MAULANA AZAD NATIONAL URDU UNIVERSITY HYDERABAD

A Project Report
on

PERSONAL MAIL SECURITY

Submitted in partial fulfillment of the requirements for the award of the degree
Of

MASTER OF COMPUTER APPLICATION

Submitted by:
ASHRAF NOMANI
Roll No: 18MMCA015HY

Guided by:
Asst. Prof. Mr. MOHTASHAM PASHA QADRI

Head of the Department:
Prof. SYED IMTIYAZ HASSAN

Submitted To

SCHOOL OF CS & IT MANUU HYDERABAD

ACKNOLEDGEMENT

Before we get into think of the project report I would like to add a few heartfelt words for the people who were part of this project in numerous ways…people who gave unending support right from the stage the project idea was conceived.

I wish to Acknowledge with a great sense of gratitude to my guide ………………………

Asst. Prof. Mr. MOHTASHAM PASHA QADRI for their valuable guidance, suggestion and correction, I have received from them. They not only encouraged me throughout the work, but I also thank them for reviewing the entire manuscript with painstaking attention for details.

It gives me immense pleasure to acknowledge my indebtedness to the various faculty members of **MAULANA AZAD NATIONAL URDU UNIVERSITY HYDERABAD** for helping me to design the project work as a part of my course curriculum.

ASHRAF NOMANI

SELF CERTIFICATE

This is to certify that the Project Report entitled " **PERSONAL MAIL SECURITY**" is done by me is an authentic

work carried out for the partial fulfillment of the requirements for the award of the degree of

Master of Computer Application under the guidance of ……………………………………

Asst. Prof. Mr. MOHTASHAM PASHA QADRI.

The matter embodied in this project work has not been submitted earlier for award of any degree

or diploma to the best of my knowledge and belief.

Signature of the student

Name of the Student	: ASHRAF NOMANI
Roll No	: 18MMCA015HY

CERTIFICATE

This is to certify that this project entitled "**PERSONAL MAIL SECURITY**" submitted in partial fulfillment of the degree of **MASTER OF COMPUTER APPLICATIONS** from **MAUJLANA AZAD NATIONAL URSU UNIVERSITY HYDERABAD** done by **ASHRAF NOMANI** Roll No. **18MMCA015HY** is an authentic work carried out by him at **MANUU HYDERABAD** Under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

ASHRAF NOMANI　　　　　　Asst. Prof. Mr. MOHTASHAM PASHA QADRI

Signature of the student:- __________　　　　**Signature of the Guide: __________**

Roll No. 18MMCA015HY

Signature of H.O.D.　　　　　　**Signature of Dean**

------------------------------　　　　　　--------------------------

Name & Signature of Class Co-Ordinator

____

Table of contents

ABSTRACT

ABOUT THE PROJECT

Businesses usually adopt a common sense approach when it comes to spending the IT budget with cost being a major influence in purchasing decisions. However, when it comes to choosing a mail server, many businesses seem to pay less attention to the costs and, as a result, end up spending far more money than is necessary.
E-mail travels on the web so they are exposed to the introducers. So privacy of emails may be compromised between senders and receivers side without giving any warning.in todays electronic world email become the backbone of the most organization daily activity . As we know e mail become most frequent in the world so e mail security become most important for the security organization s must control the situation by taking any approach and invest wisely including all the solutions.
Let consider the services provided by the e-mail to the business , email storage and management can be broken down into a number of components like flow of the mail storage of the mail how do we exchange public key , how do we assign trust and how user access the emails these issues are the part of total security agenda .

Today, email is absolutely mission-critical. Communication and collaboration keep your business running. Email and electronically enabled collaboration have become so embedded in normal day-to-day operations that many businesses simply could not function without them. Many businesses, however, have found that the cost of providing employees with the latest in messaging and collaboration technology is rapidly escalating. To meet modern business needs, mail servers have had to become more complex – and with that additional complexity come additional management burdens and costs. Furthermore, some mail servers have an upgrade process that is both extremely complex and extremely costly and which may necessitate the purchase of replacement server hardware. Combined, these factors place a considerable drain on corporate resources. The problem is especially severe for small and medium sized businesses (SMBs) which usually do not have access to the same financial or technical resources as large enterprises.

Email-System Is An Application Program That Sends Electronic Message From One Computer To Another. We will be developing our Project To keep in Mind the Problem of Organization and also We Will Try To Minimize Cost for Organization with secure manner.

Introduction

INTRODUCTION

This project deals with the Mailing System. This project is having different modules like new User creation form named it as a Sign-Up form and already existing user can logged into the Mailing System named it as a Sign In form.

Totally furnished with the security with personal use ,there is a fear of message tempered by the hacker in the time of sending of the message the way or medium of message sending hacker can hack it so in my system we have to encrypt the message first after that sending the message ,hacker can hack the message but this is not a readable form, so this is very useful for personal level security, Here in this message won't be tempered between sender and receiver.

This program provide you privacy as well as high level of security.

Advantages:

.

E-Mails are most frequently used in today's commerce. Now-a-days these are the most convenient way of communication
for ordinary users. E-Mails are public and can be seen by everyone at every point of communication between two users. Hence, because of their exposed nature we can't write sensitive information in ordinary e-mails.
So we can encrypt the message first then send through our personal mail security website .
In which hacker can not be able to decrypt the message so this is highly secured with the help of this user cam send the message confidentially and user have no doubt for hacking or tempering the message so this is very benificial for sending secured message.

Existing System with Limitations:

Presenting Intranet Mailing is manually providing services to employees of departments of an Organization. Employees have to go departments to know some particular information. Some times information is passed by manually between departments. This manual system will take time to pass the information and sometimes it causes lost of information also. This causes lost of employee time also.

There is existing mail system like gmail Hotmail etc where there is no guarantee the message is secured or not, in this we are totally dependent on gmail or Hotmail .there is no guarantee of our message is secured or not.

Proposed System Features:

Now a day the organizations are growing fastly and are increasing in size also. So there organizations are divided into departments. In the fast growing world the information is need as fast as possible. This can be accomplished by passing the information quickly. Quick passing of mails is not possible in load manual systems. Because in manual system the mails are passed through persons from one department to another. But it takes much time and risk also. This leads the inconsistency of information. So we need a system, which is both quick and accurate. This can be achieved by mailing system.

In present organization structure most of the work is done using software applications. In order to improve service for customers we need effective applications. Similarly considering need of work flow we need Personal mail security app for securing our message , This is very good for any company because we can send message in fully secured manner..

SPECIFICATION REQUIREMENT

SPECIFICATION REQUIREMENT

Requirement analysis for web applications encompasses three major tasks: formulation, requirements gathering and analysis modeling. During formulation, the basic motivation and goals for the web application are identified, and the categories of users are defined. In the requirements gathering phase, the content and functional requirements are listed and interaction scenarios written from end-user's point-of-view are developed. This intent is to establish a basic understanding of why the web application is built, who will use it, and what problems it will solve for its users.

Software requirement Specification:

A set of programs associated with the operation of a computer is called software. Software is the part of the computer system, which enables the user to interact with several physical hardware devices.

The minimum software requirement specifications for developing this project are as follows:

Operating System	:	**Window 2000, XP,Win 7,Win 8.**
Presentation layer	:	**Java, JSP.**
Database	:	**Mysql**
Documentation Tool	:	**Ms Office**

Hardware Requirement Specification:

The collection of internal electronic circuits and external physical devices used in building a computer is called the Hardware. The minimum hardware requirement specifications for developing this project are as follows:

Processor	**:** Standard processor with a speed of 1.6 GHz or more
RAM	**:** 256 MB RAM or more
Hard Disk	**:** 20 GB or more
Monitor	**:** Standard color monitor

TECHNOLOGIES

USED

TECHNOLOGIES USED

Java:-

Java is a programming language and platform and java is a high level robust ,secured and Object oriented programming language.

The java programming language was first developed in 1990 by an engineer at sun-microsystem named james Gosling.He was unhappy using the c++ programming language , so he created an new programming language that he named Oak, consequently sun named the language java simply because that named sound cool and made it freely available in 1995.This language is used in backend..

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be **Write Once, Run Anywhere.**

Java is −

- **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

- **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed** − Java is designed for the distributed environment of the internet.
- **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

History of Java

James Gosling initiated Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called ‘Oak’ after an oak tree that stood outside Gosling's office, also went by the name ‘Green’ and ended up later being renamed as Java, from a list of random words.

Sun released the first public implementation as Java 1.0 in 1995. It promised **Write Once, Run Anywhere** (WORA), providing no-cost run-times on popular platforms.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

Tools You Will Need

For performing the examples discussed in this tutorial, you will need a Pentium 200-MHz computer with a minimum of 64 MB of RAM (128 MB of RAM recommended).

You will also need the following softwares −

- Linux 7.1 or Windows xp/7/8 operating system
- Java JDK 8
- Microsoft Notepad or any other text editor

This tutorial will provide the necessary skills to create GUI, networking, and web applications using Java.

JSP (Java server Pages) :-

- It stands for **Java Server Pages**.
- It is a server side technology.
- It is used for creating web application.
- It is used to create dynamic web content.
- In this JSP tags are used to insert JAVA code into HTML pages.
- It is an advanced version of Servlet Technology.
- It is a Web based technology helps us to create dynamic and platform independent web pages.
- In this, Java code can be inserted in HTML/ XML pages or both.
- JSP is first converted into servlet by JSP container before processing the client's request.

JSP pages are more advantageous than Servlet:

- They are easy to maintain.
- No recompilation or redeployment is required.
- JSP has access to entire API of JAVA .
- JSP are extended version of Servlet.

Features of JSP

- **Coding in JSP is easy** :- As it is just adding JAVA code to HTML/XML.
- **Reduction in the length of Code** :- In JSP we use action tags, custom tags etc.

- **Connection to Database is easier** :-It is easier to connect website to database and allows to read or write data easily to the database.
- **Make Interactive websites** :- In this we can create dynamic web pages which helps user to interact in real time environment.
- **Portable, Powerful, flexible and easy to maintain** :- as these are browser and server independent.
- **No Redeployment and No Re-Compilation** :- It is dynamic, secure and platform independent so no need to re-compilation.
- **Extension to Servlet** :- as it has all features of servlets, implicit objects and custom tags

JSP syntax

Syntax available in JSP are following

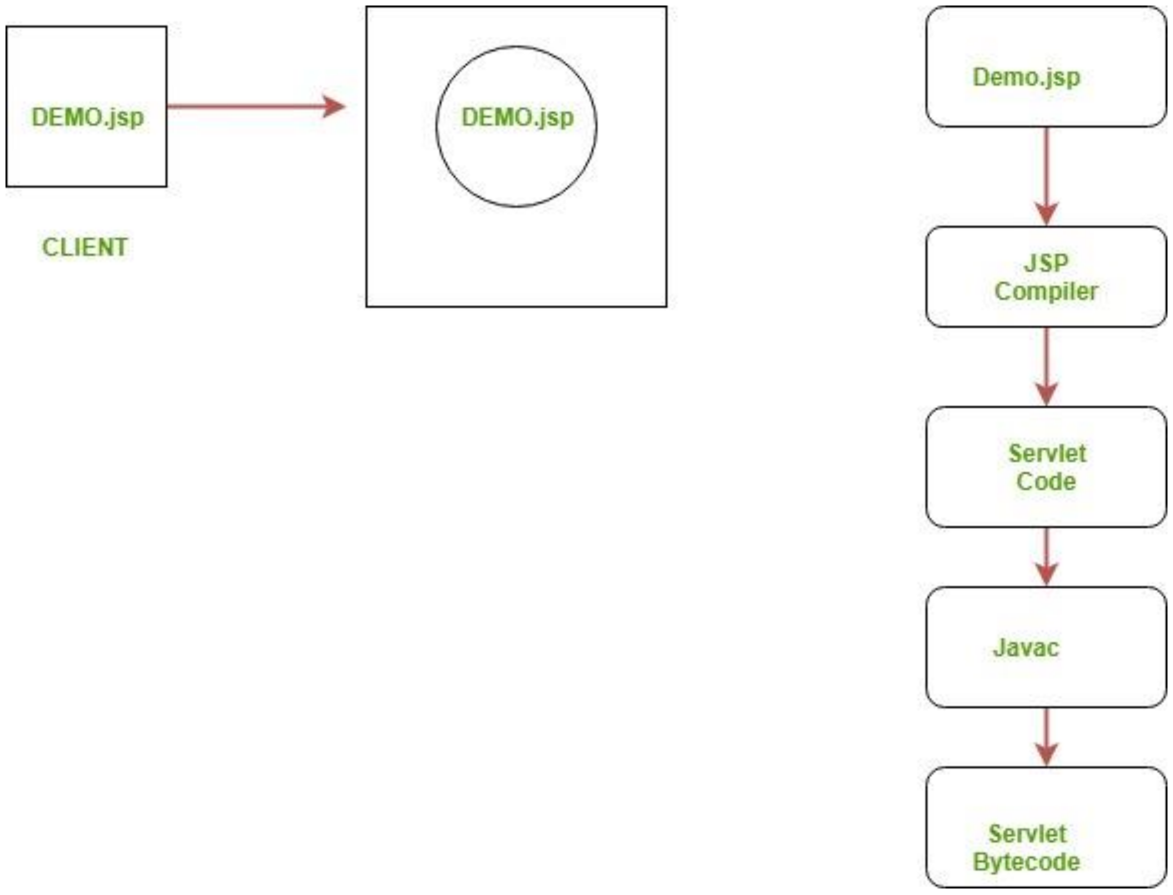
1. **Declaration Tag** :-It is used to declare variables.
2. **Syntax:-**
3. `<%!  Dec var  %>`
4. **Example:-**
5. `<%! int var=10; %>`

6. **Java Scriplets** :- It allows us to add any number of JAVA code, variables and expressions.
7. **Syntax:-**
8. `<% java code %>`
9. **JSP Expression** :- It evaluates and convert the expression to a string.
10. **Syntax:-**
11. `<%= expression %>`
12. **Example:-**
13. `<% num1 = num1+num2 %>`
14. **JAVA Comments** :- It contains the text that is added for information which has to be ignored.
15. **Syntax:-**
16. `<% -- JSP Comments %>`

Process of Execution

Steps for Execution of JSP are following:-

- Create html page from where request will be sent to server eg try.html.
- To handle to request of user next is to create .jsp file Eg. new.jsp
- Create project folder structure.
- Create XML file eg my.xml.
- Create WAR file.
- Start Tomcat
- Run Application



Example of Hello World

We will make one .html file and .jsp file

```
demo.jsp
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
```

```
<title>Hello World - JSP tutorial</title>
</head>
<body>
    <%= "Hello World!" %>
</body>
</html>
```

Advantages of using JSP

- It does not require advanced knowledge of JAVA
- It is capable of handling exceptions
- Easy to use and learn
- It can tags which are easy to use and understand
- Implicit objects are there which reduces the length of code
- It is suitable for both JAVA and non JAVA programmer

Disadvantages of using JSP

- Difficult to debug for errors.
- First time access leads to wastage of time
- It’s output is HTML which lacks features.

Java mail API:-

In any software application, sending and receiving electronic messages, more specifically e-mails are an essential part. Emails are a medium of communication between different parties who are using the application. In most of the standard programming languages, email APIs are available for communication, and Java is also not an exception. Java provides e-mail APIs which are platform and protocol independent. The mail management framework consists of various abstract classes for defining an e-mail communication system.

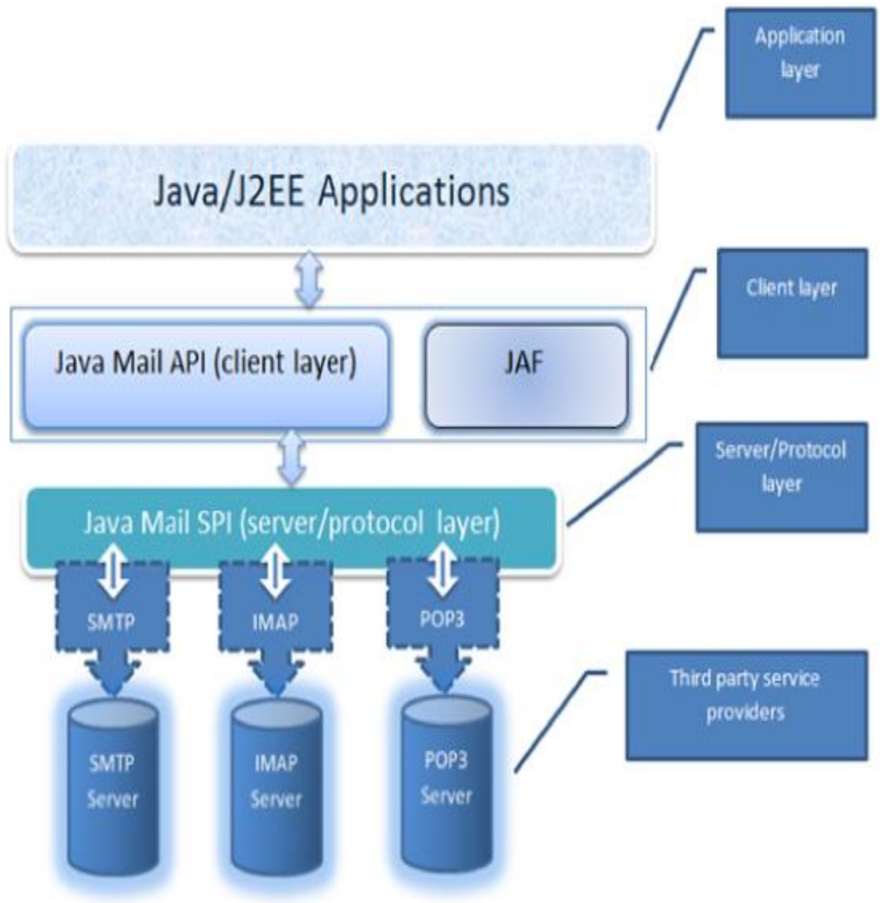
In this article, we will discuss about the Java E-mail management framework and its important components. We will also work with some coding examples.

JavaMail architecture overview Java mail API comes as a default package with java EE platform, and it is optional for java SE platform. Java mail framework is composed of multiple components. JavaMail API is one such component used by the developers to build mail applications. But, these APIs are just a layer between the Java application (mail enabled) and the protocol service providers.

Let us have a look at different layers of Java mail architecture.

- **Java Mail APIs:** These are the Java interfaces to send and receive e-mails. This layer is completely independent of the underlying protocols.
- **JavaBeans Activation Framework (JAF):** This framework is used to manage mail contents like URL, attachments, mail extensions etc.
- **Service Provider Interfaces:** This layer sits between the protocol implementers and the Java applications, more specifically Java mail APIs. SPIs understand the protocol languages and hence create the bridge between the two sides.
- **Service protocol implementers:** These are the third party service providers who implements different protocols like *SMTP*,*POP3* and *IMAP* etc. In this context we must have some ideas on the following protocols.
    - **SMTP**– *S*imple *M*ail *T*ransfer *P*rotocol is used for sending e-mails.
    - **POP3–** *P*ost *O*ffice *P*rotocol is used to receive e-mails. It provides one to one mapping for users and mail boxes, which is one mail box for one user.
    - **IMAP–** *I*nternet *M*essage *A*ccess *P*rotocol is also used to receive e-mails. It supports multiple mail boxes for single user.
    - **MIME–** *M*ultipurpose *I*nternet *M*ail *E*xtensions is a protocol to define the transferred content.

Following is the Java mail architecture diagram. There are mainly four layers in the system. The top layer is the Java application layer. The second layer is the client API layer along with JAF. Third layer contains server and protocol implementers. And, the bottom layer is the third party service providers.



Environment setup Before we start working on the code examples, let us complete the environment setup first. For Java mails, we need to download some JAR files and add them in the CLASSPATH. Following are the two which needs to be installed in your system.

- Download JavaMail API and complete the installation
- Download JavaBeans Activation Framework (JAF) and install in your system
- Add the mail.jar and activation.jar files in your CLASSPATH
- Install any SMTP server for sending emails. In our example we will be using Jango SMTP.

Now the environment setup is complete and we will jump into the coding part.

How to send and receive e-mails?

Send Email: In our first example we will check how an email can be sent by using Java mail API and SMTP server. Following are the steps to be followed.

- Setup ‘From’ and ‘To’ address along with the user id and password.
- Setup SMTP host
- Setup properties values
- Create session object
- Form the message details
- Send the message by using Transport object.

This example followed the above steps.

Listing1: Sample code to send emails

```
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;

public class DemoSendEmail {

  public static void main(String[] args) {

   //Declare recipient's & sender's e-mail id.

   String destmailid = "destemail@eduonix.com";

   String sendrmailid = "frmemail@eduonix.com";

   //Mention user name and password as per your configurat
ion

   final String uname = "username";

   final String pwd = "password";
   //We are using relay.jangosmtp.net for sending emails

   String smtphost = "relay.jangosmtp.net";

    //Set properties and their values

   Properties propvls = new Properties();

   propvls.put("mail.smtp.auth", "true");

   propvls.put("mail.smtp.starttls.enable", "true");

   propvls.put("mail.smtp.host", smtphost);

   propvls.put("mail.smtp.port", "25");

   //Create a Session object & authenticate uid and pwd

   Session sessionobj = Session.getInstance(propvls,

     new javax.mail.Authenticator() {

     protected PasswordAuthentication getPasswordAuthentic
ation() {

       return new PasswordAuthentication(uname, pwd);

   }

     });

   try {

   //Create MimeMessage object & set values

  Message messageobj = new MimeMessage(sessionobj);

  messageobj.setFrom(new InternetAddress(sendrmailid));

  messageobj.setRecipients(Message.RecipientType.TO,Intern
etAddress.parse(destmailid));

  messageobj.setSubject("This is test Subject");
  messageobj.setText("Checking sending emails by using Jav
aMail APIs");

 //Now send the message

  Transport.send(messageobj);

  System.out.println("Your email sent successfully....");

   } catch (MessagingException exp) {

     throw new RuntimeException(exp);

   }

  }

}
```

After compilation and running the application you will get the following output.

Your email sent successfully….

Receive Email: Now in the second example we will check how to receive emails by using Java Mail APIs.

Please follow the steps below to complete the application

- Set up properties values for POP3 server
- Create session object
- Create POP3 store object and connect
- Create folder object and open it
- Retrieve email messages and print them in a loop
- Close folder and store object

Following code sample follows the steps described above.

