UNIVERSITY OF LIBERAL ARTS BANGLADESH

Department of Computer Science and Engineering

# Open  Ended Lab Report - 02

**Submitted To:**

FERDOUS BIN HAFIZ
Deptertment of CSE
University of Liberal Arts Bangladesh

**Submitted By:**

Name:  AHADUL ISLAM RAHUL
ID:  223014152
Course Name:  Object Oriented Programming Lab
Course ID:  CSE 2104
Section:  02

**DATE: 18-12-2023**

## ➢ TITLE:

Job Management System **“BEKARerJOB”.**

## ➢ INTRODUCTION:

**“BEKARerJOB”** is a Java-based Job Management System. It has a graphical user interface based on GUI swing in java. This Job Management System allows you to manage your job posts and job seekers profiles. You can easily add, update and delete records. The system include more features like file I/O for data persistence and separates classes for job posts and seekers to enhance modularity. The application serves as the basis for a simple and easy-to-use job recruitment tool.

## ➢ PROBLEM UNDERSTANDING:

**“BEKARerJOB”** is a Java Swing based Job Management System that provides a graphical user interface for a Job Management System. It allows users to create job posts and seeker profiles. Key features include adding, updating, and deleting records, displayed in tables. The system supports file I/O for data persistence.

## ➢ BACKGROUND THEORY:

The provided Java code implements a Job Management System, primarily focusing on the graphical user interface （GUI） for managing job posts and seeker profiles.

**1. Graphical User Interface (GUI):**

Java Swing: Java Swing is a set of GUI components for building desktop applications in Java.

**2. Object-Oriented Programming (OOP):**

The code follows object-oriented principles by defining classes for “JobPost” and “Seeker”. This promotes code organization, encapsulation, and code reuse.

**3. Event-Driven Programming:**

GUI applications often follow an event-driven paradigm where user interactions (“button clicks”) trigger events. The code handles events using action listeners (“jButton1ActionPerformed”), responding to user actions.

**4. File Input/Output (I/O):**

The code demonstrates file I/O operations, allowing the system to save job post and seeker data to text files (“txtfile.txt” and “seeker.txt”). This provides a basic form of data persistence.

**5. Swing Components:**

The code utilizes various Swing components, such as “JTable”, “JComboBox”, “JTextField”, “JRadioButton”, and “JDateChooser”, to create a user-friendly interface for data input and display.

**6. Exception Handling:**

While not extensively implemented in the provided code, exception handling is a crucial aspect of robust programming. Proper handling of exceptions ensures that the application can gracefully manage errors or unexpected situations.

**7. Separation of Concerns:**

The code attempts to separate concerns by having distinct classes for job posts (“JobPost”) and seekers (“Seeker”). This separation enhances code modularity and readability.

**8. Data Persistence:**

The code uses file I/O to save and load data. Data persistence is vital for preserving information between different program runs.

## ➢ ALGORITHM DESIGN:

**1. Initialization:**

- Create an item of **UiUx**.
- Create an item of **UiPanel**.
- Set the visibility of **UiPanel** to true.

**2. JobPost Class:**

- Define a class **JobPost** to represent job posts.
- Include attributes such as ***cm, jm, jt, gen, sal, age, edu, skl.***
- Implement a constructor to initialize these attributes.

**3. Seeker Class:**

- Define a class **Seeker** to represent job seekers.
- Include attributes such as ***nm, dis, gn, dob, ed, sk, mn.***
- Implement a constructor to initialize these attributes.

**4. UiPanel Class:**

- Initialize **ArrayLists** for job posts (**t1**) and seekers (**t2**).
- Initialize *DefaultTableModel* for job posts (**gjp**) and seekers (**ajs**).
- Implement **GUI** components for job posts and seekers (text fields, combo boxes, etc.).
- Handle button events for ***adding, updating, and deleting*** records.

- Implement file I/O methods to save and load job posts and seekers.

**5. File Operations Class:**

- Define a separate class for file I/O operations (***FileOperations***).
- Include methods to save job posts to ***"txtfile.txt"*** and load from it.
- Include methods to save seekers to ***"seeker.txt"*** and load from it.

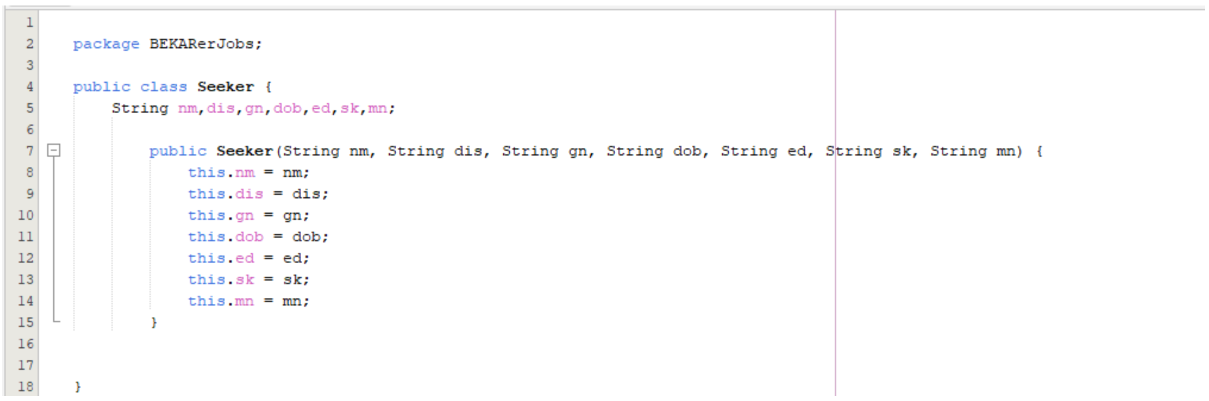
**6. Main Execution:**

- In the main method of **UiUx** class:
- Create an item of **UiUx.**
- Create an item of **UiPanel.**
- Set the **UiPanel** to be visible.

## ➢ MODERN TOOLS:

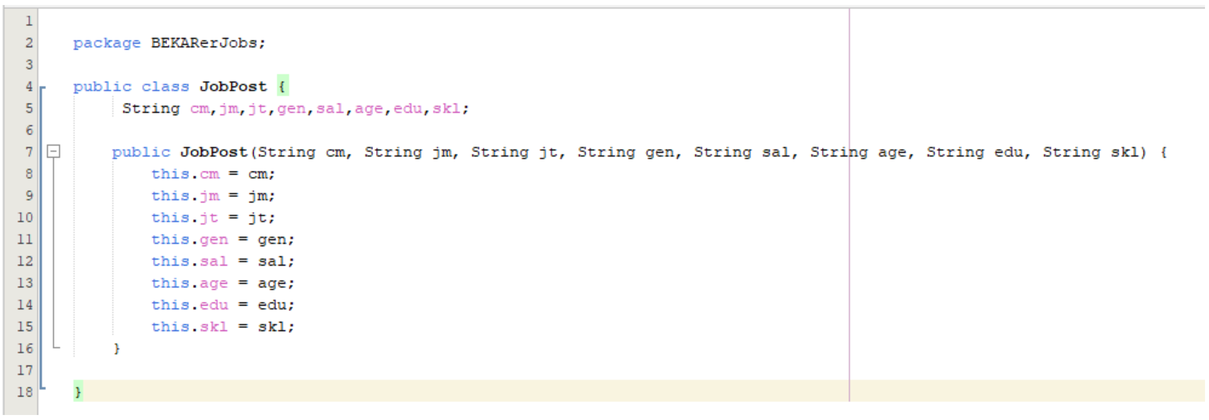
i. **Software:** Apache NetBeans 19
ii. **Compiler:** Apache NetBeans 19 IDE

