**TEST PLAN FOR NUMBER CONVERSION API**

**1. Introduction**

**1.1 Objective**

The objective of this test plan is to define the testing strategy for the Number Conversion API, which converts numerical inputs into their corresponding word representation. The goal is to:

* Validate API functionality, performance, and reliability.
* Ensure proper error handling and boundary conditions.
* Conduct security assessments and compliance testing.
* Measure response times under varying loads.

**1.2 API Overview**

* **Type:** SOAP Web Service
* **Endpoint:** https://www.dataaccess.com/webservicesserver/NumberConversion.wso
* **Request Format:** XML (SOAP)
* **Response Format:** XML
* **Authentication:** No authentication required
* **Methods Supported:**
  + NumberToWords(ubiNum): Converts a number to words.
  + NumberToDollars(ubiNum): Converts a number to dollar format.

**2. Scope**

**2.1 In-Scope**

* **Functional Testing:** Verify that the API correctly converts numbers into words.
* **Boundary Testing:** Test API limits for small, large, and extreme values.
* **Error Handling:** Ensure the API correctly handles invalid inputs.
* **Performance Testing:** Measure API response times under different loads.
* **Security Testing:** Validate API against common vulnerabilities (e.g., injection attacks).
* **Regression Testing:** Ensure that new changes do not introduce defects.

**2.2 Out of Scope**

* **Database testing:** The backend database logic is outside the scope.
* **UI testing:** Since this is a web service, front-end UI is not tested.
* **Integration with other services:** Only the Number Conversion API itself is tested.

**3. Test Strategy**

**3.1 Testing Approach**

The API will be tested using a **black-box testing** approach, focusing on the request and response structure without accessing internal code.

**3.2 Types of Testing**

| **Test Type** | **Description** |
| --- | --- |
| **Functional Testing** | Verify that the API returns correct results for valid inputs. |
| **Boundary Testing** | Validate API behavior with minimum, maximum, and out-of-range values. |
| **Negative Testing** | Send invalid, malformed, or missing parameters and check API responses. |
| **Performance Testing** | Check response times under normal and high loads using JMeter. |
| **Security Testing** | Ensure data integrity and check for vulnerabilities like injection attacks. |
| **Regression Testing** | Run tests after changes to ensure no new defects are introduced. |

**3.3 Test Data Strategy**

* **Valid Data:** 0, 123, 999, 1000000
* **Boundary Data:** -1, 2147483647 (max int), -2147483648 (min int)
* **Invalid Data:** "abc", "" (empty), null, special characters (@#%!)

**4. Test Environments**

| **Component** | **Details** |
| --- | --- |
| **Operating Systems** | Windows 10, macOS, Linux |
| **Browsers** | Chrome, Firefox, Edge (for API tools) |
| **Network Conditions** | Wi-Fi, Cellular, Wired |
| **Testing Tools** | Postman, SOAP UI, JMeter, Curl |
| **Logging Tools** | Splunk, Portainer |
| **Defect Tracking** | JIRA, Bugzilla |

**5. Test Cases**

**5.1 Functional Test Cases**

| **Test Case ID** | **Scenario** | **Input** | **Expected Output** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_01 | Convert a valid number | 123 | "One Hundred Twenty-Three" | Pass/Fail |
| TC\_02 | Convert zero | 0 | "Zero" | Pass/Fail |
| TC\_03 | Convert a large number | 999999999 | "Nine Hundred Ninety-Nine Million Nine Hundred Ninety-Nine Thousand Nine Hundred Ninety-Nine" | Pass/Fail |

**5.2 Negative Test Cases**

| **Test Case ID** | **Scenario** | **Input** | **Expected Output** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_04 | Pass a non-numeric value | "abc" | Error: Invalid input | Pass/Fail |
| TC\_05 | Pass a negative number | -5 | Error: Not Supported | Pass/Fail |
| TC\_06 | Send an empty request | "" | Error: Missing Input | Pass/Fail |

**6. Defect Reporting**

**6.1 Defect Classification**

| **Severity** | **Description** |
| --- | --- |
| **Critical** | API is down or does not return a response. |
| **High** | Incorrect conversion logic. |
| **Medium** | Slow response times. |
| **Low** | Minor formatting issues in response. |

**6.2 Defect Logging Procedure**

1. Log the defect in **JIRA** or **Bugzilla**.
2. Provide clear **steps to reproduce**.
3. Attach **screenshots, request payload, and response**.
4. Assign severity and priority.
5. Developer fixes and tester retests.

**7. Entry & Exit Criteria**

**7.1 Entry Criteria**

* API is deployed in a **test environment**.
* Test data is prepared.
* Test cases are **reviewed and approved**.

**7.2 Exit Criteria**

* All test cases are executed.
* No **critical defects** remain open.
* Final **test summary report** is generated.

**8. Tools**

| **Tool** | **Usage** |
| --- | --- |
| **Postman** | Functional API testing |
| **SOAP UI** | SOAP request validation |
| **JMeter** | Performance testing |
| **Curl** | Manual request testing |
| **JIRA** | Defect tracking |

**9. Risks & Mitigation**

| **Risk** | **Mitigation** |
| --- | --- |
| API downtime | Use mock servers for testing |
| Slow response times | Optimize API calls and load balancing |
| Incorrect number conversion | Validate against multiple test cases |

**10. Approvals**

| **Role** | **Name** | **Approval Status** |
| --- | --- | --- |
| Test Lead | [Your Name] | ✅ Approved |
| QA Manager | [Manager Name] | ✅ Approved |
| Project Manager | [PM Name] | ✅ Approved |

**11. Conclusion**

This test plan ensures that the Number Conversion API is thoroughly tested for functionality, performance, security, and reliability. All identified defects will be tracked and resolved before final deployment.