Listing2: Sample code to receive emails

```
import java.util.Properties;

import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.NoSuchProviderException;
import javax.mail.Session;
import javax.mail.Store;
public class DemoCheckEmail{
  public static void main(String[] args) {
   //Set mail properties and configure accordingly
   String hostval = "pop.gmail.com";
   String mailStrProt = "pop3";
   String uname = "uname@gmail.com";
   String pwd = "password";
  // Calling checkMail method to check received emails
   checkMail(hostval, mailStrProt, uname, pwd);
   }
  public static void checkMail(String hostval, String mail
StrProt, String uname,String pwd)
   {
   try {
   //Set property values
   Properties propvals = new Properties();
   propvals.put("mail.pop3.host", hostval);
   propvals.put("mail.pop3.port", "995");
   propvals.put("mail.pop3.starttls.enable", "true");
   Session emailSessionObj = Session.getDefaultInstance(pr
opvals);
   //Create POP3 store object and connect with the server
   Store storeObj = emailSessionObj.getStore("pop3s");
   storeObj.connect(hostval, uname, pwd);
   //Create folder object and open it in read-only mode
   Folder emailFolderObj = storeObj.getFolder("INBOX");
   emailFolderObj.open(Folder.READ_ONLY);
   //Fetch messages from the folder and print in a loop
   Message[] messageobjs = emailFolderObj.getMessages();
   for (int i = 0, n = messageobjs.length; i < n; i++) {
     Message indvidualmsg = messageobjs[i];
     System.out.println("Printing individual messages");
     System.out.println("No# " + (i + 1));
     System.out.println("Email Subject: " + indvidualmsg.ge
tSubject());
     System.out.println("Sender: " + indvidualmsg.getFrom()
[0]);
     System.out.println("Content: " + indvidualmsg.getConte
nt().toString());
   }
   //Now close all the objects
   emailFolderObj.close(false);
   storeObj.close();
   } catch (NoSuchProviderException exp) {
     exp.printStackTrace();
   } catch (MessagingException exp) {
     exp.printStackTrace();
   } catch (Exception exp) {
     exp.printStackTrace();
   }
   }
```

JDBC (Java Data Base Connectivity)

Introduction

JDBC or Java Database Connectivity is a specification from Sun microsystems that provides a standard abstraction(that is API or Protocol) for java applications to communicate with various databases. It provides the language with java database connectivity standard. It is used to write programs required to access databases. JDBC along with the database driver is capable of accessing databases and spreadsheets. The enterprise data stored in a relational database(RDB) can be accessed with the help of JDBC APIs.

Definition of JDBC(Java Database Connectivity)

JDBC is an API(Application programming interface) which is used in java programming to interact with databases.

The classes and interfaces of JDBC allows application to send request made by users to the specified database.

Purpose of JDBC

Enterprise applications that are created using the JAVA EE technology need to interact with databases to store application-specific information. So, interacting with a database requires efficient database connectivity which can be achieved by using the ODBC(Open database connectivity) driver. This driver is used with

JDBC to interact or communicate with various kinds of databases such as Oracle, MS Access, Mysql and SQL server database.

Components of JDBC

There are generally four main components of JDBC through which it can interact with a database. They are as mentioned below:

1. **JDBC API:** It provides various methods and interfaces for easy communication with the database.It provides two packages as follows which contains the java SE and java EE platforms to exhibit WORA(write once run everywhere) capabilities.
2. `1. java.sql.*;`
3. `2. javax.sql.*;`

   It also provides a standard to connect a database to a client application.
4. JDBC Driver manager**:** It loads database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.
5. **JDBC Test suite:** It is used to test the operation(such as insertion, deletion, updation) being performed by JDBC Drivers.
6. **JDBC-ODBC Bridge Drivers**: It connects database drivers to the database.This bridge translates JDBC method call to the ODBC function call.It makes the use of
   `sun.jdbc.odbc`

   package that includes native library to access ODBC characteristics.

Description:

1. **Application:** It is a java applet or a servlet which communicates with a data source.
2. **The JDBC API:** The JDBC API allows Java programs to execute SQL statements and retrieve results. Some of the important classes and interfaces defined in JDBC API are as follows:

DriverManager
Driver
Connection
Statement
PreparedStatement
CallableStatement
ResultSet
SQL data

3. **DriverManager:** It plays an important role in the JDBC architecture.It uses some database-specific drivers to effectively connect enterprise applications to databases.
4. **JDBC drivers:** To communicate with a data source through JDBC, you need a JDBC driver that intelligently communicates with the respective data source.

My Sql Data Base:-

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as **Foreign Keys**.

A **Relational DataBase Management System (RDBMS)** is a software that −

- Enables you to implement a database with tables, columns and indexes.

- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology

Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

- **Database** − A database is a collection of tables, with related data.
- **Table** − A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column** − One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row** − A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- **Redundancy** − Storing data twice, redundantly to make the system faster.
- **Primary Key** − A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
- **Foreign Key** − A foreign key is the linking pin between two tables.
- **Compound Key** − A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index** − An index in a database resembles an index at the back of a book.
- **Referential Integrity** − Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.

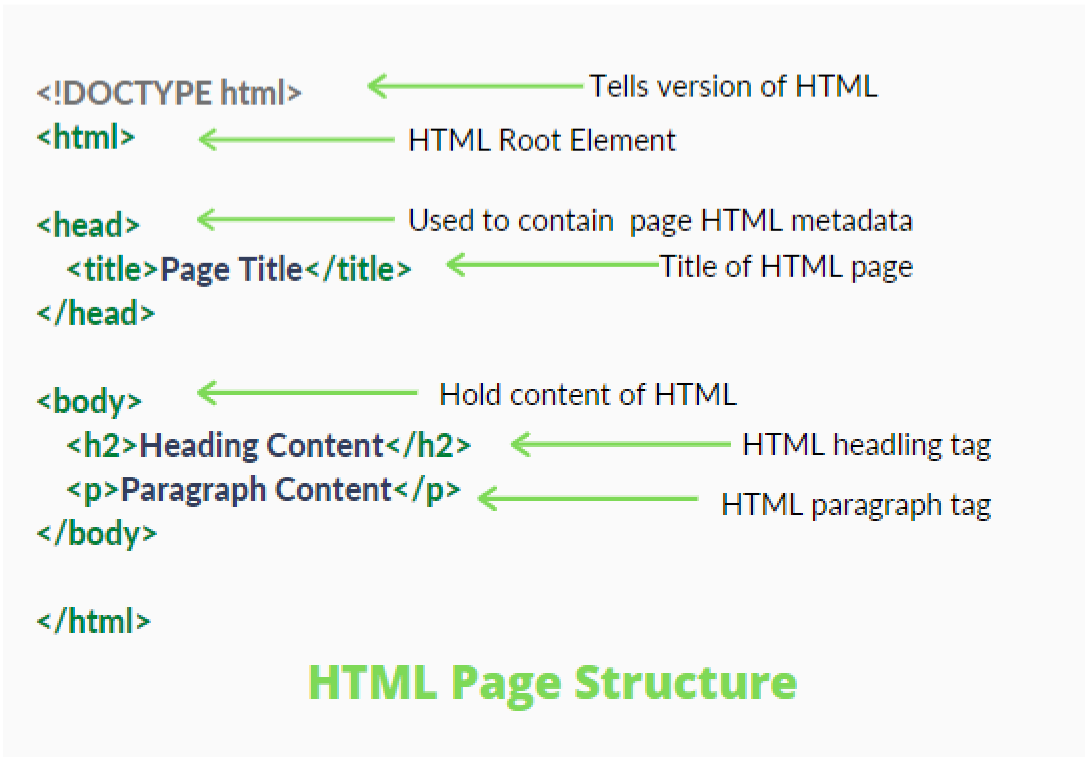
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

Front End HTML CSS:-

HTML stands for HyperText Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text. HTML is a markup language used by the browser to manipulate text, images, and other content, in order to display it in the required format. HTML was created by Tim Berners-Lee in 1991. The first-ever version of HTML was HTML 1.0, but the first standard version was HTML 2.0, published in 1999.

Elements and Tags: HTML uses predefined tags and elements which tell the browser how to properly display the content. Remember to include closing tags. If omitted, the browser applies the effect of the opening tag until the end.

HTML page structure: The basic structure of an HTML page is laid out below. It contains the essential building-block elements (i.e. doctype declaration, HTML, head, title, and body elements) upon which all web pages are created.



```
<!DOCTYPE html>
<html>

<head>
  <title>Page Title</title>
</head>

<body>
  <h2>Heading Content</h2>
  <p>Paragraph Content</p>
</body>

</html>
```

<DOCTYPE! html>: This is the document type declaration (not technically a tag). It declares a document as being an HTML document. The doctype declaration is not case-sensitive.
<html>: This is called the HTML root element. All other elements are contained within it.
<head>: The head tag contains the “behind the scenes” elements for a webpage. Elements within the head aren’t visible on the front-end of a webpage. HTML elements used inside the <head> element include:

- <style>
- <title>
- <base>
- <noscript>
- <script>
- <meta>
- <link>

<body>: the body tag is used to enclose all the visible content of a webpage. In other words, the body content is what the browser will show on the front-end.
An HTML document can be created using any text editor. Save the text file using **.html** or **.htm**. Once saved as an HTML document, the file can be opened as a webpage in the browser.

NOTE: Basic/built-in text editors are Notepad (Windows) and TextEdit (Macs). Basic text editors are entirely sufficient for when you're just getting started. As you progress, there are many feature-rich text editors available which allow for greater function and flexibility.

Here's an example of an HTML webpage:

- html

```
<!DOCTYPE html>

<html>



<head>

    <title>Demo Web Page</title>

</head>



<body>

    <h1>ASHRAF NOMANI</h1>



<p>A computer science portal for geeks</p>



</body>
```

```
</html>
```

Output: ASHRAF NOMANI

Features of HTML:
- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to text.
- It is a markup language.

Why learn HTML?
- It is a simple markup language. Its implementation is easy.
- It is used to create a website.
- Helps in developing fundamentals about web programming.
- Boost professional career.

Advantages:
- HTML is used to build websites.
- It is supported by all browsers.
- It can be integrated with other languages like CSS, JavaScript, etc.

Disadvantages:
- HTML can only create static webpages. For dynamic webpages, other languages have to be used.
- A large amount of code has to be written to create a simple web page.
- The security feature is not good.

CSS : -

This definition explains the meaning of CSS (cascading style sheets) and how using them with HTML pages is a user interface (UI) development best practice that complies with the separation of concerns design pattern.

CSS is the standard and preferred mechanism for formatting HTML pages.

Conforming with the separation of concerns design pattern and best practice, cascading style sheets provide a central location in which information about what various fonts, foreground colors, background colors, italicization and emphasization should be applied to various HTML elements within a webpage. Cascading style sheets can also control how various parts of a page, such as the header, footer, body, article content, sections and asides, are laid out on the page. This is extremely helpful when content must be laid out in a dramatically different fashion depending upon whether it is being viewed on a desktop, tablet or a smartphone.

Proper use of CSS

In the early days of the World Wide Web (WWW), it was common for HTML files to include not only markup language and content, but formatting information and JavaScript as well. This made webpages difficult to write, difficult to read, difficult to update and difficult to maintain. As the web matured, it became a best practice to divide HTML, scripting content and style information into separate, easy-to-maintain files. As such, a modern webpage is typically made up of three separate entities: a cascading style sheet, a JavaScript file and the HTML file itself.

Implementing CSS formatting

The cascading nature of CSS files is attributed to the fact that style information for a webpage can be defined in any of three different places, also known as *style levels*.

The preferred practice is to put style information in a separate file with a .css extension. Using formatting information contained within an external cascading style sheet is accomplished via the HTML link tag. A webpage can link to zero, one or may different external CSS files by using multiple link tags.

```
<link rel="stylesheet" type="text/css" href="what-is-
css.css">
```

However, on smaller projects or in cases where a given webpage is interested in overriding some of the style information in an external CSS file, style information can be written within a `<style>` tag inside the webpage. This is known as an *internal* *style level*. Internal style level information within a webpage will override any style information provided by an external cascading style sheet.

Cascading style rules

Furthermore, all HTML5 tags have a style property that one can use to override any style information defined at either the page style level or in an external style sheet. Using an HTML tag to define CSS information is referred to as an *inline style*. The fact that style rules dictate that parent-level styles are overridden by page-level styles and page-level styles are overridden by tag-level styles is what is meant by style sheets being *cascading*.

Style sheet language

CSS syntax is relatively simple. The name of the element to style, referred to as the *CSS selector*, is followed by braces, within which various attributes, such as `font-size` and `background-color` are assigned values. The World Wide Web Consortium standards organization defines the CSS attributes, although various browsers may offer supplemental support by defining their own custom fields. This is often done for proposed attributes that are expected to be included in future CSS releases.

CSS selectors

CSS selectors can be HTML tags, class attributes assigned to HTML tags and even states of a given element, such as the disabled state of an input field or the hover state of an anchor link. Making it possible to customize the style of components depending upon their state or how they are classified on a page provides the graphic designer a great deal of flexibility in determining how a webpage will be rendered by the browser.

CRYPTOGRAPHY:-

Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. The prefix "crypt-" means "hidden" or "vault" -- and the suffix "-graphy" stands for "writing."

In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms, to transform messages in ways that are hard to decipher. These deterministic algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet, and confidential communications such as credit card transactions and email.

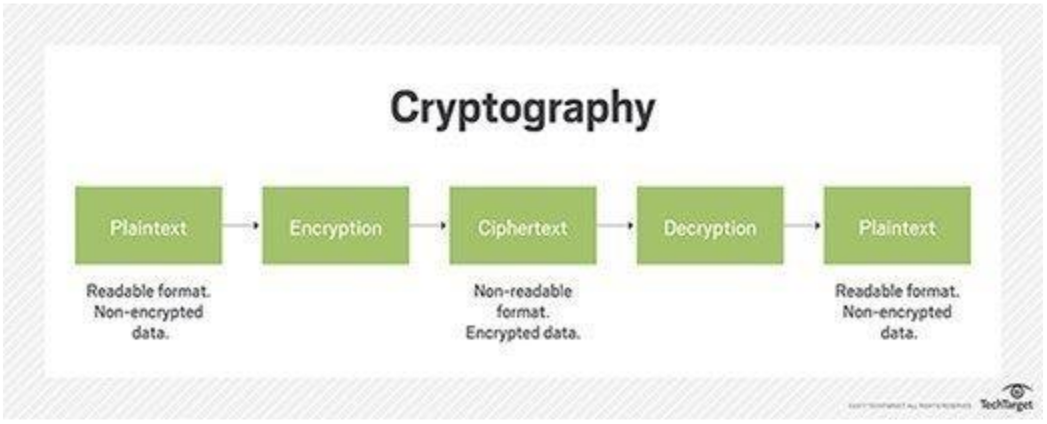
Cryptography techniques

Cryptography is closely related to the disciplines of cryptology and cryptanalysis. It includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. However, in today's computer-centric world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as cleartext) into ciphertext (a process called encryption), then back again (known as decryption). Individuals who practice this field are known as cryptographers.

Modern cryptography concerns itself with the following four objectives:

1. **Confidentiality**: the information cannot be understood by anyone for whom it was unintended
2. **Integrity:** the information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected
3. **Non-repudiation**: the creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information
4. **Authentication**: the sender and receiver can confirm each other's identity and the origin/destination of the information

Procedures and protocols that meet some or all of the above criteria are known as cryptosystems. Cryptosystems are often thought to refer only to mathematical procedures and computer programs; however, they also include the regulation of human behavior, such as choosing hard-to-guess passwords, logging off unused systems, and not discussing sensitive procedures with outsiders.


Cryptography
Plaintext
Encryption
Ciphertext
Decryption
Plaintext
Readable format. Non-encrypted data.
Non-readable format. Encrypted data.
Readable format. Non-encrypted data.
TechTarget

Cryptographic algorithms

Cryptosystems use a set of procedures known as cryptographic algorithms, or ciphers, to encrypt and decrypt messages to secure communications among computer systems, devices such as smartphones, and applications. A cipher suite uses one algorithm for encryption, another algorithm for message authentication, and another for key exchange. This process, embedded in protocols and written in software that runs on operating systems and networked computer systems, involves public and private key generation for data encryption/decryption, digital signing and verification for message authentication, and key exchange.

Types of cryptography

Single-key or symmetric-key encryption algorithms create a fixed length of bits known as a block cipher with a secret key that the creator/sender uses to encipher data (encryption) and the receiver uses to decipher it. Types of symmetric-key cryptography include the Advanced Encryption Standard (AES), a specification established in November 2001 by the National Institute of Standards and Technology as a Federal Information Processing Standard (FIPS 197), to protect sensitive information. The standard is mandated by the U.S. government and widely used in the private sector.

In June 2003, AES was approved by the U.S. government for classified information. It is a royalty-free specification implemented in software and hardware worldwide. AES is the successor to the Data Encryption Standard (DES) and DES3. It uses longer key lengths (128-bit, 192-bit, 256-bit) to prevent brute force and other attacks.

Public-key or asymmetric-key encryption algorithms use a pair of keys, a public key associated with the creator/sender for encrypting messages and a private key that only the originator knows (unless it is exposed or they decide to share it) for decrypting that information. The types of public-key cryptography include RSA, used widely on the internet; Elliptic Curve Digital Signature Algorithm (ECDSA) used by Bitcoin; Digital Signature Algorithm (DSA) adopted as a Federal Information Processing Standard for digital signatures by NIST in FIPS 186-4; and Diffie-Hellman key exchange.

To maintain data integrity in cryptography, hash functions, which return a deterministic output from an input value, are used to map data to a fixed data size. Types of cryptographic hash functions include SHA-1 (Secure Hash Algorithm 1), SHA-2 and SHA-3.

Cryptography concerns

Attackers can bypass cryptography, hack into computers that are responsible for data encryption and decryption, and exploit weak implementations, such as the use of default keys. However, cryptography makes it harder for attackers to access messages and data protected by encryption algorithms.

Growing concerns about the processing power of quantum computing to break current cryptography encryption standards led the National Institute of Standards and Technology (NIST) to put out a call for papers among the mathematical and science community in 2016 for new public key cryptography standards. Unlike today's computer systems, quantum computing uses quantum bits (qubits) that can represent both 0s and 1s, and therefore perform two calculations at once. While a large-scale quantum computer may not be built in the next decade, the existing infrastructure requires standardization of publicly known and understood algorithms that offer a secure approach, according to NIST. The deadline for submissions was in November 2017, analysis of the proposals is expected to take three to five years.

History of cryptography

The word "cryptography" is derived from the Greek *kryptos*, meaning hidden. The origin of cryptography is usually dated from about 2000 B.C., with the Egyptian practice of hieroglyphics. These consisted of complex pictograms, the full meaning of which was only known to an elite few. The first known use of a modern cipher was by Julius Caesar (100 B.C. to 44 B.C.), who did not trust his messengers when communicating with his governors and officers. For this reason, he created a system in which each character in his messages was replaced by a character three positions ahead of it in the Roman alphabet.

In recent times, cryptography has turned into a battleground of some of the world's best mathematicians and computer scientists. The ability to securely store and transfer sensitive information has proved a critical factor in success in war and business.

Because governments do not wish certain entities in and out of their countries to have access to ways to receive and send hidden information that may be a threat to national interests, cryptography has been subject to various restrictions in many countries, ranging from limitations of the usage and export of software to the public dissemination of mathematical concepts that could be used to develop cryptosystems. However, the internet has allowed the spread of powerful programs and, more importantly, the underlying techniques of cryptography, so that today many of the most advanced cryptosystems and ideas are now in the public domain.

AES(Advancced Encryption Standard):

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

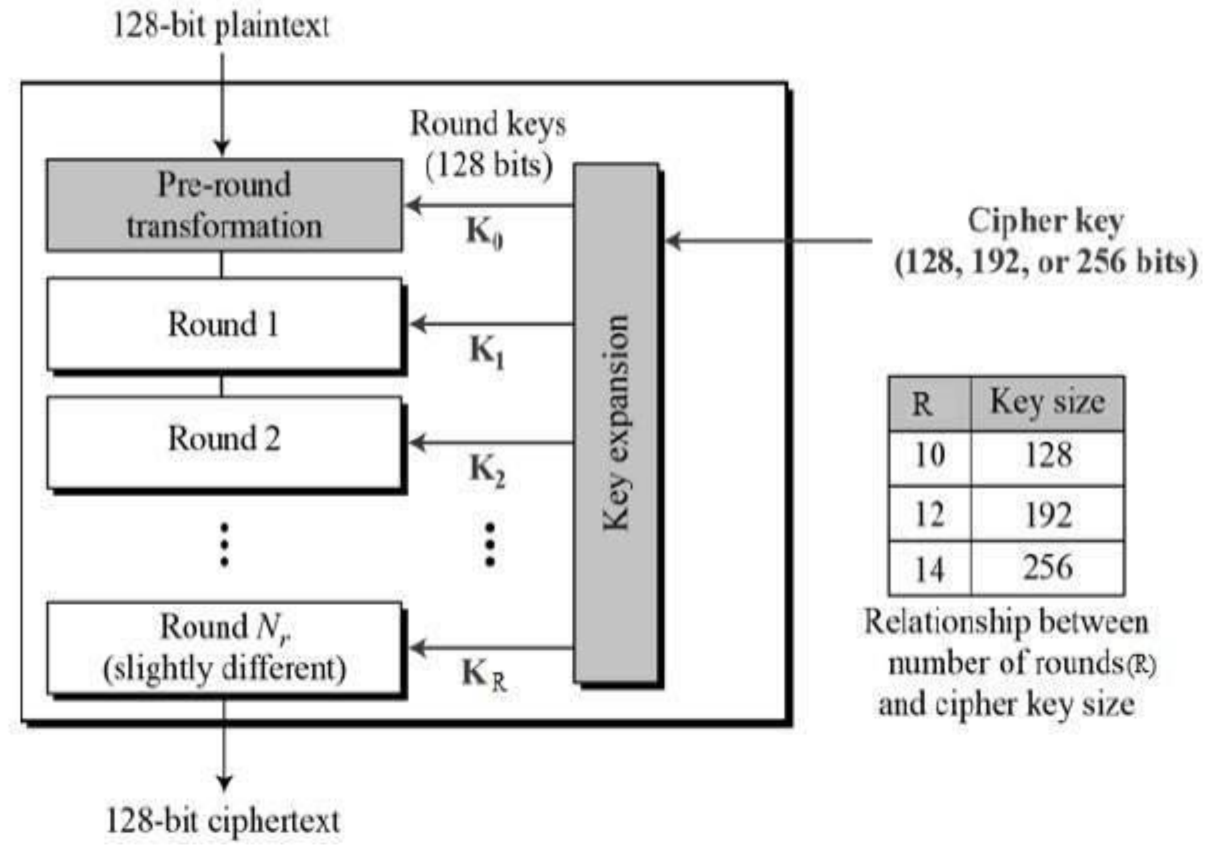
Operation of AES

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

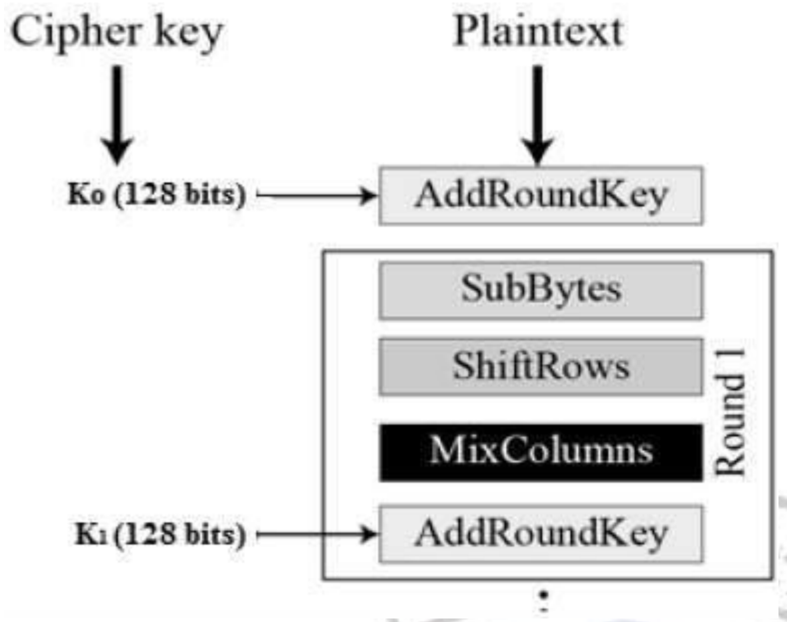
The schematic of AES structure is given in the following illustration −



R	Key size
10	128
12	192
14	256

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −



Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows −

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

AES Analysis

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES has been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of ‘future-proofing’ against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.

