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

PROJECT DESCRIPTION

Inquisitive 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

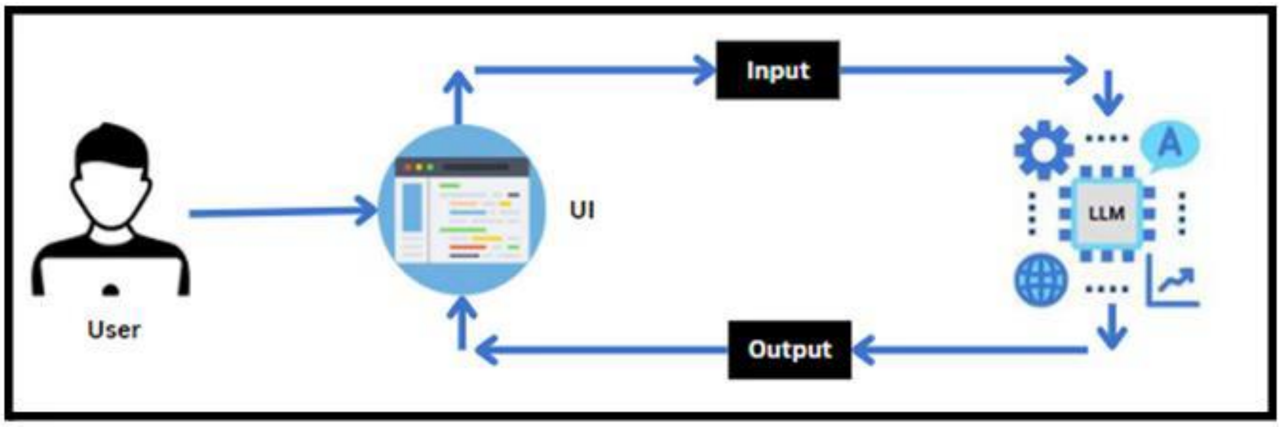
Enhance corporate training programs by incorporating Inquisitive: The GenAI Question Generator. Employees can input training materials, and the application generates quizzes, fostering active learning and reinforcing key concepts. This automated process saves time for trainers and encourages self-paced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

- **Users input text into the UI of Inquisitive.**
- **The input text is then processed and analyzed by the PALM architecture, which is integrated into the backend.**
- **PALM autonomously generates questions based on the input text.**
- **The generated questions are sent back to the frontend for display on the UI.**
- **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**
  -

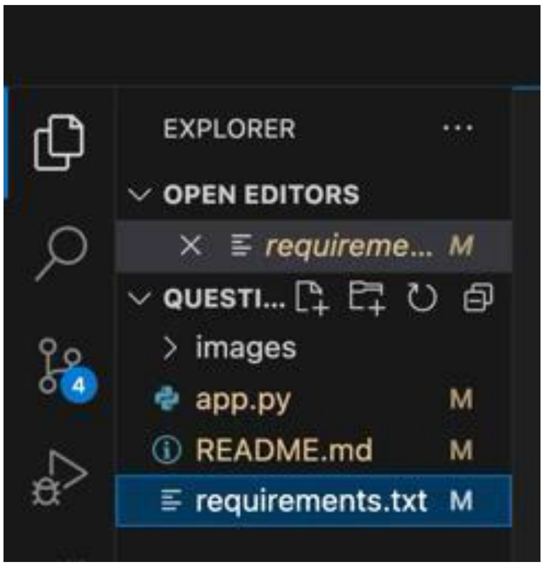
Prior Knowledge:

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:

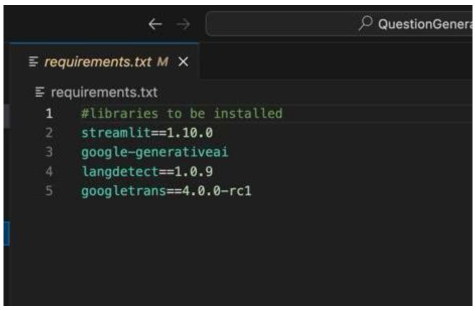


- **app.py: It serves as the primary application file housing both the model and Streamlit UI code.**
- **requirements.txt: It enumerates the libraries necessary for installation to ensure proper functioning.**
- **Images : It is established to store image images utilised under UI**
- **README.RD: It gives the overview and introduction to inquisitive streamline app**

Milestone 1: <u>Requirements Specification</u>

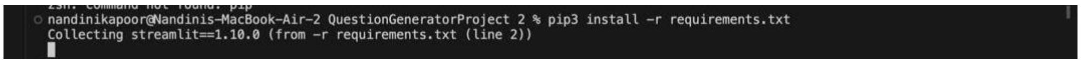
Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment.

Activity 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.**



```
zsh: command not found: pip
nandinikapoor@Nandinis-MacBook-Air-2 QuestionGeneratorProject 2 % pip3 install -r requirements.txt
Collecting streamlit==1.10.0 (from -r requirements.txt (line 2))
```

