Project Report

Date of Submission	20/07/2024
Team ID	SWTID1720459891
Project Name	Inquisitive: A Multilingual AI Question Generator Using PaLM's Text-Bison-001

TEAM MEMBERS

Name	Student Mail ID
Siddarth SR Pillai	siddarth.srpillai2021@vitstudent.ac.in
Akshaj Anil	akshaj.anil2021@vitstudent.ac.in
Chirag Mahesh	chirag.mahesh2021@vitstudent.ac.in
Akshay R	akshay.r2022@vitstudent.ac.in

Inquisitive:A Multilingual AI Question Generator Using Palm's Text-Bison-001

Inquisitive is an application that leverages the robust capabilities of the PALM architecture to analyze user input text and autonomously generate questions from it. This multilingual application enables seamless interaction, extracting key information from the text to formulate relevant questions in various languages. It enhances comprehension and engagement through dynamic question generation, facilitating deeper exploration of textual content across linguistic barriers.

Scenario 1: Corporate Training Programs

Enhance corporate training programs by incorporating Inquisitive: The GenAI Question Generator. Employees can input training materials, and the application generates quizzes, fostering active learning and reinforcing key concepts. This automated process saves time for trainers and encourages self-paced learning, leading to improved knowledge retention and skill development.

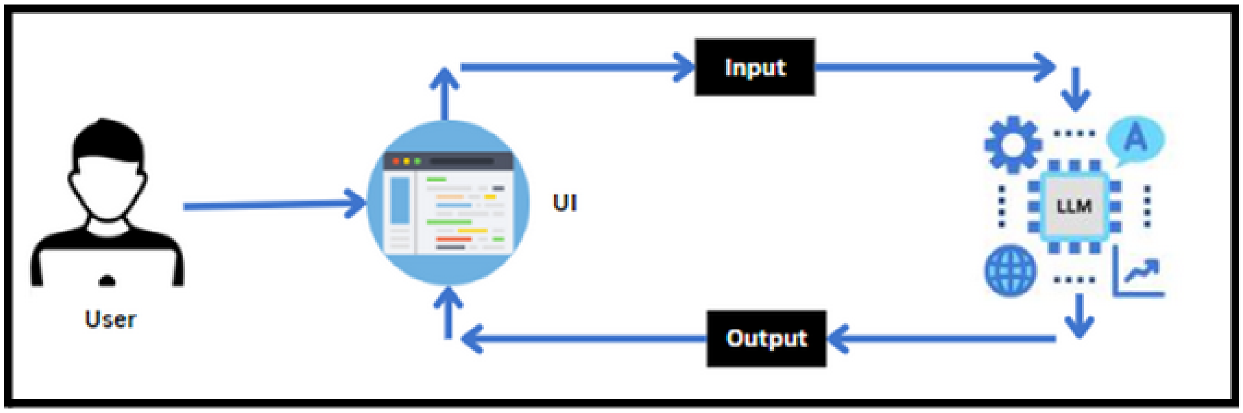
Scenario 2: Content Creation for Marketing

In marketing, utilize Inquisitive to transform product descriptions or promotional content into interactive quizzes or FAQs. This engages customers, encourages interaction with the brand, and provides valuable insights into consumer preferences and understanding. The dynamic nature of generated questions keeps content fresh and engaging, driving customer interest and retention.

Scenario 3: Knowledge Management in Meetings

During corporate meetings, employ Inquisitive to summarize discussions in real-time and generate follow-up questions. This ensures thorough understanding among participants, stimulates critical thinking, and clarifies any ambiguities. By automating question generation, the tool streamlines the meeting process, promotes active engagement, and facilitates deeper exploration of topics, ultimately leading to more productive and insightful discussions.

Technical Architecture



Project Flow

1. Users input text into the UI of Inquisitive.
2. The input text is then processed and analyzed by the PALM architecture, which is integrated into the backend.
3. PALM autonomously generates questions based on the input text.
4. The generated questions are sent back to the frontend for display on the UI.
5. Users can view the dynamically generated questions and interact with them to gain deeper insights into the content.

To accomplish this, we have to complete all the activities listed below,

- **Initializing the PALM**
  - Generate PALM API
  - Initialize the pre-trained model
- **Interfacing with Pre-trained Model**
  - Questions Generator
- **Model Deployment**
  - Deploy the application using Streamlit

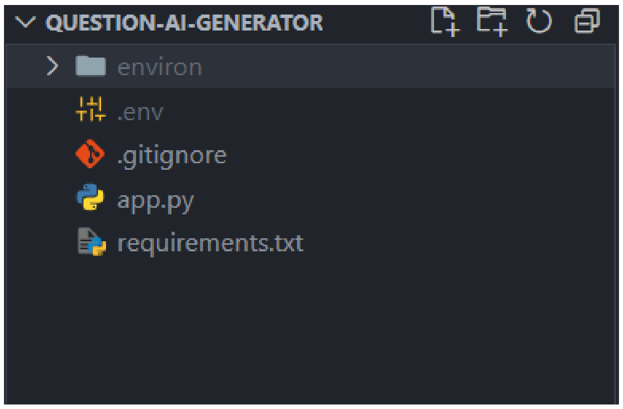
Prior Knowledge

You must have prior knowledge of the following topics to complete this project.

- LLM & PALM: https://cloud.google.com/vertex-ai/docs/generative-ai/learn-resources
- Google Translate API: https://pypi.org/project/googletrans/
- Streamlit: https://www.datacamp.com/tutorial/streamlit

Project Structure

Create the Project folder which contains files as shown below:



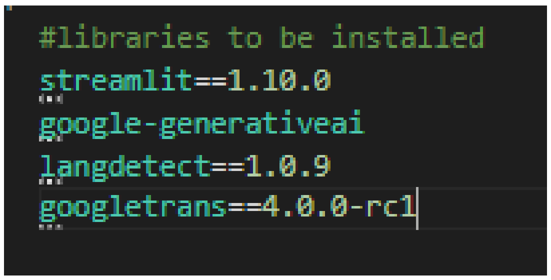
```
QUESTION-AI-GENERATOR
  > environ
    .env
    .gitignore
    app.py
    requirements.txt
```

- app.py: It serves as the primary application file housing both the model and Streamlit UI code.
- requirements.txt: It enumerates the libraries necessary for installation to ensure proper functioning.
- .gitignore : This file is used in Git to specify which files and directories should be ignored and not tracked by version control, helping to keep the repository clean from unnecessary files such as build artifacts, temporary files, and sensitive information.
- .env : This file stores all the environment variables required for your application. In this case, it is the google api key
- environ : This is a virtual environment under which the application is written so as to isolate the libraries installed in the application from the system.

