

Participant P16

00:02:14

Participant

Shall I start by reading to find out what the purpose of the activities is?

00:02:21

To understand the context.

00:04:19

Okay, alright? Well, then I'll start executing the project

00:04:23

that was proposed to see what's inside it.

00:04:29

So I'll just run it initially to

00:04:34

see what was done.

00:04:39

What I need to do is change the color of the button

00:04:51

> **Viewing file** (src/index.js)

00:04:54

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

00:04:57

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

00:05:07

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

00:05:12

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

- 00:05:22

00:06:31

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

00:07:02

Chat (window: 0) (**canceled**)

> **Agent**

github.copilot.workspace

> **Context**

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

> **Message**

@workspace main.jsx Practical Task A - Change the color When there are no tasks, a red box appears with the message "Nenhum afazer aqui ainda". Change the color of this box to blue (#007bff). Verification: Update the application. Confirm that the message "Nenhum afazer aqui ainda" appears inside a blue box. Practical Task B - Change the placeholder The placeholder text in the input field currently displays the question "O que precisa ser feito?". Change the text to "Digite um afazer aqui". Verification: Update the application. Confirm that the placeholder now displays "Digite um afazer aqui". Practical Task C - Sort tasks Currently, tasks are displayed in the order they were added. Modify the application so that tasks are sorted alphabetically. Verification: Add the following tasks, in this order: "o", "rápido", "123", "marrom", "raposa". Confirm that the tasks are displayed in the following order (alphabetically): "123", "marrom", "o", "rápido", "raposa".

> **Response**

00:07:23

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

copilot-interaction @workspace main.jsx Practical Task A - Change the color When there are no tasks, a red box appears with the message "Nenhum afazer aqui ainda". Change the color of this box to blue (#007bff). Verification: Update the application. Confirm that the message "Nenhum afazer aqui ainda" appears inside a blue box. Practical Task B - Change the placeholder The placeholder text in the input field currently displays the question "O que precisa ser feito?". Change the text to "Digite um afazer aqui". Verification: Update the application. Confirm that the placeholder now displays "Digite um afazer aqui". Practical Task C - Sort tasks Currently, tasks are displayed in the order they were added. Modify the application so that tasks are sorted alphabetically. Verification: Add the following tasks, in this order: "o", "rápido", "123", "marrom", "raposa". Confirm that the tasks are displayed in the following order (alphabetically): "123", "marrom", "o", "rápido", "raposa".

> Response

Practical Task A - Change the color

To change the color of the box to blue (#007bff), update the style of the `.todo-list-empty-container` class. If the style is in a CSS file, make the following change:

```
File: src/todo/styles/main.css
.todo-list-empty-container {
  background-color: #007bff; /* Changed to blue */
  color: white; /* Adjust for contrast */
  padding: 10px;
  border-radius: 5px;
  text-align: center;
}
```

If the style is inline, you can add the `style` attribute directly in the JSX:

```
<div className="todo-list-empty-container" style={{ backgroundColor:
"#007bff", color: "white", padding: "10px", borderRadius: "5px",
textAlign: "center" }}>
  <span className="todo-list-empty-message">Nenhum afazer aqui
ainda</span>
</div>
```

Practical Task B - Change the placeholder

Locate the component that contains the input field (probably an [input](#) ([node_modules/@types/react/index.d.ts](#)) in another file). Update the placeholder attribute to "Digite um afazer aqui". Example:

```
<input
  type="text"
  className="new-todo"
  placeholder="Digite um afazer aqui" /* Changed */
  autoFocus
/>
```

Practical Task C - Sort tasks

To sort tasks alphabetically, modify the [useMemo](#) ([node_modules/@types/react/index.d.ts](#)) that calculates [visibleTodos](#) ([src/todo/components/main.jsx](#)). Add a step to sort the items:

```
const visibleTodos = useMemo(
  () =>
    todos
      .filter((todo) => {
        if (route === "/active") return !todo.completed;
        if (route === "/completed") return todo.completed;
        return todo;
      })
      .sort((a, b) => a.text.localeCompare(b.text)), //
  [todos, route]
);
```

Alphabetical sorting

Verification

1. **Task A:** Update the application and verify that the message "Nenhum afazer aqui ainda" appears inside a blue box.
2. **Task B:** Verify that the input field's placeholder displays "Digite um afazer aqui".
3. **Task C:** Add the tasks "o", "rápido", "123", "marrom", "raposa" and confirm that they appear in the order: "123", "marrom", "o", "rápido", "raposa".

