Test Strategy for Football Data API Test Suit

### Introduction

The Football Data API Test Suite is designed to validate the functionality and correctness of the Football Data API endpoints related to areas, competitions, matches, and other resources. This document outlines the test strategy to ensure comprehensive testing and reliable API validation.

### Test Objectives

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

* Functional Testing: Verify that each API endpoint functions as expected, providing accurate and relevant data.
* Security Testing: Ensure that the API enforces proper authentication and authorization mechanisms.
* Error Handling: Validate that the API handles various error scenarios gracefully and provides appropriate error messages.
* Performance Testing: Assess the performance of API responses under normal and peak loads.
* Compatibility Testing: Confirm that the API is compatible with different platforms, browsers, and devices.

### Test Scope

The test scope includes the following aspects:

* API Endpoints: Test all relevant endpoints such as areas, competitions, matches, etc.
* Authentication: Verify the API's authentication mechanisms, including valid and invalid token scenarios.
* Authorization: Ensure that users have the appropriate permissions to access specific resources.
* Data Validity: Validate the correctness and completeness of the data returned by the API.

### 4. Test Types

#### **4.1** **Functional Testing**

Functional testing will cover the following scenarios:

* Retrieving data
* Handling invalid or missing authentication tokens.
* Handling authorization issues

#### **4.2 Security Testing**

Security testing will include:

* Validating that the API requires proper authentication for access.
* Verifying that valid tokens grant appropriate access.
* Confirming that invalid tokens result in the expected error responses.
* Ensuring that sensitive information is not exposed in error messages.

#### **4.3 Performance Testing**

Performance testing will involve:

* Assessing the API response times under normal load conditions.
* Analysing the API's ability to handle a higher volume of requests.
* Identifying any performance bottlenecks and suggesting optimizations.

**4.4 Compatibility Testing**

Compatibility testing will cover:

* Verifying that the API works consistently across different browsers.
* Confirming that the API is accessible on various devices and platforms.

### 5. Test Environment

The test environment includes:

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

### 6. Test Data

Test data required for a successful run include:

* Valid data for successful API requests.
* Invalid data to trigger error responses.
* Data with different permission levels for authorization testing.

### 7. Test Execution

Tests will be executed using pytest, either individually, using appropriate marks (tags) or as part of the entire test suite. Test execution will be automated to ensure repeatability and efficiency.

### 8. Test Deliverables

The following deliverables will be produced:

* Automation regression suite
* Test scripts for functional, security, performance, and compatibility testing.
* Test reports containing results, issues, and recommendations.

### 9. Entry & Exit Criteria User stories

* to be tested must meet the defined 'Ready for Testing' criteria.
* Testing completes when all test cases execute with no critical defects outstanding.

### 10. Review and Update

This test strategy document will be reviewed regularly and updated as needed to reflect any changes in the API or testing requirements.