**Documentation**

# A. System details – multi-step form:

* Web Interface: Multi-step form for submitting procurement documents for compliance verification.
* Simulation of the Multi-Step Form:
  + Accepts only two types of documents: DOCX and PDF.
  + Accepts documents with a size between 0 and 10 KB (0 KB not included).
  + Contains required fields.
  + Includes a review page for filled data.
* Steps of the Simulated Multi-Step Form:
  + User information and document upload.
  + Document metadata: title, description, category.
  + Review page.
  + Confirmation page.

# B. Scope and test cases

* Performed Functional Regression Tests:
  + Test each field on a page individually.
  + Test the end-to-end flow through a multi-step form.
* Test Cases:
  + Uploading Documents with Different Sizes to Verify Upload Field:
    - Size 8 MB: Path/Valid Test
    - Size 20 MB: Path/Invalid Test
    - Size 10 MB: Boundary Test
    - Size 0 MB: Boundary Test
  + Uploading Documents of Different Types:
    - Extensions .docx/.pdf: Valid Test
    - Extensions .jpg/.zip: Invalid Test
  + Testing Required Fields for Each Form Step:
    - Required fields for personal information and document upload form step.
    - Required fields in document metadata form step.
  + Testing Field Content on the Review Page
  + Cross-Browser Testing: Investigate the Solution
  + Testing Connection Field During Form Submission \* Not yet implemented :
    - Simulate a network failure during form submission and ensure the system handles it gracefully.
    - Verify that a second attempt contains the previously entered data and that no data is missing.
  + Dynamic Element Handling

# C. Documentation

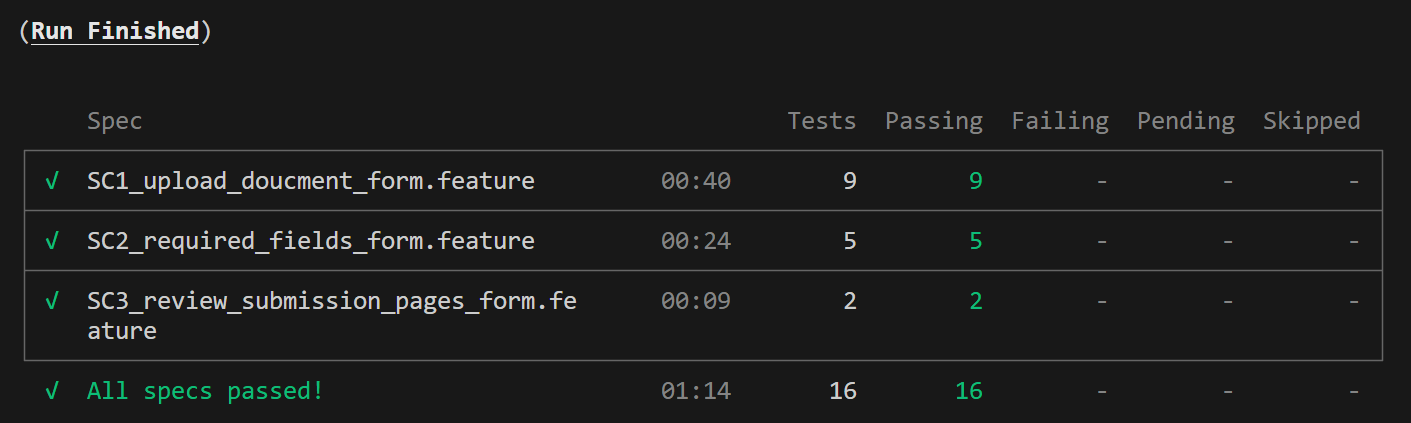
* BEST PRACTICES:
  + BDD (Gherkin) - used to integrate business members in test scenario design (3 amigos meeting).
  + Using POM (Page Object Model) - used for readable step implementation and scalable testing projects.
  + DDT (Data-Driven Tests) - to have a clear and readable single source for different metadata used throughout web application interfaces. Usually, we use Excel for structured data.
* STEPS:
  + MULTI-STEPS FORM SIMULATION - Use the website JotForm [https://www.jotform.com/build/242396167398572] to personalize the form to match the constraints defined in the specifications.
  + FAKE FILE GENERATOR - Use the fake file generator website [https://www.fakefilegenerator.com/] to produce files with different sizes and extensions.
  + SET UP PROJECT ARCHITECTURE - Defined in the next point.
  + PLUGIN CONFIGURATION - Documented in technical\_documentation.txt.
  + IMPLEMENT POM CLASSES - Defined in the point after the next.
  + Write the corresponding Gherkin scenarios.
  + Implement the Gherkin scenarios.
  + Refactor the code.
* PROJECT ARCHITECTURE:
  + TEST DATA: Exists in the cypress/fixtures folder.
  + POM CLASSES: Exist in the cypress/pages folder.
    - Base PAGE: For operations, which a user can perform across the multi-step form.
    - FirstFormStep PAGE: Corresponds to the first step form page selectors and operations - personal information and document upload.
    - SecondFormStep PAGE: Corresponds to the second step form page selectors and operations - document metadata.
    - ThirdFormStep PAGE: Corresponds to the third step form page selectors and operations.
    - Review PAGE: Corresponds to the review page selectors and operations.
    - Confirmation PAGE: Corresponds to the confirmation page selectors and operations - content typed across the 3-step form.
  + TEST SCENARIOS: Feature files - Gherkin, located in cypress/e2e/features.
    - SC1\_upload\_document\_form: To verify the upload field functionality against different document sizes and formats.
    - SC2\_required\_fields\_form: To verify the required fields functionality for each form step.
    - SC3\_review\_page: To verify the review page functionality, submission, and confirmation page.
  + STEP IMPLEMENTATION: For all 3 feature files - located in cypress/e2e/step\_definitions/UI/step\_implementation\_form.js.
  + SCRIPTS: To run the feature files across different browsers - located in scripts/.
  + ALLURE REPORTS: Generated in allure\_results/ and allure\_reports/.
* INTEGRATE THE SOLUTION TO CI/CD PIPELINE:
  + Upload the automation script to a remote branch of the company repository.
  + Prepare the CI/CD pipeline for different events: example, for GitLab, we use .gitlab-ci.yml.
    - Configure the pipeline to so it triggered for different events - for example, after each pull request to detect any regression or scheduled to run every midnight.
    - Configure path artifacts - for example, reports including screenshots.
* FURTHER ENHANCEMENTS:
  + Use Typescript for a more stable automation script.
  + Implement the DDT (Data-Driven Test) pattern for scalable automation scripts - for example, using Excel as a source of data for test data samples and the expected behavior of the system.
  + Use JavaScript models to facilitate data changes through the test automation script.
  + Use the Singleton pattern.
  + Tests on the content of files - implementation for the form & automation script.

