**SMARTINTERNZ**

Session Aug-Sept 2024

**PROJECT REPORT**

**On**

**“GEMINIDECODE: MULTILANGUAGE DOCUMENT EXTRACTION BY GEMINI PRO”**

Submitted in partial fulfilment of the requirement for the award of the completion of

**GOOGLE CLOUD GENERATIVE AI INTERNSHIP**

**Submitted to-**

**MOSES)**

**Submitted by-**

**Student**

**UMIKA KHUGSAL**

# ACKNOWLEDGEMENT

I express my sincere gratitude towards Smartinternz and Google for valuable oppturnity and resources provided throughout the work. Without their kind patronage and guidance, the project would not have taken shape. I would also like to express my sincere regards for kind approval of the project and thankfulness for time to time counselling and advises.

Date: 24th September 2024 Student name: **Umika Khugsal**

**Table of contents**

**Acknowledgement**

**Chapter 1 Introduction**

**Chapter 2 Methodology**

**Chapter 3 Problem Statement**

**Chapter 4 System Design**

**Chapter 5 Technology Bucket**

**Chapter 6 Output screens**

**Appendices** Implementation code

**References**

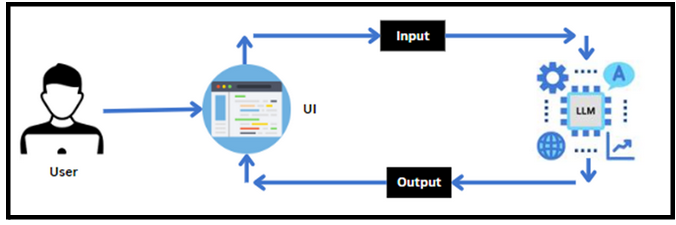
**Chapter 1**

**Introduction**

**Concept**

The GeminiDecode manages the extraction of information in Documents and Images by using the most crucial information based on Google cloud API supplied by a user and other criteria that take into account the user's choice and interest.

It finds out the match between user and item and imputes the similarities between users and items for recommendation of results in users desired language and Format.

GeminiDecode is a cutting-edge solution designed to extract and process data from documents in multiple languages with unparalleled efficiency. Leveraging advanced natural language processing (NLP) and machine learning algorithms, it seamlessly identifies, extracts, and categorizes information from diverse document formats, ensuring accuracy and speed. Ideal for global businesses, GeminiDecode supports over 50 languages, providing robust data extraction capabilities that streamline workflows, enhance productivity, and improve decision-making processes.



**Feature Selection**

Feature selection is a dimensionality reduction method that removes unnecessary features from a dataset. It's different from feature extraction, which creates new features, while feature selection keeps a subset of the original features. It Improves accuracy by reducing the amount of misleading data, feature selection helps models perform better. Reduces overfitting, Fewer redundant data points means fewer chances of making decisions based on noise. Reduces training time , With less data algorithms can run faster. Avoids the curse of dimensionality in a dataset, which can prevent the data points from becoming sparse and statistically insignificant. Improves interpretability and makes it easier to interpret than feature extraction.

**Initial Model Training**

The project does not include a traditional model training phase in the sense of training machine learning models from scratch. Instead, it utilizes pre-trained models from Google Generative AI and constructs a pipeline for processing PDF documents, extracting text, and facilitating question-answer interactions. The steps include configuring and using these models for embeddings, creating a vector store for similarity search, and setting up a conversational chain with pre-defined prompts.

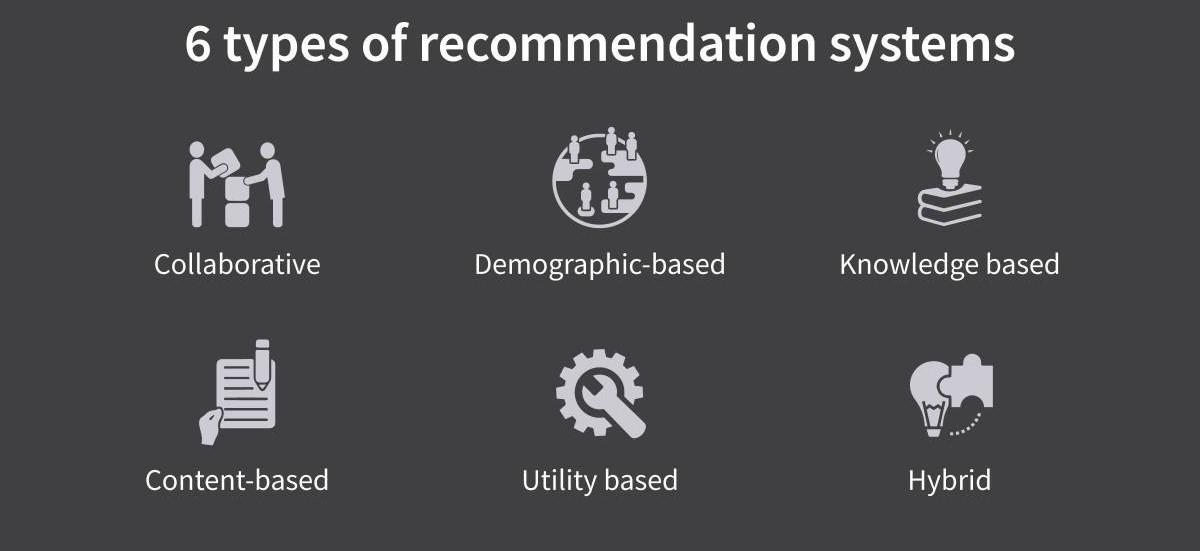
* Model Validation and Evaluation Report: **Model:** Gemini Pro Model 1.5 **Accuracy:** 99.7%

