SIX WEEKS INDUSTRIAL TRAINING PROJECT REPORT

ON

DIGITAL LETTER BOX

AT

HEWLETT PACKARD ENTERPRISE

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

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Palak Aggarwal

11501311

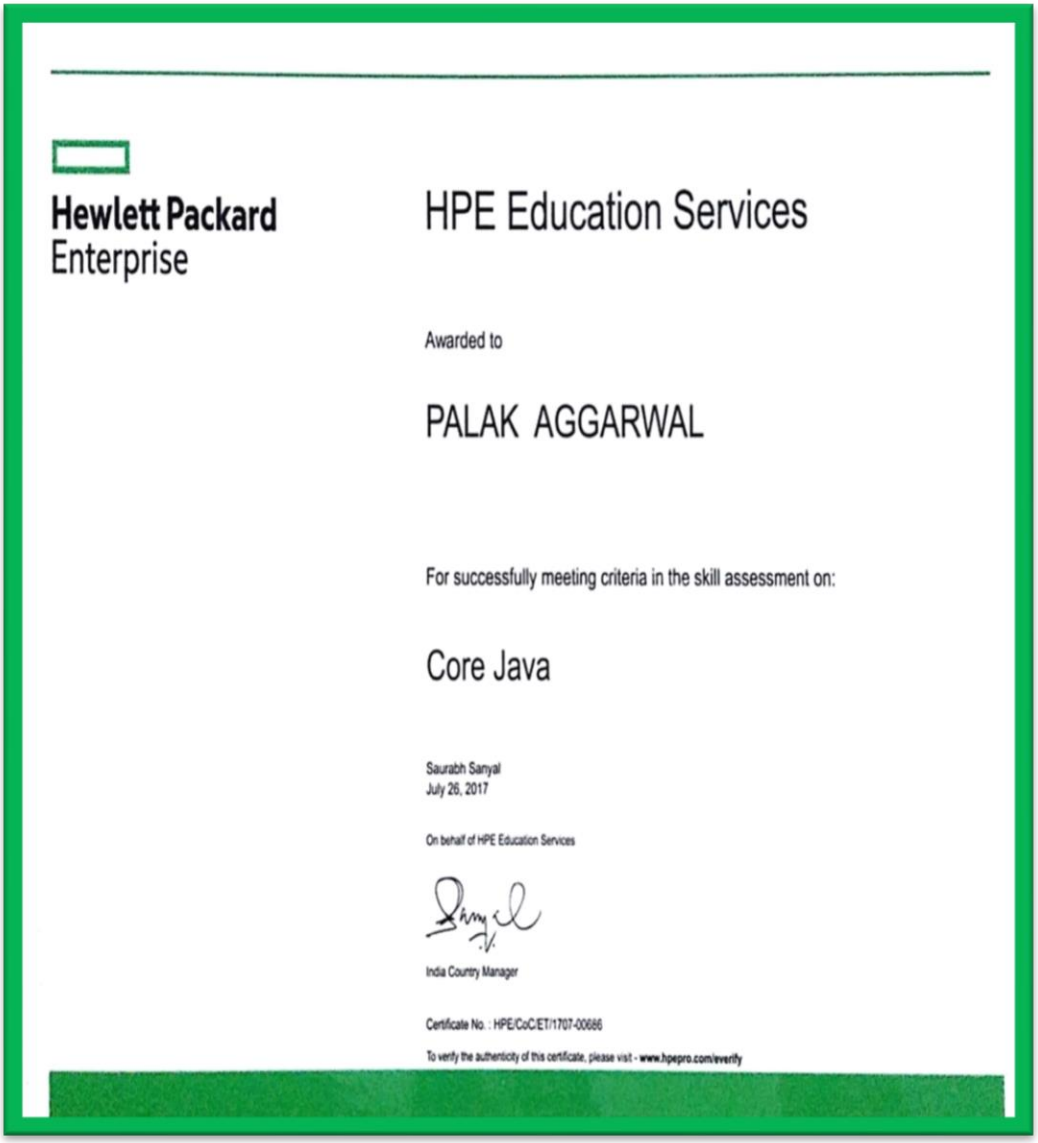
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DAV UNIVERSITY

JALANDHAR-PATHANKOT NATIONAL HIGHWAY, NH 44,

SARMASTPUR PUNJAB – 144012

CERIFICATE



Hewlett Packard Enterprise

HPE Education Services

Awarded to

PALAK AGGARWAL

For successfully meeting criteria in the skill assessment on:

Core Java

Saurabh Sanyal
July 26, 2017

On behalf of HPE Education Services

India Country Manager

Certificate No. : HPE/CoC/ET/1707-00686

To verify the authenticity of this certificate, please visit - www.hpepro.com/everify

DECLARATION

I hereby declare that the Industrial Training Report is an authentic record of my own work as requirements of Industrial Training during the period from 15th June, 2017 to 15th July, 2017 for the award of degree of B. Tech in Computer Science and Engineering, DAV University, Sarmastpur, Jalandhar.

Palak Aggarwal

Date: **11501311**

(C118A26)

ACKNOWLEDGEMENT

I express my gratitude to all those who helped us in various stages of the development of this project. First, I would like to express my sincere gratitude indebtedness to Dr A. K. Paul (Vice-Chancellor), Dr Jasbir Rishi (Dean Academics) and Er Nishi (Coordinator, Department of Computer Science and Engineering) DAV University for allowing me to undergo the six weeks industrial training at Hewlett Packard Enterprise House Summer Training.

I am also thankful to all faculty members of Department of Computer Science and Engineering, for their true help, inspiration and for helping me for the preparation of the final report and presentation. Lastly, I pay my sincere thanks and gratitude to all the Staff Members of Hewlett Packard Enterprise for their support and for making our training valuable and fruitful.

CONTENTS

LIST OF FIGURES

CHAPTER – 1
INTRODUCTION

"DIGITAL LETTER BOX" is based on Java Programming.

1.1 Training Area – Core Java Programming

1.1.1 History

The history of java starts from Green Team.

1) James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991. The small team of sun engineers called Green Team.

2) Firstly, it was called "Greentalk" by James Gosling and file extension was .gt.

3) After that, it was called Oak as Oak is a symbol of strength and chosen as a national tree of many countries like U.S.A., France, Germany, Romania etc. and was developed as a part of the Green project.

4) In 1995, Oak was renamed as "Java" because it was already a trademark by Oak Technologies.

5) They wanted something that reflected the essence of the technology: revolutionary, dynamic, lively, cool, unique, and easy to spell and fun to say. According to James Gosling "Java was one of the top choices along with Silk". Since java was so unique, most of the team members preferred java

6) Java is an island of Indonesia where first coffee was produced (called java coffee).

7) In 1995, Time magazine called Java one of the Ten Best Products of 1995.

8) JDK 1.0(Java Development Kit version 1.0) released in (January 23, 1996).

9) There are various versions of Java has been released till today. Current available release of Java is Java SE8. Earlier versions are as following:

10) **The Java Development Kit (JDK)** is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

11) **JVM**

A specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.

An implementation Its implementation is known as JRE (Java Runtime Environment).

Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

1.1.2 Features

Features of java are also known as java buzzwords. The Java Features given below are simple and easy to understand.

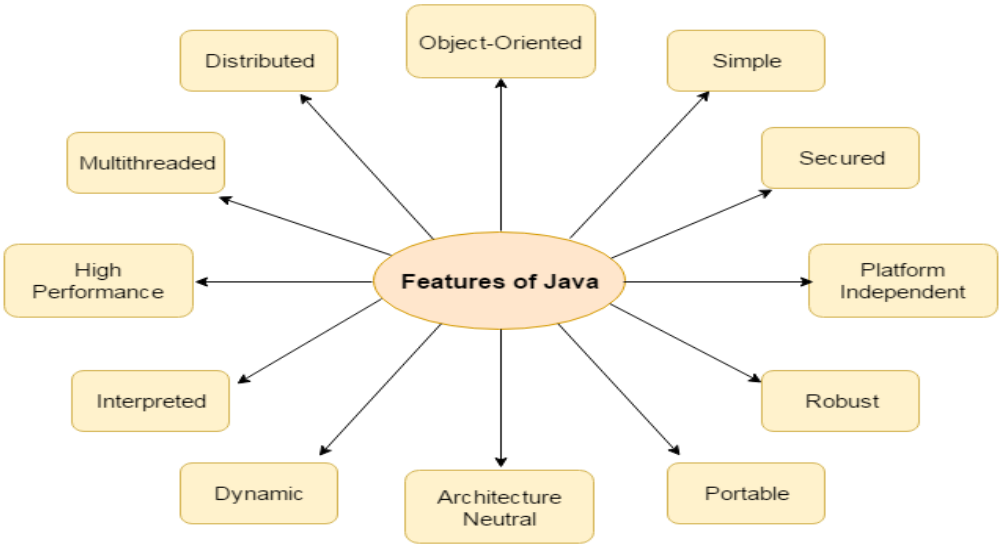


Figure 1.1: Features of Java

1.1.2.1 **Simple**

Java language is simple because:

1. Syntax is based on C++ (so easier for programmers to learn it after C++).
2. Removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc.
3. No need to remove unreferenced objects because there is Automatic Garbage Collection in java.

1.1.2.2 **Object-Oriented**

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour.

Object-oriented programming (OOPs) is a methodology that simplifies software

development and maintenance by providing some rules.

Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

1.1.2.3 **Platform Independent**

A platform is the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform. The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris and Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere (WORA).

