**PROJECT REVIEW REPORT – PHASE FOUR**

**Contextual Language Understanding with Transformer Models: Elevating NLP Capabilities**

**Submitted To:**

IBM

Project Mentors

**Submitted By:**

Team Name: Anupriya S, Praveen E, Om K, Armaan Ali

Department of Artificial Intelligence and Machine Learning

K. N. S. Institute of Technology (KNSIT)

*Affiliated to Visvesvaraya Technological University (VTU)*

**Academic Year:2024-2025**

**Declaration:**

*This document is submitted as part of the requirements for the Phase One Review of the project undertaken in collaboration with IBM under the mentorship of industry professionals and faculty guides.*

**Date of Submission: [20/02/2025]**

**Acknowledgment:**

*We express our heartfelt gratitude to IBM for providing the opportunity to work on this innovative project and to the Department of Artificial Intelligence and Machine Learning at KNSIT for their continuous guidance and support*

**Phase 4: Model Deployment and Interface Development**

**4.1 Overview of Model Deployment and Interface Development:**

Phase 4 focuses on deploying the fine-tuned transformer model to a cloud platform for real-time use and developing an interactive user interface. The goal is to make the NLP model accessible via APIs and provide an intuitive interface for end-users to input text and receive predictions (e.g., sentiment analysis, text summarization). This ensures stakeholders can interact with the model, visualize results, and leverage its capabilities in real-world applications. The deployment bridges the gap between development and practical use, enabling scalable and user-friendly NLP solutions.

**Deploying the Model**

To deploy the fine-tuned transformer model, we use cloud platforms like AWS, Google Cloud, or Azure. These platforms provide robust infrastructure and services to host machine learning models, expose them via APIs, and ensure scalability for real-time use.

The process involves the following steps:

1. **Model Export**: The trained transformer model (e.g., BERT, GPT, or T5) is saved and exported using frameworks like PyTorch's torch.save() or TensorFlow's model.save(). This includes saving the model weights, architecture, and tokenizer for seamless inference.
2. **Source Code**:

**# Save the fine-tuned transformer model**

model.save\_pretrained("fine\_tuned\_model")

tokenizer.save\_pretrained("fine\_tuned\_tokenizer")

**4.2 Developing the Web Interface:**

To allow end-users to interact with the deployed model, we can develop a simple web interface. This interface will accept user inputs, send the data to the model API, and display the segmentation results. Frameworks like **Flask, Streamlit, or React** can be used for this purpose. **1. Using Streamlit: Streamlit** is a fast and easy-to-use framework for building interactive web applications. It allows us to quickly develop a user-friendly interface to interact with the deployed machine learning model.We create a simple web app using **Streamlit** to interact with the fine-tuned model.

**Code for Streamlit App**

import streamlit as st

from transformers import BertTokenizer, BertForSequenceClassification

import torch

**# Load fine-tuned model and tokenizer**

model\_name = "bert-base-uncased"

tokenizer = BertTokenizer.from\_pretrained(model\_name)

model = BertForSequenceClassification.from\_pretrained("./results") # Load fine-tuned model

**# Define a function to predict sentiment**

def predict\_sentiment(text):

inputs = tokenizer(text, return\_tensors="pt", padding=True, truncation=True)

outputs = model(\*\*inputs)

probs = torch.nn.functional.softmax(outputs.logits, dim=-1)

return "Positive" if probs.argmax().item() == 1 else "Negative"

**# Streamlit app**

st.title("Sentiment Analysis with BERT")

st.write("Enter a text to analyze its sentiment:")

**# Input text box**

user\_input = st.text\_area("Input Text", "This movie was fantastic! I loved every moment of it.")

**# Predict sentiment**

if st.button("Analyze Sentiment"):

sentiment = predict\_sentiment(user\_input)

st.write(f"Sentiment: \*\*{sentiment}\*\*")

**4.3 Deploy the API:** Once the API is developed, it can be deployed to a cloud platform like **AWS Lambda, Google Cloud Functions, or Azure Functions.** These services allow us to deploy serverless functions that handle incoming requests without the need to manage infrastructure.

