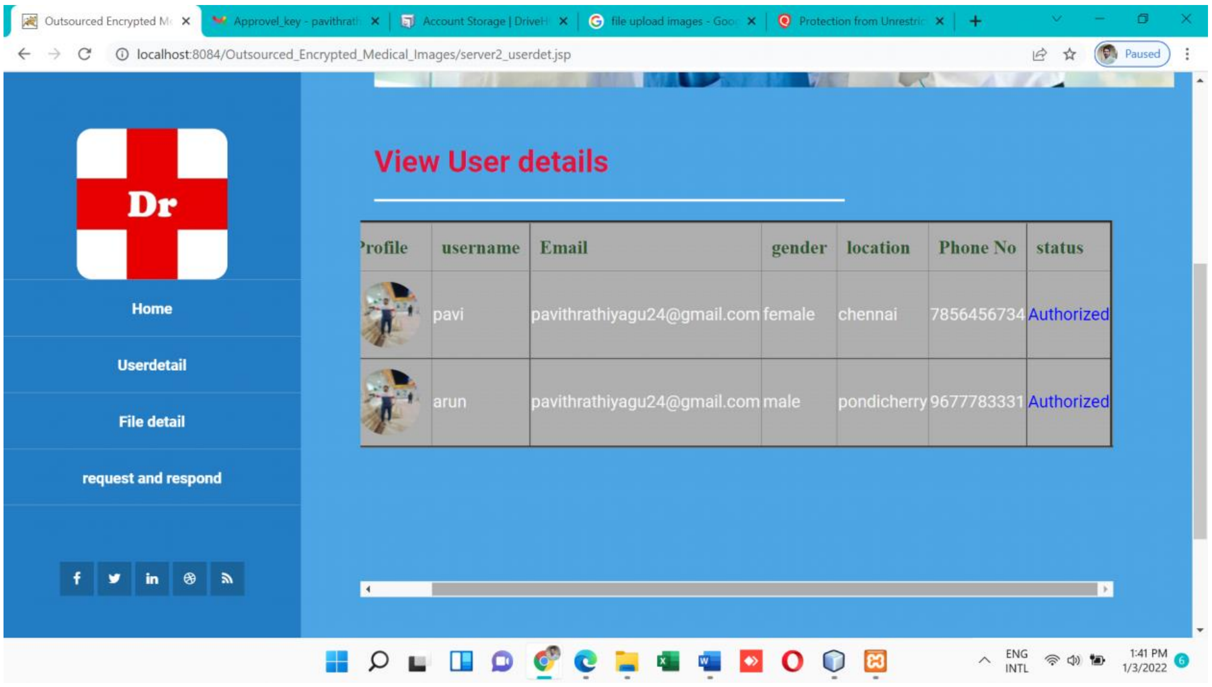
List file details

Owner name	filename	descrip	akey1	akey2	filetype	Request
thiyagu	brain tumer	A risk factor is anything that increases a person?s chance of developing a brain tumour	tumour	brain	textfile	Request File
thiyagu	brain tumour	Knowing your risk factors and talking about them with your doctor may help you make more informed decisions	tumour	brain	image	Request File



Cloud Server2 Login

User Name	server2
Password	•••••••



View User details

Profile	username	Email	gender	location	Phone No	status
	pavi	pavithrathiyagu24@gmail.com	female	chennai	7856456734	Authorized
	arun	pavithrathiyagu24@gmail.com	male	pondicherry	9677783331	Authorized

6.2 Conclusion

Cloud-based electronic healthcare systems will be increasing popular, particularly due to the capability to share and access data in real-time across organizations .From several attack tests that have been carried out on ciphertext to determine the resistance of the AES method, it was found that the determinant of the success or failure of the decryption process of the image file depends on the pixel value. When the pixel value of the encrypted image is changed, the decryption process have been successful, but it cannot restore the plaintext imagewe presented a secure and efficient scheme to locate the exact nearest neighbour over encrypted medical images stored.To over come storage problem we split storage space into different way we have created multiple folders. The Advanced Encryption Standard (AES) algorithm was successfully applied to encrypt an image. In the decryption process, this method can restore plaintext as clear as before. Attack test is given on the ciphertext by cropping, blurring, and enhancing. It is found that this method can recognize plaintext clearly for cropping attacks. The performance of our scheme is evaluated using real-world medical images.

