Integration with Test Modeller

- Overview
- Main features
  - Test Modeller quick overview and core concepts
  - Mapping of concepts
- Setup
- Login example
- Use cases
  - Test Modeller to create manual Test cases in Xray
  - Test Modeller as a modeling and automation tool and Xray as single-source of truth
- Tips
- Learn more

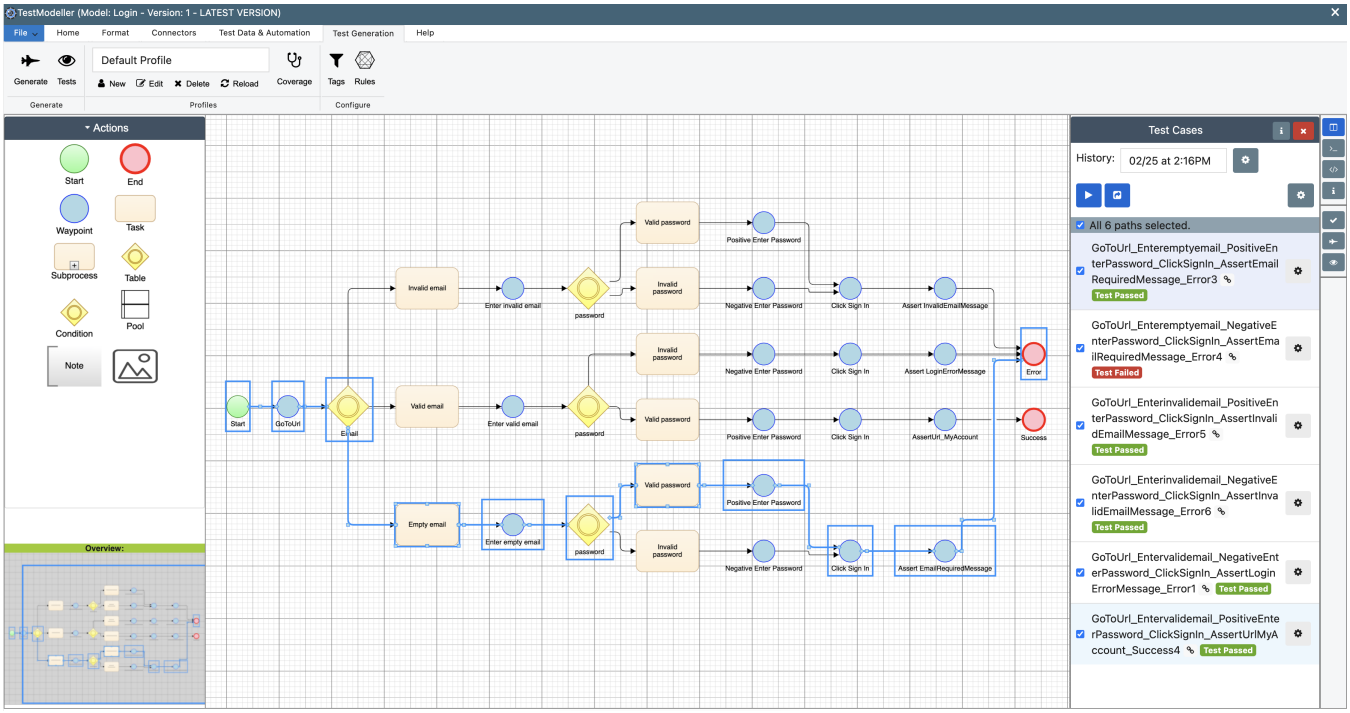
Overview

Test Modeller (TM) is a modelling tool that can be used to model parts of the system behaviour and easily implement test automation for it.

By using visual models, team members can visualize and collaborate more effectively to implement testing with greater coverage and in less time.

Models can be built by hand or using a web scanner (i.e. a Chrome extension), in case you're targetting web-based systems. The later will create objects that will contain the necessary identifiers (e.g. XPath, CSS), so they can be later on run by an automation framework.

More details about Test Modeller here.



Main features

This integration provides:

- integration with Xray cloud
- automatic provisioning of Tests in Xray, corresponding to each generated path
- ability to link tests/paths to an existing "requirement" in Jira/Xray
- ability to track automation results in Xray from testing performed using Test Modeller and underlying automation frameworks

Test Modeller quick overview and core concepts

In Test Modeller, all happens in the context of a working project.

A project is tied to an automation framework, defined upon the moment of creation, which can be one from a set of supported ones including custom frameworks.

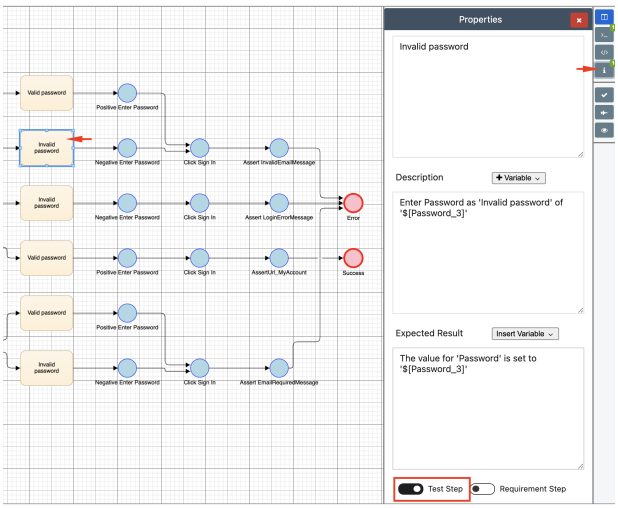
Inside a project, we can have:

- one or more **models**, each one abstracting a certain behaviour of the SUT
- **page objects/collections**, each one abstracting a certain web page; each page object contains one or more inner objects corresponding to the web/UI elements we can interact with
- **module objects/collections**, which provide an implementation layer for the page object (e.g. in Java, or other)

A model has a starting node and one or more ending nodes. The model is used as a source to generate test automation code, containing automated tests based on **test cases** (i.e. user journeys) generated by Test Modeller.

TM provides a configurable algorithm to fine-tune coverage (i.e. the test cases that will be created based on all possible paths derived from the model).

Mapping of concepts

Test Modeller	Xray	Examples
Test/Test case /Test path	Test (manual/structured) • Test' Summary field is based on Test Path's name on TM. Please see the Setup section ahead, namelly the configuration of "Field Mappings" • It may be linked to one requirement, during the export operation Note: Tests are auto-provisioned whenever doing an explicit exporting of the test suite using the Xray connector profile	
Action (e.g. Task, Waypoint)	Test step Note: only Actions having the property Test Step enabled will be mapped to a test step in Xray.	
results, of a given Test Case	Test Run	

Setup

In order to setup the integration, we need to configure and add a Xray connector profile to your project in Test Modeller.

You'll need to provide the URL of your Jira Cloud instance, Jira's username, create a Jira API token, specify the target Jira project, and also Xray API credentials (client id+client secret).

It's also important to configure the "Field Mappings", as these are used to identify how information will be mapped into the Test issue and to the step fields.

