ONLINE JOB PORTAL

A PROJECT REPORT

Submitted by

MUKESH M 2303811724321071

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"ONLINE JOB PORTAL"** is the bonafide work of **MUKESH M 2303811724321071** who carried out the project work during the academic year 2024 - 2025 under my supervision.

[illegible]

Signature

Dr. T. AVUDAIAPPAN M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

[illegible]

Signature

Mrs. S. GEETHA M.E.,

SUPERVISOR,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24 .

[illegible]

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on "**ONLINE JOB PORTAL** " is the result of original work done by me and best of my knowledge, similar work has not been submitted to "**ANNA UNIVERSITY CHENNAI**" for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING.**

Signature

[illegible]

MUKESH M

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

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

I extend our sincere acknowledgement and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.,** for having provided 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 our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.,** Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D**., Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E**., Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

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

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conductive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The **Online Job Portal** project simplifies recruitment by offering a seamless platform for interaction between **job seekers** and **employers**. It allows **job seekers** to register, manage their profiles, and search for jobs based on their skills and preferences. On the other hand, **employers** can register their company, post job openings, and (in future updates) view applications.This project uses **Java Programming**, demonstrating the power of **Object-Oriented Programming (OOP)** concepts like encapsulation and modularity. Data is managed dynamically using Java's **Collections Framework**, including **HashMap** and **ArrayList**, to store and retrieve information efficiently.The project adopts a **menu-driven console interface**, making it interactive for users while maintaining simplicity. Its design focuses on modularity and scalability, ensuring adaptability for future enhancements, such as integrating graphical interfaces or databases. This project addresses the inefficiencies of traditional job search methods, providing an accessible and automated solution.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Online Job Portal Application is a Java-based project designed to connect job seekers and employers efficiently through an intuitive graphical user interface built with the Abstract Window Toolkit (AWT). The application allows job seekers to register their profiles by providing personal details and skillsets, enabling them to log in securely and explore job listings tailored to their qualifications. Employers can also register to post job openings and specify the skills required, allowing them to identify suitable candidates from the portal. The platform ensures seamless interaction between job seekers and employers, simplifying the recruitment process while providing a user-friendly environment for both parties to manage profiles, job postings, and applications effectively.

1.2 OBJECTIVE

The primary objective of the Online Job Portal System is to develop a robust, scalable, and interactive application that bridges the gap between job seekers and employers in the hiring process. The system aims to:

1. Automate recruitment tasks: Reducing manual intervention for employers while simplifying job searches for seekers.
2. Enhance user convenience: Providing an intuitive interface for tasks like registration, job posting, profile updates, and job searching.
3. Demonstrate the power of Java programming: Utilizing advanced features such as Object-Oriented Programming (OOP), data encapsulation, and Java collections for real-time data handling.

4. Address traditional inefficiencies: Replacing outdated methods of job search and recruitment with a streamlined, digital solution.

5. Facilitate personalized recommendations: Implementing algorithms to match job seekers with relevant job opportunities based on their profiles, skills, and preferences.

6. Strengthen data security and privacy: Ensuring the protection of user information through robust encryption and secure data handling practices.

7. Improve accessibility: Designing the application to be mobile-responsive and accessible to users with different devices and varying levels of technical proficiency.

8. Enable real-time communication: Incorporating features like chat systems or messaging capabilities for direct interaction between job seekers and employers.

