Training Report – Day 19

Topic Covered Today:

- Understanding **Frontend Integration with LLaMA Model**
- Using APIs to connect LLaMA backend with a web interface
- Designing a simple **web-based chatbot** frontend
- Tools and technologies used for LLaMA deployment

Key Learning:

Introduction to LLaMA Frontend Integration:

Today I learned how to connect the **LLaMA (Large Language Model Meta AI)** backend model with a **frontend interface** so that users can interact with it like a chatbot or assistant.

The goal was to understand how to create a **user-friendly interface** that allows text input and displays LLaMA-generated responses, similar to how ChatGPT or Hugging Face web demos work.

A **frontend** is the part of an application that users interact with (built using HTML, CSS, JavaScript, or frameworks like React). The **backend** (Python + LLaMA model) handles data processing, model predictions, and responses.

Architecture Overview:

User Interface (Frontend)
↓
HTTP Request / API Call
↓
Backend (Flask or FastAPI)
↓
LLaMA Model (Text Generation)
↓
Response sent back to Frontend

This flow helps users type messages in the browser, which are sent to the backend API where LLaMA processes the input and returns a response to be displayed on the web page.

Technologies Used:

- **Frontend:** HTML, CSS, JavaScript (or React.js)
- **Backend:** Python Flask / FastAPI
- **Model:** LLaMA (from Hugging Face Transformers or Meta AI)
- **API Communication:** RESTful API using JSON data

Implementation Example (Flask + HTML):

Backend (Python – Flask):

```
from flask import Flask, request, jsonify
from transformers import AutoTokenizer, AutoModelForCausalLM

app = Flask(__name__)

tokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-2-7b-chat-hf")
model = AutoModelForCausalLM.from_pretrained("meta-llama/Llama-2-7b-chat-hf")

@app.route('/generate', methods=['POST'])
def generate_response():
    user_input = request.json['prompt']
    inputs = tokenizer(user_input, return_tensors='pt')
    outputs = model.generate(**inputs, max_new_tokens=50)
    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return jsonify({'response': response})

if __name__ == '__main__':
    app.run(debug=True)
```

Frontend (HTML + JavaScript):

```
<!DOCTYPE html>
<html>
<head>
  <title>LLaMA Chatbot</title>
</head>
<body>
  <h2>□ LLaMA Chat Interface</h2>
  <textarea id="input" placeholder="Type your message..."></textarea><br>
  <button onclick="sendMessage()">Send</button>
  <p><b>Response:</b> <span id="response"></span></p>

  <script>
    async function sendMessage() {
      const input = document.getElementById("input").value;
      const response = await fetch("/generate", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ prompt: input })
      });
      const data = await response.json();
      document.getElementById("response").innerText = data.response;
    }
  </script>
</body>
</html>
```

This setup creates a simple **chat interface** where users can type messages, and the Flask backend communicates with the LLaMA model to generate AI-based responses.

Key Concepts Learned:

- How **frontend and backend** communicate through HTTP requests.
- The use of **APIs** to send user input to the LLaMA model.
- How to design a **minimal web UI** for AI model interaction.
- Understanding the **response flow** between browser and model.

Activities / Assignments:

- Reviewed how **LLaMA** models can be deployed through Flask.
- Designed a **basic HTML frontend** for chatbot-style interaction.
- Connected the frontend with the backend using **POST API requests**.
- Tested multiple prompts and observed LLaMA’s generated responses.
- Studied **deployment options** for hosting the application (local server or cloud).

Personal Reflection for Day 19:

Today’s session helped me understand the complete integration process of an AI model with a web interface. I realized that even the most powerful model like **LLaMA** needs an interactive **frontend** to become user-friendly and accessible.

It was exciting to see how backend Python code connects seamlessly with a web page using Flask and JavaScript. Creating a simple chatbot interface gave me real-world exposure to **AI web application development**.

This session bridged my knowledge of **machine learning and web development**, showing how full-stack AI applications are built and deployed.