Activity 2 : Install The Required Libraries

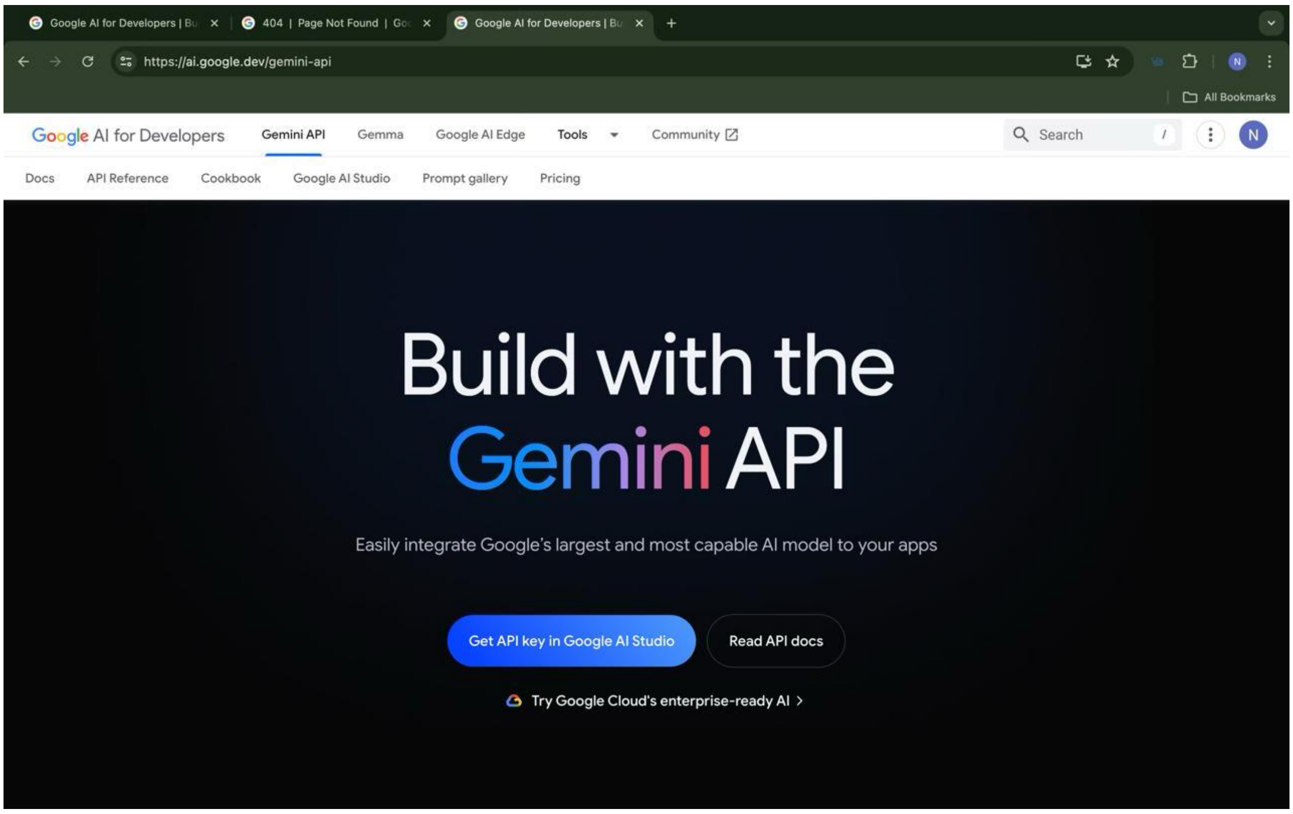
-
- **Open the terminal.**
- **Run the command: pip3 install -r requirements.txt**
- **This command installs all the libraries listed in the requirements.txt file**

Milestone 2 : Initialization of Google API KEY

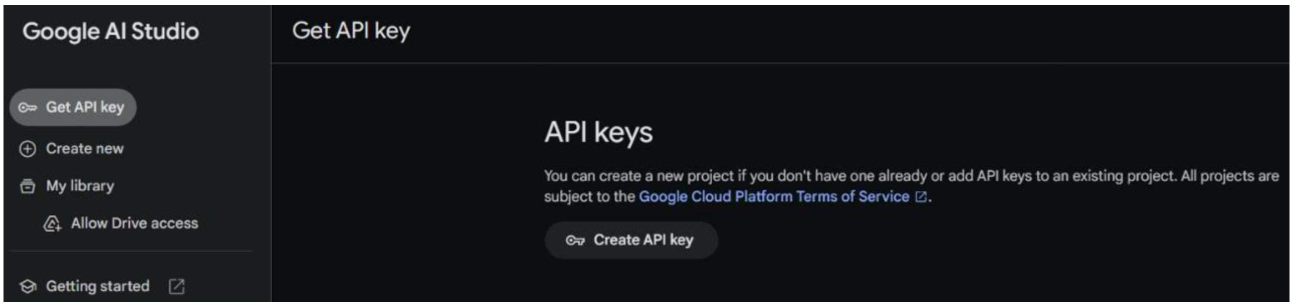
The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API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

Activity 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". (Refer the below images)**
- **Copy the API key.**

