JOB PORTAL MANAGEMENT SYSTEM

### A MINI PROJECT REPORT

Submitted by

|  |  |
| --- | --- |
| **Shreya Mridula G** | **220701271** |
| **Vijayetha S T** | **220701320** |
|  |  |

In partial fulfillment for the award of the degree of BACHELOR OF ENGINEERING

IN COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE(AUTONOMOUS) THANDALAM

CHENNAI-602105

2023 - 2024

**BONAFIDE CERTIFICATE**

Certified that this Project Report “**JOB PORTAL”** is the Bonafide work of **“VIJAYETHA S T (220701320), SHREYA MRIDULA G (220701271)”** who carried out the project work under my supervision.

### Submitted for the Practical Examination held on

**SIGNATURE SIGNATURE**

**Dr R SABITHA**

**Professor and II Year Academic Head Computer Science and Engineering,**

**Rajalakshmi Engineering College (Autonomous),**

**Thandalam, Chennai - 602 105**

**Ms V JANANEE**

**Assistant Professor(SG),**

**Computer Science and Engineering,**

**Rajalakshmi Engineering College (Autonomous),**

**Thandalam, Chennai - 602 105**

### INTERNAL EXAMINER EXTERNAL EXAMINER

**ABSTRACT**

The Job Portal project is designed to streamline and enhance the recruitment process by connecting job seekers with potential employers through a comprehensive and user-friendly platform. This portal integrates advanced search functionalities, resume building tools, and job recommendation algorithms to provide a tailored experience for users. Job seekers can create detailed profiles, upload resumes, and apply for jobs that match their skills and career aspirations. Employers, on the other hand, have access to a vast database of candidates, enabling them to post job openings, search for qualified candidates, and manage the recruitment process efficiently. The platform's intuitive design ensures ease of use, making the job search and hiring process more effective and less time-consuming.

This project also incorporates various features to enhance user engagement and satisfaction, such as personalized job alerts, company reviews, and interview preparation resources. Security measures are implemented to protect user data, ensuring a safe and trustworthy environment. Additionally, the portal supports seamless integration with social media platforms, allowing users to leverage their professional networks. By utilizing modern web technologies and a robust backend infrastructure, the Job Portal aims to bridge the gap between job seekers and employers, facilitating a more dynamic and responsive job market.

Top of Form

Bottom of Form

**TABLE OF CONTENTS**

### Chapter 1

1. **INTRODUCTION**
   1. [Introduction](#_TOC_250012)
   2. Objectives
   3. Scope Of Work
   4. Problem Statement

[Chapter 2](#_TOC_250003)

2.SYSTEM SPECIFICATION

2.1 Software Specifications

2.2 Hardware specifications

### Chapter 3

#### 3.CODING

#### 3.1 Program Coding

#### Chapter 4

#### 4. SCREENSHOTS

#### 4.1 Application interface

#### 4.2 Database connectivity

#### Chapter 5

#### 5. CONCLUSION AND FUTURE ENHANCEMENT

#### 5.1 Conclusion

#### 5.2 Future Enhancement

#### 5.3 Reference

#### 

# CHAPTER 1

**1.INTRODUCTION**

**1.1 Introduction**

The Job Portal project was developed to address the growing need for a streamlined and efficient platform that connects job seekers with potential employers. In today’s fast-paced job market, both job seekers and employers require a centralized system that not only simplifies the recruitment process but also enhances the overall user experience. This project aims to provide a comprehensive solution by integrating various features that cater to the needs of both parties. The portal offers advanced search functionalities, personalized job recommendations, resume building tools, and a user-friendly interface, making it easier for job seekers to find relevant job opportunities and for employers to discover qualified candidates.

The development of this Job Portal involved leveraging modern web technologies and a robust backend infrastructure to ensure reliability and scalability. Key features of the portal include detailed user profiles, job posting and application management, company reviews, and interview preparation resources. Security and privacy were prioritized to protect user data and build trust within the community. Additionally, the portal supports integration with social media platforms to enable users to expand their professional networks. By bridging the gap between job seekers and employers, this project aims to facilitate a more dynamic and responsive job market, ultimately contributing to more efficient and successful hiring processes.

This project was a collaborative effort, incorporating feedback from both job seekers and employers to refine and enhance the platform's functionality. Throughout its development, emphasis was placed on creating an intuitive and accessible user interface to cater to a diverse user base with varying levels of technical proficiency. Comprehensive testing was conducted to ensure the platform's performance and reliability, addressing potential issues and optimizing the user experience. The Job Portal stands as a testament to innovative design and practical application, aiming to revolutionize the way job seekers and employers interact in the digital age.

**1.2 Objectives**

The Job Portal project aims to revolutionize the recruitment landscape by providing a comprehensive, user-friendly platform that bridges the gap between job seekers and employers. This project integrates advanced search functionalities, personalized job recommendations, resume-building tools, and detailed user profiles to create a seamless and efficient job search experience. Job seekers can easily find and apply for positions that match their skills and aspirations, while employers benefit from a vast database of qualified candidates, streamlined job posting processes, and efficient application management. The portal's intuitive design and robust backend infrastructure ensure reliability, scalability, and ease of use, making the recruitment process more effective and less time-consuming.