## ➢ CODE:



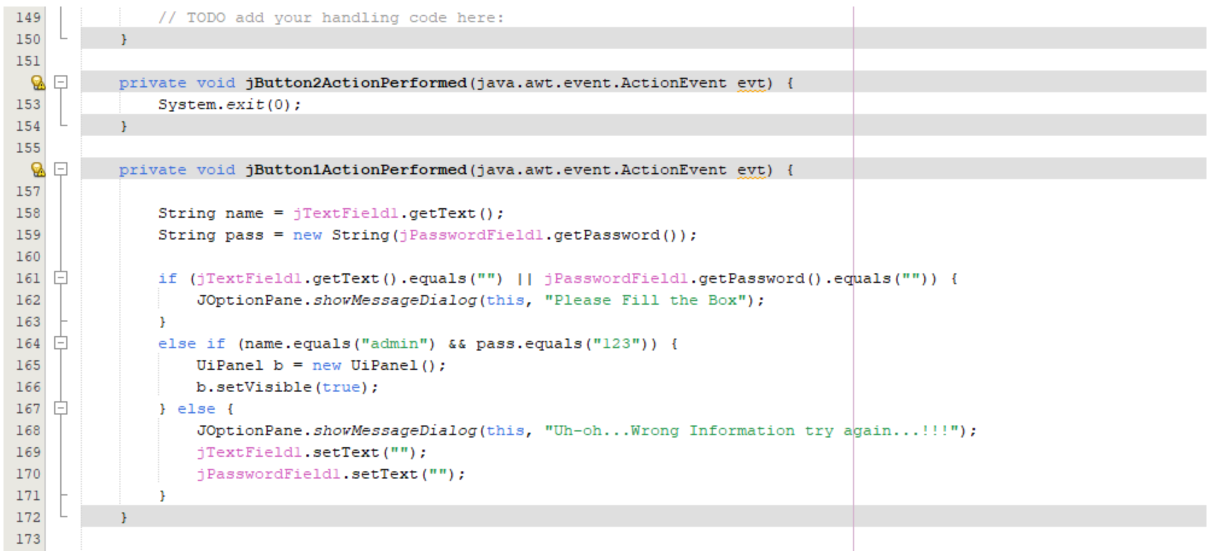
```java

package BEKARerJobs;

public class Seeker {
    String nm,dis,gn,dob,ed,sk,mn;

        public Seeker(String nm, String dis, String gn, String dob, String ed, String sk, String mn) {
            this.nm = nm;
            this.dis = dis;
            this.gn = gn;
            this.dob = dob;
            this.ed = ed;
            this.sk = sk;
            this.mn = mn;
        }


}
```

Figure 1: Seeker Class with properties



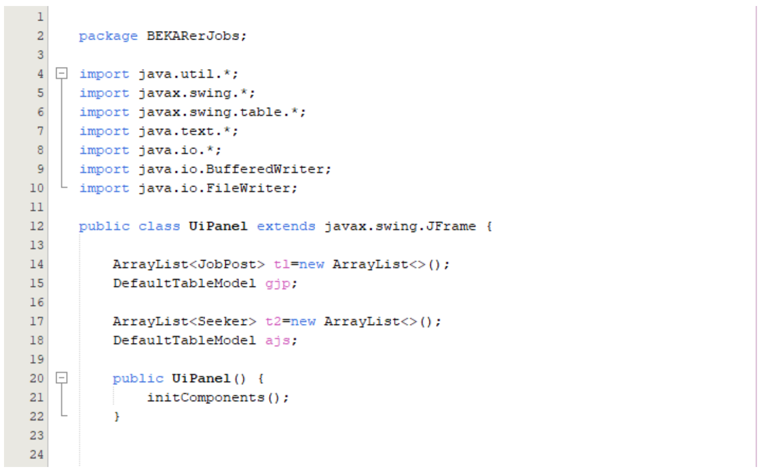
```java

package BEKARerJobs;

public class JobPost {
     String cm,jm,jt,gen,sal,age,edu,skl;

    public JobPost(String cm, String jm, String jt, String gen, String sal, String age, String edu, String skl) {
        this.cm = cm;
        this.jm = jm;
        this.jt = jt;
        this.gen = gen;
        this.sal = sal;
        this.age = age;
        this.edu = edu;
        this.skl = skl;
    }

}
```

Figure 2: JobPost Class with properties



```java
        // TODO add your handling code here:
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        String name = jTextField1.getText();
        String pass = new String(jPasswordField1.getPassword());

        if (jTextField1.getText().equals("") || jPasswordField1.getPassword().equals("")) {
            JOptionPane.showMessageDialog(this, "Please Fill the Box");
        }
        else if (name.equals("admin") && pass.equals("123")) {
            UiPanel b = new UiPanel();
            b.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(this, "Uh-oh...Wrong Information try again...!!!");
            jTextField1.setText("");
            jPasswordField1.setText("");
        }
    }

```

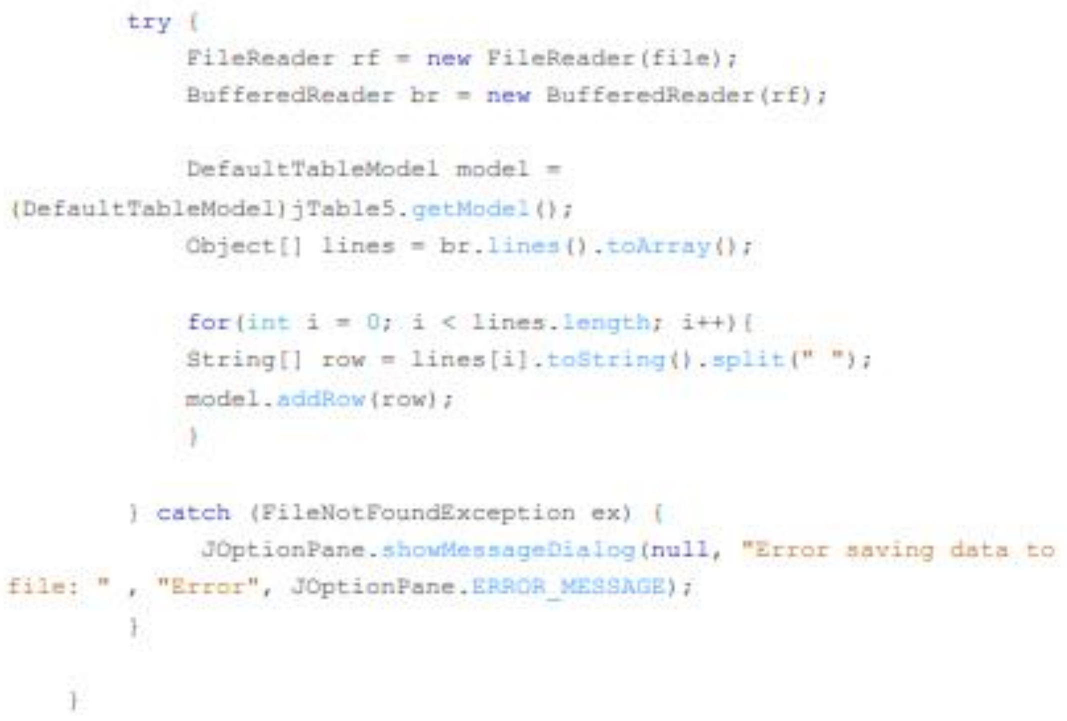
Figure 3: UiUx JFrame Form with properties



```java

package BEKARerJobs;

import java.util.*;
import javax.swing.*;
import javax.swing.table.*;
import java.text.*;
import java.io.*;
import java.io.BufferedWriter;
import java.io.FileWriter;

public class UiPanel extends javax.swing.JFrame {

    ArrayList<JobPost> t1=new ArrayList<>();
    DefaultTableModel gjp;

    ArrayList<Seeker> t2=new ArrayList<>();
    DefaultTableModel ajs;

    public UiPanel() {
        initComponents();
    }


