

## Assignment – 5 (Graphical User Interface Testing)

Name: Prasad Aggarwal

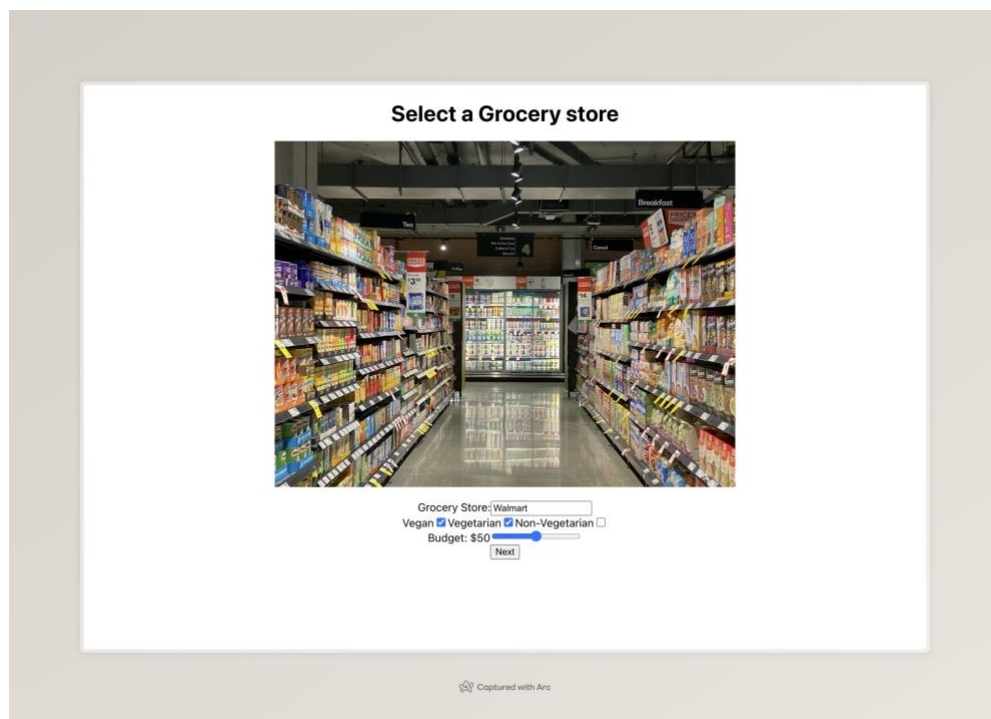
ASURITE ID: paggar10

### Application Version 1:


I have developed an application that helps user plan where they want to buy their groceries from (Walmart, Costco, etc.). Then there is a checkbox where users can select vegan, vegetarian, non-vegetarian options based on their dietary restrictions/preferences. Finally, there is a slider for a budget that the user can set, ranging from 0-100\$.

First page is the GroceryStore page, when the user clicks on next, they are taken to the CartBuilder page. The user builds their shopping cart here, using an input text box which adds the items user is shopping for and adds them to a bulleted list. Then there are two radio buttons to choose between a delivery or a pickup option. From this page the user can navigate to the payment review page.

The FinalReview page is responsible for getting user information through input boxes and form data, selecting the mode of statement delivery, and then finally entering their card details for the order payment.



### Build Your Shopping Cart for Walmart



Add Item:


- Apple
- Tomatoes
- Bananas
- Milk

**Select Mode:**

☐ Delivery ☒ Pickup

Captured with Arc

### Review and Confirm



**Your Cart:**

Satisfaction Rating: 5

**Preferred Method of Statement Delivery:**

☒ Email ☐ Postal Mail

Grocery Store: Walmart

Items:

Mode of Delivery: pickup

**Enter Your Payment details**

Name:  Email:  Phone:

**Payment Information**

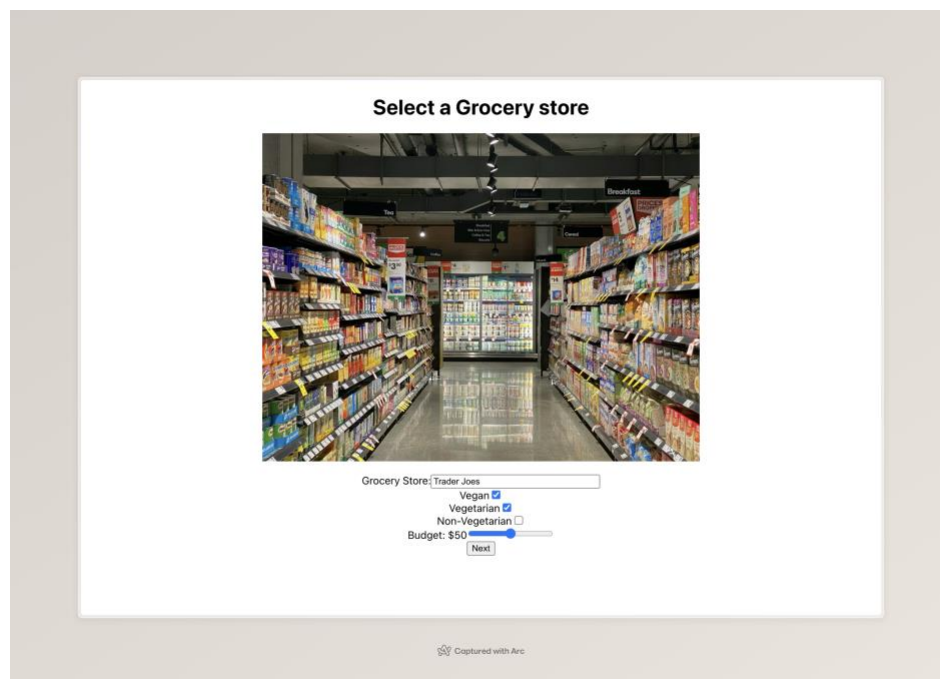
Card Number:  Expiry Date:  CVV:

Captured with Arc

## Application Version 2:

In version 2, I made the following changes:

- Changed the width of the input box for grocery store to 250px.
- Made sure that the checkboxes for dietary preferences are using a flex layout with a flex-direction of column.
- On page 2, I change the flow by shifting the radio buttons for delivery/pickup at the top. This would change the location of the input box for cart items as well.
- Centered the shopping cart list items on the page.
- Gave the back and next buttons more meaningful names.
- On the final page, I added flex-direction column to all HTML divs to change the structural skeleton of the application.
- Changed the flow of the final page by asking for User's personal details before the card details and adding a separate section to check for order details confirmation.



## Build Your Shopping Cart for Trader Joes

Select Mode:

☐ Delivery ☒ Pickup



Add Item:

- Apple
- Tomatoes

 Captured with Arc

### Enter Your Personal details

Name:   
Email:   
Phone:

### Review and Confirm



### Your Order details:

Selected Grocery Store: Trader Joes

Selected Items:

- Apple
- Tomatoes

Mode of Delivery: pickup

Preferred Method of Statement Delivery:

☒ Email ☐ Postal Mail

### Payment Information

Card Number:   
Expiry Date:   
CVV:

Satisfaction Rating: 5

 Captured with Arc

## Tool Used for Testing Purposes: Cypress

- *Real-time Testing Experience:* Cypress executes tests in the same run-loop as the application, providing real-time feedback and interactive visual testing. This unique feature enables developers to see exactly what happens at each step of the test directly in the browser, facilitating quicker debugging and development cycles.
- *Simplified Architecture:* Unlike Selenium, which requires separate drivers and servers (like the Selenium WebDriver and browser-specific drivers), Cypress operates directly within the browser. This allows it to interact with the DOM and the application's JavaScript more naturally and efficiently, reducing the complexity of setup and maintenance.
- *Automatic Waiting:* Cypress intelligently waits for commands and assertions before moving on. This means there is no need to define explicit waits or sleep in the tests, making the tests more readable and less prone to timing errors, a common issue in Selenium tests where explicit waits need to be defined.
- *Built-in Test Spies, Stubs, and Clocks:* Cypress includes support for test spies, stubs, and clocks, which allow developers to easily verify and control the behavior of server responses, timers, or functions. This is highly useful in GUI testing where external systems' responses may affect the UI behavior.
- *Comprehensive Debugging Capabilities:* Cypress provides rich debugging capabilities by automatically taking snapshots at each test step. Developers can hover over a command in the test script to see exactly what happened at each step, making it easier to identify where a test failed and why. Additionally, Cypress integrates seamlessly with Chrome Developer Tools.
- *Community and Ecosystem:* Cypress has a robust and active community that contributes to a growing ecosystem of plugins. This ecosystem extends the capabilities of Cypress, making it adaptable to a wide range of testing needs, from simple unit tests to complex end-to-end tests for web applications.

## Other Tool Considered: Selenium

- *Execution Environment:* While Selenium tests operate through a server communicating over a network with browser-specific drivers, Cypress runs directly inside the browser. This allows Cypress to execute commands much faster and more reliably due to its direct access to browser APIs and DOM.
- *Test Stability and Flakiness:* Selenium tests can sometimes be flaky and require adjustments for timing issues, such as adding waits or retries. Cypress' architecture

reduces flakiness by automatically managing these waits and retries, making tests more stable out of the box.

- *Ease of Setup:* Setting up Cypress is generally simpler because it doesn't rely on external dependencies like browser drivers. Selenium, however, requires setup of the WebDriver and browser-specific drivers, which can be challenging and time-consuming.
- *Language Support:* Selenium supports a variety of programming languages including Java, C#, Python, and Ruby, which can be a significant advantage in diverse development environments. Cypress, on the other hand, is primarily focused on JavaScript, which, while limiting language flexibility, can be an advantage in JavaScript-heavy development teams.

