Job Portal Web Application (Flask)

Pratik Pawar

12-02-2026

—

Python Developer Trainee Program

—

Technology Used: Flask, SQLite, HTML, CSS, Bootstrap

Abstract

The Job Portal Web Application is a web-based platform developed using the Flask framework in Python. The system is designed to connect job seekers and employers in a structured and efficient manner. The application allows employers to post job listings, job seekers to search and apply for jobs, and an administrator to manage users and job postings.

The application implements secure user authentication, role-based access control, database management using SQLAlchemy ORM, and a responsive user interface using Bootstrap and custom CSS. The project demonstrates practical implementation of backend development concepts, database relationships, authentication mechanisms, and UI/UX design principles.

This project reflects real-world web application architecture and follows best practices for clean code and structured development.

Introduction

In today's competitive job market, online job portals have become essential tools for recruitment and employment. Companies use digital platforms to publish job openings, while job seekers rely on them to search and apply for suitable positions.

This project aims to build a simplified yet functional Job Portal system using Flask. The system provides three primary roles:

1. Job Seekers
2. Employers
3. Administrator

Each role has specific permissions and responsibilities within the system.

The project simulates a real-world recruitment platform and provides hands-on experience with full-stack web development using Python.

Problem Statement

Manual job recruitment processes are inefficient and time-consuming. There is a need for a centralized digital platform that:

- Allows employers to post job vacancies
- Enables job seekers to browse and apply for jobs
- Provides administrative control for monitoring activity

The goal of this project is to design and implement such a system using Flask and SQLite.

Objectives

The primary objectives of this project are:

- To develop a web-based job portal using Flask.
- To implement secure user authentication.
- To design role-based access control.
- To allow employers to post and manage job listings.
- To enable job seekers to apply for jobs.
- To store and manage data using SQLAlchemy ORM.
- To create a responsive and user-friendly interface.
- To simulate real-world project structure and deployment readiness.

Technologies Used

Technology	**Description**
Python	Backend programming language
Flask	Lightweight web framework
SQLite	Database system
SQLAlchemy	ORM for database interaction
Flask-Login	Authentication and session management
HTML	Structure of web pages
CSS	Styling and UI design
Bootstrap	Responsive front-end framework
Werkzeug	Password hashing and security

System Architecture

The application follows a modular structure similar to the MVC (Model-View-Controller) architecture.

7.1 Models

Defines the database structure:

- User
- Job
- Application

7.2 Views (Routes)

Handles HTTP requests and business logic:

- Login
- Register
- Post Job
- Apply for Job
- Admin Dashboard

7.3 Templates

HTML files rendered dynamically using Jinja2 templating.

7.4 Static Files

Contains CSS and styling resources.

Database Design

The system uses SQLite as the database and SQLAlchemy as ORM.

8.1 User Table

Field	Description
id	Primary key
username	User's name
email	Unique email
password	Hashed password
role	jobseeker / employer

8.2 Job Table

Field	Description
id	Primary key
title	Job title
description	Job description
company	Company name
location	Job location
salary	Salary information
posted_by	Employer ID
created_at	Timestamp

8.3 Application Table

7y	Description
id	Primary key
job_id	Foreign key to Job
user_id	Foreign key to User
applied_at	Timestamp

Functional Modules

9.1 Authentication Module

- User Registration
- Secure password hashing
- Login and Logout
- Session management
- Role-based access restriction

Passwords are hashed using Werkzeug security utilities before storing in the database.

9.2 Employer Module

Features available to employers:

- Post new job listings
- View their own job postings
- Manage job information
- Restricted access to job seeker features

Employers cannot apply for jobs.

9.3 Job Seeker Module

Features available to job seekers:

- Browse available jobs
- View job details
- Apply for jobs
- Prevent duplicate applications
- View previously applied jobs

9.4 Admin Module

The administrator has full control over the system:

- View all users
- View all job postings
- Delete inappropriate job listings
- Restricted access to admin-only pages

Admin access is validated based on a predefined admin email.

User Interface Design

The UI was developed using Bootstrap and custom CSS.
Design features include:

- Gradient background
- Card-based layout
- Responsive design
- Smooth hover animations
- Clean navigation bar
- Flash messages for feedback

The UI ensures accessibility and readability across devices.