Configure Connector

Details | Connection | Field Mappings

Connector Type: Xray (JIRA Cloud) - Tests

Profile Name: xray_cloud

Shared

Validate | Save | Cancel

Configure Connector

Details | Connection | Field Mappings

Url: https://sergiofreire.atlassian.net

JIRA Username: sergio@example.com

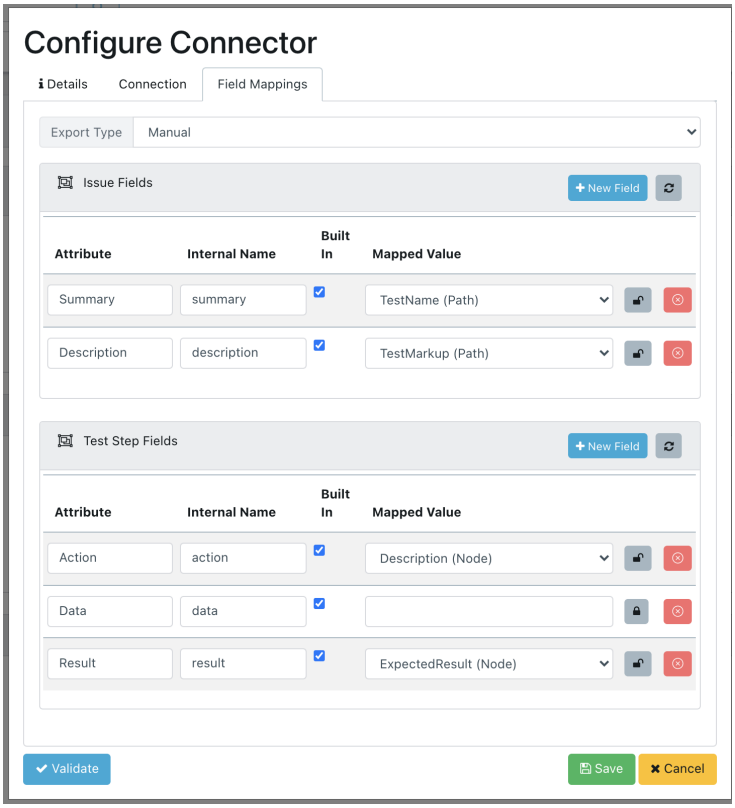
JIRA API Key: •••••

Project Key: CALC

XRay Client ID: 215FFD69FE4644728C72000000000000

XRay Secret Key: •••••

Validate | Save | Cancel

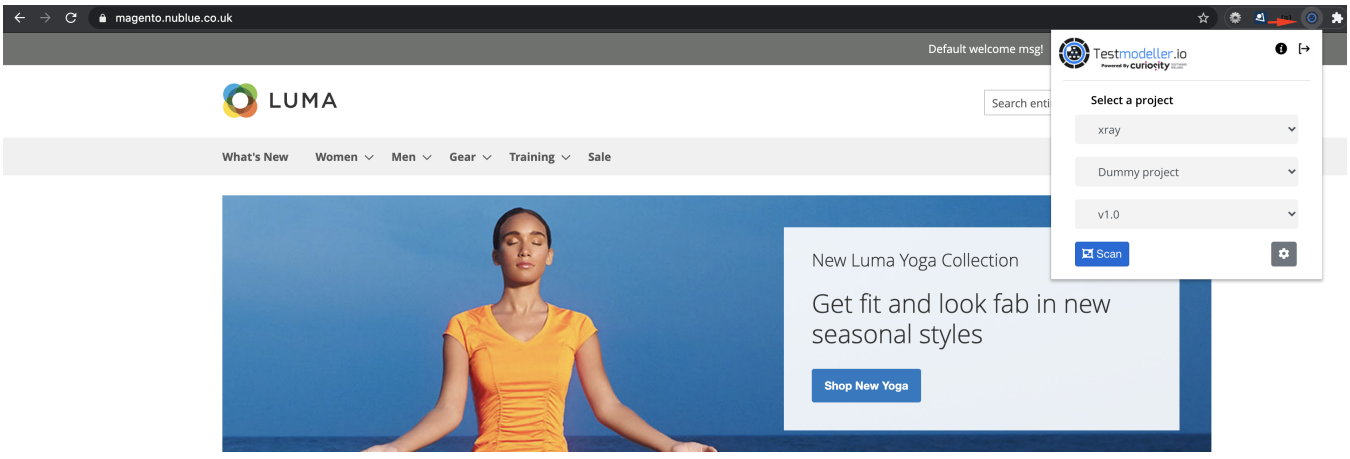


The following video shows how you can see how to setup the integration between Test Modeller and Xray.

Your browser does not support the HTML5 video element

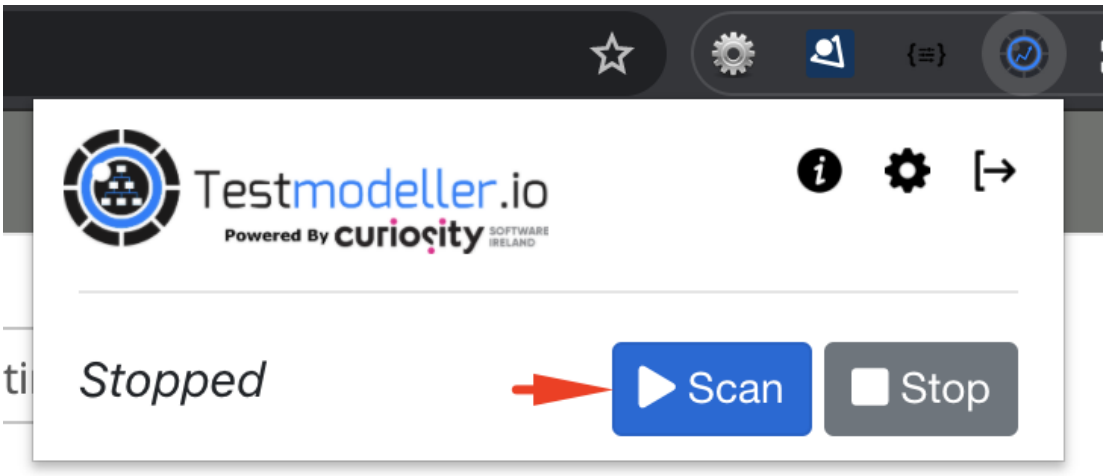
Login example

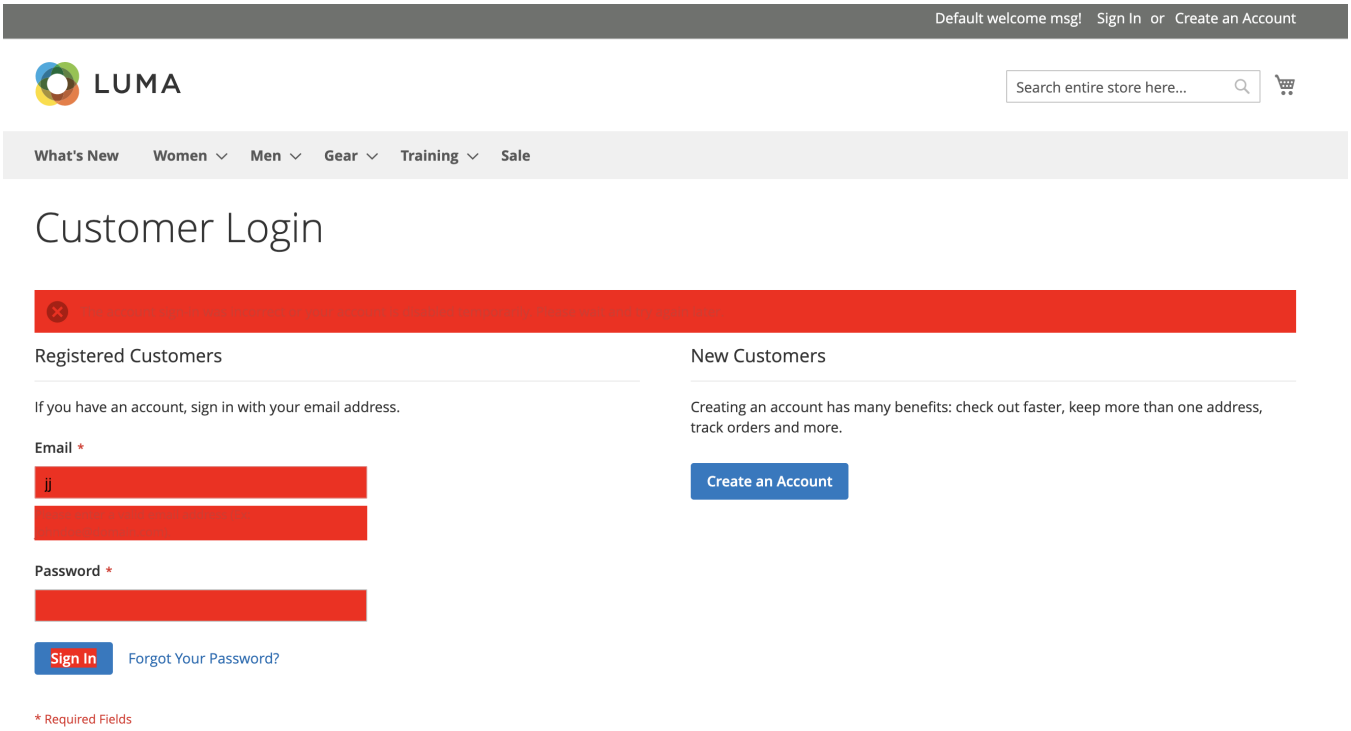
For the purpose of detailing possible use cases, we will use a simple example: test the login process of a website.



We start by using Test Modeller Explorer, a Chrome extension, to help us scan web elements and upload them to TM.

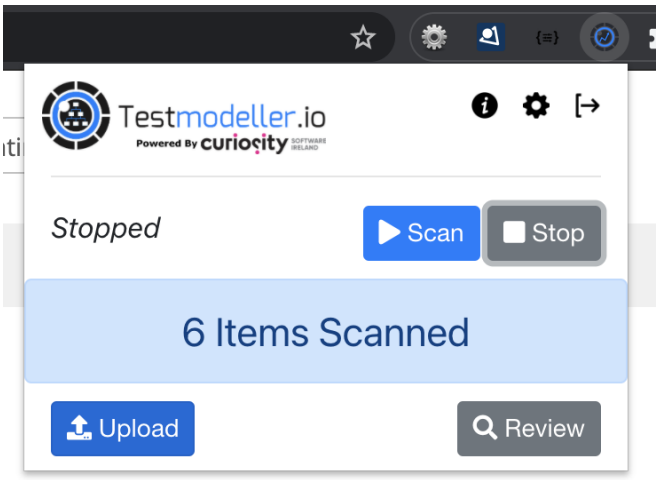
It's important to scan all elements that we need to interact with or assert (e.g. error messages.

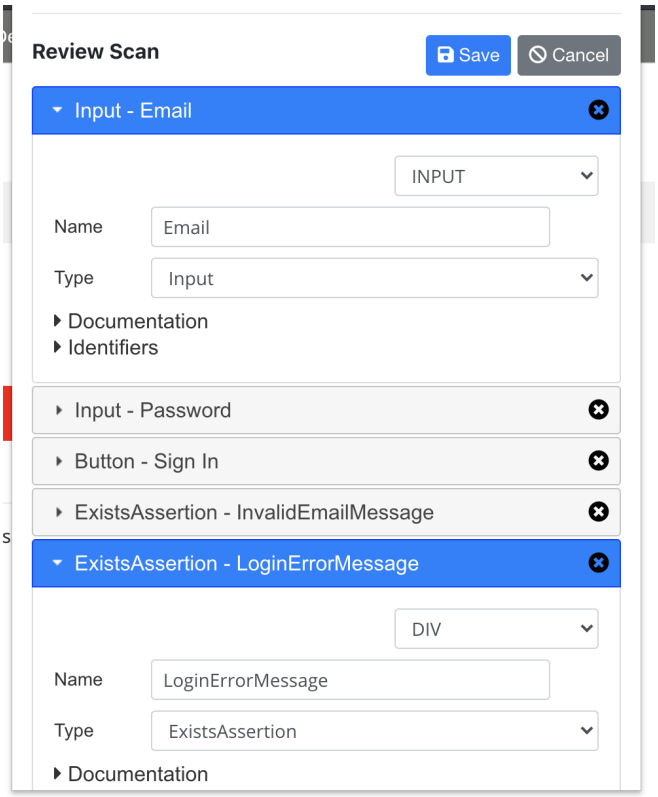




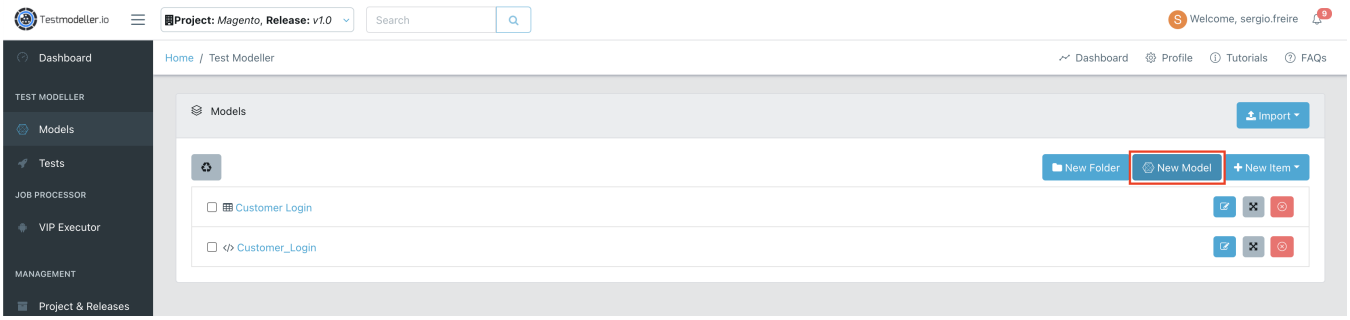
When we finish scanning, we may need to review the items to adjust the identifiers (i.e. the CSS and XPath locators) and also the type; if you're looking at some message that you want to assert, then you should use the type "ExistsAssertion".

