**Solution Approach for** **Data-Driven Testing in Cypress Using POM with CICD integration using Git-Action**

**Data-Driven Testing in Cypress Using POM**

To automate the given web application workflow using Cypress, we'll follow a data-driven testing approach using fixtures and implement the Page Object Model (POM) to maintain clean and scalable test code.

**1. Steps to Implement**

1. **Set Up Cypress Project** (Skip if already set up)

npm init -y

npm install cypress csv-parse --save-dev

* + csv-parse helps read CSV files.

1. **Create Test Data in CSV (cypress/fixtures/testData.csv)**

dob,postalCode,app\_id,loanAmount,tenure,sanctionedAmount,selectedTenure,aprInterest,interest,principalAmount

1990-05-15,12345,app123,50000,3,45000,36,12.5%,5000,50000

1. **Modify the Page Object Model (POM)**
   * Add takeScreenshot() before submission on each page.
   * Read test data dynamically from CSV.

**Folder structure**

cypress-project/

│── cypress/

│ ├── fixtures/

│ │ ├── testData.csv # CSV file containing test data

│ ├── integration/

│ │ ├── loanTest.spec.js # Main test (reads data from CSV)

│ ├── pages/ # Page Object Model (POM) files

│ │ ├── LoginPage.js # Login Page

│ │ ├── LoanPage.js # Loan Page

│ │ ├── QuoteInitiatePage.js # Quote Initiate Page

│ │ ├── CustomerFinancialPage.js # Customer Financial Details Page

│ │ ├── SignPage.js # Sign Page

│ ├── screenshots/ # Cypress captures screenshots here

│ ├── videos/ # Cypress records test execution videos

│ ├── support/

│ │ ├── commands.js # Custom Cypress commands (if needed)

│ │ ├── index.js # Cypress support configuration

│── node\_modules/ # Installed dependencies (auto-generated)

│── cypress.config.js # Cypress configuration settings

│── package.json # Project dependencies and scripts

│── package-lock.json # Dependency lock file

│── README.md # Documentation for the project

**2. Rough Implementation Code**

**Login Page (cypress/pages/LoginPage.js)**

class LoginPage {

visit(baseURL, appId) {

cy.visit(`${baseURL}/${appId}`);

}

enterDOB(dob) {

cy.get('#dob').type(dob);

}

enterPostalCode(postalCode) {

cy.get('#postalCode').type(postalCode);

}

takeScreenshot() {

cy.screenshot('LoginPage');

}

submit() {

this.takeScreenshot();

cy.get('#submitBtn').click();

}

}

export default LoginPage;

**Loan Page (cypress/pages/LoanPage.js)**

class LoanPage {

enterLoanAmount(amount) {

cy.get('#loanAmount').type(amount);

}

selectTenure(tenure) {

cy.get(`#tenure option[value="${tenure}"]`).click();

}

validateLoanDetails(expected) {

cy.get('.loan-details-table').within(() => {

cy.get('td').eq(1).should('contain', expected.sanctionedAmount);

cy.get('td').eq(2).should('contain', expected.selectedTenure);

cy.get('td').eq(3).should('contain', expected.aprInterest);

cy.get('td').eq(4).should('contain', expected.interest);

cy.get('td').eq(5).should('contain', expected.principalAmount);

});

}

takeScreenshot() {

cy.screenshot('LoanPage');

}

submit() {

this.takeScreenshot();

cy.get('#loanSubmitBtn').click();

}

}

export default LoanPage;

**Quote Initiate Page (cypress/pages/QuoteInitiatePage.js)**

class QuoteInitiatePage {

selectAddress() {

cy.get('#addressDropdown').select(1);

}

takeScreenshot() {

cy.screenshot('QuoteInitiatePage');

}

submit() {

this.takeScreenshot();

cy.get('#quoteSubmitBtn').click();

}

}

export default QuoteInitiatePage;

