**Test Plan Document - iTunes Store API: Constructing Searches**

**1. Introduction**

* Purpose: The purpose of this test plan is to verify the functionality of the iTunes Store API for constructing searches and displaying results.
* Scope: The testing will focus on constructing search queries, sending requests, parsing JSON responses, and validating the results.
* Objectives: To ensure that the search functionality works as expected, returns accurate results, and handles various input scenarios.

**2. Test Environment**

* API Endpoint: The fully-qualified URL for the iTunes Store search API.
* Tools and Software Requirements:
  + REST-Assured to test REST services
  + Jackson databind dependency to achieve Serialization and Deserialization( parsing JSON responses)
  + Eclipse/IntelliJ IDE for project development in java
  + Maven Cucumber reporting from <https://github.com/damianszczepanik/maven-cucumber-reporting> to generate test reports
  + TestNG dependency for assertions
  + Cucumber and Junit dependencies to create cucumber project and run Junit tests.

**3. Test Scenarios**

* Scenario 1: Search for content by providing the "term" parameter with valid text string.
* Scenario 2: Search for content by providing the "country" parameter with valid two-letter country codes.
* Scenario 3: Search for content by specifying the "media" parameter with valid media types.
* Scenario 4: Search for content by specifying the "entity" parameter with valid entities based on the media type.
* Scenario 5: Search for content by specifying the "attribute" parameter with valid attributes based on the media type.
* Scenario 6: Search for content with a combination of multiple parameters.
* Scenario 7: Search for content and verify the number of search results using the "limit" parameter.

**4. Test Data**

* Prepare test data for each test scenario based on the required parameter values from the below table.
* Include test data that covers different media types, entities, attributes, and parameter combinations.

The following table defines the parameter keys and values you can specify to search for content within the iTunes Store, App Store, iBooks Store and Mac App Store:

| **Parameter Key** | **Description** | **Required** | **Values** |
| --- | --- | --- | --- |
| term | The URL-encoded text string you want to search for. For example: jack+johnson. | Y | Any URL-encoded text string.  Note: URL encoding replaces spaces with the plus (+) character and all characters except the following are encoded: letters, numbers, periods (.), dashes (-), underscores (\_), and asterisks (\*). |
| country | The two-letter country code for the store you want to search. The search uses the default store front for the specified country. For example: US. The default is US. | Y | [See http://en.wikipedia.org/wiki/ ISO\_3166-1\_alpha-2](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2) for a list of ISO Country Codes. |
| media | The media type you want to search for. For example: movie. The default is all. | N | movie, podcast, music, musicVideo, audiobook, shortFilm, tvShow, software, ebook, all |
| entity | The type of results you want returned, relative to the specified media type. For example: movieArtist for a movie media type search. The default is the track entity associated with the specified media type. | N | [For a list of available entitites, see Table 2-1](https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/iTuneSearchAPI/Searching.html#//apple_ref/doc/uid/TP40017632-CH5-SW2). |
| attribute | The attribute you want to search for in the stores, relative to the specified media type. For example, if you want to search for an artist by name specify entity=allArtist&attribute=allArtistTerm. In this example, if you search for term=maroon, iTunes returns “Maroon 5” in the search results, instead of all artists who have ever recorded a song with the word “maroon” in the title.  The default is all attributes associated with the specified media type. | N | [For a list of available attributes, see Table 2-2](https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/iTuneSearchAPI/Searching.html#//apple_ref/doc/uid/TP40017632-CH5-SW3). |
| callback | The name of the Javascript callback function you want to use when returning search results to your website. | Y, for cross-site searches | wsSearchCB |
| limit | The number of search results you want the iTunes Store to return. For example: 25. The default is 50. | N | 1 to 200 |
| lang | The language, English or Japanese, you want to use when returning search results. Specify the language using the five-letter codename. For example: en\_us. The default is en\_us (English). | N | en\_us, ja\_jp |
| version | The search result key version you want to receive back from your search. The default is 2. | N | 1,2 |
| explicit | A flag indicating whether or not you want to include explicit content in your search results. The default is Yes. | N | Yes, No |

**5. Test Execution**

* Automation Test Execution:
  + Cucumber Test Runner - Create Cucumber test runner to execute test scenarios with/without the help of tags
  + Implement Maven Command Line run facility to run the tests from cmd with the help of maven commands
  + Implement cucumber reporting mechanism for graphical representation of result reports

**6. Test Reporting**

Implement maven cucumber reporting mechanism to achieve below:

* Test Execution Reports: Document the execution status of each test scenario.
* Representation: Graphical representation of result reports for easy and better understanding of executed scenarios.
* Defect Reports: Log any defects or issues encountered during testing.

**7. Risks and Mitigation**

NA

**8. Test Schedule**

NA

**9. Test Closure**

NA

**10. Appendix**

* References:
  + Maven Cucumber reporting from <https://github.com/damianszczepanik/maven-cucumber-reporting> to generate test reports