**DECCAN COLLEGE OF ENGINEERING & TECHNOLOGY**

**Dar-us-Salam, Hyderabad -500 001.**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**



**SOFTWARE ENGINEERING LAB REPORT**

**SUBJECT CODE: (PC 651 CS)**

**B.E. CSE VI SEMESTER(CBCS)**

**FACULTY:Mrs. Razia Begum**

**Mr. ShaikMuqtadar**

**2019-2020**

A Project Report

on

**SOFTWARE ENGINEERING APPLICATION**

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

of

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

**SULTAN. MOHIUDDIN 160317733049**

**HAMED MOHIUDDIN REHAN 160317733043**

**MOHAMMED SAJJAD UDDIN 160317733048**

**MOHAMMED AQUIB JAVAID 160317733039**

**MUSTAFA MOHIUDDIN AMER 160317733059**

**Under the guidance**

**Of**

**Mrs. RAZIA BEGUM**

**Assistant Professor**

**Department Of CSE, DCET**



**Department of Computer Science Engineering**

**Deccan College of Engineering and Technology**

**(Affiliated to Osmania University)**

**Hyderabad**

**2020**

**DECCAN COLLEGE OF ENGINEERING AND TECHNOLOGY**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SOFTWARE ENGINEERING LAB**

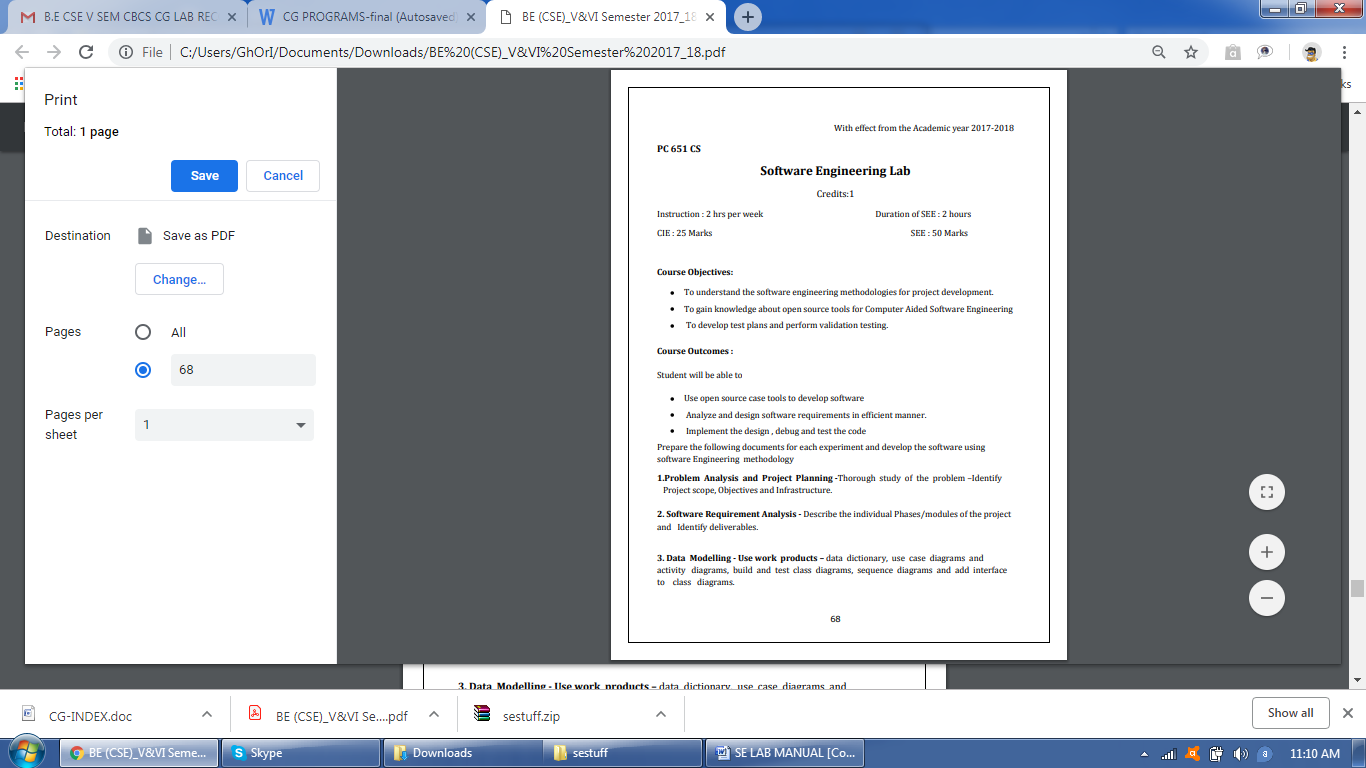


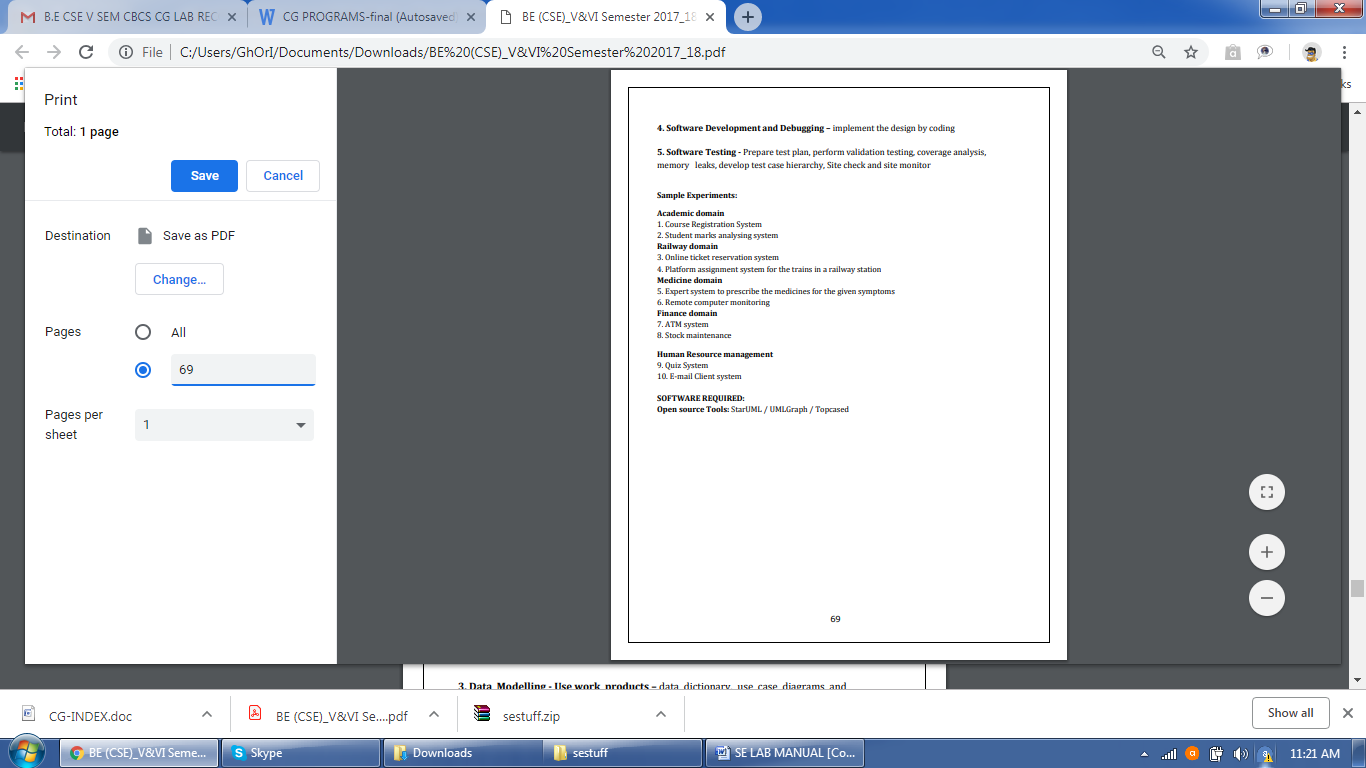
**CERTIFICATE**

**This is to certify that \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, bearing Roll No.43, 48, 49, 39& 59\_\_\_\_\_students of B.E VIsemester(CBCS), C.S.E branch has successfully completed Software Engineering Lab in Deccan College of Engineering and Technology, Dar-us-Salam, Hyderabad during the academic year 2018-2019.**

**Internal Examiner External Examiner**

**SYLLABUS**





**Abstract**

An e-Mail client is an email program; software designed to collect and send electronic mail. It is also referred to as an email program, or mail user agent (MUA) . The term “mail user agent” is less familiar to the average person, but is used in email headers. The headers of the email supply information to the mail servers or computers that handle transferring messages across networks like the Internet.

