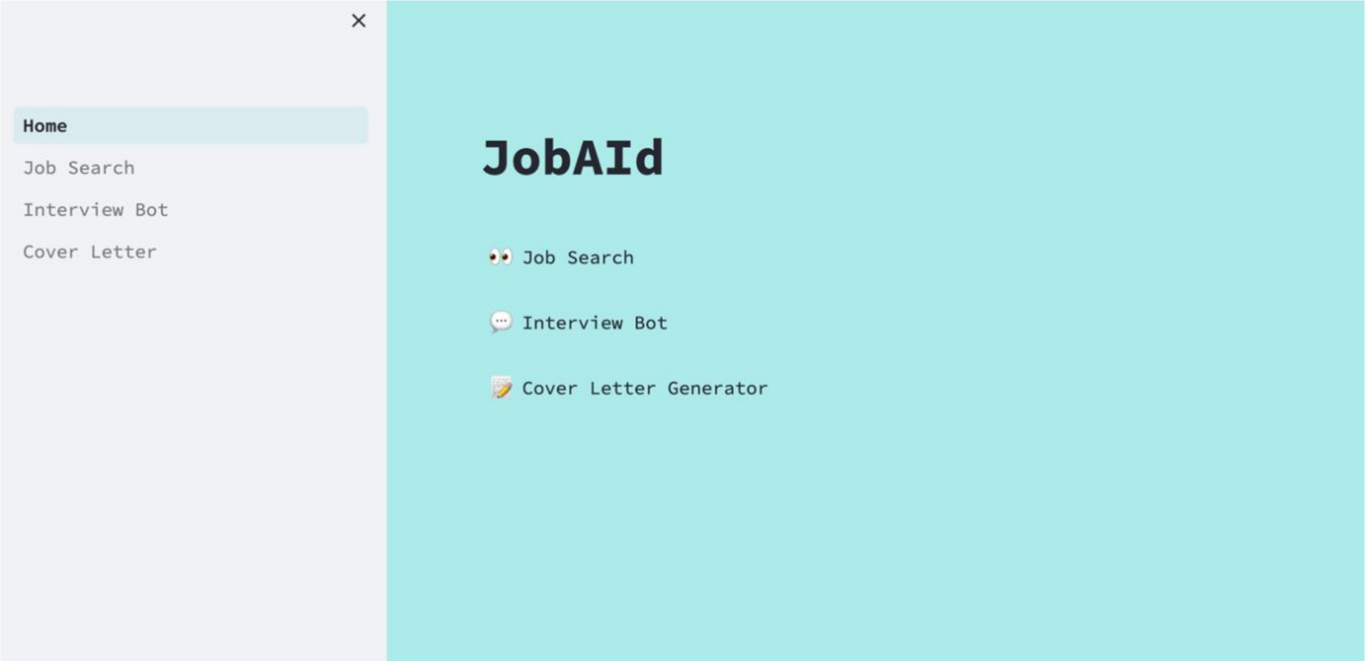
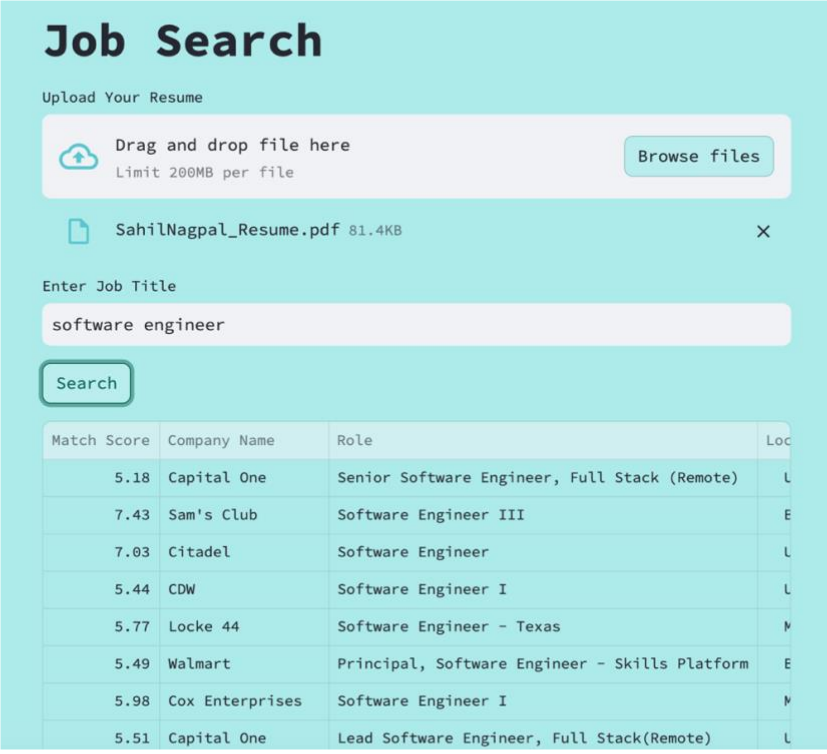
Generative AI
Final Project
JobAId: A Job Search Portal
(http://jobaid.servehttp.com:8501/)



