College Name: VIT Bhopal
Student Name: Shreya Sahu 22BCG10027

GEN AI PROJECT PHASE 2 SUBMISSION DOCUMENT

Phase 2: Project Execution and Demonstration

1. Project Title:

Startup Pitch Generator using Generative AI (LoRA Fine-Tuned GPT-2)

2. Objective Recap

The objective of this project is to develop a startup pitch generation system that leverages Generative AI. We fine-tuned a **GPT-2 model using LoRA (Low-Rank Adaptation)**, enabling the model to generate startup-style pitches given a user-defined theme or problem. This application supports rapid ideation and creative writing assistance in entrepreneurial and educational contexts.

3. Technologies Used

- Python
- HuggingFace Transformers
- PEFT (Parameter-Efficient Fine-Tuning using LoRA)
- PyTorch
- Streamlit (for web interface)
- Google Colab / Jupyter Notebook
- Dataset: Custom startup pitch dataset in instruction-response format
- Pre-trained base model: GPT-2

4. Proposed Solution

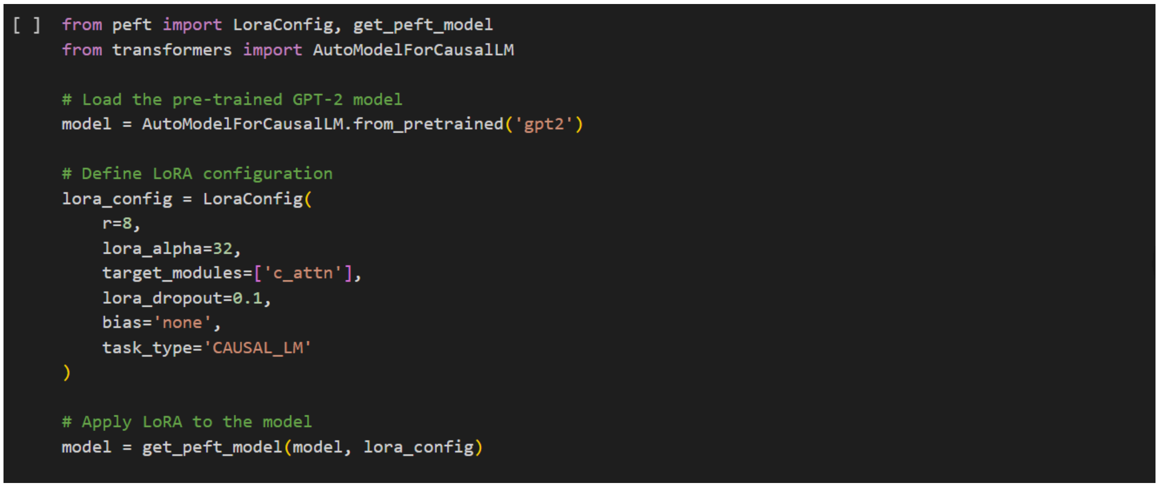
To enable context-aware, pitch-specific text generation, we fine-tuned the **GPT-2** model using **LoRA adapters** via the peft and transformers libraries.

- ***Training Process:***
  Dataset Format:
  We prepared a **JSON/JSONL** dataset with fields like:

JavaScript

```
{
  "instruction": "Generate a startup pitch for a fintech app that simplifies taxes for freelancers.",
  "output": "Introducing TaxEase — a smart fintech assistant designed for freelancers..."
}
```

- ***Model Setup:***
  - **Base Model**: `gpt2`
  - **Fine-tuning method**: LoRA using the HuggingFace PEFT library
  - Training was done on Google Colab using PyTorch