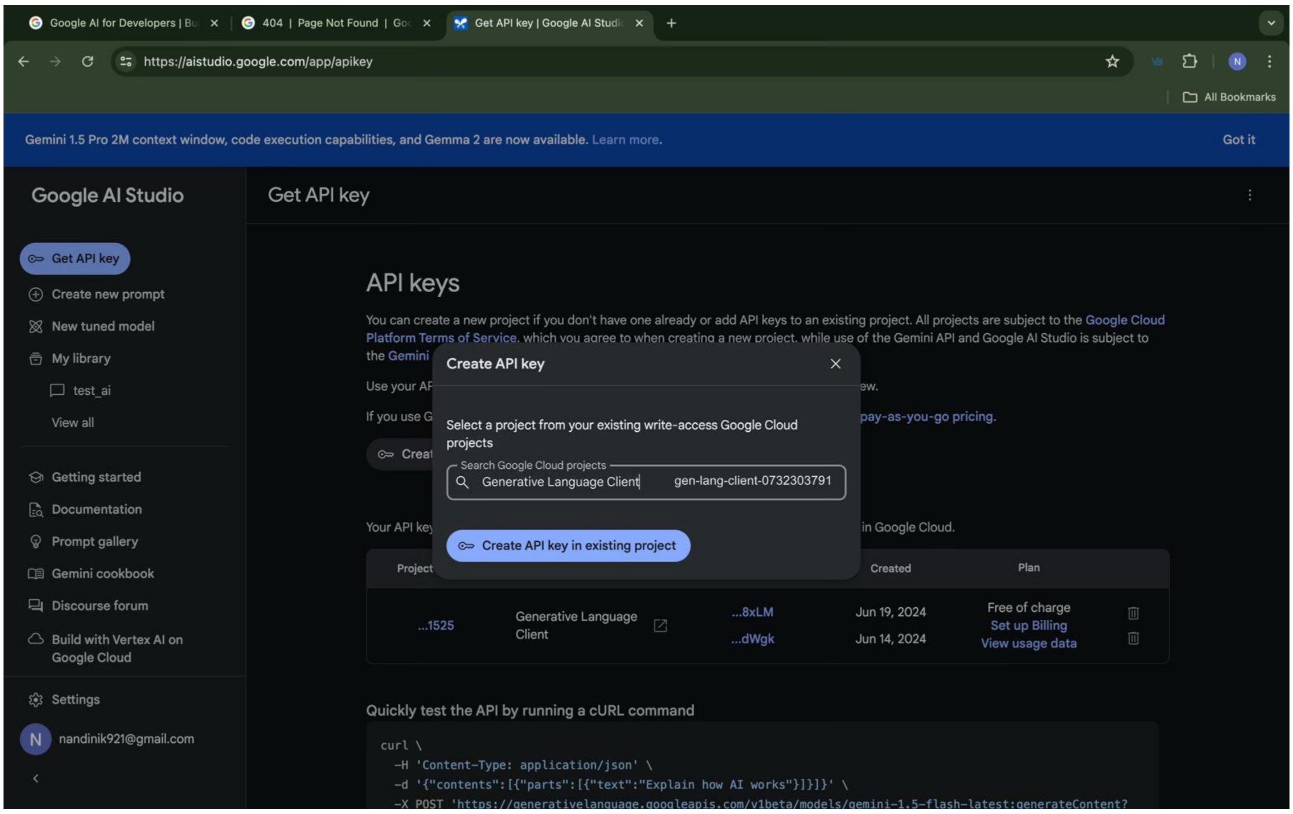

https://ai.google.dev/gemini-api
Google AI for Developers
Gemini API
Gemma
Google AI Edge
Tools
Community
Search
Docs
API Reference
Cookbook
Google AI Studio
Prompt gallery
Pricing
Build with the Gemini API
Easily integrate Google's largest and most capable AI model to your apps
Get API key in Google AI Studio
Read API docs
Try Google Cloud's enterprise-ready AI

After signing in to your account, navigate to the 'Get an API Key' option. Clicking on this option will redirect you to another webpage as shown below



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.

Next, click on 'Create API Key' and choose the generative language client as



Gemini 1.5 Pro 2M context window, code execution capabilities, and Gemma 2 are now available. Learn more.

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

Use your AP ew.

If you use G pay-as-you-go pricing.

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

Generative Language Client gen-lang-client-0732303791

Your API key in Google Cloud.

Project			Created	Plan
...1525	Generative Language Client	...8xLM	Jun 19, 2024	Free of charge
		...dWgk	Jun 14, 2024	Set up Billing View usage data

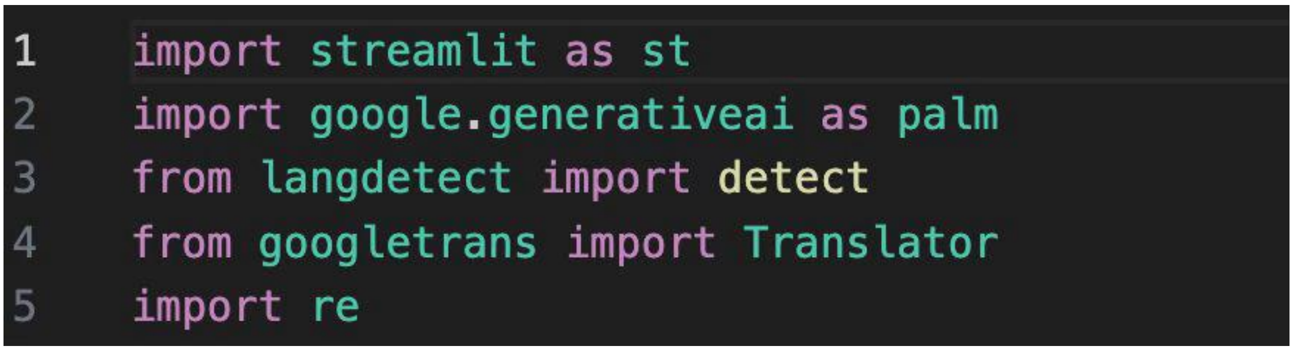
Quickly test the API by running a cURL command

```
curl \
  -H 'Content-Type: application/json' \
  -d '{"contents":[{"parts":[{"text":"Explain how AI works"}]}]}' \
  -X POST 'https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash-latest:generateContent?
```

the project. Then, select 'Create API key in existing project'.

Copy the newly generated API key as it is required for loading the pre-trained model.