Security and privacy are paramount in the Job Portal, with robust measures in place to protect user data and foster a trustworthy environment. Additional features, such as company reviews, interview preparation resources, and social media integration, enhance user engagement and satisfaction. By leveraging modern web technologies and incorporating feedback from users, the Job Portal offers a dynamic and responsive solution to the challenges of the job market. This project not only facilitates more efficient hiring processes but also contributes to a more dynamic and responsive job market, ultimately benefiting both job seekers and employers.

Furthermore, the Job Portal emphasizes accessibility and inclusivity, ensuring that users with varying levels of technical proficiency can easily navigate and utilize the platform. The portal also supports multiple languages and regional settings, catering to a global audience and making it a versatile tool for international recruitment. Continuous updates and improvements based on user feedback and market trends ensure that the portal remains relevant and effective in addressing the evolving needs of the job market. By fostering a more connected and efficient recruitment ecosystem, the Job Portal project sets a new standard for digital job search and hiring solution

# 1.3 Scope of Work

The scope of the Job Portal project encompasses the design, development, and deployment of a comprehensive online platform that facilitates the job search and recruitment process for job seekers and employers. This includes the following key components:

1. **User Registration and Profile Management:**
   * Development of a user-friendly registration process for both job seekers and employers.
   * Creation and management of detailed user profiles, including personal information, educational background, work experience, and skill sets.
2. **Job Search and Application:**
   * Implementation of advanced search functionalities allowing job seekers to find job openings based on various criteria such as location, industry, job title, and experience level.
   * Development of a streamlined application process, enabling users to apply for jobs directly through the portal.
3. **Job Posting and Recruitment Management:**
   * Provision of tools for employers to post job vacancies, including detailed job descriptions, requirements, and benefits.
   * Development of a recruitment management system for employers to track applications, shortlist candidates, and schedule interviews.
4. **Resume Building and Management:**
   * Creation of resume-building tools that assist job seekers in creating professional resumes.
   * Secure storage and easy access to uploaded resumes for job applications.
5. **Personalized Recommendations and Alerts:**
   * Implementation of algorithms to provide personalized job recommendations based on user profiles and preferences.
   * Development of notification systems for job alerts, application status updates, and interview schedules.
6. **Additional Features:**
   * Integration of company reviews and ratings to help job seekers make informed decisions.
   * Provision of interview preparation resources, including tips and common questions.
   * Integration with social media platforms to facilitate networking and sharing of job postings.
7. **Security and Data Privacy:**
   * Implementation of robust security measures to protect user data and ensure privacy.
   * Compliance with relevant data protection regulations and best practices.
8. **User Interface and Experience:**
   * Design of an intuitive and accessible user interface to cater to a diverse user base.
   * Continuous user feedback incorporation to enhance the user experience and functionality.
9. **Testing and Deployment:**
   * Comprehensive testing to ensure the reliability, performance, and security of the platform.
   * Deployment of the platform on a scalable and reliable infrastructure.
10. **Maintenance and Updates:**
    * Ongoing maintenance to address any issues and ensure the platform operates smoothly.
    * Regular updates based on user feedback and evolving market trends to keep the platform relevant and effective.

This scope of work outlines the critical elements necessary to create a robust Job Portal that effectively serves the needs of both job seekers and employers, fostering a more efficient and dynamic job market.

### 1.4 Problem Statement

In the current job market, both job seekers and employers face significant challenges in connecting efficiently and effectively. Job seekers often struggle to find job opportunities that align with their skills, experience, and career aspirations due to the fragmented nature of existing job search platforms. They encounter difficulties in navigating multiple job boards, creating professional resumes, and receiving timely notifications about suitable job openings. Additionally, the lack of personalized job recommendations and comprehensive interview preparation resources further complicates their job search process.

On the other hand, employers face the challenge of identifying and attracting qualified candidates in a timely and cost-effective manner. The recruitment process is often hampered by the inefficiencies of managing numerous job postings, sifting through a high volume of unqualified applications, and coordinating interviews. Moreover, existing platforms do not adequately address the need for secure and reliable data handling, leaving both job seekers and employers concerned about the privacy and security of their personal information.

The Job Portal project aims to address these issues by creating a centralized, user-friendly platform that streamlines the job search and recruitment process. By integrating advanced search functionalities, personalized job recommendations, resume-building tools, and robust security measures, the portal seeks to bridge the gap between job seekers and employers, facilitating a more dynamic and responsive job market.

# CHAPTER 2

**2.SYSTEM SPECIFICATION**

**2.1** **Software Specification for Job Portal Project**

**1. Introduction**

This document outlines the software specifications for the development of a job portal application. The application aims to provide a robust, scalable, and user-friendly platform for job seekers and employers.

**2. System Overview**

The job portal application will include functionalities such as user registration, login, job search, job postings, job updates, and account deletion. The system will utilize modern web technologies for both the frontend and backend to ensure performance and scalability.

**3. Functional Requirements**

**3.1 User Registration**

Description: Allow users to register with their first name, last name, email, and password.

