ONLINE JOB PORTAL

A PROJECT REPORT

Submitted by

KIRUBHA SHRIVAISHNAVI G (8115U23EC051)

in partial fulfillment of requirements for the award of the course

EGB1201 - JAVA PROGRAMMING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER - 2024

K. RAMAKRISHNAN COLLEGE OF ENGINEERING (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **“ONLINE JOB PORTAL”** is the bonafide work of **KIRUBHA SHRIVAISHNAVI G (8115U23EC051)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

[illegible]	[illegible]
SIGNATURE	**SIGNATURE**
Dr. T. M. NITHYA, M.E.,Ph.D.,	Mr.V.KUMARARAJA, M.E.,(Ph.D.,),
HEAD OF THE DEPARTMENT	**SUPERVISOR**
ASSOCIATE PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Engineering (Autonomous)	K.Ramakrishnan College of Engineering (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 6.12.24

[illegible]	[illegible]
INTERNAL EXAMINER	EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“ONLINE JOB PORTAL”** isthe result of original work done by us and best of our knowledge, similar work has not beensubmitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **EGB1201 - JAVA PROGRAMMING.**

[illegible]

Signature

Kirubha ShriVaishnavi G

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Engineering (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr.V.KUMARARAJA, M.E., (Ph.D.,),** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To achieve a prominent position among the top technical institutions.

MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the artinfrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edgetechnologies.
- M3: To provide education for developing high-quality professionals to transform the society.

VISION OF DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting qualityeducation.

MISSION OF DEPARTMENT

M1: To provide technical exposure in the field of Computer Science and Engineering throughstate of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of ComputerScience and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects foraddressing the societal needs.

PROGRAM EDUCATIONAL OBJECTIVES

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.
- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activitieswith an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assesssocietal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and needfor sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leaderin diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Online Job Portal is a comprehensive web-based application designed to bridge the gap between job seekers and employers. The primary objective of this system is to streamline the job search and recruitment processes by providing a user- friendly platform where employers can post job vacancies and job seekers can find suitable employment opportunities.

Key features of the Online Job Portal include user registration and authentication, job posting, job search, resume upload, and application management. Job seekers can create profiles, upload their resumes, search for job openings based on various criteria, and apply for positions that match their qualifications and interests. Employers can register on the portal, post job advertisements, search for potential candidates, and manage the applications they receive.

The portal leverages advanced technologies to ensure a seamless and efficient experience for users. The backend is developed using Java, incorporating robust object-oriented programming principles. Technologies such as Java Servlets, Java Server Pages (JSP), and JDBC are utilized for dynamic content generation and database connectivity. A relational database, such as MySQL, is employed to store user profiles, job listings, and application data securely.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Online Job Portal is a comprehensive web-based application designed to bridge the gap between job seekers and employers. The primary objective of this system is to streamline the job search and recruitment processes by providing a user- friendly platform where employers can post job vacancies and job seekers can find suitable employment opportunities.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 OBJECTIVE

The Online Job Portal aims to create a user-friendly platform connecting job seekers with employers. It facilitates efficient job searches, applications, and recruitment processes. Job seekers can search for jobs, apply directly, and manage their profiles, while employers can post job vacancies, search for candidates, and track applications. The system ensures data security, provides a seamless user experience, and promotes equal opportunity employment. By offering advanced search options and an intuitive interface, the portal simplifies job matching and candidate evaluation, making the recruitment process more streamlined and effective.

1.2 OVERVIEW

The Online Job Portal is a web-based application designed to connect job seekers with employers. It streamlines the job search and recruitment processes by offering a user-friendly platform. Job seekers can search for job listings, apply directly, and manage their profiles, while employers can post job vacancies, search for candidates, and manage applications. The system ensures data security, provides an intuitive interface, and promotes equal opportunity employment, making the recruitment process efficient and effective.

The online job portal is a Java-based application designed to streamline the interaction between job seekers and employers. It provides job seekers with features such as registration, profile management, job search, and application submission, while enabling employers to post job openings and manage applications. The project employs object-oriented programming concepts like classes, inheritance, and polymorphism, with AWT and Swing for the user interface. Event handling facilitates user interaction, while file handling or database integration ensures data persistence. This system is designed to be simple yet scalable, offering secure authentication, data management, and a modular architecture for future enhancements.

