System Test Report

Project name: NinjaManager

Team name: CtrlAltDelete

Team Members

Elina Grigoryan

John Vardanyan

Jeremy Vuong

Navid Baghaei

Arjun Bhargava

Madusha Yakupitiyage

Nikolay Chkhaylo

Ashley Palencia - Wisniewski

Date: 05/03/2024

TABLE OF CONTENTS

Test Environment Setup

IDE: Visual Studio Code (version 1.89.0)

Testing Software:

Jest (version 29.7.0)

Selenium (webdriver version 4.19.0)

Operating System: Windows 10 OS (19045.4291) version 22H2

How client will access files and place them in the repository in VsCode?

Jest Configuration

Open a terminal by selecting **Terminal + New Terminal** in the upper left-hand corner of the IDE. Make sure you are in the repository's main file (**CtrlAltDelete**).

- Enter **jest –init**

  When prompted, enter in or choose the following:
  Y
  Y
  Jsdom
  Babel
  Y

  jest –init will have created a **jest.config.js** file. Please ensure the following configurations are set:

  ```
  const config = {
  clearMocks: true,
  collectCoverage: true,
  setupFilesAfterEnv: ['./jest.setup.js'],
  moduleNameMapper: { '\\.(css|less|scss|sass)$': 'identity-obj-proxy' },
  transform: { '^.+\\.(js|jsx|ts|tsx)$': ['babel-jest', { configFile:
  "./babel.config.js" }] },
  coverageDirectory: "coverage",
  testEnvironment: "jest-environment-jsdom",
  verbose: true,
  };
  ```

- Create a **jest.setup.js** file with the following:

  ```
  global.fetch = jest.fn((url, config) => {
  ```

```
// Mock responses based on URL
if (url.includes('/api/user/employees')) {
// Respond with an empty array of employees or whatever is appropriate for
your tests
return Promise.resolve({
ok: true,
json: () => Promise.resolve([]),
});
}
// You can add more conditions for other API endpoints
// ...
// Default to rejecting fetch calls that aren't explicitly mocked:
return Promise.reject(new Error(`Unmocked endpoint: ${url}`));
});
});
afterEach(() => {
global.fetch.mockClear();
});
afterAll(() => {
global.fetch.mockRestore();
});
```

- Enter **npm install –save-dev jest**
- Enter **npm install --save-dev jest-environment-jsdom**
- Enter **npm install --save-dev identity-obj-proxy**
- Enter **npm install node-fetch@2**
- Enter **npm install jest-fetch-mock --save-dev**
- All test files should be placed in **CtrlAltDelete/Frontend/Src/Pages**. All Jest testing files have the following naming convention: **<filename>.test.js**.
- To run all test files, enter "**npm test**." To run a specific file, enter "**npm test <test file name>**"

Selenium Configuration

Open a terminal by selecting **Terminal + New Terminal** in the upper left-hand corner of the IDE. Make sure you are in the repository's main file (**CtrlAltDelete**).

- Enter **npm install selenium-webdriver**
- Enter **npm install chromedriver**

- All test files should be placed in **CtrlAltDelete/Frontend/Src/Pages**. All Selenium testing files have the following naming convention: **<filename>SeleniumTest.js**.

- To run a selenium test, open a new terminal in VSCode by selecting **Terminal + New Terminal** in the upper left-hand corner. In the main directory (**CtrlAltDelete**), start the application by entering "**npm start**."

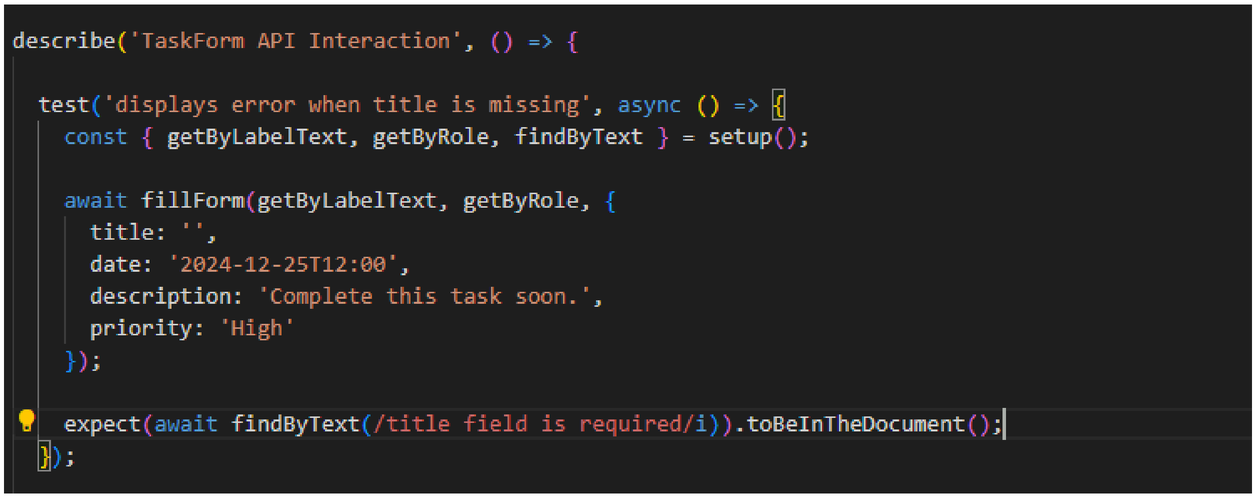
- Open another terminal by selecting **Terminal + New Terminal** in the upper left-hand corner of VSCode. Navigate to **CtrlAltDelete/Frontend/Src/Pages** (where test files should be located). To run a test file, enter "**node <filename>SeleniumTest.js**"

<u>Jest Testing</u>

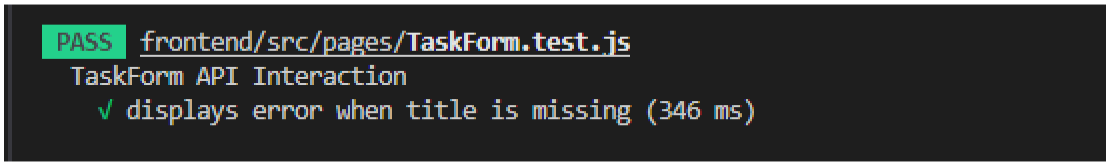
Create Task Page

1.0 Create Task Page - Display an error message when title is missing

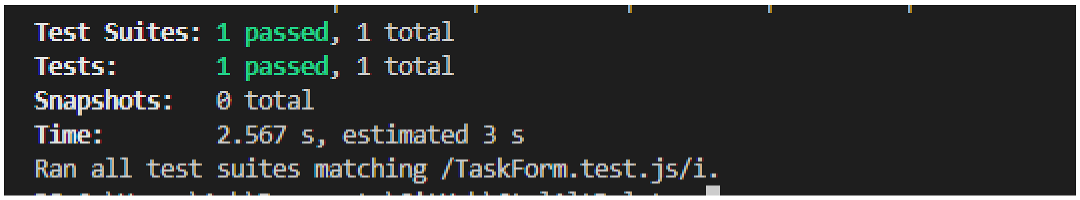
1.1 Description: Displays field-specific error message when title is missing
1.2 Valid Test Inputs: date: "2024-12-25T12:00," description: "Complete this task soon.", priority: "High"
1.3 In-Valid Test Inputs: title: " "
1.4 Tester: Ashley Palencia
1.5 Recorder: Elina Grigoryan
1.6 Date Tested: 4/24/2024
1.7 Results
    1.7.1 Did it pass? Yes



```
describe('TaskForm API Interaction', () => {

  test('displays error when title is missing', async () => {
    const { getByLabelText, getByRole, findByText } = setup();

    await fillForm(getByLabelText, getByRole, {
      title: '',
      date: '2024-12-25T12:00',
      description: 'Complete this task soon.',
      priority: 'High'
    });

    expect(await findByText(/title field is required/i)).toBeInTheDocument();
  });
```



```
 PASS  frontend/src/pages/TaskForm.test.js
  TaskForm API Interaction
    √ displays error when title is missing (346 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.567 s, estimated 3 s
Ran all test suites matching /TaskForm.test.js/i.
```

2.0 Create Task Page - Displays an error message when date is missing

2.1 Description: Displays field-specific error message when date is missing

2.2 Valid Test Inputs: title: “New Task”, description: “Complete this task soon.”, priority: “High”

2.3 In-Valid Test Inputs: date: “ “

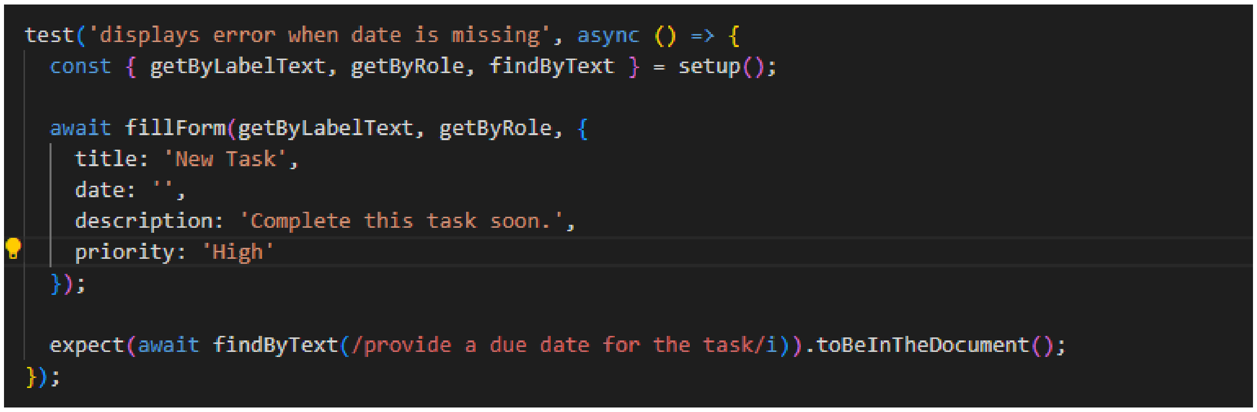
2.4 Tester: Elina Grigoryan

2.5 Recorder: Ashley Palencia

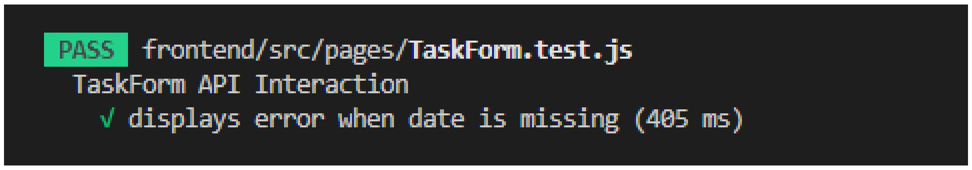
2.6 Date Tested: 4/24/2024

2.7 Results

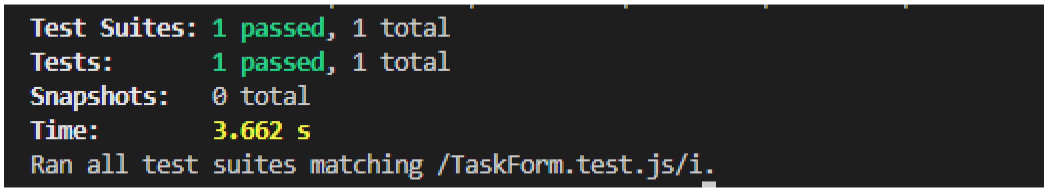
2.7.1 Did it pass? Yes



```
test('displays error when date is missing', async () => {
  const { getByLabelText, getByRole, findByText } = setup();

  await fillForm(getByLabelText, getByRole, {
    title: 'New Task',
    date: '',
    description: 'Complete this task soon.',
    priority: 'High'
  });

  expect(await findByText(/provide a due date for the task/i)).toBeInTheDocument();
});
```



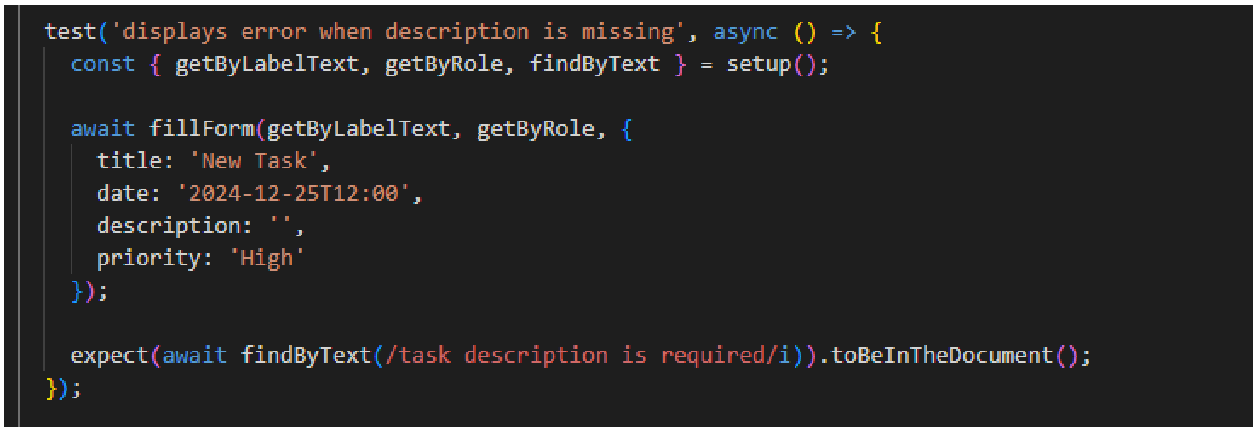
```
 PASS  frontend/src/pages/TaskForm.test.js
  TaskForm API Interaction
    √ displays error when date is missing (405 ms)
```



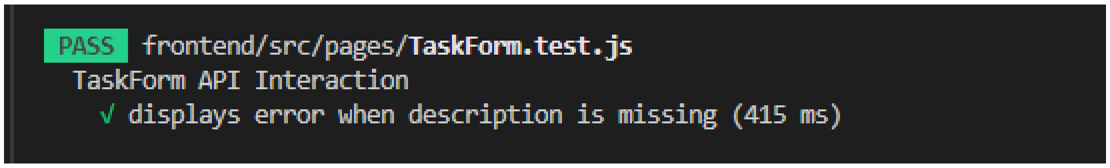
```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.662 s
Ran all test suites matching /TaskForm.test.js/i.
```

3.0 Create Task Page - Displays an error message when description is missing

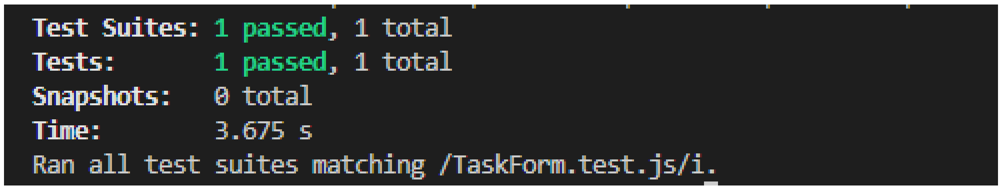
3.1 Displays field-specific error message when description is missing
3.2 Valid Test Inputs: title: “New Task,” date: “2024-12-25T12:00,”, priority: “High”
3.3 In-Valid Test Inputs: description: “ “
3.4 Tester: Ashley Palencia
3.5 Recorder: Elina Grigoryan
3.6 Date Tested: 4/25/2024
3.7 Results
3.7.1 Did it pass? Yes



```
test('displays error when description is missing', async () => {
  const { getByLabelText, getByRole, findByText } = setup();

  await fillForm(getByLabelText, getByRole, {
    title: 'New Task',
    date: '2024-12-25T12:00',
    description: '',
    priority: 'High'
  });

  expect(await findByText(/task description is required/i)).toBeInTheDocument();
});
```



```
 PASS  frontend/src/pages/TaskForm.test.js
  TaskForm API Interaction
    √ displays error when description is missing (415 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.675 s
Ran all test suites matching /TaskForm.test.js/i.
```

4.0 Create Task Page - Displays error when priority field is missing

4.1 Description: Displays field-specific error message when priority is missing

4.2 Valid Test Inputs: title: “New Task,” date: “2024-12-25T12:00,” description: “Complete this task soon.”

4.3 In-Valid Test Inputs: priority: “ ”

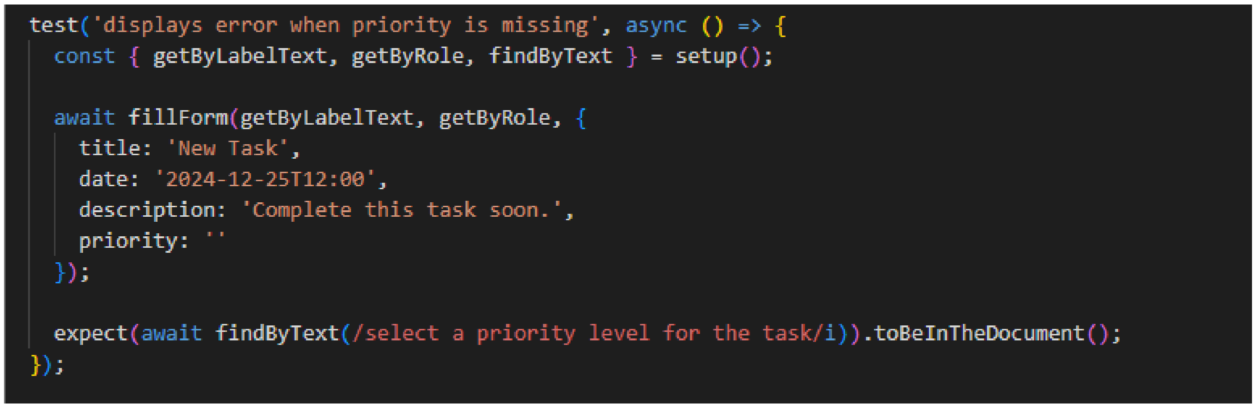
4.4 Tester: Elina Grigoryan

4.5 Recorder: Ashley Palencia

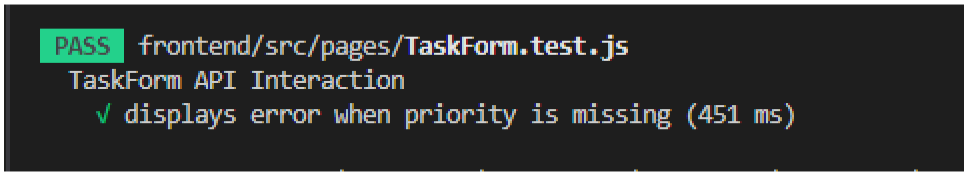
4.6 Date: 04/25/2024

4.7 Results

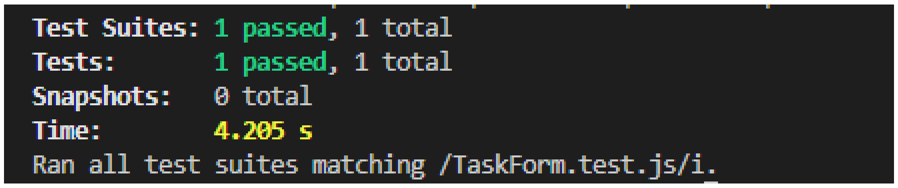
4.7.1 Did it pass? Yes



```
test('displays error when priority is missing', async () => {
  const { getByLabelText, getByRole, findByText } = setup();

  await fillForm(getByLabelText, getByRole, {
    title: 'New Task',
    date: '2024-12-25T12:00',
    description: 'Complete this task soon.',
    priority: ''
  });

  expect(await findByText(/select a priority level for the task/i)).toBeInTheDocument();
});
```



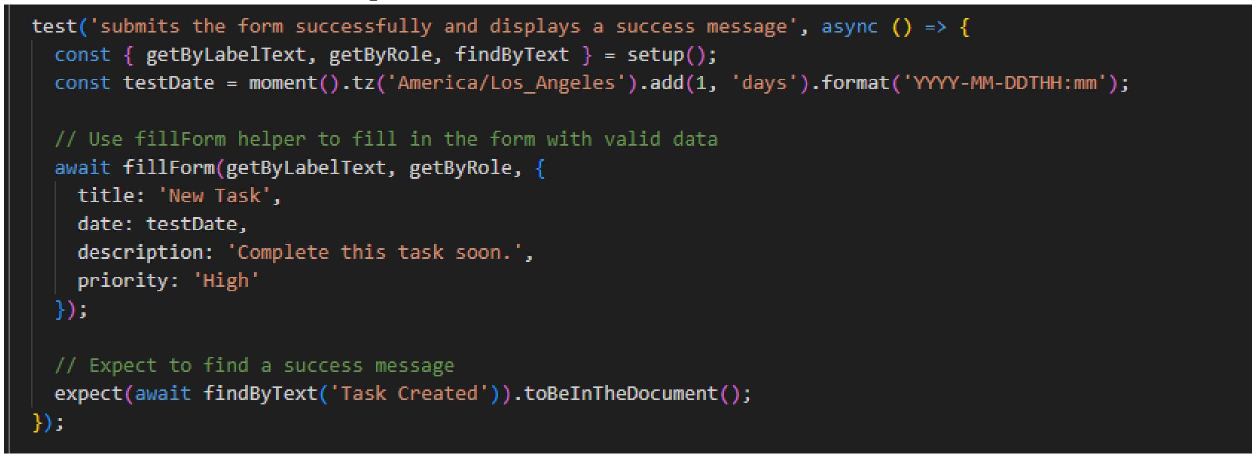
```
 PASS  frontend/src/pages/TaskForm.test.js
  TaskForm API Interaction
    √ displays error when priority is missing (451 ms)
```



