



SUSPICIOUS MAIL DETECTION SYSTEM

A Report for the Evaluation 3 of Project 2

Submitted by

SHASHANK PAL

(16SCSE101619)

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Mr. RAVINDER AHUJA SIR, M.Tech, Professor

April/May-2020



**SCHOOL OF COMPUTING AND SCIENCE AND
ENGINEERING**

BONAFIDE CERTIFICATE

Certified that this project report “**SUSPICIOUS MAIL DETECTION SYSTEM**” is the bonafide work of “**SHASHANK PAL (16SCSE101619)**” who carried out the project work under my supervision.

SIGNATURE OF HEAD

**School of Computing Science &
Engineering**

SIGNATURE OF SUPERVISOR

Mr. Ravinder Ahuja Sir

Professor

**School of Computing Science &
Engineering**

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|--------------------|---|-----------------|
| 1. | INTRODUCTION | 4 |
| 2. | SYSTEM ANALYSIS | 6 |
| 3. | SOFTWARE REQUIREMENTS SPECIFICATION | 8 |
| 4. | SYSTEM DESIGN | 12 |
| 5. | SYSTEM ENVIRONMENT | 24 |
| 6. | CODE TEMPLATES / IMPLEMENTATION | 30 |
| 7. | TESTING | 48 |
| 8. | OUTPUT | 54 |
| 9. | BIBLIOGRAPHY | 59 |

CHAPTER 1

INTRODUCTION

1.1 ABOUT

Suspicious email detection is a kind of mailing system where suspicious users are identified by determining the keywords used by him/her. The keywords such as bomb, RDX., are found in the mails which are sent by the user. All these blocked mails are checked by the administrator and identify the users who sent such mails.

It consists 5 modules:

The users of this system are compose mails to the other users who are authenticated already. If the composed mails consist of the keywords such as bomb, RDX, Terrorist etc. These suspected mails are blocked or discarded by the administrator so that they cannot be forwarded. This system is designed such a way that the users can easily interact with the system with minimum knowledge to browse the internet.

The Second chapter explains the exact Definition of the Problem and evolves out with the Feasibility Study of the product/part.

The Third chapter is System Analysis which deals about the Hardware and Software Specifications, and Software Requirement Specification, under this SRS Formal Description and Module Description.

The Fourth chapter describes the system deign, under this two levels of design , they are

- High level design (Data design, functional & interface design).
- Low level design (Pseudo code & detail description of functions).

The Fifth chapter fully deals about Testing and Implementation of the whole project.

The Sixth chapter deals the Conclusion and Foreseeable Enhancements of the system.

The Seventh chapter deals about the Bibliography of this Project.

The Eight chapter is the final one which deals about the language used, tools used, Screen layouts and Reports

1.2 PROBLEM DEFINITION

To create or develop a new system first we have to study the prior system, Analysis difficult problems faced by the operator of that system. System Analysis therefore understands such problems and proposes a new system in which the above problems are rectified.

In this project, suspicious users are identified by determining the keywords used by him/her. The keywords such as bomb, RDX ,are found in the mails which are sent by the user . All these blocked mails are checked by the administrator and identify the users who sent such mails.

1.3 TASK

The task of the project is to identify the suspicious mails for users and block them if the account holder wishes to. The project mainly constitute of the following tasks:

- Identify mails
- Block e mails
- Trace IP address of the sender

CHAPTER 2

SYSTEM ANALYSIS

2.1 PURPOSE

The purpose of this project is to suspect the E-mails which consist of offensive, anti-social elements and block them which helps in identifying the suspicious user.

In this project, suspicious users are identified by determining the keywords used by him/her. The keywords such as bomb, RDX. , are found in the mails which are sent by the user. All these blocked mails are checked by the administrator and identify the users who sent such mails.

2.2 FEASIBILITY STUDY

It is necessary and prudent to evaluate the feasibility of a project at the earliest possible time. There may be different ways of checking whether a system is feasible or not. The following feasibility studies were performed to gauge the feasibility of the system.

2.2.1 Operational Feasibility

In this test, the operational scope of the system is checked. The system under consideration should have enough operational reach. It is observed that the proposed system is very user friendly and since the system is built with enough help, even persons with little knowledge of windows can find the system very easy.

2.2.2 Technical Feasibility

This test includes a study of function, performance and constraints that may affect the ability to achieve an acceptable system. This test begins with an assessment of the technical viability of the proposed system. One of the main factors to be assessed is the need of various kinds of resources for the successful implementation for the proposed system.

2.2.3 Economical Feasibility

An evaluation of development cost weighed against the ultimate income or benefit derived from the development of the proposed system is made. Care must be taken that incurred in the development of the proposed of the system should not exceed from the system. The income can be in terms of money or goodwill, since the software brings in both, the system is highly viable.

2.3 EXISTING SYSTEM

In the existing system, the mails are sent to the authenticated users who are intended to be received.

Some defects in existing system are:

- Suspicious mails cannot be detected.
- Offensive users cannot be identified.

2.4 PROPOSED SYSTEM

In the proposed system the suspicious users are detected and the offensive mails are blocked.

Features of proposed system:

- This helps in finding out anti social elements.
- This provides the security to system which adapts it.
- This also helps the intelligence bureau, crime branch etc.

2.5 Overview

The software should be developed according to the system. The user interface module should be developed in such a way that the user can easily operate the system. It is the responsibility of the developer to give training to the user. His most important responsibility is maintenance. He is responsible to give support to the customer when they are getting problem related to the software.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Hardware and Software Specifications

The development of this project deals with the following environment

- Hardware requirements
- Software requirements

Hardware Requirements

The selection of hardware is very important in the existence and proper working of any software. In the selection of hardware, the size and the capacity requirements are also important.

The suspicious email detection can be efficiently run on Pentium system with at least 128 MB RAM and Hard disk drive having 20 GB. Floppy disk drive of 1.44 MB and 14 inch Samsung color monitor suits the information system operation.(A Printer is required for hard copy output).

- Pentium processor ----- 1.2 GHZor above
- RAM Capacity ----- 2 GB
- Hard Disk ----- 180 GB

Software Requirements

One of the most difficult tasks is that, the selection of the software, once system requirement is known is determining whether a particular software package fits the requirements. After initial selection further security is needed to determine the desirability of particular software compared with other candidates. This section first summarizes the application requirement question and then suggests more detailed comparisons.

- Operating System :: Windows NT/2000
- Server Side :: JSP with Tomcat Server
- Client Side :: HTML ,JavaScript
- Services :: JDBC
- Database :: My SQL
- Integrated Development Environment :: My Eclips

3.2 Fields Specification

Administrator: login name, password, login type.

User: user name, password.

3.2.1 Formal Description

Module Description

The main modules of this “suspicious email detection” are broadly divided into five. They are

- **Login Module**
- **Registration Module**
- **Administration Module**
- **User Module**
- **Mailing Module**

Login Module

This module is used by administrator and users (who are authenticated) to login into the secure mail. The login details of the specified person will be entered and hence can enter into the secure mail.

Registration Module

This module is used by the unauthenticated users who are unregistered. The users must register themselves such that they can login into the secure mail.

Administration Module

This module is used by the administrator to perform the functions like managing the keywords, entering new keywords and to check out the block list of the discarded mails.

User Module

This module is used by the users to do operations like composing mail, checking out the mails in inbox and finally sending the mails to the authenticated users by attaching a message.

Mailing Module

This module is used by the users perform mailing system. The mailing system consists of composing the mails, sending the mails and checking out the mails in inbox.

3.3 User Interface Requirement

To achieve the objectives and benefits expected from the computer based system, it is essential for people who will be involved to be confident of their role in the new system. This involves them in understanding the overall system. As the system becomes more complex the need for education and training is more and more important. Education of the user should really have taken place much earlier in the project when they were being involved in the investigation and design work. Once the

staff has been trained the system can be tested System testing is an expensive but critical process that can make as much as fifty percentage of the budget of the program development. The common view of testing held by the user is that it is performed to prove that there are no errors in the program.

Therefore, the most practical approach is with understanding that testing is the process of executing programs with the intention of finding errors.

3.4 Performance Requirement

Considering the interactive nature of the task the system must have the following characters.

- Efficient CPU utilization
- Less Memory space
- High reliability
- High flexibility
- User friendly

3.5 General Constraints

As the clients is not used to an automated environment they didn't impose any stringent constraints over the system.

But they put forth some important comment that is desirable for the proposed system.

- Secure e mail system
- To be able to tack the suspicious sender of e mails.