6.3 Future Enhancements

Future study will include locating a real-world healthcare organisation to construct and deploy a prototype of the proposed technique. This will enable us to evaluate the proposed system's real-world use as well as its practical scalability. It will also allow us to find any weaknesses or constraints that we were previously unaware.

REFERENCES

[1] J. Li, L. Huang, Y. Zhou, S. He, Z. Ming, “Computation partitioning for mobile cloud computing in big data environment,” IEEE Trans. Ind. Informat., vol. 13, no. 4, pp. 2009-2018, Feb. 2017.

[2] K.-K. R. Choo, “Cloud computing: Challenges and future directions,” Trends & Issues in Crime and Criminal Justice, vol. 400, no. 400, pp. 1– 6, Oct. 2010.

[3] M. Pajic, R. Mangharam, O. Sokolsky, D. Arney, J. M. Goldman, and I. Lee, “Model-driven safety analysis of closed-loop medical systems,” IEEE Trans. Ind. Informat., vol. 10, no. 1, pp. 3–16, Feb. 2014.

[4] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, and F. Bu, “Ubiquitous data accessing method in IoT-based information system for emergency medical services,” IEEE Trans. Ind. Informat., vol. 10, no. 2, pp. 1578– 1586, May. 2014.

[5] G. Yang et al., “A health-IoT platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box,” IEEE Trans. Ind. Informat., vol. 10, no. 4, pp. 2180–2191, Nov. 2014

. [6] H. Huang, T. Gong, N. Ye, R. Wang, and Y. Dou, “Private and Secured Medical Data Transmission and Analysis for Wireless Sensing Healthcare System,” IEEE Trans. Ind. Informat., vol. 13, no.3 pp. 1227-1237, June. 2017.

[7] M. Li, S. Yu, W. Lou, and Y. T. Hou, “Toward privacy-assured cloud data services with flexible search functionalities,” in Proc. ICDCSW. IEEE, Macau, CHN, 2012, pp. 466–470. [

8] P. Williams, R. Sion, and B. Carbunar, “Building castles out of mud: practical access pattern privacy and correctness on untrusted storage,” in Proc. CCS. ACM, Alexandria, VA, USA, 2008, pp. 139–148.

[9] M. S. Islam, M. Kuzu, and M. Kantarcioglu, “Access pattern disclosure on searchable encryption: Ramification, attack and mitigation,” in NDSS, San Diego, CA, USA, 2012.

[10] D. E. Knuth, “Sorting and searching,” in The art of computer programming, vol. 3, Boston, USA: Addison-Wesley, 1973.

[11] D. Song, D. Wagner, and A. Perrig, “Practical Techniques for Searches on Encrypted Data,” in Proc. of IEEE S&P, DC, USA, 2000, pp. 44-55.

[12] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions,” J. Comput.Secur., vol. 19, no. 5, pp. 895-934, 2011.

[13] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic Searchable Symmetric Encryption,” in Proc. of ACM CCS, Raleigh, NC, USA, 2012, pp. 965–976.

[14] S. Kamara, C. Papamanthou, “Parallel and dynamic searchable symmetric encryption,” Financial Cryptography and Data Security, Springer Berlin Heidelberg, 2013, pp. 258-274.

[15] G. S. Poh, J.–J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, “Searchable Symmetric Encryption: Designs and Challenges,” ACM Comput. Surv. vol. 50, no. 3, pp. 40:1-40:37, 2017.

[16] W. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, “Secure KNN Computation on Encrypted Databases,” in Proc. ACM SIGMOD, Providence, RI, USA, 2009, pp. 139–152.

`[17] H. Hu, J. Xu, C. Ren, and B. Choi, “Processing Private Queries over Untrusted Data Cloud through Privacy Homomorphism,” in Proc. IEEE ICDE, Hannover, NI, GER, 2011, pp. 601–612.

[18] Y. Zhu, Z. Huang, and T. Takagi, “Secure and controllable k-NN query over encrypted cloud data with key confidentiality,” Journal of Parallel & Distributed Computing, vol. 89, pp. 1-12, Mar. 2016.

[19] L. Zhou, Y. Zhu, and A. Castiglione, “Efficient k -NN query over encrypted data in cloud with limited key-disclosure and offline data owner,” Computers & Security, vol. 69, pp. 84-96, Aug. 2017. [20] P. Indyk and R. Motwani, “Approximate nearest neighbours: Towards removing the curse of dimensionality,” In Proc. STOC, Dallas, TX, USA, 1998, pp. 604– 613.