./

Genesis Shadow Project Report

**Manual and Automation Test of FADR web application**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
| 1 | 8th Dec 2020 | Amit Das |  |  |  |
| 2 | 21st Dec 2020 | Amit Das |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc59526583)

[Table of Figures 4](#_Toc59526584)

[Table of Tables 4](#_Toc59526585)

[Shadow Project Topic: Manual and Automation Testing of FADR webpage (UI & API) 5](#_Toc59526586)

[Project Description: 5](#_Toc59526587)

[Tools and Technology Used: 5](#_Toc59526588)

[Topics Covered: 5](#_Toc59526589)

[Objectives 6](#_Toc59526590)

[Requirements 6](#_Toc59526591)

[High Level Requirements 6](#_Toc59526592)

[Low Level Requirements 6](#_Toc59526593)

[Manual Test Plan for FADR Customer Registration – API 7](#_Toc59526594)

[Manual Test plans for FADR Customer Registration – UI 10](#_Toc59526595)

[Getting Familiar with POSTMAN tool and REST API 12](#_Toc59526596)

[Test Automation scripts for Login into FADR application using Cucumber BDD, Selenium and Rest-assured – UI and API 15](#_Toc59526597)

[Reports generated by standalone cucumber BDD 15](#_Toc59526598)

[Test Automation scripts for Login into FADR application using Serenity – UI and API 17](#_Toc59526599)

[Serenity single page report type 17](#_Toc59526600)

[Full-fledged serenity reports 18](#_Toc59526601)

[Terminal Execution of the Test Suite 22](#_Toc59526602)

[Entire test suite execution 22](#_Toc59526603)

[Tag Specific features execution from terminal 23](#_Toc59526604)

# Table of Figures

[Figure 1: Postman Test suite for Login 12](#_Toc59467887)

[Figure 2: Postman test for invalid email format 12](#_Toc59467888)

[Figure 3: Postman test for valid login 13](#_Toc59467889)

[Figure 4: Postman test for unauthorized access 13](#_Toc59467890)

[Figure 5: Postman test for wrong password 14](#_Toc59467891)

[Figure 6: Postman test for empty attribute 14](#_Toc59467892)

[Figure 7: Auto-generated report for automation test not using Serenity 15](#_Toc59467893)

[Figure 8: Pretty report using cucumber 16](#_Toc59467894)

[Figure 9: Report using Pretty plugin 16](#_Toc59467895)

[Figure 10: Serenity single page report-1 17](#_Toc59467896)

[Figure 11: Serenity single page report-2 17](#_Toc59467897)

[Figure 12: Comprehensive report- Overall results 18](#_Toc59467898)

[Figure 13: Functional coverage 18](#_Toc59467899)

[Figure 14: Integrated requirement documentation 19](#_Toc59467900)

[Figure 15: Feature expansion 19](#_Toc59467901)

[Figure 16: Detailed data table for scenarios 20](#_Toc59467902)

[Figure 17: Steps with screenshots 20](#_Toc59467903)

[Figure 18: Steps breakdown before action with screenshot 21](#_Toc59467904)

[Figure 19: Steps breakdown after action with screenshot 21](#_Toc59467905)

[Figure 20: Run entire test suite 22](#_Toc59467906)

[Figure 21: Entire test suite execution completion 22](#_Toc59467907)

[Figure 22: Tag specific feature execution 23](#_Toc59467908)

[Figure 23: Tag specific feature execution (API SECURITY) 23](#_Toc59467909)

[Figure 24: Tag specific test execution 23](#_Toc59467910)

[Figure 25: System configuration in single page report 23](#_Toc59467911)

# Table of Tables

[Table 1: High Level Requirements 6](#_Toc59526523)

[Table 2: Low Level Requirements 6](#_Toc59526524)

[Table 3: Manual test plans for API 9](#_Toc59526525)

[Table 4: Manual test plans for UI 11](#_Toc59526526)

# Shadow Project Topic: Manual and Automation Testing of FADR webpage (UI & API)

## Project Description:

The goal of the project is to learn to write automation scripts for web applications. The web application being used by us is the Factory Assisted Device Reset(FADR) for EATON. The test cases must be written in a Behavior Driven Development(BDD) framework, to make it easy for a person from non-technical background to easily understand what test cases are being executed.

Tools and Technology Used:

* Excel
* POSTMAN
* JAVA scripting
* Cucumber BDD
* Selenium
* Rest-assured
* Junit

## Topics Covered:

* Understanding the requirements
* Writing manual test plan for UI and API
* Using the POSTMAN tool to test REST APIs
* Using Cucumber BDD to write test scenarios
* Using Selenium with Cucumber BDD for test automation of UI
* Using Rest-assured with Cucumber BDD for test automation of API
* Using Serenity with Cucumber BDD and Rest-assured for automation testing of API
* Using Serenity with Cucumber BDD and selenium for automation testing of UI
* Report generation using Serenity
* Running test suites on different environments
* Tag specific feature execution

## Objectives

* Write manual test plan for Customer Registration UI in FADR application.
* Write manual test plan for Customer Registration API of FADR application.
* Write test automation scripts for negative scenarios for login page(UI) for FADR application using cucumber BDD and selenium.
* Write test automation scripts for negative scenarios for login API for FADR application using cucumber BDD and Rest-assured.
* Write the test automation script for Login API using Serenity.
* Write the test automation script for Login page(UI) using Serenity.

## Requirements

Test automation of select features of the FADR web application.

High Level Requirements

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| HL\_01 | Automate testing of Login API. |
| HL\_02 | Automate testing of Customer Registration API. |
| HL\_03 | Automate testing of Logout API. |
| HL\_04 | Automate testing of UI of login webpage. |
| HL\_05 | Automate testing of UI of customer registration webpage. |

Table 1: High Level Requirements

Low Level Requirements

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| HL\_01\_LL\_01 | Test Login API for blank and invalid attributes. **(Negative scenario)** |
| HL\_01\_LL\_02 | Test Login API for valid attributes. **(Positive scenario)** |
| HL\_02\_LL\_01 | Test Customer registration API for blank attributes. **(Negative scenario)** |
| HL\_02\_LL\_02 | Test Customer registration API for invalid, expired and blank authentication. **(Negative scenario)** |
| HL\_03\_LL\_01 | Test Logout API with valid userID and valid authentication. **(Positive scenario)** |
| HL\_03\_LL\_02 | Test Logout API with invalid, blank and expired authentication.  **(Negative scenario)** |
| HL\_03\_LL\_03 | Test Logout API with invalid and blank userID.   **(Negative scenario)** |
| HL\_04\_LL\_01 | Test Login UI with correct credentials.   **(Positive scenario)** |
| HL\_04\_LL\_02 | Test Login UI with incorrect and blank credentials.  **(Negative scenario)** |
| HL\_04\_LL\_03 | Test password masking and unmasking in Login page.  **(Neutral scenario)** |
| HL\_04\_LL\_04 | Test forgot password and contact support section of Login page. **(Neutral scenario)** |
| HL\_05\_LL\_01 | Test Customer registration page with blank fields. **(Negative scenario)** |
| HL\_05\_LL\_02 | Test Customer registration page with incorrect phone and email format. **(Negative scenario)** |
| HL\_05\_LL\_03 | Test Customer registration page input fields character limits. **(Neutral scenario)** |
| HL\_05\_LL\_04 | Test the Reset button on Customer registration page. **(Neutral scenario)** |

Table 2: Low Level Requirements

## Manual Test Plan for FADR Customer Registration – API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No.** | **Test Scenario** | **Precondition** | **Actions** | **Expected Results** |
| 1 | Login API with wrong email and valid password | Availability of login end point | Enter Invalid email format Enter Valid password     Make POST Request to the endpoint.       Validate Response Code and Response Body received. | **Request Body** { "password": "Somerandompassword", "email": "admin@eaton" }  Response should be sent successfully  **1. Response Code:** 400 (Bad Request) **2. Response Body:** { "errorCode": 9007, "errorDescription": "Please provide  valid email address" } |
| 2 | Login API with correct email and wrong password | Availability of login end point | Enter valid email Enter wrong password     Make a POST Request to the endpoint.      Validate the Response Code and Response Body received. | **Request Body** { "password": "Somerandompassword", "email": "fadr\_support\_admin@eaton.com" }  Response should be sent successfully  **1. Response Code:** 401 (Unauthorized) **2. Response Body:** { "errorCode": 9001, "errorDescription": "Unauthorized access" } |
| 3 | Login API with blank email and  valid password | Availability of login end point | Leave the email attribute blank. Enter valid password     Make a POST Request to the endpoint.       Validate the Response Code and  Response Body received. | **Request Body** { "password": "Somerandompassword", "email": "" }  Response should be sent successfully  **1. Response Code:** 400 (Bad Request) **2. Response Body:** { "errorCode": 1001, "errorDescription": "User name or password  cannot be null or empty" } |
| 4 | Login API with valid email and  blank password | Availability of login end point | Enter the valid email Leave the password attribute blank.     Make a POST Request to the endpoint.       Validate the Response Code and  Response Body received. | **Request Body** { "password": "", "email": "fadr\_support\_admin@eaton.com" }  Response should be sent successfully.  **1. Response Code**: 400 (Bad Request) **2. Response Body:** { "errorCode": 1001, "errorDescription": "User name or password  cannot be null or empty" } |
| 5 | Login API with blank email and  blank password | Availability of login end point | Leave the email attribute blank. Leave the password attribute blank.     Make a POST Request to the endpoint.       Validate the Response Code and  Response Body received. | **Request Body** { "password": "", "email": "" }  Response should be sent successfully.  **1. Response Code:** 400 (Bad Request) **2. Response Body:** { "errorCode": 1001, "errorDescription": "User name or password  cannot be null or empty" } |
| 6 | Login API with unregistered  email and valid password | Availability of login end point | Enter a unregistered email Enter a valid password    Make a POST Request to the endpoint.      Validate the Response Code and  Response Body received. | Request Body{ "password": "Somerandompassword", "email": "admin@eaton.com" }  Response should be sent successfully  **1. Response Code:** 401 (Unauthorized) **2. Response Body:** { "errorCode": 9001, "errorDescription": "Unauthorized access" } |
| 7 | Login API without email attribute | Availability of login end point | Enter a any password Do not pass an email attribute   Make a POST Request to endpoint       Validate the Response Code and response body received | **Request Body**{ "password": "Somerandompassword" }  Response should be sent successfully  1. **Response Code**: 400 (Bad Request) 2. **Response Body**: { "errorCode": 1001, "errorDescription": "User name or password  cannot be null or empty" } |
| 8 | Login API without password attribute | Availability of login end point | Enter any email Do not pass a password attribute   Make a POST Request to endpoint       Validate the Response Code and response body received | **Request Body**{ "email": "fadr\_admin@eaton.com" }  Response should be sent successfully  1. **Response Code**: 400 (Bad Request) 2. **Response Body**: { "errorCode": 1001, "errorDescription": "User name or password  cannot be null or empty" } |

