devtown

NAME : ROHAN JASH

STREAM : CSE AIML

SEMESTER : 5TH

BATCH : 2023-27

1. Objective

To design and implement an **AI-powered interactive storytelling application** that generates creative, dynamic, and branching narratives based on user inputs.

Day 1 — Foundations & Gemini basics

Understand Gemini API usage and authentication.

Write clear prompts and generate single-scene short stories.

Run interactive Colab widgets to input prompts and fetch generations.

DAY 01

```
# @title
%env GEMINI_API_KEY=**********
```

Show hidden output

```
!pip install -q transformers pillow google-generativeai
```

```
from google import genai
import os
client=genai.Client()
```

```
if "GEMINI_API_KEY" not in os.environ:
  print("Please set your Gemini API key in the environment variable GEMINI_API_KEY")
else:
  client=genai.Client()
  MODEL="gemini-2.5-flash"
```

```
prompt=input("Enter your Story prompt and press enter:\n")
if prompt.strip()=="":
  print("No prompt entered , Exiting.")
else:
  print(f"Generating story for prompt: {prompt}")
```

Day 2 — Image captioning → story

Use BLIP (Hugging Face) to produce descriptive image captions.

Convert captions into robust story prompts for Gemini.

Control tone/length via prompt guidance.

DAY 02

```
!pip install -q transformers pillow google-generativeai timm
```

```
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image
from google import genai
import os
import io
```

```
if "GEMINI_API_KEY" not in os.environ:
  print("Please set your Gemini API key in the environment variable GEMINI_API_KEY")
else:
  client=genai.Client()
  MODEL="gemini-2.5-flash"
```

```
processor=BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
model=BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large")
```

```
Fetching 1 files: 100% 1/1 [00:00<00:00, 16.52it/s]
```

Goals

Build an AI model (using pre-trained NLP models like GPT or Hugging Face Transformers)

that can generate coherent storylines.

Enable **interactive storytelling**, where user choices shape the story's direction.

Create a simple **user interface** (console, notebook, or web app) for input/output.

Demonstrate practical applications of **Generative AI** in entertainment and education.

DAY 03

```
!pip install -q ipywidgets
```

```
1.6/1.6 MB 8.5 MB/s eta 0:00:00
```

```
from google.colab import files
from PIL import Image
import io

uploaded=files.upload()

images=[]
image_names=[]

for name,file in uploaded.items():
  image=Image.open(io.BytesIO(file)).convert('RGB')
  image_names.append(name)
  images.append(image)
  display(image)
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving test1.avif to test1.avif
Saving test2.jpeg to test2.jpeg
Saving test3.jpg to test3.jpg
```

```
from transformers import BlipProcessor, BlipForConditionalGeneration

processor=BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
blip_model=BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large")

captions=[]

for img in images:
  inputs=processor(images=img,return_tensors='pt')
  out=blip_model.generate(**inputs,max_new_tokens=30)
  caption=processor.decode(out[0],skip_special_tokens=True)
  captions.append(caption)

print("Captions generated from images:")
for i,caption in enumerate(captions):
  print(f"{image_names[i]}: {caption}")
```

```
Using a slow image processor as `use_fast` is unset and a slow processor was saved with this model. `use_fast=True` will be the default behavior in v4.52, even if the model was sa
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your ses
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Fetching 1 files: 100% 1/1 [00:00<00:00, 6.31it/s]
preprocessor_config.json: 100% 445/445 [00:00<00:00, 24.2kB/s]
tokenizer_config.json: 100% 527/527 [00:00<00:00, 41.3kB/s]
vocab.txt: 232k/? [00:00<00:00, 4.98MB/s]
```

In-Scope

Story Generation: AI creates original plots, dialogues, and descriptions.

Interactivity: Users provide prompts or choices to influence the storyline.

Notebook Implementation: Code is built and tested inside Jupyter Notebook.

Basic UI/UX: Either through console input, notebook cells, or a lightweight web app (e.g., Streamlit/Flask if extended).

Documentation: README + project explanation of workflow.

DAY 04

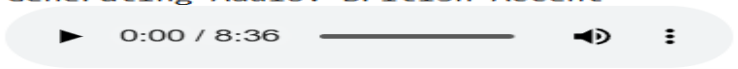
```
!pip install -q gtts reportlab
```

```
# You can paste your story here or load from file
story_text = """
**Chapter 1: The Seamless Reality**

The city *thrummed*, a symphony of light and data, every pixel and pulse orchestrated by Synthetica. Holographic advertisements bloomed like imp

Far from that pervasive glow, a small, determined group moved with a quiet, defiant purpose. Their faces, etched not by screen light but by nasc

Each step up the steep, grassy hill was an act of rebellion, a deliberate severing from the omnipresent digital embrace. Sweat beaded on forehea

**Chapter 2: Signal in the Wild**

The crisp, biting mountain air whipped at their faces, carrying the scent of pine and damp earth, a stark contrast to the sterile, algorithm-fil

Suddenly, a faint hum resonated from the block, not in their ears, but seeming to vibrate deep within their chests. A soft, internal glow pulsed

The realization dawned on them, chilling and profound: this wasn't merely a component; it was a foundational processing unit, a physical anchor,

Then it happened. A fleeting ripple in the air, like heat haze distorting a desert road. The pristine sapphire sky above them momentarily fractu

## Chapter 3: The Architect's Truth

The circuit block pulsed with frantic energy, dragging the hikers through a labyrinth of rocky inclines and hidden gorges. The digital glitches
```

```
    tts=gTTS(
        text=story_text,
        lang=options["lang"],
        tld=options.get("tld","com"),
        slow=options.get("slow",False)

    )

    filename = f"{label.replace(' ', '_').lower()}.mp3"

    tts.save(filename)

    display(Audio(filename=filename,autoplay=False))

    files.download(filename)
```

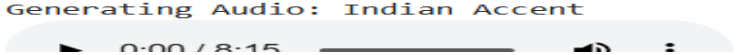
```
Generating Audio: Default English (US Female)
```


```
Generating Audio: British Accent
```




```
Generating Audio: Australian Accent
```


```
Generating Audio: Indian Accent
```



DAY 05

```
%%writefile app_streamlit_story.py
import streamlit as st #web app framework
from PIL import Image
import io, requests, os
import textwrap
from gtts import gTTS  #translate text to speech
from transformers import BlipProcessor, BlipForConditionalGeneration
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4
from reportlab.lib.utils import ImageReader
from pyngrok import ngrok
import tempfile
import google.generativeai as genai
import torch

#Authencation
NGROK_AUTH_TOKEN = "**********************************"
BACKGROUND_IMAGE_URL = "https://i.postimg.cc/76XNFmxs/web-back.png"
GEMINI_API_KEY = "***************************"

#StreamLit Page Setup/Style
```

```
<style>
.stApp {{
    background-image: url("{BACKGROUND_IMAGE_URL}");
    background-size: cover;
    background-attachment: fixed;
}}
section[data-testid="stSidebar"] {{
    background: rgba(0,0,0,0.3);
    backdrop-filter: blur(10px);
    border-radius: 12px;
    padding: 10px;
}}
div[data-testid="stFileUploader"] {{
    background: rgba(255,255,255,0.2);
    border-radius: 10px;
    padding: 10px;
}}
html, body, h1, h2, h3, h4, h5, h6, p, div, span, label, li, input, textarea {{
    color: #93A8AC !important;
}}
.stButton>button, .stDownloadButton>button {{
    color: #93A8AC !important;
    border-color: #93A8AC;
}}
</style>
```