

Participant P22

00:02:22 **Researcher**
oK, you can start.

00:02:23 **Participant**
OK

00:03:22 > **Viewing file** (src/todo/components/footer.jsx)

00:03:26 > **Viewing file** (src/todo/components/header.jsx)

00:03:28 > **Viewing file** (src/todo/components/input.jsx)

00:03:30 > **Viewing file** (src/todo/components/item.jsx)

00:03:38 > **Viewing file** (src/todo/components/main.jsx)

00:04:51 **Researcher**
What are you thinking now?

00:04:53 **Participant**
I'm analyzing the first task, color change
So I'll just ask the copilot to change it for me.

00:05:01 **Researcher**
OK.

00:05:06 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
selection (src/todo/components/main.jsx)
> **Message**
I need to change the color of the "no tasks" message box to blue (#007bff)
> **Response**
To change the color of the "No tasks here yet" message box to blue (#007bff), you can add or modify the CSS style associated with the class [todo-list-empty-container \(src/todo/components/main.jsx\)](#). Here's how you can do it:

Step 1: Update the CSS

If you are using a separate CSS file, add or edit the rule for the `.todo-list-empty-container` class:

```
File: src/todo/styles/main.css
.todo-list-empty-container {
  background-color: #007bff; /* Blue color */
  color: white; /* Adjust text color for contrast */
  padding: 10px; /* Add internal spacing */
  border-radius: 5px; /* Rounded borders (optional) */
  text-align: center; /* Center the text */
}
```

Step 2: Check the CSS Import

Make sure the CSS file is being imported into your project. For example, in the file [src/todo/components/main.jsx](#) or in a main entry file:

```
File: src/todo/components/main.jsx
import "../styles/main.css";
```

Step 3: Test the Change

Restart the development server (if necessary) and check if the change has been applied correctly.

If you are using a CSS-in-JS approach or a library like styled-components, let me know so I can adjust the solution.

00:05:45
- 00:05:49

> **Response rendering**

00:06:54

Participant

is it right?

00:07:25

Chat (window: 0)

> **Agent**

github.copilot.workspace

> **Context**

workspace, selection (src/todo/components/main.jsx)

> **Message**

@workspace help me with this task according to the project structure

> **Response**

The component [Main \(src/todo/components/main.jsx\)](#) has an issue with the import and use of the component [paginationControls \(src/todo/components/main.jsx\)](#). It is being imported as if it were a component, but it should actually be [PaginationControls \(src/todo/components/paginationControls.jsx\)](#), which is the correct name of the component defined in [src/todo/components/paginationControls.jsx](#).

Here is the necessary correction in the file [src/todo/components/main.jsx](#):

```
File: src/todo/components/main.jsx
import { PaginationControls } from "../paginationControls"; // Corrected
the component name
```

```
export function Main({ todos, dispatch }) {
  // ...existing code...

  return (
    <main className="main" data-testid="main">
      {visibleTodos.length > 0 ? (
        <div className="toggle-all-container">
          <input className="toggle-all" type="checkbox"
            id="toggle-all" data-testid="toggle-all" checked=
{visibleTodos.every((todo) => todo.completed)} onChange={toggleAll} />
          <label className="toggle-all-label"
            htmlFor="toggle-all">
            Toggle All Input
          </label>
        </div>
        <ul className={classnames("todo-list")} data-
testid="todo-list">
          {visibleTodos.map((todo, index) => (
            <Item todo={todo} key={todo.id} dispatch=
{dispatch} index={index} />
          ))}
        </ul>
      ) : (
        <div className="todo-list-empty-container">
          <span className="todo-list-empty-message">No tasks
here yet</span>
        </div>
      )}

      <PaginationControls
        currentPage={currentPage}>
```

```

        itemsPerPage={itemsPerPage}
        setCurrentPage={setCurrentPage}
        totalItems={visibleTodos.length}
      />
    </main>
  );
}

```

Changes made:

1. Corrected the import of [PaginationControls](#) ([src/todo/components/paginationControls.jsx](#)) at the top of the file.
 2. Replaced `<paginationControls />` with `<PaginationControls />` in the JSX.
- With this correction, the component [PaginationControls](#) ([src/todo/components/paginationControls.jsx](#)) will be rendered correctly in the component [Main](#) ([src/todo/components/main.jsx](#)).

> Agent

github.copilot.workspace

> Context

workspace, selection (src/todo/components/main.jsx)

> Message

@workspace I need to change the color of the message box from "nenhum afazer aqui ainda" to blue (#007bff)

> Response

To change the color of the message box "Nenhum afazer aqui ainda" to blue (#007bff), you can update the corresponding style in the CSS file [src/todo/app.css](#). Here is the necessary modification:

```
File: src/todo/app.css
.todo-list-empty-message {
  padding: 10px;
  background: #007bff; /* Changed to blue */
  color: white;
  border-radius: 5px;
  font-weight: bolder;
}
```

This change will apply the new blue background color to the message displayed when there are no tasks.