Features of the Application

1. **Job Search**
   Supports job search based on the resume of the applicant and job keyword provided by the user.

   Steps:
   1. Use the Spacy based ResumeParser to extract attributes of the uploaded resume. Write relevant attributes such as experience, skills (not including personal details) to a text file.
   2. Load the text file using Langchain TextLoader and get embeddings of the file using FAISS.
   3. Use the SerpAPI to get job postings from Google jobs using the query provided by user.
   4. Extract relevant attributes from the job listing such as the job requirements and description.
   5. Use the embeddings to calculate the similarity score (L2 distance) between the resume and the job listing.
   6. Convert the similarity score into a range of 1 to 10.
   7. Create links for cover letter generation and the Google jobs listing.
   8. Display the resulting dataframe using Streamlit.



Job Search

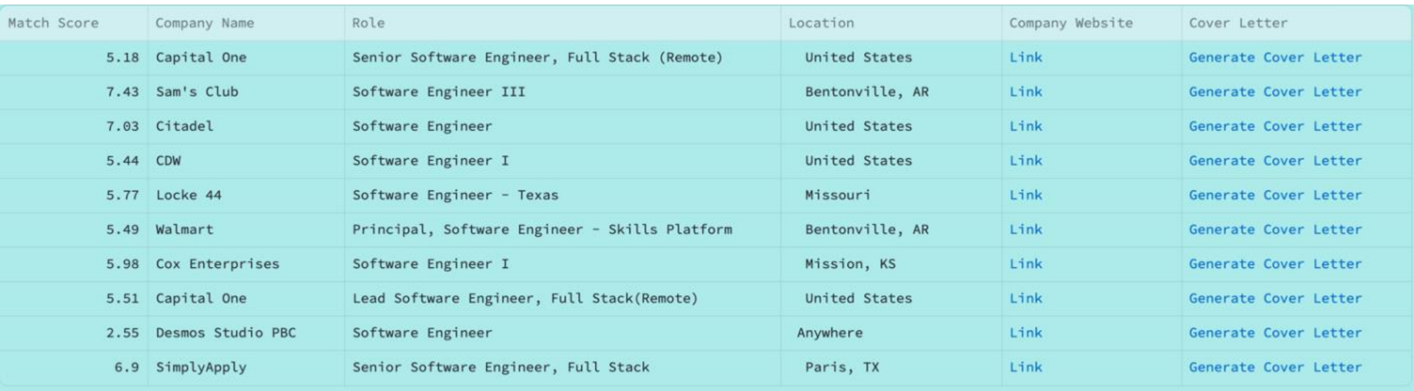
Upload Your Resume

SahilNagpal_Resume.pdf 81.4KB

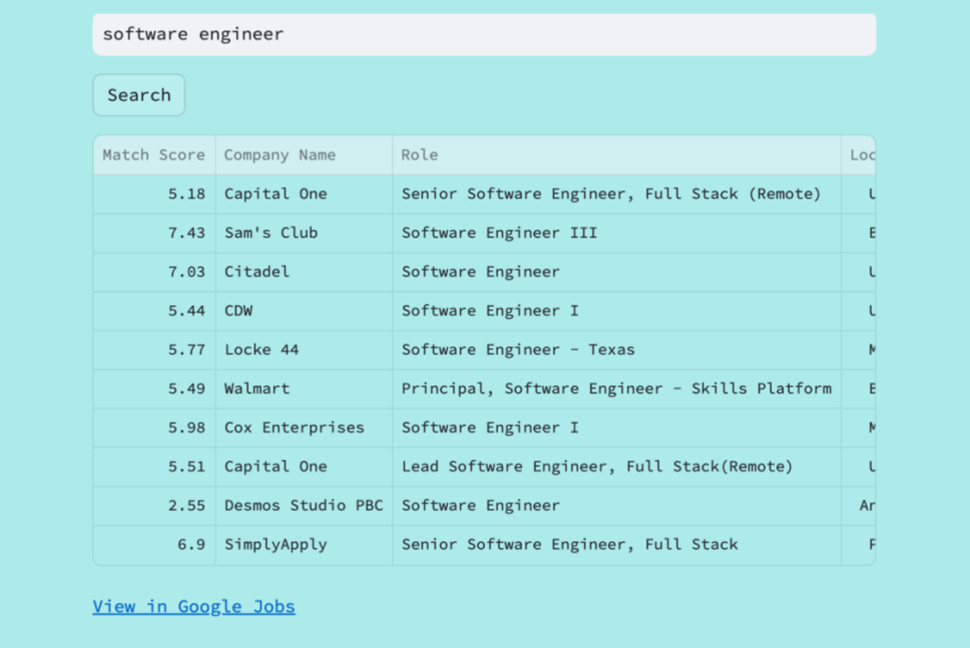
Enter Job Title

software engineer

Match Score	Company Name	Role	Loc
5.18	Capital One	Senior Software Engineer, Full Stack (Remote)	U
7.43	Sam's Club	Software Engineer III	B
7.03	Citadel	Software Engineer	U
5.44	CDW	Software Engineer I	U
5.77	Locke 44	Software Engineer - Texas	M
5.49	Walmart	Principal, Software Engineer - Skills Platform	B
5.98	Cox Enterprises	Software Engineer I	M