Milestone 1: Requirements Specification

Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment.

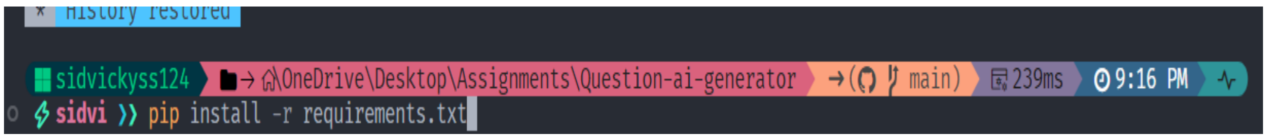
Step 1: Create a requirements.txt file to list the required libraries



```
#libraries to be installed
streamlit==1.10.0
google-generativeai
langdetect==1.0.9
googletrans==4.0.0-rc1
```

- **streamlit** : Streamlit is a powerful framework for building interactive web applications with Python.
- **google-generativeai** : Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
- **langdetect==1.0.9** : Language detection library is used to identify the language of a given text.
- **googletrans==4.0.0-rc1** : Google Translate API wrapper for Python allows translation of text between different languages using Google Translate service.
- **python-dotenv**: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.

Step 2 : Install the Required Libraries

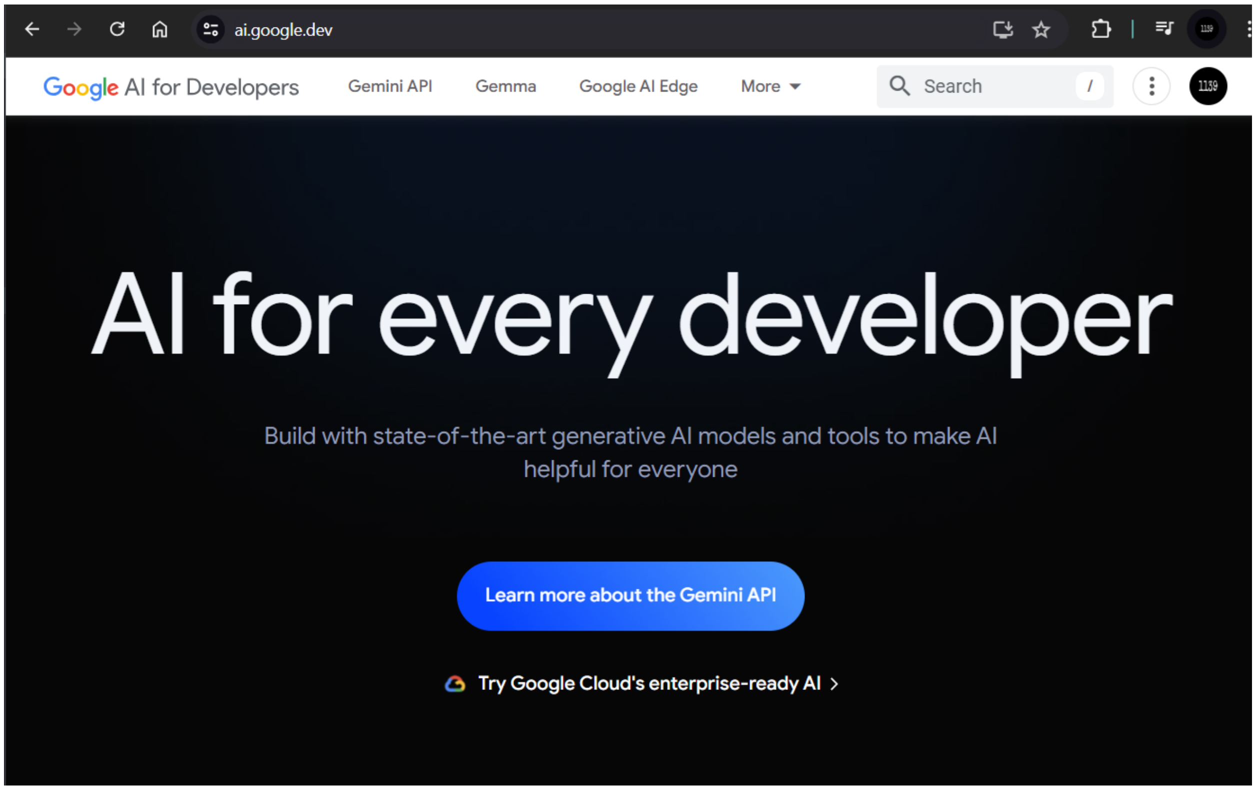


```
× History restored

sidvickyss124 → OneDrive\Desktop\Assignments\Question-ai-generator → ( main) 239ms 9:16 PM
sidvi ❯❯ pip install -r requirements.txt
```

- Open the terminal.
- Run the command: pip install -r requirements.txt
- This command installs all the libraries listed in the requirements.txt file

