**Evaluation of the Best Prompt Engineering Template for Web Landing Page Requirements Analysis [DRAFT]**

Lin\*\*\*\*\*\*\*\*\*\*

Dept of Electrical Engineering and Computer Science, Florida Atlantic University

GenAI Software Dev. Lifecycles: Assignment 1 Prompt Engineering

Professor: Dr. Fernando Koch

February 15, 2025

Table of Contents

[Abstract 3](#_Toc190541909)

[Method 4](#_Toc190541910)

[Tools Used and Environment 4](#_Toc190541911)

[Files and General Settings Used for Configuration 4](#_Toc190541912)

[Results 4](#_Toc190541913)

[Test 1: Zero-Shots Prompting 5](#_Toc190541914)

[Test 2: Few-Shots Prompting 5](#_Toc190541915)

[Test 3: Prompt-Template Prompting 5](#_Toc190541916)

[Test 4: Chain-of-Thought 5](#_Toc190541917)

[Test 5: Meta Prompting 5](#_Toc190541918)

[Test 5: Chain-of-Thought 5](#_Toc190541919)

[Test 6: Self-Consistency 5](#_Toc190541920)

[Conclusion 6](#_Toc190541921)

# Abstract

Requirement Analysis involves listing tasks needed to complete a certain project that may be part of a larger-scoped project. While consistency with a single organization is encouraged, there are instances where certain functional areas may prefer or have a web landing page that is intuitative for their functional area of expertise. A Generative-AI tool can help business analysts to create requirements that can be delivered to software engineers to efficiently and quickly build the web pages for each functional area.

A fictional software web application development company, “EasyApps”, has been awarded a contract to build a single web application for a fictional company, “Structured Mechanics Enterprises”, but are instructed to make the user-facing landing page reflective of a functional area within “Structured Mechanics Enterprises”.

There are several Prompt Engineering techniques that “Easy Apps” would like to evaluate to provide the “Structured Mechanics Enterprises” business analysts to create requirements before they begin building the web application.

# 

**Evaluation of the Best Prompt Engineering Template for Web Landing Page Requirements Analysis**

At EasyApps there is a need to ensure … save time and money for the customer, in this case “Structured Mechanics Enterprises”. Our customer, Structured Mechanical Enterprises, requires a customized application. They prefer that we do not use third party tools to create the web pages. They would like their administrators to be able to dynamically create future web landing pages, as needed.

EasyApps will need to complete the Requirements Tools the SDLC of Problem Identification, Solution Ideation (generate ideas) and Requirements Analysis (needs, conditions any conflicts)

The goal is to research building one web landing page and have it dynamically layout the controls/elements for each of the six user groups, as shown in Table 1. The screen size varies from large “curved”, laptops, tablets, and mobile phones. Developers set the CSS style to media: screen then use 100%, however there are more detailed work to ensure a dynamically changing web page will always portion the elements on the page that is visually appealing.

“Structured Mechanics Enterprises” has these functional areas; each with a number of topics that can increase or decrease through time. The goal is to make increases or decreases without changing the code.

For the finished product the customer, “Structured Mechanical Enterprises”, would like their administrators to have privileges to input no more than 2 controls or selections to dynamically create the web page.

that some people have trouble asking humans the right questions in-order to get the right answer, so the prompt pipeline will greatly assist in this matter.

# Method

This research process is built on a computer/machine with software tools. Configurations are set to achieve the proper results.