5.51	Capital One	Lead Software Engineer, Full Stack(Remote)	U



Match Score	Company Name	Role	Location	Company Website	Cover Letter
5.18	Capital One	Senior Software Engineer, Full Stack (Remote)	United States	Link	Generate Cover Letter
7.43	Sam's Club	Software Engineer III	Bentonville, AR	Link	Generate Cover Letter
7.03	Citadel	Software Engineer	United States	Link	Generate Cover Letter
5.44	CDW	Software Engineer I	United States	Link	Generate Cover Letter
5.77	Locke 44	Software Engineer - Texas	Missouri	Link	Generate Cover Letter
5.49	Walmart	Principal, Software Engineer - Skills Platform	Bentonville, AR	Link	Generate Cover Letter
5.98	Cox Enterprises	Software Engineer I	Mission, KS	Link	Generate Cover Letter
5.51	Capital One	Lead Software Engineer, Full Stack(Remote)	United States	Link	Generate Cover Letter
2.55	Desmos Studio PBC	Software Engineer	Anywhere	Link	Generate Cover Letter
6.9	SimplyApply	Senior Software Engineer, Full Stack	Paris, TX	Link	Generate Cover Letter



software engineer

Match Score	Company Name	Role	Loc
5.18	Capital One	Senior Software Engineer, Full Stack (Remote)	U
7.43	Sam's Club	Software Engineer III	B
7.03	Citadel	Software Engineer	U
5.44	CDW	Software Engineer I	U
5.77	Locke 44	Software Engineer - Texas	M
5.49	Walmart	Principal, Software Engineer - Skills Platform	B
5.98	Cox Enterprises	Software Engineer I	M
5.51	Capital One	Lead Software Engineer, Full Stack(Remote)	U
2.55	Desmos Studio PBC	Software Engineer	Ar
6.9	SimplyApply	Senior Software Engineer, Full Stack	P