**Model Optimization and Tuning**

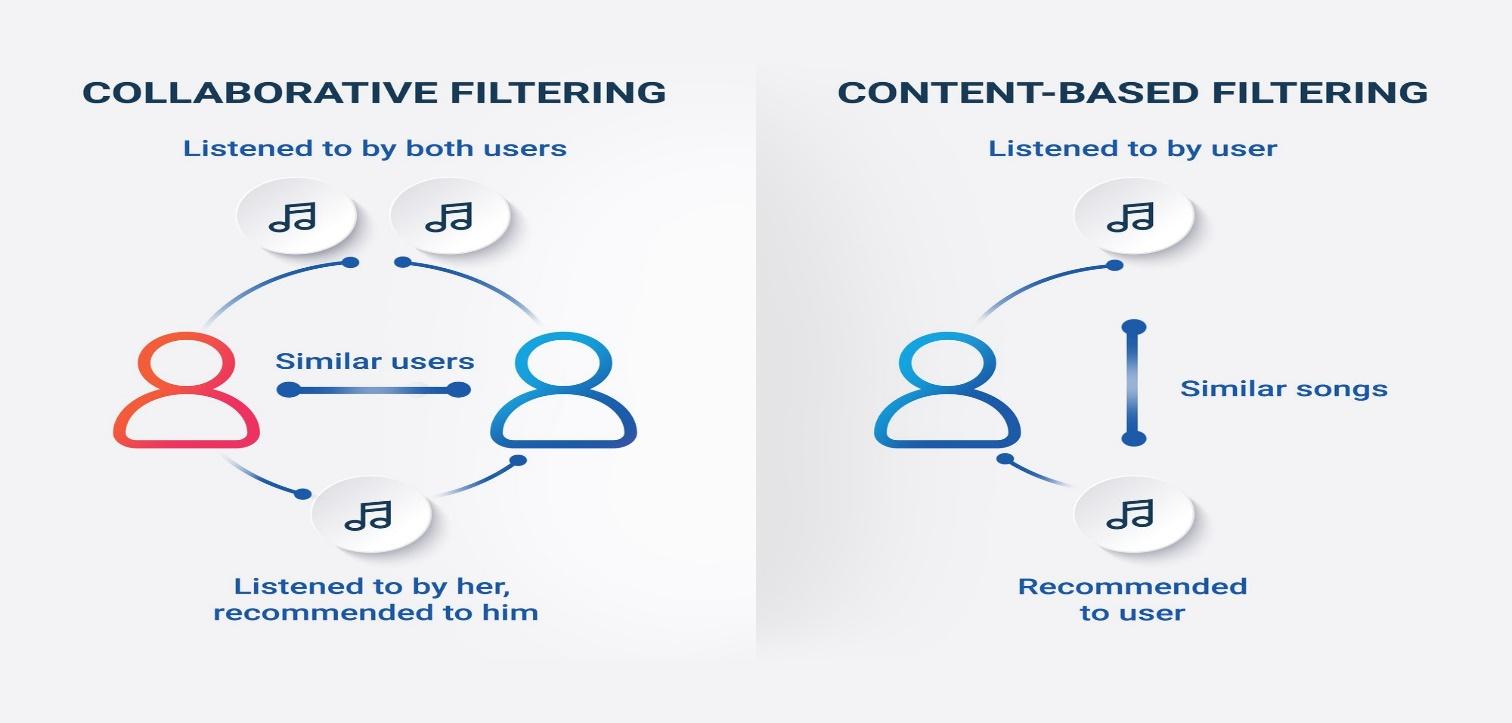
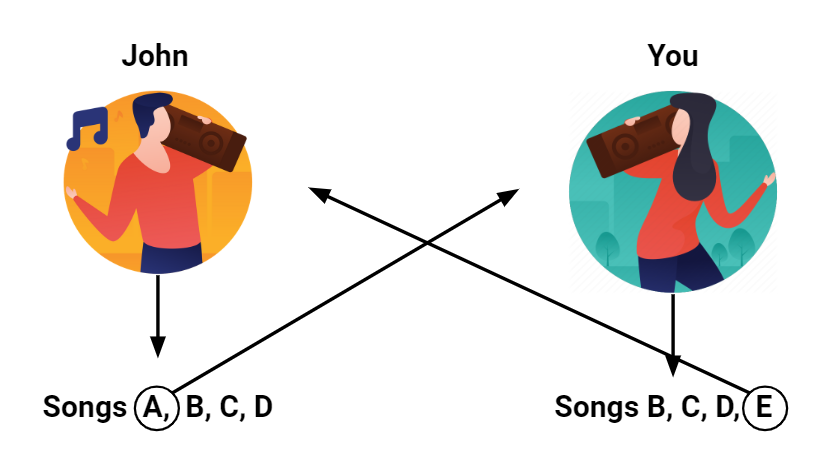
During this phase, machine learning models are optimized for maximum efficiency. To improve predicted accuracy and efficiency, it involves fine-tuning hyperparameters, comparing performance metrics, and defending the choice of the final model. We are employing Google's pre-trained model, Gemini Pro 1.5, which has been fine-tuned and optimized by Google's cutting-edge artificial intelligence technologies. Modern technology is used by this model to provide extremely accurate forecasts without the requirement for human hyperparameter adjustment. Because of Google's optimization, the model operates successfully and efficiently right out of the box, saving us time when it comes to customizing it for our unique use cases.

* **Hyperparameter Tuning** Model Gemini Pro 1.5 tuned with Temperature = 0.9 got Accuracy = 99.7%
* **Performance Metrics Comparison** Model Gemini Pro 1.5 got Accuracy of 99.7% with Temperature Set to 0.9
* **Final Model Selection Justification** Since Gemini Pro 1.5 is a powerful generative AI model created and tuned by Google, it was selected for its higher performance and dependability. This model is quite appropriate for our project requirements because it is great at deciphering intricate patterns and producing precise forecasts. Its pre-tuned nature also eliminates the need for laborious manual hyperparameter tweaking, enabling us to take use of Google's model optimization expertise and deploy the model rapidly and effectively.

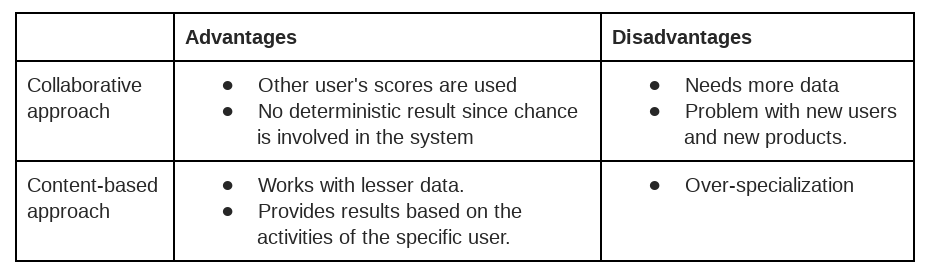
**Different types of recommendation systems:**



