Week 7: Industrial Training Report

Overview

Week 7 of the industrial training at **Auribises Technologies Pvt. Ltd.** was focused on **Project Development**, where students transitioned from learning individual technologies to **building a complete, functional project**.

This phase emphasized **project planning**, **modular coding**, **integration of frontend and backend**, and **debugging practices**.

The week simulated real-world software development workflows, including version control, testing, and performance optimization.

Day 31: Project Planning and Architecture Design

The thirty-first day marked the beginning of project development. Students learned how to analyze project requirements and convert them into a **system architecture**.

They created design documents, flowcharts, and user interface wireframes to visualize the project structure before writing any code.

Topics Covered:

- Understanding software development life cycle (SDLC)
- Requirement analysis and feasibility study
- Designing architecture: frontend, backend, and database flow
- UI/UX wireframing using Figma or hand-drawn models
- Setting up GitHub repositories for collaboration

Example: Basic Project Folder Structure

```
project/
│
├── frontend/
│   ├── index.html
```

```
│   ├── style.css
│   └── script.js
│
├── backend/
│   ├── app.py
│   ├── database.py
│   └── models/
│
├── assets/
│   ├── images/
│   └── icons/
│
└── README.md
```

Day 32: Backend Integration and Database Connectivity

Day 32 focused on integrating the **Python backend** with the **frontend interface**.
Students used **Flask or Streamlit** to connect user input forms with AI functionalities and databases such as **MongoDB** or **SQLite** for storing persistent data.

Topics Covered:

- Flask/Streamlit backend setup
- API route creation and form handling
- Connecting backend with MongoDB
- Fetching and displaying data dynamically
- Debugging connection and response issues

Example: Flask Route Integration

```
from flask import Flask, request, render_template
app = Flask(__name__)
```

```
@app.route("/", methods=["GET", "POST"])
def home():
    if request.method == "POST":
        name = request.form["username"]
        return f"Welcome, {name}!"
    return render_template("index.html")

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

Day 33: Frontend and AI Component Integration

On Day 33, students learned to **connect their AI chatbot or data-processing components** with the project's frontend.
They integrated the **OpenAI API** with the web interface, allowing real-time communication between users and the AI model.

Topics Covered:

- Linking OpenAI API with backend logic
- Handling asynchronous API calls
- Displaying AI-generated responses dynamically on the frontend
- Testing user inputs and refining prompts
- Synchronizing chat or query data with the database

Example: Backend Integration with OpenAI

```
from openai import OpenAI
client = OpenAI(api_key="your_api_key")

def get_response(user_query):
```

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

Day 34: Debugging and Testing

Day 34 focused on **debugging**, **error handling**, and **application testing**.
Students practiced identifying runtime, logical, and syntax errors, using debugging tools and logging mechanisms to trace issues.
They also performed **unit testing** and **integration testing** to ensure all components worked together seamlessly.

Topics Covered:

- Debugging techniques using `print()` and `logging` modules
- Identifying syntax vs. logical errors
- Unit testing with `unittest` and `pytest`
- Performance testing for API response time
- Error handling using `try-except` blocks

Example: Error Handling

```
try:
    result = 10 / 0
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
finally:
    print("Operation completed.")
```

Example: Logging

```
import logging
logging.basicConfig(level=logging.INFO)
logging.info("Application started successfully.")
```

Day 35: Final Project Assembly and Review

The final day of Week 7 was dedicated to **assembling and reviewing the complete project**.
Students combined all components — frontend, backend, AI integration, and database — into a single deployable application.
Mentors conducted a **code walkthrough** and provided feedback on efficiency, design, and scalability.

Topics Covered:

- Final integration and UI testing
- Bug fixing and performance optimization
- Preparing the project for demonstration
- Documentation and README creation
- Code review and mentor evaluation

Example: README Highlights

```
# ClarifAI - Smart Study Assistant
An AI-powered application built using Python, Streamlit, and OpenAI API.
Features:
- Summarizes notes and videos
- Answers student queries
- Stores data using MongoDB
```

Summary

Week 7 marked a crucial turning point where students transitioned from learning to **creating**.

Through hands-on project development, they experienced the complete software life cycle — from planning and architecture to integration and debugging.