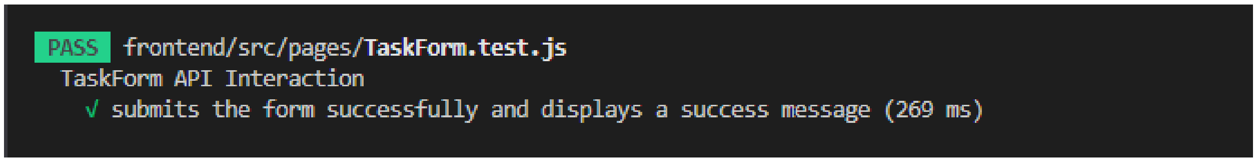
```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.205 s
Ran all test suites matching /TaskForm.test.js/i.
```

5.0 Create Task - Displays a success message with valid input

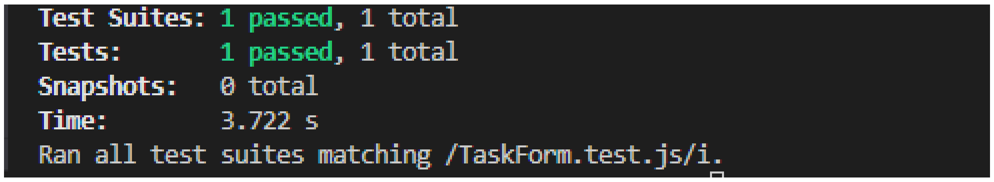
5.1 Description: Displays message "Task Created" when the task form is submitted successfully.
5.2 Valid Test Inputs: title: "New Task," date: "testDate," description: "Complete this task soon.," priority: "High"
5.3 In-Valid Test Inputs: None
5.4 Tester: Elina Grigoryan
5.5 Recorder: Ashley Palencia
5.6 Date: 04/25/2024
5.7 Results
5.7.1 Did it pass? Yes



```
test('submits the form successfully and displays a success message', async () => {
  const { getByLabelText, getByRole, findByText } = setup();
  const testDate = moment().tz('America/Los_Angeles').add(1, 'days').format('YYYY-MM-DDTHH:mm');

  // Use fillForm helper to fill in the form with valid data
  await fillForm(getByLabelText, getByRole, {
    title: 'New Task',
    date: testDate,
    description: 'Complete this task soon.',
    priority: 'High'
  });

  // Expect to find a success message
  expect(await findByText('Task Created')).toBeInTheDocument();
});
```



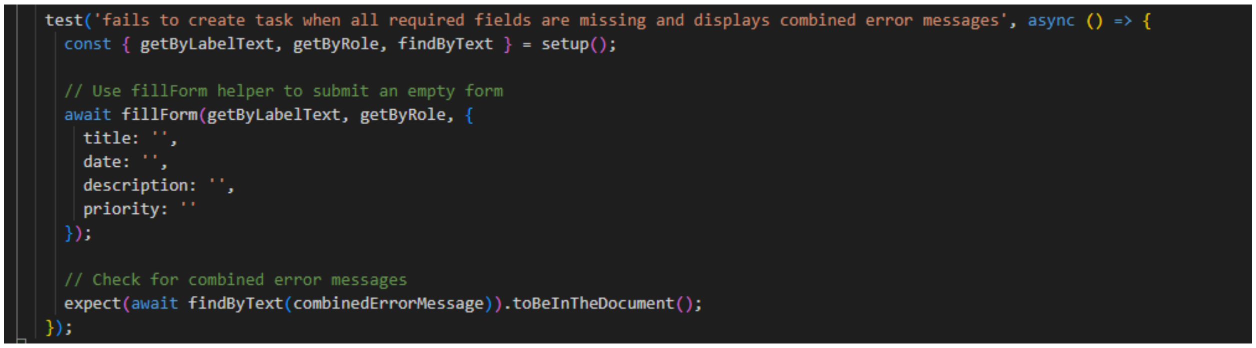
```
PASS frontend/src/pages/TaskForm.test.js
  TaskForm API Interaction
    √ submits the form successfully and displays a success message (269 ms)
```



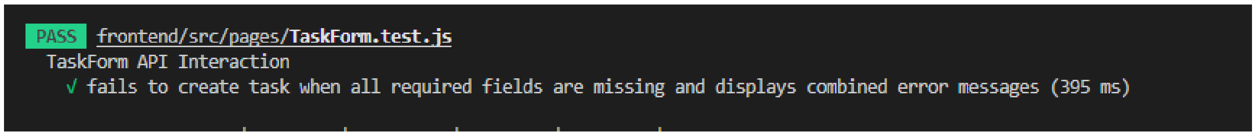
```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        3.722 s
Ran all test suites matching /TaskForm.test.js/i.
```

6.0 Create Task - Displays error message when all required fields are missing

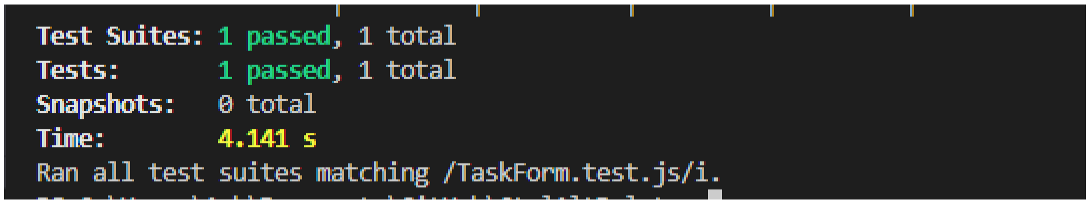
6.1 Description: Displays combined error message for each required field missing.
6.2 Valid Test Inputs: None
6.3 In-Valid Test Inputs: title:" " date: " " description: " " priority: " "
6.4 Tester: Elina Grigoryan
6.5 Recorder: Ashley Palencia
6.6 Date: 04/25/2024
6.7 Results
6.7.1 Did it pass? Yes



```
test('fails to create task when all required fields are missing and displays combined error messages', async () => {
  const { getByLabelText, getByRole, findByText } = setup();

  // Use fillForm helper to submit an empty form
  await fillForm(getByLabelText, getByRole, {
    title: '',
    date: '',
    description: '',
    priority: ''
  });

  // Check for combined error messages
  expect(await findByText(combinedErrorMessage)).toBeInTheDocument();
});
```



```
PASS  frontend/src/pages/TaskForm.test.js
  TaskForm API Interaction
    √ fails to create task when all required fields are missing and displays combined error messages (395 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.141 s
Ran all test suites matching /TaskForm.test.js/i.
```

Overview Page

7.0 Overview Page - Filter results update to show only tasks of selected priority.

7.1 Brief Description of Test - Validates that only tasks with 'Medium' priority are displayed when this filter is applied and tasks with 'Low' and 'High' priority are not shown.

7.2 Valid Test Inputs: Description: The tester selects 'Medium' from the priority dropdown.Values: "Medium"

7.3 In-Valid Test Inputs: Description: The tester selects a non-existent priority option. Values: "Ultra-High", "", null

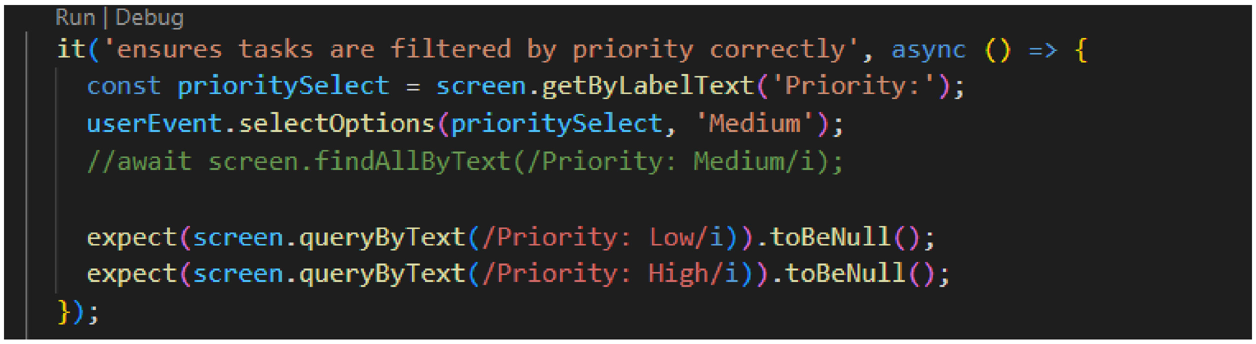
7.4 Tester Name - Navid Baghaei

7.5 Recorder - Madusha Yakupitiyage

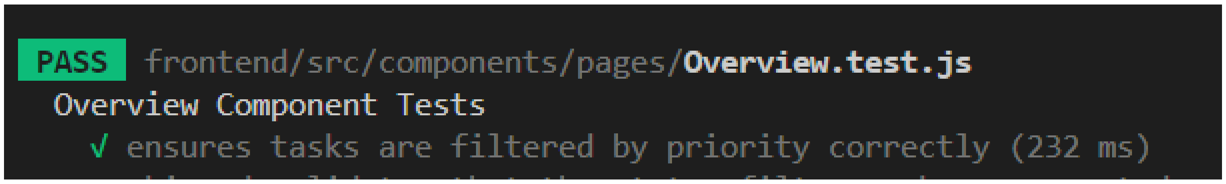
7.6 Date Tested - 05/02/24

7.7 Results

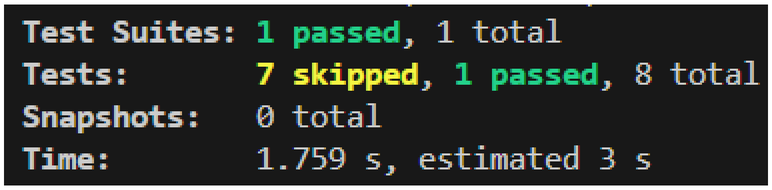
7.7.1 Did it pass? Yes



```
Run | Debug
it('ensures tasks are filtered by priority correctly', async () => {
  const prioritySelect = screen.getByLabelText('Priority:');
  userEvent.selectOptions(prioritySelect, 'Medium');
  //await screen.findAllByText(/Priority: Medium/i);

  expect(screen.queryByText(/Priority: Low/i)).toBeNull();
  expect(screen.queryByText(/Priority: High/i)).toBeNull();
});
```



```
PASS  frontend/src/components/pages/Overview.test.js
  Overview Component Tests
    √ ensures tasks are filtered by priority correctly (232 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        1.759 s, estimated 3 s
```

8.0 Overview Page - Filter results update to show only tasks marked as 'Past Due'.

8.1 Brief Description of Test - Tests if tasks can be filtered correctly by status, specifically verifying that only tasks with 'Past Due' status are visible, excluding other statuses like 'In Progress'.

8.2 Valid Test Inputs: Description: The tester selects 'Past Due' from the status dropdown. Values: "Past Due"

8.3 In-Valid Test Inputs: Description: Description: Tester selects a status that does not exist. Values: "Almost Finished", "", null

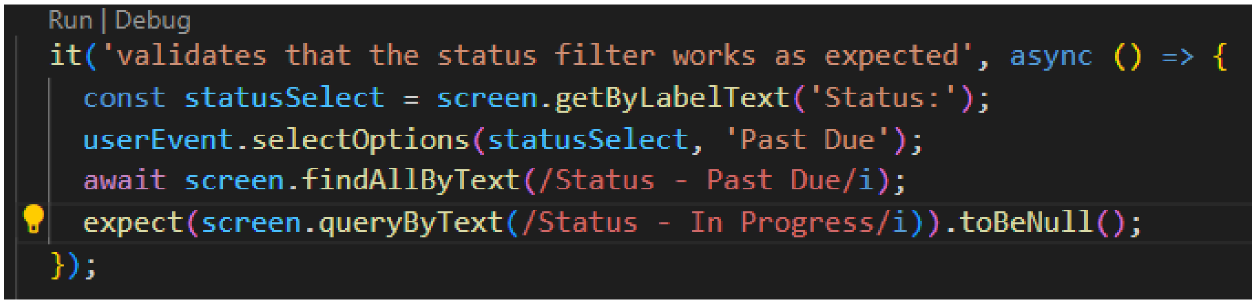
8.4 Tester Name - Navid Baghaei

8.5 Recorder - Madusha Yakupitiyage

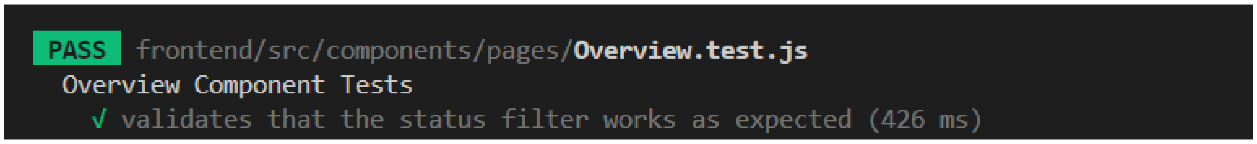
8.6 Date Tested - 05/02/24

8.7 Results

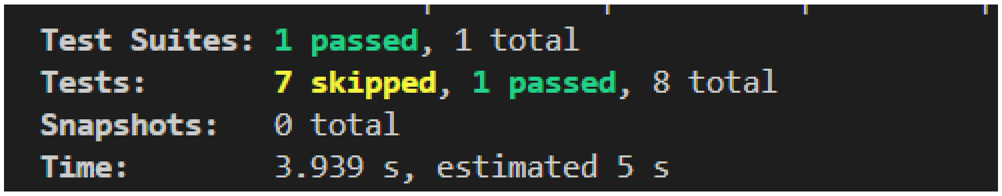
8.7.1 Did it pass? Yes



```
Run | Debug
it('validates that the status filter works as expected', async () => {
  const statusSelect = screen.getByLabelText('Status:');
  userEvent.selectOptions(statusSelect, 'Past Due');
  await screen.findAllByText(/Status - Past Due/i);
  expect(screen.queryByText(/Status - In Progress/i)).toBeNull();
});
```



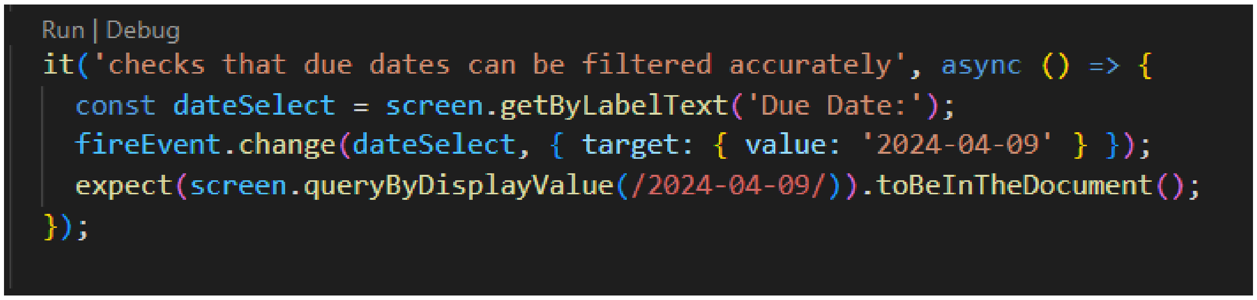
```
 PASS  frontend/src/components/pages/Overview.test.js
  Overview Component Tests
    √ validates that the status filter works as expected (426 ms)
```



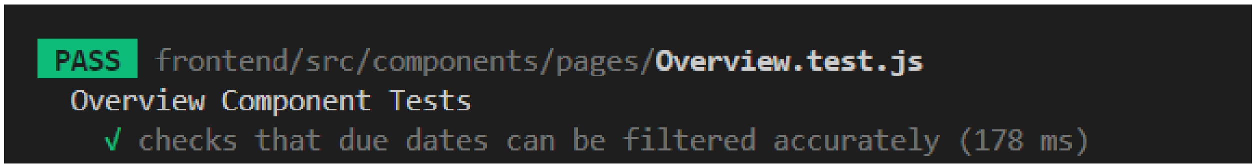
```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        3.939 s, estimated 5 s
```

9.0 Overview Page - Display only tasks that match the specified due date.

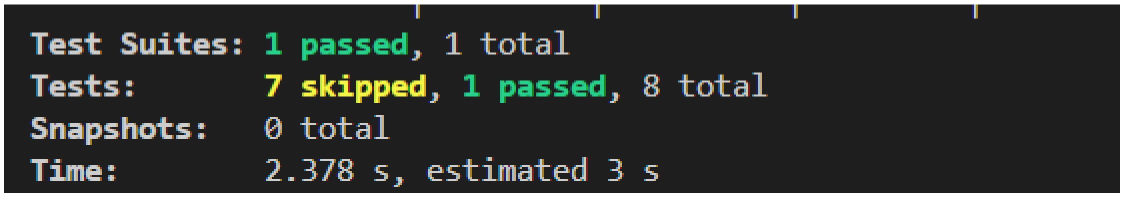
9.1 Brief Description of Test - Ensures the due date filter correctly displays tasks that are due on '2024-04-09', excluding tasks due on other dates.
9.2 Valid Test Inputs: Description: The tester enters a specific date in the due date filter. Values: '2024-04-09'
9.3 In-Valid Test Inputs: Description: Tester enters an invalid date or format. Values: '2024-13-01', '09/04/2024', 'invalid-date'
9.4 Tester Name - Navid Baghaei
9.5 Recorder - Madusha Yakupitiyage
9.6 Date Tested - 05/02/24
9.7 Results
    9.7.1 Did it pass? Yes



```
Run | Debug
it('checks that due dates can be filtered accurately', async () => {
  const dateSelect = screen.getByLabelText('Due Date:');
  fireEvent.change(dateSelect, { target: { value: '2024-04-09' } });
  expect(screen.queryByDisplayValue(/2024-04-09/)).toBeInTheDocument();
});
```



```
PASS  frontend/src/components/pages/Overview.test.js
  Overview Component Tests
    √ checks that due dates can be filtered accurately (178 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        2.378 s, estimated 3 s
```

10.0 Overview Page - Tasks that include the text 'nhhhgg' in their content are displayed.

10.1 Brief Description of Test - Confirms that the search functionality filters and displays only tasks containing the specific string 'nhhhgg', while excluding others.

10.2 Valid Test Inputs: Description: The tester types a string into the search field. Values: 'nhhhgg'

10.3 In-Valid Test Inputs: Description: The tester enters characters or strings that do not match any task. Values: 'xyz123', '', ' ' (spaces)

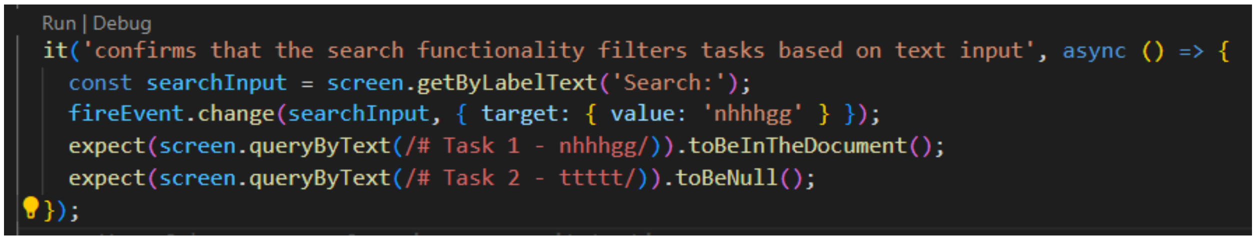
10.4 Tester Name - Navid Baghaei

10.5 Recorder - Madusha Yakupitiyage

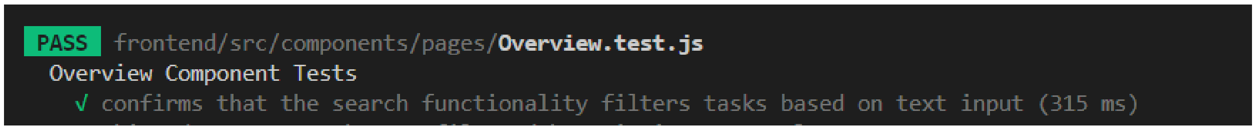
10.6 Date Tested - 05/02/24

10.7 Results

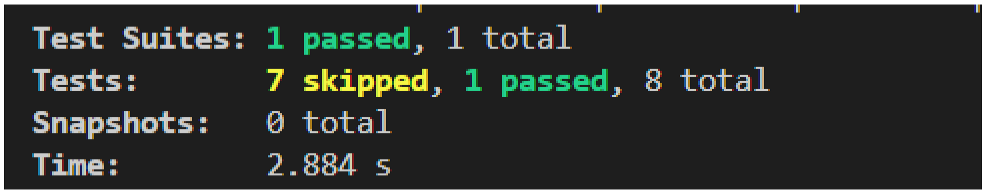
10.7.1 Did it pass? Yes



```
Run | Debug
it('confirms that the search functionality filters tasks based on text input', async () => {
  const searchInput = screen.getByLabelText('Search:');
  fireEvent.change(searchInput, { target: { value: 'nhhhgg' } });
  expect(screen.queryByText(/# Task 1 - nhhhgg/)).toBeInTheDocument();
  expect(screen.queryByText(/# Task 2 - ttttt/)).toBeNull();
});
```



```
PASS  frontend/src/components/pages/Overview.test.js
  Overview Component Tests
    √ confirms that the search functionality filters tasks based on text input (315 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        2.884 s
```

11.0 Overview Page - Reset All Filters to Default States

11.1 Brief Description of Test - This test verifies that clicking the 'Reset Filters' button effectively resets all set filters (Priority, Status, Due Date, and Search) back to their default settings.