Email sent from an Internet Service Provider (ISP) is handled by that ISPs mail server, which is the equivalent of an electronic post office. The mail comes first to the mail server, is processed, and forwarded towards the destination. Anther mail server resides at the destination. It receives all incoming mail and electronically sorts it into mailboxes. The recipient picks up email by using his or her email program to connect to the mail server which request items from the mailbox.

` This project develops a similar e-mail client application using Java and some Java APIs. We use standard sockets and networking and with them additional Java Mail API. The protocols used here are SMTP, POP3. The POP3 is used to receive the mails which has been send to our address.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| Contents | | | Page no. |
| CERTIFICATE | | | I |
| ABSTRACT | | | II |
| SYLLABUS | | | III |
| **Chapter 1 Problem Analysis & Project Planning** | | |  |
| 1.1 | Introduction | |  |
| 1.2 | Study of the Problem | |  |
| 1.3 | Scope | |  |
| 1.4 | Objective | |  |
| 1.5 | Infrastructure | |  |
| **Chapter 2 Software Requirements Analysis** | | |  |
| 2.1 | | SRS Specifications |  |
| 2.2 | | Phases/Modules of the Project |  |
| 2.3 | | Identify Deliverables |  |
| **Chapter 3 Data Modeling** | | |  |
| 3.1 | Work Products | |  |
| 3.2 | UML Diagrams | |  |
| 3.3 | Forward Engineering | |  |
| 3.4 | Reveerse Engineering | |  |
| 3.5 | Data Dicitionary | |  |
| 3.6 | Data Flow Diagrams | |  |
| **Chapter 4 Software Development & Debugging** | | |  |
| 4.1 | Sample Code | |  |
| 4.2 | Application Running | |  |
| 4.3 | Screen Shots | |
| **Chapter 5Software Testing** | | |
| 5.1 | Test Description | |
| 5.2 | Validation (Test –Cases) | |
| 5.3 | Integration Testing | |
| **Chapter 6 Conclusion** | | |
| 9.1 | Conclusion | |
| **Chapter 7 Future Enhancement** | | |
| 10.1 | Future Work | |
| **References** | | |

**Chapter 1**

**Problem Analysis & Project Planning**

* 1. **Introduction**

  An e-Mail client is an email program; software designed to collect and send electronic mail. It is also referred to as an email program, or mail user agent (MUA) . The term “mail user agent” is less familiar to the average person, but is used in email headers. The headers of the email supply information to the mail servers or computers that handle transferring messages across networks like the Internet.