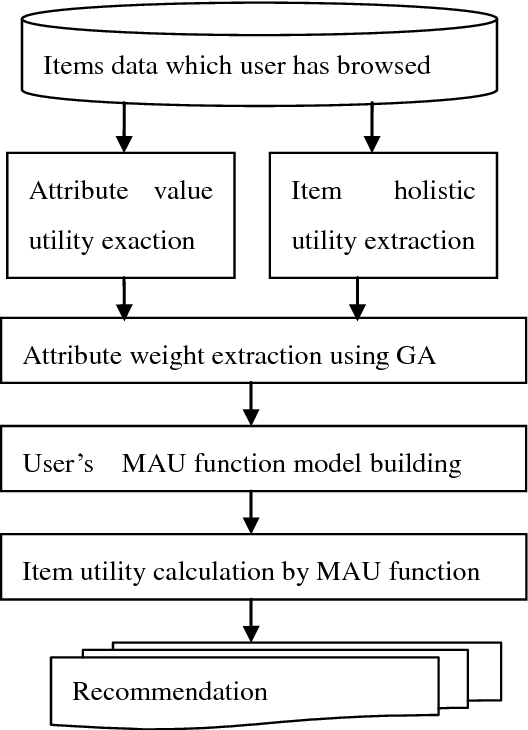
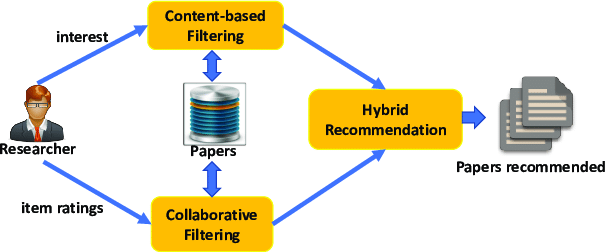
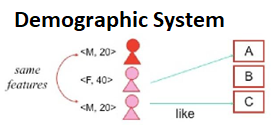
### **Collaborative Recommender system:**Collaborative recommender systems aggregate ratings or recommendations of objects, recognize commonalities between the users on the basis of their ratings, and generate new recommendations based on inter-user comparisons. The greatest strength of collaborative techniques is that they are completely independent of any machine-readable representation of the objects , work well for complex objects where variations in taste are responsible for much of the variation in preferences.



### **Content-based recommender system:** Objects are mainly defined by their associated features. A content-based recommender learns a profile of the new user’s interests based on the features present, in objects the user has rated. It’s basically a keyword specific recommender system here keywords are used to describe the items.



### **Demographic based recommender system:** This system aims to categorize the users based on [attributes](https://www.engati.com/glossary/attributes) and make recommendations based on demographic classes. Many industries have taken this kind of approach as it’s not that complex and easy to implement. In Demographic-based recommender system the algorithms first need a proper market research in the specified region.



**Utility based recommender system:** It makes suggestions based on computation of the utility of each object for the user. Of course, the central problem for this type of system is how to create a utility for individual users. In utility-based system, every industry will have a different technique for arriving at a user specific utility function and applying it to the objects under consideration.

**Knowledge based recommender system:** This type of recommender system attempts to suggest objects based on inferences about a user’s needs and preferences. Knowledge based recommendation works on functional knowledge: they have knowledge about how a particular item meets a particular user need, and can therefore reason about the relationship between a need and a possible recommendation.

**Hybrid recommender system:** Combining any of the two systems in a manner that suits a particular industry is known as Hybrid Recommender system. This is the most sought-after approach.

**Chapter 2**

**Methodology**

The methodology for building this system is fairly simple, an individual can follow

all the activities listed below:

* Requirements Specification
  + Create a requirements.txt file to list the required libraries.
  + Install the required libraries
* Initialization of Google API Key
  + Generate Google API Key
  + Initialize Google API Key
* Interfacing with Pre-trained Model
  + Load the Gemini Pro pre-trained model
  + Implement a function to get gemini response
  + Implement a function to read PDF content
  + Write a prompt for gemini model
* Model Deployment
  + Integrate with Web Framework
  + Host the Application

**Project Flow:**

* User interacts with the UI to enter the input.
* User input is collected from the UI and transmitted to the backend using the Google API key.
* The input is then forwarded to the Gemini Pro pre-trained model via an API call.
* The Gemini Pro pre-trained model processes the input and generates the output.
* The results are returned to the frontend for formatting and display.

**Chapter 3**

**Problem Statements**

Scenario:1

Legal Sector: In the legal sector, GeminiDecode proves invaluable by swiftly extracting and organizing multilingual legal documents. Law firms dealing with international clients benefit immensely as the solution ensures compliance with various legal standards and reduces manual processing time significantly. It enhances the management of cases by accurately handling contracts, affidavits, and other critical documents, thereby allowing legal professionals to focus on strategy and client interaction.

Scenario:2

Financial Institutions: Financial institutions, such as banks and investment firms, utilize GeminiDecode to process loan applications, financial statements, and other financial documents in various languages. This capability enables efficient handling of international clients, ensuring accurate data entry and compliance with global financial regulations. By automating the extraction of key financial data, banks can expedite loan approval processes, conduct thorough financial analysis, and improve overall customer service.

Scenario:3

Healthcare Industry: In the healthcare industry, hospitals and clinics leverage GeminiDecode to extract patient information from multilingual medical records. This functionality is crucial for providing high-quality patient care, as it ensures that healthcare providers have quick and accurate access to critical data, regardless of language barriers. By streamlining the management of patient records, medical histories, and treatment plans, GeminiDecode helps healthcare professionals deliver timely and informed care, ultimately improving patient outcomes.

**Objective and Scope**

This system made with the help of python and machine learning aims to provide users with technology that solves their queries and provides user with the solution based on Gemini Pro.

Language is a method of communication with the help of which we can speak, read and write. Natural Language Processing (NLP) is a subfield of Computer Science that deals with Artificial Intelligence (AI), which enables computers to understand and process human language.

This Project is based on Gemini Pro pre-trained model via an API call and data uploaded is compared with all others and then results are drawn.

Process is usually done in 4 phases

Collection →Storing →Analysing →Filtering

**Chapter 4**

**System Design**

|  |
| --- |
| **Programming and software Tools used** |

The backend as well as the front end of the application is handled by Python using Jupyter Notebook, Github, GoogleColab. Many modules of python along with google cloud Api are used for application.

Streamlit and Chrome (Web Browser) are used to access the application to get music recommendations.