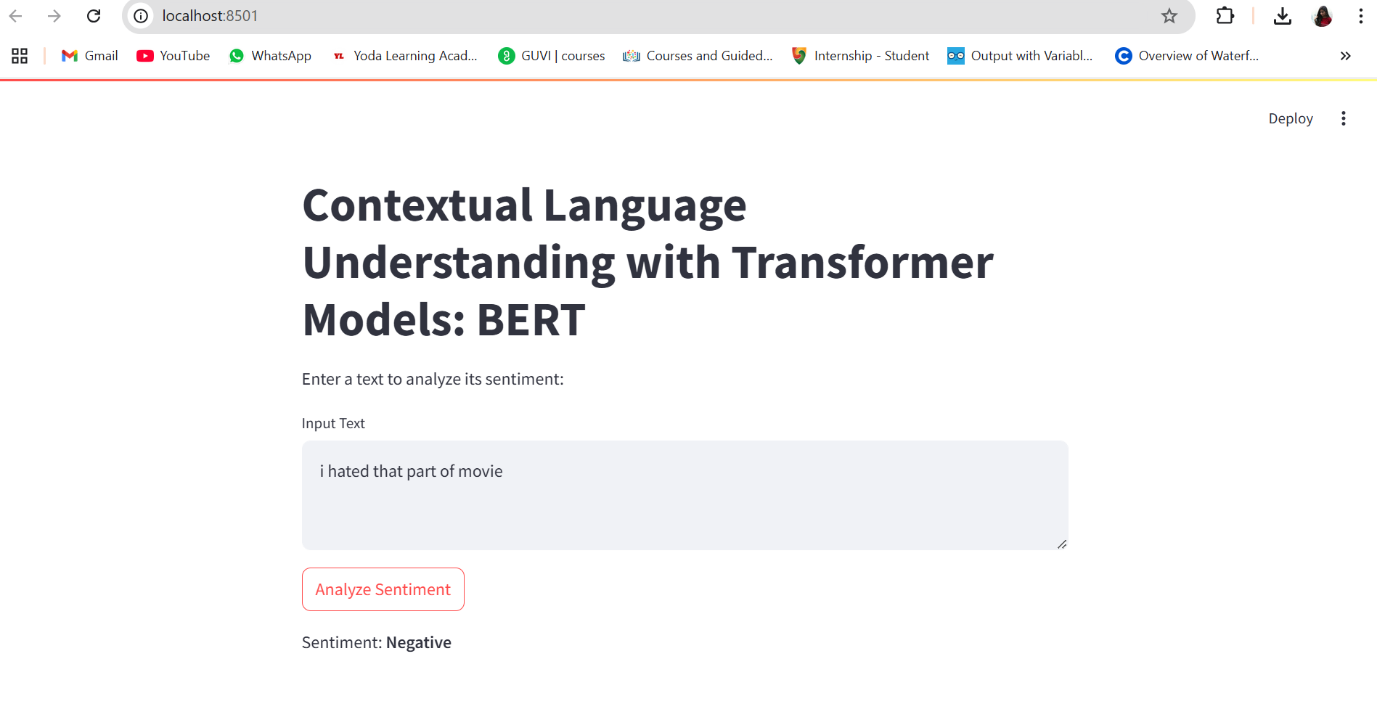
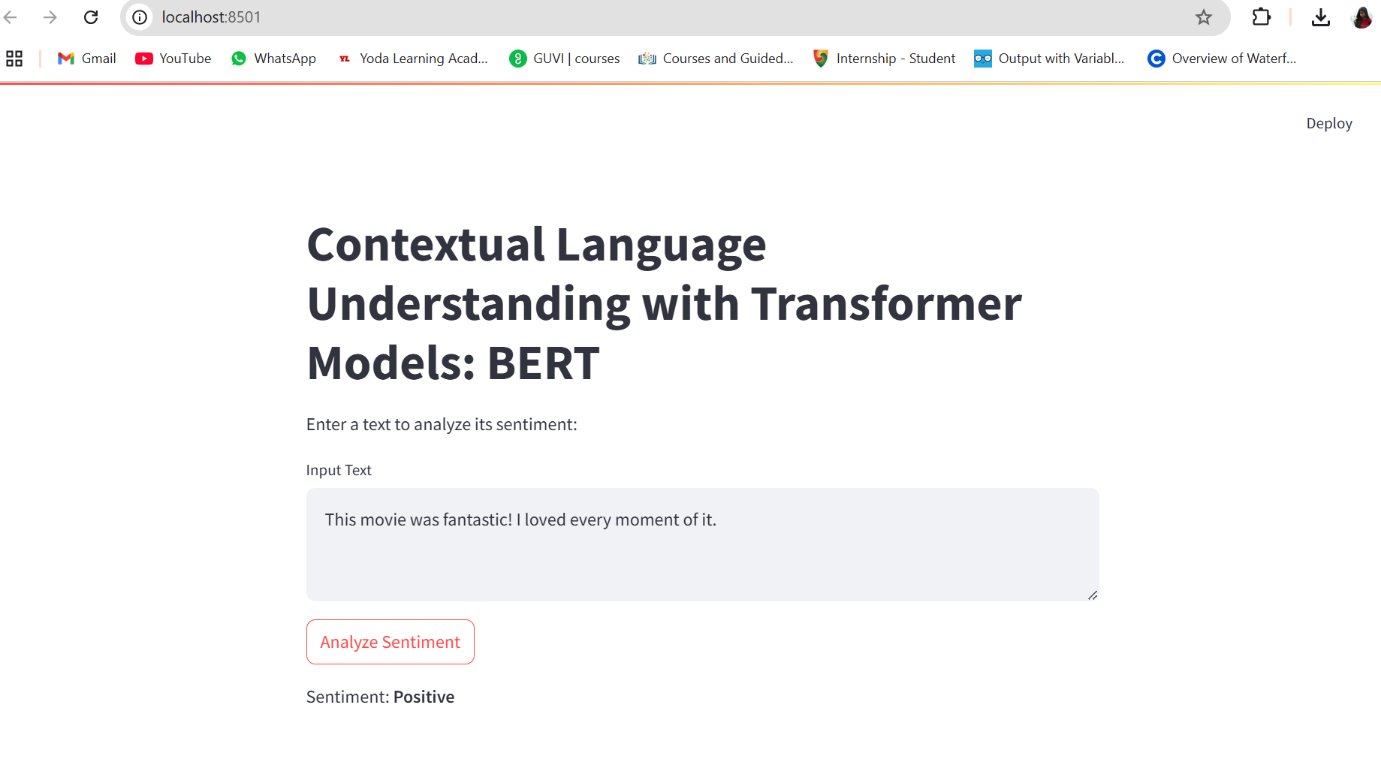
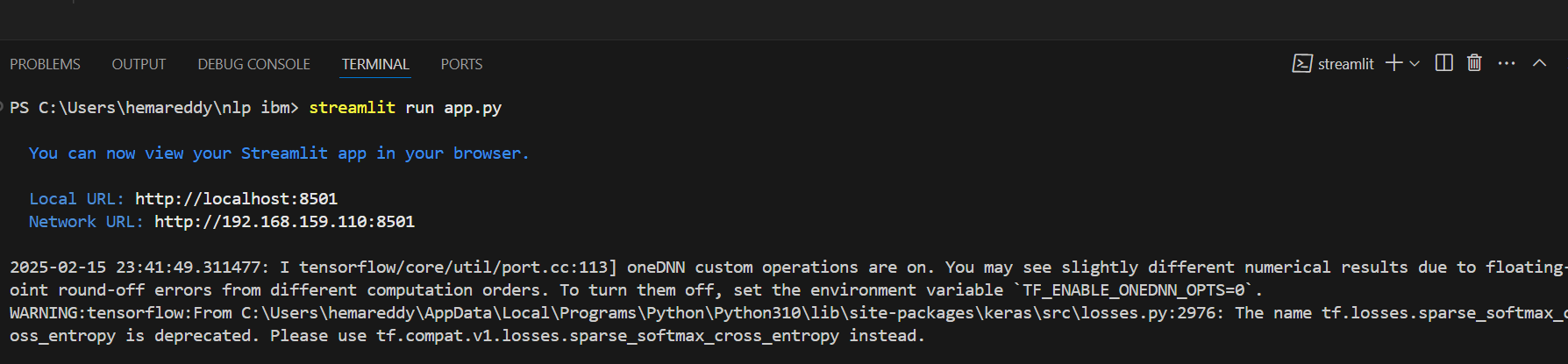
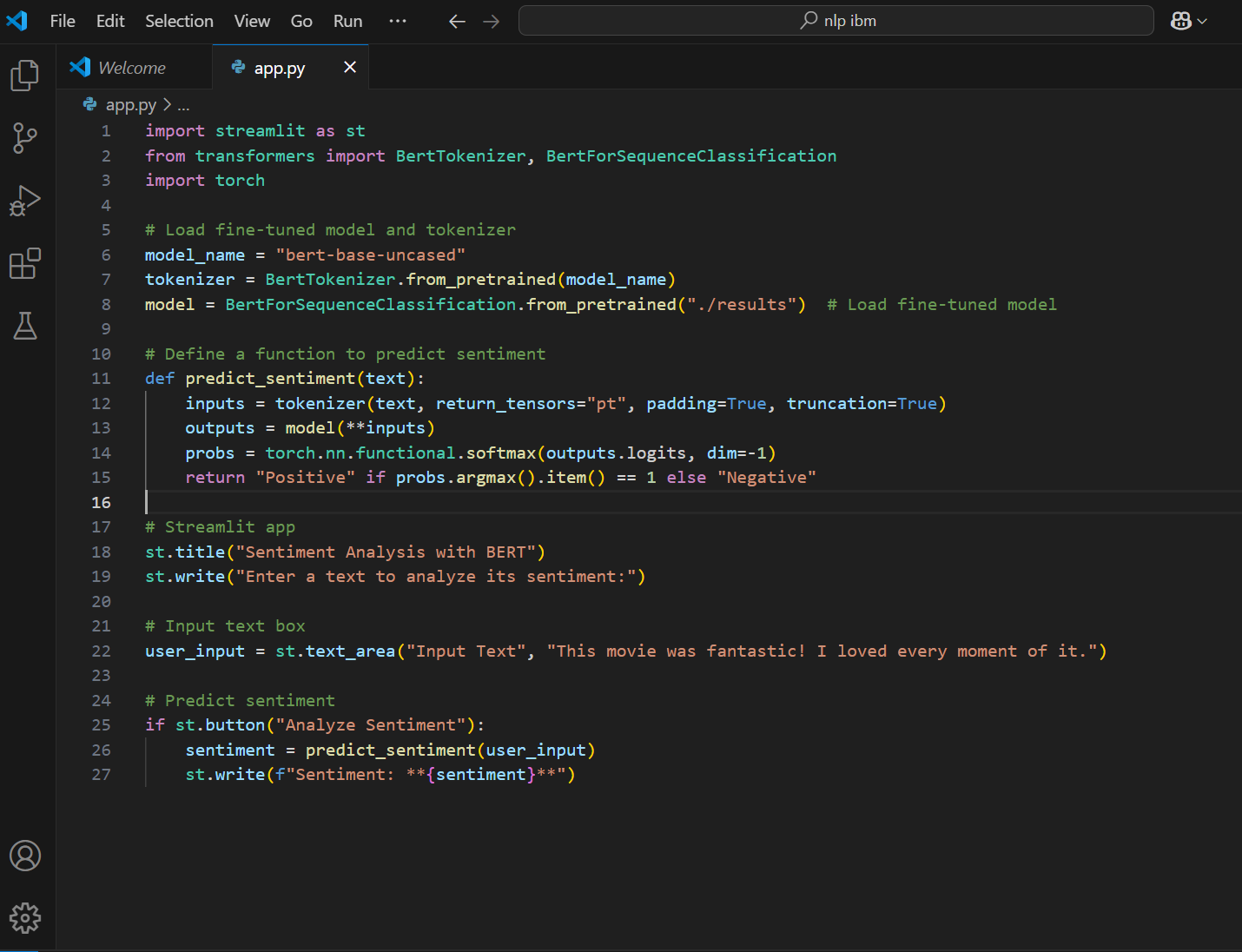
