

Participant P26

00:05:56 > **Viewing file** (src/todo/app.css)

00:06:13 **Participant**
So the in these case since I'm not.
00:06:16 I think it is a JavaScript right, web application and I'm not familiar with
00:06:21 that.
00:06:21 So then I will be like a beginner asking GitHub Copilot. How can I do this?
00:06:29 Is that.
00:06:31 Right procedure here?
00:06:34 **Researcher**
How to do it this for you to figure out, but that is within the the allowed limits,

00:06:37 **Participant**
OK.

00:06:39 **Researcher**
yes.

00:06:40 **Participant**
Yeah. OK.
00:06:42 So what I need to do is to first figure out how to run the application and then come
00:06:48 back here to continue with the task?
00:06:52 OK.
00:07:23 **Chat** (window: 0)

> **Agent**
github.copilot.workspace

> **Context**
workspace, viewport (src/todo/app.css)

> **Message**
@workspace I am not familiar with this codebase and Javascript. Can you please tell me how to run this application?

> **Response**
To run this application, follow these steps:

1. Install dependencies by running:

```
npm install
```
2. Start the development server with:

```
npm run dev
```

This will launch the app in your browser using the configuration in [webpack.dev.js](#).
The entry point is [src/index.js](#), which renders the React app.

00:07:26 - 00:07:27 > **Response rendering**

00:07:41 **Participant**
Install dependency by running.
00:07:44 Is the dependency already installed?
00:07:48 Or I need to also follow the steps to install the dependency?
00:07:54 **Researcher**
I can't give you any any more hints here.

00:07:56 **Participant**
OK, OK.

00:08:01 **Researcher**
Sorry, yeah, you need to allow this pop up.

00:08:04 **Participant**
Mm hmm.

00:08:32

Hmm.

00:08:37

OK, it works.

00:08:40

Researcher

Perfect.

00:08:42

So when you're ready, feel free to get started on the practice

00:08:44

tasks.

00:08:51

Participant

With the message not todo here yet.

00:08:55

Change the color.

00:08:59

Change this color. To blue.

00:09:38

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

viewport (src/todo/app.css)

> **Message**

Can you tell me which part of code implement this feature: there is a red box with the message "No TODOs here yet".

> **Response**

The feature you described—a red box with the message "No TODOs here yet"—is styled by the following CSS classes in your [src/todo/app.css](#) file:

```
.todo-list-empty-container {  
  padding: 5px;  
  display: flex;  
  justify-content: center;  
}  
  
.todo-list-empty-message {  
  padding: 10px;  
  background: #b83f45;  
  color: white;  
  border-radius: 5px;  
  font-weight: bolder;  
}
```

The actual message "No TODOs here yet" is likely rendered in a React component (probably in a file like `TodoList.jsx` or similar). Look for code that checks if the todo list is empty and then renders a `<div>` or similar element with the class `todo-list-empty-message`. For example:

```
{todos.length === 0 && (  
  <div className="todo-list-empty-container">  
    <div className="todo-list-empty-message">No TODOs here yet</div>  
  </div>  
)}
```

To find the exact implementation, search your codebase for `todo-list-empty-message` or the text "No TODOs here yet". This will lead you to the component responsible for rendering this feature.

00:09:39
- 00:09:42

> **Response rendering**

00:09:51

Participant

Mm hmm.

00:09:54

Researcher

So what are you thinking right now?

00:09:57

Participant

So I'm trying to locate the part of the code which implements the box.

00:10:03

And since this one.

00:10:09

Let me.

00:10:11
00:10:19
00:10:26
00:10:34

Sorry, empty container. So I'm now trying to find the piece of code.
Which is right here?

How can I change the color from red to blue?

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

selection (src/todo/app.css)

> **Message**

How can I change the color from red to blue?

> **Response**

To change the color of the red box to blue, update the background property in the .todo-list-empty-message CSS class.

Change this:

```
background: #b83f45;
```

to:

```
background: #007bff;
```

So the updated CSS will look like this:

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

00:10:35
- 00:10:38

> **Response rendering**

00:10:52

> **Editing file** (src/todo/app.css)

00:11:01

Participant

So now I save it.

00:11:05

I save the change and I want to.

00:11:09

Run it again.

00:11:15

Researcher

Yeah. Just so you know, in the future you can just refresh the web page and the changes should show.

00:11:18

00:11:20

Participant

OK.

00:11:22

So now it seems refreshed. OK, refresh the application, confirm that it's in a blue box.

00:11:27

00:11:29

So I think this one is finished.

00:11:33

And the second one, the placeholder text of the input field

00:11:37

is currently what need to be done.

00:11:40

Change the text to OK, since in the previous step I already know where implement this interface, so I now search.

00:11:47

00:11:57

Needs to be done.

00:11:59

Actually, I want to search in entire code base.

00:12:02

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

00:12:04

Participant

So I found it here and then.

00:12:09

Change it to enter a task here.

00:12:21

- 00:12:27

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

00:12:35

Participant

OK.

00:12:37

I think this one is also finished and task C, sort todos. The todos are currently displayed in the order they were added.

00:12:43

Modify the application so that the todo are sorted alphabetically instead.

00:12:55

So I want to know where in the code base.

00:13:33

It actually display the todos.

00:13:37

Chat (window: 0)

00:14:06

> Agent

github.copilot.default

> Context

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

> Message

where in the codebase implements the display of TODOs?

> Response

The display of TODOs is typically handled by a component that renders the list of todo items. In a codebase like this, look for a file named something like `list.jsx`, `todo-list.jsx`, or similar, usually in the [src/todo/components/](#) directory.

