AI-Powered Job Application Analyzer & Resume Enhancer: ResuMate

Khalid Karim
Alexander Potiagalov
Mohamed Refaai
Manan Mehta

Repository link:
https://github.com/CMPT-276-SPRING-2025/final-project-lakes

1) With approval from the TA, finalize the 2 APIs you will be using for the project
- Include a brief description of each API and how it will be used in the project.

For this project, we have finalized two APIs: **JSearch API** and **OpenAI API**. The **JSearch API** is used to fetch real-time job listings, providing access to the latest opportunities from Google for Jobs. This ensures that users can apply to the most current postings while benefiting from advanced filtering options based on salary, qualifications, and job responsibilities. Additionally, it provides detailed job descriptions, allowing users to make well-informed decisions before applying. The **OpenAI API** enhances the job application process by generating personalized resumes and cover letters, extracting key skills and location details for better job matching, and offering AI-powered interview preparation to help users perform confidently in interviews.

2) Determine the features you will be implementing for each API (6 total)
- This should include a detailed description of each feature and how it will benefit the user
- Include any changes made to the features since the proposal

JSearch API (3 Features):

1) Real-Time Job Listings
- Access the latest job postings and salary information from Google for Jobs in real-time. This ensures that users can apply for the most current opportunities available in the job market.
2) Advanced Job Filtering
- Send info from the OpenAI API to filter job postings based on user preferences. This includes filtering by job salary, description, qualifications, and responsibilities.
3) Job Salary, description, qualifications and responsibilities
- Provide detailed job salary, description, qualifications, and responsibilities for each listing. This helps users make informed decisions about the positions they are interested in.

OpenAI API (3 Features):

1) Resume and Cover Letter Personalization:
- Generate highly personalized resumes and cover letters based on user qualifications and job descriptions. This helps users create tailored documents that stand out to potential employers.
2) Skill, Location, and Other Things Extraction for Matching:
- Extract key skills, location, and other relevant details for optimized resume content and job matching. This ensures that users match their most relevant experiences and qualifications to the job postings.
3) Interview Preparation:
- Generate potential interview questions and answers based on the user's resume and the job description. This feature helps users prepare effectively for job interviews by providing tailored practice questions and tips.

3) Convert the low-fidelity storyboard to a mid-fidelity prototype

- This should include a more detailed design of the application interface
- The prototype should be interactive and demonstrate the user flow
- It is expected that the prototype will be developed using a design tool (list of tools available here)

Use the below link to open mid-fidelity prototype in Figma :
https://www.figma.com/proto/5o2TcnyMHyYblsEtf1zmjd/Untitled?node-id=4-80&p=f&t=aPWfKZDl3BMjS7fZ-1&scaling=scale-down&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=4%3A

4) Choose a SDLC model that you will be following for the project

- This SDLC must be based on one of the variations introduced in class
- Briefly explain why you chose this model

For this project, we will follow the **Agile (Scrum) SDLC model** due to its flexibility, iterative approach, and suitability for a small team working with evolving API dependencies. Agile allows us to develop ResuMate in incremental sprints, ensuring continuous improvements based on feedback. Given that our project integrates multiple external APIs (OpenAI, JSearch, Indeed, and Glassdoor), Agile enables us to quickly adapt to any changes, such as API limitations or performance issues. Additionally, Agile promotes efficient collaboration among our four team members, with well-defined roles, regular check-ins, and continuous deployment using Vercel. This model ensures that key features—resume analysis, job matching, and cover letter generation—are tested and refined progressively, leading to a more reliable and user-friendly final product.

