GeminiCare

Gemini powered App to assist Autism Care

Overview

The primary goal is to develop a Multimodal Autism Monitoring & Analysis System that leverages Gemini AI to assist medical professionals by providing an efficient method of diagnosing and monitoring autistic patients. This application will analyze multimodal data inputs - video recordings, speech patterns, and textual information from patient interactions to detect and predict symptom patterns and severity, aiding medical professionals in making informed treatment decisions.

Development Prerequisites

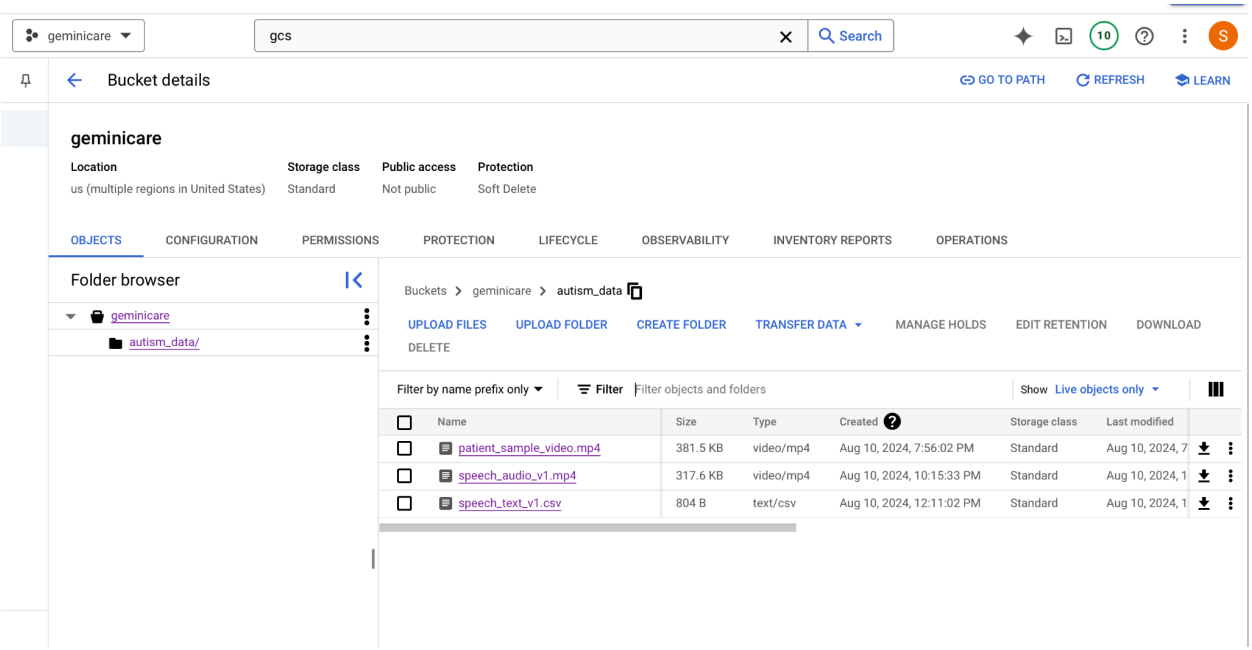
- Cloud Environment: GCP with access to Vertex AI, GCS, VM
- App Framework: Streamlit
- Python Library: streamlit, streamlit-modal, google-cloud-aiplatform

Development Steps

- Setup Keys in the Environment
    - This won't be required if the Application is running in the GCP VM which has permission to access the GCS and Vertex AI through the service account.
    - It would be required to be specified in the .env file or copied into a specific folder if the Application is running locally or in a VM which is not attached to the service account.

- Store Sample Dataset in GCS Bucket

  The uploaded files will be added to GCS Bucket. Following screenshot shows the sample files.



