Week 5: Industrial Training Report

Overview

Week 5 of the industrial training at **Auribises Technologies Pvt. Ltd.** focused on **AI integration using the OpenAI API** and the development of **intelligent chatbot systems**. The week emphasized understanding **Large Language Models (LLMs)**, API-based communication, and building end-to-end interactive chatbots using **Python**. Trainees implemented small-scale projects that simulated real-world AI workflows — integrating API calls, handling responses, and creating user-friendly conversational interfaces.

Day 21: Introduction to AI Integration and OpenAI API Setup

The twenty-first day introduced the concept of **AI-driven applications** and **API-based interaction**. Students learned about **REST APIs**, **API keys**, and **how to connect Python applications with external AI models** like OpenAI's GPT.
We discussed the workflow of sending a request to the OpenAI API, receiving a structured response, and integrating it into Python programs.

Topics Covered:

- Introduction to AI as a Service (AIaaS)
- OpenAI API overview and account setup
- Understanding API keys and environment variables
- Making API calls using Python's `requests` and `openai` libraries
- Handling authentication securely

Example: Basic OpenAI API Call

```
from openai import OpenAI

client = OpenAI(api_key="your_api_key")



response = client.chat.completions.create(
```

```
    model="gpt-4",
    messages=[{"role": "user", "content": "Hello, how are you?"}]
)
print(response.choices[0].message.content)
```

Day 22: Building a Simple AI Chatbot

Day 22 was focused on building a **basic chatbot** that interacts with users in natural language. Students learned how to capture user input, send it to the API, and display intelligent responses. We discussed **prompt design**, **context retention**, and **conversation flow management** for better chatbot performance.

Topics Covered:

- Building conversational loops in Python
- Designing prompts for meaningful responses
- Context and session-based conversation management
- Displaying responses dynamically

Example: Simple Chatbot Program

```
from openai import OpenAI

client = OpenAI(api_key="your_api_key")

while True:
    user_input = input("You: ")
    if user_input.lower() == "exit":
```

```
        break
    response = client.chat.completions.create(
        model="gpt-4",
        messages=[{"role": "user", "content": user_input}]
    )
    print("Bot:", response.choices[0].message.content)
```

Day 23: Streamlit Integration – Creating a Web-Based Chat Interface

On Day 23, students learned to **integrate AI chatbots with Streamlit**, enabling a **graphical web interface** for real-time interaction.
We explored **frontend–backend communication**, **state management**, and **UI enhancement** for chat applications.

Topics Covered:

- Introduction to Streamlit framework
- Building an interactive chat UI
- Maintaining chat history using session state
- Handling input/output dynamically in the browser

Example: Streamlit Chatbot UI

```
import streamlit as st
from openai import OpenAI


st.title("🤖 AI Chat Assistant")
client = OpenAI(api_key="your_api_key")
```

```
if "history" not in st.session_state:
    st.session_state.history = []

user_input = st.text_input("Type your message:")

if user_input:
    response = client.chat.completions.create(
        model="gpt-4",
        messages=[{"role": "user", "content": user_input}]
    )
    bot_reply = response.choices[0].message.content
    st.session_state.history.append(("You", user_input))
    st.session_state.history.append(("Bot", bot_reply))

for role, msg in st.session_state.history:
    st.write(f"**{role}:** {msg}")
```

Day 24: Advanced Chatbot Features and Context Handling

Day 24 was dedicated to enhancing chatbot intelligence by integrating **context awareness**, **memory**, and **tool-assisted reasoning**. Students explored **embedding previous user queries**, using **JSON responses**, and creating **multi-turn conversations**.

Topics Covered:

- Multi-turn conversational context
- Embedding message history for continuous dialogue
- Formatting responses with Markdown and JSON
- Error handling and rate-limit management

Example: Maintaining Chat Context

```
conversation = []


def ask_bot(prompt):
    conversation.append({"role": "user", "content": prompt})
    response = client.chat.completions.create(
        model="gpt-4",
        messages=conversation
    )
    reply = response.choices[0].message.content
    conversation.append({"role": "assistant", "content": reply})
    return reply


print(ask_bot("Who is Alan Turing?"))
print(ask_bot("What was his contribution to AI?"))
```

Day 25: Project – Smart Study Assistant Chatbot

The final day of the week was dedicated to developing a **project-based chatbot system** integrating **OpenAI API, Python, and Streamlit** — named **"ClarifAI – Smart Study Assistant."**
Students built a chatbot capable of summarizing notes, answering academic queries, and interacting intelligently with stored data. The session concluded with deployment practice and testing.

Topics Covered:

- Project: ClarifAI – AI Study Assistant
- Combining prompt engineering and file parsing
- Implementing summarization and Q&A logic
- Exporting results and deployment overview

Example: Project Workflow

1. User uploads study notes (`.txt` or `.pdf`).
2. System summarizes content using OpenAI API.
3. Chatbot answers user queries based on uploaded notes.
4. Streamlit displays summarized notes and chat history interactively.

Summary

Week 5 provided deep practical exposure to **AI integration**, **OpenAI API usage**, and **chatbot development** using Python. Students gained skills in connecting backend logic with frontend UIs through Streamlit and deploying interactive AI-driven systems.
The **ClarifAI chatbot project** served as a real-world implementation of intelligent conversation systems — combining data handling, natural language processing, and user interface design.
This week marked a significant milestone, transforming theoretical knowledge of Python into practical AI applications ready for future innovation.