This project will be using the following technology stack:

1. Python
2. Python libraries from

* streamlit
* streamlit\_extras
* google-generativeai
* python-dotenv
* PyPDF2
* Pillow
* langchain
* langchain\_community
* langchain\_google\_genai

1. Streamlit
2. Google API
3. Machine Learning
4. Generative AI

**Python Libraries Used:**

**PILLOW:** Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it. Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

**Streamlit:** Streamlit is an open-source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

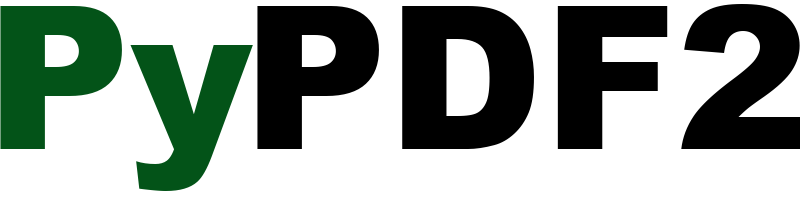
**Use *st.sidebar* to organize the input widgets for a cleaner app interface**

* **Use *st.forms* and *st.form\_submit\_button* to batch input widgets together and submit using one button**
* **Add tooltips to input widgets to provide helpful information to app users**
* **Use *st.expander* to insert a container into your app that shows extra information only when expanded**
* **Use *st.image* to add your brand logo to the app**

****

**PyPDF2:** PyPDF2 is a free and open-source pure-python PDF library capable of splitting, [merging](https://pypdf2.readthedocs.io/en/stable/user/merging-pdfs.html), [cropping, and transforming](https://pypdf2.readthedocs.io/en/stable/user/cropping-and-transforming.html) the pages of PDF files. It can also add custom data, viewing options, and [passwords](https://pypdf2.readthedocs.io/en/stable/user/encryption-decryption.html) to PDF files. PyPDF2 can [retrieve text](https://pypdf2.readthedocs.io/en/stable/user/extract-text.html) and [metadata](https://pypdf2.readthedocs.io/en/stable/user/metadata.html) from PDFs as well.

**LangChain:** LangChain is a software framework that helps facilitate the integration of large language models into applications. As a language model integration framework, LangChain's use-cases largely overlap with those of language models in general, including document analysis and summarization, chatbots, and code analysis.

**Chapter 5**

**Technology Bucket**

**Frontend Description:**

For creating frontend, we used Streamlit library. To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code. 

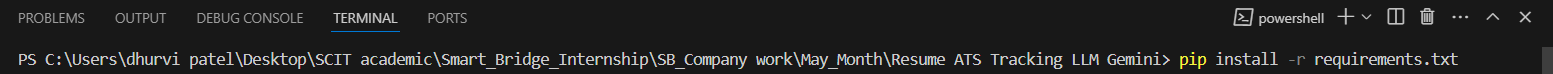
**What is Streamlit used for?**

Streamlit is an open-source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

**Backend Descreption:**

* images folder: It is established to store the images utilized in the user interface.
* .env file: It securely stores the Google API key.
* 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.
  + streamlit: Streamlit is a powerful framework for building interactive web applications with Python.
  + streamlit\_extras: Additional utilities and enhancements for Streamlit applications.
  + google-generativeai: Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
  + python-dotenv: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.
  + PyPDF2: It is a Python library for extracting text and manipulating PDF documents.
  + Pillow: Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats.
* Additionally, ensure proper file organization and adhere to best practices for version control.

**Install the required libraries**



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

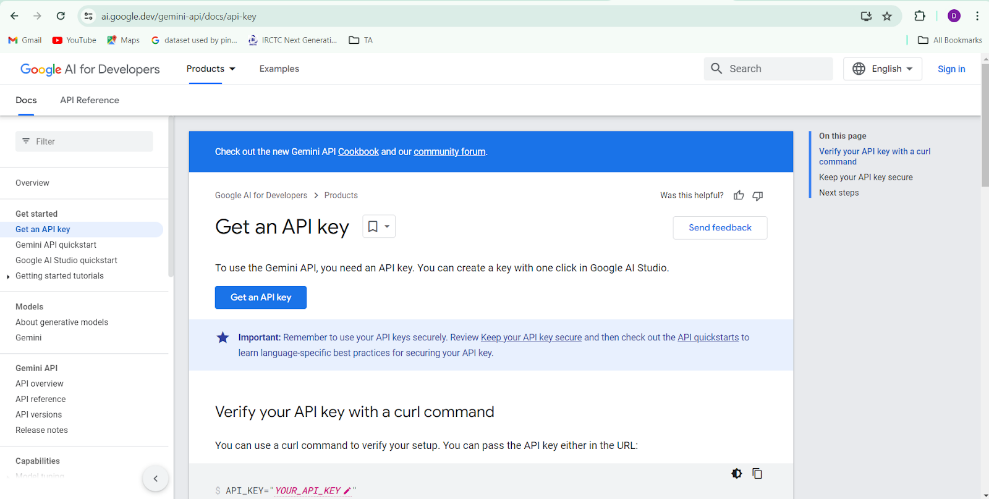
### Initialization of Google API Key

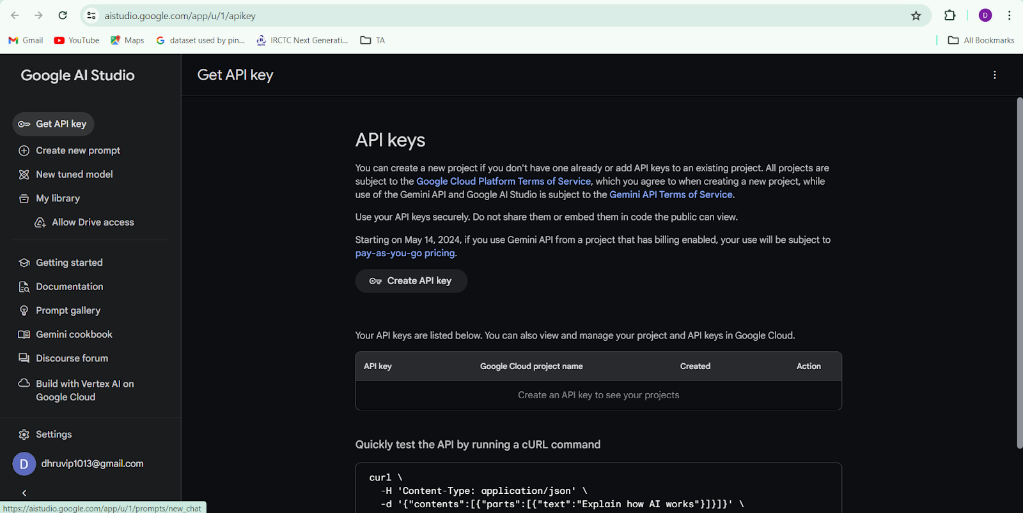
The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API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

