STP Document

Table of Contents

Planned Schedule

A step in the project process	Start date	End date
Preparation of STP document	08.03.2024	08.03.2024
Preparation of STD document	08.03.2024	08.03.2024
Round of tests #1	08.03.2024	08.03.2024
Round of tests #2	08.03.2024	08.03.2024
Round of tests #3	08.03.2024	08.03.2024
Round of tests #4	09.03.2024	09.03.2024
Round of tests #5	09.03.2024	09.03.2024
Round of tests #6	09.03.2024	09.03.2024

Purpose Of This Document

The purpose of this document is to outline a structured approach for testing the collegefootballdata API. It aims to ensure the API's reliability, efficiency,

and correctness, providing a seamless experience for users accessing college football data.

Description of the College Football Data API

The College Football Data API is a robust and comprehensive solution designed to offer developers and college football enthusiasts access to a wide range of college football data. This API aims to serve as a central hub for retrieving detailed information about teams, players, games, rankings, and much more, catering to the needs of application developers, data analysts, and fans looking for in-depth college football statistics and information.

Key Features and Functions:

The system is composed of three main layers:

1. **Games Information: Access to schedules, scores, and detailed game statistics, including player performances, team stats, and historical data going back several years.**
2. **Team Data: Information on college football teams, including rosters, team statistics, rankings, and team-specific news and updates.**
3. **Player Statistics: Comprehensive player data, including season and career statistics, player bios, and performance metrics.**
4. **Rankings: Up-to-date rankings from major polls and committees, including the AP Top 25 and College Football Playoff rankings.**
5. **Weather Data: Game-specific weather reports, providing insights into how weather conditions might impact game outcomes.**
6. **Venue Information: Details on college football venues, including stadium capacities, locations, and historical data on games played at each venue.**

7. **Media Information: Links to relevant media coverage, including game highlights, interviews, and analysis.**

Glossary and Abbreviations

Glossary

- GUI (Graphical User Interface): The design of user interfaces based on specified requirements.
- Functional Testing: Verification that fundamental system functions operate correctly.
- Maintenance Testing: Examination of the functionality of a modified system following changes, updates, or alterations in the working environment.
- STP (System Test Plan): A comprehensive project planning document encompassing strategy, schedule, and topic tree.
- STD (System Test Design): Detailed documentation outlining the testing plan.
- Traceability Matrix: A document that correlates any two baselined documents that require a many-to-many relationship to determine the completeness of the relationship.

Abbreviations

- QA: Quality Assurance
- CEO: Chief Executive Officer
- HR: Human Resources

Testing Plan

Resources: 1 tester will be assigned to this project.

Hardware/ Software Needs:

1. **Computers:** High-performance desktops or laptops to run test scripts and perform manual testing.
2. **Network Equipment:** Routers and switches to simulate different network conditions for testing.
3. **Virtual Machines:** Set up virtual machines for testing on different operating systems and browser combinations.
4. **Operating Systems:** Windows for testing.
5. **Browsers**: Latest versions of popular browsers (Google Chrome, Microsoft Edge, Safari) for cross-browser compatibility testing.
6. **Database Management System:** Database systems (Postman) for testing data handling and retrieval functionality.

Before the start of testing rounds, **functional tests** will be performed, including:

1. **Unit Testing**: To test individual units or components of the GamerPowerAPI website.
2. **Integration Testing**: To verify the interactions and interfaces between different components or systems within the application.
3. **Regression Testing:** To ensure that new code or changes do not affect the existing functionality of the website.
4. **API Testing:** To validate the functionality of API endpoints by testing their request-response mechanisms.

5. **UI Testing:** To validate that the user interface elements and interactions function correctly according to the design.
6. **End-to-End Testing:** To evaluate the entire system's functionality from start to finish, simulating real user scenarios and interactions.

Afterwards, the following **non-functional tests** will be conducted including:

1. **Performance Tests:** Assess how well the website performs under various conditions, including heavy loads and stressful situations.
2. **Security Testing:** Ensure the website is secure against vulnerabilities such as SQL injection, cross-site scripting, and unauthorized access.
3. **Compatibility Testing:** Ensure the website works seamlessly across different devices, browsers, and operating systems.
4. **Usability Testing**: Evaluate the user-friendliness and overall user experience of the website to ensure it meets user expectations.
5. **Reliability Testing:** Test the stability and reliability of the website under normal and extreme usage conditions.
6. **Scalability Testing:** Test the website's ability to handle increased workload and user traffic without degradation in performance.

Starting and exiting Criteria

- **Criteria for starting the tests:**

- 100% of planned sanity tests were carried out and passed successfully.
- 100% of planned functional and non-functional test cases have been created and reviewed.
- A traceability matrix is established, linking each test case to specific requirements.
- The testing environment is prepared and verified.
- Sufficient and accurate test data for both positive and negative scenarios is available.
- The test plan has been reviewed and approved.
- **Completion/Release Criteria:**
-

- 100% of planned functional and non-functional tests have been executed, and results have been documented.
- 84% of test cases passed successfully.
- The remaining bugs are at low severity levels, with no high-severity issues affecting functionality.

