

Participant P10

00:07:01 **Participant**
I can do a test with the copilot just
00:07:05 to see if
00:07:05 **Researcher**
Sure, sure, you can use it however you want.
00:07:10 **Participant**
I'm going to open a chat with it here.
00:07:13 so just commenting on how I might talk
00:07:17 to someone who is... some peer who is with me.
00:07:21 I've never used the copilot much in my work,
00:07:26 so I'm also discovering a bit how it works.
00:07:30 So I think the first thing I would do is see if it can understand...
00:07:36 this project? So I would
00:07:43 try to ask it to describe what this project is just to see if...
00:07:49 anyway, just to give me an overview of the
00:07:51 project to see if I can get some idea that might also help in the future and
00:07:57 also to see how assertive it
00:07:59 is regarding the code that is,
00:08:03 so, to validate if it is giving me a coherent or hallucinated response.
00:08:20 I'm thinking about the best way to describe it to it.
00:08:34 and then I send it to it.
00:08:35 **Chat (window: 0)**
> Agent
github.copilot.workspace
> Context
workspace
> Message
@workspace can you tell me what is this project about?
> Response
This project is a **TodoMVC application** built using **React**. It allows users to manage a list of tasks (todos) with features like adding, updating, toggling completion, and removing tasks. The project also includes pagination controls, filtering (all, active, completed), and persistent state storage using [localStorage\(src/todo/reducer.js\)](#).

Key Features:

1. Task Management:

- Add new tasks.
- Edit existing tasks.
- Mark tasks as completed or active.
- Remove individual or all tasks.
- Clear completed tasks.

2. Filtering:

- View all tasks.
- Filter tasks by active or completed status.

3. Pagination:

- Navigate through tasks using pagination controls.

4. State Persistence:

- Saves the state of tasks in [localStorage](#) ([node_modules/typescript/lib/lib.dom.d.ts](#)) to persist data across page reloads.

5. Responsive UI:

- Styled using CSS with accessibility features (e.g., visually hidden elements for screen readers).

6. Routing:

- Uses react-router-dom for navigation between different task filters.

Development Setup:

- **Webpack** is used for bundling the application.
- **Babel** is configured to transpile modern JavaScript and JSX.
- **ESLint** is set up for linting with React-specific rules.

How to Run:

- Use `npm run dev` to start the development server with hot reloading.

This project is a React-based implementation of the classic TodoMVC app, showcasing best practices in React development, state management, and component-based architecture.

00:08:42
- 00:08:48

> Response rendering

00:08:55

Participant

Well, so it's telling me that the

00:08:57

application was made in react. Also, commenting that it allows users to manage a task list.

00:09:03

00:09:07

With features of editing, updating

00:09:10

00:09:15

Anyway, looking at it from the perspective of someone who hasn't read any of the code and who was watching, I would almost say it's an idea of a

00:09:20

00:09:23 jira a trello. An idea more or less like that.
00:09:26 so that's what I'm interpreting in what he
00:09:31 is commenting.
00:09:34 But that's not all he says there is
00:09:38 Pagination control. I think that's the name, "filtering".
00:09:42 You have the storage.
00:09:46 I'm not going to read with such detail,
00:09:51 but it was more for me to see what if he can actually tell me,
00:09:56 what is happening.
00:10:00 And he even gives me a bit of the way to start the project.
00:10:04 I wanted more to make sure he was giving me answers consistent with what
00:10:10 the project is.
00:10:13 I'll go back here to read the tasks to understand.
00:10:19 "When there is no task" ... the practical task A, change the color when there are no tasks,
00:10:24 **Researcher**
Sorry just to, just to, just to make it clear, right?
00:10:25 **Participant**
appears, uh-huh.
00:10:27 **Researcher**
These are the very initial tasks for you
00:10:30 ... your first steps and as soon as you complete them, I'll send you the other 3 tasks.
00:10:36 It's another PDF.
00:10:38 **Participant**
Alright, cool. "When there are no tasks, a red box appears with the message"
00:10:44 "none to do here yet change the color of this box to blue verification,"
00:10:49 "update the application as confirm that the message none to do here yet"
00:10:54 "appears inside a blue box." Cool, so now
00:11:02 I'm going to leave the... I'm going to ignore this message to configure VScode. I'll leave
00:11:09 it as it is and let's play a little with the code.
00:11:16 > **Viewing file** (public/index.html)
00:11:16 **Participant**
I'm seeing what's a little in each
00:11:19 file, in each folder just to see what
00:11:23 there really is. Although I know that the file, the heart of the project really,
00:11:27 the source is in this folder. But it was just to see a little of
00:11:31 what is
00:11:34 configured
00:11:38 the project as a whole.
00:11:40 > **Viewing file** (src/index.js)
00:11:41 **Participant**
So, I'm just going to take a look.
00:11:48 And well, I'm not a front-end either,
00:11:52 so it's also not something I'm so used to,
00:11:55 > **Viewing file** (src/todo/reducer.js)
00:11:56 **Participant**
but it's just to let you know.
00:12:03 Let me see.
00:12:10 Just seeing a little, what's a little in the files...
00:12:11 > **Viewing file** (src/todo/constants.js)
00:12:13 > **Viewing file** (src/todo/app.jsx)
00:12:16 > **Viewing file** (src/todo/app.css)
00:12:28 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:12:38 > **Viewing file** (src/todo/components/main.jsx)
00:12:41 > **Viewing file** (src/todo/components/item.jsx)
00:12:43 > **Viewing file** (src/todo/components/input.jsx)

