CSE 565: Software Verification, Validation and Testing

Assignment 5: Graphical User Interface Testing

Introduction

This report presents an in-depth analysis of the News Application, a graphical user interface (GUI) project developed as part of a GUI testing assignment. Built using HTML, CSS, and JavaScript, this application offers a user-friendly platform for navigating, searching, and interacting with news content. The News Application comprises three core pages: the Home Page, the Explore News Page, and the Article Page, each contributing to a seamless browsing experience. The application's design prioritizes accessibility and usability, allowing users to search for articles by keywords and categories, switch between light and dark themes, adjust text zoom levels, and add comments to news items. This report outlines the application's structure, detailing the purpose of each page, the technologies employed, and the primary features implemented, serving as a foundation for the subsequent GUI testing and evaluation.

Description of the GUI News Application

Version 1:

1. **Structure of the Application:**
   The news application is divided into three primary pages, each serving a unique purpose within the user flow. These pages include:
   I. **Home Page (index.html)**: The entry point where users can explore news categories, perform keyword searches, and select predefined categories.
   II. **News Feed Page (news.html)**: Displays a list of articles related to the selected category, allowing users to view summarized content and navigate to detailed articles.
   III. **Article Detail Page (article.html)**: Provides the full content of a selected or searched article and allows users to add comments.

2. **Page-by-Page Breakdown:**

   I. **Home Page:** The home page serves as the primary landing page, introducing users to the application and providing them with options to navigate through various topics. Below is the screenshot of the Home page for the News application. It shows the 2 search options namely the Keyword search and the Category search.

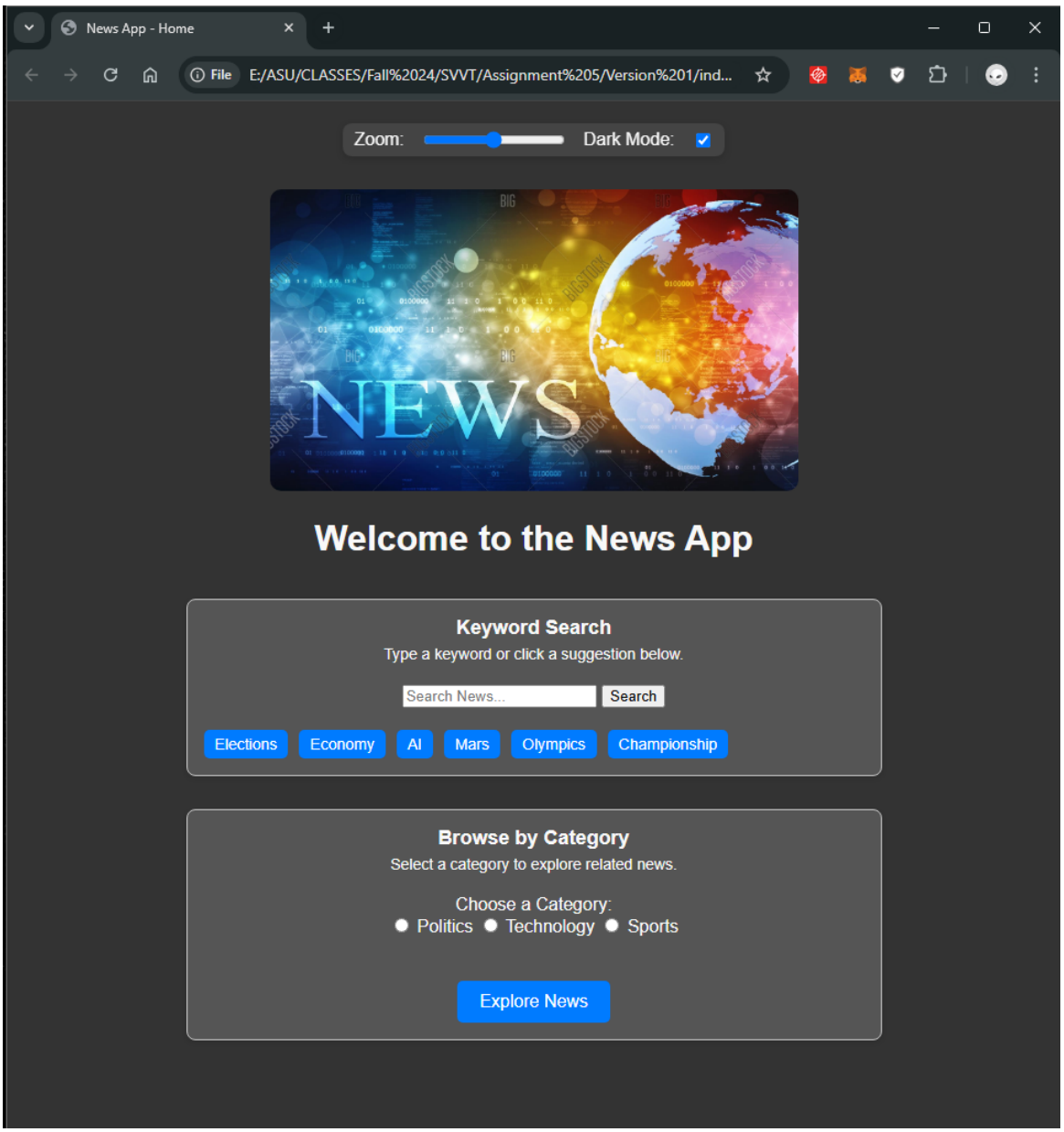

Image 1: Home Page of the News App

- **Structure**: This page is structured with a banner image at the top, followed by a "Browse by Category" section and a “keyword search” area.
- **Functionality**:
  - **Category Selection**: Users can select predefined categories, such as Politics, Technology, and Sports, using radio buttons. This selection redirects them to the news feed page to view articles within the chosen category.
  - **Keyword Search**: Users can type a keyword in the search box, or they can click on one-word suggestions displayed below the search box. Clicking the search button displays relevant results or an alert if no matching content is found.
  - **Responsive Design**: The page layout is designed to adjust based on the screen size, ensuring an optimal viewing experience on both desktop and mobile devices.
- **Technologies Used**: Built with HTML for structure, CSS for styling, and JavaScript for interactive functionalities.

II. **News Feed Page:** The news feed page displays articles based on the selected category from the home page. Users can see article summaries and navigate to full articles for more

detailed reading. Below is the screenshot of the News page feed which shows all the summary of the all the news articles in card format -