- Use Gemini Model for analyzing the content

  Model: gemini-1.5-flash-001

```
project_id="geminicare"
vertexai.init(project=project_id, location="us-central1")
model = GenerativeModel("gemini-1.5-flash-001")
# Set model parameters
```

```
generation_config = GenerationConfig(
    temperature=0,
    top_p=0,
    max_output_tokens=8192,
)

safety_settings = [
    SafetySetting(

category=SafetySetting.HarmCategory.HARM_CATEGORY_HATE_SPEECH,

threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
    ),
    SafetySetting(

category=SafetySetting.HarmCategory.HARM_CATEGORY_DANGEROUS_CO
NTENT,

threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
    ),
    SafetySetting(

category=SafetySetting.HarmCategory.HARM_CATEGORY_SEXUALLY_EXPLI
CIT,

threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
    ),
    SafetySetting(

category=SafetySetting.HarmCategory.HARM_CATEGORY_HARASSMENT,

threshold=SafetySetting.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE
    ),
]
```

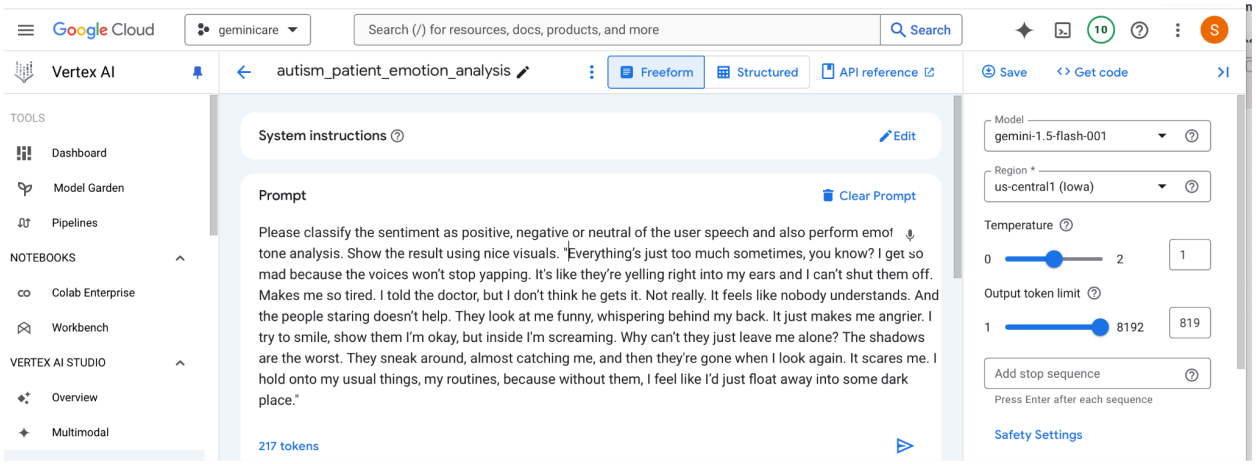
Text Analysis Prompts:
Examples of Chaining Prompts