# D. Test Report Summary

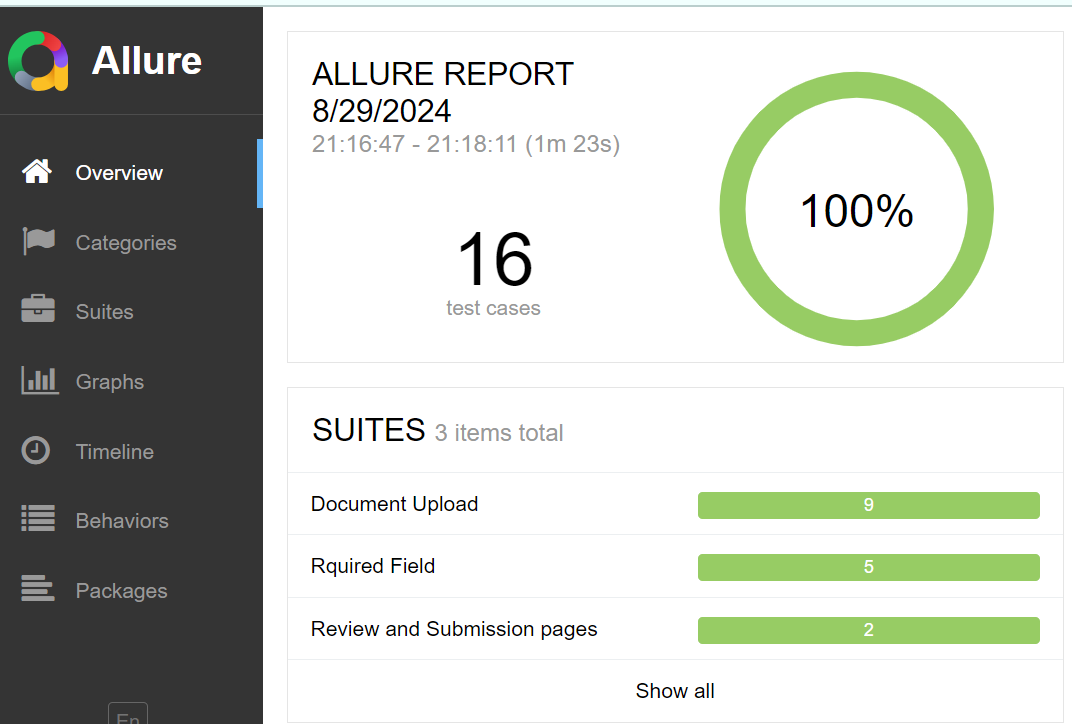
* Environment
  + Browser: Chrome
  + Evidence: Allure and Cypress screenshots
* Test Execution Overview
  + Total Test Cases: 16
  + Executed/Passed: 16
  + Failed: 0
  + Not Implemented: 1
  + Pass Rate: 100%
* Test Suite Breakdown