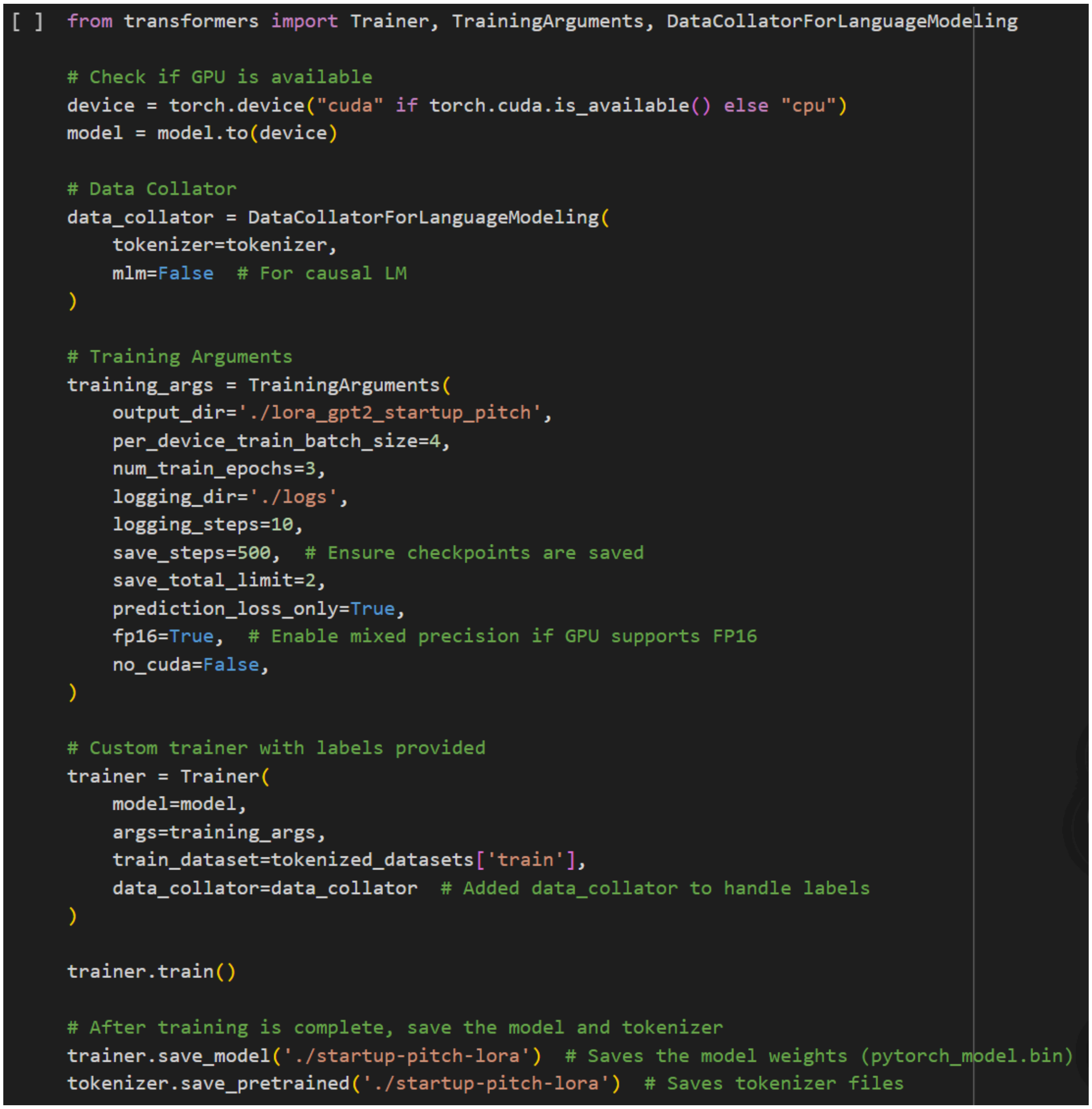
```
from peft import LoraConfig, get_peft_model
from transformers import AutoModelForCausalLM

# Load the pre-trained GPT-2 model
model = AutoModelForCausalLM.from_pretrained('gpt2')

# Define LoRA configuration
lora_config = LoraConfig(
    r=8,
    lora_alpha=32,
    target_modules=['c_attn'],
    lora_dropout=0.1,
    bias='none',
    task_type='CAUSAL_LM'
)

# Apply LoRA to the model
model = get_peft_model(model, lora_config)
```

- ***Why LoRA?***
  LoRA allows us to update only a small number of low-rank matrices during training, drastically reducing computational cost and training time without compromising performance.