11.2 Valid Test Inputs:

Description: The tester sets various filters and then resets them.

Values: Priority - 'Low', Status - 'In Progress', Due Date - '2024-04-17', Search - 'task

11.3 In-Valid Test Inputs:

Description: The tester clicks 'Reset Filters' without setting any filters initially.

Values: Filters remain unchanged as no prior settings were altered.

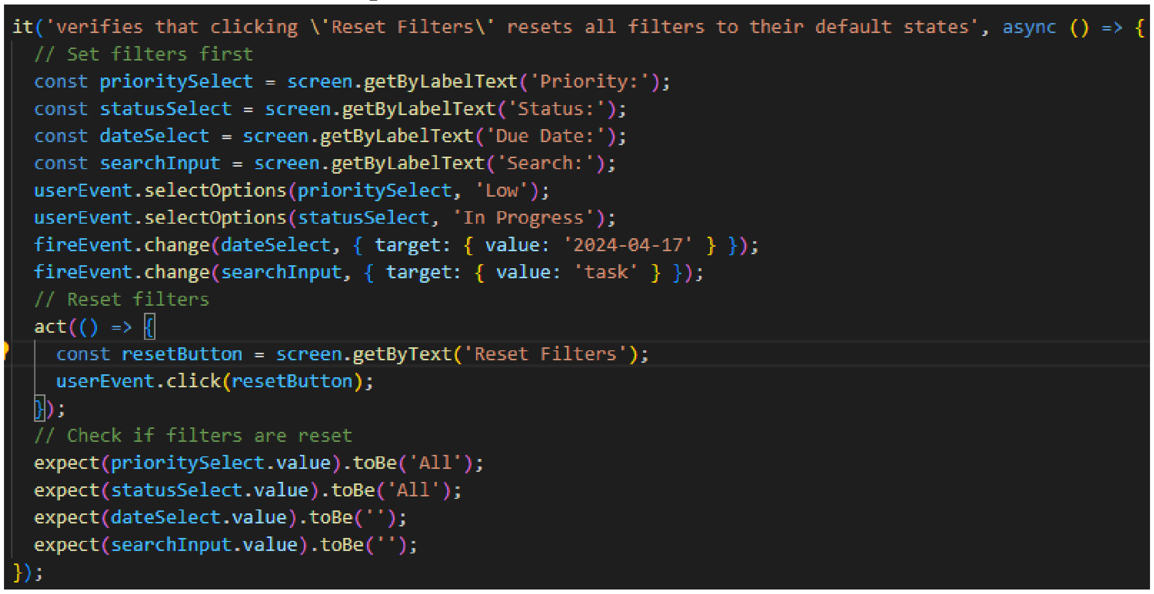
11.4 Tester Name - Madusha Yakupitiyage

11.5 Recorder - Navid Baghaei

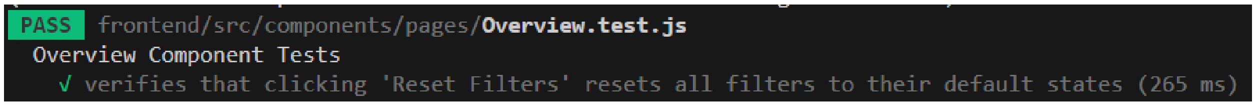
11.6 Date Tested - 05/02/24

11.7 Results

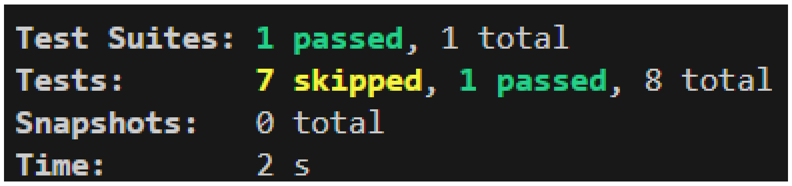
11.7.1 Did it pass? Yes



```
it('verifies that clicking \'Reset Filters\' resets all filters to their default states', async () => {
  // Set filters first
  const prioritySelect = screen.getByLabelText('Priority:');
  const statusSelect = screen.getByLabelText('Status:');
  const dateSelect = screen.getByLabelText('Due Date:');
  const searchInput = screen.getByLabelText('Search:');
  userEvent.selectOptions(prioritySelect, 'Low');
  userEvent.selectOptions(statusSelect, 'In Progress');
  fireEvent.change(dateSelect, { target: { value: '2024-04-17' } });
  fireEvent.change(searchInput, { target: { value: 'task' } });
  // Reset filters
  act(() => {
    const resetButton = screen.getByText('Reset Filters');
    userEvent.click(resetButton);
  });
  // Check if filters are reset
  expect(prioritySelect.value).toBe('All');
  expect(statusSelect.value).toBe('All');
  expect(dateSelect.value).toBe('');
  expect(searchInput.value).toBe('');
});
```



```
PASS  frontend/src/components/pages/Overview.test.js
  Overview Component Tests
    √ verifies that clicking 'Reset Filters' resets all filters to their default states (265 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        2 s
```

12.0 Overview Page - Verify Task Information Display

12.1 Brief Description of Test - Ensures that each task box correctly displays essential information such as Status, Priority, Due Date, Assigned Employee, Task Description, and Edit History.

12.2 Valid Test Inputs:

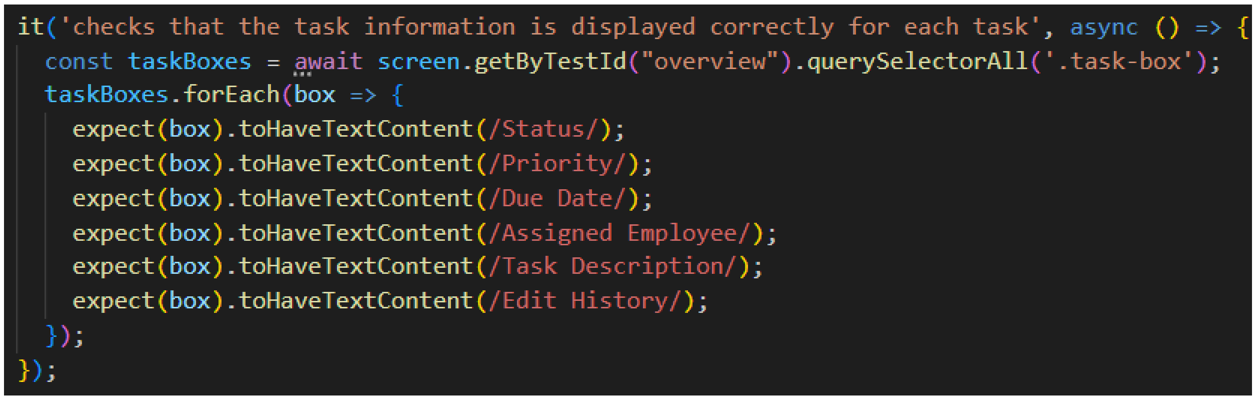
Description: The tester views the task boxes to confirm correct data display.

Values: Display properties of task boxes.

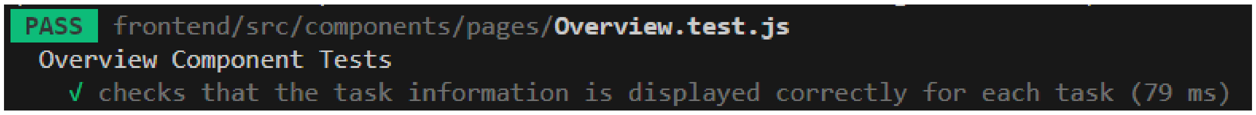
12.3 In-Valid Test Inputs:

Description: Corrupted or incomplete task data.

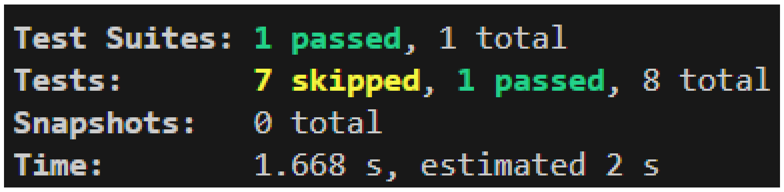
Values: Incomplete information in the task details.
12.4 Tester Name - Madusha Yakupitiyage
12.5 Recorder - Navid Baghaei
12.6 Date Tested - 05/02/24
12.7 Results
12.7.1 Did it pass? Yes



```
it('checks that the task information is displayed correctly for each task', async () => {
  const taskBoxes = await screen.getByTestId("overview").querySelectorAll('.task-box');
  taskBoxes.forEach(box => {
    expect(box).toHaveTextContent(/Status/);
    expect(box).toHaveTextContent(/Priority/);
    expect(box).toHaveTextContent(/Due Date/);
    expect(box).toHaveTextContent(/Assigned Employee/);
    expect(box).toHaveTextContent(/Task Description/);
    expect(box).toHaveTextContent(/Edit History/);
  });
});
```



```
 PASS  frontend/src/components/pages/Overview.test.js
  Overview Component Tests
    √ checks that the task information is displayed correctly for each task (79 ms)
```

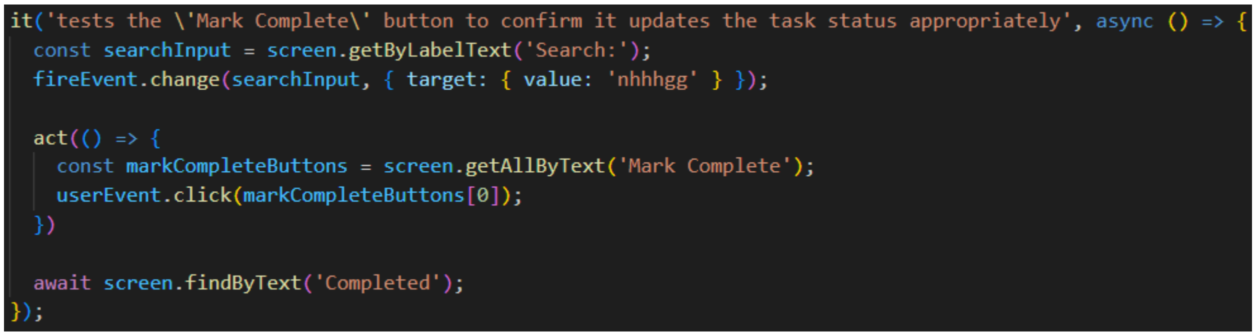


```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        1.668 s, estimated 2 s
```

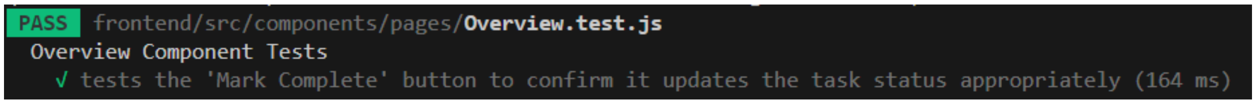
13.0 Overview Page - Functionality of 'Mark Complete' Button. Task status is updated to 'Completed' upon clicking the 'Mark Complete' button.

13.1 Brief Description of Test - Tests whether the 'Mark Complete' button updates the task's status appropriately.
13.2 Valid Test Inputs:
Description: The tester uses the search function to find a specific task and marks it as complete.
Values: Search - 'nhhhgg', Mark Complete action.
13.3 In-Valid Test Inputs:
Description: The tester attempts to mark a task that is already completed as complete again.
Values: Task already marked as completed.
13.4 Tester Name - Madusha Yakupitiyage

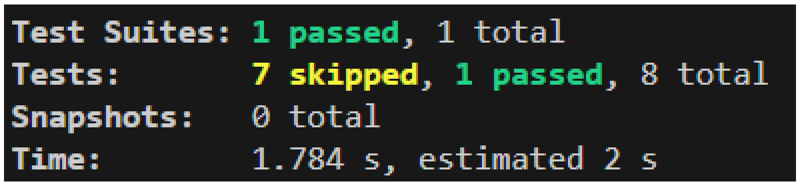
13.5 Recorder - Navid Baghaei
13.6 Date Tested - 05/02/24
13.7 Results
    13.7.1 Did it pass? Yes



```
it('tests the \'Mark Complete\' button to confirm it updates the task status appropriately', async () => {
  const searchInput = screen.getByLabelText('Search:');
  fireEvent.change(searchInput, { target: { value: 'nhhhgg' } });

  act(() => {
    const markCompleteButtons = screen.getAllByText('Mark Complete');
    userEvent.click(markCompleteButtons[0]);
  })

  await screen.findByText('Completed');
});
```



```
PASS  frontend/src/components/pages/Overview.test.js
  Overview Component Tests
    √ tests the 'Mark Complete' button to confirm it updates the task status appropriately (164 ms)
```

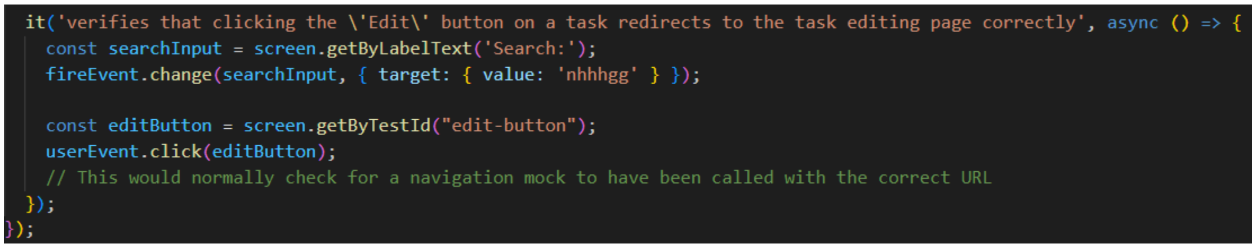


```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        1.784 s, estimated 2 s
```

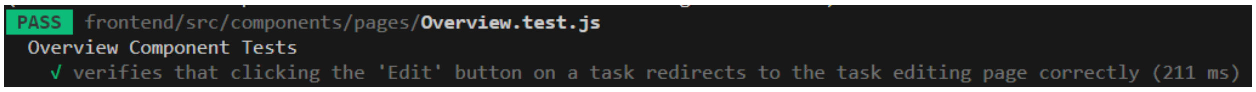
14.0 Overview Page - Redirect to Task Editing Page. The system redirects to the task editing page after 'Edit' button is clicked.

14.1 Brief Description of Test - Verifies that the 'Edit' button on a task correctly redirects to the task editing page.
14.2 Valid Test Inputs:
    Description: The tester searches for a task and clicks the 'Edit' button.
    Values: Search - 'nhhhgg', Edit action.
14.3 In-Valid Test Inputs:
    Description: The tester clicks the 'Edit' button with an invalid or corrupted task identifier.
    Values: Corrupted task data or invalid task ID.
14.4 Tester Name - Madusha Yakupitiyage
14.5 Recorder - Navid Baghaei
14.6 Date Tested - 05/02/24
14.7 Results

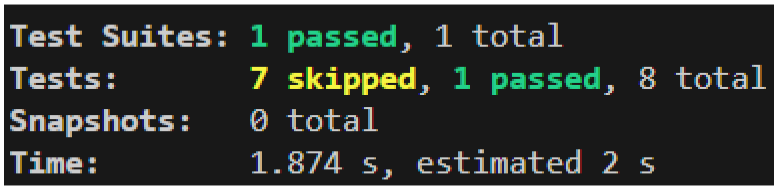
14.7.1 Did it pass? Yes



```
it('verifies that clicking the \'Edit\' button on a task redirects to the task editing page correctly', async () => {
  const searchInput = screen.getByLabelText('Search:');
  fireEvent.change(searchInput, { target: { value: 'nhhhgg' } });

  const editButton = screen.getByTestId("edit-button");
  userEvent.click(editButton);
  // This would normally check for a navigation mock to have been called with the correct URL
});
});
```



```
PASS  frontend/src/components/pages/Overview.test.js
 Overview Component Tests
   √ verifies that clicking the 'Edit' button on a task redirects to the task editing page correctly (211 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       7 skipped, 1 passed, 8 total
Snapshots:   0 total
Time:        1.874 s, estimated 2 s
```

Home Page

15.0 Home Page - Rendering TaskDetails Test. The system renders the tasks correctly

15.1 Brief Description of Test - Verifies that the task details are rendered.

15.2 Valid Test Inputs:

Description: Displays tasks that contain the title of the tasks that are mocked.

Values: title: “Test Task In Progress”, title: “Test Task Past Due”.

15.3 In-Valid Test Inputs:

Description: Fails if task is not one of the mocked tasks.

Values: Corrupted task data or invalid title.

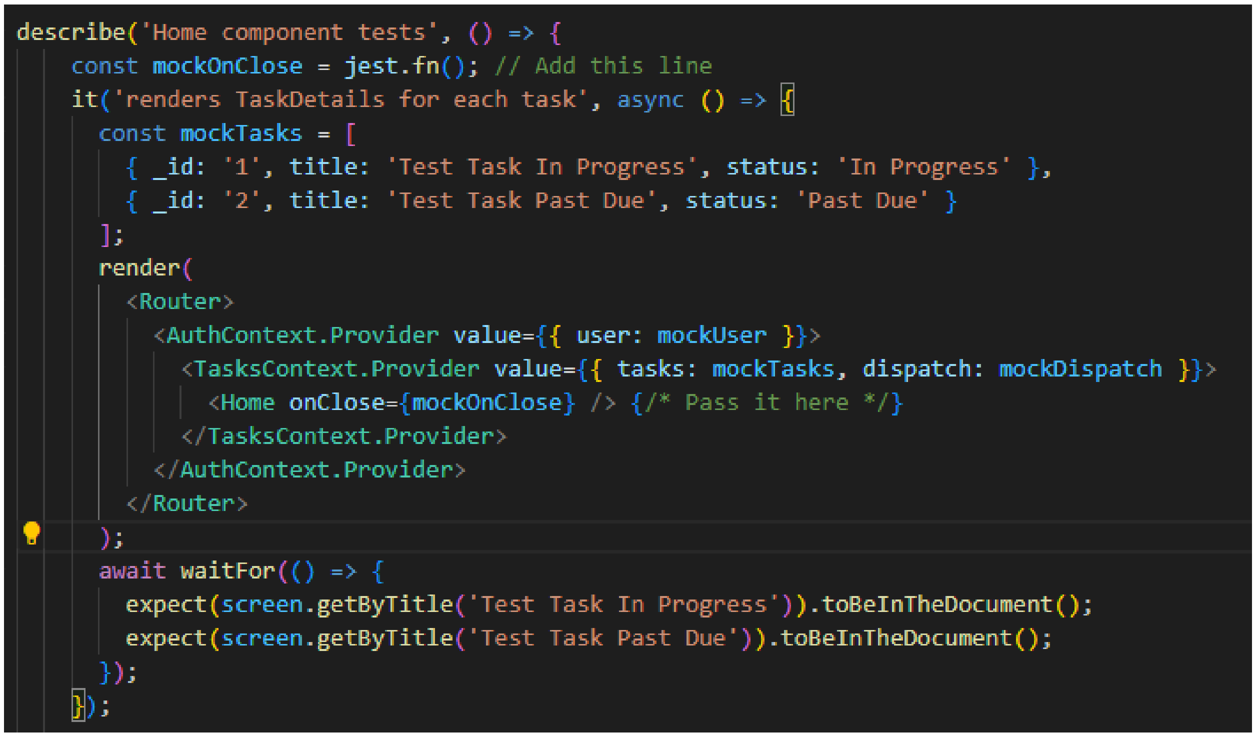
15.4 Tester Name - Arjun Bhargava

15.5 Recorder - John Vardanyan

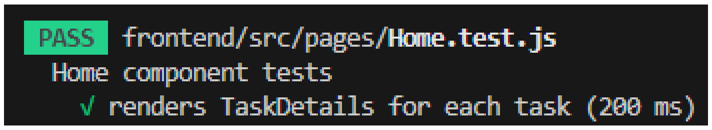
15.6 Date Tested - 05/02/24

15.7 Results

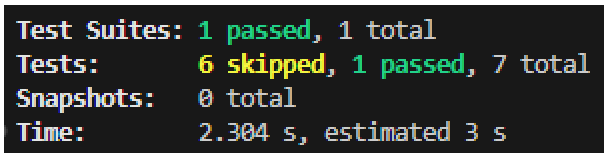
15.7.1 Did it pass? Yes



```
describe('Home component tests', () => {
    const mockOnClose = jest.fn(); // Add this line
    it('renders TaskDetails for each task', async () => {
      const mockTasks = [
        { _id: '1', title: 'Test Task In Progress', status: 'In Progress' },
        { _id: '2', title: 'Test Task Past Due', status: 'Past Due' }
      ];
      render(
        <Router>
          <AuthContext.Provider value={{ user: mockUser }}>
            <TasksContext.Provider value={{ tasks: mockTasks, dispatch: mockDispatch }}>
              <Home onClose={mockOnClose} /> {/* Pass it here */}
            </TasksContext.Provider>
          </AuthContext.Provider>
        </Router>
      );
      await waitFor(() => {
        expect(screen.getByTitle('Test Task In Progress')).toBeInTheDocument();
        expect(screen.getByTitle('Test Task Past Due')).toBeInTheDocument();
      });
    });
```



```
 PASS  frontend/src/pages/Home.test.js
  Home component tests
    √ renders TaskDetails for each task (200 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       6 skipped, 1 passed, 7 total
Snapshots:   0 total
Time:        2.304 s, estimated 3 s
```

16.0 Home Page - High Priority Filter Testing. The system displays the tasks with its corresponding filter that is selected for priority.

16.1 Brief Description of Test - Creates two tasks with different priority values of high and low and displays only tasks with high priority filter being selected.