00:12:47

Participant

Alright, well, so I have to find... how was it again?

00:12:51

It was to see a red box with the message "none to do here yet"

00:12:56

change this color to make the box blue.

00:13:03

How would I do it? before executing anything....

00:13:07

the way I would do it is to search the message.

00:13:14

Is it in English or Portuguese? "There is nothing to do here."

00:13:18

Is the message in Portuguese or not?

00:13:19

Researcher

Yes.

00:13:20

It's in Portuguese, exactly as it is there.

00:13:28

Participant

None to do...

00:13:32

None to do... So it's here.

00:13:34

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

00:13:41

Participant

Now let's see...

00:13:45

Well, let's use our copilot to know

00:13:50

how to change

00:13:53

the color of the

00:14:05

container. From what I understand...

00:14:07

let me see if it's...

00:14:14

Change container agreement

00:14:33

todo-list container, class color.

00:14:34

Chat (window: 0)

> Agent

github.copilot.vscod

> Context

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

> Message

How to change the todo-list-empty-container class color?

> Response

Use the **workbench.colorCustomizations** setting to override the color of the todo-list-empty-container class.

```
{
  "workbench.colorCustomizations": "{ \"todo-list-empty-container\":
  \"#FF0000\" }"
}
```

Ensure that the todo-list-empty-container class is defined in your CSS or theme file and is properly scoped to your project. If the color is not applied, check for conflicting styles in your CSS or other theme settings.

00:14:40

Participant

It's bringing, I had to put the workspace together

00:14:44

for it to get the files...

00:14:49

that maybe it has to access to change the color.

00:14:53

Let's see if it responds with something

00:14:58
- 00:14:58

> Response rendering

00:14:58

Participant

that gives me an idea.

00:15:32

look, it's telling me...

00:15:37

todo-list-empty-container class

00:15:41

ensure that the todo-list... class is defined in your CSS or theme file and is

00:15:48

properly scoped to your project. Well, I wanted it to have given me...

00:15:56

maybe the file that

00:16:01

this class is defined in.

00:16:03 So I can see if it's within this class that I see this
00:16:08 I'm not going to use it.
00:16:09 I'm going to search here to see where I'm going to use in the search tab... of
00:16:13 > **Viewing file** (src/todo/app.css) through file search
00:16:14 **Participant**
search... to see where it's defined.
00:16:21 So I see here that the container doesn't mention the color.
00:16:24 > **Viewing file** (src/todo/components/main.jsx) through file search
00:16:25 **Participant**
So, maybe I'll look at the message one, but then I think it would be your message's
00:16:30 color.
00:16:38 > **Viewing file** (src/todo/app.css) through file search
00:16:44 **Participant**
Ah, okay. So maybe it's the color... the background
00:16:48 Change the color of this box to blue.
00:17:06 I'm assuming it's the background color of the message.
00:17:10 > **Editing file** (src/todo/app.css)
00:17:20 **Participant**
A question: can I run the application here too?
00:17:23 **Researcher**
You have to run the application, yes, it's as if it's in a state as if you
00:17:28 had just cloned the project.
00:17:31 So you have to do the whole process.
00:17:39 **Participant**
Let's go.
00:17:51 I'll ask him what commands I should... ask the
00:17:55 copilot, what commands I should
00:17:57 execute to wake up the application.
00:18:11 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, viewport (src/todo/app.css)
> **Message**
@workspace what are the commands I should use to run the app?
> **Response**
To run the app in development mode, use the following command:

npm run dev

This will start the development server using webpack-dev-server and open the app in
your default browser.
00:18:20 > **Viewing file** (webpack.common.js)
00:18:22 > **Response rendering**
- 00:18:25
00:18:23 > **Viewing file** (webpack.dev.js)
00:18:27 > **Viewing file** (package-lock.json)
00:18:36 **Participant**
So I'll follow what the copilot told me. Developer mode.
00:18:57 > **Viewing file** (webpack.dev.js)
00:19:10 **Participant**
There was some error here... let's see. I'll copy the response and ask for his help.
00:19:19 **Researcher**
I think you needed to accept something
00:19:24 he said to be able to paste and copy.
00:19:27 But I think everything went well.