Activity 2 : Initialize The Pre-Trained Model



```python
import streamlit as st
import google.generativeai as palm
from langdetect import detect
from googletrans import Translator
import re
```

Import necessary files

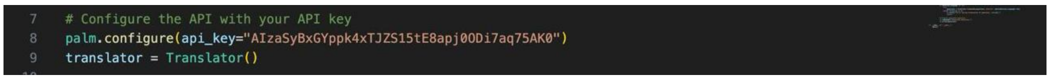
- **Streamlit, a popular Python library, is imported as st, enabling the creation of user interfaces directly within the Python script.**

- **Importing the palm module: This line imports the palm module from the google.generativeai package.**

- **The langdetect library is imported for language detection functionality, allowing the application to identify the language of user-input text.**

- **The code imports the Translator class from the googletrans library, enabling translation capabilities within the application, crucial for handling multilingual text input and output.**

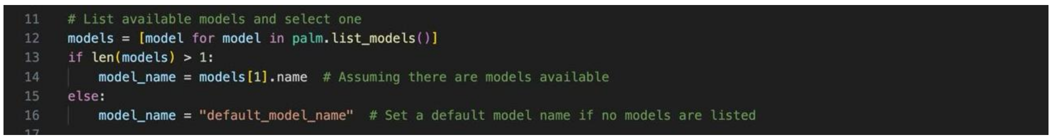
Configuration of the PALM API with the API key and initialize translator



```python
# Configure the API with your API key
palm.configure(api_key="AIzaSyBxGYppk4xTJZS15tE8apj0ODi7aq75AK0")
translator = Translator()
```

- **Configuring the API key: The configure function is used to set up or configure the Google API with an API key. The provided API key, in this case, is "AIzaSyBxGYppk4xTJZS15tE8apj00Di7aq75AK0"'**

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



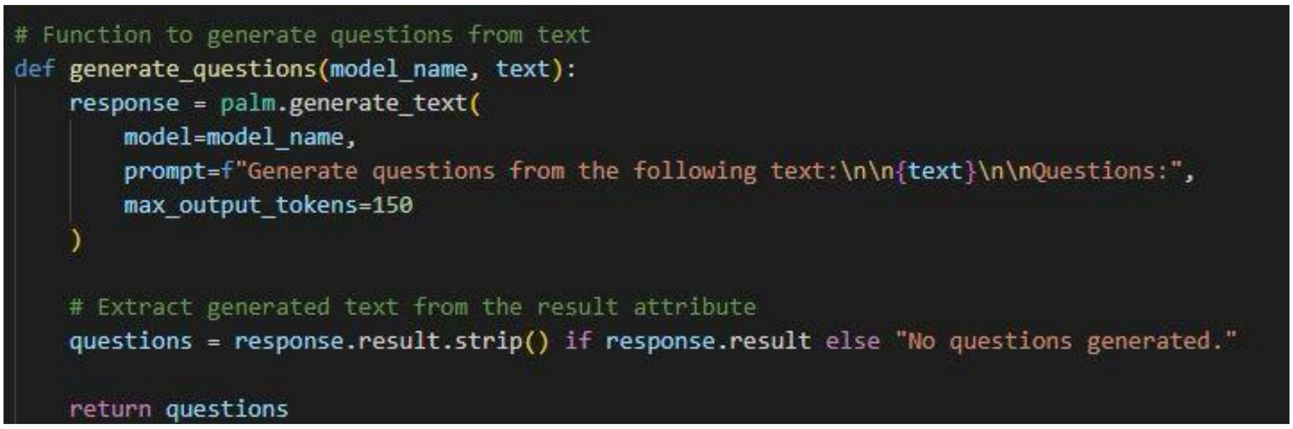
```python
# List available models and select one
models = [model for model in palm.list_models()]
if len(models) > 1:
    model_name = models[1].name  # Assuming there are models available
else:
    model_name = "default_model_name"  # Set a default model name if no models are listed
```

List available models and select one

- **Listing available models: This line retrieves a list of available models using the list_models function provided by the palm module. It creates a list (models) containing information about each available model.**
- **Fetching a model name: It extracts the name of the second model from the list of models (models). Note that Python uses 0-based indexing, so models[1] refer to the second model in the list. The name of this model is then stored in the variable model_name.**

Milestone 3 : Interfacing With Pre-Trained Model

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



```python
# Function to 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
    )

    # Extract generated text from the result attribute
    questions = response.result.strip() if response.result else "No questions generated."

    return questions
```

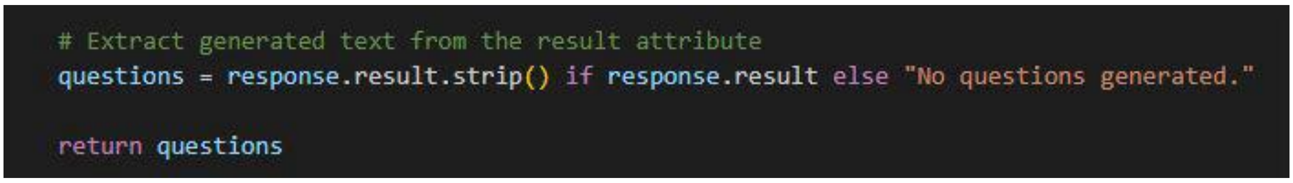
Activity 1 : Generate Questions From Text