**Generate Google API Key**

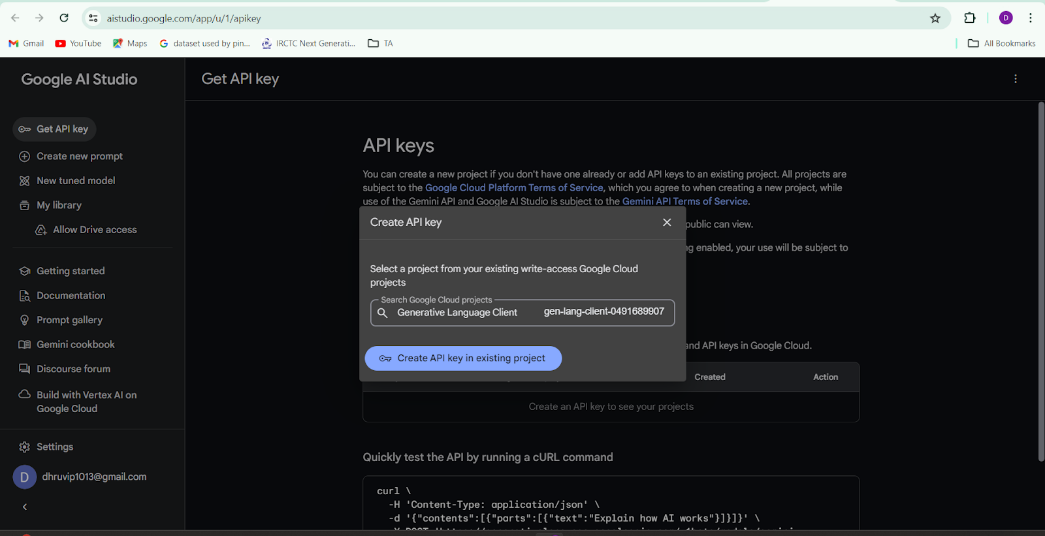
Click the provided link to access the following webpage.

Link:<https://ai.google.dev/gemini-api/docs/api-key>

[](https://ai.google.dev/gemini-api/docs/api-key)

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.

Next, click on 'Create API Key' and choose the generative language client as the project. Then, select 'Create API key in existing project'.



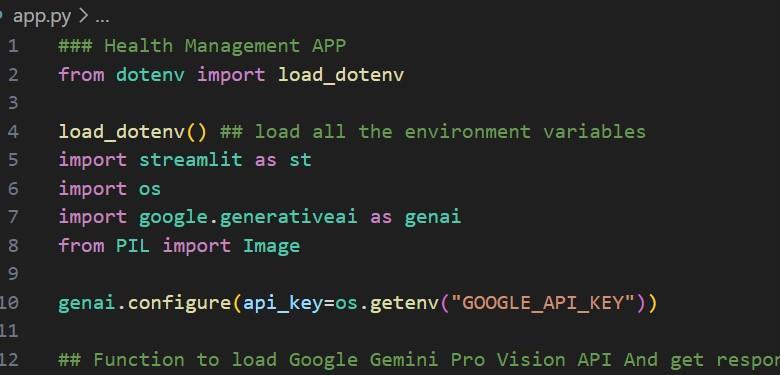
Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model.

**Initialize Google API Key**



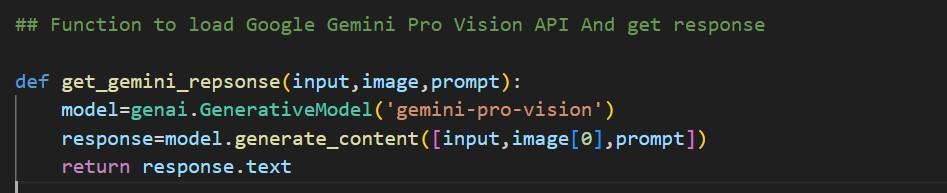
* Create a .env file and define a variable named GOOGLE\_API\_KEY.
* Assign the copied Google API key to this variable.
* Paste the API key obtained from the previous steps here.

### **Load the Gemini Pro API**



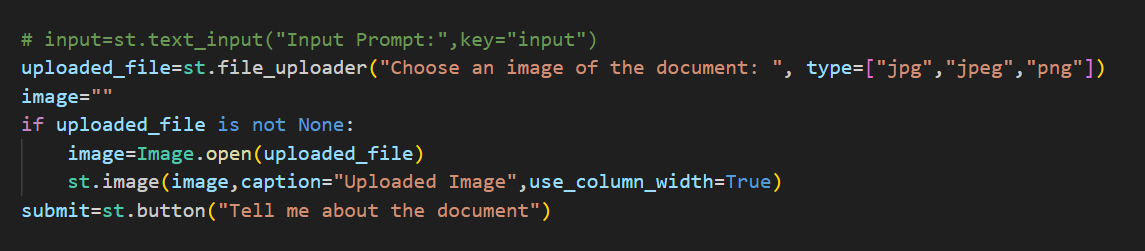
This code snippet is for initializing a health management application using Streamlit, an open-source app framework, and Google Generative AI services. The script starts by loading environment variables from a .env file using the load\_dotenv() function from the dotenv package. It then imports necessary libraries: streamlit for creating the web app interface, os for accessing environment variables, google.generativeai for utilizing Google's Generative AI capabilities, and PIL.Image for image processing. The genai.configure() function is called to set up the Google Generative AI API with the API key retrieved from the environment variables, ensuring secure and authorized access to the AI services.

**Implement a function to get gemini response**



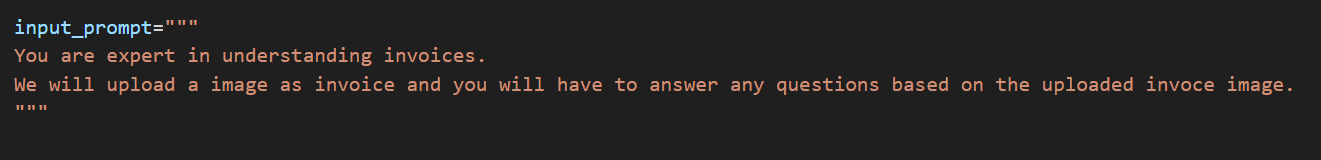
* The function get\_gemini\_response takes an input text as a parameter.
* It calls the generate\_content method of the model object to generate a response.
* The generated response is returned as text.

### **Implement a function to read the Image and set the image format for Gemini Pro model Input**



The function input\_image\_setup processes an uploaded image file for a health management application. It first checks if a file has been uploaded. If a file is present, it reads the file's content into bytes and creates a dictionary containing the file's MIME type and its byte data. This dictionary is then stored in a list named image\_parts, which is returned by the function. If no file is uploaded, the function raises a FileNotFoundError, indicating that an image file is required but not provided. This setup ensures that the uploaded image is correctly formatted and ready for further processing or analysis in the application.