5) Develop a work breakdown structure (WBS) of all the tasks associated with the project

Prioritize tasks based on dependencies and importance
By the end of this milestone, these tasks should be written as tickets (Github Issues) and added to the github project

1️⃣ Project Setup & Planning

- Set up GitHub repository (Project board, Issues, Milestones, Branches)
- Finalize UI wireframe & design
- Set up Next.js project (with ShadCN UI)

2️⃣ Frontend Development (React)

2.1 Home Page (Landing Page)

- Create a landing page UI with an introduction to ResuMate
- Add navigation buttons: "Upload Resume", "Analyze Resume", "Find Jobs"

2.2 Resume Upload & Parsing (Client-side)

- Implement resume file upload (Accept .pdf, .docx, .txt)
- Extract text content from the resume (Using JavaScript libraries like pdf.js)
- Store extracted text in local state for further processing

2.3 Resume Analysis (OpenAI API)

- Send extracted resume text to OpenAI API for feedback
- Display AI-generated resume score
- Show feedback on ATS compatibility, improvements, and missing skills

2.4 Cover Letter Generator (OpenAI API)

- Implement text input for job description
- Send job description & resume content to OpenAI to generate a tailored cover letter

- Allow users to edit, copy, or regenerate cover letters

2.5 Keyword Optimization (OpenAI API)

- Extract keywords from job description
- Suggest missing keywords based on AI analysis
- Provide an option for users to manually add keywords
- Store optimized resume keywords in local state for job searching

2.6 Job Search & Matching (LinkedIn/Indeed API)

- Implement search bar for job listings
- Use LinkedIn Jobs API or Indeed API to fetch relevant job listings
- Display jobs that match the user's resume & skills
- Implement filters (location, experience level, remote/in-person)

2.7 Job Matching System (AI-Based Ranking)

- Implement a ranking system using AI-generated insights
- Sort job listings based on best fit for the user
- Add "Apply Now" buttons linking to external job applications

---

3️⃣ API Handling & Frontend Optimization

- Implement proper API call handling (rate limits, retries, error messages)
- Optimize API requests for performance & minimal latency
- Handle API failures gracefully (fallback UI, alternative data suggestions)

4️⃣ UI/UX Improvements

- Add animations and UI enhancements
- Ensure mobile responsiveness (Tailwind CSS breakpoints)
- Improve form usability (clear instructions, validation messages)

5️⃣ Testing & Deployment

- Perform unit testing on major components
- Ensure API responses are correctly handled
- Deploy the application on Vercel
- Finalize technical documentation & user guide

Task Prioritization & Dependencies

✅ High Priority (Critical Path)

- Resume Upload & Parsing (Text extraction must work before AI analysis)
- Resume Analysis (OpenAI API) + Cover Letter Generator
- Keyword Optimization & Job Matching

✅ Medium Priority

- Job Search UI & Filters
- Client-side API handling optimizations

✅ Lower Priority (Enhancements)

- Animations & UI refinements
- Alternative search API integrations (if LinkedIn API is limited)

6) Project schedule with milestones and deadlines

This should provide a timeline for the project, including when each feature will be developed and tested
It is recommended to schedule internal deadlines and include buffer days between the internal and actual deadlines

Milestone,Task,Deadline:

1) Frontend Development - Part 1,"Implement Home Page UI, Resume Upload & Parsing", Due: Mar 9
2) Frontend Development - Part 2,"Resume Analysis (OpenAI API), Cover Letter Generator", Due: Mar 13
3) Frontend Development - Part 3,"Keyword Optimization, Job Search & Matching (LinkedIn/Indeed API)", Due: Mar 16
4) M1.5 Check-in,Internal report preparation + meeting with TA, Mar 17-21
5) Frontend Development - Part 4,"Job Matching System (AI-Based Ranking), API Handling & Optimization", Due: Mar 23
6) UI/UX Enhancements,"Improve animations, mobile responsiveness, and usability", Due: Mar 27
7) Testing & Bug Fixes,"Unit testing, handling API failures, user testing", Due: Apr 2
8) Deployment & Final Report,"Deploy on Vercel, finalize documentation, prepare presentation", Due: Apr 5
9) M2 Final Delivery,"Submit report, present final project, code handover", Due: Apr 8

7) A risk assessment of potential issues that may arise during the project and how you plan on mitigating them

Risk Assessment & Mitigation Plan

Low-Risk Issues

1. API Rate Limits
    - Risk: Exceeding request limits for OpenAI API or job listing API.
    - Mitigation: Implement caching, request batching, and monitor API usage.
2. Minor UI/UX Bugs
    - Risk: Small display inconsistencies or usability issues.
    - Mitigation: Regular UI testing, user feedback collection, and iterative improvements.
3. Feature Scope Creep
    - Risk: Adding unnecessary features beyond project requirements.
    - Mitigation: Stick to the defined feature list, prioritize MVP, and set deadlines.
4. Version Control Conflicts
    - Risk: Merge conflicts when multiple team members work on the same files.
    - Mitigation: Use Git branches effectively, conduct code reviews, and resolve conflicts early.
5. Data Format Inconsistencies
    - Risk: Job data from different sources may have variations in formatting.
    - Mitigation: Implement standardized data processing and validation functions.

