

Participant P13

00:05:46 > **Viewing file** (package.json)
00:06:01 **Participant**
Good to execute. I imagine that there's a dev script here, right?
00:06:06 So if I give here... well, I imagine this is a react
00:06:09 > **Viewing file** (webpack.dev.js)
00:06:11 **Participant**
webpack... I don't know if I have NPM available
00:06:12 > **Viewing file** (webpack.common.js)
00:06:13 > **Viewing file** (package.json)
00:06:15 **Participant**
here.
00:06:16 Ah, I do, okay? So I imagine it should be MPM run
00:06:20 should present the project. No? Ah, okay. I think I have to do MPM install
00:06:25 first. I don't know if I share the environment here
00:06:28 of the codespace and it leaves things installed.
00:06:34 **Researcher**
No, we just imported the project for you.
00:06:39 **Participant**
Alright, well, so I'll do an NPM install here to
00:06:43 install dependencies.
00:06:45 It installed, so in theory, if I do it now, will it work?
00:06:50 It worked and it is.
00:06:54 it's redirecting to the project.
00:07:01 Hmm, it is.
00:07:05 so it's just to understand the project,
00:07:08 if you come here and edit, I don't know...
00:07:12 It adds here, if I press enter, it adds here to the list
00:07:17 There are states here, right? "Active", "completed," "all," "clear." If I click.
00:07:21 Here it adds style.
00:07:30 No, this is something else.
00:07:35 well, so now I'm going to look at that task.
00:07:38 So when there are tasks, a red box appears with the message "no
00:07:43 tasks here yet". So, if I see here it's this message here, okay?
00:07:48 Change the box color to blue.
00:07:54 And what qualification updates the application. What was the message?
00:08:00 So the first step. I see here that the project.
00:08:04 > **Viewing file** (public/index.html)
00:08:04 **Participant**
has 2 folders, it's a public... From what I see here, the basic session,
00:08:10 The structure of the application, right? What is the title of this here?
00:08:15 "None to do here yet" Let's copy this here just to have
00:08:19 as a reference, right?
00:08:21 In case we get too lost and here there's the search, right?
00:08:22 > **Viewing file** (src/index.js)
00:08:24 **Participant**
Which I imagine this index here is the base of the project and that it has the
00:08:28 **Researcher**
Hmm, Hmm.
00:08:30 **Participant**
pages with the components, right? And this here is a,
00:08:35 I don't know if it's a button, something like that.

00:08:40 It's like a little space, a button, it's not exactly the button.
00:08:44 > **Viewing file** (src/todo/components/footer.jsx)
00:08:44 **Participant**
Ah, this is important, okay? We have a header,
00:08:44 > **Viewing file** (src/todo/components/header.jsx)
00:08:46 > **Viewing file** (src/todo/components/input.jsx)
00:08:47 **Participant**
we have an input which is our little box.
00:08:47 > **Viewing file** (src/todo/components/item.jsx)
00:08:53 **Participant**
More that we have the todo-inputs.
00:08:58 > **Viewing file** (src/todo/components/main.jsx)
00:08:59 **Participant**
Well, it probably defines several things here,
00:09:03 let me hide here... "none to do here",
00:09:07 what's the message here? Ah, okay,
00:09:11 this guy here that we want, alright, okay? So it has a color here that is.
00:09:17 In the document that this is a non-specific red color and he wants to change to
00:09:22 this hexadecimal, no, I forgot the name of this here, but anyway,
00:09:26 wants to change to this hashtag here
00:09:30 So it's this here and an HTML element
00:09:35 and it is being applied in style here which is this todo empty message. OK,
00:09:42 we're going to come here, in our styles and look for it,
00:09:46 I imagine that
00:09:48 > **Viewing file** (src/todo/app.css)
00:09:49 **Participant**
maybe it's in this style here, let's see exactly if it is
00:09:54 inside here of the... it probably has all the styles of the components here, right?
00:09:58 So its background, this red, I will come here and change to ...
00:10:02 > **Editing file** (src/todo/app.css)
- 00:10:14
00:10:05 **Participant**
I will change to this one that he wants.
00:10:11 Ah, let me ... Ah, it accepts string like this
00:10:15 it changed here, then I don't know if
00:10:20 if the command I ran, it updates automatically
00:10:25 I imagine so. If I refresh here what will it do?
00:10:32 Ah, good. It changed the button color, so I think the first task
00:10:38 checked
00:10:39 When the application updated and confirmed that a blue box appears
00:10:43 Now, the practical task B, change the placeholder
00:10:46 The placeholder text in the input field
00:10:50 "what needs to be done?" Which is this one, Ok.
00:10:53 change the text to
00:10:54 "Type a task here" and once again, click to update the text.
00:10:58 So we will take this text here that he wants.
00:11:04 Right?
00:11:04 We can come here, right? We saw here that we have our
00:11:08 > **Viewing file** (src/todo/components/main.jsx)
00:11:11 **Participant**
main here.
00:11:11 Is there any information?
00:11:12 > **Viewing file** (src/todo/components/input.jsx)
00:11:13 **Participant**
Yes. I imagine it must have the text in