00:19:50

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (webpack.dev.js)

> Message

@workspace I got this error while trying to run the app: > todomvc-react@1.0.0 dev > webpack serve --open --config webpack.dev.js sh: 1: webpack: not found

> Response

The error indicates that the webpack command is not available, likely because the dependencies have not been installed yet. To resolve this, run the following command to install all the required dependencies:

```
npm install
```

After the installation is complete, try running the app again with:

```
``sh npm run dev
```

00:19:55

Participant

Ask him what might have gone wrong.

00:20:01
- 00:20:01

> Response rendering

00:20:09

Participant

it's because the dependencies weren't installed, so I need to install them.

00:20:39

I'll read the other tasks while it's running there.

00:20:44

Researcher

OK.

00:20:49

Participant

Practical task B, change the placeholder. The placeholder text in the input field currently displays the question "what needs to be done?" change it to "type a task here."

00:20:53

00:20:57

00:21:02

00:21:04

00:21:07

00:21:10

00:21:13

00:21:19

00:21:23

00:21:38

00:21:46

00:21:58

00:22:05

00:22:10

00:22:27

00:22:34

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

Participant

"what needs to be done?" to "type a task here"

00:22:37
- 00:22:41

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

00:22:46

Participant

Was it saved? Do I need to... is there hot reload?

00:22:49

I don't know if I need to... I'll ask him.

00:22:57

I'll ask the copilot

00:23:00

just out of curiosity, to understand

00:23:07

if this application has hot reload, and what needs to be done,

00:23:14

if it's just to refresh the page

00:23:21

to have the changes. It was to translate

00:23:27

now I need to bring it to English. I think a decision... something I didn't mention but

00:23:33

I'm speaking English with copilot, because I think maybe the questions...

00:23:40
00:23:43
00:23:47
00:23:50
00:23:54
00:24:09
00:24:21
00:24:26
00:24:46

well, I understand that
it was better trained in an English environment than in Portuguese,
so many times I do it in English to get answers,
maybe a little more accurate than doing it in Portuguese.

So that's a decision I make.

I'm asking if this project has hot reload.

Asking what I should do so that given
a change, see if it has already gone to the page.

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

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

> Message

@workspace is this project support hotreload? What should I do to see if my changes
are already in the page?

> Response

Yes, this project supports hot reloading because it uses webpack-dev-server with the
[devServer \(webpack.dev.js\)](#) configuration in [webpack.dev.js](#). The webpack serve
command automatically reloads the page when changes are made to the source files.

Steps to See Your Changes:

1. Start the development server using:

```
npm run dev
```

2. Make changes to your source files (e.g., header.jsx).

3. Save the file. The browser should automatically reload the page to reflect your
changes.

If the page does not reload automatically, ensure the development server is running and
check the browser console for errors.

00:24:52
- 00:24:56

> Response rendering

00:25:11

Participant

So I'll press F5 here to see if...

00:25:16

That's it, so okay, I need to see here what he commented a

00:25:21

little, that in theory it already has hotreload, so, in fact,

00:25:26

just pressing F5 already makes the change, so task B is done now...

00:25:32

task C to sort tasks currently already tasks they are

00:25:37

displayed in the order they were added, modify the application so that the

00:25:40

tasks are sorted alphabetically.

00:25:43

Well, just out of curiosity

00:25:45

I'm going to test the application here.

00:25:55

how to add?

00:25:56

Researcher

Enter.

00:26:09

Participant

So let's see how to add alphabetically.

00:26:20

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

00:26:20

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

00:26:28

> Viewing file (src/index.js)

00:26:32

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

00:26:34

Participant

I'm first trying... just to

00:26:35

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

00:26:36

Participant

00:26:41 explain what I'm thinking. I'm trying to see if I can find manually,
00:26:45 the place of the logic for this.
00:26:45 > **Viewing file** (src/index.js)
00:26:45 **Participant**
It's more in this way that I
00:26:49 am doing. And if I don't find it, then I'll talk to the copilot to
00:26:51 > **Viewing file** (src/todo/reducer.js)
00:26:56 **Participant**
help me.
00:27:04 Yeah, I just found it strange that VScode tells me there's something
00:27:09 wrong here, but I'll ignore it for now.
00:27:13 > **Viewing file** (src/todo/constants.js)
00:27:17 > **Viewing file** (src/todo/reducer.js)
00:27:20 **Participant**
And then I'll see that there.
00:27:21 > **Viewing file** (src/todo/app.jsx)
00:27:26 > **Viewing file** (src/todo/app.css)
00:27:28 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:27:35 **Participant**
I'll ask him.
00:27:37 > **Viewing file** (src/index.js)
00:27:41 **Participant**
I'll ask...
00:27:50 where... first, I'll ask for the location...
00:27:54 of the file, maybe?
00:28:00 This file where I can make the logic change.
00:28:13 Well, it's even hard to think... but it's the insertion logic
00:28:18 of tasks. But I think he won't understand if I
00:28:22 say tasks.
00:28:47 Let's see if he understands my question, about which file
00:28:55 deals with the display of the inputs I will insert.
00:29:00 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, viewport (src/index.js)
> **Message**
@workspace what is the file that handle the display of the input values?
> **Response**
The file that handles the display and submission of input values is
<src/todo/components/input.jsx>.

