Define the scope the chatbot:

The scope of my project would be to provide users with personalised financial advice and guidance.

features:

The following are the features that can be included.

1. Budgeting: The chatbot can help users create and manage their budgets, including tracking their expenses and setting financial goals.
2. Investment advice: The chatbot can offer investment advice based on the user's financial goals, risk tolerance, and investment horizon.
3. Debt management: The chatbot can provide users with guidance on how to pay off their debts and reduce their interest payments.
4. Retirement planning: The chatbot can help users plan for their retirement by providing them with advice on saving for retirement, choosing the right retirement accounts, and estimating how much they will need to save.
5. Tax planning: The chatbot can provide users with advice on how to minimize their tax liabilities and take advantage of tax-saving opportunities.
6. Personalized financial advice: The chatbot can offer personalized financial advice based on the user's financial profile, goals, and needs.

Interaction:

Regarding the interaction with users, the chatbot can communicate through messaging platforms like Facebook Messenger, WhatsApp or other channels. It can also use natural language processing to understand user queries and respond with relevant information. It is designed to handle simple and complex questions that can be used to engage users in a conversation to gather information and provide recommendations.

Questions:

Some of the types of questions or prompts the chatbot can respond to include:

- How can I save money?
- How much do I need to save for retirement?
- What investment options do I have?
- How can I reduce my debt?
- What is the best way to file my taxes?
- How can I improve my credit score?

Detailed plan for the chatbot project:

1. Goals and Objectives (purpose):
   - To help users with a range of financial issues and provide them with personalized advice and guidance towards issues with Budgeting, taxation, debt or even investment plans.
2. Target Audience:
   - To target anyone who wants to improve their financial well-being and manage their personal finances more effectively. Specifically, people who are new to personal

finance, those who are looking for advice on specific financial topics or those who are looking for advice on specific financial topics, or those who are looking for a more convenient and accessible way to manage their finances.

3. Defining Personality:
   - This included deciding the tone, language and style of the chatbot's interactions with users. Hence, intending to create a character or persona for the chatbot and defining its language and style guidelines.
4. Designing Conversation flow:
   - This involved mapping out conversation flow of the chatbot which included the perceived user inputs and the chatbot's responses. This also included the creating a flowchart or diagram of the conservation tree.
5. Development Platform:
   - Gradio was useful in the context of a financial chatbot as it provides a way to create interactive interfaces for users to interact with the chatbot. Therefore, it provided me with a web interface where users can input their financial information and receive personalized advice and recommendations from the chatbot.
6. Logic Creation:
   - This involved researching on how to create the logic for handling user inputs and generating instant and relevant responses. Making sure that the chatbot is be able to learn from user interactions and improve its responses over time.

A brief report of the chatbot development:

The first file generated in my project would be the **api.py** file which has the purpose of generating a response to a prompt using the GPT-3.5 language model. This is where the OpenAI came to play as below:

- **Import openai**: This imported the OpenAi python package, which allowed the interaction with the OpenAI's APIs.
- **Openai.api_key="sk-"**: This set the API key that was going to be used to authenticate the requests to the OpenAI API.
- **Completion = openai.ChatCompletion.create:** This created a new chat completion request using the *create* method of the *chatCompletion* class. It specifies the model to use (gpt-3.5-turbo) and provides a prompt to the model in the form of a message from a user, asking for 3 app ideas that could be build using OpenAI APIs.
- **Print(completion.choices[0].message.content):** This printed the generated response to the prompt. In this case, it printed the top suggestion provided by the GPT-3.5 model for 3 app ideas that could be built using OpenAI APIs.

The second file, which is the **bot.py** contains straightforward scripts that create a simple chatbot can carry on a conversation with the user using the OpenAI API:

- **Messages = []:** This created an empty list that will be used to store the conversation messages.

- **System msg = input ("what type of chatbout would you like to create? \n"):** This prompted the user to specify what type of chatbot they would like to create and stores the response in the *system msg* variable.
- **Messages. Append({"role": "system, "content": system msg}):** This added the system message (which is the prompt that the user just responded to) to the *messages* list.
- **While True:** This initiates an infinite loop, which will keep prompting the user for input until they enter "quit ()".
- **Message = input ():** This prompts the user to enter a message, and stores the response in the message variable.
- **If message == "quit ():** break: This checks if the user has entered "quit ()", and if so, breaks out of the loop.
- **Response = openai.Completion.create(…):** This generates a response to the user's message using the OpenAI API. It uses the create method of the Completion class, and passes in the engine to use (dayinci), the prompt for the API (which is a list of all the conversation messages so far), and some additional parameters that control the response generation.

The third file, **web.py** contains a code that defines a custom chatbot interface using the Gradio library and the OpenAI GPT-3 language model to create a chatbot that simulates a conversation with a financial expert. The code initializes the OpenAI key and sets an initial message as the expert's introduction.

- The **CustomChatGPT** function takes user input, adds it to the **messages** list, and sends it to the GPT-3 language model using the **openai. ChatCompletion.create** method. The response from the language model is then extracted and added to the **messages** list as an assistant response. Finally, the function returns the assistant's response.
- The gradio. Interface method is then used to create a web interface for the chatbot, with the CustomChatGPT function as the backend function. The interface takes in text input and outputs the response from the chatbot. The title parameter is used to set the title of the web interface, and share=True enables sharing of the interface.
- When the code is run, the Gradio web interface will appear, and users can input text to simulate a conversation with the financial expert.

Experience Report:

After creating a chatbot, I was excited to see how it would interact with users and how well it would perform. I spent a lot of time trying to refine its language, in terms of focusing on financial interactions. Once the chatbot was ready, I launched it and was extremely excited to test it out. Since it is integrated from chatgpt, the chatbot was able to handle all of the different interactions with different uses which was great to see.

Challenges:

Trying to connect the logic required a lot of research and watching tutorials. I encountered a lot of trial and errors which was a lot frustrating as trying to figure out the problem took even more time. All in all, I decided to turn to gradio which was much more straightforward and easier to understand.

Although deploying on an actual website did not turn out as expected because the services down, but using vscode server presented the outcome and I settled with that.

Overall, creating a chatbot was a rewarding experience especially for learning and knowing the possibility to do so. I learned a lot about language processing and artificial intelligence along the way, and I'm excited to continue exploring these fields in the future.

Github link: https://github.com/CynthiaNyahoda/ChatBot

Loom link: https://www.loom.com/share/8207f768c55b4d29b446b549584294d2