16.2 Valid Test Inputs:

Description: Shows tasks that are listed as “High Priority” in the priority filter.

Values: Priority: “High”

16.3 In-Valid Test Inputs:

Description: Hides tasks that are NOT listed as “High Priority”. In this case, Low priority tasks.

Values: Priority: Low

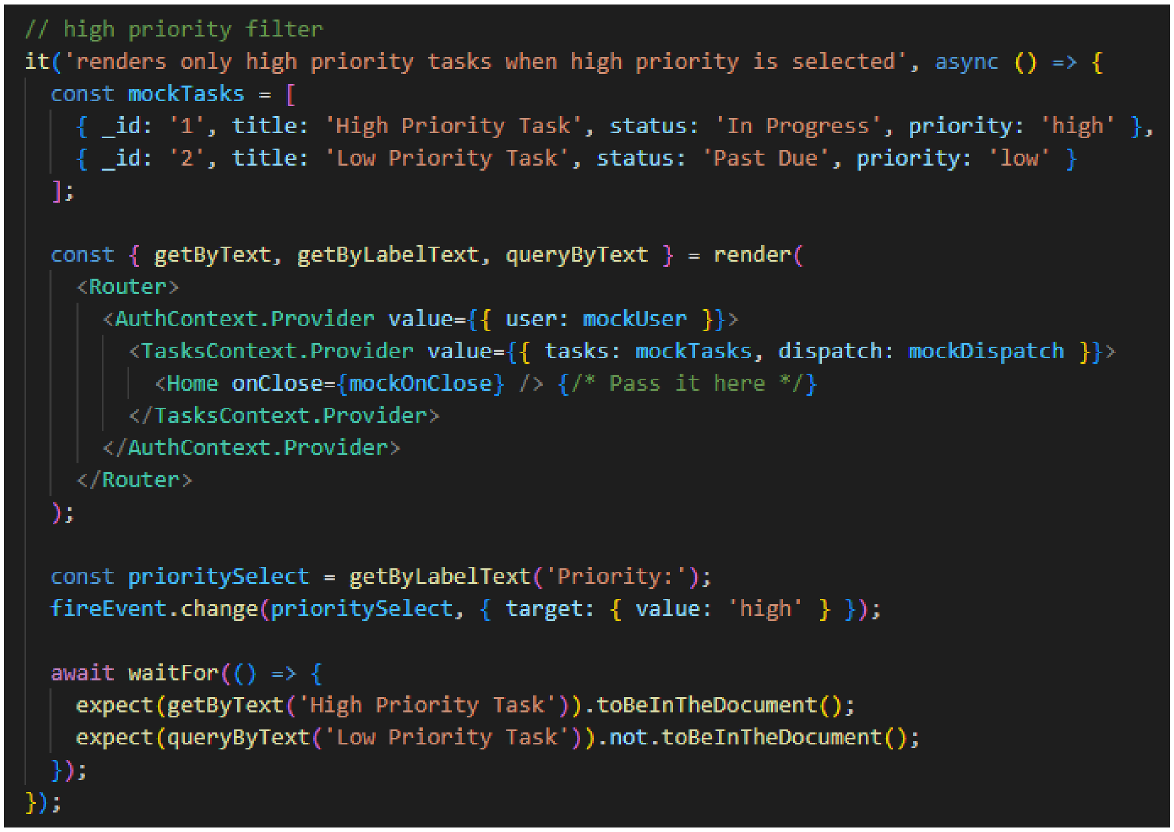
16.4 Tester Name - Arjun Bhargava

16.5 Recorder - John Vardanyan

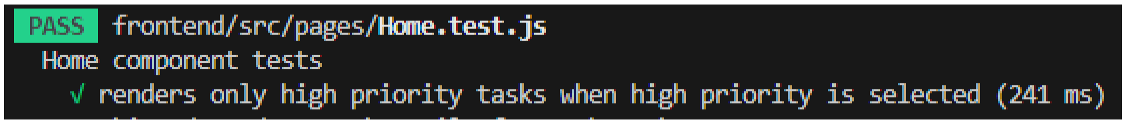
16.6 Date Tested - 05/02/24

16.7 Results

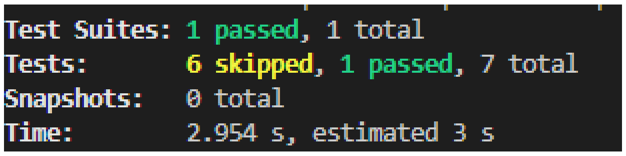
16.7.1 Did it pass? Yes



```
// high priority filter
it('renders only high priority tasks when high priority is selected', async () => {
  const mockTasks = [
    { _id: '1', title: 'High Priority Task', status: 'In Progress', priority: 'high' },
    { _id: '2', title: 'Low Priority Task', status: 'Past Due', priority: 'low' }
  ];

  const { getByText, getByLabelText, queryByText } = render(
    <Router>
      <AuthContext.Provider value={{ user: mockUser }}>
        <TasksContext.Provider value={{ tasks: mockTasks, dispatch: mockDispatch }}>
          <Home onClose={mockOnClose} /> {/* Pass it here */}
        </TasksContext.Provider>
      </AuthContext.Provider>
    </Router>
  );

  const prioritySelect = getByLabelText('Priority:');
  fireEvent.change(prioritySelect, { target: { value: 'high' } });

  await waitFor(() => {
    expect(getByText('High Priority Task')).toBeInTheDocument();
    expect(queryByText('Low Priority Task')).not.toBeInTheDocument();
  });
});
```



```
 PASS  frontend/src/pages/Home.test.js
  Home component tests
    √ renders only high priority tasks when high priority is selected (241 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       6 skipped, 1 passed, 7 total
Snapshots:   0 total
Time:        2.954 s, estimated 3 s
```

17.0 Home Page - Medium Priority Filter Testing. The system displays the tasks with its corresponding filter that is selected for priority.

17.1 Brief Description of Test - Creates two tasks with different priority values of Medium and Low and displays only tasks with Medium priority filter being selected.

17.2 Valid Test Inputs:
Description: Shows tasks that are listed as “Medium Priority” in the priority filter.
Values: Priority: “Medium”

17.3 In-Valid Test Inputs:
Description: Hides tasks that are NOT listed as “Medium Priority”. In this case, Low priority tasks.
Values: Priority: Low

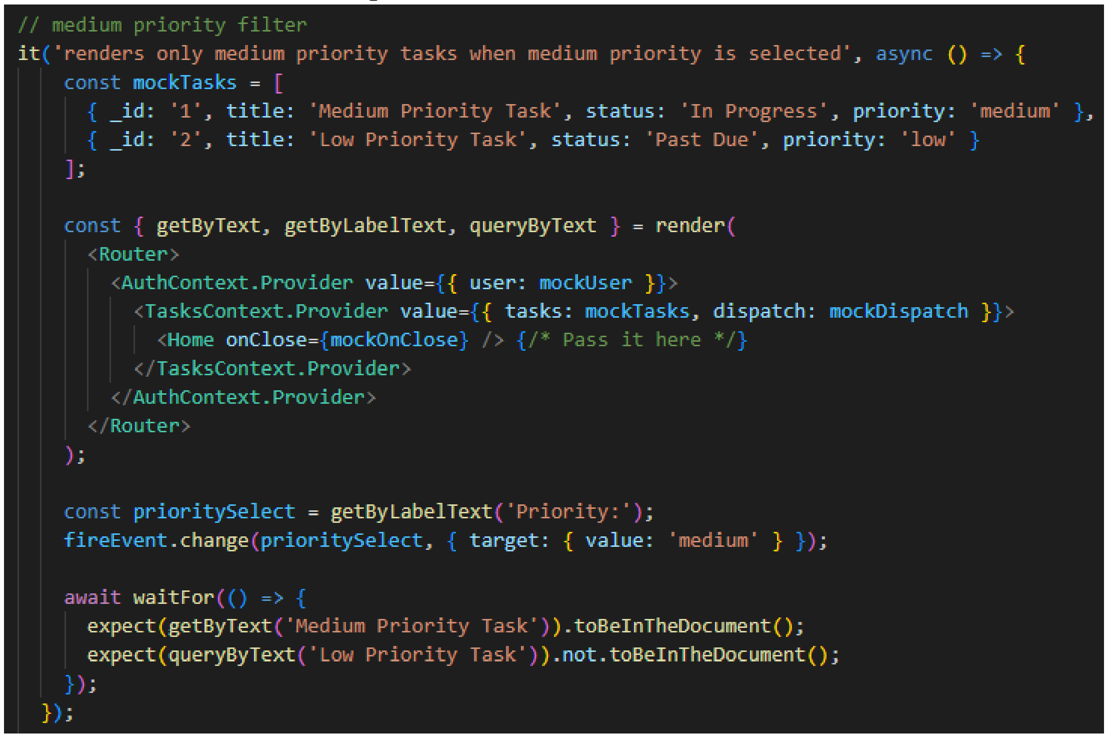
17.4 Tester Name - Arjun Bhargava

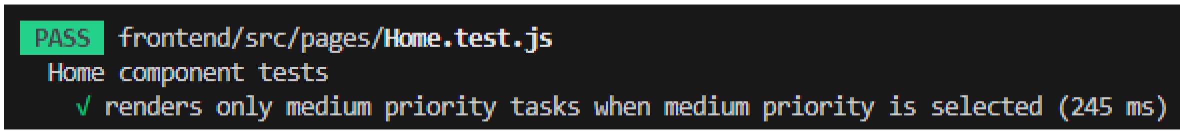
17.5 Recorder - John Vardanyan

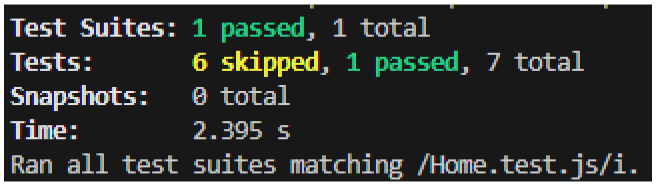
17.6 Date Tested - 05/02/24

17.7 Results

17.7.1 Did it pass? Yes


```
// medium priority filter
it('renders only medium priority tasks when medium priority is selected', async () => {
    const mockTasks = [
      { _id: '1', title: 'Medium Priority Task', status: 'In Progress', priority: 'medium' },
      { _id: '2', title: 'Low Priority Task', status: 'Past Due', priority: 'low' }
    ];

    const { getByText, getByLabelText, queryByText } = render(
      <Router>
        <AuthContext.Provider value={{ user: mockUser }}>
          <TasksContext.Provider value={{ tasks: mockTasks, dispatch: mockDispatch }}>
            <Home onClose={mockOnClose} /> {/* Pass it here */}
          </TasksContext.Provider>
        </AuthContext.Provider>
      </Router>
    );

    const prioritySelect = getByLabelText('Priority:');
    fireEvent.change(prioritySelect, { target: { value: 'medium' } });

    await waitFor(() => {
      expect(getByText('Medium Priority Task')).toBeInTheDocument();
      expect(queryByText('Low Priority Task')).not.toBeInTheDocument();
    });
  });
```


```
 PASS  frontend/src/pages/Home.test.js
  Home component tests
    √ renders only medium priority tasks when medium priority is selected (245 ms)
```


```
Test Suites: 1 passed, 1 total
Tests:       6 skipped, 1 passed, 7 total
Snapshots:   0 total
Time:        2.395 s
Ran all test suites matching /Home.test.js/i.
```

18.0 Home Page - Low Priority Filter Testing. The system displays the tasks with its corresponding filter that is selected for priority.

18.1 Brief Description of Test - Creates two tasks with different priority values of Medium and Low and displays only tasks with Low priority filter being selected.

18.2 Valid Test Inputs:
Description: Shows tasks that are listed as “Low Priority” in the priority filter.
Values: Priority: “Low”

18.3 In-Valid Test Inputs:
Description: Hides tasks that are NOT listed as “Low Priority”. In this case, Medium priority tasks.
Values: Priority: “Medium”

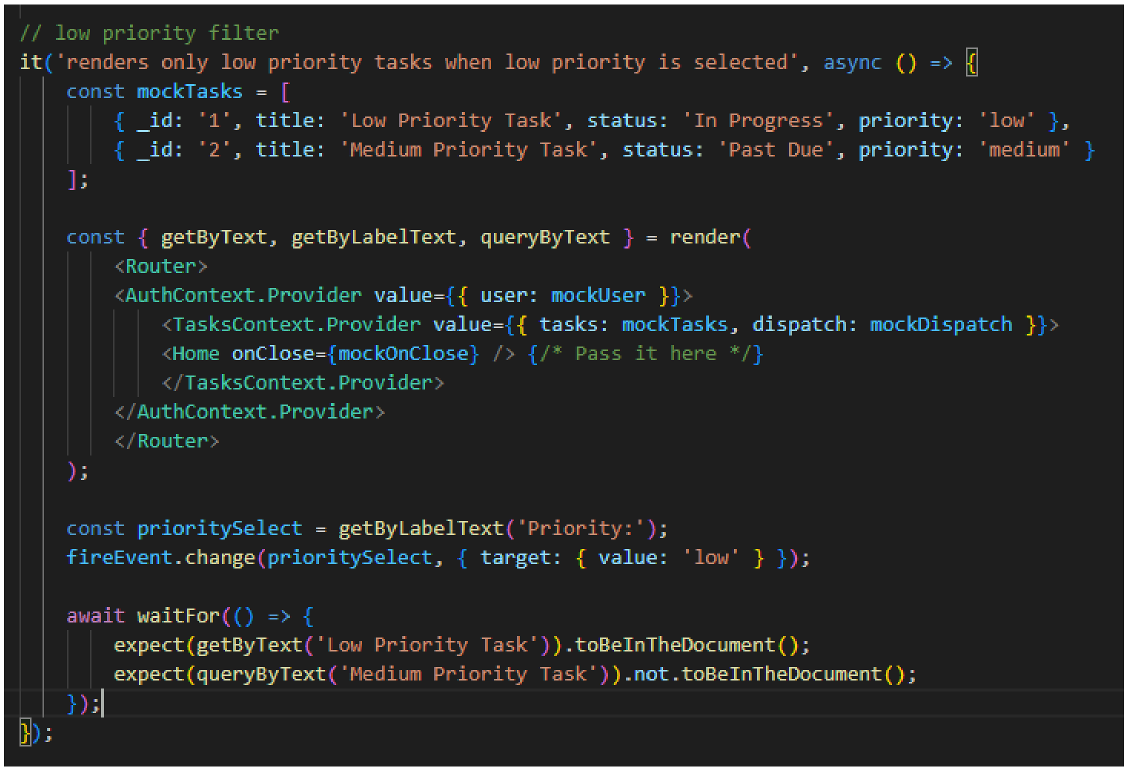
18.4 Tester Name - Arjun Bhargava

18.5 Recorder - John Vardanyan

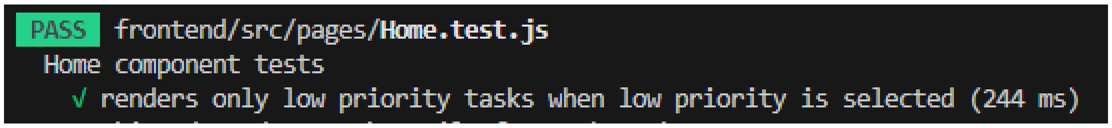
18.6 Date Tested - 05/02/24

18.7 Results

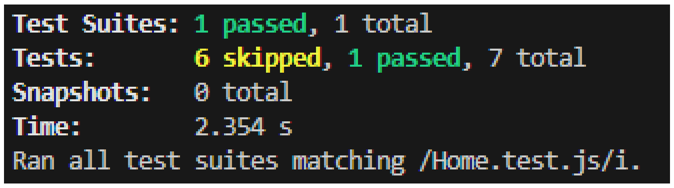
18.7.1 Did it pass? Yes



```
// low priority filter
it('renders only low priority tasks when low priority is selected', async () => {
    const mockTasks = [
        { _id: '1', title: 'Low Priority Task', status: 'In Progress', priority: 'low' },
        { _id: '2', title: 'Medium Priority Task', status: 'Past Due', priority: 'medium' }
    ];

    const { getByText, getByLabelText, queryByText } = render(
        <Router>
        <AuthContext.Provider value={{ user: mockUser }}>
            <TasksContext.Provider value={{ tasks: mockTasks, dispatch: mockDispatch }}>
            <Home onClose={mockOnClose} /> {/* Pass it here */}
            </TasksContext.Provider>
        </AuthContext.Provider>
        </Router>
    );

    const prioritySelect = getByLabelText('Priority:');
    fireEvent.change(prioritySelect, { target: { value: 'low' } });

    await waitFor(() => {
        expect(getByText('Low Priority Task')).toBeInTheDocument();
        expect(queryByText('Medium Priority Task')).not.toBeInTheDocument();
    });
});
```



```
 PASS  frontend/src/pages/Home.test.js
  Home component tests
    √ renders only low priority tasks when low priority is selected (244 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       6 skipped, 1 passed, 7 total
Snapshots:   0 total
Time:        2.354 s
Ran all test suites matching /Home.test.js/i.
```

19.0 Home Page - In Progress Filter Testing. The system displays the tasks with its corresponding filter that is selected for status.

19.1 Brief Description of Test - Displays the tasks that have their status set to in progress.

19.2 Valid Test Inputs:
Description: Shows tasks that are listed as “In Progress” in the priority filter.
Values: Status: “In Progress”

19.3 In-Valid Test Inputs:
Description: Hides tasks that are NOT listed as “In Progress”. In this case, Past Due tasks.
Values: Status: In Progress

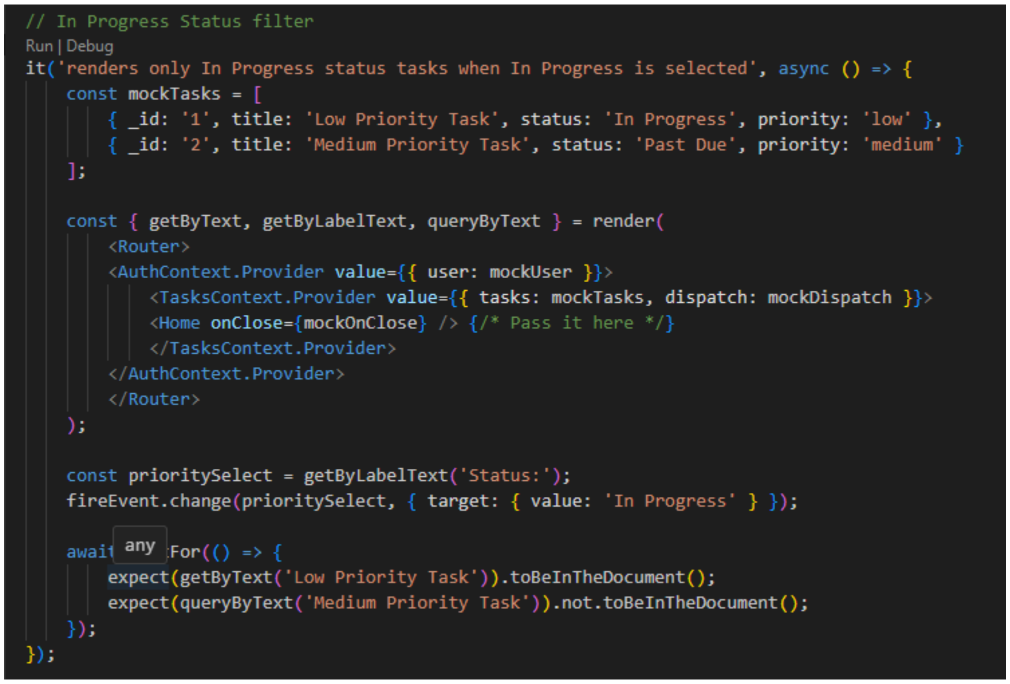
19.4 Tester Name -John Vardanyan

19.5 Recorder - Arjun Bhargava

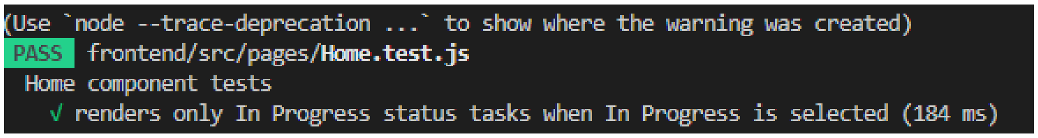
19.6 Date Tested - 05/02/24

19.7 Results

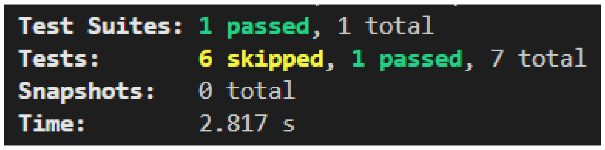
19.7.1 Did it pass? Yes



```
// In Progress Status filter
Run | Debug
it('renders only In Progress status tasks when In Progress is selected', async () => {
    const mockTasks = [
        { _id: '1', title: 'Low Priority Task', status: 'In Progress', priority: 'low' },
        { _id: '2', title: 'Medium Priority Task', status: 'Past Due', priority: 'medium' }
    ];

    const { getByText, getByLabelText, queryByText } = render(
        <Router>
        <AuthContext.Provider value={{ user: mockUser }}>
            <TasksContext.Provider value={{ tasks: mockTasks, dispatch: mockDispatch }}>
            <Home onClose={mockOnClose} /> {/* Pass it here */}
            </TasksContext.Provider>
        </AuthContext.Provider>
        </Router>
    );

    const prioritySelect = getByLabelText('Status:');
    fireEvent.change(prioritySelect, { target: { value: 'In Progress' } });

    await any For(() => {
        expect(getByText('Low Priority Task')).toBeInTheDocument();
        expect(queryByText('Medium Priority Task')).not.toBeInTheDocument();
    });
});
```



```
(Use `node --trace-deprecation ...` to show where the warning was created)
 PASS  frontend/src/pages/Home.test.js
  Home component tests
    √ renders only In Progress status tasks when In Progress is selected (184 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       6 skipped, 1 passed, 7 total
Snapshots:   0 total
Time:        2.817 s
```

20.0 Home Page - Past Due Filter Testing. The system displays the tasks with its corresponding filter that is selected for status.