Table 3: Manual test plans for API

## Manual Test plans for FADR Customer Registration – UI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl. No.** | **Test Scenario** | **Precondition** | **Actions** | **Expected Results** |
| 1 | Login with Valid email and wrong password | The login page should be open in browser | 1. The correct email address is entered "fadr\_support\_admin@eaton.com" 2. A wrong password is entered "somerandompassword"  3. Click on Login button | The user is returned to Home page of FADR with the  message "Invalid credentials" |
| 2 | Login with incorrect email and valid password | The login page should be open in browser | 1. An incorrect email is entered "admin@eaton.com" 2. Valid password is entered "ValidPassword"  3. Click on the Login button | The user is returned to Home page of FADR with the  message "Invalid credentials" |
| 3 | Login with incorrect email and  incorrect password | The login page should be open in browser | 1. An incorrect email is entered "admin@eaton.com" 2. An invalid password is entered "Somerandompassword"  3. Click on the Login button | The user is returned to Home page of FADR with the  message "Invalid credentials" |
| 4 | Login with invalid email format | The login page should be open in browser | 1.An incorrect email format is entered in the username input box "admin@eaton" | An error message below the email input field should be displayed |
| 5 | Remember Me checkbox | The login page should be open in browser | 1. Login into the application with correct credentials and remember me check box uncheck. 2. Logout and return to login page.  3. Verify is the username field is auto filled.    4. Login into the application with correct credentials and remember me check box checked. 5. Logout and return to login page.  6. Verify is the username field is auto filled. | The username field should not be auto filled.        The username field should be filled with email of last user. |
| 6 | Forgot password functionality | The login page should be open in browser | Click on the forgot password link on the login page | A new page labelled "Forgot password opens"  The page contains an input field to submit the  email associated with the account.  The page has emergency contact number  available. |
| 7 | Contact EATON support functionality | The login page should be open in browser | Click on the link "Contact an Eaton Support Representative" | A new help section dialog box pops up.  It contains a "General Questions" section with the contact email address.  It contains a "Emergency Support" section with the contact number. |
| 8 | Password masking and unmasking | The login page should be open in browser | 1. Enter correct email id "fadr\_support\_admin@eaton.com" 2. Enter any password "somerandompassword"  3. Click the eye button once.  4. Click the eye button once more. | The Password should be masked.  The Password should be unmasked.  The Password should be masked. |