You can search for keywords like: - `todos.map` (common in React for rendering lists) - `TodoList` - `list`

The component will likely receive the list of todos as a prop and render each item, possibly using another component for each todo (like `TodoItem`). For example, you might see code like:

```
<ul>
  {todos.map(todo => (
    <TodoItem key={todo.id} {...todo} />
  ))}
</ul>
```

Summary:

Look for a component such as `TodoList` or similar in the [src/todo/components/](#) folder. That component is responsible for displaying the TODOs.

00:14:08

- 00:14:11

> Response rendering

00:14:35

Participant

I think probably I was not asking.

00:14:39

Question about this workspace, so I want to ask it again.

00:14:39

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

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

> Message

@workspace where in the codebase implements the display of TODOs?

> Response

The display of TODOs is implemented in the [Main\(src/todo/components/main.jsx\)](#) component in [src/todo/components/main.jsx](#). This component renders the list of TODO items using the [Item\(src/todo/components/item.jsx\)](#) component for each todo. The list is shown inside a `<ul>` with the class [todo-list\(src/todo/components/item.jsx\)](#).

00:14:41

- 00:14:42

> Response rendering

00:14:45

Participant

In the main component.

00:14:49

OK.

00:14:54
00:15:00
00:15:03
00:15:09
00:15:15
00:16:02

In main.

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

Participant

Renders the list of todo items using the item component for each todo.

The list is shown inside a ul with the class.

Todo.

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

I want to take display the entered TODOs in an alphabetical order. Can you suggest the change?

> **Response**

To display the TODOs in alphabetical order, you should sort the [visibleTodos \(src/todo/components/main.jsx\)](#) array by the todo title before rendering. You can do this by updating the [visibleTodos \(src/todo/components/main.jsx\)](#) calculation in your [useMemo \(node_modules/@types/react/index.d.ts\)](#) hook.

Here's how you can modify it:

```
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;
      })
      .slice() // create a shallow copy to avoid mutating the
original array
      .sort((a, b) => a.title.localeCompare(b.title)), // sort
alphabetically by title
  [todos, route]
);
```

This will ensure your TODOs are always displayed in alphabetical order by their title.

> **Response rendering**

Participant

You should sort the visible todo.

Array by the todo title before rendering, you can do this by updating.

This one.

> **Viewing file** (node_modules/%40types/react/index.d.ts) through Copilot link

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

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

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

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

Researcher

So what are you thinking now?

Participant

Now I'm so I just copy paste the suggested code.

00:16:03
- 00:16:05
00:16:15
00:16:20
00:16:26
00:16:27
00:16:54
00:17:03
- 00:17:06
00:17:07
00:17:16
00:17:35
00:17:36

00:17:42 And then I'm now looking at this code snippet.
00:17:46 This is the part that was added since I don't know JavaScript,
00:17:51 so now I'm now I'm trying to reason and see if this correct and I think it is
00:17:57 correct and now I'm going to validate the result.
00:18:03 So the quick 123.
00:18:18 How can I add the todos?
00:18:26 The following todos in this order.
00:18:38 Double Click to add it A to do.
00:18:43 So I do not completely understand this application.
00:18:47 **Researcher**
OK.
00:18:50 Could you try to figure out how it how it works?
00:18:54 **Participant**
Let me try.
00:18:56 No idea yet.
00:19:00 Is this a button?
00:19:41 Yes, I think it is correct.
00:19:45 **Researcher**
Perfect. Thanks.
00:19:47 Now those were the practice tasks. I'll send you the actual tasks. I see you
00:19:54 have.
00:19:57 27, 26 minutes left.
00:20:02 So yeah, don't worry if there's not enough time to
00:20:04 complete all of them, that's no problem.
00:20:08 **Participant**
OK.
00:20:12 [unintelligible]
00:20:22 Even after refresh. Ensure that each todo state, whether it is completed or active.
00:20:29 Is also preserved.
00:20:32 An attempt to implement this functionality was made,
00:20:35 but it's not working correctly at this moment.
00:20:51 So I'm going to just do.
00:20:58 A lazy thing here.
00:21:00 I copy paste the text here and then I say please suggest the fix.
00:21:10 **Chat** (window: 0)
> Agent
github.copilot.workspace
> Context
workspace, viewport (src/todo/components/main.jsx)
> Message
@workspace Currently, all TODOs disappear when you refresh the page. Modify the application so that TODOs remain even after a refresh. Ensure that each TODO's state (whether it is completed or active) is also preserved. An attempt to implement this functionality was made, but it is not working correctly at this moment. Please suggest the fix
> Response
To persist TODOs across refreshes, you need to:

1. **Fix the typo in your reducer:**
In [src/todo/reducer.js](#), the action type is misspelled as [LOADSTATE](#) ([src/todo/reducer.js](#)) instead of [LOADSTATE](#) ([src/todo/constants.js](#)).
2. **Initialize state from localStorage in the app:**
Use the [loadState](#) ([src/todo/reducer.js](#)) function as the initial state in your [useReducer](#) ([node_modules/@types/react/index.d.ts](#)) in [src/todo/app.jsx](#).
3. **Save state to localStorage under the correct key:**
Your [saveState](#) ([src/todo/reducer.js](#)) function uses "todos" as the key, but [loadState](#) ([src/todo/reducer.js](#)) uses "todoS". Make sure both use "todos".

Here are the changes you need:

```
File: src/todo/reducer.js
// ...existing code...
export const loadState = () => {
  const state = localStorage.getItem("todos"); // fix key to "todos"
  return state ? JSON.parse(state) : [];
};
// ...existing code...
    case LOAD_STATE: // fix typo here
      newState = action.payload;
      break;
// ...existing code...

File: src/todo/app.jsx
// ...existing code...
import { loadState, todoReducer } from "../reducer";
// ...existing code...
export function App() {
  const [todos, dispatch] = useReducer(todoReducer, [], loadState);
  // use loadState as initializer
  // ...existing code...
}
```

With these changes, TODOs and their completed/active state will persist after a refresh.