20.1 Brief Description of Test - Displays the tasks that have their status set to past due.

20.2 Valid Test Inputs:

Description: Shows tasks that are listed as “Past Due” in the priority filter.

Values: Status: “Past Due”

20.3 In-Valid Test Inputs:

Description: Hides tasks that are NOT listed as “Past Due”. In this case, In Progress tasks.

Values: Status: Past Due

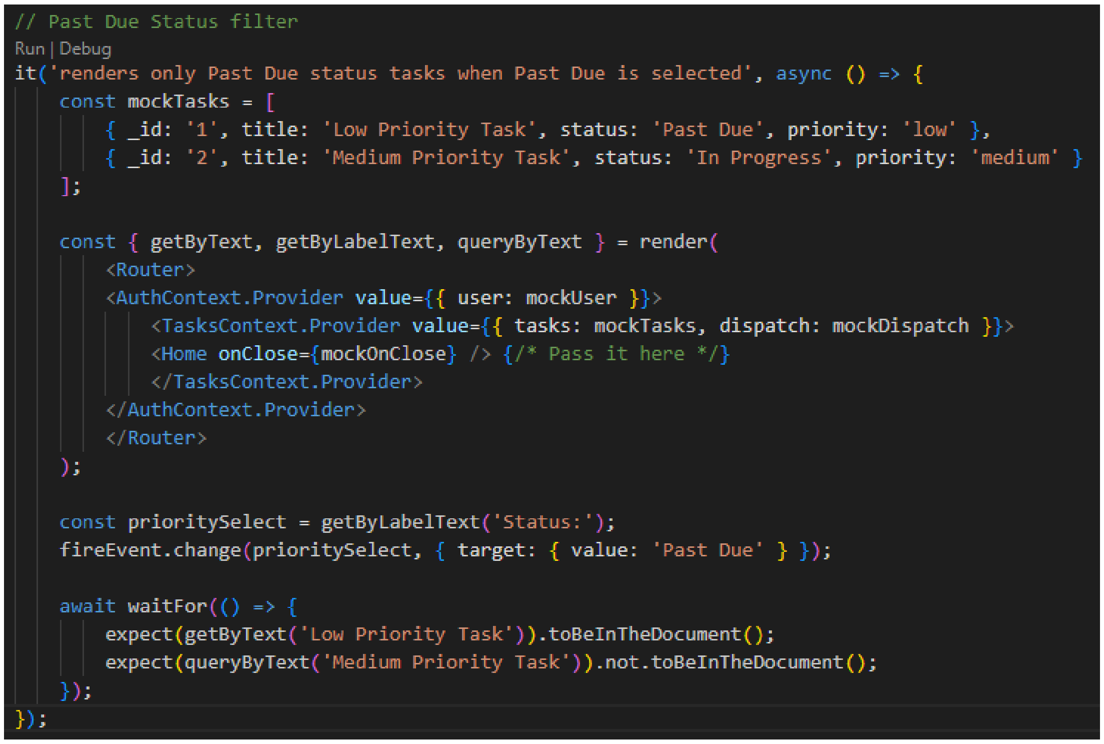
20.4 Tester Name -John Vardanyan

20.5 Recorder - Arjun Bhargava

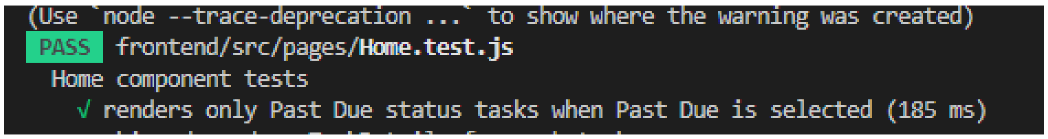
20.6 Date Tested - 05/02/24

20.7 Results

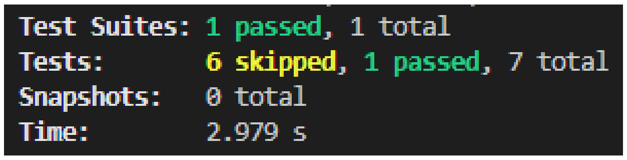
20.7.1 Did it pass? Yes



```
// Past Due Status filter
Run | Debug
it('renders only Past Due status tasks when Past Due is selected', async () => {
    const mockTasks = [
        { _id: '1', title: 'Low Priority Task', status: 'Past Due', priority: 'low' },
        { _id: '2', title: 'Medium Priority Task', status: 'In Progress', priority: 'medium' }
    ];

    const { getByText, getByLabelText, queryByText } = render(
        <Router>
        <AuthContext.Provider value={{ user: mockUser }}>
            <TasksContext.Provider value={{ tasks: mockTasks, dispatch: mockDispatch }}>
            <Home onClose={mockOnClose} /> {/* Pass it here */}
            </TasksContext.Provider>
        </AuthContext.Provider>
        </Router>
    );

    const prioritySelect = getByLabelText('Status:');
    fireEvent.change(prioritySelect, { target: { value: 'Past Due' } });

    await waitFor(() => {
        expect(getByText('Low Priority Task')).toBeInTheDocument();
        expect(queryByText('Medium Priority Task')).not.toBeInTheDocument();
    });
});
```



```
 (Use `node --trace-deprecation ...` to show where the warning was created)
 PASS  frontend/src/pages/Home.test.js
  Home component tests
    √ renders only Past Due status tasks when Past Due is selected (185 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       6 skipped, 1 passed, 7 total
Snapshots:   0 total
Time:        2.979 s
```

21.0 Home Page - Due Date Filter Testing. The system displays the tasks with its corresponding task that is selected for Due Date.

21.1 Brief Description of Test - Displays the tasks that have their status set to specific due date.

21.2 Valid Test Inputs:

Description: Shows tasks based on Due Date input.

Values: Due Date: “2024-05-28”

21.3 In-Valid Test Inputs:

Description: Hides tasks that are NOT listed as the specific ‘Due Date’. In this case, a task titled “Medium Priority Task”.

Values: Due Date: any date that is **not** “2024-05-28”.

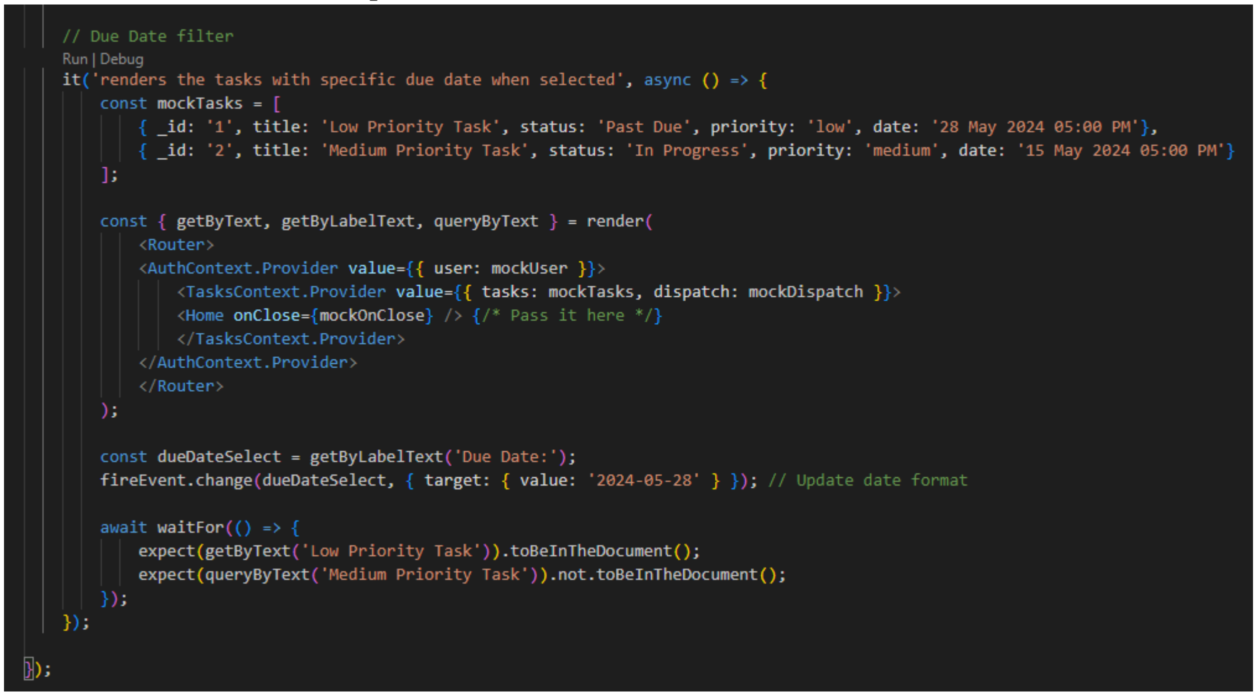
21.4 Tester Name -John Vardanyan

21.5 Recorder - Arjun Bhargava

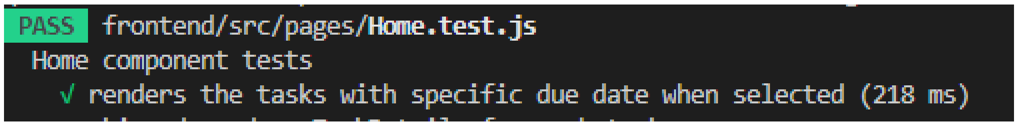
21.6 Date Tested - 05/02/24

21.7 Results

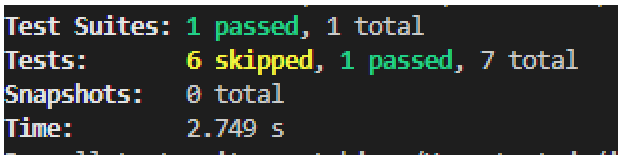
21.7.1 Did it pass? Yes



```
// Due Date filter
Run | Debug
it('renders the tasks with specific due date when selected', async () => {
  const mockTasks = [
    { _id: '1', title: 'Low Priority Task', status: 'Past Due', priority: 'low', date: '28 May 2024 05:00 PM'},
    { _id: '2', title: 'Medium Priority Task', status: 'In Progress', priority: 'medium', date: '15 May 2024 05:00 PM'}
  ];

  const { getByText, getByLabelText, queryByText } = render(
    <Router>
    <AuthContext.Provider value={{ user: mockUser }}>
      <TasksContext.Provider value={{ tasks: mockTasks, dispatch: mockDispatch }}>
      <Home onClose={mockOnClose} /> {/* Pass it here */}
      </TasksContext.Provider>
    </AuthContext.Provider>
    </Router>
  );

  const dueDateSelect = getByLabelText('Due Date:');
  fireEvent.change(dueDateSelect, { target: { value: '2024-05-28' } }); // Update date format

  await waitFor(() => {
    expect(getByText('Low Priority Task')).toBeInTheDocument();
    expect(queryByText('Medium Priority Task')).not.toBeInTheDocument();
  });
});

});
```



```
PASS  frontend/src/pages/Home.test.js
  Home component tests
    √ renders the tasks with specific due date when selected (218 ms)
```



```
Test Suites: 1 passed, 1 total
Tests:       6 skipped, 1 passed, 7 total
Snapshots:   0 total
Time:        2.749 s
```

Calendar Page

22.0 Calendar Page - Displays today's date correctly

22.1 Description: Tests if today’s date is correctly displayed in the side component on the calendar page.

22.2 Valid Test Input:

- The only valid string would be “Today is: May 3, 2024” since we are expecting today’s date to be displayed. At the time of testing, “today’s” date is May 3, 2024.

22.3 In-Valid Test Input:

- Any day that is not May 3, 2024 in our example which could be something like December 1,2025

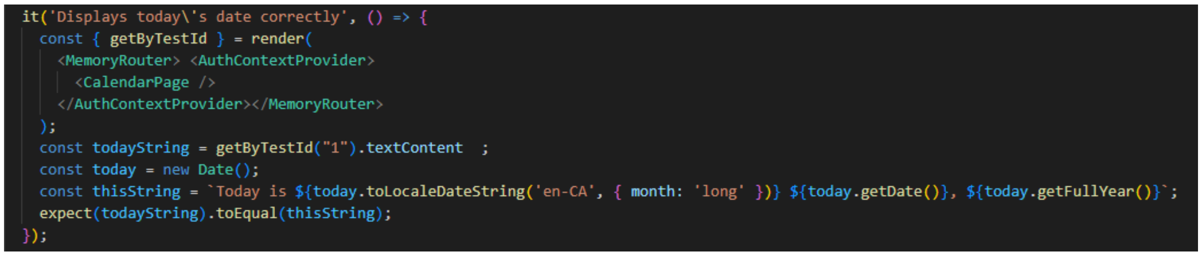
22.4 Tester Name: Nikolay Chkhaylo

22.5 Recorder/Observer Name: Jeremy Vuong

22.6 Date Tested: 05/03/2024

22.7 Results

22.7.1 Did it pass? Yes



```
it('Displays today\'s date correctly', () => {
  const { getByTestId } = render(
    <MemoryRouter> <AuthContextProvider>
      <CalendarPage />
    </AuthContextProvider></MemoryRouter>
  );
  const todayString = getByTestId("1").textContent  ;
  const today = new Date();
  const thisString = `Today is ${today.toLocaleDateString('en-CA', { month: 'long' })} ${today.getDate()}, ${today.getFullYear()}`;
  expect(todayString).toEqual(thisString);
});
```

23.0 Calendar Page - Displays the agenda labels correctly

23.1 Description: Tests if agenda labels are displayed correctly on the calendar page.

23.2 Valid Test Inputs:

- Inputs that test today's date

23.3 In-Valid Test Inputs:

- Inputs that test any day except for today's date since our test is testing the current date.

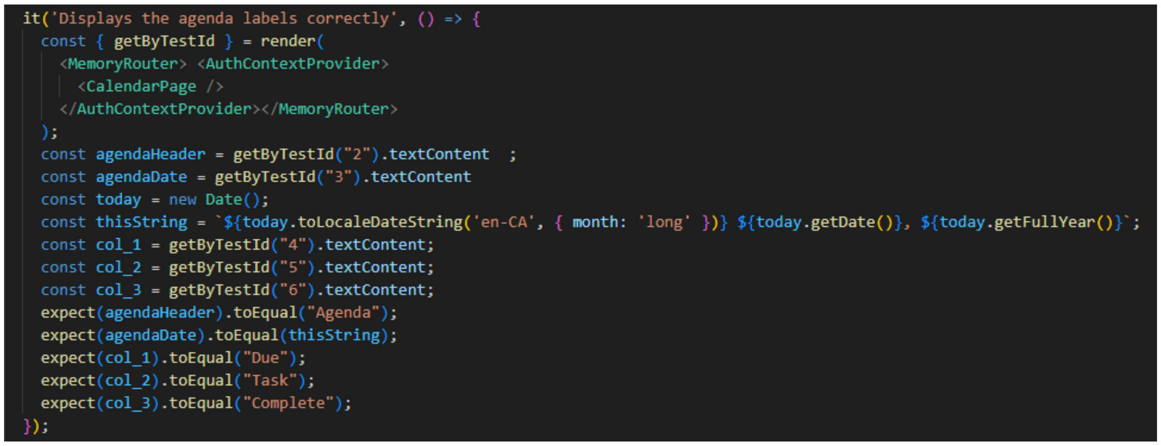
23.4 Tester Name: Nikolay Chkhaylo

23.5 Recorder/Observer Name: Jeremy Vuong

23.6 Date Tested: 05/03/2024

23.7 Results

23.7.1 Did it pass? Yes



```
it('Displays the agenda labels correctly', () => {
  const { getByTestId } = render(
    <MemoryRouter> <AuthContextProvider>
      <CalendarPage />
    </AuthContextProvider></MemoryRouter>
  );
  const agendaHeader = getByTestId("2").textContent  ;
  const agendaDate = getByTestId("3").textContent
  const today = new Date();
  const thisString = `${today.toLocaleDateString('en-CA', { month: 'long' })} ${today.getDate()}, ${today.getFullYear()}`;
  const col_1 = getByTestId("4").textContent;
  const col_2 = getByTestId("5").textContent;
  const col_3 = getByTestId("6").textContent;
  expect(agendaHeader).toEqual("Agenda");
  expect(agendaDate).toEqual(thisString);
  expect(col_1).toEqual("Due");
  expect(col_2).toEqual("Task");
  expect(col_3).toEqual("Complete");
});
```

24.0 Calendar Page - React date picker functions as expected

24.1 Description: Tests the functionality of the previous month button, next month button, and selectable dates in the react date picker component of the calendar page.

- The previous month button (back arrow) should navigate the date of react date picker to the previous month,
- The next month button (forward arrow) should navigate the date of react date picker to the next month,
- Selecting a date on react date picker sets the date of the agenda and calendar to the corresponding date.

24.2 Valid Test Inputs:

- Selecting a date within the React date picker should update the selected date in the calendar/agenda component to match the selected date in the date picker.

24.3 In-Valid Test Inputs:

- Selecting a date outside of the selectable range in the React date picker (if there are restrictions on which dates can be selected) should not update the selected date in the calendar/agenda component.

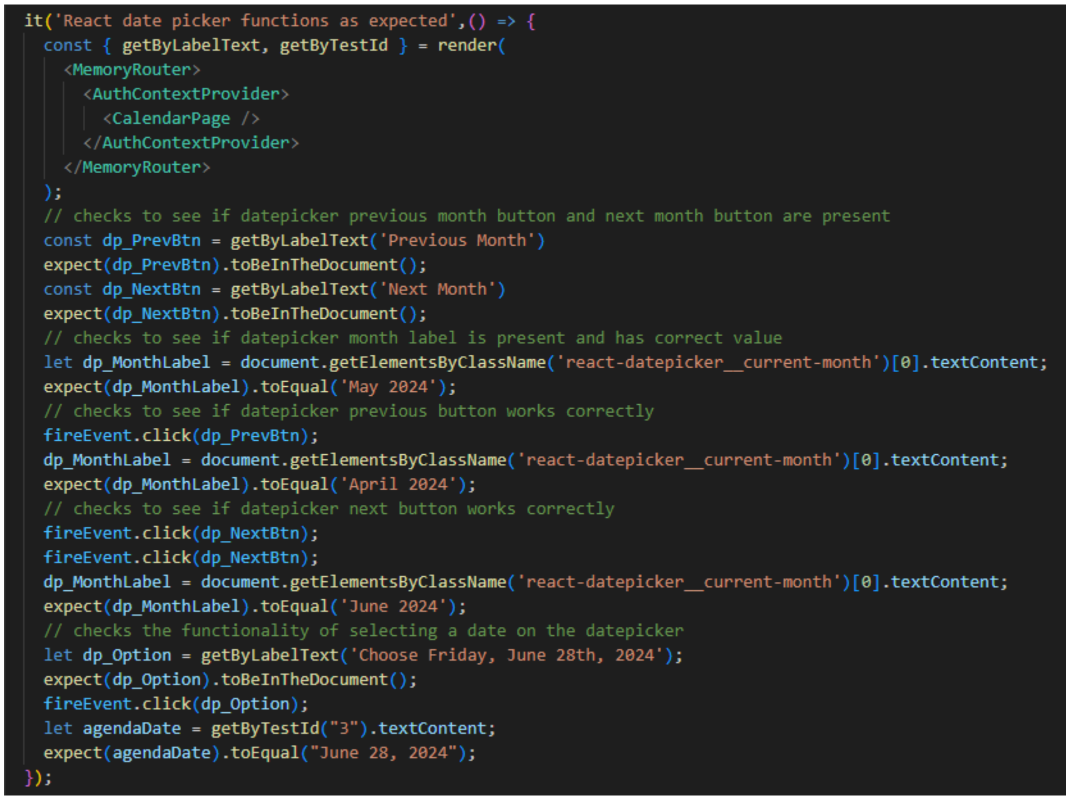
24.4 Tester Name: Jeremy Vuong

24.5 Recorder/Observer Name: Nikolay Chkhaylo

24.6 Date Tested: 05/03/2024

24.7 Results

24.7.1 Did it pass? Yes



```
it('React date picker functions as expected',() => {
  const { getByLabelText, getByTestId } = render(
    <MemoryRouter>
      <AuthContextProvider>
        <CalendarPage />
      </AuthContextProvider>
    </MemoryRouter>
  );
  // checks to see if datepicker previous month button and next month button are present
  const dp_PrevBtn = getByLabelText('Previous Month')
  expect(dp_PrevBtn).toBeInTheDocument();
  const dp_NextBtn = getByLabelText('Next Month')
  expect(dp_NextBtn).toBeInTheDocument();
  // checks to see if datepicker month label is present and has correct value
  let dp_MonthLabel = document.getElementsByClassName('react-datepicker__current-month')[0].textContent;
  expect(dp_MonthLabel).toEqual('May 2024');
  // checks to see if datepicker previous button works correctly
  fireEvent.click(dp_PrevBtn);
  dp_MonthLabel = document.getElementsByClassName('react-datepicker__current-month')[0].textContent;
  expect(dp_MonthLabel).toEqual('April 2024');
  // checks to see if datepicker next button works correctly
  fireEvent.click(dp_NextBtn);
  fireEvent.click(dp_NextBtn);
  dp_MonthLabel = document.getElementsByClassName('react-datepicker__current-month')[0].textContent;
  expect(dp_MonthLabel).toEqual('June 2024');
  // checks the functionality of selecting a date on the datepicker
  let dp_Option = getByLabelText('Choose Friday, June 28th, 2024');
  expect(dp_Option).toBeInTheDocument();
  fireEvent.click(dp_Option);
  let agendaDate = getByTestId("3").textContent;
  expect(agendaDate).toEqual("June 28, 2024");
});
```

25.0 Calendar Page - All react big calendar toolbar buttons work as expected

25.1 Description: Tests the functionality of all the buttons in the react big calendar toolbar.

- The ‘Today’ button should set the calendar’s date to the current date.
- The ‘Back’ button should set the calendar’s date to the previous month, week, or day depending on which view is currently selected.
- The ‘Next’ button should set the calendar’s date to the next month, week, or day depending on which view is currently selected.
- The ‘Month’ button should change the current view of the calendar to the month view.
- The ‘Week’ button should change the current view of the calendar to the week view.
- The ‘Day’ button should change the current view of the calendar to the day view.
- When the date of the calendar is changed, the react big calendar toolbar label will change to display the selected month, week, or day depending on which view is currently selected.