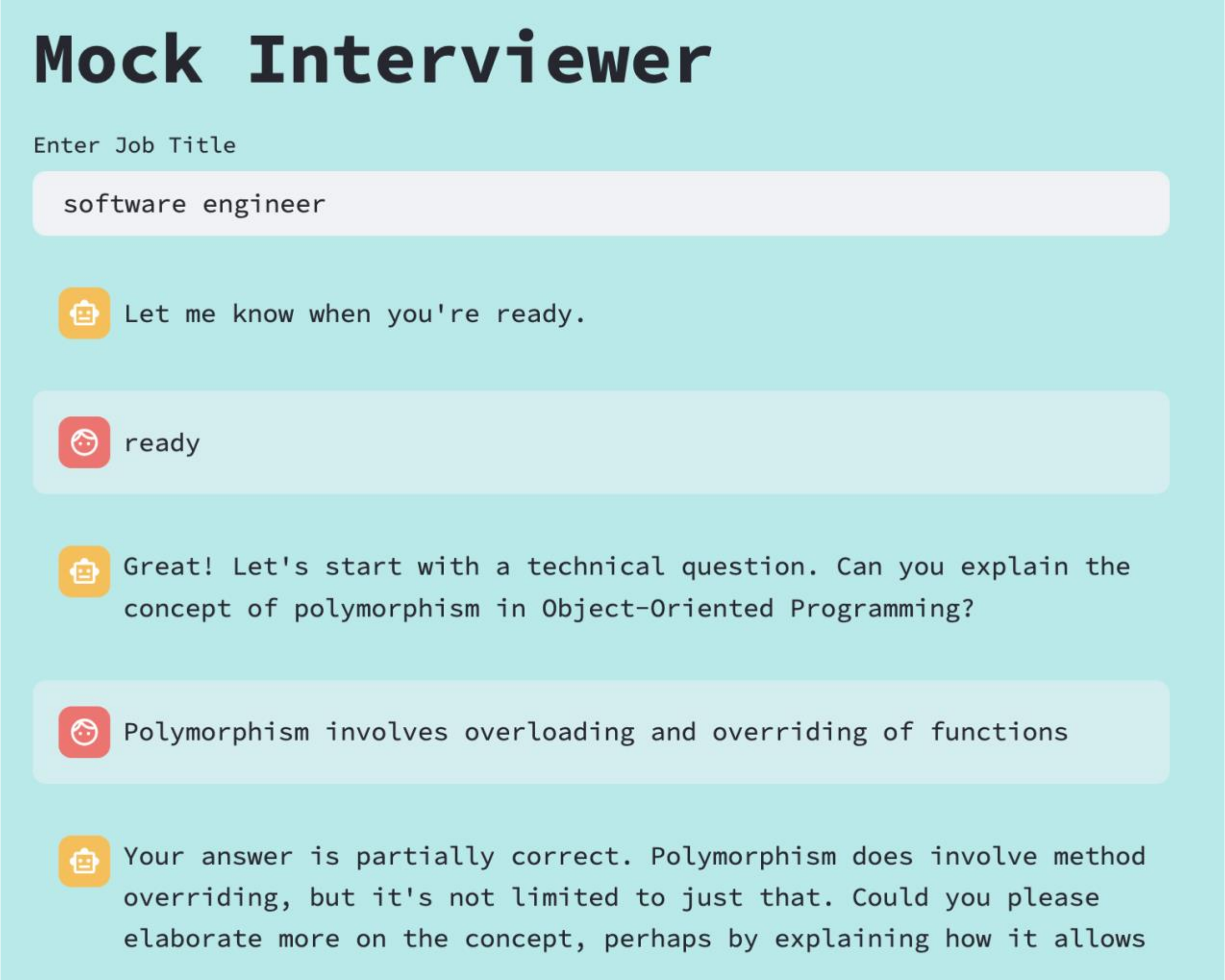
View in Google Jobs

2. **Interviewer Bot**

   Poses as an interviewer for the provided job role and asks relevant questions while guiding the applicant to appropriate answers.

   Steps:

   1. Create system message using COSTAR framework using the job title entered by the user.
   2. Create a Langchain ConversationChain comprising of ConversationBuffer Memory.
   3. If the last message in the message history was generated by the user, use the conversation chain to predict the response based on the history and job title.
   4. Display the response to the user.
   5. If the user changes the job title, clear the message history, and create a new system message based on the job title. Thereafter, create a new conversation chain and predict responses based on the new job title.