00:21:14
- 00:21:18

> Response rendering

00:21:25

Participant

Instead of load state.

00:21:27

> Viewing file (src/todo/reducer.js)

00:21:34

Participant

Mm hmm.

00:21:36
- 00:21:39

> Editing file (src/todo/reducer.js)

00:21:42

Participant

Initialize state from local storage in the app.

00:21:45

Use the loadstate function as the initial state in your use reducer.

00:21:56

I guess you need save it.

00:22:03

Save to local storage under the correct key.

00:22:07

Your save state function uses todos as the key, but load state used todo S.

00:22:13

Make sure both used todo S.

00:22:45

So I'm now trying to understand the logic here.

00:22:51

Save state to local storage under the correct key.

00:22:54

Your save state function uses todo as the key.

00:22:59 Yes.
00:23:02 But load state uses todo
00:23:06 S, make sure.
00:23:11 > **Editing file** (src/todo/reducer.js)
00:23:14 **Participant**
OK.
00:23:46 No, it's not working.
00:23:54 [unintelligible]
00:24:24 I'm now trying to understand the suggestion here.
00:24:30 Since it is in the chatbox,
00:24:33 not in line, so I'm trying to understand where.
00:24:40 Yeah. Where is the part that need to be fixed?
00:25:01 OK.
00:25:26 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (src/todo/reducer.js)
> **Message**
where is export function App()?
> **Response**
The export function App() is most likely defined in the file:

/workspaces/copilot-study-todo/src/todo/app.jsx

This is a common convention for React applications, where the main application component is named App and placed in app.jsx. You can open this file to view or edit the App component.
00:25:27
- 00:25:29
00:25:40 > **Response rendering**
00:25:40 > **Viewing file** (src/todo/app.jsx)
00:25:40 **Participant**
In app.
00:25:43 App.
00:25:45 It suggests.
00:25:50 Load state as initializer.
00:26:02 > **Implementing Copilot code** by copying part of snippet (src/todo/app.jsx)
00:26:26 > **Implementing Copilot code** by copying part of snippet (src/todo/app.jsx)
00:26:48 **Participant**
Completed, refresh the page and confirm both the todos and their state remains unchanged. So this one completed.
00:26:54 OK.
00:27:00 Edit some todos, refresh the change and confirm.
00:27:01 Yes.
00:27:12 And delete some todos, refresh the change.
00:27:14 Let's see this one.
00:27:20 Yes.
00:27:23 OK. The second one, if a user refreshes the page while typing
00:27:27 a new todo, the text in the input field disappears.
00:27:32 Modify the application so that the input field automatically saves drafts every few seconds
00:27:42 and restore them when the page reloads, OK.
00:27:48 **Chat** (window: 0)
00:28:17 > **Agent**
github.copilot.workspace
> **Context**
workspace, viewport (src/todo/app.jsx)
> **Message**

@workspace If a user refreshes the page while typing a new TODO, the text in the input field disappears. Modify the application so that the input field automatically saves drafts every few seconds, and restore them when the page reloads. Please suggest the changes

> **Response**

To implement auto-saving and restoring drafts for the TODO input:

1. **Persist the draft in localStorage as the user types.**
2. **Restore the draft from localStorage when the input mounts.**
3. **Auto-save every few seconds (e.g., 2 seconds) while typing.**

Update [Input\(src/todo/components/input.jsx\)](#) in [src/todo/components/input.jsx](#) as follows:

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

const DRAFT_KEY = "todo-input-draft";

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

export function Input({ onSubmit, placeholder, label, defaultValue,
onBlur }) {
  const [value, setValue] = useState(() => {
    // Restore draft from localStorage if present
    return localStorage.getItem(DRAFT_KEY) || defaultValue || "";
  });
  const saveTimeout = useRef();

  // Auto-save draft every 2 seconds
  useEffect(() => {
    if (saveTimeout.current) clearTimeout(saveTimeout.current);
    saveTimeout.current = setTimeout(() => {
      localStorage.setItem(DRAFT_KEY, value);
    }, 2000);
    return () => clearTimeout(saveTimeout.current);
  }, [value]);

  // Clear draft on submit
  const handleKeyDown = useCallback(
    (e) => {
      if (e.key === "Enter") {
        const trimmed = value.trim();
        if (!hasValidMin(trimmed, 2)) return;
        onSubmit(trimmed);
        setValue("");
        localStorage.removeItem(DRAFT_KEY);
      }
    },
    [onSubmit, value]
  );

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

  const handleBlur = useCallback(() => {
    if (onBlur) onBlur();
  }, [onBlur]);