00:11:18 question. Ah, there's a placeholder here.
00:11:22 Ah, ok, this is a component, right?
00:11:26 Probably it's the main that... let me do something
00:11:29 on my keyboard here, hold on... done. so this here, probably
00:11:35 > **Viewing file** (src/todo/components/main.jsx)
00:11:36 **Participant**
it's imported here, where is it? Ah, a question, can I use the function to see,
00:11:41 > **Viewing file** (src/todo/components/item.jsx)
00:11:41 **Participant**
for example, I have this input here.
00:11:43 > **Viewing file** (src/todo/components/input.jsx)
00:11:44 **Participant**
Can I use the function to see its references?
00:11:48 **Researcher**
Yes, yes, of course.
00:11:51 **Participant**
So, Ah, and it's being used here in the item
00:11:52 > **Viewing file** (src/todo/components/item.jsx)
00:11:54 **Participant**
edit todo input which is its label, so I have to come here to change this
00:11:57 > **Editing file** (src/todo/components/item.jsx)
00:12:00 **Participant**
to "type a task here". OK. We come here, update.
00:12:13 Why didn't it edit? what did I do wrong here?
00:12:17 This here is an input, right? let me see.
00:12:21 It's "what needs to be done?".
00:12:25 Could it be if I
00:12:26 update again it works?
00:12:28 No, so I think I changed it in the wrong place. Let me see.
00:12:31 > **Editing file** (src/todo/components/item.jsx)
00:12:34 **Participant**
I don't think I. Ah, it's not right. What was, what needs to be done?
00:12:38 Ah no, it's right.
00:12:41 Actually, the text is "what needs to be done?" and "type a task here?"
00:12:43 and it's not changing.
00:12:47 > **Editing file** (src/todo/components/item.jsx)
00:12:51 **Participant**
I think it's here.
00:12:57 Could I find more occurrences of this "what needs to be done?"
00:13:02 I'll try to find here in the project the occurrences of this text here to see
00:13:08 > **Viewing file** (src/todo/components/header.jsx) through file search
00:13:08 **Participant**
if I'm missing something. Hmm, Ah, okay, Ah, this is another one.
00:13:12 > **Viewing file** (src/todo/components/item.jsx)
00:13:14 **Participant**
I got confused.
00:13:17 > **Editing file** (src/todo/components/item.jsx)
00:13:18 **Participant**
Or maybe I had to put both.
00:13:20 There's this other input here, which I ended up misleading here.
00:13:21 > **Viewing file** (src/todo/components/header.jsx)
00:13:26 > **Editing file** (src/todo/components/header.jsx)
- 00:13:37
00:13:26 **Participant**
Come here.

00:13:29 And I put
00:13:32 Type a task here.
00:13:42 If I come here
00:13:48 Ah, great. So it's this one... probably if
00:13:52 it's here, it must be some other guy that appears when
00:13:55 I type, right?
00:13:58 Great, now order tasks in the order in,
00:14:00 which they were added, modify the app so that the
00:14:04 tasks are inserted alphabetically. Hmm,
00:14:10 alphabetical order.
00:14:13 Okay? So I, for example, if I come here and type one.
00:14:22 It's not, I think not...
00:14:29 Let me type here, two.
00:14:34 Wait, I think I did something wrong that it
00:14:37 isn't.... did it give an error here? what did it give?
00:14:44 Ah, okay, it's not this one. I think it was before. Where? Type a,
00:14:48 task here... but this one was from before, why is it giving this error?
00:14:52 Let me see.
00:14:59 Hmm?
00:15:01 > **Viewing file** (src/todo/components/item.jsx)
00:15:04 > **Viewing file** (src/todo/components/header.jsx)
00:15:04 **Participant**
Edit, todo-input I don't know why at first it's
00:15:08 giving this problem.
00:15:10 > **Viewing file** (src/todo/components/input.jsx) through file search
00:15:12 **Participant**
Let's go here ...
00:15:16 Ah, and there's an input here, there are several placeholders, then there's this input here that is
00:15:22 used in the.
00:15:25 to-do, if I have somewhere that now
00:15:28 is giving this problem? but I don't think so.
00:15:30 > **Viewing file** (src/todo/components/header.jsx)
00:15:34 **Participant**
new label, type a to-do here... practically everything is
00:15:38 all right.
00:15:47 Let me do the following, I will cancel the execution,
00:15:50 I will run it again to clear the terminal so it doesn't get confusing
00:15:53 **Researcher**
Hmm, Hmm.
00:15:53 **Participant**
with the error message.
00:15:58 Double click to edit.
00:16:01 It was adding before, now it's not, what a thing,
00:16:05 I don't know ... Hold on, the change I made
00:16:09 before was this one in the first task and this
00:16:12 one in the second.
00:16:17 Hmm?
00:16:25 Ah, okay, let me see here.
00:16:29 Why was it before, and now it's not? let me see.
00:16:30 > **Viewing file** (src/todo/app.css)
00:16:32 > **Viewing file** (src/todo/components/header.jsx)
00:16:36 **Participant**
For example, if I undo here what I did and
00:16:39 go back to the previous state, will it go? Let's see.
00:16:43 > **Editing file** (src/todo/components/header.jsx)

00:16:46

Participant

We come here and now I will run again.

00:17:03

Ah, okay, there's a rule that I can't add,

00:17:06

probably... I can't add things that have a so

00:17:10

understood, understood, okay, so it's right,

00:17:13

so let me go back here.

00:17:14

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

00:17:15

Participant

What's up? We're going to do here, okay?

00:17:22

actually I think I might have to edit the rule here so I can

00:17:27

add this "o" here, because before, it didn't allow that.

00:17:33

Well, so let's go, there's this item here.

00:17:42

Which is probably what we use, right? To add here.

00:17:45

If I come here, for example, in this component.

00:17:48

It has item label, right? Let's just confirm if that's it, right?

00:17:54

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

00:17:59

Participant

Huh? So, hold on, it's ADD_ITEM, payload, title, usecallback.

00:18:00

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

00:18:12

Participant

ADD_ITEM ?

00:18:19

It has an action here of type ADD_ITEM.

00:18:20

and it sends this payload, and this action here, what does it do?

