ScopeX Fintech Assessment:

**Manual Testing Test case:**

**Smoke testing**: Checking basic and critical features first

1. First, we should always start with Smoke testing. Able to login first either google login method or sign-up method (create account)
2. Once signed in the user should be redirected to the dashboard.
3. Then able to add recipient. After verifying in same recipient details should be available in recipient list
4. Able to log out and return the user to the login screen.

**Function testing**:

1. Click on login button. It should display login page
2. Verifying once again using google account able to login.
3. Click on sign up button. it should display sign up page.
4. Test scenario for Name, Email and referral code text box

Name Text box:

Positive scenarios:

* Verify first able to write something in name text box
* Verify Able to write name with both upper- and lower-case alphabets
* Verify name with max and min alphabets defined by developer (for ex: 50 characters it will accept we should test with (0,1,2,10,20,30,40, 49,50,51)

Negative scenario:

* Verify It should not accept special characters, numbers, Doller symbol, and combination of number with alphabets, combination of special characters with alphabets and other combinations etc.

Password text box:

* Verify first able to write something in name text box
* Verify Able to write name with both upper- and lower-case alphabets
* Verify name with max and min alphabets defined by developer (for ex: 50 characters it will accept we should test with (0,1,2,10,20,30,40, 49,50,51)
* Verify by giving special characters, numbers, Doller symbol, and combination of number with alphabets, combination of special characters with alphabets and other combinations etc.

Referral code text box:

* Verify first able to write something in Referral code text box
* Verify Able to write Referral code with both upper- and lower-case alphabets
* Verify Referral code with max and min alphabets defined by developer (for ex: 50 characters it will accept we should test with (0,1,2,10,20,30,40, 49,50,51)
* Verify by giving special characters, numbers, Doller symbol, and combination of number with alphabets, combination of special characters with alphabets and other combinations etc.

**Integration testing b/w Email and name:**

1. Verify by giving wrong credentials either name or email and click on signup button. It should throw error.

**Ad-hoc testing**

1. Verify by leaving empty on both name and email text box. Click on sign up button it should throw error

**End to End testing or System testing:**

1. From begging login, able to add recipient, verify in recipient list and logout.

**Add recipient test case:**

**Functional testing:**

**Recipient name and nick name:**

Positive scenarios:

* Verify first able to write something in name text box
* Verify Able to write name with both upper- and lower-case alphabets
* Verify name with max and min alphabets defined by developer (for ex: 50 alphabets it will accept we should test with (0,1,2,10,20,30,40, 49,50,51)

Negative scenario:

* Verify It should not accept special characters, numbers, Doller symbol, and combination of number with alphabets, combination of special characters with alphabets and other combinations etc.

**Bank account number:**

Positive scenarios:

* Verify bank account number text box with numbers
* Verify name with max and min numbers defined by developer (for ex: 15 numbers it will accept we should test with (0,1,2,10,20,30,40, 49,50,51).

Negative scenarios:

* Verify it should not accept alphabets and special characters and combination of both

**IFSC Code:**

Positive Test Scenarios

1. Valid IFSC Code

* Description: Verify that a valid IFSC code is accepted.
* Steps:
  1. Enter a valid IFSC code (e.g., SBIN0001234).
  2. Submit the form.
* Expected Result: The system should accept the IFSC code and proceed without errors.

2. Valid IFSC Code with Leading/Trailing Spaces

* Description: Verify that the system trims leading and trailing spaces from the IFSC code.
* Steps:
  1. Enter a valid IFSC code with leading/trailing spaces (e.g., SBIN0001234).
  2. Submit the form.
* Expected Result: The system should trim the spaces and accept the IFSC code.

3. Valid IFSC Code in Lowercase

* Description: Verify that the system accepts a valid IFSC code in lowercase.
* Steps:
  1. Enter a valid IFSC code in lowercase (e.g., sbin0001234).
  2. Submit the form.
* Expected Result: The system should convert it to uppercase and accept the IFSC code.

