ICSI499 Capstone Project Report

GenAI-based Multimodal User Profile Generator

Project Team

Alexander Crowe (001339469)
Manfred Fong (001483891)
Samuel Ginzburg (001470893)
Bryan Gomez (001472439)
Elijuwon Mitchel(001487571)

.....
College of Engineering and Applied Sciences
University at Albany, SUNY

Project Sponsor

Pradeep Atrey & Chidansh Bhatt
patrey@albany.edu & chidansh@gmail.com

May 6, 2024

Acknowledgements

A heartfelt thank you goes out to our sponsor, Doctor Chidansh Bhatt, for his management throughout this project over the past semester. We thank him for his team management, organization, knowledge, and emphasis on deadline adherence, regular weekly meetings, and professional standards. We would also like to thank Professor Pradeep Atrey for extending his hand and reaching out to his peers to allow us to work on such a captivating project. This experience has expanded our knowledge and skill set and is a pivotal stepping stone in the journey we will take down the road as we continue our careers.

We would also like to thank our friends and family for supporting us personally, emotionally, and financially throughout our college careers to pursue higher education and better futures. Too many to name, thank you all for your encouragement and support throughout this process.

Abstract

Our project introduces a chatbot which was created to assist users in job applications and hiring. Developed primarily using Python and various libraries within the language,the Langchain framework, OpenAI as the prominent LLM, Streamlit for user interface, Proxy-Curl for web scraping, and Tavily Search Engine as a general internet request interface. By applying LangChain agents accompanied by Prompt Engineering, the chatbot can provide individualized tips to users who input their resumes and LinkedIn profiles. By means of our chatbot, users can polish their resumes, enrich their resume-relevant skills, and search for appropriate job vacancies. This project demonstrates the practical side of chatbots, which showcases its capability as a chat assistant with the integration of AI technology and gives tailored advice to job seekers.

Artificial intelligence (AI), a transformative force shaping modern society, has experienced increasing adoption and advancement over recent years. A myriad of AI-powered tools now empower users to manipulate various facets of data and processes, from analyzing patterns and making predictions to automating tasks and more. However, amidst these strides, a fundamental aspect of AI—the ethical considerations and potential biases inherent in its algorithms—often goes unaddressed.

Introducing the Generative AI chatbot for job seekers. Harnessing the power of natural language processing and machine learning algorithms, our chatbot allows users to upload their resumes and job listings, providing personalized feedback and suggestions for enhancing their resumes. Whether fine-tuning your professional summary or aligning your skills with job requirements, this tool bridges the gap between job seekers and their dream positions.

Contents

1 Problem Analysis

1.1 What problem you are trying to solve?

In today's competitive job market, it's imperative to differentiate yourself as an applicant. A one-size-fits-all resume no longer suffices amidst the abundance of applications. It's crucial to customize your resume according to the specific positions you're pursuing. While this tailored approach may be time-consuming, especially given the high volume of applications, it's an essential step toward securing opportunities. Our project aims to streamline the tailoring process by providing users with an AI assistant that enhances user's resumes based on specific job descriptions and requirements. The core problem we're addressing is the inefficiency of traditional resume editing and tailoring methods which this AI attempts to solve.

1.2 Why is this an important problem?

Crafting a resume that effectively showcases one's skills and experiences for a specific job can be challenging and time-consuming, often leading to missed opportunities or mismatches between candidates and positions. Streamlining this process can lead to a higher quality of match between employee and employer, leading to longer-lasting employees and a higher efficacy of work produced over the lifetime of the job.

1.3 How is this a challenging problem to solve?

Editing resumes and reading job applications is time-consuming. Often, certain skills and certifications are omitted from a resume as they are not relevant to the larger proportion of jobs available and most often applied to. In applying to different jobs we come into the issue of a particular candidate's skills may not fit on a simplified resume. Additionally, the range of industries and applications is vast and comes in many different forms, this is where we utilize AI to adapt to different situations. However, AI is not a perfect tool and comes with its own challenges such as 'hallucinations', incorrect information returned from the LLM with a high degree of certainty, and unpredictable outcomes if not prompted and handled properly.

1.4 What has been done to solve the problem or what are the existing solutions?

OpenAI's API was the primary agent, we exposed it to data provided by the user, ProxyCurl API, and utilized prompting techniques to allow AI agents to analyze key points about the person and the job they are applying to. With proper guidance, we were able to pull consistent results from the AI to aid the user. This technique gives us the benefit of automatic analysis, removing the burden from the user, and eliminating the process of manually filling in boilerplate templates.

