COSC617 Group Project Proposal

Fatima Malik

Devere Weaver

Betelhem Woldemichael

Amanda James

Ahmed Hamza

Overall Goal

The goal of the project is to develop an intelligent web application that automates and optimizes the entire job search process for users. This application will help users track their applications, organize their workflow and, also it will provide them with tailored job recommendations based on their resume and LinkedIn profile. By automating time-consuming tasks and offering personalized insights, the system will ensure that job seekers stay organized and focused throughout their job search journey.

Use cases

The Job Search Management web application automates the job search experience by connecting with LinkedIn to collect job details and relevant data effectively. The platform evaluates user profiles after they register and submit resumes to provide tailored job recommendations. Users can monitor the status of their job applications, saved, in progress, received, rejected or pending by using a dashboard. This functionality guarantees that no potential job openings go unnoticed.

The application sends reminders and alerts to remind users to check on their applications and get ready for interviews. It keeps users updated on the employment opportunities by checking LinkedIn in real time.

The Automated Job Search Management web app offers users an efficient way to manage their job hunt effectively and take charge of their employment journey, with tips and updates tailored to their needs and ambitions, in the job market.

Proposed Features

Core Features:

1. User profile:
    a. Allow users to set up an account and user profile to keep track of their related information
2. Job Collection and Aggregation:
    a. The app will extract job listings and descriptions from LinkedIn (and potentially other platforms like Indeed or Glassdoor). It will store relevant job data such as the job title, company, location, description, salary range, and application deadlines.
3. Personalized Job Recommendations:
    a. Based on a user's uploaded resume, LinkedIn profile, and preferences, the system will suggest jobs that match the user’s experience, skills, and

career aspirations. The suggestions will improve over time as the app learns from user feedback and interaction.

4. Job Application Tracking:
    a. Users will be able to track their job applications in a centralized dashboard, with status indicators (e.g., "Saved," "Applied," "Interview Scheduled," "Accepted," "Rejected," "Waiting"). The app will notify users about upcoming deadlines, follow-ups, and next steps.
5. Automated Reminders and Follow-Ups:
    a. To ensure users stay on top of their applications, the app will send timely reminders to follow up on applications, interviews, and other important tasks. This feature helps eliminate missed opportunities by keeping users proactive.
6. Dashboard for Progress Visualization:
    a. A user-friendly dashboard will display a comprehensive view of the job application status, showing both upcoming tasks and long-term progress. It will also give insights on trends, such as the number of applications sent, interview invitations, and response rates.
7. Job Search Insights:
    a. The system will analyze patterns in users' job search activities and offer insights, such as which skills are most commonly requested in job postings or areas where the user might need further development.

Other Features:

To build the web application, we'll need to leverage some of the following technical features. The primary technical features are what we'll need to implement a bare minimum working web application, while the other technical features will ensure it is fully featured and deployed for user interaction.

Primary Technical:

1. Database Management

- Use either and Nosql documents store (e.g. MongoDB) or an SQL store (Postgres, MySQL, etc.) to persist user data, this choice will depend on how we choose to model our data

2. Frontend

- We'll use the React library (or Next.js framework which is built on top of the React
  library) to create a dynamic user interface

3. Backend

- In keeping with the course objectives, we can use Node.js with Express.js for our backend logic

Other Technical:

4. API Integration

- If possible, we can use different API integrations to pull data from job hosting sites (e.g. LinkedIn, Indeed, Glassdoor, etc.) that we can then store and use to recommend for our users

5. Security and Authentication

- Open Authorization (OAuth) - might be necessary if we implement the features that allow the web application to connect to third-party services (e.g. LinkedIn, Glassdoor, etc.) to pull user account information and/or job postings; also allows users to login in with different account services using our web app
- OAuth is also potentially useful for allowing notifications from our web application when using third-party services (e.g. Gmail, Slack, Google Calendar, etc.)
- Password Hashing - to store user data we'll need to securely hash them before storage, a library such as argon2 or bcrypt.js will be used for this task.

6. Deployment

- There're several ways to deploy the final web app (e.g. AWS, GCP, Heroku, etc.) however it might be easiest to use Firebase which offers a free tier and is very easy to set up and deploy live web applications

7. Job Recommendation

- Since we're proposing to have an "intelligent job recommendation" as part of the core features, we'll likely need some kind of ML technique and libraries
- As an example, we can use a library such as LangChain & LangGraph for prompt parsing to extract resume skills & experience from user documents, reinforcement learning via user interactions, and semantic matching for matching jobs with resumes.

8. Web Scraping

- This isn't a necessary component but is potentially useful if there are hard limits on the APIs we intend to integrate into our platform, this exists as a backup plan to pull data from these sites

GitHub

https://github.com/AmurdAmzer/Job-Search