1.3 JAVA PROGRAMMING CONCEPTS

The **Online Job Portal** project leverages various core Java programming concepts. These concepts are essential for understanding the functionality and implementation of the application.

- **Class and Object**:
  - Classes define the blueprint (e.g., User, JobPost), and objects are instances of these classes that hold data and perform actions.
- **Inheritance**:
  - Allows the creation of specialized classes (e.g., Employer and JobSeeker) that inherit common properties and methods from a base class like User.
- **Polymorphism**:
  - **Method Overloading**: Enables multiple methods with the same name but different parameters (e.g., searchJob(String title) and searchJob(String title, String location)).

- **Method Overriding**: Allows subclasses to provide specific implementations of methods defined in the parent class.

 Encapsulation:

- Protects data by declaring fields as private and providing public getter and setter methods to access and update them securely.

 Interfaces and Abstract Classes:

- Interfaces define common behaviors (e.g., Searchable interface for job searching).
- Abstract classes provide a shared base for specialized subclasses while allowing specific method implementations.

 AWT and Swing:

- AWT provides basic GUI components like buttons and text fields, while Swing enhances the GUI with additional features like tables and dialog boxes.

 Event Handling:

- Manages user interactions, such as button clicks or form submissions, by implementing listeners like ActionListener.

 Exception Handling:

- Ensures the program continues running smoothly by catching and handling errors like invalid inputs or file not found exceptions using try-catch blocks.

 File Handling:

- Stores and retrieves data (e.g., user details, job posts) using file operations like reading and writing files.

 Collections Framework:

- Uses data structures like ArrayList or HashMap to store and manage dynamic collections of jobs, users, or applications.

 Constructor (Default and Parameterized):

- Used to initialize objects, with parameterized constructors allowing the setting of initial values for fields.

- **Control Structures**:
  - Includes if-else conditions, loops (for, while), and switch cases to implement decision-making and repetitive tasks.
- **Access Modifiers**:
  - Controls the visibility of classes, methods, and variables using modifiers like public, private, and protected.

CHAPTER 2
PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed work for the online job portal focuses on building a platform that connects job seekers with employers, facilitating an efficient recruitment process. The system will allow users to register, log in securely, and manage their profiles. Job seekers can upload resumes, search for jobs using filters like location and skills, and apply directly through the portal. Employers will have the ability to post job listings, update or delete them, and track applications from candidates.

The platform will feature a user-friendly graphical interface built with AWT and Swing to ensure smooth interaction. Data management will be handled using file handling or database integration to store and retrieve user profiles, job details, and application statuses. Additionally, secure authentication mechanisms will be implemented to protect user data, and robust exception handling will ensure a seamless user experience even in the event of errors. This proposed work aims to deliver a functional, secure, and scalable system for job seekers and employers.