00:18:22

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

00:18:28

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

00:18:41

Participant

It adds the title, right?

00:18:42

That I sent to it, okay? So I imagine it must

00:18:47

based on that component, right?

00:18:49

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

00:18:52

Participant

That we see there on the page. It's probably referring to this one, right?

00:18:55

At some point it looks at this one,

00:18:56

It returns a label with the title, right?

00:19:02

Which is what's filled in there

00:19:04

We found, Where this is done

00:19:07

new item

00:19:32

So we have here.

00:19:41

This one is the...

00:19:45

well, I imagine all this refers to

00:19:56

this specific item here.

00:19:59

it probably takes this

00:20:17

and iterates

00:20:21

Because each of these is an item, In a list

00:20:22

ordered, but here it is ordered by the order of

00:20:27

addition, right?

00:20:31

Which is the return of this function.

00:20:35

So let me see where this is called.

00:20:38

No, but it is a component, right? It must be like this.

00:21:04

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

00:21:08

Participant

Here it is? visible all. so here it is iterating in all point ID.

00:21:18

And what is this?

00:21:21

00:21:32 Well, so I have two options,
00:21:37 I imagine, which is either insert ordered or sort them later
00:21:40 I imagine that either of the 2 is a good option.
00:22:06 Hmm, visible todo point length greater than zero.
00:22:12 Uh-huh. And then he takes it.
00:22:17 I have to sort by title, right, so.
00:22:23 I might have to sort before this one here.
00:22:30 Probably because it controls the display, right?
00:22:38 Uh-huh.
00:22:38 now I don't remember if the map...
00:22:43 I think I'm going to
00:22:47 I think the JavaScript map iterates sequentially
00:22:51 in the list.
00:22:52 **> Editing file** (src/todo/components/main.jsx)
00:22:52 **Participant**
00:22:57 So I'm going to have to sort this list before we go through this map.
00:23:01 I don't know if I can do this here,
00:23:01 where is it?
00:23:35 These effects, probably used when there's a
00:23:38 change.
00:23:49 Let me see here.
00:23:57 There's a problem I have
00:24:01 to consider is how this is rendered, right?
00:24:09 Hmm, yeah. Maybe it's interesting for me to consider,
00:24:12 I think I'm going to ask for a suggestion
00:24:17 here in our chat to solve this, let's see, how could I?
00:24:23 How am I going to ask the question? Yeah, Ah, one thing,
00:24:25 can I ask in English or Portuguese? Or does it not matter, I don't know?

00:24:28 **Researcher**
00:24:33 You can ask in both English and Portuguese.

00:24:33 **Participant**
00:24:38 Okay, so, because it's been a while since I worked with
00:24:41 frontend, in this case, let's ask here
00:24:45 I would like to add
00:24:45 No... better.... I would like the items.
00:24:51 Let me see... I'm using the hashtag codebase.
00:24:53 It considers as context all that are open, right?
00:24:57 I don't need to specify this when writing here, right? If it's open.
00:25:00 **Researcher**
00:25:03 If you put the codebase
00:25:07 it considers the whole project. If you don't put the codebase
00:25:11 it considers as context only those that are open.

00:25:11 **Participant**
00:25:13 Ah, okay, so, well, in this case, I think it reduced.
00:25:17 The context is always good, right? Because otherwise it will keep considering
00:25:19 too many things, right?
00:25:25 I would like the items added
00:25:25 to the list to always be in alphabetical order.
00:25:35 Any suggestion
00:25:38 for that? Yeah, I'll specify here... "That is,
00:25:44 The.
00:25:46 Component," then I'll be very specific here.
00:25:54 "Present in main.jsx

00:25:56 **Chat** (window: 0)
> Agent
github.copilot.default

> Context

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

> Message

I would like the items added to the list to always be in alphabetical order, that is, the `<Item>` component present in main.jsx. Any suggestions for this?

> Response

To ensure that the items in the list are displayed in alphabetical order, you can sort the [visibleTodos \(src/todo/components/main.jsx\)](#) before rendering them. This can be done directly in the [useMemo \(node_modules/@types/react/index.d.ts\)](#) that already filters the items based on the route. Just add a sorting step to the filtered array.

Here is the modification in the code:

```
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.title.localeCompare(b.title)), // Sorts
  the items alphabetically by title
  [todos, route]
);
```

Explanation:

1. [filter \(src/todo/components/main.jsx\)](#): Maintains the existing logic to filter the items based on the route (/active, /completed, or all).
2. **sort**: Adds a sorting step using the `localeCompare` method, which compares strings in a language-sensitive manner. Here, I assume that each [todo \(src/todo/components/main.jsx\)](#) has a `title` property that contains the item's text.

Result:

Now, the items will be displayed in alphabetical order in the list, regardless of the order in which they were added.

If the property that contains the item's text is not `title`, replace `a.title` and `b.title` with the correct property name.

> Response rendering

Researcher

I want to tell you a little thing, just so you don't click close on the x of any of the chats. Just so we can save it later.

Participant

Ah, okay, great.

Researcher

Thank you.

Participant

Ah.

He said... Ah, made a good suggestion here.

Hmm, okay, so here it's filtering, right?

He uses this `localeCompare`.

> **Implementing Copilot code** using apply button (src/todo/components/main.jsx)

00:25:59
- 00:26:07

00:26:08

00:26:12

00:26:14

00:26:17

00:26:18

00:26:19

00:26:23

00:26:35

00:26:44
- 00:26:57

00:26:44

Participant

I'm going to use Apply directly here.

00:26:46

Will he complain? Let's see.

00:26:58

So, maybe it would have been faster if I had done this directly, copying, but anyway.