- ***Training Code Summary:***

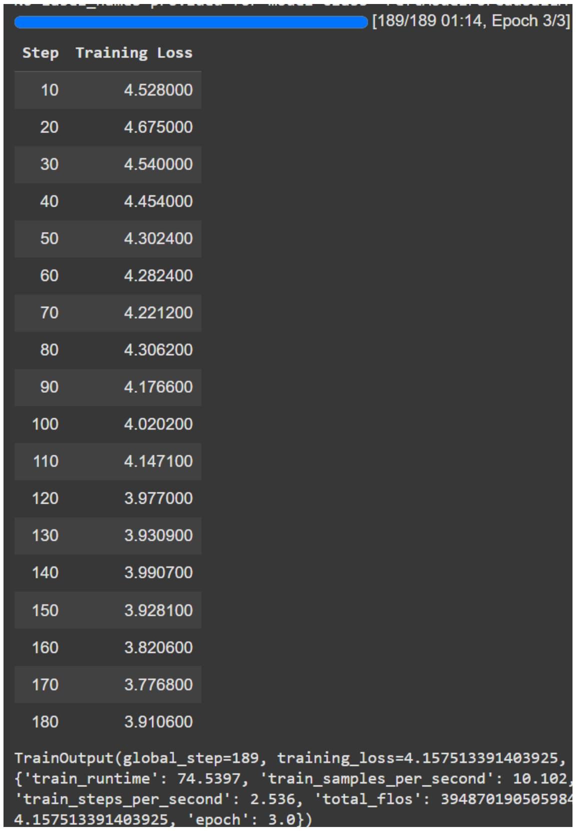
Python

```
from peft import get_peft_model, LoraConfig, TaskType
from transformers import AutoModelForCausalLM, Trainer, TrainingArguments

model = AutoModelForCausalLM.from_pretrained("gpt2")
peft_config = LoraConfig(task_type=TaskType.CAUSAL_LM, r=8,
lora_alpha=32, lora_dropout=0.1)
model = get_peft_model(model, peft_config)
```



```
from transformers import Trainer, TrainingArguments, DataCollatorForLanguageModeling

# Check if GPU is available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Data Collator
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False  # For causal LM
)

# Training Arguments
training_args = TrainingArguments(
    output_dir='./lora_gpt2_startup_pitch',
    per_device_train_batch_size=4,
    num_train_epochs=3,
    logging_dir='./logs',
    logging_steps=10,
    save_steps=500,  # Ensure checkpoints are saved
    save_total_limit=2,
    prediction_loss_only=True,
    fp16=True,  # Enable mixed precision if GPU supports FP16
    no_cuda=False,
)

# Custom trainer with labels provided
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    data_collator=data_collator  # Added data_collator to handle labels
)

trainer.train()

# After training is complete, save the model and tokenizer
trainer.save_model('./startup-pitch-lora')  # Saves the model weights (pytorch_model.bin)
tokenizer.save_pretrained('./startup-pitch-lora')  # Saves tokenizer files
```

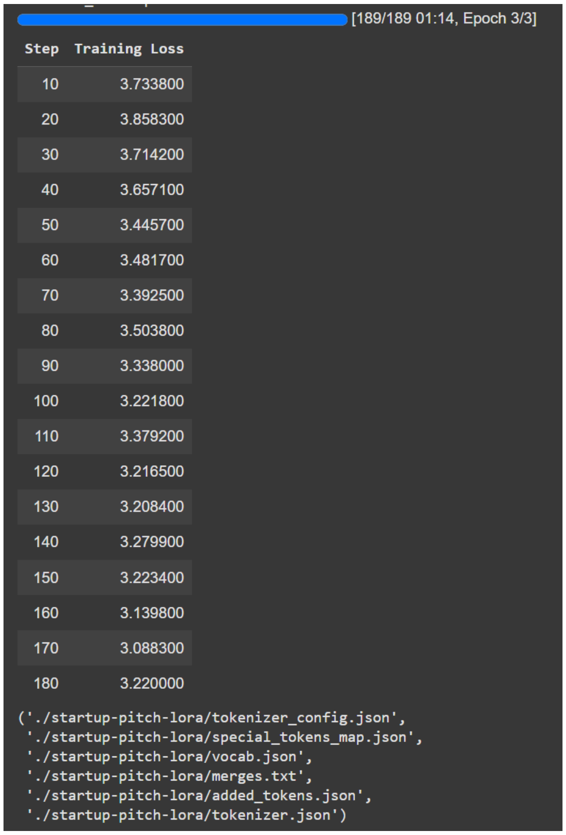
- ***Results:***
  The fine-tuned model learned to generate well-structured, creative, and contextually accurate startup pitches based on a variety of inputs.