The [Input\(src/todo/components/input.jsx\)](src/todo/components/input.jsx) component in this file is responsible
for rendering the input field and handling user interactions, such as submitting the input
value when the "Enter" key is pressed.
00:29:03 **Participant**
Let's see if he understands me.
00:29:06 > **Response rendering**
- 00:29:07
00:29:14 > **Viewing file** (src/todo/components/header.jsx) through Copilot link
00:29:20 **Participant**
Ah, great, I found it. He gave me...
00:29:24 > **Viewing file** (src/todo/components/input.jsx)
00:29:49 **Participant**
rendering the input field and handling user interactions, such as submitting the input value

00:29:55 when the "Enter" key is pressed.

00:30:17 Okay, I think here he takes the

00:30:22 value, which is supposed to be... If it took, hold on...

00:30:27 I'm trying to understand the logic of... this is where he takes...

00:30:41 I pressed enter... He will take the value that is there and will

00:30:47 store it here.

00:31:08 **> Viewing file** (src/todo/components/header.jsx)

00:31:16 **Participant**

Let me just see here.

00:31:27 **> Viewing file** (src/todo/components/item.jsx)

00:31:42 **Participant**

Let me just take a look at... I'm trying...

00:31:45 I understood what he told me here that maybe it's this part,

00:31:46 **> Viewing file** (src/todo/components/header.jsx)

00:31:49 **Participant**

where he deals with the input, but he's...

00:31:53 It's just that, actually... no, not here. This one is storing what are the

00:31:55 **> Viewing file** (src/todo/components/input.jsx)

00:31:59 **Participant**

input values.

00:32:03 **> Viewing file** (src/todo/components/header.jsx)

00:32:03 **Participant**

Now I want to see where it is

00:32:08 where it is done the...

00:32:13 **> Viewing file** (src/todo/components/input.jsx)

00:32:14 **Participant**

Wait a minute, I'm trying to understand, here he presses enter...

00:32:19 so is it in this place that it gives the... It would be good to ask him if in this...

00:32:25 would be where the input values will appear.

00:32:31 It's a...

00:32:36 "Which function is it that...

00:32:43 show

00:32:46 the input works?"

00:32:49 **Chat** (window: 0)

> Agent

github.copilot.workspace

> Context

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

> Message

@workspace what is the function that shows the input values?

> Response

The function that handles and shows the input values is the [onSubmit \(src/todo/components/header.jsx\)](#) callback passed as a prop to the [Input \(src/todo/components/input.jsx\)](#) component in [src/todo/components/input.jsx](#).

This function is triggered when the "Enter" key is pressed, and it processes the trimmed input value if it meets the minimum length requirement ([hasValidMin \(src/todo/components/input.jsx\)](#)). The [onSubmit \(src/todo/components/header.jsx\)](#) function is defined by the parent component, such as in [src/todo/components/header.jsx](#), where it dispatches an action to add a new todo item.

> Response rendering

00:32:58 **Participant**

– 00:33:02 Well, I'll try to ask him if...

00:33:09

00:33:21
00:33:32

If I want to sort the input values, how would I do it?

Chat (window: 0)

> **Agent**

github.copilot.workspace

> **Context**

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

> **Message**

@workspace if I want to order the input values how would I do?

> **Response**

To order the input values (todos), you can modify the [visibleTodos](#) ([src/todo/components/main.jsx](#)) array in the [Main](#) ([src/todo/components/main.jsx](#)) component. You can use JavaScript's sort method to sort the todos based on a specific property, such as the [title](#) ([src/todo/components/header.jsx](#)) or [id](#) ([node_modules/@types/react/index.d.ts](#)).

Example: Sorting Todos by Title

Update the [visibleTodos](#) ([src/todo/components/main.jsx](#)) definition in [Main](#) ([src/todo/components/main.jsx](#)) to include sorting:

```
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)), // Sort by
title alphabetically
  [todos, route]
);
```

Example: Sorting by Creation Time

If you want to sort by creation time, ensure the [generateUniqueID](#) ([src/todo/reducer.js](#)) in [reducer.js](#) ([src/todo/reducer.js](#)) includes a timestamp, and sort by that timestamp:

```
.sort((a, b) => a.id - b.id) // Sort by ID (assuming ID includes a
timestamp)
```

This will ensure the todos are displayed in the desired order. Save the file, and the changes will reflect automatically if hot reloading is enabled.

