**Documentation for Gemini-Decode Application**

**Overview**

The **Gemini Decode** application is a Streamlit-based tool designed to extract information from multilingual documents. Utilizing the Gemini Pro AI model, it allows users to upload document images and ask questions regarding their content. This application efficiently bridges language barriers, making information retrieval straightforward and accessible.

**For Creating this we need three files which are as follows:**  
A. .env   
B. app.py

c. requirements.txt  
  
In .env file you need to write your GOOGLE\_API\_KEY which can be access from the link below  
[LINK TO GET GOOGLE\_API\_KEY.](https://ai.google.dev/gemini-api/docs/api-key)

In requirement.txt file we’ll write all the necessary library of python which are as follows:

streamlit

google-generativeai

python-dotenv

langchain

PyPDF2

chromadb

faiss-cpu

streamlit-extras

Pillow

Along with that also ensure that you have python and pip instal in your system, if not then installed them first.

**For installing all the library need to open terminal and do as below:-  
Run the command: pip install -r requirements.txt**

**After this just just save the both the file .env and requirements.txt.**

**In .env file write the api key as given below:-  
GOOGLE\_API\_KEY=”your key value from the link ”**

**NOW LET’S START THE CODE STRUCTURE OF OUR GEMINI-DECODE APPLICATION:-  
  
IMPORT ALL IMPORTANT LIBRARY:**  
import os

from PIL import Image  
from dotenv import load\_dotenv  
import streamlit as st  
import base64  
import google.generativeai as genai

**Load Environment Variables**  
load\_dotenv()

api\_key = os.getenv("GOOGLE\_API\_KEY")

print(f" Loaded API Key: {api\_key}")

**Configure Generative AI Model**

genai.configure(api\_key=api\_key)

model = genai.GenerativeModel('gemini-1.5-flash')

**Define Helper Function**

def get\_response(input\_text, image):

if input\_text != "":

response = model.generate\_content([input\_text, image])

else:

response = model.generate\_content(image)

return response.text

**Initialize Streamlit Application**

st.set\_page\_config(page\_title="Gemini Decode: Multilanguage Document Extraction by Gemini Pro")

st.header("Gemini Decode: Multilanguage Document Extraction by Gemini Pro")

**User Input Section**

input\_prompt = st.text\_input("Input:", key="input")

uploaded\_file = st.file\_uploader("Choose an image of the document:", type=["jpg", "jpeg", "png"])

image = None

**Image Display**

if uploaded\_file is not None:

image = Image.open(uploaded\_file)  
 st.image(image, caption="Uploaded Image", use\_column\_width=True)

**Submit Button**

submit = st.button("Submit")

**Response Display**

if submit and image is not None:

response = get\_response(input\_prompt, image) # Pass input\_prompt and image

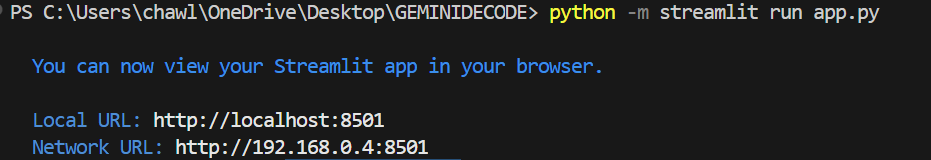
st.subheader("Bot Response:")

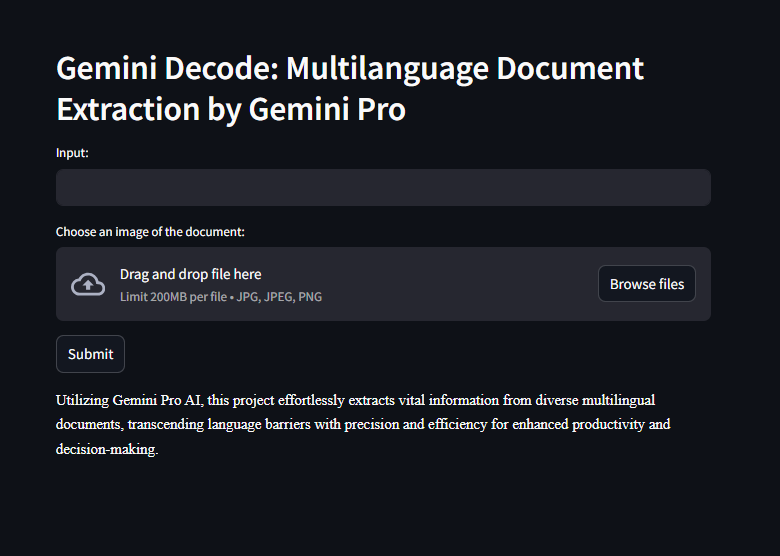
st.write(response)