[189/189 01:14, Epoch 3/3]

Step	Training Loss
10	4.528000
20	4.675000
30	4.540000
40	4.454000
50	4.302400
60	4.282400
70	4.221200
80	4.306200
90	4.176600
100	4.020200
110	4.147100
120	3.977000
130	3.930900
140	3.990700
150	3.928100
160	3.820600
170	3.776800
180	3.910600

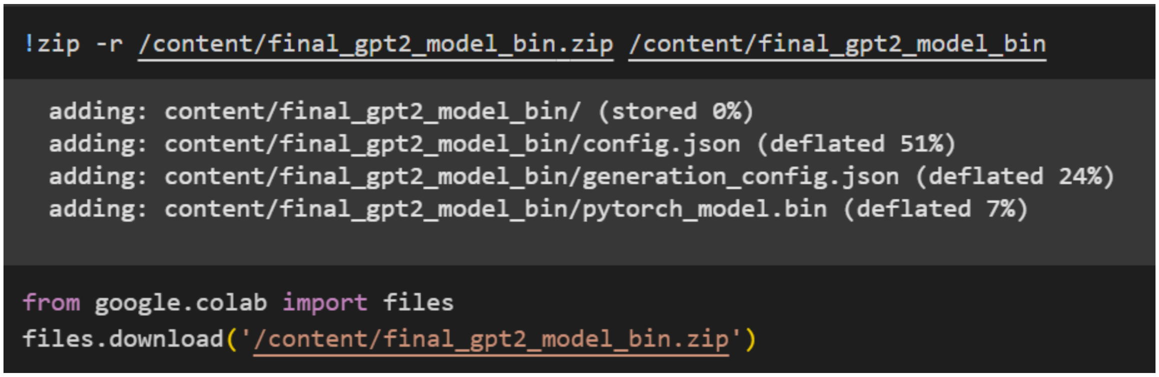
```
TrainOutput(global_step=189, training_loss=4.157513391403925,
{'train_runtime': 74.5397, 'train_samples_per_second': 10.102,
'train_steps_per_second': 2.536, 'total_flos': 394870190505984
4.157513391403925, 'epoch': 3.0})
```



[189/189 01:14, Epoch 3/3]

Step	Training Loss
10	3.733800
20	3.858300
30	3.714200
40	3.657100
50	3.445700
60	3.481700
70	3.392500
80	3.503800
90	3.338000
100	3.221800
110	3.379200
120	3.216500
130	3.208400
140	3.279900
150	3.223400
160	3.139800
170	3.088300
180	3.220000

```
('./startup-pitch-lora/tokenizer_config.json',
 './startup-pitch-lora/special_tokens_map.json',
 './startup-pitch-lora/vocab.json',
 './startup-pitch-lora/merges.txt',
 './startup-pitch-lora/added_tokens.json',
 './startup-pitch-lora/tokenizer.json')
```



```
!zip -r /content/final_gpt2_model_bin.zip /content/final_gpt2_model_bin

  adding: content/final_gpt2_model_bin/ (stored 0%)
  adding: content/final_gpt2_model_bin/config.json (deflated 51%)
  adding: content/final_gpt2_model_bin/generation_config.json (deflated 24%)
  adding: content/final_gpt2_model_bin/pytorch_model.bin (deflated 7%)

from google.colab import files
files.download('/content/final_gpt2_model_bin.zip')
```

5. Full Code Implementation

- *Step 1: Install Required Libraries*

```
pip install transformers peft accelerate streamlit
```

- *Step 2: Import Required Libraries*

Python

```
from transformers import AutoModelForCausalLM, AutoTokenizer
from peft import PeftModel
import streamlit as st
```

- *Step 3: Load the Fine-Tuned Model and Tokenizer*

Python

```
base_model = AutoModelForCausalLM.from_pretrained("gpt2")
tokenizer = AutoTokenizer.from_pretrained("gpt2")
peft_model = PeftModel.from_pretrained(base_model, "startup-pitch-lora")
peft_model.eval()
```