1.5 Why existing solutions are insufficient or what are their key weaknesses and limitations?

The shortcomings of existing solutions, exemplified by template resumes, are multifaceted and hinder the efficiency of the job application process. While templates offer a starting point, they still necessitate significant manual effort from the user. Individuals must painstakingly review and reprocess their resumes, discerning which key points to emphasize and which to omit, a time-consuming and often laborious task. This manual approach not only consumes valuable time but also introduces the potential for oversight or misrepresentation of one's qualifications. In contrast, our innovative solution addresses these shortcomings by offering an automated and expedited process. By leveraging advanced algorithms and natural language processing capabilities, our platform streamlines the resume customization process, empowering users to tailor their resumes swiftly and seamlessly for each job application. This automation not only saves time but also ensures a more precise alignment between the applicant's qualifications and the job requirements, enhancing their chances of securing meaningful employment opportunities.

1.6 What is your core idea and solution?

Our core idea and solution involve leveraging AI technology to provide personalized resume assistance tailored to each user's unique skills and experiences, as well as the specific requirements of the job they're applying for. By analyzing job descriptions, requirements, and recommended qualifications, our AI assistant can suggest relevant skills to highlight, recommend additional information to include, and guide formatting and presentation.

1.7 What are the key characteristics of the proposed solution?

ProxyCurl is an API that we utilize that is fundamental to the application. It allows us access to data for a user's LinkedIn profile and a job description which is then provided to the LLM for processing. In addition, we used LangChain, a Python framework designed by Harrison Chase for developing AI applications. This framework provided us with agents that can see the functions and tools we provided and behave to use a Large Language Model (LLM) as the decision-making engine for our processes. It was the bridge that connected the data we collected, and the large language model that we utilized. To gather additional data from the internet the Tavily search engine was utilized. The Tavily API allows our application to gather information from various web pages to answer the user's prompts best.

1.8 How your solution is novel and better than existing solutions?

Our platform takes away a lot of effort that a user would have to put in compared to template resumes, which are nothing but a start toward building a resume. Our automated system customizes the resume instantly, highlights all the important pointers, and maximizes the content for each application. Our conversational feature, allows users to re-prompt the chatbot allowing it to act as a proper virtual AI assistant guaranteeing the proper results.

The combination of all these features saves time and reduces the chances of missing out on some details or misrepresenting qualifications. Our solution provides a sophisticated algorithm and natural language processing, which guarantees that the candidate's skills are aligned well with the requirements of the job. This increases the candidate's probability of getting an opportunity for gainful employment.

1.9 Key Contributions

- Design and develop an intuitive and simple interface similar to existing chatbots to decrease the time taken to become familiar with the application.
- Develop a conversation feature, that utilizes the LLM and co-reference memory, allowing our chat-bot to act as a proper virtual assistant
- Give users the ability to upload and save their resume so that our virtual assistant can provide proper recommendations to improve it based on the data provided through prompts or data found through web searches.

1.10 Organization of Report

This report examines the issue of optimizing resume content to align with specific job listings. It describes the shortcomings of conventional methods, such as manual resume editing, and introduces our innovative solution, the GenAI-based Multimodal User Profile. Generator. By harnessing AI technology, our chatbot facilitates automatic resume updates tailored to the requirements of individual job postings, offering a more efficient and effective approach for job seekers.

2 Proposed System/Application/Study

2.1 Overview

Our proposed system is an AI-powered assistant designed to streamline the process of resume tailoring for job applications. By leveraging AI technology, our system analyzes job descriptions and user resumes to provide personalized recommendations for enhancing resume content and formatting. This section will provide an overview of the system's architecture, components, and functionality. Readers can expect to learn about the key features of our AI assistant and how it addresses the challenges associated with traditional resume editing methods. Additionally, we will outline the organization of subsequent sections, including methodology, implementation, and evaluation.

2.2 Project Requirements

2.2.1 User Classes

Member Users: Member users will have access to the entire capabilities of the platform. This includes general chat features, resume alteration capabilities, and analysis. Membership registration requires the user to provide their resume and LinkedIn profile. The platform is exclusively available on desktop and mobile as a web application.

2.2.2 Design Requirements

Device Compatibility: The platform is compatible with both desktop and mobile devices as a web application.