00:33:38
- 00:33:43

> **Response rendering**

00:33:49

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

00:34:10

Participant

Here he generates the sort method.

00:34:14

So, how he shows me is...

00:34:24

Okay, let's do this

00:34:26

that he recommended.

00:34:31

So, from what I understand

00:34:33

Visible all. It would be...

00:34:40

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

00:34:46

Participant

00:34:58 Is this correct here?
00:35:23 Let me see then... refresh the page here.
00:35:28 It's right, great.
00:35:30 Ah, I have to do it
00:35:31 as it is here, right?
Researcher
It doesn't have to be exactly, no, no, it's fine. If it's all right for you,
00:35:37 we.
00:35:38 stay a few more minutes... I can send you the second part just
00:35:42 for you to start and then we do the final questions.
00:35:45 It depends on how your time is. If not, we can finish now.
00:35:47 **Participant**
You can, no, I'm fine, you can. I ended up taking
00:35:53 too long. Sorry.
00:35:54 **Researcher**
No, no, no. Not every input we have is valid,
00:35:59 no problem, no! Let me just check this.
00:36:04 **Participant**
I don't know if it contributes, but I think that...
00:36:06 I think that when I do the tasks.
00:36:09 I also go in an attempt to understand and learn what it is.
00:36:15 I'm commenting because since it's not a job interview
00:36:21 I don't need to do it quickly and just pass.
00:36:24 So, I think in the attempt to also learn,
00:36:27 I think I try
00:36:29 to do things with this approach... I don't know,
00:36:32 it's not to justify myself, but maybe to give input for your
00:36:38 research.
00:36:39 **Researcher**
No. Yes, yes, you can keep talking about what you are ...
00:36:42 your impression and how you are... The experience you are having,
00:36:46 you know? This is really nice for us to understand
00:36:49 really.
00:36:49 It's not just what you do, how you use and don't use, but also the reasons, right?
00:36:54 So what you just said is very valid for us to understand.
00:36:59 Very cool. I sent a link there, if you can,
00:37:03 and then we just start and do one of the tasks and then we
00:37:09 discuss how it went. If that's all right with you.
00:37:12 **Participant**
Great.
00:37:15 Uh-huh. Yes, I opened it here, let me see.
00:37:17 **Researcher**
Another thing, if you want to close the first... the
00:37:21 steps, just to avoid confusion.
00:37:29 **Participant**
So, tasks, implement the following requirements,
00:37:32 in the order you consider most appropriate and make sure that all verification steps
00:37:37 are successful. Task A persist. Currently
00:37:41 all to-dos disappear when the page is reloaded,
00:37:45 modify the application so that the to-dos remain even after the
00:37:50 reload. Ensure that the state of each
00:37:53 to-do is completed or
00:37:54 active is also preserved.
00:37:57 An attempt to implement this functionality was made,
00:38:01 but at the moment it is not working correctly, fix or
00:38:06 reimplement the functionality. Verification, add the to-dos, confirm,

00:38:11 mark some as completed, reload, edit some...

00:38:15 Confirm that the changes were kept. I will read the others just

00:38:20 to see which... Save draft.

00:38:24 If a user reloads the page while typing a new

00:38:28 to-do the text in the field disappears.

00:38:30 Modify the application so that the input field automatically saves drafts

00:38:36 and restores them upon reloading the page and... sorry...

00:38:44 Pagination. Currently, the to-do list application displays

00:38:48 all tasks in a single list. In this task,

00:38:51 you implement pagination to limit

00:38:54 the number of to-dos displayed per page, allowing users to easily navigate

00:38:59 their to-dos.

00:39:02 Well, I'll start with task A.

00:39:04 **Researcher**

00:39:08 Again, the intention is this. The intention is not to finish all the

00:39:12 it's just for us... It's that they increase in

00:39:17 complexity. But you can choose one of them and start the

00:39:19 implementation.

Participant

Yes, cool.

00:39:24 Cool, let's do this task A.

00:39:29 And now let's see... there was an attempt to

00:39:35 implement this functionality.

00:39:40 I'm going, by... thinking of being a little faster, I'll... Not for us here,

00:39:45 but also to simply try to solve the task.

00:39:49 but also to simply try to solve the task.

00:39:57 I'll ask him how he would persist the input values

00:40:03 of input.

00:40:26 **Chat** (window: 0)

> Agent

github.copilot.workspace

> Context

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

> Message

@workspace how would I do if I want to persist the input values even if I refresh the page?