4. Valid IFSC Code with Special Characters

* Description: Verify that the system rejects IFSC codes with special characters.
* Steps:
  1. Enter an IFSC code with special characters (e.g., SBIN@001234).
  2. Submit the form.
* Expected Result: The system should reject the IFSC code and display an appropriate error message.

Negative Test Scenarios

1. Invalid IFSC Code

* Description: Verify that an invalid IFSC code is rejected.
* Steps:
  1. Enter an invalid IFSC code (e.g., SBIN123456).
  2. Submit the form.
* Expected Result: The system should reject the IFSC code and display an error message (e.g., "Invalid IFSC code").

2. IFSC Code with Incorrect Length

* Description: Verify that the system rejects IFSC codes with incorrect lengths.
* Steps:
  1. Enter an IFSC code with fewer than 11 characters (e.g., SBIN1234).
  2. Enter an IFSC code with more than 11 characters (e.g., SBIN000123456).
  3. Submit the form.
* Expected Result: The system should reject the IFSC code and display an error message (e.g., "IFSC code must be 11 characters long").

3. IFSC Code with Invalid Characters

* Description: Verify that the system rejects IFSC codes with invalid characters (e.g., special characters or lowercase letters).
* Steps:
  1. Enter an IFSC code with invalid characters (e.g., SBIN@001234 or sbin0001234).
  2. Submit the form.
* Expected Result: The system should reject the IFSC code and display an error message (e.g., "IFSC code must contain only uppercase letters and numbers").

4. Empty IFSC Code Field

* Description: Verify that the system rejects an empty IFSC code field.
* Steps:
  1. Leave the IFSC code field empty.
  2. Submit the form.
* Expected Result: The system should display an error message (e.g., "IFSC code is required").

5. IFSC Code for Non-Existent Bank/Branch

* Description: Verify that the system rejects IFSC codes for non-existent banks or branches.
* Steps:
  1. Enter a valid format IFSC code for a non-existent bank/branch (e.g., ABCD0123456).
  2. Submit the form.
* Expected Result: The system should reject the IFSC code and display an error message (e.g., "Bank/Branch not found").

6. IFSC Code with Spaces in Between

* Description: Verify that the system rejects IFSC codes with spaces in between characters.
* Steps:
  1. Enter an IFSC code with spaces in between (e.g., SBIN 001234).
  2. Submit the form.
* Expected Result: The system should reject the IFSC code and display an error message (e.g., "IFSC code cannot contain spaces").

Test Data for IFSC Code Validation

Valid IFSC Codes:

* SBIN0001234 (State Bank of India)
* HDFC0000123 (HDFC Bank)
* ICIC0000001 (ICICI Bank)

Invalid IFSC Codes:

* SBIN123456 (Invalid branch code)
* ABCD0123456 (Non-existent bank)
* SBIN@001234 (Special character)
* SBIN 001234 (Space in between)
* sbin0001234 (Lowercase letters)

**Country:**

Positive scenario:

* verify by click on India button.

**Integration testing b/w Email and name:**

* 1. Verify by giving wrong credentials either name or nick name or bank account number or ifsc code and click on submit button. It should throw error.

**Ad-hoc testing**

* 1. Verify by leaving empty on all text boxes. Click on sign up button it should throw error

