SOFTWARE TEST PLAN (STP)

NASA API website

Nidaa AbuSaleh

TABLE OF CONTENTS

INTRODUCTION

1. Purpose:

The purpose of this Software Test Plan (STP) is to define the strategy, scope, resources, and timeline for conducting quality assurance (QA) tests through automation on the NASA API website (https://api.nasa.gov/). The primary goal is to ensure the website's functionality, reliability, and performance meet specified standards, and any identified issues are effectively addressed.

2.Project Overview:

This project involves the implementation of QA automation tests for the NASA API website. The API provides access to a wealth of NASA's data, including imagery, astronomy information, and space-related content. The automation tests will comprehensively cover different aspects of the API, ensuring that it operates as intended, meets user expectations, and remains reliable across various scenarios.

3.Website Overview:

The NASA API website serves as a gateway for developers and enthusiasts to access NASA's extensive collection of data and resources programmatically. Users can retrieve information about space missions, celestial bodies, and other space-related content. The website supports interactions through API requests, making it crucial to validate the functionality, accuracy, and reliability of these interactions.

Test Strategies

1.Test Objectives:

The QA automation tests for the NASA API website will focus on the following key objectives:

a. Functionality: Verify that API endpoints return the expected data and respond appropriately to valid and invalid requests.

b. Usability: Assess the clarity and consistency of the API documentation to ensure users can easily understand and implement API requests.
c. Performance: Evaluate the response time of API requests under normal and peak load conditions to ensure timely and efficient data retrieval.
d. Compatibility: Validate the website's compatibility across different web browsers, ensuring a seamless experience for users on various platforms.

2. Feature to be tested:

Module Name	**Applicable Roles**	**Description**
Endpoint Functionality	Developer	Verify the functionality of API endpoints, ensuring they return accurate and relevant data based on specified parameters.
Data Accuracy	Developer	Validate the accuracy and consistency of data retrieved from different API endpoints to ensure reliability.
Documentation Clarity	Developer	Assess the clarity and completeness of API documentation to facilitate ease of use and implementation for developers.
Response Time	Developer	Evaluate the response time of API requests under normal and peak load conditions to ensure optimal performance.
Compatibility	Developer	Validate the compatibility of the API across various programming languages and environments.

Authentication and Authorization	Developer	Ensure that authentication and authorization mechanisms are effective in controlling access to restricted data and functionalities.
Data Format Support	Developer	Test the API's ability to handle different data formats (e.g., JSON, XML) during both requests and responses.
Documentation Examples	Developer	Verify the accuracy and relevance of code examples provided in the API documentation for ease of integration.

3.Test Assumptions:

- The NASA API website is stable and accessible for testing purposes.
- Test environments accurately reflect the production environment in terms of configuration and usage.
- Test data, representative of actual usage scenarios, is available and can be manipulated to simulate various conditions.

4.Test Approach:

- Automation: Utilize automated testing tools, such as Postman or Insomnia, to enhance test efficiency and coverage, focusing on API endpoints and responses.
- Comprehensive Testing: Conduct a combination of functional, compatibility, performance, security, and usability testing to ensure a thorough evaluation of the API's capabilities.

5.Data Approach:

- Synthetic Data: Generate synthetic data to simulate various scenarios, ensuring thorough testing of API responses and handling of diverse data inputs.
- Test Data Management: Use tools and processes for efficient generation, manipulation, and management of test data to support testing requirements effectively.
- Data Privacy: Ensure compliance with data protection regulations, and maintain data privacy and confidentiality throughout the testing process.

6.Levels of Testing:

- Unit Testing: Validate individual API endpoints and functions to ensure they operate as expected, providing accurate and reliable data.
- Integration Testing: Verify the integration of different API endpoints and functionalities to ensure seamless communication and data flow.
- System Testing: Validate the overall system behavior, encompassing multiple API interactions, to ensure consistency and reliability.
- Acceptance Testing: Involve developers and other stakeholders to conduct acceptance testing, ensuring that the API meets user expectations in terms of functionality, performance, and security.

Execution Strategy

1.Test Execution:

- Automation Framework Selection: Opt for a suitable automation framework for API testing, such as Postman, RestAssured, or other API testing tools, to efficiently automate and validate API endpoints.
- Continuous Integration (CI): Integrate automated API tests into the

CI/CD pipeline, allowing for automatic execution with each code commit. This ensures early detection of potential issues and maintains a reliable testing process.

2.Test Environment:

- Multiple Test Environments: Utilize multiple test environments, including development, staging, and production-like environments, to simulate real-world conditions and test the API under various scenarios.
- Environment Configuration Management: Ensure that test environments accurately replicate the production environment in terms of configuration and usage. This includes identical setups for databases, servers, and other dependencies, minimizing environment-related discrepancies.

3.Test Data Management:

- Data Generation: Utilize automated tools to generate diverse and realistic test data, covering various scenarios to ensure thorough testing of API responses and functionalities.
- Data Reusability: Promote the reusability of test data across different test scenarios, reducing duplication efforts and enhancing efficiency in testing. Maintain a centralized repository for test data to ensure consistency.

4.Test Execution Process:

- Test Prioritization: Prioritize API test cases based on risk and criticality, employing risk-based testing methodologies. Focus testing efforts on high-impact areas to ensure comprehensive coverage of critical functionalities.
- Test Execution Flow: Define a standardized test execution flow, specifying preconditions, test steps, and expected outcomes using tools like Postman or other API testing frameworks. Ensure consistency in test execution to facilitate reusability and maintainability of test cases.

5.Test Coverage:

Coverage Analysis: Analyze test coverage reports generated from API testing tools, identifying areas of the API that lack adequate test coverage. Ensure that all critical functionalities, API endpoints, and potential edge cases are thoroughly tested, providing confidence in the overall reliability of the NASA API.

TEST CRITERIA

- Suspension Criteria:

  Test Case Failure: If 20% or more of the API test cases fail, testing will be suspended until the development team addresses and resolves all the failed cases.

  User Experience: If the tests indicate a significant degradation in the quality of user experience or functionality, testing may be temporarily suspended until issues are addressed.

- Exit Criteria:
  - Run Rate: The run rate for API tests must be 100%, unless there is a clear and documented reason for not achieving this rate. This ensures that all planned tests are executed, providing comprehensive coverage.
  - Pass Rate: The pass rate for API tests should be 80% or higher. Achieving the pass rate is mandatory to proceed to the next phase, demonstrating that the API meets the specified quality standards

Human Resource

Member	Tasks
Project Manager	Oversee project, define goals, secure resources for website testing.
Test Team Lead	Lead team in test strategy, execution, defect reporting. Coordinate with outsourced teams. Verify and assess Test Approach.
Test Engineers	Develop and execute test cases, collaborate with developers for comprehensive coverage.
SQA Analysts	Oversee quality assurance, ensure alignment with requirements, provide continuous improvement feedback.

SCHEDULE & ESTIMATION

The testing activities will be conducted according to the following schedule:
a. Test Planning: 07/03/24
b. Test Development: 08/03/243
c. Test Execution: 08/03/23
d. Defect Reporting and Resolution: Ongoing throughout the testing phases
e. Final Test Report: 09/03/24