

Participant P21

00:01:25 **Researcher**
You can start

00:02:15 **Participant**
OK?

00:04:20 let me understand.
00:04:23 this project is an app written in JavaScript

00:04:32 **Researcher**
yes.
00:04:33 This function can execute the project.

00:05:05 **Participant**
Change the background color of the text
00:05:18 > **Viewing file** (src/todo/components/main.jsx)

00:05:35 **Participant**
I'll look for where this is being used
00:05:41 class name.
00:06:14 > **Viewing file** (src/todo/app.css)

00:06:17 > **Viewing file** (src/todo/components/main.jsx)

00:06:18 **Participant**
Here is the class name, this one here.
00:06:21 > **Viewing file** (src/todo/app.css)

00:06:23 > **Editing file** (src/todo/app.css)
- 00:06:37

00:06:25 **Participant**
I'll change the color

00:06:34 **Researcher**
OK

00:06:45 **Participant**
correct?

00:06:46 **Researcher**
yes, go to task b.

00:06:48 **Participant**
Change of.
00:06:53 placeholder change. Change to this one here.
00:07:07 I'll copy to find it.

00:07:08 > **Viewing file** (src/todo/components/header.jsx)

00:07:10 **Researcher**
ok.

00:07:11 **Participant**
placeholder change to 'type a task'.
00:07:18 > **Editing file** (src/todo/components/header.jsx)
- 00:07:19

00:07:28 **Participant**
placeholder?

00:07:30 **Researcher**
yes

00:07:35 **Participant**
sorting task now
00:07:39 Modify the app so that tasks are entered in alphabetical order.

00:08:19 > **Viewing file** (src/todo/components/input.jsx)

00:08:34 > **Viewing file** (src/todo/components/item.jsx)

00:08:59 > **Viewing file** (src/todo/app.jsx)

00:09:06 > **Viewing file** (src/todo/components/header.jsx)
00:09:06 > **Viewing file** (src/todo/components/footer.jsx)
00:09:15 > **Viewing file** (src/todo/components/item.jsx)
00:09:21 **Researcher**
Remembering that you can use the copilot whenever you want, okay?
00:09:25 **Participant**
ok.
00:10:06 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, selection (src/todo/components/item.jsx)
> **Message**
@workspace I want to sort my listing in alphabetical order, currently it is being listed in insertion order
> **Response**
To sort the to-do list in alphabetical order, you can modify the rendering logic in the [Main component \(src/todo/components/main.jsx\)](#). In the section where the visible to-dos are filtered, you can add a sort before rendering them.

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


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


This change uses the sort method with localeCompare to sort the to-dos by title in alphabetical order. Now, the to-dos will be displayed in alphabetical order in the interface.
00:10:12 > **Response rendering**
- 00:10:15
00:10:24 > **Viewing file** (src/todo/components/main.jsx)
00:10:39 > **Implementing Copilot code** using apply button (src/todo/components/main.jsx)
- 00:10:42
00:11:35 **Participant**
O, 123, brown, fox.
00:11:40 **Researcher**
ok.
00:11:41 **Participant**
The sorting is correct I believe
00:11:45 **Researcher**
ok.
00:11:49 **Participant**
This made it easier to find the function that was doing the sorting.
00:11:52 **Researcher**
cool.
00:11:53
00:11:59 Great, cool, in the chat there's the link to the tasks, you can open it and start doing it please

00:12:05 You can do it in the order you think is best. Validate all the requested items for each task before moving on to the next

00:12:27 ask, okay?

00:13:10 **Participant**
Just to explain what I imagined here.

00:13:14 He wants the tasks to remain even with the reload,

00:13:19 so I have to save this somehow.

00:13:21 **Researcher**
correct.

00:13:23 **Participant**
And a global variable or in the localStore which I think is the simplest to do.

00:13:28 **Researcher**

00:13:30 Yes, that's right.

00:14:07 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
codebase
> **Message**
#codebase To avoid data loss, I want to add my data to LocalStorage, so that when I refresh the page, I don't lose my tasks.
> **Response**
To save and load task data in LocalStorage, you can adjust the [todoReducer \(src/todo/reducer.js\)](#) and the main application component to synchronize the data with local storage. Here's how you can do it:

1. Update the [todoReducer \(src/todo/reducer.js\)](#) to save to LocalStorage

The [todoReducer \(src/todo/reducer.js\)](#) already has a [saveState \(src/todo/reducer.js\)](#) function to save data to LocalStorage. Make sure it is being called after each state change.

