**Test Plan Document for testing GraphQL API**

**1.** **Introduction**

This test plan outlines the functional testing approach for the GraphQL API. The objective is to ensure that the API correctly processes queries, mutations, and handles errors effectively and ensures data integrity.

**2.** **Scope**

This test plan focuses on:

* Query operations
* Mutation operations
* Input validation
* Error handling
* Performance considerations

**3. Approach**

Testing Strategy:

* Automated testing for functional, regression and performance
* Perform manual exploratory testing.
* Conduct edge-case testing.

Test Automation strategy:

* Automated service level component tests: These tests will be executed as a part of CI

pipeline, mocking all external dependencies.

* These tests will be completed along with the development phase.
* Automated End-to-End tests: These tests will be performed in an integration

environment like QA where all services are deployed and test data can be managed.

* The tests will be grouped into different subgroups eg: Smoke test, Regression test,

Feature-based etc enabling QAs to run automation on a targeted section as and when

required.

* Framework needs to be set up and Tests need to be written before system integration

phase.

Testing Types:

| **Testing Types** | **Required for Release** | **Notes** |
| --- | --- | --- |
| Unit Testing | ✅ |  |
| Component Testing | ✅ | Isolated testing for the service by mocking external dependencies |
| System Integration testing | ✅ |  |
| Regression Testing | ✅ |  |
| Exploratory Testing | ✅ |  |
| Performance Testing | ✅ |  |
| Production Smoke Test | ✅ |  |
| UAT Testing | ✅ |  |

**Entry and Exit Criteria:**

Entry Criteria:

● Test environment is set up

● Test cases are prepared and reviewed

● Build is stable

Exit Criteria:

● All test cases executed

● Critical bugs fixed

● Test reports reviewed and approved

**4.** **Test Environment**

* API Endpoint: http://demo-api.tc.staging.catapult.com:3000/graphql
* Database: [DB type/version]
* Tools: DogAPI, Playground, JAVA, RestAssured,Cucumber, JUnit, Gatling.

**5. Test Cases**

### **5.1 Feature: Validate Books API**

| **Test Case ID** | **Scenario** | **Expected Result** |
| --- | --- | --- |
| 1 | Validate bookAll query | Returns a list of books with correct attributes |
| 2 | Validate bookById query | Returns correct book details or an error if not found |
| 3 | Validate createBook Mutation | Successfully creates a book and returns its details |
| 3.1 | Validate createBook Mutation with creating new auther | Successfully creates a book and its auther |
| 3.2 | Validate createBook Mutation with connecting existing auther | Successfully creates a book and connects an existing auther |
| 4 | Validate updateBook Mutation | Updates the book with new values |
| 5 | Validate removeBook mutation | Deletes a book and confirms deletion |

### 

### 

### 

### **5.2 Feature: Validate Author API**

| **Test Case ID** | **Scenario** | **Expected Result** |
| --- | --- | --- |
| 6 | Validate authorAll query | Returns a list of authors with correct attributes |
| 7 | Validate authorById query | Returns correct author details or an error if not found |
| 8 | Validate createAuthor Mutation | Successfully creates an author and returns details |
| 9 | Validate updateAuthor Mutation | Updates author and verifies changes |
| 10 | Validate removeAuthor mutation | Deletes an author and confirms deletion |

### 

### **5.3 Feature: Validate book and author**

| **Test Case ID** | **Scenario** | **Expected Result** |
| --- | --- | --- |
| 11 | Validate books are listed for the author correctly when the author was created or linked during the creation of the book | Linking between the author and book should be correct |
| 12 | Validate author is listed for the book correctly when the book was created or linked during the creation of the author | Linking between the book and author should be correct |
| 13 | Validate success while trying to create existing books during createAuthor Mutation with skipDuplicates set to true | Auter gets created successfully |
| 14 | Validate author update should be reflected in the queryBook response | Updates get reflected |
| 15 | Validate author delete should be reflected in the queryBook response | Updates get reflected |

### 

### **5.4 Input Validation**

| Test Case ID | Scenario | Expected Result |
| --- | --- | --- |
| 16 | Missing required fields | Returns validation error |
| 17 | Invalid data types | Returns type error |
| 18 | Exceeding character limits | Returns validation error |

### 

### 

### 

### **5.5 Error Handling**

| Test Case ID | Scenario | Expected Result |
| --- | --- | --- |
| 19 | Query non-existent book | Returns an appropriate error message |
| 20 | Invalid query syntax | Returns syntax error |
| 21 | Validate error thrown for duplicate createBook Mutation | Returns an appropriate error message |
| 22 | Validate error thrown for duplicate createAuthor Mutation | Returns an appropriate error message |
| 23 | Validate error thrown while trying to create an existing author during createBook Mutation | Returns an appropriate error message |
| 24 | Validate error thrown while trying to connect an invalid author during createBook Mutation | Returns an appropriate error message |
| 25 | Validate error thrown while trying to create an existing book during createAuthor Mutation | Returns an appropriate error message |
| 26 | Validate error thrown while trying to connect an invalid book during createAuthor Mutation | Returns an appropriate error message |
| 27 | Validate error thrown while trying to create existing books during createAuthor Mutation | Returns an appropriate error message |

### 

**5.** **Test Data**

* Sample books and authors with varied attributes.
* Edge-case data such as empty fields, long strings, and special characters.

**6. Reporting and Tracking**

* Use test management tools (JIRA, TestRail) for tracking.
* Document test results, issues, and resolutions.

**7. Test Schedule**

Milestones:

● Test Plan Review: [Date]

● Test Case Development: [Date]

● Test Execution: [Date]

● Bug Fixes and Re-testing: [Date]

● Final Review and Approval: [Date]

Test Deliverables:

● Test Plan Document

● Test Cases and Test Scripts

● Test Reports

● Bug Reports

**8.** **Key Risks**

● Unavailability of test Data

● Test Environment Outage

● Delays in build delivery

● Unexpected bugs in third-party services

● Outage in downstream services

**9. Conclusion**

This structured test plan aims to ensure that the GraphQL API operates as expected, handling all edge cases efficiently. Continuous testing and automation will improve overall quality.