```

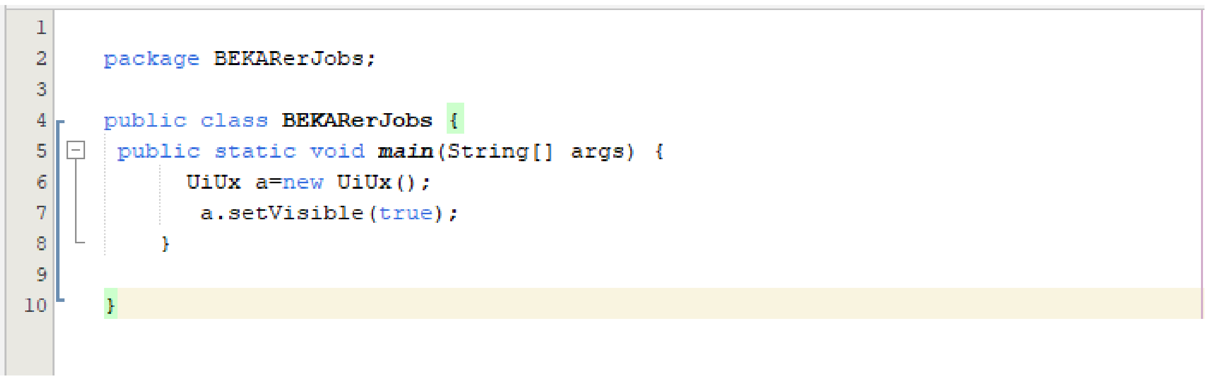
```java
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    gjp=(DefaultTableModel)jTable1.getModel();
    if(jTable1.getSelectedRowCount()==1){
    int i=jTable1.getSelectedRow();
    int p = JOptionPane.showConfirmDialog(this, "Do you really want to
delete?","Delete",JOptionPane.YES_NO_OPTION,3);
    if(p==0){
    gjp.removeRow(i);
    tl.remove(i);
}
    else
    JOptionPane.showMessageDialog(this, "Thanks");
    }
    else
    JOptionPane.showMessageDialog(this,"Select  Row");
    }
    private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
  try{
    int i=jTable1.getSelectedRow();
        String a=gjp.getValueAt(i,0).toString();
        String b=gjp.getValueAt(i,1).toString();
        String c=gjp.getValueAt(i,2).toString();
        String d=gjp.getValueAt(i,3).toString();
        String e=gjp.getValueAt(i,4).toString();
        String f=gjp.getValueAt(i,5).toString();
        String g=gjp.getValueAt(i,6).toString();
        String h=gjp.getValueAt(i,7).toString();

   jTextField1.setText(a);
   jTextField2.setText(b);
   jComboBox1.setSelectedItem(c);
   jComboBox2.setSelectedItem(d);
   jComboBox3.setSelectedItem(e);
  jComboBox4.setSelectedItem(f);
  jComboBox5.setSelectedItem(g);

 if(h.equalsIgnoreCase("Male"))
     jRadioButton1.setSelected(true);
 else
    jRadioButton2.setSelected(true);
}
catch(Exception e){
    }
   }

   private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

       gjp=(DefaultTableModel)jTable2.getModel();
           String nm=jTextField3.getText();
           String mn = jTextField4.getText();
           String dis=jComboBox6.getSelectedItem().toString();
           String ed = jComboBox7.getSelectedItem().toString();
           String sk = jComboBox8.getSelectedItem().toString();
           String gn = null, dob;

       if(jRadioButton1.isSelected())
           gn="Male";
       if(jRadioButton2.isSelected())
           gn="Female";

dob=((JTextField)jDateChooser1.getDateEditor().getUiComponent()).getText();
           int j=jTable2.getSelectedRow();

           if (j != -1 && j < t2.size()){
               t2.get(j).nm=nm;
               t2.get(j).dis=dis;
               t2.get(j).gn=gn;
               t2.get(j).dob=dob;
               t2.get(j).ed=ed;
               t2.get(j).sk=sk;
               t2.get(j).mn=mn;

               gjp.setValueAt(nm, j,0);
               gjp.setValueAt(dis, j, 1);
               gjp.setValueAt(gn, j, 2);
               gjp.setValueAt(dob, j, 3);
               gjp.setValueAt(ed, j, 4);
               gjp.setValueAt(sk, j, 5);
```

```
                gjp.setValueAt(mn, j, 6);

                }
            else{
                JOptionPane.showMessageDialog(this, "Please select a valid
row");
                }

    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

String nm = jTextField3.getText();
String mn = jTextField4.getText();
String dis = jComboBox6.getSelectedItem().toString();
String ed = jComboBox7.getSelectedItem().toString();
String sk = jComboBox8.getSelectedItem().toString();
String gn = null, dob;

if (jTextField3.getText().equals("") || mn.equals("") ||
jComboBox6.getSelectedIndex() == 0 || jComboBox7.getSelectedIndex() == 0 ||
jComboBox8.getSelectedIndex() == 0)
    JOptionPane.showMessageDialog(this, "Please Fill the Box");
 else {
    if (jRadioButton3.isSelected()) {
        gn = "Male";
    }
    if (jRadioButton4.isSelected()) {
        gn = "Female";
    }

    dob = ((JTextField)
jDateChooser1.getDateEditor().getUiComponent()).getText();

    t2.add(new Seeker(nm, dis, gn, dob, ed, sk, mn));
    ajs = (DefaultTableModel) jTable2.getModel();
    ajs.setRowCount(0);
    Object s[] = new Object[7];
    for (Seeker y : t2) {
        s[0] = y.nm;
        s[1] = y.dis;
        s[2] = y.gn;
        s[3] = y.dob;
        s[4] = y.ed;
        s[5] = y.sk;
        s[6] = y.mn;

        ajs.addRow(s);
    }

    jTextField3.setText("");
    jTextField4.setText("");
    jComboBox6.setSelectedIndex(0);
    jComboBox7.setSelectedIndex(0);
    jComboBox8.setSelectedIndex(0);
    buttonGroup2.clearSelection();
    jDateChooser1.setCalendar(null);
}


    }

    private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

        ajs=(DefaultTableModel)jTable2.getModel();
            if(jTable2.getSelectedRowCount()==1){
        int i=jTable2.getSelectedRow();
        int q = JOptionPane.showConfirmDialog(this, "Do you really want to
delete?","Delete",JOptionPane.YES_NO_OPTION,3);
            if(q==0){
            ajs.removeRow(i);
            t2.remove(i);
            }
            else
            JOptionPane.showMessageDialog(this, "Thanks");
            }
            else
            JOptionPane.showMessageDialog(this,"Select Row");
            }
            private void jTable2MouseClicked(java.awt.event.MouseEvent evt)
{
        try{
            int i=jTable2.getSelectedRow();
                String a=ajs.getValueAt(i,0).toString();
                String b=ajs.getValueAt(i,1).toString();
                String c=ajs.getValueAt(i,2).toString();
                String d=ajs.getValueAt(i,3).toString();
                String e=ajs.getValueAt(i,4).toString();
                String f=ajs.getValueAt(i,5).toString();
                String g=ajs.getValueAt(i,6).toString();
            jTextField3.setText(a);
            jComboBox5.setSelectedItem(b);
            jComboBox7.setSelectedItem(e);
            jComboBox8.setSelectedItem(f);
            jTextField4.setText(g);
                if(c.equalsIgnoreCase("Male"))
                jRadioButton3.setSelected(true);
                else
                jRadioButton4.setSelected(true);
            Date w=new SimpleDateFormat("MMM d,
y").parse((String)e.toString());
            jDateChooser1.setDate(w);
            }
        catch(Exception e){
        }
    }

    private void jRadioButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
    }

    private void jComboBox6ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
       String filePath = "txtfile.txt";
       File file = new File(filePath);

       try {
    FileWriter fw = new FileWriter(file);
    BufferedWriter bw = new BufferedWriter(fw);

        for(int i = 0; i < jTable1.getRowCount(); i++){
        for(int j = 0; j < jTable1.getColumnCount(); j++){
        bw.write(jTable1.getValueAt(i, j).toString()+" ");
            }
        bw.newLine();
            }
        bw.close();
        fw.close();

            JOptionPane.showMessageDialog(null, "Data Saved successfully!!
");

        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null, "Error saving data to file:
" , "Error", JOptionPane.ERROR_MESSAGE);

        }


    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
       String filePath = "seeker.txt";
       File file = new File(filePath);

       try {
    FileWriter wb = new FileWriter(file);
    BufferedWriter bw = new BufferedWriter(wb);

        for(int i = 0; i < jTable2.getRowCount(); i++){
        for(int j = 0; j < jTable2.getColumnCount(); j++){
        bw.write(jTable2.getValueAt(i, j).toString()+" ");
                }
        bw.newLine();
            }

        bw.close();
        wb.close();

            JOptionPane.showMessageDialog(null, "Data Saved successfully!!
");

        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null, "Error saving data to file:
" , "Error", JOptionPane.ERROR_MESSAGE);

        }


    }
    private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {
        String filePath = "txtfile.txt";
        File file = new File(filePath);

        try {
            FileReader fr = new FileReader(file);
            BufferedReader br = new BufferedReader(fr);

            DefaultTableModel model =
(DefaultTableModel)jTable3.getModel();
            Object[] lines = br.lines().toArray();

            for(int i = 0; i < lines.length; i++){
            String[] row = lines[i].toString().split(" ");
            model.addRow(row);
            }

        } catch (FileNotFoundException ex) {
             JOptionPane.showMessageDialog(null, "Error saving data to
file: " , "Error", JOptionPane.ERROR_MESSAGE);
        }

    }

    private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {

       UiUx a=new UiUx();
       a.setVisible(true);

    }

    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {
        String filePath = "seeker.txt";
        File file = new File(filePath);
```



