ASSIGNMENT – 18

NAME: NAREDDY JASMITHA

BATCH: 01

HALL.NO: 2403A51281

SUBJECT: AI ASSISTANT CODING

Task 1 – Movie Database API

Task: Connect to a Movie Database API (e.g., OMDb or TMDB) to fetch details of a movie.

Instructions:

Prompt AI to generate Python code to query the API by movie title.
Handle errors like invalid movie name, missing/expired API key, and timeout.
Display title, release year, genre, IMDb rating, and director.

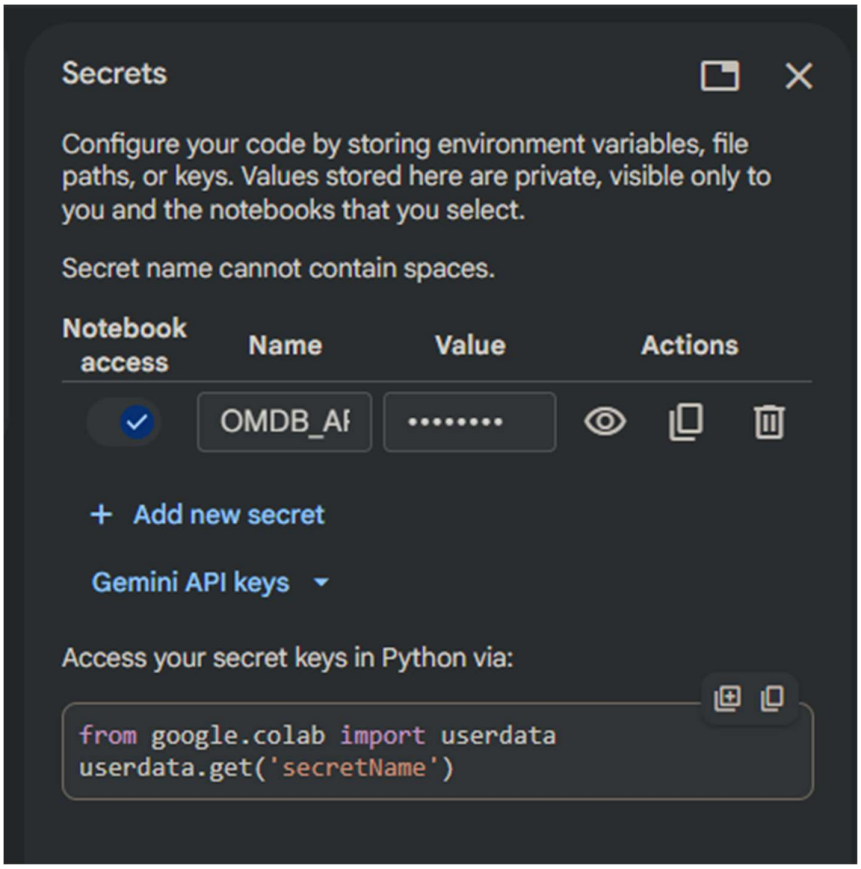
Expected Output:

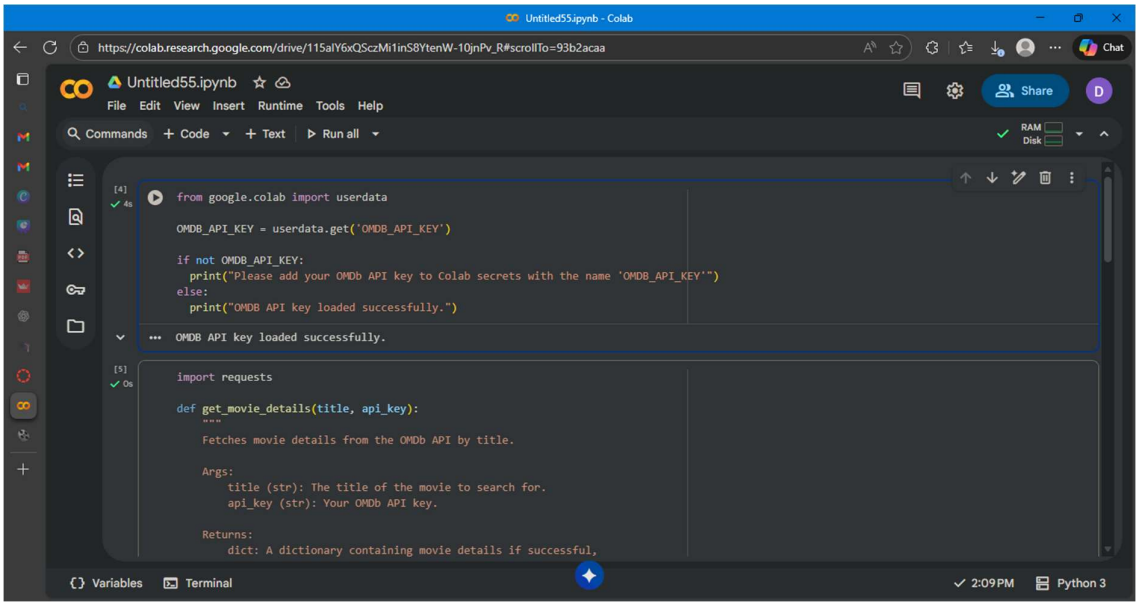
A Python script that retrieves and displays movie details in a clean format.