- **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**

Activity 2 :Extract Generated Text From The Result Attribute



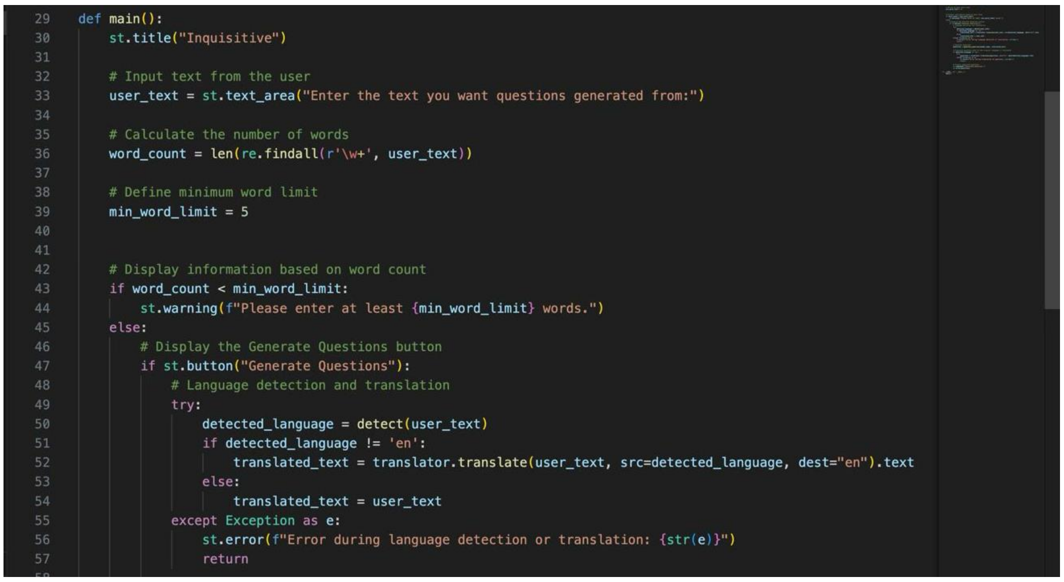
```python
# Extract generated text from the result attribute
questions = response.result.strip() if response.result else "No questions generated."

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.

Activity 1 : Take Input From The User And Detect The Language



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

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

    # Calculate the number of words
    word_count = len(re.findall(r'\w+', user_text))

    # Define minimum word limit
    min_word_limit = 5


    # Display information based on word count
    if word_count < min_word_limit:
        st.warning(f"Please enter at least {min_word_limit} words.")
    else:
        # Display the Generate Questions button
        if st.button("Generate Questions"):
            # Language detection and translation
            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
            except Exception as e:
                st.error(f"Error during language detection or translation: {str(e)}")
                return
```

1. **User Input: Users enter text into a text area in the Streamlit application labeled "Enter the text you want questions generated from."**

2. **Word Count Check: The code counts the number of words in the input text using a regular expression. It requires at least 5 words for further processing.**

3. **User Guidance: If the input text has fewer than 5 words, a warning message is displayed, prompting the user to input more text.**

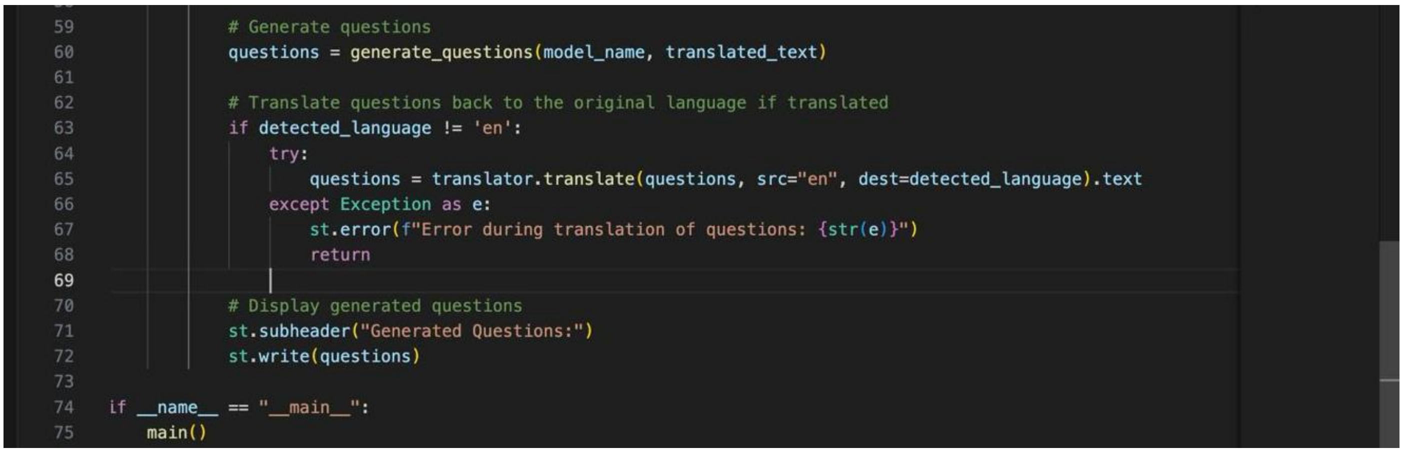
4. **Generate Questions Button: Once the word count requirement is met, a button labeled "Generate Questions" appears.**

5. **Language Handling: Upon clicking the button, the code attempts to detect the language of the input text. If the text is not in English, it translates it to English using a translation service.**

6. **Error Handling: The code includes error handling to manage exceptions that may occur during language detection or translation, displaying an error message if there's an issue.**

7. **Next Steps: The application is designed to proceed with generating questions based on the processed text, but the specific logic for generating questions is not detailed in the provided snippet. This would typically involve further natural language processing (NLP) techniques**.

