



ONLINE JOB PORTAL



A PROJECT REPORT

Submitted by

ELUMALAI S (2303811710421039)

In partial fulfillment of requirements for the award of the course

CGB1201-JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K . RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM-621112

NOVEMBER-2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)

SAMAYAPURAM-621112

BONAFIDE CERTIFICATE

Certified that this project report on **“ONLINE JOB PORTAL”** is the bonafide work of **ELUMALAI S (2303811710421039)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E., PD

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

CGB1201-JAVA PROGRAMMING
Mr. M. SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

MR.M. SARAVANAN, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 02.12.2024.

CGB1201-JAVA PROGRAMMING
Mr. M. ARMANAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Dr. R. SETHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“ONLINE JOB PORTAL”** is the result of original work done by us and best of our knowledge, similar work has not been submitted 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 **CGB1201- JAVA PROGRAMMING.**

Signature



ELUMALAI S

Place: Samayapuram

Date: 02.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (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. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, 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. M.SARAVANAN, M.E.**, 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 serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

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 activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, 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 need for 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 leader in 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

This program focuses on developing a Job Portal System that enables both employers and job seekers to interact efficiently. The core functionalities include job posting by employers, job searching by domain for job seekers, and the submission of job applications. Employers can view the list of submitted applications, while job seekers can search for available jobs in various domains and apply for positions. The system is implemented using Java, utilizing its object-oriented programming features to create a modular, scalable, and maintainable architecture. To enhance the user experience, the project incorporates the Abstract Window Toolkit (AWT) to design an intuitive and visually appealing Graphical User Interface (GUI). The interface includes components such as buttons, labels, text fields, and choice boxes, ensuring seamless interaction for both employers and job seekers.

ABSTRACT WITH Pos AND PSOs MAPPING

CO5:BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POsMAPPED	PSOsMAPPED
<p>The system is implemented using Java, leveraging its powerful features to ensure reliability, modularity, and effective error handling. Object-oriented programming principles are employed to create a maintainable, scalable, and flexible architecture. Additionally, the project incorporates the Abstract Window Toolkit (AWT) to design a graphical user interface (GUI), enhancing user experience with intuitive and visually engaging components. Core functionalities include job posting by employers, job searching by domain for job seekers, viewing job details, and submitting job applications. Employers can also view submitted applications, while job seekers are provided with a seamless application process.</p>	<p>PO1-3 PO2-3 PO3-3 PO4-3 PO5-3 PO6-3 PO7-3 PO8-3 PO9-3 PO10-3 PO11-3 PO12-3</p>	<p>PSO1-3 PSO2-3 PSO3-3</p>

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

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	3
	2.1 Proposed Work	3
	2.2 Block Diagram	5
3	MODULE DESCRIPTION	6
	3.1 Main Menu Module	6
	3.2 Employer Portal Module	6
	3.3 Job Seeker Portal Module	6
	3.4 Application Handling Module	7
4	RESULTS AND DISCUSSION	8
5	CONCLUSION	10
	REFERENCES	11
	APPENDIX (Source Code)	12
		17

CHAPTER 1

INTRODUCTION

1. Objective:

The objective of the Job Portal program is to provide a simple, user-friendly application that simulates the functionalities of a job portal, including job posting by employers, job searching by domain for job seekers, application submission, and viewing submitted applications. It utilizes Java AWT for an interactive graphical interface, object-oriented programming principles for managing job and application data, and basic error handling to ensure a smooth user experience. This project serves as an educational tool to demonstrate how programming concepts can be applied to develop practical, real-world applications.

The main objective of the Job Portal is to create an intuitive and interactive application that facilitates seamless interaction between employers and job seekers. Using Java AWT for the interface and object-oriented principles for data management, the program provides a practical example of building a user-centric and functional recruitment system.

2. Overview:

The Job Portal program is a Java-based application designed to replicate the core functionalities of an online job portal. It provides a graphical user interface (GUI) built with Java AWT to enable users to interact with the system in an intuitive manner. The application supports two main user roles: employers who can post job listings and job seekers who can search and apply for jobs. The program leverages object-oriented programming principles, with Job and Application classes to manage job details, job seeker applications, and related data. It also incorporates basic error handling to address scenarios like incomplete job postings or job application submissions, ensuring a smooth and efficient user experience.