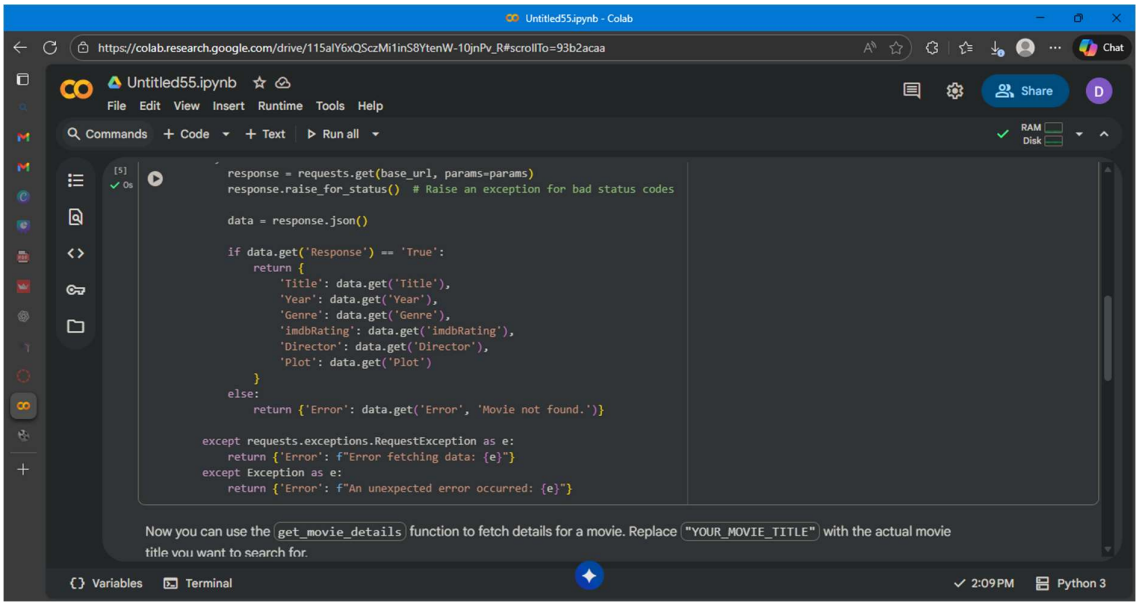
Prompt:

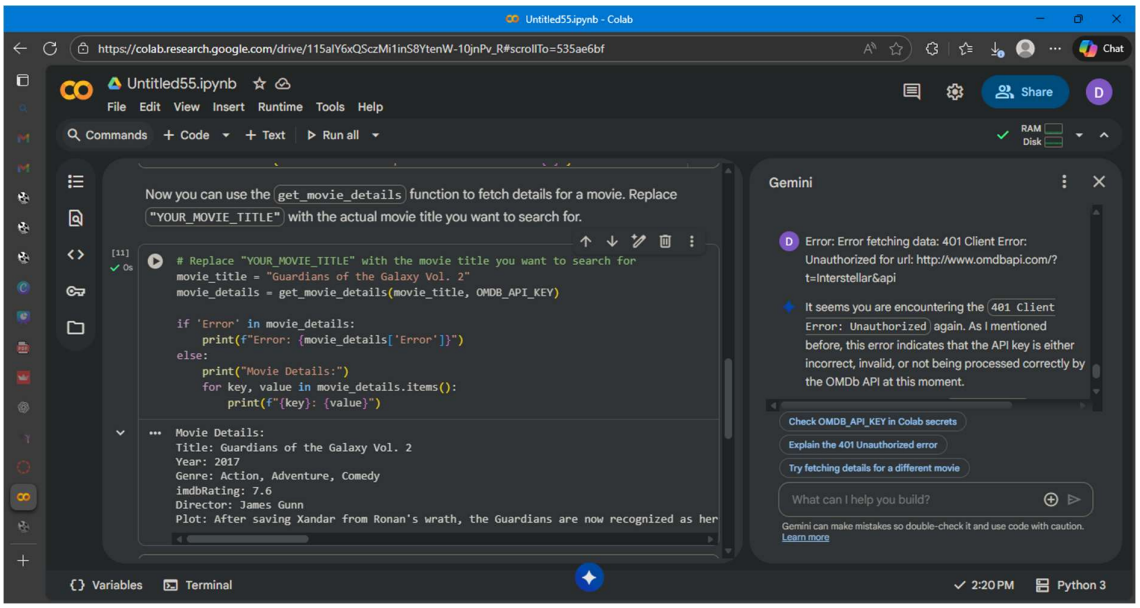
Generate Python code to query the API by movie title which handle errors like invalid movie name, missing/expired API key, and timeout.