  Networking standards have been established to make exchange of messages possible from a virtual unlimited pool of clients. To this end every mail user agent handles text messages, graphics and attached messages in a similar manner. When a user sends an email an array of information is sent with the message to ensure it reaches its destination. This extra “addressing” information is included in the headers, along with the name of the email program as the “mail user agent.”

  Email sent from an Internet Service Provider (ISP) is handled by that ISPs mail server, which is the equivalent of an electronic post office. The mail comes first to the mail server, is processed, and forwarded towards the destination. Anther mail server resides at the destination. It receives all incoming mail and electronically sorts it into mailboxes. The recipient picks up email by using his or her email program to connect to the mail server which request items from the mailbox.

` This project develops a similar e-mail client application using Java and some Java APIs. We use standard sockets and networking and with them additional Java Mail API. The protocols used here are SMTP, POP3. The POP3 is used to receive the mails which has been send to our address.

* 1. **Study of the Problem**
* In former to Mail client there exists a system called Mail through browser
* We need to use mail services like Yahoo, Google, and Hotmail.
* They use HTTP port 80 to access the mails.

**Limitations**

* The browsers are not safe to send confidential messages.
* Some companies create own mail services to send mails.
* They can be easily hacked by hackers.
* The valuable data is modified and it may be lost.
  1. **Scope**

It has a broad future **scope** as new features can be incorporated in the present proposed **system**. The **system** can be used for online sharing of data without the involvement of authority by which users can read message from any part of the world. It has a broad future **scope** as new features can be incorporated in the present proposed **system**. The **system** can be used for online sharing of data without the involvement of authority by which users can read message from any part of the world.

* 1. **Objective**

The objective of Email Client System is to allow the administrator of any organization to send a digital message electronically from one computer to one or more other computers. Emails are flexible and can be used for giving instructions, serving as documentation, providing confirmation, communicating rules and procedures, making recommendations, providing a status update, making **an** inquiry.

* 1. **Infrastructure**

Email infrastructure is a system built to fuel the delivery of all newsletters or transactional emails you send out. It typically consists of the following components: IP addresses, mail agents, feedback loops, and email reputation management tools.

**IP address**

IP addresses are strings of numbers that the domain system name (DNS) attaches to domains. Thanks to these, we can use letters (google.com and such) instead of numbers (256.58.217.06) to visit websites.

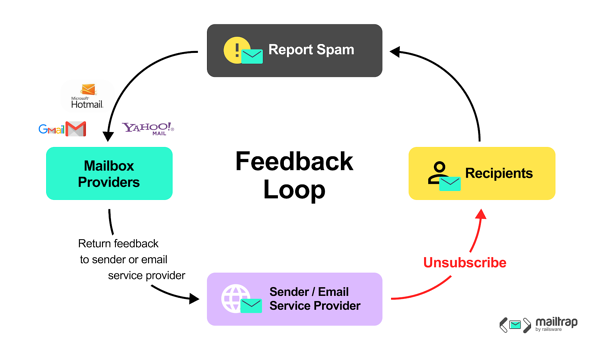
**SMTP server**

An SMTP server is an application that a chosen email client uses to exchange emails between users. Every hosting provider has a different SMTP address - the general look is: **smtp.serveraddress.com**

For example, Gmail’s SMTP is smtp.gmail.com, Outlook’s is smtp.outlook.com, and so on.

### Feedback loop

A feedback loop (FBL) is one of the key processes empowered by email infrastructure. If a campaign sender sets up a feedback loop, he will be able to find out how the reader interacted with an email. This way, marketing teams can remove poorly engaged subscribers from the list to make sure the content is fully relevant to the audience.



A visual breakdown of an email feedback loop

To register for an FBL, a business owner needs to meet a set of conditions:

* Own a domain or an IP or have admin privileges
* Have an active @postmaster address attached to the domain
* Good email reputation score (fair or high).

Monitor feedback loops frequently to assess the relevance of the content you share with subscribers and keep the subscriber list clean.

**Chapter 2**

**Software Requirement Analysis**

Software Requirements Specification is the starting point of Software development activity. It is the most difficulty activity and the most error prone Software Requirements Specification is a means of translating the ideas in the minds of clients into one formal document.

This document is used for the Software Project Management. In the organization the Admin, Project Manager, Project Leader, and Developer the uses the system for different purposes like Employee Details, Employee performance, Client Details, Project status, Module status, report generations etc.

**2.1 SRS Specification**

SRS plays an important role in the designing a project. SRS meets the needs for project designing and developing. The entire information which is needed for the project is specified in this SRS.

SRS contains the feasibility concepts of the projects software, hardware requirements, duration of the project, cost effort of the project etc comes under SRS.

SRS goes as an input to design phase for the project. First we have to know the detailed structure of the project which is nothing but SRS, which tells us the project or task then proceed to design phase. Based on the feasibility study the SRS is prepared as below.

***2.1.1Purpose:-***

The purpose of this document is to describe all the external requirements for a organization. It also describes the interfaces for the system.

***2.1.2 Scope:-***

This document describes about the requirements of the banking system. It will be used by the developers and will be the basis of validating the final delivered system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for clarifications where necessary and will not make any alterations without the permission of the client.

***2.1.2.1 References:-***

Existing Manual Software Project Management

***2.1.2.2 Developers responsibilities overview:-***

The developer is responsible for

1. Developing the system
2. Installing the software on the client’ s hardware
3. For conducting any user training that might be needed for using the system.
4. Maintaining the system for a period of time after installation.