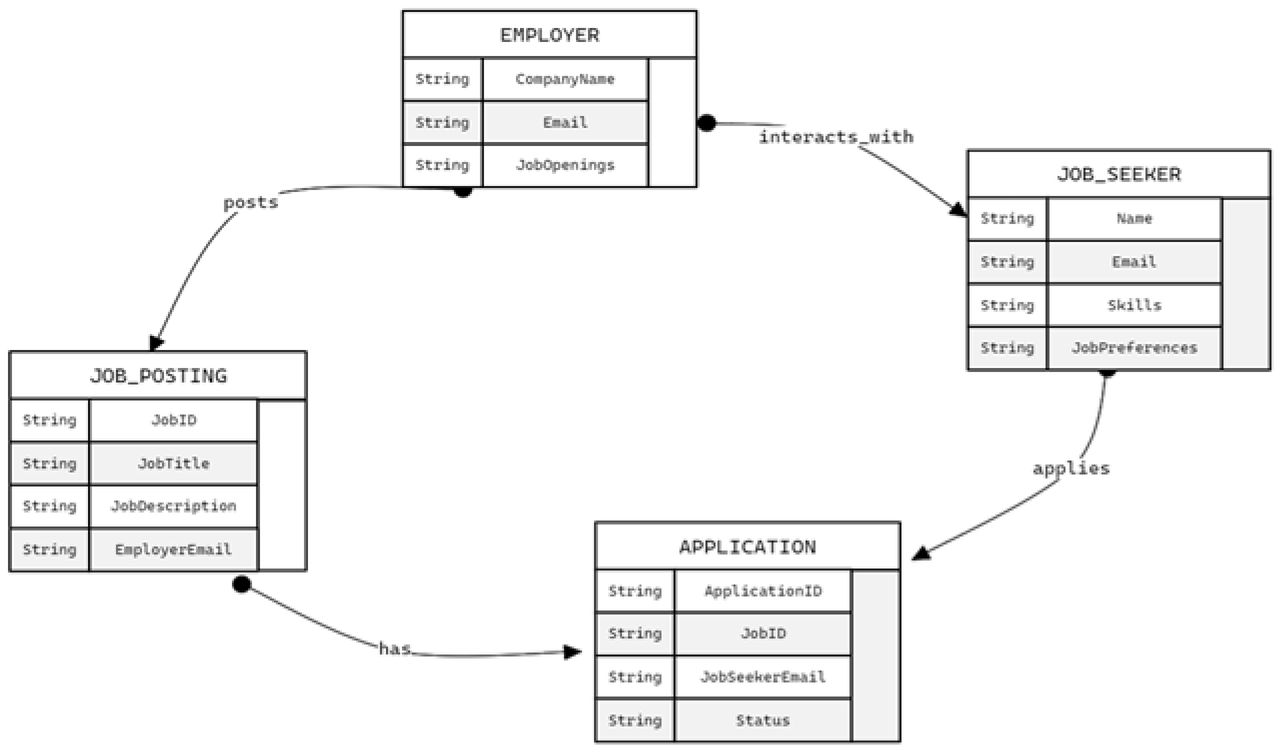
CHAPTER 2
PROJECT METHODOLOGY

2.1 PROPOSED WORK

The project aims to create an intuitive, efficient, and user-friendly job portal system. The core objective is to develop a Java-based application that provides seamless interaction between job seekers and employers. The methodology focuses on:

1. Understanding Requirements:
    - Identifying needs for user registration, secure login, and job management.
    - Analyzing existing systems for inspiration and limitations.
2. System Design:
    - Architecting modules for functionality, scalability, and ease of use.
3. Implementation:
    - Writing modular Java code.
    - Employing data structures like HashMap and ArrayList for data storage.
4. Testing and Validation:
    - Testing each module independently for functionality and integration.
    - Ensuring error-free data processing.
5. Deployment and Feedback:
    - Providing a functional graphical user interface (GUI) for end users.

2.2 BLOCK DIAGRAM



CHAPTER 3
JAVA PROGRAMMING CONCEPTS

3.1 OBJECT-ORIENTED PROGRAMMING (OOP):

- Classes and Objects: Represent job seekers, employers, and system functionalities.
- Encapsulation: Securely manage user data with private fields and public getters/setters.
- Inheritance: Enhance modularity and code reusability (extendable for future updates).

2. Java Collections Framework:
   - HashMap: Efficiently stores user information like credentials and profiles.
   - ArrayList: Dynamically manages job postings, skills, and preferences.
3. Control Flow Mechanisms:
   - Loops and Conditional Statements: Facilitate seamless navigation within menus.
   - Switch Case: Organizes user choices in the main and module-specific menus.
4. Input Handling with Scanner:

Captures and validates user input for actions like registration, login, and job search.

3.2 GUI COMPONENTS

1. Frame: Main window container.
2. Label: Displays text.
3. Button: Triggers actions like registration or login.
4. TextField: Accepts user input.