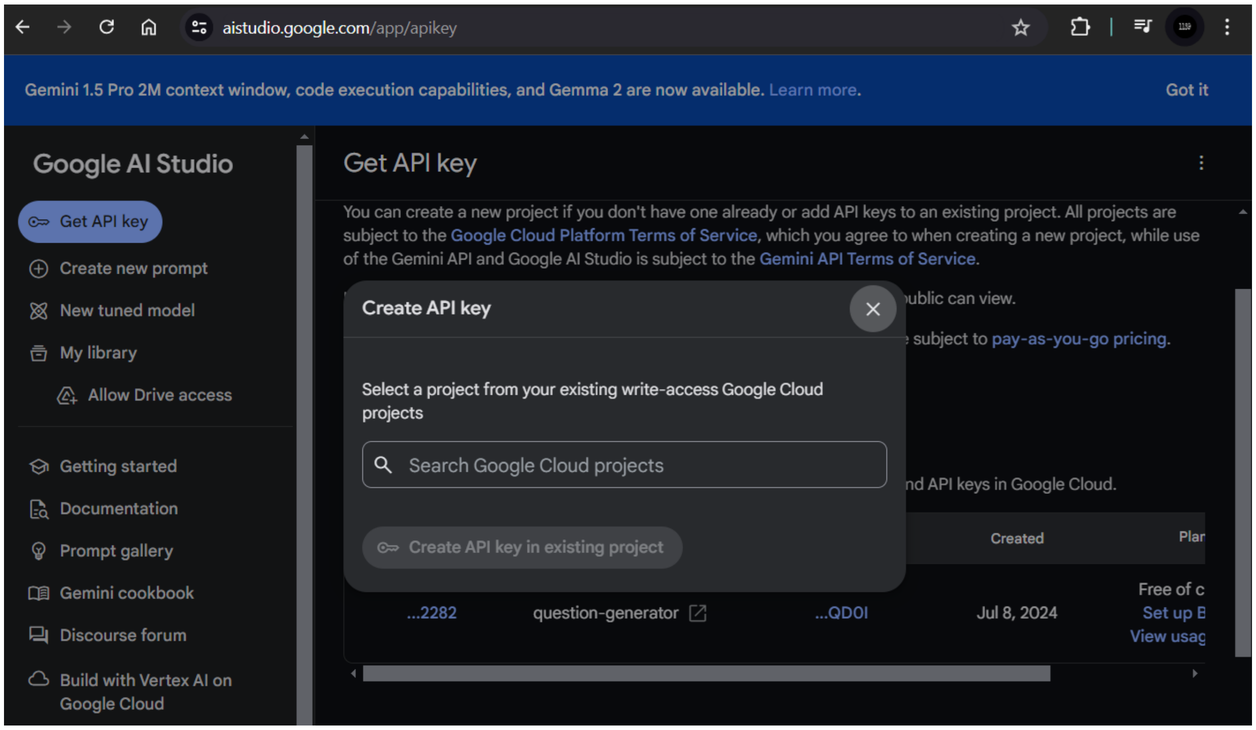
Milestone 2 : Initialization The Model

The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API requests, ensuring that only authorized users can access and utilize Google's services.


ai.google.dev
Google AI for Developers
Gemini API
Gemma
Google AI Edge
More
Search
AI for every developer
Build with state-of-the-art generative AI models and tools to make AI helpful for everyone
Learn more about the Gemini API
Try Google Cloud's enterprise-ready AI

Step 1 : Generate PALM API key

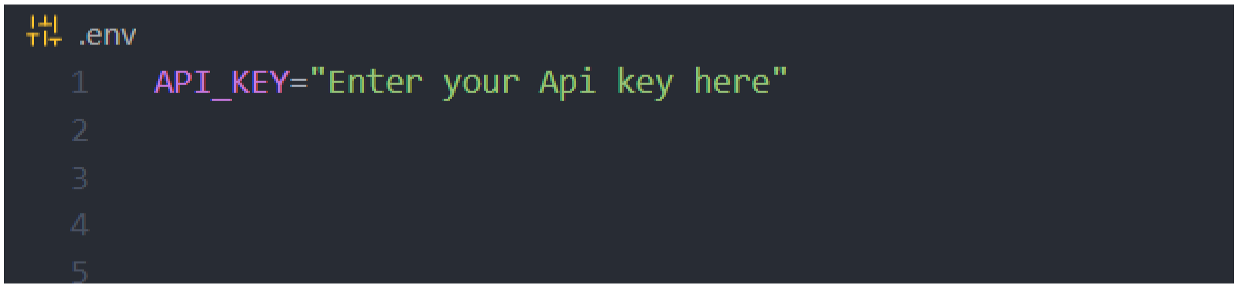
- Click on the link (https://developers.generativeai.google/).
- Then click on “Get API key in Google AI Studio”.
- Click on “Get API key” from the right navigation menu.
- Now click on “Create API key”.
- Copy the API key.



You can create a new project if you don't have one already or add API keys to an existing project. All projects are subject to the Google Cloud Platform Terms of Service, which you agree to when creating a new project, while use of the Gemini API and Google AI Studio is subject to the Gemini API Terms of Service.

Select a project from your existing write-access Google Cloud projects

Step 2 : Initialize Google API Key



```
.env
API_KEY="Enter your Api key here"
```

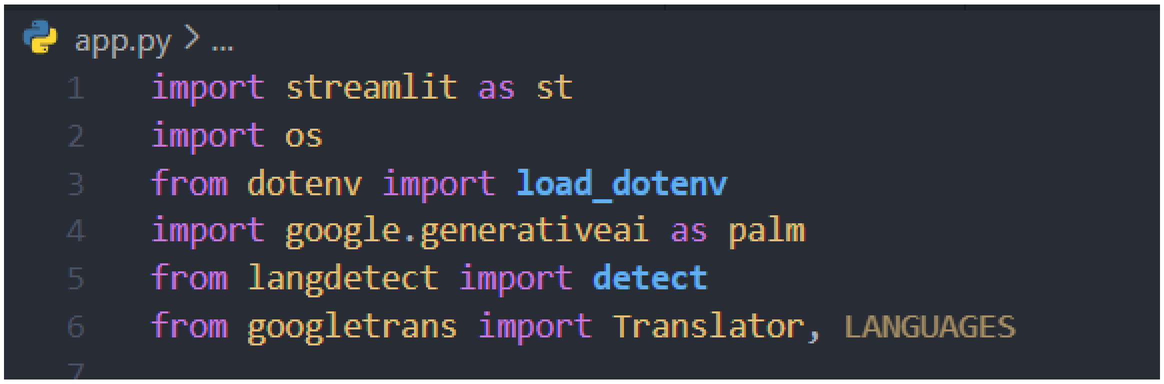
● Create a ***.env*** file and define a variable named ***API_KEY***.

● Assign the copied Google API key to this variable.

● Paste the API key obtained from the previous steps here.

Milestone 3 : Initialize The Pre-Trained Model

Step 1 : Import necessary files

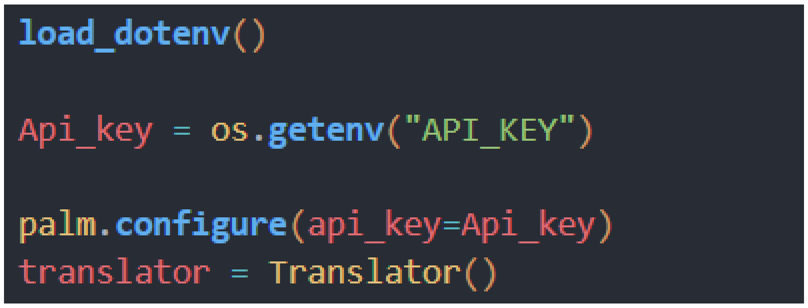


```
app.py > ...
import streamlit as st
import os
from dotenv import load_dotenv
import google.generativeai as palm
from langdetect import detect
from googletrans import Translator, LANGUAGES
```

- Imports the **Streamlit** library, which is used for creating web applications.
- Imports the **OS** module, which provides functions for interacting with the operating system.

- Imports the **load_dotenv** function from the **dotenv** library, used to load environment variables from a .env file.
- Imports the **Google Generative AI** library, aliasing it as palm, for accessing generative AI functions provided by Google.
- Imports the **detect** function from the **langdetect** library, which is used to detect the language of a given text.
- Imports the **Translator** class and **LANGUAGES** constant from the googletrans library, used for translating text between languages and accessing a list of supported languages, respectively.