Medium-Risk Issues

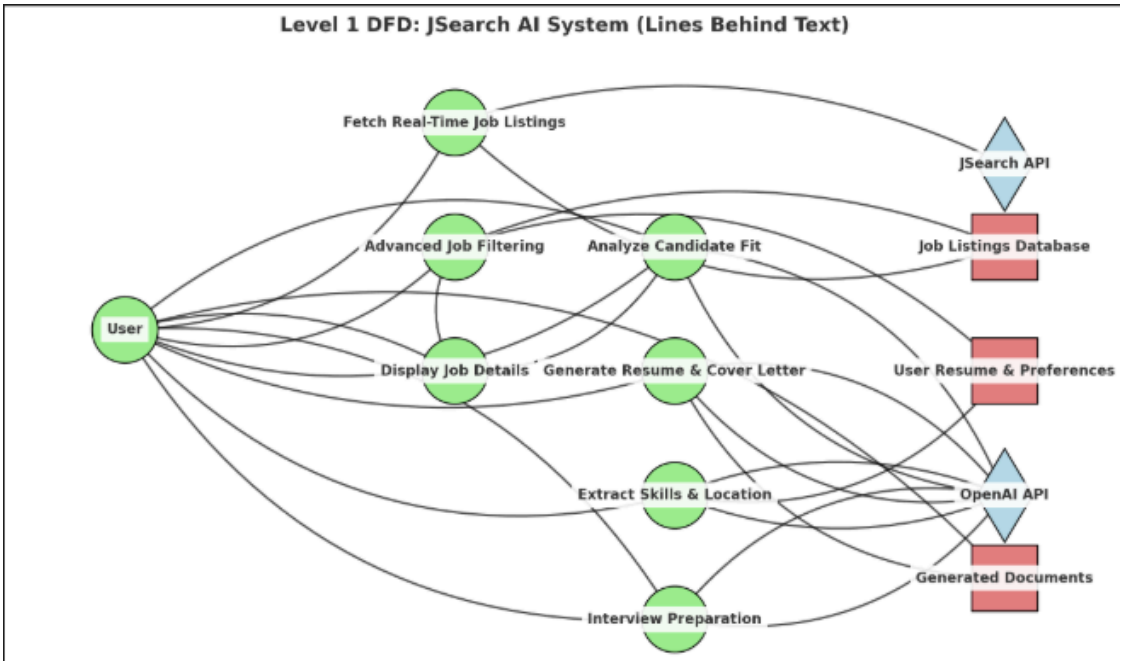
6. Data Privacy Concerns
    - Risk: Handling sensitive user information like resumes.
    - Mitigation: Avoid storing personal data, use encryption for transmission, and comply with privacy regulations.
7. Dependency on Third-Party APIs
    - Risk: Service downtime or API changes from Google Jobs or OpenAI.
    - Mitigation: Implement error handling, caching for recent results, and explore alternative APIs.
8. Bias in AI Recommendations
    - Risk: OpenAI's model may introduce unintended biases in job matching and cover letter generation.
    - Mitigation: Implement bias-checking mechanisms and allow user adjustments.
9. Scalability Issues
    - Risk: Performance degradation with a high number of users.
    - Mitigation: Optimize API calls, use pagination, and conduct performance testing.
10. User Adoption & Engagement