2.2.3 Functional Requirements

- **Function Requirement: Streamlit**
  - This functionality will allow an 'out of the box' front-end to our application that can be easily deployed to the cloud.
- **Function Requirement: Upload a LinkedIn profile of the user**
  - This functionality will allow users to upload their profile to allow the GenAI to pull information from it during analysis and use that information to modify its certainty on resume topics.
- **Function Requirement: Upload a resume document**
  - Users should be able to upload a document of their resume, to allow the AI to analyze for suggestions. This document is temporarily stored in a hashed state

to prevent data leaks but allows for updates to resumes on the cloud for faster processing.

- **Function Requirement: Chat features**
  - Users should be able to interact with the AI as a regular chatbot, speaking casually as well as focusing on their specific task of updating and enhancing their resume.
- **Function Requirement: Resume analysis and output**
  - With the necessary data provided by the user, AI should be able to do an analysis with the provided information and return an edited resume tailored to the job application.
- **Function Requirement: Algorithm Design and Analysis**
  - Our system utilizes AI capabilities for processing and comparing large volumes of textual data, ensuring fast and accurate responses to user queries.

2.2.4 Non-functional Requirements

Performance. The cornerstone of an excellent Virtual assistant is appropriately generated output and desirable responses. Regardless of the algorithmic intricacies of the LLM, users should not experience incorrect or inappropriate responses e.g. 'hallucinations'. We aspire to deliver the correct results with the proper citations based on previously provided user context.

Security. Resume and online information are personal, records of a person's professional experience. Every data point uploaded and the information saved, therefore, should be properly encrypted within the local memory and MongoDB. In addition to the user's uploaded data being encrypted, due to the nature of a chatbot, all conversation sessions and each input of the user and the output of the chatbot are saved and, therefore, must be encrypted. Each conversation session and message widget is assigned a session ID and is encrypted using Python's secrets library and Shah-256. This encryption offers proper security and protects the user's information from malicious cyber-attacks.

Reliability. The generated response should invariably reflect the user's previous prompts and provide context, to properly provide the most accurate and credible answers.

Availability. The chatbot should act as an assistant, helping the user form or update their resume. Due to this, the user needs a proper internet connection and the right prompts so that the chatbot can generate a desirable response.

Maintainability. Our group will strive for clearly and concisely documented and modular code to ensure readability and ease future maintenance.

2.2.5 Design and Implementation Constraints

Our project faced significant implementation constraints, primarily due to our choice of technology and the time available for development. We decided to use LangChain, a relatively new Python framework that, while promising, presented challenges due to it being a relatively new framework. This means there was limited availability of resources and community support, which added to the learning curve we faced. Furthermore, frequent updates to the framework meant that we had to continually adapt, leading to a less stable development process. These technological challenges were further intensified by the time restraints we faced as students, which limited our ability to delve deeply into the framework's capabilities or optimize our solution comprehensively.

In terms of Design content. Our primary front-end tool, Streamlit, runs scripts from top to bottom. Whenever a user interaction happens, such as pressing a button or adjusting a slider, the entire script is re-executed. This linear execution model conflicts with the concurrent execution model that asyncio is designed for, which is based on running tasks concurrently to manage I/O-bound and high-level structured network code. This prevented the use of some of the powerful tools that were previously developed and required us to switch to a subscription-based API that handled these tasks and abstracted away much of the technical conflicts.

2.3 Solution Architecture

Our unique generative AI chat assistant, developed using the top-down design technique, stands out for its carefully arranged UI and seamless user interface transition. We laid the foundation for the GUI framework and then focused on the main activities of the application, showcasing the technical capabilities and features of our AI chat assistant. Custom tools were developed to enable the chatbot to perform specific tasks: an app search engine based on Tavily API for general web searches, an in-app web scraper using ProxyCurl API to pull data from dedicated webpages, and a utility that can generate resumes based on entered data. The chatbot can autonomously lead the activity through prompt engineering, and LLM calls are directed to the agent executor, thus providing the most appropriate tool for the user issue.

These tools remain undetected in the background, giving the discretion to the agent executor to decide whether and in what order they should be used for a given task. The deployment of separate Chatbot utilities and Streamlit's highly intuitive UI components aided in improving the project's workflow and enabled an efficient implementation of the Generative AI system.