Mock Interviewer

Enter Job Title

software engineer

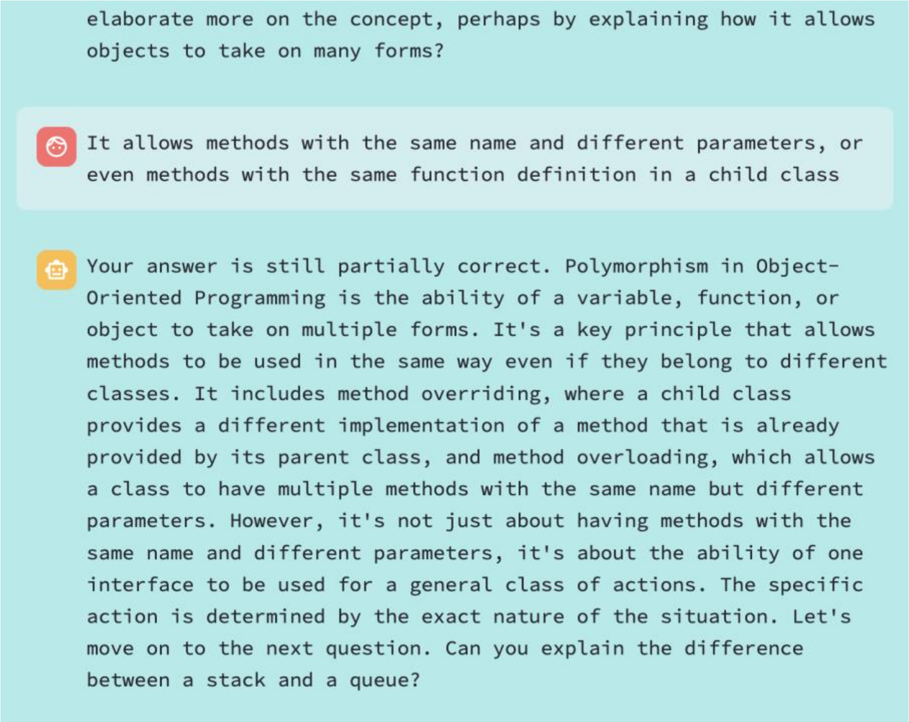
Let me know when you're ready.

ready

Great! Let's start with a technical question. Can you explain the concept of polymorphism in Object-Oriented Programming?

Polymorphism involves overloading and overriding of functions

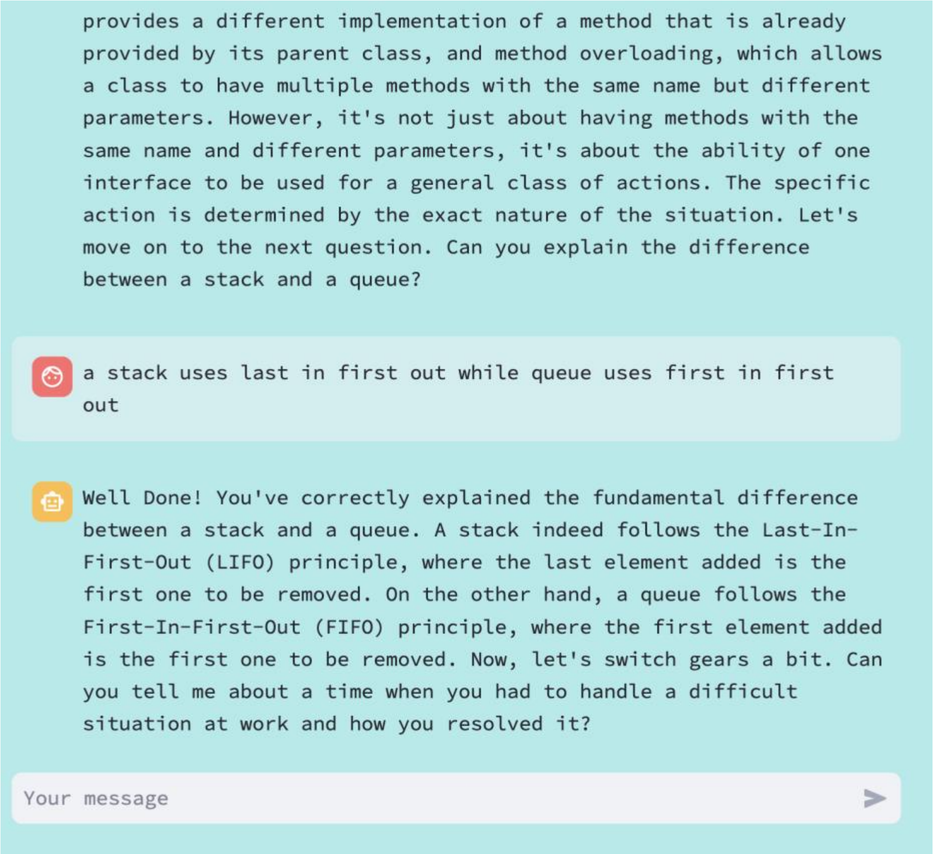
Your answer is partially correct. Polymorphism does involve method overriding, but it's not limited to just that. Could you please elaborate more on the concept, perhaps by explaining how it allows