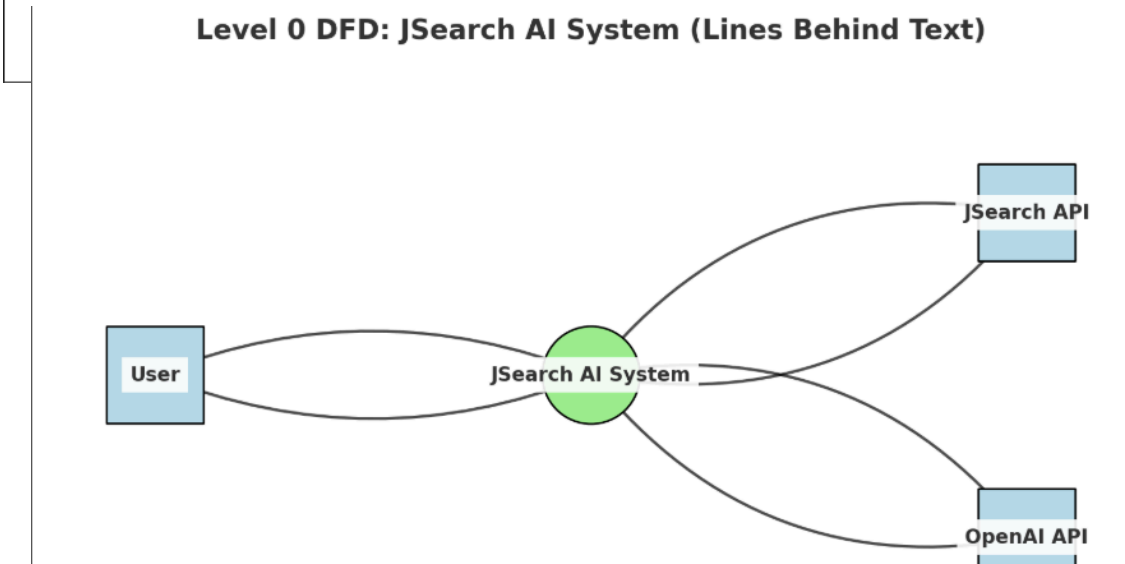
- Risk: Users may not find the tool intuitive or useful.
- Mitigation: Gather user feedback, improve UI/UX, and iterate based on user needs.

High-Risk Issues

11. Inaccurate Job Matching
- Risk: AI may suggest irrelevant job matches.
- Mitigation: Improve NLP filtering, allow user feedback, and refine keyword extraction.
12. Legal & Compliance Risks
- Risk: Copyright, licensing, or compliance violations with job data or AI-generated content.
- Mitigation: Review API terms, add disclaimers, and avoid storing job data.
13. Security Vulnerabilities
- Risk: API abuse, unauthorized access, or data breaches.
- Mitigation: Implement authentication, secure API keys, and follow security best practices.
14. Project Timeline Delays
- Risk: Missed deadlines due to unforeseen issues.
- Mitigation: Use agile development, set milestones, and track progress with weekly check-ins.
15. AI Hallucinations & Inaccuracies
- Risk: OpenAI API generating incorrect or misleading job descriptions/responses.
- Mitigation: Implement verification mechanisms and allow manual user adjustments.

