Test Documentation for Sticky Notes and To-Do List

Sticky Notes Tests

Test: Verify Created Notes

This test verifies that all notes in the dummyNotesList are rendered correctly on the screen. The test renders the StickyNotes component and iterates through the dummyNotesList, asserting each note title is present in the document.

Implementation:

```
test("verify created notes", () => {
  render(<StickyNotes />);
  dummyNotesList.forEach(note => {
    const titleElement = screen.getByText(note.title);
    expect(titleElement).toBeInTheDocument();
  });
});
```

Test: Update a Note

This test verifies the functionality of updating an existing note. It renders the StickyNotes component and finds the note elements by their original titles. It then uses the `fireEvent.input` to change the innerHTML of both title and content elements to new values. Finally, it asserts the new title and content are present in the document.

Implementation:

```
test("update a note", async () => {
  render(<StickyNotes />);
  const titleInput = screen.getByText("test note 1 title");
  const contentInput = screen.getByText("test note 1 content");

  fireEvent.input(titleInput, { target: { innerHTML: "new title" } });
  fireEvent.input(contentInput, { target: { innerHTML: "new content" } });

  const newTitle = screen.getByText("new title");
  const newContent = screen.getByText("new content");

  expect(newTitle).toBeInTheDocument();
  expect(newContent).toBeInTheDocument();
});
```

Test: Delete a Note

This test verifies the delete functionality of a note. It renders the StickyNotes component, finds the title of a specific note, and simulates a click on the delete button associated with that note. It then asserts the title element is no longer present in the document.

Implementation:

```
test("delete a note", () => {
  render(<StickyNotes />);
  const titleInput = screen.getByText("test note 1 title");
  const deleteButton = screen.getByTestId("xbut1");

  fireEvent.click(deleteButton);

  expect(titleInput).not.toBeInTheDocument();
});
```

Test: Note Deletion Synchronizes with Favorite List

This test checks if deleting a note also deletes the corresponding entry in the favorite list. It renders the StickyNotes component, finds a note title, and simulates clicking the like button and then the delete button. Finally, it asserts that the note title is not present in the document after deletion.

Implementation:

```
test("note deletion synchronizes with favorite list", () => {
  render(<StickyNotes />);
  const titleInput = screen.getByText("test note 1 title");
  const likeButton = screen.getByTestId("likebut1");
  const deleteButton = screen.getByTestId("xbut1");

  fireEvent.click(likeButton);
  fireEvent.click(deleteButton);

  expect(titleInput).not.toBeInTheDocument();
});
```

To-Do List Tests

Test: Verify Displaying Items

This test verifies that all items in the dummyGroceryList are displayed in the To-Do List. The test renders the ToDoList component and asserts that each item name is present in the document.

Implementation:

```
test("verify displaying items", () => {
  render(<ToDoList />);
  dummyGroceryList.forEach(item => {
    const itemName = screen.getByText(item.name);
    expect(itemName).toBeInTheDocument();
  });
});
```

Test: Verify Number of Checked Items

This test checks if the number of checked items is displayed correctly. It renders the ToDoList component, finds all checkboxes, and randomly checks a few of them. Then it retrieves the text that shows the number of bought items and asserts that the number is equal to the number of checked checkboxes.

Implementation:

```
test("verify num of checked items", () => {
  render(<ToDoList />);
  const checkboxes = screen.getAllByRole('checkbox') as HTMLInputElement[];

  fireEvent.click(checkboxes[2 % checkboxes.length]); // randomly check some
boxes
  fireEvent.click(checkboxes[4 % checkboxes.length]);
  fireEvent.click(checkboxes[7 % checkboxes.length]);

  const itemsBoughtText = screen.getByText(/Items bought:/i);
  const content = itemsBoughtText?.textContent || "Items bought: 0";
  const numBought = parseInt(content.match(/\d+/)?.[0] || '0', 10);

  const checkedCheckboxes = checkboxes.filter(checkbox => checkbox.checked);

  expect(numBought).toBe(checkedCheckboxes.length);
});
```