Table 4: Manual test plans for UI

## Getting Familiar with POSTMAN tool and REST API

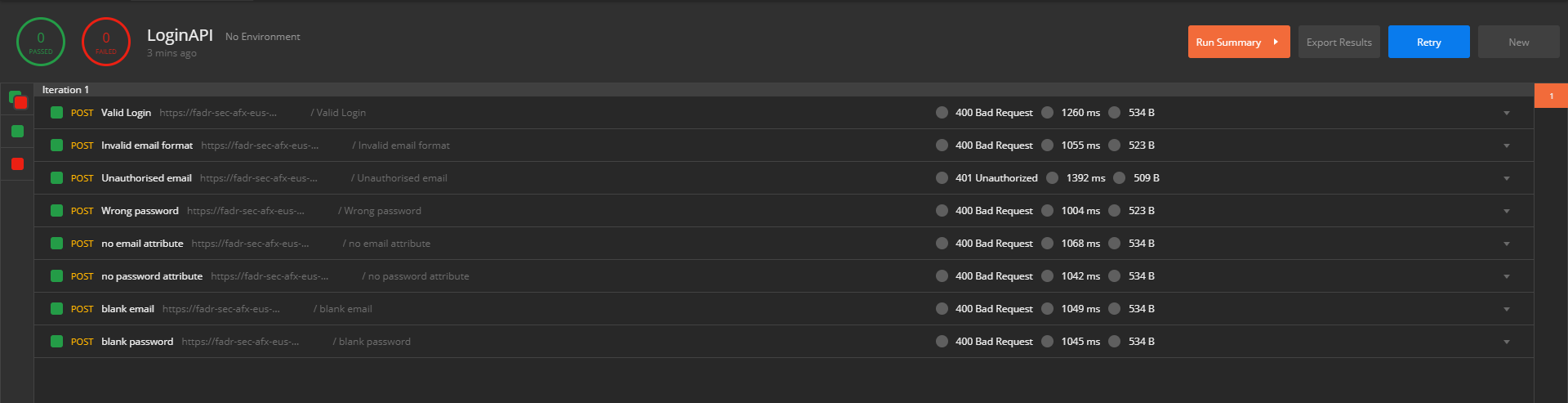


Figure 1: Postman Test suite for Login

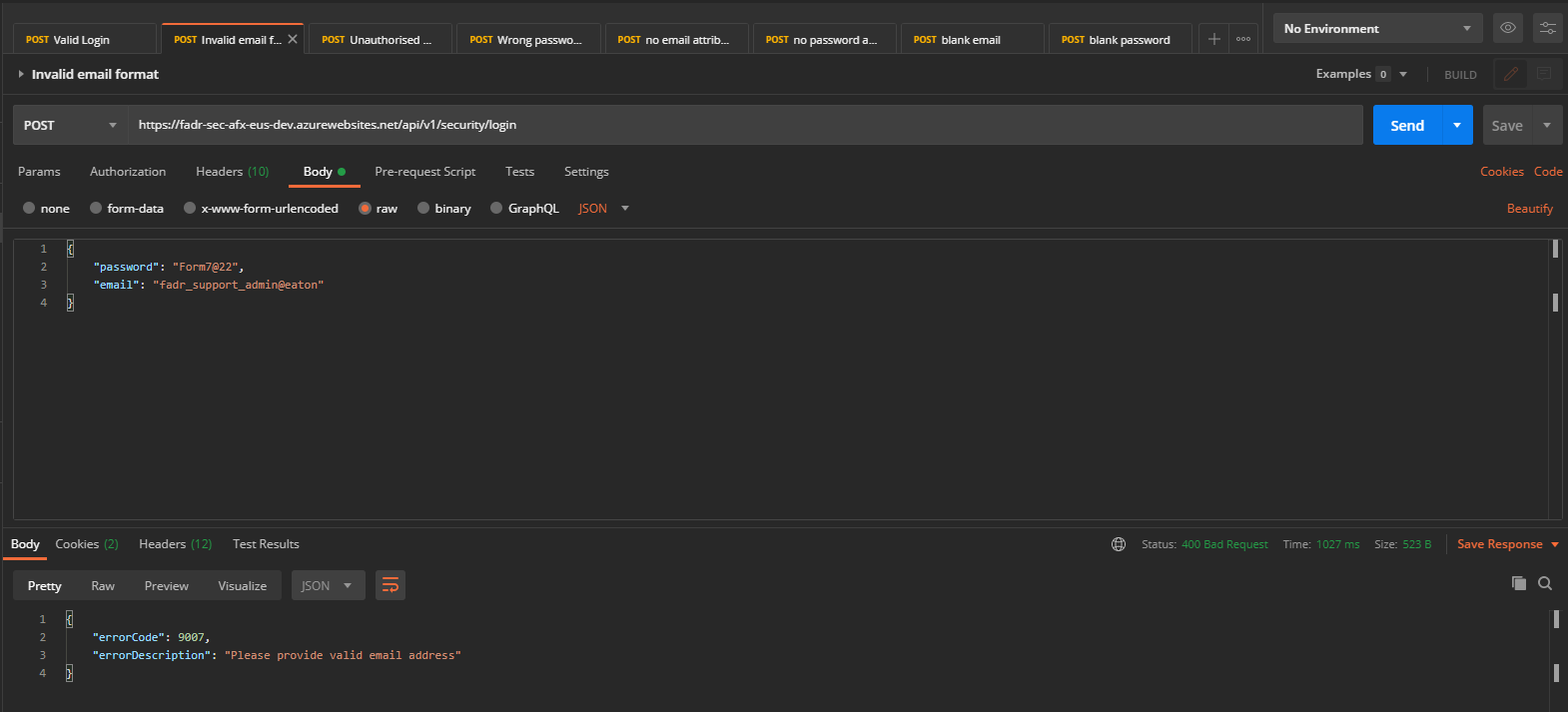


Figure 2: Postman test for invalid email format

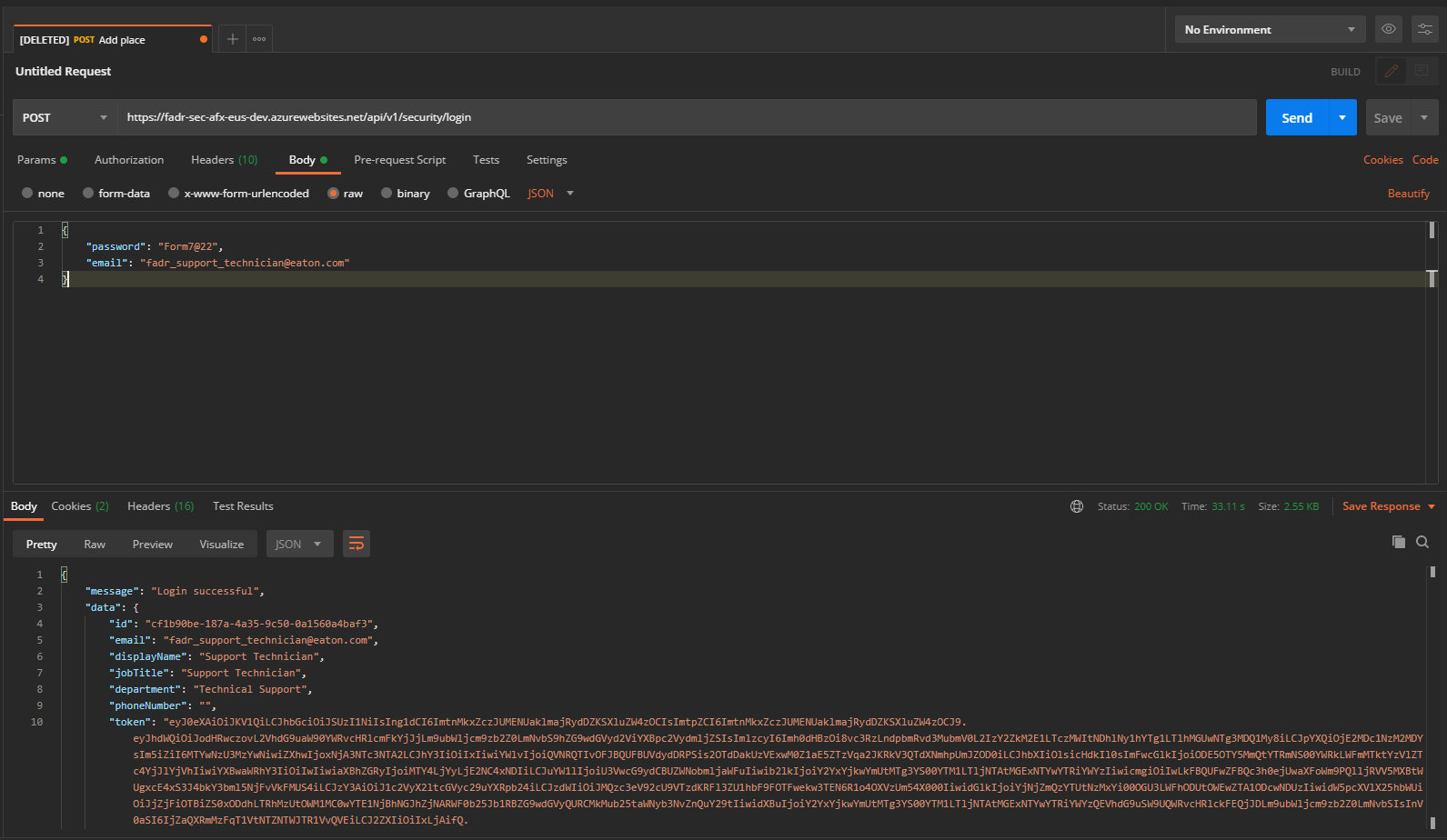


Figure 3: Postman test for valid login

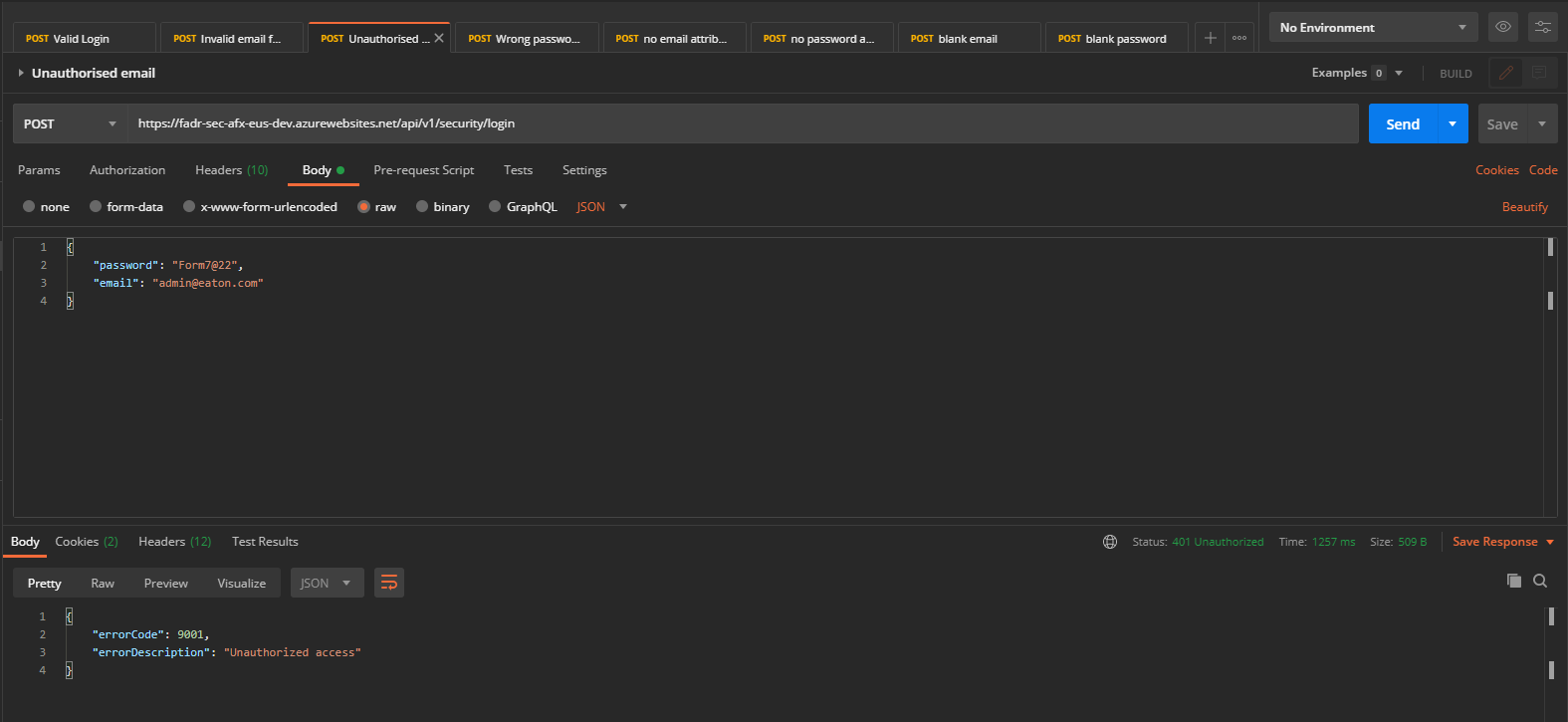


Figure 4: Postman test for unauthorized access

