Dynamic Job Portal Scraping and Application Tracking Website (GitHub)

By- Ryan Matthew (J053), Soumya Dass (J063), Raadnya Apte (J077), Siddharth Bakshi (J062)

Introduction:

This project is a web application that allows users to search for job postings on various job portals, save jobs they are interested in, and track their applications. The website is built using Django for the back end, HTML/CSS/JavaScript for the front end, Selenium for webscraping and Docker for containerization and deployment. It features two main pages: the **Search Tab** and the **Saved Jobs Tab**. The application also includes a MySQL database to store user data, job postings, and saved jobs.

Objective:

The primary objective of this project is to provide users with a streamlined experience for searching, saving, and tracking job applications from various online job portals.

Functional Requirements:

1. User Management

- **Sign-Up:**
    - Users can create an account by providing details like username, password, and email.
    - User details will be stored in the user_info MySQL table.
    - Password validation rules must be met (e.g., minimum length, special characters, etc.).
- **Login:**
    - Users can log in using their credentials (username and password).
    - Form validation ensures that all required fields are filled out and meet criteria.

2. Job Search

- **Search Tab:**
    - Users can search for job postings by entering a job role and location.
    - Upon clicking the submit button, a web-scraping script is triggered.
    - The web scraper retrieves job postings from multiple job portals (e.g., Indeed, LinkedIn) and stores them in the job_posting_info MySQL table.

- Each job posting contains:
    - Company Name
    - Role
    - Location
    - Link to the job posting
- The data is then retrieved and displayed on the website.

3. Saving Job Postings

- **Saved Jobs Tab:**
    - Users can save job postings they are interested in or have applied to by clicking the **+** button next to each job posting.
    - Saved job postings are stored in the saved_jobs MySQL table.
    - The saved job postings are then displayed on the **Saved Jobs** tab, showing the same job details (Company Name, Role, Location, Link).

4. Managing Saved Jobs

- Users can delete a saved job by clicking the **-** button next to each saved job posting.
- Deleted jobs are removed from the saved_jobs MySQL table and the interface is updated accordingly.

5. Data Management

- The website uses three MySQL tables:
    1. user_info – stores user details for sign-up and login.
    2. job_posting_info – stores job postings scraped from job portals.
    3. saved_jobs – stores jobs that users have saved.

System Architecture:

1. Front-End:

- HTML, CSS, and JavaScript are used to build the user interface.
- The interface includes a search form for role and location input, and buttons for saving or deleting jobs.

2. Back-End:

- Django framework handles user authentication, job searches, and data management.
- A selenium web scraping script is triggered upon user searches, fetching data from job portals.

3. Database:

- MySQL is used to store user information, job postings, and saved jobs.
- Tables:
    1. user_info: stores user data (username, email, hashed password).
    2. job_posting_info: stores job details (company name, role, location, link).
    3. saved_jobs: stores the user's saved job postings.

Project Workflow Report

1. Planning Phase:

- Requirements were gathered and analyzed.
- The project was divided into three major components: Front-End, Back-End, and Database.
- Technologies were selected: Django for back-end, MySQL for the database, HTML/CSS/JS for front-end, Selenium for webscraping, and Docker for deployment.

2. Design Phase:

- **Front-End Design:**
    - A simple, user-friendly interface was designed using HTML, CSS, and JavaScript.
    - Two main tabs were created: the **Search Tab** for searching jobs and the **Saved Jobs Tab** for tracking applied jobs.
- **Back-End Design:**
    - Django was used to handle user registration, authentication, and job searches.
    - The selenium web scraping script was integrated into the search process to retrieve job postings dynamically.

3. Development Phase:

- **User Management:**
    - A user authentication system was built, allowing users to sign up and log in.
    - The MySQL table user_info was created to store user credentials securely.
- **Web Scraping:**
    - A Python Selenium script using web scraping libraries was developed to scrape job postings from various job portals.
    - The scraped data was stored in the job_posting_info table.
- **Job Search and Saving Jobs:**
    - After users submit search parameters, the scraping process is triggered, and the job postings are displayed.

- Users can save jobs they are interested in by clicking the **+** button. Saved jobs are stored in the saved_jobs table.

- **Managing Saved Jobs:**
  - Users can view saved jobs in the **Saved Jobs Tab** and delete them by clicking the **-** button.

4. Testing Phase:

- The system was tested for:
  1. **User authentication**: Ensuring only valid users can access their accounts.
  2. **Web scraping**: Checking that the correct job postings are retrieved and displayed.
  3. **Saving and deleting jobs**: Verifying that saved jobs are added to and removed from the saved_jobs table correctly.

ER Diagram:-