Implementation Details

11.1 Role-Based Access Control

Role-based restrictions are implemented using conditional checks on user roles before allowing access to certain routes.
Example:

- Only employers can access /post-job
- Only job seekers can apply for jobs
- Only admin can access /admin

11.2 Duplicate Application Prevention

Before saving an application, the system checks:

- If the same user has already applied for the same job
- If yes, it prevents duplicate submission

11.3 Database Handling

The application uses SQLAlchemy ORM for:

- Creating tables
- Inserting records
- Querying data
- Deleting records

Challenges Faced

During development, the following challenges were encountered:

- Database schema mismatch after model updates
- Template syntax errors in Jinja
- Handling role-based route restrictions
- Managing session authentication
- Preventing duplicate job applications

These issues were resolved through debugging, database resets, and structured logic improvements.

Testing and Validation

The system was tested under the following conditions:

✓ Registration and login validation
✓ Incorrect password handling
✓ Role-based access restriction
✓ Employer job posting
✓ Job seeker job application
✓ Duplicate application prevention
✓ Admin deletion functionality

All modules were tested successfully.

Future Enhancements

The system can be further improved by adding:

- Resume upload feature
- Advanced job search filters
- Email notification system
- Pagination for job listings
- Password reset feature
- Deployment to cloud server
- Integration with external job APIs
- JWT-based authentication
- REST API version of system

Conclusion

The Job Portal Web Application successfully demonstrates the implementation of a full-stack web system using Flask. The system integrates authentication, database management, and role-based access control into a structured application.
The project provides hands-on experience with:

- Backend development
- Database relationships
- Authentication systems
- UI design principles
- Real-world application workflow

This project reflects practical understanding and application of Python web development concepts.

Learning Outcomes

Through this project, the following skills were developed:

- Flask application structure
- Database modeling using SQLAlchemy
- Secure password handling
- Role-based system design
- Debugging and error handling
- UI/UX enhancement using Bootstrap
- Writing clean and modular code

References

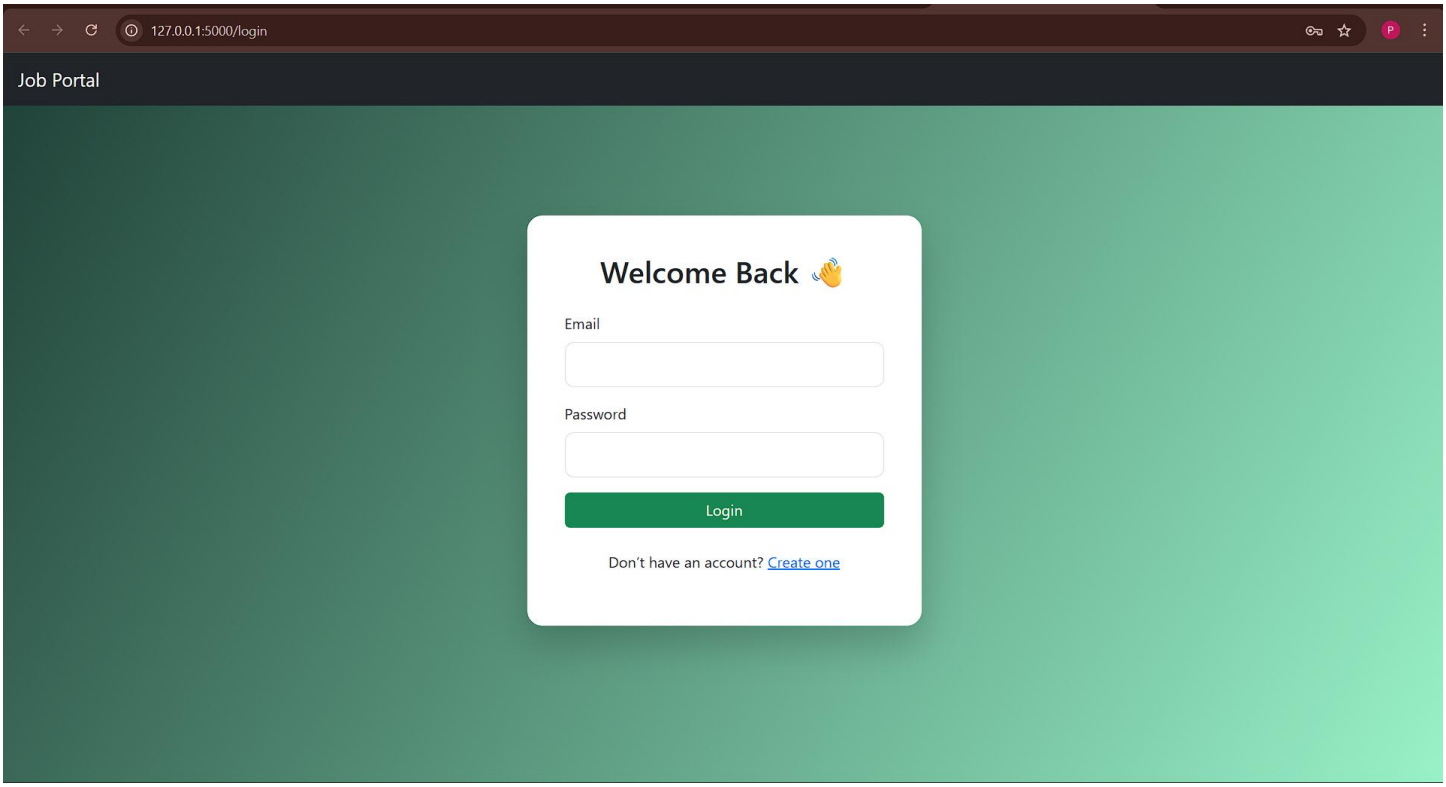
- Flask Official Documentation
- SQLAlchemy Documentation
- Bootstrap Documentation
- Python Official Documentation

