

✓ Enhance ShopAssist AI (ShopAssist 2.0)

Project by Neeraj Kumar Bhola

✓ Part 1: Introduction

✓ Project Background

In today's digital age, online shopping has become the go-to option for many consumers. However, the overwhelming number of choices and the lack of personalized assistance can make the shopping experience daunting. To address this, we have developed **ShopAssist AI, a chatbot that combines the power of large language models and rule-based functions to ensure accurate and reliable information delivery.**


Problem Statement

Given a dataset containing information about laptops (product names, specifications, descriptions, etc.), build a chatbot that parses the dataset and provides accurate laptop recommendations based on user requirements.

You can load the data and see it here.

```
1 # Install OpenAI library
2 !pip install -q openai tenacity
3
```

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

 Mounted at /content/drive

```
1 # Import other libraries
2 import pandas as pd
3 from IPython.display import display, HTML
4 pd.set_option('display.width',100)
```

```
1 filepath = '/content/drive/MyDrive/GenAI_Project/'
```

Approach:

1. **Conversation and Data Collection:** The chatbot engages in natural dialogue using LLMs to understand user preferences and requirements.
2. **Information Extraction:** Once key information is gathered, rule-based functions identify the top three laptops that best match the user's needs.
3. **Personalized Recommendations:** The chatbot continues the conversation to answer follow-up questions and assist users in selecting the ideal laptop.

✓ Part 2: System Design

Dataset

We have a dataset `laptop_data.csv` where each row describes the features of a single laptop and also has a small description at the end. The chatbot that we build will leverage LLMs to parse this `Description` column and provide recommendations

✓ Workings of the Chatbot

The chatbot should ask a series of questions to determine the user's requirements. To make it simple, we have used 6 features to encapsulate the user's needs.

To identify user needs, the chatbot focuses on six key features: - GPU intensity - Display quality - Portability - Multitasking - Processing speed - Budget

- Confirm if the user's requirements have been correctly captured at the end.

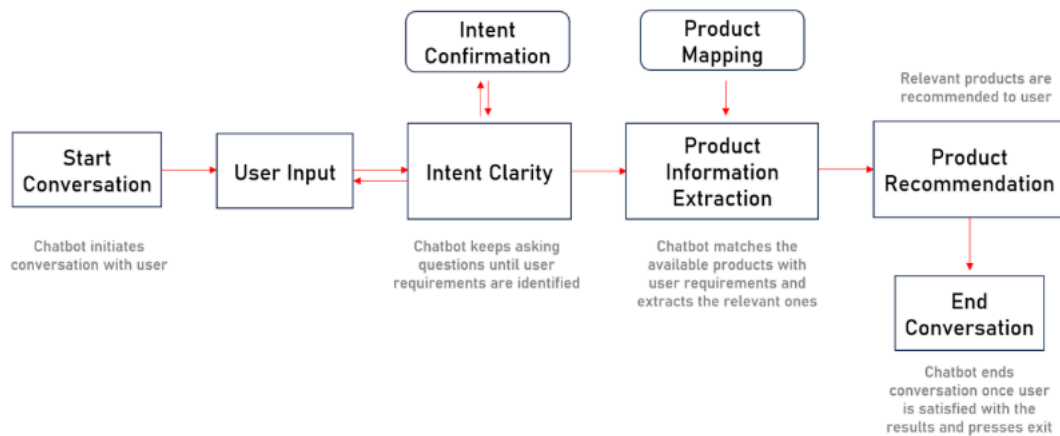
After confirming the user's preferences, the chatbot lists the top 3 matching laptops and continues to assist with further queries.

✓ ShopAssist 1.0

✓ Building the Chatbot

Below is the system design for the chatbot for **ShopAssist 1.0**.

CHATBOT SYSTEM DESIGN



Chatbot System Design

Stage 1

- Intent Clarity Layer
- Intent Confirmation Layer

Stage 2

- Product Mapping Layer
- Product Information Extraction Layer

Stage 3

- Product Recommendation Layer

✓ Major functions behind the Chatbot ShopAssist

Let us now look at a brief overview of the major functions that form the chatbot and then deep dive later:

- `initialize_conversation()` : This initializes the variable conversation with the system message.
- `get_chat_completions()` : This takes the ongoing conversation as the input and returns the response by the assistant
- `moderation_check()` : This checks if the user's or the assistant's message is inappropriate. If any of these is inappropriate, it ends the conversation.
- `intent_confirmation_layer()` : This function takes the assistant's response and evaluates if the chatbot has captured the user's profile clearly. Specifically, this checks if the following properties for the user has been captured or not GPU intensity, Display quality, Portability, Multitasking, Processing speed, Budget
- `dictionary_present()` : This function checks if the final understanding of user's profile is returned by the chatbot as a python dictionary or not. If there is a dictionary, it extracts the information as a Python dictionary.
- `compare_laptops_with_user()` : This function compares the user's profile with the different laptops and come back with the top 3 recommendations.
- `initialize_conv_reco()` : Initializes the recommendations conversation

In the next sections, we will see how to write the code for the above functions

✓ ShopAssist 2.0

But in development of **ShopAssist2.0**, we leverage from Function Calling feature for improved performance and simple architecture.