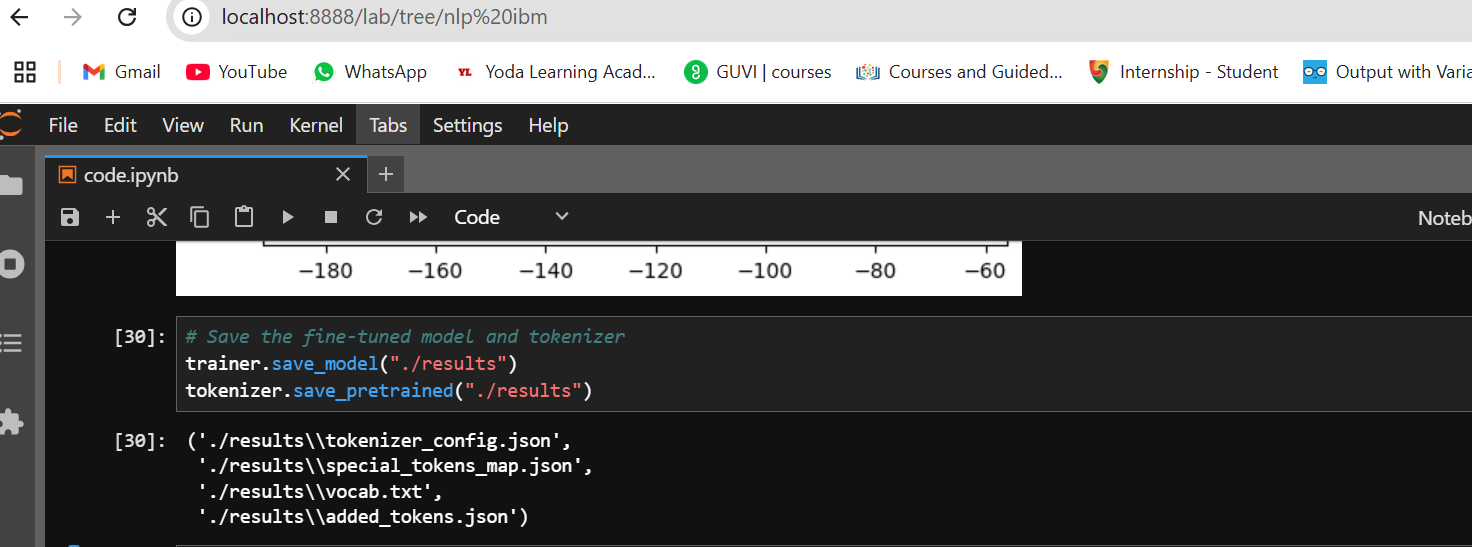
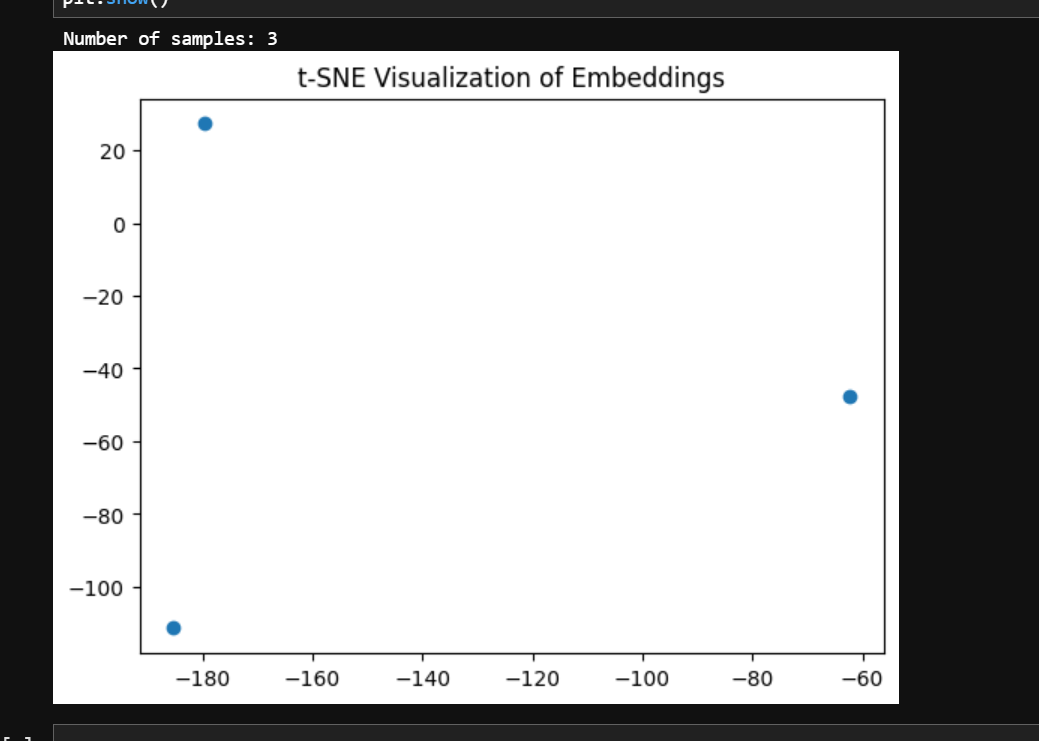
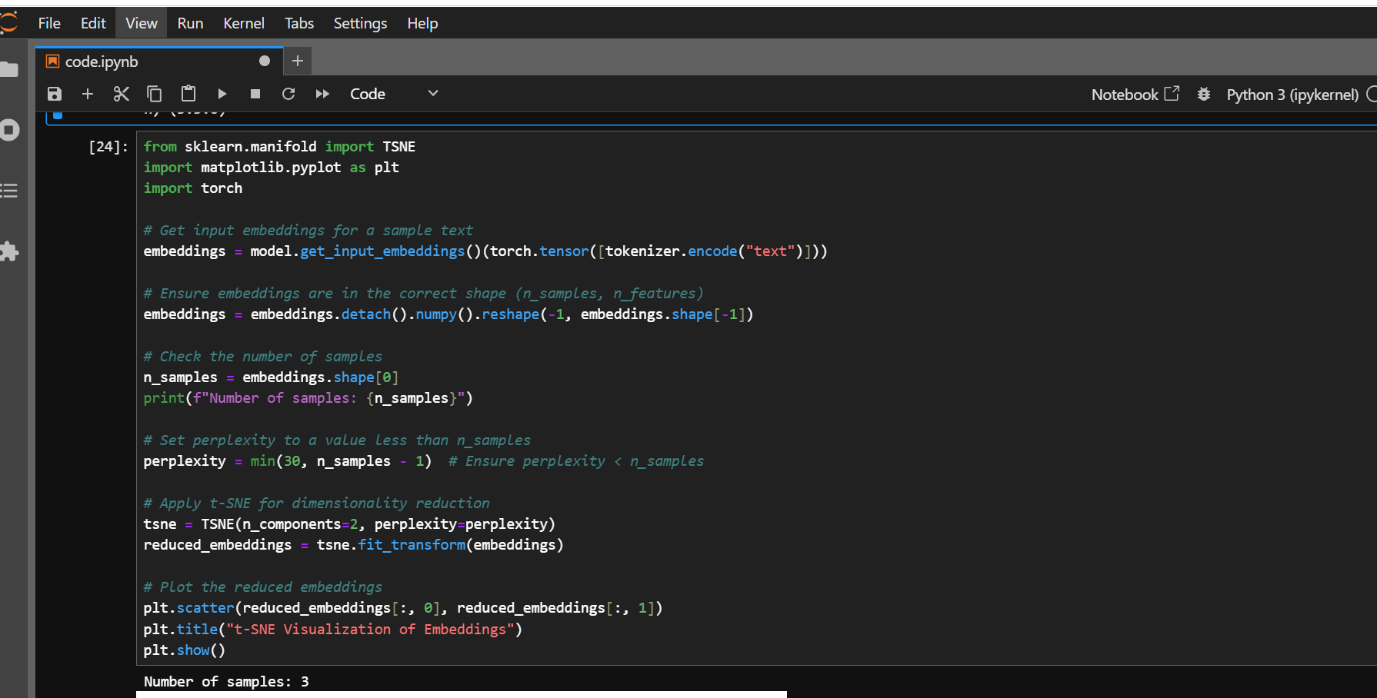