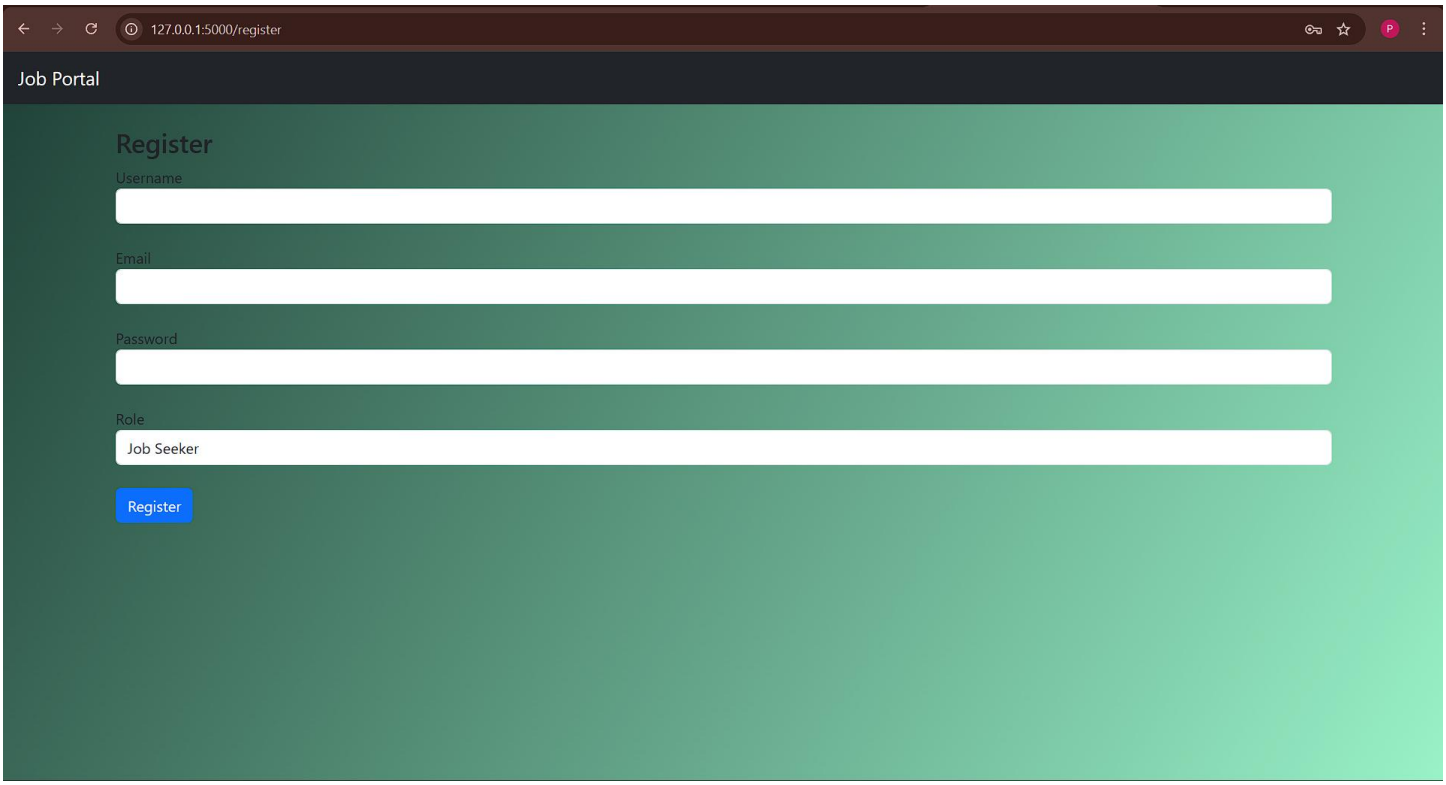
Final Statement

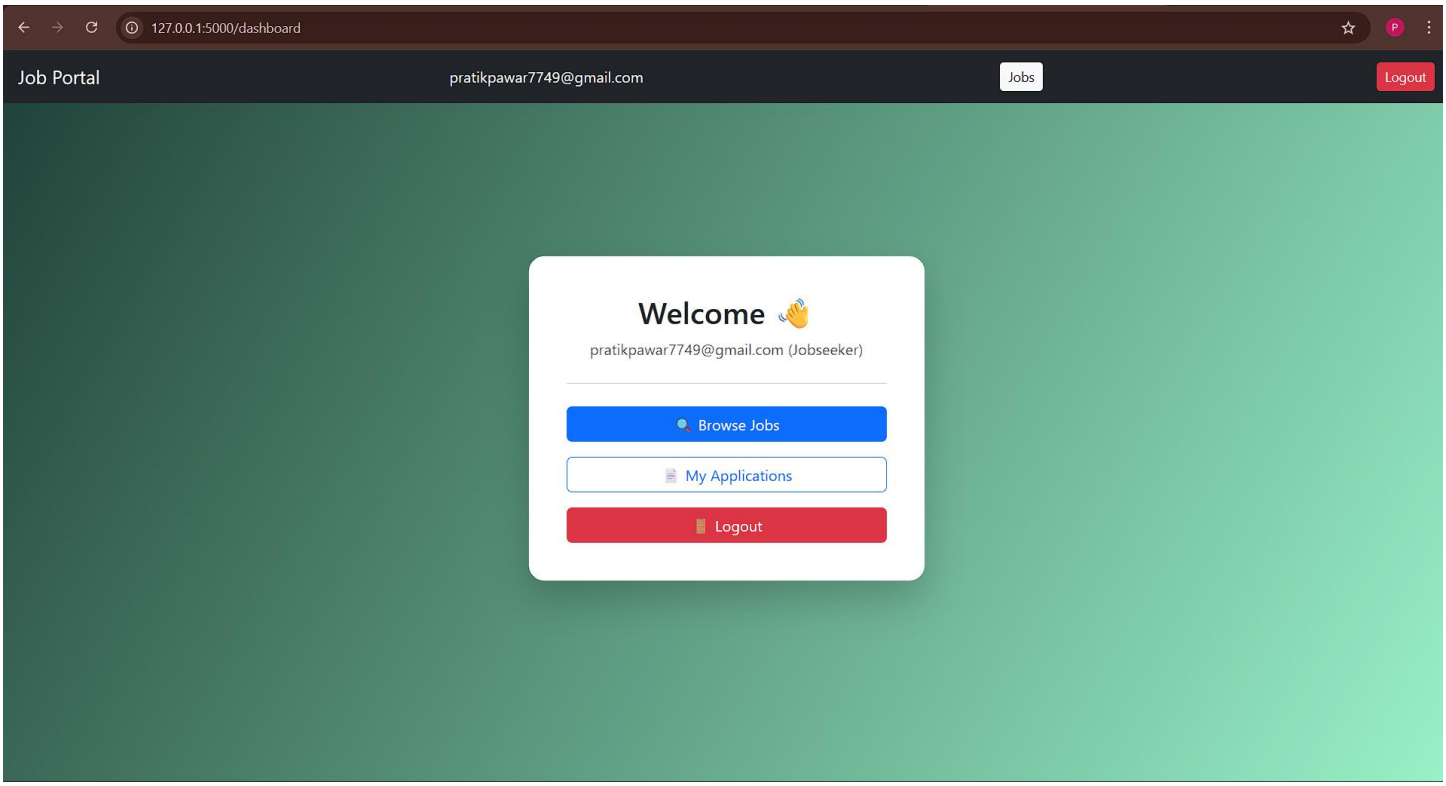
This project represents a complete and functional job portal system developed during the internship. It demonstrates technical skills, structured thinking, and real-world development capability.