The architecture looks as shown below:

Stage 1

- Intent Clarity Layer

Stage 2

- Product Mapping Layer

Stage 3

- Product Recommendation Layer

The following layers from ShopAssist 1.0 **are removed** as their purpose are filled by now using **Function Calling** in OpenAI API.

1. intent_confirmation_layer()
2. dictionary_present()
3. initialize_conv_reco()

Part 3: Implementation

✓ Part-2 System Design and Implementation - Stage 1

✓ 1.1 - Import the libraries

Importing libraries that are require for this project:

- openai
- os, json, ast
- pandas

Make sure the api key is stored in the text file `OPENAI_API_Key.txt` in your google drive linked above.

```
1 # Import the libraries
2 import os, json, ast
3 import openai
4 from tenacity import retry, wait_random_exponential, stop_after_attempt
```

```
1 # Read the OpenAI API key from Gdrive
2 with open(filepath + "OpenAI_API_Key.txt", "r") as f:
3     openai.api_key = ' '.join(f.readlines())
```

1.2 - Implementing Intent Clarity Layer

- initialize_conversation()

✓ initialize_conversation():

This initializes the variable conversation with the system message. Using prompt engineering and chain of thought reasoning, the function will enable the chatbot to keep asking questions until the user requirements have been captured in a dictionary. It also includes Few Shot Prompting (sample conversation between the user and assistant) to align the model about user and assistant responses at each step.

```
1 # initialize_conversation function
2 def initialize_conversation():
3     '''
4     Returns a list [{'role':'system','content': system_message}]
5     '''
6     delimiter = '####'
7
8
9     system_message = f'''
10    You are very intelligent laptop gadget expert and your goal is to find out best laptop for a user for given input.
11    You need to ask relevant questions to the user and understand the user profile by analysing the user's responses.
12    Your final objective is to get following parameters from user's conversation. The parameters are 'GPU intensity','Display quality
13    You need to ask probing questions to the user in case any of these parameters are not obtained from user's interaction.
14    The value for the 'Budget' should be numerical value extracted from user's response.
15    The value of all keys except 'Budget' should be either 'high' or 'medium' or 'low' based on the importance of the corresponding
16    All the values in the example dictionary are only representative values.
17    {delimiter}
18    Here are some instructions around the values for the different keys. if you do not follow this, you will be heavily penalized.
```

```

19 - The value of all keys except 'Budget' should be strictly either 'high' or 'medium' or 'low' based on the importance of the corr
20 - The value for the 'Budget' should be numerical value extracted from user's response.
21 - 'Budget' value needs to be greater than or equal to 24999 INR. If the user says less than that, please mention to user that the
22 - Please do not randomly assign any value to the parameters.
23 - The value needs to be inferred from the user's response.
24 - Please ask one question at a time to capture values for the parameters.
25 {delimiter}
26
27 Once you have obtained all the parameter, your goal is to extract details of the top 3 laptops matching to the parameters obtained
28 And this you will do by function calling with compare_laptops_with_user function.
29
30 {delimiter}
31 Once you get the list of top 3 laptops you will need to present the details in the personalized format it and show the recommendation
32 user in the following format:
33 1. <Laptop name>: <Basic laptop specs in brief>, <price of laptop>
34 2. <Laptop name>: <Basic laptop specs in brief>, <price of laptop>
35 3. <Laptop name>: <Basic laptop specs in brief>, <price of laptop>
36 {delimiter}
37
38 Follow below steps when interacting with user.
39
40 Step 1 : Start with a short welcome message and encourage the user to share their requirement precisely on the laptop needed.
41 Remember, you only recommend on laptop, if user is asking anything else then you apologize and remind user that you are laptop assistant
42
43 Step 2 : Based on the user input obtained from conversation, you try to get values for 'GPU intensity','Display quality','Portability'
44
45 Step 3 : In case, details are not clear , ask clarifying question to user to get the details for above parameters.
46
47 Step 4 : When all the parameters are available you invoke function calling with compare_laptops_with_user function.
48
49 Step 5 : Personalize the recommendation obtained from compare_laptops_with_user function and present to the user.
50
51 Step 6 : If user changes any of the request then repeat step 2 to 5.
52
53
54 {delimiter}
55 Here are some notes which you should take care of. If you do not follow this you will be heavily penalized.
56 - The values for all keys, except 'Budget', should strictly be either 'low', 'medium', or 'high' based on the importance of the key
57 - The value for 'budget' should be a numerical value extracted from the user's response. It needs to be greater than or equal to 24999 INR
58 - Do not ask questions to user for more 1 parameter at a time.
59 - Do not ask too many questions in one shot.
60 - Do not reply on any other subject other than laptop and its parameters.
61
62
63 {delimiter}
64 Here is a sample conversation between user and AI assistant:
65 User Input : 'Hi, I am an editor.'
66 Assistant : 'Great! As an editor, you likely require a laptop that can handle demanding tasks. Hence , the laptop should have high performance'
67 User Input : ' I primarily work with after effects. '
68 Assistant : ' Thank you for providing that information. Working with after effects involves working with graphics, animations and video editing'
69 User Input : ' Yes, sometimes I work with 4K videos as well. '
70 Assistant : ' Thank you for the information. Processing 4K videos will require a good and high GPU. I think we have already determined that'
71 User Input : ' Yes, sometimes I travel but do not carry my laptop. '
72 Assistant : ' Could you kindly let me know your budget for the laptop ? This will help me to find the options that fit within your budget'
73 User Input : ' My maximum budget is 1.5 lakh INR. '
74
75 {delimiter}
76
77 {delimiter}
78 Here is another sample conversation between the user and assistant:
79 User Input: "I am a coder and i need higher end laptop with high multitasking feature"
80 Assistant: "Great! As a coder, you likely require a laptop that can handle demanding tasks. Hence, the laptop should have high performance and multitasking capabilities"
81 User Input: "Yes, you are right"
82 Assistant: "Great! Do you prefer a laptop that is easy to carry around or do you primarily use it at a fixed location?"
83 User Input: "I only need it for home use."
84 Assistant: "Thank you for specifying that. Do you often have multiple tasks running on your laptop simultaneously or primarily for coding?"
85 User Input: "I usually have multiple tasks running"
86 Assistant: "Thank you for sharing that. Do you prefer a laptop with fast processing speed for quick performance?"
87 User Input: "yes i need it to be very quick"
88 Assistant: "Appreciate that input. Finally, what is your budget range for the laptop you are looking to purchase?"
89 User Input: "i would like if it is not more than 125000 INR"
90
91
92 {delimiter}
93 Here is another sample conversation between the user and assistant:
94 User Input: "I am a basic business user"
95 Assistant: "As a basic business user, it's important to have a reliable laptop for your day-to-day tasks. Based on your usage as a business user, I recommend a laptop with good performance, portability, and battery life."
96 User Input: "Yes, you right"
97 Assistant: "Great! Do you prefer a laptop that is easy to carry around or do you primarily use it at a fixed location?"
98 User Input "I need it to be easy to carry around as I travel frequently on business trips"
99 Assistant: "Thank you for specifying that. Do you often have multiple tasks running on your laptop simultaneously or primarily for business tasks?"
100 User Input: "i usually have multiple tasks running"