3.6 Other Non Functional Requirements

Nonfunctional requirements define system properties and constraints it arises through user needs, because of budget constraints or organizational policies, or due to the external factors such as safety regulations, privacy registration and so on. These are:

- Reliability
- Maintainability
- Portability
- Reusability
- Security

CHAPTER 4

SYSTEM DESIGN

Design Description

Design is essentially a blue print or it acts as a bridge between the requirement specification and the final solution for satisfying the requirements.

Based on the work-flow described above we can draw the following conclusions for the Software System that has to be developed:

- The System needs to be a web-based system so that it allows the admin & clients to access the secure mail over the Internet.
- Being a web-based system, it enables the users to send e-mails to other users who are already registered. An added advantage is since the e-mail is delivered instantly, there could be instant responses from the admin if any suspicious emails are detected.

- The whole process depends on communications between admin & the users. If all these communications are done through a web-based system, then the time period for the whole process can be considerably brought down.
- The System needs to store the details of all the users.
- The System needs to store the details of all the information (sent mails, composed mails etc) held by all the users.
- The System needs to store the details of all the requirements held by the different users.
- Since it is a web-based system, a Login authorization should be provided so that Admin and users will be able to lookup & use options that are specific to them.
- The System should allow the users to enter their details.
- The System should provide an option to generate an user Report.
- The System should provide an option to generate block mails Report.
- The System should provide an option to generate selected users Report.

4.1 High level design

Data design

Table Name: Users

Description: This table is used to maintain the registered users information.

| SL.NO | FIELD NAME | DATA TYPE | DESCRIPTION |
|--------------|-------------------|------------------|--|
| 1 | USERNAME | Varchar2(10) | This is unique identifier given to user to identify him uniquely. This the Primary Key of the table. |
| 2 | PASSWORD | Varchar2(20) | This the password of the user |

Table Name: block list

Description: This table is used for client's mailing information.

| SL.NO | FIELD NAME | DATA TYPE | DESCRIPTION |
|--------------|-------------------|------------------|--|
| 1 | MAIL TO | Varchar2(10) | To whom the user wants to send the mails |
| 2 | MAIL FROM | Varchar2(20) | From whom the users got the mails. |
| 3 | SUBJECT | Varchar2(40) | The subject present in the mail. |
| 4 | MESSAGE | Varchar2(40) | The message or the data present in the mail. |

Table Name: keywords

Description: This table consists of the keywords of the mailing system.

| SL.NO | FIELD NAME | DESCRIPTION |
|--------------|-------------------|--------------------------------------|
| 1 | keyword | The suspected keywords of the mails. |

4.2 Input Design

Input design is the process of converting a user-oriented description of the inputs to a computer-based business system into a programmer-oriented specification. The design decision for handling input specify how data are accepted for computer processing. Input design is a part of overall design that needs careful attention.

The collection of input data is considered to be the most expensive part of the system design. Since the inputs have to be planned in such a way so as to get the relevant information, extreme care is taken

to obtain the pertinent information. If the data going into the system is incorrect then the processing and outputs will magnify these errors.

The goal of designing input data is to make data entry as easy, logical and free from error as possible.

The following are the objectives of input design:

- To produce a cost effective method of input.
- To ensure validation.

Effort has been made to ensure that input data remains accurate from the stage at which it is recorded and documented to the stage at which it is accepted by the computer. Validation

procedures are also present to detect errors in data input, which is beyond control procedures.

Validation procedures are designed to check each record, data item or field against certain criteria.

4.3 Output Design

The output design phase of the system design is concerned with the conveyance of information to the end users in user-friendly manner. The output design should be efficient, intelligible so that the system relationship with the end user is improved and thereby enhancing the process of decision making.

The output design is an ongoing activity almost from the beginning of the project, efficient and well-defined output design improves the relation of the system and the user. The primary considerations in the design of the output are the requirement of the information and the objective of the end user.

4.4 Architectural Design

4.4.1 DFD

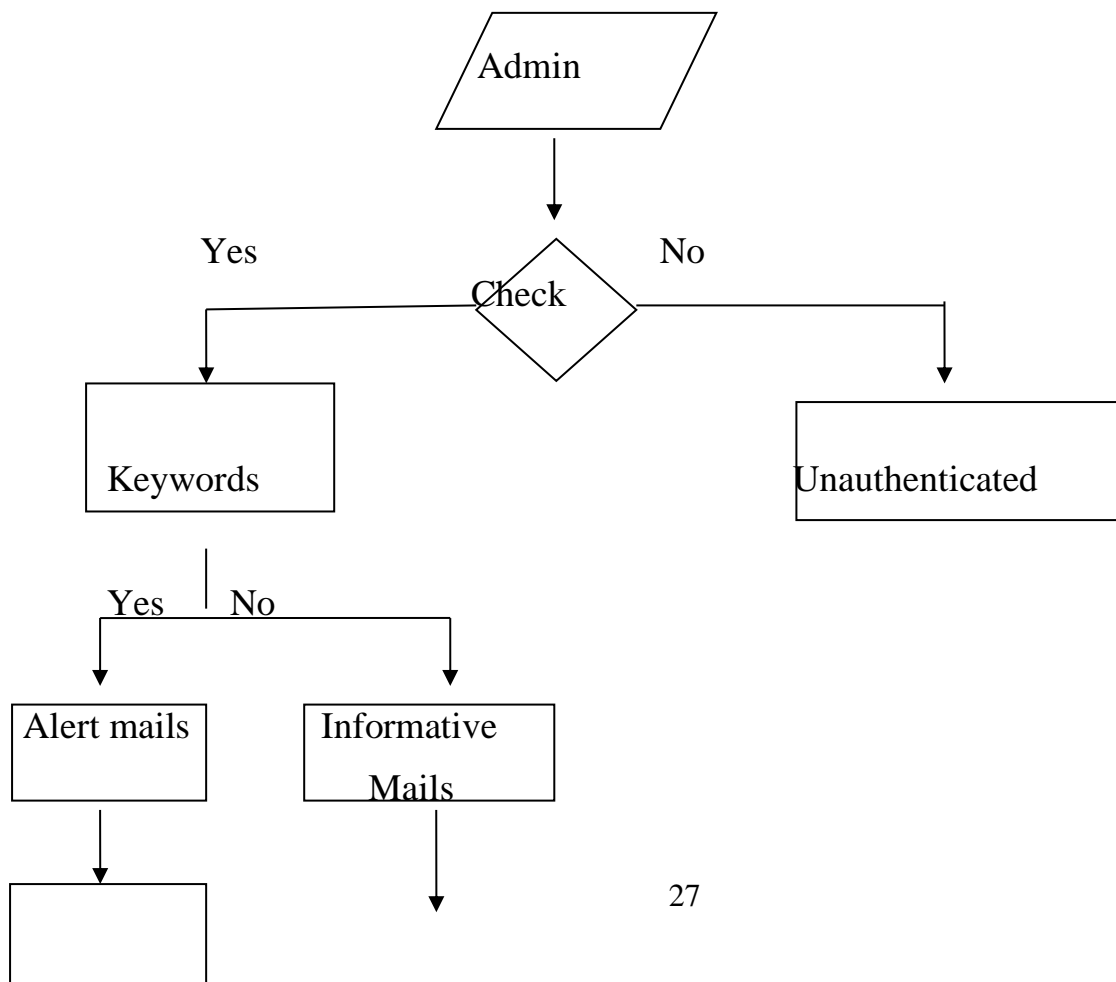
Data flow diagram (DFD) was first developed by Larry Constantine as a way of representing system requirements in a graphical form; this led to modular design. A DFD describes what data flow (logical) rather than how they are processed, so it does not depend on hardware, software, and data structure or file organization. It is also known as 'bubble chart'. A data Flow Diagram is a structured

analysis and design tool that can be used for flowcharting in place of, or in association with, information-oriented and process-oriented system flowcharts. A DFD is a network that describes the flow of data and the processes that change, or transform, data throughout a system. This network is constructed by using a set of symbols that do not imply a physical implementation. It has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design phase that functionally decomposes the requirement specifications down to the lowest level of detail.

Four steps are commonly used to construct a DFD

- Process should be named and numbered for easy reference. Each name should be representative of the process.
- The direction of flow is from top to bottom and from left to right.
- When a process is exploded into lower level details they are numbered.
- The names of data stores, sources and destinations are written in capital letters.

