AI Web Lab Report

Revon Moore

December 15, 2025

Task 1

This task involved setting up the Node.js development environment and installing required dependencies, including the OpenAI SDK and dotenv for environment variable management. The project structure was verified using terminal commands, and initial scripts were executed to confirm proper configuration.

Task 2

I tested multiple prompts and attempted different OpenAI models. While quota limits prevented full responses, terminal output confirms model switching and prompt execution. Differences include response structure and error handling behavior.

Task 3

This task demonstrates both non-streaming and streaming OpenAI API calls. In non-streaming mode, the response is returned as a complete message after processing. In streaming mode, tokens are returned incrementally, allowing real-time output. Although quota limits prevented full completion, terminal logs confirm correct implementation and execution flow.

Task 4

Task 4 focused on implementing an OpenAI agent using the Agents SDK. A recipe-generation agent was created with defined instructions and executed through a TypeScript file. The agent configuration and execution were confirmed through terminal output, verifying correct setup despite quota-related execution limits.

Task 5

This task implemented a Retrieval-Augmented Generation (RAG) pipeline using a PDF resume as input. The application performs PDF parsing, text chunking, embedding generation, and vector storage before producing contextual responses. Execution was confirmed via terminal prompts requesting a PDF file path and initializing the RAG workflow.

Task 6

Task 6 involved running a web-based RAG resume analyzer. A local server was started, and a browser interface allowed PDF upload for analysis. The application generated two perspectives: a hiring manager review and a critical assessment. Successful server startup and UI rendering confirmed correct functionality.