```

```

101 Assistant: "Thank you for sharing that. Do you prefer a laptop with fast processing speed for quick performance?"
102 User Input: "yes i need it to be very quick"
103 Assistant: "Appreciate that input. Finally, what is your budget range for the laptop you are looking to purchase?"
104 User Input: "i would like if it is not more than 1 lakh"
105
106
107 '''
108
109 conversation = [{'role' : 'system' , 'content' : system_message}]
110
111 return conversation

```

Let's now look at the next function.

- `get_chat_completions()`: This takes the ongoing conversation as the input and returns the response by the assistant. We'll use the Chat Completions function for performing LLM calls to OpenAI.

✓ `get_chat_completions()`:

This function perform LLM call using the Chat Completions API to get the LLM response.

```

1 # Define a Chat Completions API call
2 # Retry up to 6 times with exponential backoff, starting at 1 second and maxing out at 20 seconds delay
3 @retry(wait=wait_random_exponential(min=1, max=20), stop=stop_after_attempt(6))
4 def get_chat_completions(input, json_format = False):
5
6     MODEL = 'gpt-4o-mini'
7
8     system_message_json_output = """<<. Return output in JSON format to the key output.>>""
9
10
11     try:
12         # If the output is required to be in JSON format
13         if json_format == True:
14             # Append the input prompt to include JSON response as specified by OpenAI
15             messages = input + [{"role": "system", "content": system_message_json_output}]
16
17             # JSON return type specified
18             chat_completion_json = openai.chat.completions.create(
19                 model = MODEL,
20                 messages = messages,
21                 response_format = { "type": "json_object"},
22                 seed = 1234)
23
24             output = json.loads(chat_completion_json.choices[0].message.content)
25
26         # No JSON return type specified
27         else:
28             message = input
29             chat_completion = openai.chat.completions.create(
30                 model = MODEL,
31                 messages = input,
32                 seed = 2345)
33
34             output = chat_completion.choices[0].message.content
35         return output
36
37     # Handling exception
38     except Exception as e:
39         print(f"An error occurred while calling OpenAI API : {e}")
40         return "None"

```

✓ `moderation_check()`:

This checks if the user's or the assistant's message is inappropriate. If any of these is inappropriate, you can add a break statement to end the conversation.

```

1 # Define a function called moderation_check that takes user_input as a parameter.
2
3 def moderation_check(user_input):
4     # Call the OpenAI API to perform moderation on the user's input.
5     response = openai.moderations.create(input=user_input)
6
7     # Extract the moderation result from the API response.
8     moderation_output = response.results[0].flagged
9     # Check if the input was flagged by the moderation system.