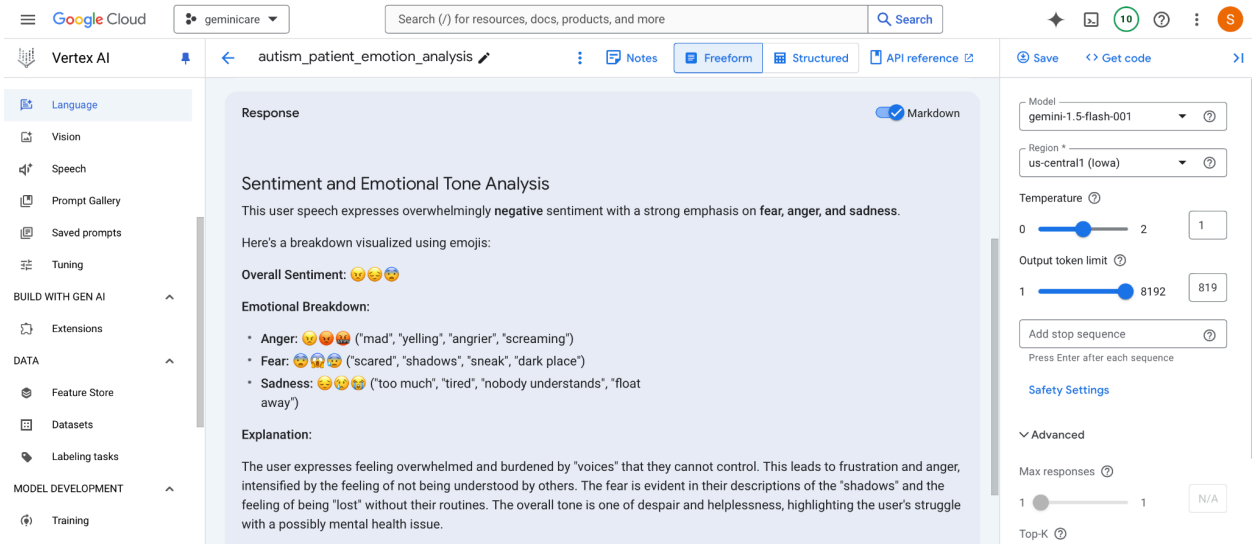
```
sentiment_analysis_prompt1 = "Please classify the sentiment as positive,
negative or neutral of the text provided at the end and construct the sentence
properly for this text:\n\n"+str(patient_text)
```

```
response1a =
model.generate_content(sentiment_analysis_prompt1,generation_config=genera
tion_config, safety_settings=safety_settings)
sentiment_text = response1a.text

sentiment_analysis_prompt2 = "Perform sentiment analysis and Generate html
markup code with a table header 'Sentiment Analysis' and display the name of
the sentiment using proper color ('Negative' in red, 'Positive' in green, 'Neutral' in
blue)  and show explanation in a separate big text area for the given text and
don't show the given text, only provide generated html code: \n\n" +
str(sentiment_text)

response1b =
model.generate_content(sentiment_analysis_prompt2,generation_config=genera
tion_config, safety_settings=safety_settings)
classified_sentiment = response1b.text.replace("```","").replace('html',"")

tone_analysis_prompt1 = "Please find all the different tones and Create a JSON
structure with  'tone' and 'score' where score is between 1 to 100 for this text:
\n\n" + str(patient_text)

response2a =
model.generate_content(tone_analysis_prompt1,generation_config=generation_
config, safety_settings=safety_settings)
emotional_text = response2a.text

tone_analysis_prompt2 = "Generate html markup code to display the result in a
html table with columns 'tone' and 'score' (which you need to load from the json
structure) with header 'Tone Analysis' and show an additional column
'Explanation' and don't show the given text, only provide generated html code:
\n\n" + str(emotional_text)

response2b =
model.generate_content(tone_analysis_prompt2,generation_config=generation_
config, safety_settings=safety_settings)
emotional_tones = response2b.text.replace("```","").replace('html',"")

keyword_prompt1 = "Based on the given text, find the key intents in maximum 5
keywords: \n\n" + str(patient_text)
response3a =
model.generate_content(keyword_prompt1,generation_config=generation_config
, safety_settings=safety_settings)
intent_text = response3a.text
```

```
keyword_prompt2 = "Generate html markup code to display the result in a html table with columns 'Intent Keyword' and 'Explanation' with table header 'Keyword Analysis' and don't show the given text, only provide generated html code: \n\n" + str(intent_text)

response3b = model.generate_content(keyword_prompt2,generation_config=generation_config, safety_settings=safety_settings)
intent_keywords = response3b.text.replace("```","").replace('html',"")

 # Append the insights result into a list.
 sentiment_content_list.append([patient_text, classified_sentiment, emotional_tones, intent_keywords])
```