Inputs: First name, last name, email, password.

Outputs: Registration success or error message.

Process:

Collect user inputs.

Hash the password.

Store user details in the database.

**3.2 User Login**

Description: Allow users to log in using their email and password.

Inputs: Email, password.

Outputs: Login success or error message.

Process:

Collect user inputs.

Retrieve and verify hashed password.

Issue JWT on successful authentication.

**3.3 Job Search**

Description: Allow users to search for jobs based on various criteria.

Inputs: Job title, country, city, company.

Outputs: List of matching jobs.

Process:

Collect search criteria.

Query the database using Elasticsearch for advanced search functionalities.

Display results in the user interface.

**3.4 Job Postings**

Description: Allow employers to post new jobs.

Inputs: Job title, description, requirements, location, company name.

Outputs: Job posting success or error message.

Process:

Collect job details.

Store job information in the database.

**3.5 Job Update**

Description: Allow employers to update job details.

Inputs: Job ID, updated job details.

Outputs: Job update success or error message.

Process:

Retrieve job by ID.

Update job details in the database.

**3.6 Account Deletion**

Description: Allow users to delete their account.

Inputs: Email, password.

Outputs: Account deletion success or error message.

Process:

Verify user credentials.

Delete user account and associated data from the database.

**4. Non-Functional Requirements**

**4.1 Security**

Use JWT for secure user authentication.

Enforce SSL/TLS for secure data transmission.

Implement data encryption for sensitive information.

Comply with GDPR for user data privacy.

**4.2 Performance**

Optimize database queries.

Implement caching strategies.

Ensure the system can handle multiple concurrent users.

**4.3 Usability**

Design an intuitive and responsive user interface.

Provide clear error messages and guidance for users.

**5. System Architecture**

**5.1 Client-Server Model**

**Client:**

Frontend developed using React.js or Angular.js.

Responsive UI design with Bootstrap or Material-UI.

Server:

Backend developed using Node.js with Express.js or Django.

RESTful API for client-server communication.

**6. Database Design**

**6.1 Users Table**

Columns: id (PK), first\_name, last\_name, email (unique), password.

Constraints: Email must be unique.

**6.2 Jobs Table**

Columns: id (PK), title, description, requirements, location, company\_name.

Constraints: None specified.

**6.3 Applications Table**

Columns: id (PK), user\_id (FK), job\_id (FK), status, applied\_date.

Constraints: Foreign keys to Users and Jobs tables.

**7. Technology Stack**

**7.1 Frontend**

Framework: React.js or Angular.js.

Languages: HTML5, CSS3, JavaScript.

UI/UX Design: Bootstrap or Material-UI for responsive design.

Libraries: Redux for state management, Axios for API calls.

Tools: Webpack or Gulp for module bundling and task automation.

**7.2 Backend**

Framework: Node.js with Express.js or Django.

Languages: JavaScript (Node.js) or Python (Django).

Database: PostgreSQL or MongoDB

API: RESTful API.

Authentication: JWT for secure user authentication and session management.

Search: Elasticsearch for advanced search functionalities.

**8. Hosting and Deployment**

**8.1 Cloud Platform**

AWS, Google Cloud Platform, or Azure for cloud hosting.

**8.2 Web Server**

Nginx or Apache.

**8.3 CI/CD**

Jenkins, GitLab CI/CD, or GitHub Actions for continuous integration and deployment.

**9. Security**

**9.1 Encryption**

SSL/TLS for secure data transmission.

Data encryption for sensitive information.

**9.2 Authentication**

JWT for secure user sessions.

Implement MFA and secure password policies.

**10. Testing**

**10.1 Unit Testing**

Jest or Mocha/Chai for frontend and backend unit tests.

**10.2 Integration Testing**

Selenium or Cypress for end-to-end testing.

**10.3 Load Testing**

JMeter or Locust to ensure performance under high traffic.

**11. Version Control**

**11.1 System**

Git.

**11.2 Repository**

GitHub, GitLab, or Bitbucket.

**12. Project Management**

**12.1 Tools**

JIRA, Trello, or Asana for task management and collaboration.

**12.2 Documentation**

Confluence or Markdown-based documentation in the repository.

**13. Additional Features**

**13.1 Email Notifications**

SendGrid or AWS SES for email services.

**13.2 Social Media Integration**

OAuth for connecting with LinkedIn, Google, and Facebook.

# 2.2Hardware Specification

# Hardware Specifications for Job Portal Project

# To support the development, deployment, and operation of the Job Portal project, the following hardware specifications are recommended. These specifications ensure optimal performance, scalability, and reliability of the platform.

