SOFTWARE TEST DESCRIPTION

PetStore Swagger UI Site

Team Members

Project Manager: Yair Amon

Test Team Lead: Tzahi Anedghr

Test Engineers: Wael Otman

TABLE OF CONTENT

1. INTRODUCTION

1.1. PURPOSE

The purpose of the Software Test Documentation (STD) is to provide a comprehensive overview of the testing process, methodologies, strategies, and resources utilized for testing the Swagger Petstore API. This document serves as a guide for the testing team, stakeholders, and other project members, outlining the approach to ensure the quality, reliability, and functionality of the API.

1.2 Scope

The scope of the Software Test Documentation (STD) for the Swagger Petstore API encompasses various aspects related to testing, including:

- Functional Testing: Verification of the API's functionality according to its specifications and requirements. This includes testing of API endpoints, data input/output, error handling, and response validation.
- Non-Functional Testing: Evaluation of non-functional aspects such as performance, security, reliability, and usability of the API. This involves assessing response times, throughput, security vulnerabilities, and user experience.
- Compatibility Testing: Testing the compatibility of the API with different platforms, operating systems, web browsers, and devices to ensure consistent behavior across various environments.
- Integration Testing: Validation of the API's integration with external systems, databases, and third-party services to ensure seamless data exchange and interoperability.
- Regression Testing: Verification that existing functionalities remain intact after changes or updates to the API. This involves rerunning previously executed test cases to detect any regressions.
- API Documentation: Review and validation of the API documentation to ensure completeness, accuracy, and clarity. This includes verifying that all endpoints, parameters, and response formats are accurately documented.
- Test Environment Setup: Configuration and setup of the test environment to replicate production conditions and facilitate testing activities. This involves setting up servers, databases, testing tools, and other necessary infrastructure components.

2. OVERVIEW

2.1.Background

The website https://petstore.swagger.io/ hosts the Swagger Petstore API, demonstrating API development features using modern web technologies. It showcases CRUD operations for managing pets, user authentication, and order management. The API is accompanied by comprehensive documentation and a Swagger UI interface for easy exploration and testing. It serves as a valuable resource for developers to learn about API development and Swagger tools.

2.2.Goals

The goals of the Software Test Documentation (STD) and testing for the Swagger Petstore API site are to:

1. Verify functionality, including CRUD operations and authentication.
2. Ensure compatibility across platforms and devices.
3. Validate security measures.
4. Evaluate performance under varying loads.
5. Review and update documentation for clarity.
6. Assess usability of the Swagger UI interface.
7. Prevent regression issues.
8. Ensure compliance with standards and regulations.
9. Test error handling for user-friendliness.
10. Ultimately, aim for stakeholder satisfaction by delivering a reliable and secure API.

2.3.glossary

- **Swagger Petstore API:** Web service managing pet store inventory, enabling pet-related operations like adding, deleting, and updating pets, as well as managing users and orders.
- **Endpoint:** URL representing a specific action or resource within the API, such as /pets or /users.
- **RESTful:** Architecture emphasizing stateless communication via HTTP methods (GET, POST, PUT, DELETE) for data exchange between systems.
- **API:** Set of rules and tools facilitating communication and data exchange between software systems.
- **JSON:** Lightweight data-interchange format for easy human readability and machine parsing.
- **HTTP:** Protocol for web communication, used by clients to request resources and exchange data with servers.
- **CRUD:** Fundamental operations of persistent storage: Create, Read, Update, Delete.
- **Authorization:** Process determining if a user, application, or system is permitted to perform a requested action or access a resource.

- **Authentication:** Process verifying the identity of a user, application, or system through credentials like username and password.
- **Response Code:** Numeric code indicating the status of a client request, such as 200 for success or 404 for not found.
- **STP (Software Test Plan):**

  Definition: Document outlining testing goals, methods, resources, and schedule.

  Explanation: Guides testing process, detailing what, how, and by whom testing will be conducted.

- **STD (Software Test Description):**

  Definition: Detailed document specifying individual test cases.

  Explanation: Describes test scenarios, inputs, expected outcomes, and execution steps.

- **STR (Software Test Report):**

  Definition: Summary document detailing testing results.

  Explanation: Provides insights into testing outcomes, including defects and overall software quality.

- **Bugs:**

  Definition: Software errors causing unexpected behavior.

  Explanation: Range from minor to critical issues, affecting functionality or security.

- **Test Case:**

  Definition: Set of steps to verify software functionality.

  Explanation: Guides systematic testing of software features.

TEST STRATEGY

2.4.Test Tree

A) Functional tests:

1) Pet Operations

a. Add a new pet.
   - Verify successful addition of a pet with valid information
   - Verify error handling for invalid pet data (e.g., missing required fields)

b. Update an existing pet
    - Verify successful update of pet information
    - Verify error handling for updating non-existent pet

c. Find pet by ID
    - Verify successful retrieval of pet by valid ID
    - Verify error handling for non-existent pet ID

d. Find pet by status
    - Verify successful retrieval of pets by valid status (e.g., available, pending)
    - Verify error handling for invalid status input

2) User Operations

a. Create a new user:
    - Verify successful creation of a user with valid information
    - Verify error handling for invalid user data (e.g., missing required fields)

b. Update an existing user
    - Verify successful update of user information
    - Verify error handling for updating non-existent user

c. Find user by username
    - Verify successful retrieval of user by valid username
    - Verify error handling for non-existent username