  return (
    <div className="input-container">
```

```

        <input
          className="new-todo"
          id="todo-input"
          type="text"
          data-testid="text-input"
          autoFocus
          placeholder={placeholder}
          value={value}
          onChange={handleChange}
          onBlur={handleBlur}
          onKeyDown={handleKeyDown}
        />
        <label className="visually-hidden" htmlFor="todo-input">
          {label}
        </label>
      </div>
    );
  }

```

This will auto-save the draft every 2 seconds and restore it on reload. The draft is cleared when a TODO is submitted.

> Response rendering

Participant

Save every few seconds, while typing.

So now it suggests a piece of code, but it doesn't exactly tell where should

I make this change?

Oh no, it suggests here input JSX.

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

Participant

Update the input function.

So it seems that the code snippet is actually for entire file.

What I will do is I copy paste.

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

Participant

Comment this one out.

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

Participant

No, it doesn't work.

Persist the draft in the local storage as the user types.

Restore the draft on from local storage when the input mounts.

00:30:04 Auto save every few seconds.
00:30:16 So it suggests.
00:30:20 Change to this file.
00:30:20 What I did is I copy paste but it doesn't work.
00:30:25 So now I need to understand what exact change it actually suggests.
00:30:31 So what's the difference between the the the one suggested by GitHub Copilot and
00:30:37 the original one?
00:31:20 Auto save draft every two seconds.
00:31:26 Which might also mean that I should type and wait for several seconds.
00:31:36 And then refresh it.
00:31:38 Then I see it.
00:31:39 So actually the code is correct.
00:31:43 The suggested change is correct, but when I tested it,
00:31:46 I did not wait for two seconds.
00:31:50 So.
00:31:51 Let me test everything else.
00:31:54 Type a new todo without submitting.
00:31:57 Refresh the page and confirm the input is still there.
00:32:00 So this one I already verified. Type a new todo, submit it, refresh and confirm it is
00:32:06 still there.
00:32:07 Here. OK, type it and submit it and then refresh.
00:32:14 It's still there.
00:32:19 So then I can safely remove this piece of code.
00:32:22 **> Editing file** (src/todo/components/input.jsx)
- 00:32:24

00:32:29

Participant

00:32:34 Come back to this. The last one. Currently the todo application displays
00:32:38 all tasks in a single list. Implement.
00:32:48 Pagination to limit the number of displayed todos per page.
00:32:53 To limit the number of displayed todos, allowing users to easily navigate through their
00:32:57 tasks.
00:33:04 A fixed number of todos per page with options to show 1, 3, 5.
00:33:07 At a time, also implement navigation controls to
00:33:10 allow users to move between pages.
00:33:13 Of todos.
00:33:13 Should have next, previous button to navigate between pages and [unintelligible].
00:33:19 Disabled and grey when the user is on the last page and the previous button should
00:33:24 be disabled and grey when the user is on the first page.
00:33:28 The page that the user is currently on should also be indicated.
00:33:33 An attempt to implement the navigation functionality was made but it.
00:33:38 Was not working.
00:33:41 Please fix it.
00:33:42 So my impression is that this a very big feature.
00:33:48 And maybe I shall.
00:33:52 Decompose it into several smaller features.
00:33:58 And then use GitHub Copilot to help me implement the idea.
00:34:38 **Chat** (window: 0)

> Agent

github.copilot.workspace

> Context

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

> Message

@workspace Currently, the TODO application displays all tasks in a single list. Implement pagination to limit the number of displayed TODOs per page, allowing users to easily navigate through their tasks. Please suggest how to modify the application to display a fixed number of TODOs per page, with options to show 1, 3, or 5 (the default value) TODOs at a time.

> Response

To implement pagination with a selectable number of TODOs per page (1, 3, or 5), follow these steps:

1. Add a selector for items per page

Add a dropdown in [Main \(src/todo/components/main.jsx\)](#) to let users choose how many TODOs to show per page.

2. Update state and pagination logic

Track [itemsPerPage \(src/todo/components/main.jsx\)](#) in state and use it to slice the displayed TODOs.

3. Fix the import and usage of [PaginationControls \(src/todo/components/paginationControls.jsx\)](#)

You should import and use [PaginationControls \(src/todo/components/paginationControls.jsx\)](#) as a component, not as a function.

Here are the changes:

```
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, useCallback, useState, useEffect } from "react";
import { PaginationControls } from "../paginationControls"; // Fix
import

export function Main({ todos, dispatch }) {
  const { pathname: route } = useLocation();
  const [currentPage, setCurrentPage] = useState(1);
  const [itemsPerPage, setItemsPerPage] = useState(5); // Add state
  for items per page
  const [lastAddedItem, setLastAddedItem] = useState(null);

  // Filter and sort todos
  const filteredTodos = useMemo(
    () =>
      todos
        .filter((todo) => {
          if (route === "/active") return !todo.completed;
          if (route === "/completed") return todo.completed;
          return todo;
        })
        .slice()
        .sort((a, b) => a.title.localeCompare(b.title)),
    [todos, route]
  );

  // Calculate paginated todos
  const totalPages = Math.ceil(filteredTodos.length / itemsPerPage);
```



```

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

// Reset to first page if itemsPerPage or filter changes
useEffect(() => {
  setCurrentPage(1);
}, [itemsPerPage, route]);

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

useEffect(() => {
  if (todos.length > 0) {
    const lastItem = todos[todos.length - 1];
    setLastAddedItem(lastItem);
  }
}, [todos, lastAddedItem]);

// Handler for items per page selector
const handleItemsPerPageChange = (e) => {
  setItemsPerPage(Number(e.target.value));
};

return (
  <main className="main" data-testid="main">
    <div style={{ display: "flex", justifyContent: "flex-end",
marginBottom: 10 }}>
      <label htmlFor="items-per-page" style={{ marginRight: 8
}}>Show per page:</label>
      <select id="items-per-page" value={itemsPerPage}
onChange={handleItemsPerPageChange}>
        <option value={1}>1</option>
        <option value={3}>3</option>
        <option value={5}>5</option>
      </select>
    </div>
    {filteredTodos.length > 0 ? (
      <>
        <div className="toggle-all-container">
          <input
            className="toggle-all"
            type="checkbox"
            id="toggle-all"
            data-testid="toggle-all"
            checked={filteredTodos.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 TODOs
here yet</span>
  </div>
)

