Data Collection and Preprocessing Phase

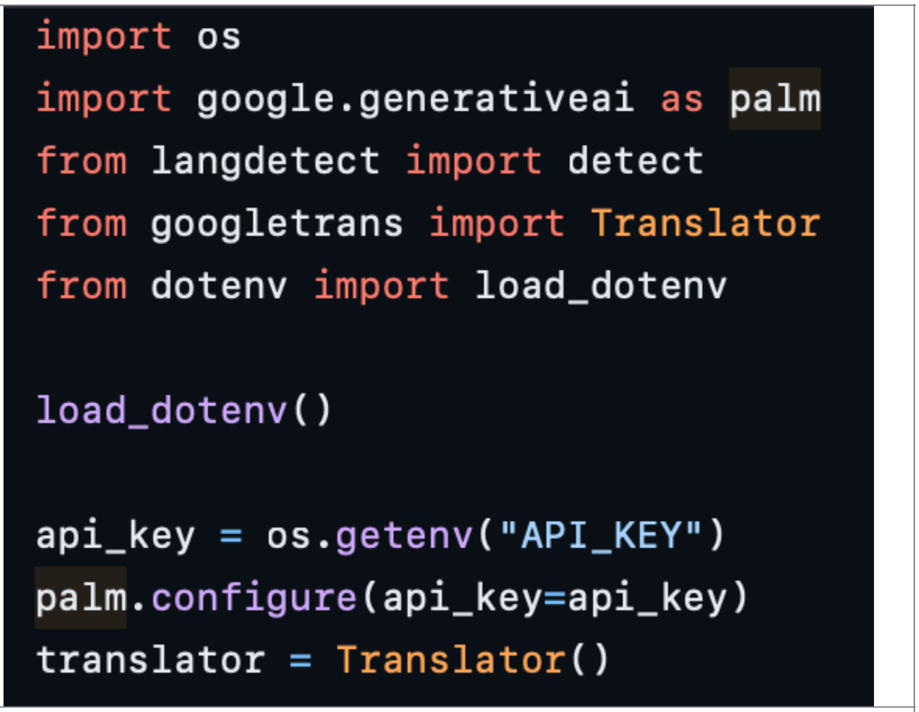
Date	19 July 2024
Team ID	SWTID1720099578
Project Title	Inquisitive: A Multilingual AI Question Generator
Maximum Marks	6 Marks

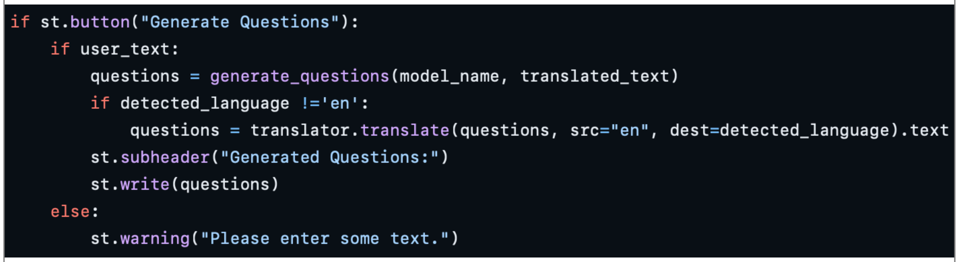
Data Exploration and Preprocessing Template

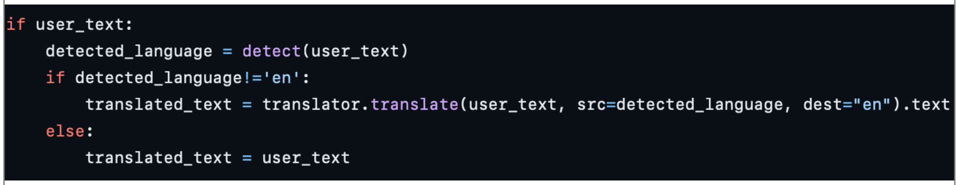
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

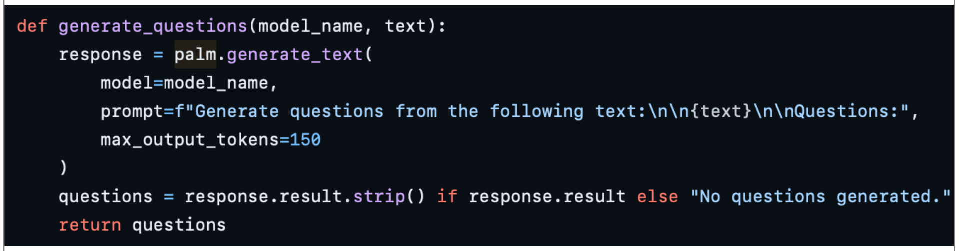
Section	**Description**
Data Overview	The project uses user-provided text inputs in various languages to generate questions. The dataset includes: • **Text**: Input text for question generation. • **Language**: Detected language of the input text. • **Translated Text**: English translation of the input text (if necessary). • **Generated Questions**: Questions generated from the translated text. **Dimensions:** • Number of records: Dependent on user input. • Number of features: 4 (Text, Language, Translated Text, Generated Questions)

Univariate Analysis	**Text**: • Analyze text length. • Common words. • Language distribution. **Language**: • Frequency distribution of detected languages. **Translated Text**: • Analysis of text length after translation. **Generated Questions**: • Number and quality of generated questions.
Bivariate Analysis	**Text Length vs. Generated Questions**: • Correlation between text length and the number or quality of generated questions. **Language vs. Generated Questions**: • Analysis to determine if certain languages produce more or better-quality questions.
Multivariate Analysis	**Language, Text Length, and Generated Questions**: • Combined effect of language and text length on the generated questions. **Sentiment of Text and Generated Questions**: • Influence of input text sentiment on the generated questions.
Outliers and Anomalies	**Outliers in Text Length**: • Identify unusually short or long texts. **Anomalies in Generated Questions**: • Detect nonsensical or irrelevant questions.
Data Preprocessing Code Screenshots	

Loading Data	
	import os
	import google.generativeai as palm
	from langdetect import detect
	from googletrans import Translator
	from dotenv import load_dotenv
	
	load_dotenv()
	
	api_key = os.getenv("API_KEY")
	palm.configure(api_key=api_key)
	translator = Translator()

Handling Missing Data	
	if st.button("Generate Questions"):
	    if 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)
	    else:
	        st.warning("Please enter some text.")

Data Transformation	
	if user_text:
	    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

Feature Engineering	
	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
	    )
	    questions = response.result.strip() if response.result else "No questions generated."
	    return questions

Save Processed Data	The primary use case is for users to input text and receive questions immediately. This real-time interaction does not require storing the questions, as they are displayed directly to the user.