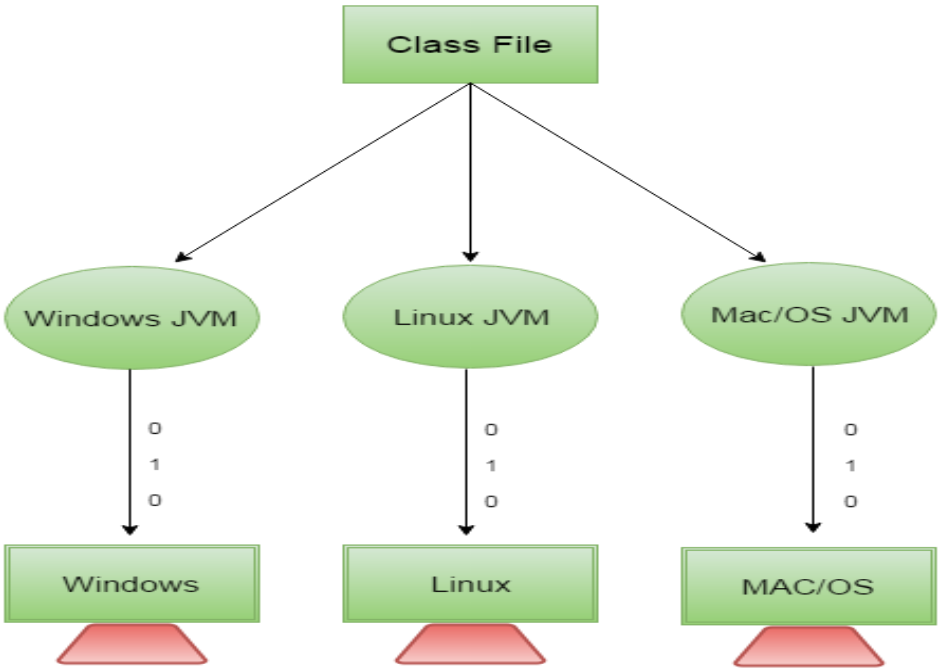


Figure 1.2: Platform Independence

1.1.2.4 **Secured**

Java is secured because:

1. No explicit pointer
2. Java Programs run inside virtual machine sandbox.

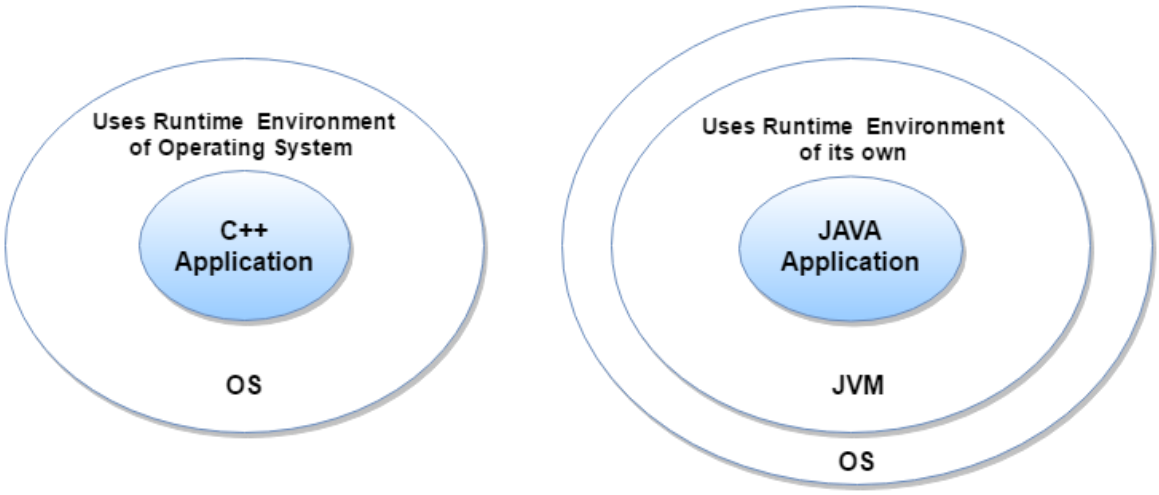


Figure 1.3: Java Runtime Environment

3. Classloader: adds security by separating the package for the classes of the local file system from those that are imported from network sources.
4. Bytecode Verifier: checks the code fragments for illegal code that can violate access right to objects.
5. Security Manager: determines what resources a class can access such as reading and writing to the local disk.

These securities are provided by java language. Some security can also be provided by application developer through SSL, JAAS and Cryptography etc.

1.1.2.5 **Robust**

Robust simply means strong. Java uses strong memory management. It lacks pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points make java robust.

1.1.2.6 **Architecture-neutral**

There are no implementation dependent features, e.g. size of primitive types is fixed. In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.

1.1.2.7 **Portable**

We may carry the java bytecode to any platform.

1.1.2.8 **High-Performance**

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++).

1.1.2.9 **Distributed**

We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

1.1.2.10 **Multi-Threaded**

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications etc.

1.1.3 Applications and Scope of Java

Java language has become the backbone of millions of applications across multiple platforms including Windows, Macintosh and UNIX-based desktops, Android-based mobiles, embedded systems and enterprise solutions. According to Oracle (that acquired Sun Microsystems in 2010), Java now runs on more than 3 billion devices.

Types of Applications that Run on Java

1.1.3.1 **Desktop GUI Applications**

Java provides GUI development through various means like Abstract Windowing Toolkit (AWT), Swing and JavaFX. While AWT contains a number of pre-constructed components such as menu, button, list, and numerous third-party components, Swing, a GUI widget toolkit, additionally provides certain advanced components like trees, tables, scroll panes, tabbed panel and lists. JavaFX, a set of graphics and media packages, provides Swing interoperability, 3D graphic features and self-contained deployment model which facilitates quick scripting of Java applets and applications.

1.1.3.2 **Mobile Applications**

Java Platform, Micro Edition (Java ME or J2ME) is a cross-platform framework to build applications that run across all Java supported devices, including feature phones and smart phones. Further, applications for Android, one of the most

popular mobile operating systems, are usually scripted in Java using the Android Software Development Kit (SDK) or other environments.

1.1.3.3 Embedded Systems

Embedded systems, ranging from tiny chips to specialized computers, are components of larger electromechanical systems performing dedicated tasks. Several devices, such as SIM cards, blue-ray disk players, utility meters and televisions, use embedded Java technologies. According to Oracle, 100% of Blu-ray Disc Players and 125 million TV devices employ Java.

1.1.3.4 Web Applications

Java provides support for web applications through Servlets, Struts or JSPs. The easy programming and higher security offered by the programming language has allowed a large number of government applications for health, social security, education and insurance to be based on Java. Java also finds application in development of ecommerce web applications using open-source ecommerce platforms, such as Broadleaf.

1.1.3.5 Web Servers and Application Servers

The Java ecosystem today contains multiple Java web servers and application servers. While Apache Tomcat, Simple, Jo!, Rimfaxe Web Server (RWS) and Project Jigsaw dominate the web server space, WebLogic, WebSphere, and Jboss EAP dominate commercial application server space.

1.1.3.6 Enterprise Applications

Java Enterprise Edition (Java EE) is a popular platform that provides API and runtime environment for scripting and running enterprise software, including network applications and web-services. Oracle claims Java is running in 97% of enterprise computers. The higher performance guarantee and faster computing in Java has resulted in high frequency trading systems like Murex to be scripted in the language. It is also the backbone for a variety of banking applications which have Java running from front user end to back server end.

1.1.3.7 Scientific Applications

Java is the choice of many software developers for writing applications involving scientific calculations and mathematical operations. These programs are generally considered to be fast and secure, have a higher degree of portability and low maintenance. Applications like MATLAB use Java both for interacting user interface and as part of the core system.

1.1.4 BASIC CONCEPTS USED IN PROECT

1.1.4.1 **Java Swings**

"Swing" refers to the new library of GUI controls (buttons, sliders, checkboxes, etc.) that replaces the somewhat weak and inflexible AWT controls.

The Swing classes eliminate Java's biggest weakness: its relatively primitive user interface toolkit. Java Swing helps you to take full advantage of the Swing classes, providing detailed descriptions of every class and interface in the key Swing packages. It shows you how to use all of the new components, allowing you to build state-of-the-art user interfaces and giving you the context you need to understand what you're doing. It's more than documentation; *Java Swing* helps you develop code quickly and effectively.

1. Main New Features
   - Lightweight. Not built on native window-system windows.
   - Much bigger set of built-in controls: Trees, image buttons, tabbed panes, sliders, toolbars, colour choosers, tables, text areas to display HTML or RTF, etc.
   - Much more customizable. Can change border, text alignment, or add image to almost any control. Can customize how minor features are drawn. Can separate internal representation from visual appearance.
   - "Pluggable" look and feel. Can change look and feel at runtime, or design own look and feel.
   - Many miscellaneous new features. Double-buffering built in, tool tips, dock able tool bars, keyboard accelerators, custom cursors, etc.
2. Components are named J*Xxx*.

   For example: JFrame, JPanel, JApplet, JDialog, JButton, etc.

   J Component

   The J Component class is the root of the Visual component class hierarchy in JFC. All Swing components are implemented as subclass of J components class, which inherits from the Container class. Swing component inherit the following functionality from J Component

   - Borders- Using the setBorder() method, you can specify the border that a component displays around its edges. You can specify that

component have extra space around its edges using an EmptyBorder instance.

- Double Buffering- It can improve the appearance of frequently changing components. Now you do not have to write the double buffering code because Swing provides it for you. By default Swing components are double buffered.
- Tool Tips- By specifying a string with the setToolTipNext() method, you can provide help to users of a components. When the cursor pauses over the components, the specified string is displayed in small window near the component.
- Look and Feel- Subject to the security restriction, you can choose the look and feel used by all Swing components by invoking the UIManager.setLookAndFeel() method.