Tomcat:-

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process.

Tomcat 5 implements the Servlet 2.4 and Java Server Pages 2.0 specifications and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

Directories and files:

$CATALINA_HOME represents the root of your Tomcat installation. When we say, "This information can be found in your $CATALINA_HOME/README.txt file" we mean to look at the README.txt file at the root of your Tomcat install.

These are some of the key tomcat directories, all relative to **$CATALINA_HOME**:

- **/bin** - Startup, shutdown, and other scripts. The *.sh files (for Unix systems) are functional duplicates of the *.bat files (for Windows systems). Since the Win32 command-line lacks certain functionality, there are some additional files in here.
- **/conf** - Configuration files and related DTDs. The most important file in here is server.xml. It is the main configuration file for the container.
- **/logs** - Log files are here by default.
- **/webapps** - This is where your webapps go.

INSTALLING TOMCAT

Installing Tomcat on Windows can be done easily using the Windows installer. **Installation as a service**: Tomcat will be installed as a Windows NT/2k/XP service no matter what setting is selected. Using the checkbox on the component page sets the service as "auto" startup, so that Tomcat is automatically startup when Windows starts. For optimal security, the service should be affected a separate user, with reduced permissions (see the Windows Services administration tool and its documentation).

- **Java location**: The installer will use the registry or the JAVA_HOME environment variable to determine the base path of the JDK or a JRE. If only a JRE (or an incorrect path) is specified, Tomcat will run but will be unable to compile JSP pages at runtime. Either all webapps will need to be precompiled (this can be easily done using the Tomcat deployed), or the lib\tools. Jar file from a JDK installation must be copied to the common\lib path of the Tomcat installation.

Architecture Overview

Server: In the Tomcat world, a Server represents the whole container. A **Server** element represents the entire Catalina servlet container. Therefore, it must be the single outermost element in the conf/server.xml configuration file. Its attributes represent the characteristics of the servlet container as a whole. Tomcat provides a default implementation of the Server interface, and this is rarely customized by users.

Engine
An Engine represents request processing pipeline for a specific Service. As a Service may have multiple Connectors, the Engine received and processes all requests from these connectors, handing the response back to the appropriate connector for transmission to the

client.

Host
A Host is an association of a network name, e.g. www.yourcompany.com, to the Tomcat server. An Engine may contain multiple hosts, and the Host element also supports network aliases such as yourcompany.com and abc.yourcompany.com. Users rarely create custom Hosts because the Standard Host implementation provides significant additional functionality.

Connector
A Connector handles communications with the client. There are multiple connectors available with Tomcat, all of which implement the Connector interface These include the Coyote connector which is used for most HTTP traffic, especially when running Tomcat as a standalone server, and the JK2 connector which implements the AJP protocol used when connecting Tomcat to an Apache HTTPD server. Creating a customized connector

is a significant effort.

Context

A Context represents a web application. A Host may contain multiple contexts, each with a unique path. The Context interface may be implemented to create custom Contexts, but this is rarely the case because the Standard Context provides significant additional functionality.

Service

A Service is an intermediate component which lives inside a Server and ties one or more Connectors to exactly one Engine. The Service element is rarely customized by users, as the default implementation is simple and sufficient: service interface.

BEHAVIORAL DESCRIPTION

BEHAVIORAL DESCRIPTION

Data Flow:

There are 2 types of Dfd's they are

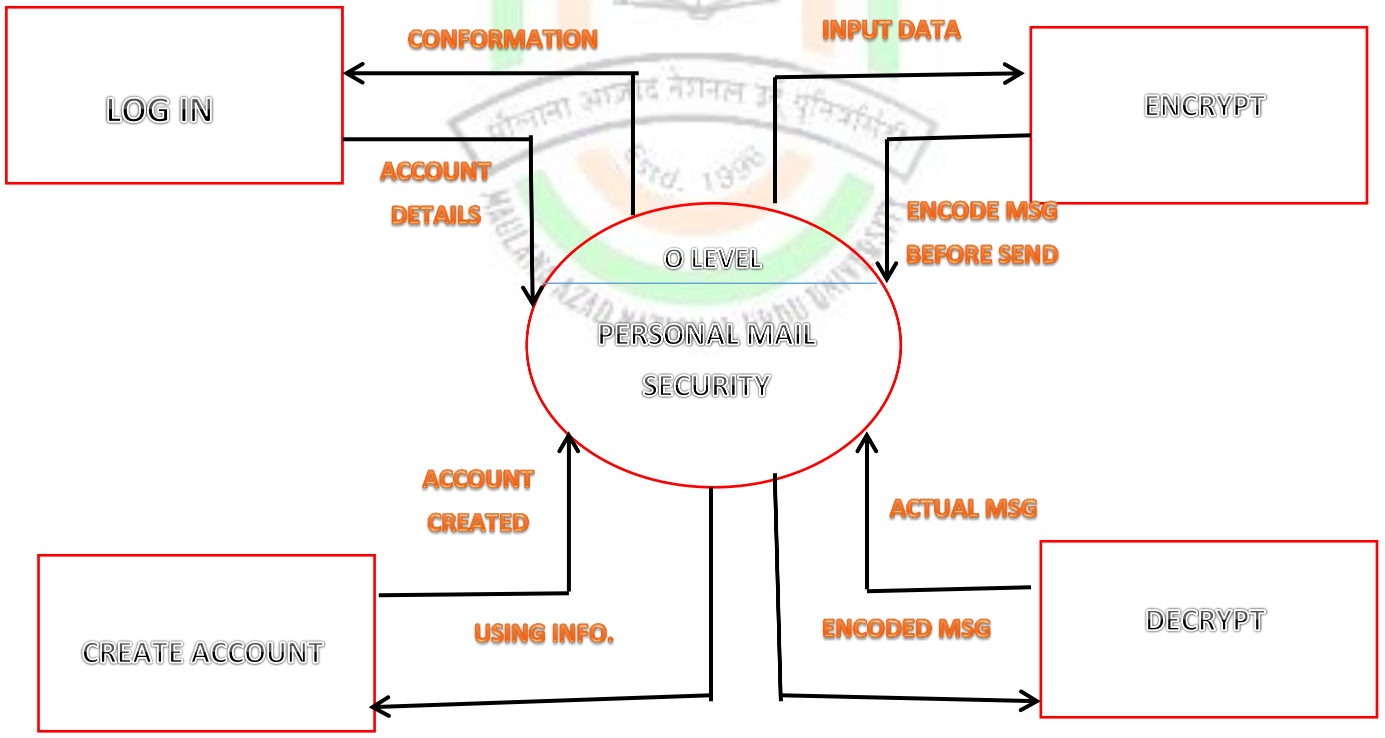
1. Context Level DFD/ O LEVEL DFD
2. Top Level DFD

Context Level DFD:

In the Context Level the whole system is shown as a single process.

- No data stores are shown.
- Inputs to the overall system are shown together with data sources (as External entities).
- Outputs from the overall system are shown together with their destinations (as External entities).

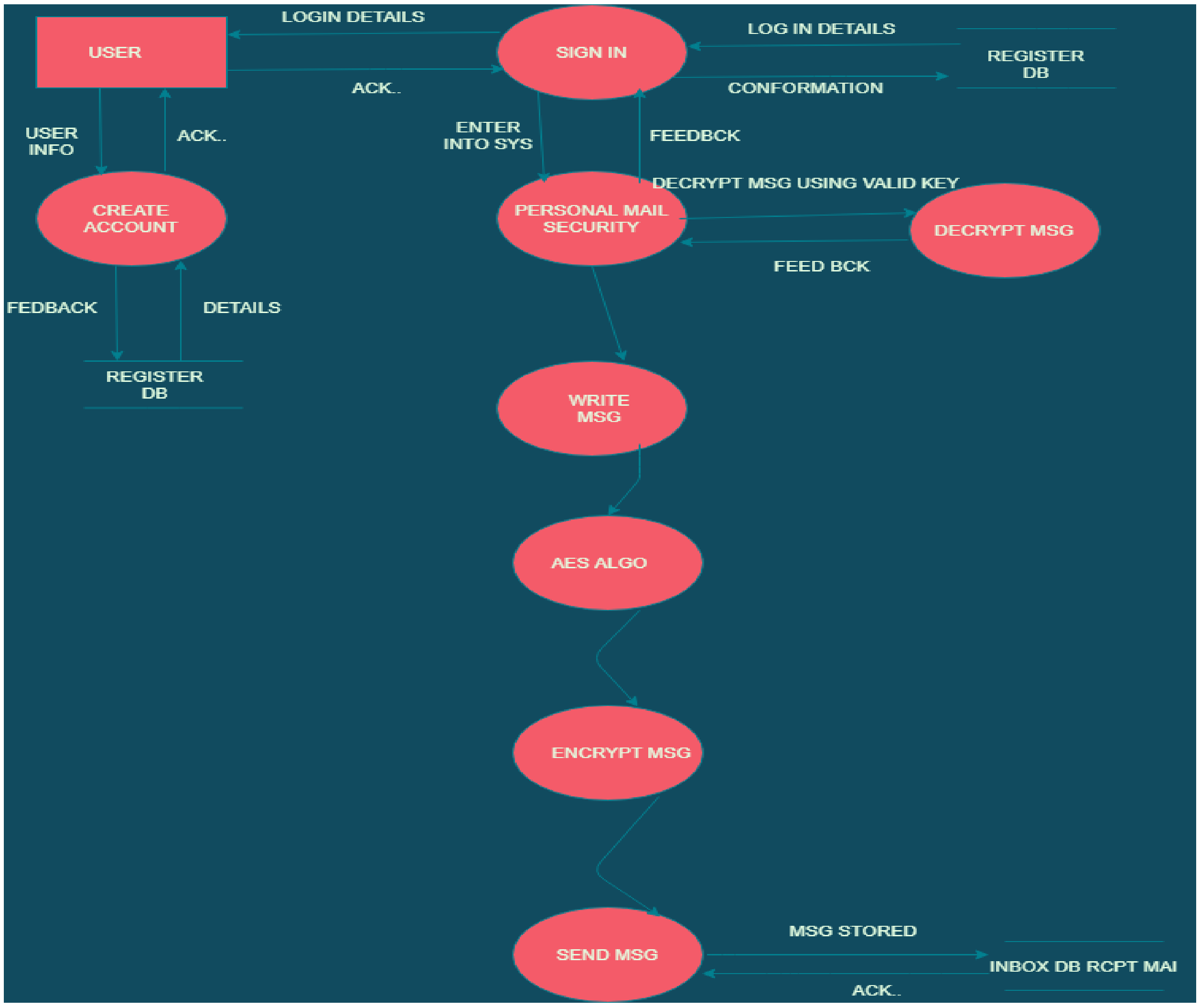
DFD 0 level:



Top Level DFD:

The Top Level DFD gives the overview of the whole system identifying the major system processes and data flow. This level focuses on the single process that is drawn in the context diagram by 'Zooming in' on its contents and illustrates what it does in more detail.

1 LEVEL DFD



4.1.2. Use Case Documentation:

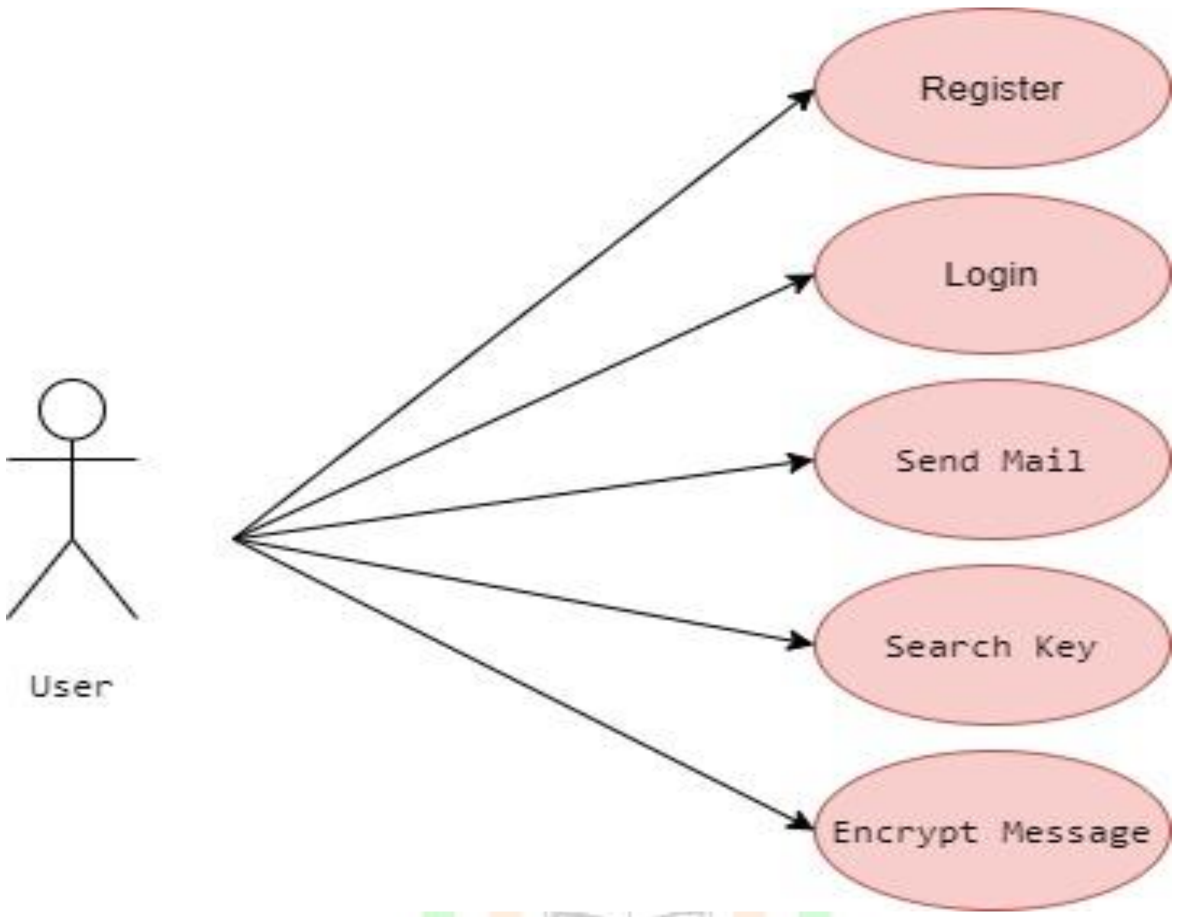
Use Case Diagram

- A use case diagram is a diagram that shows a set of use cases and actors and relationships.

Contents

- Use case commonly contain
    - Use cases
    - Actors
    - Dependency, generalization and association relationships

<u>Over all use case diagram,</u>



<u>Process Flow</u>

Activity Diagrams:

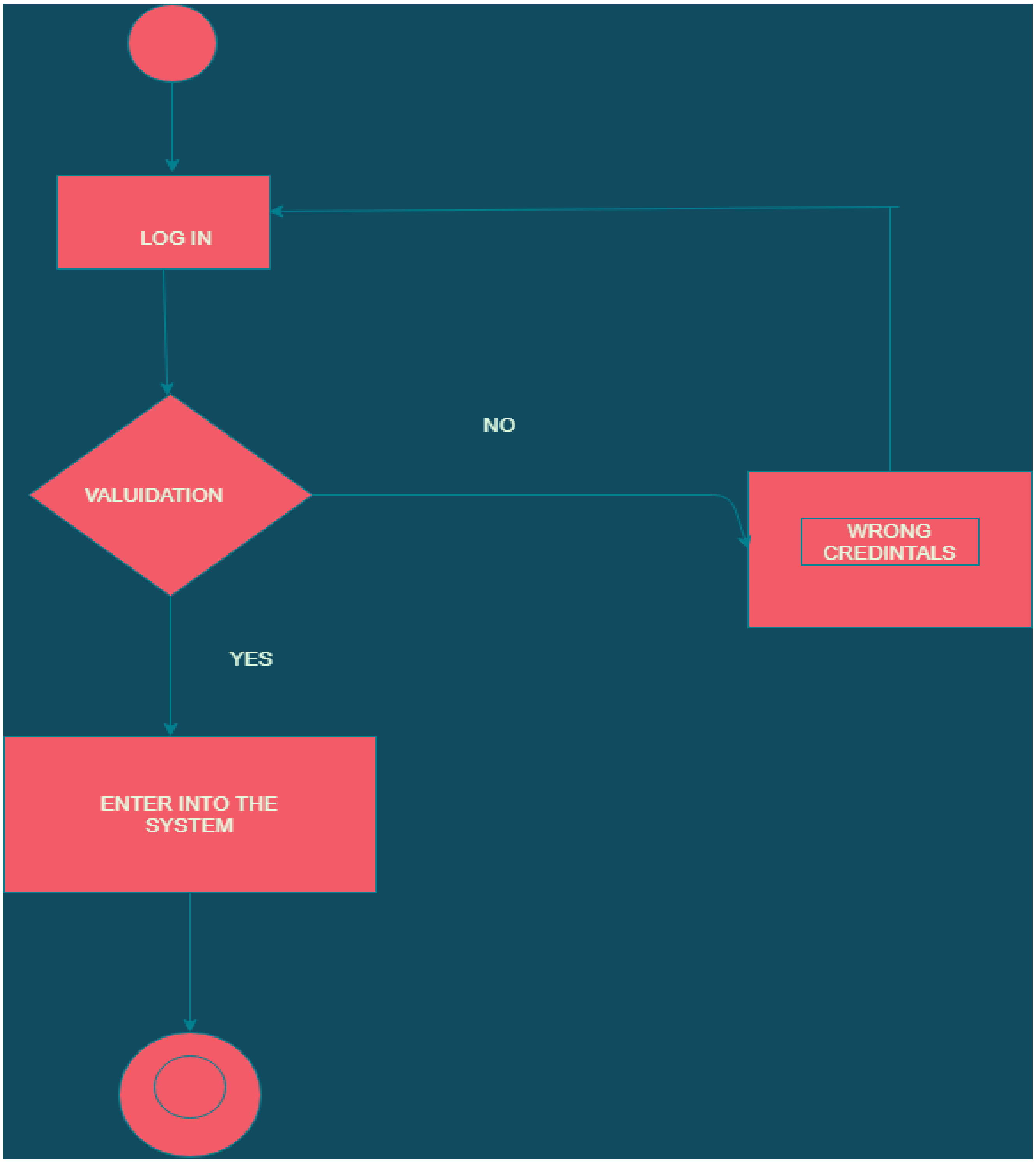
Activity Diagram:

- An activity diagram shows the flow from activity to activity. An activity is an ongoing non-atomic execution within a state machine.
- Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of the system or the return of a value.
  Activity diagrams commonly contain

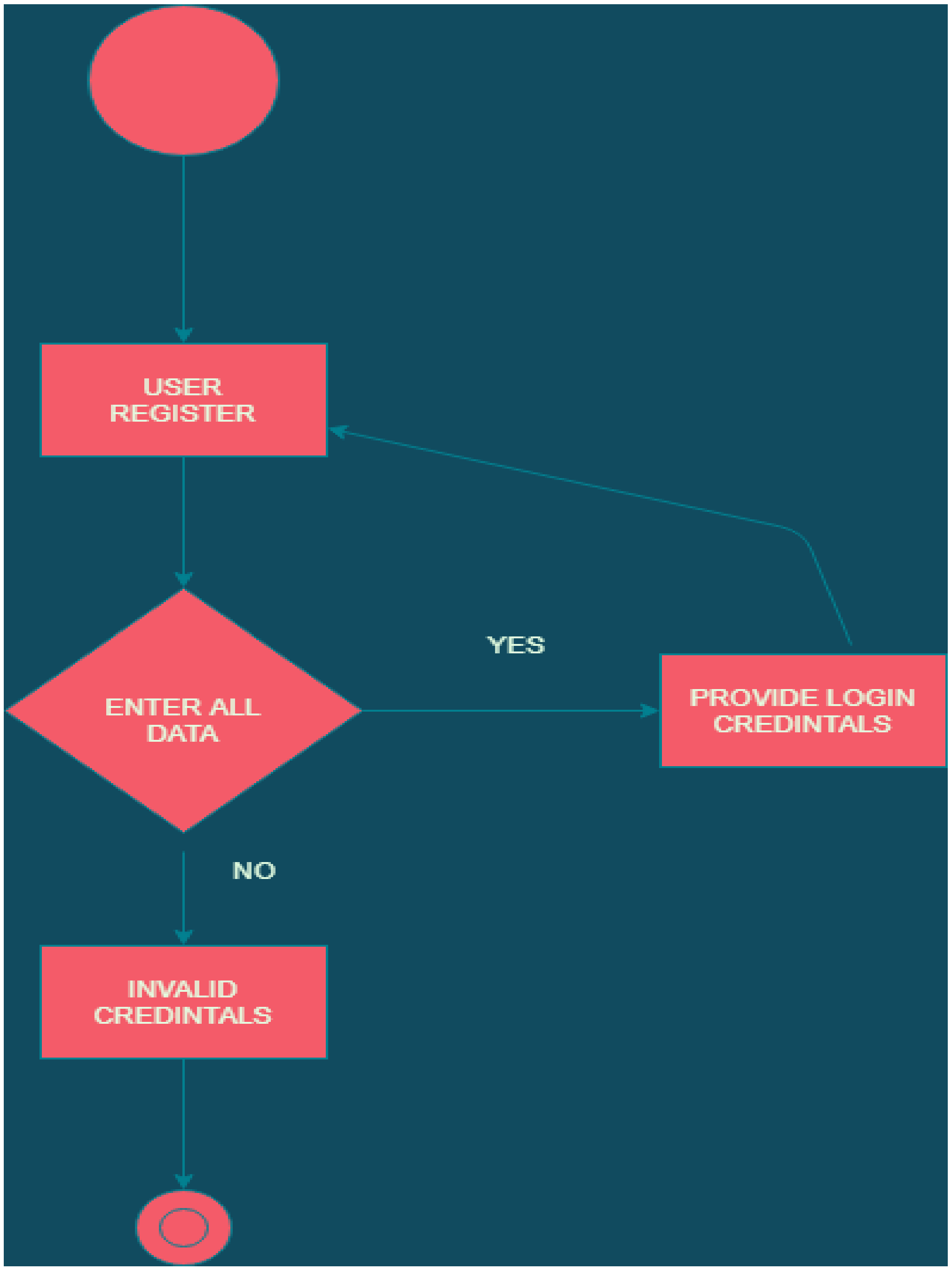
- ➢ Activity states and action states
- ➢ Transitions
- ➢ Objects

