FYP PRE DOCUMENT

software quality engineering|

ADNAN HUSSAIN

22F-3696

[Year]

**Quality and Testing Requirements for Final Year Project**

**Project Title:**

**Farming Optimization through AI:**

**This project aims to develop a web/mobile application that uses AI to analyze farming data, predict optimal crop yields, and provide actionable insights to farmers.**

**1. Non-Functional Requirements**

**1.1 Performance Requirements**

* **System Performance:**
  + API response time must be under **200ms** for 95% of requests under normal load conditions.
  + The system must handle **1,000 concurrent users** without downtime or degradation in performance.
* **Scalability:**
  + The system should scale to handle increasing requests by adding more server resources dynamically.
  + Database queries must handle datasets with **10 million records** efficiently.
* **Latency:**
  + Ensure maximum latency does not exceed **500ms** for edge-case scenarios.
* **Throughput:**
  + The system must process at least **500 API calls per second** during peak loads.

**1.2 Availability**

* The system must have **99.9% uptime**, including scheduled maintenance.
* APIs should implement health-check endpoints to monitor service status.

**1.3 Reliability**

* System should recover from failures within **10 seconds** using automated failover mechanisms.
* Data persistence must ensure **zero data loss** in case of hardware failure.

**1.4 Usability**

* The application should support **multilingual interfaces** (e.g., English, Urdu, Sindhi).
* AI recommendations should be easy to interpret by non-technical users, with graphical insights.

**1.5 Compliance**

* The system must comply with **GDPR** and **local data privacy laws**.
* APIs should follow **RESTful standards** for consistency.

**2. Security Requirements**

**2.1 Authentication and Authorization**

* Implement **OAuth 2.0** for user authentication and access control.
* All endpoints must be protected with **JSON Web Tokens (JWT)**.

**2.2 Data Protection**

* Encrypt sensitive data (e.g., user credentials, farming analytics) using **AES-256**.
* Use **HTTPS** for all data transmission to prevent eavesdropping and man-in-the-middle attacks.

**2.3 API Security**

* Follow the **OWASP REST API Security Top 10** checklist:
  + Avoid exposing sensitive data in API responses.
  + Validate all input data to prevent SQL injection and XSS attacks.
  + Rate limit API requests to prevent denial-of-service (DoS) attacks.

**2.4 Monitoring and Logging**

* Implement centralized logging for all API activities (using tools like **ELK Stack** or **Datadog**).
* Set up real-time monitoring for suspicious activities and trigger alerts.

**2.5 Penetration Testing**

* Conduct penetration testing to identify vulnerabilities in the application.
* Regularly update software dependencies to address known vulnerabilities.

**3. Software Test Plan**

**3.1 Test Strategies**

The test plan will include the following strategies:

| **Test Type** | **Purpose** |
| --- | --- |
| **Unit Testing** | Ensure individual functions (e.g., AI models, API methods) work correctly. |
| **Integration Testing** | Verify communication between AI models, APIs, and databases. |
| **Performance Testing** | Test the system under heavy loads using **JMeter** or **K6**. |
| **Security Testing** | Identify vulnerabilities using tools like **OWASP ZAP** or **Burp Suite**. |
| **Regression Testing** | Verify new changes don’t break existing functionality. |

**3.2 Testing Tools**

The project will utilize the following tools:

* **RestAssured** for API automation testing.
* **Postman** for manual API testing.
* **JMeter** or **K6** for performance and load testing.
* **OWASP ZAP** for security testing.
* **JUnit** for unit testing the Java-based application.

**3.3 Test Automation Plan**

* Automate **API functional tests** (GET, POST, PUT, DELETE) using RestAssured.
* Automate **load testing** with K6 to simulate real-world usage scenarios.
* Automate **security tests** with OWASP ZAP.

**3.4 Test Scenarios**

For each API endpoint, create the following scenarios:

| **Test Scenario** | **Description** |
| --- | --- |
| **GET /api/crops** | Fetch crop data for a specific region. |
| **POST /api/crops/recommend** | Submit farming data to generate AI-based recommendations. |
| **PUT /api/users/profile** | Update user profile information. |
| **DELETE /api/data/{id}** | Delete user farming data with the given ID. |

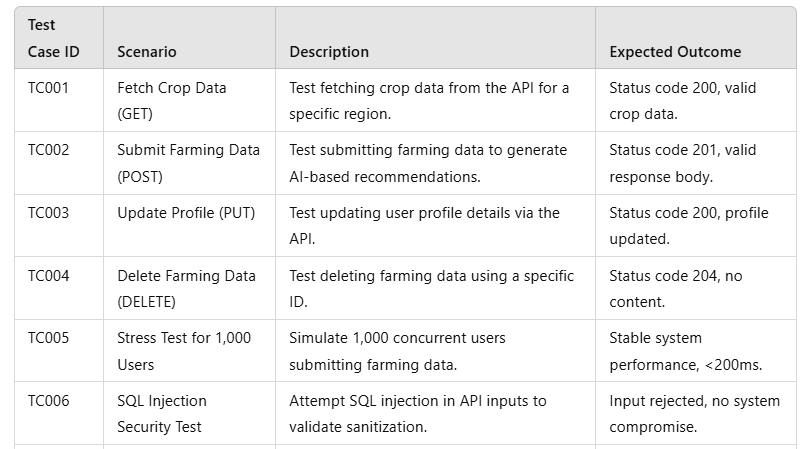
**4. Performance Requirements for Web APIs**

**4.1 Performance Benchmarks**

* API requests should meet a **200ms response time** goal.
* Average CPU usage should not exceed **70%** under load.

**4.2 Load Testing Scenarios**

* **Scenario 1**: Simulate **1,000 concurrent users** submitting farming data recommendations.
* **Scenario 2**: Simulate **5,000 requests** per minute for fetching AI insights.



**5. Security Checklist**

* **Authentication**:
  + Use **OAuth 2.0** for secure API access.
* **Input Validation**:
  + Sanitize and validate all incoming data.
* **Error Handling**:
  + Avoid exposing sensitive information in error messages.

**6. Test Automation Document**

**6.1 Tools for Test Automation**

1. **RestAssured** for functional testing of REST APIs.
2. **JMeter** or **K6** for load testing.
3. **OWASP ZAP** for penetration testing