> Response

To persist the input values (todos) even after refreshing the page, you can use [localStorage \(node_modules/@types/node/globals.d.ts\)](#) to save and load the todos. The project already has a mechanism for this in [src/todo/reducer.js](#) with the [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) functions. Here's how you can ensure persistence:

Steps to Persist Todos:

1. **Load Todos on Initialization:** The [initialState \(src/todo/reducer.js\)](#) in [src/todo/reducer.js](#) already uses the [loadState \(src/todo/reducer.js\)](#) function to load todos from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) when the app starts:

```
export const initialState = loadState();
```
2. **Save Todos on State Change:** The [saveState \(src/todo/reducer.js\)](#) function is already called in the [todoReducer \(src/todo/reducer.js\)](#) after every state change:

```
const saveState = (state) => {
  localStorage.setItem("todos", JSON.stringify(state));
};
```

3. Ensure Correct Key in `localStorage`

([node_modules/@types/node/globals.d.ts](#)): Update the `loadState` ([src/todo/reducer.js](#)) function to use the correct key (`todos` ([src/todo/components/main.jsx](#))) to match the `saveState` ([src/todo/reducer.js](#)) function:

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

4. Verify Integration: The `useReducer`

([node_modules/@types/react/index.d.ts](#)) hook in [src/todo/app.jsx](#) already initializes the state with `initialState` ([src/todo/reducer.js](#)):

```
const [todos, dispatch] = useReducer(todoReducer, []);
```

Result:

With these mechanisms in place, todos will automatically persist in `localStorage` ([node_modules/@types/node/globals.d.ts](#)) and reload when the page is refreshed. You don't need to make additional changes unless you want to customize the behavior further.

00:40:30

Participant

So I asked him here, how I would persist the

00:40:35
- 00:40:51

> Response rendering

00:40:35

Participant

input values even if I refreshed the page.

00:40:43

There's one here, called local storage.

00:40:49

> **Viewing file** (`node_modules/%40types/node/globals.d.ts`) through Copilot link

00:41:00

Participant

it's a native method

00:41:05

load the todos. The project already has a mechanism for this.

00:41:09

So he identifies that there's this method with load state.

00:41:11

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

00:41:14

Participant

ah, it was the one I saw that is wrong.

00:41:18

Well, where did I see that there was one that...?

00:41:23

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

00:41:24

Participant

Well, is it just this here?

00:41:32

State?

00:41:37

There's a save state.

00:41:54

He loads.

00:42:02

To save, okay.

00:42:09

I think here it is not being

00:42:12

called, right?

00:42:13

The save state... let's see if he identifies this.

00:42:18

Or am I not...

00:42:22

Let me see this here... Save all on state change. The savestate function is already called in the todoReducer

00:42:35

where is it called?

00:42:39

Ah here.

00:42:43

Okay.

00:42:56

Let me see.

00:43:04

ensure correct key in local storage.

00:43:05

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

00:43:09

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

00:43:09

Participant

Update the load state function to use the correct key.

00:43:21

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

00:43:21

Participant

Is it this here?

00:43:25

So?

00:43:28

verify integration, use reducer hook in app.

00:43:33

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

00:43:36

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

00:43:37

Participant

[not audible]

00:43:44

verify integration.

00:43:59

use reducer...

00:44:38

So not yet...

00:44:51

I'll ask... I thought it was just changing those...

00:44:59

those little errors that were there, and it would be enough, but maybe not...

00:45:02

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

00:45:17

Participant

What I was going to tell him is that maybe I thought I had already done it, had already corrected.

00:45:21

If it were just those little things.

00:45:24

But let me double check here, on what he...

00:45:29

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

00:45:38

Participant

local storage ...

00:45:39

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

00:45:42

Participant

initial state, loadstate function to load todos from localStorage

00:45:56

save todos on state change. The savestate function is already called in the todo reducer after every state change.

00:46:03

ensure correct key, update loadstate, function to use the correct key...

00:46:08

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

00:46:16

Participant

the correct key to match the savestate function. Ah, could it be that?

00:46:32

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

00:46:33

Participant

00:46:47

00:46:54

I'll ask him to explain better...

00:46:59

the third item

00:47:03

to see if that's it.

Chat (window: 0)

> **Agent**

github.copilot.workspace

> **Context**

workspace, viewport (src/todo/reducer.js)

> **Message**

@workspace Explain better to me the third item 'Ensure Correct Key'

> **Response**

The third item, "**Ensure Correct Key in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#))**", refers to making sure that the key used to store and retrieve data in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) is consistent. In this project, the key being used is "todos". This key is used in both the [loadState](#) ([src/todo/reducer.js](#)) and [saveState](#) ([src/todo/reducer.js](#)) functions to interact with [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)).

Why is this important?

If the key used in [saveState](#) ([src/todo/reducer.js](#)) (to store data) and [loadState](#) ([src/todo/reducer.js](#)) (to retrieve data) is not the same, the app will not be able to load the saved todos correctly. For example: - If [saveState](#) ([src/todo/reducer.js](#)) stores data under the key "todos", but [loadState](#) ([src/todo/reducer.js](#)) tries to retrieve data using a different key (e.g., "tasks"), the app will not find the saved data.

