**CS673 Software Engineering**

**Team 4 - TeamBuilder**

**Software Test Document**

****

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Vishal Sundaram | QA Leader | *Vishal Sundaram* | 22-Sep-2024 |
| Wenli Gai | Security Leader | *Wenli Gai* | 23-Sep-2024 |
| Jake Kelly | Team and Requirements Leader | *Jake Kelly* | 9-23-2024 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **0.2** | **Raymond** | **10/6/24** | **Backend Automated Testing Report** |
|  | **Vishal** | **08-Oct-2024** | **End-to-end testing** |
|  | **Vishal** | **15-Oct-2024** | **Update coverage report, more unit tests** |

[Testing Summary](#_sm5odwyvuk3j)

[Manuel Tests Reports](#_pqso2mbjyzx4)

[Automated Testing Reports](#_mtfbusfb0eq3)

[Testing Metrics](#_rijyjeu2ojqa)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Testing Summary

What was tested:

* UI components unit tests
  + Sidenav
    - Creation of sidenav component
    - should have two navigation items as a non Admin - Home, Add User
    - should call toggleAdminLogin when the slide toggle is changed
    - should display the admin login toggle with the correct default label
  + User details
    - Creation of user details component
    - Should disable save when form is invalid
    - Should enable save when form is valid
  + Auth guard
    - should not allow activation if the user is not logged in
    - should allow activation if the user is logged in
    - should be created

Testing techniques used:

* Manual
* Automated
  + Unit tests
  + End-to-end tests

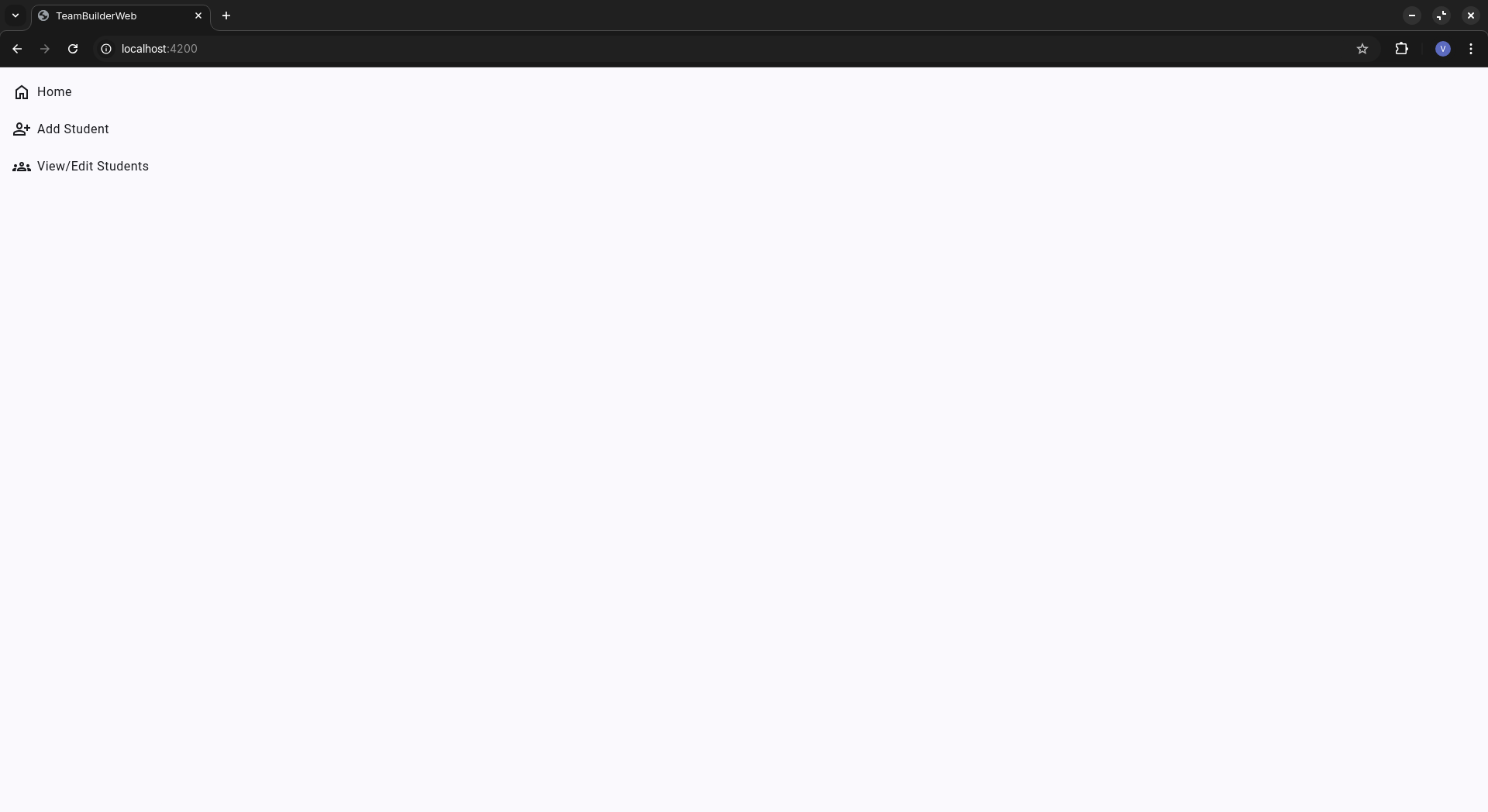
Test result:

* UI:
  + Unit Test coverage summary:
    - Statements : 40.22% ( 35/87 ) →42.73% ( 103/241 )
    - Branches : 7.14% ( 2/28 ) → 13.75% ( 11/80 )
    - Functions : 25.64% ( 10/39 ) → 27.55% ( 27/98 )
    - Lines : 38.27% ( 31/81 ) → 42.19% ( 100/237 )
  + End-to-end test coverage summary:
    - ----------------------------|---------|----------|---------|---------|-------------------
    - File | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
    - ----------------------------|---------|----------|---------|---------|-------------------
    - All files | 100 | 50 | 100 | 100 |
    - team-builder-web | 100 | 50 | 100 | 100 |
    - playwright.config.ts | 100 | 50 | 100 | 100 | 21-23
    - team-builder-web/e2e/pages | 100 | 100 | 100 | 100 |
    - user-details-page.ts | 100 | 100 | 100 | 100 |
    - team-builder-web/e2e/tests | 100 | 100 | 100 | 100 |
    - add-user.spec.ts | 100 | 100 | 100 | 100 |
    - ----------------------------|---------|----------|---------|---------|-------------------
* Server:

# Manual Testing Report

*Test 1*

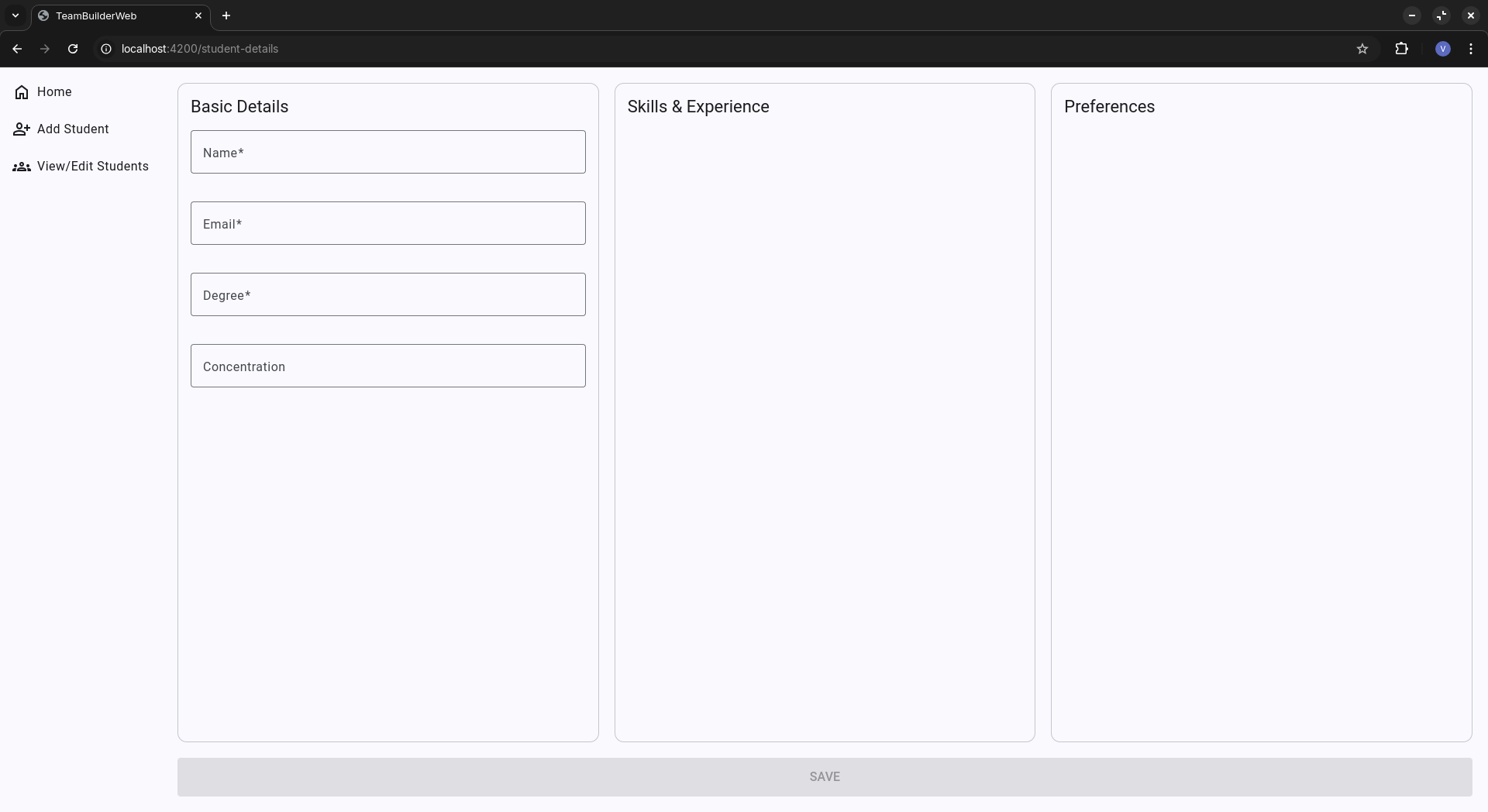
* Test case ID: 1
* Test case name: Side navigation
* New or old: new
* Test items:
  + Side navigation bar
  + navigation items
* Test priority (high/medium/low)
  + High
* input data:
* Test steps:
  + Navigate to website
  + See whether the side navigation bar is loaded to the left side of the app
  + Three navigation items should be displayed on the side navigation bar
* Expected output:
  + Side navigation bar shall be displayed on the left side of the application with three items
    - Home
    - Add Student
    - View/Edit Student
* Actual output:

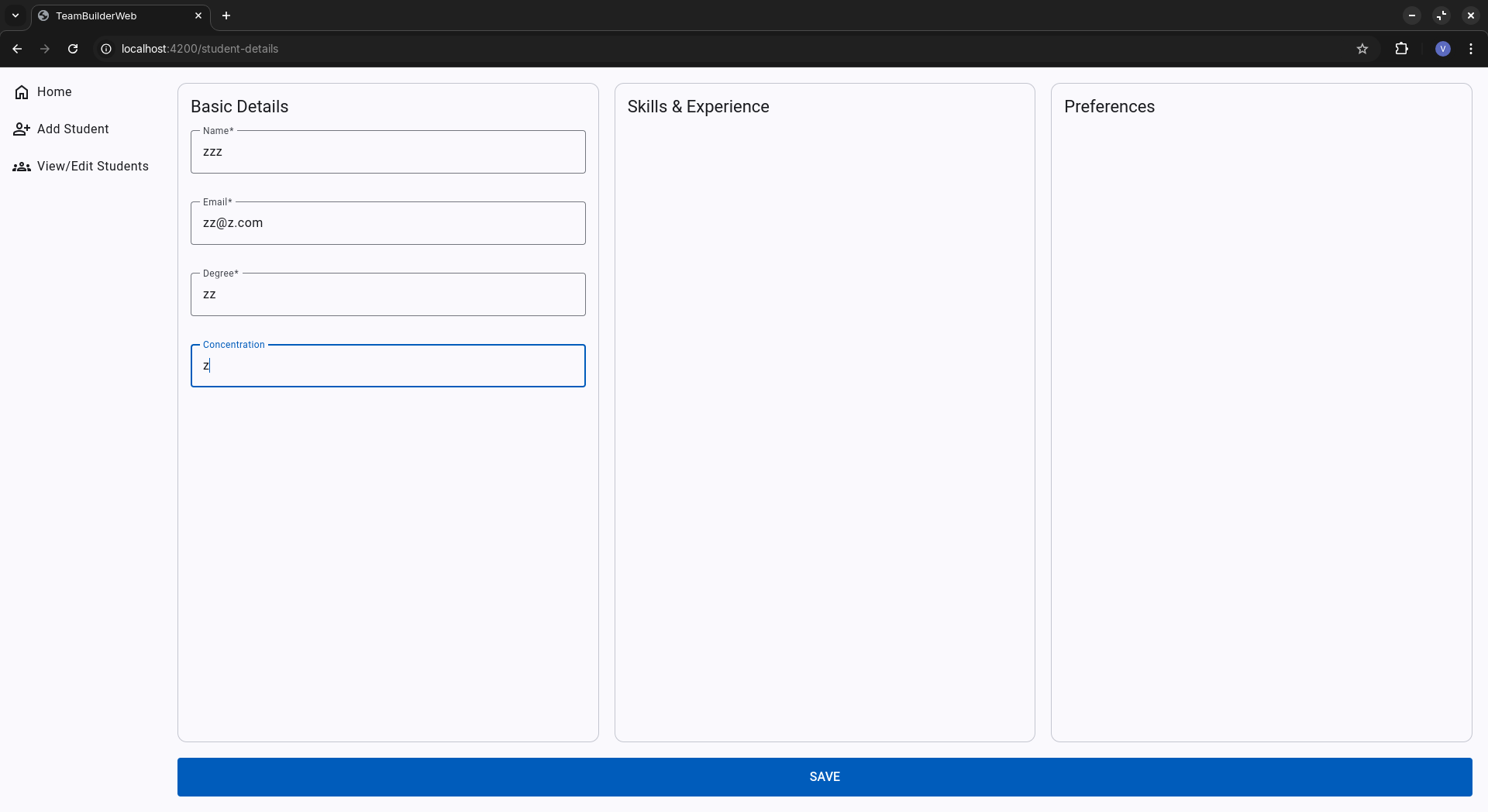


* Pass or Fail: Pass

*Test 2*

* Test case ID: 2
* Test case name: Add student details
* New or old: new
* Test items:
  + Add student page
  + Form input
  + Save button
* Test priority: high
* Test steps:
  + Navigate to website
  + On the side navigation, click on ‘Add Student’
  + On the ‘Basic details’ section, give valid and invalid inputs
* Expected output:
  + Three sections shall be displayed on ‘Add Student’ page
  + On ‘Basic details’, four input fields - Name, Email, Degree, Concentration shall be displayed
  + Given valid values, the save button shall be enabled
* Actual output:



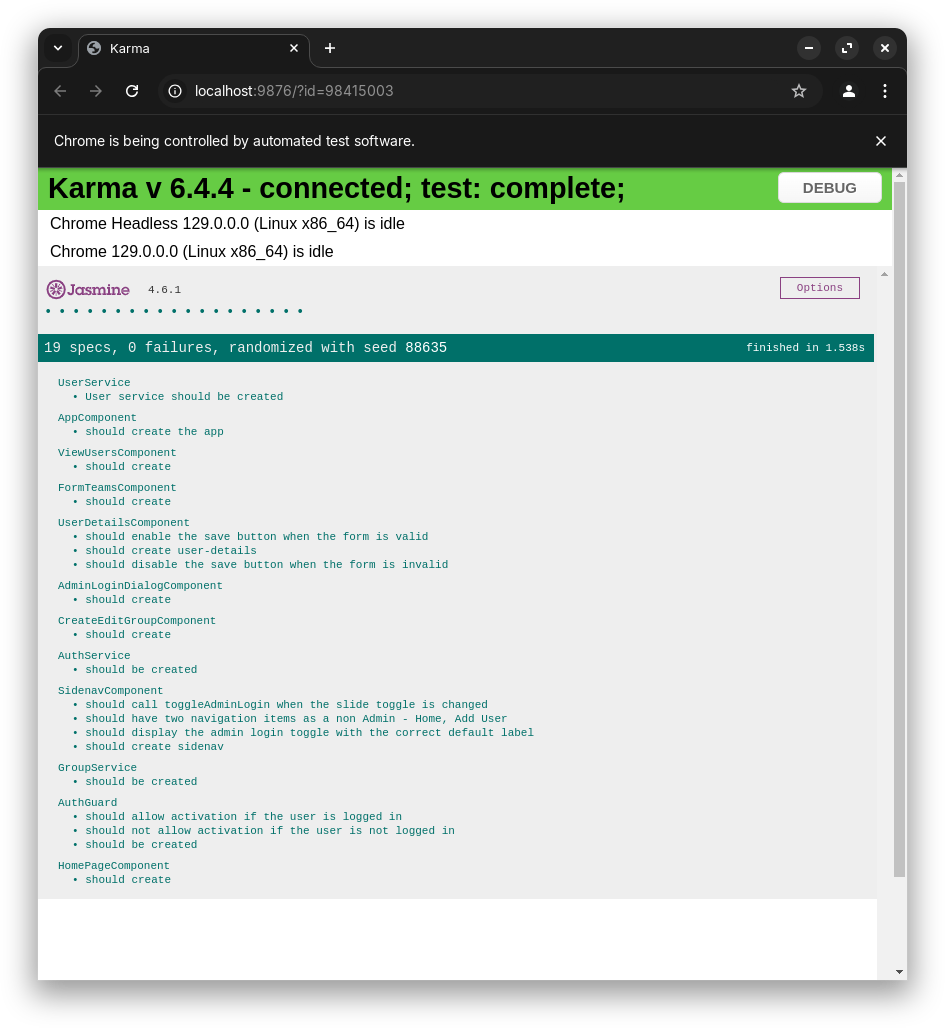


* Pass or Fail: Pass

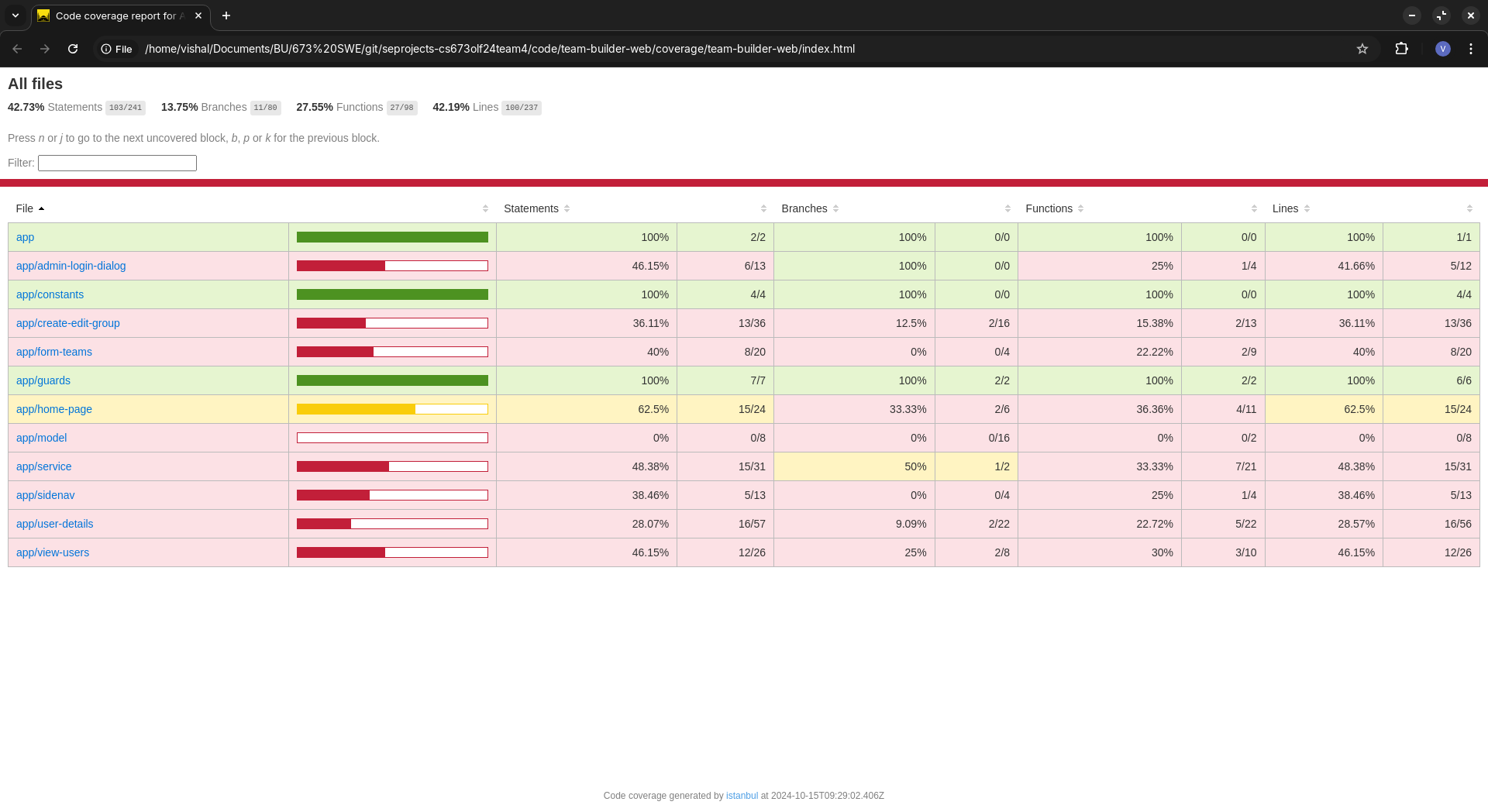
# 

# Automated Testing Report

Unit testing - UI:

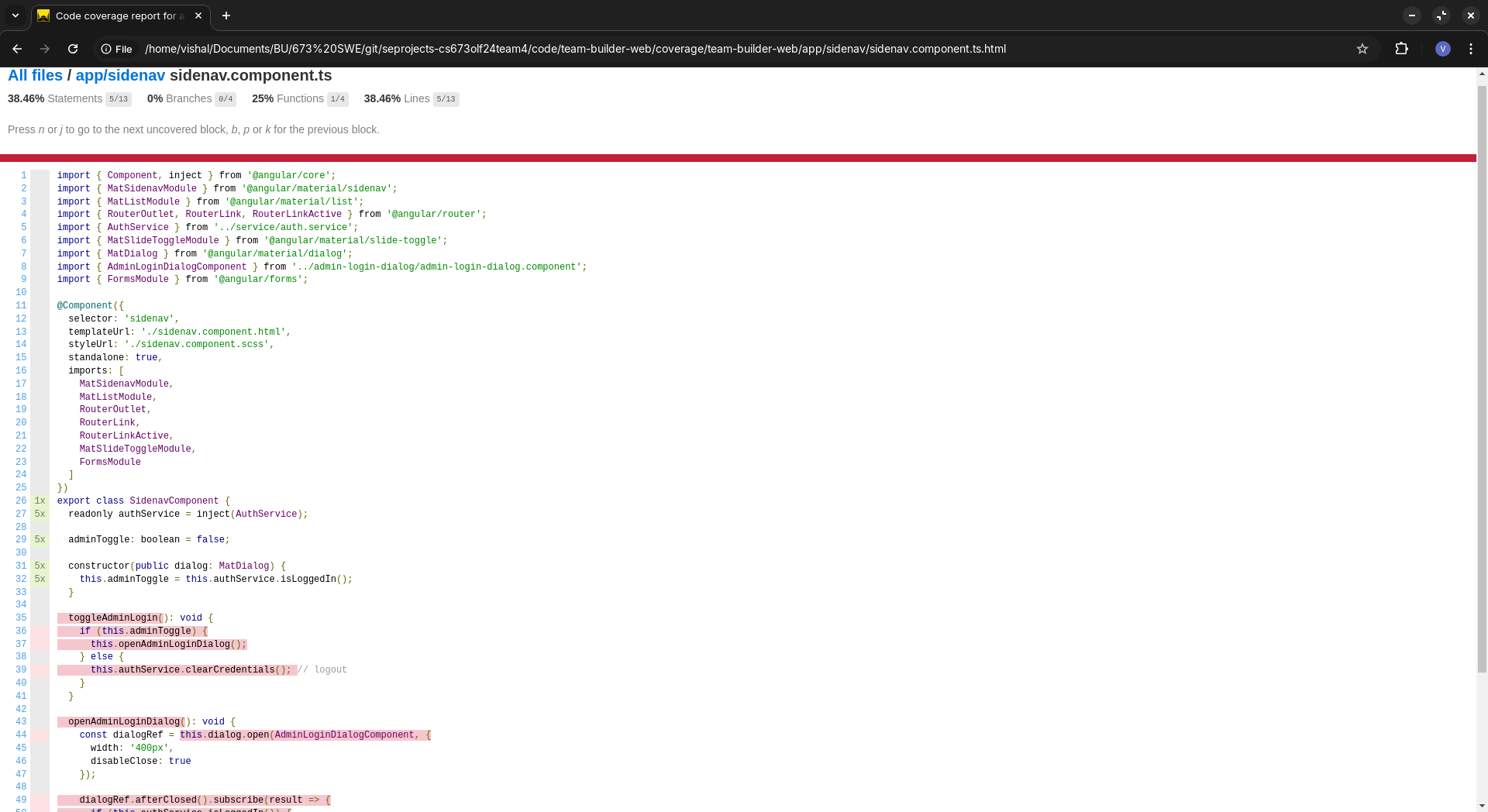


Overall Test coverage report

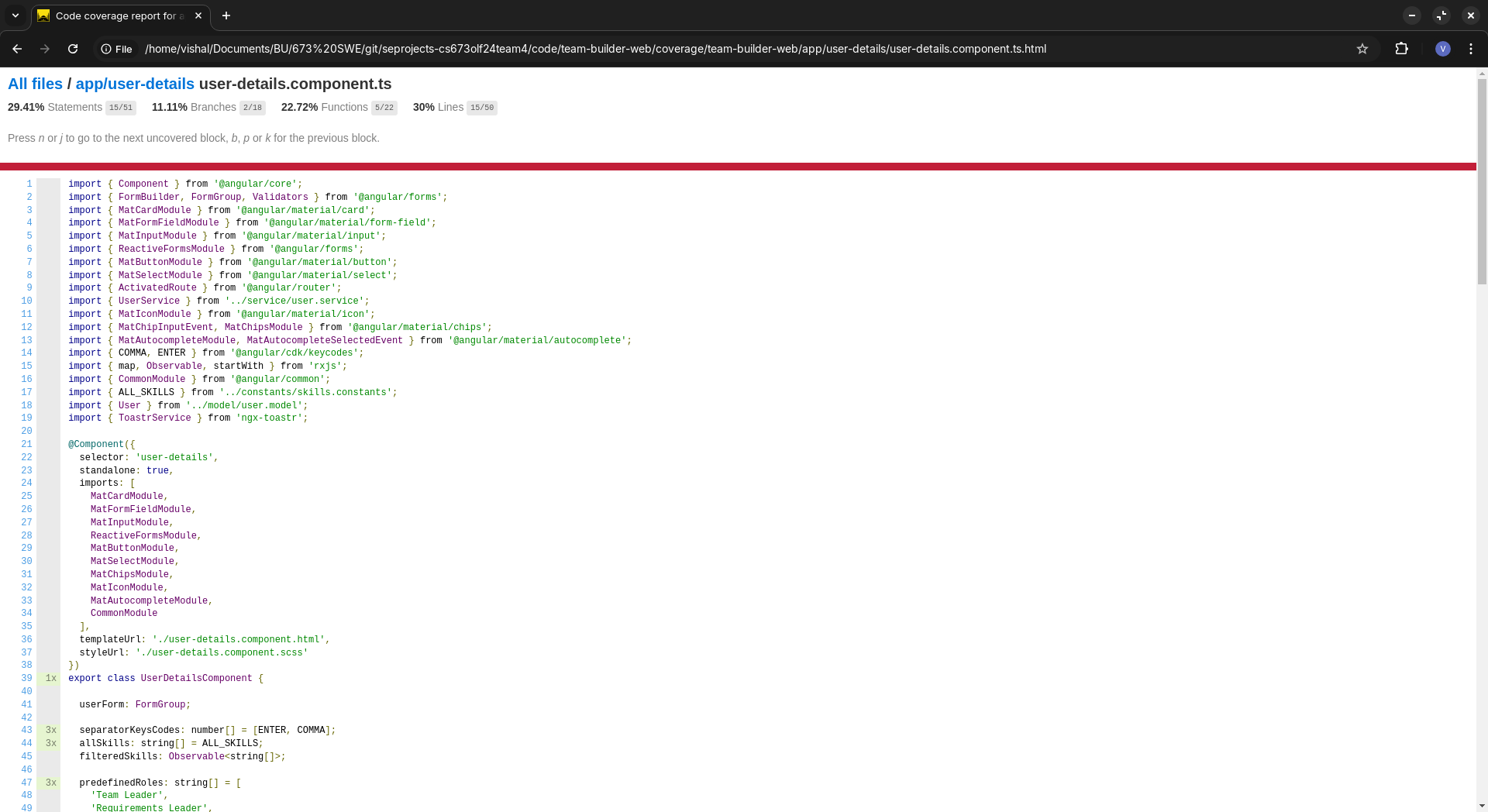


Coverage results are stored in the team-builder-web/coverage directory

Coverage report of Sidenav component:



Coverage report of User details component:



End-to-end testing - no change in test code, test passed:

Tests use Page object pattern to abstract the page details and write the test according to user actions

Test case:

test('should fill user details and enable save button', async ({ page }) => {

const userDetailsPage = new UserDetailsPage(page);

await userDetailsPage.goto();

await userDetailsPage.navigateToAddUserPage();

await userDetailsPage.expectSaveButtonToBeDisabled();

await userDetailsPage.addUserDetails('Test User', 'test.user@tb.com', ['Python', 'Flask'], 'Team Leader');

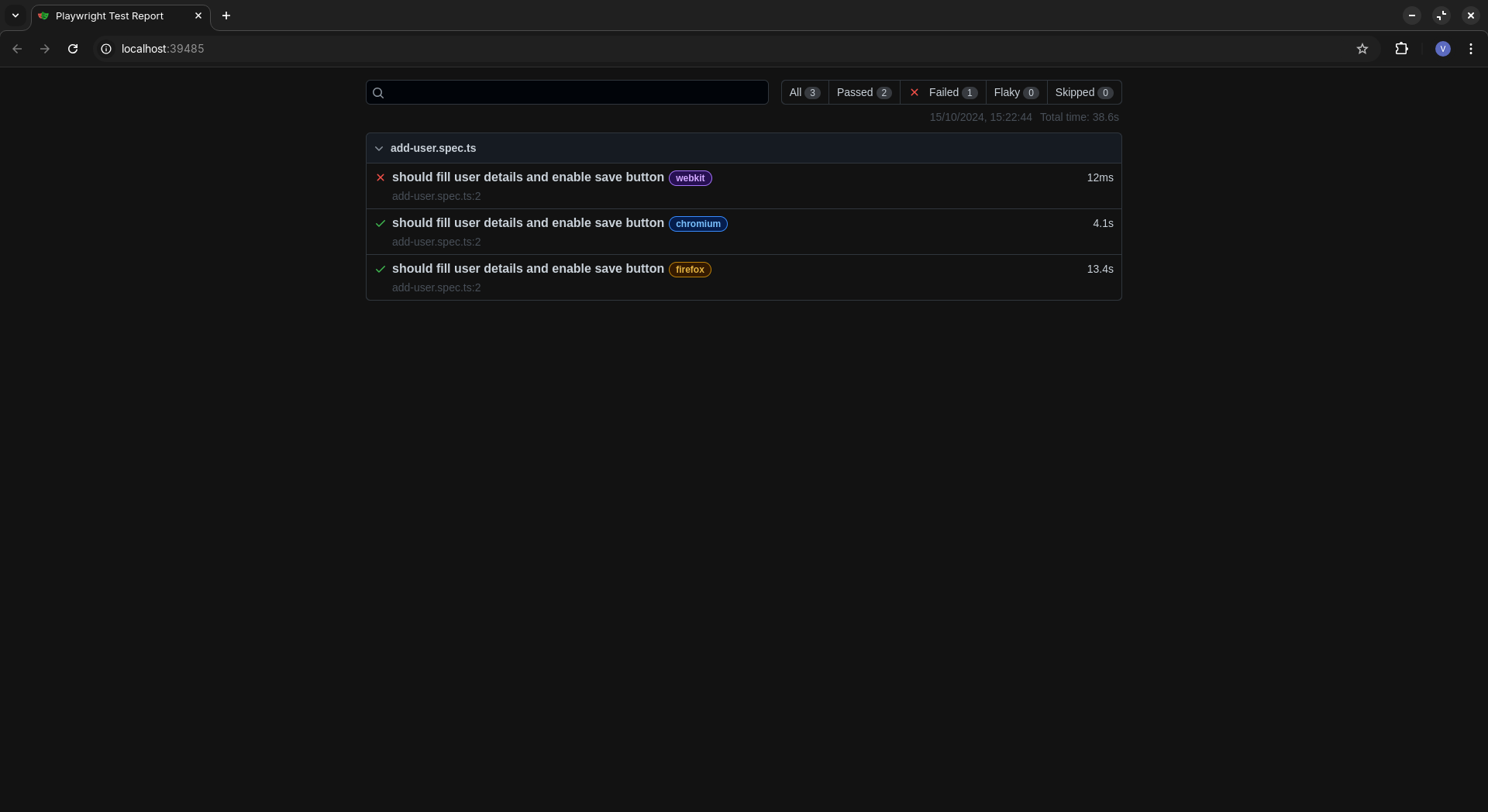
await userDetailsPage.expectSaveButtonToBeEnabled();

await userDetailsPage.saveUser();