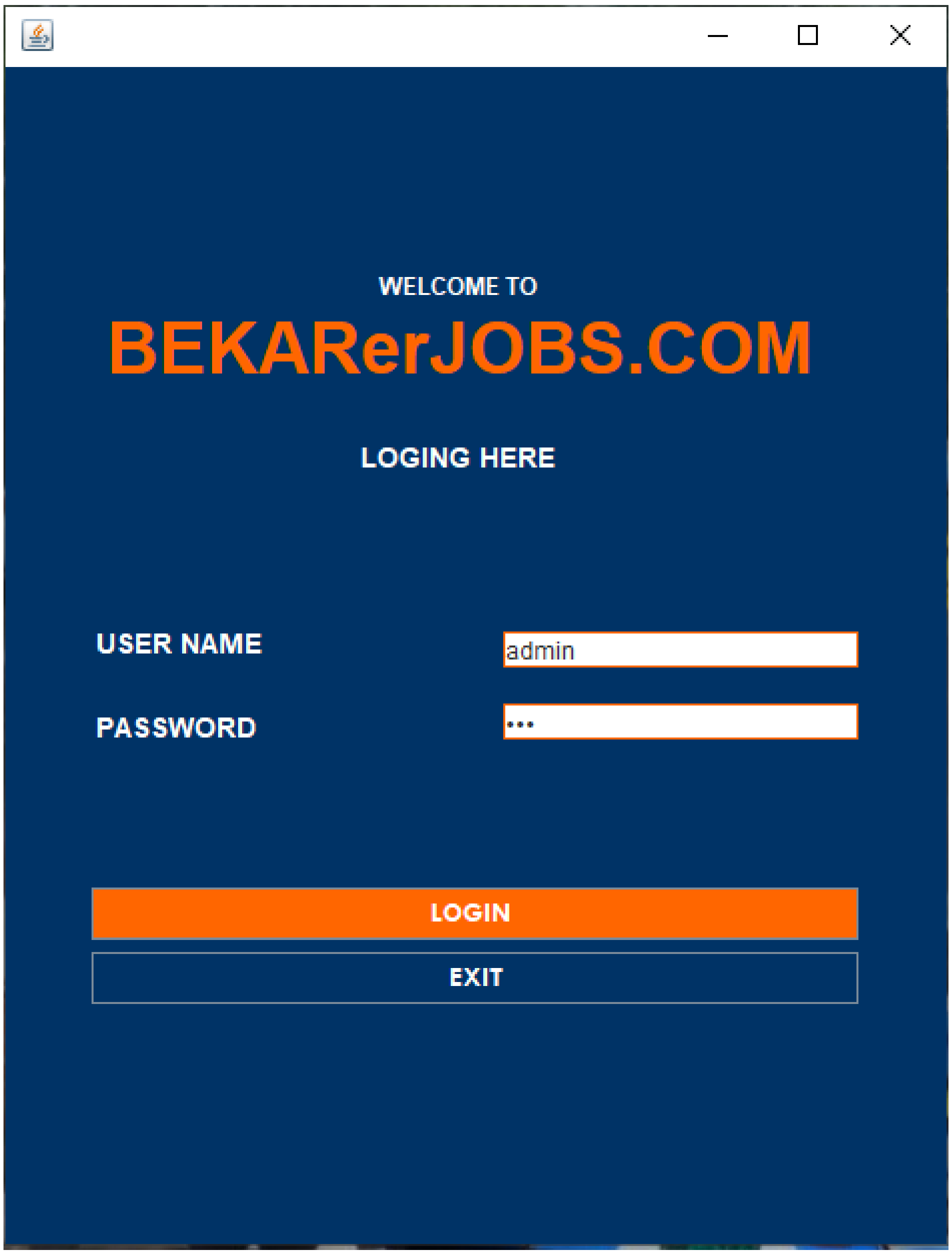
```
        try {
            FileReader rf = new FileReader(file);
            BufferedReader br = new BufferedReader(rf);

            DefaultTableModel model =
(DefaultTableModel)jTable5.getModel();
            Object[] lines = br.lines().toArray();

            for(int i = 0; i < lines.length; i++){
            String[] row = lines[i].toString().split(" ");
            model.addRow(row);
            }

        } catch (FileNotFoundException ex) {
             JOptionPane.showMessageDialog(null, "Error saving data to
file: " , "Error", JOptionPane.ERROR_MESSAGE);
        }

    }
```

**Figure 4: UiPanel JFrame Form with properties**



```

package BEKARerJobs;

public class BEKARerJobs {
 public static void main(String[] args) {
      UiUx a=new UiUx();
       a.setVisible(true);
    }

}
```

