**Test Execution document**

1. **Test cases**

**Test conditions**

TC-FE-1 : UI components for creating, updating, and deleting users, workouts, exercises, and equipment must be rendered correctly and be interactive within the React app.

TC-FE-2 : Network connectivity between the frontend and the backend API should be stable for CRUD operations.

TC-FE-3 : The frontend application must correctly handle and display responses from the API, including success and error messages.

TC-FE-4 : Input validation as implemented on the frontend must adhere to the Joi validation schemas, providing immediate feedback to the user within the UI.

TC-FE-5 : The application should correctly display modals for creating and updating entities, with pre-filled data for updates.

TC-FE-6 : Error handling in the frontend must correctly interpret and display error messages or statuses received from the API.

TC-FE-7 : The application must update the UI in real-time to reflect changes made (e.g., adding, updating, or deleting users, workouts, exercises, equipment).

**Test Case 1: Creating a New User**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC1 |
| **Test Conditions** | The user creation form is accessible in the UI. |
| **Test Basis** | User creation functionality as per application requirements. |
| **Execution Pre-conditions** | Application is loaded; user is on the users page. |
| **Input Values** | Valid and Invalid user details (name, email, password, age). |
| **Actions** | Open user creation modal; Fill in the form; Submit the form. |
| **Expected Results** | For valid inputs: User is created, success message; For invalid inputs: Error messages, no user creation. |
| **Execution Post-conditions** | UI returns to initial state; New user appears in list if created. |

**Test Case 2: Creating a New Workout**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC2 |
| **Test Conditions** | The workout creation form is accessible in the UI. |
| **Test Basis** | Workout creation functionality as per application requirements. |
| **Execution Pre-conditions** | Application is loaded; user is on the workout page. |
| **Input Values** | Valid and Invalid workout details (name, description, difficulty). |
| **Actions** | Open workout creation modal; Fill in the form; Submit the form. |
| **Expected Results** | For valid inputs: Workout is created, success message; For invalid inputs: Error messages, no creation. |
| **Execution Post-conditions** | UI returns to initial state; New workout appears in list if created. |

**Test Case 3: Creating New Equipment**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC3 |
| **Test Conditions** | The equipment creation form is accessible in the UI. |
| **Test Basis** | Equipment creation functionality as per application requirements. |
| **Execution Pre-conditions** | Application is loaded; user is on the equipment page. |
| **Input Values** | Valid and Invalid equipment details (name, description, availability). |
| **Actions** | Open equipment creation modal; Fill in the form; Submit the form. |
| **Expected Results** | For valid inputs: Equipment is created, success message; For invalid inputs: Error messages, no creation. |
| **Execution Post-conditions** | UI returns to initial state; New equipment appears in list if created. |

**Test Case 4: Creating a New Exercise**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC4 |
| **Test Conditions** | The exercise creation form is accessible in the UI. |
| **Test Basis** | Exercise creation functionality as per application requirements. |
| **Execution Pre-conditions** | Application is loaded; user is on the exercise page. |
| **Input Values** | Valid and Invalid exercise details (name, sets, reps, weight, description). |
| **Actions** | Open exercise creation modal; Fill in the form; Submit the form. |
| **Expected Results** | For valid inputs: Exercise is created, success message; For invalid inputs: Error messages, no creation. |
| **Execution Post-conditions** | UI returns to initial state; New exercise appears in list if created. |

**Test Case 5: Updating an Existing User**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC5 |
| **Test Conditions** | The user update form is accessible in the UI. |
| **Test Basis** | User update functionality as per application requirements. |
| **Execution Pre-conditions** | An existing user is selected for update. |
| **Input Values** | Valid and Invalid user details (name, email, password, age). |
| **Actions** | Open user update modal; Modify the form fields; Submit the form. |
| **Expected Results** | For valid inputs: User is updated, success message; For invalid inputs: Error messages, no update. |
| **Execution Post-conditions** | UI reflects updated user details or remains unchanged for invalid inputs. |

**Test Case 6: Updating an Existing Workout**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC6 |
| **Test Conditions** | The workout update form is accessible in the UI. |
| **Test Basis** | Workout update functionality as per application requirements. |
| **Execution Pre-conditions** | An existing workout is selected for update. |
| **Input Values** | Valid and Invalid workout details (name, description, difficulty). |
| **Actions** | Open workout update modal; Modify the form fields; Submit the form. |
| **Expected Results** | For valid inputs: Workout is updated, success message; For invalid inputs: Error messages, no update. |
| **Execution Post-conditions** | UI reflects updated workout details or remains unchanged for invalid inputs. |

**Test Case 7: Updating Existing Equipment**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC7 |
| **Test Conditions** | The equipment update form is accessible in the UI. |
| **Test Basis** | Equipment update functionality as per application requirements. |
| **Execution Pre-conditions** | An existing piece of equipment is selected for update. |
| **Input Values** | Valid and Invalid equipment details (name, description, availability). |
| **Actions** | Open equipment update modal; Modify the form fields; Submit the form. |
| **Expected Results** | For valid inputs: Equipment is updated, success message; For invalid inputs: Error messages, no update. |
| **Execution Post-conditions** | UI reflects updated equipment details or remains unchanged for invalid inputs. |

**Test Case 8: Updating an Existing Exercise**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC8 |
| **Test Conditions** | The exercise update form is accessible in the UI. |
| **Test Basis** | Exercise update functionality as per application requirements. |
| **Execution Pre-conditions** | An existing exercise is selected for update. |
| **Input Values** | Valid and Invalid exercise details (name, sets, reps, weight, description). |
| **Actions** | Open exercise update modal; Modify the form fields; Submit the form. |
| **Expected Results** | For valid inputs: Exercise is updated, success message; For invalid inputs: Error messages, no update. |
| **Execution Post-conditions** | UI reflects updated exercise details or remains unchanged for invalid inputs. |

**Test Case 9: Deleting an Existing User**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC9 |
| **Test Conditions** | User deletion option is accessible in the UI. |
| **Test Basis** | User deletion functionality as per application requirements. |
| **Execution Pre-conditions** | An existing user is selected for deletion. |
| **Input Values** | User ID or other identifier for the user being deleted. |
| **Actions** | Select and confirm the delete option for the user. |
| **Expected Results** | User is removed from the UI; Success message displayed. |
| **Execution Post-conditions** | UI no longer displays the deleted user. |

**Test Case 10: Deleting an Existing Workout**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC10 |
| **Test Conditions** | Workout deletion option is accessible in the UI. |
| **Test Basis** | Workout deletion functionality as per application requirements. |
| **Execution Pre-conditions** | An existing workout is selected for deletion. |
| **Input Values** | Workout ID or other identifier for the workout being deleted. |
| **Actions** | Select and confirm the delete option for the workout. |
| **Expected Results** | Workout is removed from the UI; Success message displayed. |
| **Execution Post-conditions** | UI no longer displays the deleted workout. |