```

```

10     if response.results[0].flagged == True:
11         # If flagged, return "Flagged"
12         return "Flagged"
13     else:
14         # If not flagged, return "Not Flagged"
15         return "Not Flagged"

```

✓ Stage 2

2.1 Implementing the Product Mapping

✓ product_map_layer():

This function is responsible for extracting key features and criteria from laptop descriptions. Here's a breakdown of how it works:

- Use a prompt that assign it the role of a Laptop Specifications Classifier, whose objective is to extract key features and classify them based on laptop descriptions.
- Provide step-by-step instructions for extracting laptop features from description.
- Assign specific rules for each feature (e.g., GPU Intensity, Display Quality, Portability, Multitasking, Processing Speed) and associate them with the appropriate classification value (Low, Medium, or High).
- Includes Few Shot Prompting (sample conversation between the user and assistant) to demonstrate the expected result of the feature extraction and classification process.

```

1 def product_map_layer(laptop_description):
2     delimiter = "#####"
3
4     lap_spec = {
5         "GPU intensity": "(Type of the Graphics Processor)",
6         "Display quality": "(Display Type, Screen Resolution, Display Size)",
7         "Portability": "(Laptop Weight)",
8         "Multitasking": "(RAM Size)",
9         "Processing speed": "(CPU Type, Core, Clock Speed)"
10    }
11
12    values = {'low', 'medium', 'high'}
13
14    prompt=f"""
15    You are a Laptop Specifications Classifier whose job is to extract the key features of laptops and classify them as per their re
16    To analyze each laptop, perform the following steps:
17    Step 1: Extract the laptop's primary features from the description {laptop_description}
18    Step 2: Store the extracted features in {lap_spec} \
19    Step 3: Classify each of the items in {lap_spec} into {values} based on the following rules: \
20    {delimiter}
21    GPU Intensity:
22    - low: <<< if GPU is entry-level such as an integrated graphics processor or entry-level dedicated graphics like Intel UHD >>> , \n
23    - medium: <<< if mid-range dedicated graphics like M1, AMD Radeon, Intel Iris >>> , \n
24    - high: <<< high-end dedicated graphics like Nvidia RTX >>> , \n
25
26    Display Quality:
27    - low: <<< if resolution is below Full HD (e.g., 1366x768). >>> , \n
28    - medium: <<< if Full HD resolution (1920x1080) or higher. >>> , \n
29    - high: <<< if High-resolution display (e.g., 4K, Retina) with excellent color accuracy and features like HDR support. >>> \n
30
31    Portability:
32    - high: <<< if laptop weight is less than 1.51 kg >>> , \n
33    - medium: <<< if laptop weight is between 1.51 kg and 2.51 kg >>> , \n
34    - low: <<< if laptop weight is greater than 2.51 kg >>> \n
35
36    Multitasking:
37    - low: <<< If RAM size is 8 GB, 12 GB >>> , \n
38    - medium: <<< if RAM size is 16 GB >>> , \n
39    - high: <<< if RAM size is 32 GB, 64 GB >>> \n
40
41    Processing Speed:
42    - low: <<< if entry-level processors like Intel Core i3, AMD Ryzen 3 >>> , \n
43    - medium: <<< if Mid-range processors like Intel Core i5, AMD Ryzen 5 >>> , \n
44    - high: <<< if High-performance processors like Intel Core i7, AMD Ryzen 7 or higher >>> \n
45    {delimiter}
46
47    {delimiter}
48    Here is input output pair for few-shot learning:
49    input 1: "The Dell Inspiron is a versatile laptop that combines powerful performance and affordability. It features an Intel Cor
50    output 1: {'GPU intensity': 'medium', 'Display quality': 'medium', 'Portability': 'medium', 'Multitasking': 'high', 'Processing speed
51

```

```

52     {delimiter}
53     """ Strictly don't keep any other text in the values of the JSON dictionary other than low or medium or high """
54     """
55     input = f"""Follow the above instructions step-by-step and output the dictionary in JSON format {lap_spec} for the following la
56     #see that we are using the Completion endpoint and not the Chatcompletion endpoint
57     messages=[{"role": "system", "content":prompt },{"role": "user","content":input}]
58
59     response = get_chat_completions(messages, json_format = True)
60
61     return response

```

1 Start coding or [generate](#) with AI.

Let's now apply this function to the entire laptop dataset

```

1 ##Run this code once to extract product info in the form of a dictionary
2 laptop_df= pd.read_csv(filepath + 'laptop_data.csv')

```

```
1 laptop_df.columns
```

```

→ Index(['Brand', 'Model Name', 'Core', 'CPU Manufacturer', 'Clock Speed', 'RAM Size',
        'Storage Type', 'Display Type', 'Display Size', 'Graphics Processor', 'Screen Resolution',
        'OS', 'Laptop Weight', 'Special Features', 'Warranty', 'Average Battery Life', 'Price',
        'Description'],
        dtype='object')