**Figure 5: Main method called *"BEKARerJobs"***



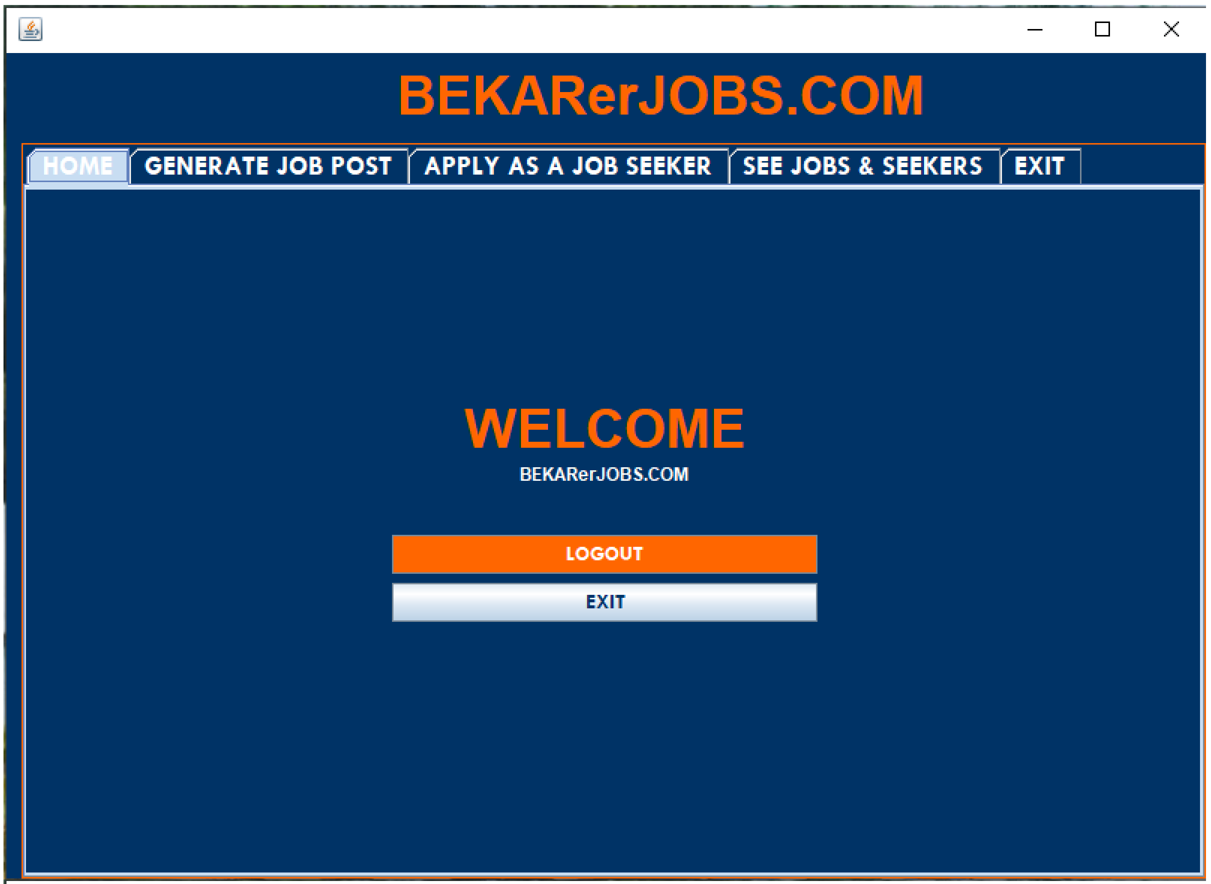
WELCOME TO

BEKARerJOBS.COM

LOGING HERE

USER NAME admin

PASSWORD •••


BEKARerJOBS.COM

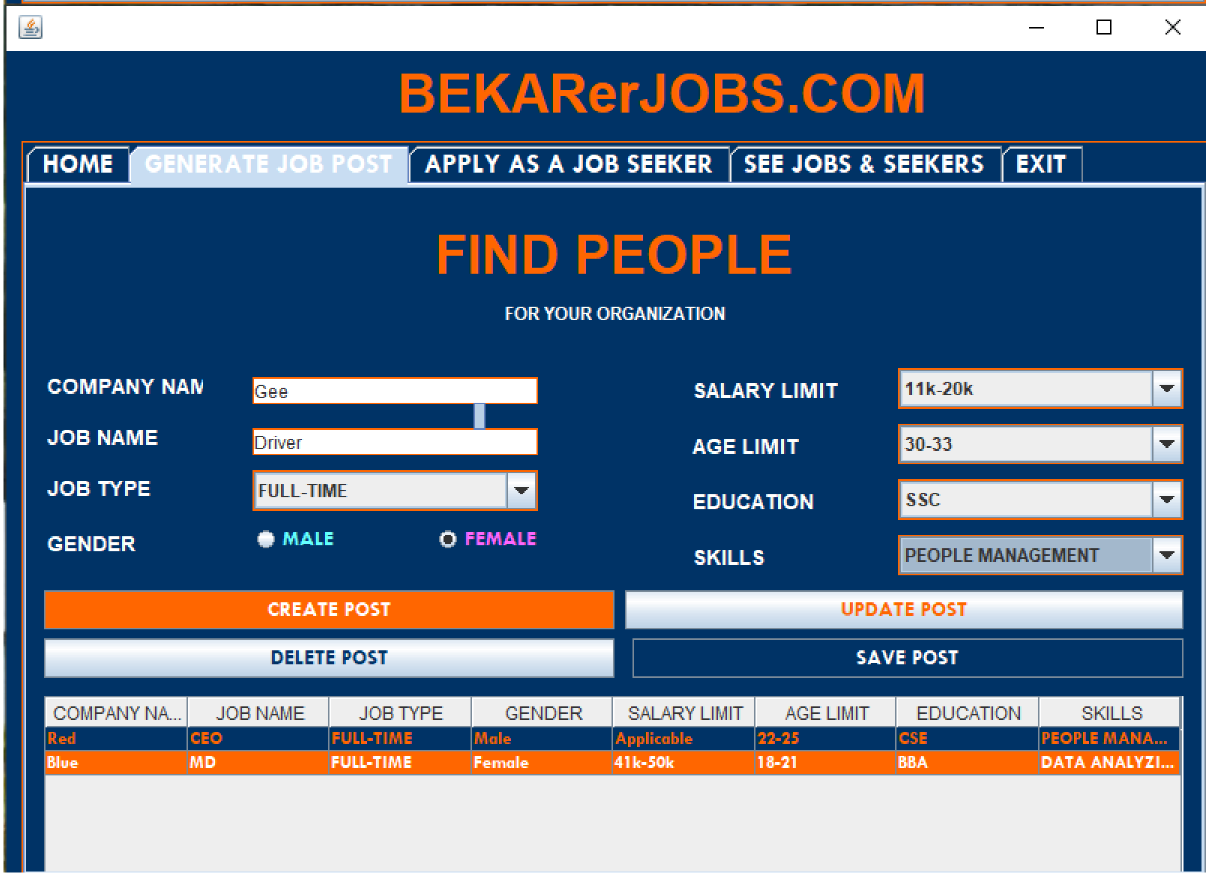
HOME | GENERATE JOB POST | APPLY AS A JOB SEEKER | SEE JOBS & SEEKERS | EXIT

WELCOME

BEKARerJOBS.COM

LOGOUT

EXIT


BEKARerJOBS.COM

HOME | GENERATE JOB POST | APPLY AS A JOB SEEKER | SEE JOBS & SEEKERS | EXIT

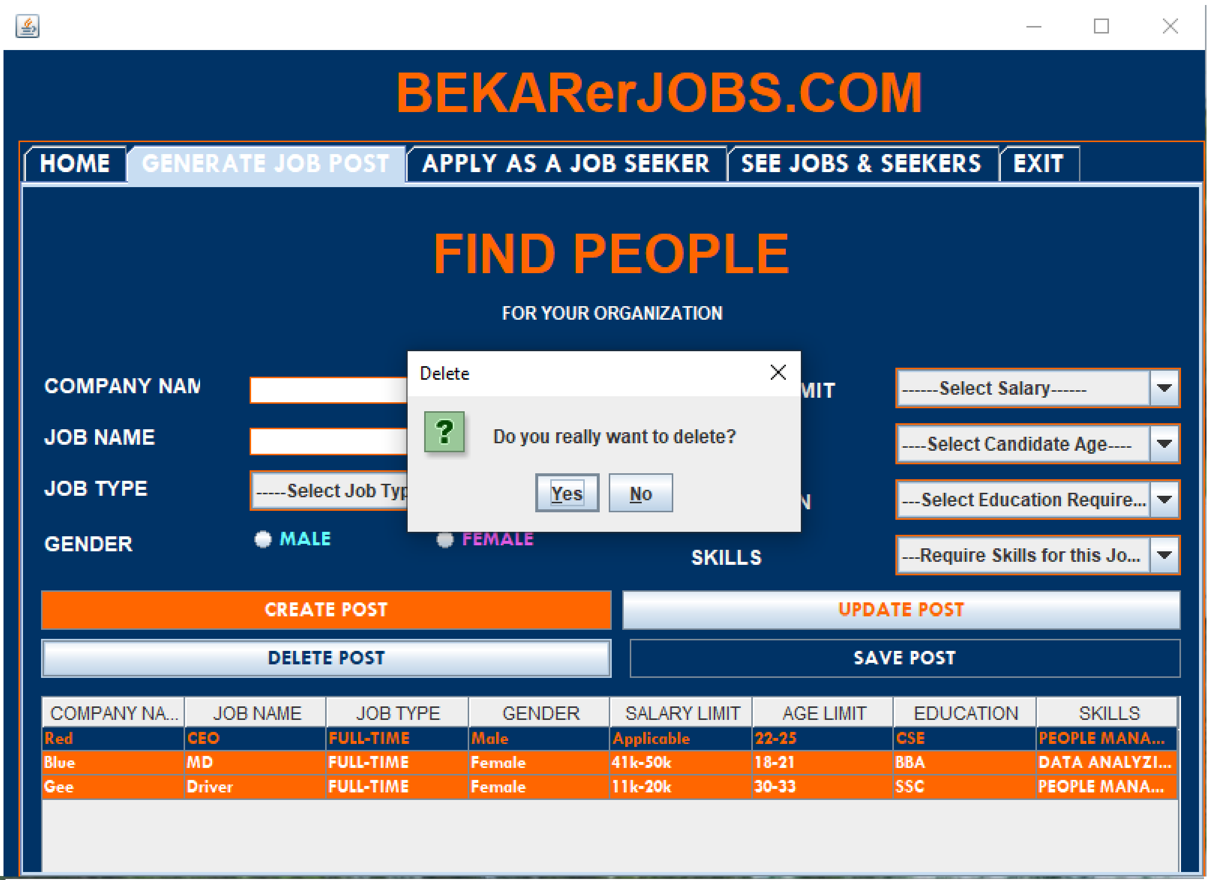
FIND PEOPLE

FOR YOUR ORGANIZATION

| | |
|---|---|
| COMPANY NAM | Gee |
| JOB NAME | Driver |
| JOB TYPE | FULL-TIME |
| GENDER | [✓ MALE] FEMALE |
| SALARY LIMIT | 11k-20k |
| AGE LIMIT | 30-33 |
| EDUCATION | SSC |
| SKILLS | PEOPLE MANAGEMENT |

CREATE POST | UPDATE POST | DELETE POST | SAVE POST

| COMPANY NA... | JOB NAME | JOB TYPE | GENDER | SALARY LIMIT | AGE LIMIT | EDUCATION | SKILLS |
|---|---|---|---|---|---|---|---|
| Red | CEO | FULL-TIME | Male | Applicable | 22-25 | CSE | PEOPLE MANA... |
| Blue | MD | FULL-TIME | Female | 41k-50k | 18-21 | BBA | DATA ANALYZI... |



BEKARerJOBS.COM

HOME | GENERATE JOB POST | APPLY AS A JOB SEEKER | SEE JOBS & SEEKERS | EXIT

FIND PEOPLE

FOR YOUR ORGANIZATION

COMPANY NAN
JOB NAME
JOB TYPE -----Select Job Typ
GENDER ● MALE ● FEMALE
------Select Salary------
----Select Candidate Age----
---Select Education Require...
SKILLS ---Require Skills for this Jo...

