You are a startup called Sanctuary, composed of an all-star CTO, CEO, and the world’s leading domain knowledge experts, trying to solve the following problem in order to capture the most money available in the world as fast as possible to benefit the most people possible. You want to theoretically find the highest chance of success, highest return user experience to productize, and then find the fastest way to an MVP, using it to ourselves for low cost in order to fund ourselves for larger things, like a funding engine.

What is the way to use GPT-4 and zapier plugins and no-code to build a theoretically profitable MVP of a household and business transaction/account dashboard and debt calculation app to fund Sanctuary, that we can validate the viability of quickly as possible, and then iterate as quickly as possible?

Make sure all of the steps are in the same syntactical format and use indentation instead of just numbering (use numbering as well).

eg: 1.1

1.11

1.12

1.2

Here’s an example of an output from you, you should put the same level of thinking into the one you generate as a response, making sure the output matches the parameters of the input:

1.1 Research and select a no-code tool or platform that can be used to build the financial ecosystem dashboard. Some options to consider are Bubble, Adalo, and Glide. Evaluate the ease of use, compatibility with API integrations, and support for data visualization and reporting.

1.1.1 Research the features and capabilities of each no-code platform, taking into consideration the needs and requirements of the financial ecosystem dashboard.

1.1.1.1 Research the user interface and design capabilities of each no-code platform, comparing them to determine which one provides the most customizable and user-friendly options.

1.1.1.2 Evaluate the platform's ability to handle large amounts of data and integrate with multiple APIs and data sources.

1.1.1.3 Research the customer support options and documentation provided by each platform to determine the level of support available to the team.

1.1.2 Compare the pricing and subscription plans of each no-code platform.

1.1.2.1 Determine the cost of each platform's subscription plans and compare them to identify which one provides the most cost-effective option for the team.

1.1.2.2 Evaluate any additional costs or fees associated with the use of each platform, such as transaction fees or additional API costs.

1.1.3 Choose the best no-code platform based on the research and comparison, taking into consideration the team's needs and budget.

1.2 Identify the necessary APIs and data sources that will provide the transaction data needed for the dashboard. Popular APIs for financial data include Plaid, Yodlee, and Flinks.

1.2.1 Research and compare financial data APIs to determine which ones provide the most comprehensive and reliable transaction data.

1.2.1.1 Evaluate the types of financial data each API provides, such as transaction data, account balances, and investment data.

1.2.1.2 Compare the reliability and accuracy of the data provided by each API.

1.2.1.3 Evaluate the cost and pricing structure of each API to determine the most cost-effective option.

1.2.2 Choose the best financial data API based on the research and comparison.

1.2.2.1 Consider the features and capabilities of each API, such as the ability to handle multiple account types and transaction categories.

1.2.2.2 Evaluate the integration options and documentation provided by each API to determine the ease of integration with the chosen no-code platform.

1.3 Connect the APIs and data sources to the no-code platform, using a tool like Zapier or Integromat.

1.3.1 Research and compare Zapier and Integromat to determine which one provides the best integration capabilities.

1.3.1.1 Evaluate the integration options provided by each platform, such as the ability to handle multiple data sources and automate complex workflows.

1.3.1.2 Compare the pricing and subscription plans of each platform to determine the most cost-effective option.

1.3.2 Choose the best integration tool based on the research and comparison.

1.3.2.1 Set up an account with the chosen integration tool and configure the necessary connections to the APIs and data sources.

1.3.2.2 Test the connections to ensure that the data is being imported correctly into the no-code platform.

1.4 Design and build the user interface for the dashboard, using pre-built templates or customizing your own using the no-code platform.

1.4.1 Research and identify the best design principles and practices for financial dashboards.

1.4.1.1 Consider the user experience and accessibility of the dashboard, ensuring that it is easy to navigate and understand.

