**Requirement Document For Api Testing**

**Non-Functional Requirements**

**1. Performance Requirements**

Performance benchmarks for Web APIs are essential to ensure that they are reliable, responsive, and efficient. Below are key performance requirements:

* **Response Time:**
  + The API should respond to 95% of requests within 200ms for GET requests and within 500ms for POST/PUT/DELETE requests.
* **Throughput:**
  + The system should handle at least 100 concurrent users with a throughput of 50 requests per second under normal load conditions**.**
* **Scalability:**
  + The API should scale horizontally to support a 25% increase in users without performance degradation.
* **Error Rate:**
  + The error rate for API calls should not exceed 0.5% under normal operation**.**
* **Stress Testing:**
  + The API should remain functional under peak load conditions of up to 50 concurrent users for a sustained period of 9 Seconds.

**2. Security Requirements**

To secure your RESTful API, you can follow the OWASP REST API Security Checklist. Below are some key security requirements:

* **Authentication and Authorization:**
  + Use JWT for securing APIs and ensuring only authorized users have access.
  + Implement role-based access control (RBAC) to restrict sensitive actions to authorized roles.
* **Input Validation:**
  + Validate all incoming data (e.g., query parameters, request bodies, headers) to prevent SQL Injection and buffer overflow attacks.
* **Data Encryption:**
  + Enforce TLS 1.2/1.3 for all communications to ensure encryption of sensitive data in transit.
* **Rate Limiting and Throttling:**
  + Implement rate limiting to prevent abuse and throttling for controlling excessive requests from clients.
* **Error Handling:**
  + Do not expose sensitive information in error messages. Use generic error codes like 401 (Unauthorized), 403 (Forbidden), etc.
* **Logging and Monitoring:**
  + Log all API access attempts (both successful and unsuccessful) and monitor them for suspicious activity using tools like ELK Stack or Splunk.
* **Other OWASP Security Recommendations:**
  + Avoid exposing sensitive endpoints.
  + Disable unused HTTP methods (e.g., TRACE, OPTIONS).

**Software Test Plan for RESTful Web APIs**

**1. Introduction**

This document defines the testing strategy for RESTful APIs. The focus will be on verifying API functionality, performance, security, and scalability.

**2. Objectives**

* Ensure the APIs function as expected with valid and invalid inputs.
* Verify performance metrics under normal and peak loads.
* Identify and fix vulnerabilities to secure sensitive data and API access.
* Automate API testing for faster and more consistent testing cycles.

**3. Testing Strategies**

**a. Functional Testing**

* Verify all CRUD operations (GET, POST, PUT, PATCH, DELETE) for each endpoint.
* Test for correct HTTP status codes and responses (e.g., **200**, **201**, **400**, **404**, **500**).
* Validate business logic by testing edge cases and corner cases.
* Test API data validation rules (e.g., required fields, data type mismatches, etc.).

**b. Performance Testing**

* Use tools like **JMeter** to test the following:
  + **Load Testing:** Evaluate API performance under expected load conditions.
  + **Stress Testing:** Identify how the API performs under extreme loads.
  + **Spike Testing:** Test API response to sudden surges in traffic.
  + **Latency Testing:** Measure response times for various operations.

**c. Security Testing**

* Use tools like **OWASP ZAP** or **Burp Suite** for:
  + Testing authentication mechanisms.
  + Verifying token expiration and refresh logic.
  + Checking for vulnerabilities like **SQL injection**, **CSRF**, and **XSS**.
  + Testing CORS policies to prevent unauthorized domain access.

**d. Integration Testing**

* Test how the API integrates with:
  + External services (e.g., payment gateways, third-party APIs).
  + Databases and other backend systems.

**f. Compatibility Testing**

* Verify that the APIs work correctly across different browsers, devices, and network environments.

**g. Usability Testing**

* Validate error messages for user-friendliness.
* Ensure API documentation (e.g., Swagger/OpenAPI) is clear and accurate.

**4. Test Automation Plan**

**a. Tools**

* **Postman/Newman:** For functional testing and integration testing.
* **RestAssured:** For Java-based test automation of REST APIs.
* **JMeter:** For performance testing.
* **OWASP ZAP/Burp Suite:** For security testing.
* **Allure Report:** For detailed and visually rich test reports.

**b. CI/CD Integration**

* Integrate API test automation into CI/CD pipelines using **Jenkins**, **GitHub Actions**, or **GitLab CI**.
* Run automated test suites for every commit to ensure the API is stable.

**c. Reporting**

* Use reporting tools (e.g., **ExtentReports**, **Allure**, **Grafana**) to present test results in a user-friendly manner.

**5. Deliverables**

* Functional test cases for each API endpoint.
* Automated test scripts for all critical endpoints.
* Performance benchmark results (e.g., response time, throughput).
* Security vulnerability scan reports.
* Test execution logs and summary reports.

**6. Testing Metrics**

* **Test Coverage:** % of API endpoints covered by tests.
* **Pass Rate:** % of test cases that pass during test execution.
* **Defect Density:** Number of bugs found per API endpoint.
* **Performance Benchmarks:** Average response time, error rate, throughput.
* **Security Metrics:** Number of vulnerabilities identified and fixed.

**7. Risks and Mitigation**

* **Risk:** Incomplete test coverage.  
  **Mitigation:** Conduct peer reviews of test cases and automate regression tests.
* **Risk:** Performance degradation under high load.  
  **Mitigation:** Optimize database queries and implement caching mechanisms.