00:27:02

That's it, right? So he is still doing filtering he is.

00:27:07

He takes the two commands, he uses the locally compare.

00:27:08

He put a comment here, I'm going to remove the comment.

00:27:13

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

- 00:28:07

00:27:19

Participant

Here, but is the editing over? if I write anything here will he continue editing?

00:27:23

Whoa, hold on.

00:27:29

Didn't understand, let me erase here.

00:27:36

I always said this here no. Hold on.

00:27:49

Was it me who did this?

00:27:53

Here.

00:27:59

But what is he complaining about here?

00:27:59

He did some weird stuff here, let me go back.

00:28:02

Ah, okay, now I just changed this here, right? In theory, I think that.

00:28:10

With luck maybe it would already solve. So if I come here, for example, update.

00:28:17

Will it be? So come here 12.

00:28:23

If I type 13, 09, he adds first, cool,

00:28:26

Only we have the requirements here, right?

00:28:30

I have to remove the

00:28:36

requirement here, right? So let's come here. Is there a moment when he checks the size here? Probably.

00:28:43

let me add, for example, this one, right?

00:28:46

And it probably has to do with some input rule that doesn't let me do this. Yeah. So let's see here.

00:28:51

I'm going to search to do it faster, right? We do here,

00:29:00

I'm going to search directly by the placeholder.

00:29:05

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

00:29:10

Participant

He sends the title here, right? But somewhere he is, he is not allowing to add, it's things with .

00:29:16

the title less than 2, right?

00:29:19

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

00:29:20

Participant

Ah, let's see.

00:29:29

Where is this guy?

00:29:34

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

00:29:39

Participant

What a thing. There's only one here.

00:29:42

It is. I'll ask here again, so apparently,

00:29:46

there is a rule that does not allow adding all.

00:30:09

Or rather, adding items with texts that have one character, one character.

00:30:17

How could I remove this restriction.

00:30:25

Chat (window: 0)

00:30:31

> **Agent**

github.copilot.default

00:30:36

> **Context**

00:30:45

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

> Message

Apparently there is a rule that does not allow adding items with texts that have 1 character. How could I remove this restriction?

> Response

The restriction of not allowing adding items with texts of only 1 character is probably implemented in the `<Input>` component or in the validation logic associated with [onSubmit \(src/todo/components/header.jsx\)](#). To remove this restriction, you will need to check the code of the `<Input>` component.

If the validation is in the [onSubmit \(src/todo/components/header.jsx\)](#) of the `<Input>`, you can adjust the [addItem \(src/todo/components/header.jsx\)](#) function in the [Header \(src/todo/components/header.jsx\)](#) to accept any text, regardless of length. Here is how you can modify the code:

```
File: src/todo/components/header.jsx
export function Header({ dispatch }) {
  const addItem = useCallback((title) => {
    // Remove any length validation here
    dispatch({ type: ADD_ITEM, payload: { title } });
  }, [dispatch]);

  return (
    <header className="header" data-testid="header">
      <h1>Afazeres</h1>
      <Input onSubmit={addItem} label="Novo afazer"
placeholder="Digite um afazer aqui" />
    </header>
  );
}
```

Next steps:

1. Check the `<Input>` component:

- If the length validation is implemented in the `<Input>` itself, you will need to edit this component to remove the restriction.

2. Test the functionality:

- After the change, try adding items with 1 character to ensure the restriction has been removed.

If you can share the code of the `<Input>` component, I can help locate and remove the validation directly in it.

> Response rendering

Participant

It's, it's on onSubmit.
the callback.

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

Participant

He suggested that maybe it's inside this component, right?
maybe we needed this prompt here because I ended up
getting confused here.

But it's okay, at least he helped me with this, right?

So he has a res here, I'm going to remove it because it already
does string processing here.

And great, I think with this we can already
do a battery of tests here.

Okay, so I have to add the quick 123,
it's the quick 123 brown
fox.

00:30:47
- 00:30:52

00:31:02

00:31:06

00:31:15

00:31:15

00:31:19

00:31:23

00:31:24

00:31:27

00:31:31

00:31:35

00:31:38

00:31:43

00:31:47

00:31:55

00:31:59 It's 123 brown the quick and fox. OK. So that's it.
00:32:05 I think we are done with this task c.
00:32:11 **Researcher**
Great.
00:32:11 I was unavailable for a few minutes, I don't know if you noticed.
00:32:15 **Participant**
No, no.
00:32:16 **Researcher**
My connection dropped. Let's go back, let's hope everything works out, we got people. I'll send you another one.
00:32:21 We still have a few minutes, I'll send you another file here
00:32:25 so you can start the first task from there and we'll go as far as
00:32:30 we can.
00:32:36 **Participant**
00:32:58 Task A: persist.
00:33:13 correct or implement functionality.
00:33:17 Hmm, Hmm.
00:33:24 So he said here that they already tried in the past, but it's having a problem.
00:33:30 Okay, so all-components. Well, so I imagine that.
00:33:33 > **Viewing file** (src/todo/components/header.jsx)
00:33:34 > **Viewing file** (src/todo/components/main.jsx)
00:33:36 **Participant**
The submit we have here.
00:33:41 It probably should do this, right?
00:33:42 > **Viewing file** (src/todo/components/input.jsx)
00:33:53 **Participant**
Ah, it's true, we have to delete this function here
00:33:55 too, that we won't use anymore.
00:34:00 Okay, but here they call submit, right? Ah, okay. It's passed by the input,
00:34:06 > **Viewing file** (src/todo/components/item.jsx)
00:34:10 **Participant**
Ah, who's passing is this handleupdate
00:34:12 here
00:34:16 Uhum. Okay? So, probably the one having a problem is this
00:34:19 update item here
00:34:30 Update item
00:34:35 callback, title dispatch here which is the update item.
00:34:47 > **Viewing file** (src/todo/constants.js)
00:34:48 **Participant**
Ah, there's this one here.
00:34:50 Where else is it used?
00:34:53 > **Viewing file** (src/todo/reducer.js) through file search
00:34:53 **Participant**
Our reducer.
00:35:02 todo-reducer.
00:35:14 Hmm?
00:35:14 And then there's this save state. It's probably what interacts with the
00:35:18 database. Ah, it saves in the local storage.
00:35:23 Hmm, okay? And then it saves the Json from the stringify.
00:35:37 For example, we of id_item, let me find something.
00:35:39 thing.
00:35:45 It's not working
00:35:47 So, the stage.com cache.
00:35:56 Okay, in this case I will
00:36:00 ask the copilot