> Response rendering

> Viewing file (src/todo/app.css)

> Implementing Copilot code by copying part of snippet (src/todo/app.css)

> Viewing file (package.json)

Researcher

yes. You can go to the next task

> Viewing file (src/todo/components/header.jsx) through file search

> Editing file (src/todo/components/header.jsx)

Participant

I'll use the search to find the place Holder

Great. I think it worked

sorting task

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (src/todo/components/header.jsx)

> Message

@workspace I need my tasks to be sorted in alphabetical order, make this change for me

> Response

To sort the tasks in alphabetical order, you can modify the calculation of [visibleTodos](#) ([src/todo/components/main.jsx](#)) in the [Main component](#) ([src/todo/components/main.jsx](#)). Just add a sorting step after the filter.

Here is the change in the file [src/todo/components/main.jsx](#):

```
File: src/todo/components/main.jsx
const visibleTodos = useMemo(
  () =>
    todos
      .filter((todo) => {
        if (route === "/active") return !todo.completed;
        if (route === "/completed") return todo.completed;
        return todo;
      })
```

00:08:41
- 00:08:43

00:08:51

00:09:05

00:09:23

00:09:43

00:11:09

00:11:21
- 00:12:52

00:11:22

00:14:52

00:14:53

00:15:31

```

    })
    .sort((a, b) => a.title.localeCompare(b.title)), // Sorts
the tasks in alphabetical order
    [todos, route]
  );

```

Change made:

1. Added the function `.sort((a, b) => a.title.localeCompare(b.title))` to sort the tasks alphabetically based on the title.
- Now, the tasks will be displayed in alphabetical order in the list.

00:15:37
- 00:15:41

> Response rendering

00:15:46

> Viewing file (src/todo/components/main.jsx)

00:16:08
- 00:16:08

> Implementing Copilot code by copying full snippet (src/todo/components/main.jsx)

00:16:35

Participant

I'll confirm if it worked

00:17:14

Researcher

Ok, very good. Now I'll send you the link to the tasks,

00:17:23

Participant

great.

00:17:29

Researcher

You don't need to do it in order. Just move to the next task after completing all the requirements, ok?

00:17:50

Participant

OK

00:19:21

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (src/todo/components/main.jsx)

> Message

@workspace I need my tasks to remain saved if I reload the page, do this for me

> Response

To ensure that the tasks remain saved even after reloading the page, you can use [localStorage \(node_modules/@types/node/globals.d.ts\)](#) to persist the data. Here's how you can implement this in the [src/todo/app.jsx](#) file:

Changes in the [src/todo/app.jsx](#) file:

Add the initial loading of tasks from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) and save the tasks whenever they are updated.

```
File: src/todo/app.jsx
import { useReducer, useEffect } from "react";
import { Header } from "../components/header";
import { Main } from "../components/main";
import { Footer } from "../components/footer";

import { todoReducer } from "../reducer";

import "../app.css";

const LOCAL_STORAGE_KEY = "todos";

export function App() {
  const [todos, dispatch] = useReducer(todoReducer, [], () => {
    const savedTodos = localStorage.getItem(LOCAL_STORAGE_KEY);
    return savedTodos ? JSON.parse(savedTodos) : [];
  });

  useEffect(() => {
    localStorage.setItem(LOCAL_STORAGE_KEY, JSON.stringify(todos));
  }, [todos]);

  return (
    <>
      <Header dispatch={dispatch} />
      <Main todos={todos} dispatch={dispatch} />
      <Footer todos={todos} dispatch={dispatch} />
    </>
  );
}
```

What was done:

1. Initial loading of tasks:

- Added a third argument in the [useReducer](#) ([node_modules/@types/react/index.d.ts](#)) function to initialize the state with the data saved in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)).

2. Persistence of tasks:

- Added a [useEffect](#) ([node_modules/@types/react/index.d.ts](#)) to save the tasks in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) whenever the [todos](#) ([src/todo/app.jsx](#)) state is updated.

Now, the tasks will be saved in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) and will remain available even after reloading the page.

00:19:26
- 00:19:32

> Response rendering

00:19:26

Researcher

What are you thinking now?

00:19:29

Participant

well, the task asks to save

00:19:32

in localStorage the to-dos, so when the page is reloaded it doesn't erase the to-dos.