elaborate more on the concept, perhaps by explaining how it allows objects to take on many forms?

It allows methods with the same name and different parameters, or even methods with the same function definition in a child class

Your answer is still partially correct. Polymorphism in Object-Oriented Programming is the ability of a variable, function, or object to take on multiple forms. It's a key principle that allows methods to be used in the same way even if they belong to different classes. It includes method overriding, where a child class provides a different implementation of a method that is already provided by its parent class, and method overloading, which allows a class to have multiple methods with the same name but different parameters. However, it's not just about having methods with the same name and different parameters, it's about the ability of one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. Let's move on to the next question. Can you explain the difference between a stack and a queue?



provides a different implementation of a method that is already provided by its parent class, and method overloading, which allows a class to have multiple methods with the same name but different parameters. However, it's not just about having methods with the same name and different parameters, it's about the ability of one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. Let's move on to the next question. Can you explain the difference between a stack and a queue?

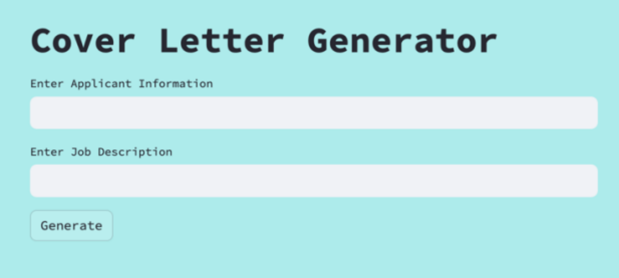
a stack uses last in first out while queue uses first in first out