Activity 2 : Generate Questions Button



```
            # Generate questions
            questions = generate_questions(model_name, translated_text)

            # Translate questions back to the original language if translated
            if detected_language != 'en':
                try:
                    questions = translator.translate(questions, src="en", dest=detected_language).text
                except Exception as e:
                    st.error(f"Error during translation of questions: {str(e)}")
                    return

            # Display generated questions
            st.subheader("Generated Questions:")
            st.write(questions)

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

Question Generation:

- **The code generates questions from user-inputted text using a specified model (`model_name`). This likely involves natural language processing (NLP) techniques to analyze and derive questions from the input text.**

Translation Handling:

- **If the detected language of the input text is not English (`detected_language != 'en'`), the generated questions are translated back into the original language using a translation service (`translator`). This ensures the questions are presented in the language of the original input text.**

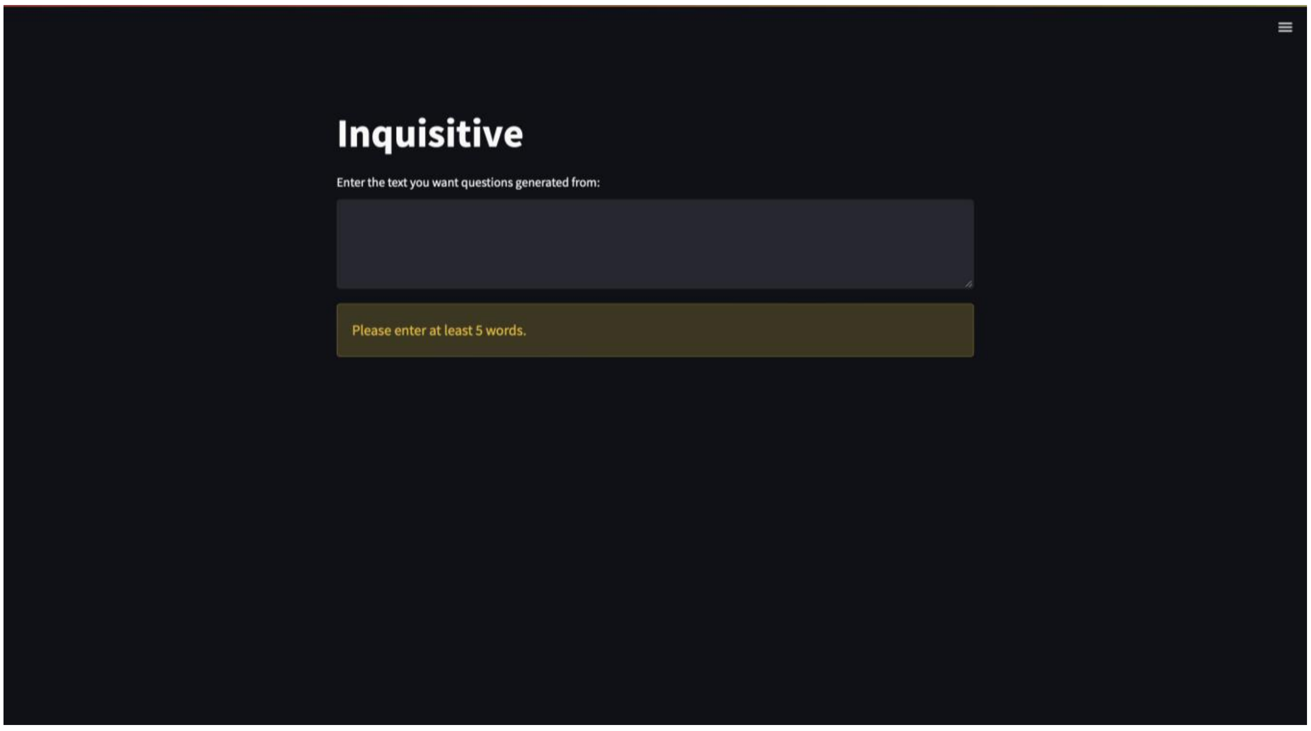
Error Management:

- **The code includes error handling for translation operations. If there is an error during the translation of questions (`translator.translate()`), it catches the exception (`Exception as e`), displays an error message using Streamlit's `st.error()`, and exits the function (`return`).**

Displaying Results:

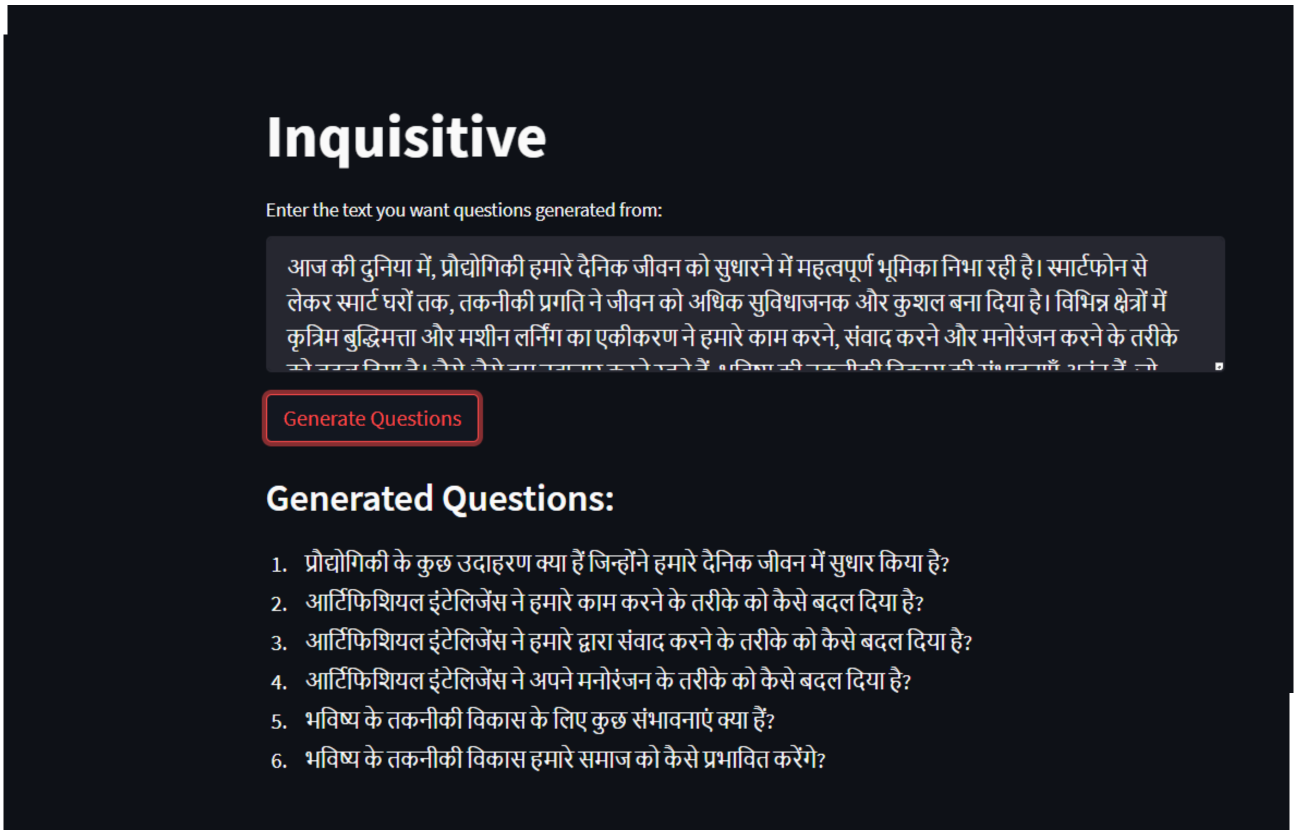
- After generating and potentially translating questions, the application displays the questions under a subheader labeled "Generated Questions" using Streamlit's `st.subheader()` and `st.write()` functions. This provides a clear and organized presentation of the output to the user.

Activity 3 : Run The Web Application


Inquisitive

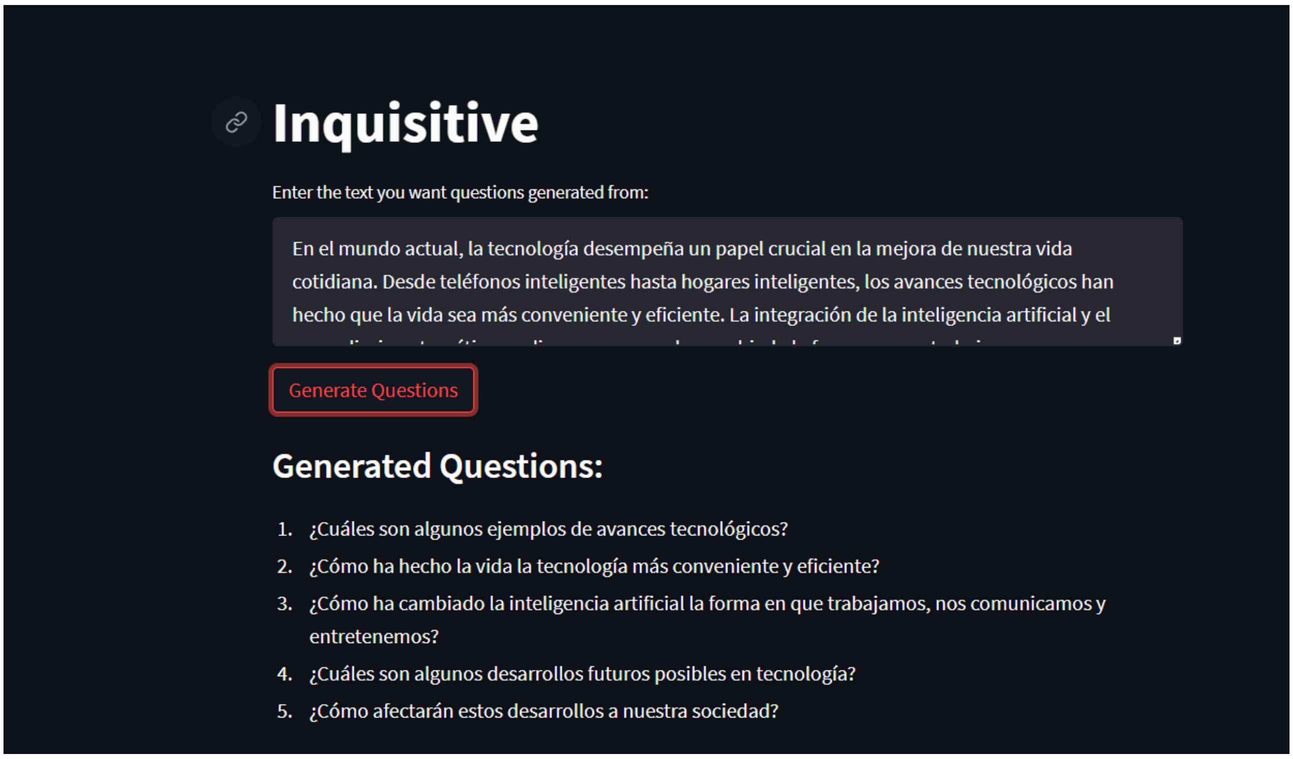
Enter the text you want questions generated from:

Please enter at least 5 words.



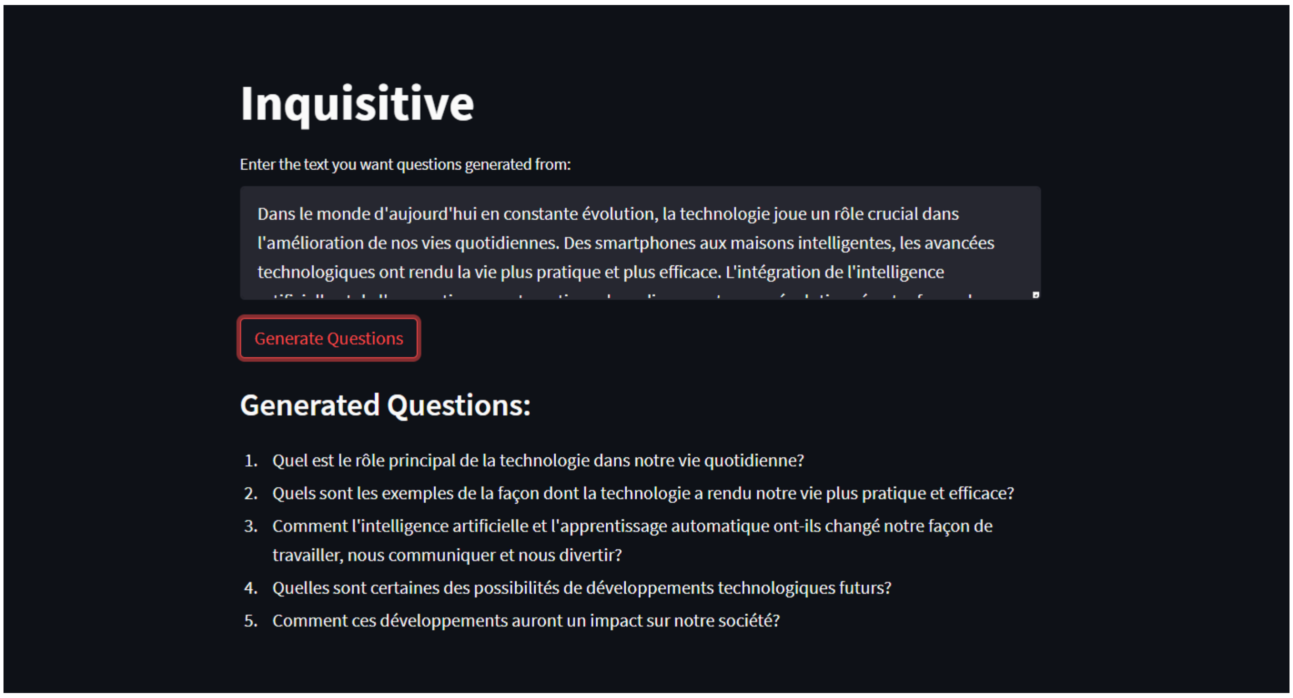
आज की दुनिया में, प्रौद्योगिकी हमारे दैनिक जीवन को सुधारने में महत्वपूर्ण भूमिका निभा रही है। स्मार्टफोन से लेकर स्मार्ट घरों तक, तकनीकी प्रगति ने जीवन को अधिक सुविधाजनक और कुशल बना दिया है। विभिन्न क्षेत्रों में कृत्रिम बुद्धिमत्ता और मशीन लर्निंग का एकीकरण ने हमारे काम करने, संवाद करने और मनोरंजन करने के तरीके

1. प्रौद्योगिकी के कुछ उदाहरण क्या हैं जिन्होंने हमारे दैनिक जीवन में सुधार किया है?
2. आर्टिफिशियल इंटेलिजेंस ने हमारे काम करने के तरीके को कैसे बदल दिया है?
3. आर्टिफिशियल इंटेलिजेंस ने हमारे द्वारा संवाद करने के तरीके को कैसे बदल दिया है?
4. आर्टिफिशियल इंटेलिजेंस ने अपने मनोरंजन के तरीके को कैसे बदल दिया है?
5. भविष्य के तकनीकी विकास के लिए कुछ संभावनाएं क्या हैं?
6. भविष्य के तकनीकी विकास हमारे समाज को कैसे प्रभावित करेंगे?



