**Weekly Progress Report**  
**Week 2: Testing Fundamentals & API Integration**  
**Duration:** APRIL 28 –MAY 2, 2025

**1. Overview**

Week 2 of the training focused on two critical aspects of modern software development: **testing fundamentals** and **API integration**. The objective of this week was to develop a strong understanding of test-driven development practices and to gain practical experience in consuming external APIs through the implementation of a mini-project. The week was structured into two phases:

* **Phase 1 (Day 1–2):** Emphasis on the theory and practice of software testing.
* **Phase 2 (Day 3–5):** Implementation of a CLI-based weather dashboard application, integrating an external weather API and applying the testing knowledge gained earlier.

**2. Phase 1: Testing Fundamentals (Day 1–2)**

**Topics Covered:**

* **Introduction to Testing Philosophy:**  
  Covered the purpose, importance, and types of software testing. Detailed exploration of:
  + **Unit Testing:** Testing individual components or functions for correctness.
  + **Integration Testing:** Ensuring multiple components work together as intended.
  + **End-to-End Testing:** Simulating real user scenarios to validate the entire flow of the application.
* **Tooling & Frameworks:**
  + For Python-based environments: **Pytest**

**Practical Implementation:**

* Set up testing environments and configuration for the Week 1 project.
* Developed and executed test cases covering:
  + Core functionalities
  + Edge cases and invalid inputs
  + Integration points between components
* Focused on writing clean, maintainable tests using assertion libraries and test runners.
* Introduced the concept of mocking to simulate external dependencies and isolate testing logic.

**3. Phase 2: Mini-Project – Weather Dashboard CLI (Day 3–5)**

**Project Objective:**

To design and implement a Command-Line Interface (CLI) tool that fetches and displays real-time weather data by integrating with a public weather API. This project provided a practical scenario to apply API handling, data formatting, and testing techniques.

**Key Functionalities Developed:**

* **API Integration:**
  + Utilized the **OpenWeatherMap API** or **WeatherAPI** to fetch live weather data.
  + Handled API authentication via secure token usage and incorporated best practices for API call management.
* **User Interface (CLI):**
  + Designed a clear and navigable CLI for user interaction.
  + Displayed current conditions such as temperature, humidity, and wind speed.
  + Extended functionality to include weather forecasts where supported by the API.
* **Local Data Persistence:**
  + Implemented a feature to save historical weather queries in a structured **JSON file**.
  + Ensured efficient read/write operations with appropriate file locking and error checks.
* **Error Handling and Optimization:**
  + Gracefully handled various error conditions including:
    - Invalid city names
    - Network failures
    - API rate limits
  + Applied caching logic to reduce redundant API calls for repeated queries.

**Testing and Validation:**

* Developed a comprehensive suite of automated tests using **Pytest**.
* Utilized **mocking frameworks** to simulate API responses for:
  + Successful data retrieval
  + API errors and failures
  + Boundary and edge cases
* Achieved high test coverage and ensured the robustness of both business logic and data storage mechanisms.

**4. Key Achievements and Learning Outcomes**

* **Technical Skills Developed:**
  + Proficient use of modern testing frameworks and tools.
  + Deep understanding of the role of tests in software lifecycle and CI/CD pipelines.
  + Hands-on experience with RESTful APIs and data serialization (JSON).
  + Error handling and optimization strategies for production-ready applications.
* **Project Management & Development Practices:**
  + Strengthened skills in problem decomposition and incremental development.
  + Practiced modular code design to facilitate testing and scalability.
  + Gained exposure to version control workflows with test-first development.
* **Collaboration & Documentation:**
  + Maintained well-documented code and tests for peer readability and future maintenance.
  + Produced clear in-code comments and README documentation for project usability.

**5. Conclusion**

This week provided a solid foundation in both testing practices and API integration, reinforcing the importance of writing reliable, maintainable code. By applying these skills in a practical mini-project, real-world development challenges such as error handling, data persistence, and external API management were effectively addressed. These experiences have laid the groundwork for more advanced backend development in the coming weeks.

**Prepared By:** Krisha Jain