**2.1.3 General Description:-**

***2.1.3.1 User Characteristics:-***

The main users of the system will be Admin, Project Leader, Project Manager, and Developer, who are literate about computers and can use programs

**2.1.4 Functional Description**

***2.3.3.1 General description of Inputs and Outputs:-***

The system has 3inputs and produces 1 major outputs. The general description of these are as follows.

**2.2 Phases/Modules of The Project**

***2.2.1 User Interface:-***

The main user of the organization is Admin. The user interface is a GUI window where the command names reflect the function they perform.

***2.2.2 Hardware Interface:-***

This project should be run on Pentium and 512 RAM should be used.

***2.2.3 Software Interface:-***

Windows XP or Windows NT operating system should be used.

**2.2.4 Performance Requirements:-**

***2.2.4.1 Static Requirements:-***

These are the requirements that are essential for the better performance of the system before the system gets executed. These include the number of computers to be supported, the number of files and their sizes the system has to process. The server side computers should be highly configured with a memory capacity to store all the class files and the database. Client side computers should be appropriately configured.

***2.2.4.2 Dynamic Requirements:-***

A high capacity RAM and high frequency processor will enable the System to perform services fastly and efficiently.

**2.3 Identify Deliverables**

**2.3.1 Feasibility Study**

Feasibility study is an important phase in the software development process. It enables the developer to have an assessment of the product being developed. It refers to the feasibility study of the product in terms of the product, operational use and technical support required for implementing it.

Feasibility study should be performed on the basis of various criteria and parameters. The various feasibility studies are:

* Economic Feasibility
* Operational Feasibility
* Technical Feasibility

**2.3.1.1 Economic Feasibility:**

It refers to the benefits or outcomes we are deriving from the

product as compared to the total cost we are spending for developing the product. If the benefits are more or less the same as older system, then it is not feasible to develop product.

**2.3.1.2 Operational Feasibility:**

It refers to the feasibility of the product to be operational. Some

Products may work very well at design and implementation but may fail in the real time environment. It includes the study of additional human resource required and their technical expertise.

**2.3.1.3 Technical Feasibility:**

It refers to whether the software that is available in the market fully supports the present application. It studies the pros and cons of using particular software for the development and its feasibility. It also studies the additional training needed to be given to the people to make the application work.

**Chapter 3**

**Data Modeling**

**3.1 Work Products**

**All the Work Products specified below must be design & Maintained for further progress Of The Software Development**

Like most client programs, an email client is only active when a user runs it. The common arrangement is for an email user (the client) to make an arrangement with a remote [Mail Transfer Agent](https://en.wikipedia.org/wiki/Mail_Transfer_Agent) (MTA) server for the receipt and storage of the client's emails. The MTA, using a suitable [mail delivery agent](https://en.wikipedia.org/wiki/Mail_delivery_agent) (MDA), adds email messages to a client's storage as they arrive. The remote mail storage is referred to as the user's [mailbox](https://en.wikipedia.org/wiki/E-mail_Mailbox). The default setting on many Unix systems is for the mail server to store formatted messages in [box](https://en.wikipedia.org/wiki/Mbox), within the user's [home directory](https://en.wikipedia.org/wiki/Home_directory). Of course, users of the system can log-in and run a mail client on the same computer that hosts their mailboxes; in which case, the server is not actually *remote*, other than in a generic sense.

**3.2 Unified Modeling Language Diagrams(UML)**

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows**.**

***3.2.1 User Model View***

* This view represents the system from the users perspective.
* The analysis representation describes a usage scenario from the end-users perspective.

***3.2.2 Structural model view***

* In this model the data and functionality are arrived from inside the system.
* This model view models the static structures**.**

***3.2.3 Behavioral Model View***

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* + 1. ***Implementation Model View***

In this the structural and behavioral as parts of the system are represented as they are to be built.

***3.2.5 Environmental Model View***

* In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.
* UML is specifically constructed through two different domains they are
* UML Analysis modeling, which focuses on the user model and structural model views of the system
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