CHAPTER 4
MODULE DESCRIPTION

4.1 MAIN APPLICATION MODULE

4.1.1 Objective

The Main Application Module serves as the entry point for the system, providing an intuitive interface for users to navigate to their respective functionalities (Job Seeker or Employer). It ensures smooth navigation and user-friendly interaction.

4.1.2 Design

- Graphical Components Used:
  - Frame: The main container for the user interface.
  - Button: Interactive elements for selecting options like "Job Seeker" or "Employer."
  - Label: Provides static text instructions.
- Navigation Workflow:
  1. User launches the application.
  2. Main menu displays options.
  3. Based on the user's choice, corresponding modules (Job Seeker or Employer) are loaded.

4.1.3 Usage

- This module acts as the first interaction point, ensuring users have a clear understanding of the application’s offerings

4.2 JOB SEEKER MODULE

4.2.1 Objective

This module allows job seekers to register, log in, and explore job opportunities tailored to their skillsets.

4.2.2 Submodules

1. Registration Submodule:
    - Objective: Captures the personal and professional details of job seekers.
    - Features:
        - Stores Name, Email, Password, and Skills.
        - Validates inputs (e.g., email format, password strength).
    - Workflow:
    1. User enters details in the form.
    2. Data is validated and saved in the system.

4.3 EMPLOYER MODULE

4.3.1 Objective

The Employer Module provides employers with functionalities to register their company, log in securely, and post job openings.

4.3.2 Submodules

1. Employer Registration:
    - Objective: Captures company details for new employers.
    - Workflow:
        - User inputs company name, email, and password.
        - Data is stored securely in the system.

2. Login System:
    - Objective: Authenticates employers using stored credentials.
    - Features: Validates credentials and allows access to job-posting functionalities.
3. Job Posting Submodule:
    - Objective: Enables employers to list job openings and specify skill requirements.
    - Features:
        - Supports multi-line job descriptions.
        - Allows tagging of required skills.

4.4 UTILITY MODULE

4.4.1 Objective

This module provides supporting functionalities like data storage, error handling, and message displays.

4.4.2 Features

1. Data Storage: Uses HashMap for in-memory storage of job seeker and employer data.
2. Message Handling: Displays success or error messages using dialog boxes.

CHAPTER 5
CONCLUSION

SUMMARY OF ACHIEVEMENTS

The **Online Job Portal Application** successfully addresses the needs of job seekers and employers through the following:

1. **For Job Seekers:**
    - A streamlined registration and login process.
    - Tailored job recommendations based on skills.
2. **For Employers:**
    - Easy-to-use interfaces for job postings.
    - Secure data handling and access.
3. **Overall System:**
    - Dynamic and scalable design, ready for future enhancements like database integration and advanced job matching algorithms.

LIMITATIONS

1. No persistent data storage (current implementation uses in-memory storage).
2. Limited GUI aesthetics (due to AWT's constraints).

FUTURE SCOPE

1. **Database Integration:** Use SQL or NoSQL databases for better scalability.
2. **Enhanced Matching Algorithm:** Implement machine learning models for personalized job recommendations.
3. **Mobile Compatibility:** Extend the application for Android and iOS platforms using JavaFX or Kotlin.
4. **Analytics Dashboard:** Add analytics for employers to track job postings and applicant statistics.

REFERENCES:

The development of this project involved the use of various resources to understand Java programming principles and implement best practices. The references include:

Books

1. *Core Java Volume I: Fundamentals* by Cay S. Horstmann and Gary Cornell
   - Comprehensive guide to Java fundamentals, including OOP concepts and collections.
2. *Effective Java* by Joshua Bloch
   - Focus on practical programming techniques and Java best practices.

Online Tutorials

1. **W3Schools Java Tutorials**: https://www.w3schools.com/java/
   - For understanding basic to advanced Java programming concepts.
2. **GeeksforGeeks Java Programming**: https://www.geeksforgeeks.org/java/
   - Detailed explanations of Java collections, object-oriented principles, and practical examples.

Documentation

1. **Oracle Java Documentation**: https://docs.oracle.com/javase/tutorial/
   - Official documentation for Java programming and the Collections
   - Framework.
2. **Java SE 17 API Documentation**: https://docs.oracle.com/en/java/javase/17/docs/api/
   - Detailed reference for Java APIs used in the project.