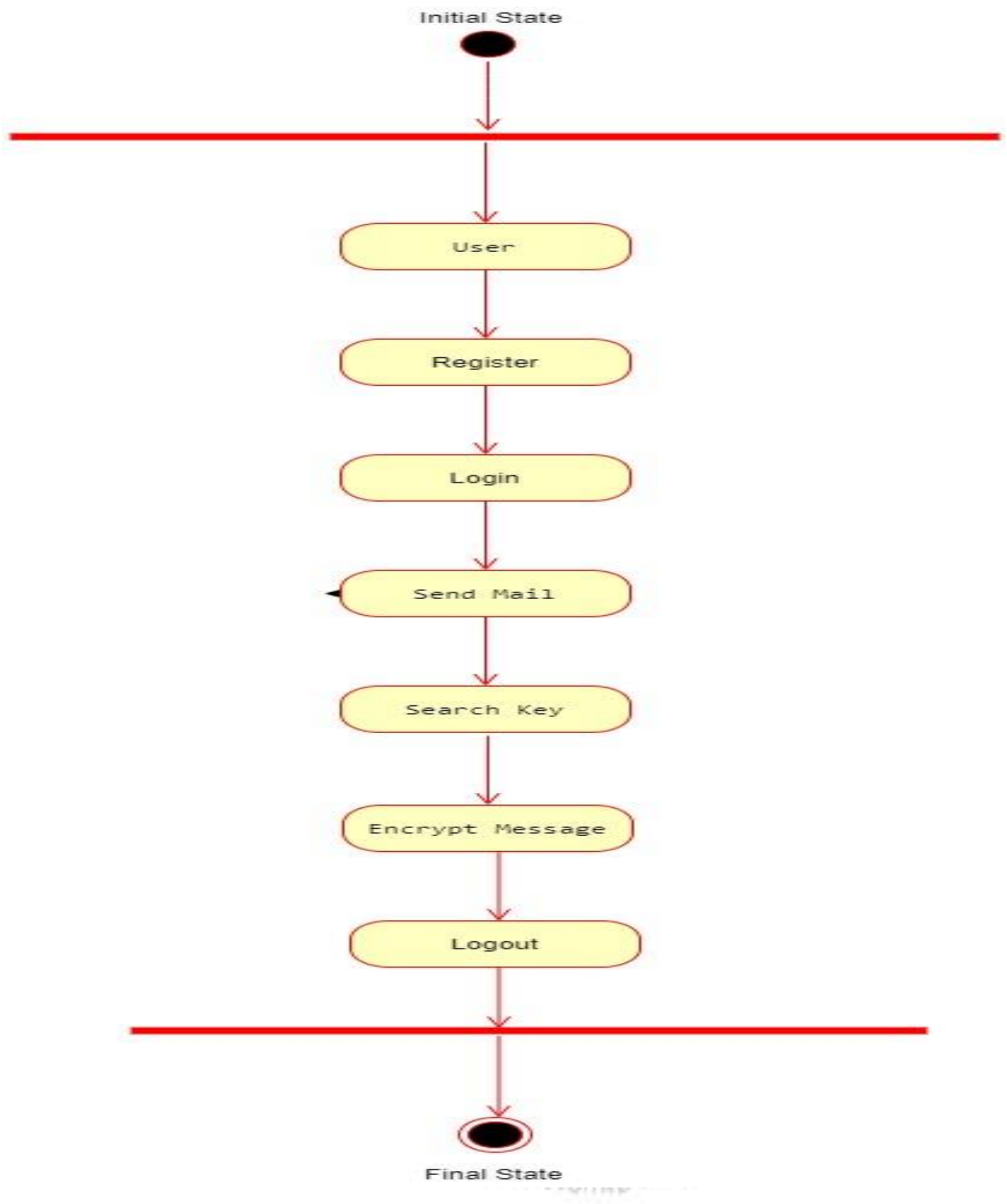
- Like all other diagrams, activity diagrams may contain notes and constrains.

Activity diagram of Login Process


LOG IN
NO
VALUIDATION
WRONG CREDINTALS
YES
ENTER INTO THE SYSTEM

Activity diagram of Registration Process :





User Activity

SYSTEM DESIGN

SYSTEM DESIGN

The main focus of the analysis phase of Software development is on “What needs to be done”. The objects discovered during the analysis can serve as the framework or Design. The class’s attributes, methods and association identified during analysis must be designed for implementation language. New classes must be introduced to store intermediate results during the program execution.

Emphasis shifts from the application domain o implementation and computer such as user interfaces or view layer and access layer. During analysis, we look at the physical entities or business objects in the system, that is, which players and how they cooperate to do the work of the application. These objects represent tangible elements of the business.

During the Design phase, we elevate the model into logical entities, some of which might relate more to the computer domain as people or employees. Here his goal is to design the classes that we need to implement the system the difference is that, at this level we focus on the view and access classes, such as how to maintain information or the best way o interact with a user or present information.

Design process:

During the design phase the classes identified in object-oriented analysis Must be revisited with a shift focus to their implementation. New classes or attribute and Methods must be an added for implementation purposes and user interfaces. The object-oriented design process consists of the following activities:

1. Apply design axioms to design classes, their attributes, methods, associations, structure

And protocols Refine and complete the static UML class diagram by adding details to the UML diagram. This step consists of following activities. *Refine attributes *Design methods and protocols by utilizing a UML activity diagram to represent the method’s algorithms.

*Refine associations between classes

*Refine class hierarchy and design with inheritance

*Iterate and refine again

2. Design the access layer

- Create mirror classes: For every business class identified and created. For example, if there are three business classes, create three access layer classes.
- Identify access layer class relationships.
- Simplify classes and their relationships: The main goal here is to eliminate redundant classes and structures.

  *Redundant classes: Do not keep two classes that perform similar translate results

  activities. Simply select one and eliminate the other.

  *Method classes: Revisit the classes that consist of only one or two methods to see if they can be eliminated or combined with existing classes.

- Iterate and refine again.

  Define the view layer classes

  - Design the macro level user interface, identifying view layer objects.
  - Design the micro level user interface, which includes these activities:

    * Design the view layer objects by applying the design axioms and corollaries.

    * Built a prototype of the view layer interface.

    - Test usability and user satisfaction
    - Iterate and refine.

3. Iterate refine the whole design process. From the class diagram, you can begin to extrapolate which classes you will have to built and which existing classes you can reuse. As you do this, also begin this, also begin thinking about the inheritance structure. If you have several classes that seem relates but have specific differences.

Design also must be traceable across requirements, analysis, design from the Requirements model.

DESIGN AXIOMS

Axioms are a fundamental truth that always is observed to be valid and for which there is no counter example or exception. Such explains that axioms may be hypothesized form a large number of observations by nothing the common phenomena shared by all cases; they cannot be proven or derived, but they can be invalidated by counter examples or exceptions. A theorem is a proposition that may not be self-evident but can be proven from accepted axioms. If therefore, is equivalent to a law or principle. A corollary is a proposition that follows from an axioms or another proposition that has been proven. Again, corollary is shown to be valid or not valid in the same manner as a theorem. In the two important axioms axiom 1 deals with relationships between system components and axiom 2 deals with the complexity of design.

The following the two important axioms:

Axiom 1: The independence axiom, which maintain the independence of the components.

Axiom 2: The information axioms that maintain the information content of the design.

Axioms1 states that, during the design process, as we go from requirement and use case to a system component, each component must satisfy that requirement without affecting other requirements.

An axiom 2 is concerned with simplicity. Scientific theoreticians often rely on a general rule known as Occam's razor, after William of Occam. He says, "The best theory explains the known facts with a minimum amount of complexity and maximum simplicity and straightforwardness."

The best designs usually involve the least complex code but not necessarily the fewest number of classes or methods. Minimizing complexity should be the goal, because that produces the most easily maintained and enhanced application. In an object-oriented system, the best way to minimize complexity is to use inheritance and the systems built in classes and to add as little as possible to what already is there.

From the two design axioms, many corollaries may be derived as a direct consequence of the axioms. These corollaries may be more useful in marking specific design decisions, since they can be applied to actual situations.

1. Uncoupled design with less information content: Highly cohesive objects can improve coupling because only a minimal amount of essential information need be passed between objects. The degree or strength of coupling between two components is measured by the amount and complexity of information transmitted between them.
2. Single purpose: Each class must have single, clearly defined purposes.
3. Large number of simple classes: Keeping the classes simple allows reusability. Large and complex classes are too specialized to be reused.
4. Strong mapping: There must be a strong association between the physical system and logical design. During the design phase, we need to design this class, design its methods, its association with other objects. So a strong mapping links classes should be identified.
5. Standardization: promote standardization by designing interchangeable and reusing existing classes or components.
6. Design with inheritance: Common behavior must be moved to super classes. The super class-sub class structure must make logical sense.

Refining attributes and methods:

Attributes identified in object oriented analyzed must be refined in the design phase. In the analysis phase, the name of the attributes was sufficient. But in the design phase, detailed information must be added to the model. The three basic types of attributes are:

1. Single valued attributes: This has only value or state.
2. Multiplicity or multivalue attributes: This has a collection of many values at any point in time.
3. Instance connection attributes: This is required to provide the mapping needed by an object to fulfill its responsibilities.

UML attribute presentation:

Visibility name: type-expression=initial-value

Visibility indicates either public visibility or protected visibility or private visibility. The public visibility indicates that the attribute can be accessible to all classes. The protected visibility indicates that the accessibility is given to the subclasses and operations of the class. The private visibility indicates that the accessibility can be given only to the operations of the class only.

Type expression is a language dependent specification of the implementation type of an attribute. Initial value is a language dependent expression for the initial value is optional.

5.1 Sequence and collaboration diagrams

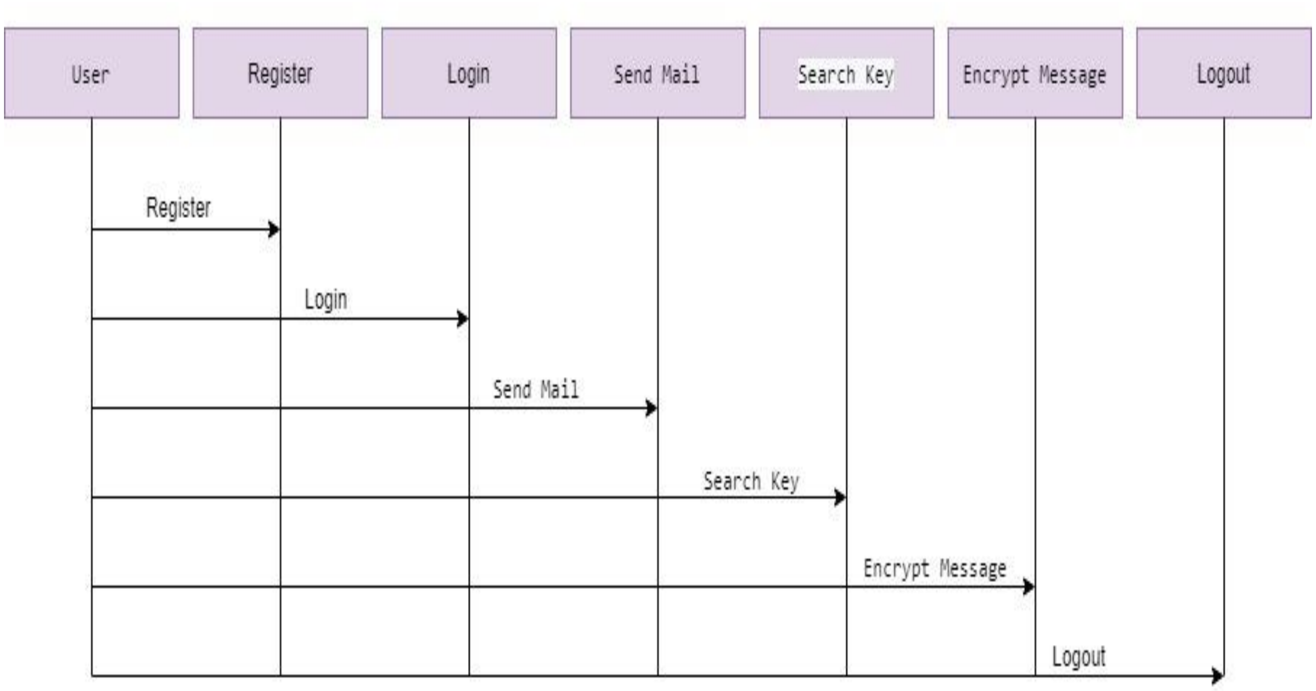
Sequence Diagram

- An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them.
- A sequence diagram is an interaction diagram that emphasizes the time ordering of messages.
- Graphically, a sequence diagram is a table that shows objects arranged along x-axis and messages, ordered in increasing time, along the y-axis.

Contents

- Sequence diagrams commonly contain the following:
    - Objects
    - Links
    - Messages

Sequence:



<u>Collaboration Diagram</u>

- Collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that's bigger than the sum of all its parts.

- Collaboration is also the specification of how an element, such as a classifier or an operation, is realized by a set of classifiers and associations playing specific roles used in a specific way

Contents

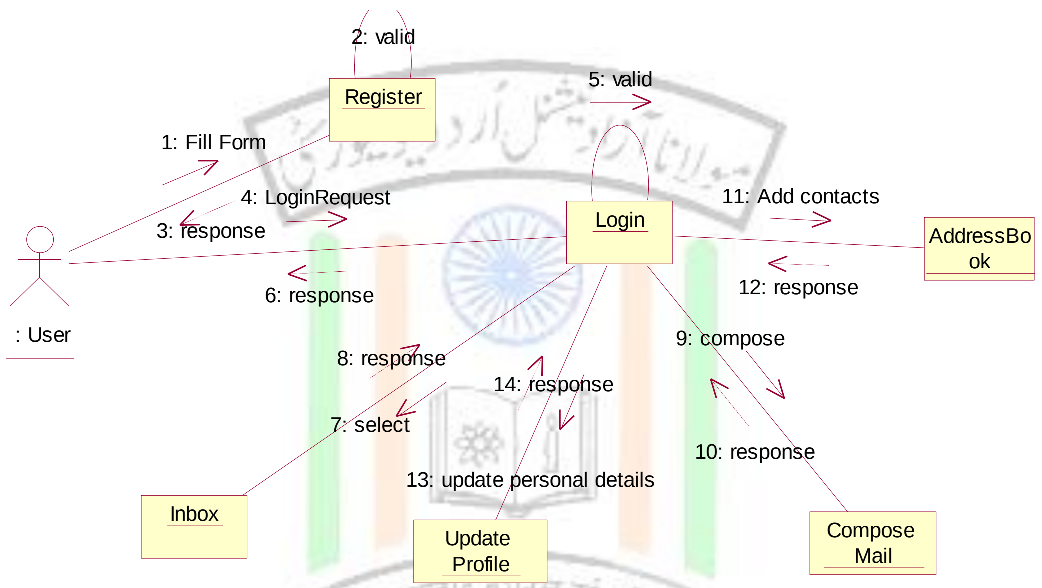
Collaboration diagrams commonly contain the following:

- Objects

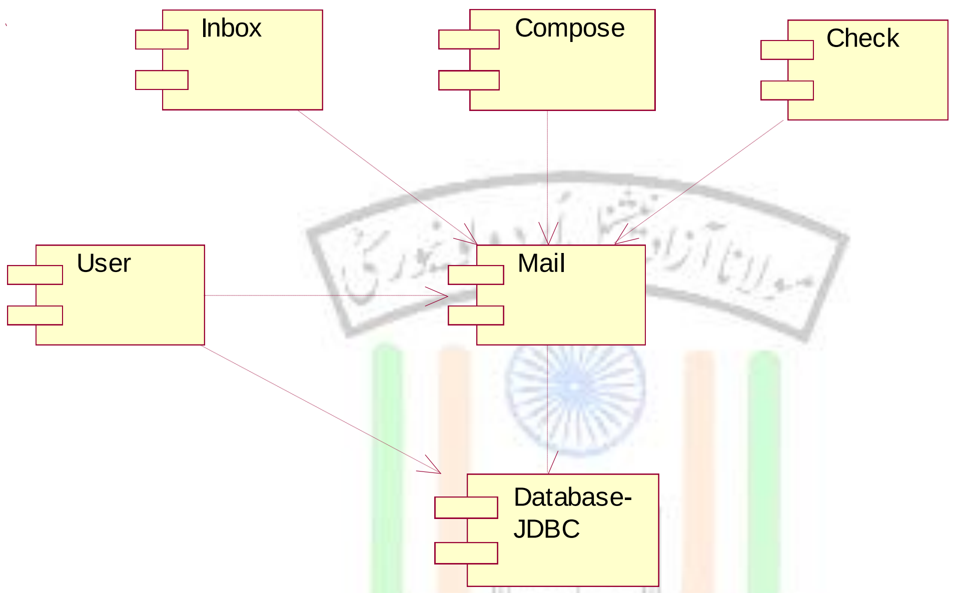
- Links
- Messages

Like all other diagrams, sequence diagrams may contain notes and constrains.

Collaboration



Component Diagram:



Deployment Diagram

- A deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them.

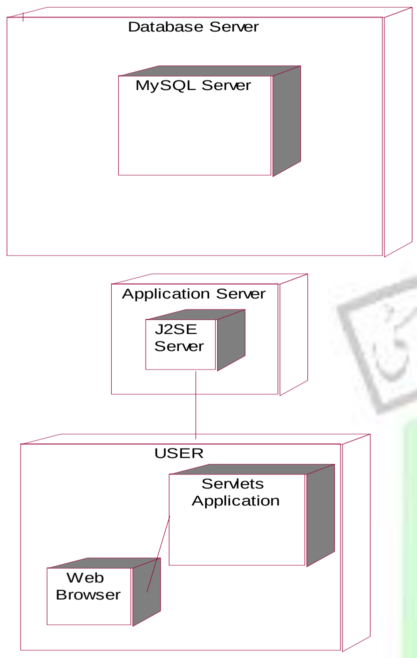
000- Graphically, a deployment diagram is collection of vertices and arcs.

Contents

- Deployment diagram commonly contain the following things:
    - Nodes
    - Dependency and association relationships
- Like all other diagrams, deployment diagrams may contain notes and constraints.

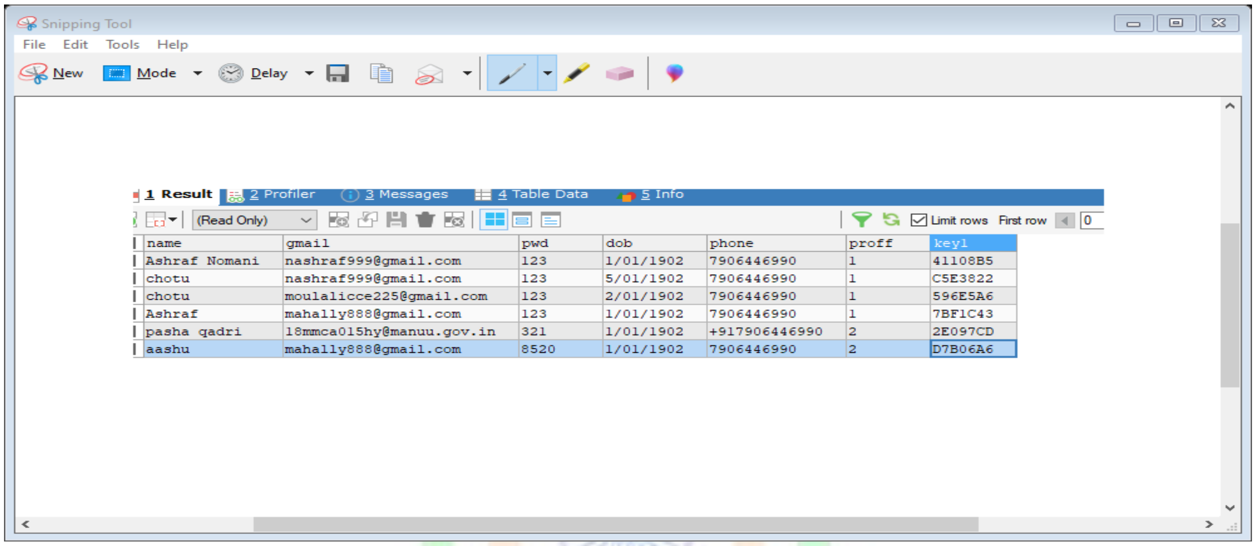
- Deployment diagrams may also contain components, each of which must live on some node.

- Deployment diagrams may also contain packages or subsystems, both of which are used to group elements of your model into larger chunks.



MYSQL DATA BASE TABLES

Table Name:- gmail



name	gmail	pwd	dob	phone	proff	key1
Ashraf Nomani	nashraf999@gmail.com	123	1/01/1902	7906446990	1	41108B5
chotu	nashraf999@gmail.com	123	5/01/1902	7906446990	1	C5E3822
chotu	moulalicce225@gmail.com	123	2/01/1902	7906446990	1	596E5A6
Ashraf	mahally888@gmail.com	123	1/01/1902	7906446990	1	7BF1C43
pasha qadri	18mmca015hy@manuu.gov.in	321	1/01/1902	+917906446990	2	2E097CD
aashu	mahally888@gmail.com	8520	1/01/1902	7906446990	2	D7B06A6

**** RECEIPENT MAIL SERVER IS USED FOR INBOX DB..**

TESTING AND IMPLEMENTATION

TESTING AND IMPLEMENTATION

Example for GUI Test cases:

T.C. No	Description	Expected value	Actual value	Result
1	Check for all the features in the screen	The screen must contain all the features		
2	Check for the alignment of the objects as per the validations	The alignment should be in proper way		

1. **Positive Test Cases:**

- The positive flow of the functionality must be considered
- Valid inputs must be used for testing
- Must have the positive perception to verify whether the requirements are justified.