2.2 BLOCK DIAGRAM

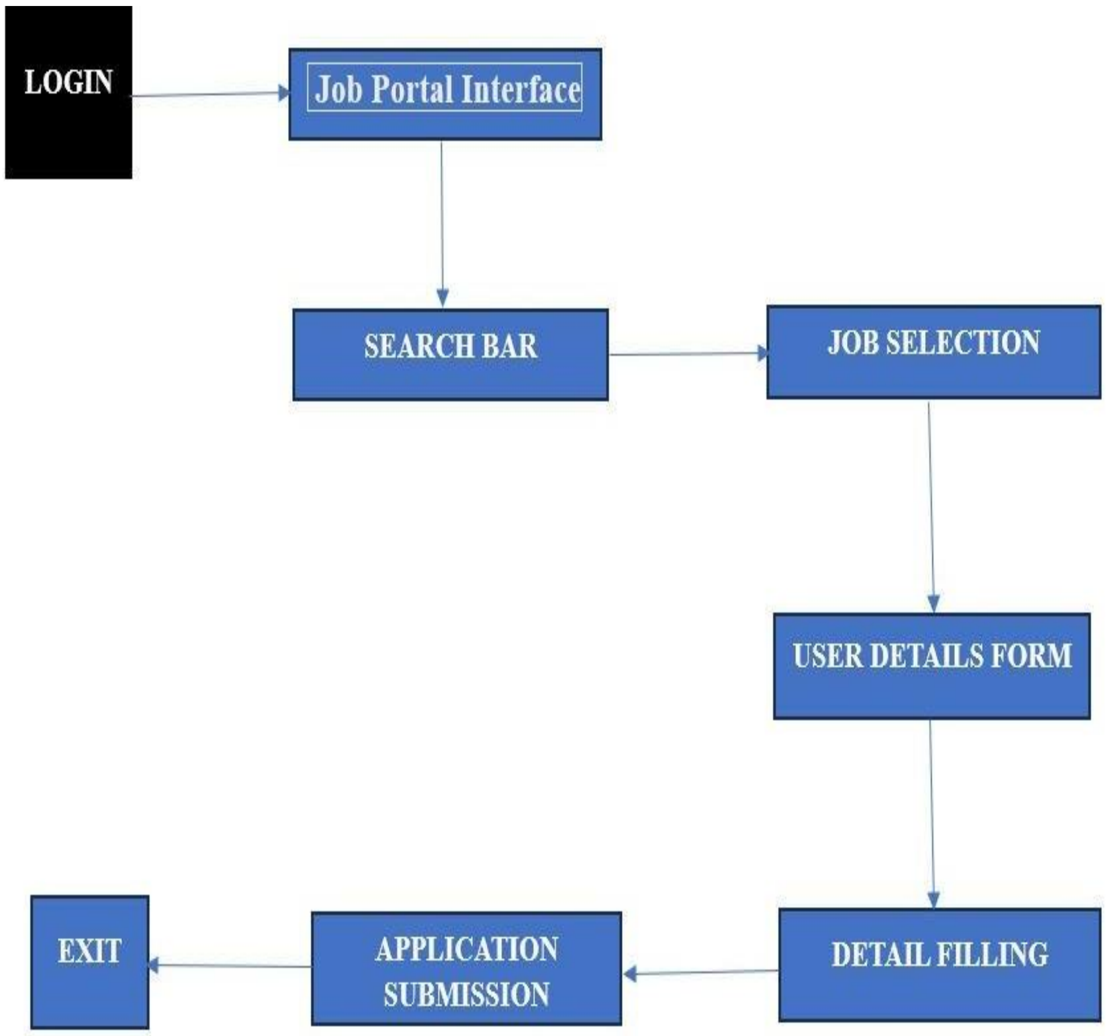


Fig 2.1 Search Engine system

CHAPTER 3

MODULE DESCRIPTION

3.1 USER MANAGEMENT MODULE

The User Management Module handles the registration, authentication, and role-based access of users. It allows job seekers and employers to create accounts, log in securely, and manage their profiles. Job seekers can upload resumes and update personal details, while employers can manage their company profiles. This module ensures a seamless user experience by differentiating roles and providing tailored features for each. Core functionalities include registration, login/logout, and user profile management, implemented using classes like User, JobSeeker, and Employer, with secure password handling and session management**.**

3.2 JOB POSTING MODULE

The Job Posting Module enables employers to create and manage job listings. Employers can add job details such as title, description, required skills, location, and salary. They can also edit or delete existing postings and view a list of their active job listings. This module serves as a crucial feature for employers to attract potential candidates. It is implemented using a JobPost class with attributes to capture job-related information, and data is stored persistently using filehandling.

3.3 JOB SEARCH MODULE

The Job Search Module is designed for job seekers to find relevant job postings based on various filters, such as job title, location, required skills, or salary range. It allows users to perform targeted searches and displays results in a structured and user-friendly format. This module ensures job seekers can efficiently find opportunities that match their preferences. The implementation uses methods like searchJob() to filter job postings dynamically, leveraging data structures like ArrayList for efficient data retrieval.

3.4 APPLICATION MANAGEMENT MODULE

The Application Management Module handles the process of job seekers applying for jobs and employers reviewing applications. Job seekers can apply for jobs by submitting their resumes and track the status of their applications (e.g., "Under Review," "Accepted"). Employers can view submitted applications, shortlist candidates, and manage the hiring process. This module ensures a streamlined application flow, bridging the gap between job seekers and employers. It involves mapping applications to specific job postings and managing data using collections or persistent storage.

3.5 AUTHENTICATION AND SECURITY MODULE

The Authentication and Security Module ensures the secure access of the system. It validates user credentials during login and enforces role-based access controls, restricting features to authorized users only. This module also protects sensitive data, such as passwords, using encryption techniques. Additionally, it manages session security to prevent unauthorized access. By implementing robust authentication mechanisms, this module safeguards user information and maintains the integrity of the system.