It's also a good momentt o name the items properly, if needed.



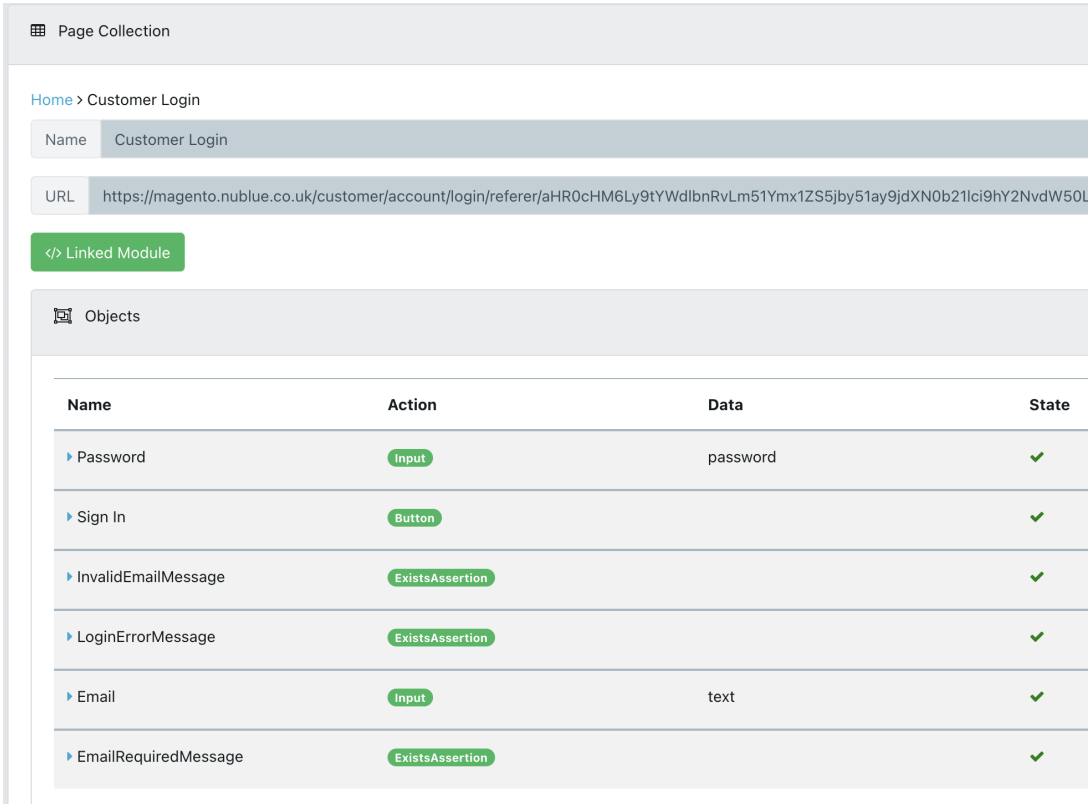


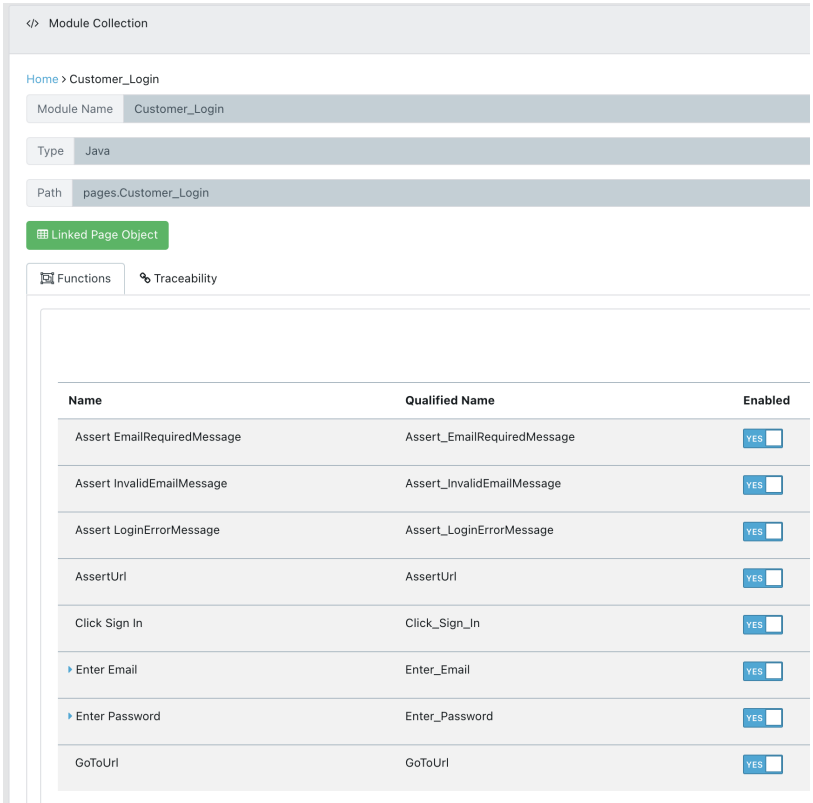
The page object can then be uploaded along with the web elements that will be provisioned as objects inside it. This can repeated for other pages and can also be done at any moment in time.



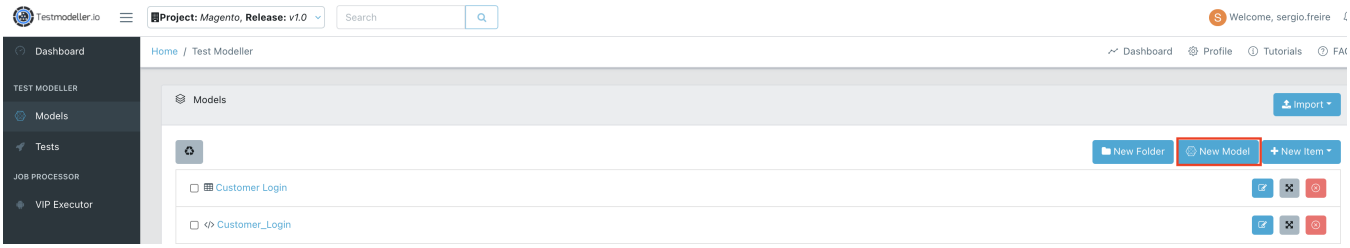
The "Page Collection" (i.e. the page object) contains inner objects that correspond to the web elements that we can interact with. It is linked to an implementation layer, also known as a Module/Module Collection.

The Module contains functions that allow us to interact with the page object, including navigating to the page, filling out some fields, clicking sign-in button, look for the presence of certain error messages.





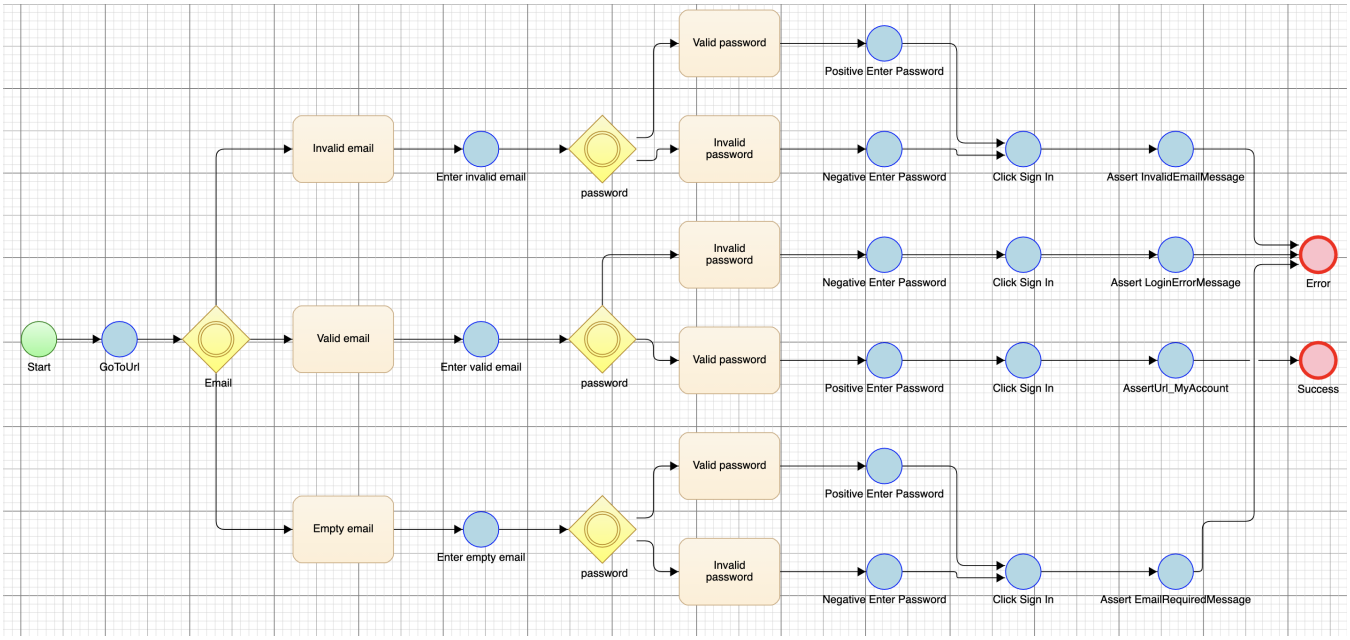
After importing or creating all the page objects, we can build a model.