```

```

1 # Understanding the text of Description column
2 print(laptop_df.Description)

```

```

→ 0    The Dell Inspiron is a versatile laptop that c...
1    The MSI GL65 is a high-performance laptop desi...
2    The HP EliteBook is a premium laptop designed ...
3    The Lenovo IdeaPad is a versatile laptop that ...
4    The ASUS ZenBook Pro is a high-end laptop that...
5    The Acer Predator is a powerhouse laptop desig...
6    The Microsoft Surface Laptop is a premium devi...
7    The Lenovo ThinkPad is a powerful laptop desig...
8    The HP Pavilion is a budget-friendly laptop th...
9    The ASUS ROG Strix G is a high-performance gam...
10   The Dell XPS 15 is a premium laptop that combi...
11   The Lenovo ThinkPad X1 Carbon is a sleek and l...
12   The Acer Swift 3 is a lightweight and affordab...
13   The Apple MacBook Air is a sleek and lightweig...
14   The MSI Prestige 14 is a compact and stylish l...
15   The ASUS ZenBook 13 is a lightweight and power...
16   The Dell Precision 5550 is a high-performance ...
17   The HP ENVY x360 is a versatile 2-in-1 convert...
18   The Razer Blade 15 is a high-performance gamin...
19   The Apple MacBook Pro is a high-end laptop tha...
Name: Description, dtype: object

```

```

1 ## Create a new column "laptop_feature" that contains the dictionary of the product features
2 laptop_df['laptop_feature'] = laptop_df['Description'].apply(lambda x: product_map_layer(x))

```

```

1 # checking for the laptop_feature column
2 laptop_df.columns

```

```

→ Index(['Brand', 'Model Name', 'Core', 'CPU Manufacturer', 'Clock Speed', 'RAM Size',
        'Storage Type', 'Display Type', 'Display Size', 'Graphics Processor', 'Screen Resolution',
        'OS', 'Laptop Weight', 'Special Features', 'Warranty', 'Average Battery Life', 'Price',
        'Description', 'laptop_feature'],
        dtype='object')

```

```

1 # checking the data in laptop_feature column
2 print(laptop_df.laptop_feature)

```

```

→ 0    {'output': {'GPU intensity': 'low', 'Display q...
1    {'output': {'GPU intensity': 'high', 'Display ...
2    {'output': {'GPU intensity': 'low', 'Display q...
3    {'output': {'GPU intensity': 'low', 'Display q...
4    {'output': {'GPU intensity': 'high', 'Display ...
5    {'GPU intensity': 'high', 'Display quality': '...
6    {'output': {'GPU intensity': 'medium', 'Displa...
7    {'output': {'GPU intensity': 'high', 'Display ...
8    {'output': {'GPU intensity': 'low', 'Display q...
9    {'output': {'GPU intensity': 'high', 'Display ...
10   {'output': {'GPU intensity': 'high', 'Display ...
11   {'output': {'GPU intensity': 'low', 'Display q...
12   {'output': {'GPU intensity': 'medium', 'Displa...
13   {'output': {'GPU intensity': 'medium', 'Displa...
14   {'output': {'GPU intensity': 'high', 'Display ...

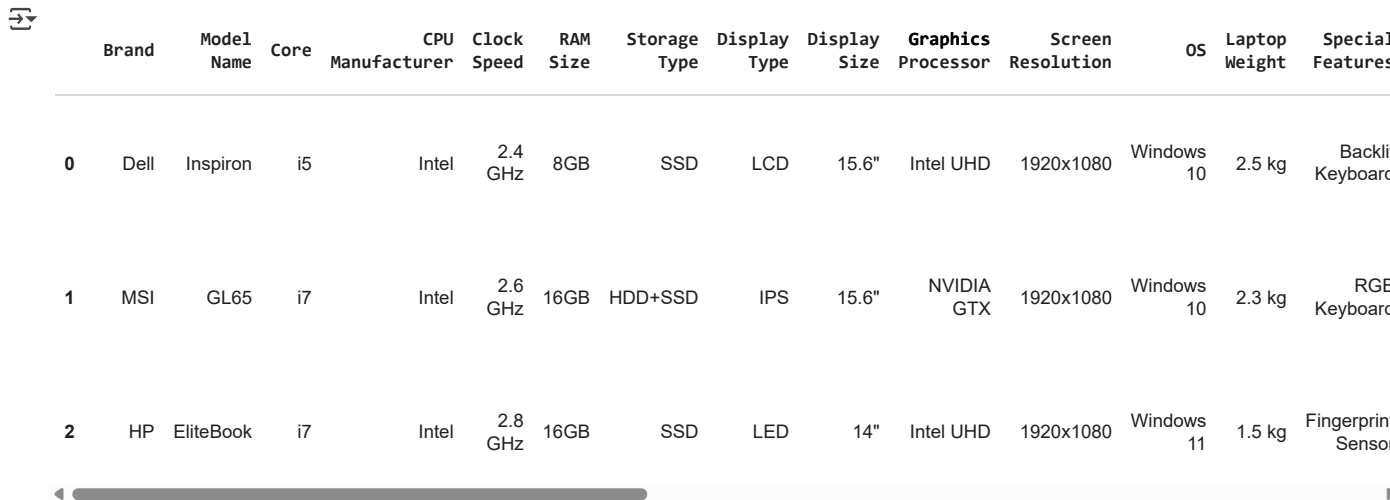
```

```

15 {'output': {'GPU intensity': 'medium', 'Displa...
16 {'output': {'GPU intensity': 'high', 'Display ...
17 {'output': {'GPU intensity': 'medium', 'Displa...
18 {'output': {'GPU intensity': 'high', 'Display ...
19 {'output': {'GPU intensity': 'medium', 'Displa...
Name: laptop_feature, dtype: object