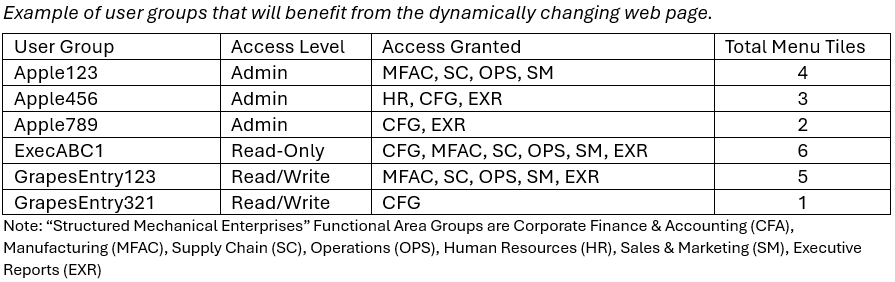
## Tools Used and Environment

* Ollama Server
* llama 3.2 Model
* Python 3.12
* Visual Studio 2022 (version 17.7.5)
* Machine: 8 Core (16 logical processors); Base Speed: 1.8 GHz; Available RAM: 7.4 GB; 12.5 MB cache memory; Windows 11 Operating System

**Files and General Settings Used for Configuration**

* \_ pipeline\_dynamicLayoutWebLanding\_Simplied.py (Level 1 and Level 2)
* \_ pipeline\_dynamicLayoutWebLanding\_Complex.py (Level 1 and Level 2)
* \_pipeline\_runALLprompts.py
* \_pipeline\_runFLEXprompts.py (Level 1 and Level 2)
* Variable(s) used by developers to input options for Prompt Method:
  + TotalMenuTiles: Enter total number of menus for the web page.
  + TileNames: Enter names or acronyms for the functional areas. (Reference Table 1)
  + GeneralInfo: Let the model know what the person is attempting to do.
  + WhichPrompt: Select from a variety of Prompt Methods [Default is Help\_Me\_Design].
  + HowMuchCTX: Determine the length of the Results.
  + PredictionsAmount: Determine the size or number of Predictions.
  + LearningLevel: This will help the model determine how to present results.
* **I**nitial Web Page Settings for each Prompt:
  + Web Page Width: 100%
  + Menu Tile Padding: 2%
  + Menu Tile (boxes on page): Height = Width
  + Set Privileged Access: Only areas where a user has access are displayed. (Access will become more granular, as users within certain groups may have fewer or more tiles.)
  + Maintain a consistence “row” format and not wrap unnecessarily.

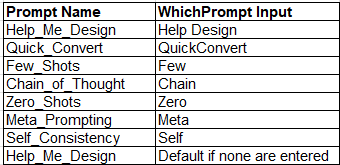
**Table 1**



**Prompt Methods**

If running the program, \_pipeline\_runALLprompts\_Complex.py, the “WhichPrompt” input selections are displayed on screen, as shown in the WhichPrompt Input column in Table 2.

**Table 2**



The ”Help\_Me\_Design” Prompt Method, is mostly <Instruction> and <Answer>. There are Level 1 and Level 2 prompts. Level 1 prompt handles the specific layout design and Level 2 retrieves results from Level 1 to indicate whether to provide additional information on creating with the Table element or Div element. The Prompt tries to steer the design toward using the Div element. Some developers prefer the Div element because it is more simplified, however, the Table element contains a solid structure. If the Table element is indicated in Level 1 then the GenAI model will return suggestions using the Table element. When using words such as box or tile; Cascading Style Sheets (CSS) or the Div element is suggested. The results can be reviewed in \_pipeline\_dynamicLayoutWebLanding\_Simplied.py and \_pipeline\_dynamicLayoutWebLanding\_Compex.py

The “Quick\_Convert” Prompt Method is a quick prompt to convert a layout value in percentage to pixels. The results can be reviewed in \_pipeline\_runALLprompts\_Complex.py.

This is an example of using the existing, Chain-of-Thought prompt method for this type of web layout requirements, for example:

“Q: What are the Total Number of Menu Tiles for User Group Kiwiw123?

A: The Total Menu Tiles are {TotalMenuTiles}.

<Instruction> Get 100 divided by {TotalMenuTiles}.

A: The additional 4% is for padding to the right. The answer is 12.5% for each Tile.”