3. Java Programming Concepts:

- **Object-Oriented Programming(OOP)**

Classes (Job, Application) encapsulate data and behavior, following OOP principles such as encapsulation and abstraction.

- **GUI Programming with AWT**

Utilizes AWT components like Frame, Label, Button, TextField, and TextArea for creating a user-friendly graphical interface.

- **AWT Layout Management**

Manages UI layouts with FlowLayout for organizing components and ensuring an intuitive user experience.

- **User Interaction and Input Handling**

Provides user input fields for job posting and application submission, validating that all required fields are filled before proceeding.

- **Data Management**

Uses ArrayList to store job and application data, allowing for dynamic updates and retrievals.

- **Error Handling and Feedback**

Displays feedback to the user, such as error messages for incomplete forms or validation issues, using labels and text fields.

- **Control Flow:**

- **If-Else Statements:** Used for validating user input during job posting and application submission.

- **Loops:** Employed in displaying all submitted applications in a TextArea.

- **String Manipulation:**

Formats and displays job and application details using string operations, ensuring clear and readable output

- **Event Handling:**

Action listeners are added to buttons to perform tasks such as posting jobs, viewing applications, and submitting applications.

- **Collections Framework:**

- **ArrayList:** Used to maintain lists of Job and Application objects for job postings and submitted applications.

CHAPTER 2

PROJECT METHODOLOGY

Proposed Work

Develop a Job Portal application that allows users to post job listings and apply for jobs through an interactive and user-friendly interface. The system should be capable of managing job postings and job applications with robust data handling.

- **Requirements Gathering**

Identify the core features: job posting, job searching by domain, job application submission, and viewing submitted applications.

- **System Design**

Outline the workflow for job posting, job searching, and applicationsubmission. Create a modular design that separates the responsibilities of job data management and application handling.

- **Project Setup**

Use the Job class to represent job listings and manage job data .Create an Application class to represent job applications and store applicant details.

- **Database Setup**

Utilizein-memory data structures such as Array List to store job and application data for simplicity and quick access during runtime.

- **Backend Development**

Implement methodsin theJobPortalAWTclassforjobposting, job searching by domain, and application submissions.

- **User Interface Development**

Design the GUI using AWT components like Frame, Label, Button, TextField,andTextArea.Create separateframesforjobseekersandemployerswith interactive forms and dialog boxes for user interaction and feedback.

- **Integration**

Combine the backend logic with the GUI components to ensure smooth user navigation and seamless operation for posting jobs, applying for jobs, and viewing job applications.

- **Testing**

Test key functionalities such as job posting, job searching, and application submission to ensure data is handled correctly. Validate user input error handling and check the responsiveness of the user interface.