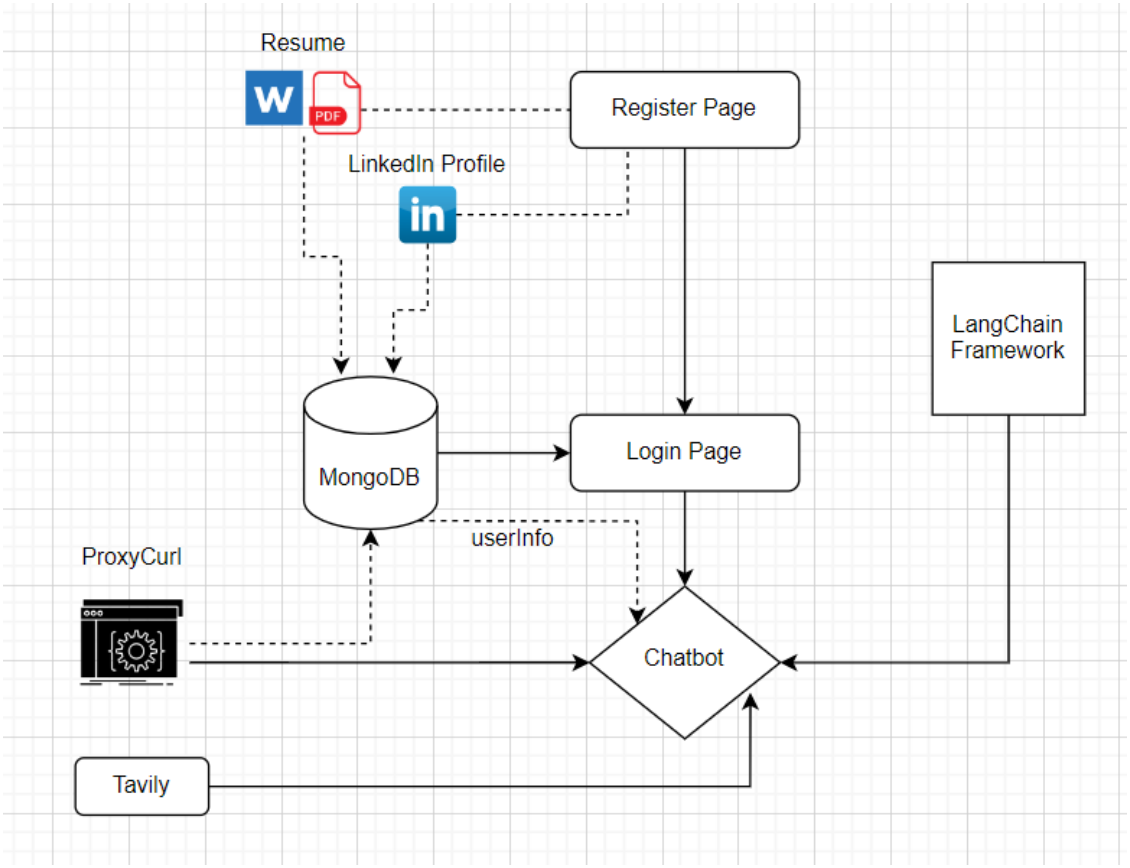

Figure 1: System flow diagram for the chatbot Application

The figure above details a system flow diagram of the chat assistant application. It shows the flow of the user experience of the screens interacting with some of the APIs and custom tools. As users interact with the application, they are brought along a logical path from registration, where they upload their data via resume upload and LinkedIn link. The mode of input is to the chatbot page, where they are presented with your typical chat page where they can prompt the A.I.
The figure above also details the data flow of the application. The API objects are used to retrieve text data from the user-uploaded resume and save it to MongoDB. The APIs and tools also scrape data from various online sources through Tavily and are saved through local memory. The generate response feature of the chatbot utilizes all this data to answer the user's prompts.

2.4 System Implementation

Since our deployment target is a web application, we opted for a fully web-based development ecosystem, using the Python programming language and Streamlit development environment to develop our user interface. For the user interface, we predominantly used the Streamlit framework dedicated to building user interfaces. To implement our conversational chatbot assistant, we utilized the Langchain framework [1] to interact with LLMs. We used the Tavily search Engine [11] and ProxyCurl API [4] frameworks to implement the web search and web scraping capabilities. We used OpenAI's [9] LLM, which grants our chatbot its decision-making capabilities.

For the primary function of our chatbot, we directly take advantage of the Langchain framework [1] and the OpenAI LLM [9] and the concepts of prompt engineering [2]. Using the above-mentioned tools, we allow the user to prompt the LLM through conversations with the A.I. Using these landmark points, the A.I first determine the primary goal of the user based on the initial prompt provided. Then, the agent executor provided by the Langchain framework and the OpenAI's LLM, the chatbot, decides which tool or API to use and the order in which they are used to generate the proper answer to the user. If the generated output is not desirable due to the conversational feature and the local memory functionality, which is capable of co-reference memory, the user can prompt the LLM such that a desirable response is generated.