00:19:39

so I just asked the copilot to generate this

00:19:43

code, but it seems that it's wrong the

00:19:51

Researcher

OK.

00:19:52

Participant

I'll ask to generate the correct code.

00:19:54

Researcher

Great.

00:19:57

> Viewing file (src/todo/app.jsx)

00:20:19
- 00:20:19

> Implementing Copilot code by copying full snippet (src/todo/app.jsx)

00:20:34

> Viewing file (src/todo/components/main.jsx)

00:21:52

Participant

now it seems that everything is right

00:21:59

Researcher

You can go to the next one

00:22:13

Participant

Alright.

00:23:27

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (src/todo/components/main.jsx)

> Message

@workspace I need to add a feature to save drafts that a user is writing in a task, I want this saving to occur every two seconds, do this for me.

> Response

To implement the feature of automatically saving drafts every two seconds, you can use [useEffect \(node_modules/@types/react/index.d.ts\)](#) with a time interval. Here's how you can do it in the component [Item \(src/todo/components/item.jsx\)](#):

Change in the file [src/todo/components/item.jsx](#):

Add automatic draft saving:

```
File: src/todo/components/item.jsx
import { memo, useState, useCallback, useEffect } from "react";
import classNames from "classnames";

import { Input } from "../input";

import { TOGGLE_ITEM, REMOVE_ITEM, UPDATE_ITEM } from "../constants";

export const Item = memo(function Item({ todo, dispatch }) {
  const [isWritable, setIsWritable] = useState(false);
  const [draft, setDraft] = useState(todo.title); // State for the
draft
  const { title, completed, id } = todo;

  const toggleItem = useCallback(() => dispatch({ type: TOGGLE_ITEM,
payload: { id } })), [dispatch]);
  const removeItem = useCallback(() => dispatch({ type: REMOVE_ITEM,
payload: { id } })), [dispatch]);
  const updateItem = useCallback((id, title) => dispatch({ type:
UPDATE_ITEM, payload: { id, title } })), [dispatch]);
```

```

const handleDoubleClick = useCallback(() => {
  setIsWritable(true);
}, []);

const handleBlur = useCallback(() => {
  setIsWritable(false);
}, []);

const handleUpdate = useCallback(
  (title) => {
    if (title.length === 0) removeItem(id);
    else updateItem(id, title);

    setIsWritable(false);
  },
  [id, removeItem, updateItem]
);

const handleDraftChange = useCallback((e) => {
  setDraft(e.target.value);
}, []);

// Automatic draft saving every 2 seconds
useEffect(() => {
  const interval = setInterval(() => {
    if (draft !== title) {
      updateItem(id, draft);
    }
  }, 2000);

  return () => clearInterval(interval); // Clears the interval on
unmount
}, [draft, title, id, updateItem]);

return (
  <li className={classnames({ completed: todo.completed })} data-
testid="todo-item">
    <div className="view">
      {isWritable ? (
        <Input
          onSubmit={handleUpdate}
          label="Edit Todo Input"
          defaultValue={draft}
          onBlur={handleBlur}
          onChange={handleDraftChange} // Updates the
draft
        />
      ) : (
        <>
          <input className="toggle" type="checkbox" data-
testid="todo-item-toggle" checked={completed} onChange={toggleItem} />
          <label data-testid="todo-item-label"
onDoubleClick={handleDoubleClick}>
            {title}
          </label>
          <button className="destroy" data-testid="todo-
item-button" onClick={removeItem} />
        </>
      )}
    </div>
  </li>
);
});

```

What was done:

1. Added the draft state:

- A local state to store the draft of the title while the user edits.

2. Added [useEffect \(node_modules/@types/react/index.d.ts\)](#) for automatic saving:

- Every 2 seconds, the draft is saved to the global state if it is different from the current title.

3. Updated the component [Input \(src/todo/components/input.jsx\)](#):

- Added the [onChange \(node_modules/@types/react/index.d.ts\)](#) event to update the draft as the user types.

Now, drafts will be automatically saved every two seconds while the user edits a task.

Participant

Here, basically as the task asks to save a draft, I'll ask

> Response rendering

Participant

For the copilot to generate.

00:23:30

00:23:32
- 00:23:50

00:23:39

00:23:42 the code that will do this basically every
00:23:46 2 seconds.
00:23:54 > **Viewing file** (src/todo/components/item.jsx)
00:23:54 **Researcher**
OK.
00:24:08 > **Implementing Copilot code** by copying full snippet (src/todo/components/item.jsx)
- 00:24:08
00:24:38 **Participant**
It didn't work.
00:24:54 > **Viewing file** (src/todo/components/item.jsx)
00:26:26 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, viewport (src/todo/components/item.jsx)
> **Message**
@workspace The draft didn't work, when I refresh the page the draft of the field is disappearing, review the code and find the problem.
> **Response**
The problem occurs because the draft ([draft\(src/todo/components/item.jsx\)](#)) is being saved in the global state ([todos\(src/todo/app.jsx\)](#)) every 2 seconds, but the initial value of the draft ([draft\(src/todo/components/item.jsx\)](#)) is not loaded from the global state when the component is mounted. This causes the draft to be lost when reloading the page.

