# GOVERNMENT COLLEGE OF TECHNOLOGY, COIMBATORE

TEAM MEMBERS: JANANI G(71772117117), VISHAL N (71772117149)

DEPARTMENT: B.E. COMPUTER SCIENCE ENGINEERING

SUBJECT: ARTIFICIAL INTELLIGENCE

COURSE CODE: 18SPC702

PROJECT TITLE: AI-ENABLED EXCEL ANALYTICS ASSISTANT

ASSIGNMENT NO: 02

## PROJECT OVERVIEW

The AI-Enabled Excel Analytics Assistant is an interactive tool designed to support users in retrieving insights from their Excel data files, bridging advanced AI capabilities with everyday data analysis needs. By utilizing Google’s generative AI models in combination with natural language processing (NLP), this system interprets and answers user queries based on the contents of their Excel files.  
This application is developed in Python, utilizing the google.generativeai API as the engine for AI responses and Streamlit for a simple, engaging interface that facilitates file uploads, question inputs, and displays answers in real time.

## Project Approach

**1. Data Handling and Preparation:**

Upon file upload, the assistant processes the Excel file using the pandas library, structuring the data into a format compatible with the AI model. The contents are converted to a readable text format that is fed to the generative model.

**2. Query Processing and Response Generation:**

After the data is loaded, the system allows users to input questions, which are then combined with the data context and sent to the AI model. The model interprets and responds to the user’s questions based on the data, delivering an intuitive answer.

**3. Streamlit-based User Interface:**

Through Streamlit, the application provides an easy-to-navigate interface for file uploads, question submissions, and viewing AI-generated responses, creating an accessible and responsive user experience.

## Code Implementation

```python  
import google.generativeai as genai  
import pandas as pd  
import time  
  
# Configure the API key  
genai.configure(api\_key="AIzaSyCdFkS-IAjpHNUoRIPhFs1jvdQe5P9AmVc")  
  
# Function to load and preview Excel data  
def load\_excel\_data(file\_path):  
 # Load the Excel file into a DataFrame  
 data = pd.read\_excel(file\_path)  
 # Convert the entire DataFrame to a readable string format for the prompt  
 data\_context = data.to\_string(index=False)  
 return data\_context  
  
# Function to send a request to the model  
def send\_request(chat, context, question):  
 try:  
 # Combine context and question into a single prompt  
 prompt = f"Here is the data:\n\n{context}\n\nQuestion: {question}"  
 response = chat.send\_message(prompt)  
 return response  
 except Exception as e:  
 print("Error occurred:", e)  
 time.sleep(5) # Wait before retrying  
 return None  
  
# Main function to load data and interact with multiple questions  
def interactive\_ask\_model\_with\_excel(file\_path):  
 # Load Excel data and format it  
 data\_context = load\_excel\_data(file\_path)  
   
 # Start a chat with the model  
 model = genai.GenerativeModel(model\_name="gemini-1.5-pro")  
 chat = model.start\_chat()  
   
 print("Interactive mode: Ask your questions about the Excel data. Type 'exit' to stop.")  
  
 while True:  
 # Prompt the user to enter a question  
 question = input("\nEnter your question: ")  
   
 # Check if the user wants to exit  
 if question.lower() == 'exit':  
 print("Ending the session.")  
 break  
   
 # Send the request with the data context and question  
 response = send\_request(chat, data\_context, question)  
   
 if response:  
 # Extract and print the answer from the model's response  
 answer = response.candidates[0].content.parts[0].text  
 print("\nAnswer:", answer)  
 else:  
 print("No response received. Try again.")  
  
# Example usage  
file\_path = "/content/Walmart -GCT - Phase 2 Sandbox login credentials. 3rd Oct.xlsx" # Replace with the path to your Excel file  
interactive\_ask\_model\_with\_excel(file\_path)  
```

## Challenges Encountered

**1. Handling Large Files:**

Managing large datasets efficiently proved challenging as it impacted processing speed. Performance optimizations were implemented to enable smooth operation with larger files.

**2. Contextual Understanding:**

Accurately interpreting complex or vague questions was challenging, and adjustments to prompt wording and model settings were required to improve answer relevance.