**Write a prompt for gemini model**



The input prompt is designed for an expert in invoice analysis. Upon uploading an image of an invoice, the expert will answer any questions related to the content of the invoice. The prompt ensures that the expert focuses on accurately interpreting and extracting details from the uploaded invoice image. This setup is ideal for scenarios where precise and reliable information from invoices is required.

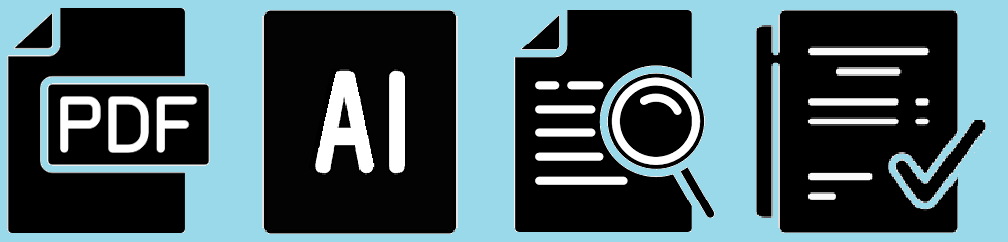
**Model Deployment**

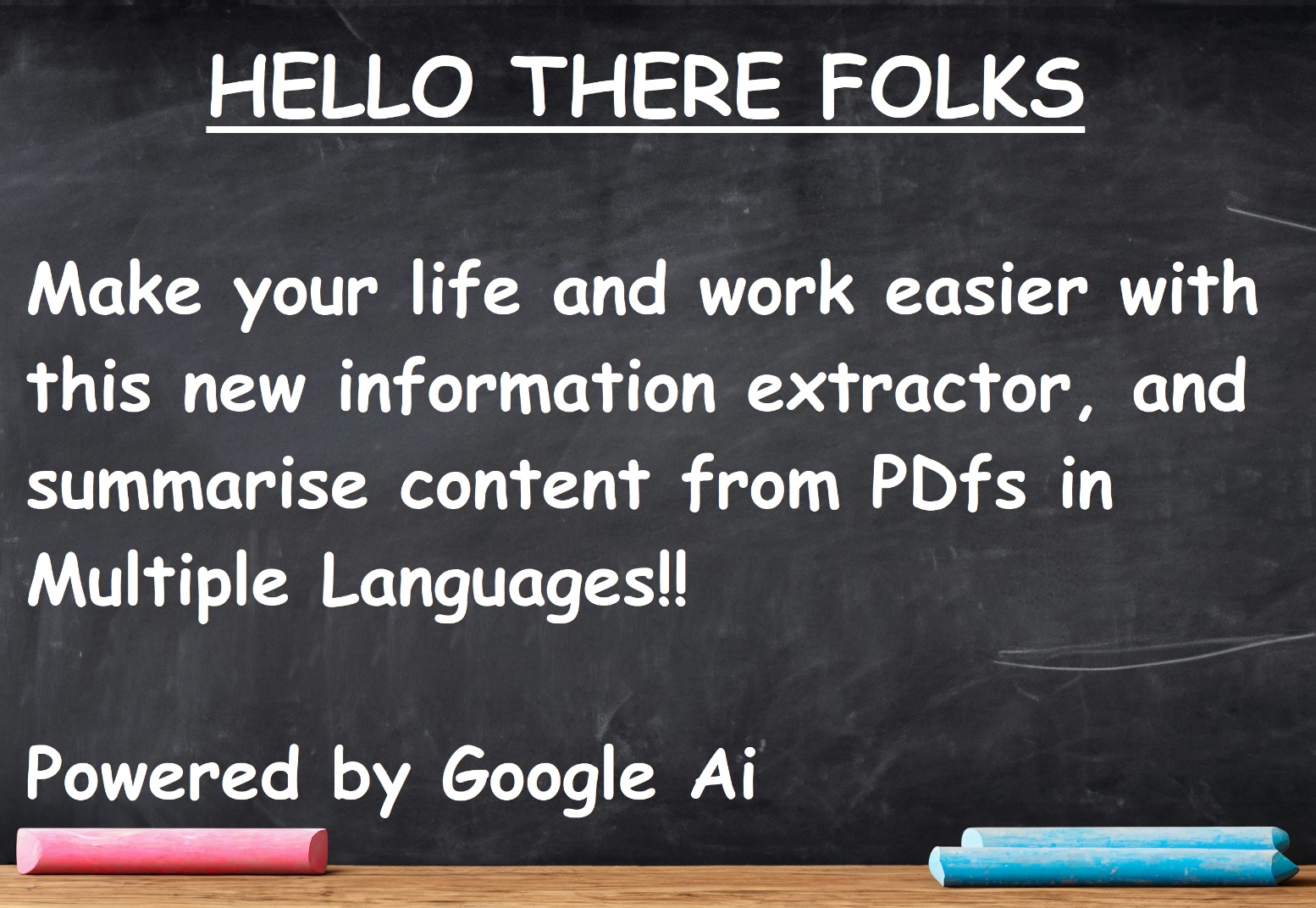
We deploy our model using the Streamlit framework, a powerful tool for building and sharing data applications quickly and easily. With Streamlit, we can create interactive web applications that allow users to interact with our models in real-time, providing an intuitive and seamless experience.

