**Report PennyFlow Frontend – Sprint 3 Detailed**

**1. Overview and Objectives**

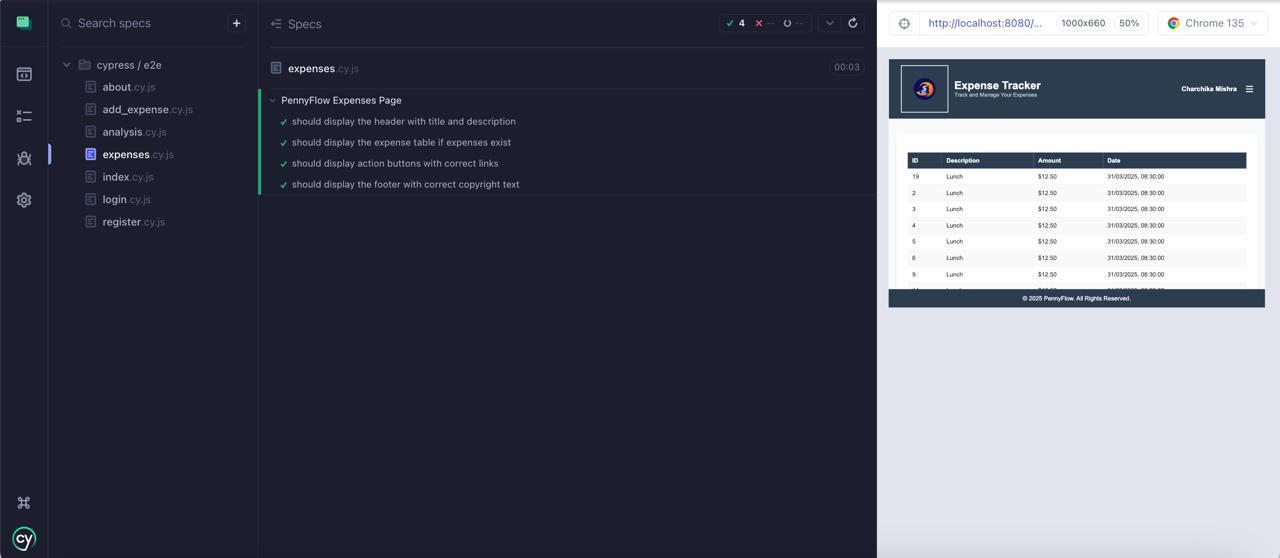
**Introduction:**In Sprint 3, the front-end team concentrated on enhancing the user interface and overall user experience for the PennyFlow application. The primary goals were to:

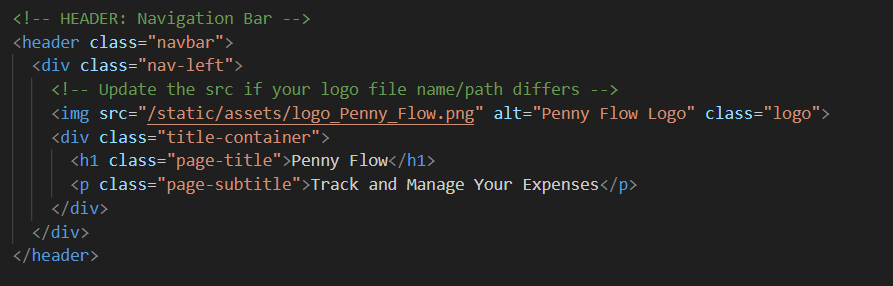
* Develop a Dynamic Expense Analysis Dashboard:  
  Create an interactive dashboard that visualizes user expenses using charts and graphs. This dashboard supports informed decision making by providing real-time data insights.
* Refine UI Components and Navigation:  
  Improve the overall design and functionality of key pages such as Login, Registration, and Expense Management. Implement a persistent navigation bar to facilitate smooth transitions between application sections.
* Strengthen API Integration:  
  Enhance communication with the backend by optimizing API calls for authentication, expense retrieval, and data analytics. Ensure that data flows seamlessly from the back-end endpoints to the user interface.
* Increase Unit Testing Coverage and Reliability:  
  Implement comprehensive unit tests for all components. Validate form inputs, API call responses, component rendering, and routing to ensure high reliability and maintainability.