2. Load the initial state from LocalStorage

In the main component ([App \(src/todo/app.jsx\)](#)), you can initialize the state with the data saved in LocalStorage.

Updated Code:

Update the [todoReducer \(src/todo/reducer.js\)](#)

Ensure that the [saveState \(src/todo/reducer.js\)](#) is being called correctly after each action.

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

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

export const todoReducer = (state, action) => {
```

```

    let newState;
    switch (action.type) {
      case ADD_ITEM:
        newState = state.concat({ id: generateUniqueID(), title:
action.payload.title, completed: false });
        break;
      case UPDATE_ITEM:
        newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, title: action.payload.title } : todo));
        break;
      case REMOVE_ITEM:
        newState = state.filter((todo) => todo.id !==
action.payload.id);
        break;
      case TOGGLE_ITEM:
        newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, completed: !todo.completed } : todo));
        break;
      case REMOVE_ALL_ITEMS:
        newState = [];
        break;
      case TOGGLE_ALL:
        newState = state.map((todo) => (todo.completed !==
action.payload.completed ? { ...todo, completed:
action.payload.completed } : todo));
        break;
      case REMOVE_COMPLETED_ITEMS:
        newState = state.filter((todo) => !todo.completed);
        break;
      case LOAD_STATE:
        newState = action.payload;
        break;
      default:
        throw Error(`Unknown action: ${action.type}`);
    }

    saveState(newState); // Save the updated state to LocalStorage
    return newState;
  };

```

Update the [App component \(src/todo/app.jsx\)](#).

Load the initial state from LocalStorage when initializing the [useReducer \(node_modules/@types/react/index.d.ts\)](#).

```

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

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

import "../app.css";

export function App() {
  const [todos, dispatch] = useReducer(todoReducer, [], loadState);

  // Synchronize the state with LocalStorage when loading the
  application
  useEffect(() => {
    const savedTodos = loadState();
    if (savedTodos.length > 0) {

```

```
        dispatch({ type: "LOAD_STATE", payload: savedTodos });
    }
}, []);
return (
    <>
        <Header dispatch={dispatch} />
        <Main todos={todos} dispatch={dispatch} />
        <Footer todos={todos} dispatch={dispatch} />
    </>
);
}
```

Result

Now, the tasks will be automatically saved to LocalStorage whenever there is a change. When you reload the page, the data will be loaded from LocalStorage and displayed in the interface.

00:14:13
- 00:14:21

> **Response rendering**

00:14:25

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

00:14:27
- 00:14:29

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

00:14:56

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

00:14:56
- 00:15:01

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

00:15:17

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

00:15:19

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

00:15:40

Participant

I'm checking what's being saved.

00:15:44

Researcher

OK

00:15:45

Participant

It seems to me that there is something in conflict

00:15:53

Researcher

ok.

00:15:54

Participant

I'll check, add some tasks to load the

00:15:57

page.

00:16:12

Mark some as completed, reload

00:16:21

These 2, as completed.

00:16:27

Alright, saved. Edit some tasks.

00:16:33

were kept.

00:16:55

Researcher

Alright, you can go to the next ones.

00:16:59

Participant

If the user reloads the page while typing,

00:17:05

The input text disappears

00:17:36

ok

00:18:30

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

codebase

> **Message**

#codebase I want the typed value in the input to be saved, even without adding the task. If the user adds the item, make sure the field is cleared.

> **Response**

To save the typed value in the [input \(node_modules/@types/react/index.d.ts\)](#) even without adding the task, you can use a local state in the [Header component \(src/todo/components/header.jsx\)](#) to store the current value of the input field. When the user adds the item, the field will be cleared.

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

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

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

```

export function Header({ dispatch }) {
  const [inputValue, setInputValue] =
  useState(localStorage.getItem("inputValue") || "");

  const addItem = useCallback(
    (title) => {
      dispatch({ type: ADD_ITEM, payload: { title } });
      setInputValue(""); // Clears the field after adding the
item
      localStorage.removeItem("inputValue"); // Removes the saved
value from LocalStorage
    },
    [dispatch]
  );

  const handleInputChange = useCallback((e) => {
    const value = e.target.value;
    setInputValue(value);
    localStorage.setItem("inputValue", value); // Saves the value
in LocalStorage
  }, []);

  return (
    <header className="header" data-testid="header">
      <h1>Tasks</h1>
      <Input
        onSubmit={addItem}
        label="New task"
        placeholder="Type a task here"
        defaultValue={inputValue}
        onChange={handleInputChange} // Updates the state when
typing
      />
    </header>
  );
}

```

Modification in the [Input component](#) ([src/todo/components/input.jsx](#))

Update the [Input component](#) ([src/todo/components/input.jsx](#)) to accept the [onChange property](#) ([node_modules/@types/react/index.d.ts](#)) and bind it to the [onChange event](#) ([node_modules/@types/react/index.d.ts](#)) of the input field.

File: [src/todo/components/input.jsx](#)