00:36:05
00:36:06
00:36:10

00:36:16
00:36:23
00:36:35
00:36:41
00:36:52
00:36:58
00:37:05

if I give a command + L I think it takes the context here, I will

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

Participant

select here to see if it takes this little context here. Done.

Okay? Ah, I don't know if the function or ...

I'll put it like this, reducer ADD_item, apparently

It's having problems saving the items

In the local storage.

I don't know. Point out some reasons

why this is happening.

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

selection (src/todo/reducer.js)

> **Message**

The ADD_ITEM reducer apparently has problems saving items in localStorage. Point out some reasons why this is happening.

> **Response**

Here are some possible reasons why the [ADD_ITEM\(src/todo/constants.js\)](#) may not be saving items correctly in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)):

1. **Incorrect key in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)):**

- In the [saveState\(src/todo/reducer.js\)](#) method, the key used to save data is "todos", but in the [loadState\(src/todo/reducer.js\)](#) method, the key used to load data is "todoS". This inconsistency can cause problems since the saved and loaded state are not using the same key.

Correction: Make sure to use the same key in both methods. For example:

```
const loadState = () => {  
  const state = localStorage.getItem("todos"); // Correct to  
  "todos"  
  return state ? JSON.parse(state) : [];  
};
```

2. **Error in the reducer flow:**

- If the [ADD_ITEM\(src/todo/constants.js\)](#) is not being called correctly or if the [action.payload.title\(src/todo/reducer.js\)](#) is empty or invalid, the item may not be added to the state, and the [saveState\(src/todo/reducer.js\)](#) will not be called with the new state.

Correction: Check if the [action.payload.title\(src/todo/reducer.js\)](#) is being passed correctly and is not empty. Add a log for debugging:

```
case ADD_ITEM:  
  console.log("Adding item:", action.payload.title); //  
  Debugging  
  newState = state.concat({ id: generateUniqueID(), title:  
    action.payload.title, completed: false });  
  break;
```

3. **Error in [generateUniqueID\(src/todo/reducer.js\)](#):**

- If the [generateUniqueID\(src/todo/reducer.js\)](#) function is not generating unique IDs correctly, there may be conflicts in the state, which can cause problems when saving to [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)).

Correction: Ensure that [generateUniqueID \(src/todo/reducer.js\)](#) is working correctly. You can test the function separately to verify that it generates unique IDs.

4. Error in [saveState \(src/todo/reducer.js\)](#):

- If the [saveState \(src/todo/reducer.js\)](#) is not being called correctly or if there is an error saving to [localStorage \(node_modules/@types/node/globals.d.ts\)](#), the data will not be persisted.

Correction: Add a log in [saveState \(src/todo/reducer.js\)](#) to check if it is being called and if the state is correct:

```
const saveState = (state) => {  
  console.log("Saving state:", state); // Debugging  
  localStorage.setItem("todos", JSON.stringify(state));  
};
```

5. Browser issues:

- If [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is disabled or full in the browser, the data will not be saved.

Correction: Check if [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is working correctly in the browser. You can test manually in the browser console:

```
localStorage.setItem("test", "testValue");  
console.log(localStorage.getItem("test"));
```

Summary:

- Fix the inconsistency in the [localStorage \(node_modules/@types/node/globals.d.ts\)](#) key ("todos" vs "todoS").
- Check if the [action.payload.title \(src/todo/reducer.js\)](#) is being passed correctly.
- Test the [generateUniqueID \(src/todo/reducer.js\)](#) function to ensure it generates unique IDs.
- Add logs in [saveState \(src/todo/reducer.js\)](#) and [ADD_ITEM \(src/todo/constants.js\)](#) for debugging.
- Ensure that [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is enabled and functional in the browser.

00:37:07
- 00:37:23

> Response rendering

00:37:20

Participant

Maybe there would be this with concat here, but let's see.

00:37:25

Incorrect key in the method ...

00:37:29

but the loadstate method ...

00:37:32

Ah, but the loadstate method it points here to the loadstate.

00:37:38

I didn't get to see this.

00:37:47

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

00:37:52

Participant

In the local storage, as if it put the key of all.

00:37:54

And then ... it's for the account of the state change.

00:37:56

It always saves a Json

00:37:58

with a snapshot of the state at the moment, I see.

00:38:04

In the correct key, I in the flow, whether in the status of 9 current by the state

00:38:11

check it's not empty.

00:38:15

but it seems like...

00:38:20

this suggestion of his seems quite plausible here.

00:38:25

doesn't seem to be any problem. So let's go, persist, be fixed, okay?

00:38:30

So he made this suggestion, it seems reasonable and now, if I,

00:38:35

for example, if I.

00:38:36

Come here and press control + shift + R ...

00:38:43

It will erase everything, because it's not the state it was, right?

