# Test Plan for Football Data API

### 1. Introduction

The Football Data API Test Plan outlines the testing approach, objectives, scope, and activities for validating the Football Data API. The primary focus is on functional, security, performance, and compatibility testing to ensure the API's reliability and correctness.

### 2. Objectives

The objectives of testing the Football Data API are as follows:

* Functional Testing: Verify the correctness of API endpoints and their functionality.
* Security Testing: Ensure that the API enforces proper authentication and authorization mechanisms.
* Compatibility Testing: Confirm that the API works consistently across various platforms and devices.

### 3. Scope

The test scope includes the following:

* API Endpoints: Test all relevant endpoints for areas, competitions, matches, and other resources.
* Authentication and Authorization: Verify the effectiveness of the authentication and authorization mechanisms.
* Data Validity: Ensure the correctness and completeness of the data returned by the API.

### 4. Test Types

#### **4.1 Functional Testing**

Area Endpoints Testing:

* Retrieve data for all areas.
* Retrieve data for a single area.
* Verify error handling for invalid or missing authentication tokens.

Competition Endpoints Testing:

* Retrieve data for all competitions.
* Retrieve data for a single competition.
* Verify error handling for authorization issues.

Match Endpoints Testing:

* Retrieve match data for all competitions.
* Verify error handling for various scenarios in match data retrieval.

#### **4.2 Security Testing**

Authentication Testing:

* Confirm that valid authentication tokens are required.
* Verify that valid tokens grant appropriate access.
* Test error responses for invalid or missing tokens.

Authorization Testing:

* Validate that users have the necessary permissions to access specific resources.
* Ensure sensitive information is not exposed in error messages.

#### **4.3 Compatibility Testing**

Browser Compatibility Testing:

* Verify that the API works consistently across different browsers.
* Device Compatibility Testing:
* Confirm that the API is accessible on various devices and platforms.

### 5. Test Environment

* Programming Language: Python
* Testing Framework: pytest
* Dependencies: requests library
* Version Control: Git

### 6. Test Data

* Use valid data for successful API requests.
* Use invalid data to trigger error responses.
* Include data with different permission levels for authorization testing.

### 7. Test Execution

* Execute tests using the pytest framework.
* Use automation for repetitive and complex test scenarios.

### 8. Test Deliverables

* Test scripts for functional, security, performance, and compatibility testing.
* Test reports with detailed results, issues, and recommendations.

### 9. Test Schedule

* Unit Testing: Verify individual API endpoints and functions.
* Integration Testing: Validate the interactions between different API components.
* System Testing: Comprehensive testing of the entire system.

### 10. Risks and Assumptions

* Assumption: The API documentation accurately reflects the expected behaviour.
* Risk: Changes to the API endpoints during testing may impact test cases.

**11. Sign-Off Criteria**

* All test cases executed.
* Test reports reviewed and approved.
* No critical defects outstanding.

**12. Review and Update**

Regularly review and update the test plan to reflect any changes in the API or testing requirements.