# 1. Development Environment

# 1.1 Developer Workstations

# Processor: Intel Core i7 or AMD Ryzen 7

# Memory: 16GB RAM or higher

# Storage: 512GB SSD or higher

# Operating System: Windows 10/11, macOS, or Linux (Ubuntu)

# Graphics: Integrated or dedicated graphics card suitable for software development

# 2. Server Environment

# 2.1 Web/Application Servers

# Processor: Intel Xeon E5 or AMD EPYC

# Memory: 32GB RAM or higher

# Storage: 1TB SSD for fast read/write operations

# Network: High-speed Ethernet (1Gbps or higher)

# Operating System: Linux (Ubuntu Server, CentOS, or Debian)

# Virtualization: Support for Docker and Kubernetes for containerization

# 2.2 Database Servers

# Processor: Intel Xeon E5 or AMD EPYC

# Memory: 64GB RAM or higher for handling large datasets

# Storage: 2TB SSD with RAID configuration for redundancy and performance

# Network: High-speed Ethernet (1Gbps or higher)

# Operating System: Linux (Ubuntu Server, CentOS, or Debian)

# Backup: Regular automated backups to external storage or cloud

# 2.3 Load Balancer

# Processor: Intel Xeon or AMD EPYC

# Memory: 16GB RAM or higher

# Network: Dual high-speed Ethernet ports (1Gbps or higher)

# Software: HAProxy, Nginx, or AWS Elastic Load Balancing (if using cloud services)

# 3. Cloud Infrastructure (if applicable)

# 3.1 Compute Instances

# Type: AWS EC2, Google Cloud Compute Engine, or Azure Virtual Machines

# Instance Type: General-purpose (e.g., t3.medium for web servers, r5.large for database servers)

# Scaling: Auto-scaling groups to handle varying loads

# 3.2 Storage

# Type: AWS S3, Google Cloud Storage, or Azure Blob Storage

# Capacity: Scalable based on demand

# Backup: Regular snapshots and backup to different regions

# 3.3 Networking

# Load Balancing: AWS Elastic Load Balancing, Google Cloud Load Balancing, or Azure Load Balancer

# Content Delivery Network (CDN): AWS CloudFront, Google Cloud CDN, or Azure CDN for faster content delivery

# 4. Development and Testing Servers

# Processor: Intel Core i5 or AMD Ryzen 5

# Memory: 16GB RAM

# Storage: 512GB SSD

# Network: High-speed Ethernet

# Virtualization: Support for virtual machines or containers for isolated testing environments

# 5. Backup and Redundancy

# 5.1 External Storage

# Type: NAS (Network-Attached Storage) with RAID configuration

# Capacity: At least 4TB, scalable based on backup nee

# 5.2 Redundancy

# Implementation: Failover servers and regular data replication to ensure high availability and disaster recovery

# 

# CHAPTER 3

## CODING

## import tkinter as tk

## from tkinter import messagebox

## from tkinter import ttk

## import mysql.connector

## from mysql.connector import Error

## import bcrypt

## selected\_job\_id = None # Global variable to store the selected job ID

## def update\_job():

## switch\_to\_job\_update()

## def delete\_account():

## switch\_to\_account\_deletion()

## # Function to create a new user in the database

## def signup():

## first\_name = entry\_fname.get()

## last\_name = entry\_lname.get()

## email = entry\_email.get()

## password = entry\_password.get()

## hashed\_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

## try:

## conn = mysql.connector.connect(

## host="localhost",

## user="root",

## password="Shreya@2005",

## database="job\_portal"

## )

## if conn.is\_connected():

## cursor = conn.cursor()

## insert\_query = "INSERT INTO users (first\_name, last\_name, email, password) VALUES (%s, %s, %s, %s)"

## cursor.execute(insert\_query, (first\_name, last\_name, email, hashed\_password))

## conn.commit()

## messagebox.showinfo("Success", "Registration Successful!")

## switch\_to\_dashboard()

## except Error as e:

## messagebox.showerror("Error", f"Error connecting to database: {e}")

## finally:

## if conn.is\_connected():

## cursor.close()

## conn.close()

## # Function to check user credentials and login

## def login():

## email = entry\_login\_email.get()

## password = entry\_login\_password.get()

## try:

## conn = mysql.connector.connect(

## host="localhost",

## user="root",

## password="Shreya@2005",

## database="job\_portal”

## )

## if conn.is\_connected():

## cursor = conn.cursor()

## select\_query = "SELECT password FROM users WHERE email=%s"

## cursor.execute(select\_query, (email,))

## result = cursor.fetchone()

## if result and bcrypt.checkpw(password.encode('utf-8'), result[0].encode('utf-8')):

## messagebox.showinfo("Success", "Login Successful!")

## switch\_to\_dashboard()

## else:

## messagebox.showerror("Error", "Invalid Email or Password")

## except Error as e:

## messagebox.showerror("Error", f"Error connecting to database: {e}")

## finally:

## if conn.is\_connected():

## cursor.close()

## conn.close()

## # Function to switch to the login page

## def switch\_to\_login():

## notebook.select(0)

## # Function to switch to the dashboard page

## def switch\_to\_dashboard():

## notebook.select(1)

## # Function to switch to the job portal page

## def switch\_to\_job\_portal()

## notebook.select(2)

## 

## # Function to switch to the job update page

## def switch\_to\_job\_update():

## notebook.select(3)

## # Function to switch to the account deletion page

## def switch\_to\_account\_deletion():

## notebook.select(4)

## # Function to update job details in the database

## def update\_job\_details():

## new\_title = entry\_new\_job\_title.get()

## new\_country = entry\_new\_country.get()

## new\_city = entry\_new\_city.get()

## new\_company = entry\_new\_company.get()

## if not selected\_job\_id:

## messagebox.showerror("Error", "No job selected for update.")

## return

## try:

## conn = mysql.connector.connect(

## host="localhost",

## user="root",

## password="Shreya@2005",

## database="job\_portal"

## )

## if conn.is\_connected():

## cursor = conn.cursor()

## update\_query = "UPDATE jobs SET title=%s, country=%s, city=%s, company=%s WHERE id=%s"

## 

## cursor.execute(update\_query, (new\_title, new\_country, new\_city, new\_company, selected\_job\_id))

## conn.commit()

## messagebox.showinfo("Success", "Job details updated successfully!")

## except Error as e:

## messagebox.showerror("Error", f"Error updating job details: {e}")

## finally:

## if conn.is\_connected():

## cursor.close()

## conn.close()

## # Function to delete user account from the database

## def delete\_user\_account():

## delete\_password = entry\_delete\_password.get()

## email = entry\_email.get()

## try:

## conn = mysql.connector.connect(

## host="localhost",

## user="root",

## password="Shreya@2005",

## database="job\_portal"

## )

## if conn.is\_connected():

## cursor = conn.cursor()

## select\_query = "SELECT password FROM users WHERE email=%s"

## cursor.execute(select\_query, (email,))

## result = cursor.fetchone()

## if result and bcrypt.checkpw(delete\_password.encode('utf-8'), result[0].encode('utf-8')):

## delete\_query = "DELETE FROM users WHERE email=%s"

## cursor.execute(delete\_query, (email,))

## conn.commit()

## messagebox.showinfo("Success", "Account deleted successfully!")

## else:

## messagebox.showerror("Error", "Invalid password for account deletion.")

## except Error as e:

## messagebox.showerror("Error", f"Error deleting user account: {e}")

## finally:

## if conn.is\_connected():

## cursor.close()

## conn.close()

## # Function to search for jobs in the database

## def search\_jobs():

## job\_title = entry\_job\_title.get()

## country = entry\_country.get()

## city = entry\_city.get()

## preferred\_company = entry\_company.get()

## try:

## conn = mysql.connector.connect(

## host="localhost",

## user="root",

## password="Shreya@2005",

## database="job\_portal"

## )

## if conn.is\_connected():

## cursor = conn.cursor()

## search\_query = "SELECT id, title, country, city, company FROM jobs WHERE title=%s OR country=%s OR city=%s OR company=%s"