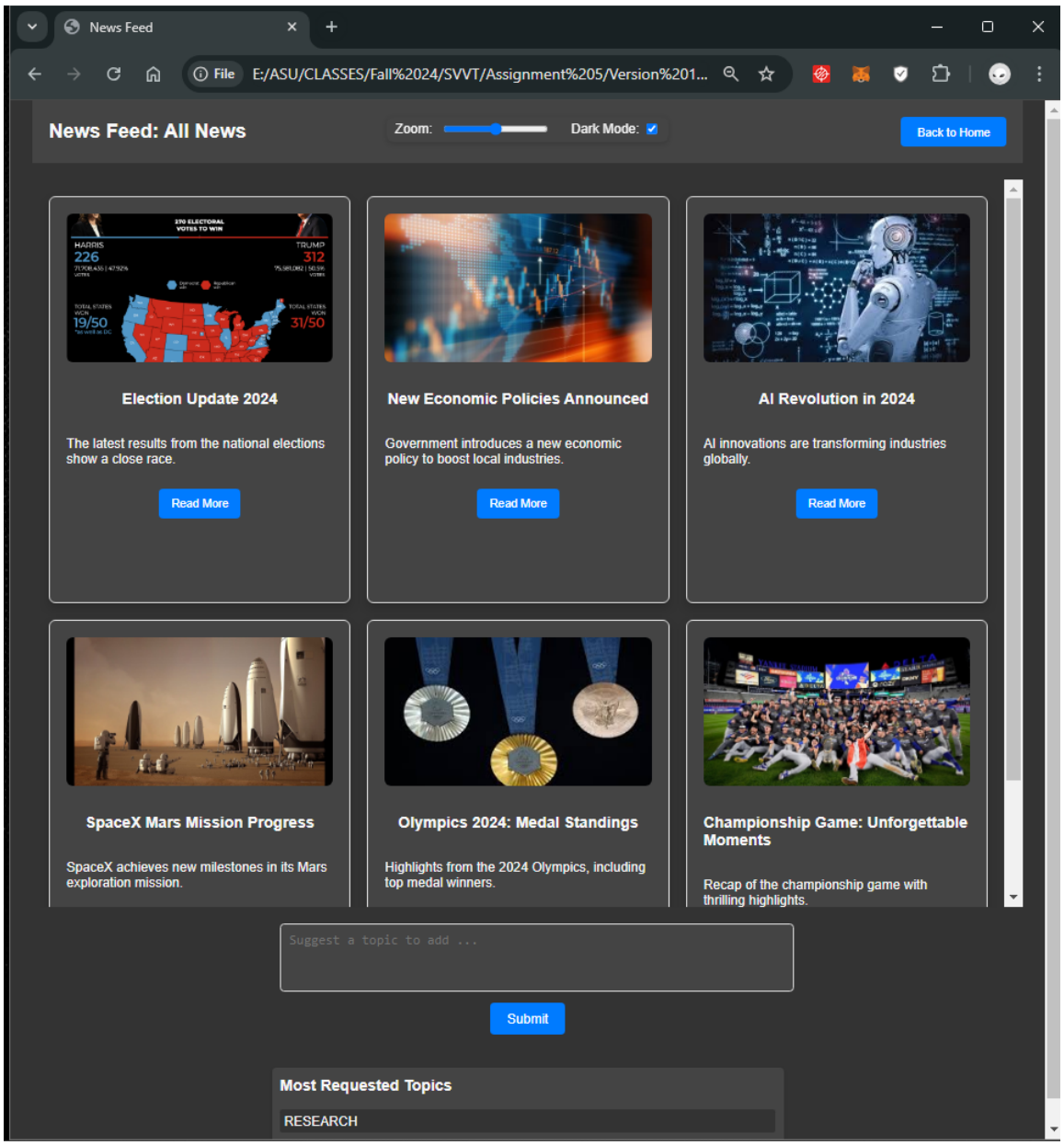

Image 2: Explore Page of the News App

- **Structure**: This page includes a top navigation section, an article list section, and a topic suggestion area.
- **Functionality**:
  - **Zoom Slider**: Allows users to adjust the zoom level of the page content for improved readability.
  - **Dark Mode Toggle**: Users can switch between light and dark modes, with their preference saved in local storage, so the mode persists across sessions.
  - **Back to Home Button**: A button that takes users back to the home page.
  - **Article List**: Displays articles in a grid format, showing a title, description, image, and a "Read More" button. Users can click on "Read More" to open the article detail page (article.html).

- **Topic Suggestion Box**: Allows users to suggest new topics, which are stored in local storage and displayed under "Most Requested Topics."

- **Technologies Used**: This page uses HTML for structure, CSS for styling and layout, and JavaScript for functionality, including local storage management for dark mode and topic suggestions.

III. **Article Detail Page:** The article detail page provides users with the complete content of an article selected from the news feed. This page also enables users to interact with the article by leaving comments. Below is the screenshot of the Article Detail Page -

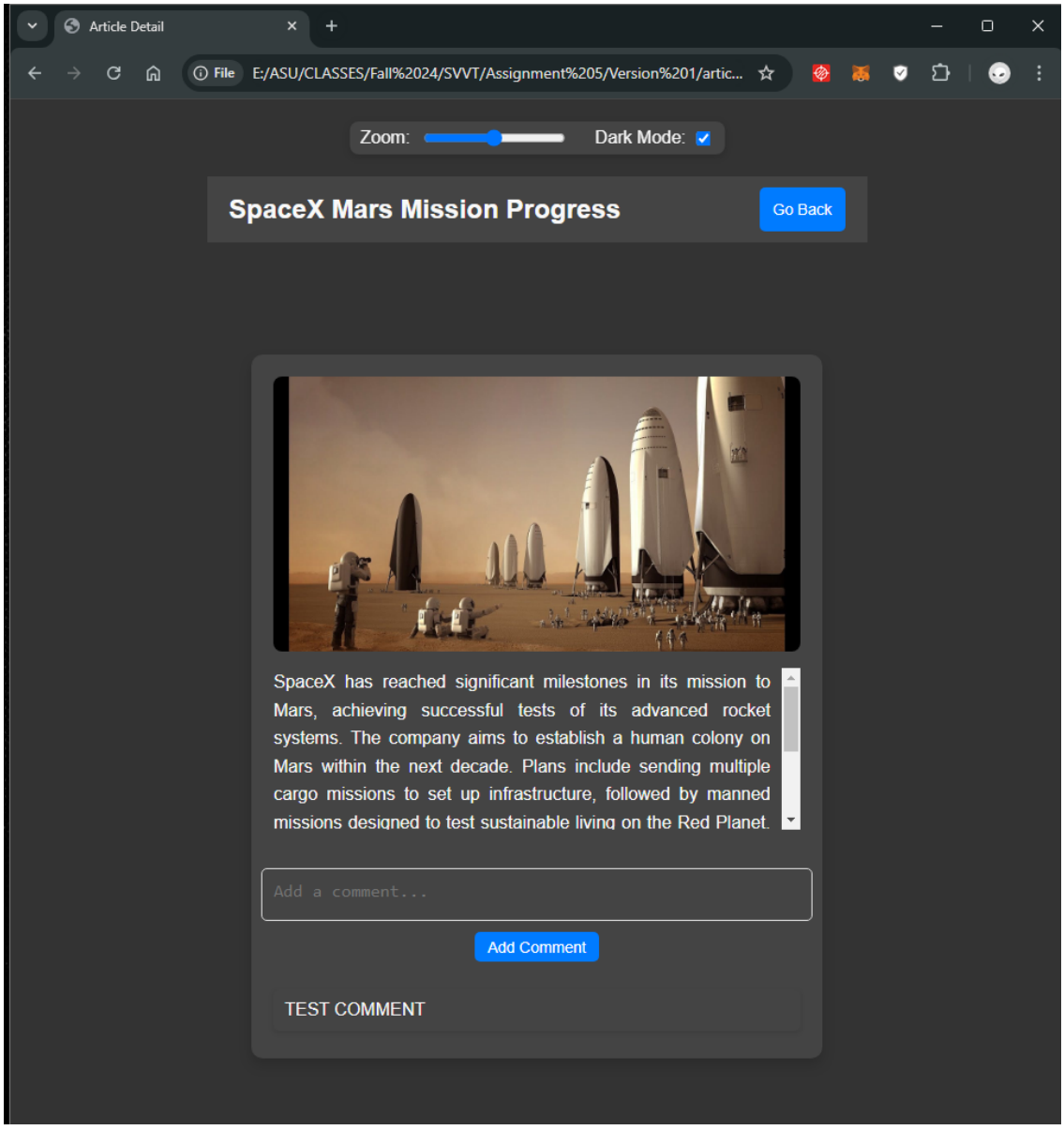

Image 3: Article Detail Page of the News App

- **Structure**: The page layout consists of a header, article content section, and a comment section.
- **Functionality**:
  - **Article Content Display**: Displays the full content of an article, including an image and text content.

- **Zoom Slider**: Similar to the news feed page, users can adjust the zoom level of the article content.
- **Dark Mode Toggle**: Enables users to switch between light and dark modes, with the setting saved across sessions.
- **Go Back Button**: Redirects users back to the news feed page.
- **Comment Box**: A text area where users can type comments. Comments are stored in local storage and persist even after page reloads.
- **Display of Comments**: A section that dynamically displays all comments submitted by users for the article.

- **Technologies Used**: Built with HTML, CSS for layout and design, and JavaScript for dynamic functionality, including comment storage and retrieval from local storage.

3. Application Flow

The application follows a structured and intuitive flow to guide users through the news browsing and reading experience. Here's how users navigate through the application's pages:

I. Home Page

- **Entry Point**: When users open the application, they start on the home page.
- **Options**:
  - **Browse by Category**: Users can select a category (e.g., Politics, Technology, Sports) and click the "Explore News" button to see articles related to that category. This selection redirects them to the news feed page (news.html), where only the articles within the selected category are displayed.
  - **Keyword Search**: Users can enter keywords in the search box or click on suggested keywords below it to search for specific articles. After typing in a keyword and pressing the search button, the application either:
    - Redirects to the article detail page (article.html) if a relevant article is found, or
    - Displays a message if no matching article is found.

II. News Feed Page

- **Purpose**: Displays a list of articles filtered by the selected category from the home page.
- **User Actions**:
  - **Read More**: Users can click on "Read More" for any article, which redirects them to the article detail page (article.html) to read the full content.
  - **Back to Home**: A button at the top of the page allows users to return directly to the home page (index.html), bringing them back to the entry point of the application.
  - **Topic Suggestion**: Users can suggest new topics by entering text in a suggestion box. Suggested topics are saved locally and displayed in the "Most Requested Topics" section for reference and future content updates.

III. **Article Detail Page**

- **Purpose**: Provides the full content of a specific article selected from the news feed page.
- **User Actions**:
  - **Add Comment**: Users can type and submit comments related to the article. These comments are stored in local storage, allowing them to persist even if the page is reloaded.
  - **Go Back**: A "Go Back" button allows users to return to the news feed page (news.html). This flow supports the user journey by allowing them to return to the list of articles and continue browsing other content within the same category.

This flow ensures that users have a seamless experience, with clear pathways to navigate between browsing categories, reading articles, and revisiting previous pages. Each navigation option is designed to bring users back to the home page or allow them to browse related content effortlessly.

4. Technologies and Tools

The news application is entirely client-side and uses the following technologies:

- **HTML**: Provides the structural foundation of each page.
- **CSS**: Controls the styling and layout, creating a responsive and visually appealing interface.
- **JavaScript**: Implements core functionality, such as search, navigation, dark mode, comment handling, and local storage management.