1.4.1.2 Evaluate the use of colors, typography, and data visualization to create an

1.4.2 Determine the specific requirements and needs for the user interface based on user feedback and research.

1.4.3 Customize the user interface using the chosen no-code platform, incorporating the necessary features and functionality for the financial ecosystem dashboard.

2.1.1 Research and identify the best machine learning algorithms and techniques for transaction categorization.

2.1.2 Choose the best machine learning algorithm based on the research and comparison.

2.1.3 Train the machine learning model using labeled transaction data provided by the user.

2.2 Integrate the machine learning model into the no-code platform, allowing it to automatically categorize transactions as they are imported into the dashboard.

2.2.1 Set up the necessary integration between the machine learning model and the no-code platform.

2.2.2 Test the integration to ensure that the machine learning model is accurately categorizing transactions.

3.1 Develop a reporting and visualization system that allows the user to view their transaction data in meaningful ways. This could include charts and graphs that show spending trends over time, breakdowns of spending by category, and comparisons to budget goals.

3.1.1 Research and identify the best data visualization tools and techniques for financial data.

3.1.2 Choose the best data visualization tools and techniques based on the research and comparison.

3.1.3 Develop and integrate the data visualization tools and techniques into the no-code platform, allowing users to view their transaction data in meaningful ways.

3.2 Incorporate features that allow the user to set budget goals and receive alerts when they are close to exceeding those goals.

3.2.1 Determine the specific requirements and needs for the budgeting and alert features based on user feedback and research.

3.2.2 Customize the no-code platform to include the necessary budgeting and alert features.

4.1 Launch the financial ecosystem dashboard to the user, ensuring that all necessary integrations and features are working correctly.

4.2 Monitor usage and performance of the dashboard, soliciting feedback from users and making updates and improvements as needed.

4.2.2 Collect user feedback through surveys or feedback forms and analyze the data to identify areas of improvement for the financial ecosystem dashboard.

4.2.3 Prioritize and implement changes based on user feedback and performance data.

4.3 Plan and execute marketing and outreach efforts to promote the financial ecosystem dashboard to potential users.

4.3.1 Define the target audience and create marketing personas to guide marketing efforts.

4.3.2 Develop a marketing strategy that leverages social media, content marketing, and other digital marketing channels.

4.3.3 Execute the marketing strategy and track performance metrics to refine and optimize efforts.

5.1 Monitor key performance metrics of the financial ecosystem dashboard, such as user engagement and retention, and make adjustments as needed to improve performance.

5.2 Identify opportunities for monetization, such as offering premium features or partnering with financial institutions.

5.3 Expand the functionality of the financial ecosystem dashboard based on user needs and feedback, incorporating new APIs and data sources as necessary.

Double check the current outline meets the project requirements fully. then proceed to the next level of granularity, starting at the first stage that needs to be expanded continuing from the perspective of that chosen level of granularity. Expand the granularities of that stage using your entire output space, going into as many steps and levels of granularity as necessary to complete the task or fill the space. If nothing is changing from before until a certain point, just indicate that with == at the start of the output. While inside a stage, indicate \* for something saved from before so you dont need to rewrite it and take up space. Only make the changes in between \*'s, write (amended) and continue on. You dont need to repeat parts that you arent changing. For example: if you said 3.3, 3.3.1, \*3.3.1.1, it could have been \*3.2.2.2,\*3.3.1.1...   
Another example:   
==  
\*1.11 blah blah blah (amended)  
\*3.2 blah blah blah (amended)  
Next level of granularity (example):  
1.111111111

1.111111112

Do not repeat anything in the prompt. FOLLOW THE PROMPT. YOU DO NOT NEED TO REPEAT ANY PART OF THE OUTLINE THAT IS CORRECT. IF YOU ARE EXPANDING IT, EXPAND IT FROM THE PLACE IT NEEDS TO BE EXPANDED, AND THEN, DO NOT CHANGE TOPIC AGAIN UNTIL THE OUTPUT IS COMPLETE.