Different scenarios can be run in \_pipeline\_runALLprompts\_Complex.py and \_pipeline\_runFLEXprompts.py

Using the existing PROMPT Few-Shots, the goal is to keep short by proving a precise calculation of the web layout requirements. For example:

CALCULATESIZE = “Calculate “ + 100 divided by {TotalMenuTiles}”

The answer will display the percentage of each of the “ + TotalMenuTiles . Different scenarios can be run in \_pipeline\_runALLprompts\_Complex.py or \_pipleline\_runFlexprompts.py.

The existing “Zero-Shot” prompt method simply sends an <Instruction> using the summarized calculation to return the size of each menu tile (box), for example:

“Calculate “ + 100% divided by {TotalMenuTiles}” - .02 [for the padding]. Only a single value or a very short length should be returned, so change the Variables “PredictionsAmount” to 1 and make the “HowMuchCTX” no more than 10.”

Different scenarios can be run in \_pipeline\_runALLprompts\_Complex.py or \_pipeline\_runFLEXprompt.py. “

While testing various prompt methods, the user may find that it is important to stay on topic or the model will return broad, unrelated answers, or it will remind the user to ask consistent questions. The prompt method, “Self-Consistency”, uses <Instruction> to direct a precise answer for suggesting requirements for the web layout. The \_pipeline\_runFLEXprompts.py offers flexibility to try different ways to return a one and two level options based on input from the user. It may work best to run the \_pipeline\_runFLEXprompts.py methods last so that the model has learned enough. However, “General Info” will keep on topic with developing web pages.

The previous prompts have been following the Meta Prompting guidelines in the way of organization and structure; however, this prompt is more refined, for example:

“<Instruction>First, research requirements for a web page with a width of 100%

Second, is it best to use percentage or pixels for the web page elements?

Third, is it best to use div element or table element or CSS to create boxes on the web page? Fourth, show an example.”

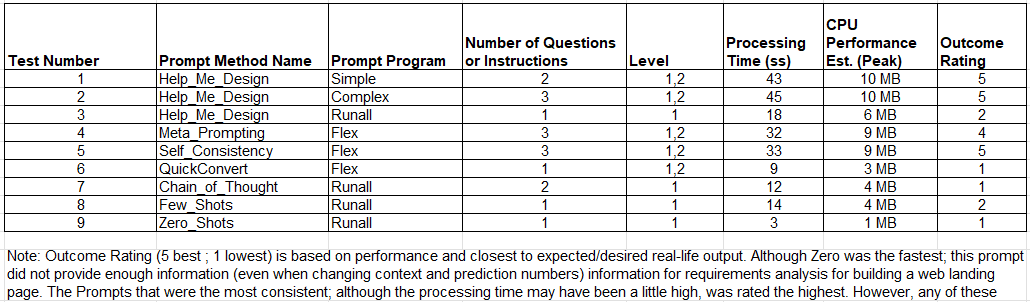
# Results

**Performance**

Prompt Engineering is designed to assist with development and offer insightful facts on different ways of getting requirements written. By running the same prompt several times, the answers become more refined. All of the results can be considered, as shown in Table 3.