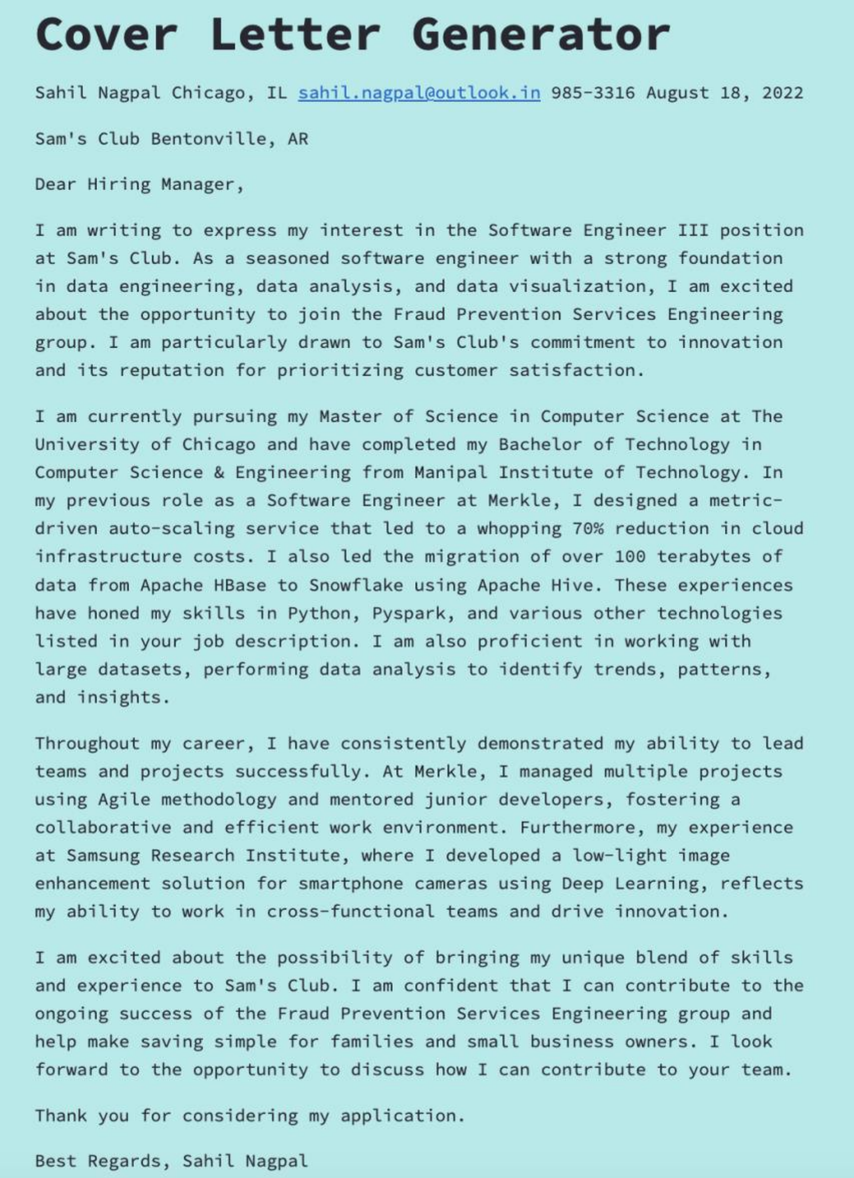
Well Done! You've correctly explained the fundamental difference between a stack and a queue. A stack indeed follows the Last-In-First-Out (LIFO) principle, where the last element added is the first one to be removed. On the other hand, a queue follows the First-In-First-Out (FIFO) principle, where the first element added is the first one to be removed. Now, let's switch gears a bit. Can you tell me about a time when you had to handle a difficult situation at work and how you resolved it?

3. **Cover Letter Generator**
   Generates a cover letter based on:
   - custom applicant information and job description.
   - resume and listings on the job search page.

Steps:

1. Create system message with guidelines for generating a cover letter.
2. If redirected from the job search page, extract the applicant information, job information and resume path from the query parameters.
3. Load the resume using Langchain PyPDFLoader and extract the text.
4. Combine the applicant information and the job description for the prompt.
5. Pass the system message and the necessary information to the custom GPT chain.
6. Print the response using Streamlit.
7. Copy the response to Clipboard using Pyperclip (only works when running on local machine)





Cover Letter Generator

Sahil Nagpal Chicago, IL sahil.nagpal@outlook.in 985-3316 August 18, 2022

Sam's Club Bentonville, AR

Dear Hiring Manager,

I am writing to express my interest in the Software Engineer III position at Sam's Club. As a seasoned software engineer with a strong foundation in data engineering, data analysis, and data visualization, I am excited about the opportunity to join the Fraud Prevention Services Engineering group. I am particularly drawn to Sam's Club's commitment to innovation and its reputation for prioritizing customer satisfaction.

I am currently pursuing my Master of Science in Computer Science at The University of Chicago and have completed my Bachelor of Technology in Computer Science & Engineering from Manipal Institute of Technology. In my previous role as a Software Engineer at Merkle, I designed a metric-driven auto-scaling service that led to a whopping 70% reduction in cloud infrastructure costs. I also led the migration of over 100 terabytes of data from Apache HBase to Snowflake using Apache Hive. These experiences have honed my skills in Python, Pyspark, and various other technologies listed in your job description. I am also proficient in working with large datasets, performing data analysis to identify trends, patterns, and insights.