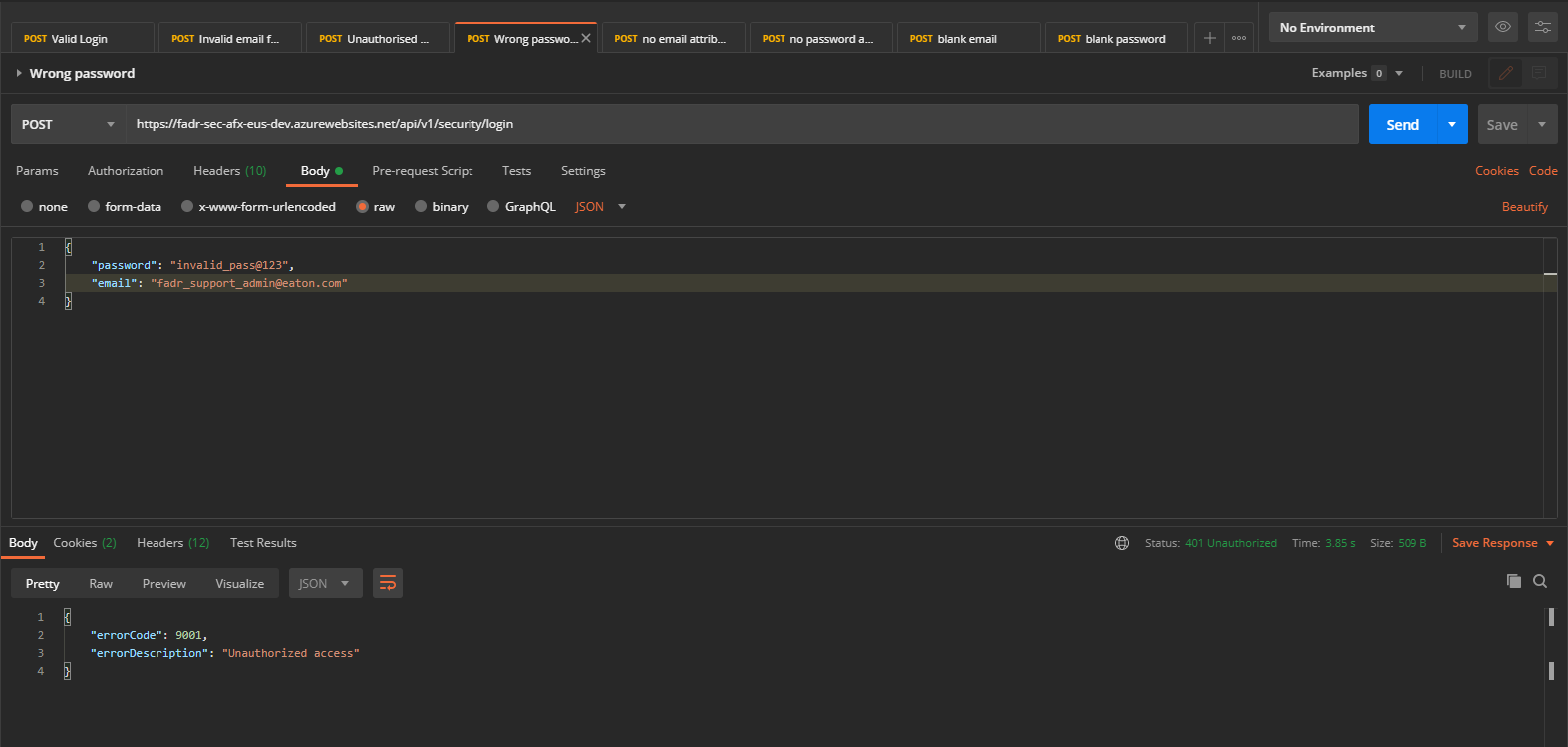


Figure 5: Postman test for wrong password

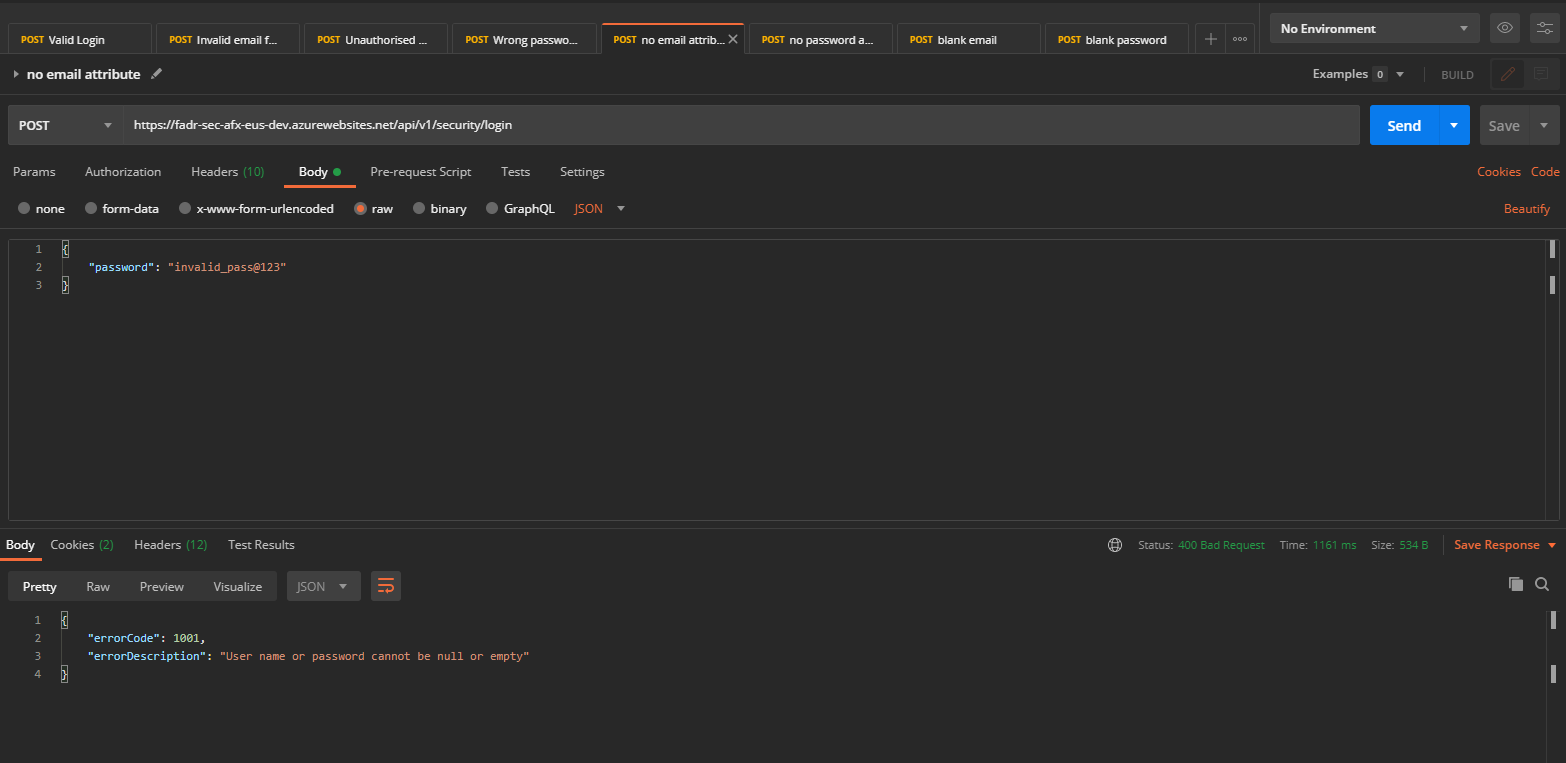


Figure 6: Postman test for empty attribute

## Test Automation scripts for Login into FADR application using Cucumber BDD, Selenium and Rest-assured – UI and API

* GitHub Repository Link: [Link to GitHub Repository](https://github.com/99002591/fadrdemoSerenity)