Data Flow Diagram



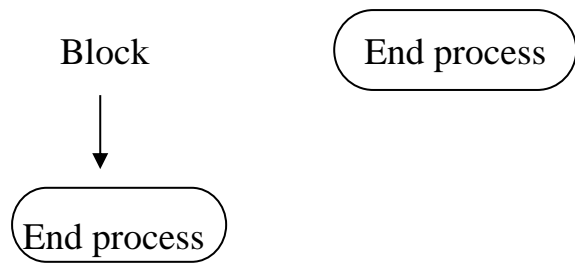


Fig.4.1 DFD for Admin

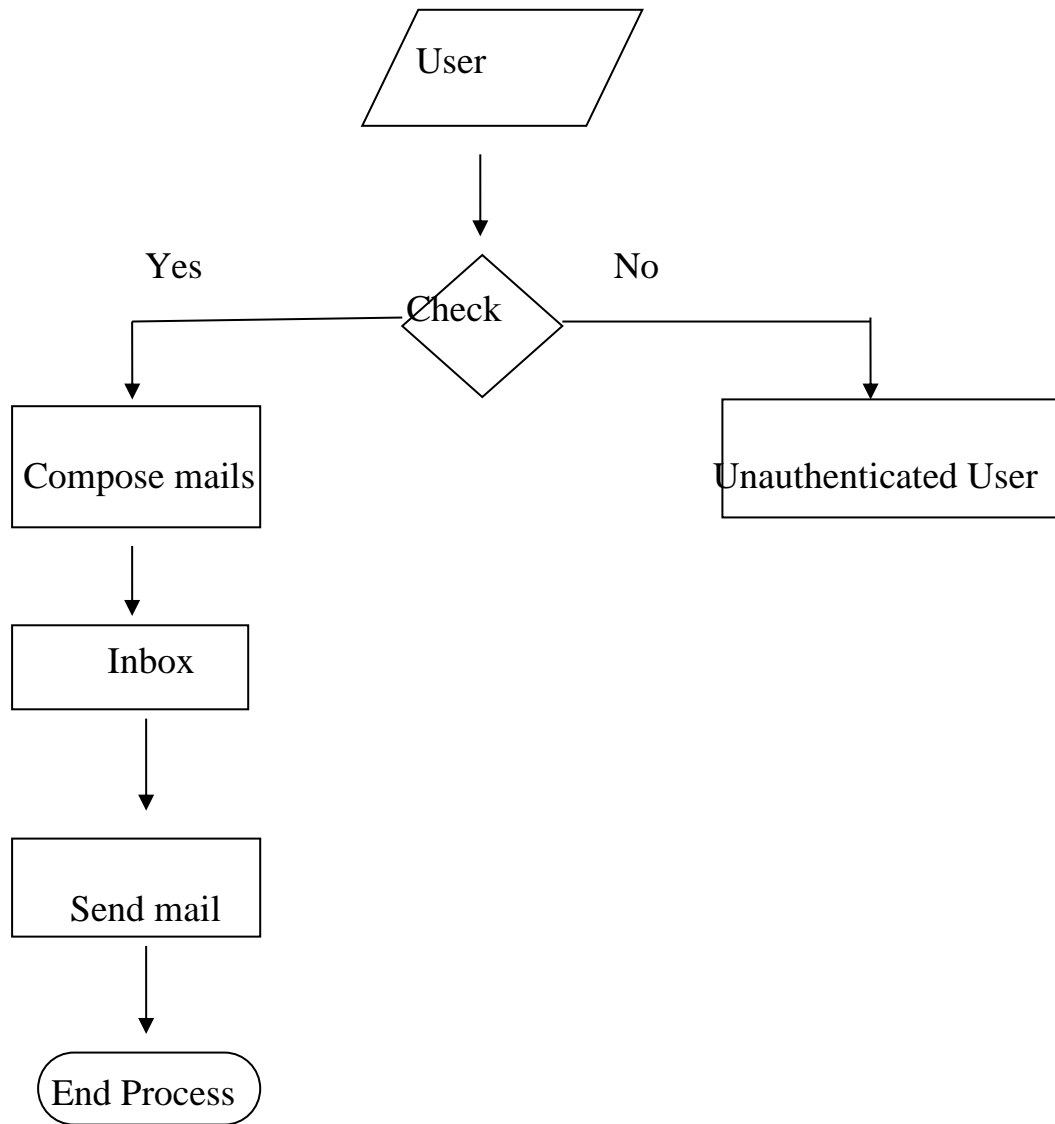


Fig.4.2 DFD For User

4.4.2 ER Diagram:

An entity–relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in a way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them.

Entity Relationship Diagram



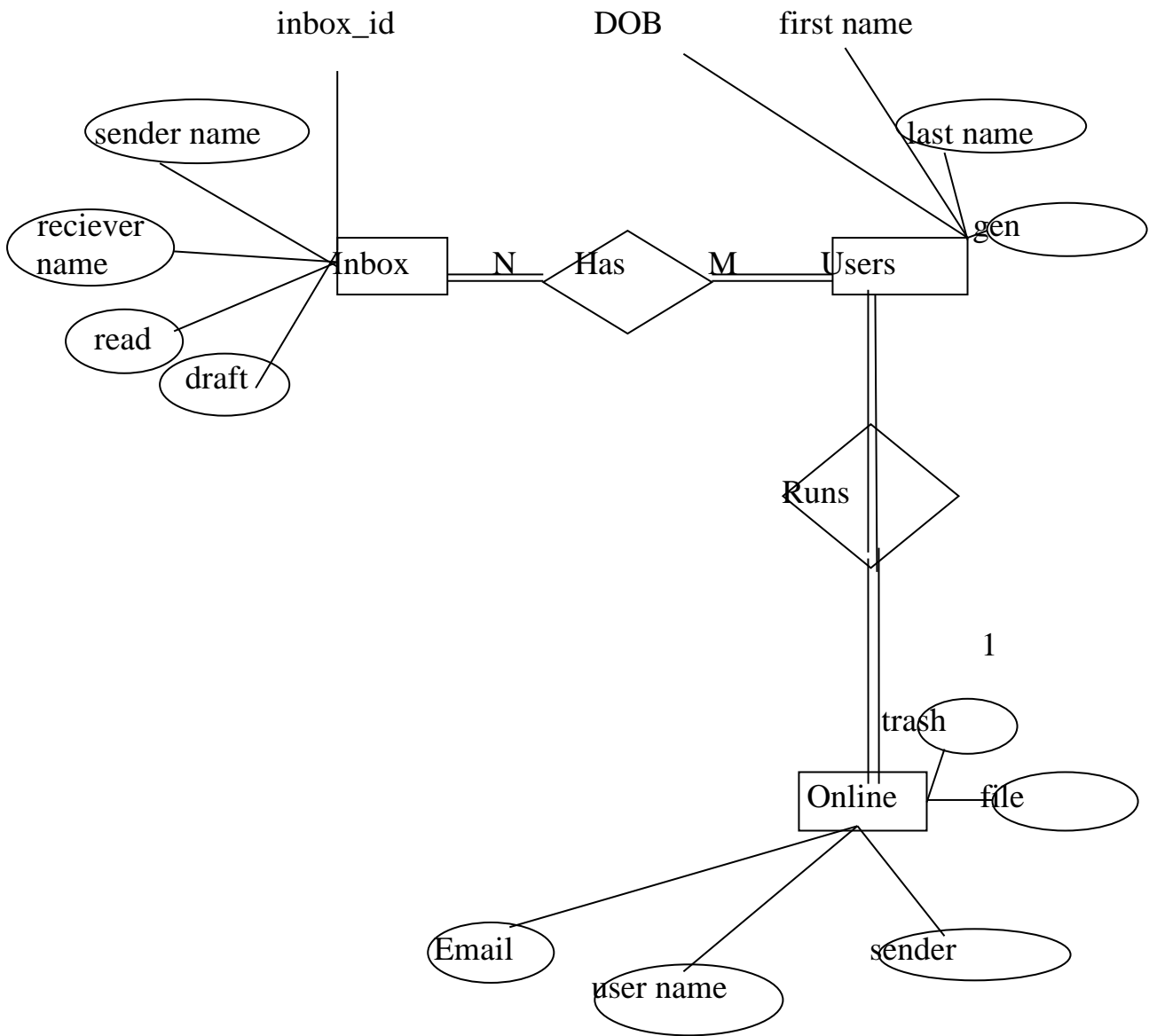


fig.4.3 E R Diagram

4.4.3 CLASS DIAGRAM

The class diagram is the main building block of **object oriented** modelling. It is used both for general **conceptual modelling** of the systematics of the application, and for detailed modelling translating the models into **programming code**.

A class with three sections.

In the diagram, classes are represented with boxes which contain three parts:

- The top part contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lowercase.