**Customer Financial Page (cypress/pages/CustomerFinancialPage.js)**

class CustomerFinancialPage {

validateLoanTable(expected) {

cy.get('.financial-table').within(() => {

cy.get('td').eq(1).should('contain', expected.sanctionedAmount);

cy.get('td').eq(2).should('contain', expected.selectedTenure);

cy.get('td').eq(3).should('contain', expected.aprInterest);

cy.get('td').eq(4).should('contain', expected.interest);

cy.get('td').eq(5).should('contain', expected.principalAmount);

});

}

takeScreenshot() {

cy.screenshot('CustomerFinancialPage');

}

submit() {

this.takeScreenshot();

cy.get('#financialSubmitBtn').click();

}

}

export default CustomerFinancialPage;

**Sign Page (cypress/pages/SignPage.js)**

class SignPage {

validateLoanTable(expected) {

cy.get('.sign-table').within(() => {

cy.get('td').eq(1).should('contain', expected.sanctionedAmount);

cy.get('td').eq(2).should('contain', expected.selectedTenure);

cy.get('td').eq(3).should('contain', expected.aprInterest);

cy.get('td').eq(4).should('contain', expected.interest);

cy.get('td').eq(5).should('contain', expected.principalAmount);

});

}

sign() {

cy.get('#signatureCanvas').click();

}

takeScreenshot() {

cy.screenshot('SignPage');

}

submit() {

this.takeScreenshot();

cy.get('#signSubmitBtn').click();

}

}

export default SignPage;

**3. Test Case Using Data-Driven Approach (cypress/integration/loanTest.spec.js)**

import LoginPage from '../pages/LoginPage';

import LoanPage from '../pages/LoanPage';

import QuoteInitiatePage from '../pages/QuoteInitiatePage';

import CustomerFinancialPage from '../pages/CustomerFinancialPage';

import SignPage from '../pages/SignPage';

import fs from 'fs';

import path from 'path';

import { parse } from 'csv-parse/sync';

describe('Loan Process Automation - Data Driven Testing', () => {

const loginPage = new LoginPage();

const loanPage = new LoanPage();

const quoteInitiatePage = new QuoteInitiatePage();

const customerFinancialPage = new CustomerFinancialPage();

const signPage = new SignPage();

let testData = [];

before(() => {

const csvPath = path.join(\_\_dirname, '../fixtures/testData.csv');

const fileContent = fs.readFileSync(csvPath, { encoding: 'utf-8' });

testData = parse(fileContent, { columns: true });

});

testData.forEach((data) => {

it(`should complete loan process for ${data.app\_id}`, function () {

const baseURL = "http://your-webapp-url.com"; // Replace with actual base URL

// Step 1: Login

loginPage.visit(baseURL, data.app\_id);

loginPage.enterDOB(data.dob);

loginPage.enterPostalCode(data.postalCode);

loginPage.submit();

// Step 2: Loan Page

loanPage.enterLoanAmount(data.loanAmount);

loanPage.selectTenure(data.tenure);

loanPage.submit();

loanPage.validateLoanDetails(data);

// Step 3: Quote Initiate

quoteInitiatePage.selectAddress();

quoteInitiatePage.submit();

// Step 4: Customer Financial Page

customerFinancialPage.validateLoanTable(data);

customerFinancialPage.submit();

// Step 5: Sign Page

signPage.validateLoanTable(data);

signPage.sign();

signPage.submit();

// Test Passes if all validations match expected data

});

});

});

**4. Cypress Configuration (cypress.config.js)**

module.exports = {

e2e: {

baseUrl: "http://your-webapp-url.com",

defaultCommandTimeout: 10000

}

};

**Key Features in This Approach**

1. **Uses CSV for Data-Driven Testing**: Reads test cases dynamically from a CSV file.
2. **Encapsulates URL with app\_id**: The visit method constructs <webbase URL>/app\_id dynamically.
3. **Takes Screenshots Before Submission**: Helps in debugging by capturing UI state.
4. **Page Object Model (POM)**: Ensures modular, reusable, and maintainable test cases.