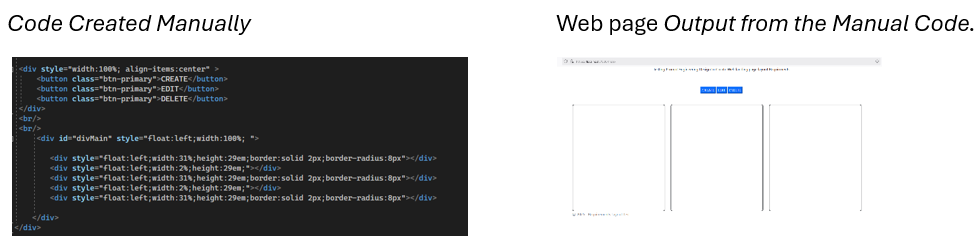
**Table 3**

*Results from several Prompt Method Tests*

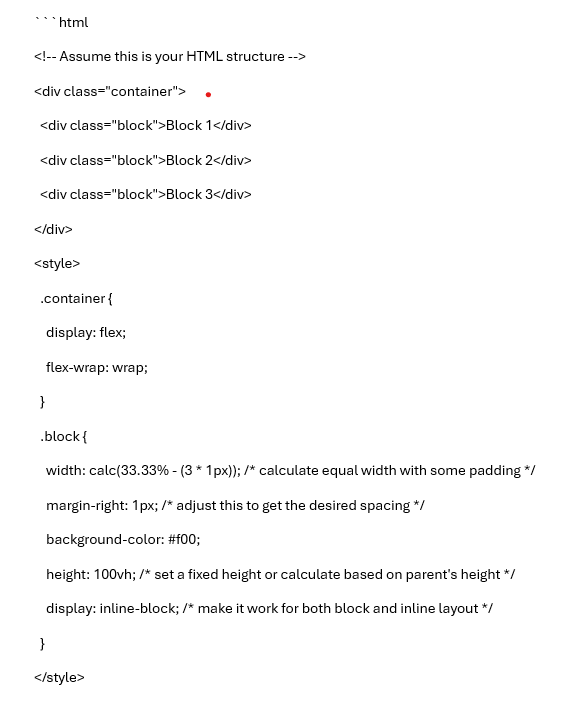
****

The expected outcome from the GenAI model should be close to matching this code created, manually, as shown in Figure 1, along with the page output.

**Figure 1**



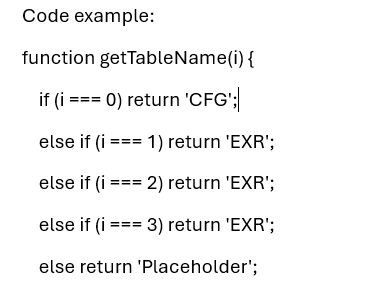
To test generated by the GenAI model, llama 3.2, the code was placed inside an HTML page. There should be some adjustments, but the GenAI model returned a close example of the layout with additions, using \_pipeline\_dynamicLayoutWebLanding\_Simplied.py.



While the final web landing page will have the titles on the tiles (boxes), as shown in Figure 2, it is important to prompt is such a way that avoid the model returning definitions of literal names that are specific to “Structured Mechanical Enterprises”.

**Figure 2**

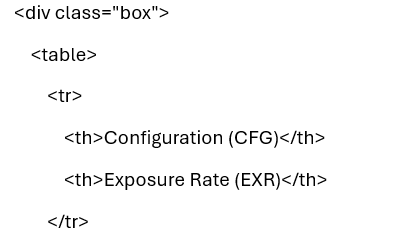
*Use names as identifiers and avoid the model looking up definitions.*



These names and acronyms are related only to Structured Mechanics Enterprises. Depending on how the prompt is engineered, the names could generate definitions more than help with the layout, as shown in Figure 3.

**Figure 3**

Avoid the lookup of definitions.

****

This is why the alternative Prompt method, “\_pipeline\_dynamicLayoutWebLanding\_Simplied.py”, may be a better option. Developers can code the names, as needed, or code the application to allow administrators of the application to input, select or edit names, or retrieve names from another system that stores the names that are needed.

The more Questions and Answers provided through the PROMPT pipeline the more refined the answers from the Prompt Engineering Tool. The GenAI continues to refine more answers about the Q and A.