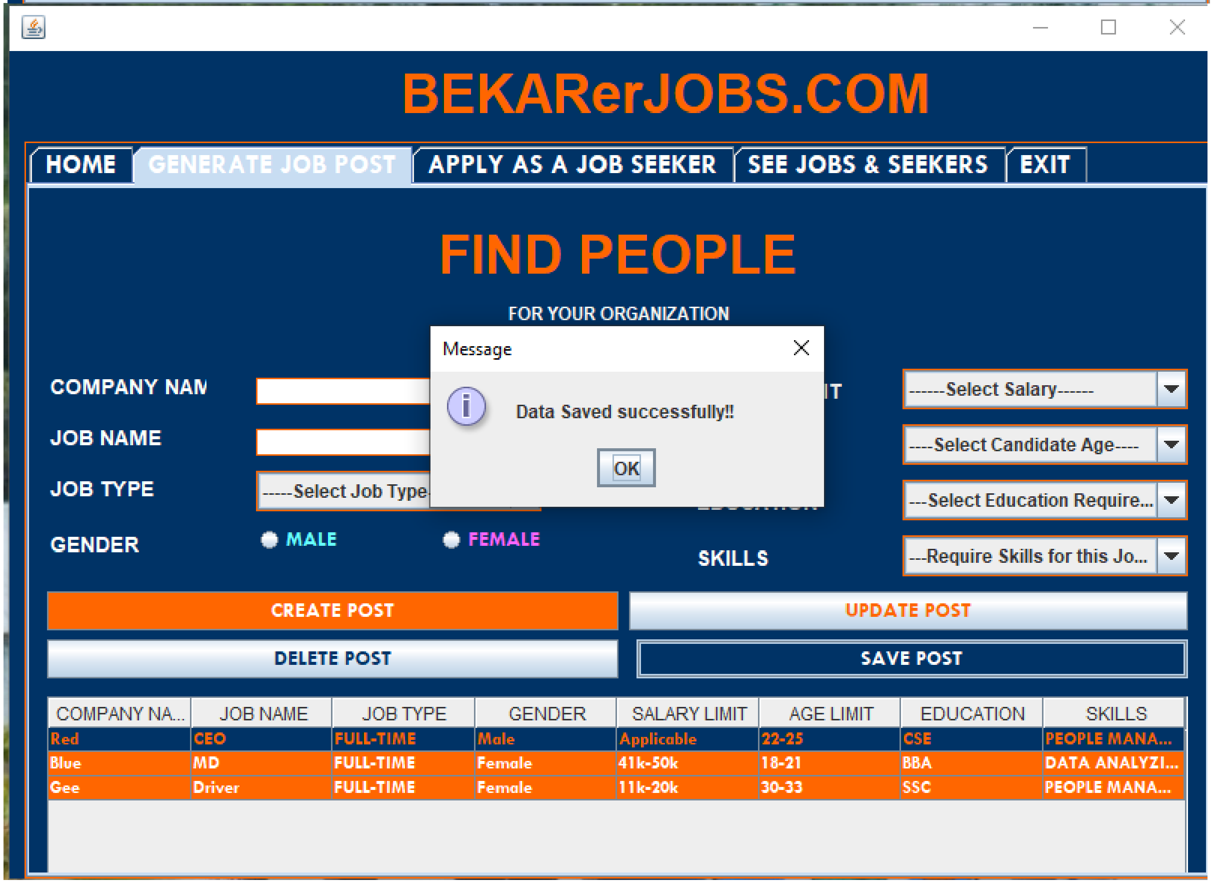
Delete

Do you really want to delete?

Yes No

CREATE POST | UPDATE POST
DELETE POST | SAVE POST

| COMPANY NA... | JOB NAME | JOB TYPE | GENDER | SALARY LIMIT | AGE LIMIT | EDUCATION | SKILLS |
|---|---|---|---|---|---|---|---|
| Red | CEO | FULL-TIME | Male | Applicable | 22-25 | CSE | PEOPLE MANA... |
| Blue | MD | FULL-TIME | Female | 41k-50k | 18-21 | BBA | DATA ANALYZI... |
| Gee | Driver | FULL-TIME | Female | 11k-20k | 30-33 | SSC | PEOPLE MANA... |



BEKARerJOBS.COM

HOME | GENERATE JOB POST | APPLY AS A JOB SEEKER | SEE JOBS & SEEKERS | EXIT

FIND PEOPLE

FOR YOUR ORGANIZATION

COMPANY NAN
JOB NAME
JOB TYPE -----Select Job Type
GENDER ● MALE ● FEMALE
------Select Salary------
----Select Candidate Age----
---Select Education Require...
SKILLS ---Require Skills for this Jo...

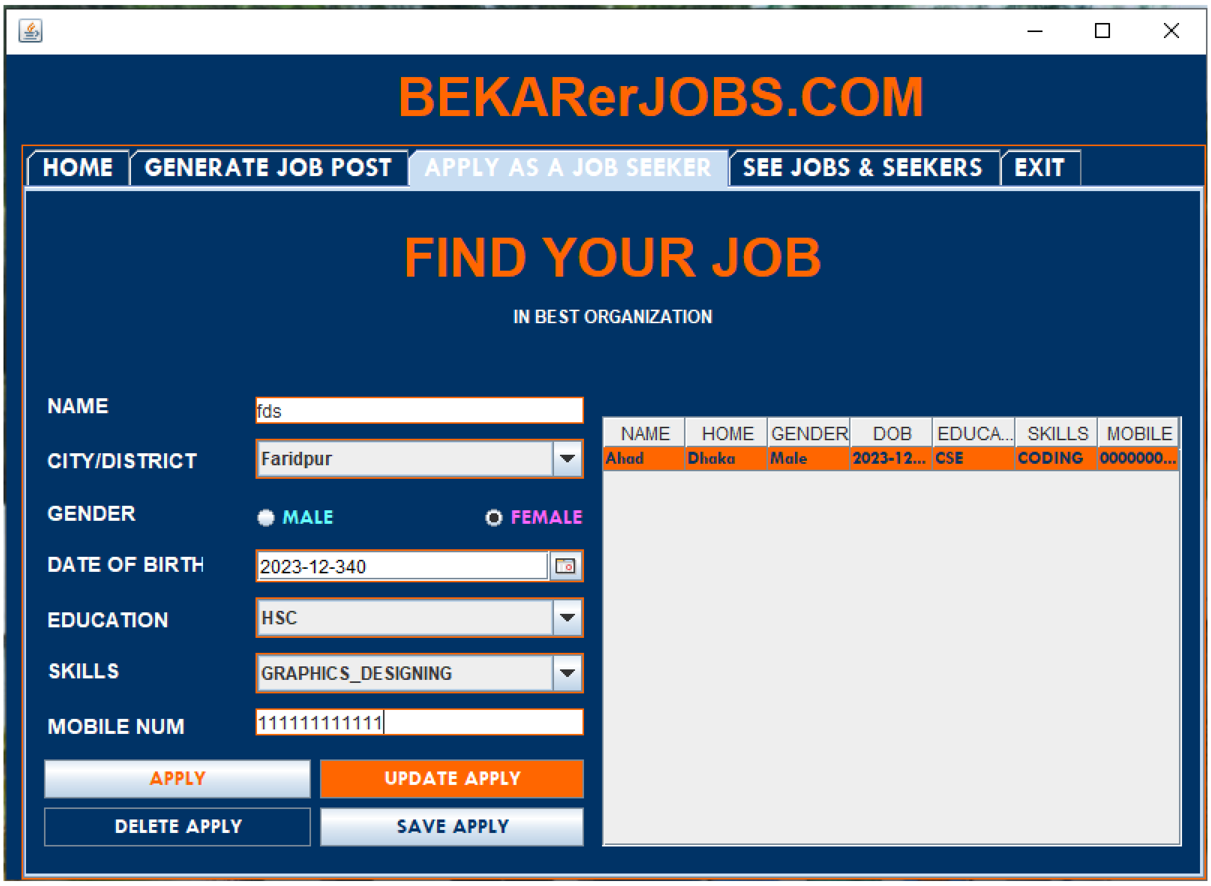
Message

Data Saved successfully!!

