**Laptop Recommendation System Documentation**

**Overview**

This code provides a conversational interface to help users find suitable laptops based on their requirements and preferences. Using OpenAI’s API, the system extracts laptop features, recommends models within a budget, and interacts with users to refine the recommendations.

**Requirements**

* **Python Libraries**:
  + openai - for interacting with the OpenAI API.
  + pandas - for data manipulation and processing.
* **Data**:
  + A CSV file named laptop\_data.csv containing laptop specifications, including a Description and Price column.

**Setup**

1. **Install the necessary libraries**:

bash

pip install openai pandas

1. **Set up the OpenAI API Key**:
   * Ensure you have an OpenAI API key.
   * Replace "your\_openai\_api\_key" in openai.api\_key = "your\_openai\_api\_key" with your actual API key.
2. **Data Preparation**:
   * Place laptop\_data.csv in the same directory as the script.
   * Ensure the file contains the columns:
     + Description: A textual description of the laptop’s specifications.
     + Price: The price of each laptop as a numeric value or as a string formatted with commas (e.g., "40,000").

**Code Structure**

**1. Data Loading and Preprocessing**

python

laptop\_df = pd.read\_csv("/content/laptop\_data.csv")

laptop\_df['Price'] = laptop\_df['Price'].str.replace(',', '').astype(int)

* **Purpose**: Loads the laptop data, removes any commas from the Price column, and converts it to an integer format for easier budget-based filtering.

**2. Feature Extraction Function**

python

def extract\_laptop\_features(laptop\_description):

...

return response.choices[0].message.content

* **Purpose**: Extracts and classifies laptop features based on a description.
* **Parameters**:
  + laptop\_description (str): Textual description of a laptop's specifications.
* **Output**: Returns extracted feature categories like GPU intensity, display quality, etc., as low, medium, or high.
* **Example Usage**:

python

features = extract\_laptop\_features("Dell Inspiron with Intel Core i5, 8GB RAM, and 1920x1080 resolution")

**3. Recommendation Processing Function**

python

def recommend\_laptops(user\_requirements):

...

return response.choices[0].message.content

* **Purpose**: Filters and ranks laptops based on user requirements within a specified budget.
* **Parameters**:
  + user\_requirements (dict): A dictionary containing user preferences, e.g., Budget, GPU intensity, etc.
* **Output**: A list of recommended laptops.
* **Example Usage**:

python

recommendations = recommend\_laptops({"Budget": "40000", "GPU intensity": "medium"})

**4. Conversation Flow Function**

python

def conversation\_flow(user\_input, user\_profile=None):

...

return response.choices[0].message.content

* **Purpose**: Provides a conversational response based on user input and profile, refining recommendations interactively.
* **Parameters**:
  + user\_input (str): The user’s question or statement.
  + user\_profile (dict, optional): A dictionary of user preferences for personalizing responses.
* **Output**: A conversational response tailored to the user’s profile and input.
* **Example Usage**:

python

user\_input = "Which laptop is ideal for travel?"

conversation\_response = conversation\_flow(user\_input, {"GPU intensity": "medium"})

**5. Sample Workflow**

The workflow demonstrates how to:

1. Extract features for each laptop in the data.
2. Get laptop recommendations based on specific user requirements.
3. Interact with the user through natural conversation.

**Code Execution**

1. **Extract Features**:

python

for index, row in laptop\_df.iterrows():

features = extract\_laptop\_features(row['Description'])

print(f"Laptop {index + 1} Extracted Features:", features)

1. **Get Recommendations**:

python

user\_requirements = {"Budget": "40000", "GPU intensity": "medium"}

recommendations = recommend\_laptops(user\_requirements)

print("Recommendations:", recommendations)

1. **Simulate a Conversation**:

python

user\_input = "Which laptop is ideal for travel?"

conversation\_response = conversation\_flow(user\_input, {"GPU intensity": "medium"})

print("Conversation Response:", conversation\_response)

**Error Handling and Tips**

* **API Rate Limits**: Be aware of API limits. Use a delay in loops if necessary to prevent hitting the limit.
* **Invalid API Key**: If the API key is incorrect or missing, the code will raise an authentication error. Ensure the API key is securely stored.
* **Data Format**: Ensure laptop\_data.csv matches the expected format to avoid KeyErrors.

**Potential Enhancements**

* **Add more User Profile fields**: Capture other user preferences like battery life or operating system.
* **Interactive CLI or Web Interface**: Build a simple UI for ease of interaction.
* **Database Integration**: Store laptops in a database for faster, more scalable querying.