The following diagram shows a possible implemention for checking the login feature.

There is a start node and two ending nodes: one for successful login and another to encapsulate all failed login attempts.

> **Please note**
>
> Models can be implemented in different ways. What you decide to represent, the risks you want to assess, that and other will shape the design process of your model.
>
> There is no right/wrong in a model but modelling is an art that requires practice.

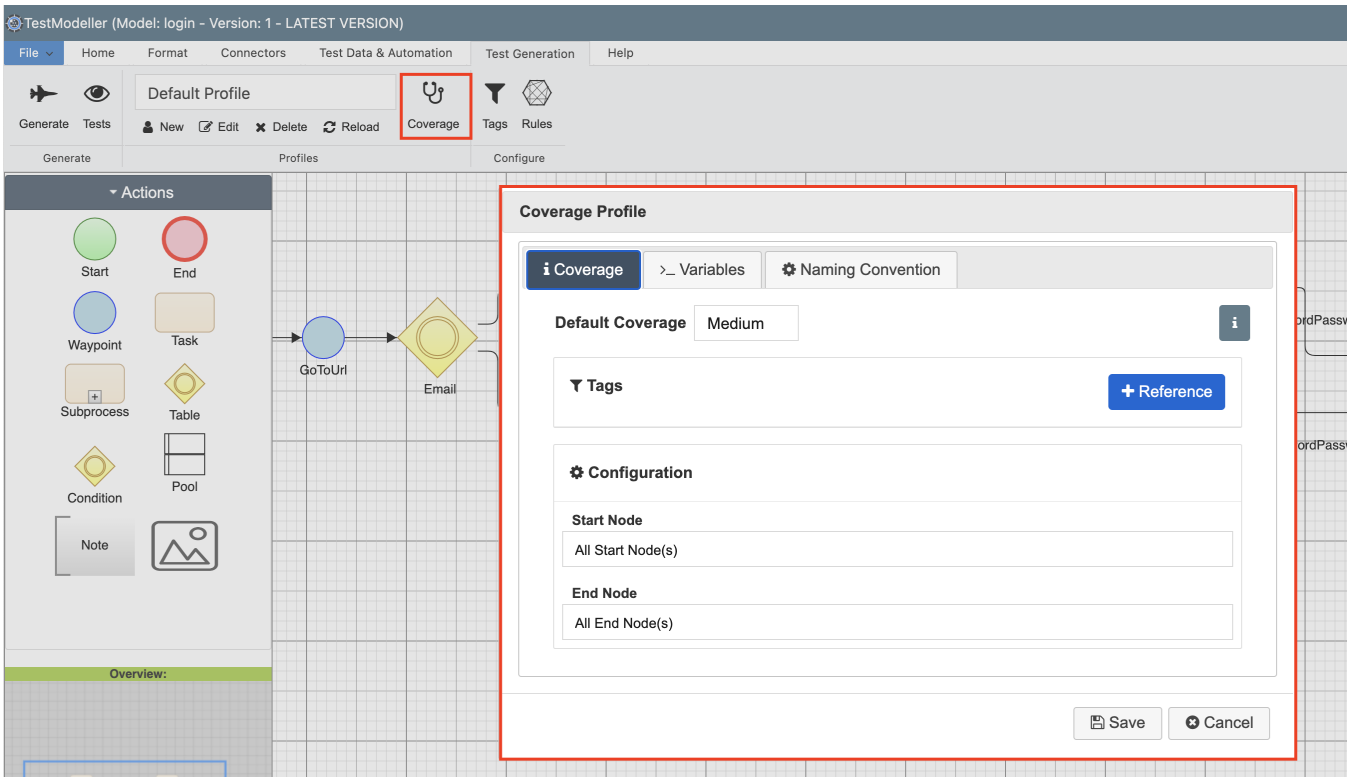


Use cases

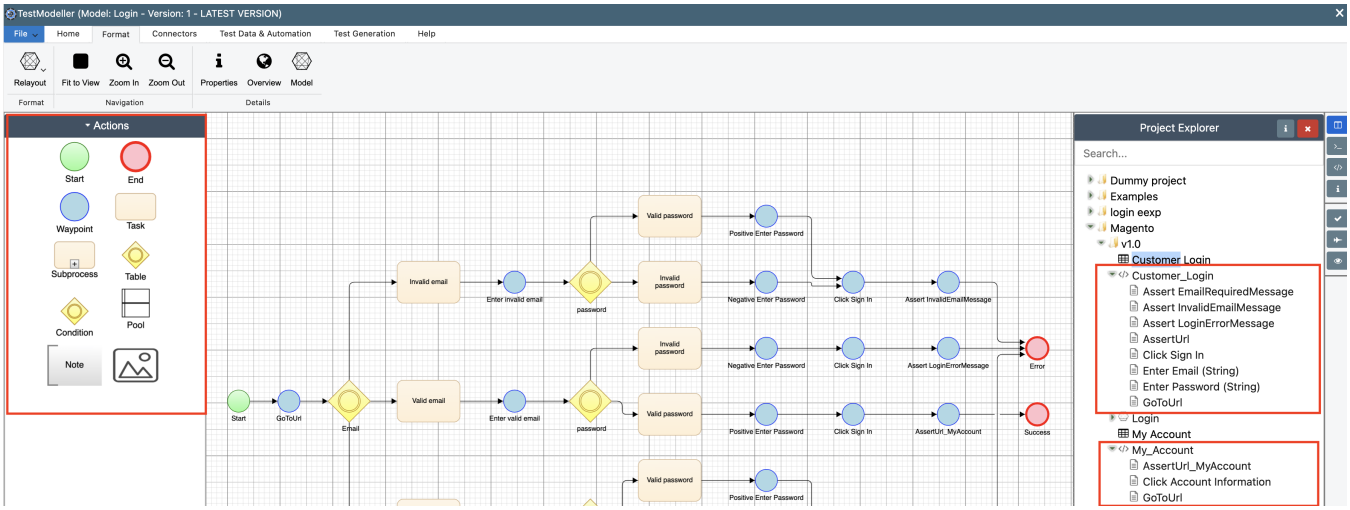
Test Modeller to create manual Test cases in Xray

In this use case, Test Modeller is used to design models and generate multiple paths/test cases (i.e. tests).

But first, the coverage profile must be defined in order to fine tune the algorithm for the generation of the test cases.



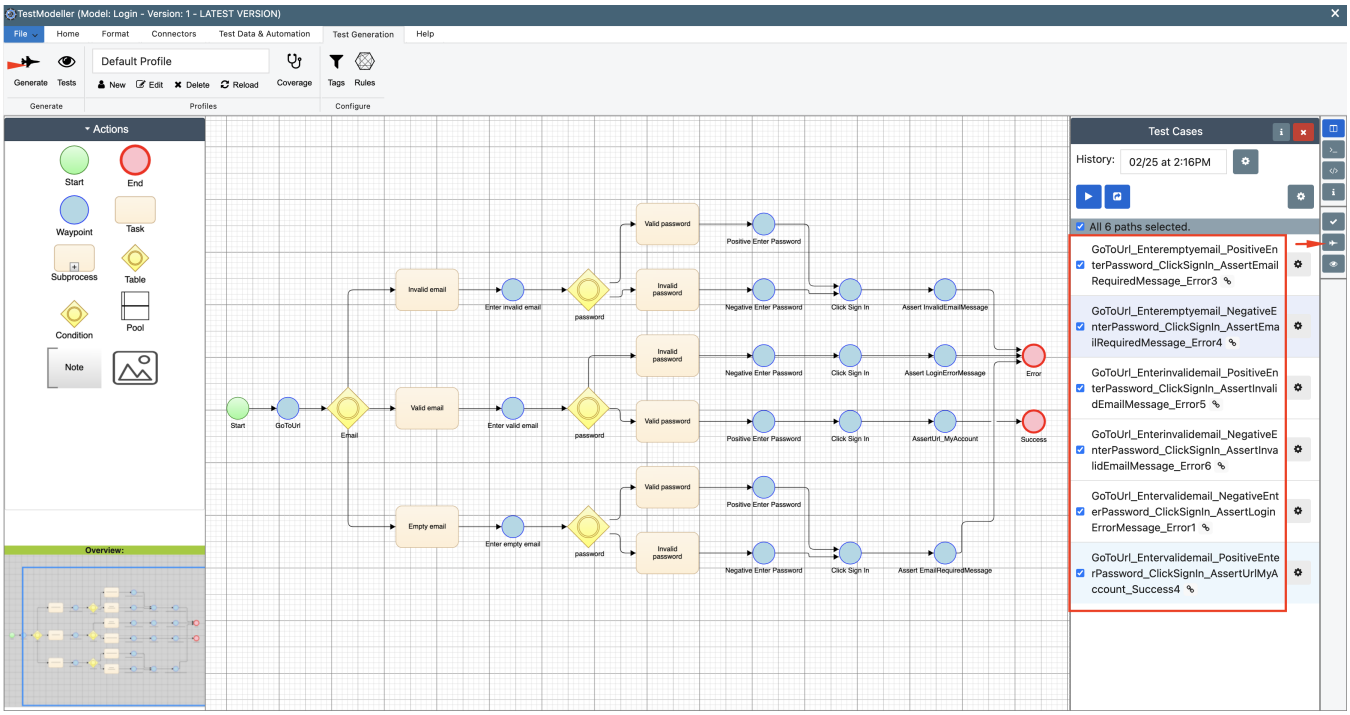
Then we can build the model using Actions or functions from the Modules we have in our project. Note that these functions are associated with operation done in the objects of the corresponding page object.