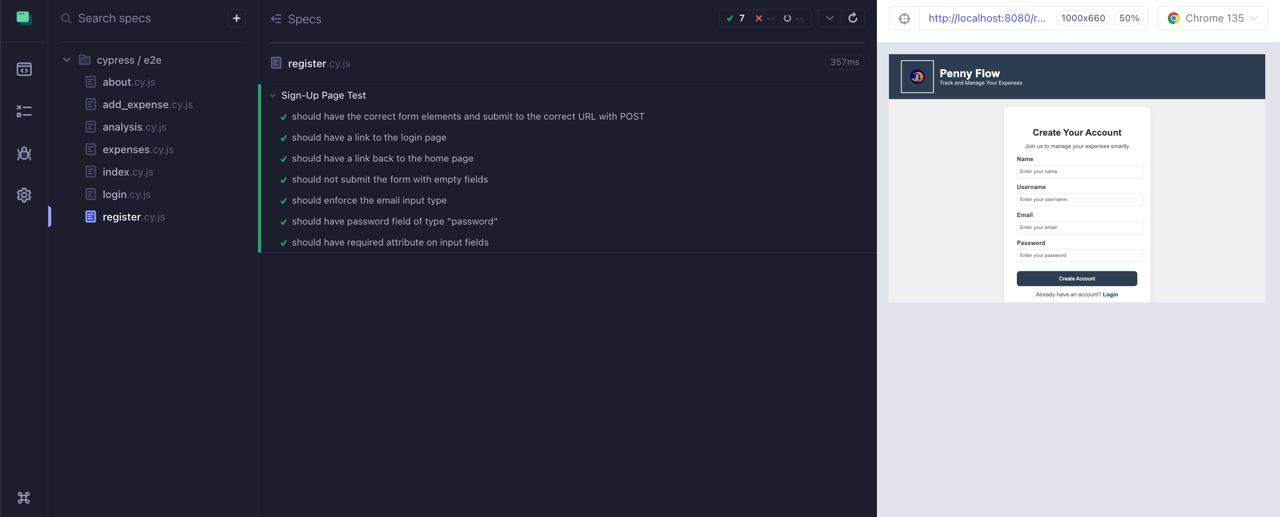
2. User Interface Enhancements and New Features

2.1 Enhanced Pages and Visual Improvements

Expense Analysis Dashboard:

* Description:  
  The newly developed Expense Analysis Dashboard provides an interactive view of the user's spending patterns. It displays charts (such as pie charts and bar graphs) derived from aggregated expense data, enabling users to see trends and breakdowns by category.
* Screenshot Instructions:
  + Insert the following image(s):
* 
  + Caption:
    - “Figure 1: Expense Analysis Dashboard displaying interactive expense charts.”
* **2.2 Navigation and Form Layouts**
* **Description:**  
  The project now features a persistent navigation bar ensuring users can easily move between sections, and standardizes forms for authentication and expense entry.

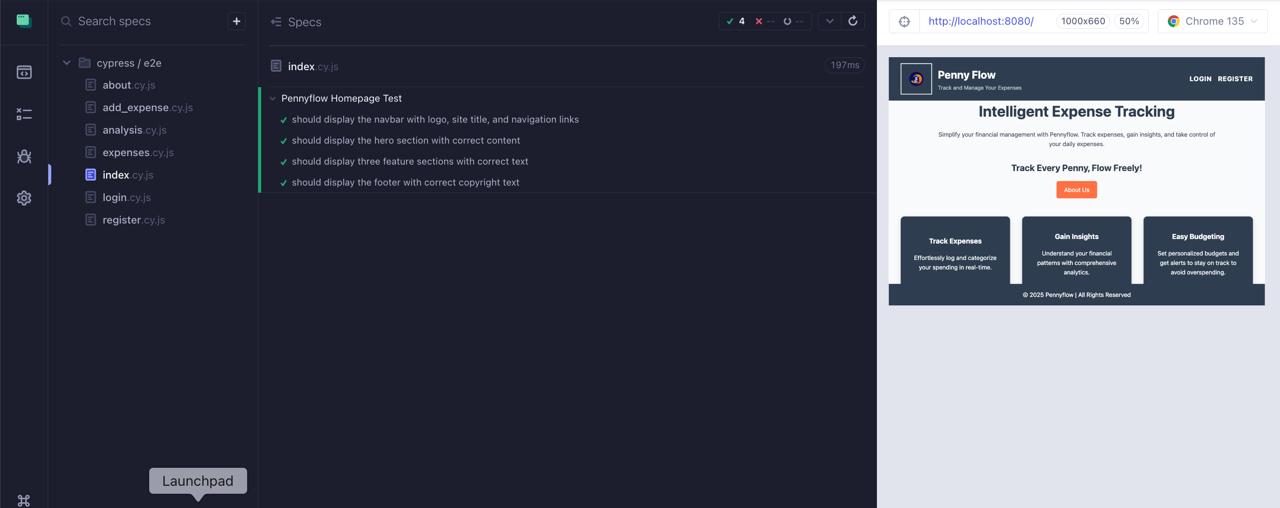


* Refined Navigation and Form Layouts:
* Description:  
  A consistent and user-friendly layout has been implemented across key pages. The persistent navigation bar now appears on every page (e.g., Dashboard, Add Expense, Analysis, About Us), and all forms (Login, Registration, Expense Entry) have been standardized. Inline validations ensure that users receive immediate feedback if they enter invalid data.
* 

**Caption:**

* “Figure 2: Persistent Navigation Bar and Login Form with in-line validations.”