Example for Positive Test cases:

T.C. No	Description	Expected value	Actual value	Result
1	Check for the date Time Auto Display	The date and time of the system must be displayed		
2	Enter the valid Roll no into the	It should accept		

	student roll no field			

2. Negative Test Cases:

- Must have negative perception.
- Invalid inputs must be used for test.

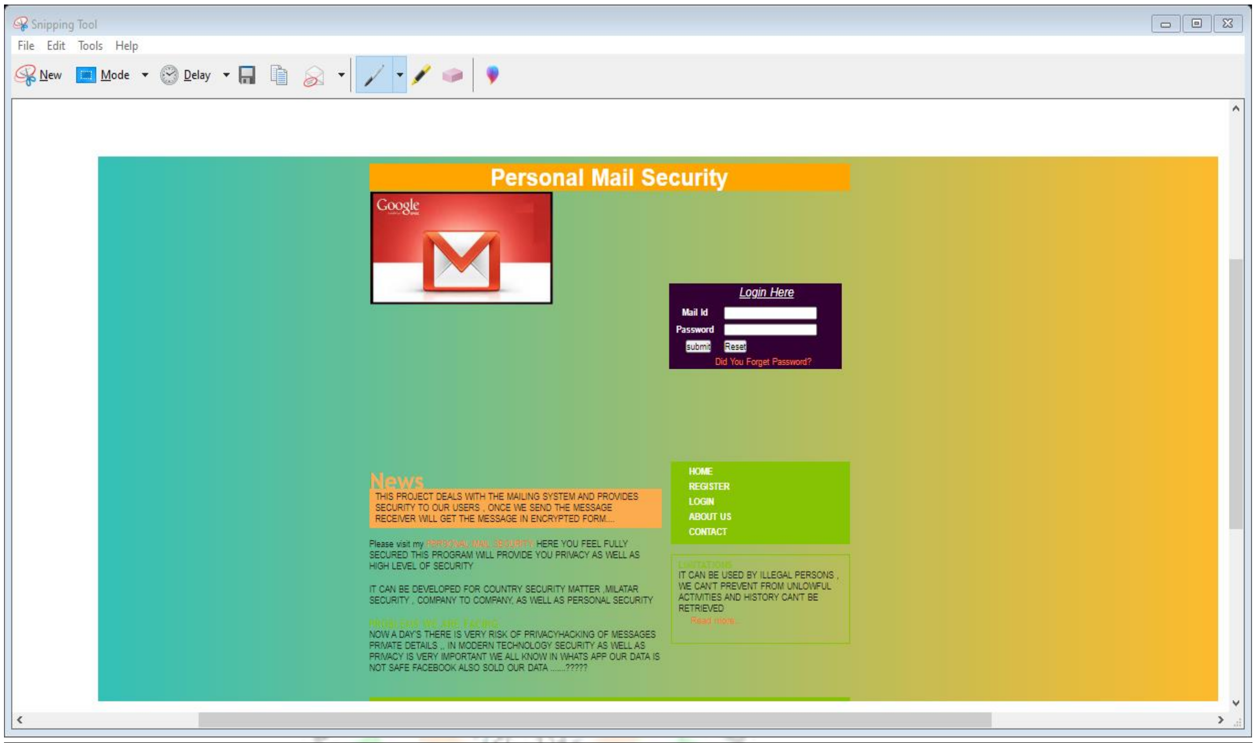
Example for Negative Test cases:

T.C. No	Description	Expected value	Actual value	Result
1	Try to modify the information in date and time	Modification should not be allow		
2	Enter invalid data in to the student details form, click on Save	It should not accept invalid data, save should not allow		

GUI

OUTPUT RESULTS

HOME.html



Personal Mail Security

Login Here

Mail Id

Password

submit Reset

Did You Forget Password?

News

THIS PROJECT DEALS WITH THE MAILING SYSTEM AND PROVIDES SECURITY TO OUR USERS , ONCE WE SEND THE MESSAGE RECEIVER WILL GET THE MESSAGE IN ENCRYPTED FORM....

Please visit my PERSONAL MAIL SECURITY HERE YOU FEEL FULLY SECURED THIS PROGRAM WILL PROVIDE YOU PRIVACY AS WELL AS HIGH LEVEL OF SECURITY

IT CAN BE DEVELOPED FOR COUNTRY SECURITY MATTER ,MILATAR SECURITY , COMPANY TO COMPANY, AS WELL AS PERSONAL SECURITY

PROBLEMS WE ARE FACING

NOW A DAY'S THERE IS VERY RISK OF PRIVACYHACKING OF MESSAGES PRIVATE DETAILS ,, IN MODERN TECHNOLOGY SECURITY AS WELL AS PRIVACY IS VERY IMPORTANT WE ALL KNOW IN WHATS APP OUR DATA IS NOT SAFE FACEBOOK ALSO SOLD OUR DATA ....?????

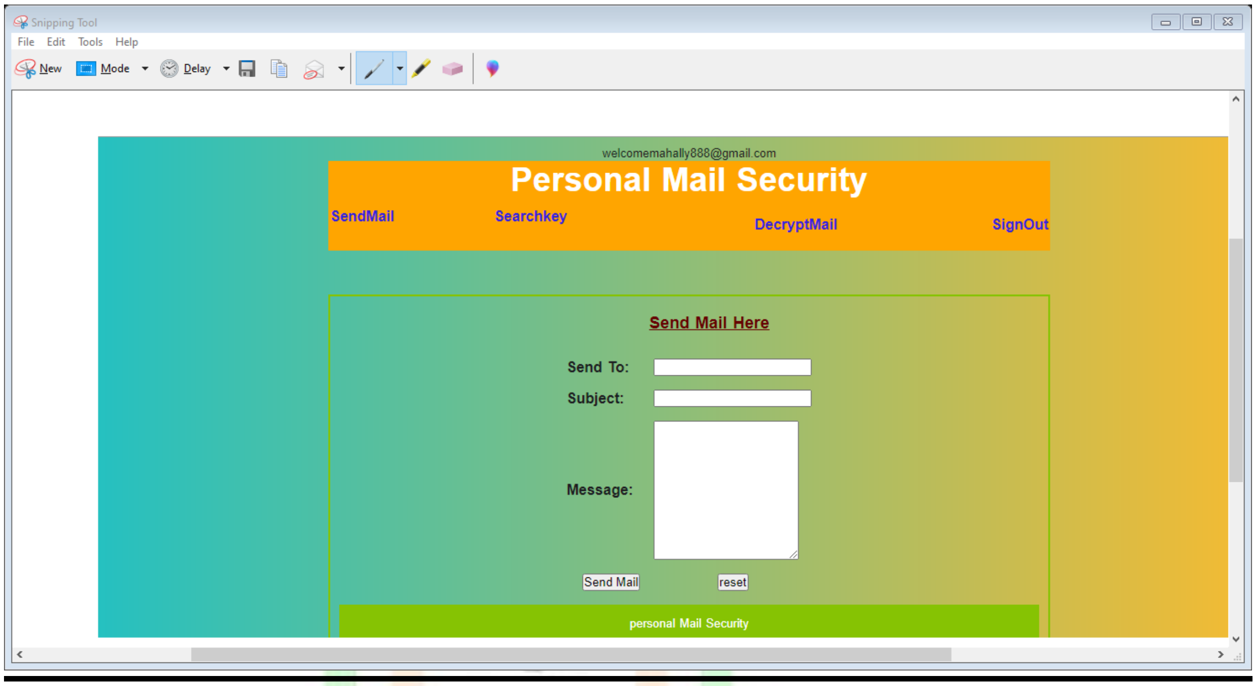
HOME
REGISTER
LOGIN
ABOUT US
CONTACT

LIMITATIONS

IT CAN BE USED BY ILLEGAL PERSONS , WE CAN'T PREVENT FROM UNLOWFUL ACTIVITIES AND HISTORY CAN'T BE RETRIEVED

Read more...

AFTER SUCCESSFUL LOGIN



welcomemahally888@gmail.com

Personal Mail Security

SendMail Searchkey DecryptMail SignOut

Send Mail Here

Send To:

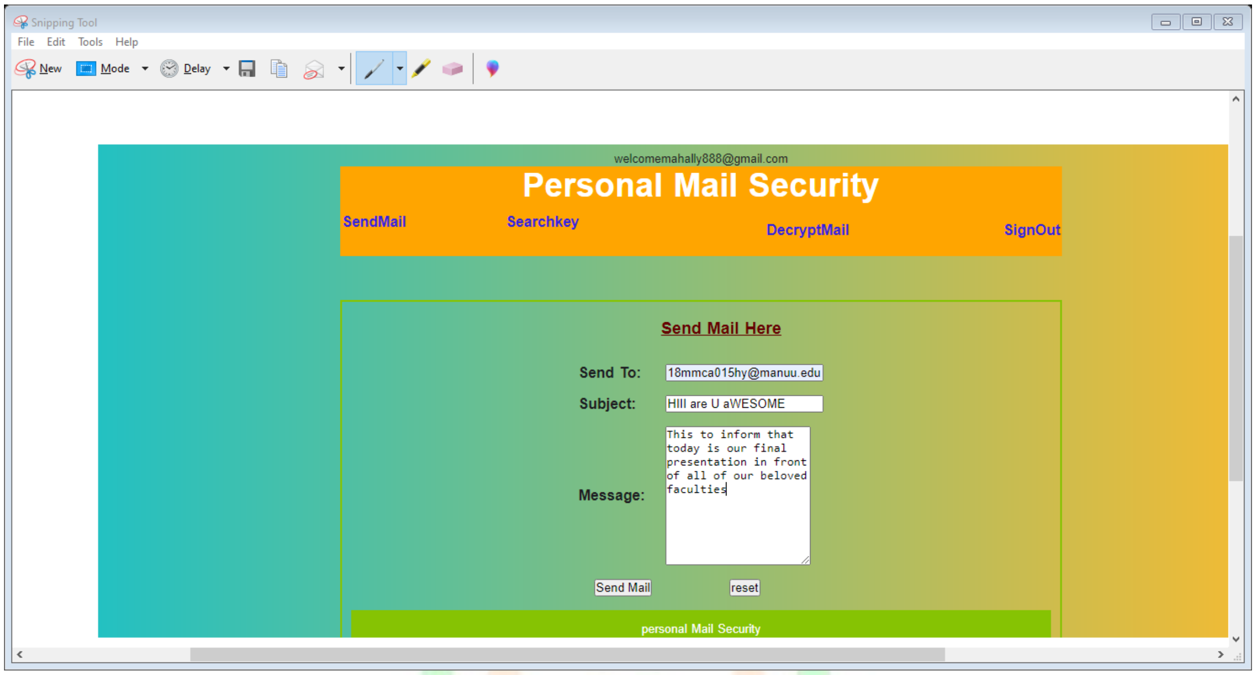
Subject:

Message:

Send Mail reset

personal Mail Security

SEND MAIL


welcomemahally888@gmail.com

Personal Mail Security

SendMail Searchkey DecryptMail SignOut

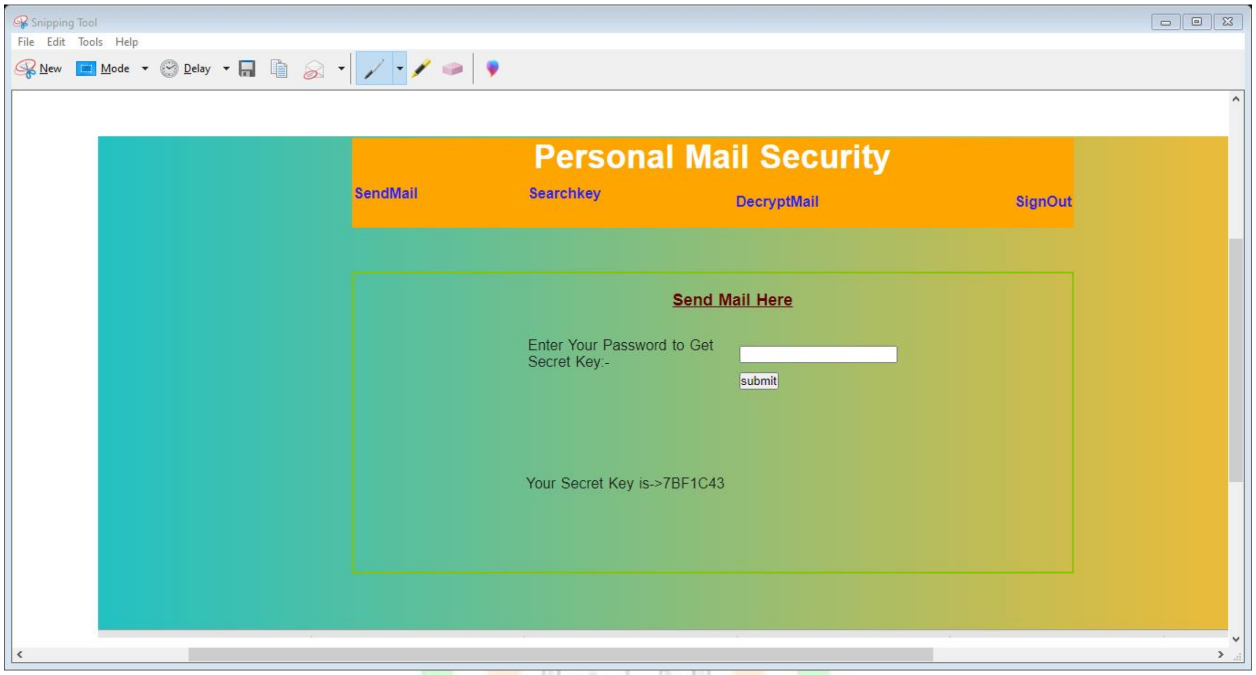
Send Mail Here

Send To: 18mmca015hy@manuu.edu

Subject: HIII are U aWESOME

Message: This to inform that today is our final presentation in front of all of our beloved faculties

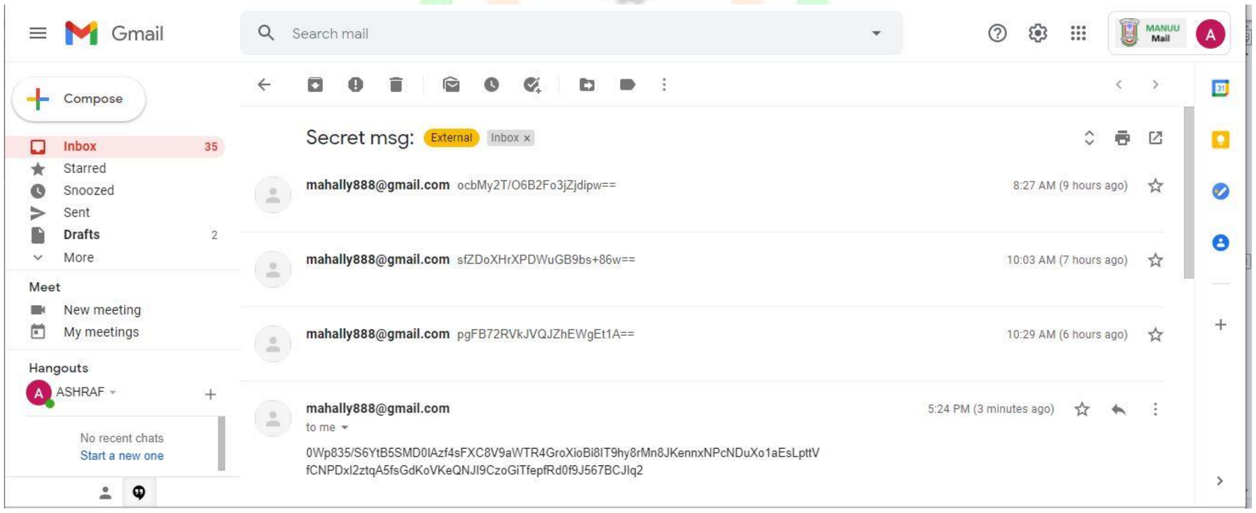
AFTER SUCCESSFUL MAIL SENT



Enter Your Password to Get Secret Key:-

Your Secret Key is->7BF1C43

AFTER SEARCHING KEY WE HAVE TO DECRYPT MESSAGE CHECK MAIL WHICH I SEND



Secret msg:

mahally888@gmail.com ocbMy2T/O6B2Fo3jZjdipw== 8:27 AM (9 hours ago)

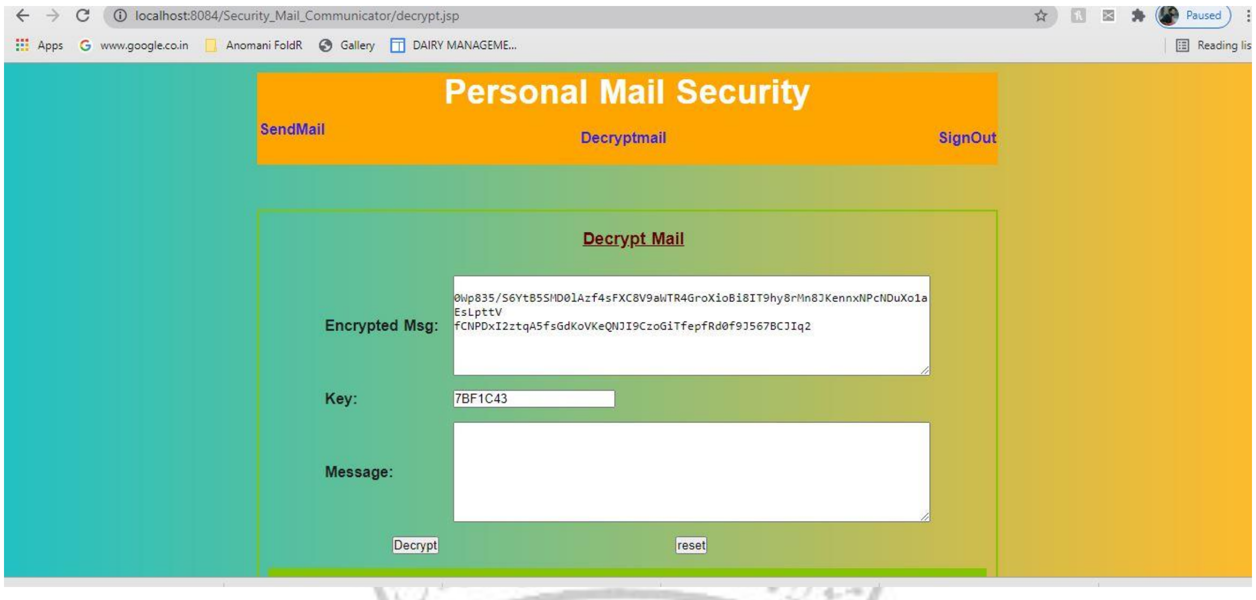
mahally888@gmail.com sfZDoXHrXPDWuGB9bs+86w== 10:03 AM (7 hours ago)

mahally888@gmail.com pgFB72RVkJVQJZhEWgEt1A== 10:29 AM (6 hours ago)

mahally888@gmail.com 5:24 PM (3 minutes ago)
to me

0Wp835/S6YtB5SMD0lAzf4sFXC8V9aWTR4GroXioBi8IT9hy8rMn8JKennxNPcNDuXo1aEsLpttV
fCNPDxl2ztqA5fsGdKoVKeQNJI9CzoGiTfepfRd0f9J567BCJlq2

WE HAVE TO COPY MESSAGE AND PASTE IT TO THE BOX


localhost:8084/Security_Mail_Communicator/decrypt.jsp

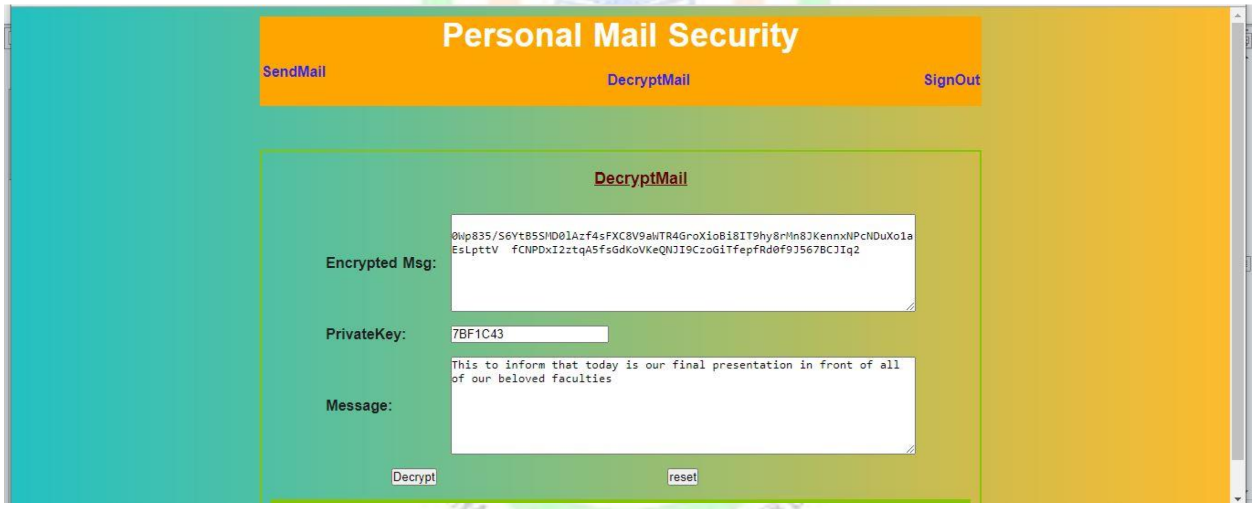
Encrypted Msg:

```
0Wp835/S6YtB5SMD0lAzf4sFXC8V9aWTR4GroXioBi8IT9hy8rMn8JKennxNPcNDuXo1a
EsLpttV
fCNPDxI2ztqA5fsGdKoVKeQNJI9CzoGiTfepfRd0f9J567BCJIq2
```

Key: 7BF1C43

Message:

ONCE I CLICK ON THE DECRYPT BUTTON THEN MESSAGE WILL BE DECRYPTED



Encrypted Msg:

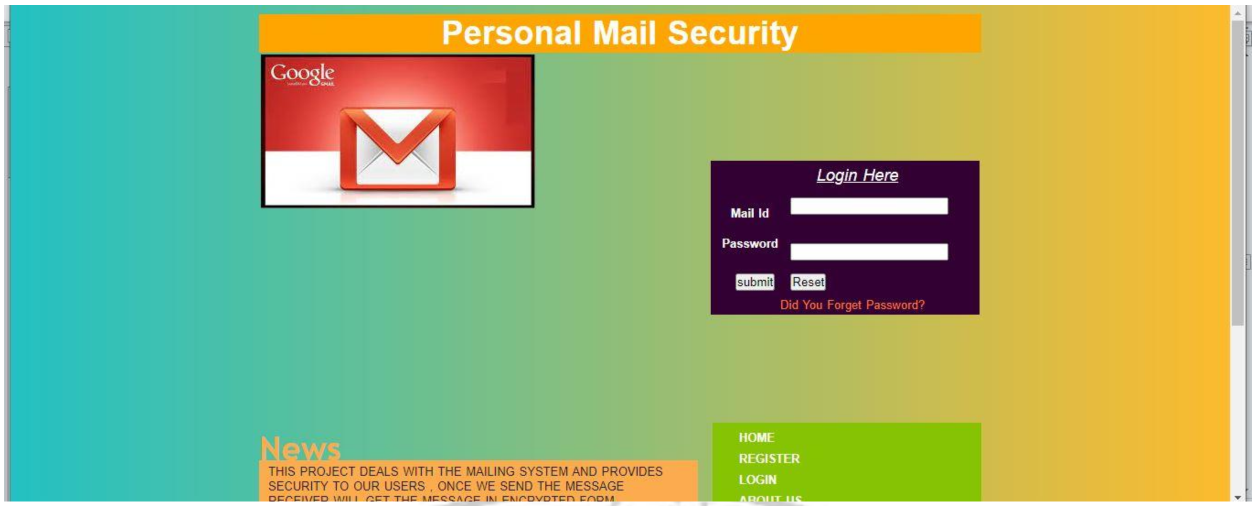
```
0Wp835/S6YtB5SMD0lAzf4sFXC8V9aWTR4GroXioBi8IT9hy8rMn8JKennxNPcNDuXo1a
EsLpttV  fCNPDxI2ztqA5fsGdKoVKeQNJI9CzoGiTfepfRd0f9J567BCJIq2
```

PrivateKey: 7BF1C43

Message:

```
This to inform that today is our final presentation in front of all
of our beloved faculties
```

AFTER THAT WE HAVE TWO OPTIONS SEND MESSAGE AGAIN OR LOG OUT



THIS PROJECT DEALS WITH THE MAILING SYSTEM AND PROVIDES SECURITY TO OUR USERS , ONCE WE SEND THE MESSAGE

I WANT TO LOG OUT WE CAN SEND MESSAGE AGAIN….