Solution:

You need to initialize the state [draft\(src/todo/components/item.jsx\)](#) with the updated value of the title ([todo.title\(src/todo/components/footer.jsx\)](#)) whenever the component is mounted or when the title changes. To fix this, use [useEffect\(node_modules/@types/react/index.d.ts\)](#) to synchronize the state [draft\(src/todo/components/item.jsx\)](#) with the value of [todo.title\(src/todo/components/footer.jsx\)](#).

Here is the corrected code:

```
File: src/todo/components/item.jsx
import { memo, useState, useCallback, useEffect } from "react";
import classNames from "classnames";

import { Input } from "../input";

import { TOGGLE_ITEM, REMOVE_ITEM, UPDATE_ITEM } from "../constants";

export const Item = memo(function Item({ todo, dispatch }) {
  const [isWritable, setIsWritable] = useState(false);
  const [draft, setDraft] = useState(todo.title); // State for the
  draft
  const { title, completed, id } = todo;

  const toggleItem = useCallback(() => dispatch({ type: TOGGLE_ITEM,
  payload: { id } })), [dispatch]);
  const removeItem = useCallback(() => dispatch({ type: REMOVE_ITEM,
  payload: { id } })), [dispatch]);
  const updateItem = useCallback((id, title) => dispatch({ type:
  UPDATE_ITEM, payload: { id, title } })), [dispatch]);

  const handleDoubleClick = useCallback(() => {
    setIsWritable(true);
  }, []);
```

```

const handleBlur = useCallback(() => {
  setIsWritable(false);
}, []);

const handleUpdate = useCallback(
  (title) => {
    if (title.length === 0) removeItem(id);
    else updateItem(id, title);

    setIsWritable(false);
  },
  [id, removeItem, updateItem]
);

const handleDraftChange = useCallback((e) => {
  setDraft(e.target.value);
}, []);

// Synchronize the draft with the initial title
useEffect(() => {
  setDraft(title);
}, [title]);

// Auto-save the draft every 2 seconds
useEffect(() => {
  const interval = setInterval(() => {
    if (draft !== title) {
      updateItem(id, draft);
    }
  }, 2000);

  return () => clearInterval(interval); // Clear the interval on
unmount
}, [draft, title, id, updateItem]);

return (
  <li className={classnames({ completed: todo.completed })} data-
testid="todo-item">
    <div className="view">
      {isWritable ? (
        <Input
          onSubmit={handleUpdate}
          label="Edit Todo Input"
          defaultValue={draft}
          onBlur={handleBlur}
          onChange={handleDraftChange} // Update the
draft
        />
      ) : (
        <>
          <input className="toggle" type="checkbox" data-
testid="todo-item-toggle" checked={completed} onChange={toggleItem} />
          <label data-testid="todo-item-label"
onDoubleClick={handleDoubleClick}>
            {title}
          </label>
          <button className="destroy" data-testid="todo-
item-button" onClick={removeItem} />
        </>
      )}
    </div>
  </li>

```

```
});
```

What was fixed:

1. Synchronization of the draft with the initial title:

- Added a [useEffect\(node_modules/@types/react/index.d.ts\)](#) to update the state [draft\(src/todo/components/item.jsx\)](#) whenever the value of [todo.title\(src/todo/components/footer.jsx\)](#) changes.

Now, the draft will be correctly loaded when reloading the page, and changes made in the input field will be saved in the global state.

00:26:42
- 00:26:50

> Response rendering

00:27:04
- 00:27:04

> Implementing Copilot code by copying full snippet (src/todo/components/item.jsx)

00:28:01

> Viewing file (src/todo/components/item.jsx)

00:29:20

Chat (window: 0)

> Agent

github.copilot.workspace

> **Context**

workspace, viewport (src/todo/components/item.jsx)

> **Message**

@workspace It's still not working. I'm typing in the text input and clicking to reload the page, and when it reloads, no saved draft appears.

> **Response**

The problem occurs because the draft ([draft\(src/todo/components/item.jsx\)](#)) is being saved in the global state ([todos\(src/todo/app.jsx\)](#)) every 2 seconds, but the global state is not being correctly persisted in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) to reflect these changes in real-time. Additionally, the draft is not being loaded from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) when the page reloads.