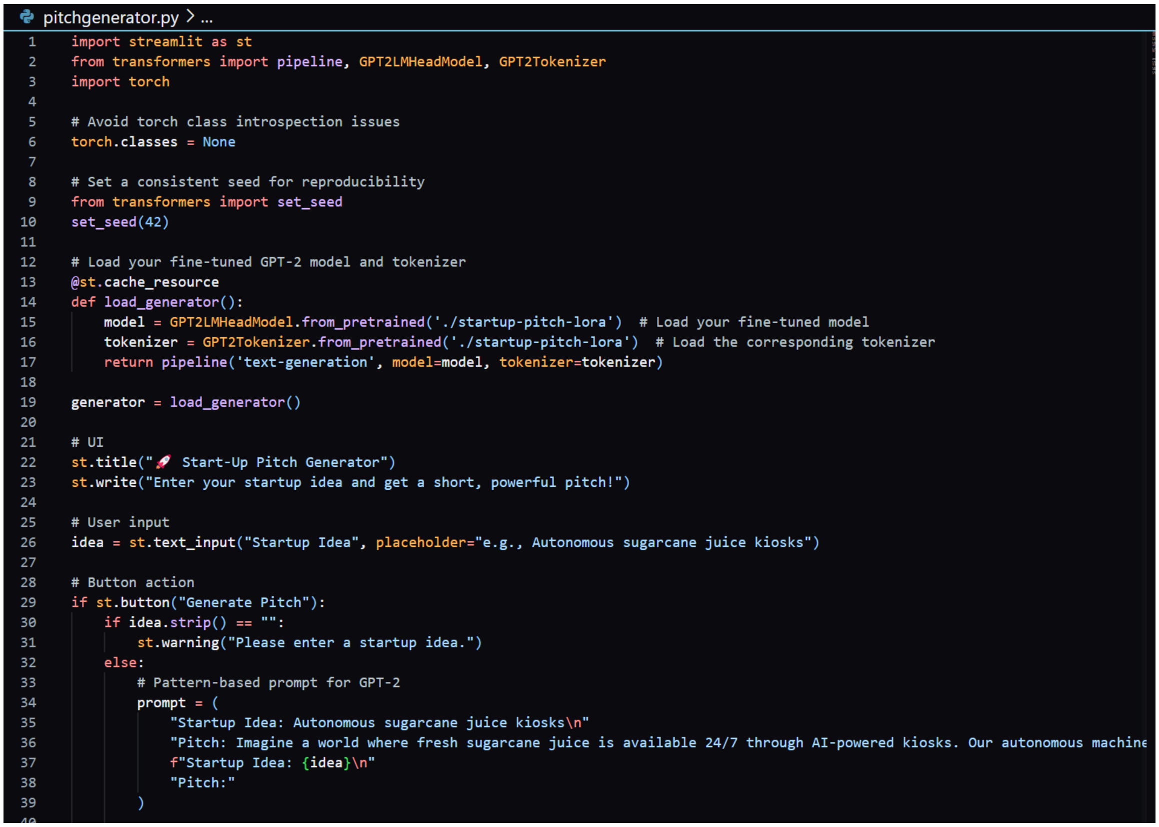
- *Step 4: Build Streamlit Interface*

Python

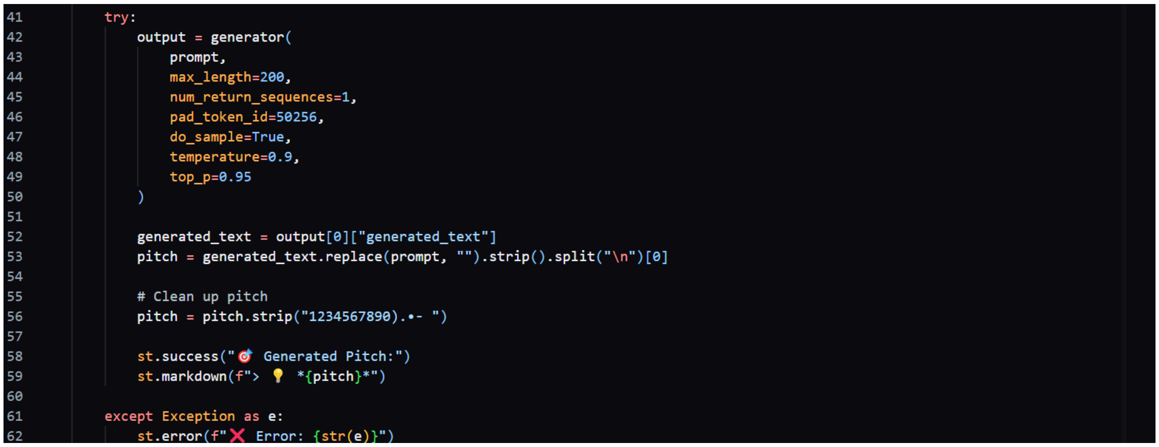
```

st.title("Startup Pitch Generator using Generative AI")
st.write("Describe your startup idea, and get a full pitch!")
input_text = st.text_area("Enter a theme, idea, or one-liner:")

if input_text:
    inputs = tokenizer(input_text, return_tensors="pt")
    outputs = peft_model.generate(**inputs, max_length=150,
num_return_sequences=1)
    result = tokenizer.decode(outputs[0], skip_special_tokens=True)
    st.subheader("Generated Pitch:")
    st.write(result)
```