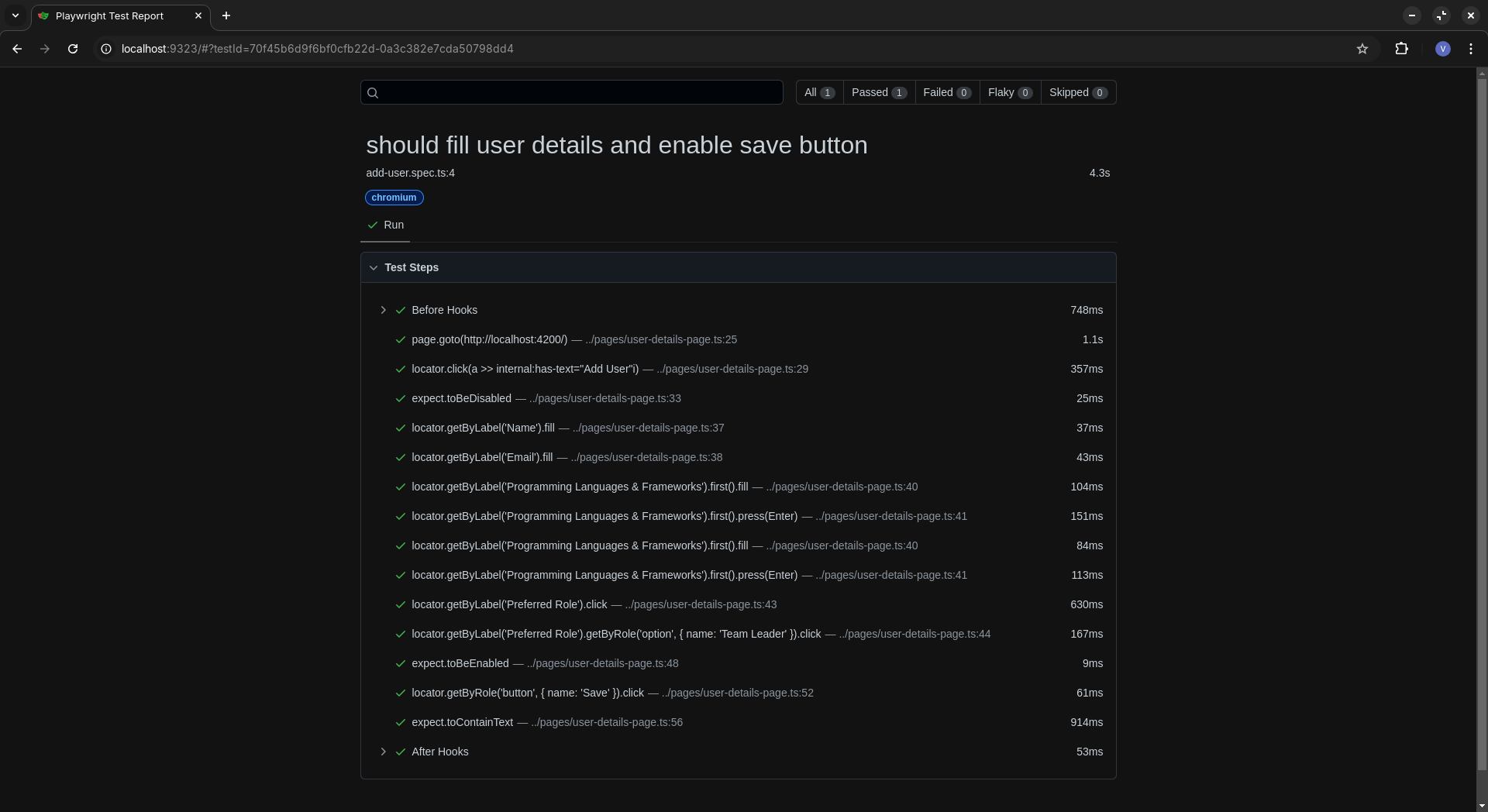
await userDetailsPage.expectSuccessNotification();

});

In chromium and firefox:



Chromium report



# Testing Metrics

UI unit tests:

Number of test cases: 8 → 19

Test coverage: 40.22% → 42.73%

End-to-end tests:

Number of tests: 1

Test coverage: 100%

Server tests:

Number of tests: 14

Test coverage: 88%

Defects: 0

# Backend Automated Testing Report

1. TeamBuilderApplicationTests:

a. testUserCreation

Creates a new user and verifies its properties

b. testGetAllUsers

Retrieves all users and checks the count

c. testUpdateUser

Updates a user's role and answers, then verifies the changes

d. testDeleteUser

Deletes a user and verifies the user count decreases

e. testTeamAssignment

Creates a group with multiple users

Assigns users to teams

Verifies team sizes

Checks role distribution across teams

Ensures Design and Implementation Leaders are in different teams

Verifies even distribution of roles with multiple users

Checks distribution of high-scoring users

2. ControllerTest:

User API Tests:

a. testCreateUser

Creates a new user via API and verifies the response

b. testGetAllUsers

Retrieves all users via API and checks the response format

c. testGetUserById

Creates a user, then retrieves it by ID and verifies the details

d. testUpdateUser

Creates a user, updates its details, and verifies the changes

e. testDeleteUser

Creates a user, deletes it, and verifies it's no longer retrievable

Group API Tests:

f. testCreateGroup

Creates a new group via API and verifies the response

g. testGetAllGroups

Retrieves all groups via API and checks the response format

h. testGetGroupById

Creates a group, then retrieves it by ID and verifies the details

i. testAssignTeams

Creates a group, assigns teams, and verifies the response

# 

# 

# References

# Glossary