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the online job portal provides an efficient platform for job seekers and employers to connect, streamlining the recruitment process. By leveraging core programming concepts such as object-oriented principles, file handling, and event-driven programming, the system ensures modularity, scalability, and user-friendliness. The inclusion of a well-designed graphical user interface, secure authentication mechanisms, and role-based access controls enhances usability and security, making the application a reliable solution for job hunting and hiring needs. The modular architecture also facilitates easy management of user accounts, job postings, applications, and search functionalities.

Looking ahead, this project lays a strong foundation for further enhancements, such as integrating advanced features like email notifications, real-time chat, or database connectivity for better data management. The system’s scalability ensures that it can evolve to meet the growing demands of modern recruitment. With its simple yet effective design, this job portal not only improves the user experience but also demonstrates the practical application of Java programming concepts in solving real-world problems.

4.2 Future Scope

The online job portal has significant potential for enhancement and expansion. One key area for future development is the integration of a **relational database** such as MySQL or PostgreSQL, which would improve data storage, retrieval, and scalability compared to file-based storage. This would also facilitate advanced query capabilities for better job search functionality, and allow for easier management of large datasets, including user profiles, job postings, and applications.

Additionally, incorporating **AI-powered job matching algorithms** could be a major advancement. By using machine learning, the system could recommend jobs to seekers based on their skills, preferences, and previous applications, while employers could receive more accurate candidate matches. Further, the addition of **real-time notifications**, **email alerts**, and a **chat system** could enhance user engagement and streamline communication between job seekers and employers. Integration with social media platforms for job sharing and professional networking could also be explored to increase the visibility of job postings and expand the pool of candidates.

Lastly, implementing a **mobile application** version of the portal could broaden accessibility, allowing users to search, apply, and manage profiles on-the-go, thus improving the overall user experience. The security mechanisms could also be upgraded with **two-factor authentication** to further safeguard user accounts and sensitive data. With these improvements, the job portal could evolve into a more robust, interactive, and feature-rich platform, catering to the evolving needs of both job seekers and employers.

APPENDIX A (SOURCE CODE)

```
        package searchengine;


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.ArrayList;

public class JobPortalApp extends Frame implements ActionListener {

// UI components for login
Label lblUsername, lblPassword, lblLoginMessage;
TextField txtUsername, txtPassword;
Button btnLogin, btnLoginBack;

// UI components for job search
Label lblSearch, lblFilter;
TextField txtKeyword;
Choice chLocation;
Button btnSearch, btnSearchBack; java.awt.List lstResults;

// UI components for application form
Label lblPersonalDetails, lblEducationDetails;
TextArea txtPersonalDetails, txtEducationDetails;
Button btnSubmitApplication, btnApplyBack;

// Job list and application details ArrayList<Job> jobDatabase; Job selectedJob;

public JobPortalApp() {
setLayout(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns, with 10px gaps

// Login UI components
lblUsername = new Label("Username:");
add(lblUsername);

txtUsername = new TextField(20);
```

```
add(txtUsername);

lblPassword = new Label("Password:");
add(lblPassword);

txtPassword = new TextField(20);
txtPassword.setEchoChar('*'); // Mask password input add(txtPassword);

btnLogin = new Button("Login");
btnLogin.addActionListener(this);
add(btnLogin);

btnLoginBack = new Button("Exit");
btnLoginBack.addActionListener(this); // Exit button for login screen add(btnLoginBack);

lblLoginMessage = new Label();
add(lblLoginMessage);

// Set frame properties for login
setSize(400, 250);
setTitle("Job Portal - Login");
setVisible(true);

// Window closing handler
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) {
dispose();
System.exit(0);
}
});
}

// Action handler for button clicks @Override
public void actionPerformed(ActionEvent e) {
if (e.getSource() == btnLogin)
{
String username = txtUsername.getText();
String password = txtPassword.getText();
// For simplicity, assuming login is successful if both fields are non-empty if
(!username.isEmpty() && !password.isEmpty()) {
```

```
lblLoginMessage.setText("Login successful!");
// Transition to the job portal screen after login showJobPortal();
} else {
lblLoginMessage.setText("Please enter valid credentials.");
}
}

// Exit button to close the application if (e.getSource() == btnLoginBack) {
dispose();
System.exit(0);
}

// Search button action
if (e.getSource() == btnSearch) {
String keyword = txtKeyword.getText().toLowerCase();
String location = chLocation.getSelectedItem();
lstResults.removeAll(); // Clear previous results
var results = JobSearch.searchJobs(keyword, location);
if (results.isEmpty()) {
lstResults.add("No jobs found matching the criteria.");
} else {
for (var job : results) {
lstResults.add("Job: " + job.title + " at " + job.company + ", Location: "
+ job.location);
}
}
}

// Submit application button action
if (e.getSource() == btnSubmitApplication) {
String personalDetails = txtPersonalDetails.getText();
String educationDetails = txtEducationDetails.getText();
// Simulate saving the application
System.out.println("Application Submitted for job: " + selectedJob.title);
System.out.println("Personal Details: " + personalDetails); System.out.println("Education
Details: " + educationDetails);
// Show confirmation
JOptionPane.showMessageDialog(this, "Your application has been submitted
successfully.", "Application Submitted", JOptionPane.INFORMATION_MESSAGE);
}
```

```
// Back button to return to job portal if (e.getSource() == btnSearchBack) {
showJobPortal();
}

// Back button to return to login screen if (e.getSource() == btnApplyBack) {
showLoginScreen();
}
}

// Show Job Portal screen public void showJobPortal() {
// Clear previous UI components
removeAll();

// Set layout for job portal UI setLayout(new GridLayout(6, 2, 10, 10));

// Job search components
lblSearch = new Label("Search Jobs:");
add(lblSearch);

txtKeyword = new TextField(20);
add(txtKeyword);

lblFilter = new Label("Location:");
add(lblFilter);

chLocation = new Choice();
chLocation.add("Any");
chLocation.add("New York");
chLocation.add("San Francisco");
chLocation.add("Seattle");
add(chLocation);

btnSearch = new Button("Search");
btnSearch.addActionListener(this);
add(btnSearch);

lstResults = new java.awt.List();
```

