SOFTWARE TEST PLAN
(STP)

Asana website

Nidaa AbuSaleh

TABLE OF CONTENTS

INTRODUCTION

1. Purpose:

The purpose of this Software Test Plan (STP) is to outline the strategy, scope, resources, and schedule for conducting quality assurance (QA) tests through automation on the Asana website. The primary goal is to ensure the website's user interface (UI) and application programming interface (API) functionalities meet specified standards, and any identified issues are effectively addressed.

2.Project Overview:

Implementing QA automation tests for the Asana website, focusing on both UI and API aspects to ensure functionality, usability, and reliability.
Objectives:

- Develop automated UI tests to validate functionality and usability.
- Create automated API tests to verify correctness and reliability.
- Ensure coverage of critical workflows, edge cases, and performance benchmarks.
- Implement a robust automation framework for easy maintenance and scalability.
- Provide comprehensive test coverage to identify and address issues in UI and API.

3.Website Overview:

The Asana website serves as a centralized platform for teams to manage tasks, projects, and collaborations efficiently. It offers features like task assignment, progress tracking, and team communication, accessible across desktop and mobile browsers.
Both UI and API automation testing are essential to maintain the quality, reliability, and performance of the Asana website, ensuring a smooth user experience for teams worldwide.

Test Strategies

1.Test Objectives:

- Functional Testing: Validate critical functionalities of the Asana website, such as task creation, assignment, and collaboration features, to ensure they meet specified requirements.
- Compatibility Testing: Verify cross-browser compatibility (Chrome, Firefox, Safari) to ensure a consistent user experience across different platforms.
- Usability Testing: Validate the website's usability and accessibility to ensure it meets the needs of diverse user groups, including those with disabilities.

UI Automation Testing:

- Validate the functionality of UI elements such as task creation forms, navigation menus, and user interactions.
- Ensure the consistency of design and layout across different pages and devices.
- Verify responsiveness and compatibility across desktop and mobile browsers.

API Automation Testing:

- Validate the functionality and reliability of Asana API endpoints for tasks, projects, and user management.
- Verify the correctness of data returned by API requests and proper handling of error scenarios.
- Assess the performance of API requests under normal and peak load conditions to ensure timely responses.

2. Feature to be tested:

Feature to be Tested	Applicable Roles	Description
1. Login	User	Verify successful and failed login attempts with various credentials and connected accounts.
2. Task Management	User	Create, update, and manage tasks within projects, including assignment, status tracking, and comments.
3. Project Creation	User	Create new projects, set project goals, and assign tasks to team members.
4. Search and Filter	User	Utilize advanced search and filter options to quickly find tasks, projects, and conversations.
5. Integration with Third-Party Tools	User	Test integration with external tools and platforms such as Google Drive and Slack for seamless collaboration.
6. User Management	User	Create, delete, and manage user accounts with appropriate permissions and access levels.
7. Reporting and Analytics	User	Generate reports and analyse project progress, task completion rates, and team performance.
8. Automation and Workflow	User	Implement and test workflow automation features such as task dependencies, recurring tasks, and custom rules.

9. Endpoint Functionality	Developer	Verify the functionality of Asana API endpoints, ensuring they return accurate and relevant data based on specified parameters.
10. Data Accuracy	Developer	Validate the accuracy and consistency of data retrieved from different API endpoints to ensure reliability.
11. Documentation Clarity	Developer	Assess the clarity and completeness of Asana API documentation to facilitate ease of use and implementation for developers.
12. Response Time	Developer	Evaluate the response time of API requests under normal and peak load conditions to ensure optimal performance.
13. Compatibility	Developer	Validate the compatibility of the Asana API across various programming languages and environments.
14. Authentication and Authorization	Developer	Ensure that authentication and authorization mechanisms are effective in controlling access to restricted data and functionalities.

3.Test Assumptions:

- The Asana website is stable and accessible for testing.
- Test environments accurately mimic production settings.
- Adequate test data is available for simulating various scenarios.
- Asana API endpoints are stable, accessible, and well-documented.
- API test environments mirror production setups.
- Relevant test data is available for validating API functionality.