```
import { useCallback } from "react";
```

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

```
export function Input({ onSubmit, placeholder, label, defaultValue,
onBlur, onChange }) {
```

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

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

        if (!hasValidMin(value, 2)) return;

```

```
        onSubmit(value);
        e.target.value = "";
    }
},
[onSubmit]
);

return (
    <div className="input-container">
        <input
            className="new-todo"
            id="todo-input"
            type="text"
            data-testid="text-input"
            autoFocus
            placeholder={placeholder}
            defaultValue={defaultValue}
            onBlur={handleBlur}
            onKeyDown={handleKeyDown}
        />
    </div>
);
```

```

        onChange={onChange} // Adds the onChange event
      />
      <label className="visually-hidden" htmlFor="todo-input">
        {label}
      </label>
    </div>
  );
}

```

Result

- The value typed in the field will be saved in LocalStorage while the user types.
- When the user adds the item, the field will be cleared and the value will be removed from LocalStorage.

00:18:37
- 00:18:45

> Response rendering

00:18:41

Participant

Doesn't it have any drafts here?

00:18:44

Researcher

No.

00:19:25

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

00:19:28
- 00:19:30

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

00:20:35

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

00:20:38
- 00:20:41

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

00:21:49

Participant

You can sound during the hand of doing this.

00:22:28

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

00:22:28

Participant

I see there's some pagination implemented here

00:22:40

This, texts here, can keep or

00:22:43

do they have to be in Portuguese.

00:22:48

Researcher

You can leave it as it is in the task

00:22:53
- 00:23:02

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

00:23:00

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

00:23:26

Participant

Need to create an options box for him to select 1, 3, or 5.

00:23:32

The initial value should be 5.

00:23:32

Researcher

Right.

00:24:49

Chat (window: 0)

> Agent

github.copilot.default

> Context

codebase

> Message

#codebase I want to create pagination in my listing. For pagination, a field with values 1, 3, or 5 (default value) should be created, which will be the number of elements on a page.

> Response

To implement pagination with the functionality to select the number of items per page (1, 3, or 5, with 5 as default), you can follow the steps below:

1. Add a selector for the number of items per page

In the component [Main \(src/todo/components/main.jsx\)](#), add a selector (<select>) to allow the user to choose the number of items per page.

Updated code for the component [Main \(src/todo/components/main.jsx\)](#)

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

export function Main({ todos, dispatch }) {
  const { pathname: route } = useLocation();
  const [currentPage, setCurrentPage] = useState(1);
  const [itemsPerPage, setItemsPerPage] = useState(5); // Default
  value: 5

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

    // Pagination: Returns only the items of the current page
    const startIndex = (currentPage - 1) * itemsPerPage;
    const endIndex = startIndex + itemsPerPage;
    return filteredTodos.slice(startIndex, endIndex);
  }, [todos, route, currentPage, itemsPerPage]);

  const totalItems = useMemo(() => {
    return todos.filter((todo) => {
      if (route === "/active") return !todo.completed;
      if (route === "/completed") return todo.completed;
      return todo;
    }).length;
  }, [todos, route]);

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

  const handleItemsPerPageChange = useCallback((e) => {
    setItemsPerPage(Number(e.target.value)); // Updates the number
of items per page
    setCurrentPage(1); // Resets to the first page
  }, []);

  return (
    <main className="main" data-testid="main">
      {todos.length > 0 && (
        <div className="toggle-all-container">
          <input
            className="toggle-all"
            type="checkbox"
            id="toggle-all"
            data-testid="toggle-all"
            checked={toggleAll} />
        </div>
      )}
    </main>
  );
}
```



```

        checked={todos.every((todo) => todo.completed)}
        onChange={toggleAll}
      />
      <label className="toggle-all-label"
htmlFor="toggle-all">
        Toggle All Input
      </label>
    </div>
  )}

  <div className="items-per-page-selector">
    <label htmlFor="items-per-page">Items per page:</label>
    <select
      id="items-per-page"
      value={itemsPerPage}
      onChange={handleItemsPerPageChange}
      data-testid="items-per-page-selector"
    >
      <option value={1}>1</option>
      <option value={3}>3</option>
      <option value={5}>5</option>
    </select>
  </div>