3) Place an order

a. Click on a search result to view item details.
    - Verify successful placement of an order with valid information
    - Verify error handling for invalid order data (e.g., missing required fields)

b. Get order by ID
    - Verify successful retrieval of order by valid ID
    - Verify error handling for non-existent order ID

c. Delete an order
    - Verify successful deletion of an order by valid ID
    - Verify error handling for deleting non-existent order

B) Non-Functional tests

1) Performance Testing

a. Load Testing:
- Test the API's response time under various loads (low, medium, high).
- Evaluate the API's throughput and resource utilization under different load conditions.

b. Stress Testing:
- Assess the API's stability and behavior under extreme loads.
- Determine the breaking point and performance degradation threshold.

c. Scalability Testing:
- Evaluate the API's ability to handle increased loads by scaling horizontally or vertically.
- Test the API's performance with increased concurrent users or requests.

2) Usability Testing

a. Documentation Review:
- Evaluate the clarity and completeness of API documentation.

b. API Usability Testing
- Assess the ease of use and intuitiveness of API endpoints and parameters.

c. Error Handling Usability
- Evaluate the user-friendliness of error messages and troubleshooting guidance.

3) Compatibility Testing

a. Browser Compatibility:
- Test the API's compatibility with different web browsers (e.g., Chrome, Firefox, Safari).

b. Platform Compatibility:
- Verify the API's compatibility with different operating systems (e.g., Windows, Linux, macOS).

4) Reliability Testing

a. Stability Testing:
- Assess the API's ability to maintain consistent performance over a prolonged period.

- Test for memory leaks or resource exhaustion under continuous usage.

b. Error Handling Testing:
   - Verify the API's response to unexpected inputs or errors.
   - Test for graceful degradation and appropriate error messages.

c. Recovery Testing
   - Evaluate the API's recovery mechanisms after a system failure or interruption.
   - Test for data consistency and integrity after recovery procedures.

3. Test Steps

Example 1:

Test Case Name	Retrieve Pet by ID
Test Objective	To verify the ability to retrieve pet details by its ID.
Preconditions	Pet with the specified ID exists in the database.
Test Data	Valid pet ID.
Test Steps	
1	Send a GET request to retrieve a pet by its ID.
Expected Result	Receive a successful response with the pet's details.

Step No.	Description	Expected Result
1	Send a GET request to the Swagger Petstore API with the pet's ID.	Receive a successful response with the pet's details.

Example 2:

Test Case	
Test Case Name	To verify the ability to add a new pet to the pet store.
Test Case Objective	To verify the ability to add a new pet to the pet store
Test Data	Valid pet details
Preconditions	Connectivity to the Swagger Petstore API, valid authentication credentials
Author	wael otman
Date	2.5.2022
Priority	1

Step No.	Description	Expected Result
1	Access the Swagger Petstore API and navigate to the "Add Pet" endpoint.	The "Add Pet" endpoint is accessible and ready for use.
2	Send a POST request to the "Add Pet" endpoint with valid pet details.	Receive a successful response indicating that the pet was added to the pet store.
3	Retrieve the added pet by its ID using a GET request.	Receive a successful response with the details of the added pet.
4	Verify that the details of the added pet match the provided input.	The details of the retrieved pet match the details provided in the POST request for adding a new pet.

4. Treacability

Test Category	Test Case	Test Condition	Description	Child Requirements	Parent Requirements
Pet Operations	Add a new pet	Valid pet information	Verify successful addition of a pet with valid information	Verify Pet Addition	Pet Operations
		Invalid pet information	Verify error handling for invalid pet data (e.g., missing required fields)	Verify Pet Addition	
	Update an existing pet	Valid pet information and ID	Verify successful update of pet information	Verify Pet Update	
		Invalid pet ID	Verify error handling for updating non-existent pet	Verify Pet Update	
	Find pet by ID	Valid pet ID	Verify successful retrieval of pet by valid ID	Verify Pet Retrieval	
		Invalid pet ID	Verify error handling for non-existent pet ID	Verify Pet Retrieval	
	Find pet by status	Valid status (e.g., available, pending)	Verify successful retrieval of pets by valid status	Verify Pet Status Retrieval	
		Invalid status	Verify error handling for invalid status input	Verify Pet Status Retrieval	
User Operations	Create a new user	Valid user information	Verify successful creation of a user with valid information	Verify User Creation	User Operations
		Invalid user information	Verify error handling for invalid user data (e.g., missing required fields)	Verify User Creation	
	Update an existing user	Valid user information and ID	Verify successful update of user information	Verify User Update	
		Invalid user ID	Verify error handling for updating non-existent user	Verify User Update	
	Find user by username	Valid username	Verify successful retrieval of user by valid username	Verify User Retrieval	
		Invalid username	Verify error handling for non-existent username	Verify User Retrieval	

Place an order	Click on a search result to view item details	Valid order information	Verify successful placement of an order with valid information	Verify Order Placement	Place an order
		Invalid order information	Verify error handling for invalid order data (e.g., missing required fields)	Verify Order Placement	
	Get order by ID	Valid order ID	Verify successful retrieval of order by valid ID	Verify Order Retrieval	
		Invalid order ID	Verify error handling for non-existent order ID	Verify Order Retrieval	
	Delete an order	Valid order ID	Verify successful deletion of an order by valid ID	Verify Order Deletion	
		Invalid order ID	Verify error handling for deleting non-existent order	Verify Order Deletion	