3. There is an almost-equivalent Swing component for most AWT components.

4. Instead of adding components directly to frames or applets, we can use the content pane.

- Add to content pane via getContentPane().add
- Replace content pane via setContentPane

5. Model-View-Controller architecture let us change the internal data representation for lists, trees, tables, etc.

6. Swing was in the com.sun.java.swing package in beta releases of 1.2. Switched to javax.swing in 1.2 final.

7. Default "look and feel" is a Java-specific one.

- Need special call to get native look
- Default called "Java look & feel"

8. Mixing AWT and Swing is doomed.

AWT components are always on top, and z-ordering problems catch you in many unexpected ways. Stick with the AWT or move completely to Swing.

1.1.4.2 JAVA DATABASE CONNECTIVITY - JDBC

The JDBC is a set of the database access classes. The very term JDBC stands for “Java Database Connectivity”. It was developed by Java Soft. JDBC technology is an API (Application Program Interface) that allows virtual access to any tabular

data source from the Java programming language by means of some connecting software called Drivers. It provides cross-DBMS connectivity to a wide range of SQL databases. JDBC defines a set of interfaces to enable developers to access data independently of the actual database product used to store the data. JDBC allow Java applets, Servlets, and application to access data in famous database management systems.

It also provides access to other tabular data sources, such as spreadsheets or flat files. The JDBC API allows developers to take advantage of the Java platform's "Write Once, Run Anywhere" capabilities for industrial strength, cross-platform applications that require access to enterprise data. With a JDBC technology-enabled driver, a developer can easily connect all corporate data even in a heterogeneous environment. The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases. The JDBC API makes it possible to do three things:

- Establish a connection with a database or access any tabular data source
- Send SQL statements
- Process the results

Steps in using JDBC

1) Create a Connection type of object (A) denoting a connection to the database.
2) Create a Statement type of object (B) using the A.
3) Use B to execute either update the database or send a query request.
4) The result of the query operation in step 3 is a Result Set type of object(C)
5) C is actually a small table (D) consisting of the result of the query.
6) D can be handled according to the user needs
7) Close C,B and A

1.1.4.3 SQL

SQL is used for creating and maintaining data in tables where ever JDBC is used. SQL stands for Structured Query Language, better known as “sequel”. It is used for:

1. Querying a database by editing the SQL statements.
2. Querying a database within a program
3. Defining data organization.
4. Administrating data

5. Accessing multiple data servers
6. Managing transaction

1.2 Training Company

1.2.1 Hewlett Packard Enterprises

HPE Authorized Education Delivery Partners gives experience to work with supremacy companies to deliver training's and meet the requirements of learning partners. **HPE** Authorized Education Delivery Partner has permission and authorization to meet the **HPE** standards approved with course ware to complete the level of competency required for certification.

HPE help customers to use technology to slash the time it takes to turn ideas into value. In turn, they transform industries, markets and lives. Some of their customers run traditional IT environments. Most are transitioning to a secure, cloud-enabled, mobile-friendly infrastructure. Many rely on a combination of both. Wherever their customers are in that journey, they provide the technology and solutions to help them succeed. The advanced research from Hewlett Packard Labs changes the world.

They are a powerful innovation engine for HPE, their customers and industry delivering breakthrough technologies and pioneering revolutionary research. They address everything from IT trends to complex consumer and social challenges. Visit Hewlett Packard Labs to learn how HPE researchers are transforming the technology industry and pioneering the future of computing.

Hewlett Packard Training in Chandigarh is an Industrial scenario that needs constant technical enhancements to serve to the fast demands. HP Training Chandigarh is the largest company in the world, which provides high quality courses. In HP Training centre in Chandigarh be the master and get the direction and depth that you may have only heard in intense technology domain.

1.2.2 Why HPE for training?

HPE visited DAV University for the House Summer Training program, as company have a lot of fame over the world as well as excellent staff members like **Mr Gagan Marken** who is ready to help us any time impressed us. He welcome our doubts even if we call him in mid night and took our training sessions for 6 to 7 hours every day That is why I opted this company for my 6 weeks training after the 4^{th} semester.

1.2.3 Work Areas

I have worked on core java. In which I worked on basic java syntax, Object Oriented Programming Concept, GUI i.e. swings, Socket Programming, Multi-Threading, File Handling.

CHAPTER – 2

DIGITAL LETTER BOX

Digital Letter Box will be transforming project if it will be implemented in any organization. How it will transform is explained below:

2.1 DEFINITION

Digital Letter Box is one of the computer based **SERVER – CLIENT** application developed in core java used to send application to the any person of institution/company to whom you want to send but sender and receiver both should be members of same organisation.

2.2 OVERVIEW

As if we see in our university i.e. DAV University, Jalandhar is a big campus, if we have to submit any kind of application to any of the higher authorities or our teachers we have to wander a lot here and there. But using this Digital Letter Box we can easily send application any of the higher authorities/teachers.

In other words, if the live server of the organisation is connected to it than the people associated with the organisation can send or receive application by logging in to application and one who have access to receive the applications can login receive the application and accept or reject it. If accepted or rejected the status will be reflected to the sender

2.3 SPECIFICATIONS

1) It can be implemented in any of the big or small organisation. It will help to reduce the little bit of work load from higher authorities. Moreover it will reduce the human effort and save time wasted in wandering here and there for submitting application.
2) Requirements

   Tools used:

   - IDE NetBeans v8.2
   - JDK SE8
   - Wamp Sql Server
   - Knowledge of Java Syntax

Programming concepts – class, packages, interfaces, swing, JDBC, Socket Programming, and File Handling

Knowledge of Database concepts

3) To send an application

   You have to login with your ID assigned, password, and contact no.

   Select department and teacher name as it is implemented for university.

   Enter application subject.

   Attach application File.

   Click on submit to send.

   Accepted or rejected status will be reflected to you in the sent box.

4) For receiver end

   Firstly login

   Check application

   Click on subject to download the file

   Click on accept or reject as per the choice

5) It require database of organisation

CHAPTER – 3

OBJECTIVES

3.1 CORE JAVA PROGRAMMING

In this program we have learnt about the following things:

- Making classes in java
- Inheritance
- Polymorphism
- Interfaces
- Packages
- File handling
- Database concepts (for implementation of JDBC)
- JDBC – Java Database Connectivity
- File Handling
- Abstract Classes Concept
- Applets
- Exception Handling
- GUI – Swings
- Multi - Threading
- Socket Programming – to make client-server connections

Tasks assigned are following:

- Basic code to print hello java
- Codes on printing various patterns
- Codes to draw hut and other objects using Applets.
- Codes to use packages and sub packages
- Codes for implementation of swings like making forms to use JFrame, JPanels, JButtons, JLables, JFileChooser, JRadioBox , JCheckBox, JComboBox etc.
- Codes to implementation of JDBC like fetch of data to table used in java code from database
- Codes for updating data to the database using GUI in Java
- Codes for the implementation of Socket Programming

- Implementation of Tic Tac Toe game over GUI using swing and for the better understanding of control statements and operators as well as GUI.

After all this work I moved to my project “DIGITAL LETTER BOX”, which require understanding of maximum of the above concepts.

3.2 GOALS OF THE DIGITAL LETTER BOX

- To digitise the procedure of submitting applications.
- To facilitate sending of any kind of applications like scholarship application, complaint applications, hostel change applications or any other application you want to send.
- To minimize human effort as sending application online instead of wandering here and there to submit application.
- To save time for wandering to submit application in search of the person to whom application to be submitted or other resources like hardcopy print outs etc.

CHAPTER – 4

WORK PLAN

I chose this particular project, as I feel it very difficult to go to registrar office again and again to submit applications. Lot of time is wasted while going to admin block and again come back. After doing these much efforts then again there are possibilities that mam would not be there, student meeting time up. If everything is in favour then don't know application will be accepted or rejected. So to end up all this mess I decide to make this application so that it will be easy task for we students and higher authorities to check lots of application.

4.1 BASIC DESIGN OF PROJECT

4.1.1 Sender End

Welcome page will be displayed having a button of sending application click it to login as shown in figure 4.1.

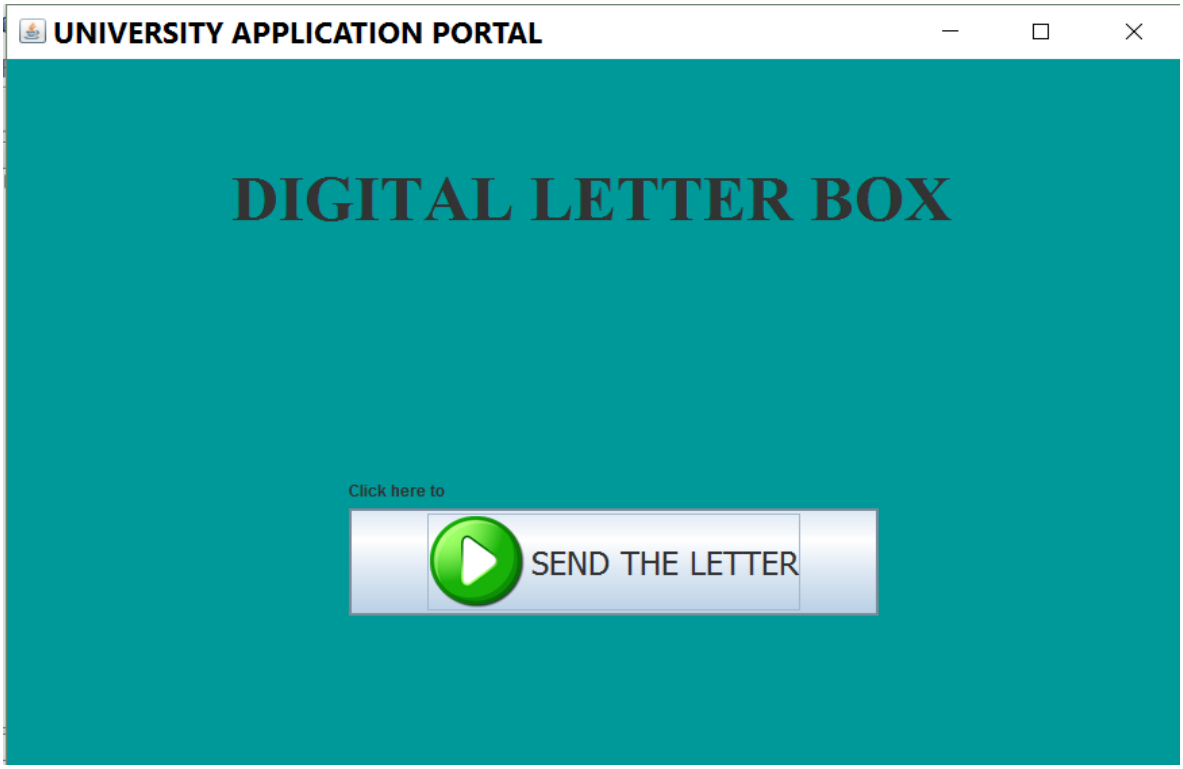


Figure 4.1: Welcome Frame

Login page will be there where sender to fill the User Id, Password, Contact No. as shown in figure 4.2 login button is there to login and go to next frame.