**3.2.6 Use Case Diagrams**

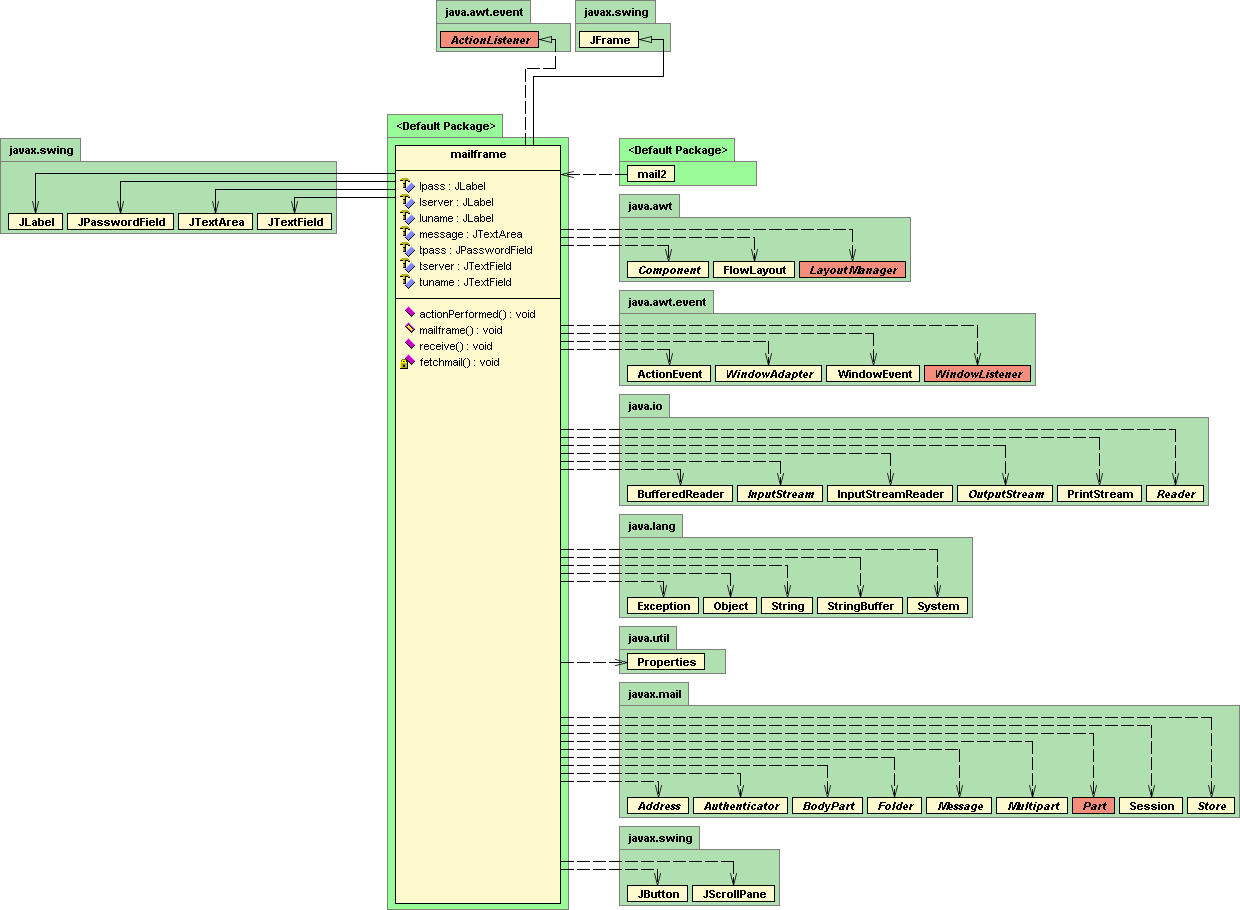
The actors identified upon the System are as follows.

* Receiver

#### **🚺**

Receiver

**3.2.7 Class Diagram**

******

**3.3 Forward Engineering**

**Problem Definition**

To perform forward engineering of the class diagram.

**Problem Description**

Forward Engineering is the process of transforming model into code.

**Pseudo code**

Following are the steps to generate java code from UML class diagram.

1. Start IBM Rational Software Architect
2. Create a class diagram
3. Go to File Menu
   1. Select New
      1. Click on Transformation configuration
      2. Provide the file name (e.g., fwd.tc)
      3. Select a transformation

3.1.3. Click UML to Java V1.3

* + 1. Click next
    2. Select the Source (The model which contains class diagram)
    3. Create a new target container( Java project to hold java code)
    4. Provide the java project name
    5. Click next
    6. Click finish

A transformation configuration file (fwd.tc) is created in the UML project.

1. Right click on fwd.tc

4.1. Click on transform

4.1.2. Click on UML to Java V1.3

1. Java code for the model is generated.

**3.4 Reverse Engineering**

**Problem Definition**

To perform Reverse Engineering.

**Problem Description**

Reverse Engineering is the process of transforming code to UML model.

**Pseudo code**

Following are the steps to generate UML model from java code.

1. Start IBM Rational Software Architect
2. Go to file

2.1 New

2.1.1 Click on Transformation configuration

* + 1. Provide the file name (e.g., rev.tc)
    2. Select a transformation

2.1.4 Click Java to UML.

* + 1. Click next
    2. Select the Source ( java project)
    3. Select a target container (UML model to hold the class diagram)
    4. Click next
    5. Click Finish

A transformation configuration file (rev.tc) is created in the Java project.

1. Right click on rev.tc
   1. Click on transform

3.1.2. Click on Java to UML

1. The Classes and interfaces present in java project will be listed in UML model.
2. Drag and drop from the model the graphical representation of classes and interfaces onto the class diagram window.
3. UML class diagram is created with appropriate relationships.Some relationships may not be reflected in the class diagram as there is loss of information during reverse engineering.

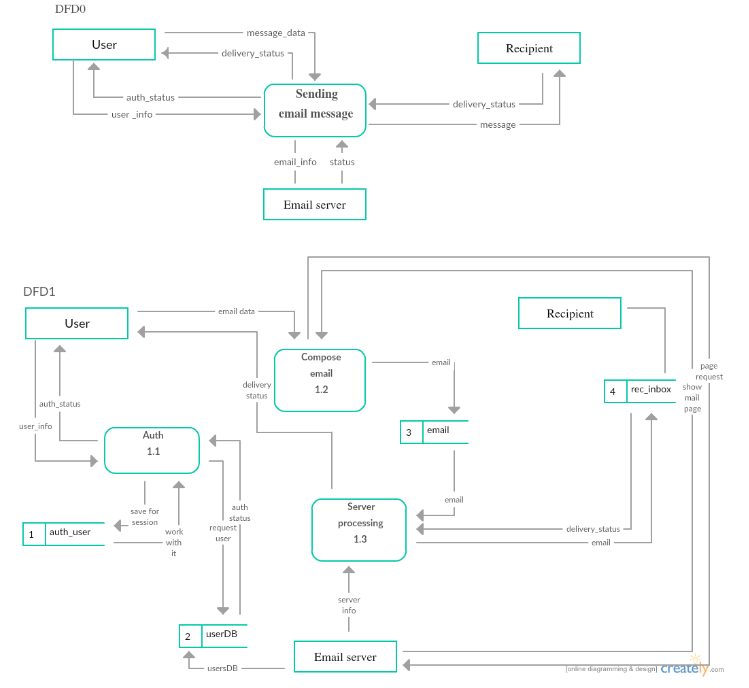
**3.5 Data Dictionary**

A data dictionary is a collection of descriptions of the [data](https://searchdatamanagement.techtarget.com/definition/data) objects or items in a data model for the benefit of programmers and others who need to refer to them. A first step in analyzing a system of [object](https://searchmicroservices.techtarget.com/definition/object)s with which users interact is to identify each object and its relationship to other objects. This process is called data modeling and results in a picture of object relationships. After each data object or item is given a descriptive name, its relationship is described (or it becomes part of some structure that implicitly describes relationship), the type of data (such as text or image or binary value) is described, possible predefined values are listed, and a brief textual description is provided. This collection can be organized for reference into a book called a data dictionary.