5. Core Functionalities

1. **Category-Based Navigation**: The home page's "Browse by Category" section allows users to filter news articles by predefined categories, which are passed to the news feed page.
2. **Keyword Search with Suggestions**: The search box on the home page provides keyword suggestions, and users can enter keywords to locate relevant articles. If the keyword doesn't match any article, an alert is displayed.
3. **Dark Mode Toggle**: Users can switch between light and dark themes on any page. This preference is saved in local storage, so it persists across sessions and pages.
4. **Zoom Control**: Both the news feed and article detail pages provide a zoom slider, allowing users to adjust the page's zoom level for better readability.
5. **Navigation Flow**: Each page has consistent navigation elements, such as "Back to Home" and "Go Back" buttons, which create a seamless user experience across pages.
6. **Comment System**: The article detail page allows users to leave comments on articles. These comments are stored in local storage and persist even after reloading the page.
7. **Topic Suggestion Box**: On the news feed page, users can suggest topics they'd like to see, which are saved in local storage and displayed in a "Most Requested Topics" section.

6. GUI Changes in Version 2

In Version 2 of the news application, the goal was to update each page with specific changes in **size**, **location**, and **orientation** of elements to meet the assignment's requirement of three modifications per page. Each change was designed to enhance the visual layout and provide a refreshed user experience while preserving core functionality from Version 1. Below is a detailed explanation of the updates implemented on each page.

Home Page

- **Intentional Bugs**:
    1. **Mirrored Banner Image**: The banner image, which should appear normally, is mirrored to simulate an orientation bug. This change helps test the orientation properties of visual elements.
    2. **Left Alignment of the Banner**: The banner is aligned to the left instead of its intended centered position. This bug tests the layout and alignment settings in the GUI.
    3. **Increased Size of Search Button**: The search button's size is exaggerated to 2x, testing if resizing affects the functionality or layout of neighboring elements.

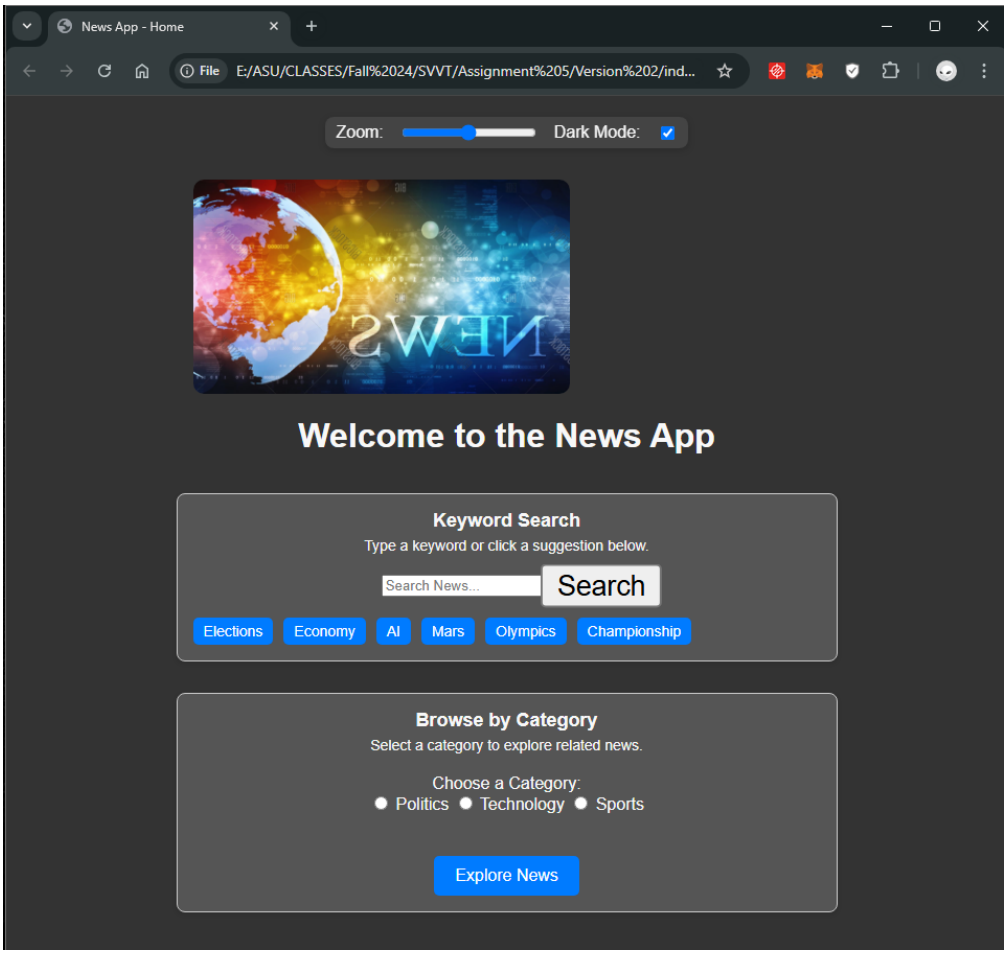

Image 4: Home page on Version 2

The screenshot highlights the three key changes on the home page in Version 2: the banner image is now left-aligned instead of centered, mirrored for orientation, and the search button beside the search box is enlarged. These modifications introduce location, orientation, and size changes for GUI testing.

News Feed Page

- **Intentional Bugs**:

    1. **Repositioned "Back to Home" Button**: The "Back to Home" button is now grouped with other controls at the top, testing navigational consistency and location-based interactions.

2. **Increased Size of "Read More" Buttons**: All "Read More" buttons are enlarged to 2x, which may interfere with layout consistency and readability, providing a test for button sizing and responsive design.
3. **Rotated Article Images**: Each article image is rotated by 180 degrees, introducing an orientation inconsistency to test if images load correctly in their intended orientation.

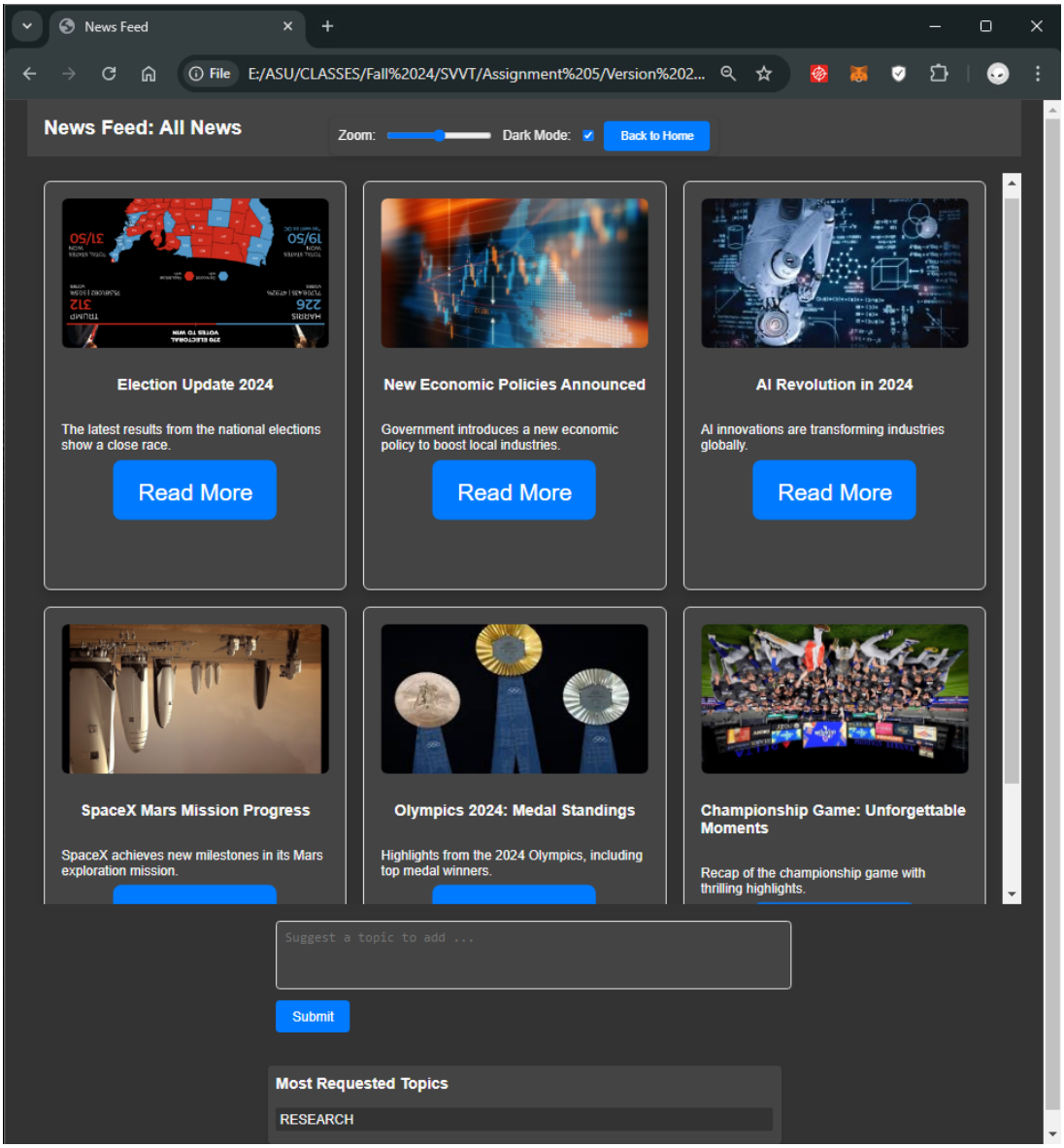

Image 5: Explore Page on Version 2

The above screenshot of the News feed page highlights the changes made. The images are inverted (rotated 180 degrees), the read more buttons have enlarged sizes and lastly the submit button's position is changed and is now aligned to the left side instead of being in the center of the page.