- ### *Step 5: Run the Streamlit App*

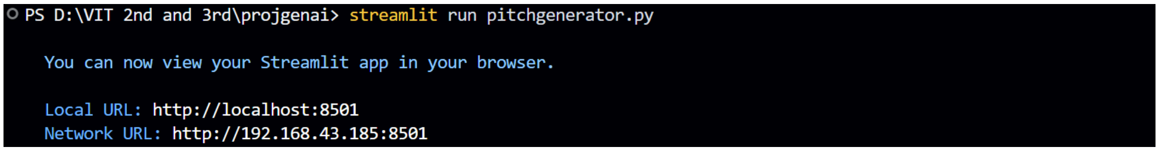
```
streamlit run pitchgenerator.py
```



```python
import streamlit as st
from transformers import pipeline, GPT2LMHeadModel, GPT2Tokenizer
import torch

# Avoid torch class introspection issues
torch.classes = None

# Set a consistent seed for reproducibility
from transformers import set_seed
set_seed(42)

# Load your fine-tuned GPT-2 model and tokenizer
@st.cache_resource
def load_generator():
    model = GPT2LMHeadModel.from_pretrained('./startup-pitch-lora')  # Load your fine-tuned model
    tokenizer = GPT2Tokenizer.from_pretrained('./startup-pitch-lora')  # Load the corresponding tokenizer
    return pipeline('text-generation', model=model, tokenizer=tokenizer)

generator = load_generator()

# UI
st.title("🚀 Start-Up Pitch Generator")
st.write("Enter your startup idea and get a short, powerful pitch!")

# User input
idea = st.text_input("Startup Idea", placeholder="e.g., Autonomous sugarcane juice kiosks")

# Button action
if st.button("Generate Pitch"):
    if idea.strip() == "":
        st.warning("Please enter a startup idea.")
    else:
        # Pattern-based prompt for GPT-2
        prompt = (
            "Startup Idea: Autonomous sugarcane juice kiosks\n"
            "Pitch: Imagine a world where fresh sugarcane juice is available 24/7 through AI-powered kiosks. Our autonomous machine
            f"Startup Idea: {idea}\n"
            "Pitch:"
        )
```