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

```

This will allow users to select how many TODOs to display per page and navigate between pages.

> Response rendering

Participant

Add a drop down in the main to let user choose how many todos show per page, update state and pagination logic. Track these items in state and use it to slice the displayed todos.

OK.

How many todos? So it again suggests a big chunk of code.

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

Participant

So when I look at the main file, I do see there's.

Issue here.

Is never read.

That was from previous change.

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

Participant

SO one thing that is not very convenient is because it's give a big chunk of the code.

So I actually want to know where exact line.

Where that needs to be changed.

Add a selector for

item per page. Add a dropdown.

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

Participant

Oh, my lazy goal again will be just copy paste everything.

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

Participant

Let me see what will happen.

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

00:34:41
- 00:34:49

00:34:48

00:34:54

00:35:01

00:35:07

00:35:13

00:35:16

00:35:18

00:35:23

00:35:30

00:35:33

00:35:40

00:35:41
- 00:35:44

00:35:52

00:35:58

00:35:58

00:36:05

00:36:07

00:36:09

00:36:21

00:36:29

00:36:34

00:36:36

00:36:38
- 00:36:41

00:36:42

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

00:37:23

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

00:37:42

Participant

Insert per page.

00:37:52

OK.

00:37:55

Items.

00:37:58

Per.

00:38:00

Page.

00:38:06

Fix the import and usage, you should import and use as a component,

00:38:10

not as a function.

00:38:12

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

00:38:18

Participant

As a component, not a function.

00:38:23

Import.

00:38:48

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

how to fix Fix the import and usage of paginationControls?

> **Response**

To fix the import and usage of [PaginationControls](#)

([src/todo/components/paginationControls.jsx](#)), make sure you:

1. **Import it with the correct name and capitalization** in your file (e.g., main.jsx):

```
import { PaginationControls } from "../paginationControls";
```

2. **Use it as a component** (with `<PaginationControls ... />`) in your JSX, not as a function call.

Example usage in main.jsx:

```
File: src/todo/components/main.jsx
// ...existing code...
<PaginationControls
  currentPage={currentPage}
  setCurrentPage={setCurrentPage}
  itemsPerPage={itemsPerPage}
  totalItems={filteredTodos.length}
/>
// ...existing code...
```

Summary:

- Import with curly braces and correct capitalization: `{ PaginationControls }` - Use as a JSX component: `<PaginationControls ... />`

00:38:50
- 00:38:53

> **Response rendering**

00:39:07

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

00:39:16

Participant

OK, now there's a dropdown.

00:39:19

So the first one was finished and also implement navigation controls to allow users to move between pages todo.

00:39:25

Allow users to move between pages todo.

00:39:30

The application should have next.

00:39:36

And previous buttons to navigate between pages.

00:39:41

Chat (window: 0)**> Agent**

github.copilot.workspace

> Context

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

> Message

@workspace Please now suggest change for this feature: The application should have "Next" and "Previous" buttons to navigate between pages. The "Next" button should be disabled and grey when the user is on the last page, and the "Previous" button should be disabled and grey when the user is on the first page. The page that the user is currently on should also be indicated.

> Response

To fully implement this feature, update [PaginationControls](#) ([src/todo/components/paginationControls.jsx](#)) to:

- Correctly calculate the total number of pages.
- Disable and style the "Previous" and "Next" buttons as required.
- Show the current page indicator.

Here is the improved component:

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

export function PaginationControls({ currentPage, setCurrentPage,
itemsPerPage, totalItems }) {
  const totalPages = Math.max(1, Math.ceil(totalItems /
itemsPerPage));

  const handlePrev = () => {
    if (currentPage > 1) setCurrentPage(currentPage - 1);
  };

  const handleNext = () => {
    if (currentPage < totalPages) setCurrentPage(currentPage + 1);
  };

  return (
    <div className="pagination-controls">
      <button
        className={classNames("pagination-button", { inactive:
currentPage === 1 })}
        onClick={handlePrev}
        disabled={currentPage === 1}
      >
        Previous
      </button>
      <span style={{ minWidth: 80, textAlign: "center" }}>
        Page {currentPage} of {totalPages}
      </span>
      <button
        className={classNames("pagination-button", { inactive:
currentPage === totalPages })}
        onClick={handleNext}
        disabled={currentPage === totalPages}
      >
        Next
      </button>
    </div>
  );
}
```

This will ensure the navigation buttons are disabled and greyed out appropriately, and the current page is displayed. The CSS in [src/todo/app.css](#) already supports the `.inactive` class for greyed-out buttons.

> Response rendering

Participant

Disable and style the previous and next buttons as required, show the current page indicator.

So this is the change that I need to do in.

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

Participant

Pagination controls.

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

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

Participant

Uhm, should be indicated.

Yes, page one of two.

Attempt to fix.

[unintelligible]

Not correctly, OK.

Add 10 or more todos, give them sequential numbers as names to easily track them.

One.

Oh.

13.

17, 18, 19, 20.

20, 21.

22.

Alright.

Confirm that users can select options 1, 3, 5 todo at the same time.

Yes.

And confirm that select different page size options.

Confirm that only the chosen number are displayed at the time.

Yes, confirm that the current page is indicated.

Correctly, yes.

Uh.

Click the next button.

Yes, click the previous one.

Yes, navigate to the first page and confirm that the previous button is disabled and gray, yes.

Is disabled and.

Navigate to the the last one, yes.

Confirm that the todo count on each page matches the selected page size.

Yes.

Confirm that the todo count on each page matches.

Sorry, this already confirmed.

Confirmed that the last page contains at least one and at most the selected number of todos.

Yes, that's true.

Confirm that all the todos added are present.

Yes.

Yeah.

Researcher

Perfect. Thanks.

00:43:44

Participant

I have to.

00:43:46
- 00:43:53

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

00:43:47

Researcher

You're right on time even.

00:43:48

Participant

Yeah.

00:43:50

Oh, thanks to Copilot.

00:43:53

To be honest, I have never really programmed in

00:43:57

JavaScript though well, in the past had some small experience in modifying JavaScript but never really used it.

00:44:03

To develop applications.

00:44:08

And and this is also very short amount of time, right?

00:44:12

And I need to perform some task.