Reports generated by standalone cucumber BDD

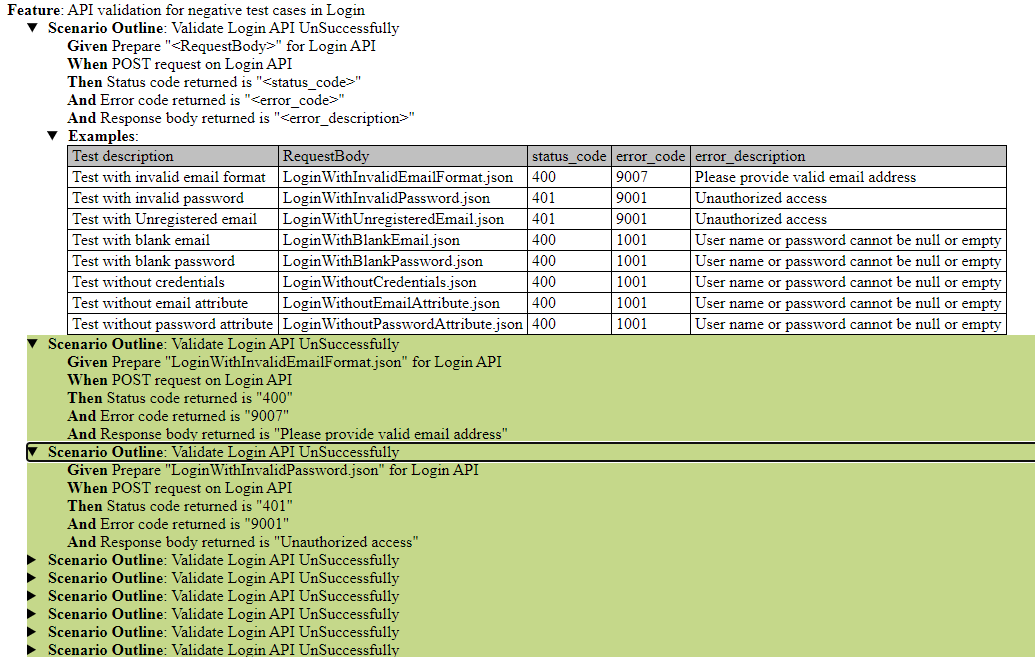


Figure 7: Auto-generated report for automation test not using Serenity

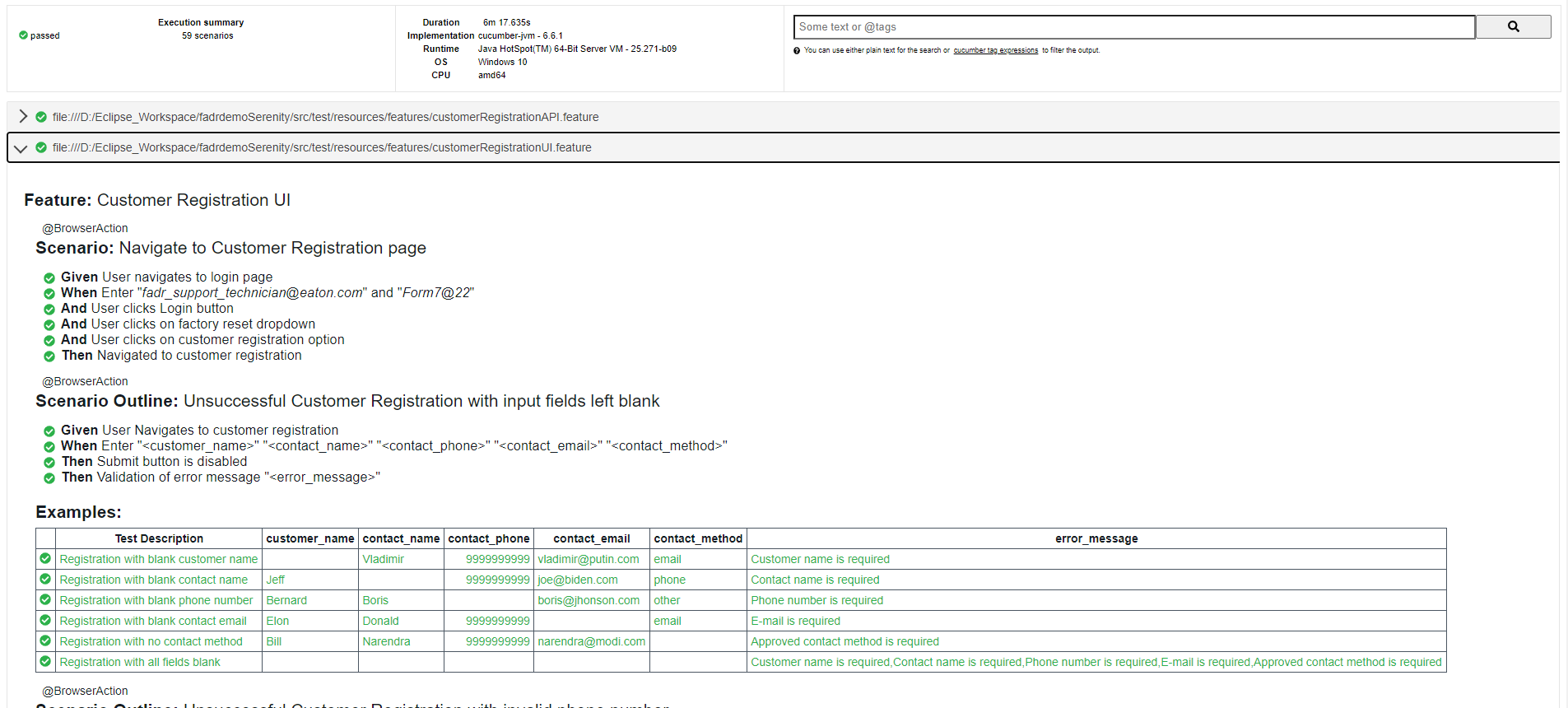


Figure 8: Pretty report using cucumber

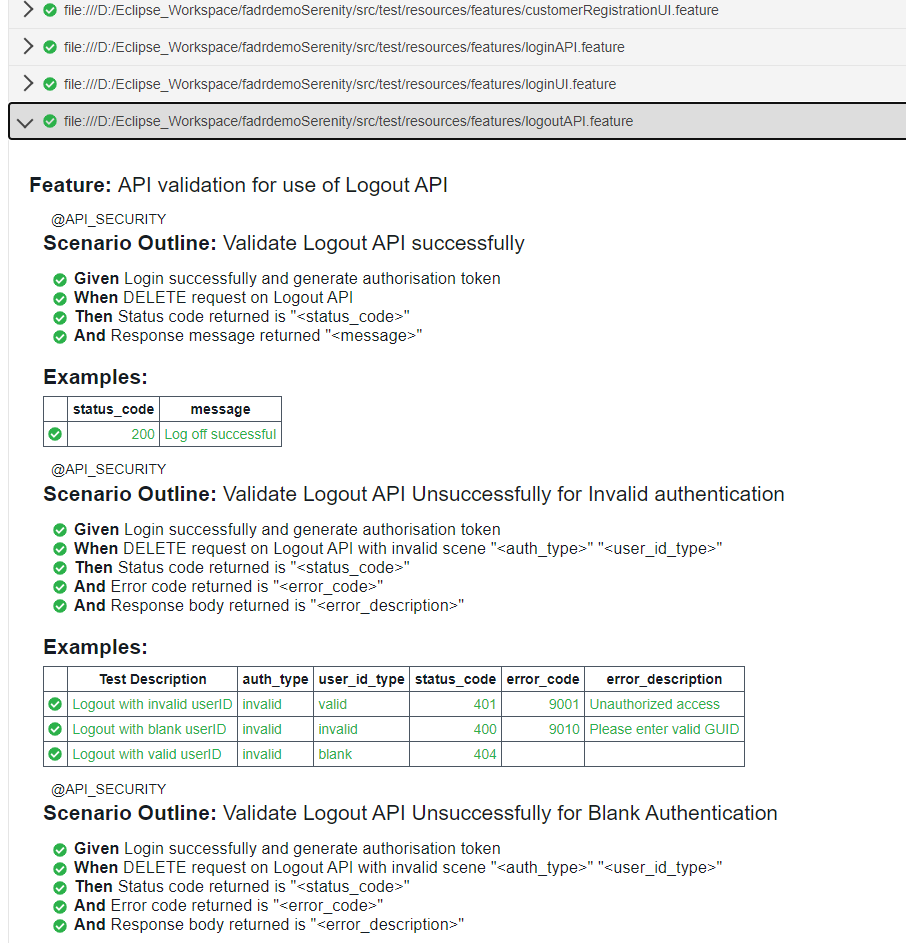


Figure 9: Report using Pretty plugin

## Test Automation scripts for Login into FADR application using Serenity – UI and API

* GitHub Repository Link: <https://github.com/99002591/fadrdemoSerenity>