```

```
1 laptop_df.head(5)
```



	Brand	Model Name	Core	CPU Manufacturer	Clock Speed	RAM Size	Storage Type	Display Type	Display Size	Graphics Processor	Screen Resolution	OS	Laptop Weight	Special Features
0	Dell	Inspiron	i5	Intel	2.4 GHz	8GB	SSD	LCD	15.6"	Intel UHD	1920x1080	Windows 10	2.5 kg	Backlit Keyboard
1	MSI	GL65	i7	Intel	2.6 GHz	16GB	HDD+SSD	IPS	15.6"	NVIDIA GTX	1920x1080	Windows 10	2.3 kg	RGB Keyboard
2	HP	EliteBook	i7	Intel	2.8 GHz	16GB	SSD	LED	14"	Intel UHD	1920x1080	Windows 11	1.5 kg	Fingerprint Sensor

Next steps:

[Generate code with laptop_df](#)
[View recommended plots](#)
[New interactive sheet](#)

▼ compare_laptops_with_user():

This function evaluates the user's profile against available laptops and returns the top recommendations. It follows these steps: - Accepts a dictionary of user requirements as input.

- Filters the laptops to include only those within the user's budget.
- Assigns a score to each laptop based on how closely it aligns with the user's preferences.
- Sorts the laptops in descending order of their scores.
- Returns the top three laptops in a JSON-formatted string.

```

1 def compare_laptops_with_user(user_req_string):
2
3     # reading updated input data for laptop details which has features details
4     df = pd.read_csv(filepath + 'laptop_dataset_with_features.csv')
5     user_requirements = user_req_string
6
7     # extract budget value from user requirement and convert into integer
8     budget = user_requirements.get('Budget',0)
9
10    # create a copy of dataframe after filtering on the budget
11    df_filtered = df.copy()
12    df_filtered['Price'] = df_filtered['Price'].str.replace(',','').astype(int)
13    df_filtered = df_filtered[df_filtered['Price'] <= budget]
14
15    # Mapping string value low medium and high to 0,1 and 2 respectively
16    mappings = {'low' : 0 , 'medium' : 1 , 'high' : 2}
17
18    # creating a new column score and initializing it to 0
19    df_filtered['Score'] = 0
20
21    # now for each laptop, calculate the score based on the input requirement
22    for index,row in df_filtered.iterrows():
23        laptop_values = ast.literal_eval(row['laptop_feature'])
24        score = 0
25
26        # now comparing the keys in the laptop values and user requirement
27        for key,user_value in user_requirements.items():
28            if key == 'Budget':
29                continue # skip the comparision
30            laptop_value = laptop_values.get(key,None)
31
32            # checking in mapping

```



```

33     laptop_mapping = mappings.get(laptop_value,-1)
34     user_mapping    = mappings.get(user_value,-1)
35
36     if laptop_mapping >= user_mapping:
37         score += 1
38
39     df_filtered.loc[index,'Score'] = score # updating the score to dataframe
40
41 # sorting laptops by score in descending order and selecting 3 products
42 top_laptops = df_filtered.drop('laptop_feature', axis = 1)
43 top_laptops = top_laptops.sort_values('Score',ascending=False).head(3)
44
45 # converting the top_laptops dataframe to json format
46 top_laptops_json = top_laptops.to_json(orient='records')
47
48 return top_laptops_json
49
50

```

✓ Stage 3

✓ 3.1: Product Recommendation Layer

The final stage is the product recommendation layer. It utilizes the output from the 'compare_laptops_with_user' function and delivers the results to the user. This layer performs the following actions:

1. Continues the dialogue to guide the recommendation process.
2. Formats and presents the recommended laptops in a user-friendly manner.
3. Engages the user with follow-up questions based on the provided recommendations.

```

1 def get_chat_completions_function_calling(input):
2
3     model = 'gpt-4o-mini'
4
5     # defining the function call parameters
6     tools = [
7         {
8             "type": "function",
9             "function": {
10                 "name": "compare_laptops_with_user",
11                 "description": "Get the top 3 laptops for the user from the catalogue available based on parameters like GPU intensi
12                 "parameters": {
13                     "type": "object",
14                     "properties": {
15                         "GPU intensity": {
16                             "type": "string",
17                             "description": "The GPU intensity requirement of the user specified as low, medium or high"
18                     },
19                     "Display quality": {
20                         "type": "string",
21                         "description": "The Display Quality requirement of the user specified as low, medium or high"
22                     },
23                     "Portability": {
24                         "type": "string",
25                         "description": "The Portability requirement of the user specified as low, medium or high"
26                     },
27                     "Multitasking": {
28                         "type": "string",
29                         "description": "The Multitasking requirement of the user specified as low, medium or high"
30                     },
31                     "Processing speed": {
32                         "type": "string",
33                         "description": "The Processing speed requirement of the user specified as low, medium or high"
34                     },
35                     "Budget": {
36                         "type": "integer",
37                         "description": "The maximum budget of the user"
38                     },
39                 },
40                 "required": [
41                     "GPU intensity",
42                     "Display quality",
43                     "Portability",
44                     "Multitasking",
45                     "Processing speed",
46                     "Budget"
47                 ]

```

```

48         }
49     }
50 }
51 ]
52
53
54 try:
55     messages = input
56
57     response = openai.chat.completions.create(
58         model = model,
59         messages = messages,
60         temperature = 0,
61         tools = tools,
62         tool_choice = 'auto',
63         seed = 4567
64     )
65
66     # check if the model wanted to call a function
67     tool_calls = response.choices[0].message.tool_calls
68
69     # call the function
70     if tool_calls:
71         available_functions = {
72             "compare_laptops_with_user": compare_laptops_with_user,
73         }
74
75         # append response given by gpt to input messages list
76         messages.append(response.choices[0].message)
77
78         for tool_call in tool_calls:
79             function_name = tool_call.function.name
80             function_to_call = available_functions[function_name]
81             function_args = json.loads(tool_call.function.arguments)
82             function_response = function_to_call(function_args)
83
84             function_call_response_dict = {
85                 "tool_call_id": tool_call.id,
86                 "role": "tool",
87                 "name": function_name,
88                 "content": function_response,
89             }
90
91             # append response_messages to the original input messages
92             messages.append(function_call_response_dict)
93
94             # make a second call to the LLM to personalize the function output
95             second_response = openai.chat.completions.create(
96                 model = model,
97                 messages = messages,
98                 temperature = 0 ,
99                 seed = 5678
100
101             )
102
103             second_response_message = [{"role": "assistant", "content": second_response.choices[0].message.content}]
104             return second_response_message
105     else:
106         response_message = [{"role": "assistant", "content": response.choices[0].message.content}]
107         return response_message
108
109     # Raise exception error
110 except Exception as e:
111     print(f"An error occurred: {e}")
112     return None

```

✓ Stage 4 - Combining all the 3 stages

Now, we will combine all three stages that we defined above.

Stage 1 + Stage 2 + Stage 3

✓ 4.1 Dialogue Management System

Bringing everything together, we create a `dialogue_mgmt_system()` function that contains the logic of how the different layers would interact with each other. This will be the function that we'll call to initiate the chatbot

```

1 def dialogue_mgmt_system():
2     ...

```

```
2 # initialize the conversation
3 conversation = initialize_conversation()
4 # First message from Assistant
5 print("\nAssistant:\nHow may I help you with your laptop selection? To exit
the conversation type 'EXIT'.\n")

6
7 user_input = ''
8
9 # Loop till user exits
10 while user_input.lower() != 'exit':
11     print("User Input: ")
12     user_input = input()
13     # Moderation Check for user input
14     moderation = moderation_check(user_input)
15     if moderation == 'Flagged':
16         print("\nAssistant:\nSorry, this message has been flagged. Please
restart your conversation.\n")
17         break
18
19     if user_input.lower() == 'exit':
20         print("\nAssistant:\nThank you for using this service. Please do
not hesitate to contact us if you need assistance. See you soon!\n")
21         break
22
23     conversation.append({"role": "user", "content": user_input})
24     response = get_chat_completions_function_calling(conversation)
25
26     # Moderation Check for LLM response
27     moderation = moderation_check(response[0]['content'])
28     if moderation == 'Flagged':
29         print("\nAssistant:\nSorry, this message has been flagged. Please
restart your conversation.\n")
30         break
31
32     print("\nAssistant:\n", response[0]['content'], '\n')
33
34     conversation += response

1 #Chatbot API - Dialogue Management System-1
2 dialogue_mgmt_system()
```



User Input:
thank you

Assistant:
You're welcome! If you have any more questions or need further assistance in the future, feel free to reach out. Happy gaming and

User Input:
exit

Assistant:
Thank you for using this service. Please do not hesitate to contact us if you need assistance. See you soon!

```
1 #Chatbot API - Dialogue Management System-2
2 dialogue_mgmt_system()
```

↩ Assistant:
How may I help you with your laptop selection? To exit the conversation type 'EXIT'.