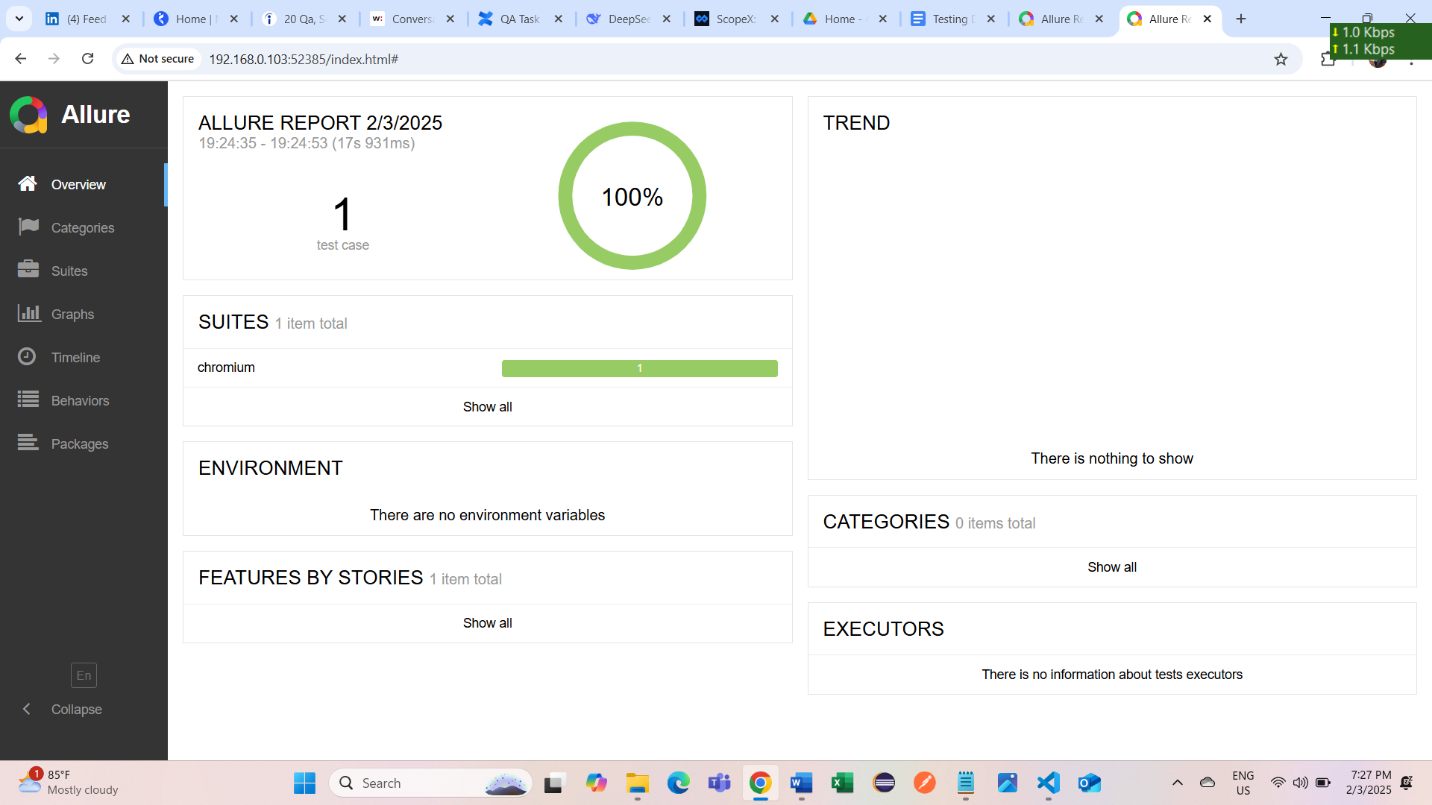
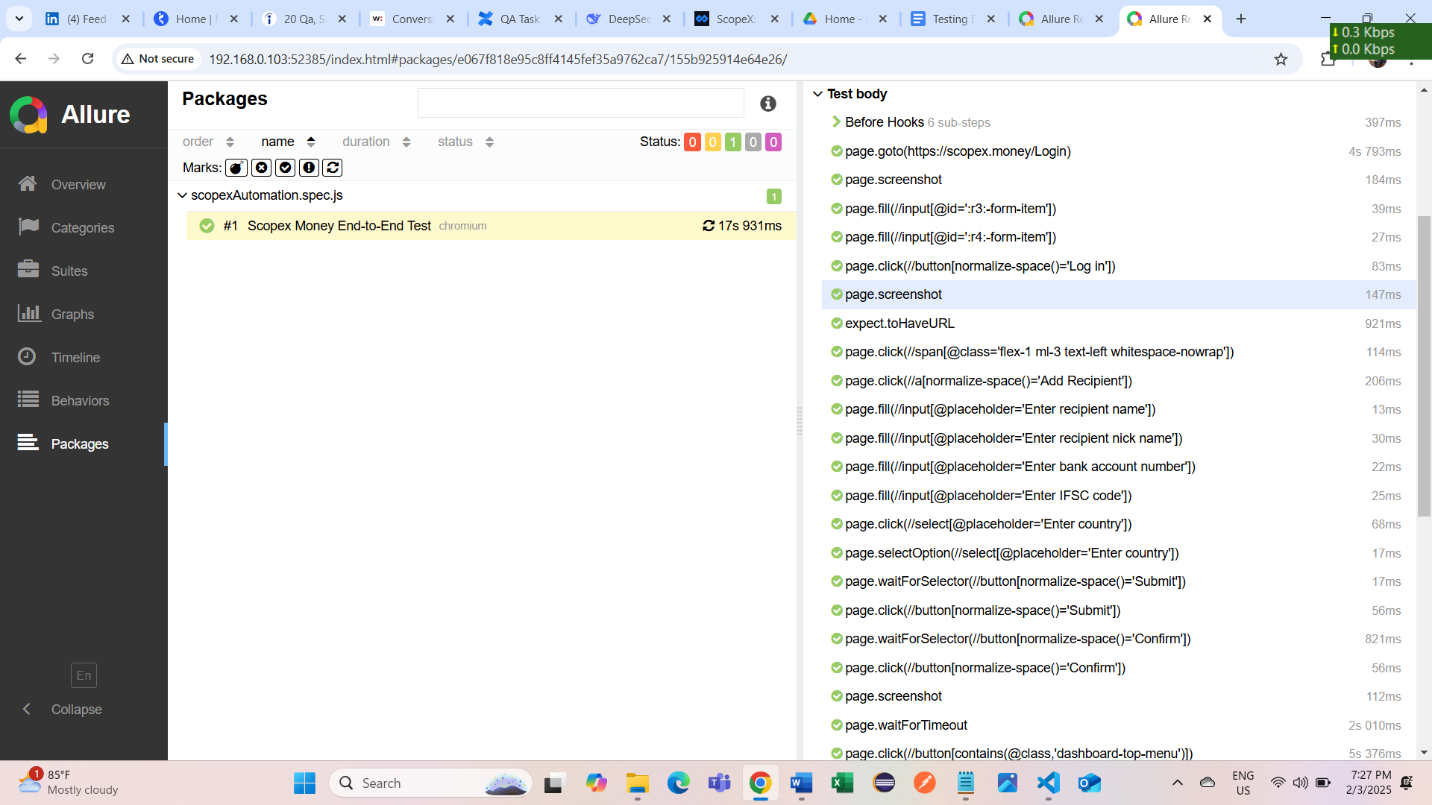
**End to End testing or System testing:**