25.2 Valid Test Inputs:

- selecting buttons that are exclusively found on the Calendar page such as Month, Day, Week, etc..

25.3 In-valid Test Inputs:

- Choosing Buttons that are not on the calendar page

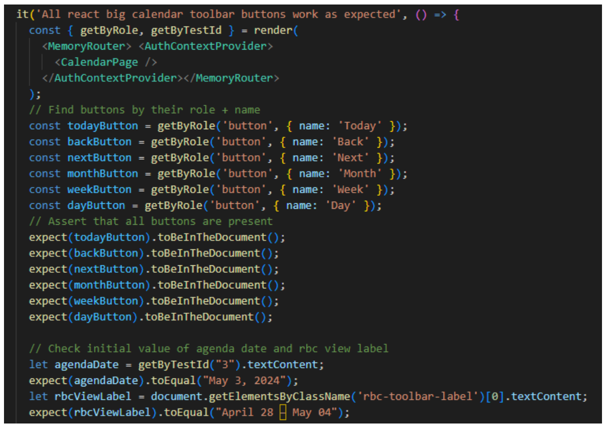
25.4 Tester Name: Jeremy Vuong

25.5 Recorder/Observer Name: Nikolay Chkhaylo

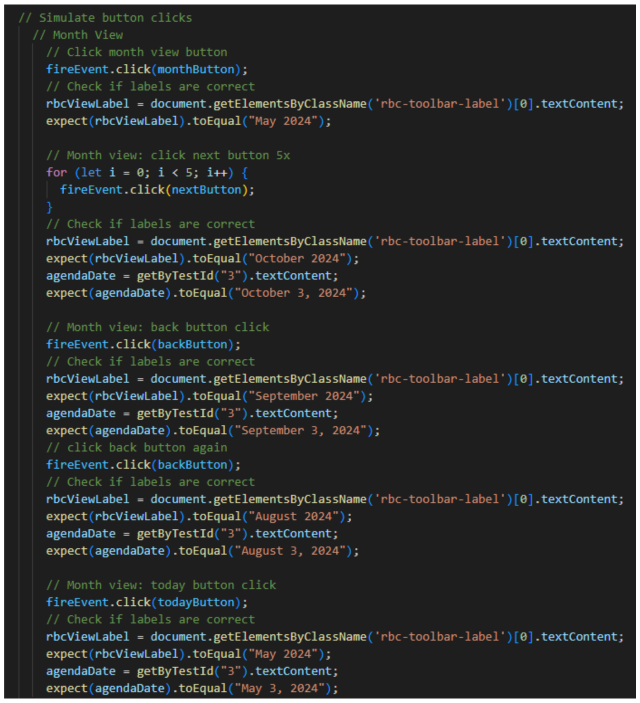
25.6 Date Tested: 05/03/2024

25.7 Results

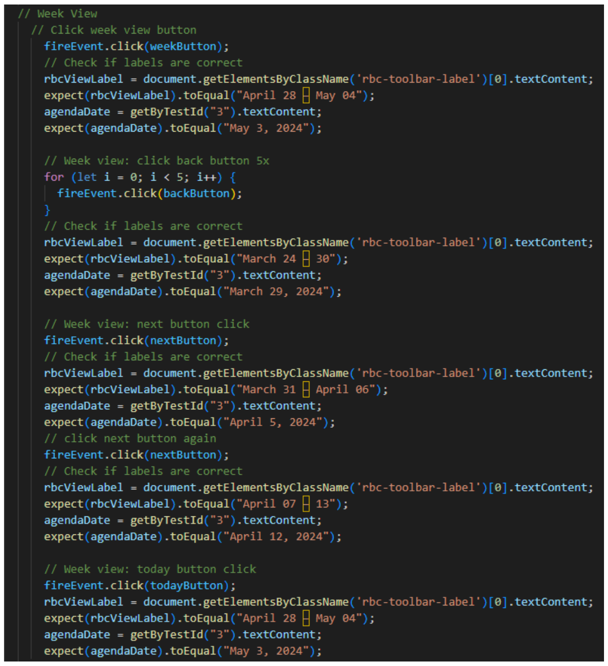
25.7.1 Did it pass? Yes



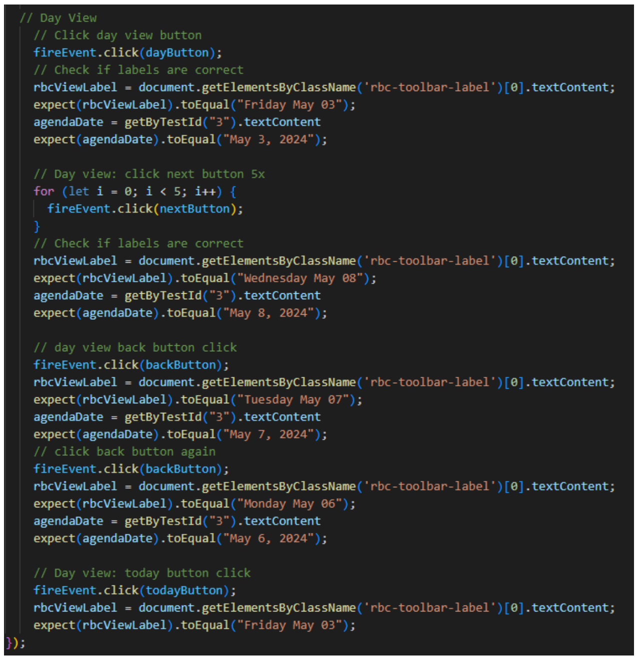
```
it('All react big calendar toolbar buttons work as expected', () => {
  const { getByRole, getByTestId } = render(
    <MemoryRouter> <AuthContextProvider>
      <CalendarPage />
    </AuthContextProvider></MemoryRouter>
  );
  // Find buttons by their role + name
  const todayButton = getByRole('button', { name: 'Today' });
  const backButton = getByRole('button', { name: 'Back' });
  const nextButton = getByRole('button', { name: 'Next' });
  const monthButton = getByRole('button', { name: 'Month' });
  const weekButton = getByRole('button', { name: 'Week' });
  const dayButton = getByRole('button', { name: 'Day' });
  // Assert that all buttons are present
  expect(todayButton).toBeInTheDocument();
  expect(backButton).toBeInTheDocument();
  expect(nextButton).toBeInTheDocument();
  expect(monthButton).toBeInTheDocument();
  expect(weekButton).toBeInTheDocument();
  expect(dayButton).toBeInTheDocument();

  // Check initial value of agenda date and rbc view label
  let agendaDate = getByTestId("3").textContent;
  expect(agendaDate).toEqual("May 3, 2024");
  let rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
  expect(rbcViewLabel).toEqual("April 28 ▯ May 04");
```



```
// Simulate button clicks
  // Month View
    // Click month view button
    fireEvent.click(monthButton);
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("May 2024");

    // Month view: click next button 5x
    for (let i = 0; i < 5; i++) {
      fireEvent.click(nextButton);
    }
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("October 2024");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("October 3, 2024");

    // Month view: back button click
    fireEvent.click(backButton);
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("September 2024");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("September 3, 2024");
    // click back button again
    fireEvent.click(backButton);
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("August 2024");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("August 3, 2024");

    // Month view: today button click
    fireEvent.click(todayButton);
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("May 2024");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("May 3, 2024");
```



```
// Week View
  // Click week view button
    fireEvent.click(weekButton);
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("April 28 ▯ May 04");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("May 3, 2024");

    // Week view: click back button 5x
    for (let i = 0; i < 5; i++) {
      fireEvent.click(backButton);
    }
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("March 24 ▯ 30");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("March 29, 2024");

    // Week view: next button click
    fireEvent.click(nextButton);
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("March 31 ▯ April 06");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("April 5, 2024");
    // click next button again
    fireEvent.click(nextButton);
    // Check if labels are correct
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("April 07 ▯ 13");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("April 12, 2024");

    // Week view: today button click
    fireEvent.click(todayButton);
    rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
    expect(rbcViewLabel).toEqual("April 28 ▯ May 04");
    agendaDate = getByTestId("3").textContent;
    expect(agendaDate).toEqual("May 3, 2024");
```



```
// Day View
  // Click day view button
  fireEvent.click(dayButton);
  // Check if labels are correct
  rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
  expect(rbcViewLabel).toEqual("Friday May 03");
  agendaDate = getByTestId("3").textContent
  expect(agendaDate).toEqual("May 3, 2024");

  // Day view: click next button 5x
  for (let i = 0; i < 5; i++) {
    fireEvent.click(nextButton);
  }
  // Check if labels are correct
  rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
  expect(rbcViewLabel).toEqual("Wednesday May 08");
  agendaDate = getByTestId("3").textContent
  expect(agendaDate).toEqual("May 8, 2024");

  // day view back button click
  fireEvent.click(backButton);
  rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
  expect(rbcViewLabel).toEqual("Tuesday May 07");
  agendaDate = getByTestId("3").textContent
  expect(agendaDate).toEqual("May 7, 2024");
  // click back button again
  fireEvent.click(backButton);
  rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
  expect(rbcViewLabel).toEqual("Monday May 06");
  agendaDate = getByTestId("3").textContent
  expect(agendaDate).toEqual("May 6, 2024");

  // Day view: today button click
  fireEvent.click(todayButton);
  rbcViewLabel = document.getElementsByClassName('rbc-toolbar-label')[0].textContent;
  expect(rbcViewLabel).toEqual("Friday May 03");
});
```

26.0 Calendar Page - Selecting a date on react big calendar works correctly

26.1 Description: Tests the functionality of selecting a date using the react big calendar.

- In the ‘week’ view, clicking on the column header will change the date of the agenda to the corresponding date.
- In the ‘month’ view, clicking of a day number will change the date of the agenda to the corresponding date.

26.2 Valid Test Inputs:

-

26.3 In-valid Test Inputs:

-

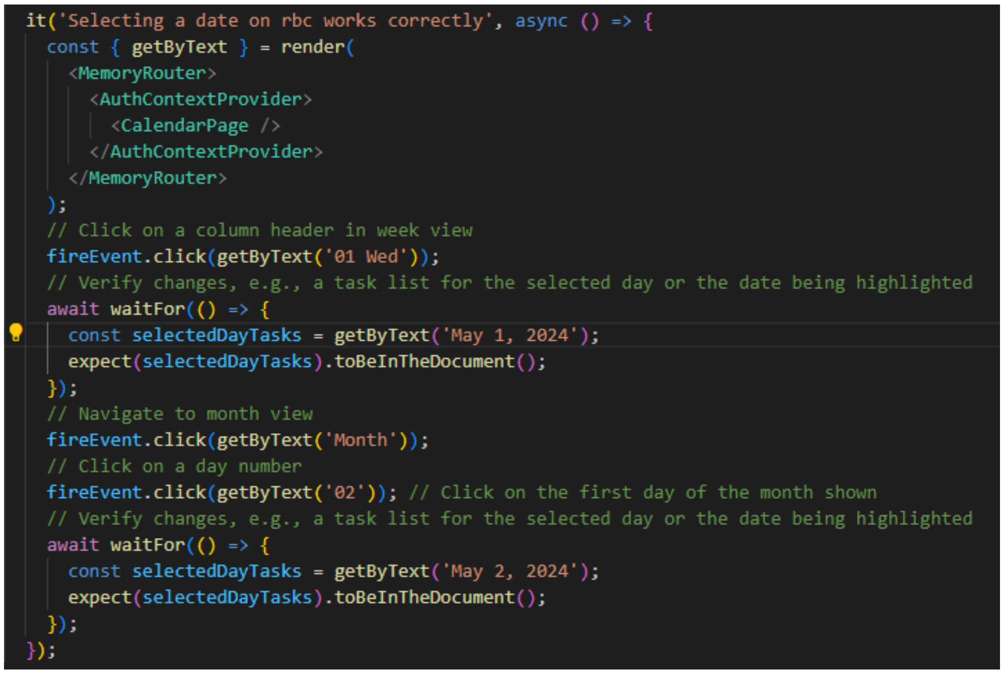
26.4 Tester Name: Jeremy Vuong

26.5 Recorder/Observer Name: Nikolay Chkhaylo

26.6 Date Tested: 05/03/2024

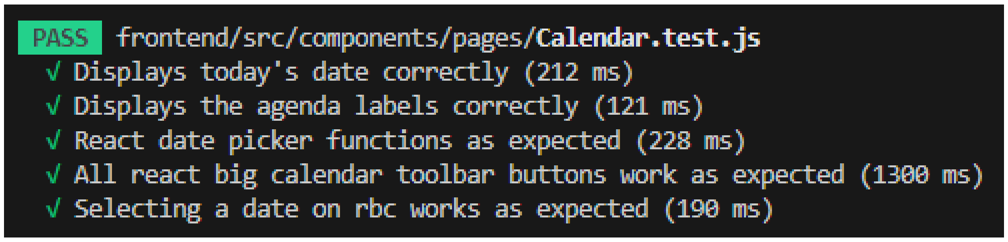
26.7 Results

26.7.1 Did it pass? Yes

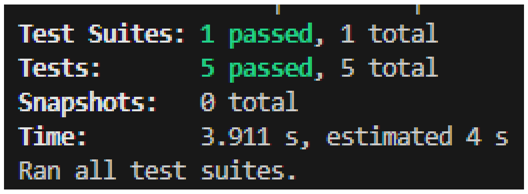


```
it('Selecting a date on rbc works correctly', async () => {
  const { getByText } = render(
    <MemoryRouter>
      <AuthContextProvider>
        <CalendarPage />
      </AuthContextProvider>
    </MemoryRouter>
  );
  // Click on a column header in week view
  fireEvent.click(getByText('01 Wed'));
  // Verify changes, e.g., a task list for the selected day or the date being highlighted
  await waitFor(() => {
    const selectedDayTasks = getByText('May 1, 2024');
    expect(selectedDayTasks).toBeInTheDocument();
  });
  // Navigate to month view
  fireEvent.click(getByText('Month'));
  // Click on a day number
  fireEvent.click(getByText('02')); // Click on the first day of the month shown
  // Verify changes, e.g., a task list for the selected day or the date being highlighted
  await waitFor(() => {
    const selectedDayTasks = getByText('May 2, 2024');
    expect(selectedDayTasks).toBeInTheDocument();
  });
});
```

Results:



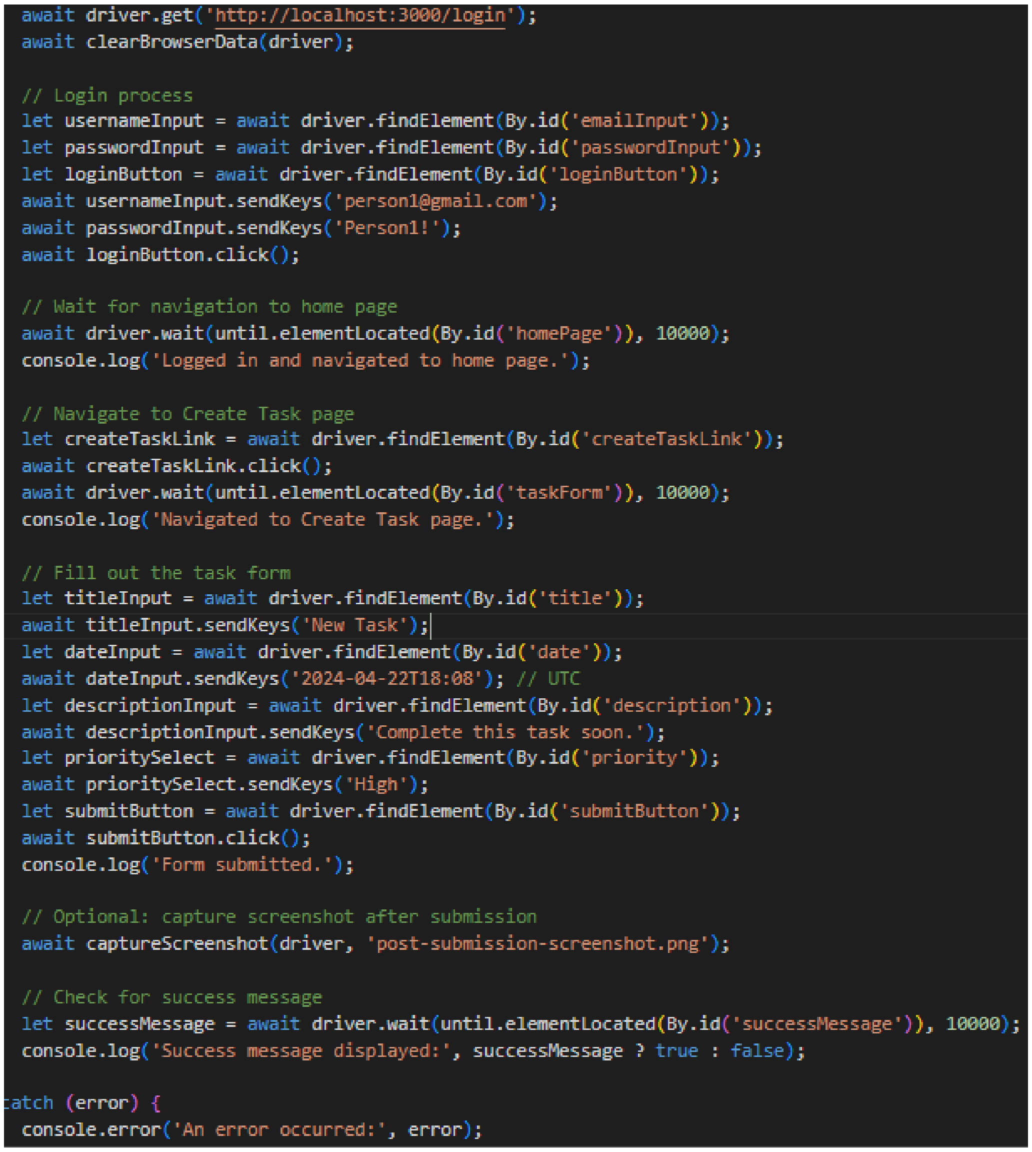
```
PASS frontend/src/components/pages/Calendar.test.js
  √ Displays today's date correctly (212 ms)
  √ Displays the agenda labels correctly (121 ms)
  √ React date picker functions as expected (228 ms)
  √ All react big calendar toolbar buttons work as expected (1300 ms)
  √ Selecting a date on rbc works as expected (190 ms)
```



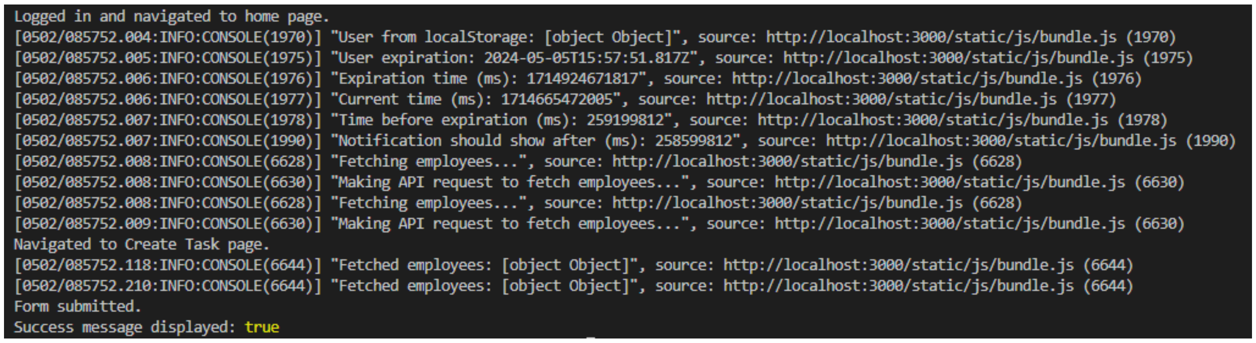
```
Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        3.911 s, estimated 4 s
Ran all test suites.
```

<u>Selenium Testing</u>

Create Task page:



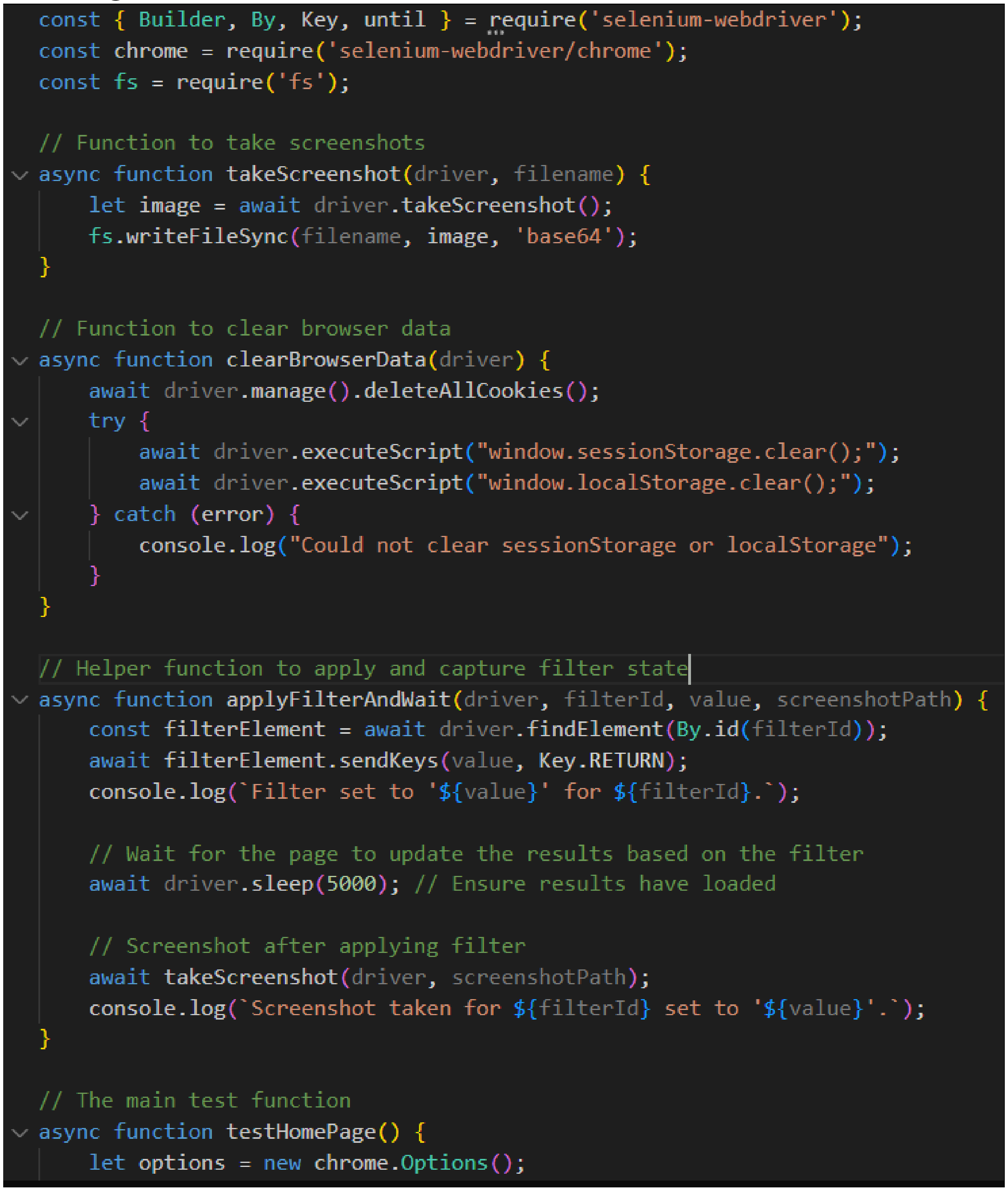
```
await driver.get('http://localhost:3000/login');
await clearBrowserData(driver);

// Login process
let usernameInput = await driver.findElement(By.id('emailInput'));
let passwordInput = await driver.findElement(By.id('passwordInput'));
let loginButton = await driver.findElement(By.id('loginButton'));
await usernameInput.sendKeys('person1@gmail.com');
await passwordInput.sendKeys('Person1!');
await loginButton.click();

// Wait for navigation to home page
await driver.wait(until.elementLocated(By.id('homePage')), 10000);
console.log('Logged in and navigated to home page.');

// Navigate to Create Task page
let createTaskLink = await driver.findElement(By.id('createTaskLink'));
await createTaskLink.click();
await driver.wait(until.elementLocated(By.id('taskForm')), 10000);
console.log('Navigated to Create Task page.');

// Fill out the task form
let titleInput = await driver.findElement(By.id('title'));
await titleInput.sendKeys('New Task');
let dateInput = await driver.findElement(By.id('date'));
await dateInput.sendKeys('2024-04-22T18:08'); // UTC
let descriptionInput = await driver.findElement(By.id('description'));
await descriptionInput.sendKeys('Complete this task soon.');
let prioritySelect = await driver.findElement(By.id('priority'));
await prioritySelect.sendKeys('High');
let submitButton = await driver.findElement(By.id('submitButton'));
await submitButton.click();
console.log('Form submitted.');

// Optional: capture screenshot after submission
await captureScreenshot(driver, 'post-submission-screenshot.png');

// Check for success message
let successMessage = await driver.wait(until.elementLocated(By.id('successMessage')), 10000);
console.log('Success message displayed:', successMessage ? true : false);

atch (error) {
console.error('An error occurred:', error);
```



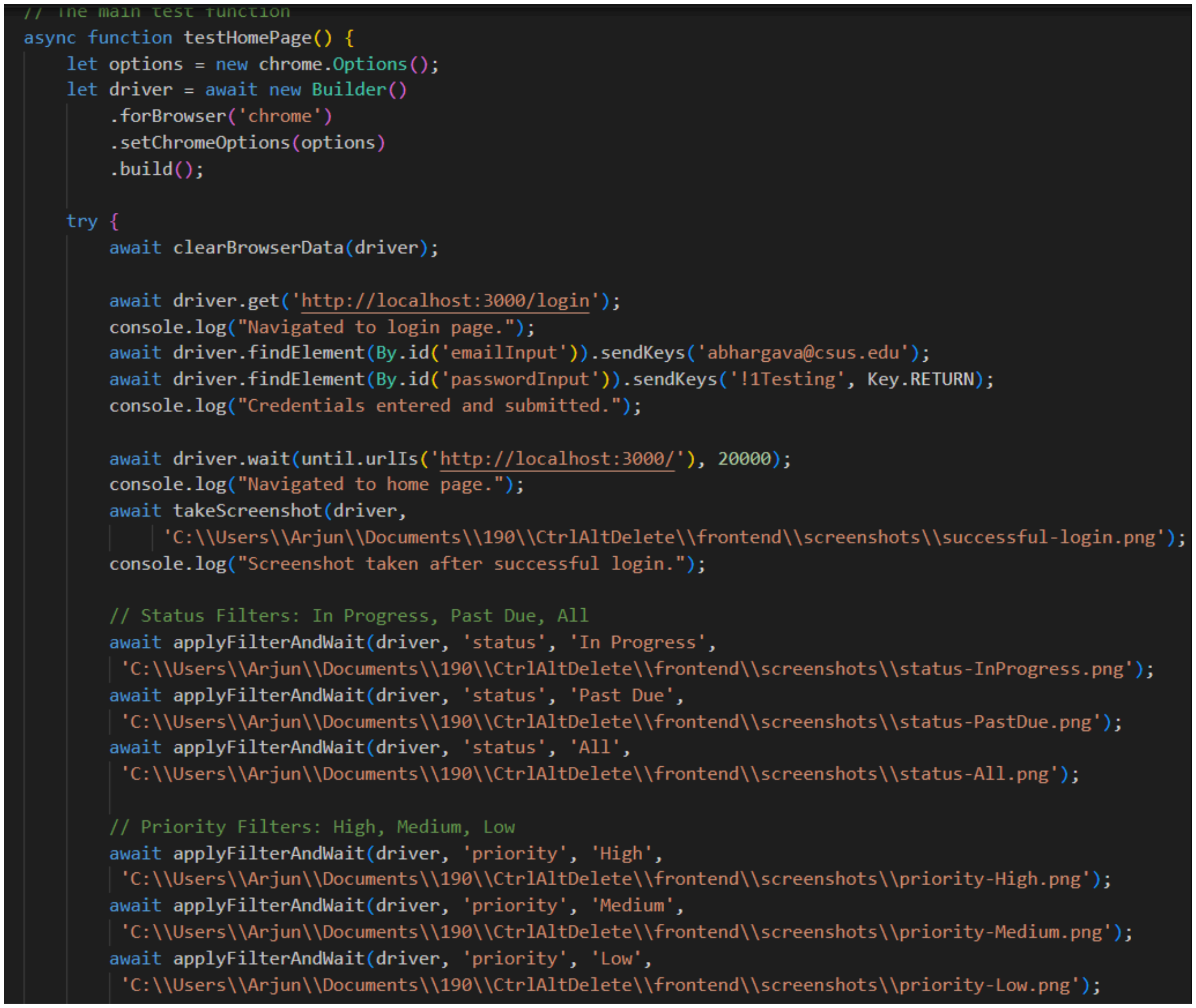
```
Logged in and navigated to home page.
[0502/085752.004:INFO:CONSOLE(1970)] "User from localStorage: [object Object]", source: http://localhost:3000/static/js/bundle.js (1970)
[0502/085752.005:INFO:CONSOLE(1975)] "User expiration: 2024-05-05T15:57:51.817Z", source: http://localhost:3000/static/js/bundle.js (1975)
[0502/085752.006:INFO:CONSOLE(1976)] "Expiration time (ms): 1714924671817", source: http://localhost:3000/static/js/bundle.js (1976)
[0502/085752.006:INFO:CONSOLE(1977)] "Current time (ms): 1714665472005", source: http://localhost:3000/static/js/bundle.js (1977)
[0502/085752.007:INFO:CONSOLE(1978)] "Time before expiration (ms): 259199812", source: http://localhost:3000/static/js/bundle.js (1978)
[0502/085752.007:INFO:CONSOLE(1990)] "Notification should show after (ms): 258599812", source: http://localhost:3000/static/js/bundle.js (1990)
[0502/085752.008:INFO:CONSOLE(6628)] "Fetching employees...", source: http://localhost:3000/static/js/bundle.js (6628)
[0502/085752.008:INFO:CONSOLE(6630)] "Making API request to fetch employees...", source: http://localhost:3000/static/js/bundle.js (6630)
[0502/085752.008:INFO:CONSOLE(6628)] "Fetching employees...", source: http://localhost:3000/static/js/bundle.js (6628)
[0502/085752.009:INFO:CONSOLE(6630)] "Making API request to fetch employees...", source: http://localhost:3000/static/js/bundle.js (6630)
Navigated to Create Task page.
[0502/085752.118:INFO:CONSOLE(6644)] "Fetched employees: [object Object]", source: http://localhost:3000/static/js/bundle.js (6644)
[0502/085752.210:INFO:CONSOLE(6644)] "Fetched employees: [object Object]", source: http://localhost:3000/static/js/bundle.js (6644)
Form submitted.
Success message displayed: true
```

(Output files (screenshots) included in zip file.)

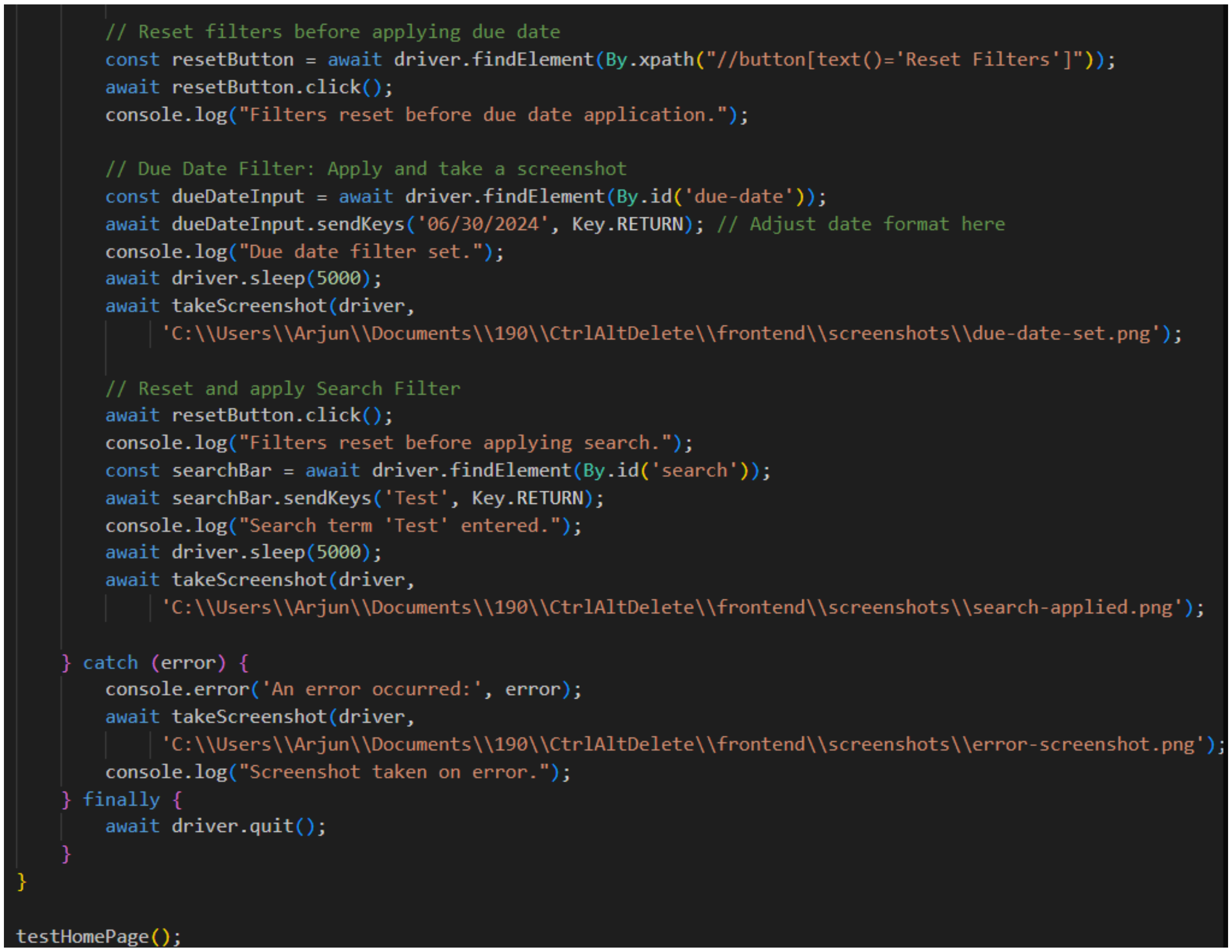
Home Page:



```
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');
const fs = require('fs');

// Function to take screenshots
async function takeScreenshot(driver, filename) {
    let image = await driver.takeScreenshot();
    fs.writeFileSync(filename, image, 'base64');
}

// Function to clear browser data
async function clearBrowserData(driver) {
    await driver.manage().deleteAllCookies();
    try {
        await driver.executeScript("window.sessionStorage.clear();");
        await driver.executeScript("window.localStorage.clear();");
    } catch (error) {
        console.log("Could not clear sessionStorage or localStorage");
    }
}

// Helper function to apply and capture filter state
async function applyFilterAndWait(driver, filterId, value, screenshotPath) {
    const filterElement = await driver.findElement(By.id(filterId));
    await filterElement.sendKeys(value, Key.RETURN);
    console.log(`Filter set to '${value}' for ${filterId}.`);

    // Wait for the page to update the results based on the filter
    await driver.sleep(5000); // Ensure results have loaded

    // Screenshot after applying filter
    await takeScreenshot(driver, screenshotPath);
    console.log(`Screenshot taken for ${filterId} set to '${value}'.`);
}

// The main test function
async function testHomePage() {
    let options = new chrome.Options();
```



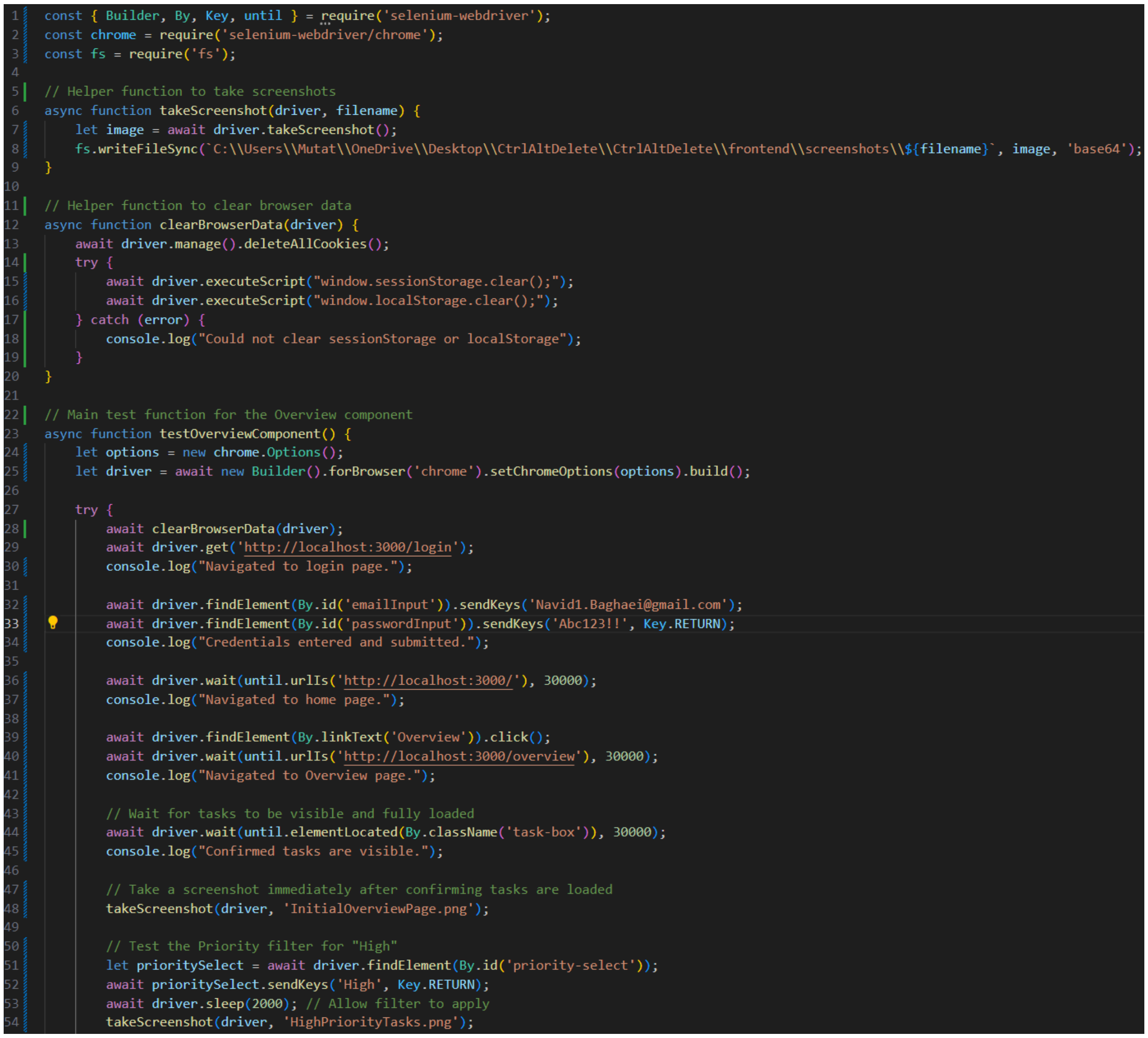
```
// The main test function
async function testHomePage() {
    let options = new chrome.Options();
    let driver = await new Builder()
        .forBrowser('chrome')
        .setChromeOptions(options)
        .build();

    try {
        await clearBrowserData(driver);

        await driver.get('http://localhost:3000/login');
        console.log("Navigated to login page.");
        await driver.findElement(By.id('emailInput')).sendKeys('abhargava@csus.edu');
        await driver.findElement(By.id('passwordInput')).sendKeys('!1Testing', Key.RETURN);
        console.log("Credentials entered and submitted.");

        await driver.wait(until.urlIs('http://localhost:3000/'), 20000);
        console.log("Navigated to home page.");
        await takeScreenshot(driver,
            'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\successful-login.png');
        console.log("Screenshot taken after successful login.");

        // Status Filters: In Progress, Past Due, All
        await applyFilterAndWait(driver, 'status', 'In Progress',
         'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\status-InProgress.png');
        await applyFilterAndWait(driver, 'status', 'Past Due',
         'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\status-PastDue.png');
        await applyFilterAndWait(driver, 'status', 'All',
         'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\status-All.png');

        // Priority Filters: High, Medium, Low
        await applyFilterAndWait(driver, 'priority', 'High',
         'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\priority-High.png');
        await applyFilterAndWait(driver, 'priority', 'Medium',
         'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\priority-Medium.png');
        await applyFilterAndWait(driver, 'priority', 'Low',
         'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\priority-Low.png');
```