Solution:

1. **Persist the global state in [localStorage](#)**

([node_modules/@types/node/globals.d.ts](#)) in real-time:

- o Ensure that the global state ([todos\(src/todo/app.jsx\)](#)) is updated in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) whenever the draft is saved.

2. **Load the draft from [localStorage](#)**

([node_modules/@types/node/globals.d.ts](#)) on initialization:

- o The draft should be loaded from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) when the component mounts.

Here is the corrected code for the component [Item\(src/todo/components/item.jsx\)](#):

```
File: src/todo/components/item.jsx
import { memo, useState, useCallback, useEffect } from "react";
import classNames from "classnames";

import { Input } from "../input";

import { TOGGLE_ITEM, REMOVE_ITEM, UPDATE_ITEM } from "../constants";

export const Item = memo(function Item({ todo, dispatch }) {
  const [isWritable, setIsWritable] = useState(false);
  const [draft, setDraft] = useState(todo.title); // State for the
  draft
  const { title, completed, id } = todo;

  const toggleItem = useCallback(() => dispatch({ type: TOGGLE_ITEM,
    payload: { id } }), [dispatch]);
  const removeItem = useCallback(() => dispatch({ type: REMOVE_ITEM,
    payload: { id } }), [dispatch]);
  const updateItem = useCallback((id, title) => dispatch({ type:
    UPDATE_ITEM, payload: { id, title } }), [dispatch]);

  const handleClick = useCallback(() => {
    setIsWritable(true);
  }, []);

  const handleBlur = useCallback(() => {
    setIsWritable(false);
  }, []);

  const handleUpdate = useCallback(
    (title) => {
```

```
        if (title.length === 0) removeItem(id);  
        else updateItem(id, title);  
  
        setIsWritable(false);  
    },  
    [id, removeItem, updateItem]
```

```

    );

    const handleDraftChange = useCallback((e) => {
        setDraft(e.target.value);
    }, []);

    // Synchronize the draft with the initial title
    useEffect(() => {
        setDraft(title);
    }, [title]);

    // Auto-save the draft every 2 seconds
    useEffect(() => {
        const interval = setInterval(() => {
            if (draft !== title) {
                updateItem(id, draft);

                // Update localStorage in real-time
                const todos = JSON.parse(localStorage.getItem("todos"));

                const updatedTodos = todos.map((todo) =>
                    todo.id === id ? { ...todo, title: draft } : todo
                );
                localStorage.setItem("todos",
                    JSON.stringify(updatedTodos));
            }
        }, 2000);

        return () => clearInterval(interval); // Clear the interval on
        unmount
    }, [draft, title, id, updateItem]);

    return (
        <li className={classnames({ completed: todo.completed })} data-
        testid="todo-item">
            <div className="view">
                {isWritable ? (
                    <Input
                        onSubmit={handleUpdate}
                        label="Edit Todo Input"
                        defaultValue={draft}
                        onBlur={handleBlur}
                        onChange={handleDraftChange} // Update the
                        draft
                    />
                ) : (
                    <>
                        <input className="toggle" type="checkbox" data-
                        testid="todo-item-toggle" checked={completed} onChange={toggleItem} />
                        <label data-testid="todo-item-label"
                            onDoubleClick={handleDoubleClick}>
                            {title}
                        </label>
                        <button className="destroy" data-testid="todo-
                        item-button" onClick={removeItem} />
                    </>
                )}
            </div>
        </li>
    );
};
});

```

What was fixed:

1. Real-time persistence in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)):

- Whenever the draft is saved, the global state ([todos\(src/todo/app.jsx\)](#)) is updated in [localStorage\(node_modules/@types/node/globals.d.ts\)](#).

2. Initial draft loading:

- The draft is synchronized with the initial title when the component mounts.

Now, when the page reloads, the draft will be correctly loaded from [localStorage\(node_modules/@types/node/globals.d.ts\)](#).

> Response rendering

> Implementing Copilot code by copying full snippet (src/todo/components/item.jsx)

Participant

Or it's not working. I'm thinking of some way to explain exactly what I want to happen.

I think maybe I might have given a generic description.

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (src/todo/components/item.jsx)

> Message

@workspace It's still not working, if I'm adding a new task and reload the page, the text that was written disappears, fix this, when I type in the input field to create a new task it should save everything I'm typing at all times, in case I reload the page, it should return all the typed characters to the creation input, and when I finish creating, delete the draft.

> Response

To fix the problem and ensure that the text typed in the input field to create a new task is continuously saved and restored after reloading the page, you can use [localStorage\(node_modules/@types/node/globals.d.ts\)](#) to store the creation input draft. Additionally, the draft should be deleted when the task is created.