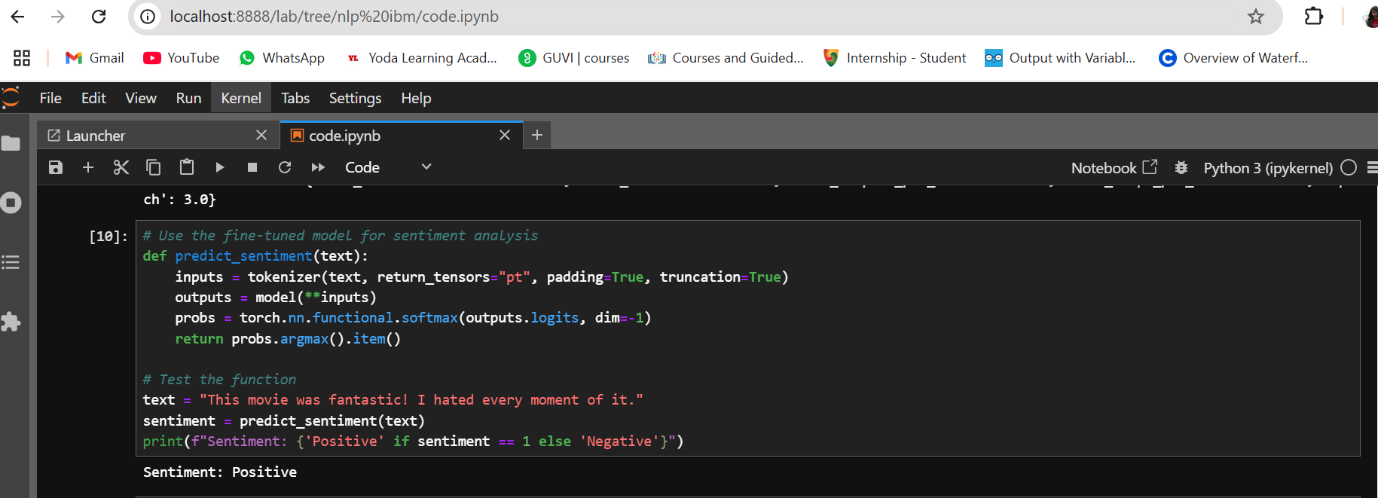
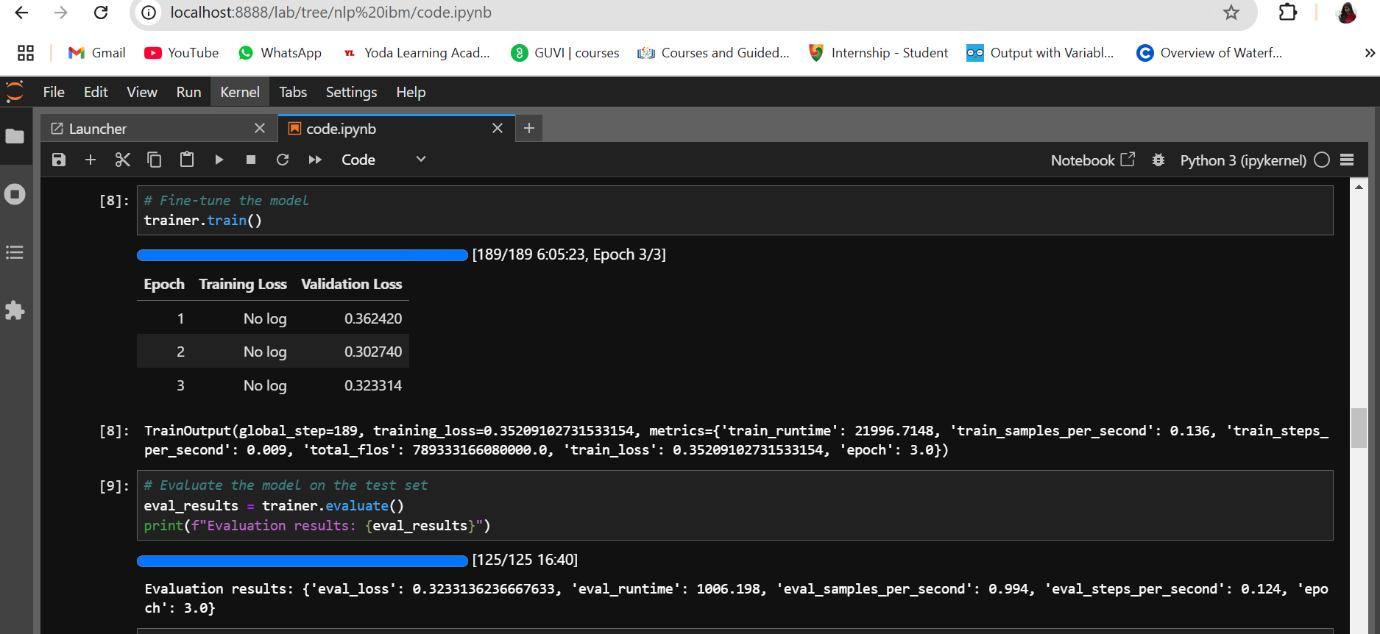
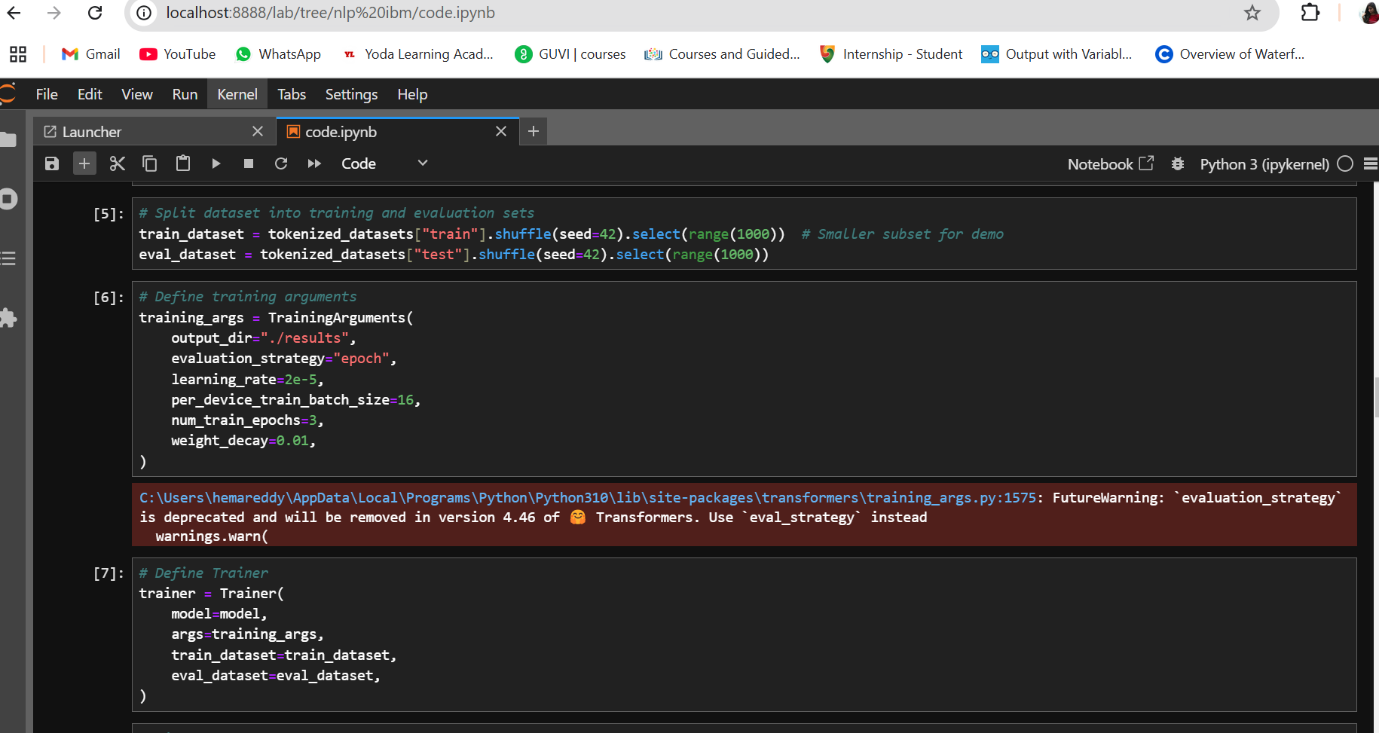
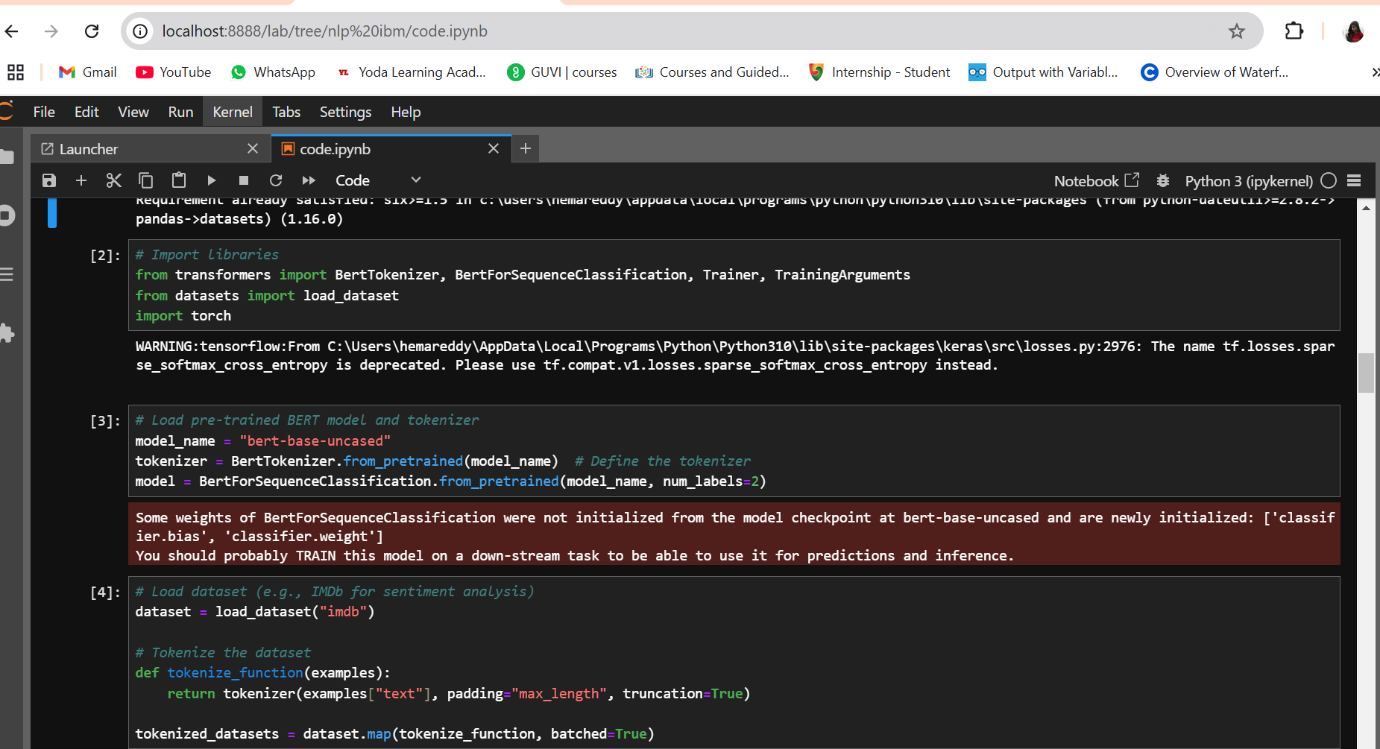