When developing programs that use the data model, a data dictionary can be consulted to understand where a data item fits in the structure, what values it may contain, and basically what the data item means in real-world terms. For example, a bank or group of banks could model the data objects involved in consumer banking. They could then provide a data dictionary for a bank's programmers. The data dictionary would describe each of the data items in its data model for consumer banking (for example, "Account holder" and ""Available credit").

**3.6 Data Flow Diagram**

A **data-flow diagram** (DFD) is a way of representing a flow of a data of a [process](https://en.wikipedia.org/wiki/Process) or a system (usually an [information system](https://en.wikipedia.org/wiki/Information_system)) The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a [flowchart](https://en.wikipedia.org/wiki/Flowchart)



**Chapter 4**

**Software Development & Debugging**

**4.1 Sample Code**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.io.BufferedReader;**

**import java.io.InputStreamReader;**

**import java.util.Properties;**

**import javax.mail.\*;**

**@SuppressWarnings("serial")**

**public class mail2 extends JFrame**

**{**

**static String popServer;**

**static String popUser;**

**static String popPassword;**

**public static void main(String args[])**

**{**

**//popServer=args[0];**

**//popUser=args[1];**

**//popPassword=args[2];**

**JFrame frame = new mailframe();**

**frame.setVisible(true);**

**}**

**}**

**@SuppressWarnings("serial")**

**class mailframe extends JFrame implements ActionListener**

**{**

**JLabel lserver =new JLabel("SMTP Server");**

**static JTextField tserver = new JTextField(15);**

**JLabel luname =new JLabel("UserName");**

**static JTextField tuname = new JTextField(15);**

**JLabel lpass =new JLabel("Password");**

**static JPasswordField tpass = new JPasswordField(15);**

**static JTextArea message = new JTextArea(30,72);**

**public mailframe()**

**{**

**this.setSize(850,500);**

**this.setLayout(new FlowLayout(FlowLayout.LEFT));**

**addWindowListener(new WindowAdapter()**

**{**

**public void windowClosing(WindowEvent e)**

**{**

**System.exit(0);**

**}**

**}**

**};**

**tpass.setEchoChar('\*');**

**this.add(lserver);**

**this.add(tserver);**

**this.add(luname);**

**this.add(tuname);**

**this.add(lpass);**

**this.add(tpass);**

**JButton b1 =new JButton("Fetch");**

**b1.addActionListener(this);**

**this.add(b1);**

**message.setEditable(false);**

**JScrollPane sp = new JScrollPane();**

**message.add(sp);**

**this.add(message);**

**}**

**public void actionPerformed(ActionEvent e)**

**{**

**System.out.println("debug button");**

**fetchmail();**

**}**

**@SuppressWarnings("deprecation")**

**private void fetchmail()**

**{**

**String popServer;**

**String popUser;**

**String popPassword;**

**popServer = mailframe.tserver.getText();**

**popUser = mailframe.tuname.getText();**

**popPassword = mailframe.tpass.getText();**

**System.out.println(popServer);**

**try**

**{**

**receive(popServer, popUser, popPassword);**

**//receive("mail.imparttechnologies.com","testproject@imparttechnologies.com", "password123");**

**}**

**catch (Exception ex)**

**{**

**System.out.println("Usage: java jmail"+" smtpServer address password "};**

**}**

**//System.exit(0);**

**}**

**public static void receive(String popServer, String popUser, String popPassword){**

**Store store=null;**

**Folder folder=null;**

**try**

**{**

**Properties props = System.getProperties();**

**Session session = Session.getDefaultInstance(props, null);**

**store = session.getStore("pop3");**

**store.connect(popServer, popUser, popPassword);**

**folder = store.getDefaultFolder();**

**if (folder == null) throw new Exception("No default folder");**

**folder = folder.getFolder("INBOX");**

**if (folder == null) throw