00:38:47

So, for example, you see that typing, I don't know,

00:38:50

is there a test he comments here? no So now ...

00:38:55

if I refresh it was supposed to keep here.. it didn't keep, so it's not correct.

00:39:15

maybe there's an item problem

00:39:18

that maybe... Ah, let's see here what else he suggested

00:39:22

in the flow.

00:39:29

Yeah, but it's probably not that,

00:39:31

because it's always empty.

00:39:38

Yeah, it could be a browser thing too

00:39:40

check if... it's always... correctly, okay?

00:39:44

Let's see here.

00:39:50

Action.payload.title. What does this refer to? what suggestion

00:39:54

of his?

00:39:59

It's empty ... invalid.

00:40:06

Let me see if how this one is being used?

00:40:12

> **Viewing file** (src/todo/constants.js) through file search

00:40:15

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

00:40:19

Participant

Payload title, so pass the item here,

00:40:22

would have to use state dot payload dot title. Is that what's happening here?

00:40:23

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

00:40:33

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

00:40:34

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

00:40:46

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

00:40:47

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

00:40:51

Participant

then he saves this new state.

00:40:58

Ah, wait. Ah, there's something wrong here too, this load

00:41:01

state too.

00:41:03
00:41:06
00:41:06

00:41:15
00:41:17
00:41:20
00:41:24
00:41:30
00:41:36
00:41:43
00:41:47
00:41:52
00:41:59
00:42:02

That's it, yeah, could be.

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

Participant

Did he get to comment on this here?

There was a typo here.

Let's see if it could be this.

Control + Shift

Then I know what 123 is, so if I do control + shift + R?

In this case I'll tell him "even

with the proposed fix, the todos are still not persisted

but this time I'll give codebase context

because maybe it's not here.

"what else could be causing

this failure?"

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

codebase

> **Message**

#codebase Even with the proposed fix, the todos are still not being persisted. What else could be causing this failure?

> **Response**

If the todos are still not being persisted in the [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) even after fixing the code, here are some other possible causes and solutions:

1. Error in the initial state loading ([initialState](#) ([src/todo/reducer.js](#)))

- The initial state is loaded using [loadState\(\)](#) ([src/todo/reducer.js](#)). If the [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) does not contain valid data or if the return of [loadState\(\)](#) ([src/todo/reducer.js](#)) is not being used correctly, the initial state may be overwritten.

Fix: Ensure that the [initialState](#) ([src/todo/reducer.js](#)) is being correctly passed to the [useReducer](#) ([node_modules/@types/react/index.d.ts](#)) in the `app.jsx` file:

```
const [todos, dispatch] = useReducer(todoReducer, initialState);  
// Use initialState correctly
```

2. Problem with [saveState](#) ([src/todo/reducer.js](#)) not being called

- The [saveState](#) ([src/todo/reducer.js](#)) is called at the end of the [todoReducer](#) ([src/todo/reducer.js](#)). If the [todoReducer](#) ([src/todo/reducer.js](#)) is not being executed correctly or if there is an error before [saveState](#) ([src/todo/reducer.js](#)), the data will not be saved.

Fix: Add logs in the [todoReducer](#) ([src/todo/reducer.js](#)) to check if the [saveState](#) ([src/todo/reducer.js](#)) is being called:

```
const saveState = (state) => {  
  console.log("Saving state to localStorage:", state); // Debug  
  log
```

```
    localStorage.setItem("todos", JSON.stringify(state));  
  };
```

3. Problem with [localStorage \(node_modules/@types/node/globals.d.ts\)](#) in the browser

- The [localStorage \(node_modules/@types/node/globals.d.ts\)](#) may be disabled or full in the browser, preventing data persistence.

Fix: Manually test in the browser console:

```
localStorage.setItem("test", "testValue");  
console.log(localStorage.getItem("test")); // Should return  
"testValue"
```

If the [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is full, you can clear the data:

```
localStorage.clear();
```

4. Problem with the action flow in [dispatch \(src/todo/app.jsx\)](#)

- If the [dispatch \(src/todo/app.jsx\)](#) is not being called correctly or if the [action.payload \(src/todo/reducer.js\)](#) is empty, the state may not be updated.

Fix: Add logs in the [Header \(src/todo/components/header.jsx\)](#) and in the [todoReducer \(src/todo/reducer.js\)](#) to check if the [dispatch \(src/todo/app.jsx\)](#) is being called correctly:

```
const addItem = useCallback((title) => {  
  console.log("Dispatching ADD_ITEM with title:", title); //  
  Debug log  
  dispatch({ type: ADD_ITEM, payload: { title } });  
}, [dispatch]);
```

5. **Problem with the `use

00:42:13
- 00:42:22

> Response rendering

00:42:31

Participant

The initial state is loaded using loadstate, if loadstate doesn't contain valid data or if the return from loadstate make sure the initial state is being passed correctly...

00:42:34

00:42:36

00:42:41

Hmm?

00:42:44

saveState is not being called, oh wow.

00:42:49

Wow! Ah, I don't think it's being called, yes.

00:42:58

Uh-huh. It's being called, yes

00:43:01

What else?

00:43:08

Ah, okay, so the problem might be in this

00:43:10

reducer here ...

00:43:14

This here seems to be quite valid.

00:43:18

Who does he talk about here?

00:43:21
- 00:43:26

> **Implementing Copilot code** using apply button (src/todo/reducer.js)

00:43:24

Participant

Wait, What did he do here? Uses reducer.

00:43:41

But I think he.

00:43:44
- 00:43:48

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

00:43:45

Participant

Is it?

00:43:46

Maybe he put it in a kind of wrong place here.

00:43:54

which file did he send here? Ah okay,

00:43:57

it's not .js. He's speaking kind of wrong.

