STD Document

Mohammad khayyo

Table of Contents

Purpose Of This Document

The purpose of this STD document is to outline a detailed test design strategy for testing the Monday.com website, covering UI testing, API testing, and integrated tests that combine both API and UI components. This document aims to provide a clear framework for executing tests that ensure the website's functionality, performance, reliability, and user experience meet the highest quality standards.

Scope

This document covers comprehensive test coverage for Monday.com's key functionalities, including user account management, task and project collaboration, workflow automation, external tool integration, data visualization, security features, communication functions, user feedback incorporation, customization options, and platform compatibility.

Goals

Comprehensive Test Coverage:

Goal: Develop STD files to provide comprehensive coverage of Monday.com's functionalities, ensuring that critical workflows and features are thoroughly tested.

Error Detection and Reporting:

Goal: Create automated test scenarios within the STD files to effectively identify and report errors or anomalies in the system behavior, providing detailed information for efficient debugging.

Cross-Browser Compatibility:

Goal: Implement STD files to validate and ensure that the Monday.com website functions seamlessly across various web browsers, guaranteeing a consistent user experience for all users.

Security Testing:

Goal: Integrate security-focused STD files to systematically test the platform for vulnerabilities, ensuring that Monday.com adheres to security best practices and safeguards user data.

User Workflow Validation:

Goal: Create STD files that mimic real user workflows to validate end-to-end scenarios, ensuring that users can smoothly navigate through key features and complete tasks without hindrance.

Documentation and Reporting:

Goal: Implement STD files that generate comprehensive test documentation and reports, providing clear insights into test results, including passed tests, detected issues, and overall system health.

Glossary

Monday.com:

The project management platform that serves as a digital workspace for teams to coordinate, collaborate, and manage tasks and projects efficiently.

Digital Workspace:

An online environment provided by Monday.com where teams can plan, organize, and execute tasks and projects collaboratively.

Task Coordination:

The process of organizing and managing individual tasks within a project to ensure they align with the project's objectives and timelines.

Project Management:

The systematic planning, coordination, and execution of a project, involving tasks, resources, and timelines to achieve specific goals.

Collaboration Hub:

Monday.com's centralized platform that facilitates seamless collaboration among team members, allowing them to share files, communicate, and stay updated on project progress.

STD (Software Test Description) File:

A document outlining the details and specifications of the software testing procedures, including test scenarios, test cases, and expected outcomes for Monday.com development.

Test Scenario:

A high-level description of a specific functionality or feature to be tested within Monday.com, providing context for the test cases.

Test Case:

A detailed set of conditions, inputs, and expected outcomes for testing a specific aspect or functionality of Monday.com.

Test Suite:

A collection of related test cases organized together in the STD file, typically focused on a specific module or feature of the Monday.com application.

Defect:

An identified deviation or inconsistency in the behavior of Monday.com compared to its expected results, documented in the STD file for resolution.

Test Cases

UI Test Cases

UI Test Case 1: Create a New Board

- **Objective**: Ensure users can create a new board through the UI.
- **Preconditions**: User must be logged in.
- **Steps**:
    1. Click on the "Add" button or "+" icon and select "New Board".
    2. Enter a name for the board and select a board type (Main/Private/Shareable).
    3. Click "Create Board".
- **Expected Results**: A new board with the given name is created and displayed in the user's dashboard.

UI Test Case 2: Update Board Name

- **Objective**: Verify that users can update a board's name through the UI.
- **Preconditions**: A board already exists.
- **Steps**:
    1. Open the board to be renamed.
    2. Click on the board's name at the top.
    3. Enter a new name and click "Save".
- **Expected Results**: The board's name is updated across the platform.

UI Test Case 3: Delete a Task

- **Objective**: Ensure users can delete a task from a board.
- **Preconditions**: A board with at least one task exists.
- **Steps**:
    1. Navigate to the board containing the task.
    2. Hover over the task to reveal action buttons.
    3. Click the "Delete" button and confirm the deletion.
- **Expected Results**: The task is removed from the board and no longer visible.

API Test Cases (GraphQL)

API Test Case 1: Query Board Details

- **Objective**: Fetch details of a specific board by ID using a GraphQL query.
- **Preconditions**: The board ID is known, and the user has appropriate permissions.
- **GraphQL Query**:

  graphqlCopy code
  ```
  query { boards(ids: 123456789) { id name description } }
  ```
- **Expected Response**: The API returns the details of the specified board, including `id`, `name`, and `description`.

API Test Case 2: Create a New Item on a Board

- **Objective**: Use a GraphQL mutation to create a new item on a specific board.
- **Preconditions**: User has the board ID where the item will be added.
- **GraphQL Mutation**:

  graphqlCopy code
  ```
  mutation { create_item(board_id: 123456789, item_name: "New Task", column_values: "{}") { id } }
  ```
- **Expected Response**: A successful response indicating the item has been created, including the new item's `id`.

API Test Case 3: Update an Item's Status

- **Objective**: Update the status of an existing item using a GraphQL mutation.
- **Preconditions**: The item ID and the board ID are known.
- **GraphQL Mutation**:

  graphqlCopy code
  ```
  mutation { change_column_value(board_id: 123456789, item_id: 987654321, column_id: "status", value: "{\"label\":\"Done\"}") { id } }
  ```
- **Expected Response**: The API confirms the status column's value has been updated to "Done" for the specified item.

Integrated Test Cases

Integrated Test Case 1: Board Creation and API Verification

- **Objective**: Verify that a board created via the UI is immediately fetchable and correctly represented via the API.
- **Steps**:
  1. User creates a new board named "Project Alpha" via the UI.
  2. Using the GraphQL API, fetch the list of boards to verify the creation.
- **Expected Results**: The API response includes "Project Alpha" among the user's boards, confirming UI-API data consistency.

Integrated Test Case 2: Task Assignment Notification

- **Objective**: Ensure that assigning a task to a user via the UI triggers a notification that can be verified through the API.
- **Steps**:
  1. Assign a task to a user via the UI.

2. Query the user's notifications using the GraphQL API.

- **Expected Results**: The API returns a list of notifications for the user, including the recent task assignment.

Integrated Test Case 3: Item Deletion and API Confirmation

- **Objective**: Confirm that deleting an item via the UI immediately reflects in API queries.
- **Steps**:
    1. Delete an existing item from a board using the UI.
    2. Use a GraphQL query to attempt to fetch the deleted item by ID.
- **Expected Results**: The GraphQL query returns an error or null, indicating the item does not exist, confirming the UI action's immediate effect on the API level.