OK

CREATE POST | UPDATE POST
DELETE POST | SAVE POST

| COMPANY NA... | JOB NAME | JOB TYPE | GENDER | SALARY LIMIT | AGE LIMIT | EDUCATION | SKILLS |
|---|---|---|---|---|---|---|---|
| Red | CEO | FULL-TIME | Male | Applicable | 22-25 | CSE | PEOPLE MANA... |
| Blue | MD | FULL-TIME | Female | 41k-50k | 18-21 | BBA | DATA ANALYZI... |
| Gee | Driver | FULL-TIME | Female | 11k-20k | 30-33 | SSC | PEOPLE MANA... |

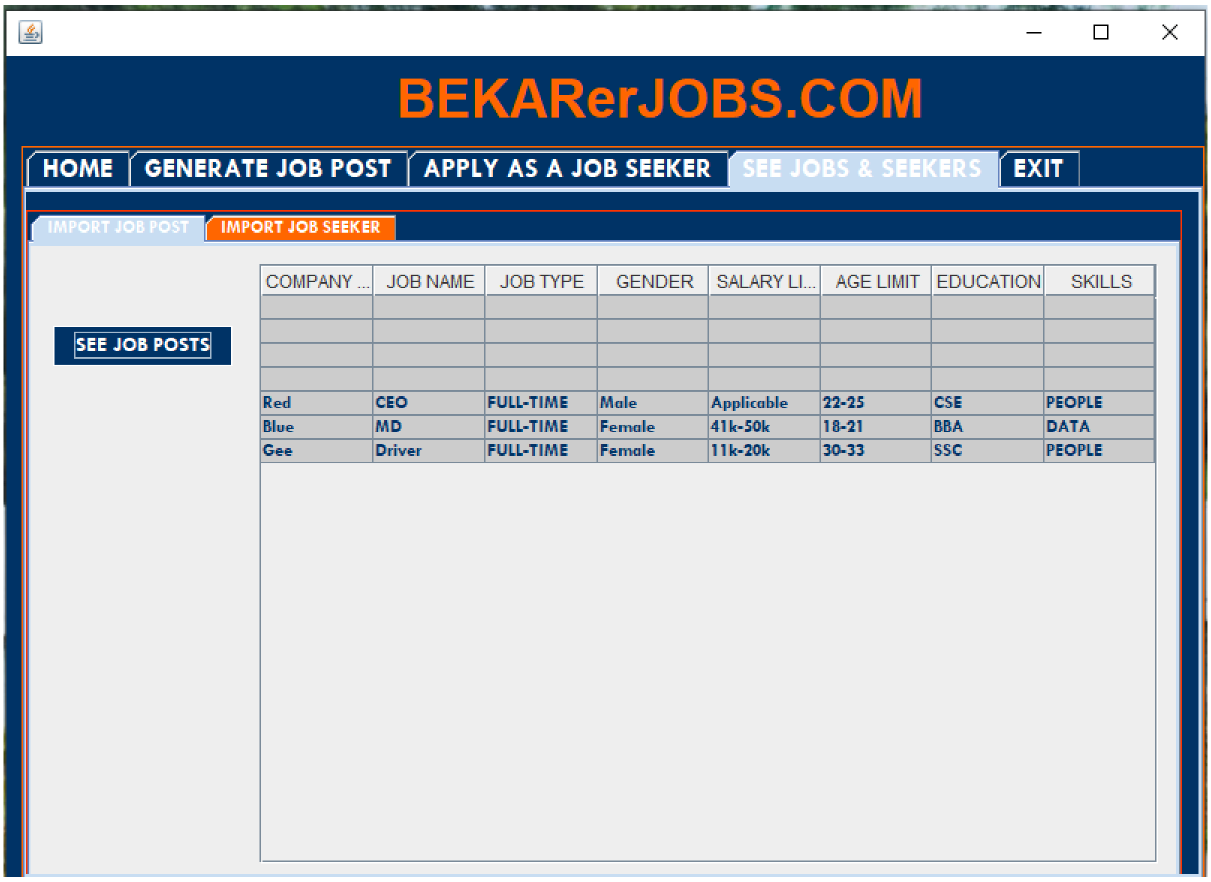


FIND YOUR JOB

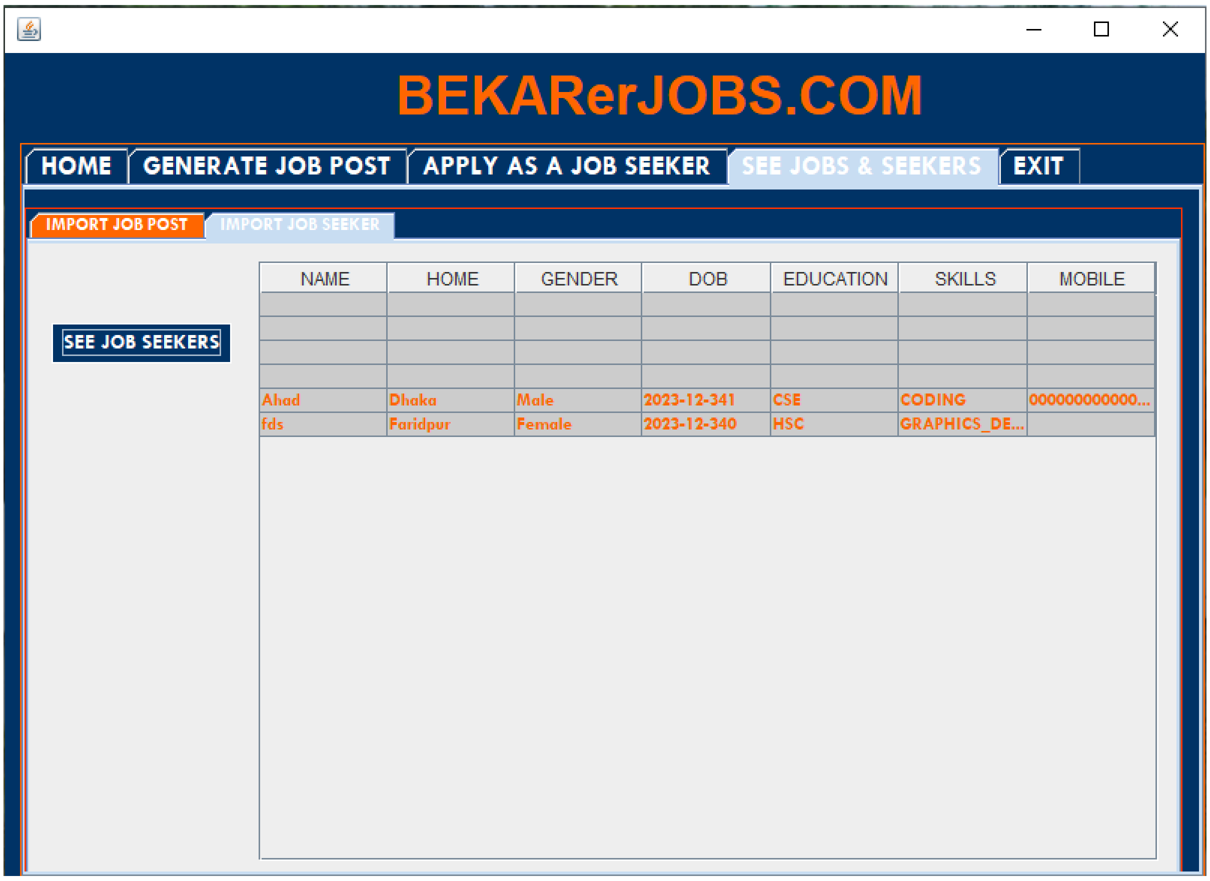
IN BEST ORGANIZATION

| | |
|---|---|
| NAME | fds |
| CITY/DISTRICT | Faridpur |
| GENDER | MALE FEMALE |
| DATE OF BIRTH | 2023-12-340 |
| EDUCATION | HSC |
| SKILLS | GRAPHICS_DESIGNING |
| MOBILE NUM | 111111111111 |

| NAME | HOME | GENDER | DOB | EDUCA... | SKILLS | MOBILE |
|---|---|---|---|---|---|---|
| Ahad | Dhaka | Male | 2023-12... | CSE | CODING | 0000000... |

