***Flavour Fusion: AI-Driven Recipe Blogging***

Category: Cloud Application Development

Skills Required:  
Machine Learning

Project Description:

flavour Fusion: AI-Driven Recipe Blogging is a web application that leverages Google's Generative AI to create unique and customized recipe blogs. The app provides users with the ability to input a topic and specify the desired word count for their recipe blog. Using the specified parameters, the AI generates detailed and engaging recipe content. Additionally, the app includes a fun feature where it tells a programmer joke to entertain users while the AI is generating the content.

Scenario 1: Creating a Vegan Recipe Blog

A food blogger specializing in vegan recipes opens the Flavour Fusion app and inputs "Vegan Chocolate Cake" with a 1200-word count. As the app generates the content, it entertains them with a programmer joke. The AI quickly delivers a detailed and creative recipe. The blogger reviews the well-crafted content and incorporates it into their blog, ready to share with their audience.

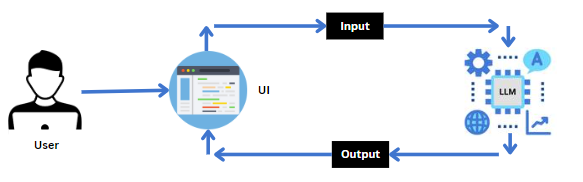
Scenario 2: Crafting a Quick Weeknight Dinner Recipe Blog

A busy professional looking for quick dinner ideas uses the Flavour Fusion app, inputting "Quick Weeknight Dinners" and specifying an 800-word count. The app provides a lighthearted joke while generating the content. The AI produces a concise yet practical recipe blog filled with quick and easy dinner ideas. The user finds the suggestions useful and incorporates them into their weeknight meal planning.

Scenario 3: Developing a Gluten-Free Baking Recipe Blog

A baker specializing in gluten-free recipes accesses the Flavor Fusion app to generate new content for their blog. They enter "Gluten-Free Bread" as the topic and select a 1500-word count. The app entertains with a joke during the content creation process. The AI delivers a comprehensive and well-detailed recipe. The baker reviews the high-quality content and publishes it on their gluten-free baking blog, confident it will be valuable to their readers.///’

***Architecture:***



***Project Flow***

* + 1. Users input a topic and specify the desired length of the blog post through the Streamlit UI.
    2. The input topic and length are sent to the Gemini 1.5 Flash language model, which is integrated into the backend.
    3. Gemini 1.5 Flash processes the input and generates a blog post based on the user's specifications.
    4. The model autonomously creates a well-structured, engaging blog post tailored to the specified topic and word count.
    5. The generated blog post is sent back to the frontend for display on the Streamlit app.
    6. Users can customize the blog post further if desired and export or copy the content for their use.
  1. To accomplish this, we have to complete all the activities listed below,
     1. Initialize Gemini 1.5 Flash:
     2. Generate Gemini 1.5 Flash  API
     3. Initialize the pre-trained model
     4. Interfacing with Pre-trained Model
        + Blog Generation
     5. Model Deployment
        + Deploy the application using Streamlit

**Prior Knowledge**

Duration: 1 Hrs

Skill Tags:

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

* LLM & Gemini 1.5 Flash :

Alarge language modelis a type of artificial intelligence algorithm that applies neural network techniques with lots of parameters to process and understand human languages or text using self-supervised learning techniques. Tasks like text generation, machine translation, summary writing, image generation from texts, machine coding, chat-bots, or Conversational AI are applications of the Large Languag.e Model. Examples of such LLM models are Chat GPT by open AI, BERT (Bidirectional Encoder Representations from Transformers) by Google, etc.

<https://www.geeksforgeeks.org/large-language-model-llm/>

<https://cloud.google.com/vertex-ai/docs/generative-ai/learn-resources>

* Streamlit:

Basic knowledge of building interactive web applications using Streamlit.

Understanding of Streamlit’s UI components and how to integrate them with backend logic.

<https://www.datacamp.com/tutorial/streamlit>

**Project Structure:**

Duration: 1 Hrs

Skill Tags:

Create the Project folder which contains application file as shown below



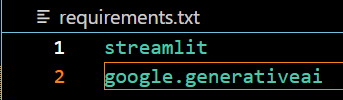
Requirements Specification

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.

**Create a requirements.txt file to list the required libraries**

Duration: 1 Hrs

Skill Tags:



**Install the required libraries.**

Duration: 1 Hrs

Skill Tags:



Generate Google API key

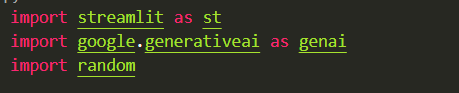
* Click on the link (<https://developers.generativeai.google/>).
* Then click on “Get API key in Google AI Studio”.
* Click on “Get API key” from the right navigation menu.
* Now click on “Create API key”. (Refer the below images)
* Copy the API key.

**Initialize the pre-trained model**

Duration: 1 Hrs

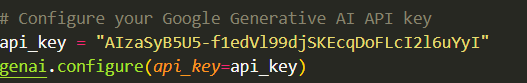
Skill Tags:

 Import necessary files



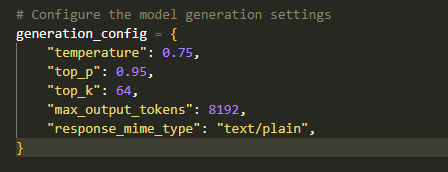
* Streamlit, a popular Python library, is imported as st, enabling the creation of user interfaces directly within the Python script.
* Google Generative AI (genai):Imported to interact with the Gemini 1.5 Flash model.