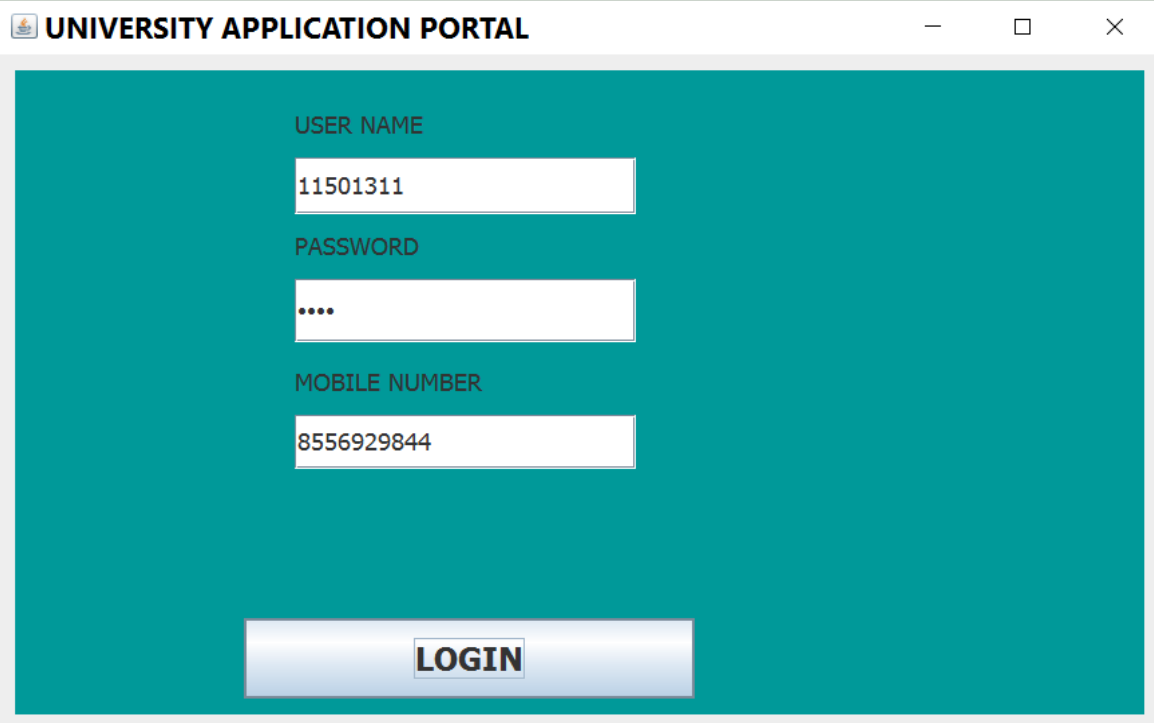


Figure 4.2: Login Frame

After login, select the department from first combo box corresponding person in particular department will be there in other combo box should be selected. Then enter the application subject and click on attach file button and after enclosing file click on OK to submit as shown in figure 4.3, 4.4, 4.5. On this frame you can also directly go to your inbox.