Here is the solution:

Change in the file [src/todo/components/header.jsx](#):

Add continuous saving of the draft in [localStorage\(node_modules/@types/node/globals.d.ts\)](#) and restore the draft when loading the page.

```
File: src/todo/components/header.jsx
import { useCallback, useEffect, useState } from "react";
import { Input } from "../input";

import { ADD_ITEM } from "../constants";

const LOCAL_STORAGE_KEY_DRAFT = "newTodoDraft";

export function Header({ dispatch }) {
  const [draft, setDraft] = useState("");

  // Load the draft from localStorage when the component mounts
  useEffect(() => {
    const savedDraft =
      localStorage.getItem(LOCAL_STORAGE_KEY_DRAFT);
    if (savedDraft) {
```

00:29:25
- 00:29:38

00:29:48
- 00:29:48

00:32:44

00:32:50

00:32:54

00:34:41

```

        setDraft(savedDraft);
    }
}, []);

// Save the draft in localStorage whenever it changes
useEffect(() => {
    localStorage.setItem(LOCAL_STORAGE_KEY_DRAFT, draft);
}, [draft]);

const addItem = useCallback(
    (title) => {
        dispatch({ type: ADD_ITEM, payload: { title } });
        setDraft(""); // Clear the draft after creating the task
        localStorage.removeItem(LOCAL_STORAGE_KEY_DRAFT); // Remove
the draft from localStorage
    },
    [dispatch]
);

const handleDraftChange = useCallback((e) => {
    setDraft(e.target.value);
}, []);

return (
    <header className="header" data-testid="header">
        <h1>Tasks</h1>
        <Input
            onSubmit={addItem}
            label="New task"
            placeholder="Type a task here"
            value={draft} // Use the draft as the input value
            onChange={handleDraftChange} // Update the draft when
typing
        />
    </header>
);
}

```

Change in the file [src/todo/components/input.jsx](#):

Update the [Input\(src/todo/components/input.jsx\)](#) component to support value control via props.

```

File: src/todo/components/input.jsx
import { useCallback } from "react";

const isValidMin = (value, min) => {
    return value.length >= min;
};

export function Input({ onSubmit, placeholder, label, value, onChange,
defaultValue, onBlur }) {
    const handleBlur = useCallback(() => {
        if (onBlur) onBlur();
    }, [onBlur]);

    const handleKeyDown = useCallback(
        (e) => {
            if (e.key === "Enter") {
                const inputValue = e.target.value.trim();

                if (!isValidMin(inputValue, 2)) return;
            }
        }
    );
}

```



```

        onSubmit(inputValue);
        e.target.value = ""; // Clear the field after
submission
    }
    },
    [onSubmit]
);

return (
    <div className="input-container">
        <input
            className="new-todo"
            id="todo-input"
            type="text"
            data-testid="text-input"
            autoFocus
            placeholder={placeholder}
            value={value} // Control the input value
            onChange={onChange} // Update the value when typing
            defaultValue={defaultValue}
            onBlur={handleBlur}
            onKeyDown={handleKeyDown}
        />
        <label className="visually-hidden" htmlFor="todo-input">
            {label}
        </label>
    </div>
);
}

```

What was done:

1. Draft persistence in [localStorage](#)

([node_modules/@types/node/globals.d.ts](#)):

- The creation input draft is continuously saved in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) whenever the user types.

2. Draft restoration when loading the page:

- The draft is loaded from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) when the [Header](#) ([src/todo/components/header.jsx](#)) component mounts.

3. Draft cleanup after creating the task:

- The draft is cleared from the state and [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) when the task is created.

4. Update of the [Input](#) ([src/todo/components/input.jsx](#)) component:

- Added support for [value \(src/todo/components/input.jsx\)](#) and [onChange \(src/todo/components/item.jsx\)](#) to control the input value externally.

Now, the text typed in the creation input will be continuously saved and restored after reloading the page. When the task is created, the draft will be deleted.

> Response rendering

> **Implementing Copilot code** by copying full snippet (src/todo/components/item.jsx)

> **Viewing file** (src/todo/components/input.jsx)

> **Implementing Copilot code** by copying full snippet (src/todo/components/input.jsx)

Participant

I think that's it.

Researcher

It worked, it worked.

You can go to task c.

Participant

OK

The task asked to implement the pagination system,

so I will pass on to it all the

specifications that are being requested.

Researcher

Cool.