**Test Case 11: Deleting Existing Equipment**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC11 |
| **Test Conditions** | Equipment deletion option is accessible in the UI. |
| **Test Basis** | Equipment deletion functionality as per application requirements. |
| **Execution Pre-conditions** | An existing piece of equipment is selected for deletion. |
| **Input Values** | Equipment ID or other identifier for the equipment being deleted. |
| **Actions** | Select and confirm the delete option for the equipment. |
| **Expected Results** | Equipment is removed from the UI; Success message displayed. |
| **Execution Post-conditions** | UI no longer displays the deleted equipment. |

**Test Case 12: Deleting an Existing Exercise**

|  |  |
| --- | --- |
| **Test Case Identifier** | TC-FE-TC12 |
| **Test Conditions** | Exercise deletion option is accessible in the UI. |
| **Test Basis** | Exercise deletion functionality as per application requirements. |
| **Execution Pre-conditions** | An existing exercise is selected for deletion. |
| **Input Values** | Exercise ID or other identifier for the exercise being deleted. |
| **Actions** | Select and confirm the delete option for the exercise. |
| **Expected Results** | Exercise is removed from the UI; Success message displayed. |
| **Execution Post-conditions** | UI no longer displays the deleted exercise. |

1. **Decision tables**
2. Creating a New Entity (User, Workout, Exercise, Equipment)

|  |  |  |  |
| --- | --- | --- | --- |
| **Conditions** | **Valid Input** | **Invalid Name** | **Invalid Specific Field** |
| Name Validity | Yes | No | Yes |
| Specific Field Validity | Yes | Yes | No |
| **Actions** |  |  |  |
| Open Creation Modal | Execute | Execute | Execute |
| Fill Form with Inputs | Execute | Execute | Execute |
| Submit Form | Execute | Execute | Execute |
| **Results** |  |  |  |
| Entity Created | Yes | No | No |
| Success Message Displayed | Yes | No | No |
| Error Message Displayed | No | Yes | Yes |

1. Updating an Existing Entity (User, Workout, Exercise, Equipment)

|  |  |  |  |
| --- | --- | --- | --- |
| **Conditions** | **Valid Input** | **Invalid Name** | **Invalid Specific Field** |
| Name Validity | Yes | No | Yes |
| Specific Field Validity | Yes | Yes | No |
| **Actions** |  |  |  |
| Open Update Modal | Execute | Execute | Execute |
| Fill Form with New Inputs | Execute | Execute | Execute |
| Submit Update Form | Execute | Execute | Execute |
| **Results** |  |  |  |
| Entity Updated | Yes | No | No |
| Success Message Displayed | Yes | No | No |
| Error Message Displayed | No | Yes | Yes |

1. Deleting an Existing Entity (User, Workout, Exercise, Equipment)

|  |  |  |
| --- | --- | --- |
| **Conditions** | **Confirmation Given** | **Confirmation Denied** |
| Entity Selected | Yes | Yes |
| **Actions** |  |  |
| Open Delete Confirmation | Execute | Execute |
| Confirm Deletion | Yes | No |
| **Results** |  |  |
| Entity Deleted | Yes | No |
| Success Message Displayed | Yes | No |
| UI Updated to Reflect Change | Yes | No |

1. **Equivalence partitioning and Boundary Value Analysis**

Creating a New User and Updating an Existing User

|  |  |  |
| --- | --- | --- |
| **Field** | **Equivalence Partition** | **Boundary Values** |
| Name | VP1: Valid Names (alphabetic) | BVA1: Min length (e.g., "A"), Max length |
| IP1: Invalid Names (empty, numeric, special chars) | BVA2: Empty string, Numeric/Special chars (e.g., "123", "@John") |  |
| Email | VP2: Valid Emails | BVA3: Valid format (e.g., "a@b.com") |
| IP2: Invalid Emails (format, empty) | BVA4: Empty string, Missing '@' (e.g., "user", "user@") |  |
| Password | VP3: Valid Passwords (min length, chars) | BVA5: Min length (e.g., "Pass@12345") |
| IP3: Invalid Passwords (length, format) | BVA6: Below min length (e.g., "Pass123"), No special chars |  |
| Age | VP4: Valid Ages (18-90) | BVA7: Min age (18), Max age (90) |
| IP4: Invalid Ages (below 18, above 90) | BVA8: Below min (17), Above max (91) |  |

Creating a New Workout and Updating an Existing Workout

|  |  |  |
| --- | --- | --- |
| **Field** | **Equivalence Partition** | **Boundary Values** |
| Name | VP5: Valid Names (alphabetic) | BVA9: Min length (e.g., "A"), Max length |
| IP5: Invalid Names (empty, numeric, special chars) | BVA10: Empty string, Numeric/Special chars (e.g., "123", "@Workout") |  |
| Difficulty | VP6: Valid Difficulty (1-10) | BVA11: Min (1), Max (10) |
| IP6: Invalid Difficulty (below 1, above 10) | BVA12: Below min (0), Above max (11) |  |

Creating a New Exercise and Updating an Existing Exercise

|  |  |  |
| --- | --- | --- |
| **Field** | **Equivalence Partition** | **Boundary Values** |
| Name | VP7: Valid Names (alphabetic) | BVA13: Min length (e.g., "A"), Max length |
| IP7: Invalid Names (empty, numeric, special chars) | BVA14: Empty string, Numeric/Special chars (e.g., "123", "@Exercise") |  |
| Sets | VP8: Valid Sets (2-5) | BVA15: Min (2), Max (5) |
| IP8: Invalid Sets (below 2, above 5) | BVA16: Below min (1), Above max (6) |  |
| Reps | VP9: Valid Reps (5-12) | BVA17: Min (5), Max (12) |
| IP9: Invalid Reps (below 5, above 12) | BVA18: Below min (4), Above max (13) |  |
| Weight | VP10: Valid Weights (numeric, "kg") | BVA19: Example valid weights (e.g., "10kg") |
| IP10: Invalid Weights (non-numeric, format) | BVA20: Non-numeric (e.g., "heavy"), Missing unit |  |

Creating New Equipment and Updating Existing Equipment

|  |  |  |
| --- | --- | --- |
| **Field** | **Equivalence Partition** | **Boundary Values** |
| Name | VP11: Valid Names (alphabetic) | BVA21: Min length (e.g., "A"), Max length |
| IP11: Invalid Names (empty, numeric, special chars) | BVA22: Empty string, Numeric/Special chars (e.g., "123", "@Equip") |  |
| Description | VP12: Valid Descriptions (text) | BVA23: Min length (e.g., "Good"), Max length |
| IP12: Invalid Descriptions (empty) | BVA24: Empty string |  |
| Availability | VP13: Valid Availability (boolean) | BVA25: True, False |
| IP13: Invalid Availability (non-boolean) | BVA26: Non-boolean values (e.g., "yes", "no") |  |

1. **State transition diagrams**

A diagram of a diagram

Description automatically generated

1. **State transition tables**

States:

Not Created: Initial state before the entity is created.

Created: The entity is successfully created and exists in the system.