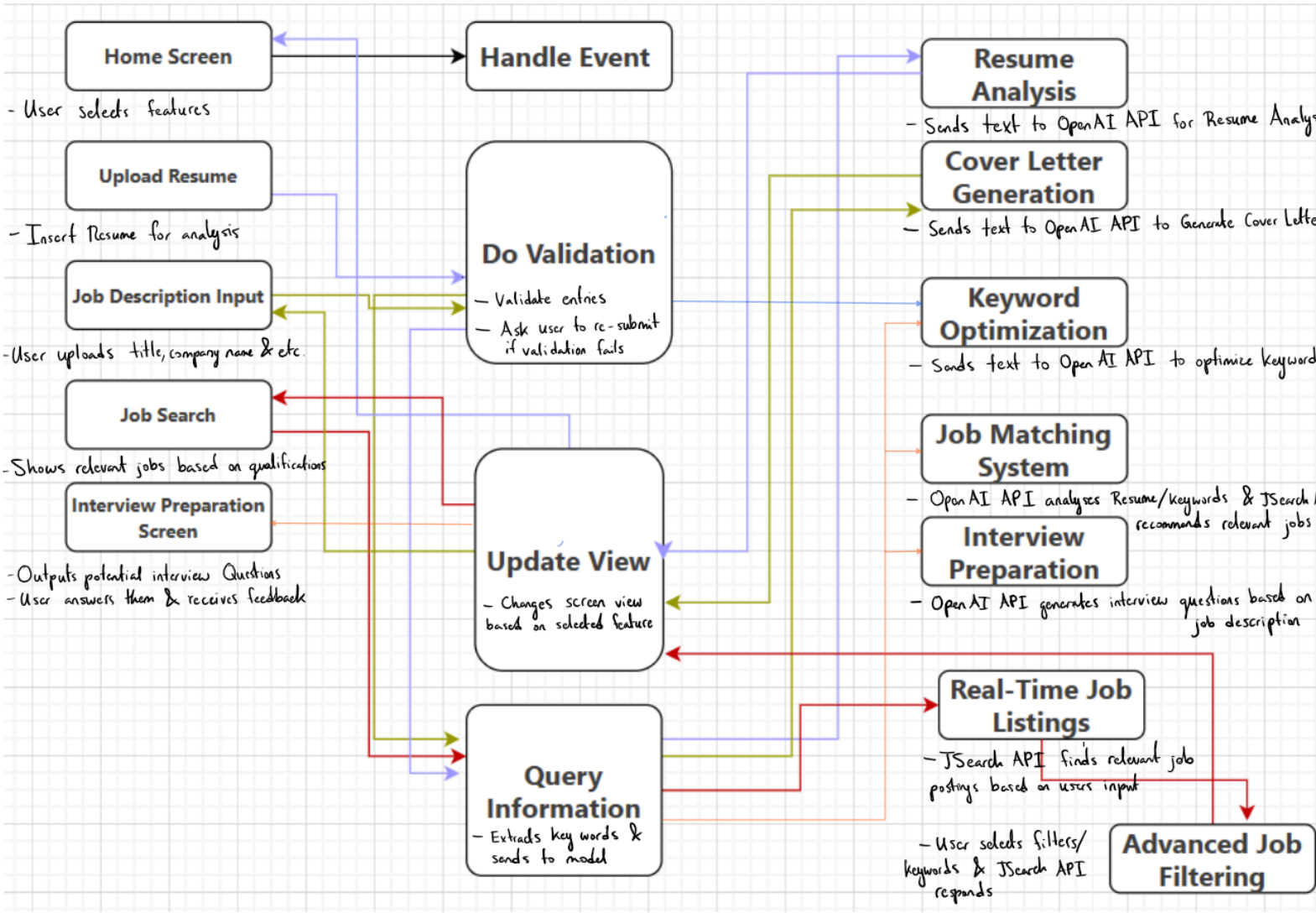
8) Use Data Flow Diagrams (level 0 and level 1) to outline how data flows within the application
Between the APIs and your application





9) MVC model diagram for the application

This should include a high-level overview of how the application will be structured and a description of each component



10) Appendix with any additional information that was not included in the main report

Provide a list of all group members and their detailed contributions to the project

Provide a changelog table that includes any revisions since the proposal

Any additional diagrams, charts, or tables that were not included in the main report

Group Members and Contributions

1. Alexander Potiagalov –Project/Repo Manager, API Integration, Job Matching System Implementation
2. Manan Mehta – Frontend Developer, UI/UX Design, Resume and Cover Letter Generator
3. Khalid Karim – AI Integration, Resume Analysis, Keyword Optimization
4. Mohamed Refaai – API Handling, Testing & Deployment, UI/UX Design

Change Log

- Merged *A brief description of the features you plan to implement* and *User stories*, addressing the first feedback: *“Make sure you're not repeating the user stories for each API feature. The sections 'A brief description of the features you plan to implement..' and 'User stories...' have almost the same content written down. Maybe merge these two under one section.”*
- Changed API keys and replaced LinkedIn API due to restrictions, opting instead for OpenAI API & JSearch API.
- Created a website prototype and established connections between pages to address the second feedback: *“Do more research on how to design low-fidelity storyboards. You need to draw your website's prototype and make connections between pages wherever needed.”*