00:44:15

I have to say I fully rely on GitHub Copilot in this case.

00:44:17

00:44:21

Researcher

Yeah, no worries. It's interesting to see.

00:45:34

Participant

How would you rate Copilot's response with regards to length?

00:45:42

I think length is neutral.

00:45:47

It's not too long.

00:45:50

Detail.

00:46:12

Yeah. So I think in this case the detail level is fine.

00:46:16

00:46:17

It's actually quite good because I need to finish the task within a certain amount of time, but I can imagine in the development process you might want to really understand the code base.

00:46:22

00:46:25

Then maybe it is a bit too short.

00:46:29

00:46:31

I would expect to have longer description.

00:46:36

Researcher

OK.

00:46:37

Participant

But so then I would say, do you want me to imagine that I'm in the real?

00:46:42

00:46:43

Software development process then I would say it is poor.

00:46:49

Researcher

I'd say rate. It towards what you need at Copilot to do in this scenario.

00:46:54

00:46:56

Participant

In this scenario, finishing certain tasks within certain amount of time. OK then I will say it's pretty good.

00:47:00

00:47:00

Researcher

Yeah.

00:47:07

Participant

Use of technical terms.

00:47:12

I think it's quite good.

00:47:13

Correctness was excellent, so I did not really need to modify anything.

00:47:19

00:47:20

I basically copy paste.

00:47:24

It's just about navigation. Understanding where to add the changes.

00:47:31

Following your instructions, pretty good.

00:47:35

Excellent. Actually. Tone.

00:47:37

Formal, confident, humorous. I did not pay attention to that.

00:47:42

But it's, then.

00:47:44

I think it is good.

00:47:46

Use of formatting, bullet point.

00:47:49 Yeah, that's pretty good.
00:47:51 It's excellent.
00:47:56 Please indicate to what extent you agree with the following statements regarding
00:48:01 your interaction with Copilot. Using Copilot in my job or study will enable me
00:48:07 to accomplish your task more quickly.
00:48:11 Extremely agree.
00:48:14 Using Copilot will improve my job of study performance.
00:48:24 Slightly agree.
00:48:28 Using Copilot in my job would increase my productivity.
00:48:33 Extremely agree.
00:48:36 Will enhance my effectiveness in my job or study.
00:48:43 Quite agree.
00:48:44 Make it easier to do my job or study.
00:48:53 Learning to operate Copilot would be easy for me.
00:48:59 I do not understand.
00:49:01 This one learning to operate Copilot will be easy for me.
00:49:07 You mean this process was easy for me to navigate,
00:49:11 to learn or it will be easier for me to achieve something? What does it ask?
00:49:17 **Researcher**
It's a learning to operate.
00:49:19 So for example, how the commands work? Learning how to?
00:49:24 What to expect from it and how to get the results you want?
00:49:30 **Participant**
Ah, OK. But I think so. I have never learned how to use Copilot,
00:49:35 but it was very easy to use already, so I would say I do not agree.
00:49:41 I think anyone with software engineering background can easily figure out how to
00:49:47 use that, and the command is not.
00:49:50 So complex.
00:49:52 **Researcher**
Sorry then I think.
00:49:53 **Participant**
Is that the intention of the, sorry?
00:49:55 **Researcher**
Yeah, yeah. But it says it's it's, it's easy.
00:49:59 So then do you disagree?
00:50:00 **Participant**
It is easy.
00:50:01 Yeah, it is very easy. Yeah.
00:50:03 **Researcher**
OK.
00:50:04 **Participant**
I thought you mean that we need some training to learn.
00:50:09 OK, then I extremely agree. I will find it easy to get Copilot.
00:50:14 To do what I wanted to do.
00:50:19 I think so.
00:50:22 Hmm.
00:50:27 I think it's also, yeah, extremely agree.
00:50:30 My interaction with Copilot will be clear and understandable.
00:50:38 My interaction with Copilot will be clear and understandable.
00:50:44 To whom?
00:50:48 Clear and understandable to GitHub, Copilot, or clear and understandable to me?
00:50:53 **Researcher**
To you.
00:50:55 **Participant**
To me.
00:51:00 I think yes, quite agree.

00:51:02 I will find Copilot to be flexible to interact with.
00:51:08 Flexible.
00:51:11 Let me look at it again.
00:51:16 Flexible.
00:51:21 It would be better, but I think you disabled the features on
00:51:24 purpose, so there's no inline.
00:51:28 Editor.
00:51:31 So you are asking whether it is what interaction with the chat box is flexible.
00:51:39 **Researcher**
Yes.
00:51:43 **Participant**
I think I agree. It will be easy for me to become skillful at using Copilot.
00:51:50 I think so.
00:51:52 I will find Copilot easy to use.
00:51:57 Quite agree.
00:52:01 Yes.
00:52:04 How mentally demanding was the session?
00:52:11 Hmm.
00:52:17 I think it was quite low.
00:52:19 How physically demanding was the session?
00:52:23 Was also very low.
00:52:28 Uh.
00:52:31 Roughly the same. How hurried or rushed was the pace of the session.
00:52:38 I think was I think it is in a good tempo so which means.
00:52:46 Zero is the right answer.
00:52:49 Like I want to say it's not too fast or not too slow.
00:52:57 So then I will select this.
00:53:01 How successful were you in the accomplishmening
00:53:04 what you were asked to do, I think it was successful.
00:53:12 How hard?
00:53:15 One question. Since I finish all the tasks and verify
00:53:18 that.
00:53:18 Is this about the fact or my perception how successful?
00:53:23 **Researcher**
About your perception of success.
00:53:26 **Participant**
OK.
00:53:27 So then I'd say 7.
00:53:27 **Researcher**
Yeah.
00:53:31 **Participant**
Because I do finish the task, but I don't think I understand all the
00:53:35 changes I made.
00:53:38 So I will say 7.
00:53:41 How hard did you have to work to accomplish your level of performance?
00:53:50 It was quite easy.
00:53:51 Not a lot of.
00:53:58 Brain energy consumption.
00:54:02 How insecure,
00:54:03 discouraged, irritated, stressed, and annoyed were you?
00:54:11 Very low.
00:54:33 How did you find the task?
00:54:41 Very easy.
00:54:48 Save draft.
00:54:50 Which one was that?
00:55:06 This is moderate.