Videos

1. **Programming with Mosh – Java Full Course** (YouTube): https://www.youtube.com/watch?v=grEKMHGYyns
   - Detailed video tutorials covering Java basics to advanced topics.
2. **CodeWithHarry – Java for Beginners** (YouTube): https://www.youtube.com/CodeWithHarry
   - Simplified explanation of Java programming concepts.

APPENDICES

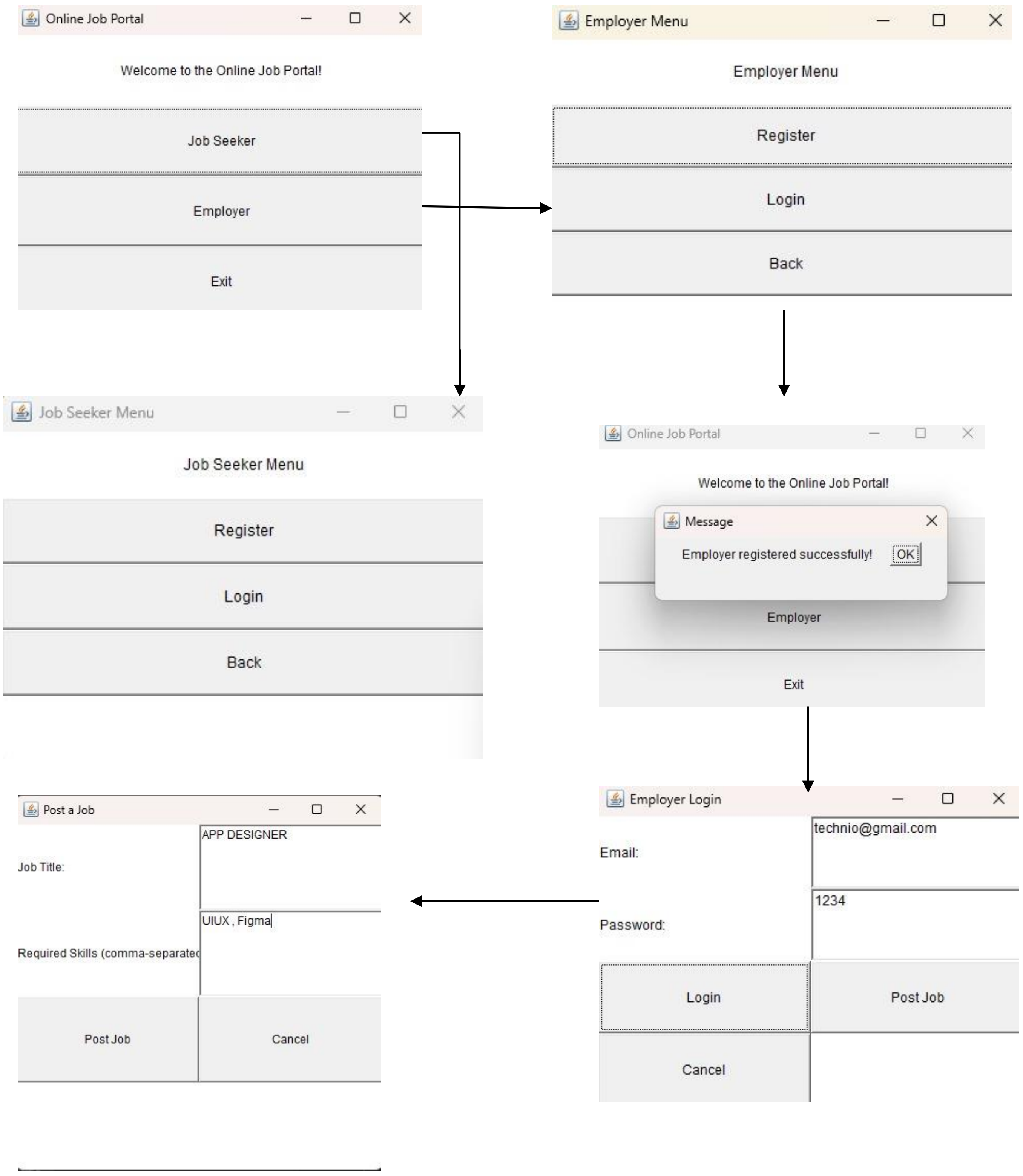
APPENDIX A – SOURCE CODE

```
import java.awt.*;
import java.util.*;
import java.util.List ;
class OnlineJobs extends Frame {

    private static Map<String, JobSeeker> jobSeekers = new HashMap<>();
    private static Map<String, Employer> employers = new HashMap<>();

    public OnlineJobs() {
        // Main Menu
        setTitle("Online Job Portal");
        setSize(400, 300);
        setLayout(new GridLayout(4, 1));

        Label welcomeLabel = new Label("Welcome to the Online Job Portal!",
Label.CENTER);
        add(welcomeLabel);

        Button jobSeekerButton = new Button("Job Seeker");
        Button employerButton = new Button("Employer");
        Button exitButton = new Button("Exit");

        add(jobSeekerButton);
        add(employerButton);
        add(exitButton);

        // Button Actions
        jobSeekerButton.addActionListener(e -> openJobSeekerMenu());
        employerButton.addActionListener(e -> openEmployerMenu());
        exitButton.addActionListener(e -> System.exit(0));

        setVisible(true);
    }

    private void openJobSeekerMenu() {
        Frame jobSeekerMenu = new Frame("Job Seeker Menu");
        jobSeekerMenu.setSize(400, 300);
        jobSeekerMenu.setLayout(new GridLayout(5, 1));

        Label jobSeekerLabel = new Label("Job Seeker Menu", Label.CENTER);
        jobSeekerMenu.add(jobSeekerLabel);

        Button registerButton = new Button("Register");
        Button loginButton = new Button("Login");
        Button backButton = new Button("Back");

        jobSeekerMenu.add(registerButton);
        jobSeekerMenu.add(loginButton);
        jobSeekerMenu.add(backButton);

        // Button Actions
        registerButton.addActionListener(e -> openJobSeekerRegistration());
        loginButton.addActionListener(e -> openJobSeekerLogin());
        backButton.addActionListener(e -> jobSeekerMenu.dispose());

        jobSeekerMenu.setVisible(true);
    }

    private void openJobSeekerRegistration() {
        Frame registrationFrame = new Frame("Job Seeker Registration");
        registrationFrame.setSize(400, 400);
        registrationFrame.setLayout(new GridLayout(6, 2));

        Label nameLabel = new Label("Name:");
        TextField nameField = new TextField();
        Label emailLabel = new Label("Email:");
        TextField emailField = new TextField();
        Label passwordLabel = new Label("Password:");
        TextField passwordField = new TextField();
        Label skillsLabel = new Label("Skills (comma-separated):");
        TextField skillsField = new TextField();

        Button registerButton = new Button("Register");
        Button cancelButton = new Button("Cancel");

        registrationFrame.add(nameLabel);
        registrationFrame.add(nameField);
        registrationFrame.add(emailLabel);
        registrationFrame.add(emailField);
        registrationFrame.add(passwordLabel);
        registrationFrame.add(passwordField);
        registrationFrame.add(skillsLabel);
        registrationFrame.add(skillsField);
        registrationFrame.add(registerButton);
        registrationFrame.add(cancelButton);

        // Button Actions
        registerButton.addActionListener(e -> {
            String name = nameField.getText();
            String email = emailField.getText();
            String password = passwordField.getText();