**Chapter 6**

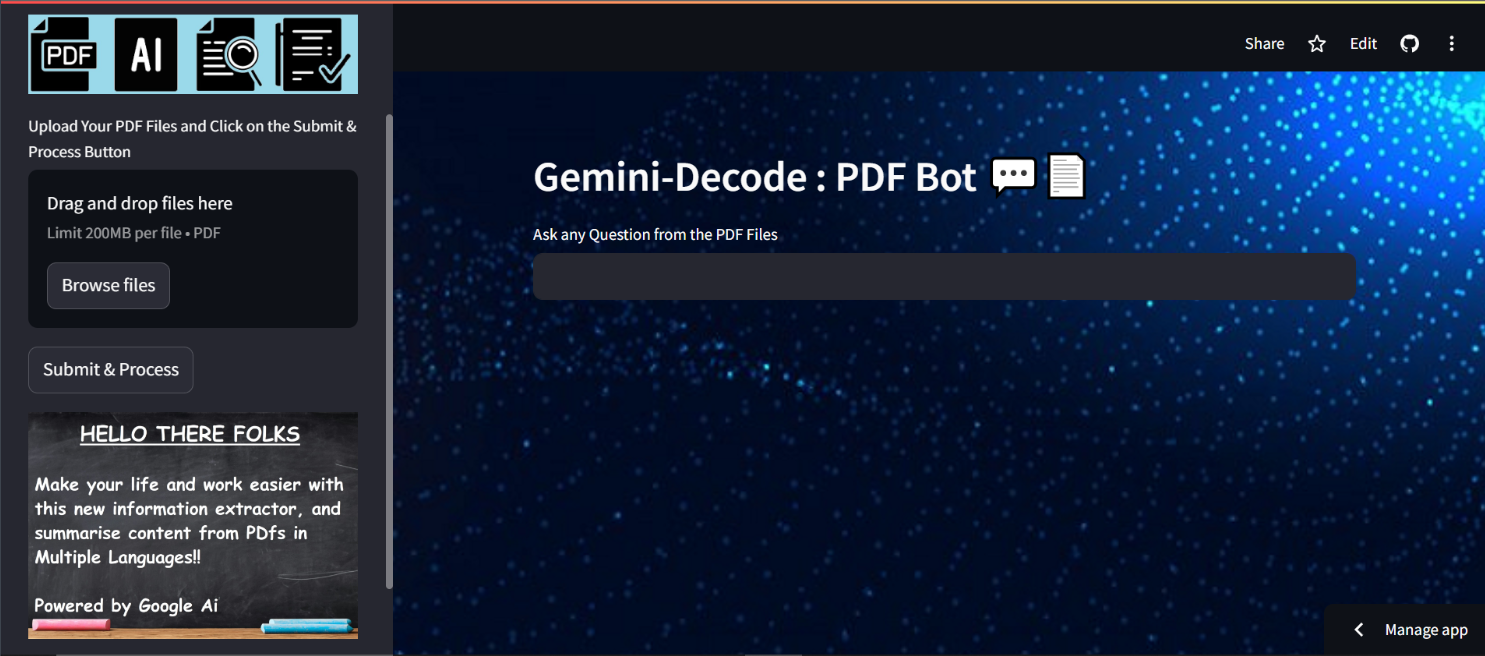
**Output Screens**

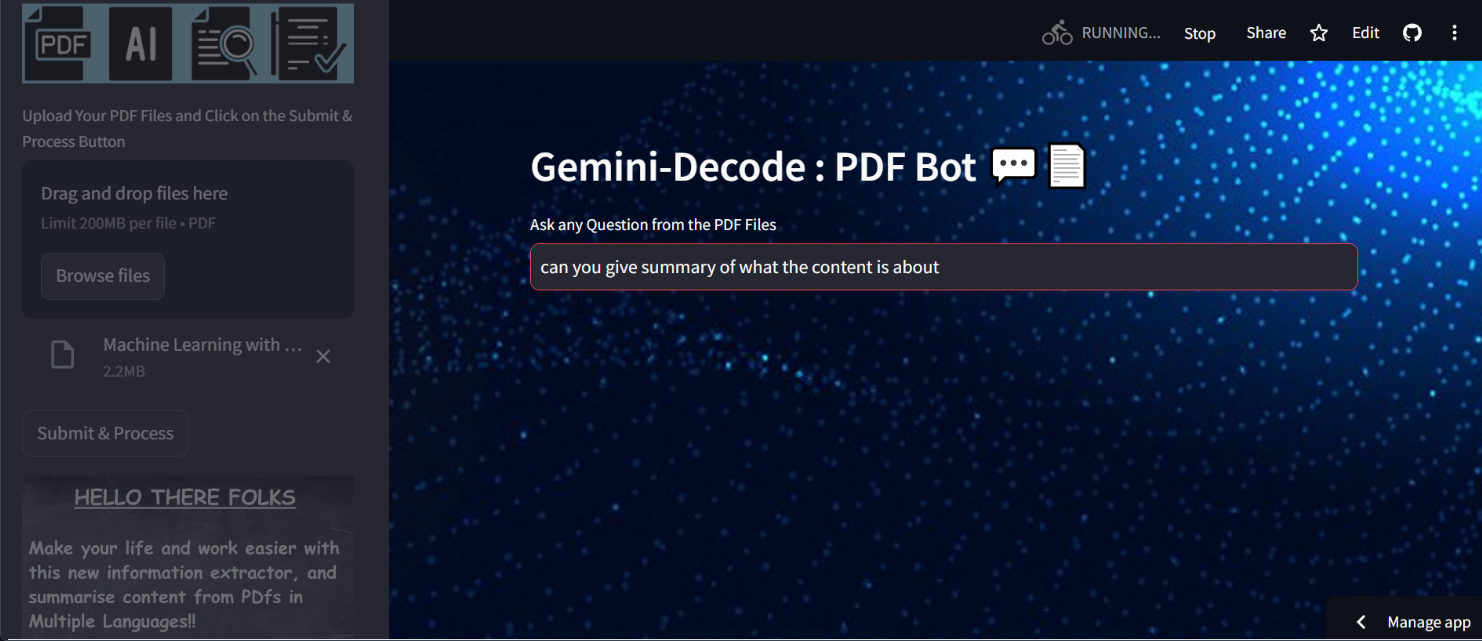
* The output of the code can be accessed by the help of chrome after running the python code with the help of Streamlit, other software programs and Google Colab Notebook.
* Any user can either upload a Document or an image.



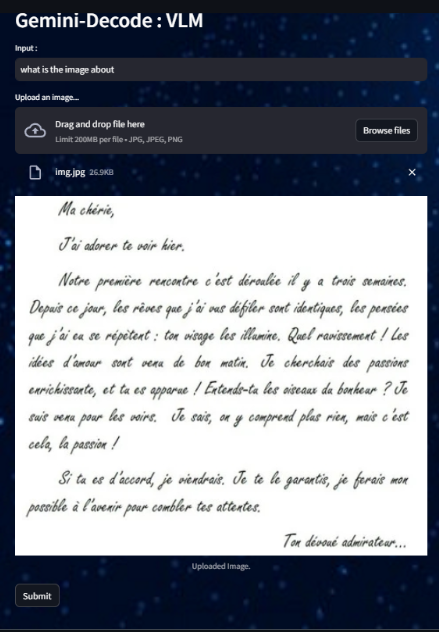
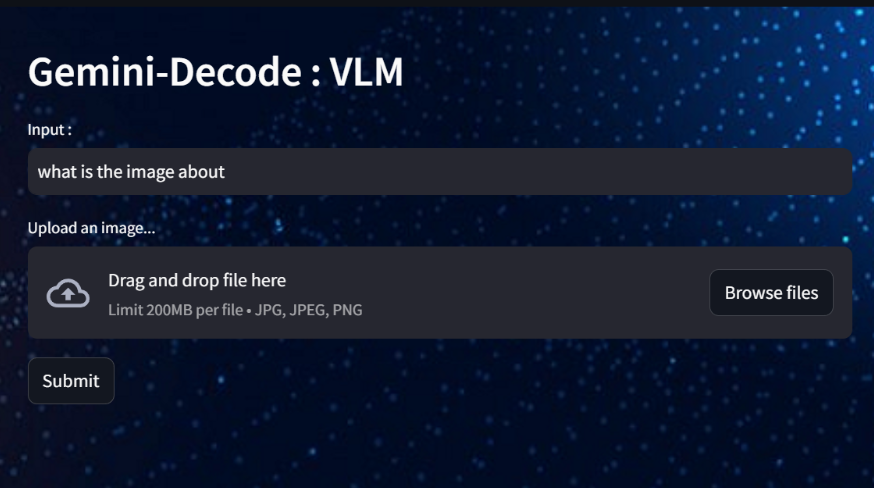