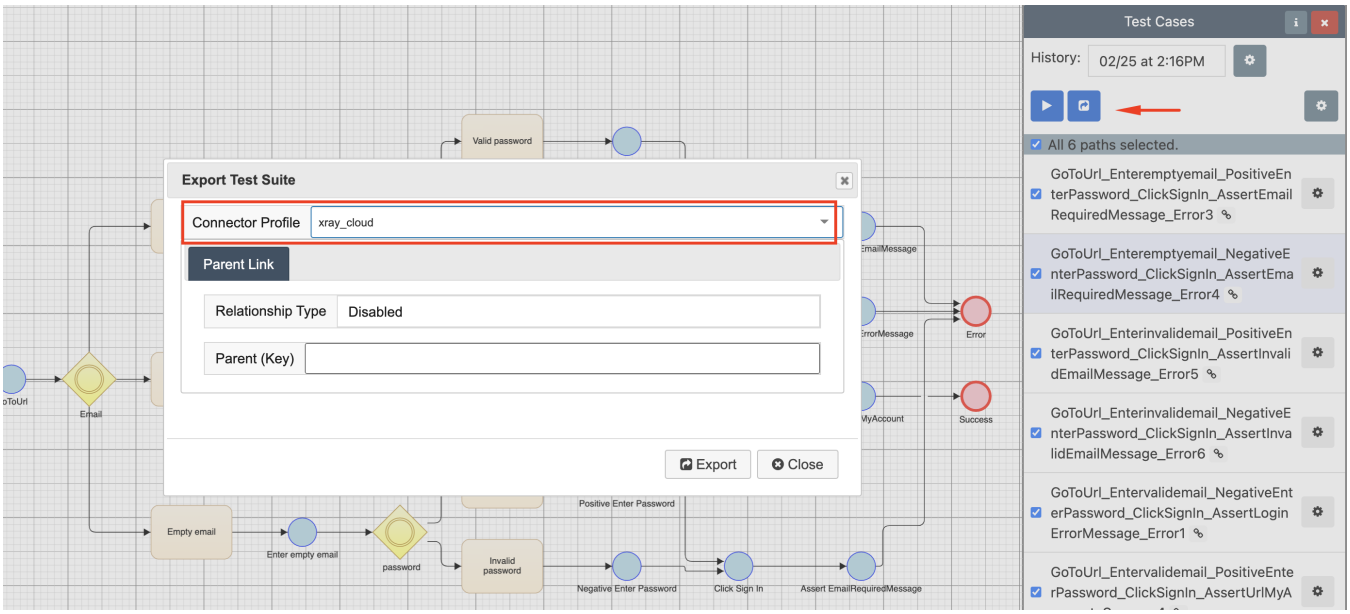
In the model different paths come to live, as a test case that represents a journey.

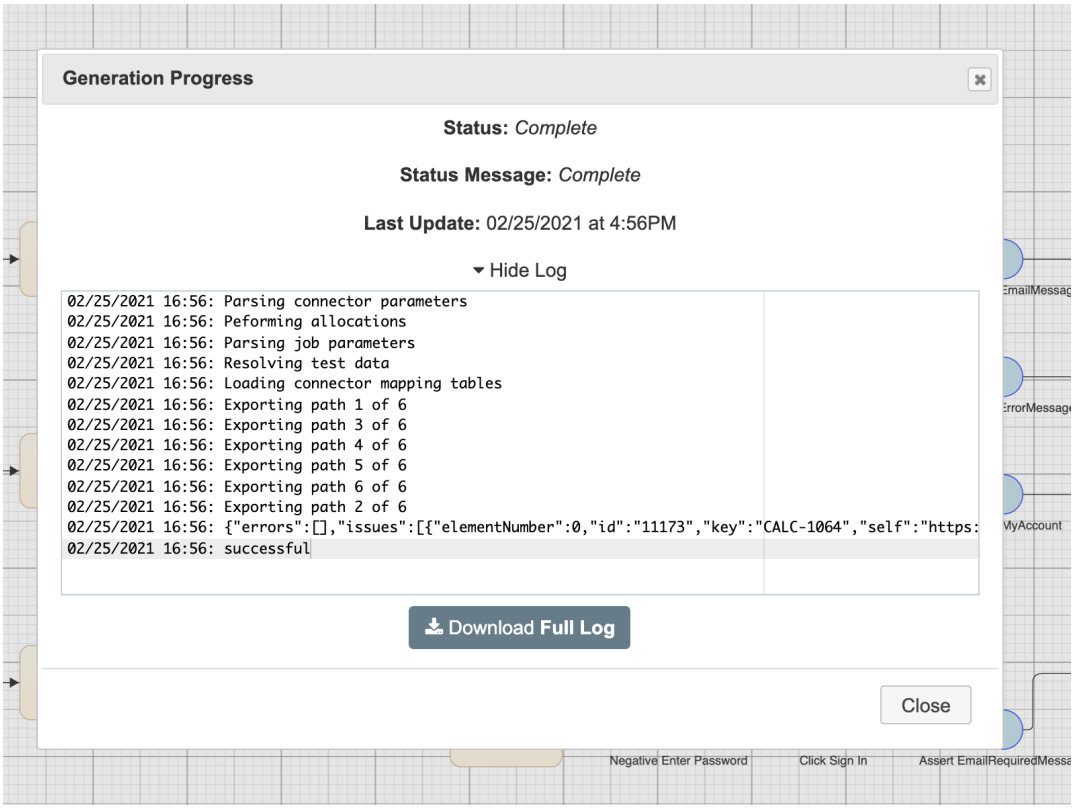
These test cases/paths will be provisioned in Xray as Test issues, by doing an export operation to Xray using the connector profile created earlier.

This can be done from within the Test Generation menu or the right-side bar. Paths will be created and will be visible on the Tests/Test Cases panel.