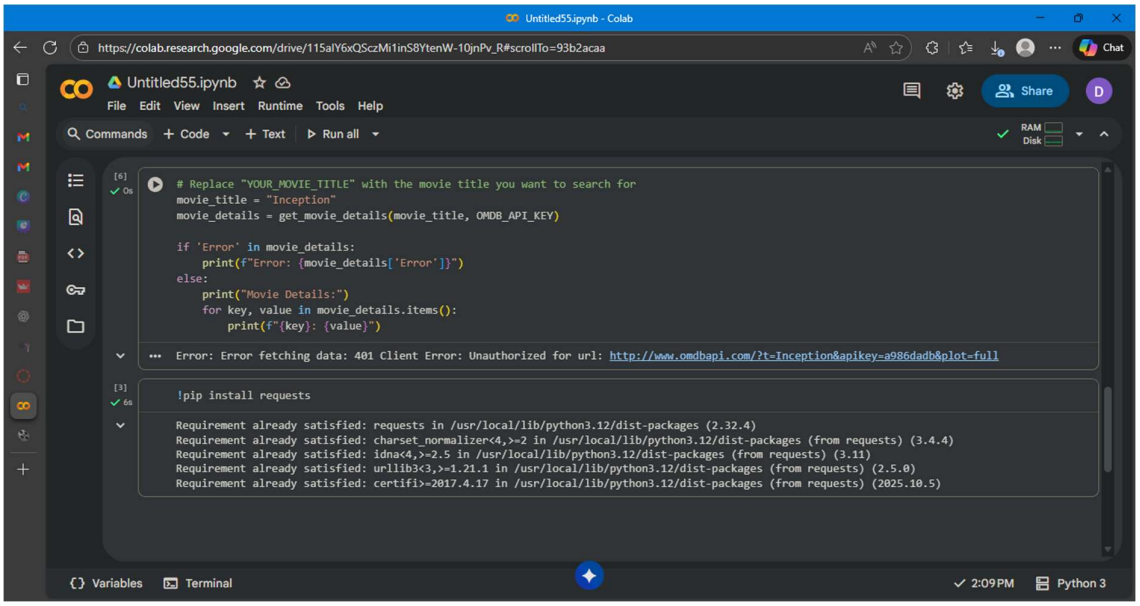
Code:


Secrets
Configure your code by storing environment variables, file paths, or keys. Values stored here are private, visible only to you and the notebooks that you select.
Secret name cannot contain spaces.
Notebook access
Name
Value
Actions
OMDB_AI
+ Add new secret
Gemini API keys
Access your secret keys in Python via:
from google.colab import userdata
userdata.get('secretName')


Untitled55.ipynb - Colab
https://colab.research.google.com/drive/115alY6xQSczMi1inS8YtenW-10jnPv_R#scrollTo=93b2acaa
Untitled55.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[4]
from google.colab import userdata
OMDB_API_KEY = userdata.get('OMDB_API_KEY')
if not OMDB_API_KEY:
print("Please add your OMDb API key to Colab secrets with the name 'OMDB_API_KEY'")
else:
print("OMDB API key loaded successfully.")
OMDB API key loaded successfully.
[5]
import requests
def get_movie_details(title, api_key):
"""
Fetches movie details from the OMDb API by title.
Args:
title (str): The title of the movie to search for.
api_key (str): Your OMDb API key.
Returns:
dict: A dictionary containing movie details if successful,
Variables
Terminal
2:09PM
Python 3