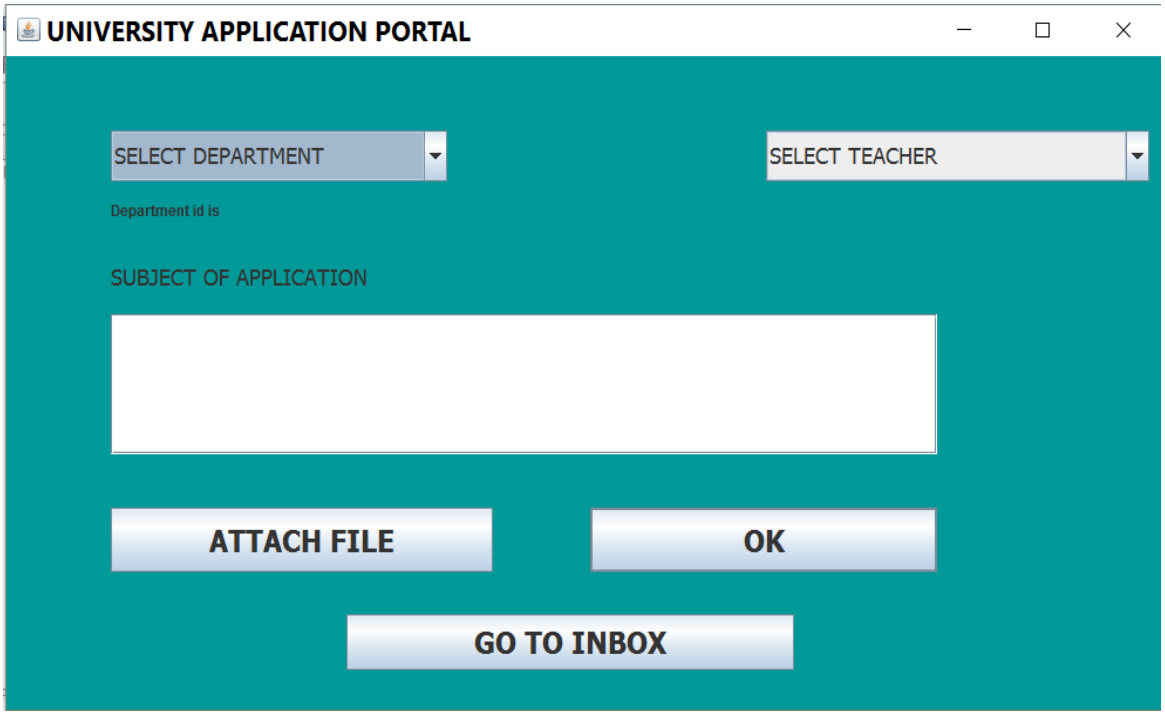


Figure 4.3: Application Compose Frame

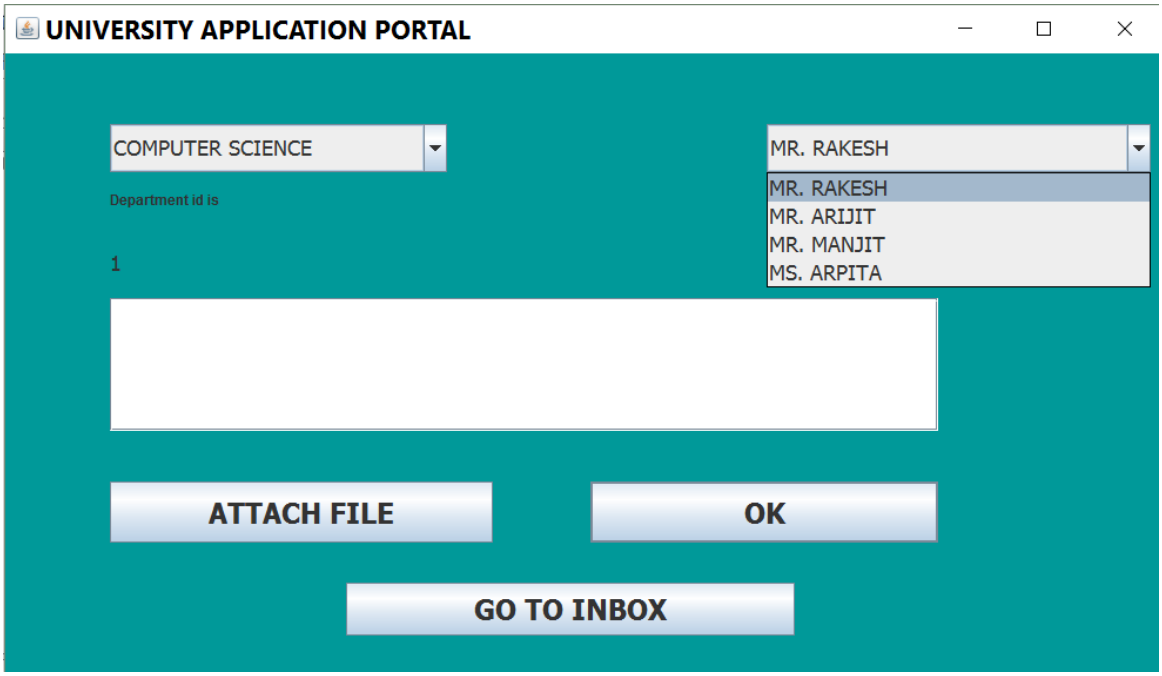


Figure 4.4: Teacher to be selected from corresponding department

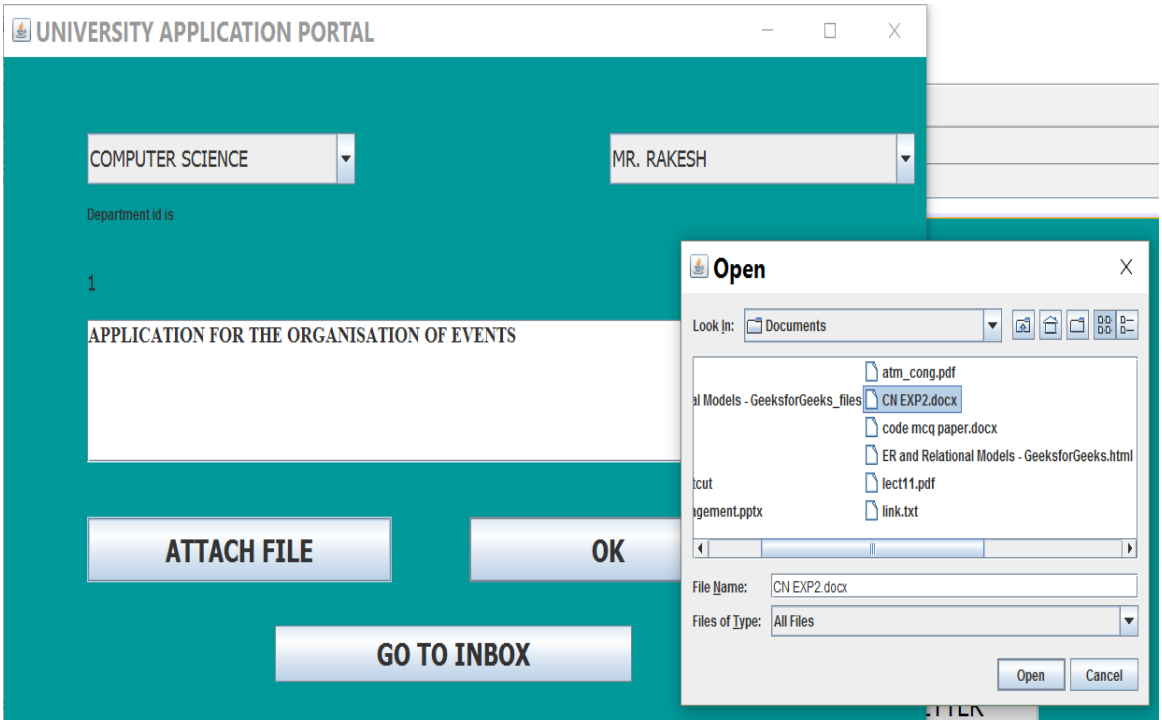


Figure 4.5: Attach file

You will be displayed a table of applications you have sent with their status either pending or accepted/rejected as shown in figure 4.6. Finally you can logout or compose a new application or clear/delete your table data.

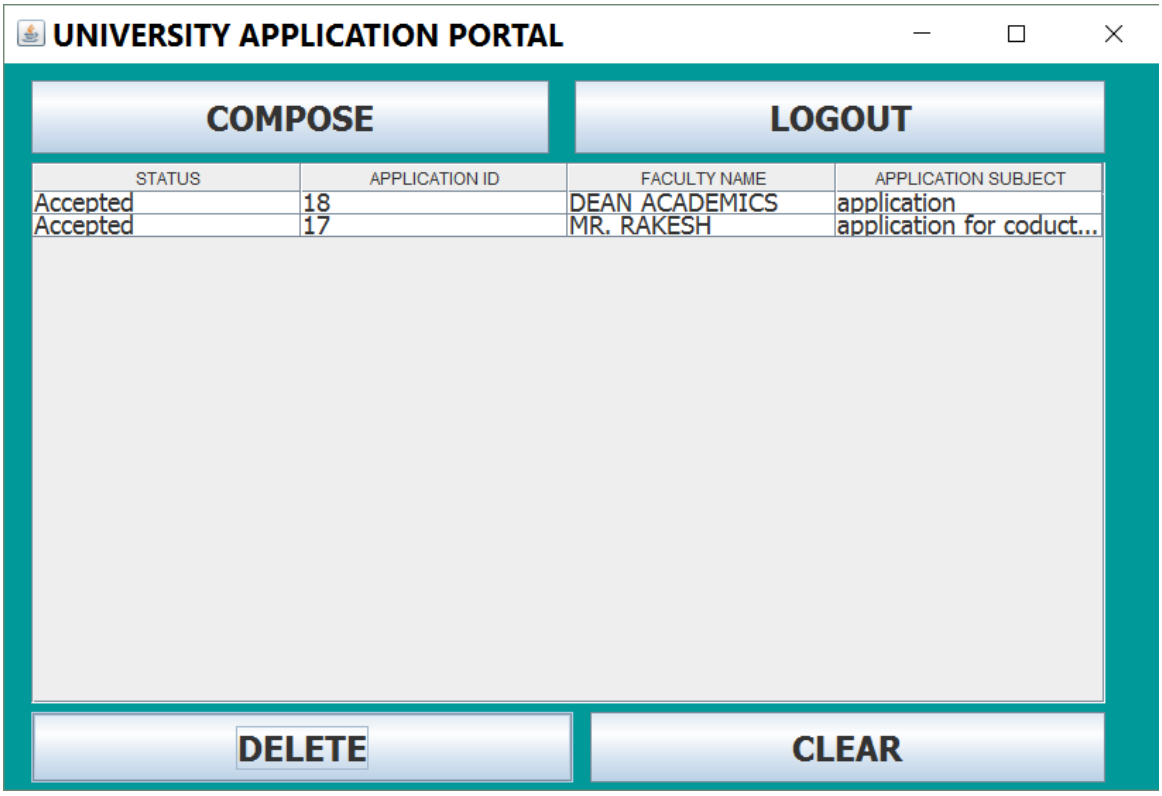


Figure 4.6: Sentbox

4.1.2 Receiver End

Welcome frame will be displayed click on check the applications button as shown in figure 4.7.

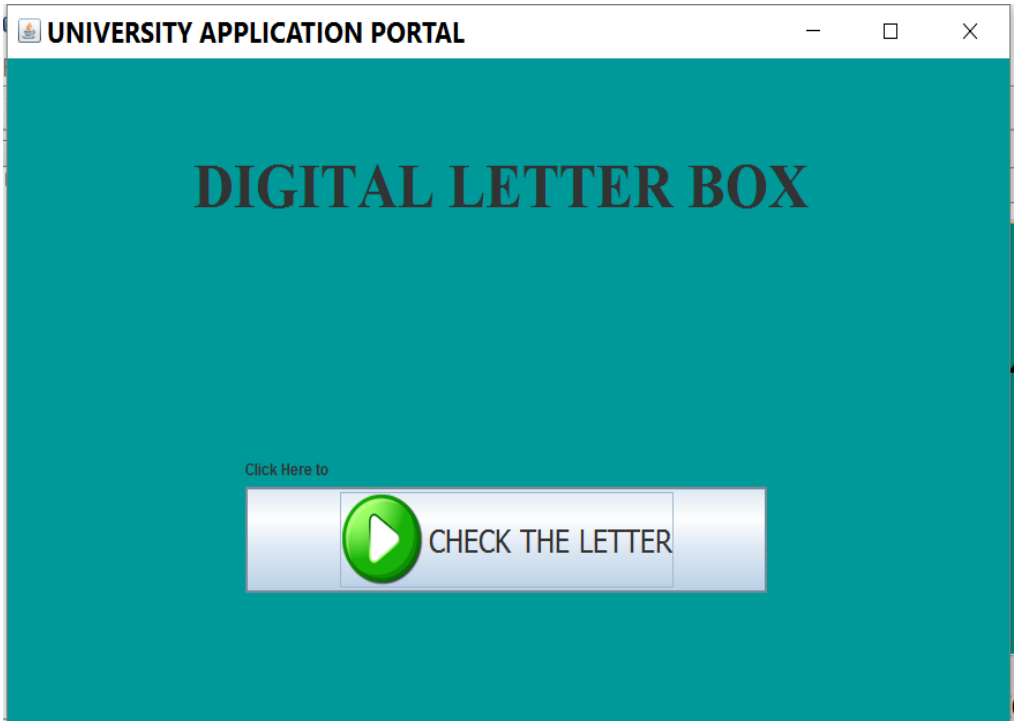


Figure 4.7 Welcome Frame

On the next frame user name and password to be entered as shown by figure 4.8.