            String skillsStr = skillsField.getText();

            List<String> skills = new ArrayList<>();
            for (String skill : skillsStr.split(",")) {
                skills.add(skill.trim());
            }

            JobSeeker seeker = new JobSeeker(name, email, password, skills);
            jobSeekers.put(email, seeker);

            showMessage("Job Seeker registered successfully!");
            registrationFrame.dispose();
        });

        cancelButton.addActionListener(e -> registrationFrame.dispose());

        registrationFrame.setVisible(true);
    }

    private void openJobSeekerLogin() {
        Frame loginFrame = new Frame("Job Seeker Login");
        loginFrame.setSize(400, 300);
        loginFrame.setLayout(new GridLayout(5, 1));

        Label emailLabel = new Label("Email:");
        TextField emailField = new TextField();
        Label passwordLabel = new Label("Password:");
        TextField passwordField = new TextField();

        Button loginButton = new Button("Login");
        Button jobPreferencesButton = new Button("View Job Preferences");
        Button cancelButton = new Button("Cancel");

        loginFrame.add(emailLabel);
        loginFrame.add(emailField);
        loginFrame.add(passwordLabel);
        loginFrame.add(passwordField);
        loginFrame.add(loginButton);
        loginFrame.add(jobPreferencesButton);
        loginFrame.add(cancelButton);

        jobPreferencesButton.setEnabled(false);

        // Button Actions
        loginButton.addActionListener(e -> {
            String email = emailField.getText();
            String password = passwordField.getText();

            if (jobSeekers.containsKey(email) &&
jobSeekers.get(email).getPassword().equals(password)) {
                showMessage("Login successful!");
                jobPreferencesButton.setEnabled(true);
            } else {
                showMessage("Invalid email or password. Please try again.");
            }
        });

        cancelButton.addActionListener(e -> loginFrame.dispose());

        jobPreferencesButton.addActionListener(e ->
openJobPreferences(emailField.getText()));

        loginFrame.setVisible(true);
    }

    private void openJobPreferences(String jobSeekerEmail) {
        JobSeeker jobSeeker = jobSeekers.get(jobSeekerEmail);
        if (jobSeeker == null) {
            showMessage("You need to log in first!");
            return;
        }

        Frame preferencesFrame = new Frame("Matching Jobs");
        preferencesFrame.setSize(400, 400);
        preferencesFrame.setLayout(new GridLayout(0, 1));

        Label headerLabel = new Label("Matching Jobs:");
        preferencesFrame.add(headerLabel);

        List<String> seekerSkills = jobSeeker.getSkills();

        boolean matchFound = false;
        for (Employer employer : employers.values()) {
            for (Job job : employer.getJobs()) {
                for (String skill : seekerSkills) {
                    if (job.getRequiredSkills().contains(skill)) {
                        Label jobLabel = new Label("Company
name:"+employer.getCompanyName() + " Offering job:" + job.getTitle());
                        preferencesFrame.add(jobLabel);
                        matchFound = true;
                        break;
                    }
                }
            }
        }

        if (!matchFound) {
            preferencesFrame.add(new Label("No matching jobs found."));
        }

        Button closeButton = new Button("Close");
        closeButton.addActionListener(e -> preferencesFrame.dispose());
        preferencesFrame.add(closeButton);

        preferencesFrame.setVisible(true);
    }

    private void openEmployerMenu() {
        Frame employerMenu = new Frame("Employer Menu");