- **Deployment**

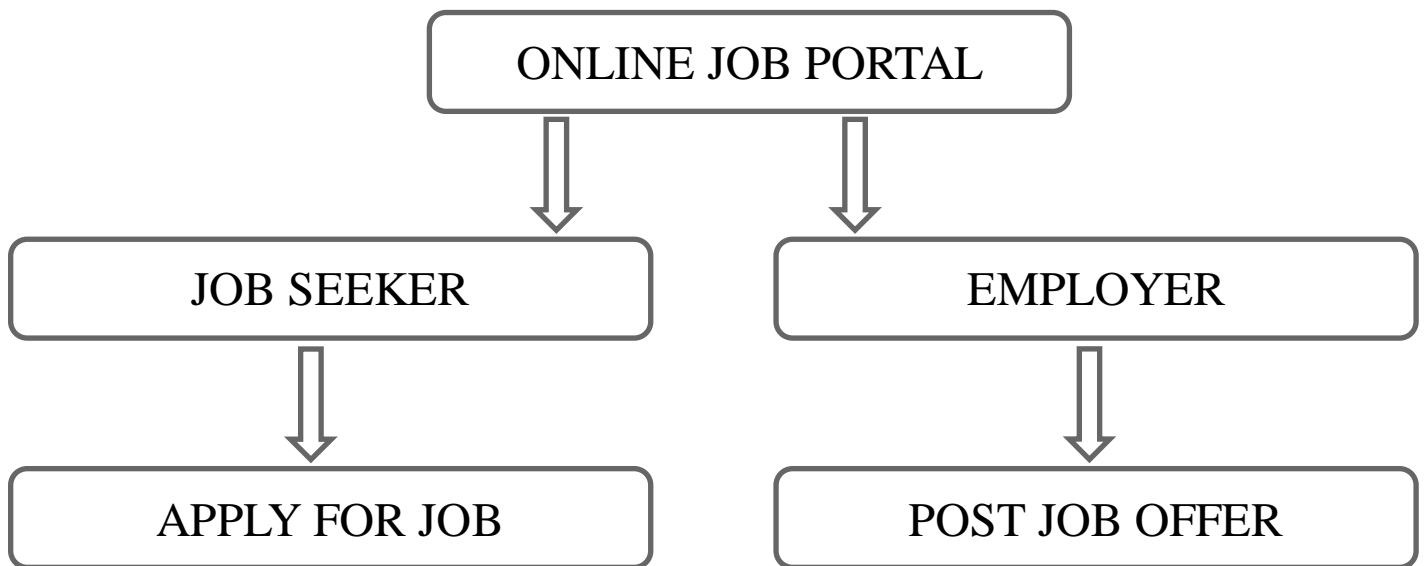
Package the application as a runnable Java program (.jar) for easy distribution and execution..

- **Maintenance**

Update features or address any bugs based on user feedback. Continuously monitor the application for performance issues and optimize for an enhanced user experience.

Block Diagram

BLOCK DIAGRAM FOR ONLINE JOB PORTAL



CHAPTER 3

MODULE DESCRIPTION

Main Menu Module

- **Purpose:** Provides the initial interface for users to choose between the "Employer" and "Job Seeker" portals.
- **Components:**
 - Frame, Button, Label.
- **Interaction:**
 - Direct users to either the employer or job seeker interface.

Employer Portal Module

- **Purpose:** Allow employers to post new job listings and view applications submitted for their jobs.
- **Components:**
 - Frame, TextField, Button, Label, TextArea.
- **Interaction:**
 - Facilitates job posting, viewing applications, and data validation.
 - `postJob()`: Adds job postings to the jobs list, `view Applications()`: Displays submitted applications for the employer

Job Seeker Portal Module

- **Purpose:** Provides job seekers with the ability to search for jobs based on domain and submit job applications.
- **Components:**
 - Frame, Choice, TextArea, Button, Label.
- **Interaction:**
 - Allows jobseekers to view available jobs and submit applications.
 - `searchJobsByDomain()`: Filters and displays jobs based on the selected domain.
 - `applyForJob()`: Submits a job application after validation.

Application Handling Module

- **Purpose :** Manages job applications submitted by jobseekers..
- **Components:**
 - List<Application>.
- **Interaction:**
 - Stores and displays job applications.