Updated: The entity has been modified at least once after creation.

Deleted: The entity has been removed from the system.

Events:

Attempt to Create: An event where the user or system initiates the creation process for the entity.

Valid Update: An update event with valid data.

Invalid Update: An update event with invalid data; the system rejects the update, and the entity remains in its previous state.

Deletion Request: User or system event to remove the entity.

State Transition Table for Users

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Event/Condition** | **Action Taken** | **Next State** |
| Not Created | Attempt to Create | Create User | Created |
| Not Created | Attempt to Create | Fail to Create User | Not Created |
| Created | Valid Update | Update User Information | Updated |
| Created | Invalid Update | Fail to Update User | Created |
| Updated | Valid Update | Update User Information | Updated |
| Updated | Invalid Update | Fail to Update User | Updated |
| Created/Updated | Deletion Request | Delete User | Deleted |
| Created/Updated | Deletion Request (Failed) | N/A | Created/Updated |

State Transition Table for Workouts

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Event/Condition** | **Action Taken** | **Next State** |
| Not Created | Attempt to Create | Create Workout | Created |
| Not Created | Attempt to Create | Fail to Create Workout | Not Created |
| Created | Valid Update | Update Workout Information | Updated |
| Created | Invalid Update | Fail to Update Workout | Created |
| Updated | Valid Update | Update Workout Information | Updated |
| Updated | Invalid Update | Fail to Update Workout | Updated |
| Created/Updated | Deletion Request | Delete Workout | Deleted |
| Created/Updated | Deletion Request (Failed) | N/A | Created/Updated |

State Transition Table for Exercises

|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Event/Condition** | **Action Taken** | **Next State** |
| Not Created | Attempt to Create | Create Exercise | Created |
| Not Created | Attempt to Create | Fail to Create Exercise | Not Created |
| Created | Valid Update | Update Exercise Information | Updated |
| Created | Invalid Update | Fail to Update Exercise | Created |
| Updated | Valid Update | Update Exercise Information | Updated |
| Updated | Invalid Update | Fail to Update Exercise | Updated |
| Created/Updated | Deletion Request | Delete Exercise | Deleted |
| Created/Updated | Deletion Request (Failed) | N/A | Created/Updated (Deletion Failed) |

State Transition Table for Equipment

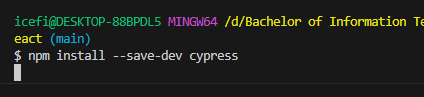
|  |  |  |  |
| --- | --- | --- | --- |
| **Current State** | **Event/Condition** | **Action Taken** | **Next State** |
| Not Created | Attempt to Create | Create Equipment | Created |
| Not Created | Attempt to Create | Fail to Create Equipment | Not Created |
| Created | Valid Update | Update Equipment Info | Updated |
| Created | Invalid Update | Fail to Update Equipment | Created |
| Updated | Valid Update | Update Equipment Info | Updated |
| Updated | Invalid Update | Fail to Update Equipment | Updated |
| Created/Updated | Deletion Request | Delete Equipment | Deleted |
| Created/Updated | Deletion Request (Failed) | N/A | Created/Updated (Deletion Failed) |

1. **Unit testing**

Using cypress because it’s the software that most accurately represents user experience.

Step 1. Setting up test environment and scripts

Install cypress



Create script in the package.json file in order for the tests to run:

A screen shot of a computer

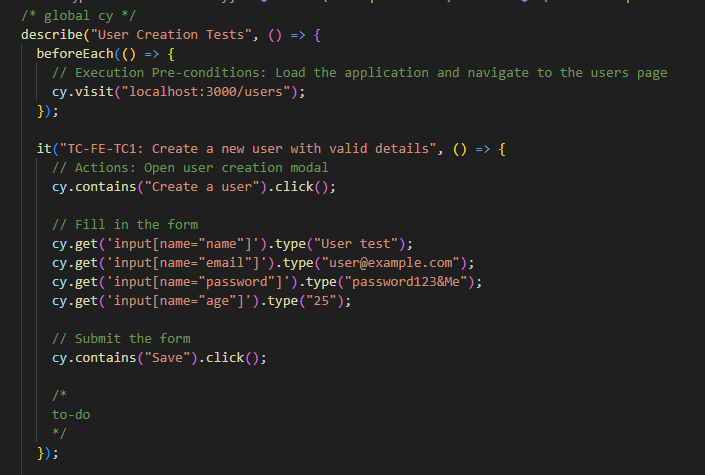
Description automatically generated

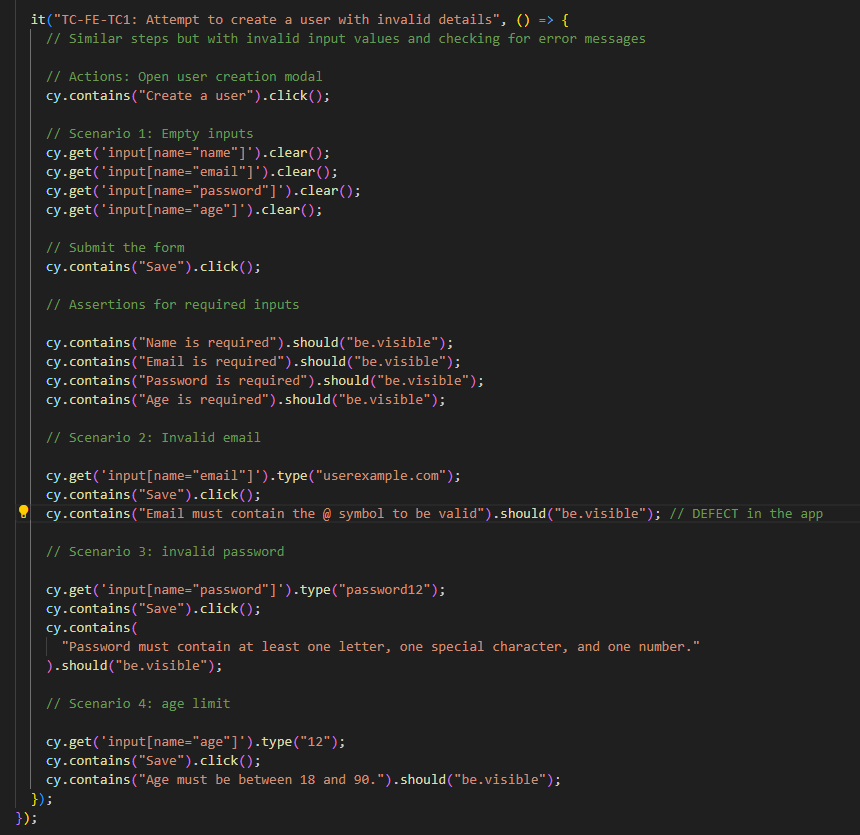
Step 2. **Integration Testing:** Code to Create test suites – for the End 2 end tests, by planning out the code structure based on test cases

I also used **regression testing** while trying to understand the basic syntax of Cypress testing in a react context by getting all of the create/update/delete data into separate sections, so that functionality doesn’t break as a lot of the code is very similar.