Coding

File Name:Mail.java

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

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 java2
 */
public class mail {

    public static boolean secretMail(String msg, String userid, String to) {
        Properties props = new Properties();
        props.put("mail.smtp.host", "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");
    // Assuming you are sending email from localhost
    Session session = Session.getDefaultInstance(props,
            new javax.mail.Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication("mahally888@gmail.com", "manuu000");
                }
            });

    System.out.println("Message   " + msg);
    try {
        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress(userid));
        message.setRecipients(Message.RecipientType.TO,
                InternetAddress.parse(to));
        message.setSubject("Secret msg: ");
        message.setText(msg);

        Transport.send(message);

        System.out.println("Done");
        return true;

    } catch (MessagingException e) {
        System.out.println(e);
        e.printStackTrace();
        return false;
        // throw new RuntimeException(e);
    }
}
```

```
}
```

File Name: CaptchaServlet.java

```
import java.awt.Color;

import java.awt.Font;

import java.awt.GradientPaint;

import java.awt.Graphics2D;

import java.awt.RenderingHints;

import java.awt.image.BufferedImage;

import java.io.*;

import java.util.Random;

import javax.imageio.ImageIO;

import javax.servlet.*;

import javax.servlet.http.*;



public class CaptchaServlet extends HttpServlet {



  /**

       *

       */

      private static final long serialVersionUID = 1L;



protected void processRequest(HttpServletRequest request,

                                HttpServletResponse response)

                 throws ServletException, IOException {


    int width = 150;
    int height = 50;

            char data[][] = {
                        {'g','y','7','0','N','n','1'},
                        {'G','v','V','t','Y','O','J'},
                        {'p','z','t','c','f','1','H'},
                        {'r','u','6','K','f','d','C'},
                        {'L','1','E','i','x','0','7'},
                        {'j','c','r','n','u','f','N'},
                        {'w','v','P','O','K','Y','I'},
                        {'f','D','J','p','i','x','h'},
                        {'p','1','3','T','Q','g','k'},
                        {'t','c','B','C','n','e','x'},
                        {'1','p','x','T','W','k','v'},
                        {'E','i','N','I','S','e','r'},
                        {'0','I','Q','W','c','n','h'},
                        {'c','C','d','W','K','F','V'},
                        {'e','8','M','A','J','7','1'},
                        {'9','A','t','h','o','f','7'},
                        {'r','j','v','P','S','d','R'},
                        {'p','J','X','F','v','M','w'},
                        {'p','W','m','4','1','7','y'},
                        {'N','a','s','H','U','w','7'},
                        {'X','f','V','C','J','P','e'},
                        {'d','K','4','V','l','F','9'},
                        {'K','Z','G','i','q','O','N'},
                        {'l','G','V','F','R','o','1'},
                        {'R','9','T','O','D','i','M'},
                        {'4','5','2','7','q','O','0'},
                        {'Y','9','W','J','F','H','T'},
                        {'a','t','f','M','L','Z','p'},
                        {'5','j','1','J','9','k','r'},
                        {'5','K','z','N','6','F','B'},
                        {'R','d','G','1','q','9','0'},
                        {'d','1','4','q','Z','f','m'},
                        {'v','g','S','u','q','C','s'},
                        {'o','Q','A','l','r','P','B'},
                        {'h','u','0','M','4','w','o'},
                        {'3','5','d','R','P','W','N'},
                        {'w','C','p','v','Q','7','q'},
                        {'h','3','z','Z','p','5','y'},
                        {'K','F','h','O','6','5','T'},
                        {'D','6','V','b','H','w','j'},
                        {'r','I','9','A','j','s','Q'},
                        {'J','0','b','p','U','l','M'},
                        {'q','C','z','B','6','9','W'},
                        {'4','j','y','c','M','f','u'},
                        {'5','z','v','F','D','h','S'},
                        {'G','J','P','q','m','a','i'},
                        {'H','J','d','Z','q','m','e'},
                        {'H','o','6','w','7','F','d'},
                        {'M','6','e','Z','m','C','V'},
                        {'v','n','O','r','X','q','t'},
                        {'d','g','R','B','j','K','E'},
                        {'9','W','m','O','f','j','U'},
                        {'3','4','u','N','h','X','S'},
                        {'a','w','l','u','Q','y','V'},
                        {'5','M','j','u','P','g','H'},
                        {'8','r','g','A','Q','W','U'},
                        {'P','n','b','8','d','x','M'},
                        {'5','P','O','j','v','Q','K'},
                        {'m','C','s','B','O','r','1'},
                        {'P','a','V','A','N','3','1'}
    };



    BufferedImage bufferedImage = new BufferedImage(width, height,
                  BufferedImage.TYPE_INT_RGB);


    Graphics2D g2d = bufferedImage.createGraphics();


    Font font = new Font("Georgia", Font.BOLD, 18);
    g2d.setFont(font);


    RenderingHints rh = new RenderingHints(
           RenderingHints.KEY_ANTIALIASING,
           RenderingHints.VALUE_ANTIALIAS_ON);


    rh.put(RenderingHints.KEY_RENDERING,
           RenderingHints.VALUE_RENDER_QUALITY);


    g2d.setRenderingHints(rh);


    GradientPaint gp = new GradientPaint(0, 0,
    Color.red, 0, height/2, Color.black, true);


    g2d.setPaint(gp);
    g2d.fillRect(0, 0, width, height);
```

```
        g2d.setColor(new Color(255, 153, 0));

        Random r = new Random();
        int index = r.nextInt(35);


        String captcha = String.copyValueOf(data[index]);
        request.getSession().setAttribute("captcha", captcha );

        int x = 0;
        int y = 0;

        for (int i=0; i<data[index].length; i++) {
            x += 10 + (Math.abs(r.nextInt()) % 15);
            y = 20 + Math.abs(r.nextInt()) % 20;
            g2d.drawChars(data[index], i, 1, x, y);
        }

        g2d.dispose();

        response.setContentType("image/png");
        OutputStream os = response.getOutputStream();
        ImageIO.write(bufferedImage, "png", os);
        os.close();
      }


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

```
            processRequest(request, response);
        }



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

File Name: Protector.java

```
import java.security.Key;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class Protector {

    private static final String ALGORITHM = "AES";
    private static final int ITERATIONS = 1;
    private static final byte[] keyValue =
        new byte[] { 'T', 'h', 'i', 's', 'I', 's', 'A', 'S', 'e', 'c',
'r', 'e', 't', 'K', 'e', 'y'};

    public String encrypt(String value, String salt) throws Exception {
        Key key = generateKey();
        Cipher c = Cipher.getInstance(ALGORITHM);
        c.init(Cipher.ENCRYPT_MODE, key);
```

```
            String valueToEnc = null;
            String eValue = value;
            for (int i = 0; i < ITERATIONS; i++) {
                valueToEnc = salt + eValue;
                byte[] encValue = c.doFinal(valueToEnc.getBytes());
                eValue = new BASE64Encoder().encode(encValue);
            }
            return eValue;
        }

        public String decrypt(String value, String salt) throws Exception {
            Key key = generateKey();
            Cipher c = Cipher.getInstance(ALGORITHM);
            c.init(Cipher.DECRYPT_MODE, key);

            String dValue = null;
            String valueToDecrypt = value;
            for (int i = 0; i < ITERATIONS; i++) {
                byte[] decordedValue = new
BASE64Decoder().decodeBuffer(valueToDecrypt);
                byte[] decValue = c.doFinal(decordedValue);
                dValue = new String(decValue).substring(salt.length());
                valueToDecrypt = dValue;
            }
            return dValue;
        }

        private static Key generateKey() throws Exception {
            Key key = new SecretKeySpec(keyValue, ALGORITHM);
```

```
                // SecretKeyFactory keyFactory =
SecretKeyFactory.getInstance(ALGORITHM);

                // key = keyFactory.generateSecret(new DESKeySpec(keyValue));

                return key;

            }

        }
```

File Name: CodeSending.java

```
        import java.io.IOException;

        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;

        import javax.servlet.ServletException;

        import javax.servlet.http.HttpServlet;

        import javax.servlet.http.HttpServletRequest;

        import javax.servlet.http.HttpServletResponse;

        public class codesending extends HttpServlet {
              private static final long serialVersionUID = 1L;


              String SMTP_HOST_NAME = "smtp.gmail.com";

               String SMTP_PORT = "465";


               String emailSubjectTxt = "Personal Mail Security Public Key";

               String emailFrom = "shiva.1000projects@gmail.com";

               String SSL_FACTORY = "javax.net.ssl.SSLSocketFactory";

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

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

                           String gmail=request.getParameter("mail");

                          String[] sendTo = new String[1];

                          sendTo[0]=gmail;

                          String emailMsgTxt = "Welcome To Security Mail
  Communicatior \n \n \t \t your public key is \t "+key;

                          try {

                                sendSSLMessage(sendTo, emailSubjectTxt,
  emailMsgTxt,

                                            emailFrom);

                                response.sendRedirect("home.html");

                          } catch (MessagingException e) {

                                // TODO Auto-generated catch block

                                e.printStackTrace();

                          }


                          }

                          public void sendSSLMessage(String recipients[],
  String subject,

                                      String message, String from) throws
  MessagingException {

                                boolean debug = true;

                                Properties props = new Properties();
                                props.put("mail.smtp.host", SMTP_HOST_NAME);
                                props.put("mail.smtp.auth", "true");
                                props.put("mail.debug", "true");
                                props.put("mail.smtp.port", SMTP_PORT);
                                props.put("mail.smtp.socketFactory.port",
  SMTP_PORT);
                                props.put("mail.smtp.socketFactory.class",
  SSL_FACTORY);
                                props.put("mail.smtp.socketFactory.fallback",
  "false");


                                Session session =
  Session.getDefaultInstance(props,
                                            new javax.mail.Authenticator() {


                                                  protected
  PasswordAuthentication getPasswordAuthentication() {
                                                        return new
  PasswordAuthentication(

        "shiva.1000projects@gmail.com", "9989765191");
                                                  }
                                            });


                                session.setDebug(debug);


                                Message msg = new MimeMessage(session);
                                InternetAddress addressFrom = new
  InternetAddress(from);
                                msg.setFrom(addressFrom);


                                InternetAddress[] addressTo = new
  InternetAddress[recipients.length];
                                for (int i = 0; i < recipients.length; i++) {
                                      addressTo[i] = new
  InternetAddress(recipients[i]);

                                }

                                msg.setRecipients(Message.RecipientType.TO,
  addressTo);


                                // Setting the Subject and Content Type

                                msg.setSubject(subject);

                                msg.setContent(message, "text/plain");

                                Transport.send(msg);


               }


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



              }


        }
```

File Name: RegisterDao.java

```
             package com.javatpoint;

             import java.sql.*;

             public class RegisterDao {



            public static int save(String firstname,String lastname,String
email,String password,String gender,Date sqldob,String city,String
state,String country,Date sqlcurrentdate){

                  int status=0;



                  try{
                        Connection con=ConnectionProvider.getConnection();

                        PreparedStatement ps=con.prepareStatement("insert 
into 
sssitmail_users(firstname,lastname,email,gender,dob,city,state,country,regist
ereddate,password) values(?,?,?,?,?,?,?,?,?,?)");

                        ps.setString(1,firstname);

                        ps.setString(2,lastname);

                        ps.setString(3,email);

                        ps.setString(4,gender);

                        ps.setDate(5,sqldob);

                        ps.setString(6,city);

                        ps.setString(7,state);

                        ps.setString(8,country);

                        ps.setDate(9,sqlcurrentdate);

                        ps.setString(10,password);


                        status=ps.executeUpdate();

                  }catch(Exception e){System.out.println(e);}



                  return status;

            }

      }
```

File Name: DecryptMessage.java

```
import java.io.IOException;

import java.security.Key;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;


import sun.misc.BASE64Decoder;


/**

 * Servlet implementation class decrypt

 */

public class decryptedmsg extends HttpServlet {






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

            // TODO Auto-generated method stub


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

            int i=0;

            String msg=null,mail=null;

            String dan="hai",enc=null;

            String check=null;

            ResultSet rs=null;

            Connection con=null;

            PreparedStatement pst=null;


            enc=request.getParameter("enc");

            System.out.println("Encrypted Message::::"+enc);

            check=request.getParameter("encdeckey");

            System.out.println("key is "+check);

             rs=null;

             System.out.println("1");

             HttpSession session=request.getSession();

            mail=session.getAttribute("gmail").toString();

            System.out.println(mail);


            try

            {

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

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



                  pst=con.prepareStatement("select key1 from register where
key1=? AND gmail=?");

                  pst.setString(1, check);

                  pst.setString(2, mail);

                  System.out.println("Query iss:::"+pst);
```

```
                    System.out.println("2");
                    rs=pst.executeQuery();

                    if(rs.next())
                    {
                          dan=rs.getString(1);
                          System.out.println("key is correct:::"+dan);
                          i=i+1;
                    }
                    System.out.println("I Value is::"+i);
                    con.close();
                    pst.close();
                    rs.close();
              } catch (Exception e1) {

                    e1.printStackTrace();
              }

              if(i>0)
              {
                    try
                    {
                    msg=decrypt(enc, check);
                    System.out.println("key is correct");

        response.sendRedirect("./decrypt2.jsp?enc="+enc+"&key="+check+"&msg="+m
    sg);


                    }
                    catch (Exception e)
                    {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }
              }
              else
              {
                    System.out.println("key not correct");

        response.sendRedirect("./decrypt2.jsp?enc="+enc+"&key="+check+"&msg="+e
  nc);
              }




        }
        public String decrypt(String value, String salt) throws Exception {
              String ALGORITHM = "AES";

            int ITERATIONS = 1;
             byte[] keyValue =
            new byte[] { 'T', 'h', 'i', 's', 'I', 's', 'A', 'S', 'e', 'c', 'r',
  'e', 't', 'K', 'e', 'y'};

              long serialVersionUID = 1L;
          Key key = new SecretKeySpec(keyValue, ALGORITHM);
          Cipher c = Cipher.getInstance(ALGORITHM);
          c.init(Cipher.DECRYPT_MODE, key);

          String dValue = null;
          String valueToDecrypt = value;
          for (int i = 0; i < ITERATIONS; i++) {
              byte[] decordedValue = new
  BASE64Decoder().decodeBuffer(valueToDecrypt);

              byte[] decValue = c.doFinal(decordedValue);

              dValue = new String(decValue);

              valueToDecrypt = dValue;

          }

          return dValue;

      }



  }
```

File Name:- SendMail.java

```
import java.io.IOException;

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;
import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class sendmail extends HttpServlet {

      private static final long serialVersionUID = 1L;


      String SMTP_HOST_NAME = "smtp.gmail.com";

       String SMTP_PORT = "465";


       String emailSubjectTxt = "hi";

       String emailFromAddress = "shiva.1000projects@gmail.com";

       String pwd="9989765191";

       String SSL_FACTORY = "javax.net.ssl.SSLSocketFactory";

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

             emailFromAddress=request.getParameter("from");

             pwd=request.getParameter("pwd");

             System.out.println(emailFromAddress);

             System.out.println(pwd);

             String emailMsgTxt=request.getParameter("msg");

                   String gmail=request.getParameter("mail");

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

                   emailSubjectTxt=request.getParameter("sub");

                  String[] sendTo = new String[1];

                  sendTo[0]=gmail;


                  try {

                       sendSSLMessage(sendTo, emailSubjectTxt, emailMsgTxt,
emailFromAddress);

                       response.sendRedirect("home.html");
```

```
                    } catch (MessagingException e) {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }


                    }
                    public void sendSSLMessage(String recipients[], String
subject,
                                String message, String from) throws
MessagingException {
                          boolean debug = true;
                          Properties props = new Properties();
                          props.put("mail.smtp.host", SMTP_HOST_NAME);
                          props.put("mail.smtp.auth", "true");
                          props.put("mail.debug", "true");
                          props.put("mail.smtp.port", SMTP_PORT);
                          props.put("mail.smtp.socketFactory.port", SMTP_PORT);
                          props.put("mail.smtp.socketFactory.class",
SSL_FACTORY);
                          props.put("mail.smtp.socketFactory.fallback",
"false");

                          System.out.println("from"+emailFromAddress);
                          System.out.println("from"+pwd);
                          Session session = Session.getDefaultInstance(props,
                                      new javax.mail.Authenticator() {




                                protected PasswordAuthentication
getPasswordAuthentication() {
                                                  return new
PasswordAuthentication(
                                                              emailFromAddress,
pwd);
                                            }
                                      });

                          session.setDebug(debug);

                          Message msg = new MimeMessage(session);
                          InternetAddress addressFrom = new
InternetAddress(from);
                          msg.setFrom(addressFrom);

                          InternetAddress[] addressTo = new
InternetAddress[recipients.length];
                          for (int i = 0; i < recipients.length; i++) {
                                addressTo[i] = new
InternetAddress(recipients[i]);
                          }
                          msg.setRecipients(Message.RecipientType.TO,
addressTo);

                          // Setting the Subject and Content Type
                          msg.setSubject(subject);
                          msg.setContent(message, "text/plain");
                          Transport.send(msg);