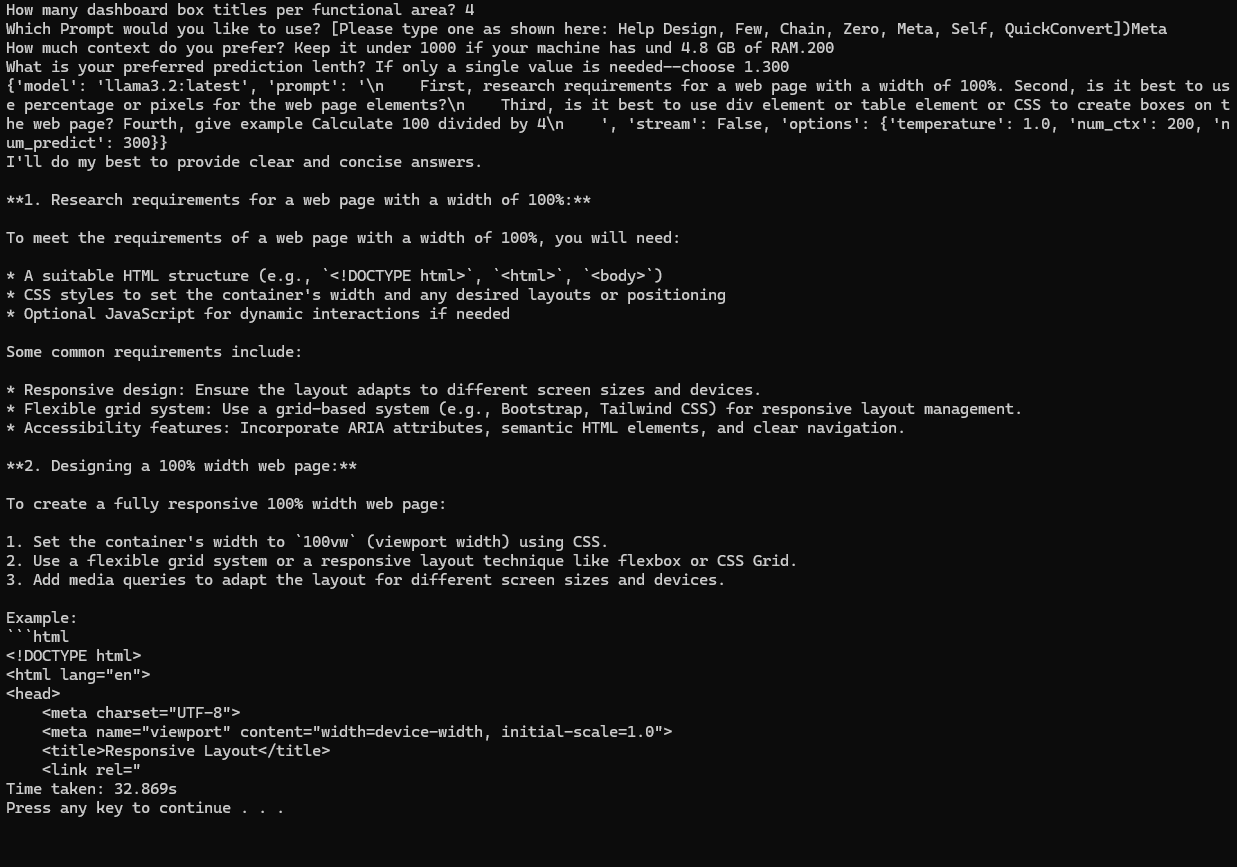
Note: Spelling is important, as the model learns from the user’s input.

**Some Examples of Outcome of Requirements Analysis based on GenAI Model llama 3.2**

Requirements can be built and analyzed by running the \_pipeline\_runFLEXprompts.py with the Meta Prompt Method, as shown in Figure 4.

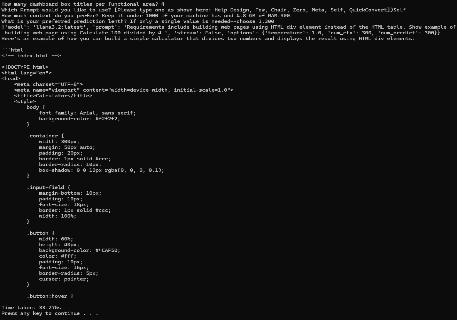
**Figure 4**

*Meta Prompting*



The following was run with using \_pipeline\_runFLEXprompts.py with the Self Consistency Prompt Method, as shown in Figure 5. This will help in building requirements.