Article Detail Page

- **Intentional Bugs**:

1. **Rotated Main Article Image**: The main article image is rotated by 180 degrees, creating an orientation issue for testing image display properties.
2. **Right-Aligned "Add Comment" Button**: The "Add Comment" button is moved to the right, disrupting its usual flow with the comment box, to test if the placement of interactive elements affects user flow.
3. **Increased Size of "Add Comment" Button**: The button size is doubled, potentially interfering with neighboring elements and testing size consistency and usability.

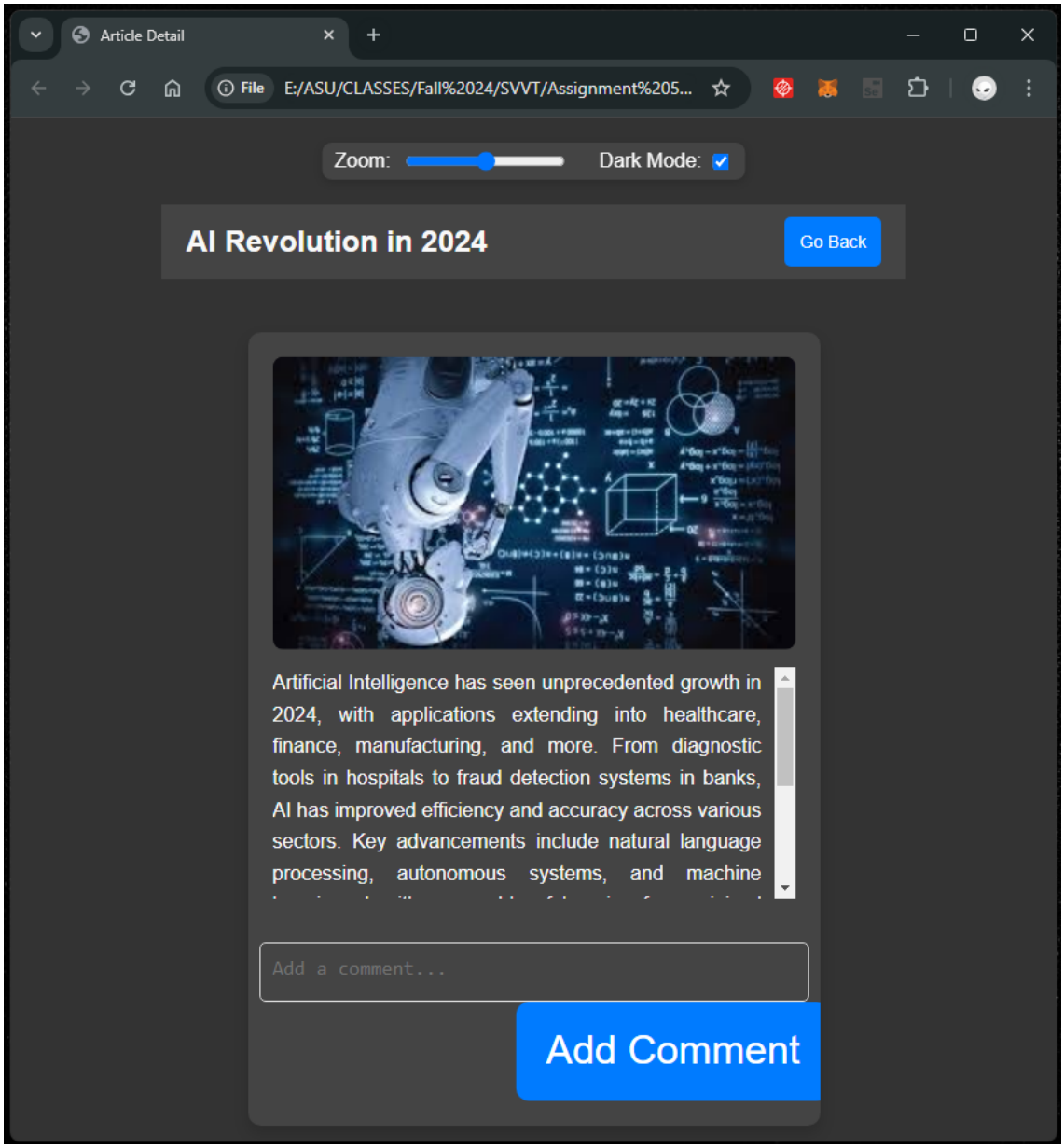

Image 6: Article Detail page of Version 2

The changes to the article page can be clearly seen in the above screenshot. The orientation of the image is rotated 180 degrees, the “Add comment” button is now positioned to the right of the article card and furthermore the size of this button is also enlarged.

Flow Modification

The application's flow remains largely unchanged from Version 1, with users navigating from the home page to the news feed page and, finally, to the article detail page. However, the intentional bugs introduce additional checkpoints in the flow to ensure all navigational elements work as expected and direct users to the correct pages.

- **Modification**: The "Go Back" button on the article detail page (article.html) is adjusted to redirect to the home page (index.html) instead of the news feed page. This introduces an unexpected navigation flow that can be tested to confirm if the "Go Back" functionality maintains consistent redirection across the application.

Conclusion

Version 2 of the news application serves as a testing environment, deliberately introducing errors across GUI elements. This controlled setup allows for comprehensive GUI testing using tools like Cypress, providing a benchmark to ensure Version 1's interface functions correctly and resiliently. Each modification in Version 2 was purposefully added to validate key elements, including orientation, size, location, and navigation, thus establishing a robust testing foundation.

Task 2: GUI Testing Tool - Cypress

For GUI testing of the news application, **Cypress** was chosen due to its modern, developer-friendly approach to end-to-end testing of web applications. Cypress is a robust testing tool that supports real-time reloading, time travel, and direct access to the browser's DOM, allowing precise control over GUI elements. Built primarily for front-end testing, Cypress operates within the same environment as the application, providing direct interaction with page elements and enabling tests that are highly reliable and resilient to changes in the application. This makes Cypress especially suitable for validating the existence, location, size, and functionality of GUI components in dynamic web applications.

Key Features of Cypress

- **Real-Time Reloading**: Cypress automatically reloads tests every time you make changes, allowing for instant feedback and efficient debugging.
- **Time Travel**: Cypress captures screenshots as it runs each test step, allowing you to "travel" back in time to see the state of the application at each stage. This is useful for identifying and troubleshooting GUI issues.
- **Automatic Waiting**: Unlike traditional testing tools, Cypress automatically waits for elements to load and commands to complete, reducing the need for manual wait times in scripts and improving test stability.
- **Direct Access to DOM**: Cypress has direct access to the DOM, which enables more accurate and efficient testing of GUI elements. It can inspect elements in real time, allowing precise testing of element properties such as location, size, and visibility.
- **Screenshots and Video Recording**: Cypress can automatically take screenshots of failed tests and record videos of the entire test suite, which is beneficial for analyzing GUI issues and understanding unexpected test results.
- **JavaScript Integration**: Cypress uses JavaScript, which is well-suited for web development and enables the integration of tests with development workflows. This allows developers to write test cases using a language they are familiar with, streamlining the testing process.

Scope of Cypress

Cypress is designed primarily for front-end testing of web applications and can handle a wide range of testing tasks, including:

- **End-to-End Testing**: Cypress excels in testing entire user flows to ensure a seamless experience across multiple pages.
- **Component Testing**: Ideal for testing individual GUI components, ensuring each element functions correctly in isolation.
- **Integration Testing**: Cypress can simulate user interactions with multiple components to validate that they work together as expected.
- **Regression Testing**: With Cypress, developers can quickly re-run test cases after changes to detect unexpected GUI issues, ensuring stability across updates.

Areas of Usage for Cypress

Cypress is highly effective in the following areas:

- **GUI Validation**: Cypress is ideal for verifying the visibility, location, and size of GUI elements, ensuring the application's layout is as expected.
- **User Interaction Testing**: Cypress can simulate real user actions, such as clicks, text input, navigation, and form submissions, to test the functionality of the GUI.
- **Cross-Browser Testing**: While Cypress's cross-browser support is growing, it currently supports Chrome, Edge, and Firefox, enabling testing across multiple environments.
- **Responsive Design Testing**: By configuring different viewport sizes, Cypress can validate that the application's GUI adapts correctly across devices.

