Test Environment:

- Operating System: Windows 11

- Browser: Google Chrome Version 90.0.4430.212

Tested Component: Search Bar

Summary:

The Search Bar component is responsible for rendering a search input field and emitting a search event to notify the parent component about the search query entered by the user.

Tested Functionality:

1. Rendering of search input field

2. Input binding and event emission

Test Cases:

1. Rendering of Search Input Field:

- Verify that the search input field is rendered.

- Confirm that the placeholder text is displayed correctly.

2. Input Binding and Event Emission:

- Enter test values in the search input field.

- Validate that the entered value is bound correctly to the searchText data property.

- Simulate input by triggering the input event.

- Verify that the handle Input method is called.

3. Reset event:

-Enter a filter text in the input field.

-Click on the "Reset" button.

Test Results:

1. Rendering of Search Input Field:

- The search input field is rendered correctly.

- The placeholder text is displayed as expected.

2. Input Binding and Event Emission:

- Test values entered in the search input field are successfully bound to the searchText data property.

- The handled Input method is called upon input.

3. Reset event:

-The filter text input field should be cleared.

-The table should display all rows without any filtering applied.

Conclusion:

The Search Bar component has undergone extensive testing, and all capabilities are functioning as expected. The typed search text is bound correctly and released as a search event, and the search input box is presented correctly. Based on the specified test cases, it can be said that the component is working well.

Tested Component: Chart Container

Summary:

The Chart Container component is responsible for rendering a chart using HTML5 canvas. It fetches chart data from a json file and allows users to hover over bars to display a tool tip with the corresponding value.

Tested Functionality:

1. Rendering of chart canvas

2. Fetching chart data from a json

3. Drawing the chart

4. Handling mouse events (mouse move, mouse leave)

Test Cases:

1. Rendering of Chart Canvas:

- Verify that the chart canvas is rendered.

- Confirm that the canvas has the correct dimensions.

- Check that the canvas has the appropriate styling.

2. Fetching Chart Data from a json file:

- Simulate the chart data fetch from the json.

- Ensure that the fetched Chart Data property is updated correctly.

3. Drawing the Chart:

- Validate that the chart is drawn correctly on the canvas.

- Check that each bar has the correct height based on the data.

- Verify that the canvas dimensions are adjusted according to the chart data.

4. Handling Mouse Events:

- Simulate a mouse move event on the canvas.

- Verify that the hovered Bar Index is updated correctly.

- Confirm that the tool tip Value are set accurately.

- Check that the show Tool tip flag is set to true.

Test Results:

1. Rendering of Chart Canvas:

- The chart canvas is rendered correctly.

- The canvas has the expected dimensions.

- The canvas has the appropriate styling.

2. Fetching Chart Data from a json:

- The fetched Chart Data property is updated correctly after the data fetch.

3. Drawing the Chart:

- The chart is drawn correctly on the canvas.

- Each bar has the correct height based on the data.

- The canvas dimensions are adjusted according to the chart data.

4. Handling Mouse Events:

- The hovered Bar Index is updated correctly upon a mouse move event.

- The tool tip Value are set accurately.

- The show Tool tip flag is set to true.

Conclusion:

All of the capabilities that have been tested for the Chart Container component are functioning as intended. The chart is accurately created, the mouse events are properly handled, the tool tip is presented, the data fetching updates the fetched Chart Data property, and the canvas is rendered correctly. Based on the specified test cases, it can be said that the component is working well.

Tested Component: Form Component

Summary:

The Form Component is responsible for rendering a form with input fields and a submit button. It allows users to enter values in the form fields and submits the form data when the submit button is clicked.

Tested Functionality:

1. Rendering of form fields

2. Input binding and value updating

3. Form submission and data handling

Test Cases:

1. Rendering of Form Fields:

- Verify that the form fields are rendered correctly.

- Confirm that the labels and input fields are displayed as expected.

2. Input Binding and Value Updating:

- Enter test values in the input fields.

- Validate that the entered values are bound correctly to the form Fields data property.

- Ensure that the input field values are updated as expected.

3. Form Submission and Data Handling:

- Simulate the form submission by clicking the submit button.

- Verify that the handle Submit method is called.

- Validate that the form data is correctly accumulated from the form Fields data property.

- Confirm that the form data is logged to the console.

Test Results:

1. Rendering of Form Fields:

- The form fields are rendered correctly.

- The labels and input fields are displayed as expected.

2. Input Binding and Value Updating:

- Test values entered in the input fields are successfully bound to the form Fields data property.

- The input field values are updated as expected.

3. Form Submission and Data Handling:

- The handle Submit method is called upon form submission.

- The form data is correctly accumulated from the form Fields data property.

- The form data is logged to the console as expected.

Conclusion:

Based on the provided test cases, the Form Component can be considered to be functioning correctly because all of the functionalities have been thoroughly tested and are functioning as expected. The form fields are rendered correctly, the entered values are bound appropriately, the form data is accumulated and logged to the console upon form submission.

Tested Component: Date Picker Component

Summary:

The Date Picker Component is responsible for displaying a date picker and handling the selected event date.

Tested Functionality:

1. Rendering of the date picker

2. Selection of the event date

Test Cases:

1. Rendering of the Date Picker:

- Verify that the event scheduler component is rendered correctly.

- Confirm that the date picker input field is displayed.

- Ensure that the event date is initially empty.

2. Selection of the Event Date:

- Simulate selecting a date from the date picker.

- Verify that the selected date is correctly bound to the component's data.

Test Results:

1. Rendering of the Date Picker:

- The event scheduler component is rendered correctly.

- The date picker input field is displayed as expected.

- The event date is initially empty.

2. Selection of the Event Date:

- Simulating the selection of a date from the date picker updates the component's data with the selected date.

- The selected date is correctly bound to the component's data.

Conclusion:

All of the capabilities that have been tested for the Event Schedule Component are functioning as intended. Users are able to choose an event date due to the component's successful rendering of the date picker. The component's data is appropriately tied to the chosen date.

Tested Component: Calendar Component

Summary:

The Calendar Component is responsible for displaying a calendar with selectable dates.

Tested Functionality:

1. Rendering of the calendar

2. Navigation to previous and next months

3. Selection of dates

Test Cases:

1. Rendering of the Calendar:

- Verify that the calendar component is rendered correctly.

- Confirm that the header displays the current month and year.

- Ensure that the table contains the days of the week and the calendar dates.

2. Navigation to Previous and Next Months:

- Simulate clicking on the previous month button.

- Verify that the displayed month and year are updated accordingly.

- Simulate clicking on the next month button.

- Confirm that the displayed month and year are updated again.

3. Selection of Dates:

- Simulate clicking on a date within the calendar.

- Verify that the selected date is correctly updated in the component's data.

Test Results:

1. Rendering of the Calendar:

- The calendar component is rendered correctly.

- The header displays the current month and year as expected.

- The table contains the days of the week and the calendar dates

2. Navigation to Previous and Next Months:

- Simulating the click on the previous month button updates the displayed month and year correctly.

- Simulating the click on the next month button updates the displayed month and year again.

3. Selection of Dates:

- Simulating the click on a date within the calendar updates the selected date in the component's data.

Conclusion:

All of the capabilities that have been tested for the calendar component are functioning as expected. The calendar is correctly rendered by the component, which also provides date selection and month navigation. The component's data is successfully updated with the selected date.