ADMIN

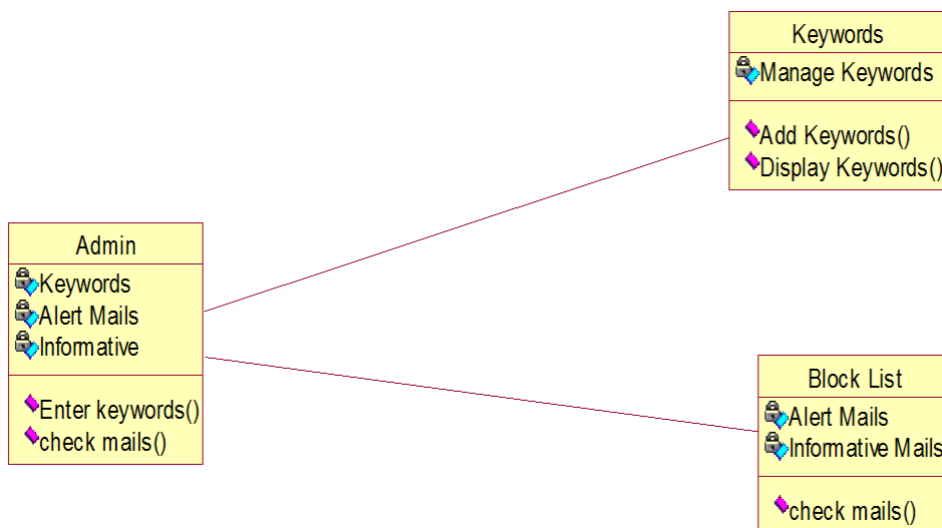


Fig4.4. Class Diagram for Admin

USER

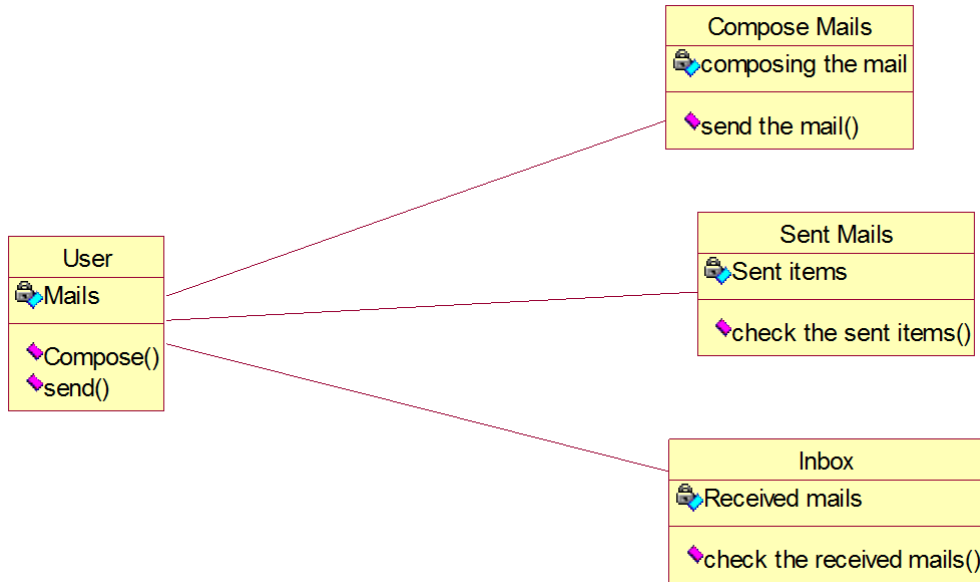


Fig.4.5 Class Diagram for User

4.4.4 USE CASE

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a [use case](#). A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual [use case](#) and will often be accompanied by other types of diagrams as well.

ADMIN

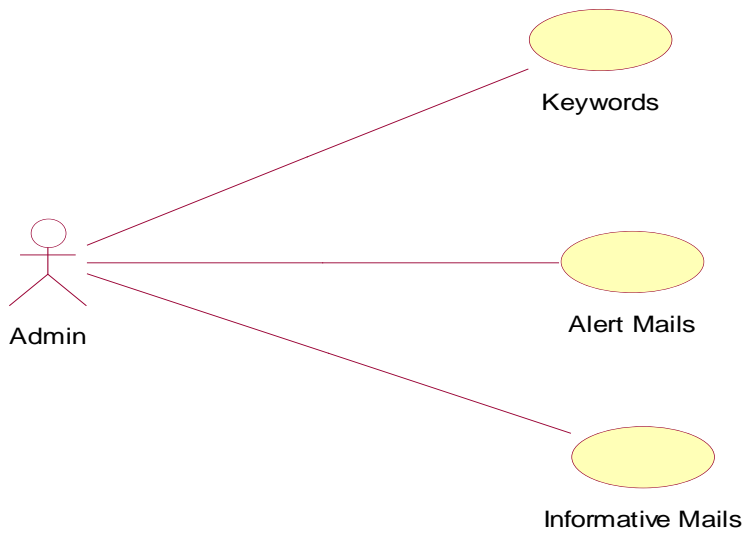


Fig. 4.6 Use case Diagram for Admin

USER

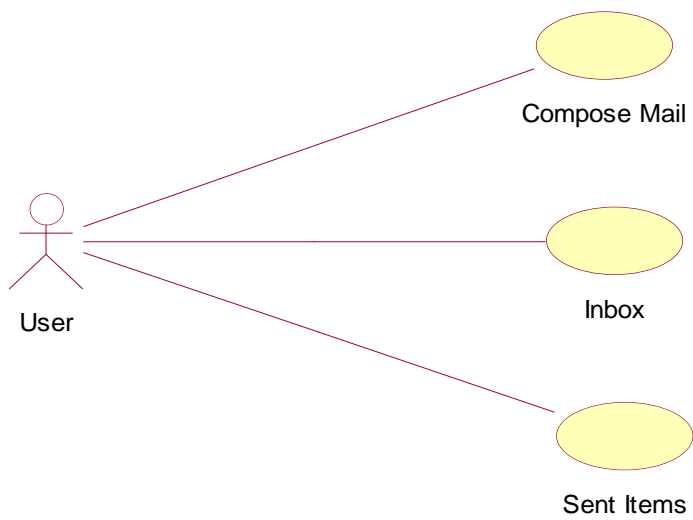


Fig.4.7 Use case Diagram for User

4.4.5 SEQUENCE DIAGRAM

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.

ADMIN

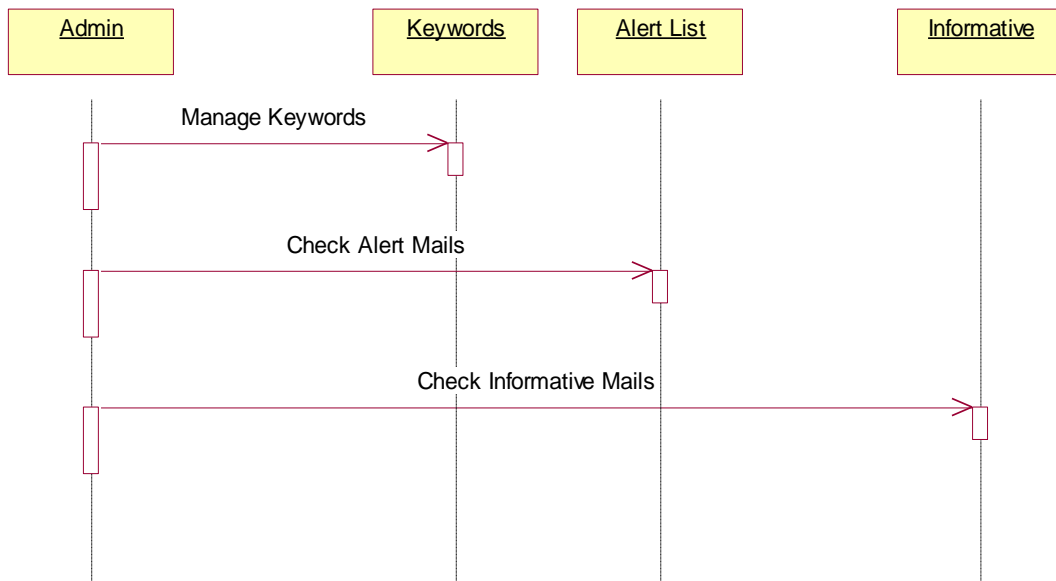


Fig. 4.8 Sequence Diagram for Admin

USER

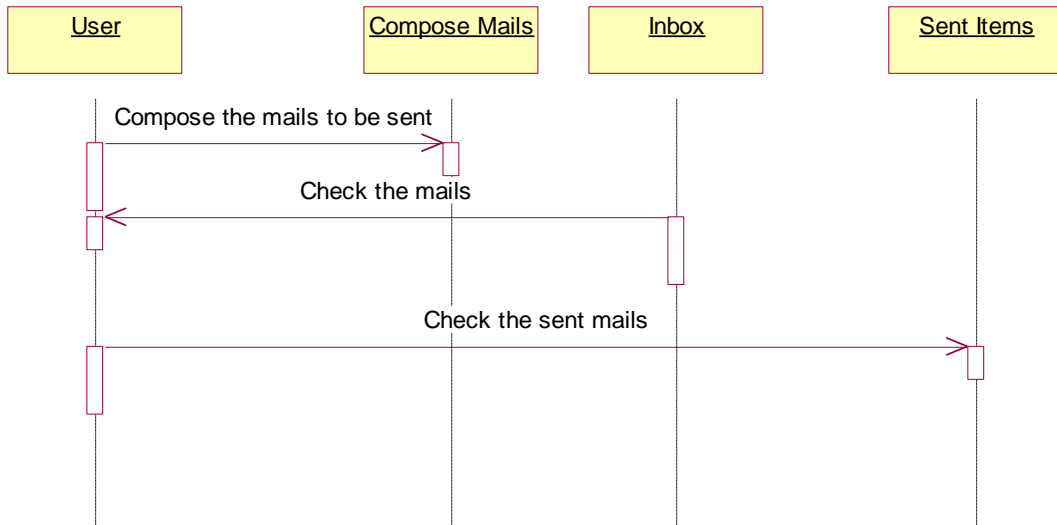


Fig. 4.9 Sequence Diagram for User

CHAPTER 5

SYSTEM ENVIRONMENT

The following hardware and software has been used for the development and deployment of the system.