new Exception("No POP3 INBOX");**

**folder.open(Folder.READ\_ONLY);**

**Message msgs[] = folder.getMessages();**

**BufferedReader reader = new BufferedReader (new InputStreamReader(System.in));**

**mailframe.message.setText("");**

**int msgNum;**

**for (msgNum = 0; msgNum < msgs.length; msgNum++)**

**{**

**System.out.println(msgNum +": "+ msgs[msgNum].getFrom()[0]+ "\t" + msgs[msgNum].getSubject());**

**System.out.println("Do you want to read message? [YES to read/QUIT to end]");**

**String line = reader.readLine();**

**String y = "yes";**

**int j = line.compareTo(y);**

**if (j == 0)**

**{**

**//System.out.println(((Message)msgs[msgNum].getContent()).toString());**

**//System.out.println(msgNum +": "+ msgs[msgNum].getFrom()[0]+ "\t" + msgs[msgNum].getSubject());**

**msgs[msgNum].writeTo(System.out);**

**//test run start -------->**

**Object content = msgs[msgNum].getContent();**

**if (content instanceof Multipart)**

**{**

**StringBuffer messageContent = new StringBuffer();**

**StringBuffer msg = new StringBuffer();**

**Multipart multipart = (Multipart) content;**

**for (int i = 0; i < multipart.getCount(); i++)**

**{**

**Part part = (Part) multipart.getBodyPart(i);**

**if (part.isMimeType("text/plain"))**

**{**

**msg = messageContent.append(part.getContent().toString());**

**String msg1 = new String();**

**String from = new String();**

**String subj = new String();**

**int k ;**

**k = msgNum;**

**msg1 = msg.toString();**

**subj = msgs[msgNum].getSubject();**

**from = msgs[msgNum].getFrom()[0].toString();**

**mailframe.message.append("Message No:"+(k+1)+"\n");**

**mailframe.message.append("Message From:"+from+"\n");**

**mailframe.message.append("Message Subject:"+subj+"\n");**

**mailframe.message.append("\n"+msg1+"\n");**

**}**

**}**

**//return messageContent.toString();**

**} else**

**{**

**//return content.toString();**

**}**

**//System.out.println("Debug 1 -> for 1 -> if yes");**

**}**

**else**

**{**

**System.out.println("Closing Previous Message. Going For Next.");**

**//break;**

**}**

**//msgs[msgNum].writeTo(System.out);**

**}**

**//System.exit(0);**

**}**

**catch (Exception ex)**

**{**

**ex.printStackTrace();**

**}**

**finally**

**{**

**try**

**{**

**if (folder!=null) folder.close(false);**

**if (store!=null) store.close();**

**}**

**catch (Exception ex2) {ex2.printStackTrace();**

**}**

**}**

**}**

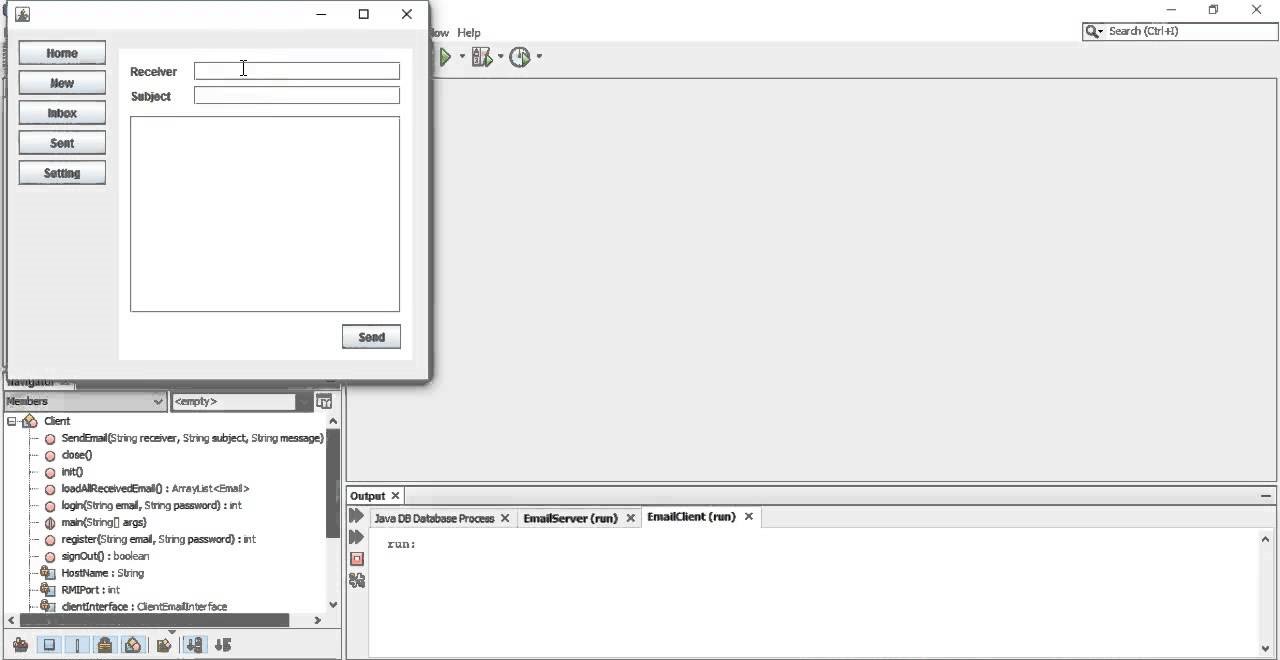
**}**

**4.2 Application Running**

* Go to **File > New Project**
* Under *Categories*, choose **Web** and then select to create a simple **Web Application**.
* Click **Next** and **Type** in *Project Name* as **JavaMail**. Be sure to select **Glassfish** as your server.
* Click **Finish** and allow NetBeans to create files for your project.

**4.3 Sample Screen Shots**

Screen shots are the screens that clearly explain the project through Graphical User Interface (GUI). The following are the screens that explain the project.



An email message travels through at least two main **SMTP servers** that belong to the senders and the recipients.

First, **SMTP** connects your client with your email provider’s server. Next, it checks the email header for relevant information about the sender and the recipient’s address.

Once a destination is determined, the server will check the location of the domain associated with the address in the [**Domain Name System**](https://www.hostinger.com/tutorials/what-is-dns).

For example, if you are trying to send a message to **emailuser@gmail.com**, the server locates **gmail.com**and relays the message to that specific computer.

Then, the recipient’s SMTP server delivers the message to the server’s mailbox until the intended user logs in to their email account. When that happens, either **POP3**or **IMAP**will forward the new message to the recipient’s email client so they can view it.