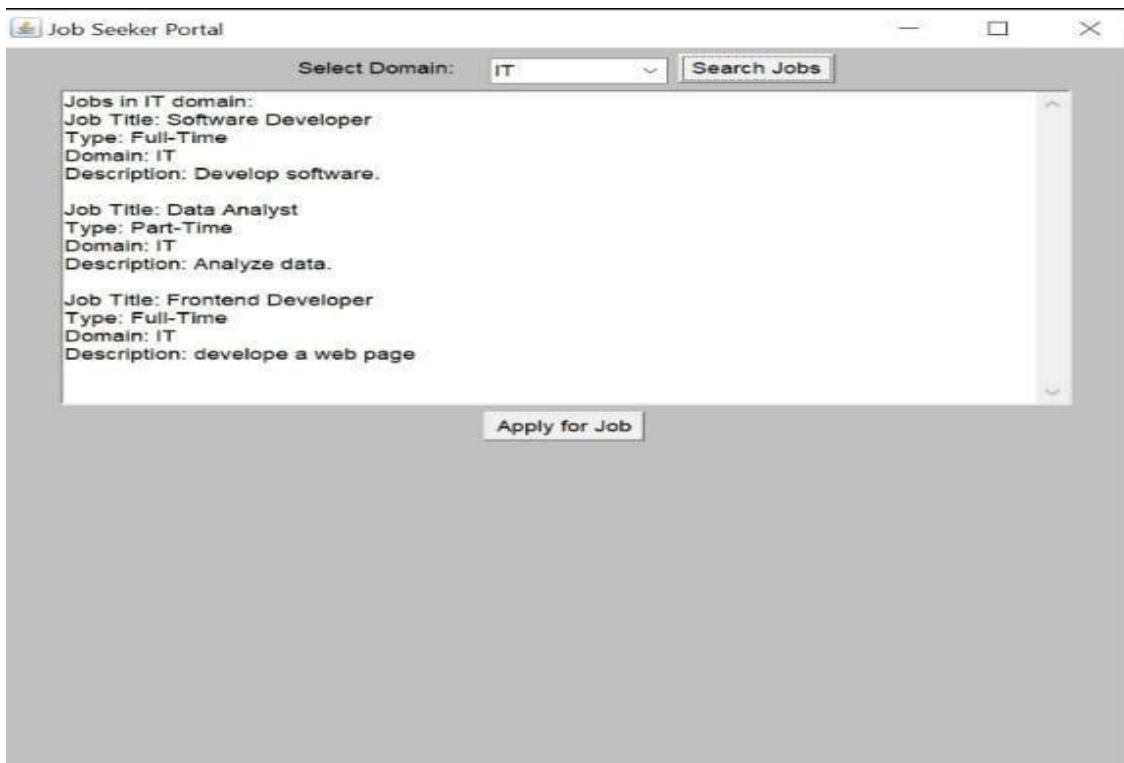
CHAPTER 4

RESULTS AND DISCUSSION

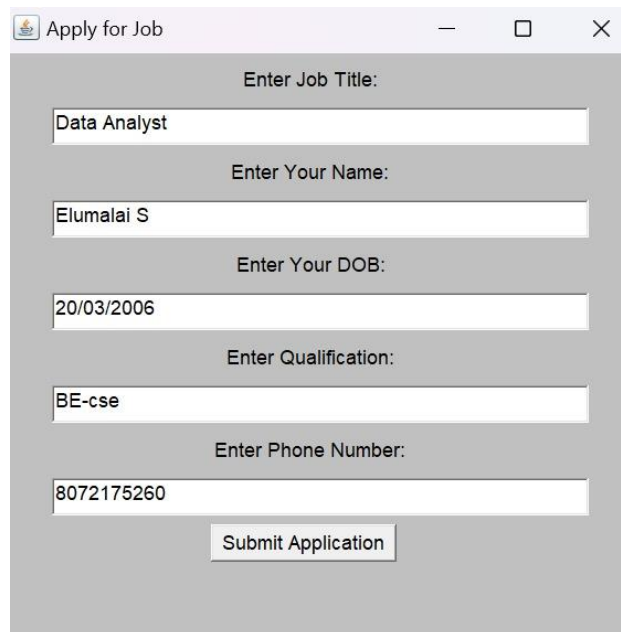
WELCOME TO JOB S0EEKER :



SEARCH JOB WITH SELECTING DOMAIN :



APPLY FOR THE JOB :



Apply for Job

Enter Job Title:

Data Analyst

Enter Your Name:

Elumalai S

Enter Your DOB:

20/03/2006

Enter Qualification:

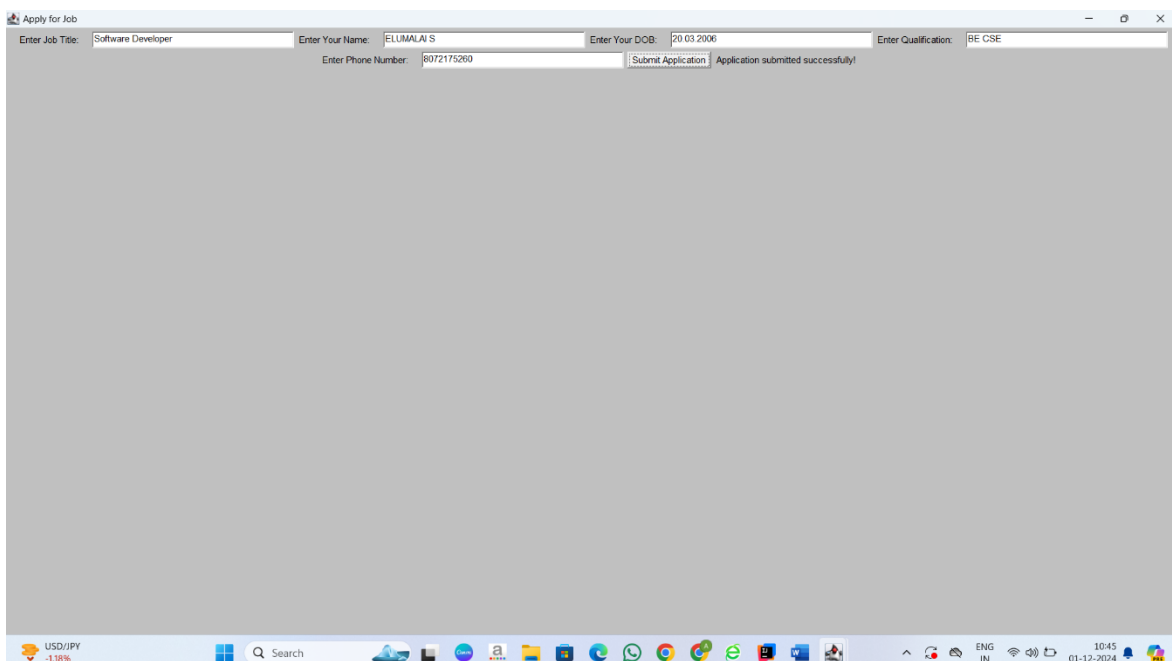
BE-cse

Enter Phone Number:

8072175260

Submit Application

APPLICATION SUCCESSFULLY SUBMITTED :



Apply for Job

Enter Job Title: Software Developer

Enter Your Name: ELUMALAI S

Enter Your DOB: 20/03/2006

Enter Qualification: BE CSE

Enter Phone Number: 8072175260

Submit Application

Application submitted successfully!

USD/INR -1.18%

Search

ENG IN

10:45 01-12-2024

CHAPTER 5

CONCLUSION

The Job Portal project effectively demonstrates the core principles of Java programming and object-oriented design, creating a platform that supports essential job- seeking and job-posting functionalities. Through the integration of employer and job seeker portals, the application facilitates job posting, application submissions, job search by domain, and viewing of submitted applications. By employing Java's Abstract Window Tool kit (AWT) for the GUI, the system ensures a user-friendly and visually engaging interface. The project showcases the practical application of modular programming, data management with collections, event handling, and error handling for a seamless user experience.

Overall, this Job Portal project serves as a robust example of how Java can be utilized to build interactive, real-world applications. It provides a comprehensive learning resource for students to understand software development concepts and serves as a solid foundation for developing more complex, feature-rich systems in the future.

REFERENCES:

1. "Java Programming : A Beginner's Guide" by Herbert Schildt.
2. "Data Structures and Algorithms in Java "by Robert Lafore.
3. Oracle Java Documentation : <https://docs.oracle.com/javase/>
4. Stack Overflow : Java- related discussions on Hash Maps, Java classes, and loops.

APPENDIX

(Project Source Code)

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.List;
public class Main extends Frame {

// Job class to store job details
    static class Job {
        String title;
        String type;
        String description;
        String domain;
        public Job(String title, String type, String description, String domain) {
            this.title = title;
            this.type = type;
            this.description = description;
            this.domain = domain;
        }
        @Override
        public String toString() {
            return "Job Title: " + title + "\nType: " + type + "\nDomain: " + domain +
"\nDescription: "
                + description;
        }
    }

// Application class to store application details
    static class Application {
        String jobTitle;
        String applicantName;
        String dob;
        String qualification;
        String phone;
        public Application(String jobTitle, String applicantName, String dob, String
qualification,String phone) {
            this.jobTitle = jobTitle;
            this.applicantName = applicantName;
            this.dob = dob;
            this.qualification = qualification;
            this.phone = phone;
        }
        @Override
        public String toString() {
```