        employerMenu.setSize(400, 300);
        employerMenu.setLayout(new GridLayout(5, 1));

        Label employerLabel = new Label("Employer Menu", Label.CENTER);
        employerMenu.add(employerLabel);

        Button registerButton = new Button("Register");
        Button loginButton = new Button("Login");
        Button backButton = new Button("Back");

        employerMenu.add(registerButton);
        employerMenu.add(loginButton);
        employerMenu.add(backButton);

        // Button Actions
        registerButton.addActionListener(e -> openEmployerRegistration());
        loginButton.addActionListener(e -> openEmployerLogin());
        backButton.addActionListener(e -> employerMenu.dispose());

        employerMenu.setVisible(true);
    }

    private void openEmployerRegistration() {
        Frame registrationFrame = new Frame("Employer Registration");
        registrationFrame.setSize(400, 300);
        registrationFrame.setLayout(new GridLayout(5, 2));

        Label companyNameLabel = new Label("Company Name:");
        TextField companyNameField = new TextField();
        Label emailLabel = new Label("Email:");
        TextField emailField = new TextField();
        Label passwordLabel = new Label("Password:");
        TextField passwordField = new TextField();

        Button registerButton = new Button("Register");
        Button cancelButton = new Button("Cancel");

        registrationFrame.add(companyNameLabel);
        registrationFrame.add(companyNameField);
        registrationFrame.add(emailLabel);
        registrationFrame.add(emailField);
        registrationFrame.add(passwordLabel);
        registrationFrame.add(passwordField);
        registrationFrame.add(registerButton);
        registrationFrame.add(cancelButton);

        // Button Actions
        registerButton.addActionListener(e -> {
            String companyName = companyNameField.getText();
            String email = emailField.getText();
            String password = passwordField.getText();
```

```
            Employer employer = new Employer(companyName, email, password);
            employers.put(email, employer);

            showMessage("Employer registered successfully!");
            registrationFrame.dispose();
        });

        cancelButton.addActionListener(e -> registrationFrame.dispose());

        registrationFrame.setVisible(true);
    }

    private void openEmployerLogin() {
        Frame loginFrame = new Frame("Employer Login");
        loginFrame.setSize(400, 300);
        loginFrame.setLayout(new GridLayout(4, 2));

        Label emailLabel = new Label("Email:");
        TextField emailField = new TextField();
        Label passwordLabel = new Label("Password:");
        TextField passwordField = new TextField();

        Button loginButton = new Button("Login");
        Button postJobButton = new Button("Post Job");
        Button cancelButton = new Button("Cancel");

        loginFrame.add(emailLabel);
        loginFrame.add(emailField);
        loginFrame.add(passwordLabel);
        loginFrame.add(passwordField);
        loginFrame.add(loginButton);
        loginFrame.add(postJobButton);
        loginFrame.add(cancelButton);

        postJobButton.setEnabled(false);

        // Button Actions
        loginButton.addActionListener(e -> {
            String email = emailField.getText();
            String password = passwordField.getText();

            if (employers.containsKey(email) &&
employers.get(email).getPassword().equals(password)) {
                showMessage("Employer login successful!");
                postJobButton.setEnabled(true);
            } else {
                showMessage("Invalid email or password. Please try again.");
            }
        });