> **Viewing file** (src/todo/components/paginationControls.jsx)

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (src/todo/components/paginationControls.jsx)

> Message

@workspace I need to add a pagination system, with options to list 1, 3, or 5 tasks per page. I also want to add previous and next buttons, so the user can navigate between pages, and if they are at one of the edges, the button should be gray and unavailable to click. Additionally, there should be information about which page the user is on. I want the page to update automatically when changing the choice of how many tasks per page.

> Response

To implement the pagination system with the described functionalities, you can modify the component [PaginationControls \(src/todo/components/paginationControls.jsx\)](#) and adjust the component [Main \(src/todo/components/main.jsx\)](#) to integrate the pagination logic.

Changes in the file [src/todo/components/paginationControls.jsx](#):

Add the functionality to select items per page and display the current page.

```
File: src/todo/components/paginationControls.jsx
import React from "react";
import classNames from "classnames";

export function PaginationControls({ currentPage, setCurrentPage,
itemsPerPage, setItemsPerPage, totalItems }) {
  const totalPages = Math.ceil(totalItems / itemsPerPage);

  const handlePageChange = (direction) => {
    setCurrentPage((prevPage) => {
      if (direction === "prev" && prevPage > 1) {
        return prevPage - 1;
      }
    });
  };
}
```

```

    }
    if (direction === "next" && prevPage < totalPages) {
      return prevPage + 1;
    }
    return prevPage;
  });
};

const handleItemsPerPageChange = (e) => {
  setItemsPerPage(Number(e.target.value));
  setCurrentPage(1); // Resets to the first page when changing
items per page
};

return (
  <div className="pagination-controls">
    <button
      className={classnames("pagination-button", { inactive:
currentPage === 1 })}
      onClick={() => handlePageChange("prev")}
      disabled={currentPage === 1}
    >
      Previous
    </button>
    <span className="pagination-info">
      Page {currentPage} of {totalPages}
    </span>
    <button
      className={classnames("pagination-button", { inactive:
currentPage === totalPages })}
      onClick={() => handlePageChange("next")}
      disabled={currentPage === totalPages}
    >
      Next
    </button>
    <select
      className="items-per-page-select"
      value={itemsPerPage}
      onChange={handleItemsPerPageChange}
    >
      <option value={1}>1 per page</option>
      <option value={3}>3 per page</option>
      <option value={5}>5 per page</option>
    </select>
  </div>
);
}

```

Changes in the file [src/todo/components/main.jsx](#):

Integrate the pagination logic to display tasks according to the current page and the number of items per page.

```

File: src/todo/components/main.jsx
import { Item } from "../item";
import classnames from "classnames";
import { TOGGLE_ALL } from "../constants";
import { useLocation } from "react-router-dom";
import { useMemo, useCallbak, useState, useEffect } from "react";
import { PaginationControls } from "../paginationControls";

export function Main({ todos, dispatch }) {
  const { pathname: route } = useLocation();

```



```

const [currentPage, setCurrentPage] = useState(1);
const [itemsPerPage, setItemsPerPage] = useState(5);

const visibleTodos = useMemo(() => {
  const filteredTodos = todos.filter((todo) => {
    if (route === "/active") return !todo.completed;
    if (route === "/completed") return todo.completed;
    return todo;
  });

  return filteredTodos.sort((a, b) =>
a.title.localeCompare(b.title));
  }, [todos, route]);

const paginatedTodos = useMemo(() => {
  const startIndex = (currentPage - 1) * itemsPerPage;
  const endIndex = startIndex + itemsPerPage;
  return visibleTodos.slice(startIndex, endIndex);
}, [visibleTodos, currentPage, itemsPerPage]);

const toggleAll = useCallback(
  (e) => dispatch({ type: TOGGLE_ALL, payload: { completed:
e.target.checked } }),
  [dispatch]

```

```

    );

    return (
      <main className="main" data-testid="main">
        {paginatedTodos.length > 0 ? (
          <>
            <div className="toggle-all-container">
              <input
                className="toggle-all"
                type="checkbox"
                id="toggle-all"
                data-testid="toggle-all"
                checked={visibleTodos.every((todo) =>
todo.completed)}
                onChange={toggleAll}
              />
              <label className="toggle-all-label"
htmlFor="toggle-all">
                Toggle All Input
              </label>
            </div>
            <ul className={classnames("todo-list")} data-
testid="todo-list">
              {paginatedTodos.map((todo, index) => (
                <Item todo={todo} key={todo.id} dispatch=
{dispatch} index={index} />
              ))}
            </ul>
          </>
        ) : (
          <div className="todo-list-empty-container">
            <span className="todo-list-empty-message">No tasks
here yet</span>
          </div>
        )}

        <PaginationControls
          currentPage={currentPage}
          setCurrentPage={setCurrentPage}
          itemsPerPage={itemsPerPage}
          setItemsPerPage={setItemsPerPage}
          totalItems={visibleTodos.length}
        />
      </main>
    );
  }
}

```

What was done:

1. Added control of items per page:

- A <select> allows the user to choose between 1, 3, or 5 items per page.
- The current page is reset to 1 when changing the number of items per page.