**Running the Application**

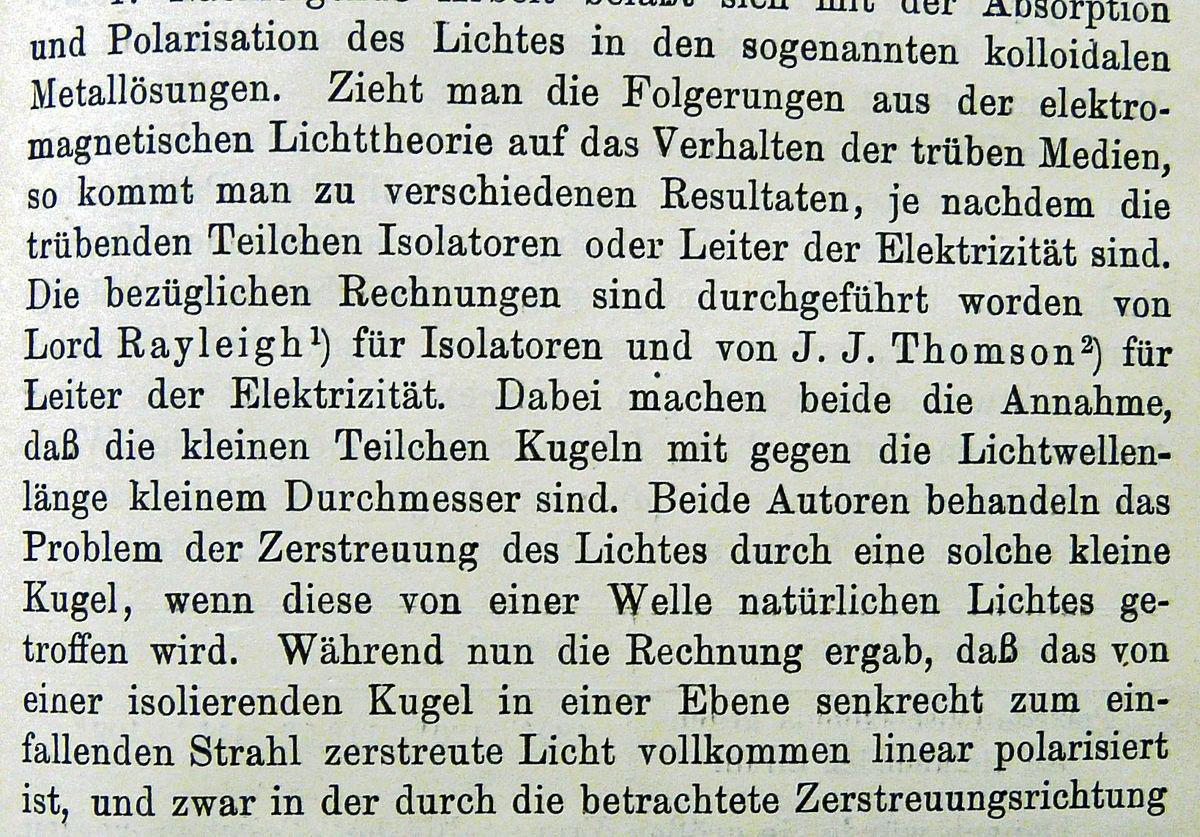
streamlit run app.py

NOW AS THE APPLICATION START IT WILL BE ON THE LOCALHOST OR LOCALSERVER   
YOU WILL GET THESE TWO LINK IN THE TERMINAL:

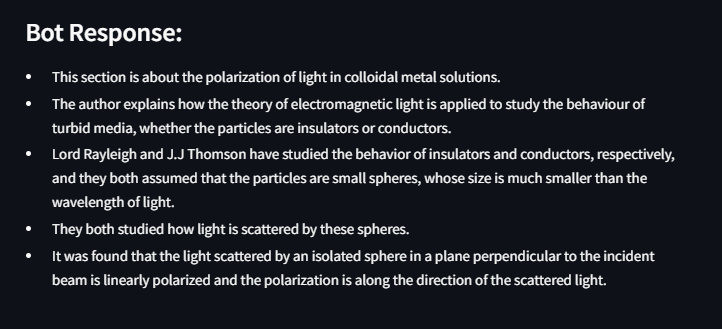


**On clicking the Local URL, you will be able to use our Gemini Decode Application  
It will look like this:**  


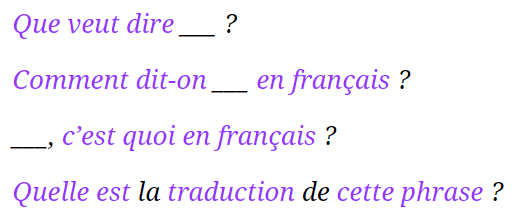
**Now Choose an image as we will choose two images in which one is of german and another image is of french   
Then in input field write : Translate the image into english (complete) in pointwise  
For both image use same input and then you will get output after clicking on submit button**

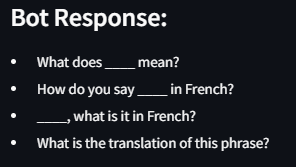
**a.For German**   
 *The input is*   
 

*And its output will be as :*



b. For French  
 The input is :



The output will be as :   
 

And that’s how this application is work .  
  
**Conclusion:**

**The Gemini Decode application provides an intuitive interface for extracting information from documents using AI. With its ability to handle multilingual content, it is an invaluable tool for enhancing productivity and decision-making.**