Conclusion

In summary, Cypress offers a comprehensive set of features tailored for modern web applications, making it an ideal choice for GUI testing of the news application. Its support for end-to-end, component, and integration testing ensures that all critical elements of the user interface are validated across various interactions and environments. The tool's real-time feedback, automated waiting, and DOM-level control enhance the efficiency and reliability of the testing process. For the news application, Cypress enables robust verification of GUI elements and user flows, ensuring consistency, usability, and performance across browsers and devices. These capabilities make Cypress a powerful solution for delivering a stable and user-friendly application interface.

TASK 3: Designing Testcases and Testing Version 1 of the News Application

Using Cypress, a set of test cases was developed to validate the GUI of Version 1 of the News Application. The focus was on verifying key aspects such as element presence, size, position, content, and flow between pages. The following describes the test cases developed and includes screenshots illustrating the results.

Test Cases Overview

Each page's test cases cover essential GUI elements, ensuring that the application meets design and functionality expectations. Key elements such as buttons, images, and navigation flows were tested.

Categorizing the testcases for Home Page (Index.html):

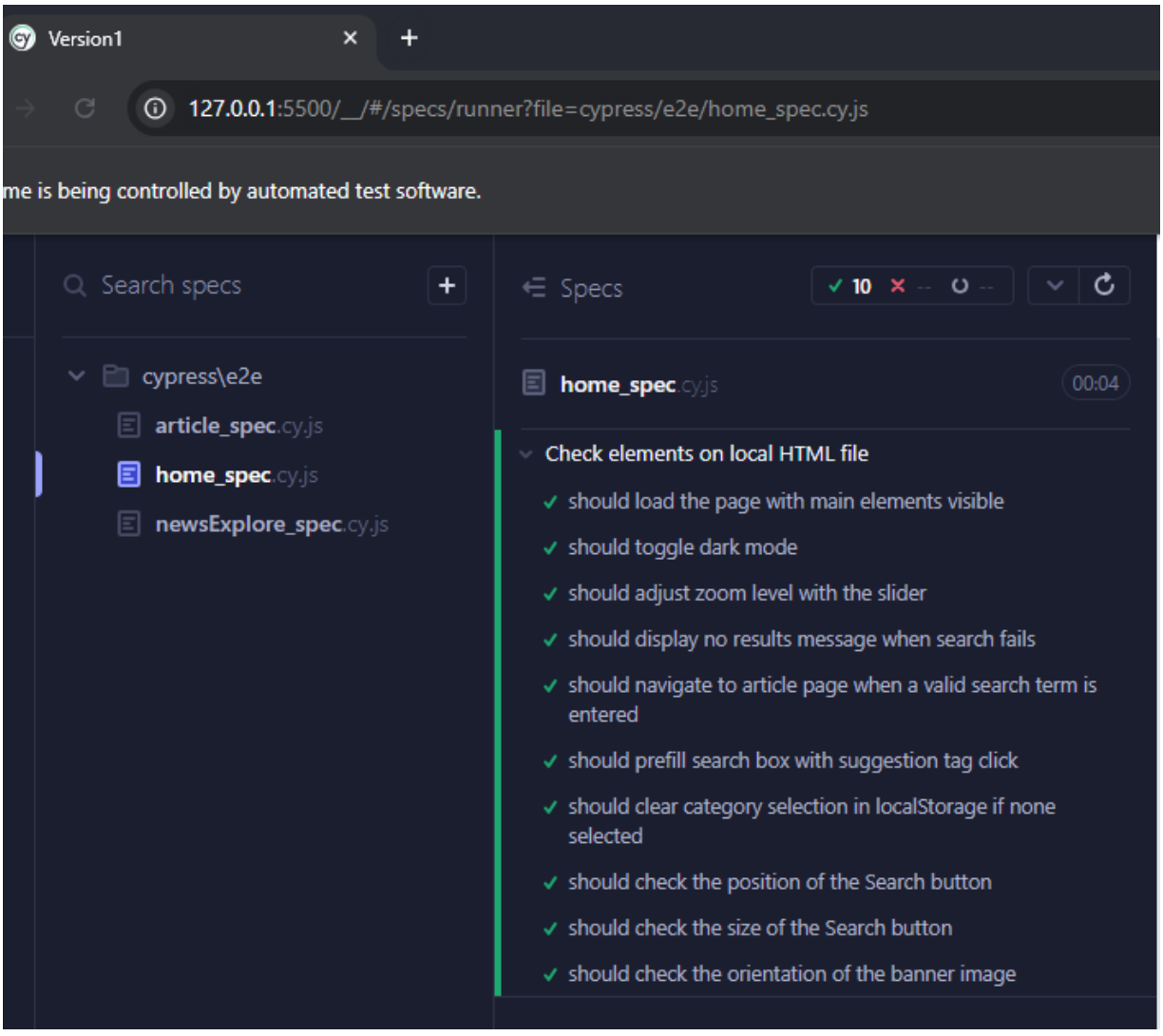

Image 7: Test case Run for Home Page using Cypress

The above screenshot shows successful test results for each test case on the index.html page, indicating that all elements were found and displayed correctly. The checkmarks confirm the proper function of dark mode, zoom control, and search functionality, along with verifying the position, size, and orientation of the elements.

- **Position Check**
  - **Check Position of the Search Button**: Verifies the exact location of the "Search" button next to the search box to ensure it aligns with design specifications.
- **Orientation Check**
  - **Check Orientation of the Banner Image**: Confirms that the banner image displays in the correct orientation without any unintended rotation or mirroring.
- **Existence Check**
  - **Verify Visibility of Main Elements**: Checks that key elements (header, main content, controls, search box) are visible on the page, ensuring that all essential components are loaded correctly.
  - **Dark Mode Toggle Presence**: Confirms the dark mode toggle button exists and is accessible in the controls section.
  - **Zoom Control Presence**: Verifies that the zoom control slider is present and correctly displayed.
- **Size Check**
  - **Check Size of the Search Button**: Ensures the dimensions of the "Search" button match the expected size for ease of access and visibility.
- **Navigation Check**
  - **Search Navigation to Article Page**: Tests that entering a valid keyword in the search box and clicking "Search" navigates to article.html with the appropriate article data.
  - **Search Navigation with Invalid Term**: Verifies that entering an invalid term in the search box displays a "No news found" message without navigating away from the page.
- **Content Coverage**
  - **Display of Search Suggestions**: Ensures the search suggestions load and appear as clickable options under the search box.
  - **Search Functionality with Predefined Keywords**: Checks that keywords such as "AI" and "Economy" return expected results.

Categorizing the testcases developed for News Explore Page (news.Html):

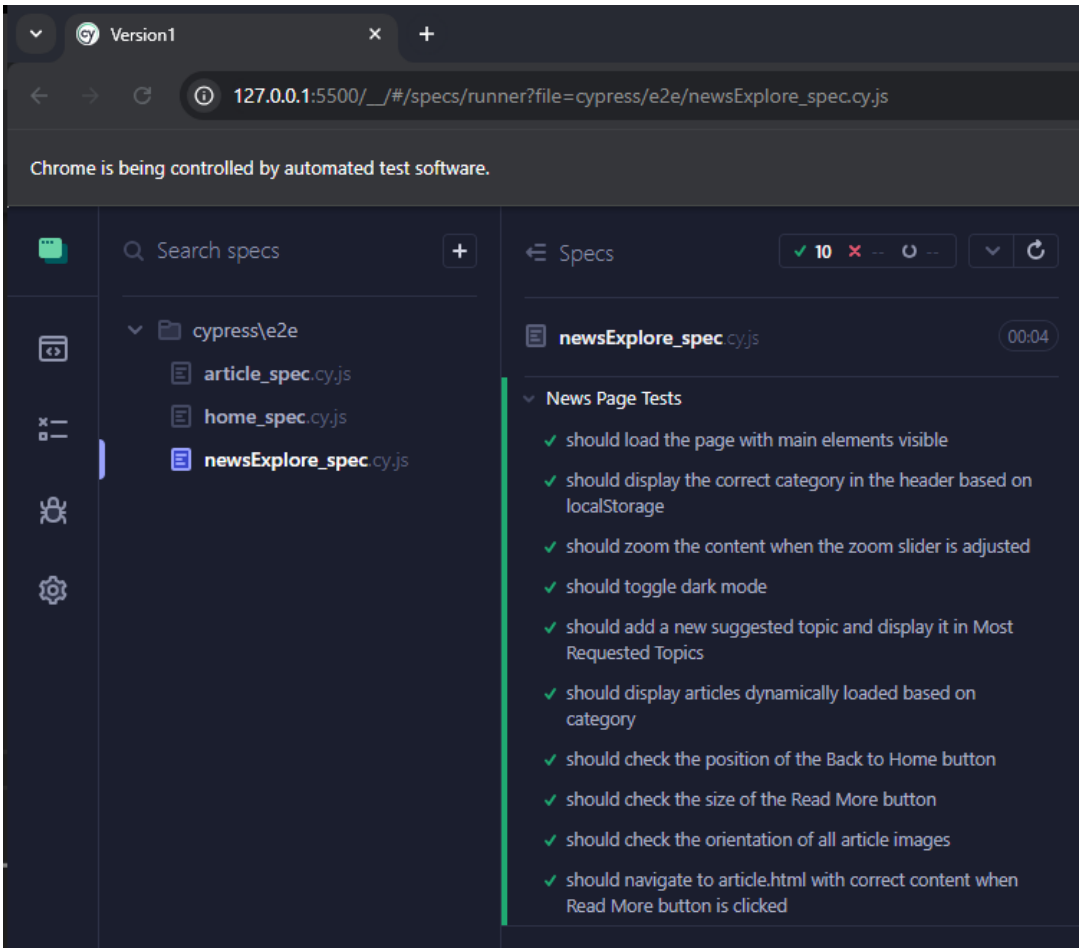

Image 8: Test case run for News Explore Page using Cypress

The above screenshot displays the results of test cases for the news.html page, with each test passing. The checks confirm that each GUI element was correctly loaded and that functionality, including dark mode, zoom, article display, and navigation, worked as expected.