**3. API Constraints:**

API limitations in handling response length and complexity occasionally impacted the clarity of output, leading to adjustments for more concise responses.

**4. User Experience Optimization:**

Ensuring smooth user interactions, especially with large files, necessitated focusing on error handling and optimizing data upload and parsing processes.

## Technologies Employed

• **Google Generative AI:**

Provides the main intelligence for understanding and answering questions based on the file data.

**• Pandas:**

Essential for loading, processing, and transforming Excel data files into a format compatible with the AI model.

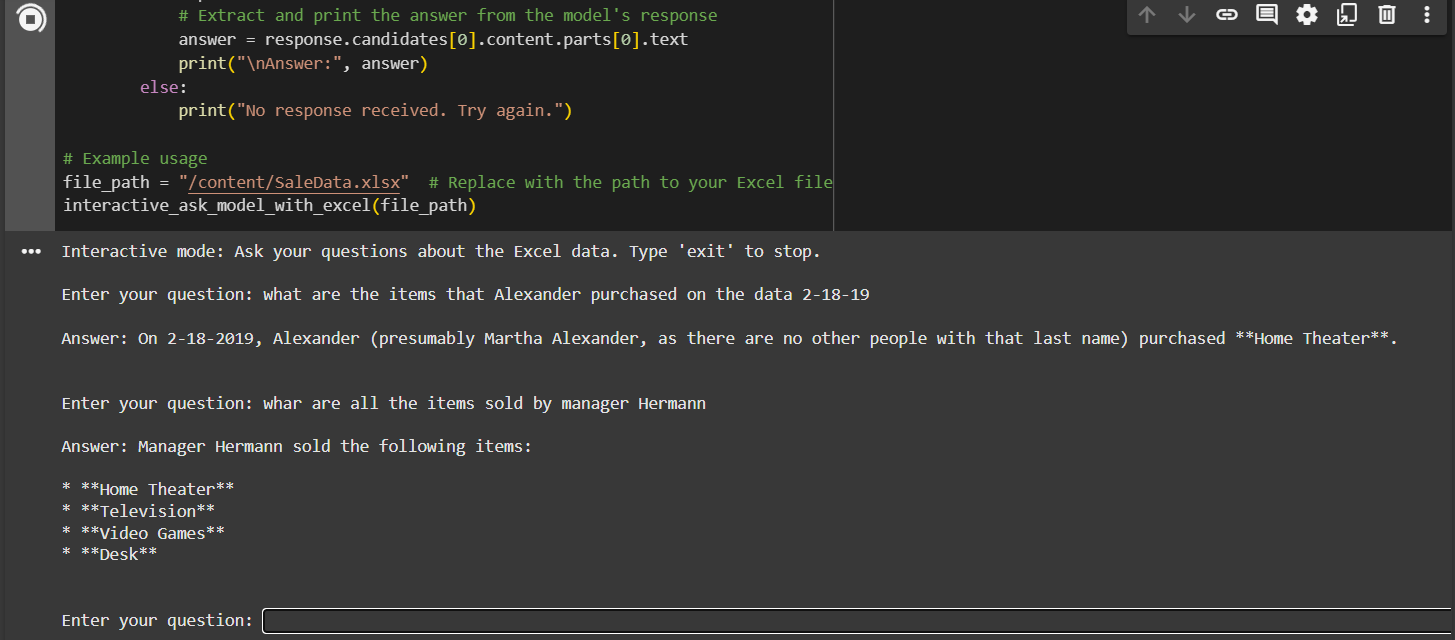
• **Streamlit:**

Powers the application interface, creating a seamless experience for file upload, question input, and answer retrieval.

• **Error Management and Time Handling:**

Utilized to manage potential issues like API timeouts and data processing errors, ensuring robust system performance.

## Sample Output



## Conclusion

The AI-Enabled Excel Analytics Assistant merges NLP techniques with an accessible interface to facilitate Excel data exploration. Through Google’s generative AI, users can query their data and obtain relevant insights, simplifying data interaction and enhancing accessibility. This project illustrates the application of generative AI to improve the user experience in data-centric tasks, showcasing how technology can make data analysis more user-friendly.