00:07:54

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

00:08:00

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

00:08:06

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

00:08:13

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

00:08:29
- 00:08:30

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

00:08:45

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

00:09:08

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

I would like all of them to be ordered this way. Currently, the tasks are displayed in the order they were added. Modify the application so that the tasks are sorted alphabetically. Verification: Add the following tasks, in this order: "o", "rápido", "123", "marrom", "raposa". Confirm that the tasks are displayed in the following order (alphabetically): "123", "marrom", "o", "rápido", "raposa".

> **Response**

To sort the tasks alphabetically, you can modify the [useMemo](#) ([node_modules/@types/react/index.d.ts](#)) that calculates [visibleTodos](#) ([src/todo/components/main.jsx](#)) to include a sorting step. Here is the necessary change:

```
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;
      })
      .sort((a, b) => a.text.localeCompare(b.text)), //
  [todos, route]
);
```

Alphabetical sorting

Explanation

1. [filter \(src/todo/components/main.jsx\)](#): Filters the tasks based on the route (/active, /completed, or all).
2. **sort**: Sorts the tasks alphabetically using `localeCompare` on the text field of each [todo \(src/todo/components/main.jsx\)](#).

Verification

1. Add the tasks in the following order: "o", "rápido", "123", "marrom", "raposa".
2. Confirm that they appear in the order: "123", "marrom", "o", "rápido", "raposa".

00:09:11

Participant

My idea is to ask the AI to do the sorting task

00:09:15

I believe it's a simple task,

00:09:19

I want to try to speed up the small

00:09:23

parts first

00:09:25

For it to do that. Meanwhile, I'll be checking the other tasks.

00:09:29

It turned out I did it a bit clumsily,

00:09:37

And now there's just this part left.

00:09:41

Researcher

Sort

00:09:50

- 00:10:00

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

00:10:08

Participant

It would be nice if the copilot created a tutorial with all the codes it showed.

00:10:45

Researcher

OK, cool.

00:10:54

Participant

for example. I even know how to make a filter

00:10:58

but I don't.

00:10:59

remember this function.

00:11:03

I never remembered it, but usually I know some codes

00:11:19

Now I'm using one at my work, Windsurf

00:11:23

but I use them in people mode, so they end up being more autonomous.

00:11:25

Researcher

OK

00:11:29

Participant

Yeah, and it's been working well for me. I think we've finished the tasks,

00:12:41

Chat (window: 0)

> Agent

github.copilot.default

> Context

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

> Message

main.jsx:26 Uncaught TypeError: Cannot read properties of undefined (reading 'localeCompare') at main.jsx:26:36 at Array.sort (<anonymous>) at main.jsx:26:14 at Main (main.jsx:15:33)

> Response

The error occurs because the property [text\(src/todo/components/main.jsx\)](#) of some object [todo\(src/todo/components/main.jsx\)](#) is undefined. To fix this, you can add a check in the [sort\(src/todo/components/main.jsx\)](#) method to ensure that both values exist before comparing them. Here is the fix:

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;  
      })  
  )
```

```

        .sort((a, b) => (a.text || "").localeCompare(b.text ||
"")), // Checks if 'text' exists
        [todos, route]
    );

```

Solution Explanation

1. `a.text || ""`: Ensures that if `a.text (src/todo/components/main.jsx)` is undefined, it will be treated as an empty string (`""`).
2. `b.text || ""`: Does the same for `b.text (src/todo/components/main.jsx)`.
3. `localeCompare (src/todo/components/main.jsx)`: Continues to work correctly, even if some value is an empty string.

Verification

1. Update the code and test again.
2. Ensure that the todos are sorted alphabetically without errors.

00:12:50

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

00:14:15

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

00:14:17
- 00:14:26

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

00:14:24

Researcher

What are you trying to do now.

00:14:28

Participant

I'm trying to fix this filter, it's not right

00:14:31

There's some error at the moment it tries to render the task that

00:14:38

apparently it's trying to render an attribute that doesn't exist by

00:14:39

Researcher

ok

00:14:55

Participant

Well, now it's working. Yeah, apparently that was the problem indeed.

00:15:04

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

00:15:33

Participant

done.

00:15:34

If it were an agent, I would ask It to do exactly what

00:15:37

I want, but since that's not the case.

00:15:40

I. When I want to do something, I go and already ask for analyzing or

00:15:57

I prefer to separate. Now let's just test, then.

00:16:30

It's correct.

00:16:37

Researcher

I'm going to send you the other tasks now

00:17:04

Participant

I'm going to have to start a project from scratch.

00:17:07

Researcher

It's not all in the same project

00:17:35

Participant

modify application so that tasks remain even after loading.

00:17:39

Make sure.

00:17:44

An attempt to solve the problem was made,

00:17:46

but at the moment it is not working correctly.

00:18:07

First thing I'm going to do is see if it's recording somewhere correctly and

00:18:12

probably from what I saw, it was using the local store,

00:18:16

so it is being saved. The problem is when I update,

00:18:20

it keeps, just doesn't retrieve and when I create another one it.

00:18:24

Should replace, probably so what is it

00:18:27

doing?

00:18:27

He's just not retrieving then for this.

00:18:33

It's load state

00:18:47

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

- 00:18:47

00:19:01

Participant

Well, I'm still looking for what might be wrong.

00:19:10

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

00:19:11

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

00:19:11

Participant

I'm seeing that this one is not being used. Probably a future problem.

00:19:15

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

- 00:19:16

00:19:17

Participant

I'll see if that's what's blocking me and sometimes it can happen from one process to another and not do

00:19:22

Anything right.

00:19:25

I'll refresh here. This wasn't the problem

00:19:28

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

00:19:36

Participant

In this case.

00:19:40

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

00:19:42

Participant

He has an action that is to load.

00:19:47

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

- 00:20:14

00:20:07

Participant

I'm also a bit.

00:20:14

confused with the load state. This has to be used somewhere.

00:20:28

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

00:20:50

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

I need the "todos" localStorage to be loaded and feed the variables.

> **Response**

To load the data from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) and feed the variables in the [Main component \(src/todo/components/main.jsx\)](#), you can use [useEffect \(node_modules/@types/react/index.d.ts\)](#) to initialize the [todos](#)

([src/todo/components/main.jsx](#)) from the state saved in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)). Here's how you can do it:

Change in [Main](#) ([src/todo/components/main.jsx](#)) to load data from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#))

Add the following code to the [Main component](#) ([src/todo/components/main.jsx](#)):

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

export function Main({ todos, dispatch }) {
  const { pathname: route } = useLocation();
  const [currentPage, setCurrentPage] = useState(1);
  const [lastAddedItem, setLastAddedItem] = useState(null);

  const itemsPerPage = 5;

  // Load data from localStorage when the component mounts
  useEffect(() => {
    const savedTodos = localStorage.getItem("todos");
    if (savedTodos) {
      const parsedTodos = JSON.parse(savedTodos);
      dispatch({ type: "LOAD_STATE", payload: parsedTodos });
    }
  }, [dispatch]);

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

  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"
            />
          </div>
        </>
      ) : null}
    </main>
  );
}
```

```

        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>
  )}
</main>
);
}

```

Change in [todoReducer \(src/todo/reducer.js\)](#) to support [LOAD_STATE \(src/todo/constants.js\)](#)

Ensure that the [todoReducer \(src/todo/reducer.js\)](#) already supports the [LOAD_STATE action \(src/todo/constants.js\)](#), as shown in the file [src/todo/reducer.js](#):

```

case LOAD_STATE:
  newState = action.payload; // Load the state from the payload
  break;

```

Explanation

1. [useEffect \(node_modules/@types/react/index.d.ts\)](#) in [Main \(src/todo/components/main.jsx\)](#): Loads data from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) when the component mounts and dispatches the [LOAD_STATE action \(src/todo/constants.js\)](#) to update the global state.
2. [LOAD_STATE \(src/todo/constants.js\)](#) in [todoReducer \(src/todo/reducer.js\)](#): Updates the global state with the data loaded from [localStorage \(node_modules/@types/node/globals.d.ts\)](#).