|  |  |  |  |
| --- | --- | --- | --- |
| Suite | Cases | Passes | Success Rate |
| File Upload Functionality | 9 | 9 | 100% |
| Required Fields Validation | 5 | 5 | 100% |
| Submission Process | 1 | 1 | 100% |
| Network Resilience | - | - | Not Implemented |

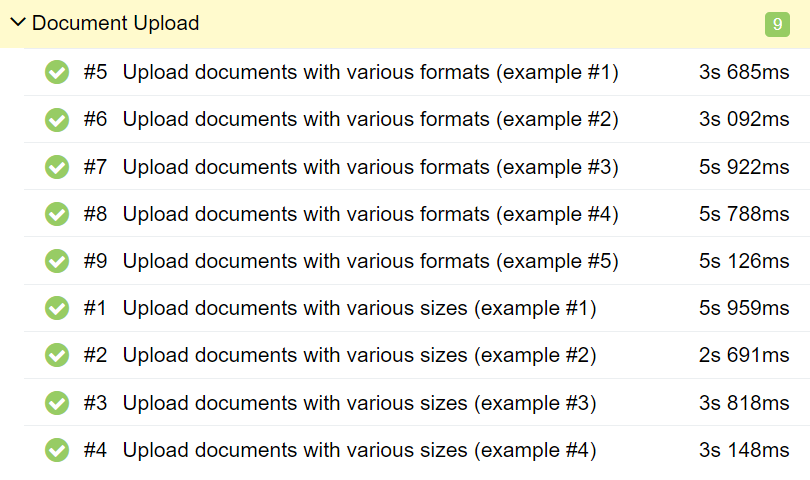
* Additional Notes
  + No flaky tests detected
  + Network interruption scenarios pending implementation
* Cypress terminal summary using headless Electron browser :



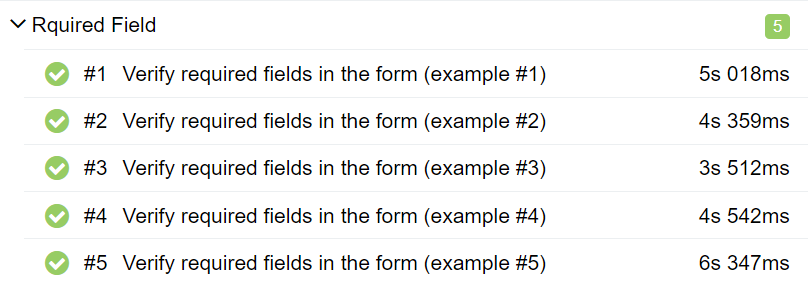
* Allure summary report:
  + Overview :



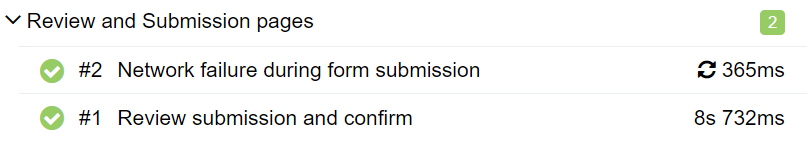
* + Upload document :



* + Required fields



* + Review and submission page:



# E. Challenges

Develop and deliver a robust, stable automation script that achieves maximum test coverage for the multi-step form functionalities outlined in the provided attachment. Implementing testing good practices and clean code.