2. Pagination with navigation buttons:

- "Previous" and "Next" buttons allow navigation between pages.
- The buttons are disabled at the edges (first and last page).

3. Display of the current page:

- Shows the current page and the total number of pages.

Now, the pagination system is complete and functional.

00:41:45
- 00:42:07

00:42:19
- 00:42:19

00:42:36

00:42:38
- 00:42:38

00:44:46

00:44:49

00:45:25

00:48:51

00:48:53

00:49:03

00:49:06

00:50:51

00:50:56

00:50:59

00:51:05

00:51:09

00:51:12

00:51:17

00:51:48

00:53:46

00:53:53

00:53:55

00:53:58

00:54:07

00:54:13

00:54:19

00:54:24

00:54:28

00:54:34

00:54:37

00:54:40

00:54:41

00:54:47

00:54:52

00:54:56

00:55:17

00:55:20

00:55:28

00:55:32

> Response rendering

> **Implementing Copilot code** by copying full snippet
(src/todo/components/paginationControls.jsx)

> **Viewing file** (src/todo/components/main.jsx)

> **Implementing Copilot code** by copying full snippet (src/todo/components/main.jsx)

Participant

I'll check if everything is right

Researcher

OK

Ok, everything is right. Thank you. You finished all the tasks on time.

Now I'll send you the post-study survey ok

Participant

ok.

Researcher

If you want you can comment on the
questions.

Participant

I agree with this because.

I believe it is quite complex, really I.

read this here and understand how to really learn all the functionalities of the
Copilot.

I'm not sure if I'm 100% sure,
how easy it would be to thoroughly learn everything

Researcher

ok

Participant

I agree a little

Researcher

Alright. Thank you. Now I will ask a few more questions ok?

Participant

OK

Researcher

Were you able to navigate through the code
and how did the copilot help you, influence you?,

Participant

I worked a little with react before, so I already knew more or less how.
How the code works a bit. I don't work anymore now I'm a backend
developer but

I knew more or less what to look for and

When I saw the code, I don't really know how this works.
file differentiation, where something is.

In the front end.

Researcher

ok.

Participant

So the copilot's suggestions helped me a lot to really not have to spend
the time looking for which file the important things are in
I think in this part it helps a lot.

Researcher

Cool.

Participant

So, whenever the copilot adds the code,
it usually puts comments on the block of code
or line comment. This helps a lot in the explanation,
it really helps to understand what's happening.

00:55:34 **Researcher**
cool.

00:55:37 And how would you have solved these tasks without the help of the copilot

00:55:45 **Participant**
Probably I would have searched the internet.

00:55:50 Considering that I didn't know how to do any,

00:55:54 of the tasks, I would search the internet.

00:56:09 I mainly use Stack Overflow.

00:56:13 **Researcher**
ok.

00:56:13 **Participant**
To look for solutions for specific problems.

00:56:33 **Researcher**
What did you like, what didn't you like about the copilot?

00:56:37 **Participant**
I already use the copilot daily.

00:56:42 I mainly like how you can.

00:56:50 Give it a lot of context.

00:56:54 In the projects I've already used,

00:56:58 it easily gets the context.

00:57:07 You don't have to keep passing it to him.

00:57:11 **Researcher**
ok

00:57:15 **Participant**
Things I don't like.

00:57:18 Hmm, difficult, I don't know if there's anything that

00:57:22 really bothers me.

00:57:25 **Researcher**
ok, no problem.

00:57:30 Did the copilot influence your strategy to solve the tasks?

00:57:50 **Participant**
I really thought about using the copilot directly
thinking more about productivity.

00:57:55 So, there are many, most of the

00:58:22 The copilot doesn't always give the correct answer, it doesn't always give the final answer.

00:58:46 From the answer, I start to tweak the code and change it,

00:58:51 but usually I start with it anyway.

00:58:54 **Researcher**
Cool.

00:59:08 Did you change your interaction strategy with the copilot during the experiment?

00:59:26 **Participant**
Yes, at a certain point

00:59:29 I didn't express myself correctly, so I had to be a bit more

00:59:33 detailed. I was sending slightly more

00:59:37 generic prompts.

00:59:39 For the productivity task, I needed to send a much more

00:59:44 specific prompt to the copilot.

00:59:58 **Researcher**
Ok, that's all. Thank you for participating in our experiment.