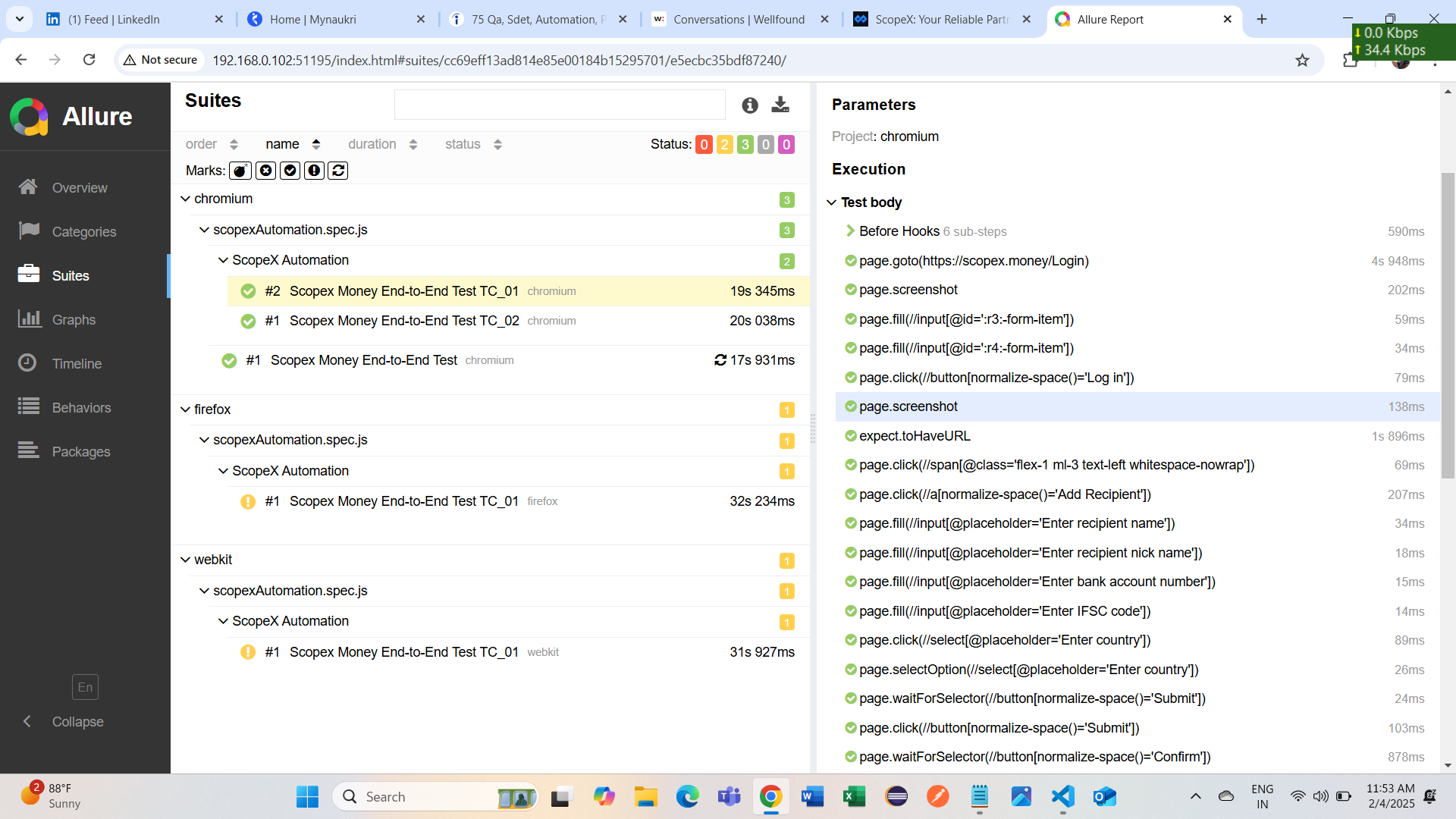
* 1. From begging login, able to add recipient, verify in recipient list and logout.

**Password reset button scenarios:**

* Verify able to click password rest button.
* Verify by giving both valid and invalid email in email address text box.
* Able to click reset password button.

**Automation report:**

****



To resolve the issue of test case failures in Firefox and WebKit browsers due to the presence of duplicate account numbers in the database, I need to introduce additional test data. Currently, I have only two account numbers, which are already present in the database, causing conflicts during the tests.

To overcome this, I will provide unique test data for each browser to avoid duplication. For Chrome, I will add two new recipient details that do not exist in the database. Similarly, for Firefox, I will introduce two different recipients, and for WebKit, I will do the same by adding two distinct recipient details. This way, each browser will have different test data that does not conflict with the database, ensuring that no errors are thrown.

Additionally, the same code can be reused for mobile view (device testing) by making modifications to the Playwright configuration file to accommodate the device-specific settings. This ensures that the tests can be executed across different browsers and devices without issues related to duplicate data in the database.