        cancelButton.addActionListener(e -> loginFrame.dispose());

        postJobButton.addActionListener(e -> openPostJob(emailField.getText()));

        loginFrame.setVisible(true);
    }

    private void openPostJob(String employerEmail) {
        Employer employer = employers.get(employerEmail);
        if (employer == null) {
            showMessage("You need to log in first!");
            return;
        }

        Frame postJobFrame = new Frame("Post a Job");
        postJobFrame.setSize(400, 400);
        postJobFrame.setLayout(new GridLayout(4, 2));

        Label jobTitleLabel = new Label("Job Title:");
        TextField jobTitleField = new TextField();
        Label skillsLabel = new Label("Required Skills (comma-separated):");
        TextField skillsField = new TextField();

        Button postButton = new Button("Post Job");
        Button cancelButton = new Button("Cancel");

        postJobFrame.add(jobTitleLabel);
        postJobFrame.add(jobTitleField);
        postJobFrame.add(skillsLabel);
        postJobFrame.add(skillsField);
        postJobFrame.add(postButton);
        postJobFrame.add(cancelButton);

        // Button Actions
        postButton.addActionListener(e -> {
            String jobTitle = jobTitleField.getText();
            String skillsStr = skillsField.getText();

            List<String> skills = new ArrayList<>();
            for (String skill : skillsStr.split(",")) {
                skills.add(skill.trim());
            }

            Job job = new Job(jobTitle, skills);
            employer.addJob(job);

            showMessage("Job posted successfully!");
            postJobFrame.dispose();
        });

        cancelButton.addActionListener(e -> postJobFrame.dispose());

        postJobFrame.setVisible(true);
    }

    private void showMessage(String message) {
        Dialog dialog = new Dialog(this, "Message", true);
        dialog.setSize(300, 100);
        dialog.setLayout(new FlowLayout());
        dialog.add(new Label(message));
        Button okButton = new Button("OK");
        okButton.addActionListener(e -> dialog.dispose());
        dialog.add(okButton);
        dialog.setVisible(true);
    }

    public static void main(String[] args) {
        new OnlineJobs();
    }
}

class JobSeeker {
    private String name;
    private String email;
    private String password;
    private List<String> skills;

    public JobSeeker(String name, String email, String password, List<String> skills)
```

```
{
        this.name = name;
        this.email = email;
        this.password = password;
        this.skills = skills;
    }

    public List<String> getSkills() {
        return skills;
    }

    public String getPassword() {
        return password;
    }
}

class Employer {
    private String companyName;
    private String email;
    private String password;
    private List<Job> jobs;

    public Employer(String companyName, String email, String password) {
        this.companyName = companyName;
        this.email = email;
        this.password = password;
        this.jobs = new ArrayList<>();
    }

    public String getCompanyName() {
        return companyName;
    }

    public String getPassword() {
        return password;
    }

    public List<Job> getJobs() {
        return jobs;
    }

    public void addJob(Job job) {
        jobs.add(job);
    }
}

class Job {
    private String title;
    private List<String> requiredSkills;

    public Job(String title, List<String> requiredSkills) {
        this.title = title;
        this.requiredSkills = requiredSkills;
    }

    public String getTitle() {
        return title;
    }

    public List<String> getRequiredSkills() {
        return requiredSkills;
    }
}
```