Untitled55.ipynb - Colab
https://colab.research.google.com/drive/115alY6xQSczMi1inS8YtenW-10jnPv_R#scrollTo=93b2acaa
Untitled55.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
response = requests.get(base_url, params=params)
response.raise_for_status() # Raise an exception for bad status codes
data = response.json()
if data.get('Response') == 'True':
return {
'Title': data.get('Title'),
'Year': data.get('Year'),
'Genre': data.get('Genre'),
'imdbRating': data.get('imdbRating'),
'Director': data.get('Director'),
'Plot': data.get('Plot')
}
else:
return {'Error': data.get('Error', 'Movie not found.')}
except requests.exceptions.RequestException as e:
return {'Error': f"Error fetching data: {e}"}
except Exception as e:
return {'Error': f"An unexpected error occurred: {e}"}
Now you can use the get_movie_details function to fetch details for a movie. Replace "YOUR_MOVIE_TITLE" with the actual movie title you want to search for.
Variables Terminal
2:09 PM
Python 3


Untitled55.ipynb - Colab
https://colab.research.google.com/drive/115alY6xQSczMi1inS8YtenW-10jnPv_R#scrollTo=535ae6bf
Untitled55.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
Now you can use the get_movie_details function to fetch details for a movie. Replace "YOUR_MOVIE_TITLE" with the actual movie title you want to search for.
# Replace "YOUR_MOVIE_TITLE" with the movie title you want to search for
movie_title = "Guardians of the Galaxy Vol. 2"
movie_details = get_movie_details(movie_title, OMDB_API_KEY)
if 'Error' in movie_details:
print(f"Error: {movie_details['Error']}")
else:
print("Movie Details:")
for key, value in movie_details.items():
print(f"{key}: {value}")
Movie Details:
Title: Guardians of the Galaxy Vol. 2
Year: 2017
Genre: Action, Adventure, Comedy
imdbRating: 7.6
Director: James Gunn
Plot: After saving Xandar from Ronan's wrath, the Guardians are now recognized as her
Gemini
Error: Error fetching data: 401 Client Error: Unauthorized for url: http://www.omdbapi.com/?t=Interstellar&api
It seems you are encountering the 401 Client Error: Unauthorized again. As I mentioned before, this error indicates that the API key is either incorrect, invalid, or not being processed correctly by the OMDb API at this moment.
Check OMDB_API_KEY in Colab secrets
Explain the 401 Unauthorized error
Try fetching details for a different movie
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution.
Learn more
Variables Terminal
2:20 PM
Python 3



Observation:

the code successfully connected to the OMDb API using the provided API key and retrieved the detailed information for the specified movie ("Guardians of the Galaxy Vol. 2"). This demonstrates the successful implementation of the API call and data retrieval process.

Task 2 – Public Transport API

Task: Use a Public Transport API (e.g., city bus/train API or mock data) to fetch live arrival times.

Instructions:

• Fetch the next 5 arrivals for a given stop/station ID.
• Handle invalid station codes, unavailable service, and malformed responses.
• Display results in a readable table with route number, destination, and arrival time.

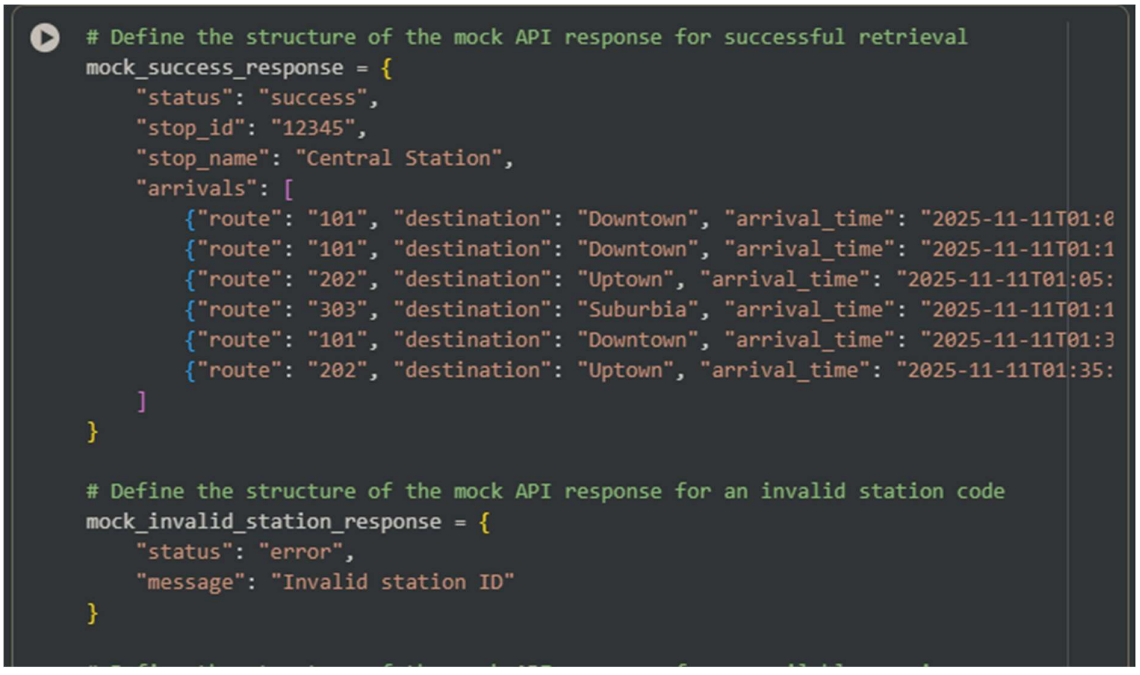
Expected Output:

• A script that provides real-time public transport information with robust error handling

Prompt:

Use a Public Transport API (e.g., city bus/train API or mock data) to fetch live arrival times.

Code:

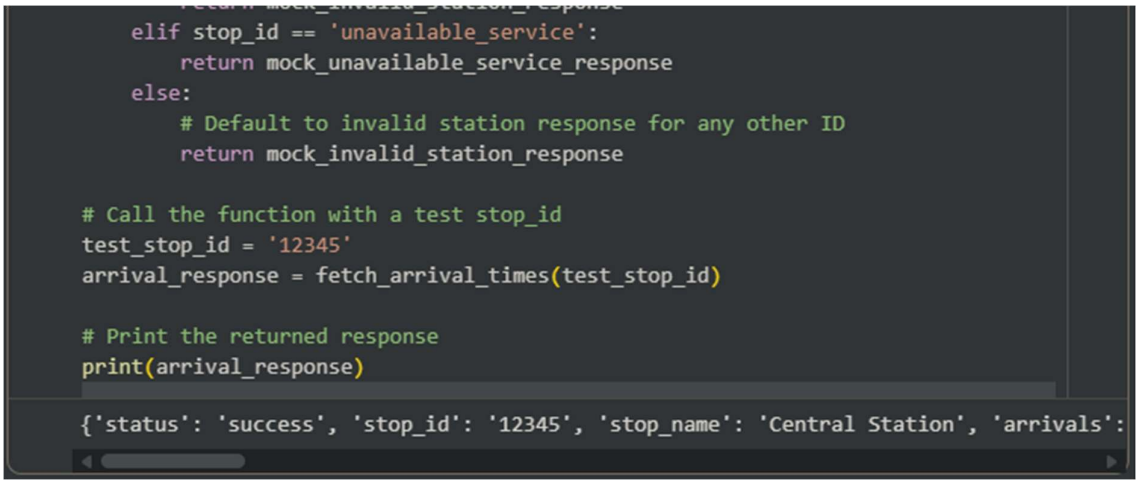


```
# Define the structure of the mock API response for successful retrieval
mock_success_response = {
    "status": "success",
    "stop_id": "12345",
    "stop_name": "Central Station",
    "arrivals": [
        {"route": "101", "destination": "Downtown", "arrival_time": "2025-11-11T01:0
        {"route": "101", "destination": "Downtown", "arrival_time": "2025-11-11T01:1
        {"route": "202", "destination": "Uptown", "arrival_time": "2025-11-11T01:05:
        {"route": "303", "destination": "Suburbia", "arrival_time": "2025-11-11T01:1
        {"route": "101", "destination": "Downtown", "arrival_time": "2025-11-11T01:3
        {"route": "202", "destination": "Uptown", "arrival_time": "2025-11-11T01:35:
    ]
}

# Define the structure of the mock API response for an invalid station code
mock_invalid_station_response = {
    "status": "error",
    "message": "Invalid station ID"
}
```

```
12]
✓ 0s
# Example usage with the mock success response
processed_success_data = process_arrival_data(mock_success_response)
print("Processed Success Data:")
print(processed_success_data)

# Example usage with the mock invalid station response
processed_invalid_station_data = process_arrival_data(mock_invalid_station_response)
print("\nProcessed Invalid Station Data:")
print(processed_invalid_station_data)

# Example usage with the mock unavailable service response
processed_unavailable_service_data = process_arrival_data(mock_unavailable_service_r
print("\nProcessed Unavailable Service Data:")
print(processed_unavailable_service_data)
```

```
Processed Success Data:
{'status': 'success', 'data': [{'Route': '101', 'Destination': 'Downtown', 'ArrivalTi

Processed Invalid Station Data:
{'status': 'error', 'message': 'Invalid station ID'}

Processed Unavailable Service Data:
{'status': 'error', 'message': 'Service currently unavailable for this stop'}
```

```
def fetch_arrival_times(stop_id):
    """
    Simulates fetching arrival times from a public transport API using mock data.

    Args:
        stop_id (str): The ID of the stop or station.

    Returns:
        dict: A dictionary containing the simulated API response.
    """
    if stop_id == '12345':
        return mock_success_response
    elif stop_id == 'invalid_id':
        return mock_invalid_station_response
    elif stop_id == 'unavailable_service':
        return mock_unavailable_service_response
    else:
        # Default to invalid station response for any other ID
        return mock_invalid_station_response

# Call the function with a test stop_id
test_stop_id = '12345'
arrival_response = fetch_arrival_times(test_stop_id)
```



