Introduction

Hafiz Ayan Ahmed
NED University of Engineering and Technology
Department of Computer Systems Engineering

System Design Document for Job Portal System

Project Overview

The Job Portal System is a web-based platform designed to streamline the hiring process. It facilitates interactions between companies and job seekers by providing features such as job posting, application management, and interview scheduling. The system uses Django as the backend framework and includes distinct roles for administrators, companies, and employees.

System Architecture

1. Architecture Type

- Three-tier architecture:

  Presentation Layer: HTML, CSS, and Django templates.

  Application Layer: Django-based logic for handling requests, authentication, and routing.

  Data Layer: SQLite database for storing user data, job postings, applications, and interview schedules.

2. Key Components

- Django Admin:

  Role: Super Administrator.

  Manages: Companies, employees, and system settings.

- Companies:

  Post and manage jobs.

  View and manage job applications.

  Schedule interviews.

- Employees:

  Search and apply for jobs.

  View application status.

Access interview details.

Functional Requirements

1. User Roles and Permissions

- Admin:

  Create, update, and delete company and employee accounts.

  View system analytics.

- Company:

  Post jobs with details such as title, description, skills, and qualifications.

  Manage job applications.

  Schedule interviews for applicants.

- Employee:

  Browse and search for jobs.

  Apply for jobs with a resume.

  Check application status and interview schedules.

2. Features

- Authentication:

  Signup, login, and logout for companies and employees.

  Role-based redirection (company dashboard, employee dashboard).

- Job Management:

  Create, read, update, and delete job postings (CRUD operations).

  View applications and status filtering.

- Application Management:

Employees can submit applications with resume uploads.

Companies can update application statuses (e.g., 'Pending,' 'Accepted,' 'Rejected,' 'Interview Scheduled').

- Interview Scheduling:

Companies can assign interview dates and times.

Employees can view interview details.

Non-Functional Requirements

1. Performance:

- The system should handle up to 500 concurrent users.

2. Scalability:

- Designed to support multiple companies and thousands of job applications.

3. Security:

- Role-based access control.

- Data validation for forms.

- Encrypted storage of passwords using Django's default authentication system.

4. Usability:

- Responsive design for web and mobile browsers.

- Intuitive navigation and interface.

Database Design

1. Tables

- Users Table

id: Primary key

username: Unique

password: Hashed

email

role: Enum (Admin, Company, Employee)

- Jobs Table

id: Primary key

title

description

required_skills

qualifications

company_id: Foreign key (linked to Users table)

- Applications Table

id: Primary key

job_id: Foreign key

user_id: Foreign key (linked to Users table)

resume: File path

status: Enum (Pending, Accepted, Rejected, Interview Scheduled)

interview_date: DateTime

API Endpoints

1. Authentication

- POST /signup/: User registration.

- POST /login/: User login.

2. Job Management

- GET /jobs/: Fetch all jobs.
- POST /jobs/: Create a new job (Company only).
- PUT /jobs/{id}/: Update job details (Company only).
- DELETE /jobs/{id}/: Delete a job (Company only).

3. Application Management
- GET /applications/: Fetch applications for a job (Company only).
- POST /applications/: Apply for a job (Employee only).
- PUT /applications/{id}/: Update application status (Company only).

Deployment Plan

1. Hosting:
- Backend: Deployed on a cloud platform such as AWS or Heroku.
- Database: Hosted on SQLite during development, can migrate to PostgreSQL for production.

2. Static Files:
- Managed using django.contrib.staticfiles and served via CDN in production.

3. Environment Variables:
- Use python-decouple to manage sensitive information such as database credentials and secret keys.

Tools and Technologies

1. Backend: Django, Python

2. Frontend: HTML, CSS, Django Templates

3. Database: SQLite (default), PostgreSQL (optional for production)

4. Version Control: Git, GitHub

5. Dependencies: Managed via pip and requirements.txt file.

Future Enhancements

1. Implement advanced filtering and search for jobs.
2. Add notifications for interview schedules.
3. Enable integration with third-party services like LinkedIn or Indeed.

Conclusion

The Job Portal System is designed to simplify the job application and hiring process while maintaining a user-friendly interface for both companies and employees. The architecture and design ensure scalability, security, and ease of use.