<u>Step 2 : Configuration of the PALM API with the API key and initialize translator</u>

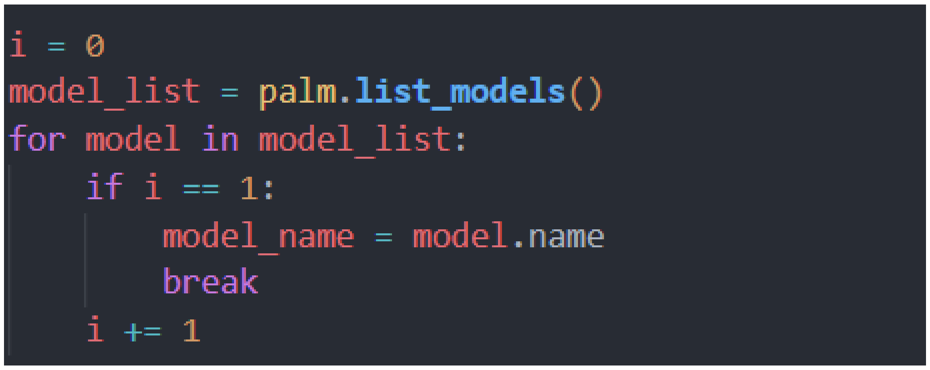


```
load_dotenv()

Api_key = os.getenv("API_KEY")

palm.configure(api_key=Api_key)
translator = Translator()
```

- The script starts by loading environment variables from a ***.env*** file using the ***load_dotenv()*** function from the dotenv package
- Configuring the API key: The ***configure*** function is used to set up Google Generative AI API with the API key retrieved from the environment variables, ensuring secure and authorized access to the AI services.

- The *Translator* class facilitates language translation capabilities within the application.

Step 3 : List Available models and select one



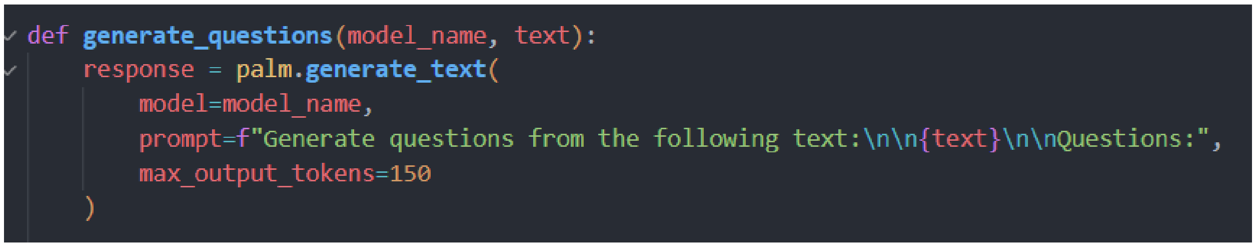
```
i = 0
model_list = palm.list_models()
for model in model_list:
    if i == 1:
        model_name = model.name
        break
    i += 1
```

1. **Model List Retrieval and Iteration:**
    - The code retrieves a list of available models using ***palm.list_models()*** and iterates through this list with a for loop.
2. **Second Model Selection:**
    - It selects the name of the second model which is text-bison-001 in the list (i == 1) and assigns it to the variable ***model_name***, then exits the loop.

Milestone 3 : Interfacing With Pre-Trained Model

In this milestone, we will build a prompt template for summarizing the input content.

Step 1 : Generate Questions from Text



```
def generate_questions(model_name, text):
    response = palm.generate_text(
        model=model_name,
        prompt=f"Generate questions from the following text:\n\n{text}\n\nQuestions:",
        max_output_tokens=150
    )
```

- This function ***generate_questions*** takes two parameters: ***model_name***, which specifies the pre-trained language model to be used, and text, which represents the input text from which questions are to be generated.
- It then utilizes the ***palm.generate_text()*** method to generate questions based on the input text.
- The prompt parameter provides a prompt for the model, instructing it to generate questions from the given text, and ***max_output_tokens*** limits the length of the generated output.

Step 2 : Extract Generated Text from the Result Attribute

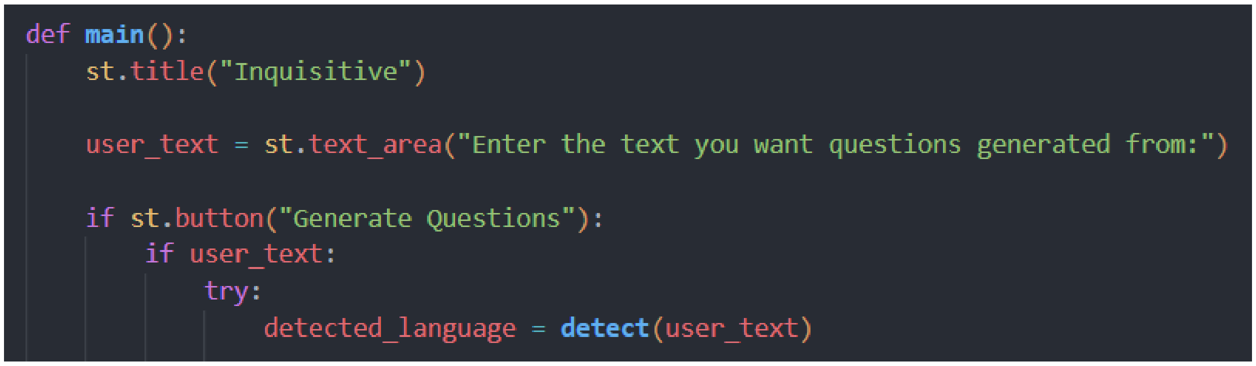
```
questions = response.result.strip() if response.result else "No questions generated"

return questions
```

- This part of the code assigns the generated questions to the variable ***questions***.
- It checks if the ***response.result*** attribute exists; if it does, it strips any leading or trailing whitespace from the result and assigns it to ***questions***.
- If ***response.result*** is empty or doesn't exist, it assigns the string "No questions generated." to ***questions***.
- Finally, it returns the ***questions*** variable, containing either the generated questions or the "No questions generated." message.

Milestone 4 : Model Deployment

In this milestone, we are deploying the created model using ***streamlit***. Model deployment using ***Streamlit*** involves creating a user-friendly web interface for deploying machine learning models, enabling users to interact with the model through a browser. ***Streamlit*** provides easy-to-use tools for developing and deploying data-driven applications, allowing for seamless integration of machine learning models into web-based applications.