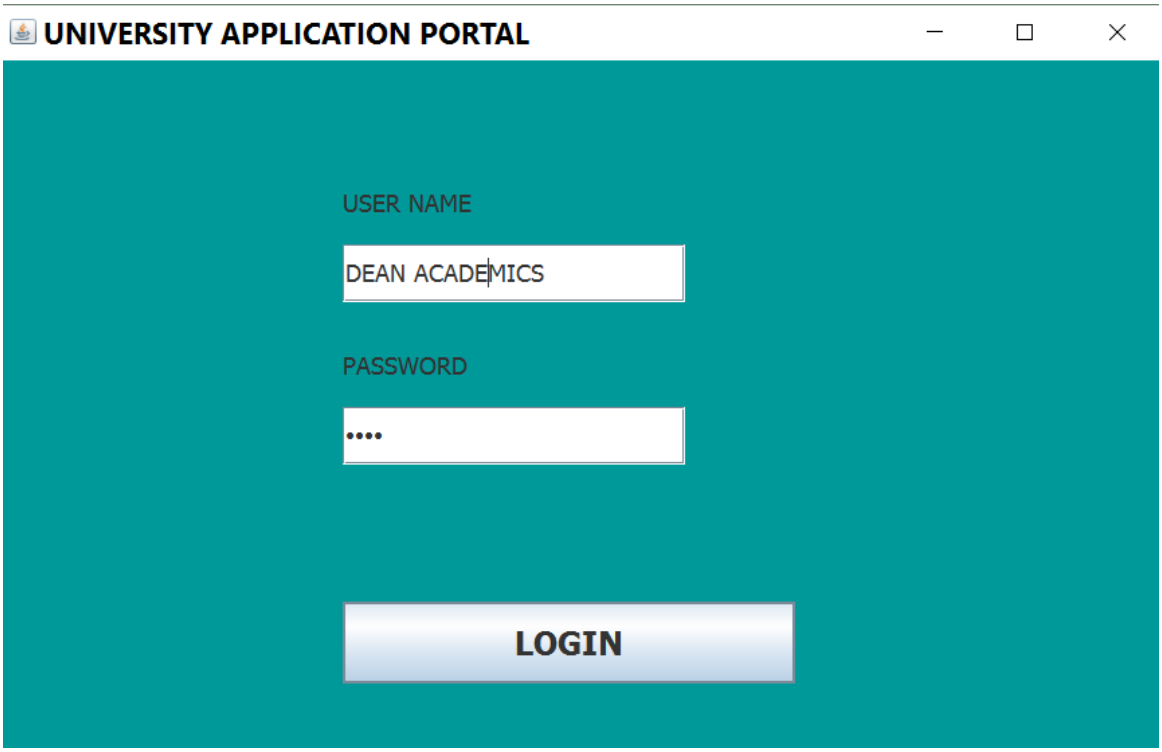


Figure 4.8 Login Frame

List of applications will be displayed along with their sender id and contact no. and click on its subject to download the application, click on accept to accept the application and reject to reject the application as shown in figure 4.9. Now simply, receiver can logout by clicking on logout button and also delete or clear its inbox.

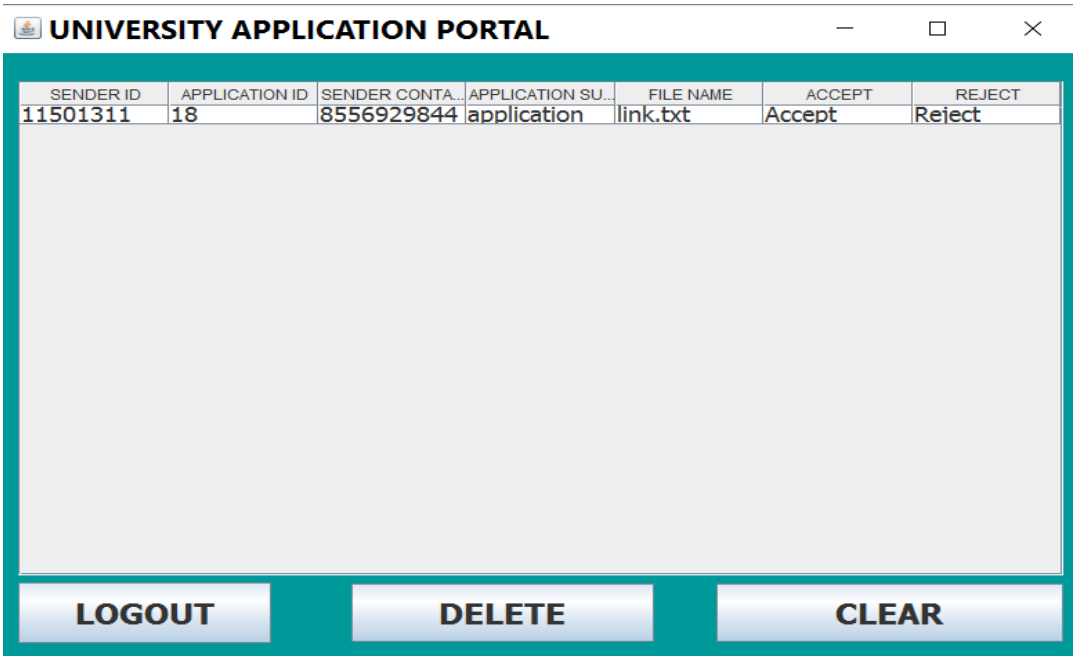


Figure 4.9 Inbox

4.1.3 Database

- Table 1 student_details have student data, who can send application.

Attributes

REDG_ID – have students' unique ID (Primary key)

STUDENT_NAME – have name of students

STUDENT_CONTACT_NO – have students contact no.

PASSWORD – have password for each unique id of student

REDG_ID	STUDENT_NAME	STUDENT_CONTACT_NO	PASSWORD
11501286	AMIT SHARMA	*NULL*	1234
11501287	ANKUSH BHANDARI	*NULL*	1234
11501289	ARJUN MAHAJAN	*NULL*	1234
11501290	BHUWAN SHARMA	*NULL*	1234
11501292	DIKSHA	*NULL*	1234
11501293	GUNTAS KAUR	*NULL*	1234
11501294	GURVEER SINGH	*NULL*	1234
11501295	GURWINDER SINGH	*NULL*	1234
11501302	MANPREET KAUR		1234
11501303	MANPREET SINGH	*NULL*	1234
11501304	MANSI	*NULL*	1234
11501305	MRIDU SOOD	*NULL*	1234
11501306	NAVLEEN KAUR	*NULL*	1234
11501307	NIKHIL SHARMA	*NULL*	1234
11501308	NILESH SHARMA	*NULL*	1234
11501309	NITALI SHARMA	*NULL*	1234
11501311	PALAK AGGARWAL	8556929844	1234
11501314	PARUL SHARMA	*NULL*	1234
11501316	POOJA	*NULL*	1234
11501317	PRANAV GUPTA	*NULL*	1234
11501328	SRISHTY MEHTA	*NULL*	1234
11501331	TWINKLE	*NULL*	1234
11501332	SANPREET		1234

Figure 4.10: Database Having Student Data

- Table 2 departmentdetails have details of all the departments.

  Attributes

  DEPARTMENTID – contain unique ID of each department.

  DEPARTMENTNAME – have name of each department.

DEPARTMENTID	DEPARTMENTNAME
1	COMPUTER SCIENCE
2	MECHANICAL
3	CIVIL
4	SPORTS
5	HIGHER AUTHORITIES
6	ELECRONICS
7	ENGLISH
8	COMMERCE
9	MATHEMATICS
10	AGRICULTURE
11	CHEMISTRY
12	NSS
13	DRAMA
14	REEJH
15	POETRY
16	BIOTECHNOLOGY
17	COMPUTER APPLICATIONS
18	YOGA
19	LIBRARY
20	HOSTEL
21	TRANSPORT
22	PHYSICS
23	ENVIRONMENT

Figure 4.11: Database Having Department Data

- Table 3 teachers have details of every teacher.

  Attributes

  TEACHERID – contain unique ID of every teacher

  TEACHERNAME – have teacher name

  PASSWORD – password for each teacherid

  TEACHERCONTACT – contains contact no. of teachers

DEPARTMENTID – foreign key from table department details have department ID for each teacher

Primary key – {TEACHERID, DEPARTMENTISD}

TEACHERID	TEACHERNAME	PASSWORD	TEACHERCONTACT	DEPARTMENTID
1	MR. RAKESH	1234	1234567889	1
2	MR. AVTAR	1234	1234567889	2
3	MR. ARIJIT	1234	1234567889	1
4	MR. MANJIT	1234	1234567889	1
5	MS. ARPITA	1234	1234567889	1
6	DEAN ACADEMICS	1234	1234567890	5
7	VC	1234	1234567890	5

Figure 4.12: Database Having Teachers Data

- Table 4 appdb have all the application details. Whole data in this table is fetched while execution

  Attributes

  Senderid – fetched while sender login

  Sendercontact – fetched while sender login

  Departmentid – fetched while sender compose application

  Teachername – fetched while sender composing the application

  Appsubject – fetched while sender compose application

  Status – while application is composed it will be pending, but will be changed to accepted/rejected as per the receiver's response to the application

  Counterc – it is used to clear data from senders sent box but data will remain in database. While data is displayed to the sender in the sent box it will be 0 otherwise it will be 1 if sender delete/clear the data.

  Counters – it is used to clear data from receiver side if application is accepted/rejected or if the inbox is cleared it will be set and data will no more be there while status is pending it will 0.

  Appid - unique ID assigned to each application composed.

  Fileupload – contains file attached.

senderid	sendercontact	departmentid	teachername	appsubject	STATUS	COUNTERC	COUNTERS	APPDBID	fileupload
11501302		1	MR. RAKESH	reappear	Accepted	0	1	1	
11501311	7986518069	5	DEAN ACADEMICS		Pending	1	1	16	
11501311	8556929844	1	MR. RAKESH	application for coducting a event	Accepted	0	1	17	The Dean Academics.pdf
11501311	8556929844	5	DEAN ACADEMICS	application	Accepted	0	0	18	link.txt
11501311		1	MR. RAKESH	czxcbbcj	Accepted	1	1	19	link.txt

Figure 4.13: Database Having Application Data

CHAPTER-5

CODING AND IMPLEMENTATION

5.1 MODULE I – SENDER SIDE

5.1.1 Welcome Frame Coding

```
1   package universityapplication;
2   public class Welcome extends javax.swing.JFrame {
3
4       public Welcome() {
5           initComponents();
6           getRootPane().setDefaultButton(b1);
7           setTitle("UNIVERSITY APPLICATION PORTAL");
8           setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
9       }
10
11      @SuppressWarnings("unchecked")
12      Generated Code
85      private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
86          this.setVisible(false);
87          new Data().setVisible(true);
88      }
89      public static void main(String args[])
90      {
91          java.awt.EventQueue.invokeLater(new Runnable()
92          {
93              public void run()
94              {
95                  new Welcome().setVisible(true);
96              }
97          });
98      }
99
100     // Variables declaration - do not modify
101     private javax.swing.JButton b1;
102     private javax.swing.JLabel l1;
103     private javax.swing.JLabel l2;
104     private javax.swing.JPanel p1;
```

Figure 5.1: Coding For Welcome Frame

A welcome frame will appear having a start button on it. This frame is not connected with any of the database but has been connected to the next frame. When you will click the send an application button, simply this frame will disappear and next frame will appear to you that will be for logging in to verify you really belong to the organization. If you're filled data will match to the student detail database then you will proceed further, otherwise you will not be able to log in or compose any of the application for sending.