We opted for Discord as our primary platform to streamline team communication and coordinate project tasks effectively. We maintained a dedicated repository on GitHub for source control and managing our codebase. GitHub served as the central hub for collaborative coding and version control. We used Streamlit for build testing and the distribution of beta tests.

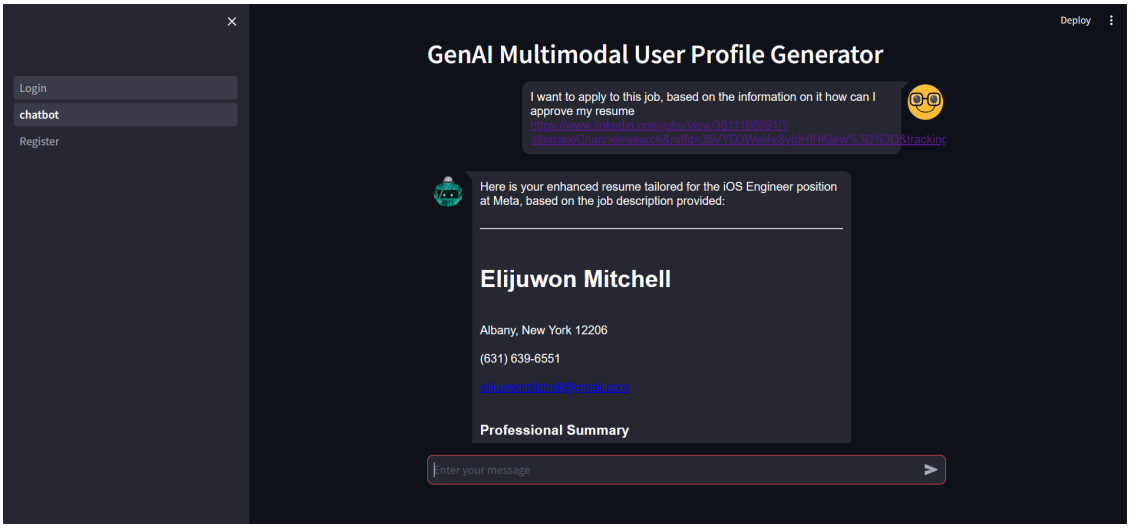

Figure 2: Screenshot of application GUI

The diagram above showcases the user journey within our application, encompassing three main screens: Register, Login, and Chatbot. Upon accessing the application, users are directed to the Register page, where they input their personal information and upload their resume. Following registration, users can log in using their credentials on the Login page to access the main functionality.

Navigating to the Chatbot page, users are greeted with a conversational interface where they can interact with the AI-powered chatbot. Here, users can prompt the chatbot with inquiries related to resume optimization and job matching. Additionally, users have the option to upload job links to provide context for the chatbot's suggestions and updates to their resume.

The seamless flow between these screens ensures a user-friendly experience, guiding individuals through the process of registering, logging in, and utilizing the chatbot to enhance their resume based on specific job requirements.

2.5 Use of Computer Science Theory and Software Development Fundamentals

2.5.1 Use of Computer Science Theories

Gen AI user-based multimodel profile generator employs several computer science theories. Amongst these theories, we note the following:

Prompt Engineering: The design of our UI was created to work as your typical messaging application. The idea behind the design was to take in user interaction as input and allow the LLM to generate an output. To ensure the proper output. We take advantage of prompt engineering. Prompt engineering is an essential tool that leads AI to make relevant suggestions or give appropriate answers to queries asked by users on job applications or hiring processes. Through the process of devising prompts suitable for each task—like the case of requesting resume feedback or job search assistance—prompt engineering ensures the AI provides useful and practical information.

Machine Learning: We incorporated the use of machine learning in our application by using pre-trained models provided by OpenAI. We used these models to implement an application that utilizes Retrieval Augmented Generative (RAG) techniques. These techniques were utilized to be provided as the foundation of our application

Database Management: In our application we utilized MongoDB and deployed database techniques to manage information for our application. This involved us designing and orchestrating data flow, and deciding what information to keep persistent.