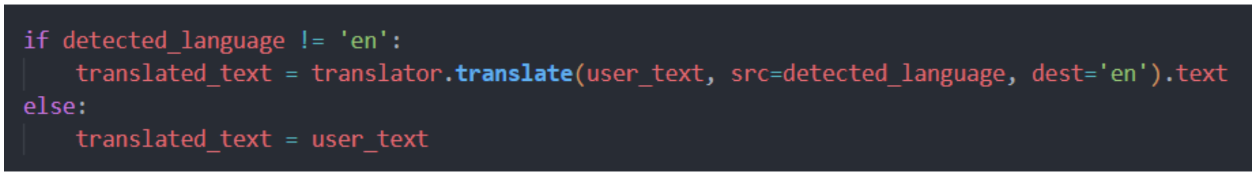
Step 1 : Take Input From the User And Detect The Language



```
def main():
    st.title("Inquisitive")

    user_text = st.text_area("Enter the text you want questions generated from:")

    if st.button("Generate Questions"):
        if user_text:
            try:
                detected_language = detect(user_text)
```

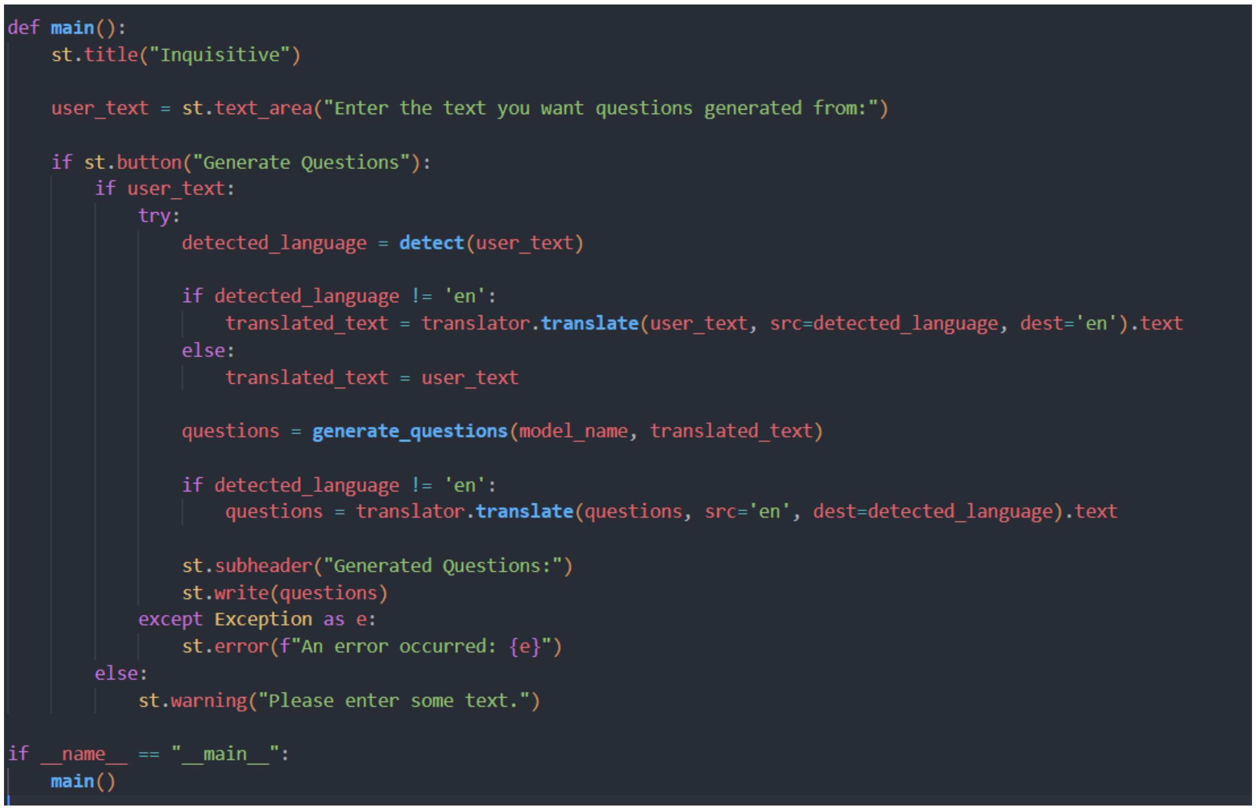
- The *main* function is defined, which contains the entire logic of the Streamlit application.
- ***st.title("Inquisitive")*** sets the title of the Streamlit app to "Inquisitive".
- ***user_text = st.text_area("Enter the text you want questions generated from:")*** creates a text area for the user to input the text from which they want questions generated.
- ***if st.button("Generate Questions"):*** checks if the "Generate Questions" button is pressed.
- ***if user_text:*** checks if the user has entered any text.
- ***detected_language = detect(user_text)*** detects the language of the user input text using the langdetect library.

Step 2 : Translate the Text To English If In Any Other Language


```python
if detected_language != 'en':
    translated_text = translator.translate(user_text, src=detected_language, dest='en').text
else:
    translated_text = user_text
```

- This segment of code checks if the detected language of the user's input text is not English.
- If the language is not English, it uses the ***translator.translate()*** method to translate the text to English, specifying the source language as the detected language and the destination language as English.
- The translated text is then stored in the variable ***translated_text***. If the detected language is already English, the original user text is retained in the ***translated_text*** variable.

Step 3 : Generate Questions Button

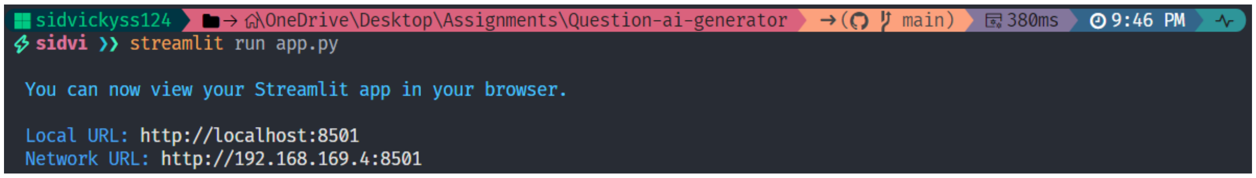


```python
def main():
    st.title("Inquisitive")

    user_text = st.text_area("Enter the text you want questions generated from:")

    if st.button("Generate Questions"):
        if user_text:
            try:
                detected_language = detect(user_text)

                if detected_language != 'en':
                    translated_text = translator.translate(user_text, src=detected_language, dest='en').text
                else:
                    translated_text = user_text

                questions = generate_questions(model_name, translated_text)

                if detected_language != 'en':
                    questions = translator.translate(questions, src='en', dest=detected_language).text

                st.subheader("Generated Questions:")
                st.write(questions)
            except Exception as e:
                st.error(f"An error occurred: {e}")
        else:
            st.warning("Please enter some text.")

if __name__ == "__main__":
    main()
```

- This block of code executes when the user clicks the "Generate Questions" button. It first checks if the user has entered some text. If text is present, it calls the ***generate_questions()*** function to generate questions based on the input text.
- If the detected language of the input text is not English, it translates the generated questions back to the original language using the ***translator.translate()*** method.
- Finally, it displays the generated questions under the subheader "Generated Questions." If no text is entered by the user, it shows a warning message prompting the user to enter some text.