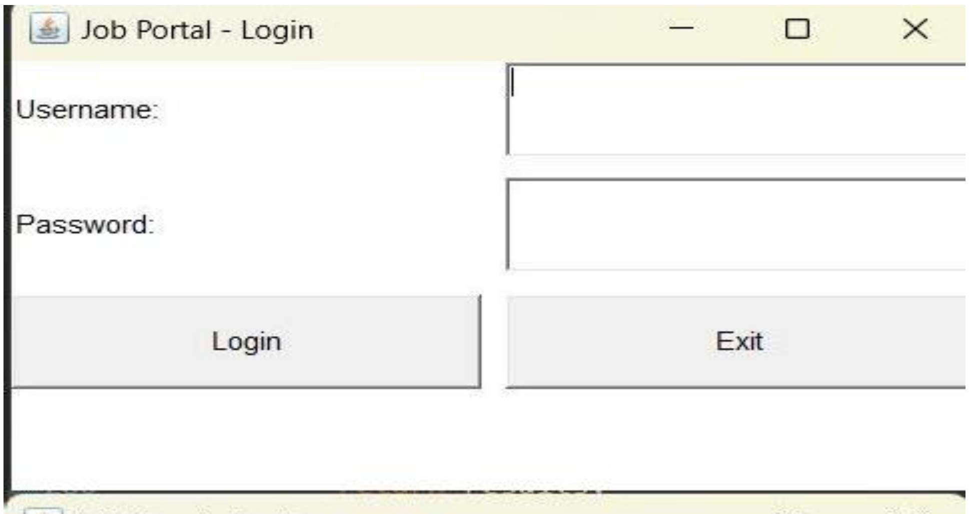
```
lstResults.addActionListener(new ActionListener() {
 @Override
public void actionPerformed(ActionEvent e) {
int index = lstResults.getSelectedIndex();
if (index != -1) {
selectedJob = JobSearch.jobDatabase.get(index);
showApplicationForm();
}
}
});
add(lstResults);

// Add Back button for job portal screen
btnSearchBack = new Button("Back to Login");
btnSearchBack.addActionListener(this);
add(btnSearchBack);

// Load job data
jobDatabase = JobSearch.jobDatabase;

// Refresh the frame validate();
repaint();
}

// Show Application Form
public void showApplicationForm() {
// Clear previous UI components removeAll();

// Set layout for application form UI setLayout(new GridLayout(5, 2, 10, 10));

lblPersonalDetails = new Label("Personal Details:");
add(lblPersonalDetails);

txtPersonalDetails = new TextArea(5, 30);
add(txtPersonalDetails);

lblEducationDetails = new Label("Education Details:");
add(lblEducationDetails);
```