2.3 -Cypress test from index.cy.js



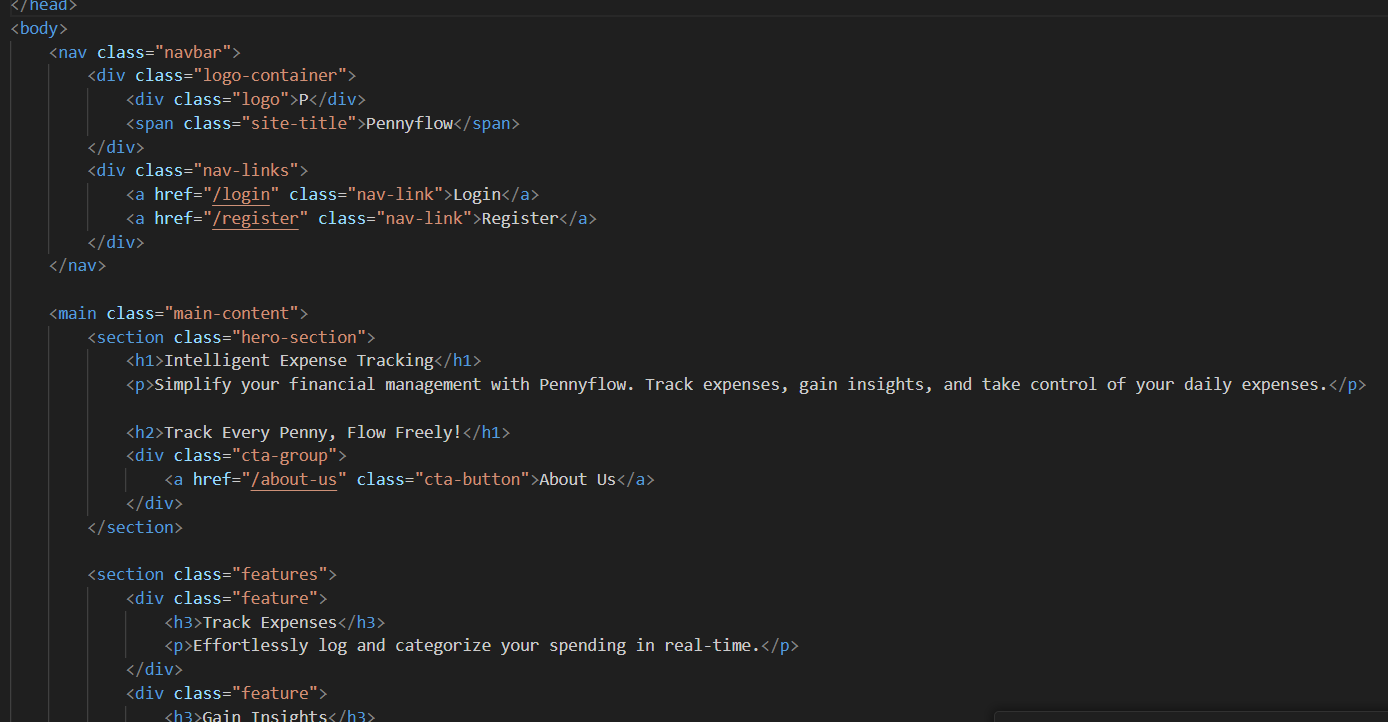
**Caption:** “Figure 4: Cypress test from index.cy.js showing the routing configuration and successful navigation between pages.”

**3. Architecture and Component Structure**

**Overview:**  
The front-end is built using React, with React Router managing client-side navigation. The application is organized into modular, reusable components, ensuring scalability and maintainability.