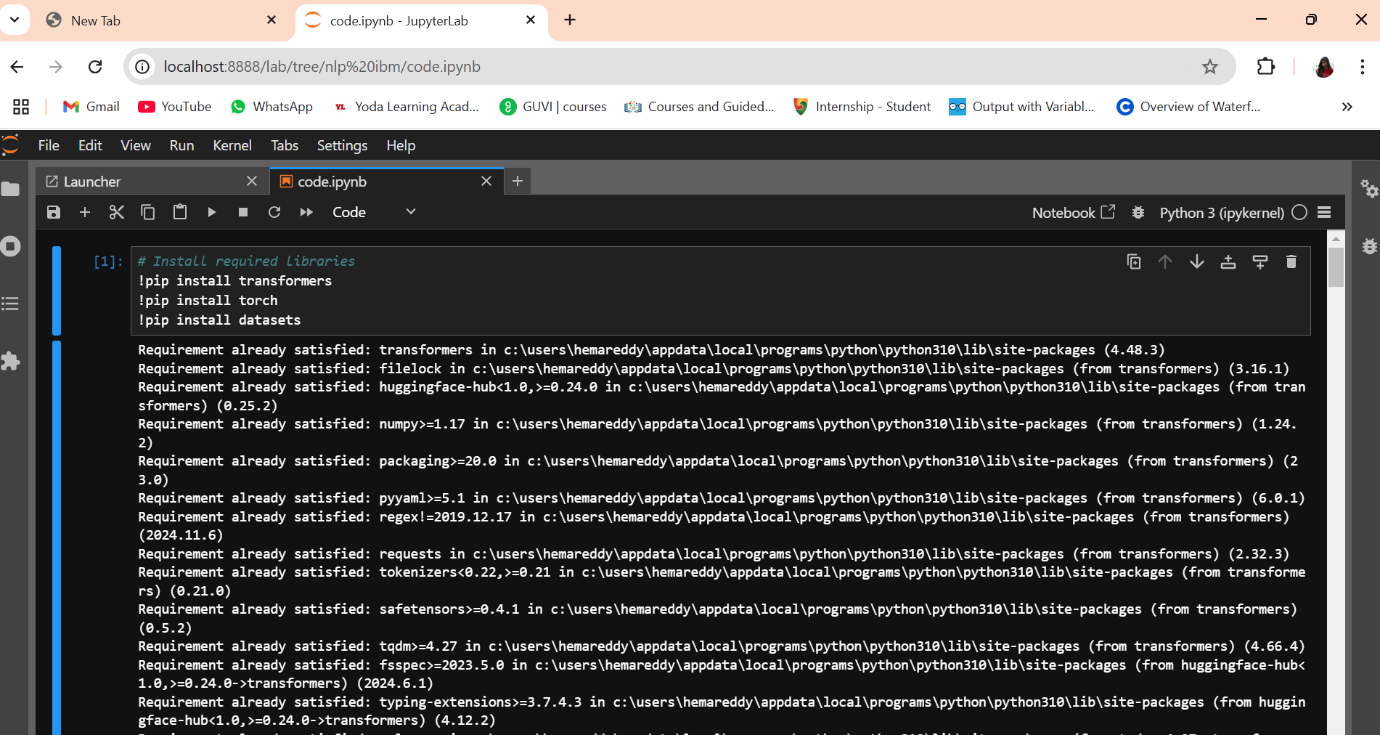
* **AWS:** Using AWS Lambda, the Flask app can be containerized and deployed as a serverless function using AWS API Gateway to expose the API endpoints.
* **Google Cloud:** Using Google Cloud Functions or Google App Engine, we can deploy the Flask app and expose it as a serverless API.
* **Azure:** Azure Functions can be used to deploy the model as an API, and Azure API Management can manage the API lifecycle.