2.5.2 Use of Software Development Fundamentals

We kept the following in mind during the development of GenAI-based Multimodal User Profile Generator:

Coding Standards: Adherence to coding standards is essential for successful software development. This is essential for developing readable and maintainable software. Our development team enforced meaningful variable and function names, consistent coding styles, and well-documented code to ensure readable code in our codebase. We use pull requests in our repository to peer review code changes and request necessary changes to enforce them. Refactoring: We made a significant effort to refactor our code to ensure it was compact and well-organized. We efficiently addressed technical debt and separated visual and functional components during implementation. Our project is well-organized, making updating APIs, algorithms, and user interfaces easy.

Refactoring: We focused on refactoring our code to make it more concise and organized. During implementation, we quickly addressed technical debt and separated visual and functional components. Our project is well-organized, making updating APIs, algorithms, and

user interfaces easy.

Version Control: We actively used GitHub for our version control system to track changes in our codebase and collaborate on code. We managed concurrent development by individually forking from our repository, branching to implement separate features, and using pull requests to make changes to the main branch. Each commit/PR was then tested with GitHub Actions, building the application to ensure continuous development was possible. These software development fundamentals allowed us to effectively work together on the same codebase without losing much time to deal with conflicts.

3 Experimental Design and Testing

3.1 Experimental Setup

3.1.1 Experiments

Experiment 1: API Web scrapping

- Objective: To test the effectiveness and accuracy of the user information provided by ProxyCurl API.
- Setup: Used a known web page with abundant and not-abundant information and sent it out to the API.
- Procedure: Analyzed the return JSON object pulled by the API for sufficiency and appropriate data.

Experiment 2: Generative Output

- Objective: To verify the accuracy of generative content from our RAG system.
- Setup: Employ a range of free-form prompts that simulate appropriate and inappropriate forms of interacting with our platform.
- Procedure: Deployed the free-form user prompts to in our platforms (considered many cases) and analyzed the RAG output.

Experiment 3: Tool Integration

- Objective: To evaluate the interaction of each tool. This included uploading user information, accessing information, analyzing the information, and output of the RAG system.
- Setup: Provided the LLM with all necessary tools and had each tool output its action input and output.
- Procedure: After integration, we fed several prompts to the platform and followed the data flow, keeping track of all the function's input and output, finally analyzing the final output made by the LLM.

3.2 Dataset

For this project, we use a humble data set about the user and information about a job posting provided by the user. The information about job postings, and LinkedIn base user information comes from LinkedIn web pages. Additional user information is provided by the user. The data set used is on a per-member basis. This means that only user-specific information provided is used for the analysis and RAG output and no other member information is utilized.

- **Real-Time Data Acquisition**: Through the constant prompts of the user. The LLM is constantly updated with new data that acts as context provided through the user's conversations and constant prompting with the chatbot. If any web page links are updated, the ProxyCurl API is rerun, updating any data that needs to be updated. Similarly, if a new resume is uploaded, the information read and stored within Mongo will also be updated.
- **Unique Data per Conversation Session**: Each conversation session and each user would generate its unique data set, which was immediately processed, analyzed, and injected as context to the LLM. This method ensured the data was directly relevant to each user's specific conditions and objectives, providing tailored insights and results.
- **Focus on Methodology Over Data Accumulation**: The main objective of our research is to capitalize on previous experience in methodology and prompt engineering techniques. The emphasis is on collecting and updating pre-existing data sets through user prompts and any additional provided context through conversation for real-time data processing and analysis.

Thus, the nature of our experimental design and objectives rendered a traditional dataset unnecessary. Our approach instead capitalized on the immediate generation and analysis of data, ensuring that our results were both relevant and timely, reflecting the real-world applicability of our system.

3.3 Results and Analysis

This section details our approach to analyzing data from the internet, the uploaded user resume, the provided user-provided LinkedIn profile, and having the chatbot generate an appropriate response.

3.3.1 Conversation and Co-reference memory

Each conversation has a local memory object that stores previous messages, either AI-generated or user-inputted. Besides the Conversational LLM, this memory object allows our chatbot to have real-time conversations with the user with the proper context.

3.3.2 Web-search and Internet Access

Using the Tavily Search Engine and the provided API. Tavily allows our chatbot to access the internet and gather information from web pages.

3.3.3 ProxyCurl and Web scrapping

We noticed that the information pulled by the API varied between profiles. We noticed that accounts with limited information pulled inconsistent results. Profiles with abundant information seemed to work best with ProxyCurl. Data cleaning was necessary whenever

there was a bad data pull. In the context of job posting, ProxyCurl behaved consistently if the correct link was provided. We noticed that LinkedIn utilized several different views for job postings, and certain views would leak to a 400 code return. However, the utilization of the most specific LinkedIn URL provided very consistent data when scrapped.