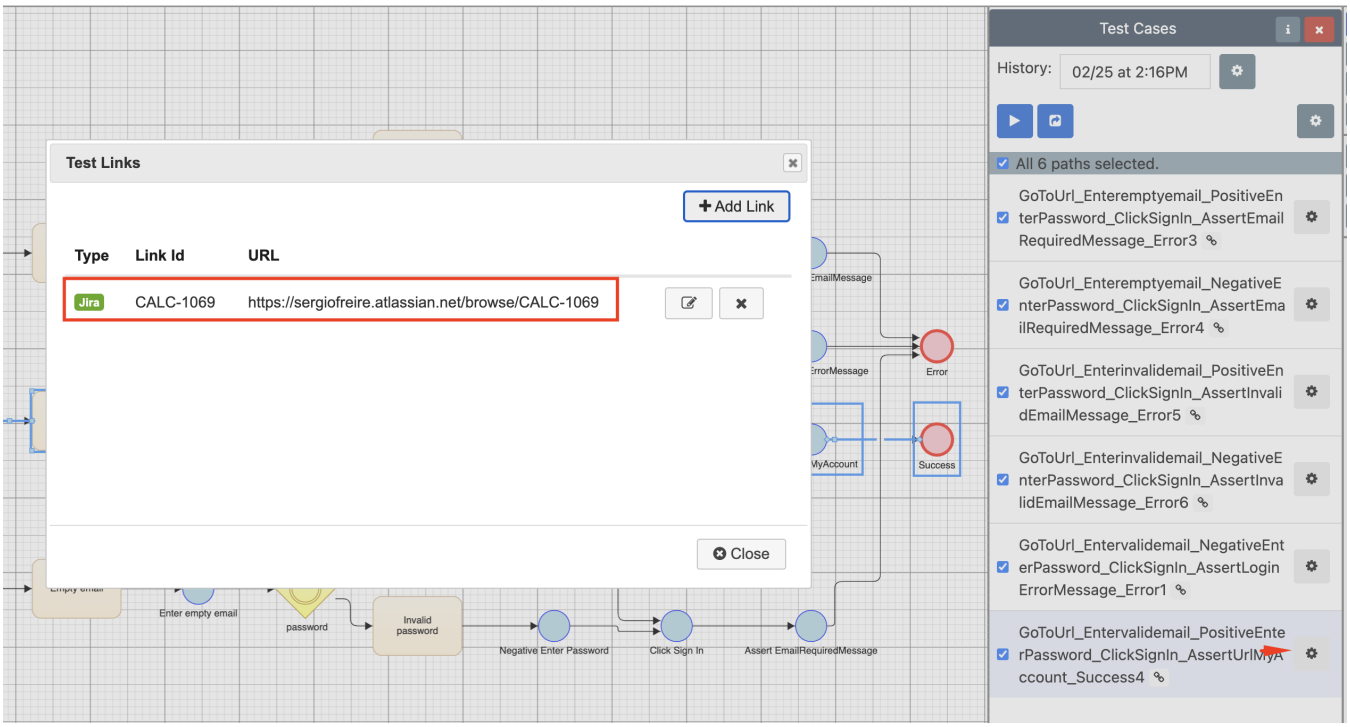


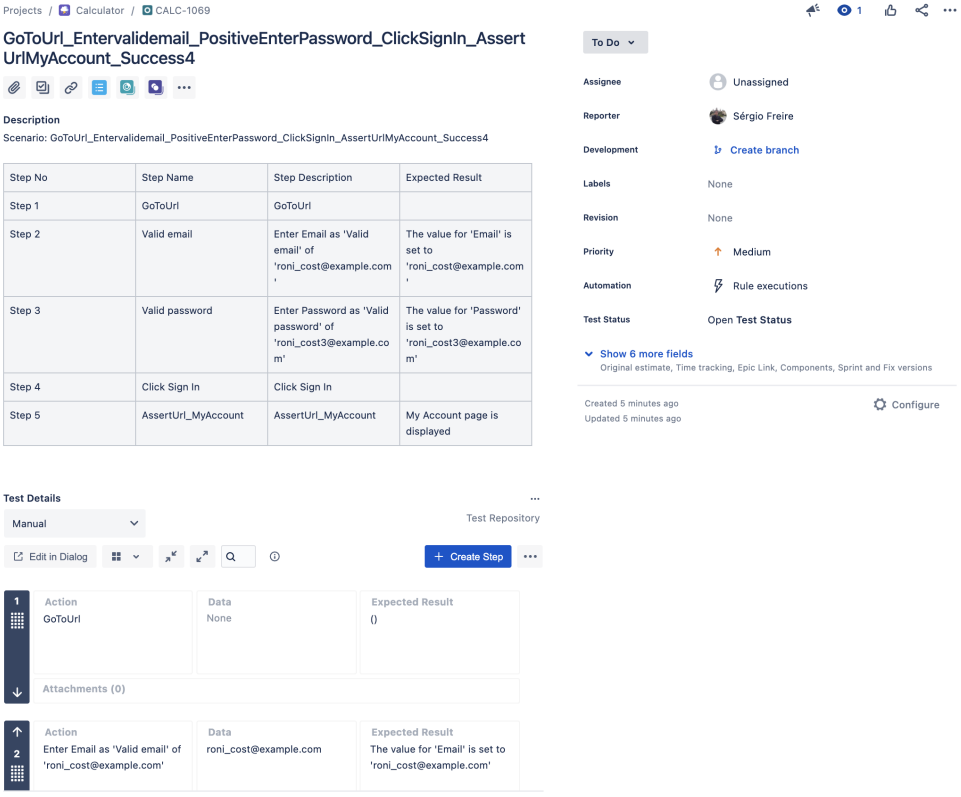
To export these Test Cases to Xray, an export operation can be triggered using the Connector Profile having the Xray settings.





In Xray, Test issues are created; each Test Modeller's test case/path has a corresponding Test issue in Xray. A link sign is shown in each path/test case in TM; the link to the Test issue in Jira can be obtained through the context options available in each path.





Test Modeller as a modeling and automation tool and Xray as single-source of truth

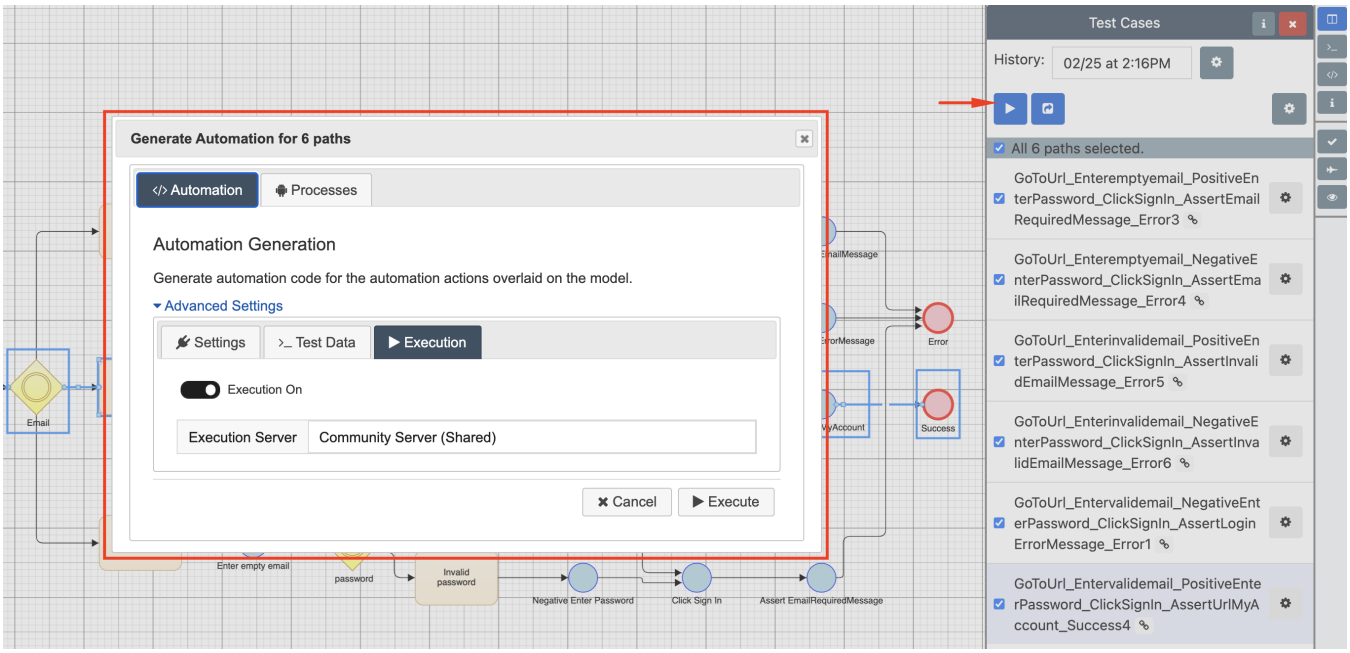
In this use case, Test Modeller is used once again to design models and generate multiple paths (i.e. tests). These paths will be provisioned in Xray as Test issues.

Test automation code will be also generated in one of different automation frameworks supported. This code will be executed by Test Modeller, or during CI, and results will be reported to Xray, to the Tests provisioned earlier.

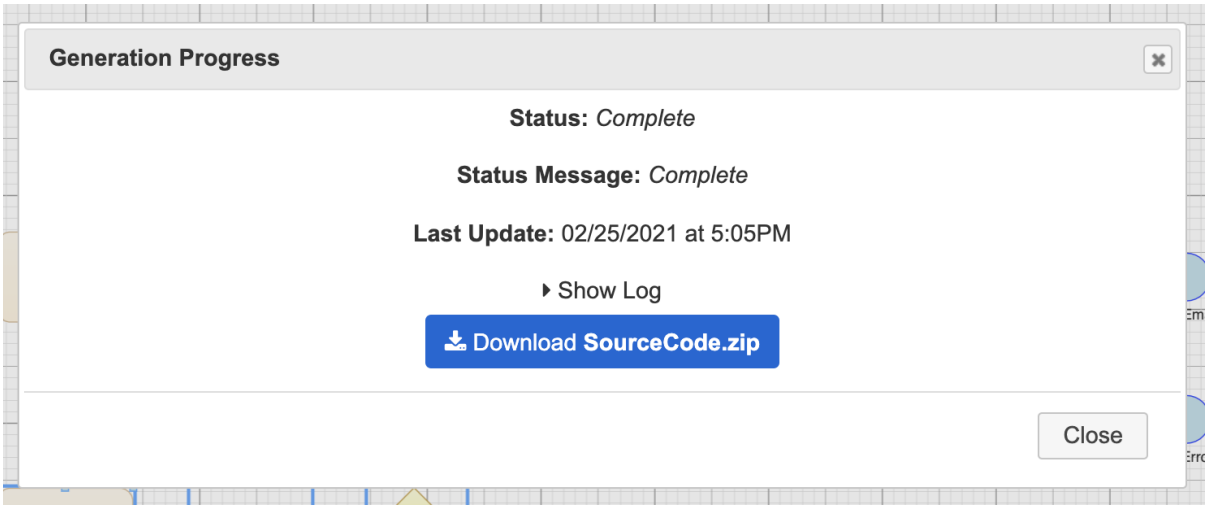
The initial steps used by this use case are essentialy the same steps described earlier for the other use case.

However, we don't want to run the tests manually; we aim to run them through the low-code automation facilities provided by TM.

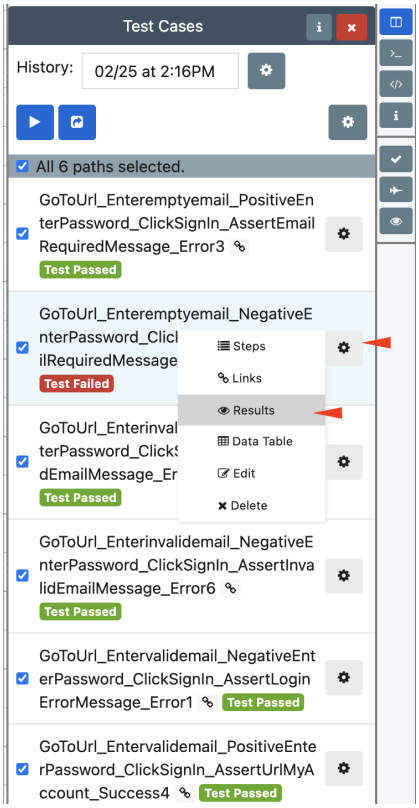
For that, we need to generate automation code for the selected paths/test cases, eventually commit it to Git, and run them.