**5. Next Steps**

* Run the test using:

npx cypress run

* Add multiple test cases in testData.csv for extensive coverage.
* Use **Cypress Dashboard** to analyze screenshots and test reports.

This ensures a **robust Cypress data-driven automation** for **loan processing validation**! 🚀 Let me know if you need refinements.

**Deploy Cypress Code to GitHub and Set Up CI/CD with GitHub Actions**

This guide will walk you through:  
1️⃣ **Pushing the Cypress automation code to GitHub**  
2️⃣ **Setting up GitHub Actions for CI/CD execution**

**1️⃣ Push Cypress Code to GitHub**

**Step 1: Initialize Git**

Inside your Cypress project folder, run the following commands:

git init # Initialize Git repository

git add . # Add all files to staging

git commit -m "Initial commit" # Commit changes

**Step 2: Create a GitHub Repository**

1. Go to [GitHub](https://github.com/) and create a new repository (e.g., cypress-automation).
2. Copy the remote repository URL (e.g., https://github.com/yourusername/cypress-automation.git).

**Step 3: Push Code to GitHub**

Run the following commands to push the code:

git remote add origin https://github.com/yourusername/cypress-automation.git

git branch -M main

git push -u origin main

Your Cypress automation project is now uploaded to GitHub. 🎉

**2️⃣ Set Up CI/CD with GitHub Actions**

To run Cypress tests automatically in CI/CD, set up **GitHub Actions**.

**Step 1: Create GitHub Actions Workflow**

Create a folder .github/workflows in your project and add a file named cypress-ci.yml:

name: Cypress Tests

on:

push:

branches: [main] # Runs on every push to the main branch

pull\_request:

branches: [main] # Runs on every PR to main

jobs:

test:

runs-on: ubuntu-latest # Runs in a Linux environment

steps:

- name: Checkout Repository

uses: actions/checkout@v4 # Pulls the code from GitHub

- name: Set Up Node.js

uses: actions/setup-node@v4

with:

node-version: 18 # Set Node.js version

- name: Install Dependencies

run: npm install # Installs Cypress and dependencies

- name: Run Cypress Tests

run: npx cypress run # Executes tests in headless mode

- name: Upload Cypress Screenshots (if tests fail)

if: failure()

uses: actions/upload-artifact@v4

with:

name: cypress-screenshots

path: cypress/screenshots # Stores failed test screenshots

- name: Upload Cypress Videos (for debugging)

uses: actions/upload-artifact@v4

with:

name: cypress-videos

path: cypress/videos # Stores recorded test execution videos

**3️⃣ Verify GitHub Actions Execution**

**Step 1: Commit & Push GitHub Actions Workflow**

After adding .github/workflows/cypress-ci.yml, push it to GitHub:

git add .github/workflows/cypress-ci.yml

git commit -m "Added GitHub Actions CI/CD for Cypress"

git push origin main

**Step 2: Check Workflow Execution**

1. Go to **GitHub Repository → Actions Tab**
2. See the Cypress test execution running under **"Cypress Tests"**
3. If tests pass ✅, your automation is working fine.
4. If tests fail ❌, review logs and screenshots for debugging.

**4️⃣ Bonus: Run Tests on Every Feature Branch (Optional)**

Modify .github/workflows/cypress-ci.yml to trigger tests on any branch:

on: [push, pull\_request]

This will run Cypress tests for every push or PR in any branch.

**5️⃣ Summary of GitHub Actions Workflow**

✅ Runs Cypress tests automatically on **push** or **pull request**  
✅ Uploads **screenshots** if tests fail  
✅ Stores **videos** for debugging  
✅ Supports **parallel test execution** (Advanced feature)

🚀 **You now have a fully automated CI/CD pipeline for Cypress!** Let me know if you need improvements. 🎯