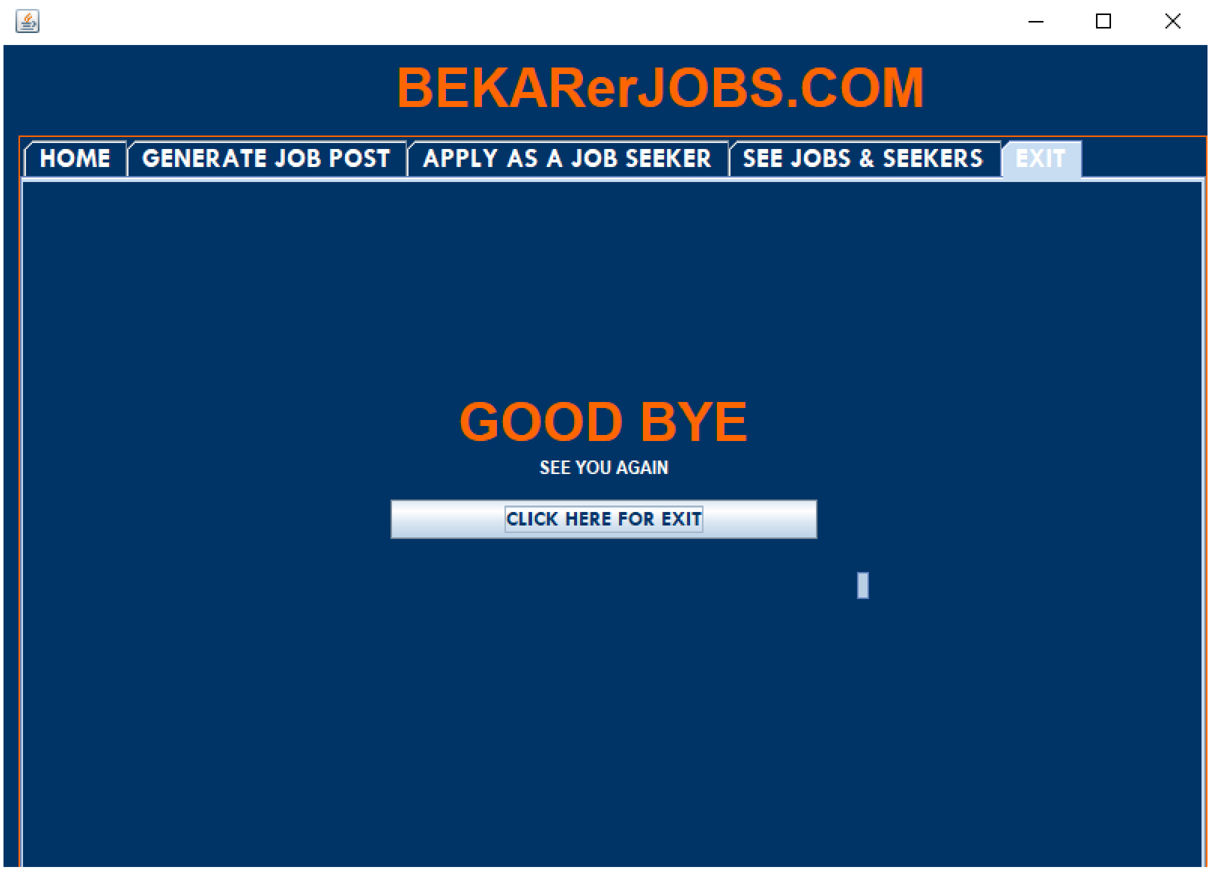


| COMPANY ... | JOB NAME | JOB TYPE | GENDER | SALARY LI... | AGE LIMIT | EDUCATION | SKILLS |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| Red | CEO | FULL-TIME | Male | Applicable | 22-25 | CSE | PEOPLE |
| Blue | MD | FULL-TIME | Female | 41k-50k | 18-21 | BBA | DATA |
| Gee | Driver | FULL-TIME | Female | 11k-20k | 30-33 | SSC | PEOPLE |



BEKARerJOBS.COM

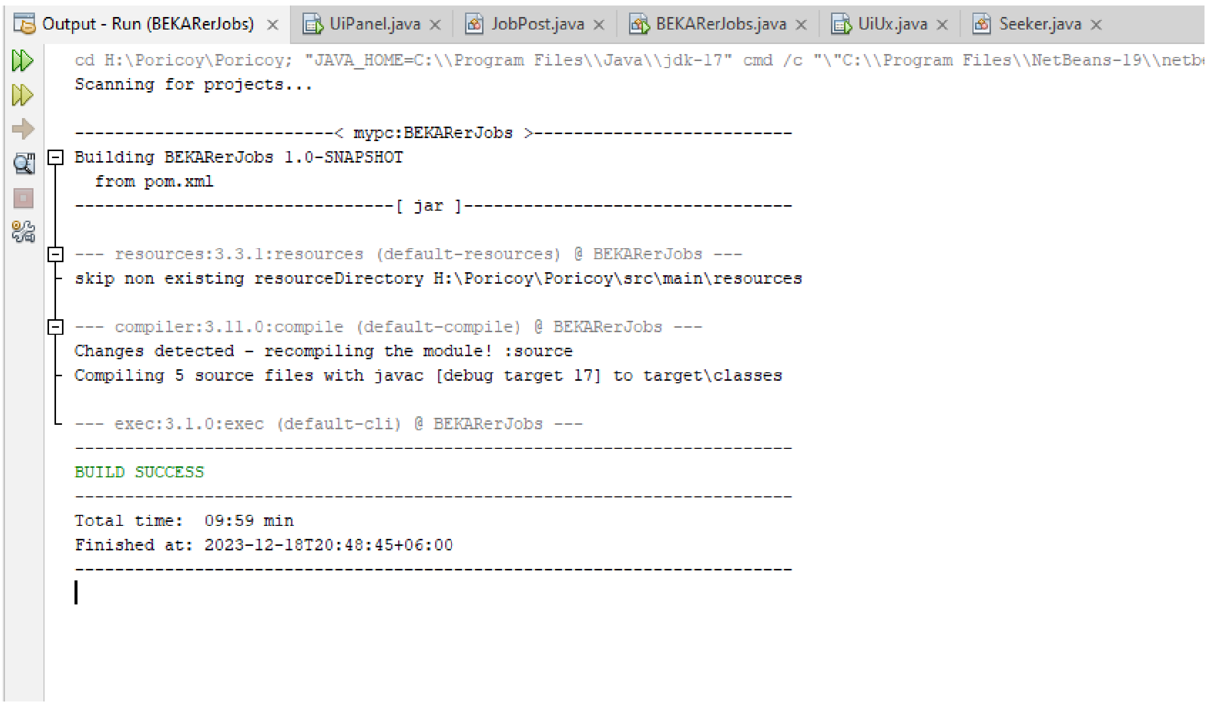
| NAME | HOME | GENDER | DOB | EDUCATION | SKILLS | MOBILE |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| Ahad | Dhaka | Male | 2023-12-341 | CSE | CODING | 000000000000... |
| fds | Faridpur | Female | 2023-12-340 | HSC | GRAPHICS_DE... | |



BEKARerJOBS.COM

GOOD BYE

SEE YOU AGAIN



```
cd H:\Poricoy\Poricoy; "JAVA_HOME=C:\\Program Files\\Java\\jdk-17" cmd /c "\"C:\\Program Files\\NetBeans-19\\netb
Scanning for projects...

--------------------------< mypc:BEKARerJobs >--------------------------
Building BEKARerJobs 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ BEKARerJobs ---
skip non existing resourceDirectory H:\Poricoy\Poricoy\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ BEKARerJobs ---
Changes detected - recompiling the module! :source
Compiling 5 source files with javac [debug target 17] to target\classes

--- exec:3.1.0:exec (default-cli) @ BEKARerJobs ---
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  09:59 min
Finished at: 2023-12-18T20:48:45+06:00
------------------------------------------------------------------------
```

**Figure 6: Output**

## ➢ METHODOLOGY:

**1. User Interface Design:**

- Design the graphical user interface **(GUI)** using Java Swing components.
- Include components for job posts **(text fields, combo boxes, radio buttons)** and seekers.
- Implement table views **(jTable1 and jTable2)** to display job posts and seeker profiles.

**2. Class Definition:**

- Define the **JobPost** class to represent job posts with attributes such as *company details, job title, etc*.
- Define the Seeker class to represent job seekers with attributes *like name, district, etc.*

**3. UiPanel Class:**

- Implement the UiPanel class to handle GUI components and user interactions.
- Initialize **ArrayLists** （**t1 and t2**） to store job posts and seekers.
- Utilize *DefaultTableModel* **(gjp and ajs)** to manage data in table views.
- Implement methods for **adding, updating, and deleting** *job posts* and *seeker profiles*.

- Handle button events **(jButton1ActionPerformed, jButton2ActionPerformed, etc.).**

**4. File Operations:**

- Create a separate class **(FileOperations)** for file input/output operations.
- Implement methods to save job posts and seekers to text files ***(txtfile.txt and seeker.txt)***.
- Implement methods to load job posts and seekers from text files.

**5. Main Execution:**

- In the main method of **UiUx** class:
- Create an instance of **UiUx** and **UiPanel**.
- Set the **UiPanel** to be visible, initiating the **GUI**.

**6. Event-Driven Programming:**

- Utilize event-driven programming to respond to user actions.
- Implement event listeners for buttons to trigger specific actions **(adding, updating, deleting records).**

**7. Data Validation:**

- Implement data validation to ensure that users enter valid information.

- Display **error messages** for **incomplete** or **incorrect** input.

**8. Testing:**

- Test the application thoroughly to identify and address any potential bugs or issues.
- Ensure that all functionalities work as expected, including file I/O operations.

## ➢ CONCLUSION:

**In this system,** Users can efficiently manage job posts and seeker profiles. The implementation includes a well-organized interface, object-oriented design, event-driven programming, and file I/O for data persistence. While the current system serves as a foundation, opportunities for improvement include refining the UI, enhancing error handling, and expanding functionalities. Overall, the code showcases fundamental concepts in GUI development and provides a basis for further customization in job recruitment applications.

**THE END**