How does this project handle it?

In this project: - The [saveState](#) ([src/todo/reducer.js](#)) function stores the todos in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) under the key "todos":

```
const saveState = (state) => {  
  localStorage.setItem("todos", JSON.stringify(state));  
};
```

- The [loadState](#) ([src/todo/reducer.js](#)) function retrieves the todos from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) using the same key "todos":

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

What should you ensure?

You should ensure that both functions consistently use the same key ("todos") when interacting with [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)). If you ever decide to change the key (e.g., to "myTodos"), you must update it in both [saveState](#) ([src/todo/reducer.js](#)) and [loadState](#) ([src/todo/reducer.js](#)) to avoid breaking the persistence functionality.

> **Response rendering**

Participant

The key being used is todos. This key is used in both the loadstate and savestate.
I'll talk to him because it seems to me that I did what he asked me,

00:47:14
- 00:47:22

00:47:36

00:49:26

00:49:32
00:49:36
00:49:41
00:49:54
00:49:57
00:50:27
00:50:31
00:50:45

but I don't know
where I'm failing to ensure
that it's being called with the same key.
Let me just redo it here to see if indeed it's not working.
I don't think so.
So I'm saying that it seems to me that they are using the same key,
but still it's not working.

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (src/todo/reducer.js)

> Message

@workspace It seems to me that the loadState and saveState are using the same key,
but still the persistence doesn't work

> Response

If the [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) functions are using the same key ("todos") but persistence still doesn't work, there are a few potential issues to investigate:

1. Initial State in [App \(src/todo/app.jsx\)](#) Component

The [App \(src/todo/app.jsx\)](#) component in [src/todo/app.jsx](#) initializes the [useReducer \(node_modules/@types/react/index.d.ts\)](#) hook with an empty array ([]) instead of the [initialState \(src/todo/reducer.js\)](#) from the reducer. This means the app is not loading the persisted state from [localStorage \(node_modules/@types/node/globals.d.ts\)](#).

Fix:

Update the [useReducer \(node_modules/@types/react/index.d.ts\)](#) initialization in [src/todo/app.jsx](#) to use [initialState \(src/todo/reducer.js\)](#):

```
File: src/todo/app.jsx  
const [todos, dispatch] = useReducer(todoReducer, initialState);
```

2. [localStorage \(node_modules/@types/node/globals.d.ts\)](#)

Availability

Ensure that [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is available in the browser. If you're running the app in an environment where [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is not supported (e.g., certain server-side rendering setups), the [loadState \(src/todo/reducer.js\)](#) function will always return an empty array.

Debug:

Add a console log to check if [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is being accessed correctly:

```
File: src/todo/reducer.js  
export const loadState = () => {  
  console.log("Loading state from localStorage...");  
  const state = localStorage.getItem("todos");  
  return state ? JSON.parse(state) : [];  
};
```

3. State Saving on Every Change

The [saveState \(src/todo/reducer.js\)](#) function is already being called after every state change in the [todoReducer \(src/todo/reducer.js\)](#). However, if the reducer throws an error or the app crashes before [saveState \(src/todo/reducer.js\)](#) is called, the state won't be saved.

Debug:

Add a console log to verify that [saveState \(src/todo/reducer.js\)](#) is being called:

```
File: src/todo/reducer.js
const saveState = (state) => {
  console.log("Saving state to localStorage...", state);
  localStorage.setItem("todos", JSON.stringify(state));
};
```

4. Browser-Specific Issues

Some browsers (or browser settings) may block [localStorage \(node_modules/@types/node/globals.d.ts\)](#) usage, especially in private/incognito mode. Test the app in a standard browser session to rule out this issue.

5. Clearing [localStorage \(node_modules/@types/node/globals.d.ts\)](#)

If the app previously saved invalid or corrupted data to [localStorage \(node_modules/@types/node/globals.d.ts\)](#), it may fail to load correctly. Clear the [localStorage \(node_modules/@types/node/globals.d.ts\)](#) and try again:

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

Summary of Fixes:

1. Update the [useReducer \(node_modules/@types/react/index.d.ts\)](#) initialization in [src/todo/app.jsx](#) to use [initialState \(src/todo/reducer.js\)](#).
 2. Add debugging logs to [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) to verify their behavior.
 3. Test in a standard browser session to rule out [localStorage \(node_modules/@types/node/globals.d.ts\)](#) restrictions.
- After applying these fixes, the persistence should work as expected.

00:50:54
- 00:51:06

> Response rendering

00:51:00

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

00:51:12

> Viewing file (src/todo/app.jsx) through Copilot link

00:51:38

Participant

Now it makes sense.

00:51:41
- 00:51:47

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

00:52:00

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

00:52:05

> Viewing file (src/todo/app.jsx) through Copilot link

00:52:12

Participant

Let's see if it's working, so I think it's done.

00:52:21

Good, great.