Verification

1. Save some tasks in [localStorage \(node_modules/@types/node/globals.d.ts\)](#) (or use the application to create some).
2. Reload the page and check if the tasks are loaded correctly.

00:20:54

Participant

Yeah, I think I did something wrong here

00:20:56

I'm asking him to do it as if,

00:20:58

an agent, but I think he will be able to

00:21:02

understand anyway and give me guidance on what might be wrong.

00:21:06

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

00:21:16
- 00:21:18

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

00:21:25
- 00:21:36

00:21:32

Participant

I still need to familiarize myself with this reduce.

00:21:37

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

00:21:39

Participant

I need to see how I can trigger something here.

00:21:44

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

00:21:44

Participant

00:21:48 Maybe there's already something in the code.
00:21:48 > **Viewing file** (src/todo/components/input.jsx)
00:21:48 **Participant**
I'll look in the input that's in the Reader
00:21:51 > **Viewing file** (src/todo/components/header.jsx)
00:21:58 > **Viewing file** (src/todo/components/main.jsx)
00:21:59 > **Viewing file** (src/todo/app.jsx)
00:22:02 > **Editing file** (src/todo/app.jsx)
- 00:22:19
00:22:19 **Researcher**
ok.
00:22:22 > **Viewing file** (src/todo/components/main.jsx)
00:22:24 > **Viewing file** (src/todo/app.jsx)
00:22:29 > **Editing file** (src/todo/app.jsx)
- 00:22:43
00:22:44 > **Viewing file** (src/todo/reducer.js)
00:22:47 > **Viewing file** (src/todo/app.jsx)
00:22:48 > **Editing file** (src/todo/app.jsx)
- 00:22:53
00:22:54 > **Viewing file** (src/todo/reducer.js)
00:22:56 > **Viewing file** (src/todo/app.jsx)
00:23:00 > **Editing file** (src/todo/app.jsx)
- 00:24:11
00:23:27 **Participant**
For now, just solving some import problems
00:23:30 routine stuff for me. Usually this type of work too.
00:23:33 **Researcher**
ok.
00:23:35 **Participant**
I like to be using AI, because it already analyzes all this
00:23:40 So you don't need to have those eagle eyes at every moment.
00:23:45 Making us a bit spoiled.
00:23:55 this load, I won't trigger anything
00:24:12 > **Viewing file** (src/todo/reducer.js)
00:24:19 > **Viewing file** (src/todo/app.jsx)
00:24:20 **Participant**
So, I would probably be asking him
00:24:22 > **Editing file** (src/todo/app.jsx)
- 00:24:30
00:24:25 **Participant**
to do it if he were in agent mode, but in text mode
00:24:29 I only ask when there's really something I can't do without it,
00:24:35 because, apparently the Time to
00:24:39 do and the.
00:24:39 Time to ask is the same. At least for me.
00:24:43 who already has experience with code for a while.
00:24:46 I sometimes prefer to do it depending on the activity, than to ask,
00:24:50 but I can understand that sometimes, for those who are starting it's much more
00:24:55 quick to ask for help.
00:24:57 Especially if it's a simple activity.
00:25:01 **Researcher**
ok.
00:25:05 **Participant**
I believe it's working now.
00:25:14 **Researcher**
ok.

00:25:18

Participant

Save draft...

00:25:28

modify the application so that the input field saves automatically,

00:25:32

drafts every few seconds and restores when the page is reloaded.

00:25:46

I believe I will start with the hardest one.

00:25:50

Researcher

ok.

00:25:52

Participant

Which would be pagination, because I remember it was already giving

00:25:56

some problems there, so I will remove the comment and try to

00:26:01

extract some error to send to AI. Meanwhile,

00:26:05

I can focus on this activity that I prefer to do manually.

00:26:09

Researcher

OK.

00:26:10

Participant

So the pagination...

00:26:12

I remember I commented on some codes here.

00:26:12

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

00:26:14

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

00:26:16
- 00:26:34

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

00:26:16

Participant

I will return.

00:26:22

Before that, there was already an error in the link

00:26:25

so probably something I can already see too.

00:26:28

The name of this import is usually uppercase,

00:26:31

so it's already wrong here.