****

****

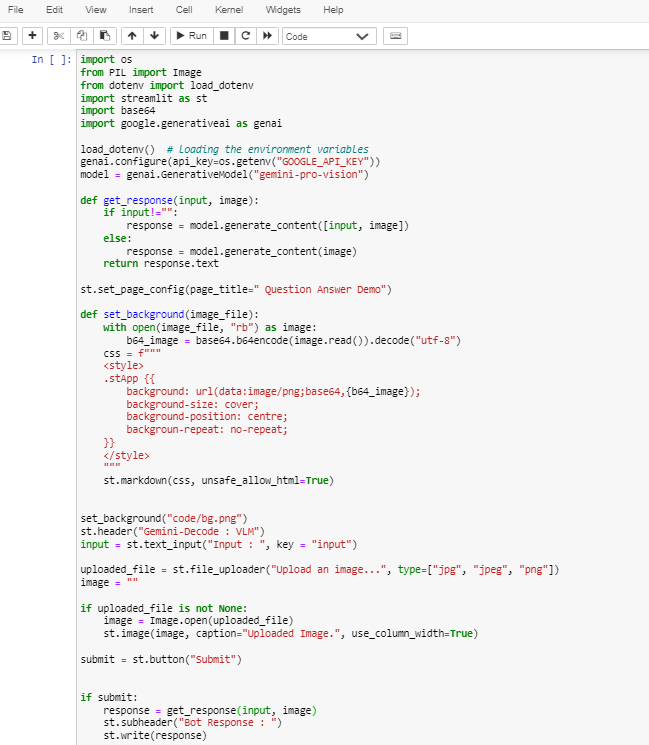
****

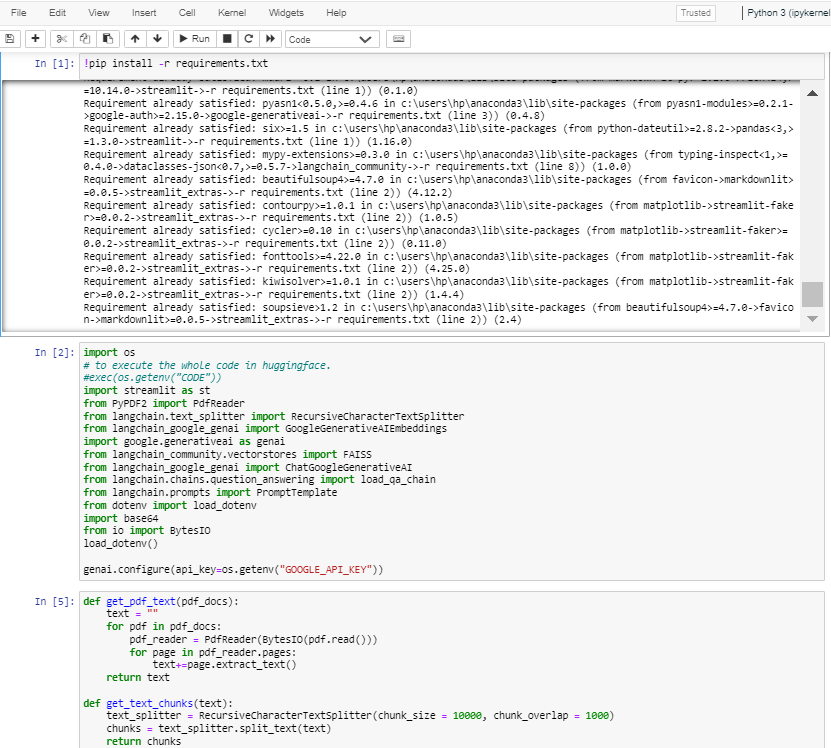
****

**APPENDICES**

**IMPLEMENTAION CODE**

**BACKEND:**





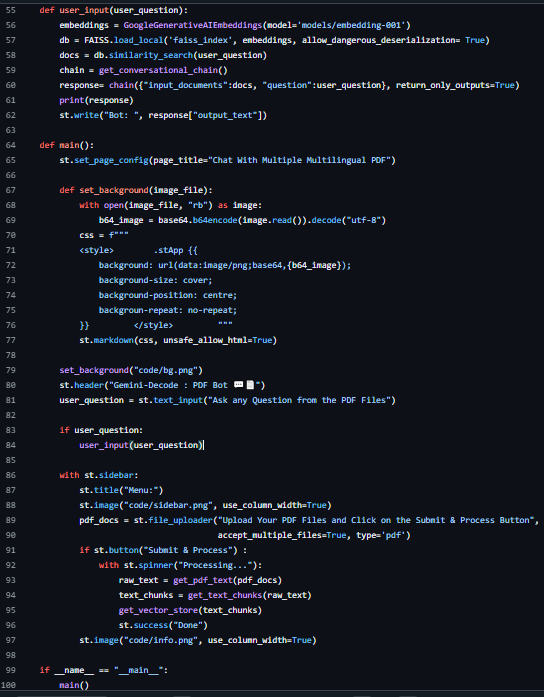
****

**App.py**

****

****

**GeminiDecode.py**

****

**REFERENCES**

* <https://jupyter.org/>
* <https://streamlit.io/>
* NLP: <https://www.tutorialspoint.com/natural_language_processing/index.htm>
* Generative AI: <https://en.wikipedia.org/wiki/Generative_artificial_intelligence>
* About Gemini: <https://deepmind.google/technologies/gemini/#introduction>
* Gemini API: <https://ai.google.dev/gemini-api/docs/get-started/python>
* Gemini Demo: <https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb>
* Streamlit: <https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/>