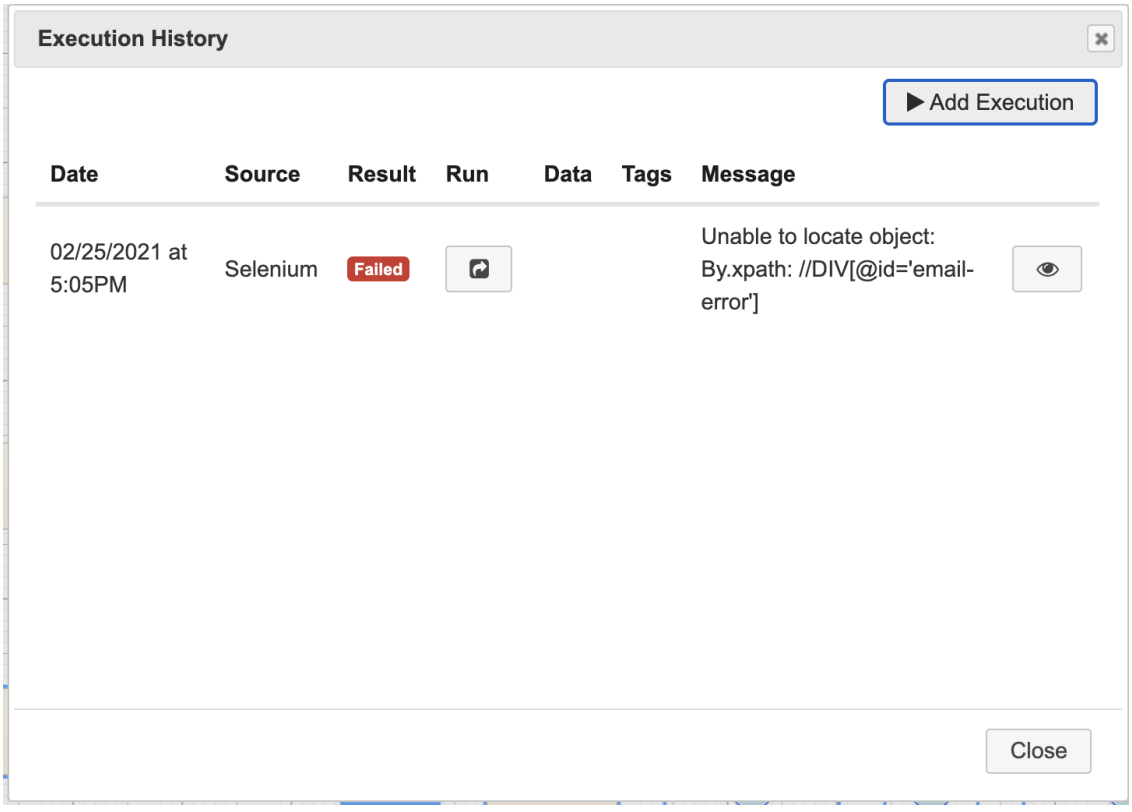


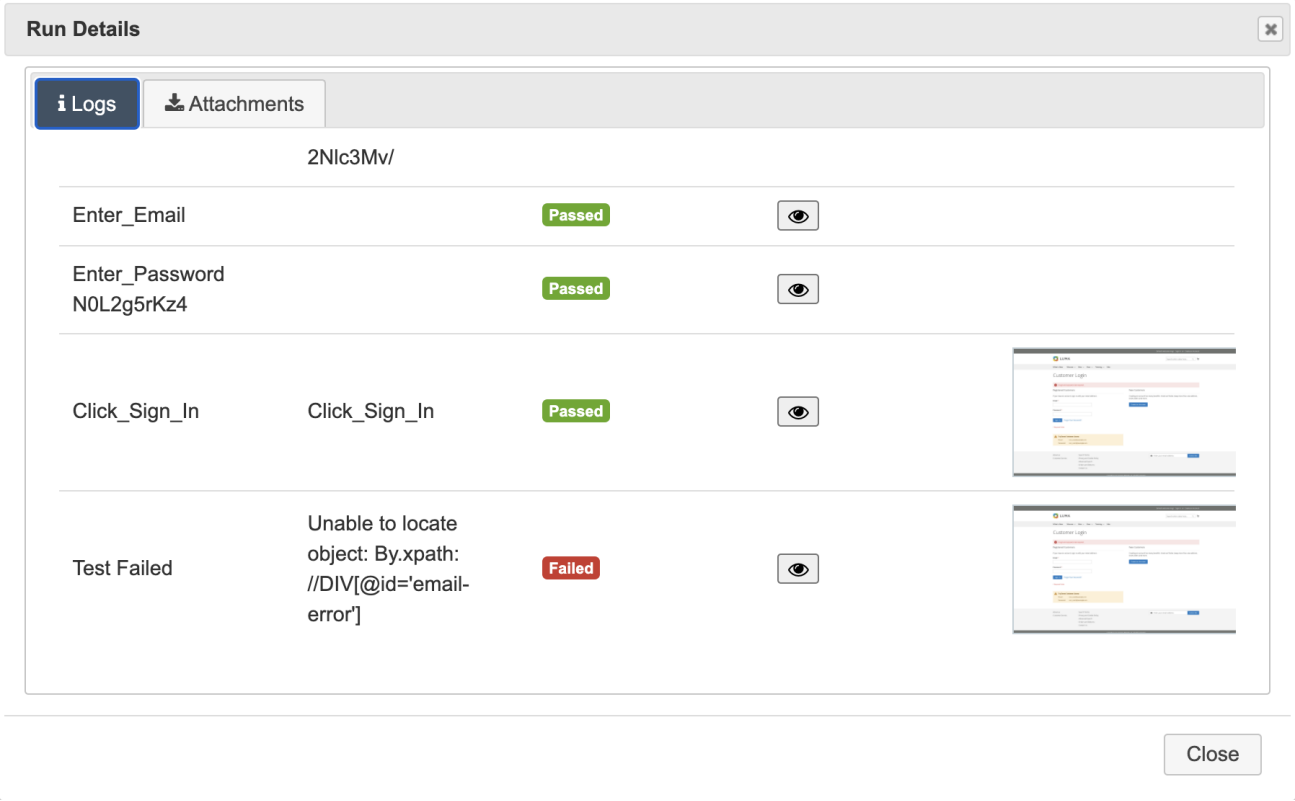
Using the automation framework assigned to the project, tests will be executed. It's also possible to download the generated source code (we can also just generate the code without executing it).



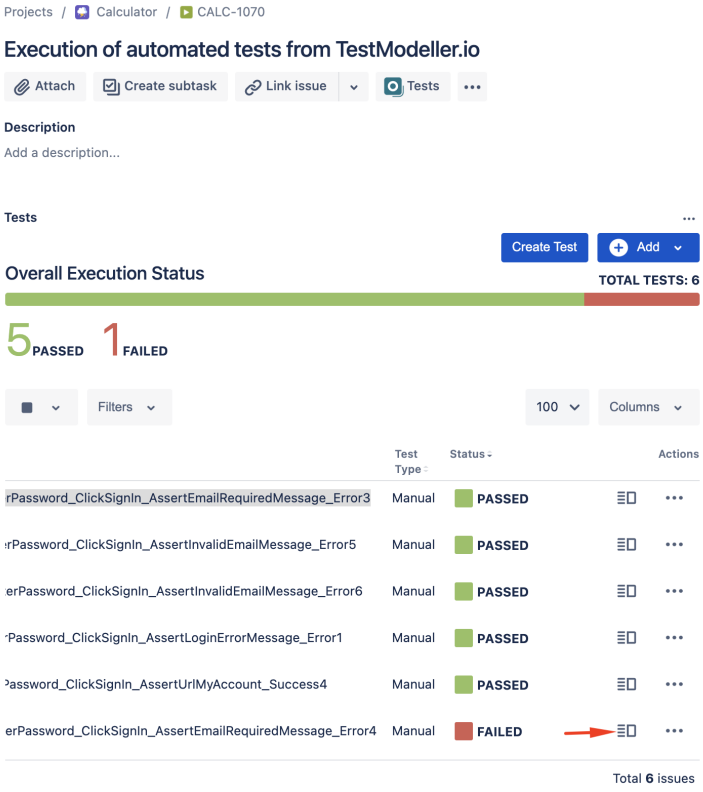
Basic result information will be shown on the Test Cases panel. It is also possible to deep-dive to see historical results or to look at the current results in more detail (e.g. step results, screenshots).


Test Cases
History:
02/25 at 2:16PM
All 6 paths selected.
GoToUrl_Enteremptyemail_PositiveEnterPassword_ClickSignIn_AssertEmailRequiredMessage_Error3
Test Passed
GoToUrl_Enteremptyemail_NegativeEnterPassword_ClicIlRequiredMessage
Test Failed
Steps
Links
Results
Data Table
Edit
Delete
GoToUrl_EnterinvalterPassword_ClickSdEmailMessage_Er
Test Passed
GoToUrl_EnterinvalidemailNegativeEnterPassword_ClickSignIn_AssertInvalidEmailMessage_Error6
Test Passed
GoToUrl_EntervalidemailNegativeEnterPassword_ClickSignIn_AssertLoginErrorMessage_Error1
Test Passed
GoToUrl_EntervalidemailPositiveEnterPassword_ClickSignIn_AssertUrlMyAccount_Success4
Test Passed