00:26:35

So, maybe it's just that.

00:26:37

That was it.

00:26:43

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

00:26:45

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

00:26:46

Participant

Now I need to set the items per page

00:26:52
- 00:26:54

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

00:26:54

Participant

Maybe this already works, maybe this is enough to

00:27:01

work.

00:27:01

Researcher

OK.

00:27:02

Participant

I don't know how many pages there are. I will try to put a lot of stuff here

00:27:07

and give a next preview. It's just not working really.

00:27:11

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

00:27:12

Participant

It's items per page.

00:27:26

I need it to, based on what we did, render only the perfect items.

00:27:34

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

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

```
}).sort((a, b) => a.title.localeCompare(b.title)), [todos, route] ); based on the
currentPage, make it render only the itemsPerPage.
```

> Response

To render only the items corresponding to the current page based on [currentPage](#) ([src/todo/components/main.jsx](#)) and [itemsPerPage](#) ([src/todo/components/main.jsx](#)), you can adjust the [useMemo](#) ([node_modules/@types/react/index.d.ts](#)) to calculate the visible items in a paginated way. Here is the modification:

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

    const startIndex = (currentPage - 1) * itemsPerPage;
    const endIndex = startIndex + itemsPerPage;

    return filteredTodos.slice(startIndex, endIndex); // Returns
only the items of the current page
  },
  [todos, route, currentPage, itemsPerPage]
);
```

Explanation

1. **filteredTodos**: Filters and sorts the todos based on the route and title.
2. **startIndex and endIndex**: Calculates the start and end indices for the current page.
3. **slice**: Returns only the items within the calculated range.

Verification

1. Test the application with more than [itemsPerPage](#) ([src/todo/components/main.jsx](#)) todos.
2. Navigate between pages using the pagination controls and confirm that only the items of the current page are displayed.

00:27:34

Participant

perfect items.

00:27:37

I'm not providing details.

00:27:39

How much would I give to a person Because I

00:27:41

imagine that the AI will already visualize this and understand what it means.

00:27:45

When this is written in a programming language,

00:27:47

which is where it is, then it will already understand what I want.

00:27:59
- 00:28:05

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

00:28:02

Participant

I really think it's something that helps productivity in general.

00:28:11

It will be 5 per page.

00:28:13

And now, here it is with the buttons locked,

00:28:17

soon I'll see why the buttons are locked.

00:28:21

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

00:28:39

Participant

This is it, I think it's something I would already ask

00:28:43

the agent to see, because it's almost mechanical work of

00:28:48

you analyzing these variables here. Although it's simple.

00:28:54 It's more fun when you do it the first times,
00:28:57 then it becomes kind of a mechanical task
00:29:00 Let's see how it will be able to help me here now.
00:29:25 > **Editing file** (src/todo/components/paginationControls.jsx)
- 00:29:26

00:29:40 **Participant**
Apparently it's correct. I think it's not passing
00:29:41 > **Viewing file** (src/todo/components/main.jsx)
00:29:42 **Participant**
correctly.
00:29:43 > **Editing file** (src/todo/components/main.jsx)
- 00:30:33

00:30:45 **Participant**
Doing it this way, it already solves
00:31:11 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:31:26 > **Editing file** (src/todo/components/paginationControls.jsx)
- 00:31:32

00:31:43 > **Viewing file** (src/todo/components/header.jsx)
00:31:47 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:31:50 > **Viewing file** (src/todo/components/main.jsx)
00:31:52 **Participant**
I usually like it too when I'm going through some problem in the
00:31:57 code, just render it somewhere, so I can see what's happening.
00:31:57 > **Editing file** (src/todo/components/main.jsx)
- 00:31:58

00:32:06 **Researcher**
OK.
00:32:19 **Participant**
