Table of Contents

[1 Scope 1](#_Toc68991749)

[2 Out of Scope 1](#_Toc68991750)

[3 Test Approach 1](#_Toc68991751)

[3.1 Manual Testing 1](#_Toc68991752)

[3.2 Test Automation 1](#_Toc68991753)

[3.2.1 Frontend Tests 1](#_Toc68991754)

[3.2.2 Backend Tests 2](#_Toc68991755)

[4 Test Environment 2](#_Toc68991756)

[5 Milestones / Deliverables 2](#_Toc68991757)

[5.1 Deliverables 2](#_Toc68991758)

# Scope

This document describes a test approach for a demo test case scenarios.

Test scenarios could be executed manual and automated.

# Out of Scope

The performance tests and exploratory tests are out of scope

# Test Approach

## Manual Testing

All test cases could be executed manually. For each execution, test status and notes/comments should be involved to test report.

At the end of each execution, the errors/fails should be reported by a bug report application like Jira. Also a execution report could be created manually or from a template and should be shared with the teams. Report should contains

* Test execution date
* Test case scope
* Errors/fails with descriptions
* Issue numbers if created on Jira
* Not tested scenarios and why (data, network etc.)
* Count of success / errors/ fails / not executed tests with a readable table

## Test Automation

Test automation can be execute for both front-end and back-end tests.

### Frontend Tests

UI behaviors should be tested by the UI Test Automation Framework. So the team can be sure which components / displays are working. But all test case should not be automated over this framework. The automation scope should be established according to risk matrices and test case priority.

For example, suppose the UI test scenarios include the following steps:

* Add description to “MyCard” card on Doing list
* Move “MyCard” to Done

The test should be like this:

* Login (**UI Automation**)
* Create a card named “MyCard” (Over Api -> if card does not exist, than create a card using Trello API over **Framework.ApiHandler** in Specflow’s “Before Scenario” hook)
* Add description (**UI Automation**)
* Move Card (**UI Automation**)

Integrate those type of tests to pipeline is a bit difficult. But after each release or bug fix some of tests could be executed as regression tests according to impact analysis.

### Backend Tests

If user interface components are not main/first priority of test scenario, than backend tests could be execute. If team wants to ensure that the services are working or not, execution of API tests are more comfortable.

Because UI tests are breakable tests and they could affected by many external environments such as network, database connections, application performance etc.

The API tests will cover full scope of the expected results and take less time. These tests could be a part of CI in your organization easily.

Back-end and front-end tests automatically generate the execution report. If an error is detected, the Jira issue must also be created manually.

Automatically generated reports and issue numbers will be sufficient to report them. If the test team has a dashboard showing all the automatic test results, only the connection of this panel will also be sufficient.

# Test Environment

Automated tests need to DEV and PRE\_PRODUCTION working environments as well. Automation tests also need database connections and test executer servers (Windows)

# Deliverables

## Deliverables

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **For** | **Date / Milestone** |
| Test Plan | Project Manager; QA Director; Test Team |  |
| Traceability Matrix | Project Manager; QA Director |  |
| Test Results | Project Manager |  |
| Test Status report | QA Manager, QA Director |  |
| Metrics | All team members |  |