## Test Cases Developed Using Cypress:

```
72 },
73
74 describe("Page Navigation from Cart Builder to Final Review", () => {
75   it("should navigate from the cart builder page to the final payment review page", () => {
76     cy.get("#grocery-store-input").type("Walmart");
77     cy.get("#next-button-destination").click();
78     cy.get("#pickup-radio-button").should("exist");
79     cy.get("#add-item-input").type("Apple");
80     cy.get("#next-button-itinerary").click();
81     cy.get("#postal-radio-button").should("exist");
82   });
83 });
84
85 describe("Existence and Visibility of Grocery Store Input Box", () => {
86   it("should check if the grocery store input box exists and is visible", () => {
87     cy.get("#grocery-store-input").should("exist").and("be.visible");
88   });
89 });
90
```

```
spec.cy.js grocery-planner-v2/... U spec.cy.js grocery-planner-v1/... X
grocery-planner-v1 > cypress > e2e > spec.cy.js > describe("Location of Vegan Check Box") callback > it("should check the exact location of the vegan check box") callback

32
33 describe("Location of Name Input Box in Final Review Page", () => {
34   it("should check the location of the name input box in the final review page", () => {
35     cy.get("#grocery-store-input").type("Trader Joes");
36     cy.get("#next-button-destination").click();
37     cy.get("#pickup-radio-button").should("exist");
38     cy.get("#add-item-input").type("Apple");
39     cy.get("#next-button-itinerary").click();
40     cy.get("#postal-radio-button").should("exist");
41     cy.get("#name-input").then(($el) => {
42       const rect = $el[0].getBoundingClientRect();
43       expect(rect.top).to.be.closeTo(880, 10);
44       expect(rect.left).to.be.closeTo(260, 10);
45     });
46   });
47 });
48
49 describe("Page Navigation from Grocery Store Selection to Cart Builder", () => {
50   it("should navigate from the grocery store selection page to the cart builder page", () => {
51     cy.get("#grocery-store-input").type("Trader Joes");
52     cy.get("#next-button-destination").click();
53     cy.get("#pickup-radio-button").should("exist");
54   });
55 });
56
57 describe("Size of Grocery Store Input Box", () => {
58   it("should check the size of the grocery store input box", () => {
59     cy.get("#grocery-store-input").then($input => {
60       expect($input).to.have.css("width", "139px");
61       expect($input).to.have.css("height", "15.5px");
62     });
63   });
64 });
65
66 describe("Content of Grocery Store Input Box", () => {
67   it("should check the content of the grocery store input box", () => {
68     const expectedInput = "Walmart";
69     cy.get("#grocery-store-input").type(expectedInput);
70     cy.get("#grocery-store-input").should("have.value", expectedInput);
71   });
72 });
73
```

```
spec.cy.js grocery-planner-v2/... U spec.cy.js grocery-planner-v1/... U X
grocery-planner-v1 > cypress > e2e > spec.cy.js > describe("Location of Vegan Check Box") callback > it("should check the exact location of the vegan check box") ca

1 describe("Location of Vegan Check Box", () => {
2   it("should check the exact location of the vegan check box", () => {
3     cy.get("#vegan-checkbox").then(($el) => {
4       const rect = $el[0].getBoundingClientRect();
5       expect(rect.top).to.be.closeTo(625, 10);
6       expect(rect.left).to.be.closeTo(405, 10);
7     });
8   });
9 });
10
11 describe("Existence and Visibility of Vegan Check Box", () => {
12   it("should check if the vegan check box exists and is visible", () => {
13     cy.get("#vegan-checkbox").should("exist").and("be.visible");
14   });
15 });
16
17 describe("Location of Grocery Store Input Box in Grocery Store Selection Page", () => {
18   it("should check the location of the grocery store input box", () => {
19     cy.get("#grocery-store-input").then(($el) => {
20       const rect = $el[0].getBoundingClientRect();
21       expect(rect.top).to.be.closeTo(610, 10);
22       expect(rect.left).to.be.closeTo(480, 10);
23     });
24   });
25 });
26
27 describe("Existence and Visibility of Grocery Store Input Box", () => {
28   it("should check if the grocery store input box exists and is visible", () => {
29     cy.get("#grocery-store-input").should("exist").and("be.visible");
30   });
31 });
32
```

## Explanation/Description of Test Cases:

- *Existence and Visibility of Grocery Store Input Box:* Tests to ensure that the grocery store input box is both present in the DOM and visible to the user.
- *Size of Grocery Store Input Box:* Validates that the grocery store input box has specific dimensions, ensuring it conforms to designed UI specifications.
- *Location of Grocery Store Input Box in Grocery Store Selection Page:* Verifies the exact position of the grocery store input box on the selection page, ensuring it aligns with the intended layout.
- *Existence and Visibility of Vegan Check Box:* Checks that the vegan option check box is available and visible, confirming its presence for user interaction.
- *Location of Vegan Check Box:* Confirms the precise placement of the vegan check box relative to the page, helping ensure consistent user experience across devices.
- *Location of Name Input Box in Final Review Page:* Assesses the location of the name input box on the final review page after navigating through several pages, ensuring it appears where expected.
- *Content of Grocery Store Input Box:* Tests if typing into the grocery store input box updates its content accurately, reflecting user input correctly.
- *Page Navigation from Grocery Store Selection to Cart Builder:* Ensures seamless navigation from the grocery store selection page to the cart builder page upon user action, verifying functional flow.
- *Page Navigation from Cart Builder to Final Review:* Confirms the ability to navigate from the cart builder to the final review page, checking the continuity and functionality of multi-page forms.



