**Introduction**   
  
I’ll be diving into my Final Year Project (FYP), where I’ll work on creating something exciting—whether it’s a game, a mobile app, a web application, or even an AI tool. No matter the type of project, I’ll be building backend web APIs to power the system. To make sure everything runs smoothly and securely, this report focuses on laying out the **Non-Functional Requirements**, **Performance Goals**, **Security Measures**, and a detailed **Testing Plan**

**1. Non-Functional Requirements**

Non-functional requirements focus on the performance, scalability, security, and usability aspects of the project. They ensure that the application is robust and can handle real-world usage scenarios effectively.

**Performance Requirements**

|  |  |
| --- | --- |
| Requirement | Description |
| Response Time | The API must respond within **200ms** for 95% of the requests under normal load. |
| Throughput | The system should handle **500 concurrent users** with an average of **5 requests per second**. |
| Scalability | The APIs should scale horizontally to support **20% traffic increase** during peak hours. |
| Latency | Internal API-to-API communication latency should not exceed **100ms**. |
| Stress Handling | The APIs should sustain up to **1000 requests/second** during stress testing. |

**Security Requirements**

|  |  |
| --- | --- |
| Authentication | Use **JWT (JSON Web Token)** for secure and stateless authentication. |
| Authorization | Implement role-based access control (RBAC) to restrict API access. |
| Encryption | Encrypt all sensitive data in transit using HTTPS and data at rest using AES-256. |
| Rate Limiting | Limit the number of requests per user to 100 requests/minute to prevent DoS attacks. |
| Input Validation | Validate all user inputs to prevent SQL injection, XSS, and path traversal attacks. |
| Error Handling | Ensure error messages do not reveal sensitive information (e.g., stack traces). |
| Logging | Log all API activities securely to detect and analyze potential breaches. |
| CORS Policy | Implement strict Cross-Origin Resource Sharing (CORS) policies. |

**Test Plan Document**

The test plan defines the strategy, objectives, scope, and tools for testing the APIs to ensure they meet the quality standards.

**Objectives**

* Validate that all API endpoints work as intended.
* Ensure APIs meet performance benchmarks under various conditions.
* Test the security of the APIs to detect and prevent vulnerabilities.
* Verify that new updates do not break existing functionality.

**Scope**

The test plan covers:

* Functional testing for CRUD operations (e.g., GET, POST, PUT, DELETE).
* Non-functional testing, including performance and security testing.
* Regression testing to ensure new updates do not introduce bugs.
* Integration testing to verify API compatibility with other components.

**Test Strategies**

|  |  |
| --- | --- |
| Test type | Objective |
| Functional Testing | For manual functional testing and automation using Newman CLI. |
| RestAssured | For load and performance testing. |
| Apache JMeter | For security testing and vulnerability scanning. |
| SonarQube | For static code analysis and identifying potential security flaws. |
| K6/Locust | For stress testing APIs under high loads. |

**Test Data**

* **Sample Requests**: Prepare test cases for both valid and invalid inputs.
* **Test Data Scenarios**:
  + Valid input data for successful operations.
  + Invalid input data to test error handling.
  + Edge cases, such as large payloads or extreme numbers.

**Automation Strategy**

1. **Set Up Test Automation Framework**:
   * Use **RestAssured** for writing automated API test scripts.
   * Integrate the scripts into the CI/CD pipeline using tools like Jenkins or GitHub Actions.
2. **Test Case Execution**:
   * Automate regression and functional test cases.
   * Schedule regular security and performance tests as part of the development lifecycle.
3. **Test Reports**:
   * Generate detailed reports using tools like **Allure Reports** or **Newman**.

**Test Metrics**

To evaluate the effectiveness of the tests, the following metrics will be tracked:

* **Pass/Fail Rate**: Percentage of test cases that passed vs. failed.
* **Response Time**: Average response time of API endpoints.
* **Error Rate**: Percentage of failed requests during stress testing.
* **Code Coverage**: Percentage of API functionality covered by tests.