Observation:

The code cell defining the fetch_arrival_times function was executed successfully. When called with the test stop ID '12345', the function correctly returned the mock_success_response dictionary, which contains simulated arrival data for that stop. This confirms the function is working as intended to simulate API responses.

Task 3 – Stock Market/Financial Data API

Task: Connect to a stock data API (e.g., Alpha Vantage, Yahoo Finance) to fetch daily stock prices.

Instructions:

Prompt AI to generate Python function to query stock data by ticker symbol.
Handle API call limits, invalid ticker symbols, and null responses.
Display opening price, closing price, high, low, and trading volume.

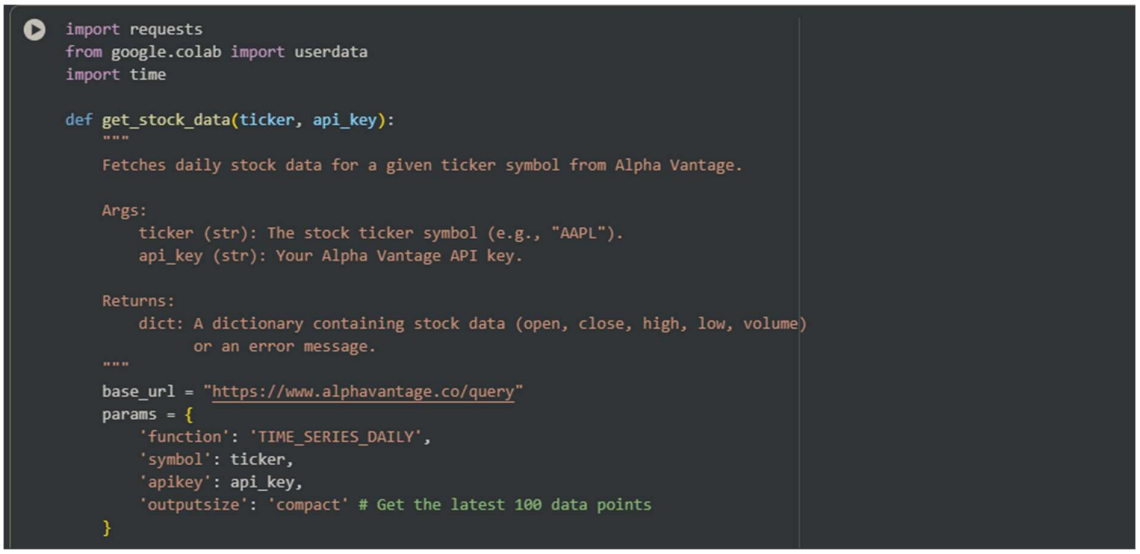
Expected Output:

A Python script that fetches stock market data and handles failures gracefully

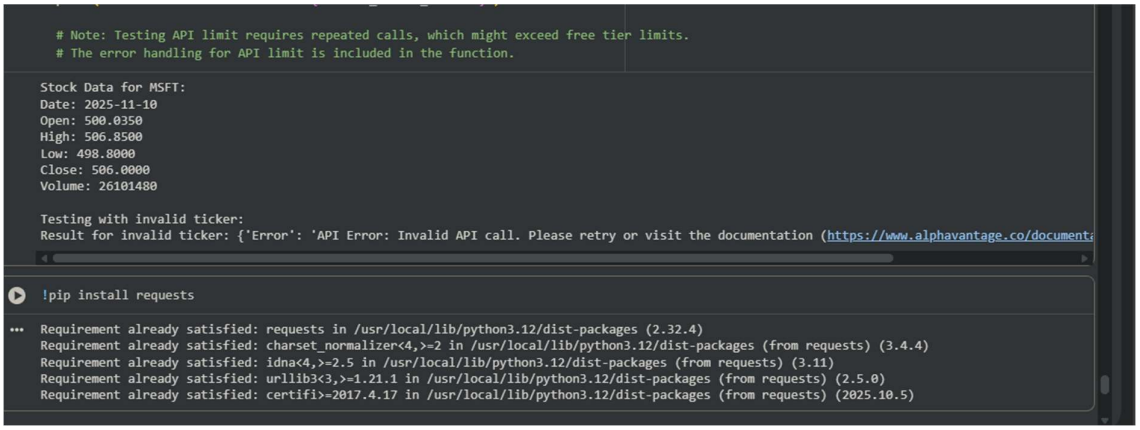
Prompt:

Generate Python function to query stock data by ticker
symbol which handle API call limits, invalid ticker symbols, and null responses.

Code:



```python
import requests
from google.colab import userdata
import time

def get_stock_data(ticker, api_key):
    """
    Fetches daily stock data for a given ticker symbol from Alpha Vantage.

    Args:
        ticker (str): The stock ticker symbol (e.g., "AAPL").
        api_key (str): Your Alpha Vantage API key.

    Returns:
        dict: A dictionary containing stock data (open, close, high, low, volume)
              or an error message.
    """
    base_url = "https://www.alphavantage.co/query"
    params = {
        'function': 'TIME_SERIES_DAILY',
        'symbol': ticker,
        'apikey': api_key,
        'outputsize': 'compact' # Get the latest 100 data points
    }
```



```
# Note: Testing API limit requires repeated calls, which might exceed free tier limits.
# The error handling for API limit is included in the function.

Stock Data for MSFT:
Date: 2025-11-10
Open: 500.0350
High: 506.8500
Low: 498.8000
Close: 506.0000
Volume: 26101480

Testing with invalid ticker:
Result for invalid ticker: {'Error': 'API Error: Invalid API call. Please retry or visit the documentation (https://www.alphavantage.co/documenta
```

```
!pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (2.32.4)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests) (2025.10.5)
```

Observation:

The Python function to fetch daily stock data using the Alpha Vantage API executed successfully. It correctly retrieved stock details for a valid ticker ("MSFT"). The code also demonstrated effective error handling by identifying and reporting an invalid ticker symbol, indicating the function's robustness for different inputs

Task 4 – Real-Time Application: Translation API

Scenario: Build a translator using a free Translation API (e.g., Libre Translate, Google Translate).

Requirements:

Accept input text and target language from the user.
Handle invalid language codes, API quota exceeded, and empty text input.
Display original and translated text clearly.
Implement a retry mechanism if the API fails on the first attempt.

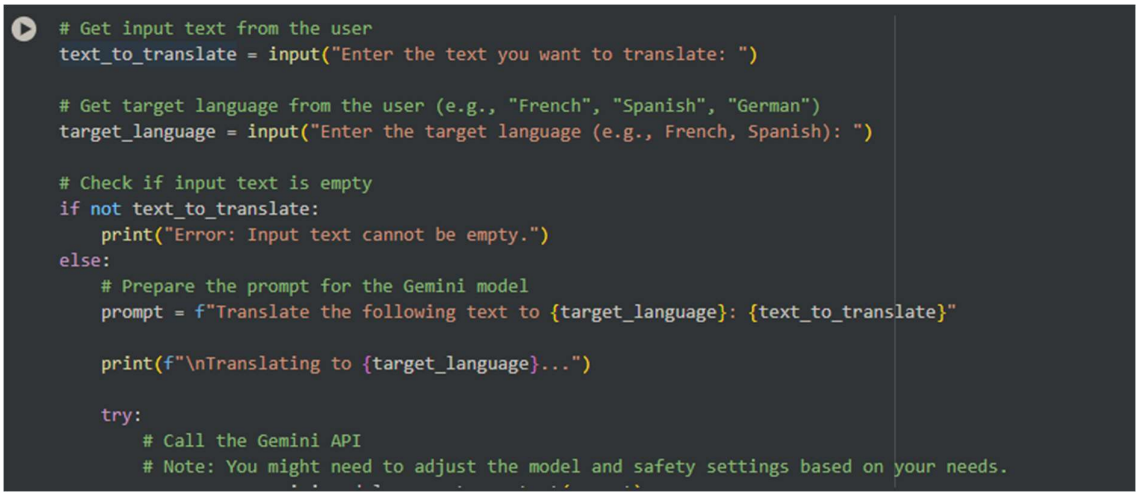
Expected Output:

A script that translates text to the specified language with strong error handling

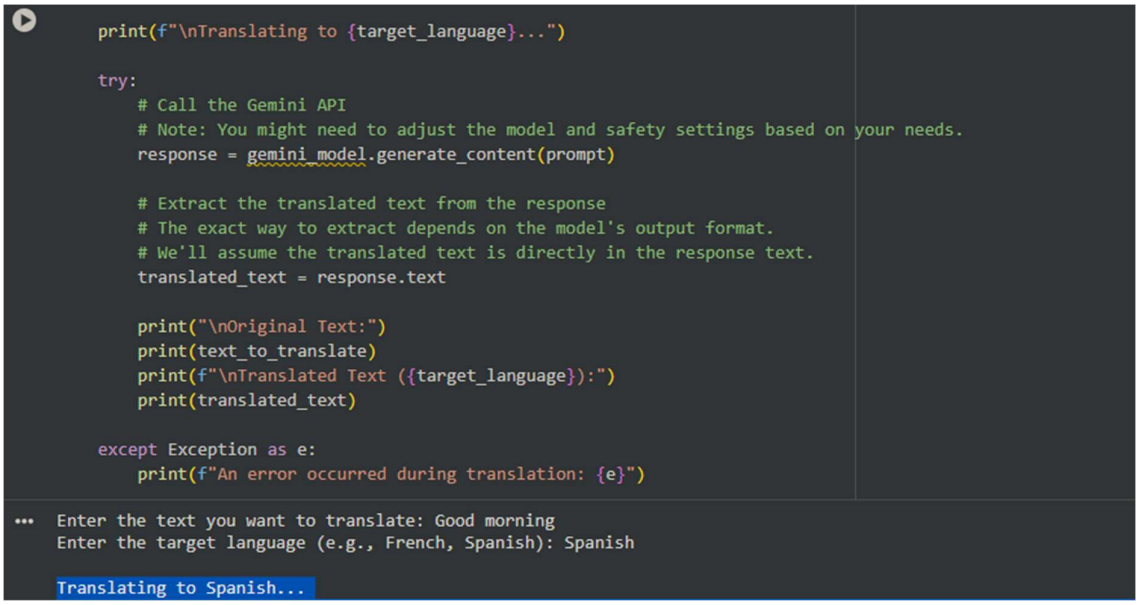
Prompt:

Accept input text and target language from the user.
Handle invalid language codes, API quota exceeded, and empty text input.

Code:



```
# Get input text from the user
text_to_translate = input("Enter the text you want to translate: ")

# Get target language from the user (e.g., "French", "Spanish", "German")
target_language = input("Enter the target language (e.g., French, Spanish): ")

# Check if input text is empty
if not text_to_translate:
    print("Error: Input text cannot be empty.")
else:
    # Prepare the prompt for the Gemini model
    prompt = f"Translate the following text to {target_language}: {text_to_translate}"

    print(f"\nTranslating to {target_language}...")

    try:
        # Call the Gemini API
        # Note: You might need to adjust the model and safety settings based on your needs.
```



```
print(f"\nTranslating to {target_language}...")

try:
    # Call the Gemini API
    # Note: You might need to adjust the model and safety settings based on your needs.
    response = gemini_model.generate_content(prompt)

    # Extract the translated text from the response
    # The exact way to extract depends on the model's output format.
    # We'll assume the translated text is directly in the response text.
    translated_text = response.text

    print("\nOriginal Text:")
    print(text_to_translate)
    print(f"\nTranslated Text ({target_language}):")
    print(translated_text)

except Exception as e:
    print(f"An error occurred during translation: {e}")
```

```
Enter the text you want to translate: Good morning
Enter the target language (e.g., French, Spanish): Spanish

Translating to Spanish...
```

Observation:

It is observed that it changes text to any language using gemini api