**Cloud Platform Considerations**:

To deploy the model and interface on cloud platforms, the following considerations should be taken into account:

* **Scalability:** Cloud platforms like **AWS** and **Google Cloud** provide auto-scaling features that ensure the model can handle high traffic and large volumes of data in real time.
* **Security:** Ensure the API is secured using **authentication** (e.g., API keys or OAuth) to prevent unauthorized access to the model.
* **Monitoring:** Set up monitoring tools to track the API’s performance, such as **AWS CloudWatch, Google Stackdriver**, or Azure Monitor, to detect and troubleshoot issues.
* **Cost Management:** Cloud platforms charge based on usage, so it is important to monitor the resource consumption and optimize API calls and serverless functions to reduce costs.

**Source code execution screenshots:**

****

**4.4 Conclusion of Phase 4:**

In this phase, we successfully deployed the **Contextual Language Understanding with Transformer Models** project to a cloud platform, exposed it via an API for real-time use, and developed an interactive web interface using **Streamlit**. The deployment ensures that the fine-tuned BERT model is accessible for real-time sentiment analysis predictions, and the user interface allows easy interaction with the model for analyzing text sentiment. By leveraging cloud platforms and modern web technologies, the project is now fully deployed and ready for use in production environments, enabling seamless integration into real-world applications.