Serenity single page report type

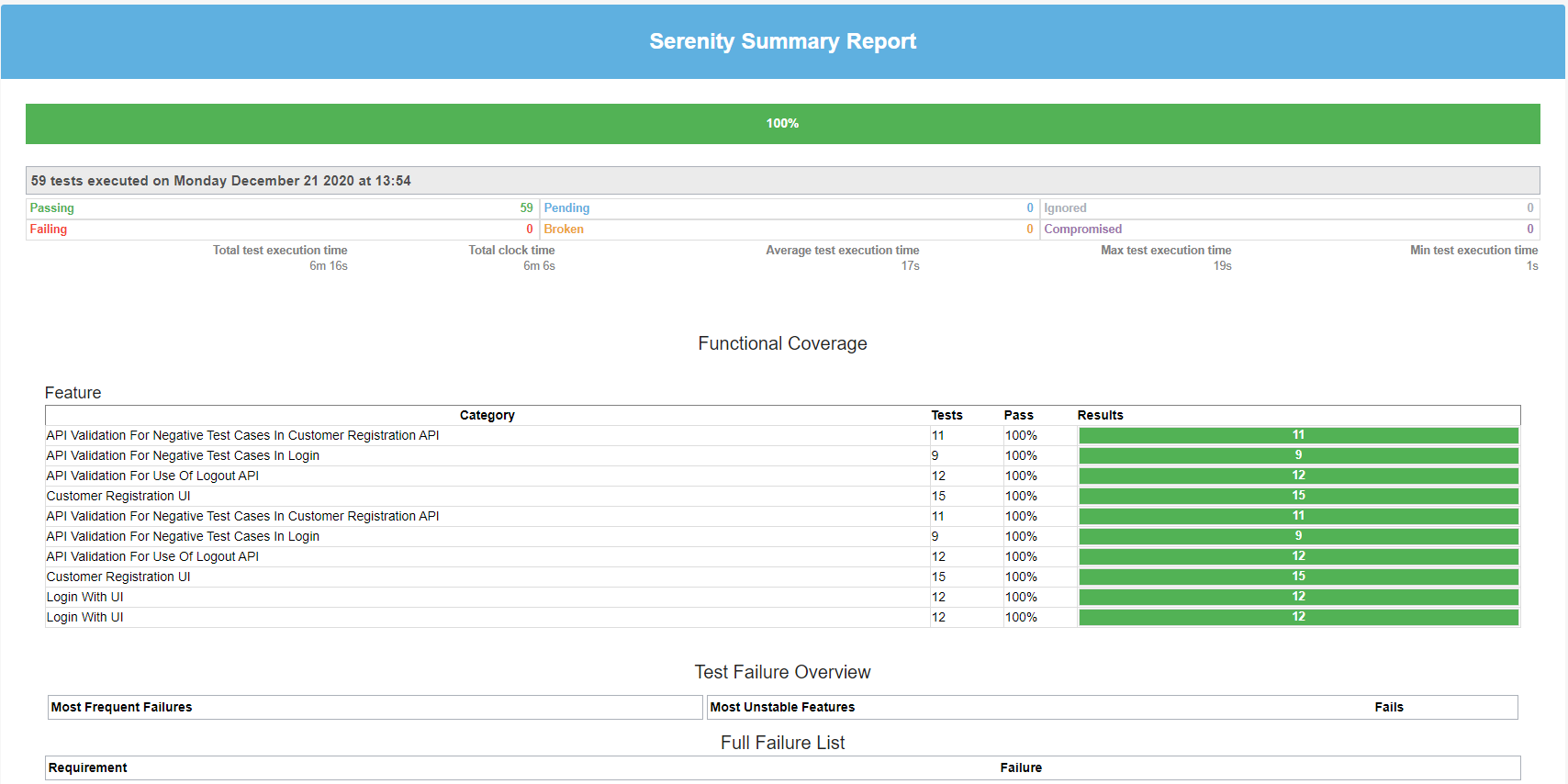


Figure 10: Serenity single page report-1

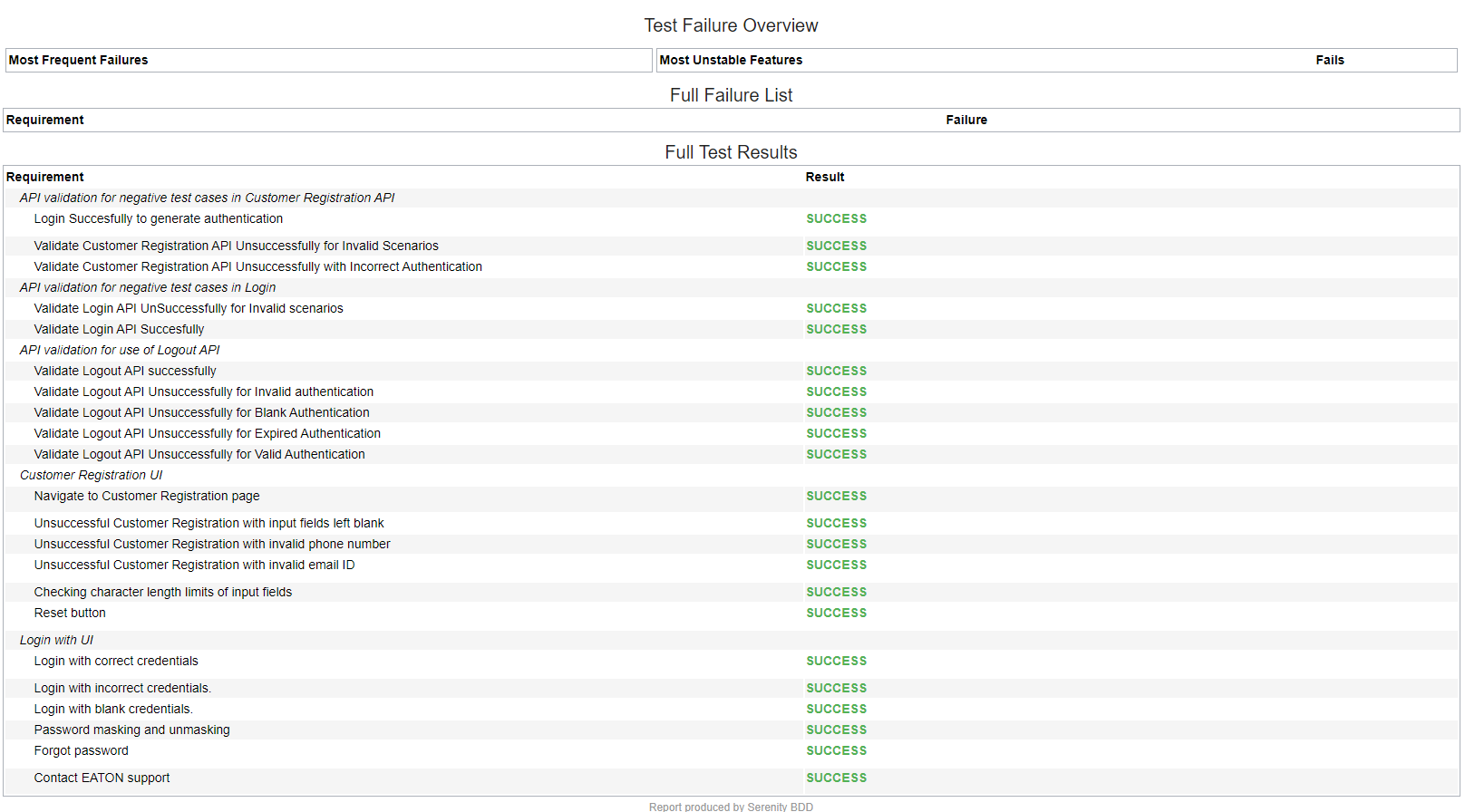


Figure 11: Serenity single page report-2

Full-fledged serenity reports

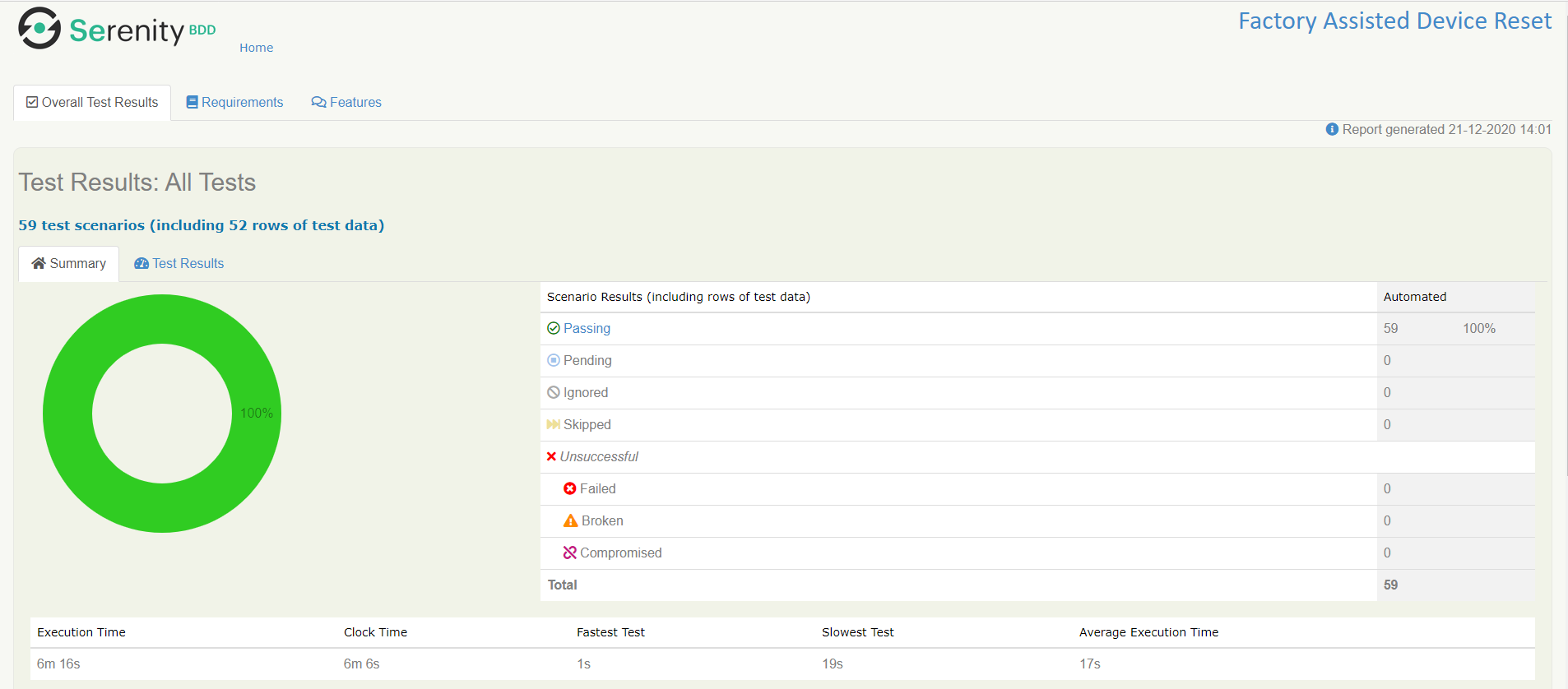


Figure 12: Comprehensive report- Overall results

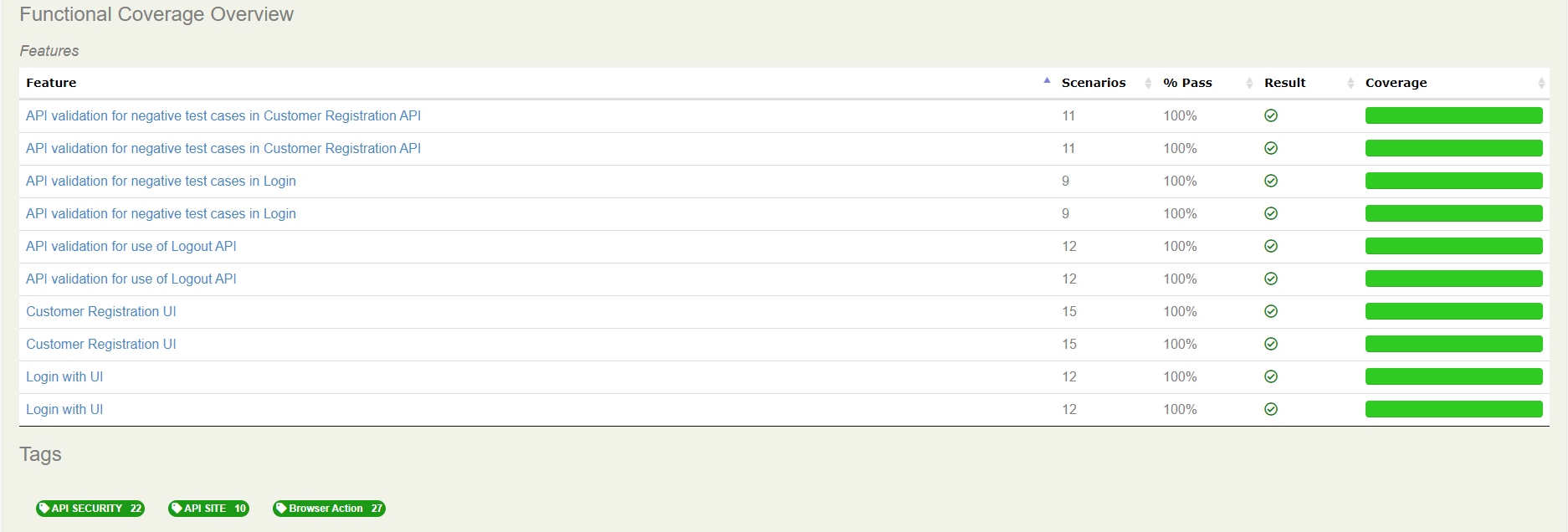


Figure 13: Functional coverage



Figure 14: Integrated requirement documentation