**3.1 Application Bootstrapping and Routing**

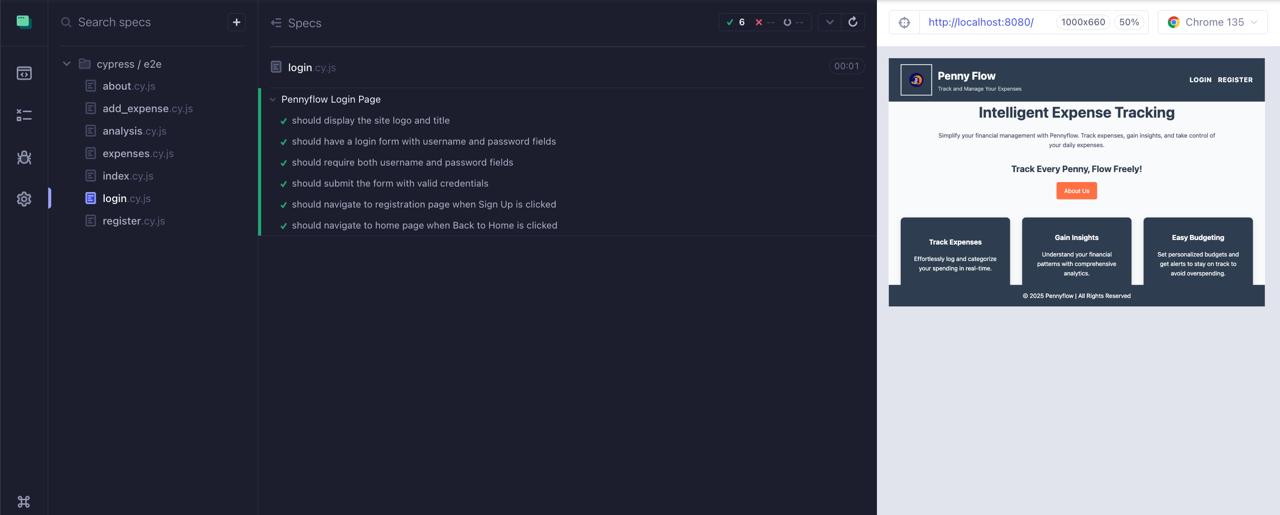
* **Entry Point (index.html):**  
  The main entry file initializes the React application and defines routing logic. This setup allows users to navigate seamlessly between different pages, such as Login, Registration, Expenses, Analysis, and About Us.



**Caption:**

* “ Implementation of the login function demonstrating API call integration.”

Cypress test for login.cy.js/register.cy.js



**4. Frontend Unit Testing and Test Cases**

**Overview:**  
Robust unit tests ensure that components render correctly, forms validate inputs, and API interactions update state as expected. We use Jest and React Testing Library to run these tests.

**4.1 Component Rendering and Interaction Tests**

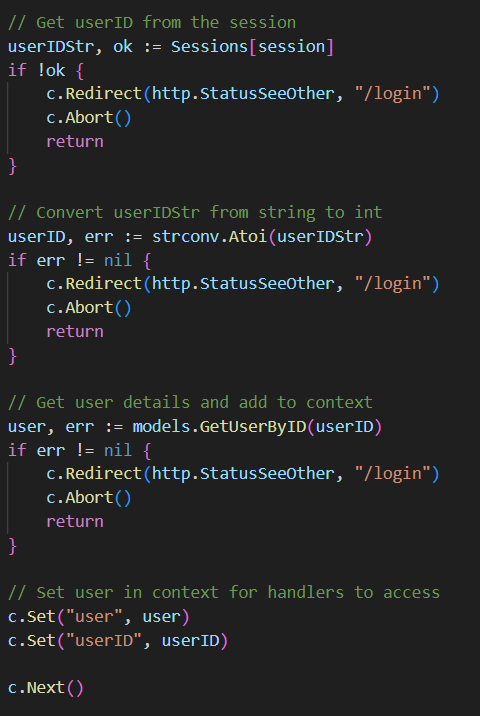
* **Test Case Example – Login Form Validation:**



**5. Integration with Backend APIs**

**Overview:**  
The front-end communicates with backend services via asynchronous HTTP requests using Fetch. This integration enables functions such as user authentication, expense data manipulation, and dynamic dashboard updates.

5.1 Authentication and Expense Data API Calls



**6. Front-End Unit Testing and Cypress Test Cases**

**Overview:**  
Our testing strategy utilizes Cypress to verify that all UI components behave as expected. We have separate Cypress test files for different aspects of the application:

* **about.cy.js:** Tests the About page for proper content and layout.
* **add\_expence.cy.js:** Validates the expense entry form functionality and its validations.
* **analysis.cy.js:** Confirms that the Expense Analysis Dashboard correctly displays and updates data.
* **expence.cy.js:** Verifies that the Expenses page renders the expense list accurately and handles CRUD operations.
* **index.cy.js:** Ensures the overall routing and navigation work correctly.
* **login.cy.js and register.cy.js:** Test authentication flows, including error cases and successful login/registration.