## cursor.execute(search\_query, (job\_title, country, city, preferred\_company))

## jobs = cursor.fetchall()

## for row in jobs\_treeview.get\_children():

## jobs\_treeview.delete(row)

## for job in jobs:

## jobs\_treeview.insert("", tk.END, values=job)

## except Error as e:

## messagebox.showerror("Error", f"Error connecting to database: {e}")

## finally:

## if conn.is\_connected():

## cursor.close()

## conn.close()

## # Function to handle job selection

## def on\_job\_select(event):

## global selected\_job\_id

## selected\_item = jobs\_treeview.selection()

## if selected\_item:

## selected\_job = jobs\_treeview.item(selected\_item)['values']

## selected\_job\_id = selected\_job[0]

## # Function to create a frame with a background image

## def create\_frame\_with\_background(notebook, image\_path):

## frame = ttk.Frame(notebook)

## frame.grid\_columnconfigure(0, weight=1)

## frame.grid\_columnconfigure(1, weight=1)

## # Load the background image

## bg\_image = tk.PhotoImage(file=image\_path)

## canvas = tk.Canvas(frame, width=bg\_image.width(), height=bg\_image.height())

## canvas.grid(row=0, column=0, columnspan=2, rowspan=10)

## canvas.create\_image(0, 0, anchor='nw', image=bg\_image)

## canvas.image = bg\_image # Keep a reference to avoid garbage collection

## return frame, canvas

## # Create the main window

## root = tk.Tk()

## root.title("Signup/Login")

## root.geometry("800x600")

## style = ttk.Style()

## style.configure("TNotebook", tabposition='n')

## style.configure("TLabel", font=('Arial', 12), padding=5, background='#E0E0E0')

## style.configure("TEntry", font=('Arial', 12), padding=5, fieldbackground='#FFFFFF')

## style.configure("TButton", font=('Arial', 12), padding=5)

## style.configure("Treeview.Heading", font=('Arial', 12, 'bold'), background='#4CAF50', foreground='white')

## style.configure("Treeview", background='#E0E0E0', foreground='#000000', rowheight=25)

## notebook = ttk.Notebook(root)

## notebook.pack(fill='both', expand=True, padx=10, pady=10)

## # Signup Frame

## signup\_frame, signup\_canvas = create\_frame\_with\_background(notebook, "/Users/user/Desktop/ss.png")

## ttk.Label(signup\_frame, text="First Name:").grid(row=0, column=0, sticky='e')

## entry\_fname = ttk.Entry(signup\_frame)

## entry\_fname.grid(row=0, column=1, pady=5)

## ttk.Label(signup\_frame, text="Last Name:").grid(row=1, column=0, sticky='e')

## entry\_lname = ttk.Entry(signup\_frame)

## entry\_lname.grid(row=1, column=1, pady=5)

## ttk.Label(signup\_frame, text="Email:").grid(row=2, column=0, sticky='e')

## entry\_email = ttk.Entry(signup\_frame)

## entry\_email.grid(row=2, column=1, pady=5)

## ttk.Label(signup\_frame, text="Password:").grid(row=3, column=0, sticky='e')

## entry\_password = ttk.Entry(signup\_frame, show="\*")

## entry\_password.grid(row=3, column=1, pady=5)

## ttk.Button(signup\_frame, text="Signup", command=signup).grid(row=4, columnspan=2, pady=10)

## notebook.add(signup\_frame, text="Signup")

## # Login Frame

## login\_frame, login\_canvas = create\_frame\_with\_background(notebook, "/Users/user/Desktop/ss.png")

## ttk.Label(login\_frame, text="Email:").grid(row=0, column=0, sticky='e')

## entry\_login\_email = ttk.Entry(login\_frame)

## entry\_login\_email.grid(row=0, column=1, pady=5)

## ttk.Label(login\_frame, text="Password:").grid(row=1, column=0, sticky='e')

## entry\_login\_password = ttk.Entry(login\_frame, show="\*")

## entry\_login\_password.grid(row=1, column=1, pady=5)

## ttk.Button(login\_frame, text="Login", command=login).grid(row=2, columnspan=2, pady=10)

## notebook.add(login\_frame, text="Login")

## # Dashboard Frame

## dashboard\_frame = ttk.Frame(notebook)

## dashboard\_frame.grid\_columnconfigure(0, weight=1)

## ttk.Label(dashboard\_frame, text="Welcome to the job portal page", font=('Arial', 16)).pack(pady=20)

## ttk.Button(dashboard\_frame, text="Go to Job Portal", command=switch\_to\_job\_portal).pack(pady=10)

## notebook.add(dashboard\_frame, text="Dashboard")

## # Job Portal Frame

## job\_portal\_frame, job\_portal\_canvas = create\_frame\_with\_background(notebook, "/Users/user/Desktop/ss.png")

## ttk.Label(job\_portal\_frame, text="Job Title:").grid(row=0, column=0, sticky='e')

## entry\_job\_title = ttk.Entry(job\_portal\_frame)

## entry\_job\_title.grid(row=0, column=1, pady=5)

## ttk.Label(job\_portal\_frame, text="Country/Region:").grid(row=1, column=0, sticky='e')

## entry\_country = ttk.Entry(job\_portal\_frame)

## entry\_country.grid(row=1, column=1, pady=5)

## ttk.Label(job\_portal\_frame, text="City/District:").grid(row=2, column=0, sticky='e')

## entry\_city = ttk.Entry(job\_portal\_frame)

## entry\_city.grid(row=2, column=1, pady=5)

## ttk.Label(job\_portal\_frame, text="Preferred Company:").grid(row=3, column=0, sticky='e')

## entry\_company = ttk.Entry(job\_portal\_frame)

## entry\_company.grid(row=3, column=1, pady=5)

## ttk.Button(job\_portal\_frame, text="Search Jobs", command=search\_jobs).grid(row=4, columnspan=2, pady=10)

## ttk.Button(job\_portal\_frame, text="Update Account", command=update\_job).grid(row=5, columnspan=2, pady=10)

## ttk.Button(job\_portal\_frame, text="Delete Account", command=delete\_account).grid(row=6, columnspan=2, pady=10)

## jobs\_treeview = ttk.Treeview(job\_portal\_frame, columns=("id", "title", "country", "city", "company"), show='headings')