En el mundo actual, la tecnología desempeña un papel crucial en la mejora de nuestra vida cotidiana. Desde teléfonos inteligentes hasta hogares inteligentes, los avances tecnológicos han hecho que la vida sea más conveniente y eficiente. La integración de la inteligencia artificial y el

1. ¿Cuáles son algunos ejemplos de avances tecnológicos?
2. ¿Cómo ha hecho la vida la tecnología más conveniente y eficiente?
3. ¿Cómo ha cambiado la inteligencia artificial la forma en que trabajamos, nos comunicamos y entretenemos?
4. ¿Cuáles son algunos desarrollos futuros posibles en tecnología?
5. ¿Cómo afectarán estos desarrollos a nuestra sociedad?



Inquisitive

Enter the text you want questions generated from:

Dans le monde d'aujourd'hui en constante évolution, la technologie joue un rôle crucial dans l'amélioration de nos vies quotidiennes. Des smartphones aux maisons intelligentes, les avancées technologiques ont rendu la vie plus pratique et plus efficace. L'intégration de l'intelligence

Generated Questions:

1. Quel est le rôle principal de la technologie dans notre vie quotidienne?
2. Quels sont les exemples de la façon dont la technologie a rendu notre vie plus pratique et efficace?
3. Comment l'intelligence artificielle et l'apprentissage automatique ont-ils changé notre façon de travailler, nous communiquer et nous divertir?
4. Quelles sont certaines des possibilités de développements technologiques futurs?
5. Comment ces développements auront un impact sur notre société?