Federica Selvini | Assignment 01

**Useful links:**

Streamlit app: https://selfede-pdai-mindthegap-a01-app-iumyao.streamlit.app/

Github repository: https://github.com/selfede/PDAI_MindTheGap_A01.git

**Solution overview**

The traditional tracking of job applications is a fragmented, low-value, and time-consuming process. First, copy-pasting each application information into a static spreadsheet like Excel or into a Notion table is time-consuming and error-prone. Secondly, students need to impractically upload a full stack of files into AI to share their academic history, copy-paste job descriptions, craft the right prompt to get personalized feedback, and later translate the latter into an attempt to find the right course to fill the gaps... Overall, a tedious work which could be conveniently centralized, streamlined, and automized. The solution is "Mind the Gap”: an AI-native ecosystem designed to help recent graduates navigate the gap from academic preparation to professional readiness.

- It eliminates the pain to manually record data by using AI to instantly extract info from the job descriptions into a structured, editable tracker.
- It allows to cross-reference the user’s academic profile against specific role descriptions, navigating with ease the gaps of one application compared to another.
- It automatically converts the tracking data into Plotly dashboards and performs an AI-generated gap analysis of the user’s skills, along with a tailored 6-month learning roadmap with direct links to bridge specific academic-to-market skill gaps.

**Target user:**

This solution is designed for university students and recent graduates transitioning into the workforce. This prototype specifically targets users who may struggle to translate their academic history (based on CV and university transcript) into professional technical requirements. It serves users who need a centralized, data-driven career center to manage high-volume job applications while identifying exactly what their strength are and how to close their market-entry gaps.

**Prototype user experience**

1. The user uploads their CV and University Transcript in the sidebar. The system extracts the text and stores it for the session.
2. In the "Application tracking" tab, the user drops multiple job description PDFs. The AI parses these to automatically populate a tracking table with the company, industry, and role.
3. The user uses a dynamic data editor to update application statuses or manually add/correct entries.
4. The user navigates to the "Analytics" tab to view interactive charts that visualize their application performance and their target industries distribution, as to visually keep track at a glance of where they’re focusing their job search efforts.
5. In the "Learning path" tab, the user selects a role from their tracker. The AI compares their profile against the job's requirements to generate a candidate summary and a list of top skill gaps, redirecting the user to targeted online courses.

**Technical stack**

For my prototype, I experimented with different Streamlit components drawing inspiration from the Streamlit API Reference, as well as browsing through Youtube. As foundation of my prototype is the OpenAI API, which I first used in the AI for Productivity and Personal Development skill seminar done in January. However, there I used credits made available from the professor. Asking Gemini for free alternatives, it recommended I used the Groq API. I set it up based on the playground instructions in Groq. Additionally, I used the libraries PyPDF2 for extraction of raw text from documents and Plotly, which we experimented with in our Python course from last term, for making the graphs.

**Prototype evolution**

Initially, I explored an idea that scraped LinkedIn profiles to compare a user's skills with those of current professionals in their target roles. However, working on a first prototype quickly made me realize that this raised critical privacy concerns and technical hurdles regarding bot detection. I also considered using a static dataset but discarded the idea as I wanted to better reflect the real-time changes in job market expectations. My second iteration involved scraping Glassdoor for job descriptions, but I again encountered authentication barriers and realized that skill requirements vary drastically across different industries for the same role, so it didn't feel very helpful. Ultimately, by reflecting on my own application experience and gathering peer feedback, I pivoted to targeting the specific job opportunities the user is interested in by analyzing PDFs directly. This change ensured the prototype was truly value-creating for the user, without compromising data privacy.

**Areas for improvement and next steps**

My primary challenge was adhering to the principles of a prototype and avoiding the "perfectionist trap". By staying loyal to the characteristics defined in class, I ensured the project remained tangible and explorable while maintaining a low-fidelity approach. It is willingly an approximation of a larger vision that will still require quite some work. This version is disposable, as I discarded several versions to validate the most effective one, and lean, as I prioritized time efficiency and utilized Groq's free tier over paid OpenAI credits.

While this current prototype identifies gaps and suggests courses, a critical next step is moving from theoretical knowledge to practical evidence. I aim to implement a "Portfolio builder" tab to bridge this gap. Instead of just listing a course, the AI will brainstorm unique projects ideas that users can build to create "proof of skill" for their portfolio, advancing the user one step further in the full skill-building lifecycle.

Future iterations will also focus on technical efficiency. I plan to enable persistent storage for development plans within the data frame so data is not lost at the end of a session but can be exported as part of the csv. Additionally, I plan to refine the AI's logic to compare different providers, such as Coursera, DataCamp, or Udemy, based on the specific technicality of the skill. Finally, I aim to smooth out the user experience, particularly making edits to the application tracking table smoother and more responsive.

**Use of AI**

Throughout the development of my prototype, I used a "patchwork" approach, combining examples from our classwork with official Streamlit documentation to build the code. Given this approach, I resorted to AI as a coach and technical mentor: it was especially helpful in debugging and in figuring out

how the Streamlit components interact and depend on each other, and where state management and widget refreshes tended to fail. I also used it to improve my prompt engineering based on the techniques learned in the AI for Productivity and Personal Development skill seminar. It was especially helpful to refine the system instructions to strictly control the model's output and force it to return a JSON-structured response for the parsing and gap analysis features.