00:55:08 How realistic did you find the tasks in reflecting real life programming
00:55:13 situations?
00:55:15 Umm.
00:55:20 I will say the last one is.
00:55:24 Quite realistic. You usually got.
00:55:30 Many changes relate to one feature, and then you have to decompose it a bit.
00:55:42 Hmm.
00:55:45 I think this one very realistic.
00:55:51 Save drafts.
00:55:54 Refresh.
00:56:01 Like this.
00:56:04 OK, I finished.
00:56:05 **Researcher**
Perfect. Thanks.
00:56:08 **Participant**
OK.
00:56:08 **Researcher**
All right. So then I have a few questions for you
00:56:12 about what you thought.
00:56:12 **Participant**
Mm hmm.
00:56:15 **Researcher**
First of all, did you manage to find your way like
00:56:19 navigate through the code base.
00:56:21 And how did Copilot influence this?
00:56:26 **Participant**
I think what I found very useful is that it can help me locate.
00:56:33 The code nippets that I need to work on.
00:56:37 This one is extremely helpful, especially when you don't know anything
00:56:42 about the code base and you just got a new one,
00:56:45 or you are dealing with a very big code base.
00:56:48 I can imagine in a real life situation you might have.
00:56:53 Even a million lines of code in your code base and you'll need to locate the part
00:56:59 that is related to the. Let's say the issues or the features you
00:57:03 got.
00:57:06 **Researcher**
All right.
00:57:08 And for your strategy of approaching the task.
00:57:13 **Participant**
Mm hmm.
00:57:15 **Researcher**
How would you have solved these tasks without Copilot?
00:57:18 And how did using Copilot influence your strategy?
00:57:23 **Participant**
It definitely influenced my strategy.
00:57:25 So if I don't have a Copilot, I will really have to look into those
00:57:31 main files and then try to understand the logic and try to understand what.
00:57:41 What file?
00:57:41 Like implement the feature that I that I need to change and how it couples with the
00:57:48 rest of the code.
00:57:52 But with GitHub Copilot you see that.
00:57:54 I basically prompt it and then I copy paste the the suggestion and my strategy
00:58:01 is that I first commented out the original code and then just try the new
00:58:07 code, see what will happen and.
00:58:11 So in this case I was lucky that the result was the suggested change was
00:58:15 actually quite quite a good already.

00:58:18 So I don't need to make a lot of changes.
00:58:22 And so then the strategy a bit changed.
00:58:26 Right. So instead of analyzing the codebase,
00:58:30 you start trying things with the suggested change and then see what will
00:58:36 happen and then I can imagine if the result is not perfect.
00:58:41 Then I have to understand the suggestion. So you see this strategy change a bit.
00:58:48 **Researcher**
OK.
00:58:48 That makes sense.
00:58:50 So do you feel like you understood the code now? And how did Copilot impact that?
00:58:56 **Participant**
So it helps roughly because it also explained with several steps. OK,
00:59:01 you need to first do this and 2nd this and 3rd this.
00:59:05 So I wouldn't say I understand that part of code perfectly,
00:59:10 but I know roughly the logic.
00:59:16 And the structure of the code.
00:59:19 **Researcher**
OK. And were you happy with your
00:59:22 understanding?
00:59:25 **Participant**
Uhm.
00:59:30 I think.
00:59:33 It will not completely replace.
00:59:36 My own analysis if I really want to.
00:59:41 You know, continue developing this code base.
00:59:43 I need to know more, so I definitely still need to spend time
00:59:48 in comprehending the code, but I think GitHub Copilot gives a like good
00:59:54 starting point and pointing out important features. Sorry.
00:59:59 Important code snippets for those features.
01:00:03 **Researcher**
Right. So then does the the setting of this
01:00:06 being a.
01:00:08 Like a short session influence how you would use Copilot.
01:00:15 **Participant**
Yes, I think a slightly, I think I it really depends on your
01:00:21 positions, right.
01:00:23 So I also try to implement some.
01:00:27 App by myself using Flutter and I know nothing about Flutter,
01:00:32 but my goal is not to, let's say develop.
01:00:37 My implementation skills, but really to do some prototype to get
01:00:41 the app working.
01:00:42 In that case, I also had a short amount of time and
01:00:45 want to achieve something right.
01:00:47 So I use the same strategy.
01:00:48 I prompt and my copy paste and then see what happens and see what need to be
01:00:52 changed.
01:00:53 But I can imagine if you want to develop your implementation skills,
01:00:58 want to be professional in software engineering and you have a long term
01:01:04 development plan and want to maintain the code base.
01:01:08 Then you definitely need to.
01:01:14 Need to know the code base a bit better, so I will say it really depends on your
01:01:18 goal.
01:01:19 **Researcher**
All right and.
01:01:23 Uh.
01:01:26 Sorry, I forgot what I was going to ask.

01:01:29 **Participant**
No problem.

01:01:31 **Researcher**
Do you feel like you changed your interaction with Copilot over time?

01:01:37 **Participant**
My interaction.

01:01:44 Can I get back to my chat box?

01:02:03 I do not think there is a big change, but since the last assignment the last
01:02:10 task is a bit more complicated than the other two.