Activity 2.2: Configuration of the Gemini 1.5 Flash API



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“AIzaSyB5U5-f1edVl99djSKEcqDoFLcI2l6uYyI”

Activity 2.3: Define the model to be used



The image shows a snippet of code that defines a dictionary in Python named `generation\_config`. This dictionary is used to configure the settings for model generation in an AI application. Here are the details of the configuration settings:

1. temperature: Set to `0.75`, this parameter controls the randomness of the output. A higher value increases the diversity, making the responses less predictable and more varied.

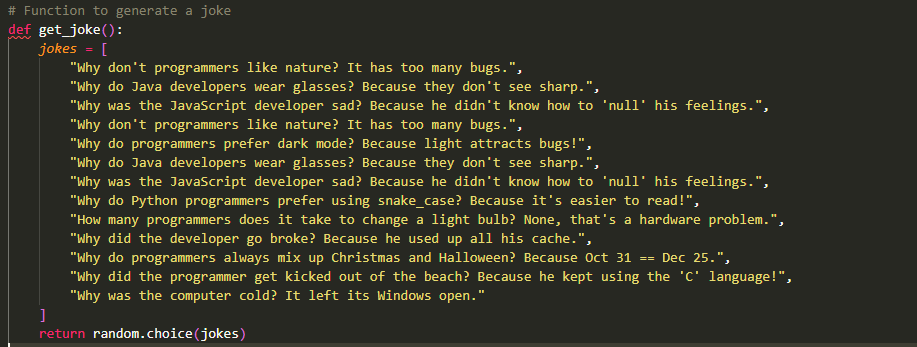
2. top\_p: Set to `0.95`, this parameter is known as nucleus sampling. It restricts the model to select token outputs from the top 95% of the probability distribution, effectively narrowing down the choices to the most likely ones.

3. top\_k: Set to `64`, this restricts the model to only consider the top 64 tokens for each choice point, filtering out less likely outputs to increase the quality of the content.

4. max\_output\_tokens: Set to `8192`, this limits the maximum number of tokens (words and punctuation) that the model can generate in a single response. This is used to control the length of the generated output.

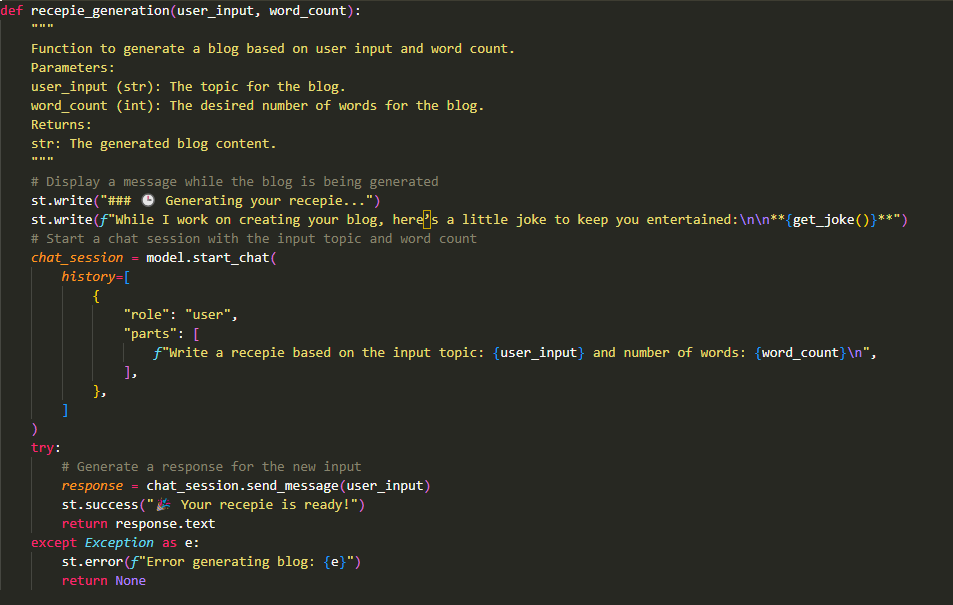
5. response\_mime\_type: Set to `"text/plain"`, indicating that the output from the model will be in plain text format.

Joke Generation (get\_joke() Function)



1. This function selects and returns a random programming joke from a predefined list
2. A list of jokes is stored in the jokes list.
3. The random.choice(jokes) function is used to randomly select and return a joke from this list.
4. This function is called within the recipe generation function to provide a light-hearted joke while the recipe is being created.

### **Recipe Generation (recepie\_generation() Function)**



1. Generates a recipe based on user input and a specified word count.
2. user\_input (str): The topic or theme for the recipe.
3. word\_count (int): The desired length of the recipe in words.
4. Displays a message indicating that the recipe is being generated.
5. Calls get\_joke() to display a joke to the user while waiting.
6. Starts a chat session using the Gemini 1.5 Flash model, passing the user input and word count.
7. Attempts to generate the recipe and returns the generated text if successful.
8. Handles any exceptions by displaying an error message.
9. This function is triggered when the user clicks the "Generate recipe" button in the Streamlit interface.

Output