- **Position Check**
  - **Check Position of the "Back to Home" Button**: Ensures the "Back to Home" button is located in the correct position on the page.
  - **Check Position of the Add Suggestion Button**: Verifies that the button to add suggestions is correctly positioned next to the suggestion box for easy access.

- **Orientation Check**
  - **Check Orientation of All Article Images**: Ensures that all article images display in their default orientation without any rotations.

- **Existence Check**
  - **Verify Visibility of Header and Content Area**: Confirms that the header, content area, and other main components load and are visible on the page.
  - **Dark Mode Toggle Presence**: Checks that the dark mode toggle button exists and functions within the page's controls.

- **Zoom Control Presence**: Verifies the presence of the zoom control slider for adjusting content scale.

- **Size Check**
  - **Check Size of the "Read More" Buttons**: Ensures that the "Read More" buttons are adequately sized for readability and accessibility.

- **Navigation Check**
  - **Navigation from "Read More" to Article Page**: Verifies that clicking the "Read More" button on an article navigates to the appropriate article.html page with the correct content.
  - **Back Button Navigation to Home Page**: Ensures that clicking the "Back to Home" button redirects the user to the index.html page.

- **Content Coverage**
  - **Display of Category-Specific Articles**: Confirms that articles dynamically load according to the selected category, showing only relevant articles.
  - **Most Requested Topics Section**: Verifies that user suggestions are saved in localStorage and displayed under "Most Requested Topics."

Categorizing the testcases developed for Article Page (article.html):

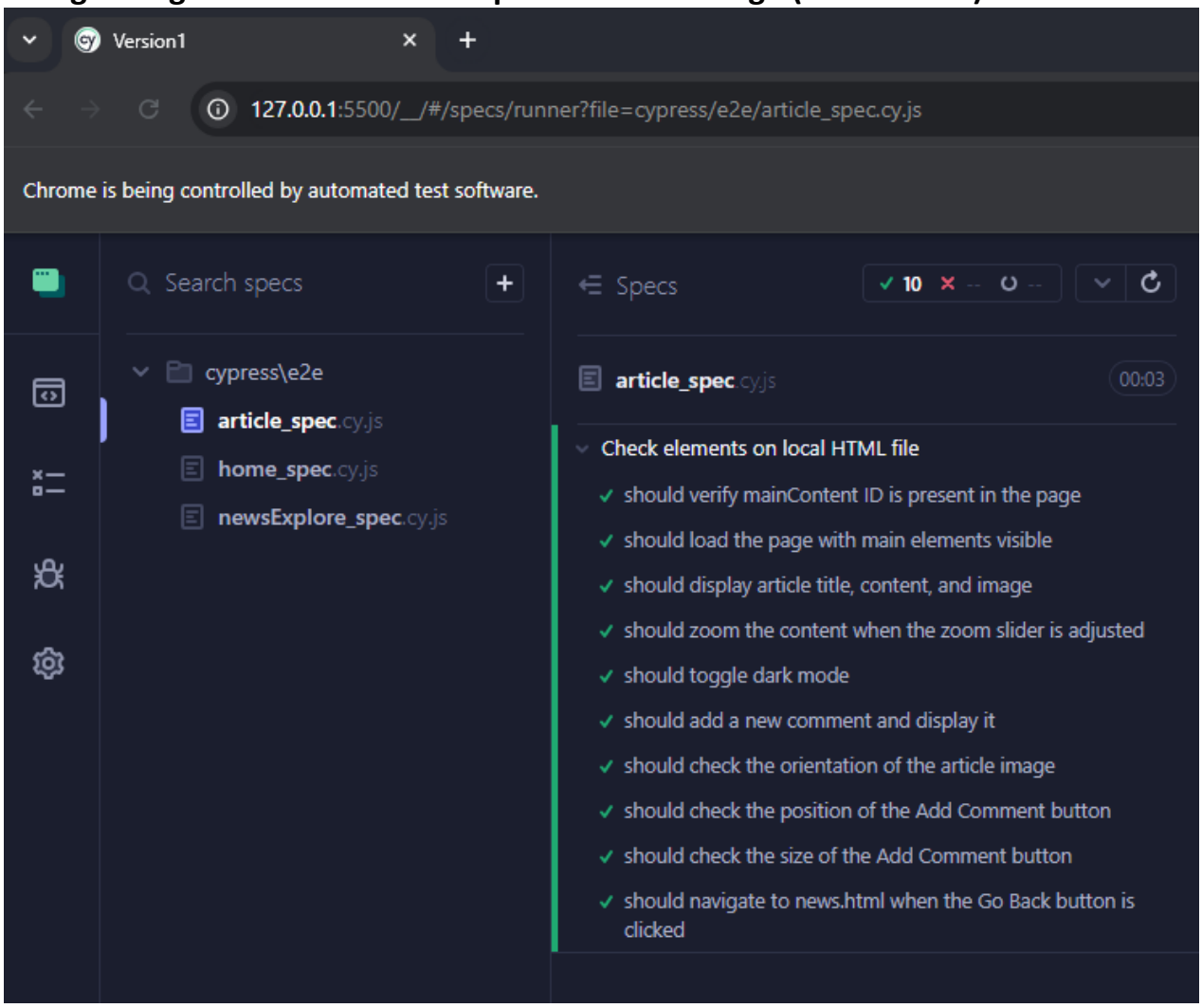

Image 9: Test case Run for Article Page using Cypress

The above screenshot captures the article.html test results, with all test cases passing successfully. The screenshot highlights the successful validation of GUI elements, including dark mode, zoom control, and the add comment feature, along with verifying the correct position and size of elements.

- **Position Check**
  - **Check Position of the "Add Comment" Button**: Ensures the "Add Comment" button is correctly positioned on the right side for better alignment with the comments box.
  - **Check Position of the Zoom and Dark Mode Controls**: Verifies that the controls are positioned at the top of the page for easy access.

- **Orientation Check**
  - **Check Orientation of the Article Image**: Ensures that the main article image displays in its default orientation without rotation or mirroring.

- **Existence Check**
  - **Verify Visibility of Main Content**: Confirms that the main content, including the article title, content area, and image, is visible upon loading the page.
  - **Dark Mode Toggle Presence**: Ensures that the dark mode toggle button is present on the page.
  - **Zoom Control Presence**: Checks that the zoom control slider is visible and accessible.

- **Size Check**
  - **Check Size of the "Add Comment" Button**: Ensures the "Add Comment" button is appropriately sized for visibility and ease of interaction.

- **Navigation Check**
  - **Go Back Button Navigation to News Page**: Confirms that clicking the "Go Back" button takes the user to news.html as intended.

- **Content Coverage**
  - **Display of Article Content from LocalStorage**: Verifies that the article title, content, and image data are correctly retrieved from localStorage and displayed on the page.
  - **Add Comment and Display in Comments Section**: Checks that adding a comment updates localStorage and displays it within the comments section.

TASK 4: Testing Version 2 of the News Application with Same Testcases

For this task I evaluated the GUI of Version 2 of the News Application using an automated test suite developed with Cypress. This version introduced specific changes in the layout, element positioning, size, orientation, and navigation flow, simulating potential issues for testing purposes. The tests conducted aimed to verify whether these changes affected the application's stability and functionality. By executing the same set of test cases as in Version 1, I observed which features remained consistent and which ones were impacted, providing a thorough assessment of the GUI's behavior in the updated version.

Test Case Results for the Home Page (index.html)

A total of 10 test cases were developed for version 1 of the application. When these test cases were used to test the version 2 of the application, 6 of the test cases passed and 4 of the test cases failed. Below is a screenshot showcasing the test results for the test cases that were run on the Version 2 of the Home Page.