```

        return "Job Title: " + jobTitle + "\nApplicant Name: " + applicantName +
"\nDOB: " + dob
        +
        "\nQualification: " + qualification + "\nPhone: " + phone;
    }
}
static List<Job> jobs = new ArrayList<>();
static List<Application> applications = new ArrayList<>();
public Main() {
    populateJobs();
    setTitle("Job Portal");
    setSize(500, 500);
    setLayout(new FlowLayout());
    setBackground(Color.LIGHT_GRAY);
    Label welcomeLabel = new Label("Welcome to the Job Portal!");
    welcomeLabel.setFont(new Font("Arial", Font.BOLD, 16));
    add(welcomeLabel);
    Button employerButton = new Button("Employer");
    Button jobSeekerButton = new Button("Job Seeker");
    add(employerButton);
    add(jobSeekerButton);
    employerButton.addActionListener(e -> handleEmployer());
    jobSeekerButton.addActionListener(e -> handleJobSeeker());
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we) {
            System.exit(0);
        }
    });
    setVisible(true);
}
private void handleEmployer() {
    Frame employerFrame = new Frame("Employer Portal");
    employerFrame.setSize(500, 600);
    employerFrame.setLayout(new FlowLayout());
    employerFrame.setBackground(Color.LIGHT_GRAY);

// Job posting section
    Label postJobLabel = new Label("Post a New Job:");
    Label titleLabel = new Label("Enter Job Title:");
    TextField titleField = new TextField(30);
    Label typeLabel = new Label("Full-Time or Part-Time:");
    TextField typeField = new TextField(30);
    Label descriptionLabel = new Label("Enter Job Description:");
    TextField descriptionField = new TextField(30);
    Label domainLabel = new Label("Enter Job Domain:");
    TextField domainField = new TextField(30);
    Button postJobButton = new Button("Post Job");
    Label postResponseLabel = new Label();
    postJobButton.addActionListener(e -> {
        String title = titleField.getText();
        String type = typeField.getText();

```

```

String description = descriptionField.getText();

String domain = domainField.getText();
    if (!title.isEmpty() && !type.isEmpty() && !description.isEmpty() &&
        !domain.isEmpty()) {
        jobs.add(new Job(title, type, description, domain));
        postResponseLabel.setText("Job posted successfully!");
        titleField.setText("");
        typeField.setText("");
        descriptionField.setText("");
        domainField.setText("");
    } else {
        postResponseLabel.setText("All fields are required to post a job.");
    }
});

// View applications section
Label viewApplicationsLabel = new Label("View Applications:");
Button viewApplicationsButton = new Button("View Applications");
TextArea applicationsArea = new TextArea(15, 40);
viewApplicationsButton.addActionListener(e -> {
    applicationsArea.setText("Submitted Applications:\n");
    if (applications.isEmpty()) {
        applicationsArea.setText("No applications submitted yet.");
    } else {
        for (Application application : applications) {
            applicationsArea.append(application.toString() + "\n\n");
        }
    }
});
employerFrame.add(postJobLabel);
employerFrame.add(titleLabel);
employerFrame.add(titleField);
employerFrame.add(typeLabel);
employerFrame.add(typeField);
employerFrame.add(descriptionLabel);
employerFrame.add(descriptionField);
employerFrame.add(domainLabel);
employerFrame.add(domainField);
employerFrame.add(postJobButton);
employerFrame.add(postResponseLabel);
employerFrame.add(viewApplicationsLabel);
employerFrame.add(viewApplicationsButton);
employerFrame.add(applicationsArea);
employerFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        employerFrame.dispose();
    }
});
employerFrame.setVisible(true);
}

private void handleJobSeeker() {

```



```

Frame jobSeekerFrame = new Frame("Job Seeker Portal");
jobSeekerFrame.setSize(600, 600);