- The code is wrapped in a try-except block to catch and display any errors that occur during execution using ***st.error(f"An error occurred: {e}")***.

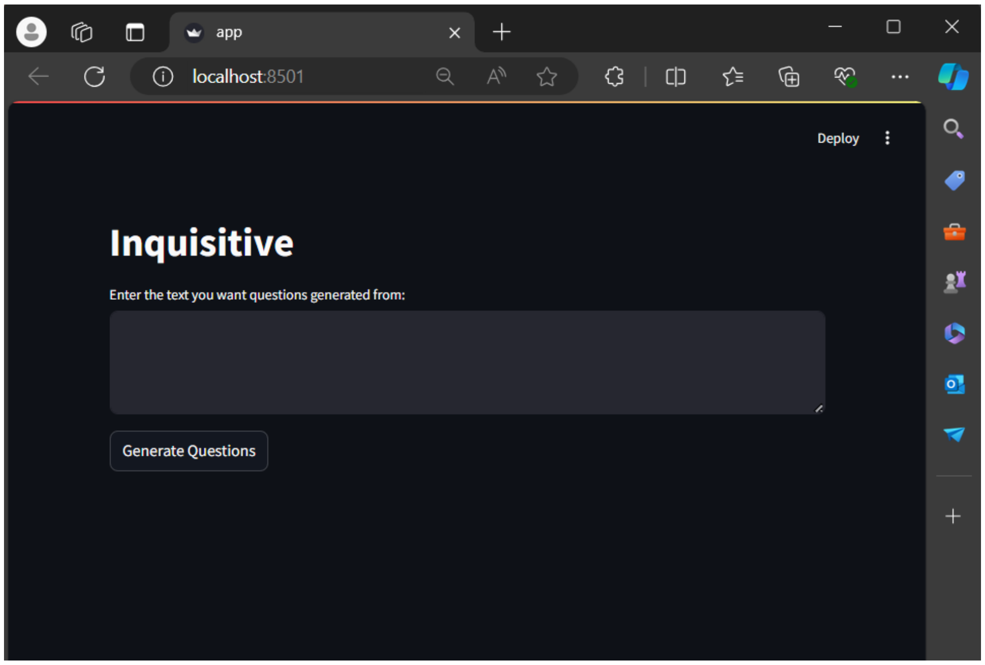
Step 4 : Run the Web Application



```
sidvickyss124  → ⌂\OneDrive\Desktop\Assignments\Question-ai-generator  →( main)  380ms  9:46 PM
sidvi ❯❯ streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.169.4:8501
```

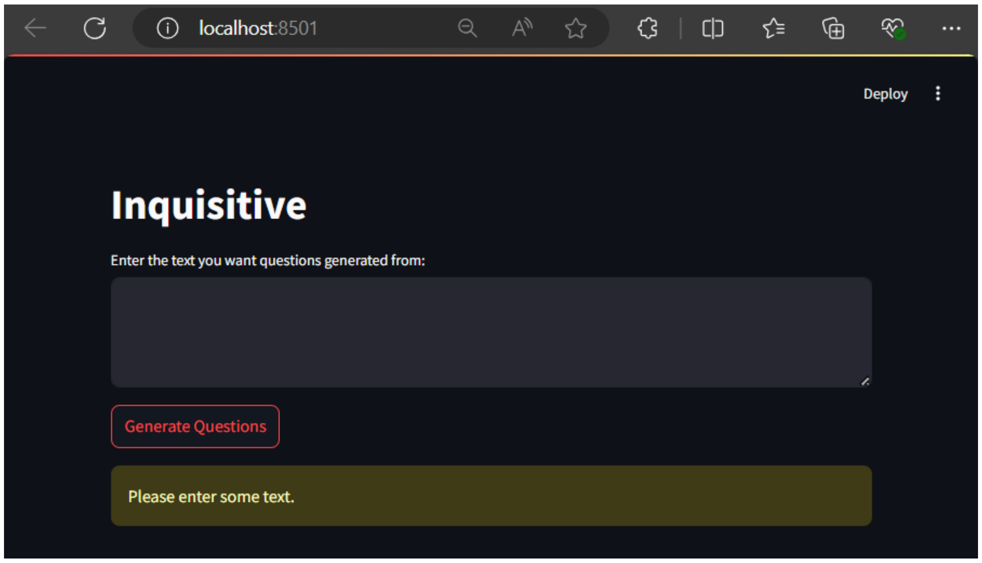
- Navigate to the folder where your Python script is.
- Open the terminal.
- Now type “streamlit run app.py” command.
- Navigate to the localhost where you can view your web page.

Now, the application will open in the web browser.



Inquisitive

Enter the text you want questions generated from:

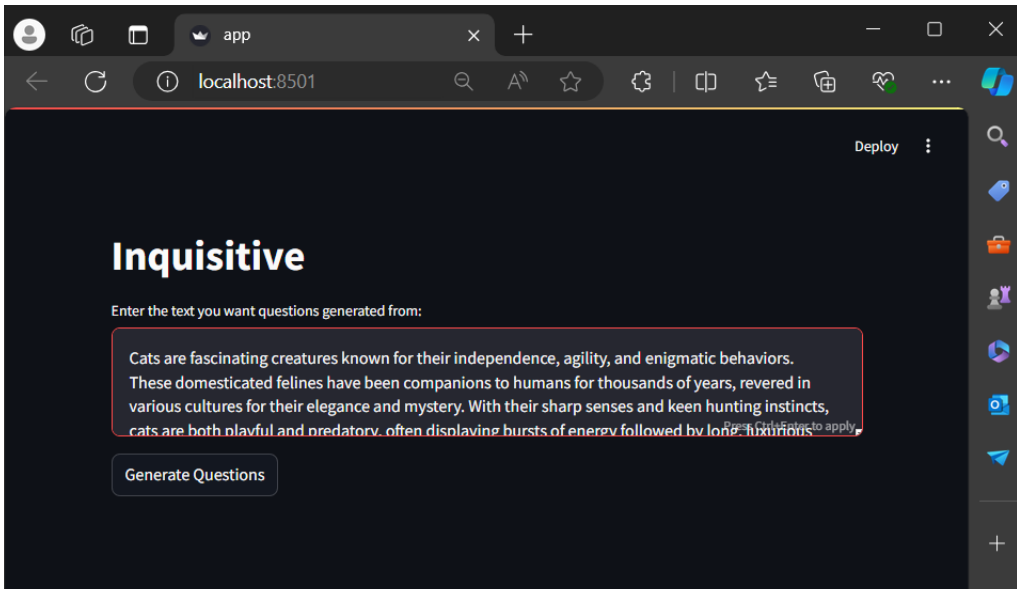


Inquisitive

Enter the text you want questions generated from:

Please enter some text.