## All Test cases passed for Application version 1:

The screenshot displays the Cypress test runner interface. The top bar shows the application name 'grocery-planner-v1' and the URL 'http://localhost:3000/'. The left sidebar contains icons for Specs, Code, Run, and Settings. The main panel shows a list of test cases under the file 'spec.cy.js'. All test cases are marked with a green checkmark, indicating they have passed. The test cases are:

- ✓ Grocery cart planner GUI Tests
  - ✓ Existence and Visibility of Grocery Store Input Box
    - ✓ should check if the grocery store input box exists and is visible
  - ✓ Size of Grocery Store Input Box
    - ✓ should check the size of the grocery store input box
  - ✓ Location of Grocery Store Input Box in Grocery Store Selection Page
    - ✓ should check the location of the grocery store input box
  - ✓ Existence and Visibility of Vegan Check Box
    - ✓ should check if the vegan check box exists and is visible
  - ✓ Location of Vegan Check Box
    - ✓ should check the exact location of the vegan check box
  - ✓ Location of Name Input Box in Final Review Page
    - ✓ should check the location of the name input box in the final review page
  - ✓ Content of Grocery Store Input Box
    - ✓ should check the content of the grocery store input box
  - ✓ Page Navigation from Grocery Store Selection to Cart Builder
    - ✓ should navigate from the grocery store selection page to the cart builder page
  - ✓ Page Navigation from Cart Builder to Final Review
    - ✓ should navigate from the cart builder page to the final payment review page

The right sidebar shows a preview of the application, displaying a grocery store selection page with various fruits and vegetables. A large 'P' is visible on the right side of the preview.

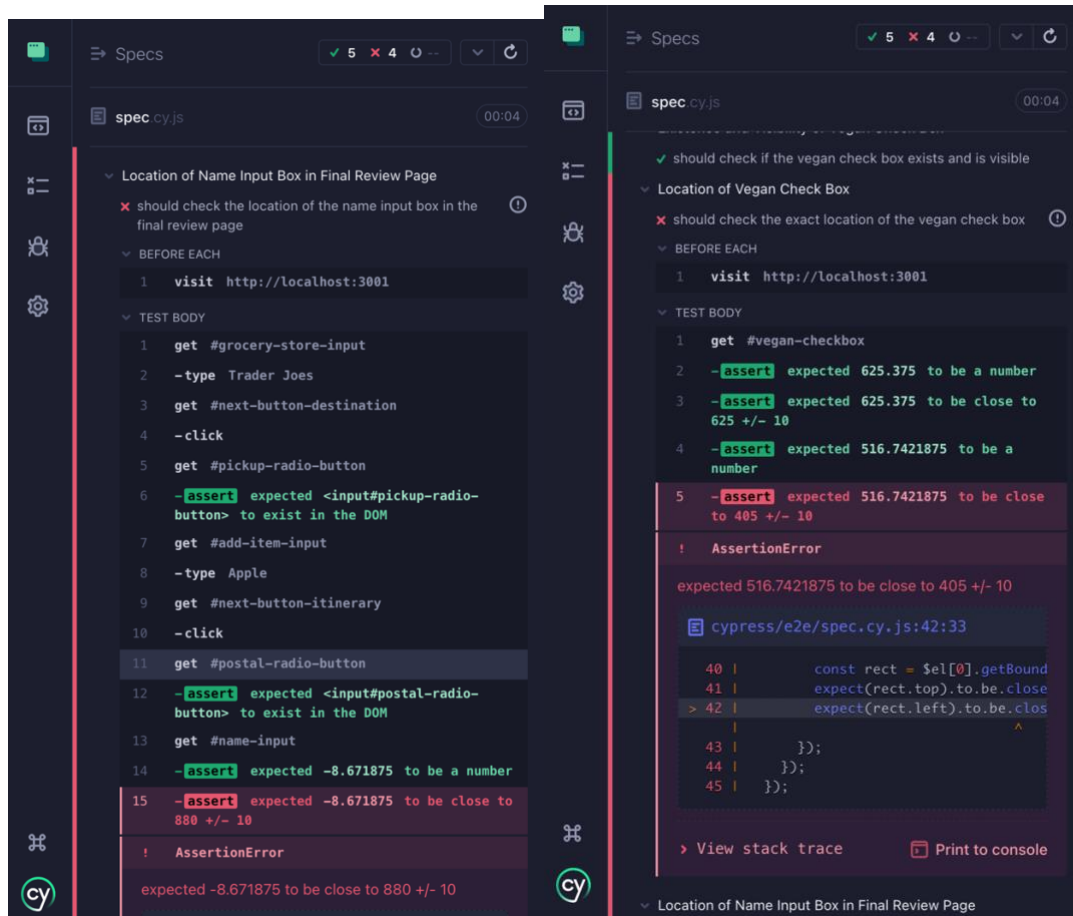
## 4 out of 9 test cases failed for Application version 2:

The screenshot displays the Cypress test runner interface. On the left, a sidebar contains icons for Specs, Source, Command Log, Runner, and Settings. The main panel shows the 'Specs' view for 'spec.cy.js', which has a duration of 00:04. A list of test cases is shown under the heading 'Grocery cart planner GUI Tests'. The test results are summarized as 5 passing (green checkmarks) and 4 failing (red X marks). The failing tests are:

- > Existence and Visibility of Grocery Store Input Box
- > Size of Grocery Store Input Box
- > Location of Grocery Store Input Box in Grocery Store Selection Page
- > Existence and Visibility of Vegan Check Box

The browser window on the right shows the URL <http://localhost:3001/> and a screenshot of a grocery store aisle with shelves stocked with various products.

## Further screenshots/explanations of Failing test cases:



- Location of name input box in final review page failed because we changed the structural skeleton and ordering of elements on the page.
- Location of vegan check box failed because we added flexbox and changed the flex-direction to column.



- Cypress also provides integration with various reporting tools and CI/CD platforms, making it easy to integrate into your development workflow.

#### **Type of coverage:**

- Cypress is primarily focused on end-to-end (E2E) testing, allowing you to test the entire user flow of your application, including the GUI.
- It can be used to test the behavior of your application from the user's perspective, ensuring that all the GUI elements and their interactions work as expected.
- Cypress also supports unit testing and integration testing, but its primary strength lies in E2E testing of the GUI.

#### **Reuse of test cases:**

- Cypress makes it easy to reuse test cases by providing a modular and maintainable codebase.
- You can create reusable helper functions and custom commands to encapsulate common actions or assertions, making it easier to write and maintain your test suite.
- The Cypress API also encourages the use of hooks and beforeEach/afterEach blocks, which can help you set up and tear down test environments consistently across your test cases.

#### **Test results produced:**

- Cypress provides detailed and informative test results, including automatic screenshots and videos of the test execution.
- The test results can be easily viewed in the Cypress Test Runner, which provides a user-friendly interface for analyzing the test runs.
- Cypress also integrates with various reporting tools, such as Mocha and Mochawesome, allowing you to generate detailed test reports and visualize the test results.

#### **Ease of usage:**

- Cypress is generally considered to be a developer-friendly testing framework, with a straightforward and intuitive API.
- The Cypress Test Runner provides a seamless development experience, allowing you to write, run, and debug your tests directly in the browser.
- Cypress also has a growing community and extensive documentation, making it easy for developers to learn and get started with the framework.

#### **Type of GUI elements that can be tested:**

- Cypress supports testing a wide range of GUI elements, including buttons, links, input fields, dropdowns, checkboxes, and more.
- It provides a comprehensive set of commands and assertions that allow you to interact with and assert on the state of these GUI elements.
- Cypress also handles common GUI-related challenges, such as waiting for elements to be present, visible, or clickable, making it easier to write stable and reliable tests.

Overall, Cypress is a powerful and feature-rich GUI testing tool that excels in end-to-end testing of web applications. Its ease of use, robust set of features, and strong community support make it a popular choice for developers looking to incorporate automated testing into their development workflow.