- Test Prompts for Sample Dataset in Vertex AI Notebook



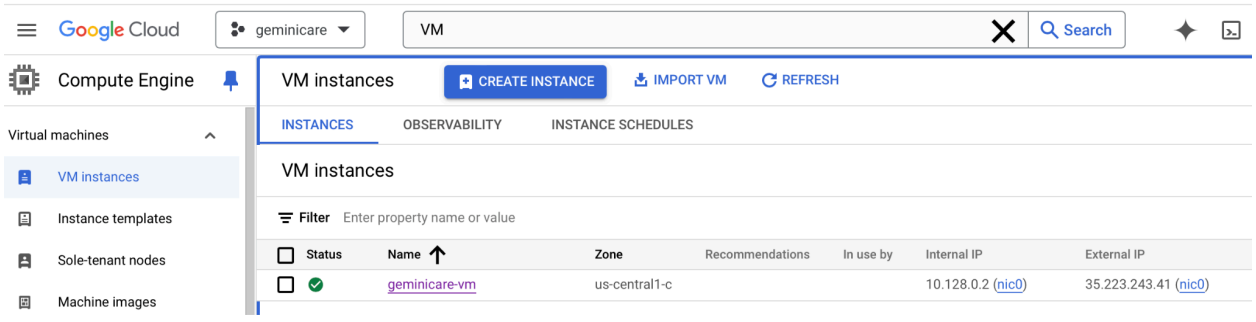


Code Repository

- Github: https://github.com/srijonmandal1/geminicare-autism-treatment

App Deployment

Create a VM in GCP



Copy the local code to GCP VM

gcloud compute scp --project="geminicare" --zone="us-central1-c” --recurse ~/Documents/GitHub/autism-care srijonmandal328@geminicare-vm:~/genai/

Open the Streamlit Port

gcloud compute --project="geminicare" firewall-rules create firewall-rules --direction=INGRESS --priority=1000 --network=default --action=ALLOW --rules=tcp:8501 --source-ranges=0.0.0.0/0

Create Virtual Env in VM

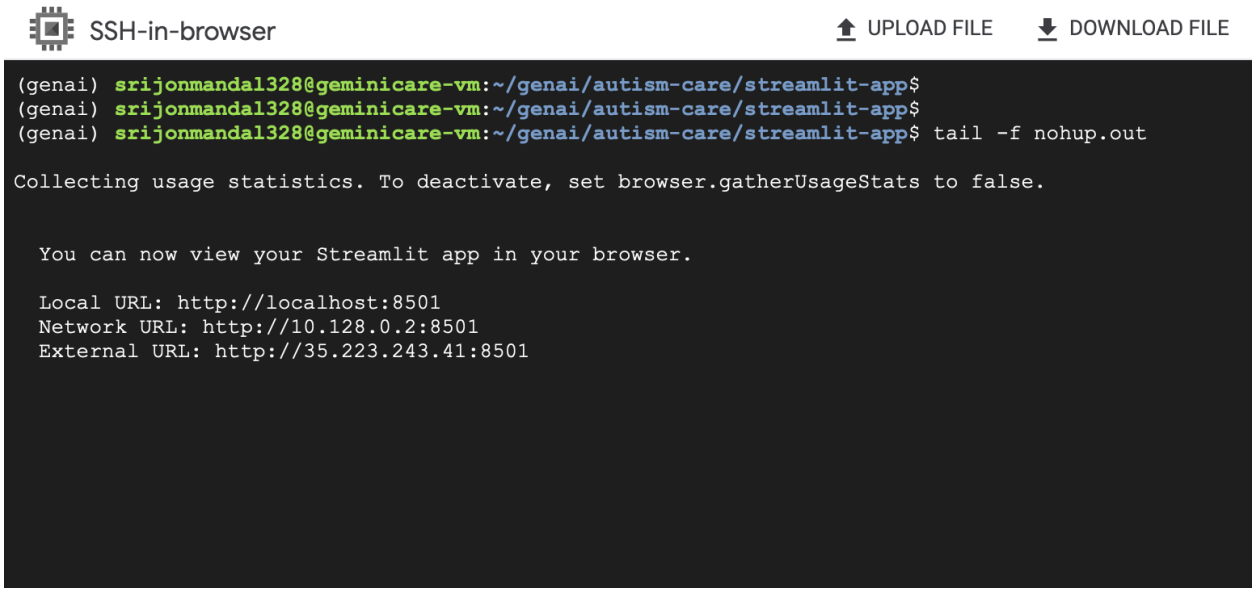
conda create -n genai

conda activate genai

pip install streamlit
pip install streamlit-modal
pip install google-cloud-aiplatform