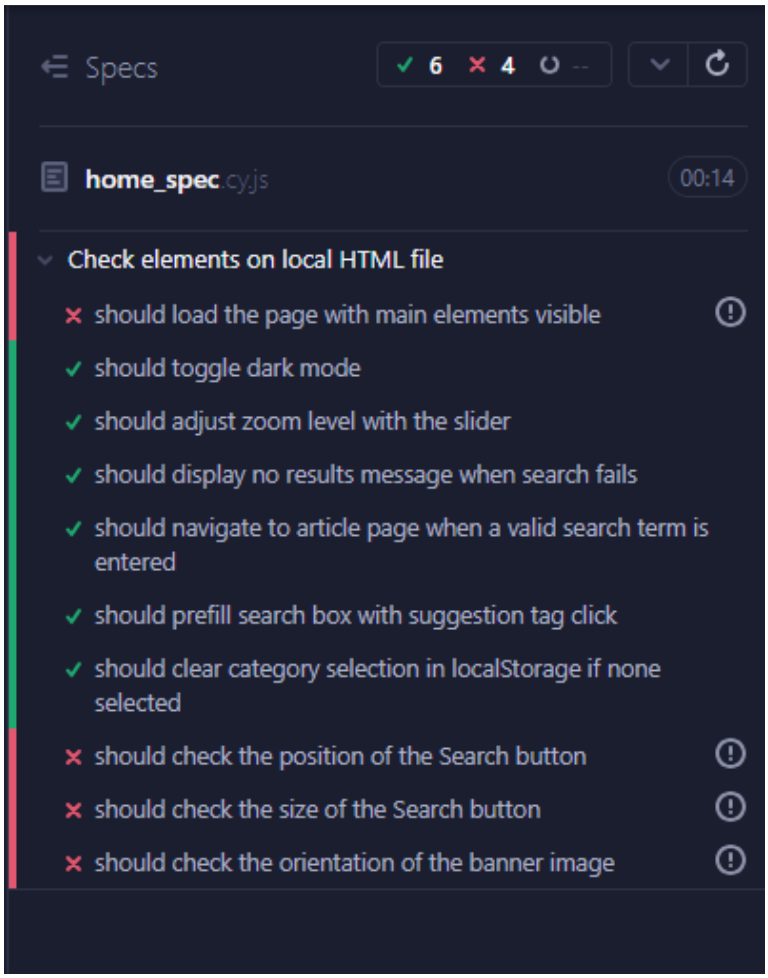

Image 10: Test Case Results for Version 2 of Home Page

Passed Test Cases:

1. **should toggle dark mode** - *Passed*: The dark mode toggle functioned as expected, allowing users to switch between dark and light themes.
2. **should adjust zoom level with the slider** - *Passed*: The zoom slider worked correctly, allowing content scaling adjustments.
3. **should display no results message when search fails** - *Passed*: Displayed an error message when an invalid search term was entered, confirming correct handling of search errors.
4. **should navigate to article page when a valid search term is entered** - *Passed*: Redirected to the article page as expected when a valid search term was entered, verifying navigation functionality.
5. **should prefill search box with suggestion tag click** - *Passed*: Clicking a suggestion tag populated the search box with the relevant term.
6. **should clear category selection in localStorage if none selected** - *Passed*: Cleared the category selection in localStorage when no category was selected, working as expected.

Failed Test Cases*:*

1. **should load the page with main elements visible** - *Failed*: Main elements on the page were either missing or misaligned, impacting initial page visibility.
2. **should check the position of the Search button** - *Failed*: The position of the "Search" button was altered, failing this test which expected it to remain in the original location.
3. **should check the size of the Search button** - *Failed*: The "Search" button size was increased, leading to a mismatch with the expected dimensions.
4. **should check the orientation of the banner image** - *Failed*: The banner image was mirrored in Version 2, which failed the test expecting the original orientation.

Test Case Results for the News Explore Page (news.html)

Similar to the home page, a total of 10 test cases were developed to test the Explore page of the News applciation. All of the test cases passed when ran on the Version 1 of the application. However, when running these same tests on the version 2 of the application, 4 test cases failed and the rest 6 test cases passed. Below is a screenshot of the results for the test cases that were run on this page.

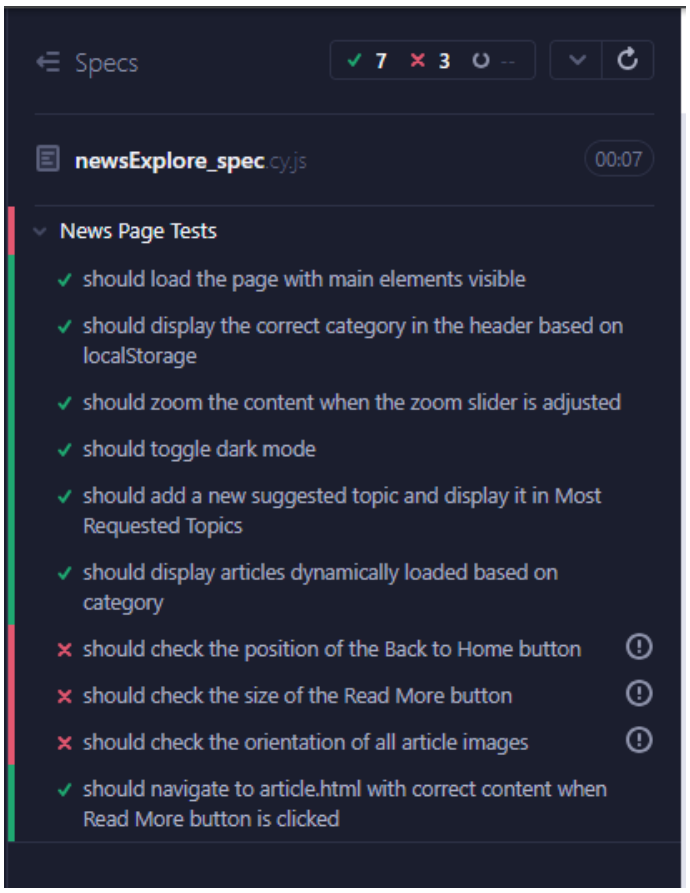

Image 11: Test Case Results for Version 2 of Explore Page

Passed Test Cases:

1. **should load the page with main elements visible** - *Passed*: All main elements loaded successfully, ensuring the core structure remained intact.
2. **should display the correct category in the header based on localStorage** - *Passed*: Displayed the correct category in the header from localStorage, confirming correct data retrieval.
3. **should zoom the content when the zoom slider is adjusted** - *Passed*: The zoom slider adjusted the content size as expected.
4. **should toggle dark mode** - *Passed*: The dark mode toggle functioned correctly, allowing theme switching.
5. **should add a new suggested topic and display it in Most Requested Topics** - *Passed*: Successfully added a new topic and displayed it under "Most Requested Topics."
6. **should display articles dynamically loaded based on category** - *Passed*: Displayed category-specific articles as expected, showing that dynamic loading remained functional.
7. **should navigate to article.html with correct content when Read More button is clicked** - *Passed*: The "Read More" button successfully navigated to the article page with the correct article loaded.

Failed Test Cases:

1. **should check the position of the Back to Home button** - *Failed*: The "Back to Home" button was repositioned, leading to a mismatch with the expected location.
2. **should check the size of the Read More button** - *Failed*: The "Read More" button size was increased in Version 2, resulting in a discrepancy with the original expected size.

3. **should check the orientation of all article images** - *Failed*: The orientation of the article images was changed, causing this test to fail.

Test Case Results for the Article Page (article.html)

Furthermore, a similar set of 10 test cases were desgined to test the article page of the application. Similar to the previous runs on the home page and the explore page, all the ten test cases passed when run on the version 1 of the article page. However, 4 of the test cases failed while the rest 6 test cases passed on the version 2 of the article page. Below is a screenshot of the test results for this page.

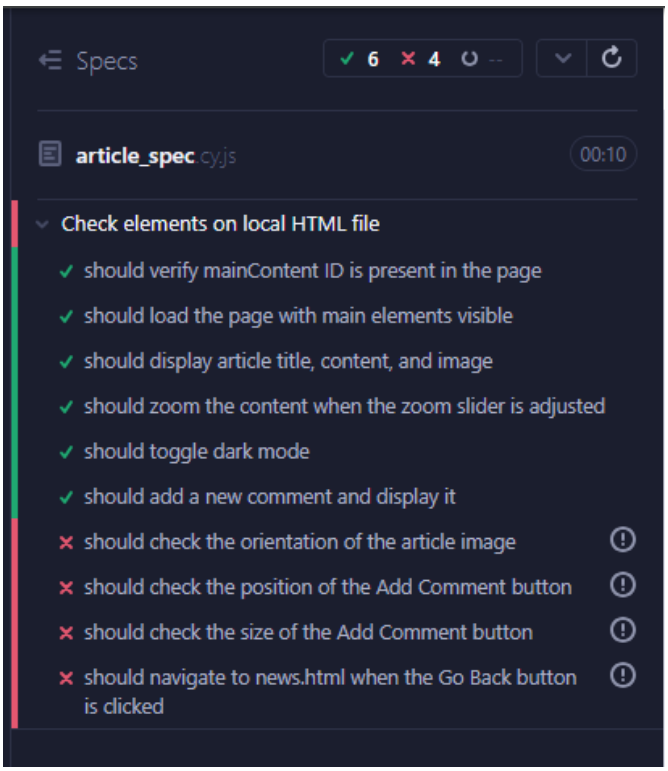

Image 12: Test Case Results for Version 2 of Article Page

Passed Test Cases:

1. **should verify mainContent ID is present in the page** - *Passed*: The `mainContent` ID was found as expected, verifying that the main content structure remained unchanged.
2. **should load the page with main elements visible** - *Passed*: All main elements, including title, content, and image, loaded correctly.
3. **should display article title, content, and image** - *Passed*: Displayed the article's title, content, and image from localStorage as expected.
4. **should zoom the content when the zoom slider is adjusted** - *Passed*: The zoom functionality worked, allowing content scaling adjustments.
5. **should toggle dark mode** - *Passed*: Dark mode was successfully toggled, switching between themes as expected.
6. **should add a new comment and display it** - *Passed*: Successfully added and displayed a comment in the comments section.