01:02:14 So for the first two tasks I basically copy paste the text into the chat and you
01:02:21 know modified it, but the third one for the third one I
01:02:27 tried to split.

01:02:29 The entire task into two or three smaller tasks.

01:02:33 So I don't know if that is, you know, that is considered to be a change of
01:02:39 interaction.

01:02:41 **Researcher**
Yeah, that makes sense.

01:02:43 I remember for the second practice task this was changing the placeholder of the
01:02:47 **Participant**
Mm hmm.

01:02:50 **Researcher**
input text.

01:02:50 You didn't use Copilot.

01:02:52 Was there a reason for this?

01:02:54 **Participant**
Oh yeah.

01:02:55 So for that one, I was thinking well, I can just search this string,
01:03:00 then I will get it even more quickly perhaps.

01:03:05 And also it's my maybe my mindset that I don't want to use AI for everything.
01:03:13 If I could use more deterministic approaches I would just use that.

01:03:21 **Researcher**
OK.

01:03:21 That makes sense.

01:03:23 I saw that sometimes when you put the put the task in Copilot you checked the
01:03:30 answer first before testing it and other times you tested it first.

01:03:38 Do you have a reason for for this?

01:03:40 **Participant**
Yeah. So if I understand it, for example, there was a task for sorting. This is a
01:03:47 very simple and small feature and the suggested change is not big.

01:03:52 Then I just got manually checked that.

01:03:56 But I think for the last one, the suggestion was like a really big
01:04:00 chunk of the code and also because it's in the chatbot, it's not inline change.

01:04:05 I cannot see exactly which line has been changed.

01:04:08 So if I want to go through all the changes.

01:04:10 Then I have to compare the before change after change side by side.

01:04:17 And and I don't think I have time to do that.

01:04:20 So that's why I just test it first.

01:04:23 **Researcher**
Alright, if you have more time or for example in a
01:04:27 real life scenario like the app you talked about,
01:04:30 would you ideally check it first?

01:04:35 **Participant**
Depends on application, so if it is just so convenient,
01:04:39 I only need to refresh then I will probably test it first and then see what
01:04:44 happened.

01:04:44 But imagine that if you develop.

01:04:48 Let's say embedded systems, cyber physical system, not web application then.
01:04:55 Running.
01:04:57 The test.
01:04:59 Or launching the application might take longer time then you you want to make
01:05:04 sure that.
01:05:06 I understand it and I validate the result before doing any testing.
01:05:12 **Researcher**
Yeah, that makes sense.
01:05:14 **Participant**
Yeah.
01:05:17 **Researcher**
Let me see.
01:05:18 So I saw you were commenting the code.
01:05:21 Why were you doing this?
01:05:23 **Participant**
I common thing because I want to keep the original.
01:05:28 Code and in case the result is really bad then I want to compare these to figure
01:05:33 out what goes wrong, what needs to be changed.
01:05:38 **Researcher**
So I see you said you primarily use ChatGPT.
01:05:42 Do you also have the same same strategy when you use that?
01:05:49 **Participant**
Yeah, same.
01:05:50 So I never just directly remove the original code.
01:05:57 **Researcher**
So you can just go.
01:05:57 **Participant**
Also, because it is also what I learn from
01:06:00 practice sometimes, especially when I use GPT 3.5 in the past.
01:06:05 It can suggest something, just not executable.
01:06:10 So that's why I developed the habit that I need to backup my previous version.
01:06:16 **Researcher**
Right. And this is maybe you don't really have a
01:06:20 question to this answer. Do you find that a nice way of doing it
01:06:26 or do you think?
01:06:27 These these chatbots could improve. These tools could improve.
01:06:32 To make it easier for you.
01:06:36 **Participant**
For the chatbot, so we only talk about chatbot and not
01:06:41 inline features, right?
01:06:42 **Researcher**
Yeah, yeah.
01:06:46 **Participant**
What I observed.
01:06:47 So if I compare this with ChatGPT which also.
01:06:52 Well, now you there's a way to see the code,
01:06:56 but I think a long time ago they did not have the way to like another window to
01:07:03 show the code, right, and back then I remember with ChatGPT they it really explains
01:07:09 which part of the code need to.
01:07:12 Be changed rather than give you like the entire.
01:07:16 File.
01:07:18 Which helps.
01:07:19 So what I will actually prefer is.
01:07:22 You give me the entire file after change after applying all the changes you
01:07:27 suggested, but also give me the fragments of changes.
01:07:32 **Researcher**

01:07:33 Right. So the exact lines.
Participant
And some explanation, yeah.

01:07:37 **Researcher**
OK.

01:07:37 So I I think in the post study survey you said that the detail now was fine,
01:07:43 but then in real life you would like to have longer answers.
01:07:48 Does this mean you would like these line by line explanations?
01:07:53 **Participant**
Yes, exactly. To help me understand what I'm changing
01:07:57 and which lines I'm changing, rather than just copy paste the entire
01:08:01 file and but giving me the entire file is good.
01:08:05 Like once I understand all the changes, then I will actually copy paste the
01:08:10 entire one to update it.
01:08:12 In one go, rather than go through all that say 10
01:08:15 changes and changes 1 by 1.
01:08:17 **Researcher**
Right. So do you like to understand all the
01:08:21 changes before you apply them?
01:08:25 **Participant**
Umm.

01:08:29 Again, depends on the goal depends on the.
01:08:34 That's the the debugging cycle.
01:08:39 So if it is really like now, I can really quickly.
01:08:45 Refresh the application then I prefer to just apply it and see what happens,
01:08:50 and then later I might take time to really understand what has been changed.
01:08:58 But it is if it is a very critical system, then I definitely will prefer to
01:09:03 understand everything before trying applying.
01:09:07 **Researcher**
That makes sense.
01:09:09 All right. Those were my questions.