4.Test Approach:

Utilize automated testing tools like Selenium WebDriver for UI testing and Postman for API testing to increase efficiency and coverage. Conduct comprehensive testing, including functional, compatibility, performance, security, and usability aspects, to ensure thorough evaluation. Prioritize testing efforts using a risk-based approach to focus on high-impact scenarios.

5.Data Approach:

- Utilize synthetic data generation techniques to simulate diverse scenarios and effectively test edge cases for both UI and API testing.
- Employ test data management tools and processes to efficiently generate, manipulate, and manage test data, supporting testing requirements.
- Ensure compliance with data protection regulations and maintain data privacy and confidentiality throughout the testing process for both UI and API automation testing

6.Levels of Testing:

- Unit Testing:Validate individual UI components and API endpoints to ensure functionality and accuracy.
- Integration Testing:Verify seamless communication and data flow between UI components and API endpoints.
- System Testing:Validate overall system behavior for UI and API, ensuring consistency and reliability.
- Acceptance Testing:Test with end users for UI; involve stakeholders to ensure API meets expectations.

Execution Strategy

1.Test Execution:

- Automation Framework Selection:
  For UI: Choose a robust and scalable automation framework, like Selenium WebDriver with Python, to automate UI test cases efficiently.
  For API: Opt for a suitable automation framework for API testing, such as Postman or other API testing tools, to automate and validate API endpoints effectively.
- Continuous Integration (CI):
  Integrate automated UI tests into the CI/CD pipeline to ensure they run automatically with each code commit, enabling early detection of UI defects.
  Similarly, integrate automated API tests into the CI/CD pipeline for automatic execution with each code commit, allowing early detection of potential API issues and maintaining a reliable testing process.

2.Test Environment:

Multiple Test Environments:

- For UI: Utilize multiple test environments such as development, staging, and production-like environments to simulate real-world conditions and conduct UI tests under various scenarios.
- For API: Employ multiple test environments to simulate different scenarios and test the API under various conditions, including development, staging, and production-like environments.

Environment Configuration Management:

- Ensure that test environments accurately replicate the production environment in terms of configuration and usage for both UI and API testing.
- This includes identical setups for databases, servers, and other dependencies to minimize environment-related discrepancies and guarantee accurate test results.

3.Test Data Management:

Data Generation:

- Utilize automated tools to generate diverse and realistic test data for both UI and API testing, covering various scenarios to ensure thorough testing of functionalities and responses.

Data Reusability:

- Promote the reusability of test data across different test scenarios for both UI and API testing, reducing duplication efforts and enhancing testing efficiency.
- Maintain a centralized repository for test data to ensure consistency and facilitate easy access and reuse across different tests.

4.Test Execution Process:

- Test Prioritization:Prioritize test cases based on risk and criticality for both UI and API testing, focusing on high-impact areas to ensure comprehensive coverage.
- Test Execution Flow:Define standardized test execution flows for UI and API testing, specifying preconditions, steps, and expected outcomes to ensure consistency and facilitate reusability.
- Defect Reporting:Report defects using standardized formats like JIRA for both UI and API testing, ensuring proper tracking, prioritization, and resolution to minimize project impact.

5.Test Coverage:

- Analyze test coverage reports to identify areas of the UI and API that lack adequate test coverage.
- Ensure all critical functionalities, UI components, and API endpoints are thoroughly tested and validated.
- Identify potential edge cases in both UI and API testing to enhance overall reliability and quality assurance of the Asana website.

TEST CRITERIA

Suspension Criteria:

- Test Case Failure: If 20% or more of the UI or API test cases fail, testing will be suspended until all failed cases are addressed by the development team.
- User Experience Degradation: If testing indicates a significant decline in user experience or functionality, testing may be temporarily suspended until the issues are resolved.

Exit Criteria:

- Run Rate: The run rate for both UI and API tests must reach 100%, unless justified by clear documentation. This ensures comprehensive test coverage.
- Pass Rate: The pass rate for both UI and API tests should be 80% or higher. Achieving this pass rate is mandatory to proceed to the next phase, demonstrating adherence to 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