APPENDIX B - SCREENSHOTS



Welcome to the Online Job Portal!

Employer Menu

Job Seeker Menu

Welcome to the Online Job Portal!

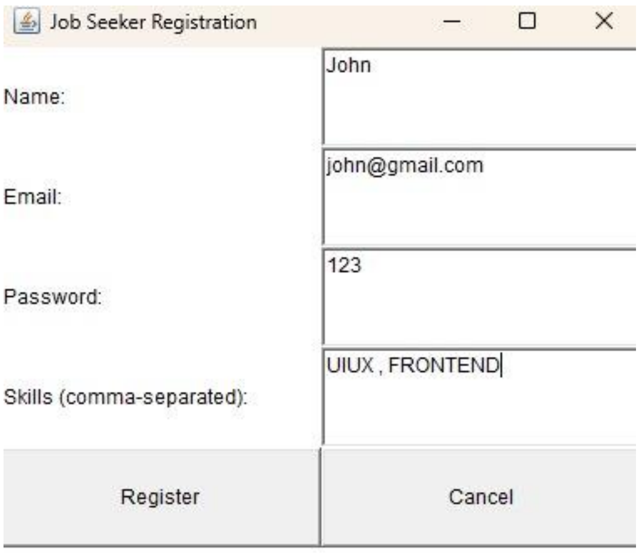
Employer registered successfully!

Email: technio@gmail.com

Password: 1234

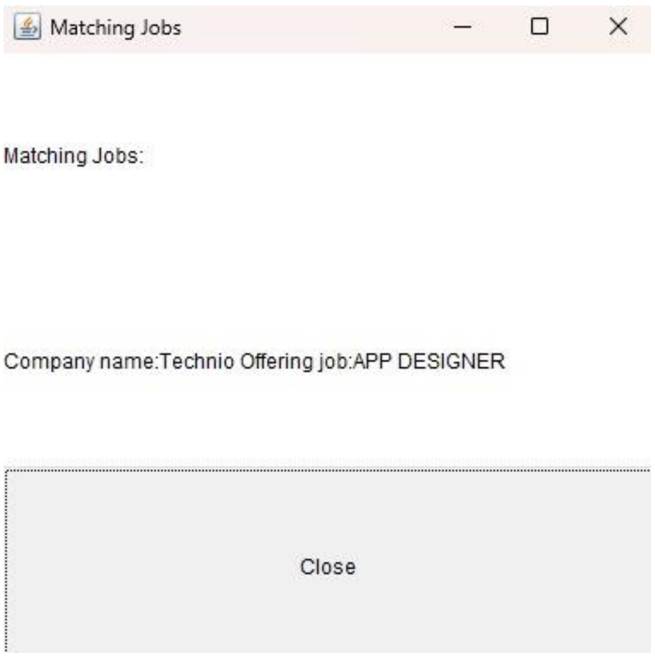
Job Title: APP DESIGNER

Required Skills (comma-separated: UIUX , Figma



Job Seeker Registration

Name:	John
Email:	john@gmail.com
Password:	123
Skills (comma-separated):	UIUX , FRONTEND



Matching Jobs

Matching Jobs:

Company name:Technio Offering job:APP DESIGNER