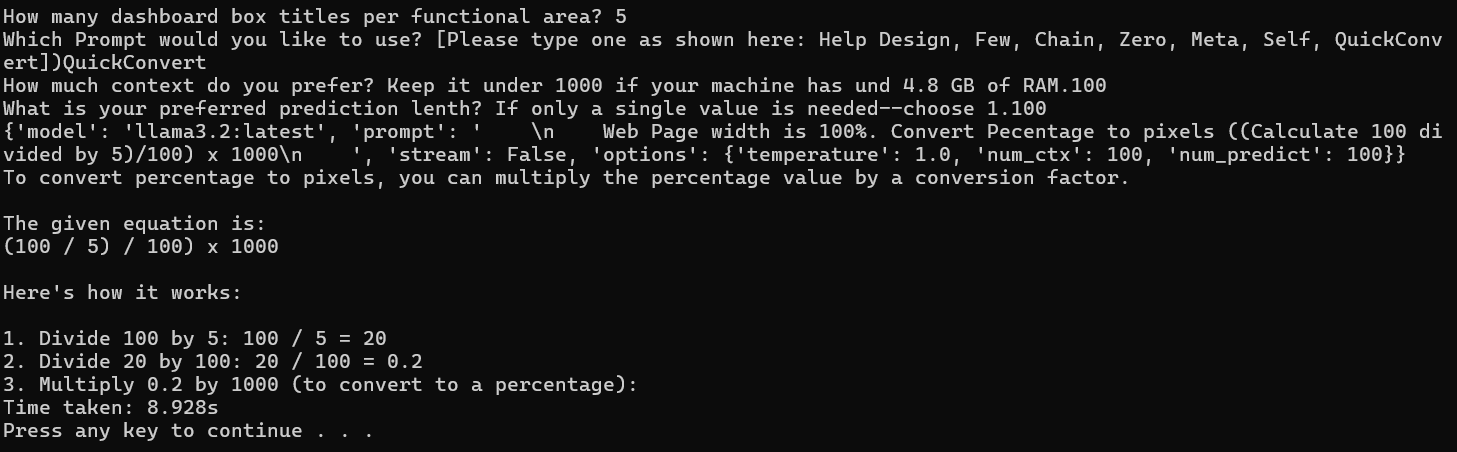
**Figure 5**

*Self Consistency Prompting*

This “Quick\_Convert” Prompt was run using the \_pipeline\_runFLEXprompts.py, as shown in Figure 6. As indicated there is not enough information to analyze the entire web page; although it is helpful.

**Figure 6**

*Quick Convert Prompting*



Running the “Help\_Me\_Design”, as shown in Figure 7, using the \_pipeline\_dynamicLayoutWebLanding\_Complex yields a significant amount of information for requirements analysis for building a web landing page. After several runs, the model eventually changed its mind from suggesting the Table was better than the Div element (Level 2 code section).