User Input:
help me with a trip to goa for 4 days

Assistant:
I'm sorry, but I'm a laptop expert and can only assist you with laptop-related queries. If you have any questions about laptops or

User Input:
exit

Assistant:
Thank you for using this service. Please do not hesitate to contact us if you need assistance. See you soon!

```
1 #Chatbot API - Dialogue Management System-3
2 dialogue_mgmt_system()
3
```

↩ Assistant:
How may I help you with your laptop selection? To exit the conversation type 'EXIT'.

User Input:
laptop

Assistant:
Hello! I'm a laptop expert and I'm here to help you find the perfect laptop for your needs. Could you please share your requirements

User Input:
for coding

Assistant:
Great! As a coder, you'll likely need a laptop that can handle demanding tasks. Could you please specify the importance of GPU inte

User Input:
High GPU, Good Display, Medium Portability, High Processing Speed, Budget < 60000

Assistant:
Thank you for sharing that information. However, I need to inform you that the budget you mentioned is below the minimum requireme

User Input:
24999

Assistant:
It seems that there are no laptops available that meet your specified criteria within your budget of 24,999 INR.

Would you like to adjust any of your requirements, such as increasing your budget or modifying the importance of any parameters?

User Input:
25000

Assistant:
Here is a laptop that matches your criteria:

1. ****Lenovo IdeaPad****: Intel Core i3 processor (2.1 GHz), 8GB RAM, 15.6" TN display (1366x768), Intel UHD **graphics**, Windows 10, Dolt

This laptop offers a balance of performance and affordability, making it suitable for coding and everyday tasks.

If you have any further questions or need more options, feel free to ask!

User Input:
exit

Assistant:
Thank you for using this service. Please do not hesitate to contact us if you need assistance. See you soon!

```
1 #Chatbot API - Dialogue Management System-4
2 dialogue_mgmt_system()
```

↻ Assistant:
How may I help you with your laptop selection? To exit the conversation type 'EXIT'.

User Input:
help me convert 1040 USD to INR

Assistant:
I'm sorry, but I can only assist you with laptop-related queries. If you have any questions about laptops or need help finding one,

User Input:

Assistant:
It seems like you didn't provide any input. If you have any questions or need assistance with finding a laptop, please let me know

User Input:
exit

Assistant:
Thank you for using this service. Please do not hesitate to contact us if you need assistance. See you soon!

```
1 #Chatbot API - Dialogue Management System-5
2 dialogue_mgmt_system()
```

↻ Assistant:
How may I help you with your laptop selection? To exit the conversation type 'EXIT'.

User Input:
how to exit

Assistant:
I'm here to assist you with laptop recommendations. If you have any questions or need help finding a laptop, feel free to ask!

User Input:
how will you help me

Assistant:
I can help you find the best laptop based on your specific needs and preferences. To do that, I'll ask you a few questions about

User Input:
how will you search laptop for me

Assistant:
I will ask you a series of questions to understand your requirements for a laptop, including factors like GPU intensity, display

Shall we begin? What specific tasks do you plan to use the laptop for?

User Input:
sure

Assistant:
Great! To start, could you please tell me what specific tasks or activities you plan to use the laptop for? This will help me und

User Input:
office work

Assistant:
Thank you for sharing that! For office work, it's important to have a reliable laptop.

Next, do you prefer a laptop with a good quality display for presentations and document work? How important is display quality to

User Input:
medium

Assistant:
Got it! You prefer a medium display quality.

Now, do you need a laptop that is easy to carry around, or do you primarily use it at a fixed location? How important is portabili

User Input:
yes

Assistant:
Thank you for that! So, portability is important to you.

Next, do you often have multiple tasks running on your laptop simultaneously, or do you primarily focus on one task at a time? How

User Input:
it's important

Assistant:

```
1 laptop df.head()
```



	Brand	Model Name	Core	CPU Manufacturer	Clock Speed	RAM Size	Storage Type	Display Type	Display Size	Graphics Processor	Screen Resolution	OS	Laptop Weight	Special Features
0	Dell	Inspiron	i5	Intel	2.4 GHz	8GB	SSD	LCD	15.6"	Intel UHD	1920x1080	Windows 10	2.5 kg	Backlit Keyboard
1	MSI	GL65	i7	Intel	2.6 GHz	16GB	HDD+SSD	IPS	15.6"	NVIDIA GTX	1920x1080	Windows 10	2.3 kg	RGB Keyboard

Next steps:

[Generate code with laptop_df](#)[View recommended plots](#)[New interactive sheet](#)

```
1 ## Save this updated csv file so that it can used next time and we do not need
  to perform this again until we have new details.
2 laptop_df.to_csv(filepath + "laptop_dataset_with_features.csv",index=False,
  header = True)
```

```
1 Start coding or generate with AI.
```