***users.cy.js***

Create





Update

A computer screen shot of a program code

Description automatically generated

A screen shot of a computer program

Description automatically generated

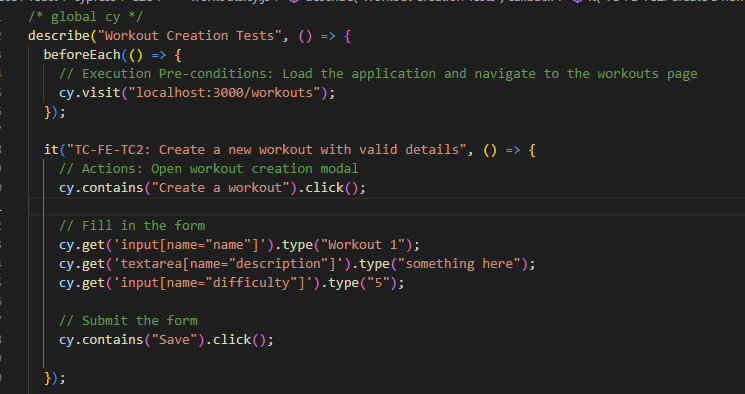
Delete

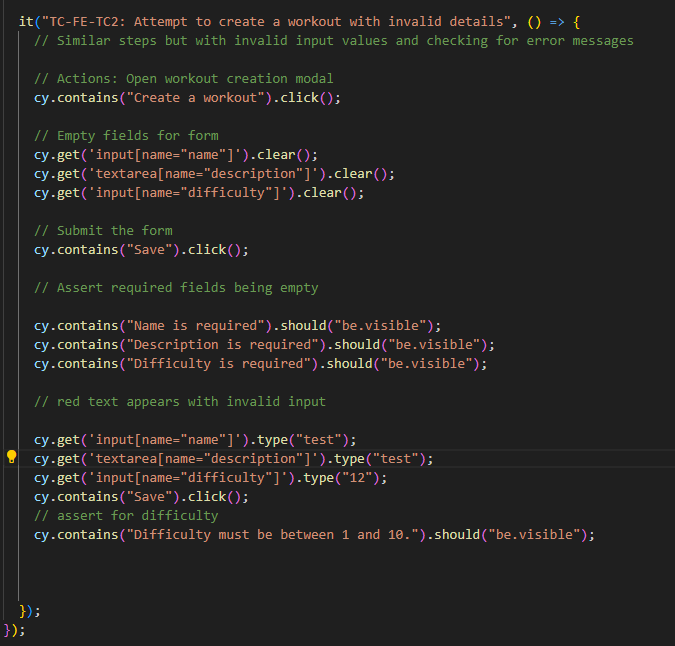
A screenshot of a computer program

Description automatically generated

***workouts.cy.js***

Create





Update

A screenshot of a computer program

Description automatically generated

Delete

A computer screen with text

Description automatically generated

***equipment.cy.js***

Create

A screenshot of a computer program

Description automatically generated

Update

A computer screen shot of a program code

Description automatically generated

Delete

A computer screen with colorful text

Description automatically generated

***exercise.cy.js***

Create

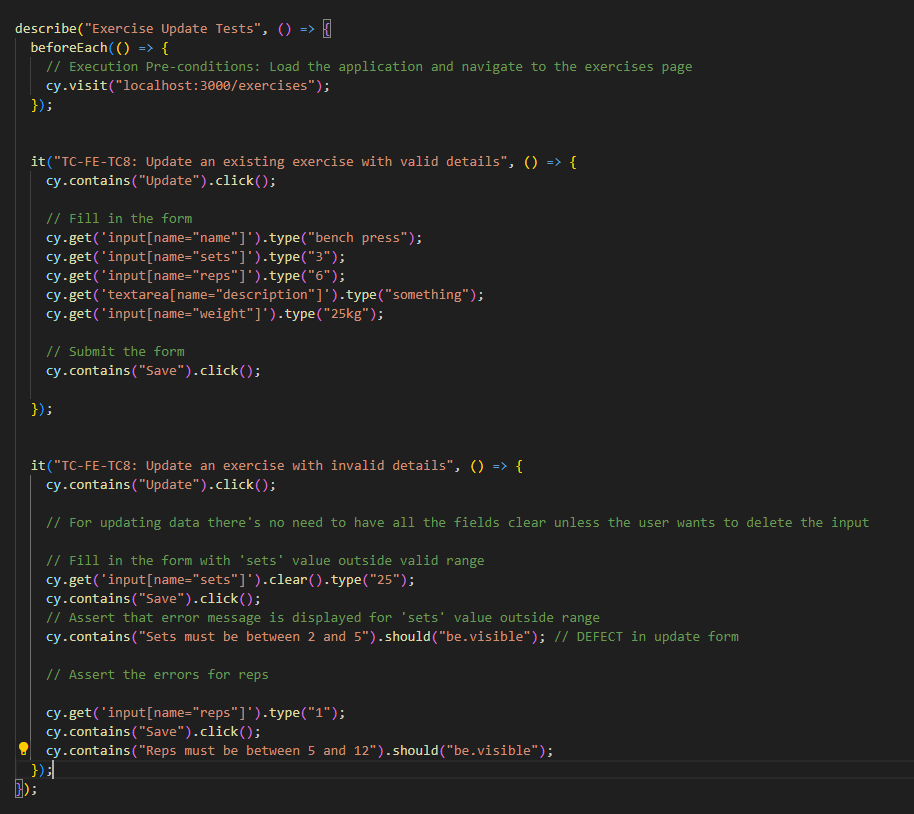
A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program

Description automatically generated

Update



Delete

A screenshot of a computer program

Description automatically generated

Step 3. Run tests and Check which tests pass and fail

Note: As there are 24 tests, with 12 test cases, 12 valid inputs, and 12 invalid inputs with various assertion types. There are 6 tests per file.

When running the “npx cypress run” command it automatically screenshots the defects for you:

A screenshot of a computer program

Description automatically generated

**Results:**

Based on the total number of tests run:

A screenshot of a computer

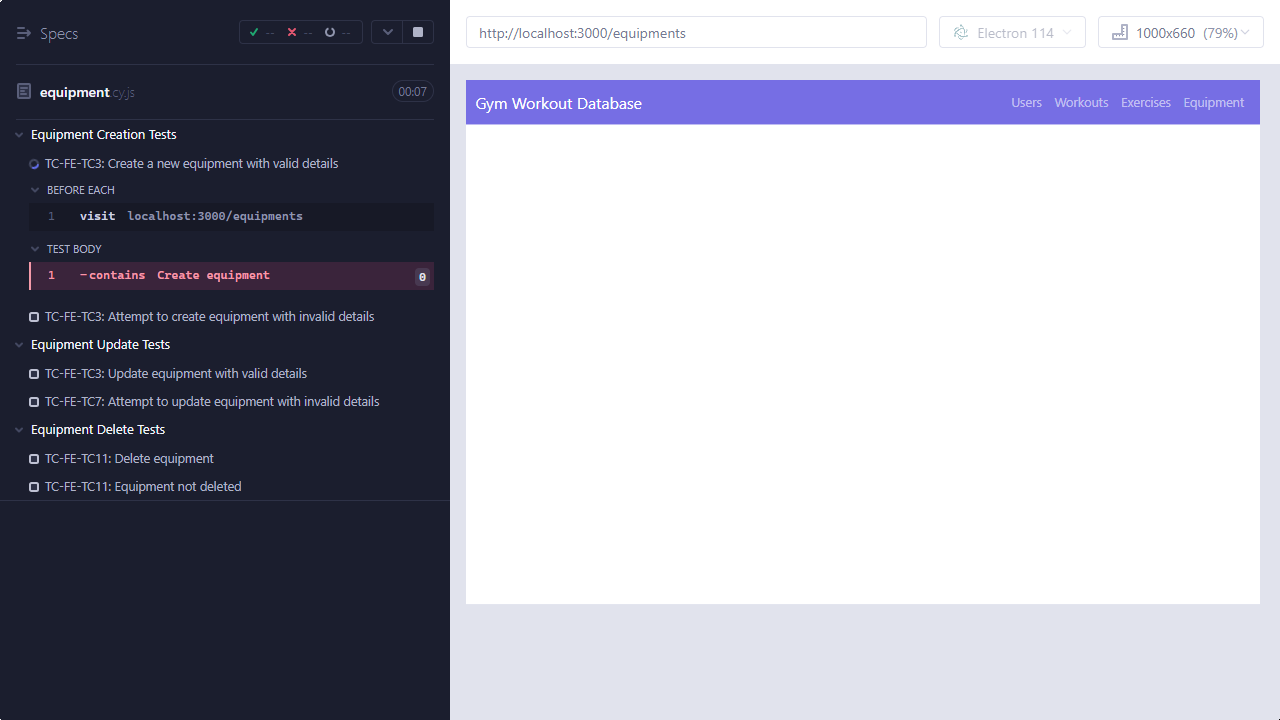
Description automatically generated

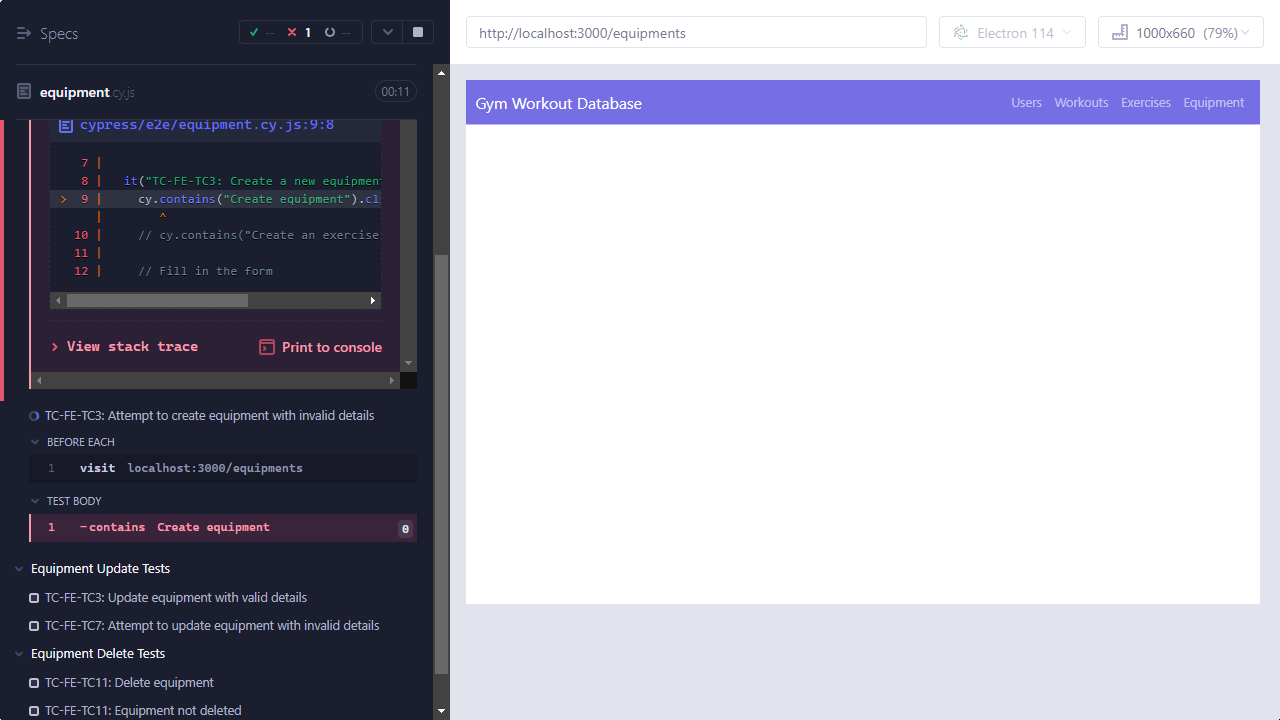
1. **Defects found and other testing types for defects**

Overall there were 9 failed tests, with the critical defect being the create equipment test case (both valid and invalid)that was initially low risk inside the test plan, which led to all the other tests failing as there was no data to update or delete.

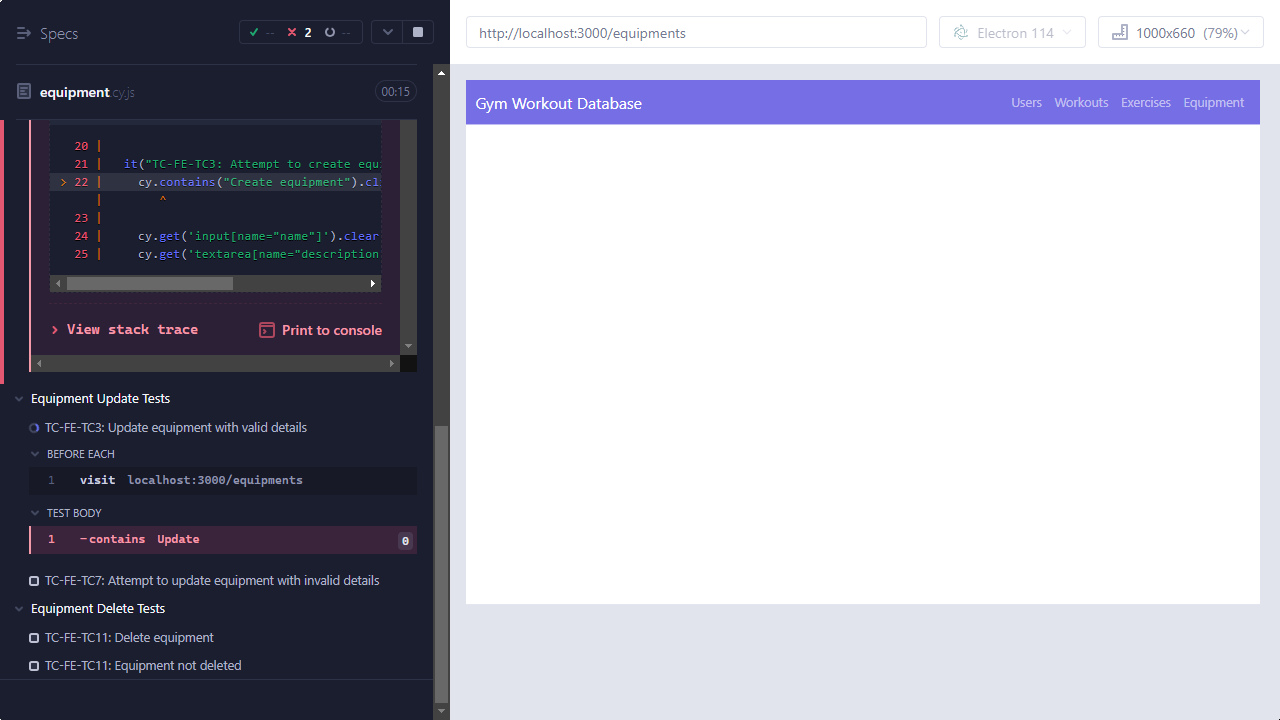
When finding out what was causing the defects, I used the technique of **Exploratory Testing** to find out what exactly is causing them within the app itself without using automation