00:43:59

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

00:43:59

Participant

but he put it here, or I don't know if I clicked wrong too.

00:44:03

let me see, it's .js

00:44:09

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

00:44:09

Participant

JFX

00:44:11

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

00:44:13

Participant

The right thing would be to use with this initState

00:44:14

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

00:44:21
- 00:44:29

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

00:44:22

Participant

Ah, but is that already happening?

00:44:31

I don't know if the suggestion is valid.

00:44:38

Local storage.

00:44:48

Ah, okay, but wait.

00:44:49

So, it always starts empty.

00:44:51

The idea is that I have to get it from there... I understand what he meant now.

00:45:02

So let's do the following.

00:45:06

How could I use the initial state as the one in local storage here?

00:45:14

Chat (window: 0)

> Agent

github.copilot.default

> Context

selection (src/todo/app.jsx)

> **Message**

How could I use the initial state as the present in local storage?

> **Response**

The problem is that the initial state of the [useReducer \(node_modules/@types/react/index.d.ts\)](#) is being set as an empty array ([]) instead of using the loaded state from localStorage (initialState). To fix this, you can pass the initialState loaded from localStorage as the second argument of the [useReducer \(node_modules/@types/react/index.d.ts\)](#).

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

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

import { todoReducer, initialState } from "../reducer"; // Import the
initial state

import "../app.css";

export function App() {
  const [todos, dispatch] = useReducer(todoReducer, initialState); //
  Use the initial state loaded from localStorage

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

Explanation:

1. initialState:

- The initialState is already being loaded from localStorage in the reducer.js file using the loadState function. Just use it as the initial state in the [useReducer \(node_modules/@types/react/index.d.ts\)](#).

2. Passing the initial state:

- The second argument of the [useReducer \(node_modules/@types/react/index.d.ts\)](#) defines the initial state. We replace the empty array ([]) with the initialState loaded from localStorage.

Result:

Now, the initial state of the [todos \(src/todo/app.jsx\)](#) will be correctly loaded from localStorage, and changes made to the state will be persisted using the saveState function in the [todoReducer \(src/todo/reducer.js\)](#).

> **Response rendering**

Participant

It's here, in this case, I think I'll ask this question with the codebase, because I think he doesn't have in initialState

00:45:17
- 00:45:30

00:45:36

00:45:40

00:45:43 but it's taking this initialstate, it imports from...
00:45:48 okay, okay true, good point. I think if I click
00:45:51 > **Implementing Copilot code** using apply button (src/todo/app.jsx)
- 00:45:53

00:45:51 **Participant**
here it will apply automatically.
00:45:52 Alright, okay, makes sense. Will it work now?
00:46:01 > **Viewing file** (src/todo/reducer.js)
00:46:02 **Participant**
Let's see. I don't know if those lint errors there
00:46:09 will...
00:46:15 Great. Well, thank you, chatGPT.
00:46:18 **Researcher**
That's it, right? It worked, right?
00:46:22 **Participant**
It worked
00:46:23 **Researcher**
Great, thank you very, very much. I will now suggest that we
00:46:28 finish with the questionnaire that I
00:46:32 will send you and then I want to ask you a few questions too,
00:52:06 And those first ones, how difficult did you find them? just to
00:52:09 complement this question.
00:52:12 **Participant**
No, they were a simple test. I, I think it contributes a bit to...
00:52:15 because the perception of difficulty is a bit difficult,
00:52:18 because it's been a while since I've had contact with front end, so to speak,
00:52:21 right?
00:52:21 Hmm?
00:52:22 Especially the webpack that's being used. So I think yes. Like,
00:52:25 if I had, I don't know, if an hour before I had read
00:52:28 something basic like that, right? Or was a professional in frontend,
00:52:31 I think it would be extremely easy.
00:52:54 **Researcher**
Great.
00:52:55 So, what I wanted to know is with your
00:52:59 experience, right? In this, this session, with the tasks,
00:53:04 were you able to orient yourself in the code how the copilot or, anyway,
00:53:09 other techniques you used helped you?
00:53:15 **Participant**
Well, uh, I think that the project is well structured, right?
00:53:19 so that helps a lot. a good project, first,
00:53:23 which is a small project, right? And then it's well structured.
00:53:27 This helps a lot in navigating, right? Without the aid of something.
00:53:33 the copilot, it helped me here.
00:53:34 It's because, there were
00:53:37 some things that I didn't remember off
00:53:40 the top of my head, right?
00:53:40 Because it's been a while since I've had contact
00:53:43 With some of the tools that are being used here
00:53:47 the Webpack and one thing or another of frontend
00:53:50 In the matter of reducer and states.
00:53:53 So I think that the copilot helped a lot in remembering some things
00:53:57 about this and also to confirm some hypotheses I have, right?
00:54:01 Sometimes I have an idea like, for example, this sort one.
00:54:04 I had two approaches, right? I thought, Ah, either I can sort
00:54:06 the list or I can insert sorted.