5.1 Hardware

- Processor-Intel(R)Core(TM) 2duo T6600
- Installed Memory(RAM)-2GB
- Mouse : Standard two button or higher
- Keyboard : Standard 101-102 key keyboard
- Display : 15" Monitor
- Other devices : Modem

5.2 Software

- Operating System :Windows 7
- Language Used : JAVA
- Tools : MY ECLIPSE 7.1

5.2.1 About the Operating System- Windows 7

The operating system used was Microsoft Windows 7. The Windows 7 provides a suitable environment for the smooth functioning of the project. Windows 7 makes personal computing easy. Power, performance, a bright new look and plenty of help when you need it. Windows 7 has it all, along with unmatched dependability and security. Windows 7 professional marks a new standard in business software combining enterprise-class performance and reliability with Unprecedented ease of use. Built on the rock-solid foundation of Microsoft's proven Windows 2000 technology, Windows

7 Professional contains all the features of Microsoft Windows XP Home Edition, and includes new and enhanced features designed especially for business and advanced use.

The all-new Help and Support Center in Windows 7 is our one-stop shop for:

- Clear how-to instructions
- Engaging start-to-finish articles
- Troubleshooting advice.

Special wizards give you step-by-step instructions to smooth the way when connecting new devices and running new software.

5.2.2 About the language- Java

Java is the first programming language designed from ground up with network programming in mind. The core API for Java includes classes and interfaces that provide uniform access to a diverse set of network protocols.

As the Internet and network programming has evolved, java has maintained its cadence. New APIs and toolkit have expanded the available options for the Java network programmer. Java is both a programming language and an environment for executing programs written in Java language. Unlike traditional compilers, which convert source code into machine level instructions, the Java compiler translates Java source code into instructions that are interpreted by the runtime Java Virtual Machine. So unlike language like C and C++, Java is an interpreted language.

This is a language that follows object-oriented concept used to create executable contents such as applications and applets. But Java is not pure object oriented language, it does not support multiple inheritance & Operator overloading.

The acquisition of Sun Microsystems by Oracle Corporation was completed by Oracle on January 27, 2010. Significantly, Oracle, previously only a software vendor, now owned both hardware and software product lines from Sun (e.g. SPARC Enterprise and Java, respectively).

A major issue of the purchase was the fact that Sun was a major competitor to Oracle, raising many concerns among antitrust regulators, open source advocates, customers, and employees. The EU Commission delayed the acquisition for several months over concerns of Oracle's plans for MySQL, Sun's competitor to the Oracle Database. The commission finally approved the takeover, apparently pressured by the U.S. to do so, according to a Wiki leaks cable released in September 2011.

5.2.2.1 Java Runtime Environment

The runtime environment used to execute the code. It is made up of the Java language and Java virtual machine. It is portable and it is platform neutral.

A Java virtual machine (JVM) is a process virtual machine that can execute Java byte code. It is the code execution component of the Java platform. Sun Microsystems has stated that there are over 5.5 billion JVM-enabled devices.

A Java virtual machine (JVM) interprets compiled Java binary code (called byte code) for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform.

JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed.

5.2.2.2 Java tools

It is used by the developers to create Java code. They include Java compiler, Java interpreter, classes, libraries and applet viewer.

5.2.2.3 Java Application

Applications are programs written in Java to carry out certain tasks on standalone local computer. Execution of a stand-alone program involves two steps.

- Compiling the source code in to byte code using javac.
- Executing byte code program using Java interpreter.

5.2.2.4 Java Applets

Java applets are pieces of Java code that are embedded in HTML document using the applet tag. When the browser encounters such code it automatically download it and execute it.

A Java applet is a small application written in Java and delivered to users in the form of byte code. The user launches the Java applet from a web page and it is then executed within a Java Virtual Machine (JVM) in a process separate from the web browser itself. A Java applet can appear in a frame of the web page, a new application window, Sun's Applet Viewer or a stand-alone tool for testing applets. Java applets were introduced in the first version of the Java language in 1995.

Java applets run at very fast speeds comparable to, but generally slower than, other compiled languages such as C++. Until approximately 2011, Java applets had been many times faster than JavaScript. Unlike JavaScript, Java applets have access to 3D hardware acceleration, making them

well suited for non-trivial, computation intensive visualizations. As browsers have gained support for hardware accelerated graphics thanks to the canvas technology (or specifically WebGL in the case of 3D graphics), as well as just in time compiled JavaScript, the speed difference has become less noticeable.

Since Java's byte code is cross-platform (or platform independent), Java applets can be executed by browsers (or other clients) for many platforms, including Microsoft Windows, FreeBSD, Unix, OS X and Linux. It is also trivial to run a Java applet as an application with very little extra code so that it can be run directly from the integrated development environment (IDE).

5.2.2.5 Java Virtual Machine

It is a specification to which Java codes must be written. All Java code is to be compiled in this nonexistent virtual machine. Writing the code that compiles in JVM ensures platform independence. A Java virtual machine (JVM) is a process virtual machine that can execute Java byte code. It is the code execution component of the Java platform. Sun Microsystems has stated that there are over 5.5 billion JVM-enabled devices.

A Java virtual machine (JVM) interprets compiled Java binary code (called byte code) for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform.

JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed.

5.2.2.6 Advantages of Java

1) Java is Robust

Robust programs are those reliable programs that are unlikely to fail even under the most unlikely conditions. Many languages like C do not have this feature because they are relaxed in terms of type checking in terms of programming errors. Java is strict about type declaration and does not allow automatic typecasting. Also it uses a pointer model that does not overwrite memory or corrupt data.

2) Java is Secure

Java allows creation of virus-free, tamper free systems to be created. It ensures security in the following ways.

- Pointers and memory allocations are removed during compile time.
- The interpreter verifies all byte codes before executing.
- All Java applets are treated as entrusted code executing in trusted environment.

Because Java was written to support distributed applications over the computer networks, it can be used with a variety of CPU and operating system architectures. To achieve this goal a compiler was created that produces architecture-neutral object files from Java code.

3) Java is Portable

Java byte code will be executed on any computer that has Java Runtime Environment. The portability is achieved in the following ways.

- Java primitive data types and the behavior of arithmetic operations on these data types are explicitly specified.
- The Java libraries include portable interfaces for each platform on which the run time environment is available.

4) Java is small

Because Java was designed to run on small computers, Java system is relatively small for a programming language. It can run efficiently on PCs with 4MB RAM or more. The Java interpreter takes up only a few hundred kilo bytes.

5) Java is garbage collected

Java programs don't have to worry about memory management. The Java system has a built in program called the garbage collector, which scans the memory and automatically frees the memory chunks that are not in use.

6) Java is dynamic

Fundamentally distributed computer environments must be dynamic. Java is capable of dynamic linking new libraries, methods and instance variables as it goes without breaking and without concern.

5.2.2.7 Java Swing

The swing classes eliminate Java's biggest weakness: Its relatively primitive user interface toolkit. Swing provides many new components and containers that allow us to build sophisticated user interfaces, far beyond what was possible with AWT. It also adds several completely new features to

Java's user interface capabilities: drag-and-drop, undo, and the ability to develop our own "Look and Feel", or the ability to choose between several standard looks. The swing components are all "lightweight", and therefore provide more uniform behavior across platforms, making it easier to test our software.