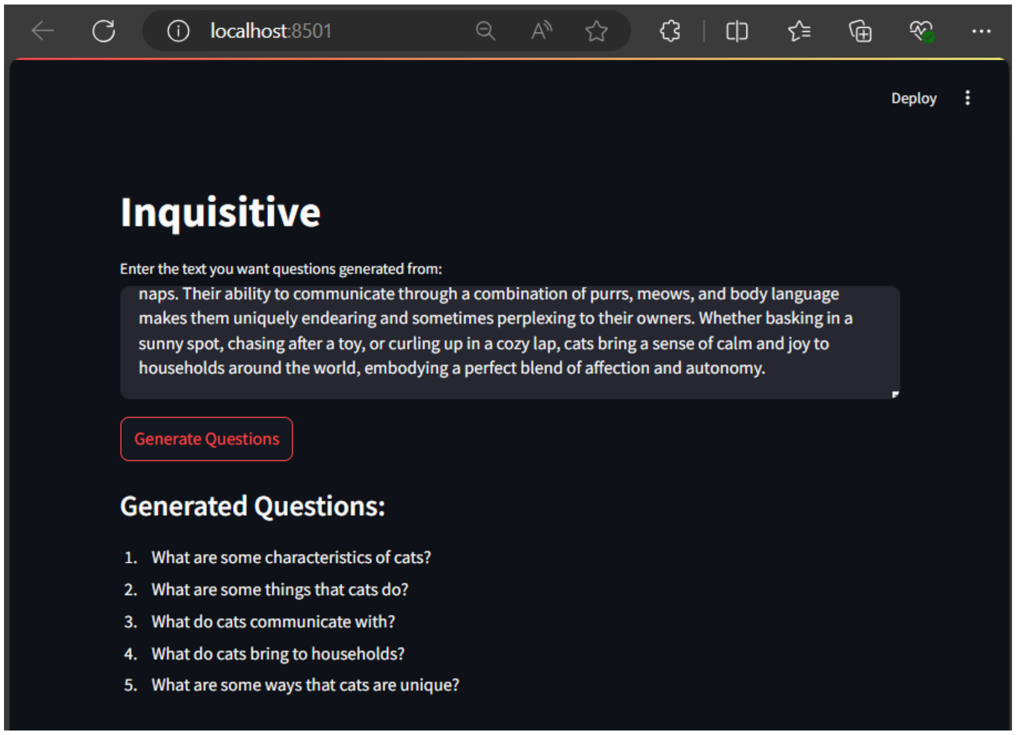
Test case 1: English Language



Inquisitive

Enter the text you want questions generated from:

Cats are fascinating creatures known for their independence, agility, and enigmatic behaviors. These domesticated felines have been companions to humans for thousands of years, revered in various cultures for their elegance and mystery. With their sharp senses and keen hunting instincts, cats are both playful and predatory, often displaying bursts of energy followed by long, luxurious



Inquisitive

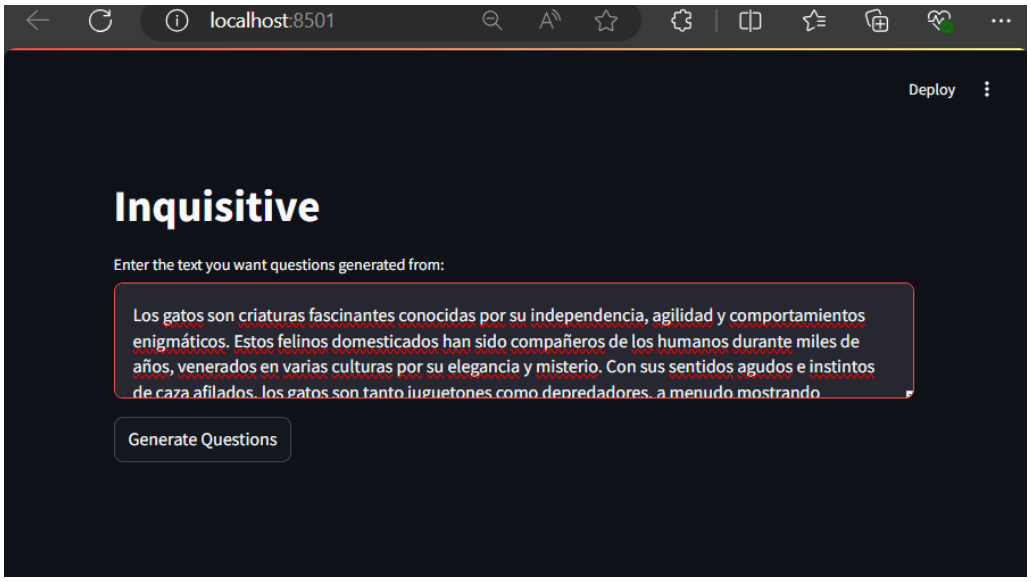
Enter the text you want questions generated from:

naps. Their ability to communicate through a combination of purrs, meows, and body language makes them uniquely endearing and sometimes perplexing to their owners. Whether basking in a sunny spot, chasing after a toy, or curling up in a cozy lap, cats bring a sense of calm and joy to households around the world, embodying a perfect blend of affection and autonomy.

Generated Questions:

1. What are some characteristics of cats?
2. What are some things that cats do?
3. What do cats communicate with?
4. What do cats bring to households?
5. What are some ways that cats are unique?

