Cypress Automation Testing Capstone Project (Using Page Object Model)

Project Title

End-to-End Cypress Automation Framework for E-Commerce Application using POM

Project Overview

This capstone project evaluates students' ability to design and implement a scalable Cypress automation framework using the Page Object Model (POM), automate real user journeys, document their work professionally, integrate automated tests into a CI/CD pipeline using GitHub Actions, and communicate their work publicly as an automation engineer.

Students will automate key workflows on a production-like e-commerce application and reflect on their learning by writing a professional LinkedIn post summarizing what was automated.

🔗 **Application Under Test (AUT)**
https://shop.qaautomationlabs.com/index.php

Objectives

By completing this capstone project, students will demonstrate the ability to:

- Build a Cypress automation framework using Page Object Model
- Write clean, reusable, and maintainable automated tests
- Implement positive and negative test scenarios
- Automate real-world e-commerce user flows
- Document automation work clearly using `README.md`
- Integrate Cypress tests into GitHub Actions (CI/CD)
- Communicate technical work professionally via a LinkedIn post

Scope of Automation

Mandatory Module

✅ Login Automation

- Positive and negative login scenarios

Plus Any Two (2) Modules from the List Below

Students must automate two additional modules, including positive and negative scenarios for each.

Automation Modules

1. Homepage Navigation

Automate validation of:

- Visibility and accessibility of menu items:
    - Home/About/Courses/Events/Shop/Testing/Blog/Contact
- Shop dropdown behavior
- Correct navigation when menu items are clicked

Negative Scenarios

- Missing menu items/Incorrect redirection/Non-functional dropdown options

2. Shopping Categories

Automate validation of:

- Navigation to:
    - Mens Wear/Womens Wear/Kids Wear/Electronics
- Correct product listing per category

- Visibility of: Product names/Pricing/Discounts/offers (if applicable)

Negative Scenarios

- Empty category results/Incorrect pricing/Product-category mismatch

3. Add to Cart Functionality

Automate validation of:

Adding products to cart /Cart icon item count updates/ Correct product details in cart / Cart persistence across navigation (if applicable)

Negative Scenarios

- Cart not updating/Incorrect quantity or price/Invalid add-to-cart actions

4. Form Filling

Automate validation of:

- Required checkout or submission form fields
- Input validation for:Name/Email format/Address/Mandatory fields/Error messages for invalid or missing inputs

Negative Scenarios

- Empty form submission/Invalid email formats/Missing required fields

---

5. End-to-End Checkout Flow

Automate a complete user journey:

- Select product >>Add to cart >>Proceed to checkout >>Fill required details >>Place order >>Verify confirmation or success message

Negative Scenarios :Checkout with empty cart/ Invalid or incomplete form submission
Order failure handling (if applicable)

Framework Architecture (Page Object Model – Mandatory)

Required Project Structure

- `cypress/`
- `  ├─ e2e/`
- `  │    ├─ login.cy.js`
- `  │    ├─ navigation.cy.js`
- `  │    ├─ cart.cy.js`
- `  │    └─ checkout.cy.js`
- `  ├─ pages/`
- `  │    ├─ LoginPage.js`
- `  │    ├─ HomePage.js`
- `  │    ├─ ShopPage.js`
- `  │    ├─ CartPage.js`
- `  │    └─ CheckoutPage.js`
- `  ├─ fixtures/`
- `  │    └─ testData.json`
- `  ├─ support/`
- `  │    ├─ commands.js`
- `  │    └─ e2e.js`
- `.github/`
- `  └─ workflows/`
- `       └─ cypress.yml`
- `README.md`
- `package.json`

POM Guidelines

- Page files contain:
    - Locators ,Page-level actions
- Test files contain: Test logic and assertions only ,No selectors inside test files, Reusable page methods across multiple tests

README Documentation (Mandatory)

Each project must include a professional `README.md` covering:

- Project overview
- Application under test
- Tools and technologies used
- Framework architecture (POM explanation)
- Installation and setup instructions
- How to run tests:
    - Cypress GUI mode
    - Headless mode
- Test coverage summary
- CI/CD integration explanation
- GitHub Actions workflow overview

CI/CD Integration (GitHub Actions)

Requirements

- Cypress tests must run automatically via GitHub Actions
- Workflow triggers:
    - On every push
    - On pull requests (optional)
- Tests must execute in headless mode
- Workflow file location:

Deliverables for Project Submission

1. **Testcase Document indicating the automated scenarios(create an addition column- stating Manual/Automated)**

2. **Cypress Automation Framework**

    - POM-based test suite
    - Login + at least one additional modules
    - Positive and negative scenarios

3. **README.md**

    - Complete, clear, and professional

4. **CI/CD Pipeline**

    - Working GitHub Actions workflow

5. **GitHub Repository**

    - Public and accessible

6. **LinkedIn Post**

    Link submitted as proof Samplepost