Traceability table

Req. ID	Requirement Description	Test Case ID	Test Case Description	Test Status
REQ-001	Retrieve game data by date	TC-001	Test retrieving game data for specific dates	To Be Done
REQ-002	Access team statistics	TC-002, TC-003	Test accessing overall and specific game statistics for a team	To Be Done
REQ-003	Fetch player performance data	TC-004, TC-005	Test fetching performance data for a specific player across different games	To Be Done
REQ-004	List rankings from major polls	TC-006	Test listing of current rankings from AP Top 25 and Coaches Poll	To Be Done
REQ-005	Provide game-specific weather reports	TC-007	Test retrieval of historical weather data for game locations and dates	To Be Done
REQ-006	Offer detailed venue information	TC-008	Test fetching detailed information on college football venues	To Be Done
REQ-007	Enable access to media information	TC-009	Test access to game highlights, interviews, and analysis through media links	To Be Done
REQ-008	Security and data protection	TC-010, TC-011	Test for common web vulnerabilities and ensure data protection	To Be Done
REQ-009	API performance under peak load	TC-012	Test API response times and error rates under simulated peak loads	To Be Done

Req. ID	Requirement Description	Test Case ID	Test Case Description	Test Status
REQ-010	User authentication and authorization	TC-013	Test authentication mechanisms and authorization levels for different user roles	To Be Done

Test Tree

College Football Data API Test Tree

1. *Games Information*

- **Functional Testing**
  - Retrieve game data by specific dates and teams.
  - Access detailed statistics for individual games.
  - Verify historical data accuracy and completeness.
- **Non-Functional Testing**
  - Performance: Evaluate response times for fetching game data.
  - Security: Check for data leaks or unauthorized access points.

2. *Team Data*

- **Functional Testing**
  - Fetch team statistics, including win/loss records and rankings.
  - Retrieve team rosters and player information.
- **Non-Functional Testing**
  - Usability: Assess the ease of accessing team data.
  - Compatibility: Ensure data is accessible across different devices and platforms.

3. *Player Statistics*

- **Functional Testing**
  - Access individual player performance data.
  - Compare player statistics across different games or seasons.
- **Non-Functional Testing**
  - Scalability: Test the system's capacity to handle large numbers of simultaneous requests for player data.

4. *Rankings*

- **Functional Testing**
  - Retrieve current and historical rankings from major polls.
  - Verify the update frequency and accuracy of ranking data.
- **Non-Functional Testing**
  - Performance: Measure the time taken to update rankings after games.

5. *Weather Data*

- **Functional Testing**
  - Fetch weather reports for past and upcoming games.
  - Assess the impact of weather on game statistics.
- **Non-Functional Testing**
  - Reliability: Ensure weather data is consistently accurate and timely.

6. *Venue Information*

- **Functional Testing**
  - Access detailed information about game venues.
  - Verify venue capacity, location, and historical game data.
- **Non-Functional Testing**
  - Compatibility: Test venue data displays correctly on various devices.

7. *API Performance and Security*

- **Performance Testing**
  - Assess API response times under peak load.
  - Evaluate the efficiency of data retrieval and processing.
- **Security Testing**
  - Identify and mitigate potential security vulnerabilities.
  - Ensure data integrity and protection against common web threats.

Hazard Table

#	Hazard	Probability	Impact	Risk Level	Hazard Description	Mitigation Action	Responsible
1	API Downtime	Medium (0.5)	High (9)	Moderate	API becomes unavailable during testing, impacting progress.	Implement retry logic and alternate testing times.	DevOps Team
2	Incomplete Test Coverage	Low (0.2)	High (9)	Low	Some API functionalities might not be thoroughly tested.	Review and update test cases regularly.	QA Lead
3	Data Inconsistency	Medium (0.5)	High (9)	Moderate	Inconsistent test data leads to unreliable test results.	Validate and standardize test data before testing.	Data Analyst
4	Overloaded Testing Environment	High (0.7)	Medium (7)	High	The testing environment may not handle the load, affecting performance tests.	Enhance testing infrastructure and monitor load.	Infrastructure Team
5	Security Vulnerabilities	Medium (0.5)	Very High (10)	High	Potential exposure to security threats and data breaches.	Conduct thorough security audits and address findings.	Security Team
6	Third-party Integration Failures	Medium (0.5)	High (9)	Moderate	Failures in third-party services could affect testing.	Establish mock services for third-party integrations.	Integration Specialist
7	Regulatory Compliance Issues	Low (0.3)	Very High (10)	Moderate	Non-compliance with data protection laws could lead to legal issues.	Ensure compliance with all relevant regulations.	Legal Advisor
8	Resource Unavailability	Medium (0.5)	High (9)	Moderate	Key personnel or testing tools not available when needed.	Plan resources adequately and have backups ready.	Project Manager

#	Hazard	Probability	Impact	Risk Level	Hazard Description	Mitigation Action	Responsible
9	Performance Bottlenecks	High (0.8)	High (9)	Very High	Undetected performance issues may impact user experience.	Use performance profiling tools and optimize accordingly.	Performance Engineer
10	Inaccurate Test Results	Medium (0.5)	High (9)	Moderate	Incorrect interpretation of test results leading to wrong conclusions.	Review test results meticulously and conduct peer reviews.	QA Analyst

- **Probability**: Estimated likelihood of the hazard occurring (Low, Medium, High).
- **Impact**: Potential impact on the project if the hazard occurs (1-10 scale).
- **Risk Level**: Overall risk posed by the hazard (Low, Moderate, High, Very High).