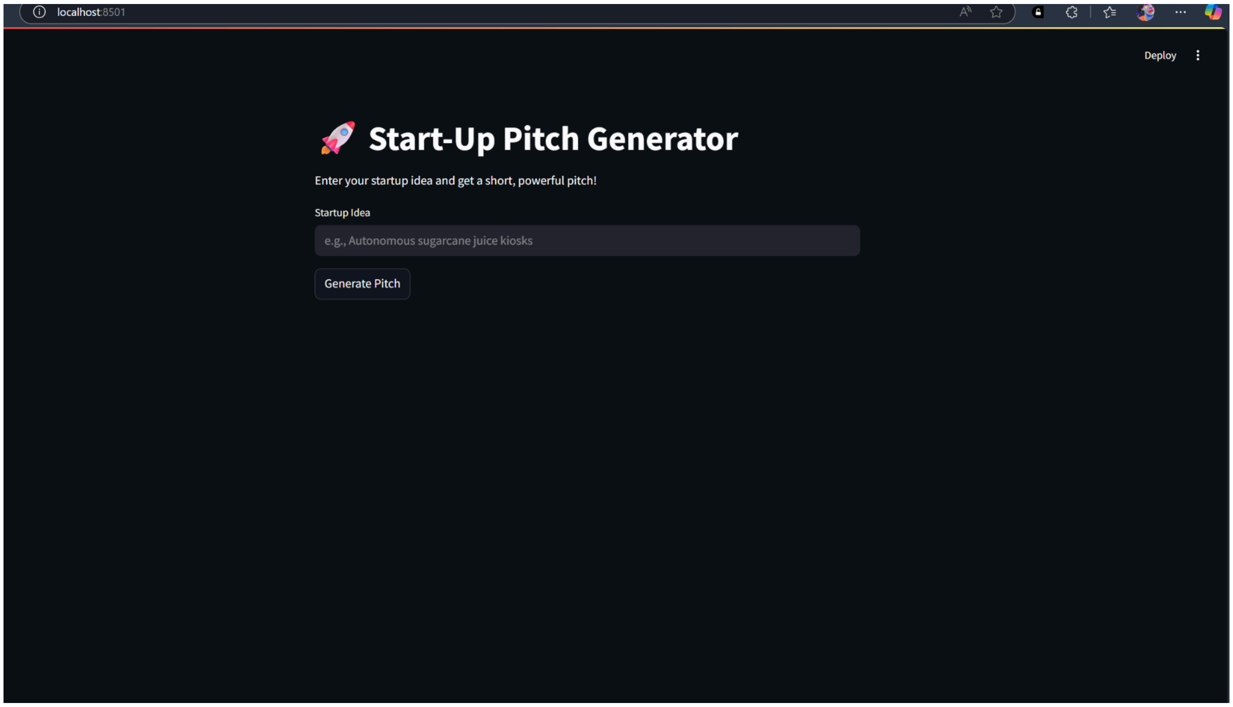
```python
        try:
            output = generator(
                prompt,
                max_length=200,
                num_return_sequences=1,
                pad_token_id=50256,
                do_sample=True,
                temperature=0.9,
                top_p=0.95
            )

            generated_text = output[0]["generated_text"]
            pitch = generated_text.replace(prompt, "").strip().split("\n")[0]

            # Clean up pitch
            pitch = pitch.strip("1234567890).•- ")

            st.success("🎯 Generated Pitch:")
            st.markdown(f"> 💡 *{pitch}*")

        except Exception as e:
            st.error(f"❌ Error: {str(e)}")
```



```
○ PS D:\VIT 2nd and 3rd\projgenai> streamlit run pitchgenerator.py

    You can now view your Streamlit app in your browser.

    Local URL: http://localhost:8501
    Network URL: http://192.168.43.185:8501
```

6. Output Screenshots

- Streamlit interface



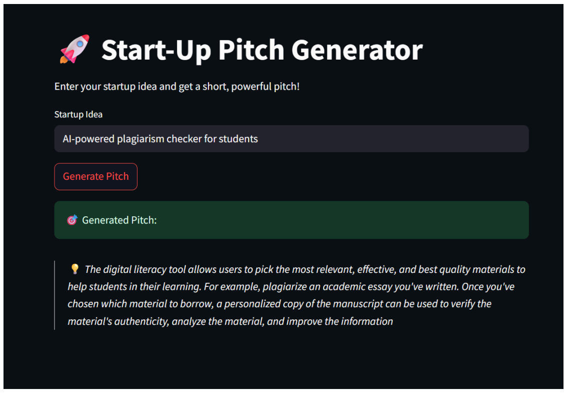
🚀 Start-Up Pitch Generator

Enter your startup idea and get a short, powerful pitch!

Startup Idea

e.g., Autonomous sugarcane juice kiosks

- Input prompt and generated startup pitch



🚀 Start-Up Pitch Generator

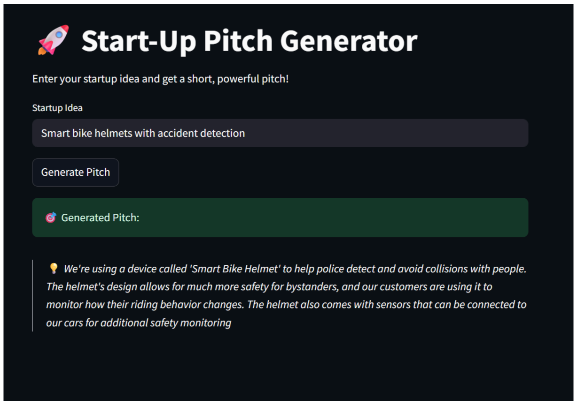
Enter your startup idea and get a short, powerful pitch!