Failed Test Cases:

1. **should check the orientation of the article image** - *Failed*: The article image was rotated by 10 degrees, failing this test expecting the original orientation.
2. **should check the position of the Add Comment button** - *Failed*: The "Add Comment" button was moved to the right, leading to a failure in the position check.
3. **should check the size of the Add Comment button** - *Failed*: The button size was altered in Version 2, causing a mismatch with the expected size.
4. **should navigate to news.html when the Go Back button is clicked** - *Failed*: The "Go Back" button now redirected to the Home Page instead of the News Page, failing the navigation flow test.

The passed test cases show that essential features, including theme switching, content loading, and navigation flow, remained stable and functional in Version 2, demonstrating the application's resilience against layout changes.

The failed test cases highlight the impact of GUI modifications on the application's consistency. By altering the layout, element positioning, size, orientation, and navigation flow, these changes caused the tests to detect deviations from the expected behaviors in Version 1.

Conclusion

The execution of the Cypress test suite on Version 2 of the News Application effectively demonstrated the impact of GUI modifications on the application's functionality and layout consistency. While core features like dark mode, zoom control, search, and article loading continued to function correctly, deliberate changes in element positioning, size, orientation, and navigation led to multiple test failures. These results emphasize the importance of thorough GUI testing to identify discrepancies across versions, ensuring that changes do not inadvertently disrupt user experience.

By successfully identifying both stable and altered elements within the application, this testing approach validated the robustness of the test suite and highlighted areas for adjustment to maintain a cohesive and functional GUI in future updates.

TASK 5: Detailed Assessment of Cypress for GUI Automation Testing

In this assignment, Cypress was used as the GUI automation tool for testing the Version 1 and Version 2 implementations of the News Application. Below is an assessment of how well Cypress performed in terms of features, coverage, reusability, test result accuracy, usability, and the type of GUI elements that it can test effectively.

1. Set of Features and Functionalities Provided

Cypress offers a comprehensive set of features tailored for front-end testing, making it highly suitable for GUI-based applications. Key features include:

- **DOM Manipulation and Assertions**: Cypress can directly access and manipulate DOM elements, making it effective for testing element visibility, location, size, and content.
- **Automatic Waiting**: Unlike other tools that often require manual wait commands, Cypress automatically waits for elements to load, which minimizes the chances of false failures due to timing issues.
- **Real-time Reloads and Debugging**: Cypress allows real-time reloading and debugging, making it easy to inspect elements, observe interactions, and adjust tests on the fly.
- **Built-in Assertions Library**: Cypress provides built-in assertions, making it easy to write tests without requiring additional assertion libraries.

These features made it straightforward to create, execute, and maintain test cases, especially for verifying the location, size, orientation, and existence of various GUI elements in the application.

2. Type of Coverage

Cypress provides comprehensive coverage for:

- **Functional Testing**: Validating that elements and features (e.g., search functionality, dark mode toggle) behave as expected.
- **UI Validation**: Verifying the layout, positioning, and orientation of elements across different pages.
- **End-to-End Testing**: Cypress supports full end-to-end testing, enabling the simulation of real user journeys across the application.
- **Error Handling and Edge Case Testing**: Cypress allows testing for error states, such as checking for messages when invalid input is provided.

The tool was effective in covering most areas of the application's GUI, ensuring that both functionality and design elements were validated.

3. Reuse of Test Cases

Cypress test cases are highly reusable. Once created, test cases can be re-run on different versions of the application to validate consistency across changes. For this project:

- **Reusability for Regression Testing**: The same test cases used for Version 1 were easily reusable for Version 2 without modification, allowing us to quickly assess which areas had changed or failed.

- **Reusable Assertions**: Assertions for checking position, size, orientation, and presence of elements were reusable, which reduced redundant code and made it easy to run the same checks across multiple pages.
- **Component Reuse**: Cypress allows the modularization of tests, so test cases for common elements (e.g., buttons or sliders) could be reused across different contexts or pages.

4. Test Results Produced

Cypress produced detailed test results, making it easy to analyze which tests passed or failed and understand the reasons:

- **Clear Pass/Fail Status**: Cypress provides a clear indication of pass and fail status, with color-coded feedback for quick identification.
- **Detailed Error Messages**: When a test fails, Cypress generates detailed error messages that help in pinpointing the exact issue. It includes information such as expected vs. actual results, making troubleshooting easier.
- **Screenshot and Video Recording**: Cypress can capture screenshots and video recordings during test execution, which provides additional context and aids in debugging.

The test results helped us quickly assess differences between Version 1 and Version 2, and understand why specific tests failed due to intentional modifications in the GUI.

5. Ease of Usage

Cypress is relatively easy to use for both novice and experienced testers:

- **Simple Installation and Setup**: Cypress is easy to install, and it comes with a user-friendly interface for configuring and running tests.
- **Intuitive Syntax**: Cypress's syntax is straightforward, making it easy to write and understand tests. The commands are simple and resemble natural language, which helps reduce the learning curve.
- **Real-time Feedback**: The interactive test runner shows real-time feedback and results, making it easier to see the outcome of tests immediately.
- **Comprehensive Documentation**: Cypress has extensive documentation, which helps users quickly find solutions to common challenges.

Overall, Cypress was user-friendly and intuitive, allowing us to focus more on writing effective tests than on troubleshooting the tool itself.

6. Type of GUI Elements That Can Be Tested

Cypress can interact with and test a wide variety of GUI elements:

- **Standard HTML Elements**: Cypress handles common HTML elements like buttons, inputs, images, and divs, which are frequently found in most web applications.
- **Dynamic Content**: It supports testing dynamic content, such as elements that appear or change based on user interaction or data loading.
- **Element Attributes**: Cypress can check properties like position, size, visibility, and orientation of elements, which was crucial in validating GUI modifications between versions.

- **CSS and Styling**: Cypress can verify CSS styles indirectly by checking the resulting appearance or dimensions of elements.
- **LocalStorage and Cookies**: The tool can interact with browser storage (like LocalStorage and Cookies), making it ideal for applications that rely on stored data for user preferences.

Cypress covered all essential GUI elements of the News Application, enabling us to verify layout consistency, functionality, and navigation flows effectively.

In conclusion, Cypress proved to be an exceptionally effective tool for GUI testing in this project, providing a comprehensive set of features that enabled detailed validation of the News Application's user interface. Its automatic waiting, real-time reloading, and intuitive syntax allowed us to create reliable and reusable test cases that covered all essential aspects of the GUI. The tool's built-in assertions and ability to interact directly with DOM elements made it easy to validate element existence, positioning, orientation, and functionality across different versions of the application. This comprehensive coverage allowed us to assess changes made in Version 2, with clear pass/fail indicators and detailed error messages that simplified troubleshooting and analysis.

Additionally, Cypress's ease of use and robust reporting capabilities, including screenshots and video recording, provided valuable insights into the application's behavior under test. The interactive test runner and modularized test structure facilitated efficient regression testing, enabling quick identification of issues while maintaining readability and consistency in the test suite. While Cypress may have limitations with more complex multi-tab interactions, it was well-suited for the single-page focus of this project. Overall, Cypress proved to be a powerful, user-friendly solution for ensuring GUI integrity, making it an excellent choice for this type of web application testing.

CONCLUSION

This assignment involved developing, modifying, and testing a GUI-based News Application, highlighting essential front-end development and GUI testing practices. Version 1 established a user-friendly interface using HTML, CSS, and JavaScript, incorporating features like dark mode, a zoom slider, and page navigation. Version 2 introduced controlled modifications to element positions, sizes, orientations, and navigation flow, simulating real-world adjustments and setting the stage for regression testing.

Using Cypress, we conducted detailed tests to verify that Version 2 changes did not compromise functionality or usability. Cypress's capabilities, including built-in assertions, automatic waiting, and detailed reporting, enabled us to identify test failures resulting from these adjustments, reinforcing the value of automated regression testing. This project demonstrated how tools like Cypress help ensure consistent application quality, offering critical insights into the development-testing cycle and the importance of thorough GUI testing in maintaining a seamless user experience.

REFERENCES

1. https://docs.cypress.io/app/get-started/why-cypress
2. https://www.w3schools.com/html/html_intro.asp
3. https://www.w3schools.com/css/
4. https://developer.mozilla.org/en-US/docs/Web/JavaScript
5. https://www.foxnews.com/