**22F-3733**

**Marsad Saqib**

**SQE-Assingment-4**

**Task-3**

**Quality and Testing Requirements for FYP**

Below is a **generic set of quality and testing requirements** for a typical Final Year Project (FYP) that includes building a backend with RESTful APIs:

1. **Non-Functional Requirements**

**Performance Requirements**

1. **Response Time**:
   * The API should respond to 95% of requests within **200ms** for read operations (GET).
   * Write operations (POST, PUT, DELETE) should complete within **500ms** under normal load.
2. **Throughput**:
   * The API should handle at least **100 requests per second** (RPS) with minimal degradation.
3. **Scalability**:
   * The system should scale horizontally to support **1,000 concurrent users** without performance drops.
4. **Availability**:
   * The API should have an uptime of **99.9%**, allowing for no more than **44 minutes of downtime per month**.
5. **Stress Handling**:
   * The API should continue functioning under a load of up to **5x peak traffic** with graceful degradation.

**Security Requirements**

1. **Authentication and Authorization**:
   * Use **OAuth 2.0** or **JWT** for secure authentication.
   * Implement **role-based access control (RBAC)** to limit resource access.
2. **Data Encryption**:
   * Enforce HTTPS to secure data in transit.
   * Use strong encryption algorithms (e.g., AES-256) for sensitive data at rest.
3. **Input Validation**:
   * Implement validation for all input parameters to prevent injection attacks.
   * Enforce strict content-type checks (application/json).
4. **Rate Limiting**:
   * Limit each user to **60 requests per minute** to prevent abuse.
5. **Error Handling**:
   * Ensure APIs return generic error messages (e.g., 500 Internal Server Error) without exposing sensitive information.
6. **OWASP API Security Guidelines**:
   * Mitigate common threats like **Injection**, **Broken Authentication**, **Excessive Data Exposure**, **Lack of Rate Limiting**, and **Broken Object-Level Authorization (BOLA)** by following the OWASP API Security Top 10.

### **2. Software Test Plan**

#### ****Purpose of the Test Plan****

To define the scope, approach, and strategy for testing RESTful APIs developed for the project, ensuring functional correctness, performance, and security compliance.

#### ****Scope****

* Testing the core backend APIs for functional correctness, performance benchmarks, and security compliance.
* Testing will be performed across development, staging, and production environments.

#### ****Test Strategies****

1. **Functional Testing**:
   * **Goal**: Verify API endpoints' behavior.
   * **Tools**:
     + Postman for manual testing.
     + RestAssured (Java) or Pytest (Python) for automated testing.
   * **Test Cases**:
     + Validate CRUD operations for each endpoint.
     + Verify status codes (200, 400, 404, 500).
     + Ensure API response matches the schema.
2. **Performance Testing**:
   * **Goal**: Ensure the API meets performance benchmarks.
   * **Tools**:
     + JMeter for load and stress testing.
     + Locust for simulating concurrent user traffic.
   * **Test Scenarios**:
     + Simulate **100 concurrent users** performing GET and POST requests.
     + Stress test the API with **1,000 RPS** and analyze response degradation.
3. **Security Testing**:
   * **Goal**: Identify vulnerabilities and secure the API.
   * **Tools**:
     + OWASP ZAP for automated vulnerability scanning.
     + Burp Suite for manual penetration testing.
   * **Test Scenarios**:
     + Test for **SQL Injection** by sending malicious queries.
     + Validate authentication bypass attempts.
     + Ensure sensitive endpoints (e.g., /admin) are inaccessible without proper permissions.
4. **Integration Testing**:
   * **Goal**: Verify seamless integration between modules (e.g., frontend, database, and backend APIs).
   * **Tools**:
     + Postman Collections for API chaining.
     + Dockerized environment for testing microservices communication.
   * **Test Scenarios**:
     + Verify API interaction with the database for transactions.
     + Simulate interactions between the frontend and backend.
5. **Regression Testing**:
   * **Goal**: Ensure new changes do not break existing functionality.
   * **Tools**:
     + Jenkins for Continuous Integration (CI) pipeline.
     + Selenium (if frontend exists) for end-to-end regression testing.
   * **Test Scenarios**:
     + Re-run all automated test suites after feature updates.
6. **Usability and Compatibility Testing**:
   * **Goal**: Ensure APIs are accessible across multiple devices or platforms.
   * **Tools**:
     + Postman or Swagger for API documentation and testing compatibility with mobile and web apps.

**Testing Tools and Frameworks**

| **Category** | **Tool** | **Purpose** |
| --- | --- | --- |
| Functional Testing | Postman, RestAssured | Manual and automated API testing |
| Performance Testing | JMeter, Locust | Load, stress, and throughput testing |
| Security Testing | OWASP ZAP, Burp Suite | Vulnerability scanning and penetration testing |
| Test Automation | Jenkins | Continuous Integration and Regression testing |
| API Documentation | Swagger | API design, documentation, and usability |

#### ****Test Reporting****

1. **Metrics to Report**:
   * Total test cases executed (passed/failed).
   * API response time and throughput.
   * Security vulnerabilities detected.
2. **Reporting Tools**:
   * Allure Report for visual test results.
   * JMeter HTML reports for performance metrics.
   * OWASP ZAP reports for security issues.