

Para Bank E2E Automation Framework – Delivery Document

Project Name: Para Bank End-to-End Automation Framework

Author: Shyamasree Sett

Date: 12/09/2025

URL Tested: <https://parabank.parasoft.com/>

Tested Environment: Chromium, Firefox, WebKit (via Playwright)

Framework: Playwright + TypeScript

Para Bank E2E Automation Framework – Delivery Document	1
1. Project Overview	1
2. Project RISKS	1
3. Key Features & Improvements	2
4. Deliverables	3
5. Test Scenarios Covered in E2E test	3
UI Test Scenarios	3
API Test Scenarios	3
Additional granular tests:	4
6. Bug discovered while running test Automation:	4
7. Test Execution Steps	
A . Running Tests Locally (Without Jenkins)	5
B. Running Tests via Jenkins(Local)	5
8. Project Structure: Page Object Model (POM)	7

1. Project Overview

Para Bank is a realistic online banking application that allows users to manage fund transactions.

The purpose of this assignment was to develop a **robust E2E automation framework** covering both **UI and API scenarios** using Playwright, including validation of transactions, account balances, and navigation workflows.

Scope of the framework includes:

1. End to end test covering all the steps provided in the assignment
2. Granular tests for easy testing and debug
3. Both UI and API steps are covered
4. Jenkins integration via pipeline jenkinsfile

2. Project RISKS

1. The ParaBank demo application is known for its instability, including frequent downtime and session data loss, which can disrupt automated testing workflows.
2. Due to stringent time constraints, there is insufficient opportunity to develop an extensive test suite, potentially leading to inadequate test coverage and increased risk of undetected defects in the application.
3. Due to high traffic, random dynamic test data generation which is unique is difficult even using Faker, UUID and date time stamp.

3. Key Features & Improvements

1. Single End to end test encompassing all steps for transactions and balance checks.
2. Granular tests to highlight reusability and modularity of suite with 1 intentionally failed test.
3. Reusable Page Objects (POM): All UI interactions are modularized for maintainability. A Base Test fixture under the `base` folder provides common setup for tests and initializes page objects, ensuring consistency across the suite.
4. Utility Helpers: Balance conversion, dynamic payee generation, API helpers, and random data generation using Faker. Random user registration with timestamp-concatenated usernames ensures uniqueness in each test execution.
5. Setup Playwright config file for cross browser testing.
6. Assertions & Error Handling: Assertions added at every step ensure correct behavior. Each UI action is wrapped in a try-catch block to capture errors for future debugging.
7. Dynamic Data & POJOs: Created a POJO for the registration form page. Data-driven approach with dynamic/randomized values ensures realistic testing scenarios.
8. Element Location Strategies: Used different types of element locators: by locator, by role, and by text, enhancing test robustness.
9. HTML Report Enhancements: Enhanced reports include screenshots, trace files, and logical step grouping for easier debugging and analysis.
10. CI/CD Ready: Playwright configuration supports integration with Jenkins (added `jenkins.bat` and `jenkinsfile` to repo), enabling automated execution in pipelines.
11. `.env` file to store secrets
12. Test Case wrapped in test steps for better test report readability

4. Deliverables

1. **E2E Test Code:** [src/tests/end2end.spec.ts](#)
2. **Granular test codes:** [src/tests/mainTests](#)
3. **Page Objects:** [pages/](#)
4. **Utility Functions:** [utils/](#)
5. **Constants & Enums:** [resources/](#)
6. **Playwright Config:** [playwright.config.ts](#)
7. **Environment Variables:** [.env](#)
8. **Jenkins setup:** [jenkinsfile](#), [jenkin.bat](#)
9. **HTML Report:** [playwright-report/](#) (generated after test execution)
10. **GitHub Repository Link:** <https://github.com/ShyamasreeSett/ParaBankTest>
11. **Demo artifacts:** <https://github.com/ShyamasreeSett/ParaBankTest/tree/main/Demo>

5. Test Scenarios Covered in E2E test

UI Test Scenarios

Steps	Scenarios	Is implemented?
1	Navigate to Para bank application.	Implemented
2	Create a new user from user registration page (Ensure username is generated randomly and it is unique in every test execution).	Implemented
3	Login to the application with the user created in step 2.	Implemented
4	Verify if the Global navigation menu in home page is working as expected.	Implemented
5	Create a Savings account from “Open New Account Page” and capture the account number.	Implemented
6	Validate if Accounts overview page is displaying the balance details as expected.	Implemented
7	Transfer funds from account created in step 5 to another account.	Implemented
8	Pay the bill with account created in step 5.	Implemented
9	Add necessary assertions at each test step whenever it is needed.	Implemented

API Test Scenarios

1	Search the transactions using “Find transactions” API call by amount for the payment transactions made in Step 8.	Implemented
2	Validate the details displayed in Json response.	Implemented

Additional granular tests:

Module	Test summary	Test ID	Status
Login	test successful login	TC-001	PASS
	test unsuccessful login with null password	TC-002	PASS
	test unsuccessful login with null username and password	TC-003	PASS
Global Navigation	test home page button after successful login	TC-004	PASS
	test ATM services availability	TC-013	FAIL
Registration	test successful registration	TC-005	PASS
Account overview	test account number is created upon new account opening	TC-006	PASS
	test account balance upon new account opening	TC-007	PASS
Open new account	test successful account opening	TC-008	PASS
Transfer funds	test transfer funds success	TC-009	PASS
	test end to end Fund Transfer Scenario	TC-010	PASS
Bill Payments	test bill payment success	TC-011	PASS
	test end to end Fund Transfer Scenario	TC-012	PASS

6. Bug discovered while running test Automation:

Bug Summary:

ATM services and Online services pages do not load successfully

Steps to reproduce:

1. Login to Parabank portal
2. Navigate to home page
3. Verify list of ATM services and Online services are displayed
4. Click on any of the links (ex Withdraw Funds)
5. Verify the respective page is displayed

Expected result:

Corresponding service page should be displayed

Actual Result:

A broken html document is displayed



7. Test Execution Steps

A . Running Tests Locally (Without Jenkins)

Step 1: Repository cloned locally: <https://github.com/ShyamasreeSett/ParaBankTest>

Step 2: Open a Terminal / Command Prompt:

Step 3: Navigate to the root of the project directory.