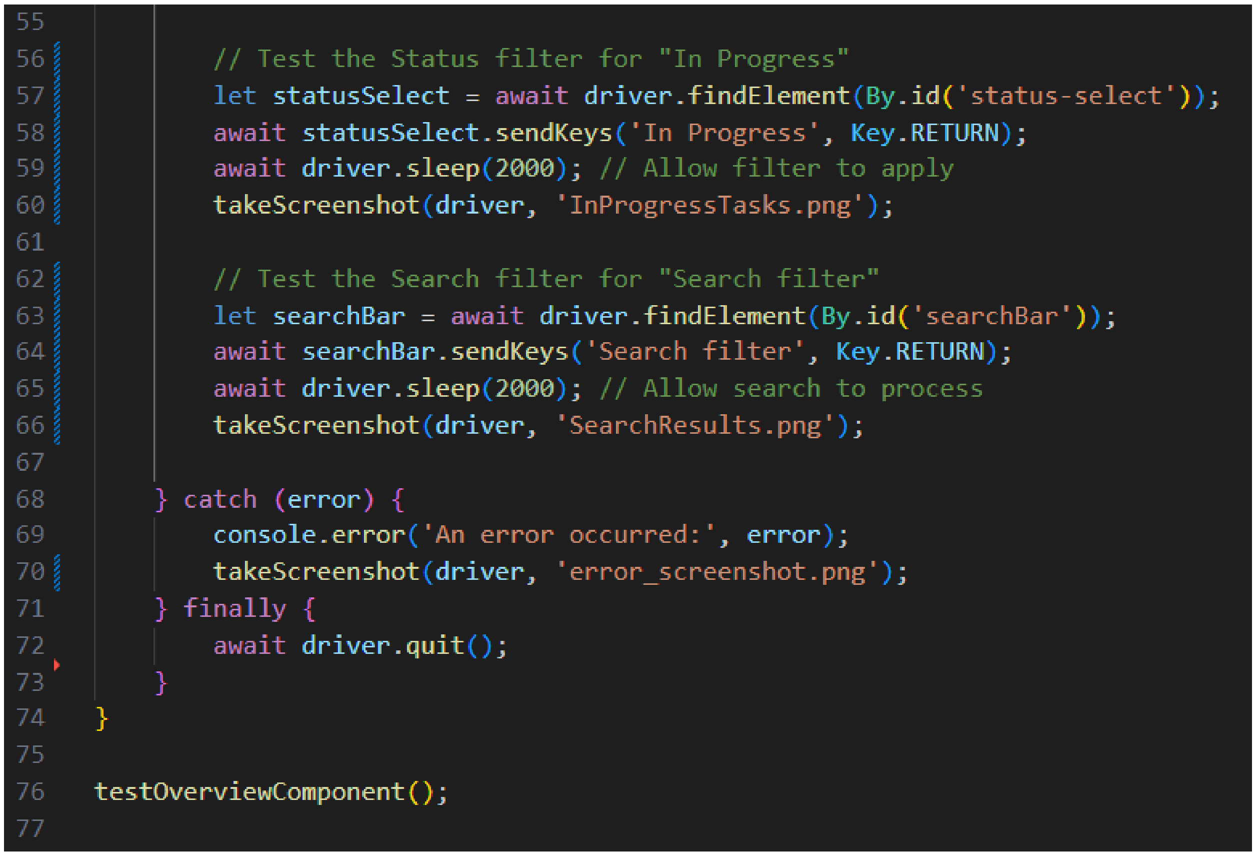
```
        // Reset filters before applying due date
        const resetButton = await driver.findElement(By.xpath("//button[text()='Reset Filters']"));
        await resetButton.click();
        console.log("Filters reset before due date application.");

        // Due Date Filter: Apply and take a screenshot
        const dueDateInput = await driver.findElement(By.id('due-date'));
        await dueDateInput.sendKeys('06/30/2024', Key.RETURN); // Adjust date format here
        console.log("Due date filter set.");
        await driver.sleep(5000);
        await takeScreenshot(driver,
            'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\due-date-set.png');

        // Reset and apply Search Filter
        await resetButton.click();
        console.log("Filters reset before applying search.");
        const searchBar = await driver.findElement(By.id('search'));
        await searchBar.sendKeys('Test', Key.RETURN);
        console.log("Search term 'Test' entered.");
        await driver.sleep(5000);
        await takeScreenshot(driver,
            'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\search-applied.png');

    } catch (error) {
        console.error('An error occurred:', error);
        await takeScreenshot(driver,
            'C:\\Users\\Arjun\\Documents\\190\\CtrlAltDelete\\frontend\\screenshots\\error-screenshot.png');
        console.log("Screenshot taken on error.");
    } finally {
        await driver.quit();
    }
}

testHomePage();
```

(Output files (screenshots) included in zip file.)

Overview Page:



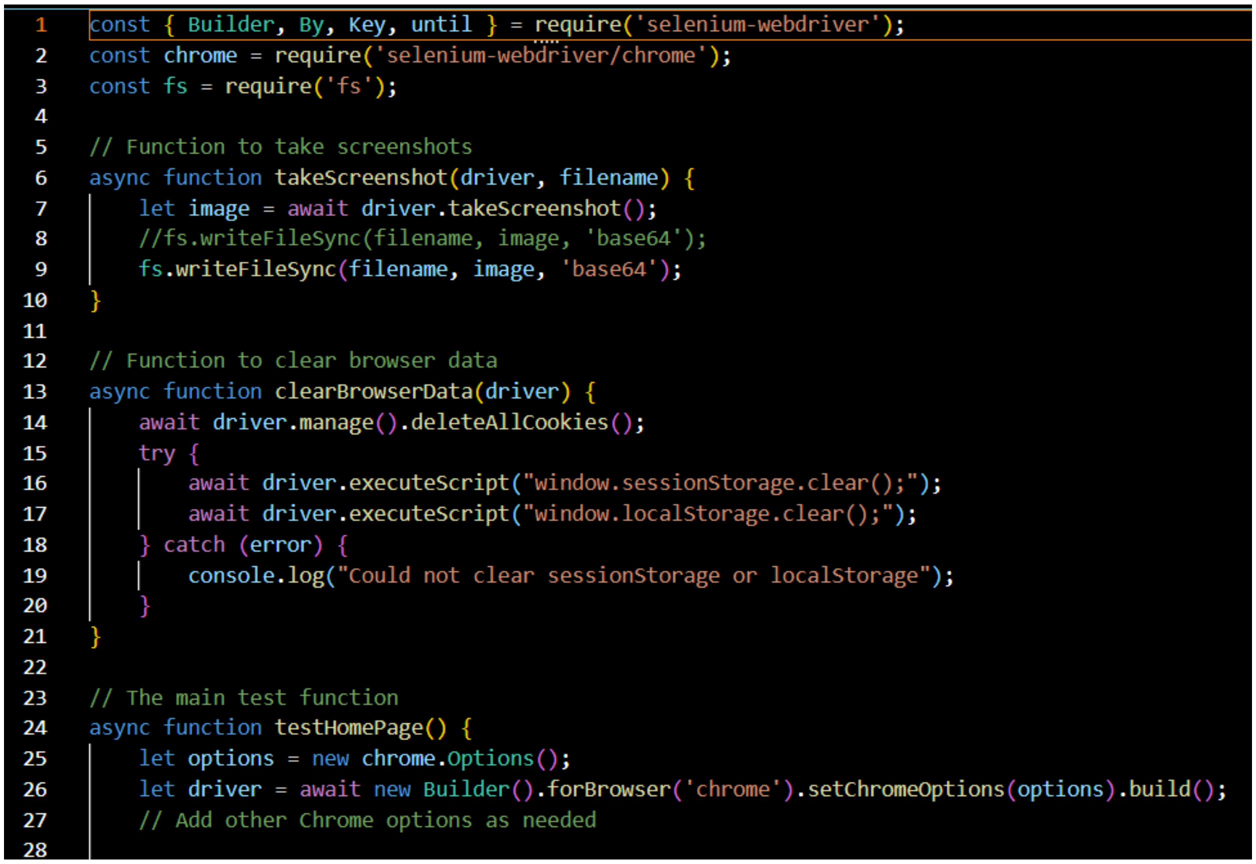
```javascript
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');
const fs = require('fs');

// Helper function to take screenshots
async function takeScreenshot(driver, filename) {
    let image = await driver.takeScreenshot();
    fs.writeFileSync(`C:\\Users\\Mutat\\OneDrive\\Desktop\\CtrlAltDelete\\CtrlAltDelete\\frontend\\screenshots\\${filename}`, image, 'base64');
}

// Helper function to clear browser data
async function clearBrowserData(driver) {
    await driver.manage().deleteAllCookies();
    try {
        await driver.executeScript("window.sessionStorage.clear();");
        await driver.executeScript("window.localStorage.clear();");
    } catch (error) {
        console.log("Could not clear sessionStorage or localStorage");
    }
}

// Main test function for the Overview component
async function testOverviewComponent() {
    let options = new chrome.Options();
    let driver = await new Builder().forBrowser('chrome').setChromeOptions(options).build();

    try {
        await clearBrowserData(driver);
        await driver.get('http://localhost:3000/login');
        console.log("Navigated to login page.");

        await driver.findElement(By.id('emailInput')).sendKeys('Navid1.Baghaei@gmail.com');
        await driver.findElement(By.id('passwordInput')).sendKeys('Abc123!!', Key.RETURN);
        console.log("Credentials entered and submitted.");

        await driver.wait(until.urlIs('http://localhost:3000/'), 30000);
        console.log("Navigated to home page.");

        await driver.findElement(By.linkText('Overview')).click();
        await driver.wait(until.urlIs('http://localhost:3000/overview'), 30000);
        console.log("Navigated to Overview page.");

        // Wait for tasks to be visible and fully loaded
        await driver.wait(until.elementLocated(By.className('task-box')), 30000);
        console.log("Confirmed tasks are visible.");

        // Take a screenshot immediately after confirming tasks are loaded
        takeScreenshot(driver, 'InitialOverviewPage.png');

        // Test the Priority filter for "High"
        let prioritySelect = await driver.findElement(By.id('priority-select'));
        await prioritySelect.sendKeys('High', Key.RETURN);
        await driver.sleep(2000); // Allow filter to apply
        takeScreenshot(driver, 'HighPriorityTasks.png');
```



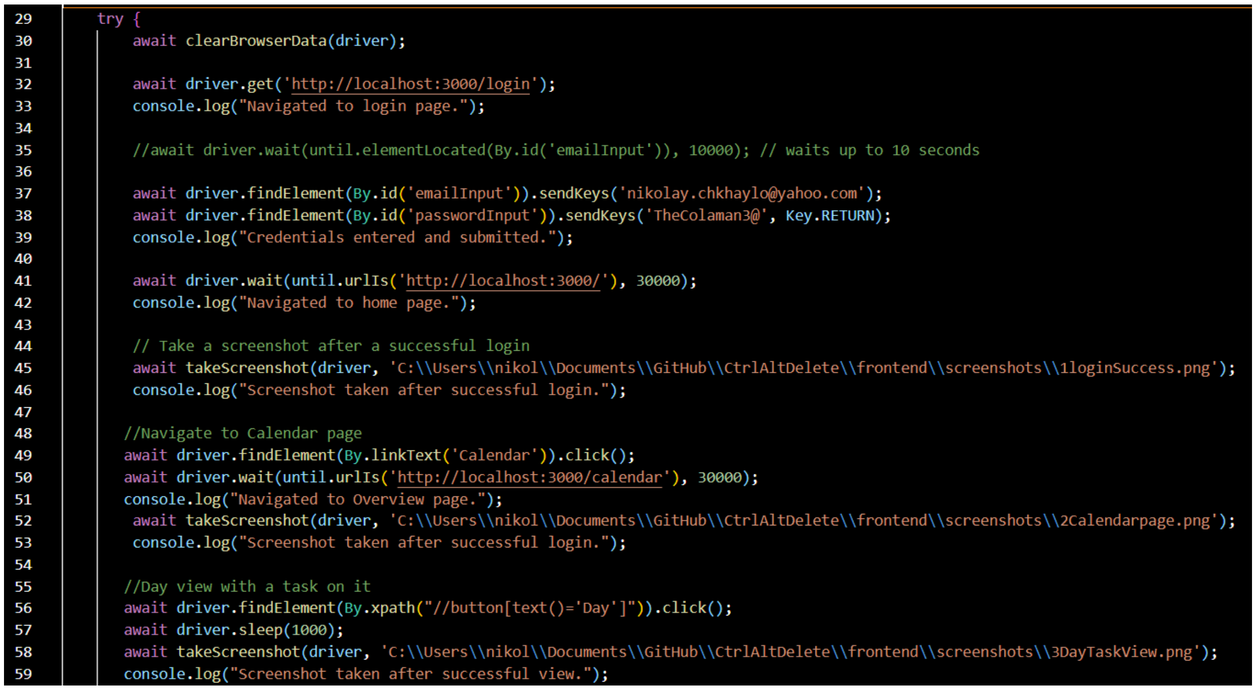
```javascript

        // Test the Status filter for "In Progress"
        let statusSelect = await driver.findElement(By.id('status-select'));
        await statusSelect.sendKeys('In Progress', Key.RETURN);
        await driver.sleep(2000); // Allow filter to apply
        takeScreenshot(driver, 'InProgressTasks.png');

        // Test the Search filter for "Search filter"
        let searchBar = await driver.findElement(By.id('searchBar'));
        await searchBar.sendKeys('Search filter', Key.RETURN);
        await driver.sleep(2000); // Allow search to process
        takeScreenshot(driver, 'SearchResults.png');

    } catch (error) {
        console.error('An error occurred:', error);
        takeScreenshot(driver, 'error_screenshot.png');
    } finally {
        await driver.quit();
    }
}

testOverviewComponent();

```

(Output files (screenshots) included in zip file.)

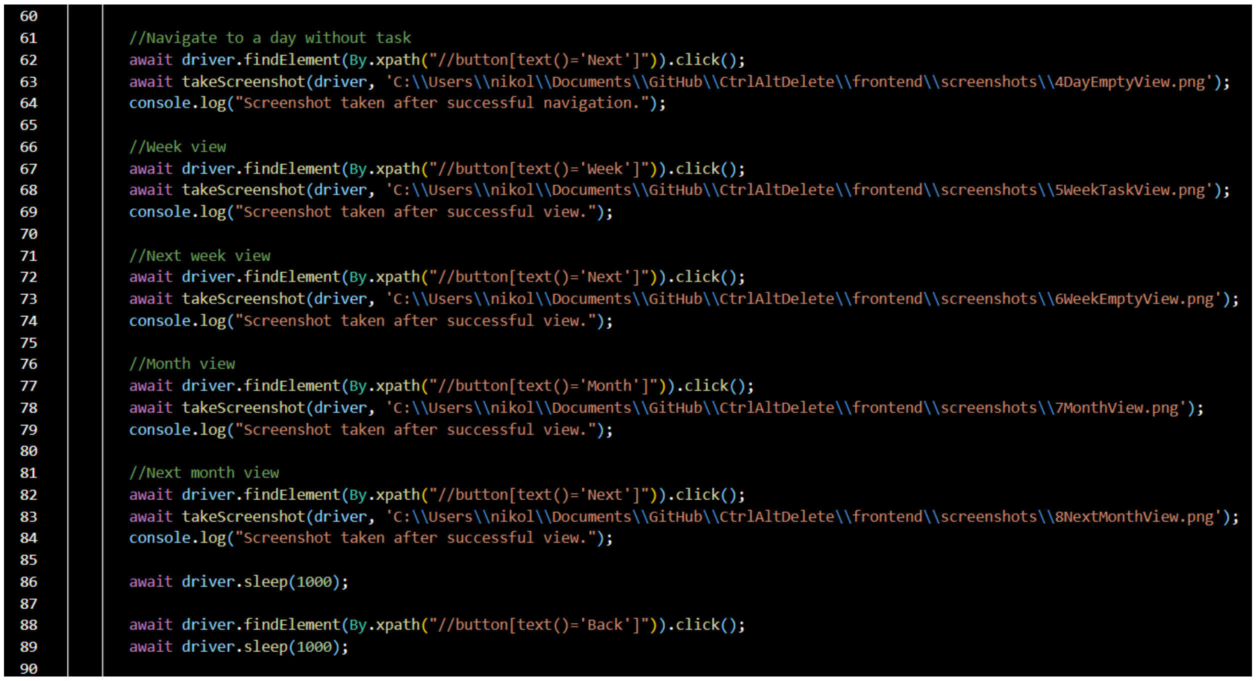
Calendar Page:



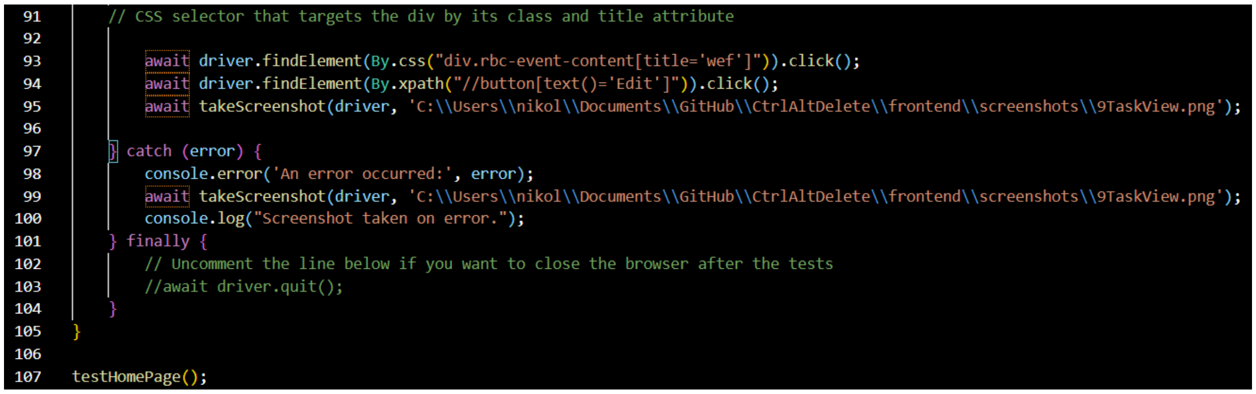
```
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');
const fs = require('fs');

// Function to take screenshots
async function takeScreenshot(driver, filename) {
    let image = await driver.takeScreenshot();
    //fs.writeFileSync(filename, image, 'base64');
    fs.writeFileSync(filename, image, 'base64');
}

// Function to clear browser data
async function clearBrowserData(driver) {
    await driver.manage().deleteAllCookies();
    try {
        await driver.executeScript("window.sessionStorage.clear();");
        await driver.executeScript("window.localStorage.clear();");
    } catch (error) {
        console.log("Could not clear sessionStorage or localStorage");
    }
}

// The main test function
async function testHomePage() {
    let options = new chrome.Options();
    let driver = await new Builder().forBrowser('chrome').setChromeOptions(options).build();
    // Add other Chrome options as needed

```



```
    try {
        await clearBrowserData(driver);

        await driver.get('http://localhost:3000/login');
        console.log("Navigated to login page.");

        //await driver.wait(until.elementLocated(By.id('emailInput')), 10000); // waits up to 10 seconds

        await driver.findElement(By.id('emailInput')).sendKeys('nikolay.chkhaylo@yahoo.com');
        await driver.findElement(By.id('passwordInput')).sendKeys('TheColaman3@', Key.RETURN);
        console.log("Credentials entered and submitted.");

        await driver.wait(until.urlIs('http://localhost:3000/'), 30000);
        console.log("Navigated to home page.");

        // Take a screenshot after a successful login
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\1loginSuccess.png');
        console.log("Screenshot taken after successful login.");

       //Navigate to Calendar page
       await driver.findElement(By.linkText('Calendar')).click();
       await driver.wait(until.urlIs('http://localhost:3000/calendar'), 30000);
       console.log("Navigated to Overview page.");
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\2Calendarpage.png');
        console.log("Screenshot taken after successful login.");

       //Day view with a task on it
       await driver.findElement(By.xpath("//button[text()='Day']")).click();
       await driver.sleep(1000);
       await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\3DayTaskView.png');
       console.log("Screenshot taken after successful view.");
```



```
        //Navigate to a day without task
        await driver.findElement(By.xpath("//button[text()='Next']")).click();
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\4DayEmptyView.png');
        console.log("Screenshot taken after successful navigation.");

        //Week view
        await driver.findElement(By.xpath("//button[text()='Week']")).click();
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\5WeekTaskView.png');
        console.log("Screenshot taken after successful view.");

        //Next week view
        await driver.findElement(By.xpath("//button[text()='Next']")).click();
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\6WeekEmptyView.png');
        console.log("Screenshot taken after successful view.");

        //Month view
        await driver.findElement(By.xpath("//button[text()='Month']")).click();
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\7MonthView.png');
        console.log("Screenshot taken after successful view.");

        //Next month view
        await driver.findElement(By.xpath("//button[text()='Next']")).click();
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\8NextMonthView.png');
        console.log("Screenshot taken after successful view.");

        await driver.sleep(1000);

        await driver.findElement(By.xpath("//button[text()='Back']")).click();
        await driver.sleep(1000);

```



```
    // CSS selector that targets the div by its class and title attribute

        await driver.findElement(By.css("div.rbc-event-content[title='wef']")).click();
        await driver.findElement(By.xpath("//button[text()='Edit']")).click();
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\9TaskView.png');

    } catch (error) {
        console.error('An error occurred:', error);
        await takeScreenshot(driver, 'C:\\Users\\nikol\\Documents\\GitHub\\CtrlAltDelete\\frontend\\screenshots\\9TaskView.png');
        console.log("Screenshot taken on error.");
    } finally {
        // Uncomment the line below if you want to close the browser after the tests
        //await driver.quit();
    }
}

testHomePage();
```

(Output files (screenshots) included in zip file.)

Signatures

Digital or Physical signature required; integrity of the document must be kept intact. Do not copy and paste signatures.

Team Members: **System Test Verification Completed: Date: ____________**

Name	Signature