So, now, the next activity is to save drafts.
00:32:22 I think this was what I wanted to do. At the same time, though.
00:32:28 There was this additional problem, which I wasn't expecting in pagination.
00:32:52 save the draft, type an a does.
00:32:55 Here.
00:32:58 > **Viewing file** (src/todo/components/input.jsx)
00:32:58 **Participant**
I'll go to the input, send the key down
00:33:12 > **Editing file** (src/todo/components/input.jsx)
- 00:33:19

00:33:19 **Participant**
get the input value and save it in local storage.
00:33:24 It's so I can retrieve the information when I type something,
00:33:32 it executes these events.
00:33:44 > **Viewing file** (src/todo/components/input.jsx)
00:33:46 > **Editing file** (src/todo/components/input.jsx)
- 00:34:42

00:34:39 **Participant**
Let's see if it will be saving
00:34:43 correctly.
00:34:53 And when the input loads.
00:34:54 > **Viewing file** (src/todo/components/input.jsx)
00:34:56 > **Editing file** (src/todo/components/input.jsx)
- 00:35:35

00:35:02 **Participant**
Currently it's using the default value, so maybe I have to get that,
00:35:36 > **Viewing file** (src/todo/components/main.jsx)
00:35:40 > **Viewing file** (src/todo/components/header.jsx)
00:35:42 > **Editing file** (src/todo/components/header.jsx)
- 00:35:45

00:35:47 > **Viewing file** (src/todo/components/input.jsx)
00:35:48 - 00:36:03 > **Editing file** (src/todo/components/input.jsx)
00:36:09 > **Viewing file** (src/todo/components/header.jsx)
00:36:10 - 00:37:27 > **Editing file** (src/todo/components/header.jsx)
00:36:23 **Participant**
I could even get it from the local store.
00:36:29 But I have to do this inside user effect,
00:36:32 so I don't have to keep doing this all the time.
00:36:45 But I think I'm being a bit slow here.
00:36:51 I should think a bit more, but it's not the kind of question I
00:36:55 would ask the copilot now, because.
00:37:00 If I were in agent mode.
00:37:03 **Researcher**
OK
00:37:26 **Participant**
Let me review what I'm doing.
00:37:30 > **Viewing file** (src/todo/components/input.jsx)
00:38:11 - 00:38:12 > **Editing file** (src/todo/components/input.jsx)
00:38:15 > **Viewing file** (src/todo/components/header.jsx)
00:38:24 - 00:38:41 > **Editing file** (src/todo/components/header.jsx)
00:38:41 > **Viewing file** (src/todo/components/input.jsx)
00:38:44 > **Viewing file** (src/todo/components/header.jsx)
00:38:46 - 00:39:50 > **Editing file** (src/todo/components/header.jsx)
00:39:51 **Participant**
Let's see if it solved it.
00:40:00 > **Viewing file** (src/todo/components/header.jsx)
00:40:03 **Participant**
Let me see if this is being saved first.
00:40:06 It changed and keeps, but doesn't pull.
00:40:13 > **Viewing file** (src/todo/components/main.jsx)
00:40:17 - 00:40:19 > **Editing file** (src/todo/components/main.jsx)
00:40:20 - 00:42:53 > **Viewing file** (src/todo/components/header.jsx)
00:40:30 **Participant**
I think when the chat is in agent mode
00:40:34 It does one more thing with you, because I'm using it a lot,
00:41:26 **Researcher**
OK.
00:41:58 **Participant**
Let's understand what's happening at this moment here
00:42:02 that it's not bringing the information that's in local storage.
00:42:30 Maybe this default and value doesn't even work.
00:42:34 Maybe I'm not even passing it
00:42:43 OK, I think it's done.
00:43:08 **Researcher**
OK. Finished the tasks.
00:43:12 **Participant**
I believe so.
00:46:58 **Researcher**
I'm going to send you the last questionnaire for you to answer.
00:47:07 After you finish answering
00:47:10 this questionnaire, I have just a few more questions,

00:47:57

Participant

One thing I noticed in Claude sonnet is that if I asked the same question, to make the filter, it would already observe that there is.

00:48:02

An item, and it would already separate that code I made

00:48:05

It would already give this suggestion or even do the more complicated part

00:48:13

Sometimes it ends up doing

00:48:19

More than it should, but.

00:48:21

If the intention is to be productive.

00:48:25

Researcher

ok.

00:48:32

Participant

I'm using it a lot and it's helping me a lot. I'm taking advantage.

00:48:45

Will the use of AI allow me to perform tasks more quickly?

00:49:26

I believe it helps, yes, but in the mode we in chat mode, no.

00:49:31

Researcher

OK.

00:49:55

Participant

