AI: An Introduction
Midterm Project

AI Chatbot with Function Calling features

Project Overview:

This project aims to develop a chatbot using the OpenAI API implements function calling to connect the chatbot to external data sources. Students will be provided a list of libraries and APIs that can be easily attached to the chatbot to expand its ability.

Libraries and APIs:
Datetime - Python library that can return the current time and date.
Psutil - Python library that can return the battery status of the device the script is running on.
NewsAPI - News aggregation API that can return the current top headlines.
OpenWeatherMap API - Weather API that returns current weather based on location.
Wolfram Alpha Short Answers API - Answer engine that returns factual information.

Project Submission:

Your project will be due on 7/12/2024. Students are given a week to complete this project.

There will be a Teams assignment created that you will use to submit your project. You can use one of the following two options to submit your project:

1. You can upload the required files directly to Teams and then submit your project.
2. You can first upload the required files to a public GitHub repository, then link to your GitHub repository as your Teams assignment submission.

The required items can be seen below.

AI: An Introduction
Midterm Project

Project Objectives:

1. Create a conversational chatbot using the OpenAI Chat Completions API.
2. Create at least 3 functions that retrieve data from the list of libraries and APIs provided.
3. Create the tool list that describes these functions and the arguments necessary for the function in natural language.
4. Append the data returned from the called function to the message list with the role “tool” and make another API call for the final response.
5. Document the entire process for reproducibility and future reference.

Project Tasks:

1. **Conversational Chatbot**
    a. Make a loop where the prompt and response pairs are appended to the message list.
2. **Data Retrieval Functions**
    a. Set up accounts and API keys for the chosen data source APIs.
    b. Use the python libraries and APIs within functions to return the desired data.
3. **Function Description**
    a. Detail the intended use of the functions in natural language so the model can understand when to use the functions properly.
    b. Determine what arguments (if any) are necessary for each function.
4. **Function Calling Implementation**
    a. Use the response object to call the function(s) chosen by the model.
    b. Append the function result to the message list for the model to generate the natural language response.
5. **Documentation**
    a. Create detailed documentation explaining the generation steps, assumptions, and decisions.
    b. Include comments in the code to make it more understandable and maintainable.
6. **Testing and Validation**

a. Test the function calling with multiple prompts that invoke different (or multiple) functions to be called.
b. Validate the results against the expected outcomes.
c. BONUS: Automate the testing using a dictionary of prompts and expected function calls.

Items for Submission:

1. A documented generation process, including code comments and explanations.
2. A script for the conversational function calling chatbot.
3. Any additional documentation or instructions for using the chatbot.

Rubric:

If you do not score a 14 or higher for your project based on the rubric below you will have the opportunity to make changes to your project and resubmit it.

	0 Points - Missing	1 Point - Does Not Meet Expectations	2 Points - Meets Expectations	3 Points - Exceeds Expectations	Points Awarded
Conversational Chatbot	This chatbot only responds to one prompt and doesn't append the response to the message list.	This chatbot doesn't loop which restricts users from prompting multiple times when running the script.	This chatbot appends user, assistant, and tool messages to the message list in a loop for users to prompt the chatbot multiple times within a run.	This chatbot manages the message list properly within a loop to allow for multiple uses when running the script without hitting the context limit.	
Data Retrieval Functions	Functions didn't use APIs or python libraries to retrieve the desired data.	Only some functions performed the proper data retrieval.	APIs and python libraries were used properly to access data and return it to the model.	APIs and python libraries were used in a multitude of ways to access different sets of	

				data to return to the model.	
Function Description	No function descriptions were provided.	Some function descriptions are vague and unclear about the actions the function take and the arguments necessary.	Function descriptions are easily understood and detailed with proper argument details.	Function descriptions are easily understood and detailed with proper argument details, required arguments, and any necessary units.	
Function Calling Implementation	The function(s) listed in the response object weren't called in the script properly.	The function results are displayed to the user but not appended to the message list for the model's informed response.	The function results are appended to the message list properly.	Parallel function calling is supported and the function results are appended to the message list properly.	
Documentation	No documentation	Only some of the needed	Code was written in the	All parts of the code were	

	was added.	documentation was added.	proper order of events and is well documented.	written in the proper order of events, and a general process was written for a non-technical user.	
Testing and Validation	No testing or validations were performed.	Only some testing and validation were done, but not enough to catch all errors or issues.	Adequate testing was performed, and the output was validated against expected responses.	A dictionary of prompts and expected responses were created to automated testing and validation.	
				Score	