jobSeekerFrame.setLayout(new FlowLayout());
jobSeekerFrame.setBackground(Color.LIGHT_GRAY);
Label domainLabel = new Label("Select Domain:");
Choice domainChoice = new Choice();
domainChoice.add("IT");
domainChoice.add("Healthcare");
domainChoice.add("Education");
domainChoice.add("Finance");
domainChoice.add("Engineering");
domainChoice.add("Marketing");
Button searchButton = new Button("Search Jobs");
TextArea jobArea = new TextArea(15, 50);
searchButton.addActionListener(e -> {
    String selectedDomain = domainChoice.getSelectedItemAt();
    jobArea.setText("Jobs in " + selectedDomain + " domain:\n");
    boolean found = false;
    for (Job job : jobs) {
        if (job.domain.equalsIgnoreCase(selectedDomain)) {
            jobArea.append(job.toString() + "\n\n");
            found = true;
        }
    }
    if (!found) {
        jobArea.setText("No jobs available in " + selectedDomain + " domain.");
    }
});
Button applyButton = new Button("Apply for Job");
applyButton.addActionListener(e -> {
    Frame applyFrame = new Frame("Apply for Job");
    applyFrame.setSize(400, 400);
    applyFrame.setLayout(new FlowLayout());
    applyFrame.setBackground(Color.LIGHT_GRAY);
    Label titleLabel = new Label("Enter Job Title:");
    TextField titleField = new TextField(30);
    Label nameLabel = new Label("Enter Your Name:");
    TextField nameField = new TextField(30);
    Label dobLabel = new Label("Enter Your DOB:");
    TextField dobField = new TextField(30);
    Label qualificationLabel = new Label("Enter Qualification:");
    TextField qualificationField = new TextField(30);
    Label phoneLabel = new Label("Enter Phone Number:");
    TextField phoneField = new TextField(30);
    Button submitButton = new Button("Submit Application");
    Label responseLabel = new Label();
    submitButton.addActionListener(ae -> {
        String jobTitle = titleField.getText();
        String name = nameField.getText();
        String dob = dobField.getText();

```

```

String qualification = qualificationField.getText();
String phone = phoneField.getText();
if (!jobTitle.isEmpty() && !name.isEmpty() && !dob.isEmpty() &&
    !qualification.isEmpty() && !phone.isEmpty()) {

boolean jobExists = jobs.stream().anyMatch(job ->
    job.title.equalsIgnoreCase(jobTitle));
    if (jobExists) {
        applications.add(new Application(jobTitle, name, dob, qualification,
phone));
        responseLabel.setText("Application submitted successfully!");
    } else {
        responseLabel.setText("Job title not found.");
    }
} else {
    responseLabel.setText("All fields are required.");
}
});
applyFrame.add(titleLabel);
applyFrame.add(titleField);
applyFrame.add(nameLabel);
applyFrame.add(nameField);
applyFrame.add(dobLabel);
applyFrame.add(dobField);
applyFrame.add(qualificationLabel);
applyFrame.add(qualificationField);
applyFrame.add(phoneLabel);
applyFrame.add(phoneField);
applyFrame.add(submitButton);
applyFrame.add(responseLabel);
applyFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        applyFrame.dispose();
    }
});
applyFrame.setVisible(true);
});
jobSeekerFrame.add(domainLabel);
jobSeekerFrame.add(domainChoice);
jobSeekerFrame.add(searchButton);
jobSeekerFrame.add(jobArea);
jobSeekerFrame.add(applyButton);
jobSeekerFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        jobSeekerFrame.dispose();
    }
});
jobSeekerFrame.setVisible(true);
}

private void populateJobs() {

```

```

jobs.add(new Job("Software Developer", "Full-Time", "Develop software.", "IT"));
jobs.add(new Job("Data Analyst", "Part-Time", "Analyze data.", "IT"));
jobs.add(new Job("Nurse", "Full-Time", "Provide healthcare.", "Healthcare"));
jobs.add(new Job("Teacher", "Full-Time", "Teach students.", "Education"));
jobs.add(new Job("Accountant", "Full-Time", "Manage accounts.", "Finance"));
jobs.add(new Job("Civil Engineer", "Full-Time", "Design infrastructure.",
"Engineering"));
jobs.add(new Job("Marketing Manager", "Full-Time", "Manage campaigns.",
"Marketing"));
}
public static void main(String[] args) {
    new Main();
}
}

```