Update 4 – Progress Update Report

CSE 499A: Senior Design I | Section: 15 | Group: 03
Md. Kamrul Islam (2211745642) | Farhan Ishraque (2212002042) | Atikul Islam Nahid (2211978042) | Iram Shehzad (2131614642)

Week Update Summary

This week, three primary AI features were completed on the JobCore project. Md. Kamrul Islam successfully completed the primary backend job search agent (job_search_agent.py) utilizing LangGraph and LangChain to integrate five different job search tools using OpenAI GPT and debug a key web-scraping dependency, ddgs. Simultaneously, Farhan Ishraque finalized the resume checker by connecting the GPT-3.5-powered Flask backend to a new React front-end and contributed to the project report. To complete the feature set, Iram Shehzad implemented and secured the two main API endpoints (/get-question and /get-feedback) for the AI mock interview feature, integrating Google's gemini-1.0-pro model and resolving multiple debugging issues in real-world scenarios, including API errors and port conflicts that occurred.

Contribution

Md. Kamrul Islam:

I built and completed the main AI agent for the JobCore backend, which I implemented in job_search_agent.py. I utilized LangGraph for state management in the agent and LangChain to manage the five job search tools: search_jobs_indeed, search_jobs_linkedin, search_jobs_glassdoor, search_remote_jobs, and get_job_market_insights. I accomplished this using the langchain-openai library to access the OpenAI GPT model for natural language understanding and ddgs for scraping web results. To keep the API key secure, I learned to store my OPENAI_API_KEY in a .env file, which was protected through the use of a .gitignore file. I also debugged multiple real-world errors, including a notable import error that occurred when I switched from the deprecated duckduckgo-search package to the latest DDG and subsequently refactored all tools to handle API changes, enabling the agent to successfully collect real job listings.

Farhan Ishraque:

This week I tried to solve the api problem and I fixed it. So basically my backend is complete and then I tried to connect the backend with my front-end. I used the GPT 3.5 turbo model of Open Ai for the resume checker part. Then I designed the front-end for the resume checker with react js and flask according to the backend.To keep the API key secure, I learned to store my OPENAI_API_KEY in a .env file, which was protected through the use of a .gitignore file also. I used a flask to connect the backend and frontend smoothly without any error. In this process I faced a lot of problems but I tried my best and overcame them. I helped on the report writing and all the outputs are pushed on github as screenshots.

Iram Shehzad:

For this week I have solved and created two main API endpoints for the AI mock interview: POST /get-question, which uses a dynamic prompt to ask the Google AI (gemini-1.0-pro) for a new, non-repeated question, and POST /get-feedback, which sends the user's answer to the AI for structured coaching. To do this, I used the @google/generative-ai library to talk to the gemini-1.0-pro model. For security, I learned to keep my secret API key secure in a .env file and debugged several real-world errors, including updating the AI model name when it was "Not Found," fixing the EADDRINUSE port conflict, and using the correct PowerShell Invoke-WebRequest syntax to test my API successfully from the terminal.

Atikul Islam Nahid:

I focused on the back-end for the AI Mock Interview, Here's what I did. I started by investigating AI models from related journals. I then worked on building the routes with JavaScript with Node.js and Express to get the questions and evaluate the answers on the back-end part of the app. I have encountered a blocker with respect to the front-end integration. I have identified that the problem is related to the API connection and am currently working to resolve this.

Completed Task:

- Resume checker updated and completed.
- We can search for jobs, and we are in an initial state in this process.

Work in Progress

- Resume Checker backend and frontend integration with GPT3.5 turbo.
- Mock Interview questions, answers, and scoring based on the answers.
- Job Search from many job posting platforms.
- Testing various AI models.

Next Target

- Complete the chatbot with the concept of memory in LLM and context.
- Job Search from many job posting platforms.
- Resume Checker backend and frontend integration.
- Complete mock interview questions, answers, and a scoring system.
- Test will be complete of the ai models and finalize.

Challenges

- Unavailability of the LLM model’s API.
- Unavailability of enough research papers in our domain.
- Costly existing tools or websites in the current market.

- Api key errors
- The AI models don't work.