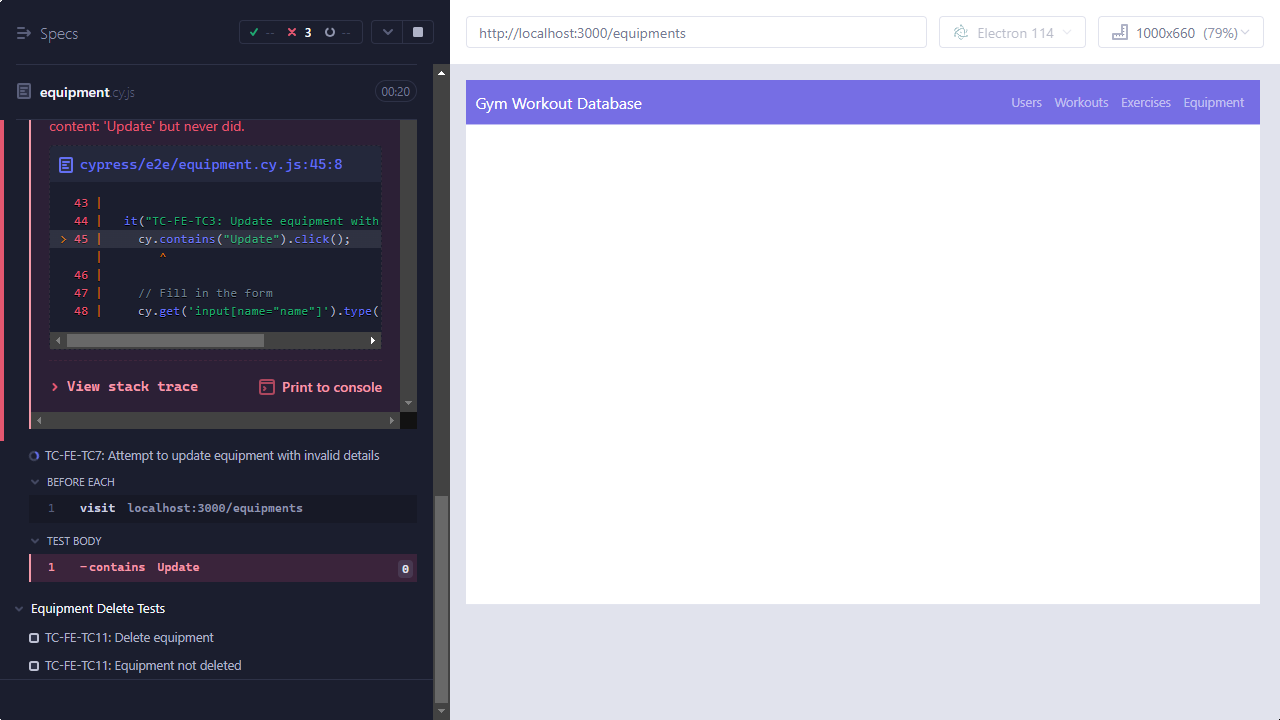
In equipments.cy.js in order of create, update and delete