4 Legal and Ethical Practices

4.1 Legal Considerations

In the development and deployment of the application, the team must take care to acknowledge the necessary legal considerations. An agreement must be made on ownership and intellectual property rights as well as ownership of the code base. Streamlit, MongoDB, LangChain, OpenAI, and ProxyCurl all have their legal agreements for using their respective technologies. To this end, the legal considerations of this application were handled as such. [5],[6],[7],[8],[10]

Privacy Concerns:

- **Access to Private LinkedIn Profiles** - This application uses ProxyCurl's scraping API to scrape Private LinkedIn Profiles. LinkedIn's privacy policy does not address web scraping as explicitly against their terms of service as this API has not received a cease & desist.
- **Access to User's Resume** - This application requires and stores a user's resume as part of its process. This data is stored in a hashed state so there is no persistent storage however data leaks can occur. The user to use the application must accept usage and storage of their resume for the app to work.

Licensing

- **LGPL** - this application is licensed under the GNU Lesser General Public License - meaning that the developed application is legally open source and that forks of the existing codebase are allowed as long as the code itself is also open source. [3].

4.2 Ethical Considerations

While our application offers significant advancements in gaze analysis technology, it also introduces potential ethical challenges, particularly concerning misuse by malicious actors. These concerns are heightened despite the fact that all processing is done on-device, ensuring data privacy. Key ethical considerations include:

- **Potential for Misuse**: While the GenAI User Profile Generator offers valuable assistance in tailoring resumes to job listings, there are potential avenues for misuse. Individuals could potentially exploit AI capabilities to fabricate or exaggerate qualifications, leading to dishonest representations on their resumes. Also, malicious users might attempt to manipulate the system to prioritize certain keywords or skills, distorting the accuracy of the resume-match process and undermining the integrity of job applications. Additionally, if the AI algorithms are not regularly updated or monitored for bias, there's a risk of perpetuating existing inequalities or favoritism within the hiring process.

- **Consent and Awareness**: With the use of cloud-based computing and Generative AI models, clear consent mechanisms must be in place, ensuring users are aware of and agree to the data processing and storage that occurs.
- **Data Security and Encryption**: Due to the cloud nature of this app, ensuring the highest standards of data security is essential. This includes robust encryption methods to protect the data from unauthorized access, especially if any data is temporarily stored or used for processing purposes.
- **Limiting Functionality for Privacy Protection**: To mitigate the risk of misuse, it might be necessary to limit certain functionalities of the application. This could include restrictions on how long resumes and user data are stored, even on-device, or disabling certain features that could be more prone to abuse.

In conclusion, although our application's cloud processing enhances privacy and security measures, it is crucial to preemptively tackle any ethical concerns, particularly regarding the possibility of exploitation by nefarious individuals. By upholding ethical standards and conscientiously deploying technology, we strive to guarantee that the application is utilized in a manner that upholds both personal privacy and societal ethics.

5 Effort Sharing

5.1 Overview

The Gen AI, generative user profile project's success is a testament to the collaborative and balanced efforts of our team, comprising Alexander Crowe, Manfred Fong, Bryan Gomez, Elijuwon Mitchell, and Samuel Ginzburg. This section details the equitable distribution of tasks and the contributions of each team member.

5.2 Individual and Joint contributions

- **Joint Efforts**: We worked together in developing the idea and supporting material, drafting the first repository, designing the central structures, and finally, strategically designing the system. Team meetings became the communication channels that enabled us to share ideas, track progress, incorporate individual contribution styles, and adhere to the coding standards, readability, and documentation. Discord provided the main means of team communication and was a powerful instrument to foster teamwork.

- **Alexander Crowe (20%)**: Designed and implemented the file upload and reading feature. Focused on providing the LLM with the proper data based on the provided context gathered from the user uploaded resume. Acted as the group leader/project manager. Facilitated different tasks among team members and ensured that each member shared the same workload. In addition, he also managed the code base, ensuring that our git commits, git pushes, and branches remain clean and readable, allowing for proper teamwork and refactoring when necessary.

- **Samuel Ginzburg (20%)**: Researched the appropriate tools, libraries, and frameworks for project implementation. Implemented the resume comparison feature to assess the similarity between the skills and experiences in the resume and predefined reference data. Ensured the generated resume, when prompted, provided a desirable response.