Startup Idea

AI-powered plagiarism checker for students

🎯 Generated Pitch:

> 💡 *The digital literacy tool allows users to pick the most relevant, effective, and best quality materials to help students in their learning. For example, plagiarize an academic essay you've written. Once you've chosen which material to borrow, a personalized copy of the manuscript can be used to verify the material's authenticity, analyze the material, and improve the information*



🚀 Start-Up Pitch Generator

Enter your startup idea and get a short, powerful pitch!

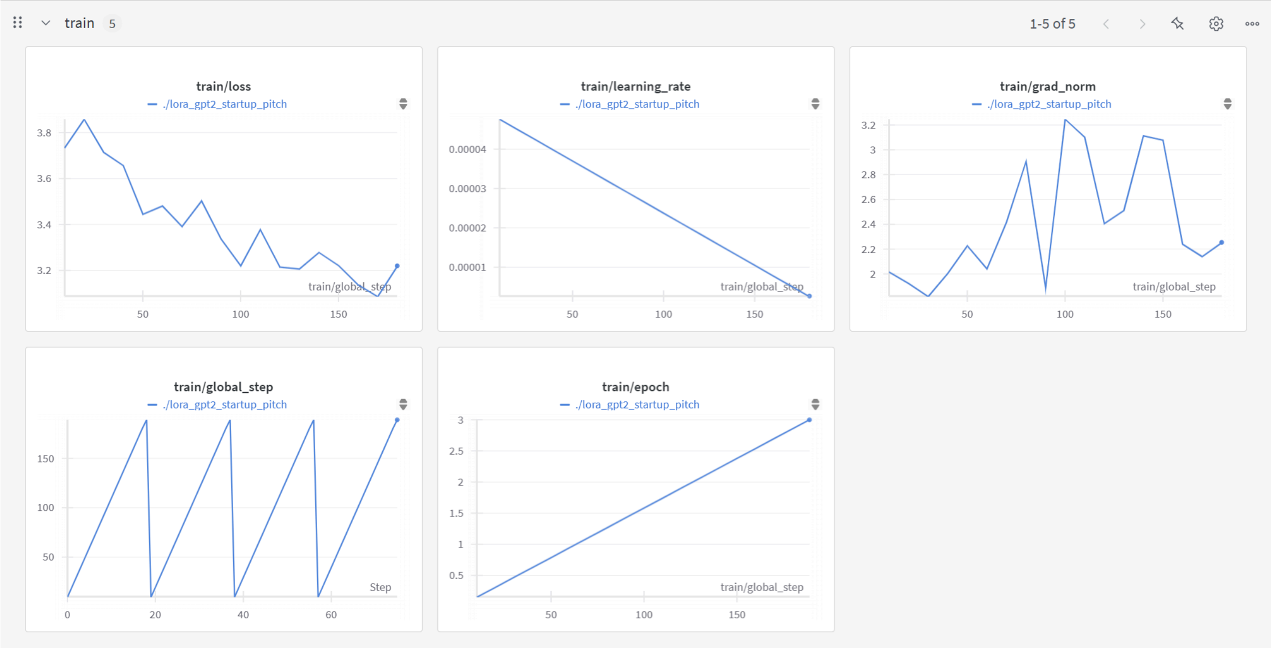
Startup Idea

Smart bike helmets with accident detection

🎯 Generated Pitch:

> 💡 *We're using a device called 'Smart Bike Helmet' to help police detect and avoid collisions with people. The helmet's design allows for much more safety for bystanders, and our customers are using it to monitor how their riding behavior changes. The helmet also comes with sensors that can be connected to our cars for additional safety monitoring*

- Training snippets or token loss graph



7. Conclusion

This project demonstrates how to use parameter-efficient fine-tuning with LoRA to adapt a generative model (GPT-2) for a highly specific domain – in this case, **startup pitch generation**. The use of a **lightweight LoRA** approach made training faster and more accessible without requiring heavy GPU resources. The web interface built with Streamlit offers a user-friendly experience for generating customized, high-quality pitch outputs.

8. References

- HuggingFace Transformers Documentation: Hugging Face - Documentation
- PEFT LoRA Docs: GitHub - huggingface/peft: 🤗 PEFT: State-of-the-art Parameter-Efficient Fine-Tuning.
- OpenAI GPT-2: https://openai.com/research/gpt-2
- Similar Open-Source Projects on AI-based Pitch Assistants on GitHub
- Papers on Parameter-Efficient Fine-Tuning for NLP (LoRA)