Throughout my career, I have consistently demonstrated my ability to lead teams and projects successfully. At Merkle, I managed multiple projects using Agile methodology and mentored junior developers, fostering a collaborative and efficient work environment. Furthermore, my experience at Samsung Research Institute, where I developed a low-light image enhancement solution for smartphone cameras using Deep Learning, reflects my ability to work in cross-functional teams and drive innovation.

I am excited about the possibility of bringing my unique blend of skills and experience to Sam's Club. I am confident that I can contribute to the ongoing success of the Fraud Prevention Services Engineering group and help make saving simple for families and small business owners. I look forward to the opportunity to discuss how I can contribute to your team.

Thank you for considering my application.

Best Regards, Sahil Nagpal

Concepts Used

1. **Retrieval Augmented Generation (RAG)**
   Leveraged Facebook's FAISS library to generate embeddings of the resume and calculate the L2 distance (similarity metric) between the applicant's profile and the job listing.

2. **Langchain**
   - Conversation chain for implementing the mock interviewer. This feature also included conversational memory to store the current conversation.
   - ChatPromptTemplate for cover letter generation using GPT.
   - Document loaders (TextLoader and PyPDFLoader) for importing the resume.

3. **Streamlit**
   The entire frontend of the application. Used st.cache_resource on the job search page while generating document embeddings and calculating similarity score.

4. **Prompt Engineering**
   Used COSTAR framework for implementing the mock interviewer and followed best practices, including asking for structured output, using delimiters, checking if conditions were satisfied, and specifying steps for the process, for the cover letter system message.
5. **Temperature**
   Set model temperature to 0 for the interviewer bot to ensure factual correctness.

Experimentation

1. **Calculation of similarity between resume and job application**
   - Attempted to load the resume with PyPDFLoader. However, this contained too many personal details and extraneous text, which led to inaccurate similarity scores. To resolve this issue by extracting relevant information, ResumeParser (pyresparser) powered by the Spacy NLP library was used. The extracted attributes were written to a text file which was subsequently read using a TextLoader.
   - Tried to incorporate the Google Jobs tool present on the Langchain hub. The resultant string contained irrelevant text such as job benefits, company overview etc. To circumvent this, the Serpapi API for Google Search was used (the langchain tool is also based on this). It returns a dictionary of job attributes, which enabled the extraction of relevant job details to improve the accuracy of the similarity score.
   - Tested the feature with various resumes and job titles to estimate the range of similarity score values. The general range was found to be between 0.2 and 0.6. Since lower values of these scores signify higher similarity, these scores were converting to a 0-10 range where 0 is for any score >= 0.6 and 10 is for any score <= 0.2.

2. **Support for conversational memory in the interviewer bot**
   - Tried using agent with ChatPromptTemplate (as discussed in class). Did not fit the use case due to the absence of additional tools.
   - Had to store the conversation chain, template, and role in the Streamlit session state to start a new chat when the job role value is changed.

3. **Links in the Job Search table**
    - Tried implementing using st.table, st .write, and st.columns. None of these support clickable links. Therefore, st.dataframe was the ideal choice.
    - Passed parameters required for cover letter generation as query string parameters.

4. **Interview Bot**
    - Tried with non-COSTAR template for system message. Received poor responses compared to COSTAR.
    - Initially attempted using chain stream for response generation. However, the stream produced one character at a time which made the text generation extremely slow. Decided to use invoke.

5. **Streamlit Miscellaneous**
    - Changed the theme of the application using config.toml.
    - Added pages for the application.
    - Copy/paste for cover letter – working on localhost using Pyperclip
    - Added modal for notifying about job role change in interview bot – cannot change the size and inconvenient in general.

Ideas for Improvement

1. Incorporate more sources for job listings. (LinkedIn, Indeed.com etc.)
2. Include pagination on the job search page for generating more than 10 jobs.
3. Add direct links to jobs by using Job Listings API.
4. Suggest resume changes based on the job listing. Can be done by dividing the document into chunks and suggesting changes for chunks with the highest L2 distance.
5. Feature to apply filters such as level of experience, location, job posting date.
6. Application autofill based on the resume.