         }

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

        }
```

```
}
```

File Name:- Register.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>



<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Registration</title>

<link rel="stylesheet" href="style.css">

<script language="javascript">


function validation()

{




      if(document.register.mail.value=="" ||
document.register.sub.value==""||document.register.msg.value=="")

      {

            alert("Please Fill The Data");

            return false;

      }

      else

{

                  var x=document.register.mail.value;

                  var atpos=x.indexOf("@");

                  var dotpos=x.lastIndexOf(".");

                  if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)

                    {
```

```
                    alert("Not a valid e-mail address");
                    return false;
                    }
                  else
                  {
                        return true;
                  }
            }


            }
</script>




</head>


<style type="text/css">
* {margin:0;padding:0;outline: none;}


body {color: #1c1d21;background: linear-gradient(to right, #22c1c3,
#fdbb2d);font: normal 13px/1.3em arial, helvetica, sans-serif;
 word-spacing: 0.1em;margin:10px 0 0;padding:0;text-align:center;}


h2 {color: #fdab4f;font-size: 2.8em;font-weight: bold;margin: 0;line-height:
0.7em;}
h3 {color: #86c303;font-size: 1.3em;margin: 0;}
h4 {color: #86c303;font-size: 1.2em;margin: 0;}
h1, h2, h3, h4 {font-family:"trebuchet MS";}
a, a:link, a:visited {color: #fd7638;text-decoration: none;}
a:hover {color: #fdab4f;text-decoration: underline;}
p { margin: 0 0 18px 0;}
```

```
span.backgroundcolor

{

background-color: #fdab4f;

}

#wrap {width:790px;height: 100%;margin: 0 auto;text-align:left;}

#top {height:400px;width:100%;background:url(frog.jpg) no-repeat top left}

#top h1 {margin:0;padding: 70px  ;line-height:100%;color: #7bbc06;font:30px
"comic sans MS", san-serif;}

#main {float:left;width: 100%;}

#content {margin-right:294px;}

#side{float: left;width: 294px;margin-left: -294px;}

#nav {background: #86c303 url(bg2.jpg);margin-top:-0px;}

#nav ul {list-style-type: none;margin: 0;padding: 5px 10px;}

#nav li a,#nav li a:link,#nav li a:visited  {

      color: #fff;line-height: 1.8em;

      padding: 0 0 0 19px;text-decoration: none;text-transform:
uppercase;font-weight:bold;}

#nav li a:hover {text-decoration: underline;}

#footer {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: #86c303 url(bg3.jpg) ;

      text-align:center;padding: 8px 0;margin:0;width: 100%;}

#header {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: orange url(bg3.jpg) ;

      text-align:center;font-size: 2.8em;font-weight: bold;padding: 8px
0;margin:0;width: 100%;}

      #header2 {background: #7bbc06;color: #fff;clear:both;line-
height:10px;background: orange url(bg3.jpg) ;

      font-size: 1em;font-weight: bold;padding: 8px 0;margin:0;width: 100%;}

#footer a, #footer a:link, #footer a:visited {

      color: #fff;font-weight: bold;text-decoration: none;}

#footer a:hover {text-decoration:underline;}


.text {padding: 16px 16px 16px 0;}
```

```
.contentBox{background:#fdab4f url(bg1.jpg);

  margin: 0 0 16px 0;padding: 5px 10px;}

  .head{background:#fdab4f;

  margin: 0 0 16px 0;padding: 15px 6px;}

.contentBox p {margin: 0;}

.sideBox {border: 2px solid #86c303;font-family:arial;

     border-bottom-width: 2px;margin: 16px 0 0 0;padding: 5px 10px;}

.more {font-size:110%;padding-left:20px;}


/* ******** the W3C checker does not allow these border shaping declarations
- but they look good (not IE)

#nav {-moz-border-radius: 1em 4em 1em 1em;border-radius: 1em 4em 1em 1em;}

#footer {-moz-border-radius: 1em 4em 1em 4em;border-radius: 1em 4em 1em 4em;}

.contentBox{-moz-border-radius: 1em 1em 1em 1em;border-radius:1em 1em 1em 1em
;}

*/


</style>


<body>


<div id="wrap">

<div id="header">Personal Mail Security</div>

<div id="header2"><table><tr><td align="left"><a
href="loginsuccess.jsp"><font
color="#3322dd"><b>SendMail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;<a href="Request.jsp"><font
color="#3322dd"><b>Searchkey</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td><td align="center"><a
href="decrypt.jsp"><font
color="#3322dd"><b>DecryptMail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
```

```
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td><td align="right"><a
href="home.html"><font
color="#3322dd"><b>SignOut</b></font></a></td></tr></table></div>

<center>

<br><br>

<div class="sideBox">

<br><font  size="+1.2"
color="#660000"><b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<u>Send Mail
Here</u></b></font><br><br>



<table cellspacing="15">

<tr><td>

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

<%@ include file="connect.jsp" %>



<form name="form1" method="post" action="Request.jsp">

  <table width="408" border="0" align="center">

    <tr>

      <td width="233"><span class="style3">Enter Your Password to Get Secret
Key:-</span></td>

      <td width="165"><label>

        <input type="password" name="pwd">

      </label></td>

    </tr>

    <tr>

      <td>&nbsp;</td>

      <td><input type="submit" name="Submit" value="Submit"></td>

    </tr>

    <tr>

      <td>&nbsp;</td>

      <td><label></label></td>
    </tr>
  </table>
  <p>&nbsp;</p>
  <p>&nbsp;</p>
</form>
</body>
</html>



<%



      try
{






      HttpSession ses = request.getSession(true);
                String gmail = ses.getAttribute("gmail").toString();
            String pwd= request.getParameter("pwd");


            if(pwd.equalsIgnoreCase(""))
            {


            }
            else
            {
             Statement stmt=connection.createStatement();
                String strQuery = "select key1 from register where
gmail='"+gmail+"' and pwd='"+pwd+"' ";
```

```
                ResultSet rs = stmt.executeQuery(strQuery);
                if(rs.next()==true){

                  String key1 =rs.getString(1);
                   System.out.println("Your Secret Key is->"+key1);
                }
                else
                {


                System.out.println("Secret Key Mismatch....");
                    %><p><a href="Search.jsp">Back</a></p>

                    </body>
                    </html>
                    <%
                }
            }

             }

        catch(Exception e)
        {
          System.out.println(e.getMessage());
        }
%>



        </div>
        <div class="article">
```

```
          <h2>&nbsp;</h2>
          <div class="clr"></div>
          <p>&nbsp;</p>
        </div>
      </div>



</body>
</html>
```

File Name :Connect.jsp

```
<%@ page import="java.sql.*"%>
<%@ page import="java.util.*" %>
<%
Connection connection = null;
  try {

      // ORACLE DBASE CONECTIVITY
/*
       Class.forName("oracle.jdbc.driver.OracleDriver");
      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      connection =  DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:insurance","scott", "tiger");
*/



/*
      //MY SQL DBASE CONNECTIVITY

       Class.forName("com.mysql.jdbc.Driver");
Connection=DriverManager.getConnection("jdbc:mysql://localhost:5053/AMES");

*/

  // MS ACCESS and MYSQL DBASE CONNECTIVITY

        Class.forName("com.mysql.jdbc.Driver");
connection = 
DriverManager.getConnection("jdbc:mysql://localhost:3306/gmail","root","root"
);
      String sql="";

}
catch(Exception e)
{
System.out.println(e);
}
%>
```

File Name: DataSet.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ page import="java.util.*" %>
        <%@ page import="java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" 
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
```

```
<body>

<%

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

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

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

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

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

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

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

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

String dob=day+"/"+month+"/"+year;

UUID idOne = UUID.randomUUID();

String hai=idOne.toString();

String u1=hai.substring(0,7);

String key1=u1.toUpperCase();

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

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/gmail","root","r
oot");

PreparedStatement pst=con.prepareStatement("insert into register
values(?,?,?,?,?,?,?)");

pst.setString(1,name);

pst.setString(2,mail);

pst.setString(3,pwd);

pst.setString(4,dob);

pst.setString(5,phone);

pst.setString(6,proff);

pst.setString(7,key1);

int i=pst.executeUpdate();

if(i>0)

{

response.sendRedirect("home.html?keymsg="+key1+"&mail="+mail+"&pwd="+pwd);
}
else
{
      System.out.println("not inserted ");
}
%>
</body>
</html>
```

Filename:decrypt.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.security.Key"%>

<%@ page import="javax.crypto.Cipher"%>
<%@ page import="javax.crypto.spec.SecretKeySpec"%>

<%@ page import="sun.misc.BASE64Decoder"%>
<%@ page import="sun.misc.BASE64Encoder"%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Registration</title>
<link rel="stylesheet" href="style.css">
<script language="javascript">
```

```
function validation()
{



      if(document.register.key.value=="" || document.register.enc.value=="")
      {
            alert("Please Fill The Data");
            return false;
      }

                  else
                  {
                        return true;
                  }
            }

</script>



</head>

<style type="text/css">
* {margin:0;padding:0;outline: none;}

body {color: #1c1d21;background: linear-gradient(to right, #22c1c3,
#fdbb2d);font: normal 13px/1.3em arial, helvetica, sans-serif;
 word-spacing: 0.1em;margin:10px 0 0;padding:0;text-align:center;}
```

```
h2 {color: #fdab4f;font-size: 2.8em;font-weight: bold;margin: 0;line-height: 
0.7em;}

h3 {color: #86c303;font-size: 1.3em;margin: 0;}

h4 {color: #86c303;font-size: 1.2em;margin: 0;}

h1, h2, h3, h4 {font-family:"trebuchet MS";}

a, a:link, a:visited {color: #fd7638;text-decoration: none;}

a:hover {color: #fdab4f;text-decoration: underline;}

p { margin: 0 0 18px 0;}

span.backgroundcolor

{

background-color: #fdab4f;

}

#wrap {width:790px;height: 100%;margin: 0 auto;text-align:left;}

#top {height:400px;width:100%;background:url(frog.jpg) no-repeat top left}

#top h1 {margin:0;padding: 70px  ;line-height:100%;color: #7bbc06;font:30px 
"comic sans MS", san-serif;}

#main {float:left;width: 100%;}

#content {margin-right:294px;}

#side{float: left;width: 294px;margin-left: -294px;}

#nav {background: #86c303 url(bg2.jpg);margin-top:-0px;}

#nav ul {list-style-type: none;margin: 0;padding: 5px 10px;}

#nav li a,#nav li a:link,#nav li a:visited  {

      color: #fff;line-height: 1.8em;

      padding: 0 0 0 19px;text-decoration: none;text-transform: 
uppercase;font-weight:bold;}

#nav li a:hover {text-decoration: underline;}

#footer {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: #86c303 url(bg3.jpg) ;

      text-align:center;padding: 8px 0;margin:0;width: 100%;}

#header {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: orange url(bg3.jpg) ;

      text-align:center;font-size: 2.8em;font-weight: bold;padding: 8px 
0;margin:0;width: 100%;}
```

```
      #header2 {background: #7bbc06;color: #fff;clear:both;line-
height:10px;background: orange url(bg3.jpg) ;

      font-size: 1em;font-weight: bold;padding: 8px 0;margin:0;width: 100%;}

#footer a, #footer a:link, #footer a:visited {

      color: #fff;font-weight: bold;text-decoration: none;}

#footer a:hover {text-decoration:underline;}


.text {padding: 16px 16px 16px 0;}

.contentBox{background:#fdab4f url(bg1.jpg);

  margin: 0 0 16px 0;padding: 5px 10px;}

  .head{background:#fdab4f;

  margin: 0 0 16px 0;padding: 15px 6px;}

.contentBox p {margin: 0;}

.sideBox {border: 2px solid #86c303;font-family:arial;

      border-bottom-width: 2px;margin: 16px 0 0 0;padding: 5px 10px;}

.more {font-size:110%;padding-left:20px;}


/* ******** the W3C checker does not allow these border shaping declarations
- but they look good (not IE)

#nav {-moz-border-radius: 1em 4em 1em 1em;border-radius: 1em 4em 1em 1em;}

#footer {-moz-border-radius: 1em 4em 1em 4em;border-radius: 1em 4em 1em 4em;}

.contentBox{-moz-border-radius: 1em 1em 1em 1em;border-radius:1em 1em 1em 1em
;}

*/


</style>


<body>

<div id="wrap">

<div id="header">Personal Mail Security</div>

<div id="header2"><table><tr><td align="left"><a
href="loginsuccess.jsp"><font
color="#3322dd"><b>SendMail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
```

```
;&nbsp;</a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td><td align="center"><a
href="decrypt.jsp"><font
color="#3322dd"><b>Decryptmail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp</td><td align="right"><a
href="home.html"><font
color="#3322dd"><b>SignOut</b></font></a></td></tr></table></div>

<center>

<br><br>

<div class="sideBox">

<br><font  size="+1.2" color="#660000"><b>&nbsp;&nbsp;<u>Decrypt
Mail</u></b></font><br><br>

<form name="register" method="post" action="./decryptedmsg" onsubmit="return
validation()">



<table cellspacing="15">

<tr><td>

<b>Encrypted Msg:</b>

</td>

<td><textarea rows="7" cols="70" name="enc"></textarea></td></tr>

<tr><td><b>Key:</b></td><td><input type="text" name="encdeckey"></td></tr>

<tr><td><b>Message:</b></td><td><textarea rows="7" cols="70"
name="msg"></textarea></td></tr>

<tr><td align="right">&nbsp;&nbsp;&nbsp;<input type="submit" value="Decrypt"
></td><td align="center"><input type="reset" value="reset"></td></tr>

</table>

</form>



<div id="footer"> Security <strong>Mail Communicator</strong>    <!-- your
site name here  -->



</div>
```

```
</div>


</center>


</div>


</body>
</html>
```

FileName:decrypt2.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.security.Key"%>

<%@ page import="javax.crypto.Cipher"%>
<%@ page import="javax.crypto.spec.SecretKeySpec"%>

<%@ page import="sun.misc.BASE64Decoder"%>
<%@ page import="sun.misc.BASE64Encoder"%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Registration</title>
<link rel="stylesheet" href="style.css">
<script language="javascript">

function validation()
{
      if(document.register.key.value=="" ||
document.register.enc.value=="")

      {

            alert("Please Fill The Data");

            return false;

      }


                  else

                  {

                        return true;

                  }

            }


</script>




</head>


<style type="text/css">

* {margin:0;padding:0;outline: none;}


body {color: #1c1d21;background: linear-gradient(to right, #22c1c3,
#fdbb2d);font: normal 13px/1.3em arial, helvetica, sans-serif;

 word-spacing: 0.1em;margin:10px 0 0;padding:0;text-align:center;}


h2 {color: #fdab4f;font-size: 2.8em;font-weight: bold;margin: 0;line-
height: 0.7em;}

h3 {color: #86c303;font-size: 1.3em;margin: 0;}

h4 {color: #86c303;font-size: 1.2em;margin: 0;}
```

```
h1, h2, h3, h4 {font-family:"trebuchet MS";}

a, a:link, a:visited {color: #fd7638;text-decoration: none;}

a:hover {color: #fdab4f;text-decoration: underline;}

p { margin: 0 0 18px 0;}

span.backgroundcolor

{

background-color: #fdab4f;

}

#wrap {width:790px;height: 100%;margin: 0 auto;text-align:left;}

#top {height:400px;width:100%;background:url(frog.jpg) no-repeat top
left}

#top h1 {margin:0;padding: 70px  ;line-height:100%;color:
#7bbc06;font:30px "comic sans MS", san-serif;}

#main {float:left;width: 100%;}

#content {margin-right:294px;}

#side{float: left;width: 294px;margin-left: -294px;}

#nav {background: #86c303 url(bg2.jpg);margin-top:-0px;}

#nav ul {list-style-type: none;margin: 0;padding: 5px 10px;}

#nav li a,#nav li a:link,#nav li a:visited  {

     color: #fff;line-height: 1.8em;

     padding: 0 0 0 19px;text-decoration: none;text-transform:
uppercase;font-weight:bold;}

#nav li a:hover {text-decoration: underline;}

#footer {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: #86c303 url(bg3.jpg) ;

     text-align:center;padding: 8px 0;margin:0;width: 100%;}

#header {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: orange url(bg3.jpg) ;

     text-align:center;font-size: 2.8em;font-weight: bold;padding: 8px
0;margin:0;width: 100%;}

     #header2 {background: #7bbc06;color: #fff;clear:both;line-
height:10px;background: orange url(bg3.jpg) ;
```

```
      font-size: 1em;font-weight: bold;padding: 8px 0;margin:0;width:
100%;}

#footer a, #footer a:link, #footer a:visited {

      color: #fff;font-weight: bold;text-decoration: none;}

#footer a:hover {text-decoration:underline;}


.text {padding: 16px 16px 16px 0;}

.contentBox{background:#fdab4f url(bg1.jpg);

  margin: 0 0 16px 0;padding: 5px 10px;}

  .head{background:#fdab4f;

  margin: 0 0 16px 0;padding: 15px 6px;}

.contentBox p {margin: 0;}

.sideBox {border: 2px solid #86c303;font-family:arial;

      border-bottom-width: 2px;margin: 16px 0 0 0;padding: 5px 10px;}

.more {font-size:110%;padding-left:20px;}


/* ******** the W3C checker does not allow these border shaping
declarations  - but they look good (not IE)

#nav {-moz-border-radius: 1em 4em 1em 1em;border-radius: 1em 4em 1em
1em;}

#footer {-moz-border-radius: 1em 4em 1em 4em;border-radius: 1em 4em
1em 4em;}

.contentBox{-moz-border-radius: 1em 1em 1em 1em;border-radius:1em 1em
1em 1em ;}

*/


</style>


<body>

<%

String enc=request.getParameter("enc");
```

```
System.out.println(enc);

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

String msg3=request.getParameter("msg");

String msg="";

if(!enc.equals(""))

{

      try{

            String ALGORITHM = "AES";

         int ITERATIONS = 1;

         byte[] keyValue = new byte[] { 'T', 'h', 'i', 's', 'I', 's',
'A', 'S', 'e', 'c', 'r', 'e', 't', 'K', 'e', 'y'};

 Key key = new SecretKeySpec(keyValue, ALGORITHM);

 Cipher c = Cipher.getInstance(ALGORITHM);

 c.init(Cipher.DECRYPT_MODE, key);


 String dValue = null;

 String valueToDecrypt = enc;

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

     byte[] decordedValue = new
BASE64Decoder().decodeBuffer(valueToDecrypt);

     byte[] decValue = c.doFinal(decordedValue);

     System.out.println(decValue);

     dValue = new String(decValue);

     valueToDecrypt = dValue;

     System.out.println(dValue);

     msg=dValue;

 }

      }

 catch(Exception e)

 {
       e.printStackTrace();

 }

 }

%>

<div id="wrap">

<div id="header">Personal Mail Security</div>

<div id="header2"><table><tr><td align="left"><a
href="loginsuccess.jsp"><font
color="#3322dd"><b>SendMail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td><td align="center"><a
href="decrypt.jsp"><font
color="#3322dd"><b>DecryptMail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td><td align="right"><a
href="home.html"><font
color="#3322dd"><b>SignOut</b></font></a></td></tr></table></div>

<center>

<br><br>

<div class="sideBox">

<br><font  size="+1.2"
color="#660000"><b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<u>Decryp
tMail</u></b></font><br><br>

<form name="register" method="post" action="decryptedmsg"
onsubmit="return validation()">



<table cellspacing="15">

<tr><td>

<b>Encrypted Msg:</b>

</td>
```

```
<td><textarea rows="7" cols="70"
name="enc"><%=enc%></textarea></td></tr>

<tr><td><b>PrivateKey:</b></td><td><input type="text" name="encdeckey"
value="<%=key2 %>"></td></tr>

<tr><td><b>Message:</b></td><td><textarea rows="7" cols="70"
name="msg"><%=msg3%></textarea></td></tr>

<tr><td align="right">&nbsp;&nbsp;&nbsp;<input type="submit"
value="Decrypt" ></td><td align="center"><input type="reset"
value="reset"></td></tr>

</table>

</form>

<div id="footer"> Â© 2021 Copyright <strong>Coign Consultants
Pvt.LTD.</strong>    <!-- your site name here  -->

<a href="http://www.coign.net/">( www.cogin.net )</a>

</div>

</div>



</center>



</div>



</body>

</html>
```

Filename: login.jsp

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

    pageEncoding="ISO-8859-1"%>

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

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



<%@page import="java.sql.DriverManager"%><html>
```

```
<head>

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

<title>Insert title here</title>

</head>

<body>

<%

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

String pwd=request.getParameter("password");

request.getSession().setAttribute("gmail",mail);

request.getSession().setAttribute("pwd",pwd);

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

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/gmail","root","r
oot");

PreparedStatement pst=con.prepareStatement("select * from register where
gmail=? && pwd=? ");

pst.setString(1,mail);

pst.setString(2,pwd);

ResultSet rs=pst.executeQuery();

int i=0;

if(rs.next())

{

	i++;

}

if(i>0)

{%>

	<jsp:forward page="loginsuccess.jsp"></jsp:forward>

<%}else

{

	System.out.println("not success");

}
```

```
%>
</body>
</html>
```

Filename:loginaccess.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>



<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Registration</title>
<link rel="stylesheet" href="style.css">
<script language="javascript">

function validation()
{



	if(document.register.mail.value=="" ||
document.register.sub.value==""||document.register.msg.value=="")
	{
		alert("Please Fill The Data");
		return false;
	}
	else
{
                  var x=document.register.mail.value;
                  var atpos=x.indexOf("@");
                  var dotpos=x.lastIndexOf(".");
                  if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
                    {
                    alert("Not a valid e-mail address");
                    return false;
                    }
                  else
                  {
                        return true;
                  }
            }


            }
</script>




</head>


<style type="text/css">
* {margin:0;padding:0;outline: none;}


body {color: #1c1d21;background: linear-gradient(to right, #22c1c3,
#fdbb2d);font: normal 13px/1.3em arial, helvetica, sans-serif;
 word-spacing: 0.1em;margin:10px 0 0;padding:0;text-align:center;}


h2 {color: #fdab4f;font-size: 2.8em;font-weight: bold;margin: 0;line-height:
0.7em;}
```

```
h3 {color: #86c303;font-size: 1.3em;margin: 0;}

h4 {color: #86c303;font-size: 1.2em;margin: 0;}

h1, h2, h3, h4 {font-family:"trebuchet MS";}

a, a:link, a:visited {color: #fd7638;text-decoration: none;}

a:hover {color: #fdab4f;text-decoration: underline;}

p { margin: 0 0 18px 0;}

span.backgroundcolor

{

background-color: #fdab4f;

}

#wrap {width:790px;height: 100%;margin: 0 auto;text-align:left;}

#top {height:400px;width:100%;background:url(frog.jpg) no-repeat top left}

#top h1 {margin:0;padding: 70px  ;line-height:100%;color: #7bbc06;font:30px
"comic sans MS", san-serif;}

#main {float:left;width: 100%;}

#content {margin-right:294px;}

#side{float: left;width: 294px;margin-left: -294px;}

#nav {background: #86c303 url(bg2.jpg);margin-top:-0px;}

#nav ul {list-style-type: none;margin: 0;padding: 5px 10px;}

#nav li a,#nav li a:link,#nav li a:visited  {

      color: #fff;line-height: 1.8em;

      padding: 0 0 0 19px;text-decoration: none;text-transform:
uppercase;font-weight:bold;}

#nav li a:hover {text-decoration: underline;}

#footer {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: #86c303 url(bg3.jpg) ;

      text-align:center;padding: 8px 0;margin:0;width: 100%;}

#header {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: orange url(bg3.jpg) ;

      text-align:center;font-size: 2.8em;font-weight: bold;padding: 8px
0;margin:0;width: 100%;}

      #header2 {background: #7bbc06;color: #fff;clear:both;line-
height:10px;background: orange url(bg3.jpg) ;
```

```
      font-size: 1em;font-weight: bold;padding: 8px 0;margin:0;width: 100%;}
#footer a, #footer a:link, #footer a:visited {
      color: #fff;font-weight: bold;text-decoration: none;}
#footer a:hover {text-decoration:underline;}


.text {padding: 16px 16px 16px 0;}
.contentBox{background:#fdab4f url(bg1.jpg);
  margin: 0 0 16px 0;padding: 5px 10px;}
  .head{background:#fdab4f;
  margin: 0 0 16px 0;padding: 15px 6px;}
.contentBox p {margin: 0;}
.sideBox {border: 2px solid #86c303;font-family:arial;
      border-bottom-width: 2px;margin: 16px 0 0 0;padding: 5px 10px;}
.more {font-size:110%;padding-left:20px;}


/* ******** the W3C checker does not allow these border shaping declarations
- but they look good (not IE)
#nav {-moz-border-radius: 1em 4em 1em 1em;border-radius: 1em 4em 1em 1em;}
#footer {-moz-border-radius: 1em 4em 1em 4em;border-radius: 1em 4em 1em 4em;}
.contentBox{-moz-border-radius: 1em 1em 1em 1em;border-radius:1em 1em 1em 1em
;}
*/


</style>
 <%
                HttpSession ses = request.getSession(true);
                String mail = ses.getAttribute("gmail").toString();
    %>
<body>
    welcome<%=mail%>
<div id="wrap">
```

```
<div id="header">Personal Mail Security</div>

<div id="header2"><table><tr><td align="left"><a
href="loginsuccess.jsp"><font
color="#3322dd"><b>SendMail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;<a href="Request.jsp"><font
color="#3322dd"><b>Searchkey</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nb
sp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td><td align="center"><a
href="decrypt.jsp"><font
color="#3322dd"><b>DecryptMail</b></font></a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td><td
align="right"><a href="home.html"><font
color="#3322dd"><b>SignOut</b></font></a></td></tr></table></div>