5.1.2 Login Frame Coding

```
3   import com.mysql.jdbc.Connection;
4   import java.sql.*;
5   import javax.swing.JOptionPane;
6
7   public class Data extends javax.swing.JFrame
8       {
9           public Data()
10          {
11          initComponents();
            setTitle("UNIVERSITY APPLICATION PORTAL");
            getRootPane().setDefaultButton(b1);
            setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
15          String uid = t1.getText();
16          }
17      @SuppressWarnings("unchecked")
18      Generated Code
132
133     public Connection conn()
134     {
135         Connection con = null;
136         try
137         {
138            Class.forName("com.mysql.jdbc.Driver");
139             con = (Connection) DriverManager.getConnection("jdbc:mysql:///application","root","");
140         }
            catch(Exception e)
142         {
143             JOptionPane.showMessageDialog(rootPane , e);
144         }
145         return con;
```

Figure 5.2: Coding For Login Frame

```
152   private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
          Connection con = null;
          PreparedStatement ps = null;
          PreparedStatement ps1 = null;
          ResultSet rs = null;
157       String sid = t1.getText();
158       String password = pwd1.getText();
159       String no = ft1.getText();
160
161       try
162       {
163           con = conn();
164           String qry = "Select *from student_data where REDG_ID = '"+sid+"' and PASSWORD = '"+password+"' ";
165           String qry1 = "update Student_data set STUDENT_CONTACT_NO  = '"+no+"' where REDG_ID = '"+sid+"' and password =
166           ps=con.prepareStatement(qry);
167           ps1=con.prepareStatement(qry1);
168           ps1.execute();
169           rs= ps.executeQuery();
170           if (rs.next()) {
171                JOptionPane.showMessageDialog(rootPane, "login sucessful");
172                  this.setVisible(false);
173                  new RecieverDetail(sid,no).setVisible(true);
174           }
175           else
176           JOptionPane.showMessageDialog(rootPane, "INVALID USER NAME AND PASSWORD");
177       }
```

Figure 5.3: Coding For Login Frame

5.1.3 Composing Application Frame Coding

```
2  import java.io.File;
3  import java.nio.file.Files;
4  import java.nio.file.Path;
5  import java.sql.*;
6  import javax.swing.JFileChooser;
7  import javax.swing.JOptionPane;
8
9  public class RecieverDetail extends javax.swing.JFrame {
10
11     String id="",mob="";
12
13     public RecieverDetail(String a,String b ) {
14         initComponents();
           setTitle("UNIVERSITY APPLICATION PORTAL");
           getRootPane().setDefaultButton(b1);
           setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
18         id=a;
19         mob=b;
             Connection con = null;
             Statement  st = null;
22           PreparedStatement ps = null;
23           String rq1 = "SELECT  DEPARTMENTNAME FROM  DEPARTMENTDETAILS";
24
25           try
26           {
                 con = conn();
28               st = con.createStatement();
29               ResultSet rs1 = st.executeQuery(rq1);
30               while (rs1.next()) {
31               C1.addItem(rs1.getString("DEPARTMENTNAME"));
```

Figure 5.4: Coding For Composing an Application

```
       PreparedStatement ps  = null;
206    String appsub = t1.getText();
207    String status = "Pending";
208
209    try
210    {
211        con = conn();
212        String qry = "insert into appdb(senderid,sendercontact,departmentid,teachername,appsubject,fileupload,status,co
213        ps = con.prepareStatement(qry);
214        ps.setString(1,id);
215        ps.setString(2,mob);
216        ps.setString(3, deptId);
217        ps.setString(4,tec);
218        ps.setString(5,appsub);
219        ps.setString(6, selectedFile.getName());
220        ps.setString(7,status);
221        ps.setInt(8,0);
222        ps.setInt(9,0);
223        ps.execute();
224    }
       catch (Exception e)
226    {
227        JOptionPane.showMessageDialog(rootPane, e);
228    }
229     this.setVisible(false);
230     new Inbox(id,mob).setVisible(true);
```

Figure 5.5: Coding For Composing an Application

```
private void C1ItemStateChanged(java.awt.event.ItemEvent evt) {
    PreparedStatement ps=null;
    Connection con=null;
    ResultSet rs=null;
    String deptID="";
          String dept = (String)C1.getSelectedItem();
          String qry = "SELECT DEPARTMENTID FROM DEPARTMENTDETAILS WHERE DEPARTMENTNAME = '"+dept+"'";
          try{
         con=conn();
         ps = con.prepareStatement(qry);
         rs=  ps.executeQuery();
              if (rs.next()) {
               deptID=rs.getString("DEPARTMENTID");
              }
          }
          catch(Exception e)
          {
              System.err.println(e);
          }
          l3.setText(deptID);
          if (!deptID.equals("")) {
        try{
              PreparedStatement ps1=null;
                  Connection con1=null;
                  con1=conn();
                  C2.removeAllItems();
          String rq2 = "SELECT TEACHERNAME  FROM  TEACHERS where DEPARTMENTID="+deptID+" ";
          ps1=con1.prepareStatement(rq2);
          ResultSet rs2 = ps1.executeQuery(rq2);
          while (rs2.next())
          {
          C2.addItem(rs2.getString("TEACHERNAME"));
```

Figure 5.6: Coding For Composing an Application

```
279     private void b2ActionPerformed(java.awt.event.ActionEvent evt) {
280         this.setVisible(false);
281         new Inbox(id,mob).setVisible(true);
282         // TODO add your handling code here:
283     }
284 File selectedFile=null;
285     private void b3ActionPerformed(java.awt.event.ActionEvent evt) {
286           JFileChooser fileChooser = new JFileChooser();
287             int returnValue = fileChooser.showOpenDialog(null);
288             if (returnValue == JFileChooser.APPROVE_OPTION) {
289
290             try {
291                  selectedFile = fileChooser.getSelectedFile();
292                File dest =new File("E:\\java projects\\UniversityApplication\\"+selectedFile.getName());
293                 Path srcpath= selectedFile.toPath();
294                 Path srcpath1= dest.toPath();
295                 Files.copy(srcpath, srcpath1);
296                }
```

Figure 5.7: Coding For Composing an Application

5.1.4 Sent Box Frame Coding

```
2  import java.sql.Connection;
3  import java.sql.DriverManager;
4  import java.sql.PreparedStatement;
5  import java.sql.ResultSet;
6  import javax.swing.JOptionPane;
7  import javax.swing.table.DefaultTableModel;
8
9  public class Inbox extends javax.swing.JFrame {
10         static String id="",mob="",aid="",file="";
11     DefaultTableModel model = new DefaultTableModel();
12     public Inbox(String a,String b) {
13         initComponents();
14         id = a;
15         mob=b;
           this.setTitle("UNIVERSITY APPLICATION PORTAL");
           setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
           displayall();
19     }
20
21        public Connection conn()
22     {
23         Connection con = null;
24         try
25         {
26            Class.forName("com.mysql.jdbc.Driver");
27            con = (Connection) DriverManager.getConnection("jdbc:mysql:///DEPARTMENT","root","");
```

Figure 5.8: Coding for Sent Box

```
36       public void displayall()
37     {
38         String cols[]={"STATUS","APPLICATION ID","FACULTY NAME","APPLICATION SUBJECT"};
39         model.setColumnIdentifiers(cols);
40         tbl.setModel(model);
           Connection con=null;
           PreparedStatement ps=null;
           ResultSet rs=null;
44         try{
45         con = conn();
46         String qry="select STATUS, APPDBID, TEACHERNAME , APPSUBJECT from APPDB where senderid = '"+id+"' AND COUNTERC = 0
47         ps=con.prepareStatement(qry);
48         rs=ps.executeQuery();
49         while(rs.next())
50         {
51            Object name=rs.getString("TEACHERNAME");
52            Object appsub=rs.getString("APPSUBJECT");
53            Object status = rs.getString("Status");
              Object aid = rs.getInt("APPDBID");
55            Object data[]={status , aid , name , appsub};
56            model.addRow(data);
57            tbl.setModel(model);
58         }
```

Figure 5.9: Coding for Sent Box

```
191    private void b2ActionPerformed(java.awt.event.ActionEvent evt) {
192            this.setVisible(false);
193            new Welcome().setVisible(true);
194    }
195
196    private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
197        this.setVisible(false);
198        new RecieverDetail( id , mob).setVisible(true);
199    }
200
201    private void b3ActionPerformed(java.awt.event.ActionEvent evt) {
202         int row=tb1.getSelectedRow();
203         aid = model.getValueAt(row, 1).toString();
204         model.removeRow(row);
           Connection con=null;
           PreparedStatement ps=null;
207        ResultSet rs=null;
208
209        try{
210        con = conn();
211        //qry to delete 1 row from table
212        String qry="update appdb set counterc =1 where senderid = '"+id+"'and appdbid = '"+aid+"'";
213        ps=con.prepareStatement(qry);
214        ps.executeUpdate();
215        }
           catch(Exception e)
217        {         }
218    }
219
220    private void b4ActionPerformed(java.awt.event.ActionEvent evt) {
```