Swing is the primary Java GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

5.2.2.8 Reason for Using Java

It is required to explore systems running different operating system. In order to do so, there should be some way to connect to bridge those operating systems so that all the differences between them are solved and the functionalities are achieved. Also the functions performed in one system should be able to transfer to another and the result should be able to reflect there properly. Java serves as a bridge between these Operating systems. Also Java is widely considered to be the best in developing network applications.

The communication happens between Java Virtual Machines running on the systems. When the client wants to perform the functionalities in another system and see the result, a method in the remote system is invoked from the client. The corresponding method in the remote system performs the job and sends the results to the client that is reflected in its interface.

CHAPTER-6

CODE TEMPLATES

AUTHENTICATION

```
package com.java.mailserverapp.helper;
```

```
import java.util.Date;
```

```
import java.util.HashSet;
```

```
import java.util.Iterator;
```

```
import java.util.List;
```

```
import java.util.Set;
```

```
import org.hibernate.HibernateException;
```

```
import org.hibernate.Query;
```

```
import org.hibernate.Session;
```

```
import org.hibernate.SessionFactory;

import com.java.mailserverapp.entity.MailDetail;
import com.java.mailserverapp.entity.UserMain;
import com.java.mailserverapp.util.HibernateUtil;

public class JsonUserMain {
    private long userId;
    private String firstName;
    private String lastName;
    private Date dob;
    private String sex;
    private String mobileNo;
    private String userName;
    private String password;
    private Date lastLoginTime;
    private String ipAddress;
    private String lastLoginFrom;
    private String userType;
    private String accountStatus;
    /**
     * @return the userId
     */
    public long getUserId() {
```



```
return userId;

}

/**
 * @param userId the userId to set
 */
public void setUserId(long userId) {
this.userId = userId;
}

/**
 \ * @return the firstName
 */
public String getFirstName() {
return firstName;
}

/**
 * @param firstName the firstName to set
 */
public void setFirstName(String firstName) {
this.firstName = firstName;
}

/**
 * @return the lastName
 */
public String getLastName() {
```

```
return lastName;

}

/**
 * @param lastName the lastName to set
 */

public void setLastName(String lastName) {

this.lastName = lastName;

}

/**
 * @return the dob
 */

public Date getDob() {

return dob;

}

/**
 * @param dob the dob to set
 */

public void setDob(Date dob) {

this.dob = dob;

}

/**
 * @return the sex
 */

public String getSex() {
```

```

return sex;

}

/**
 * @param sex the sex to set
 */

public void setSex(String sex) {

this.sex = sex;

}

/**
 * @return the mobileNo
 */

public String getMobileNo() {

return mobileNo;

}

/**
 * @param mobileNo the mobileNo to set
 */

public void setMobileNo(String mobileNo) {

    this.mobileNo = mobileNo;

}

/**
 * @return the userName
 */

public String getUserName() {

```

```

        return userName;
    }

    /**
     * @param userName the userName to set
     */
    public void setUserName(String userName) {
        this.userName = userName;
    }

    /**
     * @return the password
     */
    public String getPassword() {
        return password;
    }

    /**
     * @param password the password to set
     */
    public void setPassword(String password) {
        this.password = password;
    }

    /**
     * @return the lastLoginTime
     */
    public Date getLastLoginTime() {

```

```

        return lastLoginTime;
    }

    /**
     * @param lastLoginTime the lastLoginTime to set
     */
    public void setLastLoginTime(Date lastLoginTime) {
        this.lastLoginTime = lastLoginTime;
    }

    /**
     * @return the ipAddress
     */
    public String getIpAddress() {
        return ipAddress;
    }

    /**
     * @param ipAddress the ipAddress to set
     */
    public void setIpAddress(String ipAddress) {
        this.ipAddress = ipAddress;
    }

    /**
     * @return the lastLoginFrom
     */
    public String getLastLoginFrom() {

```

```

        return lastLoginFrom;
    }

    /**
     * @param lastLoginFrom the lastLoginFrom to set
     */
    public void setLastLoginFrom(String lastLoginFrom) {
        this.lastLoginFrom = lastLoginFrom;
    }

    /**
     * @return the userType
     */
    public String getUserType() {
        return userType;
    }

    /**
     * @param userType the userType to set
     */
    public void setUserType(String userType) {
        this.userType = userType;
    }

    /**
     * @return the accountStatus
     */
    public String getAccountStatus() {

```

```

        return accountStatus;
    }

    /**
     * @param accountStatus the accountStatus to set
     */
    public void setAccountStatus(String accountStatus) {
        this.accountStatus = accountStatus;
    }

    public static JsonUserMain getInstance(UserMain userMain){
        JsonUserMain jsonUserMain = new JsonUserMain();
        jsonUserMain.setUserId(userMain.getUserId());
        jsonUserMain.setFirstName(userMain.getFirstName());
        jsonUserMain.setLastName(userMain.getLastName());
        jsonUserMain.setDob(userMain.getDob());
        jsonUserMain.setSex(userMain.getSex());
        jsonUserMain.setMobileNo(userMain.getMobileNo());
        jsonUserMain.setUserName(userMain.getUserName());
        jsonUserMain.setPassword(userMain.getUserPassword());
        jsonUserMain.setLastLoginTime(userMain.getLastLoginDateTime());
        jsonUserMain.setLastLoginFrom(userMain.getLastLoginFrom());
        jsonUserMain.setUserType(userMain.getUserType());
        jsonUserMain.setAccountStatus(userMain.getAccountStatus());
    }

```

```
        return jsonUserMain;

    }

}
```

MAILING

```
import com.java.mailserverapp.entity.MailDetail;

public class JsonMailDetail {

    private long mailId;

    private String mailFrom;

    private String mailTo;

    private String msgSubject;

    private String msgBody;

    private Date mailSentTime;

    private boolean isSuspicious;

    private String location;

    private String ipAddress;

    private boolean isDeleted;

    /**
     * @return the mailId
     */

    public long getMailId() {

        return mailId;

    }

}
```



```
/**
 * @param mailId the mailId to set
 */
public void setMailId(long mailId) {
    this.mailId = mailId;
}

/**
 * @return the mailFrom
 */
public String getMailFrom() {
    return mailFrom;
}

/**
 * @param mailFrom the mailFrom to set
 */
public void setMailFrom(String mailFrom) {
    this.mailFrom = mailFrom;
}

/**
 * @return the mailTo
 */
public String getMailTo() {
    return mailTo;
}
```

```

/**
 * @param mailTo the mailTo to set
 */
public void setMailTo(String mailTo) {
    this.mailTo = mailTo;
}

/**
 * @return the msgSubject
 */
public String getMsgSubject() {
    return msgSubject;
}

/**
 * @param msgSubject the msgSubject to set
 */
public void setMsgSubject(String msgSubject) {
    this.msgSubject = msgSubject;
}

/**
 * @return the msgBody
 */
public String getMsgBody() {
    return msgBody;
}

```

```

/**
 * @param msgBody the msgBody to set
 */
public void setMsgBody(String msgBody) {
    this.msgBody = msgBody;
}

/**
 * @return the mailSentTime
 */
public Date getMailSentTime() {
    return mailSentTime;
}

/**
 * @param mailSentTime the mailSentTime to set
 */
public void setMailSentTime(Date mailSentTime) {
    this.mailSentTime = mailSentTime;
}

/**
 * @return the isSuspicious
 */
public boolean isSuspicious() {
    return isSuspicious;
}

```

```
/**
 * @param isSuspicious the isSuspicious to set
 */
public void setSuspicious(boolean isSuspicious) {
    this.isSuspicious = isSuspicious;
}

/**
 * @return the location
 */
public String getLocation() {
    return location;
}

/**
 * @param location the location to set
 */
public void setLocation(String location) {
    this.location = location;
}

/**
 * @return the ipAddress
 */
public String getIpAddress() {
    return ipAddress;
}
```

```

/**
 * @param ipAddress the ipAddress to set
 */
public void setIpAddress(String ipAddress) {
    this.ipAddress = ipAddress;
}

/**
 * @return the isDeleted
 */
public boolean isDeleted() {
    return isDeleted;
}

/**
 * @param isDeleted the isDeleted to set
 */
public void setDeleted(boolean isDeleted) {
    this.isDeleted = isDeleted;
}

public static JsonMailDetail getInstance(MailDetail detail){
    JsonMailDetail mailDetail = new JsonMailDetail();
    mailDetail.setMailId(detail.getMailId());
    mailDetail.setIpAddress(detail.getIpAddress());
    mailDetail.setDeleted(detail.isDeleted());
}

```

```

        mailDetail.setSuspicious(detail.isSuspicious());

        mailDetail.setLocation(detail.getLocation());

        mailDetail.setMailFrom(detail.getMailFrom());

        mailDetail.setMailTo(detail.getMailTo());

        mailDetail.setMsgSubject(detail.getMsgSubject());

        mailDetail.setMsgBody(detail.getMsgBody());

        mailDetail.setMailSentTime(detail.getMailSentTime());

        return mailDetail;}

}

```

SUSPICIOUS WORDS AND DETECTION

```

package com.java.mailserverapp.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="SUSPICIOUS_WORD")
public class SuspiciousWord{

```

```
@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

@Column(name="ID")

private long id;

@Column(name="TEXT_DESCRIPTION")

private String textDescription;

/**
 * @return the id
 */
public long getId() {
    return id;
}

/**
 * @param id the id to set
 */
public void setId(long id) {
    this.id = id;
}
```

```
}

/**
 * @return the textDescription
 */
public String getTextDescription() {
    return textDescription;
}

/**
 * @param textDescription the textDescription to set
 */
public void setTextDescription(String textDescription) {
    this.textDescription = textDescription;
}
}
```


CHAPTER7

TESTING

7.1 Introduction

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or deliver faster.