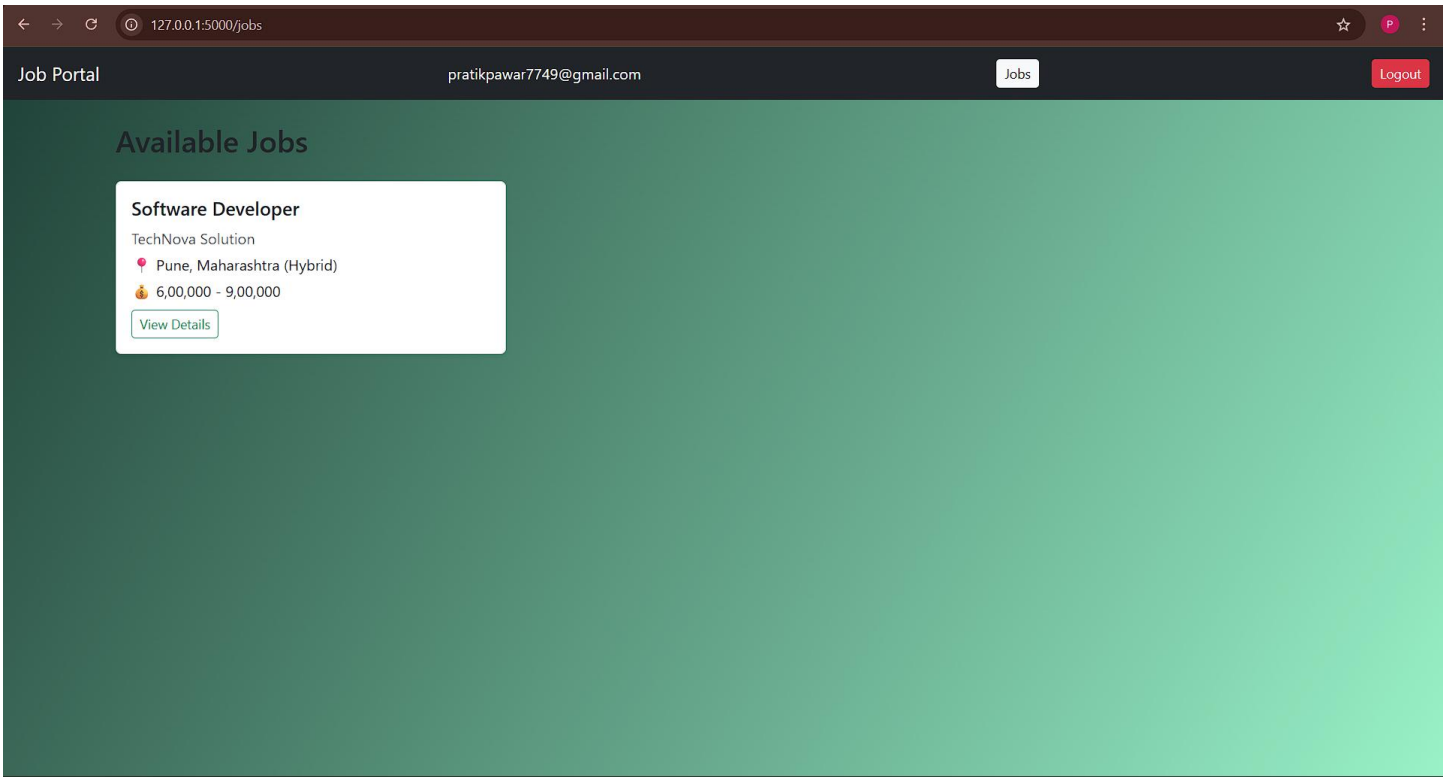
OUTPUT

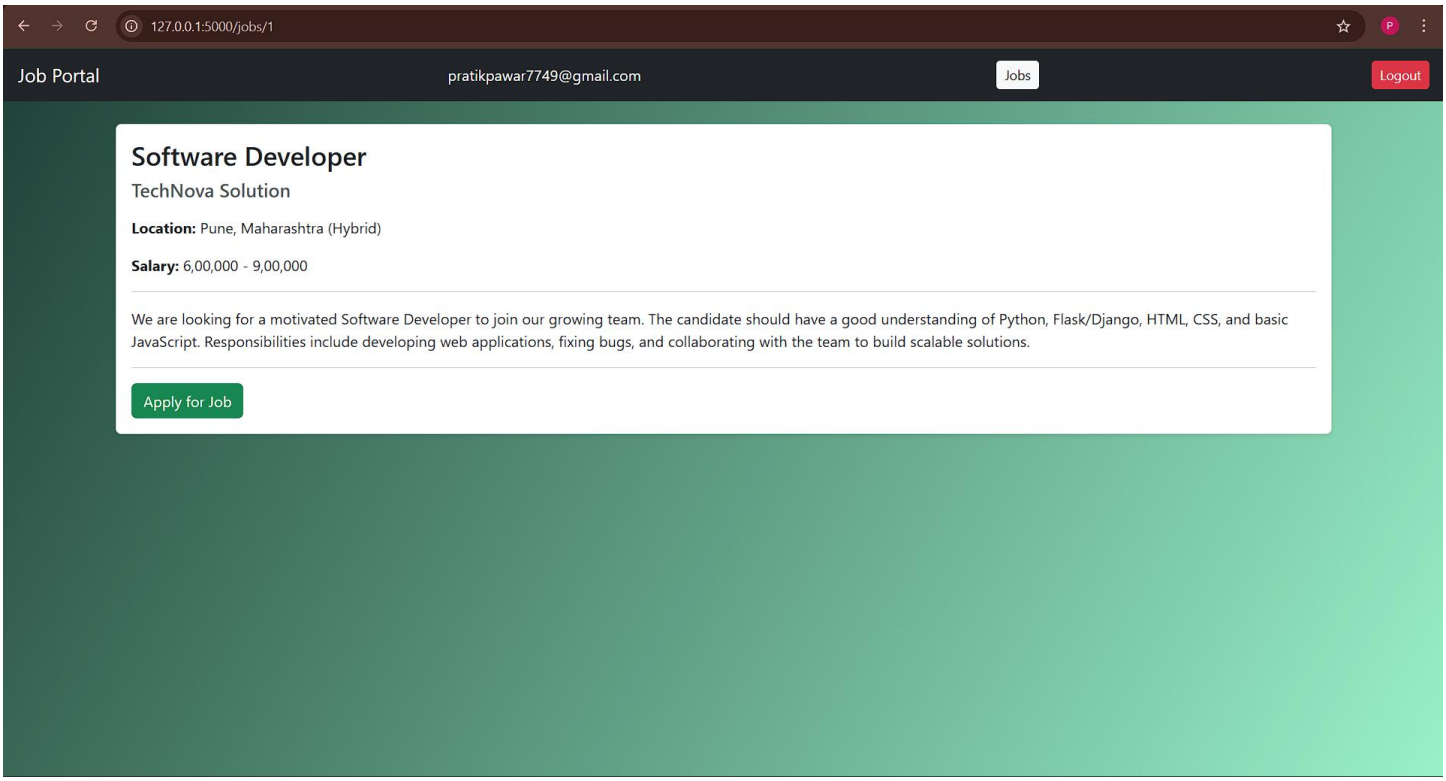
```
(venv) C:\Users\prati\OneDrive\Desktop\Python Internship Project>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 252-805-424
127.0.0.1 - - [15/Feb/2026 21:10:01] "GET / HTTP/1.1" 302 -
127.0.0.1 - - [15/Feb/2026 21:10:01] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2026 21:10:02] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [15/Feb/2026 21:10:02] "GET /favicon.ico HTTP/1.1" 404 -
```