**Chapter 5**

**Software Testing**

**5.1 Testing Description**

The purpose of testing is to assess product quality. It helps to strengthen and stabilize the architecture early in the development cycle. We can verify through testing, the various interactions, integration of components and the requirements which were implemented. It provides timely feedback to resolve the quality issues, in a timely and cost effective manner. The test workflow involves the following:

* Verifying the interactions of components.
* Verifying the proper integration of components.
* Verifying that all requirements have been implemented correctly.
* Identifying and ensuring that all discovered defects are addressed before the software is deployed.

**Types of Testing:**

After a test plan has been developed, system testing begins by testing program modules separately, followed by testing “bundled” modules as a unit. A program module may function perfectly in isolation but fail when interfaced with other modules. The approach is to test each entity with successively larger ones, up to the system test level.

System testing consists of the following steps:

**Program testing**

A program represents the logical elements of system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly with other programs. Achieving an error-free program is the responsibility of the programmer. Program testing checks for two types of errors: syntax and logic. A syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted key words are common syntax errors. These errors are shown through error messages generated by syntax errors. These errors are shown through error messages generated by the computer. A logic error, on the other hand, deals with incorrect data fields, out-of range items, and invalid combinations. Since diagnostics do not detect logic errors, the programmer must examine the output carefully for them.

**String Testing**

Programs are invariably related to one another and interact in a total system. Each program is tested to see whether it conforms to related programs in the system. Each portion of the system is tested against the entire module with both test and live data before the entire system is ready to be tested.

**System Testing**

System testing is designed to uncover weaknesses that were not found in earlier tests..This includes forced system failure and validation of the total system as it will be implemented by its user(s) in the operational environment. Generally, it begins with low volumes of transactions based on live data. The volume is increased until the maximum level for each transaction type is reached. The total system is also tested for recovery and fallback after various major failures to ensure that no data are lost during the emergency. All this is done with the old system still in operation. After the candidate system passes the test, the old system is discontinued.

**User Acceptance Testing**

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system’s procedures operate to system specifications and that the integrity of vital data is maintained. Performance of an acceptance test is actually the user’s show. User motivation and knowledge are critical for the successful performance of the system. Then a comprehensive test report is prepared. The report indicates the system’s tolerance, performance range, error rate, and accuracy.

**System Documentation**

All design and test documentation should be finalized and entered in the library for future reference. The library is the central location for maintenance of the new system. The format, organization, and language of each documentation should be in line with system standards.

**5.2 Validation (Test-Cases)**

**Mail Client Execution:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Input** | **Description** | **Expected result** | **Pass/Fail** |
| **Tc\_1** | Server name blank, user name  password blank | No server name, username,Password,  Received by server | Display error | Fail |
| **Tc\_2** | Valid server name valid user name valid password | Valid server name valid user name password given | Mails displayed | Pass |
| **Tc\_3** | Invalid server name | Invalid server name is given to server | You should enter valid server name | Fail |
| **Tc\_4** | Type wrong command | Command is wrong | Display as default | Fail |

**5.2 Integration testing**

Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group.A typical software project consists of multiple software modules, coded by different programmers. Integration Testing focuses on checking data communication amongst these modules.Hence it is also termed as **'I & T'** (Integration and Testing), **'String Testing'** and sometimes 'Thread Testing'.

Although each software module is unit tested, defects still exist for various reasons like

* A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
* At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
* Interfaces of the software modules with the database could be erroneous
* External Hardware interfaces, if any, could be erroneous
* Inadequate exception handling could cause issues.

## Example of Integration Test Case

Integration[Test Case](https://www.guru99.com/test-case.html)differs from other test cases in the sense **it focuses mainly on the interfaces & flow of data/information between the modules**. Here priority is to be given for the **integrating links** rather than the unit functions which are already tested.

Sample Integration Test Cases for the following scenario: Application has 3 modules say 'Login Page', 'Mailbox' and 'Delete emails' and each of them is integrated logically.

Here do not concentrate much on the Login Page testing as it's already been done in [Unit Testing](https://www.guru99.com/unit-testing-guide.html). But check how it's linked to the Mail Box Page.

Similarly Mail Box: Check its integration to the Delete Mails Module.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Check the interface link between the Login and Mailbox module | Enter login credentials and click on the Login button | To be directed to the Mail Box |
| **2** | Check the interface link between the Mailbox and Delete Mails Module | From Mailbox select the email and click a delete button | Selected email should appear in the Deleted/Trash folder |

## **Approaches/Methodologies/Strategies of Integration Testing:**

Software Engineering defines variety of strategies to execute Integration testing, viz.

* Big Bang Approach :
* Incremental Approach: which is further divided into the following
  + Top Down Approach
  + Bottom Up Approach
  + Sandwich Approach - Combination of Top Down and Bottom Up

**Chapter 6**

**Conclusion**

This system is implemented fulfilling all the client requirements. The interfaces designed for the system is very user friendly and attractive. It has successfully implemented the receive messages as per the client requirement.

The system has successfully passed the testing at the development site and is under the testing phase in the presence of the client. The system is waiting for the client response.

**Chapter 7**

**Future Enhancement**

This object is having a broad future scope as it can be extended to provide sending messages on line. This system can be implemented for online sharing without the intervention of the authority. If it is done so user can read messages from any where in the world. He can read any where from is login account ,SMTP server . In other words the future scope is to provide the better service.

**References**

1. [www.google.com](http://www.google.com)
2. [www.wikepedia.com](http://www.wikepedia.com)
3. [www.sunmicrosystems.com](http://www.sunmicrosystems.com)
4. [www.encyclopedia.com](http://www.encyclopedia.com)