## jobs\_treeview.heading("id", text="Job ID")

## jobs\_treeview.heading("title", text="Title")

## jobs\_treeview.heading("country", text="Country")

## jobs\_treeview.heading("city", text="City")

## jobs\_treeview.heading("company", text="Company")

## jobs\_treeview.bind("<<TreeviewSelect>>", on\_job\_select)

## jobs\_treeview.grid(row=7, columnspan=2, pady=10)

## notebook.add(job\_portal\_frame, text="Job Portal")

## # Job Update Frame

## job\_update\_frame = ttk.Frame(notebook)

## job\_update\_frame.grid\_columnconfigure(0, weight=1)

## job\_update\_frame.grid\_columnconfigure(1, weight=1)

## ttk.Label(job\_update\_frame, text="New Job Title:").grid(row=0, column=0, sticky='e')

## entry\_new\_job\_title = ttk.Entry(job\_update\_frame)

## entry\_new\_job\_title.grid(row=0, column=1, pady=5)

## ttk.Label(job\_update\_frame, text="New Country/Region:").grid(row=1, column=0, sticky='e')

## entry\_new\_country = ttk.Entry(job\_update\_frame)

## entry\_new\_country.grid(row=1, column=1, pady=5)

## ttk.Label(job\_update\_frame, text="New City/District:").grid(row=2, column=0, sticky='e')

## entry\_new\_city = ttk.Entry(job\_update\_frame)

## entry\_new\_city.grid(row=2, column=1, pady=5)

## ttk.Label(job\_update\_frame, text="New Preferred Company:").grid(row=3, column=0, sticky='e')

## entry\_new\_company = ttk.Entry(job\_update\_frame)

## entry\_new\_company.grid(row=3, column=1, pady=5)

## ttk.Button(job\_update\_frame, text="Update Job", command=update\_job\_details).grid(row=4, columnspan=2, pady=10)

## notebook.add(job\_update\_frame, text="Update Job")

## # Account Deletion Frame

## account\_deletion\_frame = ttk.Frame(notebook)

## account\_deletion\_frame.grid\_columnconfigure(0, weight=1)

## ttk.Label(account\_deletion\_frame, text="Enter Password to Delete Account:").grid(row=0, column=0, sticky='e')

## entry\_delete\_password = ttk.Entry(account\_deletion\_frame, show="\*")

## entry\_delete\_password.grid(row=0, column=1, pady=5)

## ttk.Button(account\_deletion\_frame, text="Delete Account", command=delete\_user\_account).grid(row=1, columnspan=2, pady=10)