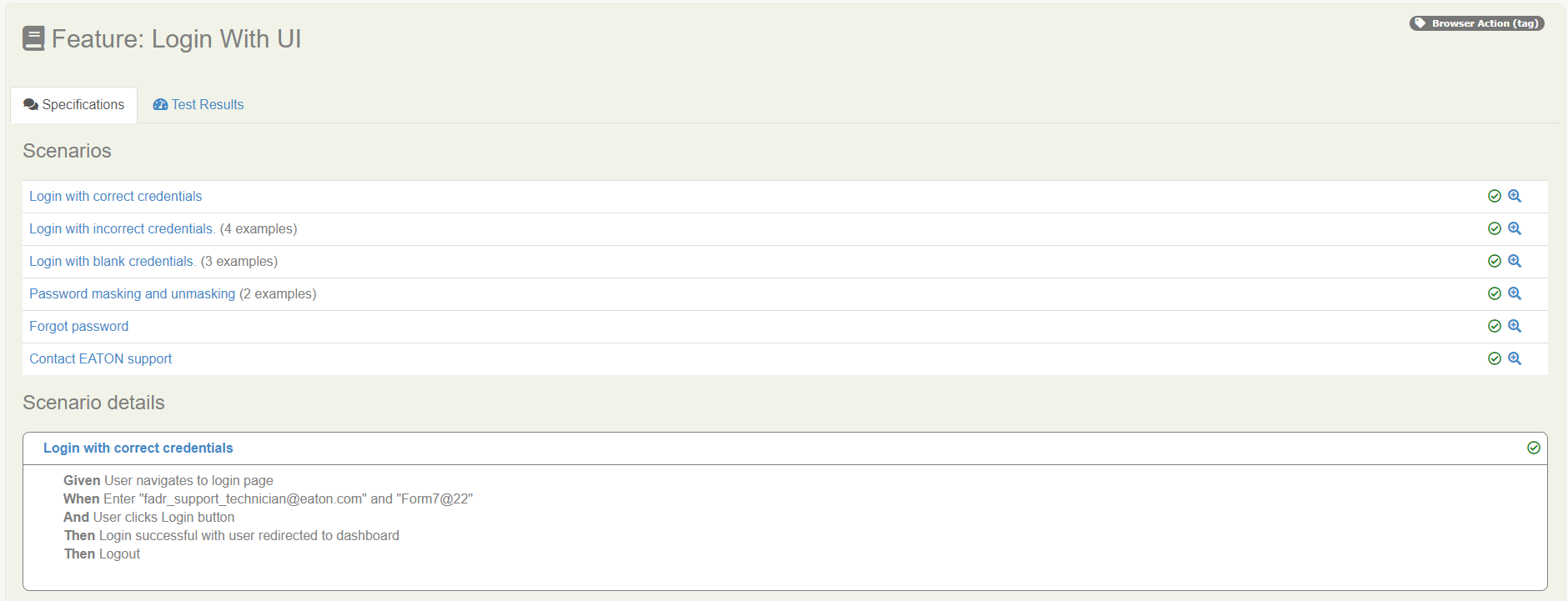


Figure 15: Feature expansion

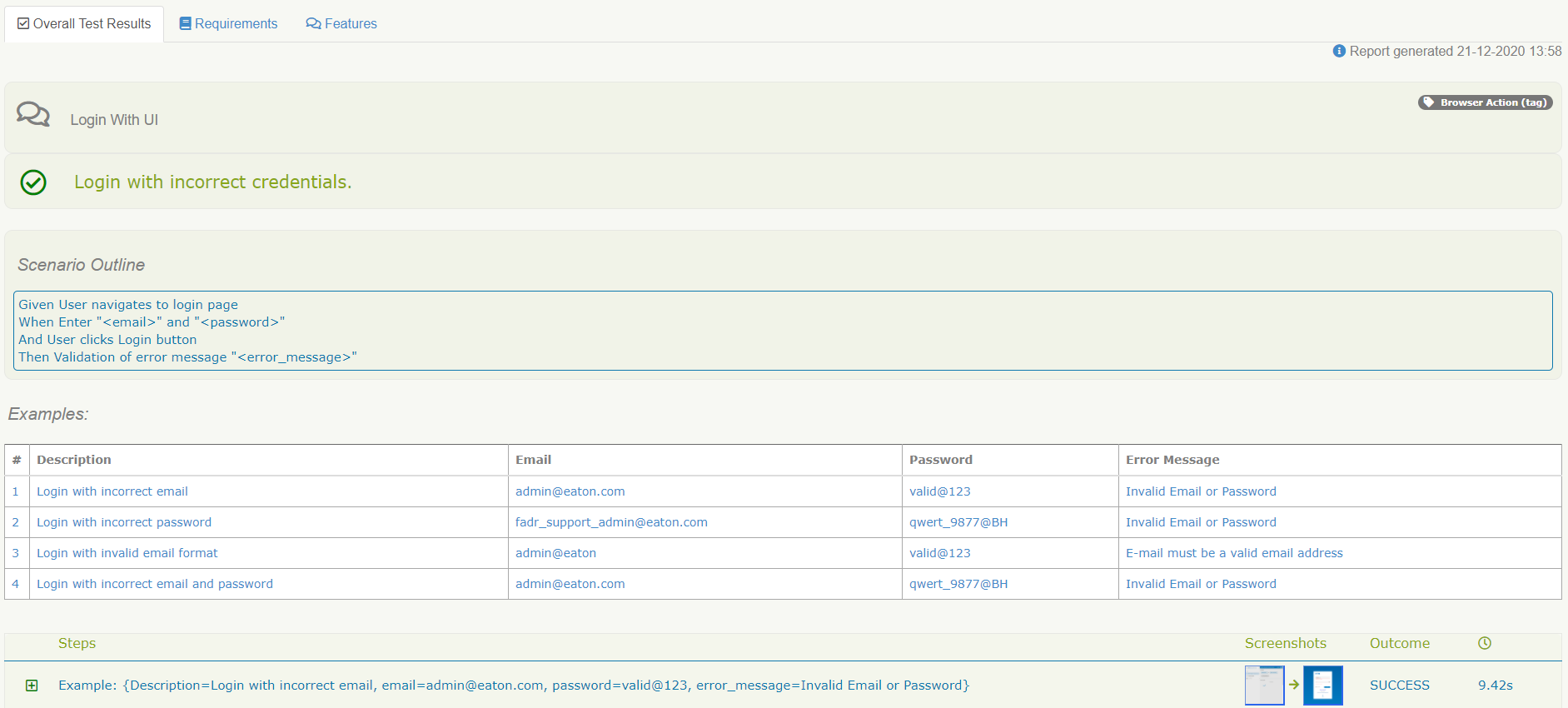


Figure 16: Detailed data table for scenarios



Figure 17: Steps with screenshots

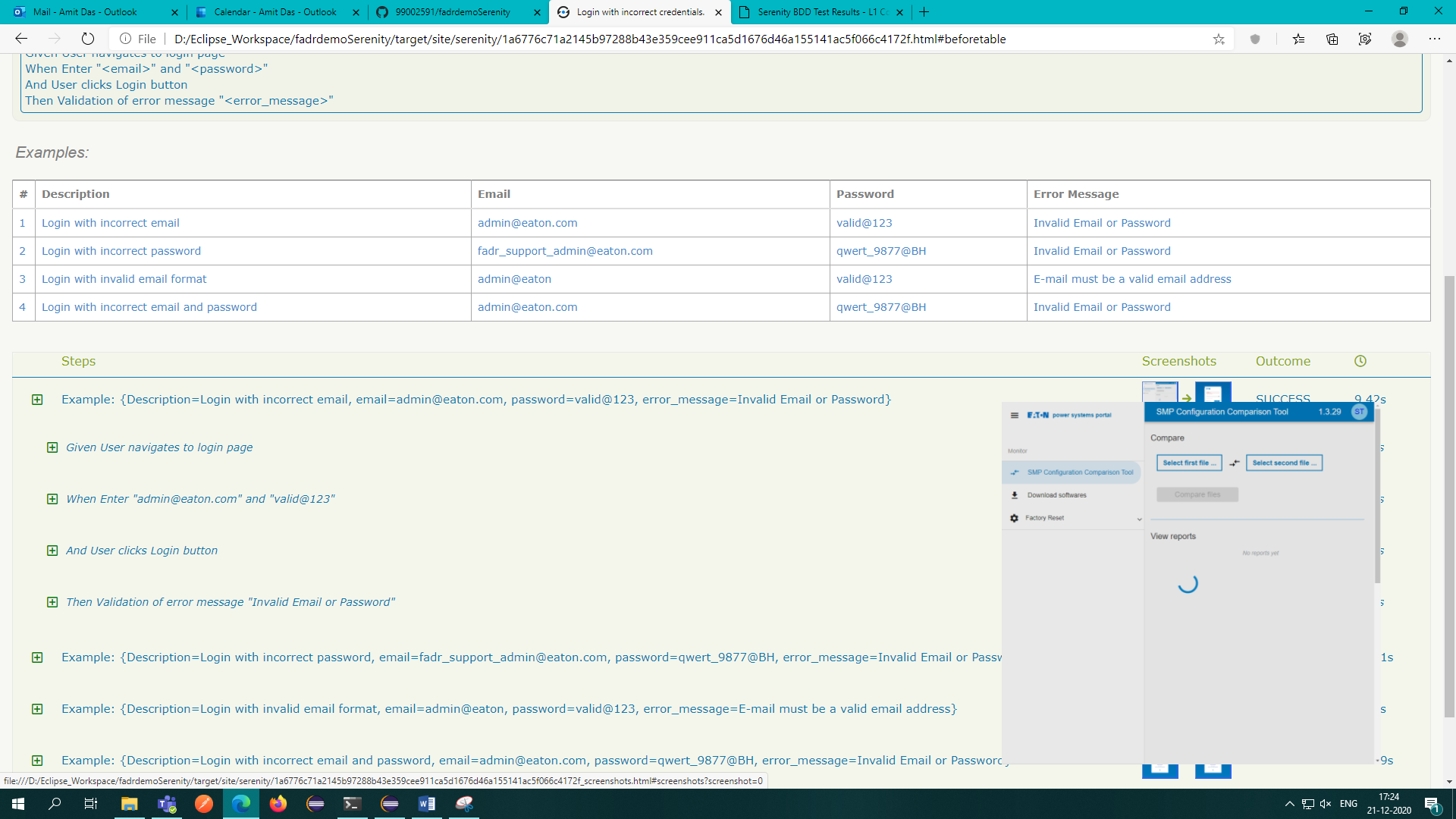


Figure 18: Steps breakdown before action with screenshot

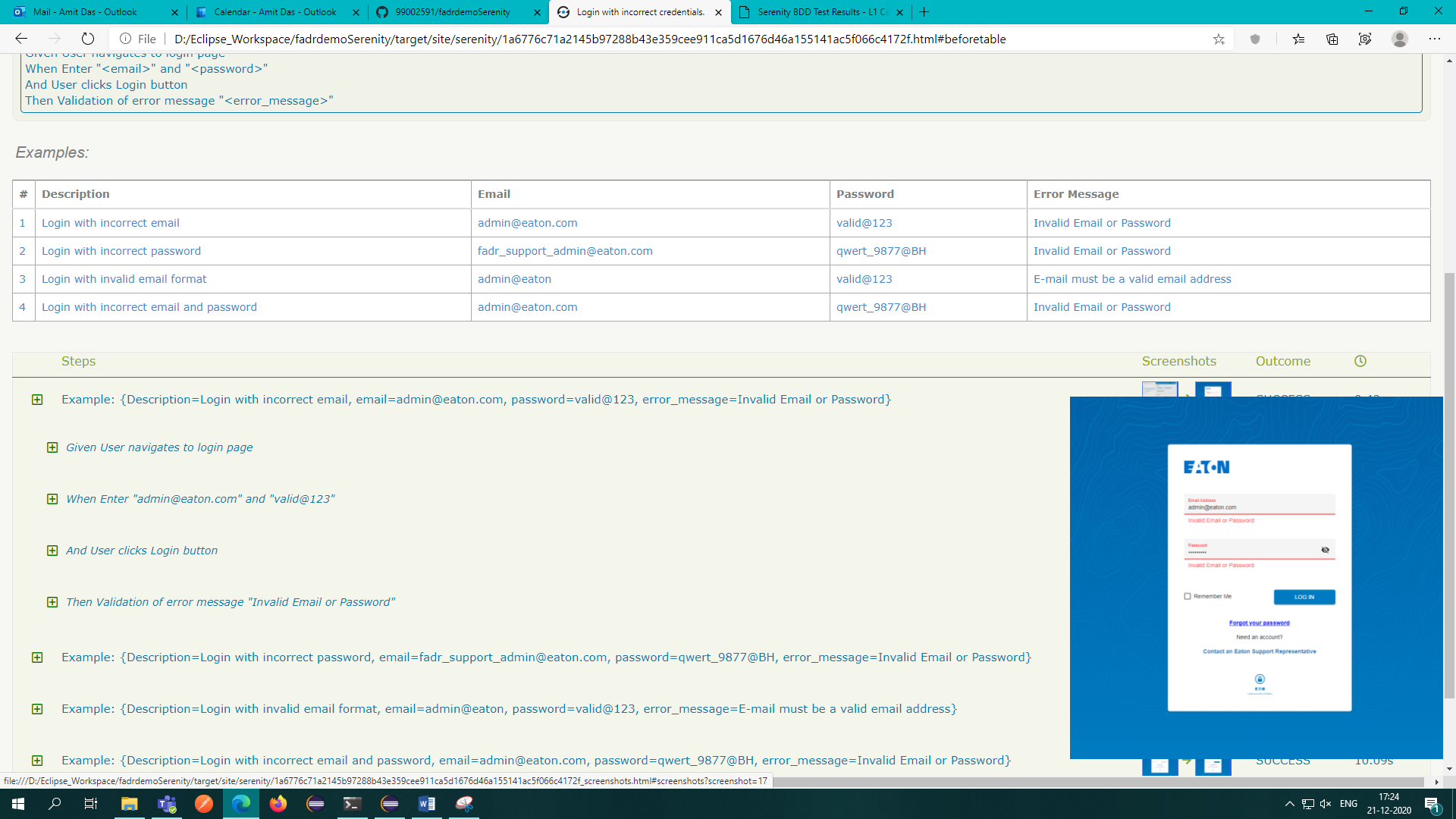


Figure 19: Steps breakdown after action with screenshot

## Terminal Execution of the Test Suite

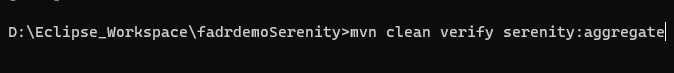