A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software.

7.2 Definition

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and

behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester[citation needed]. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one which reveals an error; however, more recent thinking suggests that a good test is one which reveals information of interest to someone who matters within the project community.

7.3 Testing Methodologies

- Black Box Testing
- White Box Testing
- Gray Box Testing

7.3.1 Black Box Testing

It is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application. Usually Test Engineers are involved in the black box testing. Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation.

The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

7.3.2 White Box Testing

It is the testing process in which tester can perform testing on an application with having internal structural knowledge. Usually The Developers are involved in white box testing.

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

7.3.3 Gray Box Testing

It is the process in which the combination of black box and white box tonics' are used. Grey-box testing involves having knowledge of internal data structures and algorithms for purposes of designing tests.

While executing those tests at the user, or black-box level. The tester is not required to have full access to the software's source code. Manipulating input data and formatting output do not qualify as grey-box, because the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

7.4 Levels of Testing

- Unit Testing
- Integration Testing
- System Testing

7.4.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software designs the module. To check whether each module in the software works properly so that it gives desired outputs to the given inputs. All validations and conditions are tested in the module level in the unit test.

Control paths are tested to ensure the information properly flows into, and output of the program unit and out of the program unit under test. Boundary conditions are tested to ensure that the modules operate at boundaries. All independent paths through the control structure ensure that all statements in a module have been executed at least once.

7.4.2 Integration Testing

The major concerns of integration testing are developing an incremental strategy that will limit the complexity of entire actions among components as they are added to the system. Developing a

component as they are added to the system, developing an implementation and integration schedules that will make the modules available when needed, and designing test cases that will demonstrate the viability of the evolving system. Though each program works individually they should work after linking them together. This is also referred to as interfacing. Data may be lost across interface and one module can have adverse effect on another. Subroutines after linking may not do the desired function expected by the main routine. Integration testing is a systematic technique for constructing program structure while at the same time conducting tests to uncover errors associated with the interface. In the testing, the programs are constructed and tested in small segments. Here our objective is that to edit, compile and execute Java programs within a single platform. Using integration test plan prepared in the design phase of the system developments guide, the integration test is carried out and all the errors found in the system are corrected for the next testing steps.

7.4.3 System Testing

When a system is developed, it is hoped that it performs properly. In practice however some errors always occur. The main purpose of testing and information system is to find the errors and correct them. A successful test is one which finds an error.

The main objectives of system testing are-

- To ensure during operation the system will perform as per specifications.
- To make sure that the system meets user's requirements during operation.
- To verify that the controls incorporated in the system function as intended.
- To see that when correct inputs are fed to the system the outputs are correct.
- To make sure that during operation incorrect input and output will be deleted.

The scope of a system test should include both manual operations and computerized. Operations system testing is a comprehensive evaluation of the programs, manual procedures, computer operations and controls. System testing is the process of checking if the developed system is working according to the original objectives and requirements. All testing needs to be conducted in accordance to the test conditions specified earlier.

7.5 Types of Testing

- Regression Testing.
- Re-Testing.
- Static Testing.

- Dynamic Testing.
- Alpha Testing.
- Beta Testing.
- Compatibility Testing.
- Installation Testing.

7.5.1 Regression Testing

It is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

7.5.2 Re-Testing

It is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

7.5.3 Static Testing

Static testing is a form of software testing where the software isn't actually used. This is in contrast to dynamic testing. It is generally not detailed testing, but checks mainly for the sanity of the code, algorithm, or document. It is primarily checking of the code and/or manually reviewing the code or document to find errors. This type of testing can be used by the developer who wrote the code, in isolation. Code reviews, inspections and Software walkthroughs are also used.

7.5.4 Dynamic Testing

Dynamic testing (or dynamic analysis) is a term used in software engineering to describe the testing of the dynamic behavior of code. That is, dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time. In dynamic testing the software must actually be compiled and run. It involves working with the software, giving input values and checking if the output is as expected by executing specific test cases which can be done manually or with the use of an automated process. This is in contrast to static testing. Unit tests, integration tests, system tests and acceptance tests utilize dynamic testing.

7.5.5 Alpha Testing

It is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

7.5.6 Beta-Testing

It is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

7.5.7 Compatibility testing

It is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers...etc) In order to check how far the product is compatible with all these environments platform combination.

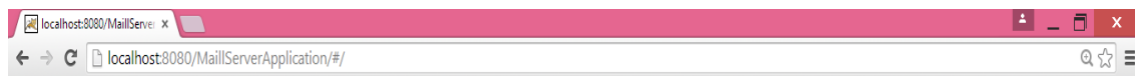
7.5.8 Installation Testing

It is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

CHAPTER8

OUTPUT SCREENS

8.1 USER INTERFACE



Sign In

Login

[Create an account](#)



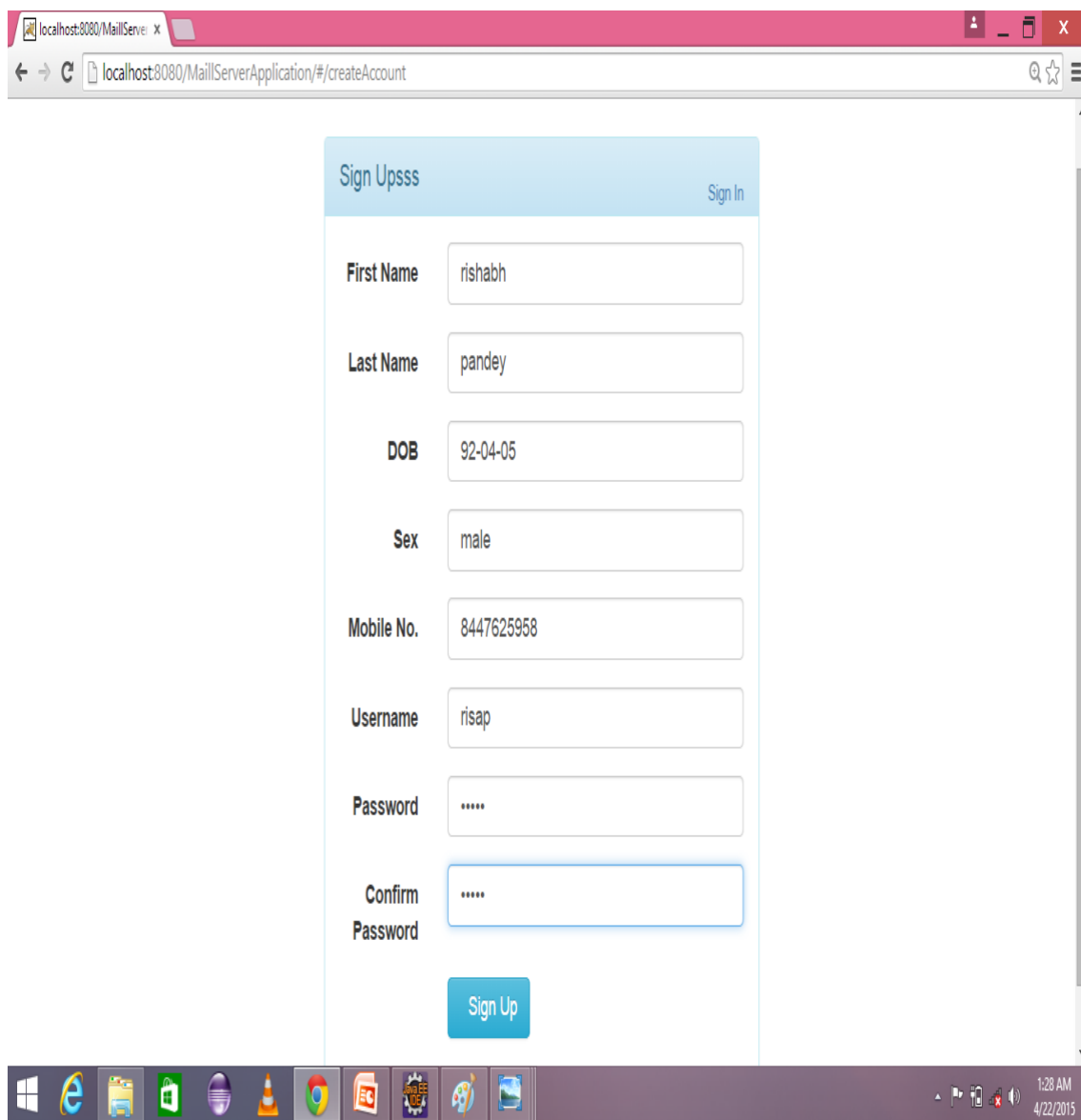
8.2 SIGN UP

The screenshot displays a web browser window with the following details:

- Browser Tab:** localhost:8080/MailServer
- Address Bar:** localhost:8080/MailServerApplication/#/createAccount
- Form Title:** Sign Ups
- Form Fields:**
 - First Name: Input field with placeholder text "First Name"
 - Last Name: Input field with placeholder text "Last Name"
 - DOB: Input field with placeholder text "Date of Birth yy-mm-dd"
 - Sex: Input field with placeholder text "Sex"
 - Mobile No.: Input field with placeholder text "Mobile Number"
 - Username: Input field with placeholder text "User Name"
 - Password: Input field with placeholder text "Password"
 - Confirm Password: Input field with placeholder text "Confirm Password"
- Buttons:** A "Sign In" link in the top right and a "Sign Up" button at the bottom.

The Windows taskbar at the bottom shows the system tray with the date and time: 1:16 AM, 4/22/2015.

8.3 REGISTRATION

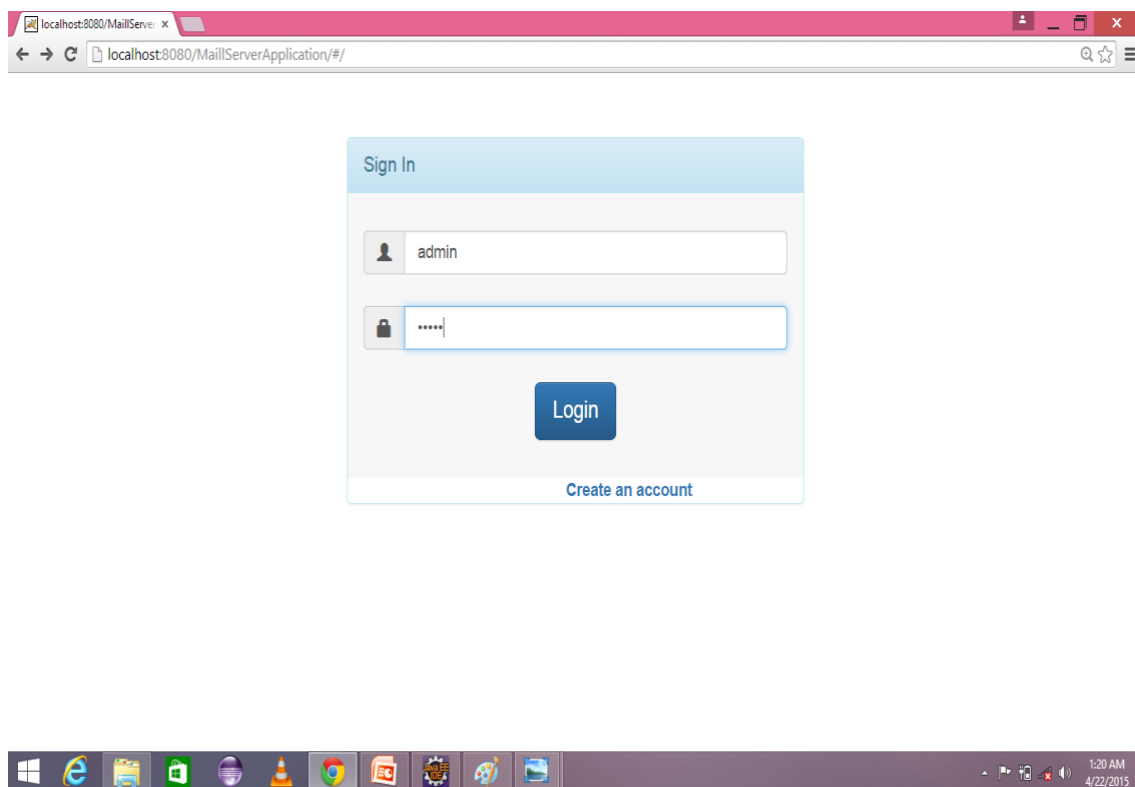


The screenshot shows a web browser window with the address bar displaying `localhost:8080/MailServerApplication/#/createAccount`. The page content is a registration form with a light blue header. The form fields are as follows:

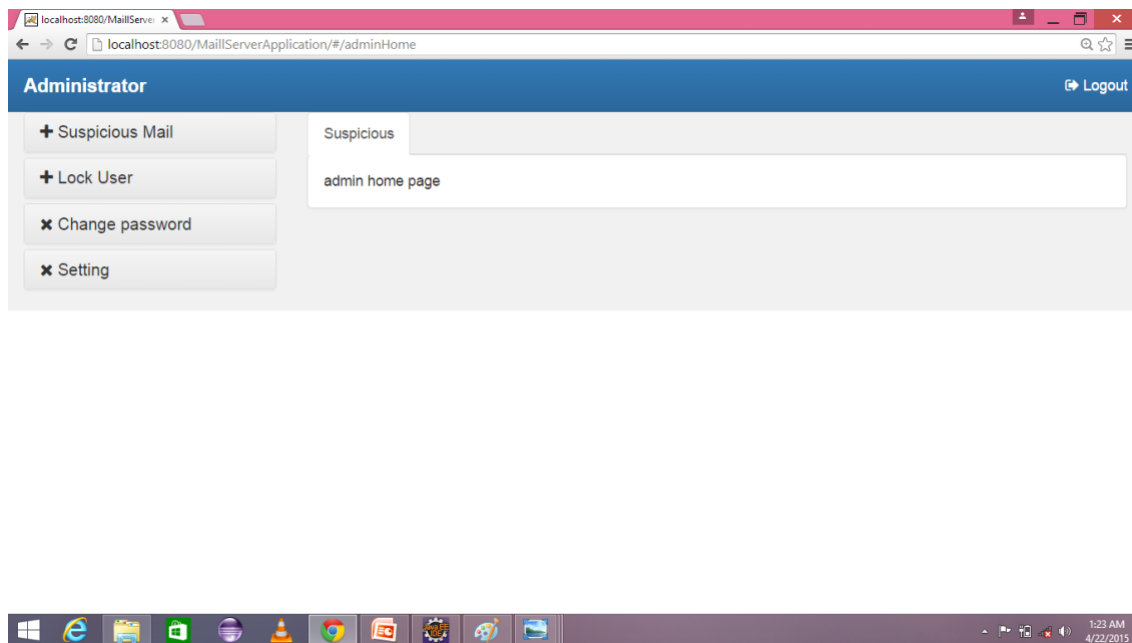
| Sign Up | | Sign In |
|--|---|-------------------------|
| First Name | <input type="text" value="rishabh"/> | |
| Last Name | <input type="text" value="pandey"/> | |
| DOB | <input type="text" value="92-04-05"/> | |
| Sex | <input type="text" value="male"/> | |
| Mobile No. | <input type="text" value="8447625958"/> | |
| Username | <input type="text" value="risap"/> | |
| Password | <input type="password" value="....."/> | |
| Confirm Password | <input type="password" value="....."/> | |
| <input type="button" value="Sign Up"/> | | |

The Windows taskbar at the bottom shows the system tray with the date and time: 1:28 AM, 4/22/2015.

8.4 ADMIN LOGIN



8.5 ADMIN INBOX



CHAPTER 9

BIBLIOGRAPHY

JAVA SERVLETS

- TATA McGraw HILL
- Karl Moss

SOFTWARE ENGINEERING

- A Practitioner's Approach
- McGraw-Hill Publications
- Roger S. Pressman.

[J2EE-Overview] - <http://java.sun.com/j2ee/overview.html>

[JS-NET] - <http://developer.netscape.com/docs/manuals/communicator/jsref/contents.htm>

[J2EE-Home] - <http://java.sun.com/j2ee/>

[J2EE-Components]

http://java.sun.com/j2ee/blueprints/platform_technologies/component/index.html

[SUN-Developer] - <http://developer.java.sun.com/developer/>