00:54:09 And then I asked chatgpt and it kind of gave me a confirmation.
00:54:16 so I'm not, I don't know, I'm not crazy, right? since chatGPT...
00:54:19 no...
00:54:20 The copilot, also suggested this to me,
00:54:23 so probably there is some basis for this hypothesis of mine,
00:54:28 so, it was very useful for that.
00:54:32 To refresh the memory of things and as it also
00:54:36 has a context
00:54:38 project links, "this component
00:54:40 is being used in another part, so you have to make this and that
00:54:44 change". I think it's very interesting this
00:54:46 tool of it suggesting here.
00:54:48 And then you see it as if it were a Git Diff of the difference.
00:54:52 I'm doing this here... the state before was this
00:54:55 and I'm adding this now. I think it's something that helps
00:54:59 a lot also for you not to just accept
00:55:02 simply its suggestions. You have a basis like, "Ah,
00:55:05 this here that you suggested really makes sense, this here too."
00:55:08 "This here that you suggested, has nothing to do."
00:55:10 So I will disregard it.
00:55:11 **Researcher**
Hmm?
00:55:13 Taking advantage of your answer?
00:55:17 Was there any... In this, in this final questionnaire here
00:55:22 there were some aspects about the response, about the interaction with the
00:55:28 copilot, that you said you liked and didn't like.
00:55:33 What were the aspects that, like, the things that stood out the most for
00:55:40 you?
00:55:40 Besides what you just mentioned, right?
00:55:43 About being able to click and have this diff,
00:55:46 what else caught your attention with the interaction with the copilot?
00:55:51 **Participant**
One thing that caught my attention a lot is the issue of references.
00:55:55 That is, it gives the answer here and gives you
00:55:57 access to the reference it used. This is a feature I found really
00:56:02 cool. I have experience
00:56:03 with Cursor. I don't remember if Cursor also
00:56:06 does this.
00:56:07 I think so, maybe in another way,
00:56:10 but these references here. I find it very interesting, right?
00:56:12 For you to see how it was before and what it's thinking.
00:56:16 As if it were explaining the thought process to you of what
00:56:18 it's doing. So these differences, I think it's a very
00:56:19 **Researcher**
Hmm?
00:56:22 **Participant**
cool highlight and it explains, like, it says,
00:56:25 gives the context it's thinking and it also gives reference at the time of the
00:56:29 response.
00:56:31 I'm considering this context here with these references, and I did this.
00:56:35 And the explanation of why I did this also has reference for that code.
00:56:41 **Researcher**
Did you have to understand the codebase?
00:56:46 How familiar did it become to you?
00:56:50 **Participant**
Is it after considering the interaction with copilot or in general?

00:56:56

Researcher

Throughout the session, compared to the beginning.

00:57:00

Participant

Ah, yes.

00:57:06

So I think well. What the exercises

00:57:09

asked for, helped you get used to

00:57:13

more with what is being done.

00:57:16

I think the basic tasks they

00:57:20

addressed important flows. Main flows,

00:57:24

like for example, adding the task and everything else.

00:57:28

Editing the placeholder... so I think as the tasks progressed

00:57:32

The basic tasks progressed.

00:57:35

as you had to change, right? Some part of the code you ended up reading,

00:57:39

Understanding a little more about how it

00:57:41

worked, because you, looking at first,

00:57:43

you think one thing and well, most things really

00:57:47

confirmed, right? Of the functioning, but

00:57:49

It helped, for sure, to understand, how the communications really worked here

00:57:56

Researcher

And if you had the same tasks,

00:58:00

but didn't have access to copilot or a similar tool, how would you,

00:58:07

solve it?

00:58:10

Participant

I would probably research. What would the approach be?

00:58:11

Researcher

Yes, the approach.

00:58:15

Participant

I think research.

00:58:17

Especially in my context since it's been a while

00:58:20

since I've seen some frontend things,

00:58:23

so I think I need a bit more on basic state usage.

00:58:27

of reducer. of how to handle that, right?

00:58:29

A refresh.

00:58:31

Well, I think also a debug, right?

00:58:34

I'd probably put or use a debug tool here from

00:58:39

VSCode itself. If that wasn't possible

00:58:43

We'd have to resort to a type of

00:58:44

Researcher

ConsoleLog.

00:58:46

Participant

Putting it here and seeing what happened.

00:58:50

Yeah, but I think that's it, it would be research and

00:58:53

use the debugging available in the IDE.

00:58:53

Researcher

Good.

00:58:59

My last question is, you are, did you feel that your interaction,

00:59:04

the way you interacted with the copilot over time has changed?

00:59:11

During this session.

00:59:18

Participant

Change, I don't know if it changed.

00:59:22

Maybe like, I. I imagine that maybe if I

00:59:27

had continued, right? For more advanced tasks,

00:59:30

maybe it would change because.

00:59:34

Well, as the tasks get more

00:59:37

complex, the desire gets harder, doesn't it

00:59:41

to explain what you really want, right? To chatGPT.

00:59:46 But I think for more basic tasks I think there wasn't any response here
00:59:49 or actually, I think there was. I don't know if I can look in the
00:59:52 history here, but anyway.
00:59:54 This here is already the history. This goes to.
01:00:00 **Researcher**
Yes.
01:00:05 **Participant**
sometimes it changed and I'm forgetting. Is that it?
01:00:08 There were some suggestions he gave here that I don't know, some hypotheses that
01:00:12 really some confirmed ones don't make much sense. I think that
01:00:17 how to interact hasn't changed. What happened here was that the fix he
01:00:21 suggested didn't fully solve the problem, right? Yeah,
01:00:25 but then I told him, although I think this can
01:00:28 be considered a change in a way, right? Because.
01:00:31 Before, I was explaining the problem, and then there was only one time that I
01:00:34 interacted with the previous response he gave, right? Which was in this case?
01:00:38 **Researcher**
Ah, yes, I understand. Aha.
01:00:38 **Participant**
That I asked, then he answered, and then
01:00:40 it was really a conversation, right? It wasn't a question, an answer,
01:00:44 like it was Google, right? You don't talk to Google,
01:00:45 **Researcher**
Uh-huh.
01:00:48 **Participant**
but you talk with the copilot. So I think this
01:00:51 was the only real moment of conversation, right?
01:00:53 That there was a change in the paradigm of question and answer.