Step 4: Install Dependencies:

```
npm ci
```

Step 5: Install Playwright Browsers:

```
npx playwright install --with-deps
```

Step 6: Run E2E Tests:

```
npm run test:e2e
```

Step 7: Run granular Tests:

```
npm run test:main
```

Step 8: View Reports: After tests finish, open the HTML report in <playwright-report/index.html> to see results.

B. Running Tests via Jenkins(Local)

Prerequisites

- Jenkins installed and running locally.
- Node.js installed or configured via Jenkins NodeJS tool.
- Repository cloned locally <https://github.com/ShyamasreeSett/ParaBankTest>

Step 1: Open Jenkins

- Open Jenkins in a browser (<http://localhost:8080> by default).

Step 2: Create a New Pipeline Job

Step 3: Configure the Pipeline

1. In the job configuration, scroll to “**Pipeline**” section.
2. Under **Definition**, select “**Pipeline script from SCM**”.
3. Choose your SCM type (e.g., Git) and enter:
 - Repository URL: Path to the local repo or remote Git URL:
<https://github.com/ShyamasreeSett/ParaBankTest>
4. Save the configuration.

Step 4: Configure NodeJS Tool (if required)

1. Go to **Manage Jenkins** → **Global Tool Configuration**.
2. Under **NodeJS**, configure a NodeJS installation:
 - Name: **NodeJS-16** (matching Jenkinsfile)
 - Install automatically or point to existing NodeJS on local machine.
3. Save.

Step 5: Build the Job

1. Click **“Build with Parameters”** and select e2e to run end to end test or main to select running granular tests.
2. Jenkins will execute the pipeline:
 - Clone the repo.
 - Run **jenkins.bat**:
 - Install dependencies: **npm ci**
 - Install Playwright browsers: **npx playwright install --with-deps**
 - Run E2E tests: **npm run test:e2e**
 - Generate HTML reports in **playwright-report/**.

Step 6: View Test Reports

After the build completes:

- Open the **specific build** in Jenkins.
- Click **“Artifacts”** in the left-hand menu.
- Open **playwright-report/index.html** to view the HTML report.

Sample test Report:

Q	All 1	✓ Passed 1	Failed 0	Flaky 0	Skipped 0
@e2e Verify end to end account fund transfer scenario					
tests/end2end.spec.ts:21					
chromium e2e					
✓ Run					
Test Steps					
> ✓ Before Hooks					
> ✓ Step 2: Create a new user from user registration page. — tests/end2end.spec.ts:24					
> ✓ Step 3: Login to the application with the user created in step 2. — tests/end2end.spec.ts:29					
> ✓ Step 4: Verify if the Global navigation menu in home page is working as expected. — tests/end2end.spec.ts:37					
> ✓ Step 5: Create a Savings account from Open New Account Page. — tests/end2end.spec.ts:50					
> ✓ Step 6: Validate if Accounts overview page is displaying the balance details as expected. — tests/end2end.spec.ts:59					
> ✓ Step 7: Transfer funds from account created in step 5 to another account. — tests/end2end.spec.ts:71					
> ✓ Step 8: Pay the bill with account created in step 5. — tests/end2end.spec.ts:92					
> ✓ Validate API response: Search the transactions using Find transactions API call by amount for the bill payment — tests/end2end.spec.ts:1...399ms					
> ✓ After Hooks					

8. Project Structure: Page Object Model (POM)

```
para-bank-e2e/
|
├─ src/
|   └─ end2end.spec.ts          # Main E2E test for UI + API
|   └─ MainTests
|       └─ login.spec.ts        #Multiple test folders
|       └─ billPayment.spec.ts
├─ pages/                       # Page Objects for modular UI interactions
|   ├─ loginPage.ts
|   ├─ registerFormPage.ts
|   ├─ homePage.ts
|   ├─ navigationPanel.ts
|   ├─ accountsOverviewPage.ts
|   ├─ openNewAccountPage.ts
|   ├─ transferFundsPage.ts
|   ├─ billPaymentFormPage.ts
|   └─ billPayPage.ts
|
├─ utils/                       # Utility/helper functions
|   ├─ apiHelper.ts             # API calls (getTransactions)
|   ├─ generateUser.ts          # create dynamic unique user
|   └─ generateBeneficiary.ts    # Generate dynamic payee data
|
├─ resources/                   # Constants & Enums
|   ├─ accountType.enum.ts
|   └─ constants.ts
|
├─ playwright.config.ts         # Playwright configuration
├─ package.json
├─ tsconfig.json
├─ jenkinsfile                  #Jenkins configuration
├─ Demo                         #Folder containing demo artifacts
|   ├─ Recording.mp4
|   └─ Test Report
└─ .env                         # Environment variables (BASE_URL)
```