**Figure 7**

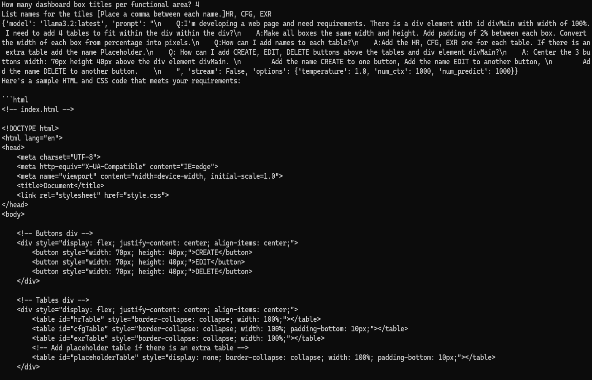
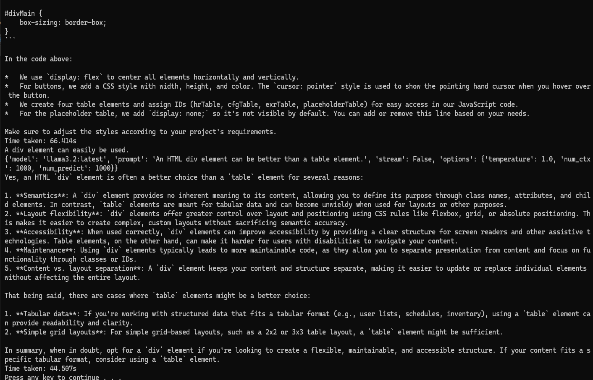
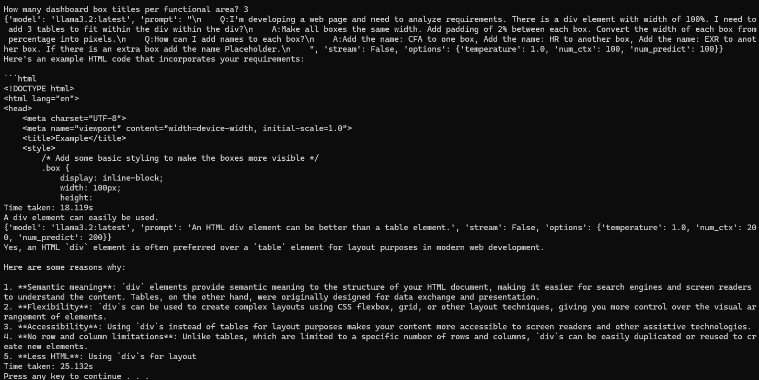


Figure 8 displays input and results from using \_pipeline\_dynamicLayoutWebLanding\_Simplied.py with Help\_Me\_Design Prompt.