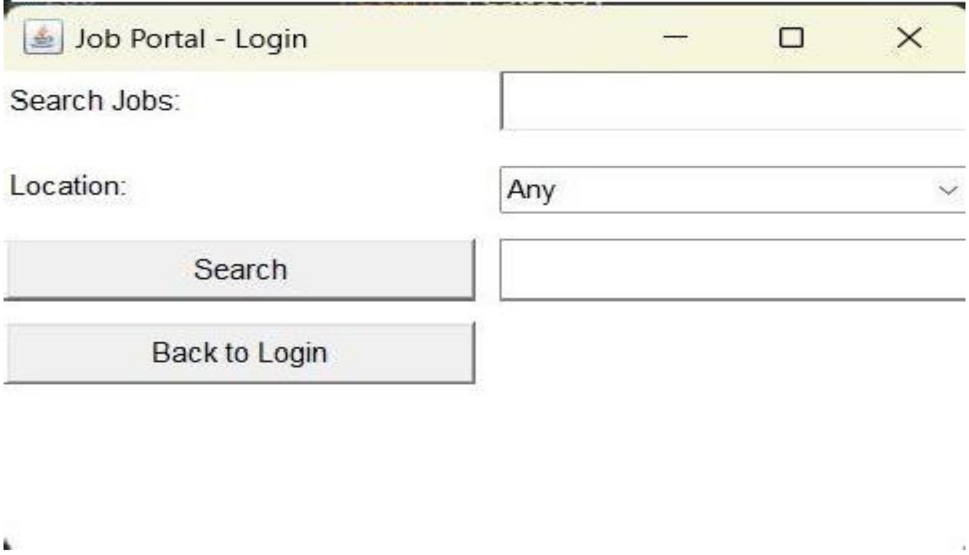
```
txtEducationDetails = new TextArea(5, 30);
add(txtEducationDetails);

btnSubmitApplication = new Button("Submit Application");
btnSubmitApplication.addActionListener(this);
add(btnSubmitApplication);

// Add Back button for application form
btnApplyBack = new Button("Back to Job Portal");
btnApplyBack.addActionListener(this);
add(btnApplyBack);

// Refresh the frame validate();
repaint();
}

// Show Login Screen
public void showLoginScreen() {
// Clear previous UI components
removeAll();

// Set layout for login screen UI
setLayout(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns

// Add login components back
lblUsername = new Label("Username:");
add(lblUsername);

txtUsername = new TextField(20);
add(txtUsername);

lblPassword = new Label("Password:");
add(lblPassword);

txtPassword = new TextField(20);
txtPassword.setEchoChar('*');
add(txtPassword);

btnLogin = new Button("Login");
btnLogin.addActionListener(this);
add(btnLogin);
```

```
btnLoginBack = new Button("Exit");
btnLoginBack.addActionListener(this); // Exit button
add(btnLoginBack);

lblLoginMessage = new Label();
add(lblLoginMessage);

// Refresh the frame
validate();
repaint();
}

// Job search logic with a mock dataset
static class JobSearch {
static ArrayList<Job> jobDatabase = new ArrayList<>();

static {
// Mock dataset
jobDatabase.add(new Job("Software Engineer", "Tech Corp", "New York"));
jobDatabase.add(new Job("Data Scientist", "Data Inc", "San Francisco"));
jobDatabase.add(new Job("Web Developer", "Web Solutions", "Seattle"));
jobDatabase.add(new Job("Product Manager", "Innovate Ltd", "New York"));
}

public static ArrayList<Job> searchJobs(String keyword, String location) {
ArrayList<Job> results = new ArrayList<>();
for (Job job : jobDatabase) {
if ((location.equals("Any") || job.location.equalsIgnoreCase(location)) &&
(job.title.toLowerCase().contains(keyword)  ||
job.company.toLowerCase().contains(keyword))) { results.add(job);
}
}
return results;
}
}

// Job class representing a job entry
static class Job {
String title;
String company;
String location;

public Job(String title, String company, String location) {
this.title = title;
this.company = company;
this.location = location;
}
}

// Main method to launch the application
public static void main(String[] args) {
new JobPortalApp();
}
}
```

APPENDIX B (SCREENSHOTS)