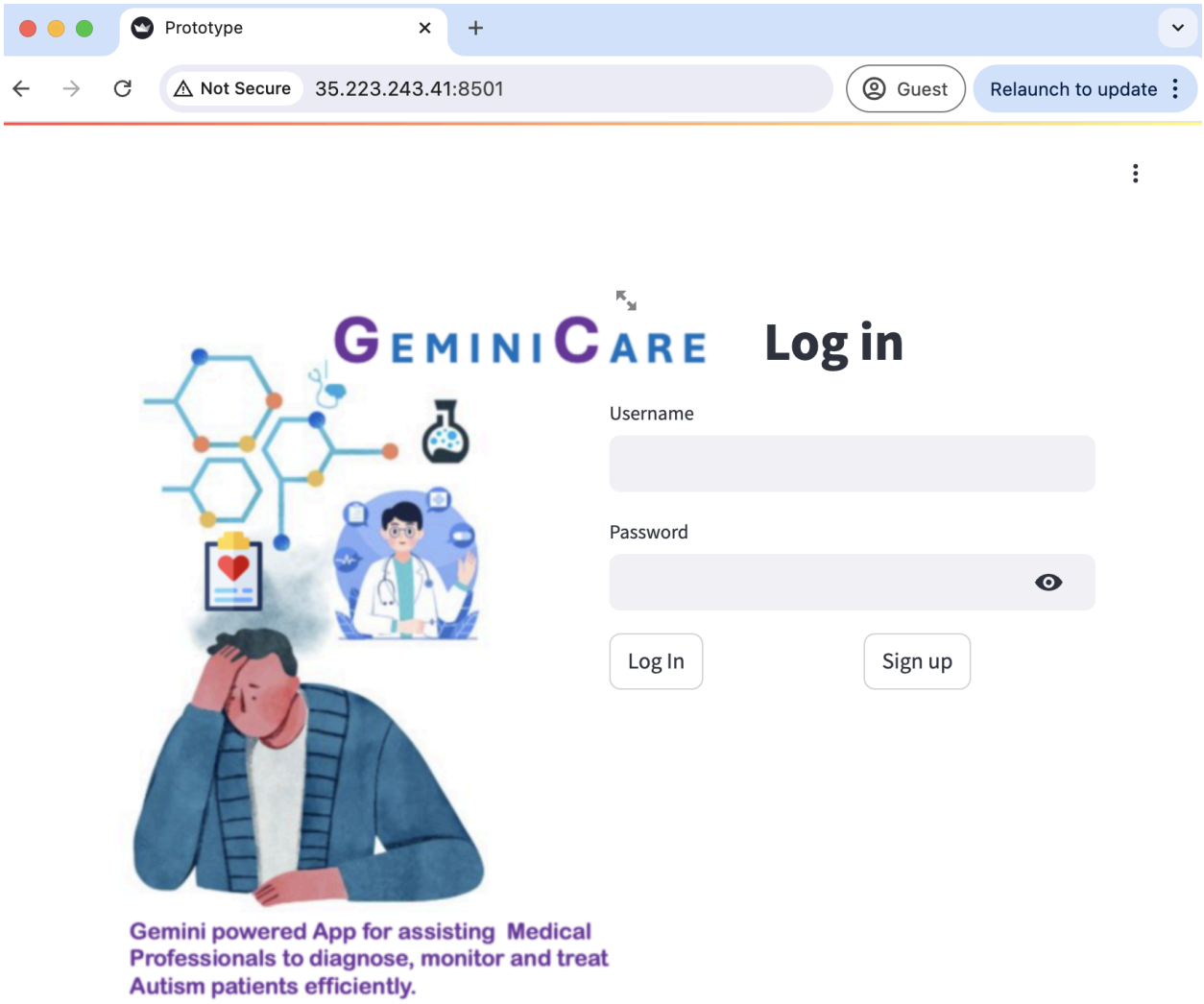
App Execution

nohup streamlit run Prototype.py &



App Usage

The App will now run at http://35.223.243.41:8501/



ToDo

- Need to use a better UI framework (React / Angular) which provides a robust user login module.
- Need to provide secure user authentication and authorization.
- Need to store properties (like gcp_project name, env etc.) in a yaml file
- Need to maintain a Dockerfile and requirements.txt
- Need to containerize and deploy the app into Cloud Run App Engine to ensure fail-over
- Need to encrypt and anonymize sensitive user details
- Need to fix the issues related storing the uploaded files into GCS bucket seamlessly

- Need to capture all the analysis actions into a database table in order to show a trend of metrics
- Need to keep the reports ready for the Clinician and Researchers to access
- Need to implement the RAG-based Chatbot which can allow Q&A based on the knowledgebase and patient diagnosis details.