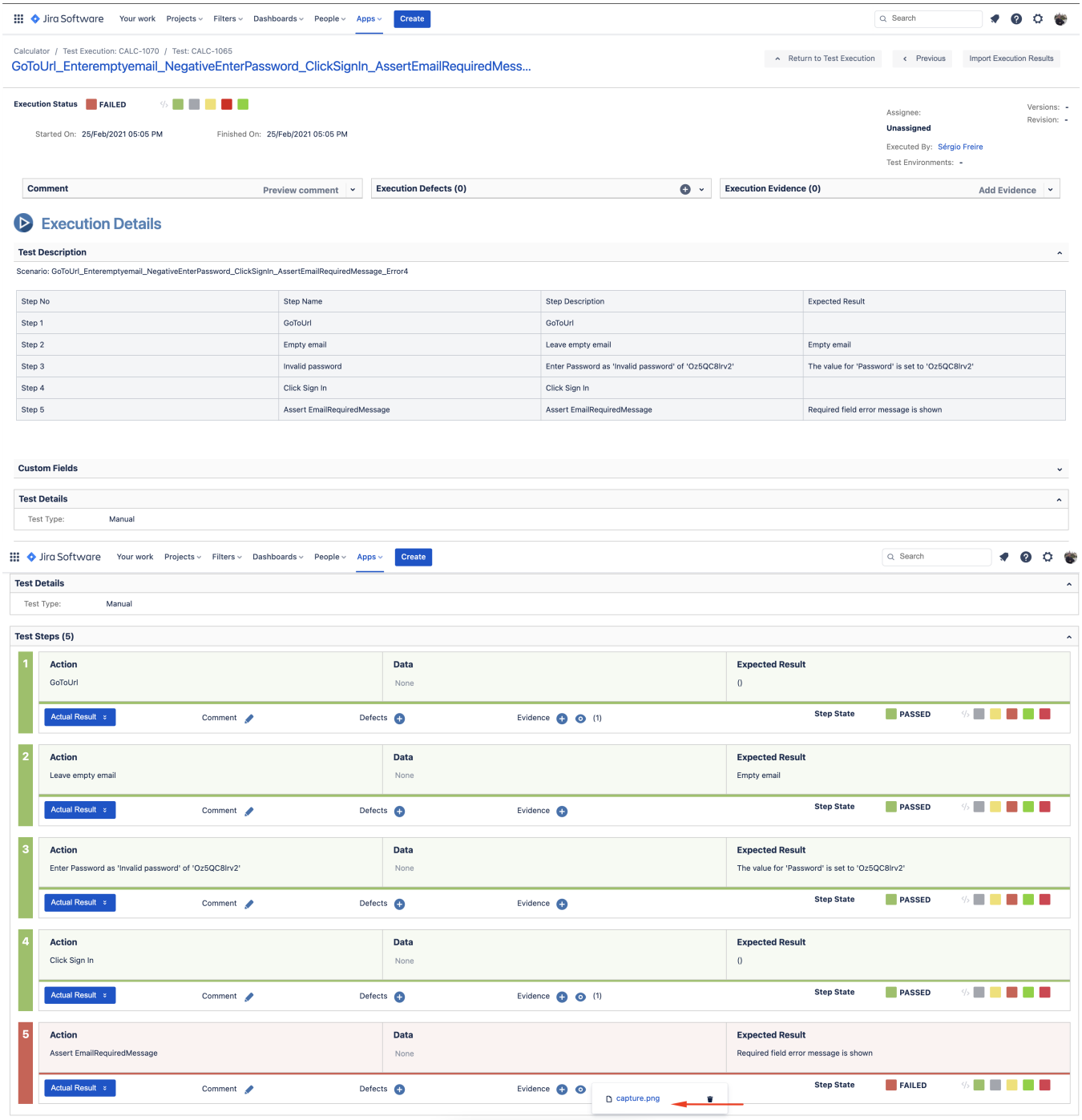

Execution History
Add Execution
Date
Source
Result
Run
Data
Tags
Message
02/25/2021 at 5:05PM
Selenium
Failed
Unable to locate object: By.xpath: //DIV[@id='email-error']
Close

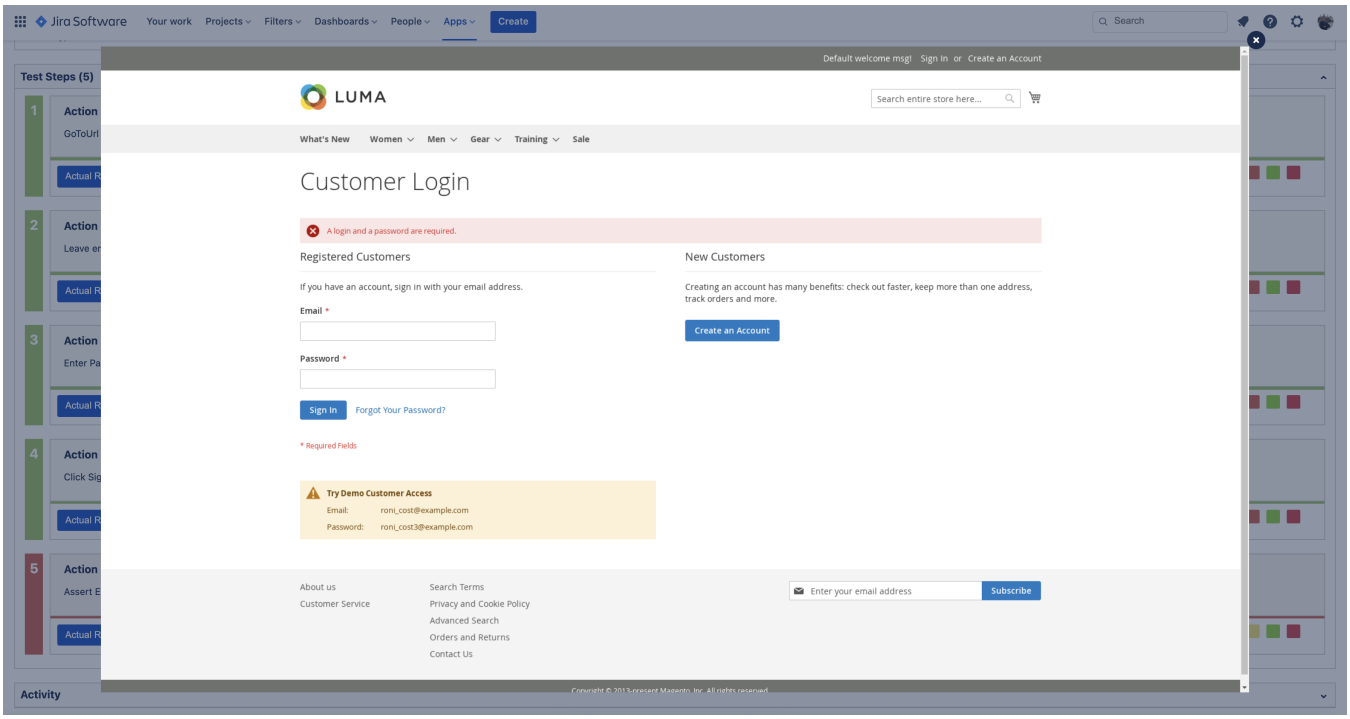


When running the automated test, and since the test cases in TM are already linked to Test issues in Xray due to a previous Export operation, a Test Execution will be created in Xray containing the results for the executed test cases/paths.

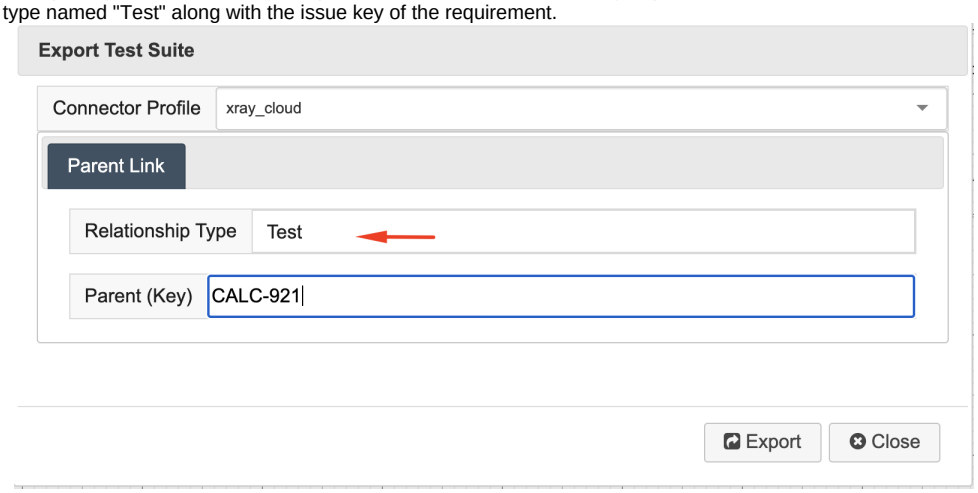


We can access the execution details for a specific Test (i.e. it's Test Run), where results per step are available, including screenshots.


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Calculator / Test Execution: CALC-1070 / Test: CALC-1065
GoToUrl_Enteremptyemail_NegativeEnterPassword_ClickSignIn_AssertEmailRequiredMess...
Return to Test Execution
Previous
Import Execution Results
Execution Status FAILED
Started On: 25/Feb/2021 05:05 PM
Finished On: 25/Feb/2021 05:05 PM
Assignee:
Unassigned
Executed By: Sérgio Freire
Test Environments: -
Versions: -
Revision: -
Comment
Preview comment
Execution Defects (0)
Execution Evidence (0)
Add Evidence
Execution Details
Test Description
Scenario: GoToUrl_Enteremptyemail_NegativeEnterPassword_ClickSignIn_AssertEmailRequiredMessage_Error4
Step No | Step Name | Step Description | Expected Result
Step 1 | GoToUrl | GoToUrl |
Step 2 | Empty email | Leave empty email | Empty email
Step 3 | Invalid password | Enter Password as 'Invalid password' of 'Oz5QC8lrv2' | The value for 'Password' is set to 'Oz5QC8lrv2'
Step 4 | Click Sign In | Click Sign In |
Step 5 | Assert EmailRequiredMessage | Assert EmailRequiredMessage | Required field error message is shown
Custom Fields
Test Details
Test Type: Manual
Test Steps (5)
1
Action
GoToUrl
Data
None
Expected Result
()
Actual Result
Comment
Defects
Evidence
(1)
Step State
PASSED
2
Action
Leave empty email
Data
None
Expected Result
Empty email
Actual Result
Comment
Defects
Evidence
Step State
PASSED
3
Action
Enter Password as 'Invalid password' of 'Oz5QC8lrv2'
Data
None
Expected Result
The value for 'Password' is set to 'Oz5QC8lrv2'
Actual Result
Comment
Defects
Evidence
Step State
PASSED
4
Action
Click Sign In
Data
None
Expected Result
()
Actual Result
Comment
Defects
Evidence
(1)
Step State
PASSED
5
Action
Assert EmailRequiredMessage
Data
None
Expected Result
Required field error message is shown
Actual Result
Comment
Defects
Evidence
capture.png
Step State
FAILED



Tips

- Whenever exporting test cases to Xray, it's possible to link them to an existing requirement and thus cover it. This can be achieved by specifying the link type named "Test" along with the issue key of the requirement.
  

- After uploading the test automation results, in Xray you can manually assign them to a specific version and also identify the Test Environment used (e.g. browser); this can be done on the respective Test Execution issues.

Learn more

- Test Modeller homepage
- Test Modeller tutorials
- Test Modeller Explorer (web scanner extension for Chrome)