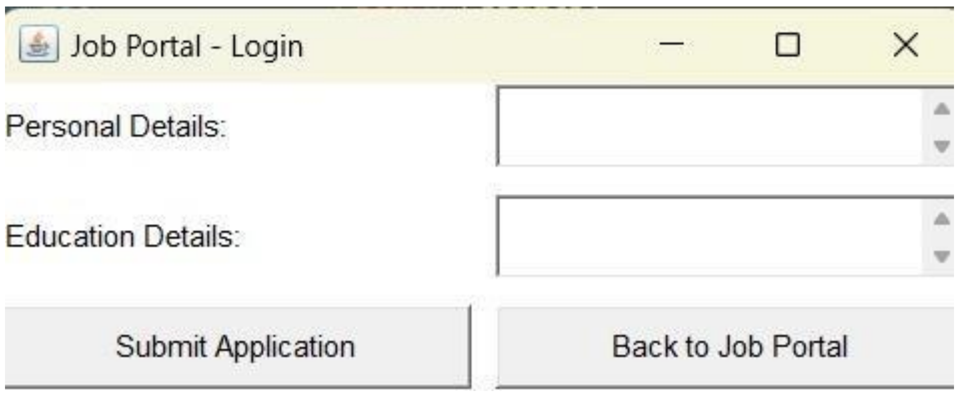
RESULT:



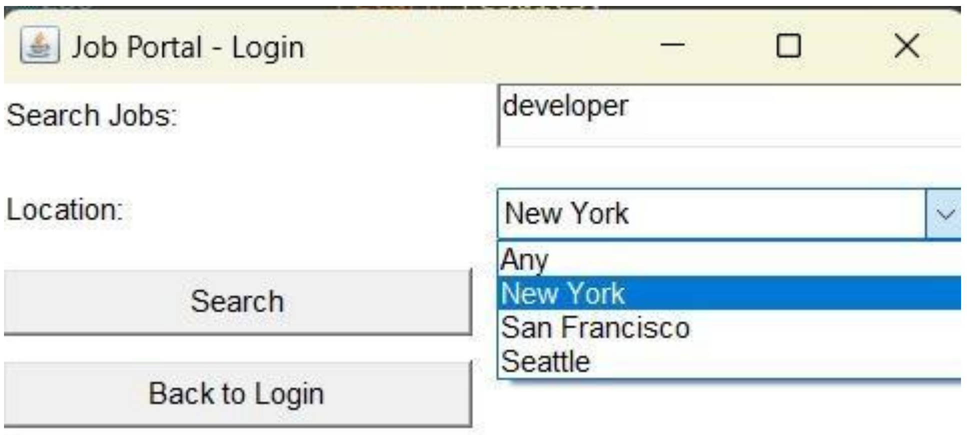


Location: Any

1. **Login module:**



2.Job search and Job Application:



Search Jobs: developer

Location: New York

REFERENCES:

1. **Java Platform SE Developer Guide – AWT** This guide focuses on using the Abstract Window Toolkit (AWT) for GUI development. Link: Java SE Developer Guide.

2. **Stack Overflow** A highly recommended platform for troubleshooting and finding examples of Java AWT programs: Search for "Java AWT examples".

3. **Reddit: LearnProgramming** The programming community often discusses topics related to Java GUI development. Link: LearnProgramming Subreddit.

4. **JavaPoint - AWT Tutorial** Comprehensive step-by-step tutorials, including examples and explanations for AWT components. Link: JavaPoint AWT Tutorial.

5. **W3Schools - Java AWT Basics** Simple and easy-to-follow guides with quick code snippets for practice. Link: W3Schools Java AWT.

6. **Telusko - Java GUI Programming** Detailed explanations with examples covering AWT and Swing.
Search for "Telusko Java GUI AWT" on YouTube. Link: Telusko Channel

7. **ProgrammingKnowledge** Practical video demonstrations of Java GUI applications using AWT and Swing. Link: ProgrammingKnowledge Channel.

8. **Java World - Advanced AWT Techniques** Offers insights into advanced uses of AWT, including event listeners and layouts. Link: Java World AWT Resources.

9. **Vogella - AWT and Swing Tutorials** Aimed at intermediate programmers, this resource covers AWT basics and advanced topics like custom components. Link: Vogella AWT Guide.

10. **Java AWT Example Projects** Repository: https://github.com/topics/awt.Explore various open-source AWT-based projects to learn practical implementations.