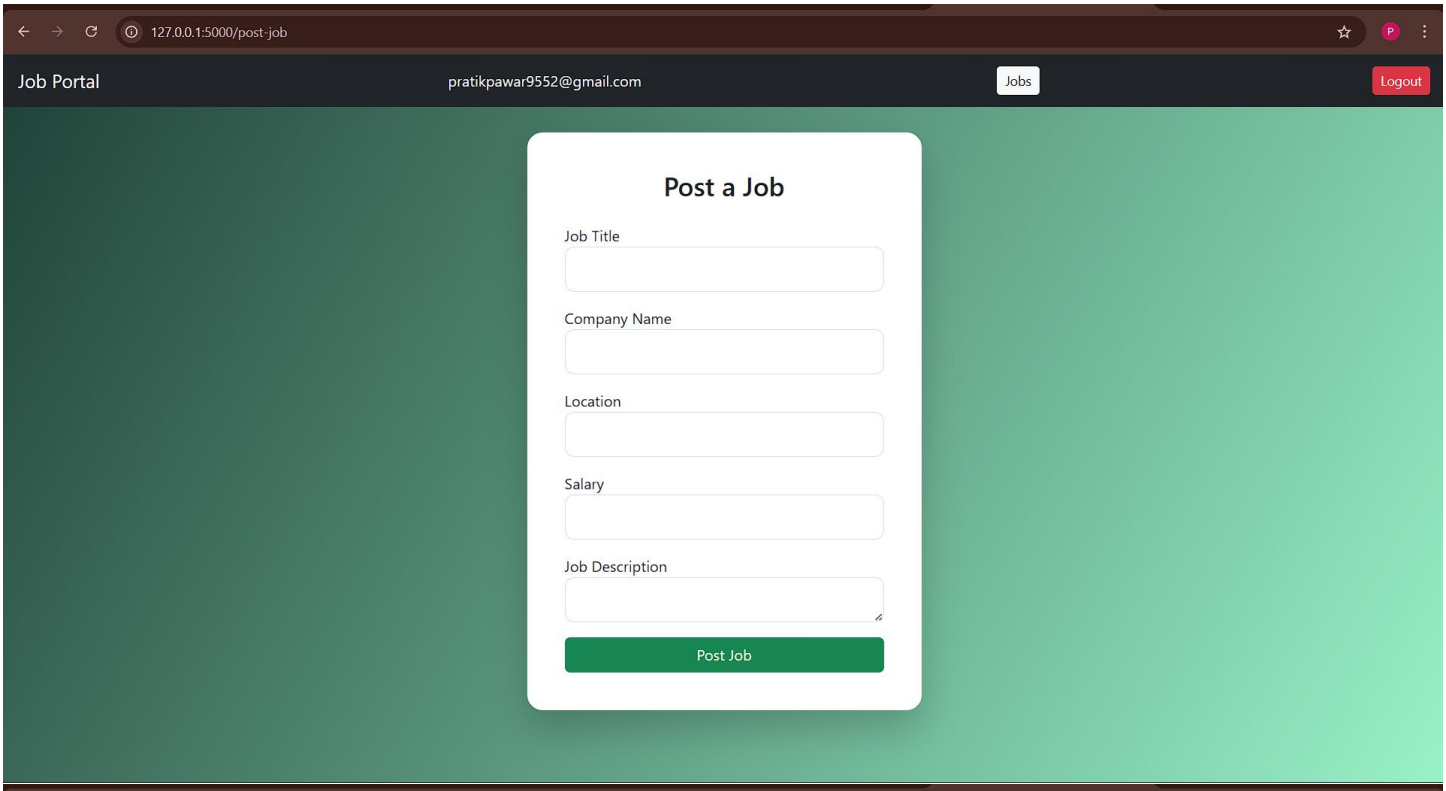


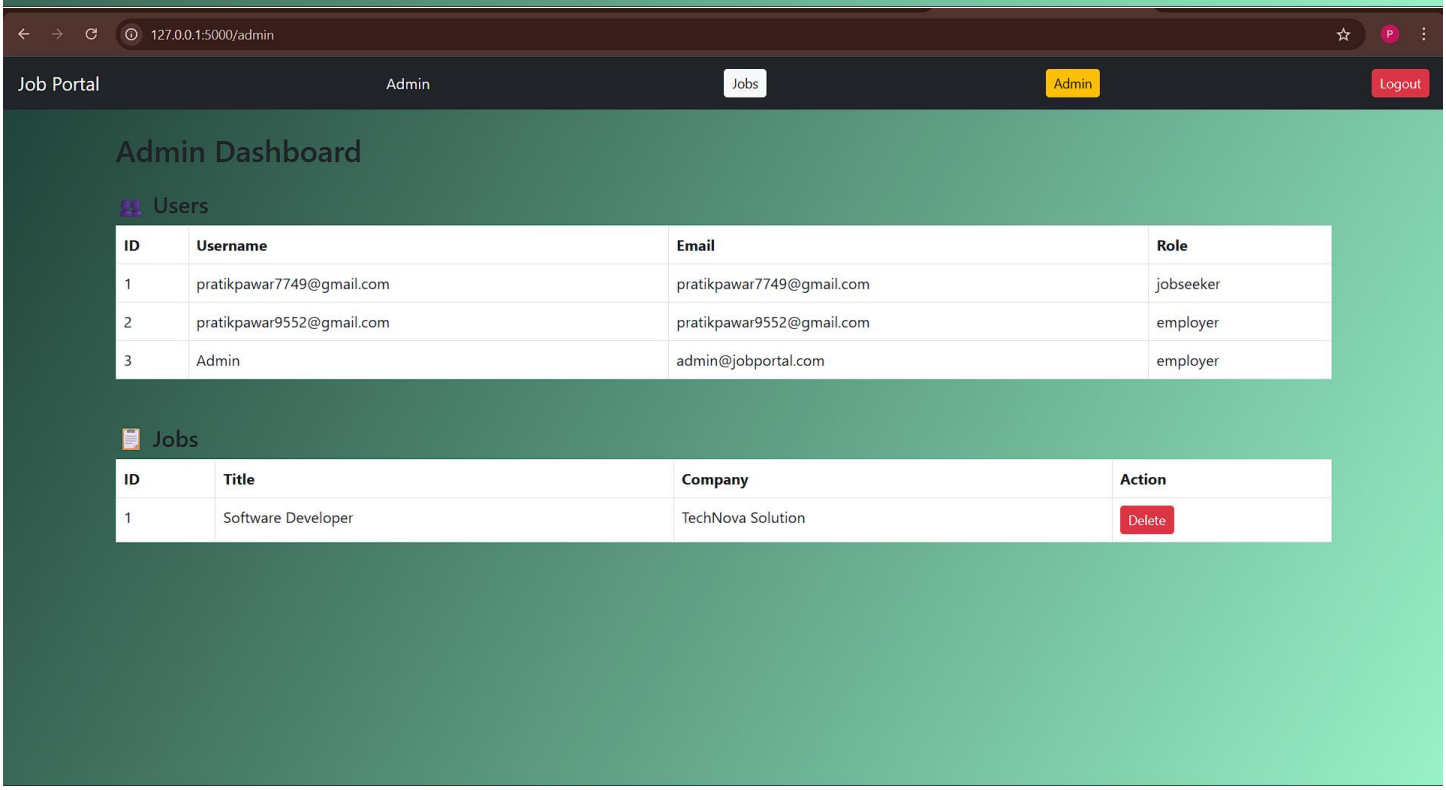

127.0.0.1:5000/register
Job Portal
Register
Username
Email
Password
Role
Job Seeker
Register


127.0.0.1:5000/dashboard
Job Portal
pratikpawar7749@gmail.com
Jobs
Logout
Welcome
pratikpawar7749@gmail.com (Jobseeker)
Browse Jobs
My Applications
Logout


127.0.0.1:5000/jobs
Job Portal
pratikpawar7749@gmail.com
Jobs
Logout
Available Jobs
Software Developer
TechNova Solution
Pune, Maharashtra (Hybrid)
6,00,000 - 9,00,000
View Details


127.0.0.1:5000/jobs/1
Job Portal
pratikpawar7749@gmail.com
Jobs
Logout
Software Developer
TechNova Solution
Location: Pune, Maharashtra (Hybrid)
Salary: 6,00,000 - 9,00,000
We are looking for a motivated Software Developer to join our growing team. The candidate should have a good understanding of Python, Flask/Django, HTML, CSS, and basic JavaScript. Responsibilities include developing web applications, fixing bugs, and collaborating with the team to build scalable solutions.
Apply for Job


127.0.0.1:5000/post-job
Job Portal
pratikpawar9552@gmail.com
Jobs
Logout
Post a Job
Job Title
Company Name
Location
Salary
Job Description
Post Job


127.0.0.1:5000/admin
Job Portal
Admin
Jobs
Admin
Logout
Admin Dashboard
Users
ID
Username
Email
Role
1
pratikpawar7749@gmail.com
pratikpawar7749@gmail.com
jobseeker
2
pratikpawar9552@gmail.com
pratikpawar9552@gmail.com
employer
3
Admin
admin@jobportal.com
employer
Jobs
ID
Title
Company
Action
1
Software Developer
TechNova Solution
Delete