<center>

<br><br>

<div class="sideBox">

<br><font  size="+1.2"
color="#660000"><b>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<u>Send Mail
Here</u></b></font><br><br>

<form name="register" method="post" action="sendmail.jsp" onsubmit="return
validation()">



<table cellspacing="15">

<tr><td>

<b>Send To:</b>

</td>

<td><input type="text" name="mail"></td></tr>

<tr><td><b>Subject:</b></td><td><input type="text" name="sub"></td></tr>

<tr><td><b>Message:</b></td><td><textarea rows="10" cols="20"
name="msg"></textarea></td></tr>

<tr><td align="right">&nbsp;&nbsp;&nbsp;<input type="submit" value="Send
Mail" ></td><td align="center"><input type="reset" value="reset"></td></tr>

</table>

</form>

 <div id="footer"> personal Mail Security</strong>
```

```
</div>
</div>


</center>


</div>


</body>
</html>
```

File name:Register.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>



<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Registration</title>
<link rel="stylesheet" href="style.css">
<script language="javascript">
function tab()
{
      if(document.register.name.value=="")
      {
            document.getElementById("loc").innerHTML="*Please Enter Name";
            return false;



      }
      else{
            document.getElementById("loc").innerHTML="";
      }
      return true;
}
function tab1(){
      if(document.register.pwd.value=="")
      {
            document.getElementById("pwdd").innerHTML="*Please Enter
Password";
      return false;
      }else{
      document.getElementById("pwdd").innerHTML="";
      }
      return true;
}
function tab2(){
      if(document.register.pwd2.value=="")
      {
            document.getElementById("pwdd2").innerHTML="*Please Enter Re-
Password";
            return false;

      }else{
            document.getElementById("pwdd2").innerHTML="";
      }
      return true;
}
function tab4(){

if(document.register.mail.value=="")
{
        document.getElementById("mail2").innerHTML="*Please Enter Re-Password";
        return false;

    }else{
        document.getElementById("mail2").innerHTML="";
    }

    return true;
    }
    function phonee(){

        if(document.register.phone.value=="")
        {
            document.getElementById("phone2").innerHTML="*Please Enter phone
    number";
            return false;

        }else{
            document.getElementById("phone2").innerHTML="";
        }

        return true;
        }

    function validation()
    {



        if(document.register.name.value=="" ||
    document.register.pwd.value==""||document.register.pwd2.value=="" ||
    document.register.mail.value==""||document.register.phone.value=="")
        {
            alert("Please Fill The Data");
            return false;
      }
      else
{
            if(document.register.pwd.value==document.register.pwd2.value)
            {
                  var x=document.register.mail.value;
                  var atpos=x.indexOf("@");
                  var dotpos=x.lastIndexOf(".");
                  if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
                    {
                    alert("Not a valid e-mail address");
                    return false;
                    }
                  else
                  {
                        return true;
                  }
            }
            else{
            return false;
}
            }

}
</script>
```

```
</head>


<style type="text/css">

* {margin:0;padding:0;outline: none;}


body {color: #1c1d21;background: linear-gradient(to right, #22c1c3,
#fdbb2d);font: normal 13px/1.3em arial, helvetica, sans-serif;

 word-spacing: 0.1em;margin:10px 0 0;padding:0;text-align:center;}


h2 {color: #fdab4f;font-size: 2.8em;font-weight: bold;margin: 0;line-height:
0.7em;}

h3 {color: #86c303;font-size: 1.3em;margin: 0;}

h4 {color: #86c303;font-size: 1.2em;margin: 0;}

h1, h2, h3, h4 {font-family:"trebuchet MS";}

a, a:link, a:visited {color: #fd7638;text-decoration: none;}

a:hover {color: #fdab4f;text-decoration: underline;}

p { margin: 0 0 18px 0;}

span.backgroundcolor

{

background: linear-gradient(to right, #22c1c3, #fdbb2d);

}

#wrap {width:790px;height: 100%;margin: 0 auto;text-align:left;}

#top {height:400px;width:100%;background:url(frog.jpg) no-repeat top left}

#top h1 {margin:0;padding: 70px  ;line-height:100%;color: #7bbc06;font:30px
"comic sans MS", san-serif;}

#main {float:left;width: 100%;}

#content {margin-right:294px;}

#side{float: left;width: 294px;margin-left: -294px;}

#nav {background: #86c303 url(bg2.jpg);margin-top:-0px;}

#nav ul {list-style-type: none;margin: 0;padding: 5px 10px;}

#nav li a,#nav li a:link,#nav li a:visited  {

      color: #fff;line-height: 1.8em;
      padding: 0 0 0 19px;text-decoration: none;text-transform:
uppercase;font-weight:bold;}

#nav li a:hover {text-decoration: underline;}

#footer {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: #86c303 url(bg3.jpg) ;

      text-align:center;padding: 8px 0;margin:0;width: 100%;}

#header {background: #7bbc06;color: #fff;clear:both;line-
height:26px;background: orange url(bg3.jpg) ;

      text-align:center;font-size: 2.8em;font-weight: bold;padding: 8px
0;margin:0;width: 100%;}

#footer a, #footer a:link, #footer a:visited {

      color: #fff;font-weight: bold;text-decoration: none;}

#footer a:hover {text-decoration:underline;}


.text {padding: 16px 16px 16px 0;}

.contentBox{background:#fdab4f url(bg1.jpg);

  margin: 0 0 16px 0;padding: 5px 10px;}

  .head{background:#fdab4f;

  margin: 0 0 16px 0;padding: 15px 6px;}

.contentBox p {margin: 0;}

.sideBox {border: 2px solid #86c303;font-family:arial;

      border-bottom-width: 2px;margin: 16px 0 0 0;padding: 5px 10px;}

.more {font-size:110%;padding-left:20px;}


/* ******** the W3C checker does not allow these border shaping declarations
- but they look good (not IE)

#nav {-moz-border-radius: 1em 4em 1em 1em;border-radius: 1em 4em 1em 1em;}

#footer {-moz-border-radius: 1em 4em 1em 4em;border-radius: 1em 4em 1em 4em;}

.contentBox{-moz-border-radius: 1em 1em 1em 1em;border-radius:1em 1em 1em 1em
;}

*/


</style>
```

```
<body background: linear-gradient(to right, #22c1c3, #fdbb2d);>

<div id="wrap">

<div id="header">Personal Mail Security</div>

<center>

<div class="sideBox">

<form name="register" method="post" action="datainsert.jsp" onsubmit="return
validation()">

<table cellspacing="15">

<tr >

<td>Name</td>

<td><input type="text" name="name" onblur="tab()">&nbsp;&nbsp;&nbsp;<span
id="loc" style="color : #f00;">&nbsp;</span></td>

</tr>

<tr>

<td>GMail</td>

<td > <input type="text"  name="mail" onblur="tab4()">&nbsp;&nbsp;&nbsp;<span
id="mail2" style="color : #f00;"></span></td>

</tr>

<tr>

<td>Password</td>

<td > <input type="password" name="pwd"
onblur="tab1()">&nbsp;&nbsp;&nbsp;<span id="pwdd" style="color :
#f00;"></span></td>

</tr>

<tr>

<td>Re-Password</td>

<td > <input type="password" name="pwd2"
onblur="tab2()">&nbsp;&nbsp;&nbsp;<span id="pwdd2" style="color :
#f00;"></span></td>

</tr>



<tr>

<td>DOB</td>
```

```
<td><select name="day" id="Day">

                    <option label="1" value="1">1</option>

                    <option label="2" value="2">2</option>

                    <option label="3" value="3">3</option>

                    <option label="4" value="4">4</option>

                    <option label="5" value="5">5</option>

                    <option label="6" value="6">6</option>

                    <option label="7" value="7">7</option>

                    <option label="8" value="8">8</option>

                    <option label="9" value="9">9</option>

                    <option label="10" value="10">10</option>

                    <option label="11" value="11">11</option>

                    <option label="12" value="12">12</option>

                    <option label="13" value="13">13</option>

                    <option label="14" value="14">14</option>
```

```
<option label="15" value="15">15</option>

<option label="16" value="16">16</option>

<option label="17" value="17">17</option>

<option label="18" value="18">18</option>

<option label="19" value="19">19</option>

<option label="20" value="20">20</option>

<option label="21" value="21">21</option>

<option label="22" value="22">22</option>

<option label="23" value="23">23</option>

<option label="24" value="24">24</option>

<option label="25" value="25">25</option>

<option label="26" value="26">26</option>

<option label="27" value="27">27</option>

<option label="28" value="28">28</option>

<option label="29" value="29">29</option>
```

```
  <option label="30" value="30">30</option>

  <option label="31" value="31">31</option>

</select>

<select name="month" id="month"  >

  <option label="01" value="01">01</option>

  <option label="02" value="02">02</option>

  <option label="03" value="03">03</option>

  <option label="04" value="04">04</option>

  <option label="05" value="05">05</option>

  <option label="06" value="06">06</option>

  <option label="07" value="07">07</option>

  <option label="08" value="08">08</option>

  <option label="09" value="09">09</option>

  <option label="10" value="10">10</option>

  <option label="11" value="11">11</option>
```

```
        <option label="12" value="12">12</option>

      </select>

    <select name="year" id="year" >

        <option label="1902" value="1902">1902</option>

        <option label="1903" value="1903">1903</option>

        <option label="1904" value="1904">1904</option>

        <option label="1905" value="1905">1905</option>

        <option label="1906" value="1906">1906</option>

        <option label="1907" value="1907">1907</option>

        <option label="1908" value="1908">1908</option>

        <option label="1909" value="1909">1909</option>

        <option label="1910" value="1910">1910</option>

        <option label="1911" value="1911">1911</option>

        <option label="1912" value="1912">1912</option>

        <option label="1913" value="1913">1913</option>
```

```
<option label="1914" value="1914">1914</option>

<option label="1915" value="1915">1915</option>

<option label="1916" value="1916">1916</option>

<option label="1917" value="1917">1917</option>

<option label="1918" value="1918">1918</option>

<option label="1919" value="1919">1919</option>

<option label="1920" value="1920">1920</option>

<option label="1921" value="1921">1921</option>

<option label="1922" value="1922">1922</option>

<option label="1923" value="1923">1923</option>

<option label="1924" value="1924">1924</option>

<option label="1925" value="1925">1925</option>

<option label="1926" value="1926">1926</option>

<option label="1927" value="1927">1927</option>

<option label="1928" value="1928">1928</option>
```

```
<option label="1929" value="1929">1929</option>

<option label="1930" value="1930">1930</option>

<option label="1931" value="1931">1931</option>

<option label="1932" value="1932">1932</option>

<option label="1933" value="1933">1933</option>

<option label="1934" value="1934">1934</option>

<option label="1935" value="1935">1935</option>

<option label="1936" value="1936">1936</option>

<option label="1937" value="1937">1937</option>

<option label="1938" value="1938">1938</option>

<option label="1939" value="1939">1939</option>

<option label="1940" value="1940">1940</option>

<option label="1941" value="1941">1941</option>

<option label="1942" value="1942">1942</option>

<option label="1943" value="1943">1943</option>
```

```
<option label="1944" value="1944">1944</option>

<option label="1945" value="1945">1945</option>

<option label="1946" value="1946">1946</option>

<option label="1947" value="1947">1947</option>

<option label="1948" value="1948">1948</option>

<option label="1949" value="1949">1949</option>

<option label="1950" value="1950">1950</option>

<option label="1951" value="1951">1951</option>

<option label="1952" value="1952">1952</option>

<option label="1953" value="1953">1953</option>

<option label="1954" value="1954">1954</option>

<option label="1955" value="1955">1955</option>

<option label="1956" value="1956">1956</option>

<option label="1957" value="1957">1957</option>

<option label="1958" value="1958">1958</option>
```

```
<option label="1959" value="1959">1959</option>

<option label="1960" value="1960">1960</option>

<option label="1961" value="1961">1961</option>

<option label="1962" value="1962">1962</option>

<option label="1963" value="1963">1963</option>

<option label="1964" value="1964">1964</option>

<option label="1965" value="1965">1965</option>

<option label="1966" value="1966">1966</option>

<option label="1967" value="1967">1967</option>

<option label="1968" value="1968">1968</option>

<option label="1969" value="1969">1969</option>

<option label="1970" value="1970">1970</option>

<option label="1971" value="1971">1971</option>

<option label="1972" value="1972">1972</option>

<option label="1973" value="1973">1973</option>
```

```
<option label="1974" value="1974">1974</option>

<option label="1975" value="1975">1975</option>

<option label="1976" value="1976">1976</option>

<option label="1977" value="1977">1977</option>

<option label="1978" value="1978">1978</option>

<option label="1979" value="1979">1979</option>

<option label="1980" value="1980">1980</option>

<option label="1981" value="1981">1981</option>

<option label="1982" value="1982">1982</option>

<option label="1983" value="1983">1983</option>

<option label="1984" value="1984">1984</option>

<option label="1985" value="1985">1985</option>

<option label="1986" value="1986">1986</option>

<option label="1987" value="1987">1987</option>

<option label="1988" value="1988">1988</option>
```

```
                    <option label="1989" value="1989">1989</option>

                    <option label="1990" value="1990">1990</option>

                    <option label="1984" value="1984">1991</option>

                    <option label="1985" value="1985">1992</option>

                    <option label="1986" value="1986">1993</option>

                    <option label="1987" value="1987">1994</option>

                    <option label="1988" value="1988">1995</option>

                    <option label="1989" value="1989">1996</option>

                    <option label="1990" value="1990">1997</option>

                  </select>


</td>
</tr>
<tr>
<td>Profession</td>
<td><select name="proff">
<option value="0"  selected="selected">---Select---</option>
<option value="1" >Student</option>
<option value="2" >Employee</option>
<option value="3" >Entrepreneur</option>
```

```
<option value="4" >Other</option>

</select>

</td>

</tr>

<tr>

<td>Phone</td>

<td><input type="text" name="phone" onblur="phonee()">&nbsp;&nbsp;&nbsp;<span
id="phone2" style="color : #f00;"></span></td>

</tr>

<tr>

<td>Enter the Below Code</td>

<td><input type="text" name="code"></td>

</tr>


</table>


<br>

<input type='image' src=http://localhost:8084/Personal Mail
Security/CaptchaServlet>

<br><br>

<input type="submit" value="SUBMIT" >

</form>

<br><br>

<%

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


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


  if (captcha != null && code != null) {


    if (captcha.equals(code)) {
        out.print("<p class='alert'>Correct</p>");

    } else {

          out.print("<p class='alert'>Incorrect</p>");

    }

  }

%>

 <div id="footer"> Personal Mail Security</strong>

</div>

</div>


</center>


</div>


</body>

</html>
```

FileName;-searchact.jsp

```
<!DOCTYPE html>

<html lang="en">

<head>



<%@ page import="javax.crypto.Cipher"%>

<%@ page import="javax.crypto.spec.SecretKeySpec"%>


<%@ page import="sun.misc.BASE64Decoder"%>

<%@ page import="sun.misc.BASE64Encoder"%>

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

<%@ include file="connect.jsp" %>
```

```
<%

try

{

HttpSession ses = request.getSession(true);

String gmail = ses.getAttribute("gmail").toString();

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

if(pwd.equalsIgnoreCase(""))

{

}

else

{

Statement stmt=connection.createStatement();

String strQuery = "select key1 from register where gmail='"+gmail+"' and
pwd='"+pwd+"' ";

ResultSet rs = stmt.executeQuery(strQuery);

if(rs.next()==true){

String key1 =rs.getString(1);

out.println("Your Secret Key is->"+key1);

}

else

{
```

```
out.println("Secret Key Mismatch....");
%><p><a href="Search.jsp">Back</a></p>

</body>
</html>
<%
}
}

}

catch(Exception e)
{
out.println(e.getMessage());
}
%>

<center><table style="width:30%" border="2">
<br><br><br><br>
<tr>
<th>Results</th>

</tr>

</table></center>
<br><br>
```

Filename:searchkey.jsp

```
_   pageEncoding="ISO-8859-1"%>
```

```
   <%@ page import="java.security.Key"%>


<%@ page import="javax.crypto.Cipher"%>
<%@ page import="javax.crypto.spec.SecretKeySpec"%>


<%@ page import="sun.misc.BASE64Decoder"%>
<%@ page import="sun.misc.BASE64Encoder"%>
<%@ page import="java.sql.*" %>
<form name="form1" method="post" action="searchact.jsp">
  <table width="408" border="0" align="center">
    <tr>
      <td width="233"><span class="style3">Enter Your Password to Get Secret
Key:-</span></td>
      <td width="165"><label>
        <input type="password" name="pwd">
      </label></td>
    </tr>
    <tr>
      <td>&nbsp;</td>
      <td><input type="submit" name="Submit" value="Submit"></td>
    </tr>
    <tr>
      <td>&nbsp;</td>
      <td><label></label></td>
    </tr>
  </table>
  <p>&nbsp;</p>
  <p>&nbsp;</p>
</form>
<%
```

```
      try
{







            String name=(String)application.getAttribute("name");


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


            if(key1.equalsIgnoreCase(""))

            {


            }

            else

            {

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


      Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/gmail","root","r
oot");

                PreparedStatement pst=con.prepareStatement("select key1 from
register where name='"+name+"' and key1='"+key1+"' ");

                  pst.setString(1, name);

                      pst.setString(2, key1);

                      ResultSet rs=pst.executeQuery();

                if(rs.next()==true){


                  String sk =rs.getString(1);

                   out.println("Your Secret Key is->"+sk);

                }
```

```
                        else
                        {



                        out.println("Secret Key Mismatch....");
                            %><p><a href="Search.jsp">Back</a></p>


                            </body>
                            </html>
                            <%
                        }
                    }


                     }

                catch(Exception e)
                {
                  out.println(e.getMessage());
                }
%>
```

FileName: searchmail.jsp

```
<%@page import="Efficient.mail"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
   <%@ page import="java.security.Key"%>


<%@ page import="javax.crypto.Cipher"%>
<%@ page import="javax.crypto.spec.SecretKeySpec"%>


<%@ page import="sun.misc.BASE64Decoder"%>
<%@ page import="sun.misc.BASE64Encoder"%>

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

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


<%@page import="java.sql.DriverManager"%><html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<%

String ALGORITHM = "AES";

int ITERATIONS = 1;

byte[] keyValue =

    new byte[] { 'T', 'h', 'i', 's', 'I', 's', 'A', 'S', 'e', 'c', 'r', 'e',
't', 'K', 'e', 'y'};


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

String salt="";

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

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/gmail","root","r
oot");

PreparedStatement pst=con.prepareStatement("select * from register where
gmail=?");

pst.setString(1,mail);

ResultSet rs=pst.executeQuery();

if(rs.next())

{

      salt=rs.getString(7);

}

String sub=request.getParameter("sub");
String msg1=request.getParameter("msg");
String msg="";
Key key = new SecretKeySpec(keyValue, ALGORITHM);
Cipher c = Cipher.getInstance(ALGORITHM);
c.init(Cipher.ENCRYPT_MODE, key);
String valueToEnc = null;
String eValue = msg1;
for (int i = 0; i < ITERATIONS; i++) {
    valueToEnc = eValue;
    byte[] encValue = c.doFinal(valueToEnc.getBytes());
    eValue = new BASE64Encoder().encode(encValue);
    msg=eValue;
}
//System.out.println("dskf"+eValue);
String from = (String) session.getAttribute("gmail");
String pwd = (String) session.getAttribute("pwd");
 mail m= new mail();
      m.secretMail(msg, mail, mail);
      response.sendRedirect("loginsuccess.jsp?m1=success");
%>

</body>
</html>
```

FileName:style.css

```
        @CHARSET "ISO-8859-1";
//{ font-size: 12px; font-family: Verdana };


input, textarea { border: 1px solid #ccc }


tr { margin: 5px; padding:5px;}
```

```css
.alert { font-size:15px; color:red; font-weight:bolder }



.body{
    background: linear-gradient(to right, #22c1c3, #fdbb2d) !important;
}
```

Filename:gmail.sql

```sql
/*
SQLyog Community Edition- MySQL GUI v7.15
MySQL - 5.5.29 : Database - gmail
*********************************************************************
*/


/*!40101 SET NAMES utf8 */;

/*!40101 SET SQL_MODE=''*/;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;

CREATE DATABASE /*!32312 IF NOT EXISTS*/`gmail` /*!40100 DEFAULT CHARACTER
SET latin1 */;

USE `gmail`;
```

```
/*Table structure for table `register` */

DROP TABLE IF EXISTS `register`;

CREATE TABLE `register` (
  `name` varchar(50) NOT NULL,
  `gmail` varchar(50) NOT NULL,
  `pwd` varchar(30) NOT NULL,
  `dob` varchar(15) NOT NULL,
  `phone` varchar(15) NOT NULL,
  `proff` varchar(15) NOT NULL,
  `key1` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `register` */

insert  into `register`(`name`,`gmail`,`pwd`,`dob`,`phone`,`proff`,`key1`)
values
('pavan','pavan.coign@gmail.com','pavan31kumar','1/01/1902','9908056223','2',
'6'),('pavan','mallikpavan@gmail.com','android31','12/07/1987','9908056223','
2','0'),('rupa','rupa.coign@gmail.com','sandeepm','14/09/1986','9959123654','
2','7946667'),('rajinikanth','rajinikanth.coign@gmail.com','jackson47','3/04/
1902','9491939191','2','27698'),('nikil','nikhith.1000projects@gmail.com','12
3','2/02/1915','9874561230','1','AB40352'),('raj','nikilp306@gmail.com','123'
,'1/01/1903','9874561230','1','0');

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
```

FileName:jDBcConnection

Conclusion

CONCLUSION

The project titled as **“Email System" has** been designed with much care, with the intention easier and the more complexity involved is presented in a simple and lucid style.

The advantages of the mailing System are

1. Security
2. Cost effective (may be Free of cost)
3. We provide easy installation and setup process
4. provide complete privacy from email service provider when used
5. Ceoss platform support since entire code in java..

.

This project deals with the mailing system and provide security to our users once we send the message

Receiver will get the message in encrypted form .

Message won’t get tempered between sender and receiver , this program will provide you privacy as well as high level of security.

BIBLIOGRAPHY

BIBLIOGRAPHY

- Head First Servlet and Java
- Murach’s Java Servlet and JSP
- Step by step guide to design websites
- Rapid SQL
- SQL Programming
- Online help Learn Java from SimpliLearn
- Java Mail API---*http://www.oracle.com/technetwork/java/javamail/index.html*
- Stack Over Flow--*http://stackoverflow.com*
- Crypto and Java help sub Edits--*http://reddit.com/r/crypto* & *http://reddit.com/r/javahelp*