- **Elijuwon Mitchell (20%)**: Implemented the initial versions of the front end. Created the login and registration pages, and the functionality of such pages. He was also in charge of data management and encryption of the backend. He personally managed the MongoDB database ensuring proper back-end functionality. Worked alongside Bryan in the chatbot's web scraping capabilities.

- **Bryan Gomez (20%)**: Concentrated on the chatbot's web scraping capabilities, ensuring that the scraped data from various web pages is correct and reliable. Worked alongside Elijuwon for tool integrations as it played a major part in the application. Implemented the ProxyCurl API web scraper tool allowing the LLM to have proper context to generate an appropriate response.

- **Manfred Fong (20%)**: Implemented the final version of the user interface. In charge of creating the primary chatbot functionality and conversational abilities. Implemented

local memory storage such that the chatbot has co-reference memory, allowing the LLM to use previous user inputs as context or references. Implemented the primary web search features through the Tavily search engine, allowing the Chatbot to access the internet and gather data from the web. In charge of message security. Ensured that each conversation, whether it is user-inputted prompts or AI-generated output, was properly encrypted and secure from malicious attacks, as the chatbot has access to a user's personal information.

5.3 Conclusion of effort

While adhering to the ideals of fairness and collective input, we ensured that our process remained above board and that all participants could rest assured that their contributions were highly regarded.

6 Conclusion and Future Work

6.1 General Statements

Developing the Generative AI chatbot has been an engaging and enlightening journey, revealing intricate facets of collaborative software development. This project presented a diverse array of challenges and learning experiences, fostering the enhancement of our team's technical prowess, interpersonal dynamics, communication skills, and management proficiency.

6.2 Challenges

One significant challenge we faced revolved around organizing our team effectively, particularly amidst diverse responsibilities and pressing deadlines. This underscored the importance of robust communication, especially when navigating differing viewpoints among team members. Conquering these hurdles not only strengthened our team's cohesion but also honed our abilities in problem-solving and negotiation, ultimately improving our management of responsibilities.

6.3 Future Prospects

The successful launch of our resume optimization app represents a major milestone, reflecting our team's unwavering commitment and diligence. Rather than considering this the end of development, we view it as a pivotal moment. Our focus remains on continuous improvement, with plans to enrich the app's capabilities and broaden its capabilities and practicality for everyday use.

6.4 Reflection

Reflecting on the development journey, it's evident that the experience has been immensely impactful. The process has produced substantial growth and resilience within our team, bolstering our capacity to navigate the dynamic landscape of software development. This project has served as a catalyst for the professional advancement of every team member, imparting invaluable skills and perspectives that are poised to shape our trajectories in the development of future products in the software development industry.

7 Bibliography

References

[1] Harrison Chase. Langchain. In *Langchain Getting started*, 2024. https://www.langchain.com/.

[2] Prompt Engineering. Prompt engineering. *Prompt Engineering Documentation*, 2024. https://www.promptingguide.ai/.

[3] GitHub, Inc. Choose an Open Source License, 2023. https://choosealicense.com/ (Accessed on May 6th, 2023).

[4] Stephen Goh. Proxycurl llc. In *ProxyCurl getting started*, 2024. https://nubela.co/proxycurl/.

[5] MongoDB Inc. MongoDB Terms of Use. In *MongoDB Development*, 2024. https://www.mongodb.com/legal/terms-of-use (Accessed on May 6th, 2024).

[6] OpenAI Inc. OpenAI Terms of Use. In *OpenAI Policies*, Jan 31, 2024. https://openai.com/policies/terms-of-use (Accessed on May 6th, 2024).

[7] Streamlit Inc. Streamlit Terms of Use. In *Streamlit Development*, 2023. https://streamlit.io/terms-of-use (Accessed on May 5th, 2024).

[8] LangChain. LangChain Terms of Use. In *LangChain Development*, 2023. https://www.langchain.com/terms-of-service (Accessed on May 5th, 2024).

[9] OpenAI. Openai api. In *OpenAI Inc.*, 2024. https://openai.com/index/openai-api.

[10] ProxyCurl. ProxyCurl Terms of Service. In *Nubela Inc.*, 2024. https://nubela.co/proxycurl/terms-of-use (Accessed on May 6th, 2024).

[11] Rotem Weiss. Tavily api. In *Tavily OPEN SRC.*, 2024. https://tavily.com/.