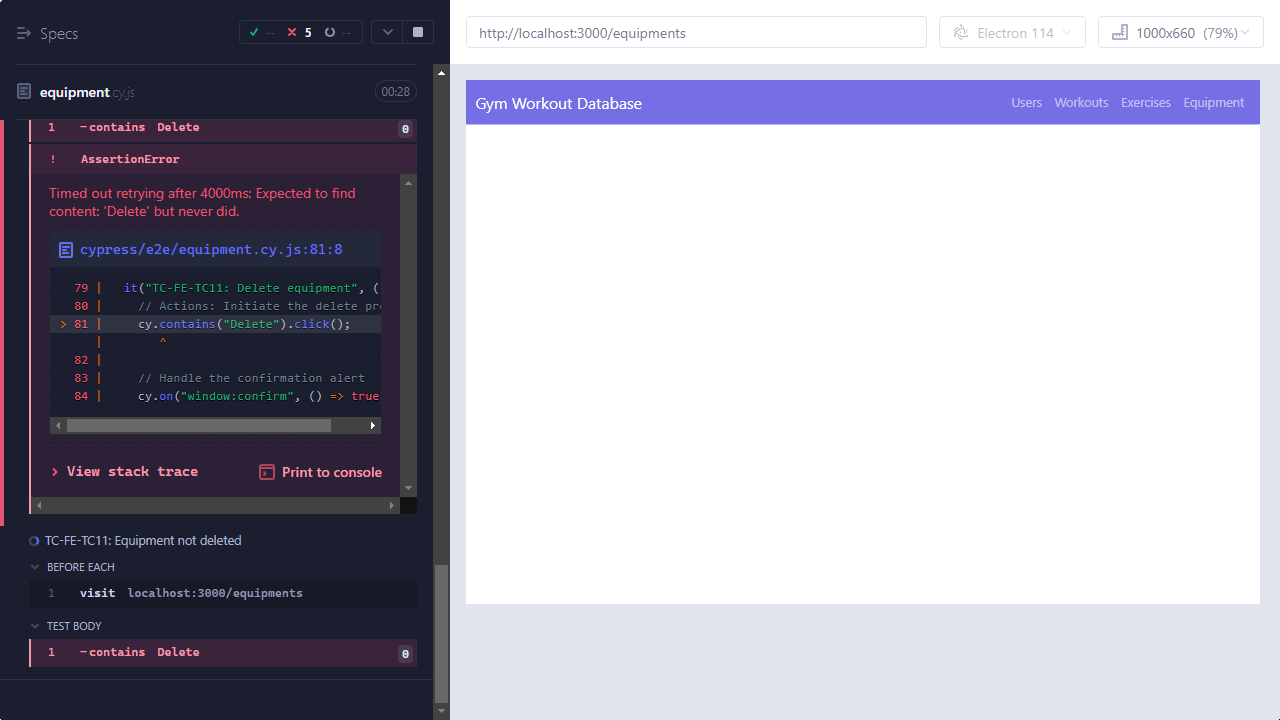
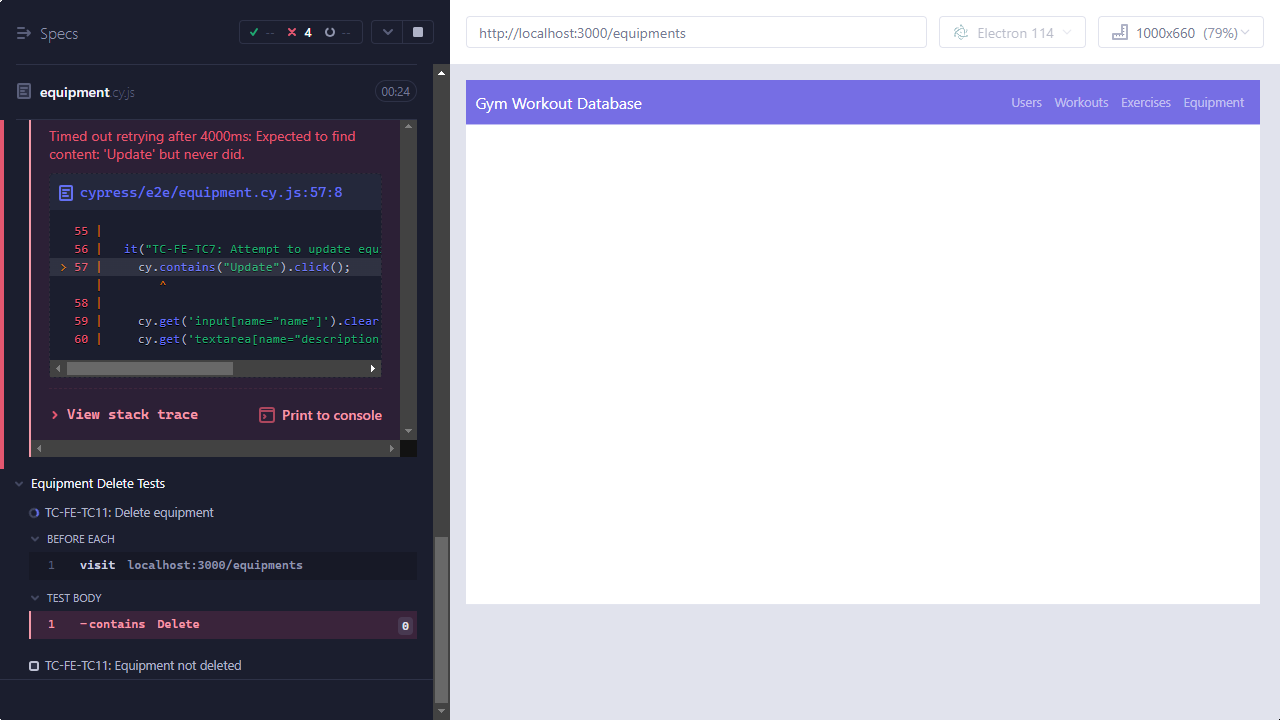




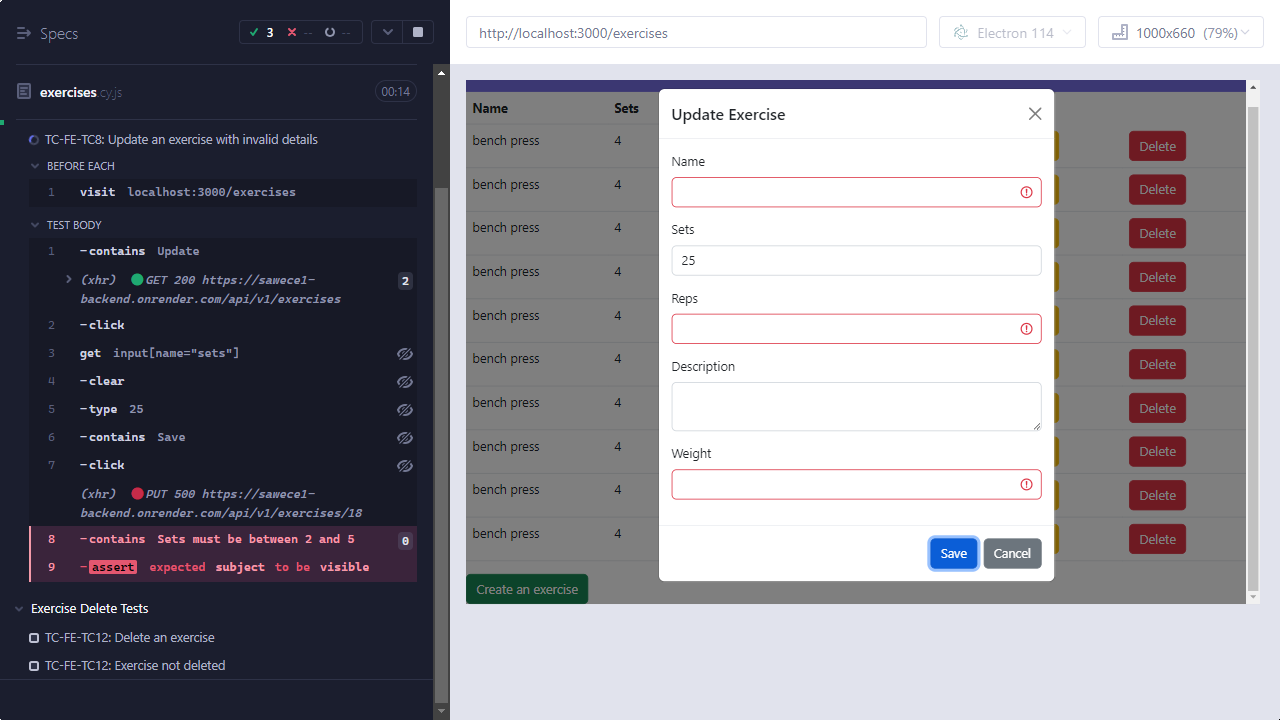
For this defect, the “Create equipment” button was instead a “Create an exercise” button.