Figure 20: Run entire test suite

Entire test suite execution

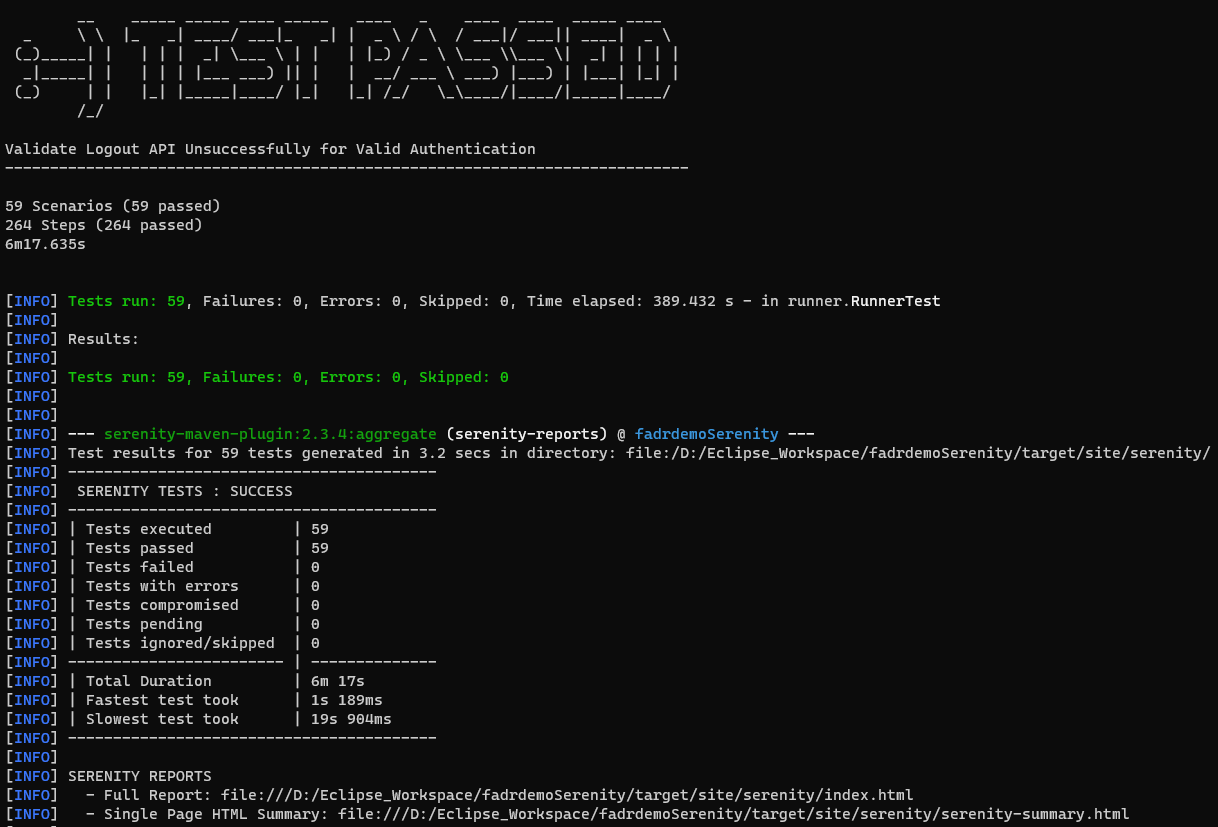


Figure 21: Entire test suite execution completion

Tag Specific features execution from terminal



Figure 22: Tag specific feature execution



Figure 23: Tag specific feature execution (API SECURITY)

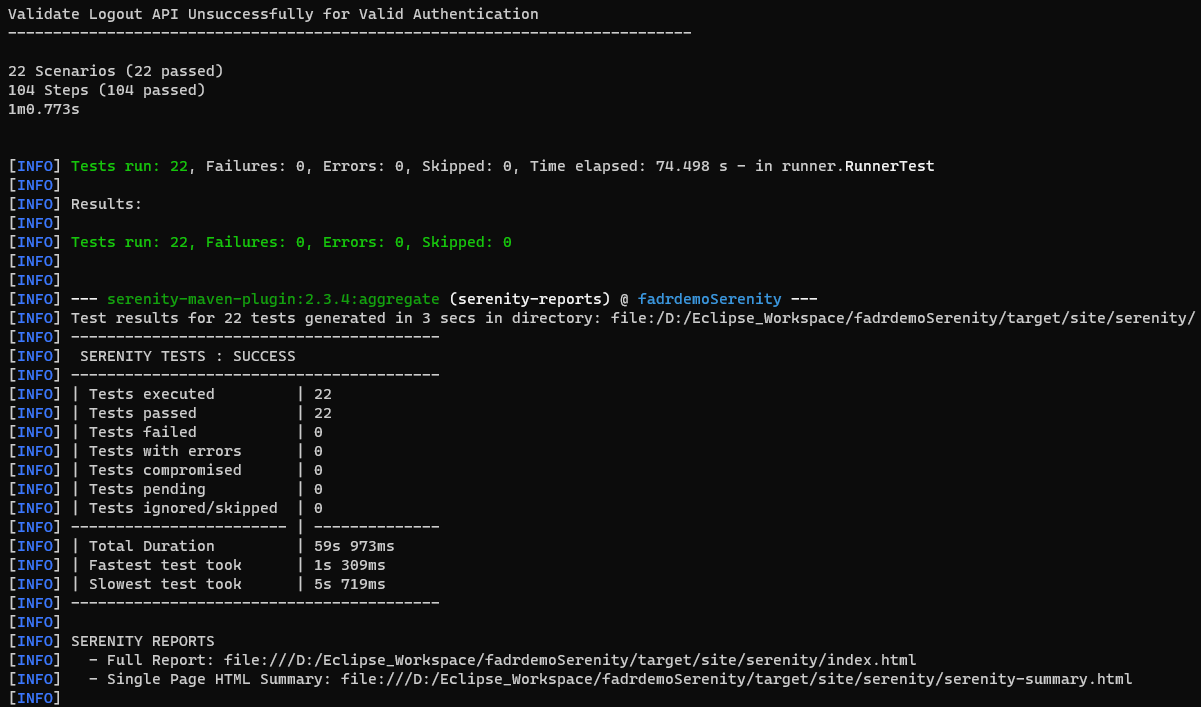


Figure 24: Tag specific test execution

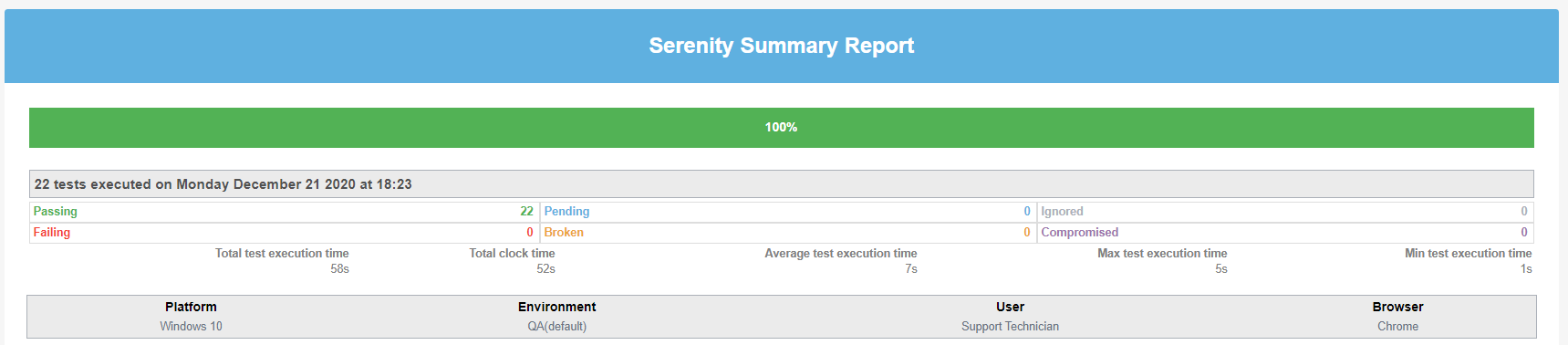


Figure 25: System configuration in single page report