00:52:35

Yes, I think so.

01:02:49

Researcher

Great, thank you very much. I just have a few questions for you.

01:02:49

Participant

Ready.

01:02:53

Researcher

We'll finish soon, sorry for taking more time

01:03:02

Were you able to orient yourself in the code, how do you see that the copilot

01:03:09

influenced you in

01:03:13

navigating through the code?

01:03:17

Participant

Yeah, it was very useful.

01:03:19

I see that without it I would have had much

01:03:24

more difficulty.

01:03:27

Especially for not knowing so much about a front-end architecture.

01:03:37

It gave me the ability to navigate with great ease.

01:03:42

Researcher

And do you feel you were able to understand the code with the tasks you performed?

01:03:47

01:03:49

Participant

I did, yes.

01:03:51

Researcher

OK, how would you have solved the tasks

01:03:55

without the copilot?

01:03:58

Participant

That's a good question. I think that

01:04:05

much would be by asking on Google, looking at documentation.

01:04:09

So, they would be more direct questions on how I,

01:04:13

for example, how do I run the code. I think that would be quite direct and

01:04:19

maybe I would know how to solve it.

01:04:22

Maybe I

01:04:26

would also be following a lot through the shortcuts

01:04:31

of the IDE. In this case, VS code, being able to...

01:04:37

Ah this class is called here, that one is there.

01:04:40

So I would be here following a tree

01:04:43

Then I get there more or less.

01:04:47

So, it would be more or less like that.

01:04:54

Researcher

And how did the copilot change your strategy or influence your strategy in solving the tasks?

01:05:00

01:05:09

Participant

Maybe like this, by not knowing so much how

01:05:14

to navigate the project

01:05:19

I would certainly have been very lost on where to go.

01:05:24

So I would be like "where is this, in the footer? in the header?"

01:05:29

Where is it? I would be navigating until I found it.

01:05:33

and the copilot..

01:05:45

the point is that

01:05:48

maybe I didn't have a strategy at the beginning.

01:05:53

and the copilot gave me insights, it provided me

01:06:02

information so that I could create a strategy. So, I ask

01:06:07

what are the possible errors for, for example, for data persistence.

01:06:13

It gave me a whole step-by-step, I was seeing.

01:06:17

Of course, when I think I was navigating the code,

01:06:19

I realized there were some parts that VS code itself

01:06:22

indicated were wrong.

01:06:23

So that also gave me confidence. Maybe that's the way.

01:06:28

So a certain strategy would be on this side of the IDE itself.

01:06:34

Then the copilot allowed me to be more assertive

01:06:39

where to look for

01:06:41

the errors or where to make the changes.

01:06:48

Researcher

Two little questions, one more is, you already mentioned how the copilot influenced your strategy, right?

01:06:54

01:06:56

And if you looked back and tried to see?

01:06:59

How your interaction with the copilot, both the questions you asked and the frequency of questions, anyway, changed over the tasks over

01:07:04

time? how do you see this happening?

01:07:09

01:07:15

Participant

Well, I was interacting more with it over the course of our session here.

01:07:19

01:07:23

I didn't, as I said, hadn't used it, so I didn't

01:07:29

know very well either.

01:07:31

So it was nice because I started kind of in doubt about how

01:07:38

functional it would be and realizing it was

01:07:42

being good, that it was giving me good answers,

01:07:46

that it was indeed useful... Then I said, "Ah, I'll trust it,"

01:07:51

So, I think there was this curve of

01:07:54

trust. The more reliable

01:07:58

what it is telling me, the more I will use it.

01:08:04

Researcher

And there in, in that last questionnaire, there were some aspects, right?

01:08:11

About the answers, the length of the answer, how clear, concise, anyway.

01:08:18

And then you say if you liked it or not.

01:08:22

Could you highlight what caught your attention?

01:08:25

Which of these aspects caught your attention about the copilot's answers?

01:08:32

Participant

I was surprised, in terms of...

01:08:37

it having like hyperlinks to the code,

01:08:43

to the file.

01:08:45

So that caught my attention a lot. I find it very easy,

01:08:48

it gives me, I click, it already goes to

01:08:52

the file I need. So that caught my attention a lot.

01:08:59

It managed to give me the answers like a step.

01:09:04

step by step. It gave me a step by step without me asking for a step by step,

01:09:08

I just said, "Ah,

01:09:13

I'm having a problem here, help me." And then it.

01:09:16

Gave me a step by step. That is definitely very...

01:09:21

It is giving me how I can solve it.

01:09:25

So I think I was very surprised by the way it structured. It's not a text,

01:09:32

it's not a single prose.

01:09:33

it's not a continuous text, but it gave me

01:09:36

it structured the form of the text in items. That for me, at least.

01:09:42

helps a lot to see what is important and what is not

01:09:48

important or what I have already seen. I don't need to see. So I think that's it.