**Figure 8**

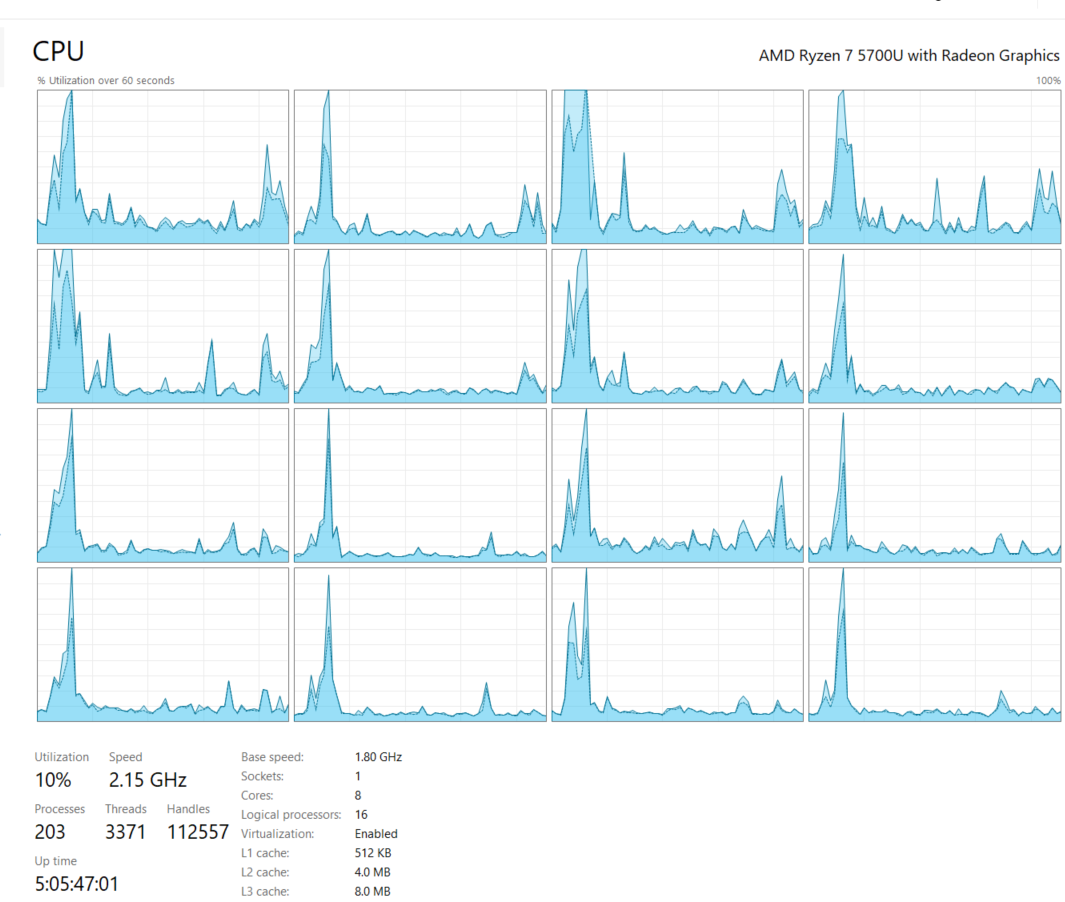
*Help Design Prompt Simplified*

**Performance**

The Ollama server is running on 8 Core multiprocessors (16 logical processors), as shown in Figure 9.

**Figure 9**

*Ollama Server Performance*



The “QuickConvert” prompt has one of the best performances, as far as CPU and GPU, using the \_pipeline\_runALLprompts\_Complex.py. One of the slowest performers was the “Help\_Me\_Design” run on pipeline\_runALLprompts\_Complex, as shown in Figure 10. Improvements can be made by running the \_\_pipeline\_runFLEXprompts.py, where adjustments can be made customized input values.

****Figure 10**

**Conclusion and Further research**

The model, llama 3.2, can also learn. One example of this is the Level 2 prompt that prompts the model to use “div elements” instead of tables. Several times, at the beginning of the results, the model suggested that the “table element” was best. Later the model suggested pros and cons of both, then finally the model began to suggest using “div elements” was better.

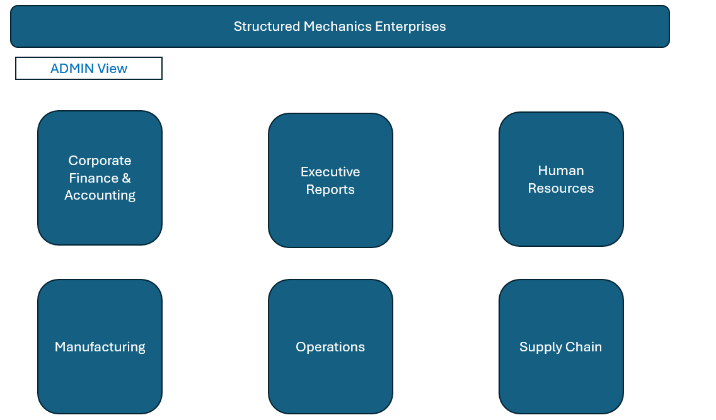
The model, llama 3.2, does not recognize the Ollama server. Different spelling were performed, however, the GenAI model indicated that it had no knowledge of Ollama.

It is possible that the research could go much further, but a cut-off point is necessary if you practice Agile methods. For example, analyzing the web landing page was the first stage.

Allowing user input to gather information from at least one person from each user group would improve the SDLC. The more input we have about how the user would like to use the system, the more behavior we can add. Humans actually supply the information; AI supplies assistance in the form of refined information to help build our product.

EasyApps can go further by using Prompt Engineering to create a web page UI with Python as the back-end code. Using Jinga and Flask typically works well for this. The Prompt Engineering program will certainly help to assist in building an entire web application. The final web layout landing page may look something like the image shown in Figure 11 (acronyms may replace the full names or be included).

**Figure 11**

*One of the dynamic web landing page’s examples*

# 