work or study, would increase my productivity.

00:50:37

What did I use here?

00:50:40

I agree a little, I think it would indeed be something useful.

00:50:41

One thing I've noticed is that after the arrival of AI, I'm reflecting more on the question.

00:51:55

Researcher

ok.

00:52:28

Participant

how physically demanding the session was.

00:53:42

How much did you have to exert yourself to reach your performance level?

00:54:37

Did you feel insecure, unmotivated, irritated, stressed, bothered?

00:54:44

I was a little hurt when I saw that

00:54:52

it wasn't an AI in agent mode.

00:54:53

Researcher

thank you, I'll ask you 2 more questions.

00:56:43

How would you have solved these tasks if you weren't using copilot?

00:57:03

Participant

First, I. I think I, in a way,

00:57:16

I did most of the way. It's as if I wasn't using the chat.

00:57:19

It's, I think I would delve more into

00:57:25

understanding where each file is, for example,

00:57:28

where, for example, The register input was there in the Reader?

00:57:31

I thought it would be in the main, so when I started to search the

00:57:35

files, I went straight to this file.

00:57:39

but it wasn't, it was in the Reader, so.

00:57:40

Maybe I should have asked this to the copilot

00:57:47

Where is everything?

00:57:51

But I think it's not a question we would ask all the time.

00:57:52

we want the AI to solve the problems and not for us to find out.

00:57:56

We don't want to do exploration.

00:58:00

We want to produce, so I think normally I would do it.

00:58:01

Still, this independent analysis, using AI or not,

00:58:06

to see where each file is

00:58:10

Opening the package.json file

00:58:21

See what it uses and see if I already know something.

00:58:22

I was following some things it was already using like local store,

00:58:39

so I continued the project.

00:58:44

Researcher

What aspects of the copilot did you like the most and what didn't you like?

00:58:54

Participant

00:59:21

00:59:24 Let me see.
00:59:30 what gave me the impression at the moment.
00:59:35 Some questions I asked here were quite trivial, like changing the color.
00:59:39 It already gave me there, but I ended up doing it by hand,
00:59:51 but it already gave me what I needed.
Researcher
Is there anything you didn't like about the copilot?
00:59:58 **Participant**
I didn't like the chat mode.
01:00:05 I think it will help me. It helps, yes
01:00:18 It was when ordering
01:00:32 that it should have already separated this part in
01:00:34 another place.
01:00:35 **Researcher**
ok
01:00:49 **Participant**
I use other platforms
01:00:53 and I see that.
01:00:54 already does this
01:00:55 It is a little more clever.
01:00:59 **Researcher**
Ah, cool, great. To wrap up
01:01:03 and during when you I saw that you used it little, right?
01:01:07 You have experience with react, you've been working with it for a long time
01:01:12 with it. It was noticeable.
01:01:15 You were already solving things without needing it
01:01:23 But during the session did you change the way you interacted with the copilot?
01:01:28 tried a different approach?
01:01:35 **Participant**
In my use.
01:01:42 I asked the question, already thinking it would.
01:01:45 Maybe give me a better shortcut
01:01:50 You can copy the code and pull, but it's me. That's useful.
01:01:58 The AI gave me the answer in a whole block and I saw
01:02:02 that it only changed this part, I took it myself.
01:02:05 And I copied this part and put it into the code.
01:02:08 **Researcher**
ok
01:02:09 **Participant**
So I think because I'm used to using the agent,
01:02:45 The part of ordering tasks, the AI brought the code with the test field,
01:02:49 but actually it's not test, it's title.
01:02:53 Being active context, the AI didn't realize that
01:02:57 this value doesn't exist.
01:03:14 Yeah, I think they're small things.
01:03:19 And if the proposal is to use the context, shouldn't it look?
01:03:25 **Researcher**
ok
01:03:28 Thank you, I will end the recording
01:03:35 Once again thank you for participating in our study.
03:56:09 > **Response rendering**
- 03:56:17
03:57:55 > **Response rendering**
- 03:58:05
04:01:26 > **Response rendering**
- 04:01:33
04:09:37 > **Response rendering**
- 04:09:52

04:16:20
- 04:16:26

> Response rendering