## notebook.add(account\_deletion\_frame, text="Delete Account")

## notebook.select(0)

## root.mainloop()

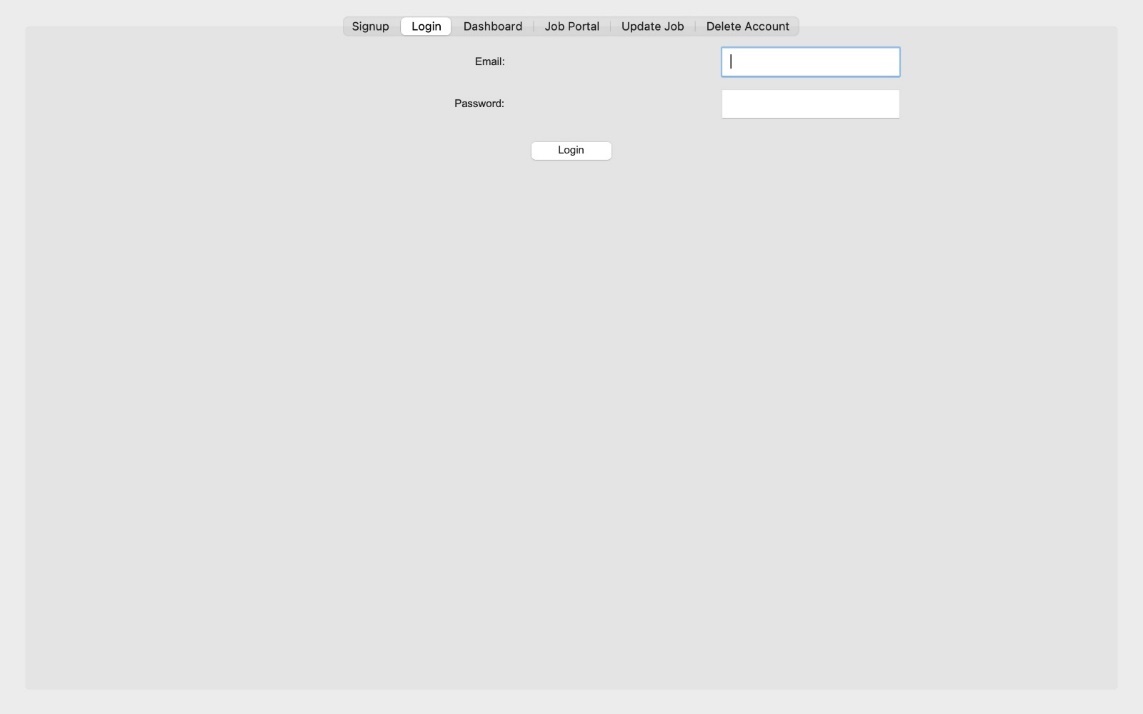
## CHAPTER 4

## 4.SCREENSHOT

## 4.1 Application Interface

## Signup page

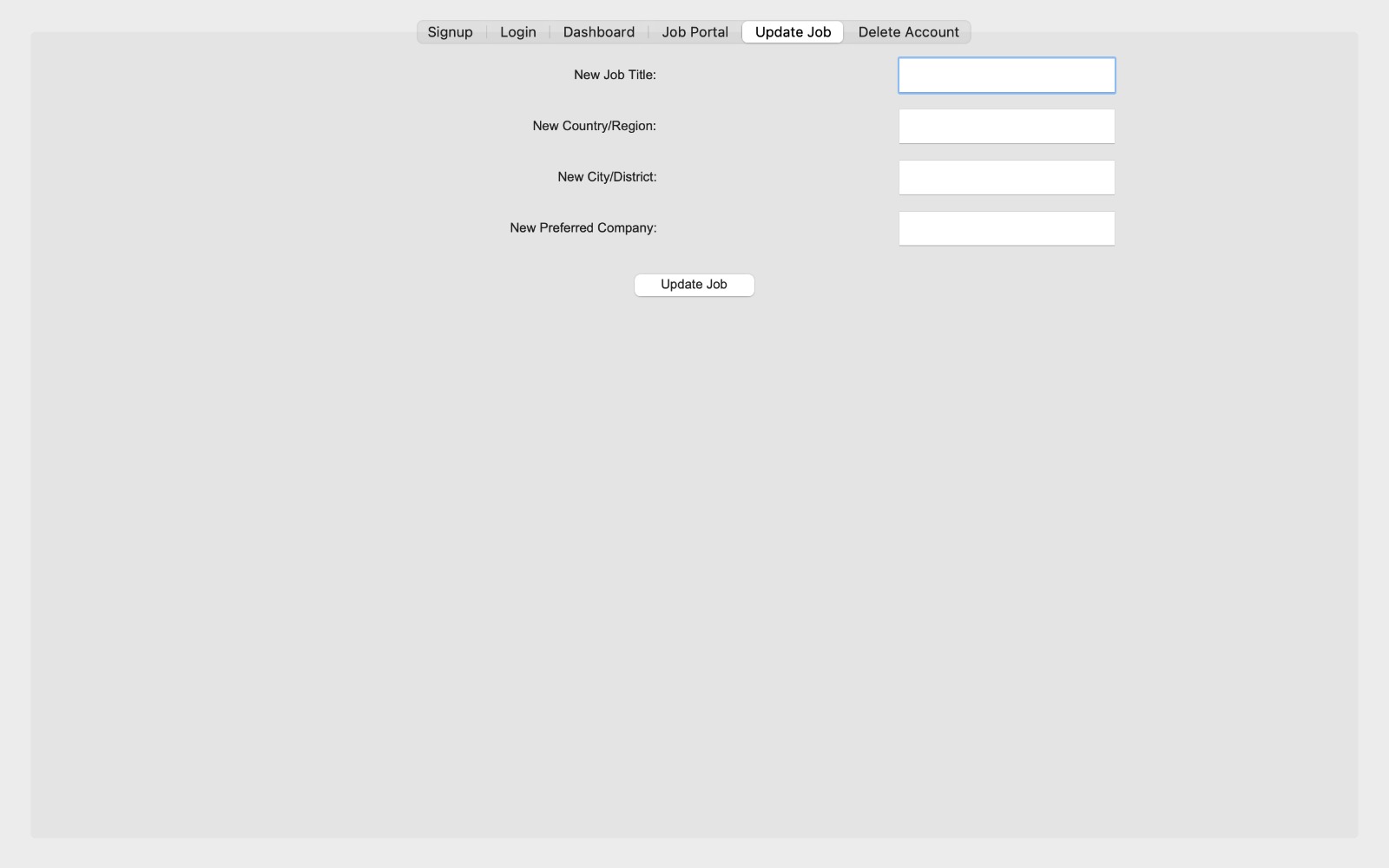
## Login Page



## Dashboard Page

## Job Portal Page

## Update Job Page



## Delete Job Page

**4.2 Database Connectivity**



## CHAPTER 5

## 5.CONCLUSION AND FUTURE ENHANCEMENTS

## 5.1 Conclusion

## The Job Portal project successfully addresses the critical challenges faced by job seekers and employers in the current job market. By providing a comprehensive, user-friendly platform, the project facilitates a seamless and efficient recruitment process. The integration of advanced search functionalities, personalized job recommendations, resume-building tools, and secure data handling creates a reliable and effective environment for users. Employers can easily manage job postings and applications, while job seekers benefit from tailored job suggestions and robust application support. The platform's modern web technologies and scalable infrastructure ensure its reliability, performance, and security, making it a valuable tool in the digital job market landscape.

## 5.2 Future Enhancements

**Future Enhancements:**

1. **AI and Machine Learning Integration:**
   * Implement advanced AI algorithms to enhance job recommendations, match job seekers with suitable roles more accurately, and predict hiring trends.
   * Use machine learning to analyze user behavior and improve the platform's personalization features.
2. **Mobile Application:**
   * Develop a mobile application for iOS and Android to provide users with on-the-go access to job searches, applications, and notifications.
3. **Video Interviewing:**
   * Integrate video interviewing capabilities to facilitate remote hiring processes, enabling employers to conduct interviews directly through the platform.
4. **Gamification:**
   * Introduce gamification elements such as badges, points, and rewards to enhance user engagement and motivate job seekers to complete their profiles and applications.
5. **Third-Party Integrations:**
   * Integrate with popular HR management systems, job boards, and professional networks to provide a seamless experience for users who utilize multiple platforms.
6. **Continuous Security Enhancements:**
   * Implement advanced security measures such as biometric authentication and regular security audits to ensure the highest level of data protection.

These future enhancements will further improve the functionality and user experience of the Job Portal, ensuring it remains a leading solution in the evolving job market. By continuously innovating and responding to user feedback, the platform will continue to meet the dynamic needs of job seekers and employers alike.

**5.3 References**

## This below website helped us in gaining more knowledge in the subject

## and completing the project

## <https://dev.mysql.com/doc/connector-python/en/>

## <https://docs.python.org/3/library/tkinter.html>

## <https://pypi.org/project/bcrypt/>