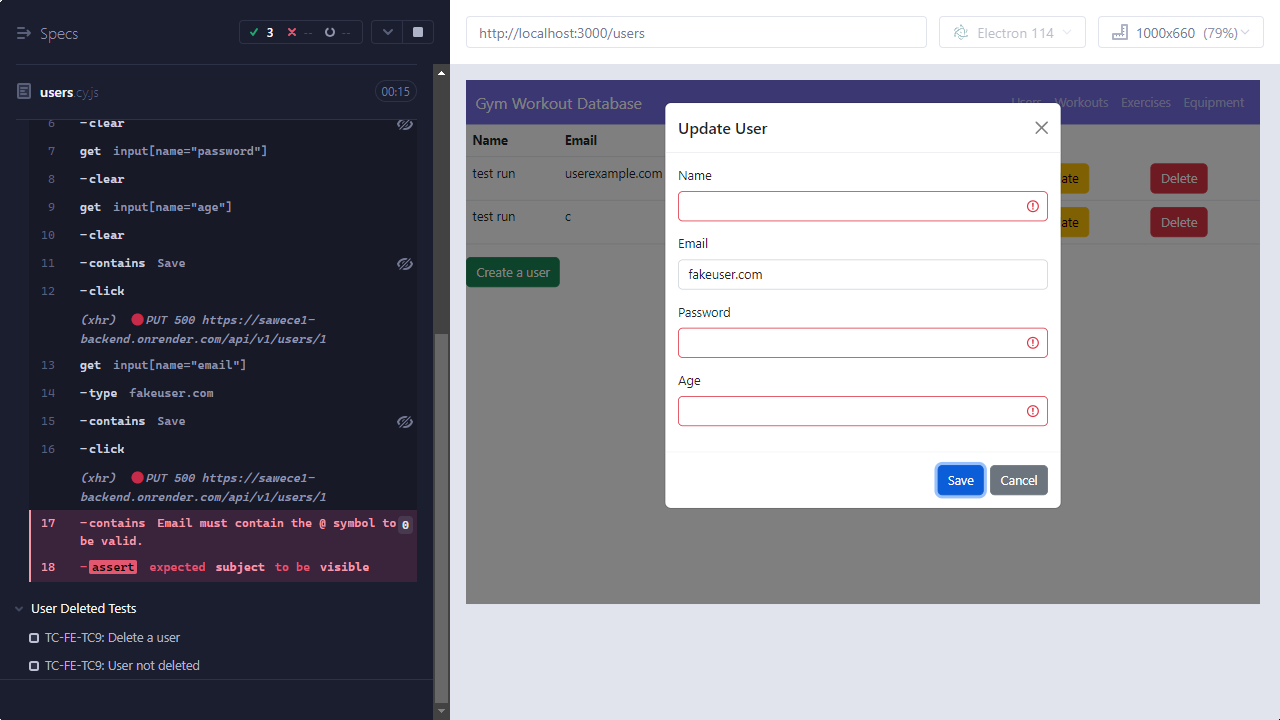


In exercises.cy.js:

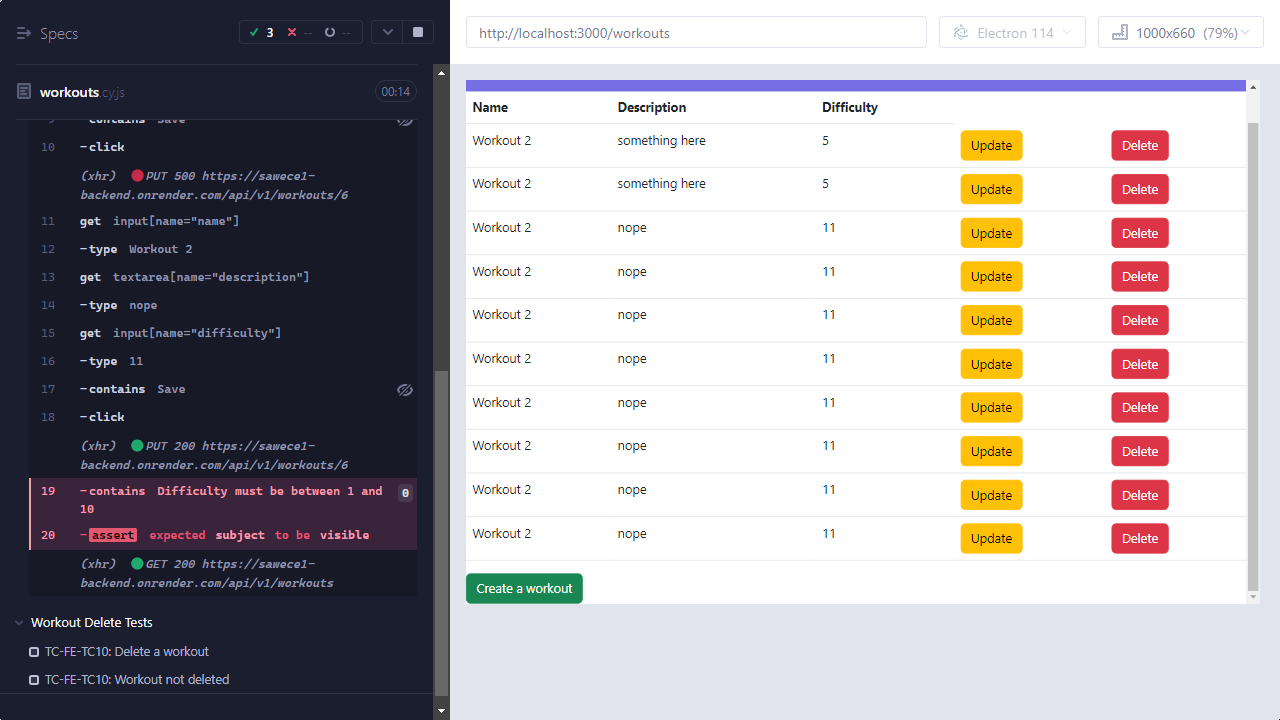


In this defect the input validation is let through when updating it, meaning it goes into the data.

In users.cy.js:



Defect is similar in that the input validation is not consistent with the creation form, same thing with the screenshot below in workouts.cy.js:

****

1. **Test Incident report**

1. Incident Identification

Project: Gym Workout Database React App

Date: 16 November 2023

Author: Crystal Sawers

Title: Defects in Equipment Creation and Inconsistent Form Validations

2. Summary

The testing process revealed critical defects in the Equipment Creation functionality, which were originally classified as low risk in the test plan. This led to the failure of all related tests after creating, such as updating, and deleting equipment data.

3. Incident Description

Location: Identified in equipment.cy.js, users.cy.js, and workouts.cy.js.

Description of Incident:

The 'Create Equipment' button was incorrectly labeled as 'Create an Exercise', preventing the creation of new equipment data. Subsequent tests for updating and deleting equipment failed due to the absence of data. Similar input validation inconsistencies were found in the users and workouts modules.

Steps to Reproduce:

1. Attempt to create equipment using the mislabeled button.
2. Observe the failure in subsequent update and delete tests.
3. Notice similar validation issues in the user and workout modules.

Expected Result: Correctly labeled buttons and consistent form validation across modules.

Actual Result: Mislabeling of buttons and inconsistent validation rules.

4. Impact

Severity: Critical

Priority: High

These defects critically impact the core functionalities of the application, especially user input, causing major disruptions in the testing activities and potentially leading to data integrity issues and security risks.