Figure 5.10: Coding for Sent Box

```
225    try{
226         con = conn();
227        //qry to delete all content from table
228        String qry="update appdb set counterc = 1 where senderid = '"+id+"' ";
229        ps=con.prepareStatement(qry);
230        ps.executeUpdate();
       }catch(Exception e)
232    {
233    }
234        model.setRowCount(0);
235        tb1.setModel(model);
236    }
```

Figure 5.11: Coding for Sent Box

5.2 MODULE II – RECEIVER SIDE

5.2.1 Welcome Frame Coding

```
1   package universityapplication;
2   public class Welcome extends javax.swing.JFrame {
3
4       public Welcome() {
5           initComponents();
6           getRootPane().setDefaultButton(b1);
7           setTitle("UNIVERSITY APPLICATION PORTAL");
8           setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
9       }
10
11      @SuppressWarnings("unchecked")
12      Generated Code
85      private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
86          this.setVisible(false);
87          new Data().setVisible(true);
88      }
89      public static void main(String args[])
90      {
            java.awt.EventQueue.invokeLater(new Runnable()
92          {
                public void run()
94              {
95                  new Welcome().setVisible(true);
```

Figure 5.12: Coding for Welcome frame

5.2.2 Login Frame Coding

```
3   import com.mysql.jdbc.Connection;
4   import java.sql.DriverManager;
5   import java.sql.PreparedStatement;
6   import java.sql.ResultSet;
7   import javax.swing.JOptionPane;
8
9   public class Login extends javax.swing.JFrame {
10
11      public Login() {
12          initComponents();
13          getRootPane().setDefaultButton(b1);
14          setTitle("UNIVERSITY APPLICATION PORTAL");
15          setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
16      }
17
18          public Connection conn()
19      {
20          Connection con = null;
21          try
22          {
23             Class.forName("com.mysql.jdbc.Driver");
24             con = (Connection) DriverManager.getConnection("jdbc:mysql:///DEPARTMENT","root","");
25          }
```

Figure 5.13: Coding for Login Frame

```
110    private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
           Connection con = null;
           PreparedStatement ps = null;
113        PreparedStatement ps1 = null;
           ResultSet rs = null;
115        String sid = t1.getText();
116        String password = pwd1.getText();
117        try
118        {
119            con = conn();
120            String qry = "Select TEACHERNAME , PASSWORD from TEACHERS where TEACHERNAME = '"+sid+"' and PASSWORD = '"+pass
121            ps=con.prepareStatement(qry);
122            rs= ps.executeQuery();
123            if (rs.next()) {
124                JOptionPane.showMessageDialog(rootPane, "LOGIN SUCCESSFUL");
125                    this.setVisible(false);
126                    new DataBase(sid).setVisible(true);
127            }
128            else
129            JOptionPane.showMessageDialog(rootPane, "INVALID USER NAME AND PASSWORD");
130        }
```

Figure 5.14: Coding for Login Frame

5.2.3 Inbox Frame Coding

```
 2  import java.io.File;
 3  import java.nio.file.Files;
 4  import java.nio.file.Path;
 5  import java.sql.Connection;
 6  import java.sql.DriverManager;
 7  import java.sql.PreparedStatement;
 8  import java.sql.ResultSet;
 9  import javax.swing.JFileChooser;
10  import javax.swing.JOptionPane;
11  import javax.swing.table.DefaultTableModel;
12
13  public class DataBase extends javax.swing.JFrame {
14      DefaultTableModel model = new DefaultTableModel();
15      String tn = "",AID="";
16      public DataBase(String a) {
17          initComponents();
18          tn = a;
            setTitle("UNIVERSITY APPLICATION PORTAL");
20          model=(DefaultTableModel)tb1.getModel();
            setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
            displayall();
23      }
24
25      @SuppressWarnings("unchecked")
26      Generated Code
127
128
129         public Connection conn()
130     {
```

Figure 5.15: Coding for Inbox Frame

```
141         return con;
142     }
143
144
145     public void displayall()
146 {
147         String cols[]={"SENDER ID","APPLICATION ID","SENDER CONTACT","APPLICATION SUBJECT","FILE NAME","ACCEPT","REJECT"},
148         model.setColumnIdentifiers(cols);
149         tb1.setModel(model);
150
151
    Connection con=null;
    PreparedStatement ps=null;
    ResultSet rs=null;
155
156     try{
157         con = conn();
158         String qry="select SENDERID, APPDBID, SENDERCONTACT , APPSUBJECT , FILEUPLOAD from APPDB where TEACHERNAME = '"+tn
159         ps=con.prepareStatement(qry);
160         rs=ps.executeQuery();
161
162         while(rs.next())
163         {
164             Object ID=rs.getString("SENDERID");
            Object appsub=rs.getString("APPSUBJECT");
```

Figure 5.16: Coding for Inbox Frame

```
166                 Object MOB = rs.getString("SENDERCONTACT");
                    Object AID=rs.getString("APPDBID");
168                 Object fd=rs.getString("FILEUPLOAD");
169                 Object data[]={ID , AID , MOB , appsub, fd , "Accept","Reject"};
170                 model.addRow(data);
171                 tb1.setModel(model);
172             }
173
174             }
        catch(Exception e)
176     {
177         JOptionPane.showMessageDialog(rootPane, e);
178         System.out.println("display");
179     }
180 }
181
182     private void b3ActionPerformed(java.awt.event.ActionEvent evt) {
        Connection con=null;
        PreparedStatement ps=null;
185     ResultSet rs=null;
186
187     try{
188
189         con = conn();
190         //qry to delete all content from table
191         String qry="update appdb set counters =1 where TEACHERNAME  = '"+tn+"' ";
192         ps=con.prepareStatement(qry);
193         ps.executeUpdate();
        }catch(Exception e)
195     {}
196
197         model.setRowCount(0);
```

Figure 5.17: Coding for Inbox Frame

```
206     private void b2ActionPerformed(java.awt.event.ActionEvent evt) {
207     int row=tb1.getSelectedRow();
208     AID = model.getValueAt(row, 1).toString();
209     model.removeRow(row);
        Connection con=null;
        PreparedStatement ps=null;
212
213     try{
214         con = conn();
215         //qry to delete 1 row from table
216         String qry="update appdb set counters=1 where teachername = '"+tn+"' AND APPDBID = '"+AID+"' ";
217         ps=con.prepareStatement(qry);
218         ps.executeUpdate();
219         }
        catch(Exception e)
221         {     }
222     }
223 File selectedFile=null;
224     private void tb1MouseClicked(java.awt.event.MouseEvent evt) {
225         int row;
226         row = tb1.getSelectedRow();
227         int col=tb1.getSelectedColumn();
            String id="",appid="";
            String sts="";
```

Figure 5.18: Coding for Inbox Frame

```
230         if (col==5) {
231             sts="Accepted";
232             id=(String)tb1.getValueAt(row, 0);
233             appid=(String)tb1.getValueAt(row, 1);
234              System.out.println(id+" "+appid);
                Connection con=null;
                PreparedStatement ps=null;
237             try{
238                 con = conn();
239                 //qry to delete 1 row from table
240                 String qry="update appdb set status='"+sts+"',counters = 1 where senderid = '"+id+"' AND APPDBID = '"+appi
241                 ps=con.prepareStatement(qry);
242                 ps.executeUpdate();
243                 JOptionPane.showMessageDialog(rootPane, "done");
244                 model.removeRow(row);
245                 }
                catch(Exception e)
247                 {
248                     JOptionPane.showMessageDialog(null,e);
249                 }
250         }
251         if (col==6) {
252             sts="Rejected";
253             id=(String)tb1.getValueAt(row, 0);
254             appid=(String)tb1.getValueAt(row, 1);
255              System.out.println(id+" "+appid);
                Connection con=null;
                PreparedStatement ps=null;
258             try{
259                 con = conn();
260                 //qry to delete 1 row from table
                    String qry="update appdb set status='"+sts+"' , counters=1 where senderid = '"+id+"' AND APPDBID = '"+appi
```

Figure 5.19: Coding for Inbox Frame

```
274             String fname=(String)tb1.getValueAt(row, 4);
275             JFileChooser fileChooser = new JFileChooser();
276             int returnValue = fileChooser.showSaveDialog(null);
277             if (returnValue == JFileChooser.APPROVE_OPTION) {
278             try {
279                selectedFile = fileChooser.getCurrentDirectory();
280                File dest =new File("E:\\java projects\\UniversityApplication\\"+fname);
281                System.out.println(dest.toPath());
282                String pp = selectedFile.toPath() +"\\"+ dest.getName();
283                File ff= new File(pp);
284                Path srcpath= ff.toPath();
285                System.out.println(selectedFile.toPath());
286                Path srcpath1= dest.toPath();
287                System.out.println(dest.toPath());
288                Files.copy(srcpath1, srcpath);
289             }
```

Figure 5.20: Coding for Inbox Frame

The entire frames are using Java database connectivity except welcome frame.

CHAPTER – 6
CONCLUSION

At last, I conclude that this project will be very beneficial if implemented. It will save lot of our time and efforts for going and submitting applications to the higher authorities. But at the same time I consider this project as an "**Incremental Software Process Model",** this is the basic version of the "**Digital letter box"**, in future I would be coming up with more enhanced features of this project and will try to make it Android based.

I strongly recommend that this project should be implemented in our university, so that my efforts would really get successful.

REFERENCES

www.javatpoint.com

www.tutorialspoint.com/java

www.geeksforgeeks.org/java/

www.oracle.com/java/

www.techopedia.com/

www.stackoverflow.com/documentation/java/topics