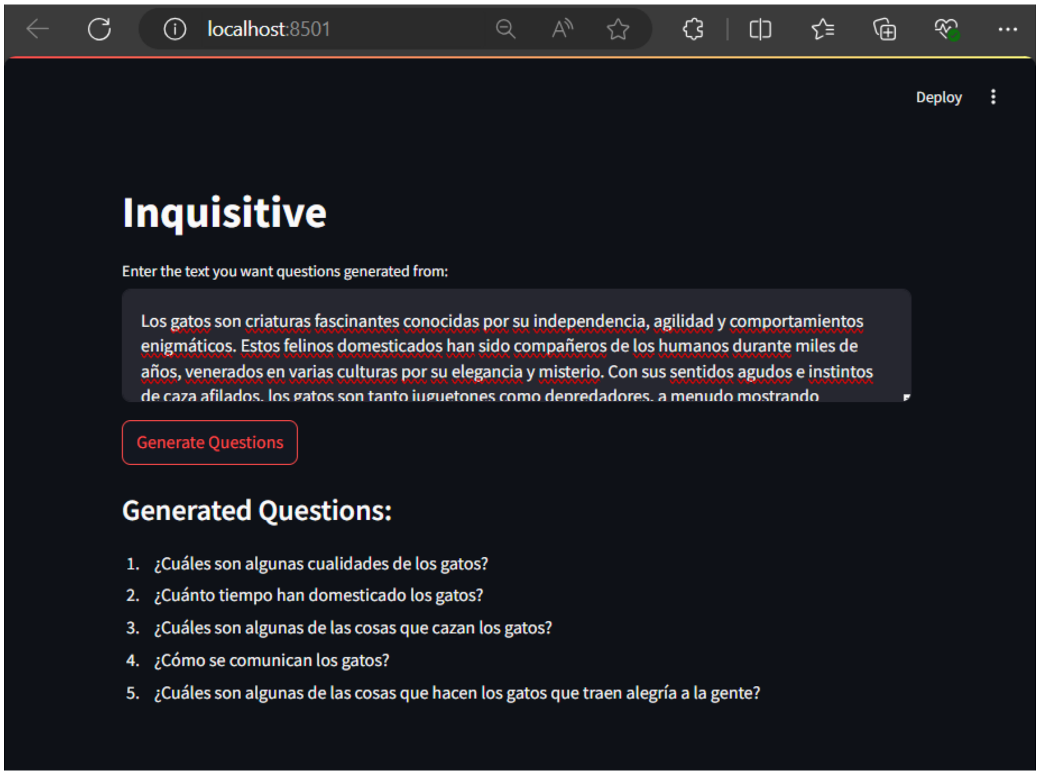
Test case 2: Spanish Language



Inquisitive

Enter the text you want questions generated from:

Los gatos son criaturas fascinantes conocidas por su independencia, agilidad y comportamientos enigmáticos. Estos felinos domesticados han sido compañeros de los humanos durante miles de años, venerados en varias culturas por su elegancia y misterio. Con sus sentidos agudos e instintos de caza afilados, los gatos son tanto juguetones como depredadores, a menudo mostrando



Inquisitive

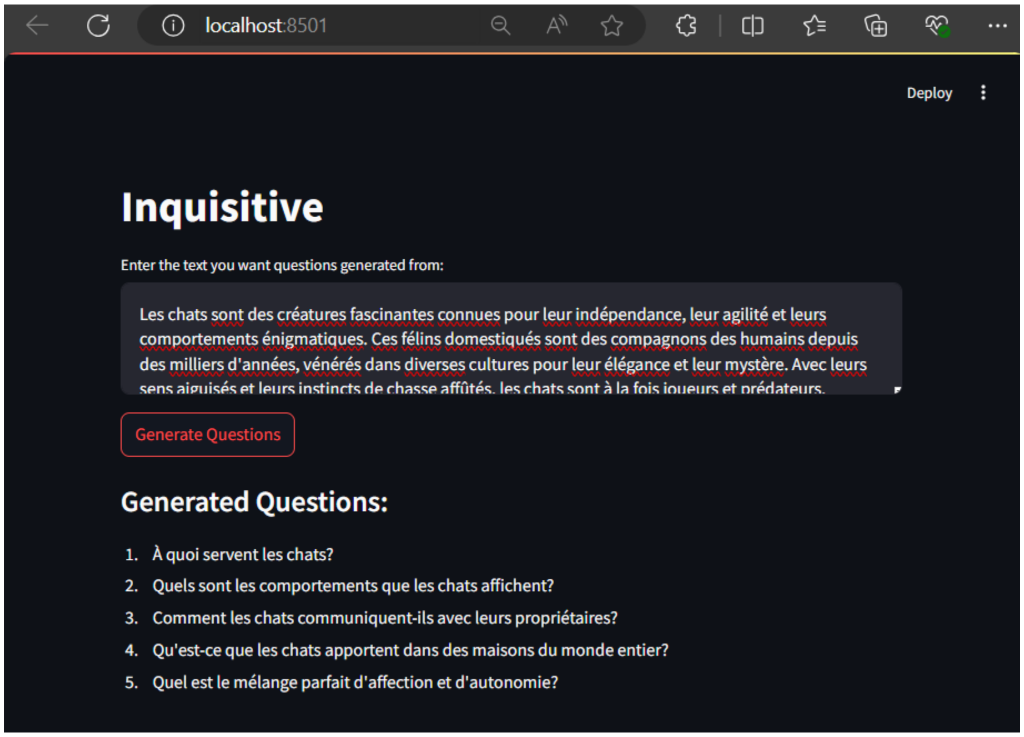
Enter the text you want questions generated from:

Los gatos son criaturas fascinantes conocidas por su independencia, agilidad y comportamientos enigmáticos. Estos felinos domesticados han sido compañeros de los humanos durante miles de años, venerados en varias culturas por su elegancia y misterio. Con sus sentidos agudos e instintos de caza afilados, los gatos son tanto juguetones como depredadores, a menudo mostrando

Generated Questions:

1. ¿Cuáles son algunas cualidades de los gatos?
2. ¿Cuánto tiempo han domesticado los gatos?
3. ¿Cuáles son algunas de las cosas que cazan los gatos?
4. ¿Cómo se comunican los gatos?
5. ¿Cuáles son algunas de las cosas que hacen los gatos que traen alegría a la gente?

Test case 3: French Language



Inquisitive

Enter the text you want questions generated from:

Les chats sont des créatures fascinantes connues pour leur indépendance, leur agilité et leurs comportements énigmatiques. Ces félins domestiqués sont des compagnons des humains depuis des milliers d'années, vénérés dans diverses cultures pour leur élégance et leur mystère. Avec leurs sens aiguisés et leurs instincts de chasse affûtés, les chats sont à la fois joueurs et prédateurs.

Generated Questions:

1. À quoi servent les chats?
2. Quels sont les comportements que les chats affichent?
3. Comment les chats communiquent-ils avec leurs propriétaires?
4. Qu'est-ce que les chats apportent dans des maisons du monde entier?
5. Quel est le mélange parfait d'affection et d'autonomie?

****