  {visibleTodos.length > 0 ? (
    <ul className={classnames("todo-list")} data-
testid="todo-list">
      {visibleTodos.map((todo, index) => (
        <Item todo={todo} key={todo.id} dispatch=
{dispatch} index={index} />
      ))}
    </ul>
  ) : (
    <div className="todo-list-empty-container">
      <span className="todo-list-empty-message">No tasks
here yet</span>
    </div>
  )}

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

```

2. Update the component [PaginationControls](#) ([src/todo/components/paginationControls.jsx](#))

Ensure that the component [PaginationControls](#)
([src/todo/components/paginationControls.jsx](#)) is correctly handling navigation
between pages.

File: [src/todo/components/paginationControls.jsx](#)
 import React from "react";
 import classnames from "classnames";

```

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

  const pageChange = (direction) => {
    setCurrentPage((prevPage) => {
      if (direction === "next" && prevPage < totalPages) {
        return prevPage + 1;
      }
      if (direction === "prev" && prevPage > 1) {
        return prevPage - 1;
      }
      return prevPage;
    });
  };

  return (
    <div className="pagination-controls">
      <button
        className={classnames("pagination-button", { inactive:
currentPage === 1 })}
        onClick={() => pageChange("prev")}
        disabled={currentPage === 1}
      >
        Previous
      </button>
      <span className="pagination-info">
        Page {currentPage} of {totalPages}
      </span>
      <button
        className={classnames("pagination-button", { inactive:
currentPage === totalPages })}
        onClick={() => pageChange("next")}
        disabled={currentPage === totalPages}
      >
        Next
      </button>
    </div>
  );
}

```

3. Style the items per page selector

Add styles for the items per page selector in the CSS file.

```

File: src/todo/app.css
.items-per-page-selector {
  margin: 10px 0;
  display: flex;
  align-items: center;
  gap: 10px;
}

.items-per-page-selector label {
  font-weight: bold;
}

.items-per-page-selector select {
  padding: 5px;
  border-radius: 5px;
  border: 1px solid #ccc;
}

```

Result

- The user can select the number of items per page (1, 3, or 5) using the selector.
- The list will be paginated based on the number of items per page.
- Navigation between pages will be done using the "Previous" and "Next" buttons.

00:24:55
- 00:25:13

> Response rendering

00:24:58

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

00:25:06
- 00:25:07

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

00:25:16

Researcher

And what are you thinking now.

00:25:20

Participant

in this component, what do I need to do to apply the pagination that is already created.

00:25:26

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

00:25:29

Participant

So here I will have to change, import the component.

00:25:33

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

00:25:39

Participant

And create the selection box.

00:26:00

Researcher

ok.

00:26:03

Participant

And make these next and previous rules.

00:26:30

I sent this chat to the copilot, because I already saw that it had created the pagination

00:26:32

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

00:26:35

Participant

somehow it will reuse what I have.

00:26:35

Researcher

cool

00:26:40

ok.

00:26:42

Participant

Here in the middle.

00:26:44

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

00:26:49

Participant

It's breaking the code

00:27:04

in the page advance check

00:27:09

That's right.

00:27:12
- 00:27:27

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

00:27:43

Participant

I'll check how this part is

00:27:52

Does it need any styling rule?.

00:27:55

Researcher

You can follow what's being asked in the task

00:28:16

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

00:28:19
- 00:28:22

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

00:28:41

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

00:28:45
- 00:28:46

> **Implementing Copilot code** using apply button (src/todo/app.css)

00:29:00

Participant

One by default, there are 5 it will go to 3, goes to 3, to one, goes to one.

00:29:09

Researcher

00:29:11 ok.
Participant
Disable the button when it's on the last page.

00:29:22 **Researcher**
ok.

00:29:35 **Participant**
Add, 10 or more tasks.

00:29:37 sequentially to monitor tracking.

00:30:50 **Researcher**
ok.

00:31:40 **Participant**
First page, it's OK If it's the last one, it's OK
this one is working too.

00:32:03 **Researcher**
ok.

00:32:11 **Participant**
additional tasks are present.

00:32:14 There are the 10 I added.

00:32:17 **Researcher**
Great, that's it.

00:32:40 Ok, finished the tasks before 40 minutes. Very good

00:34:19 Now you can answer the post-study questionnaire please.

00:35:13 **Participant**
OK.

00:35:20 I somewhat disagree with that, because sometimes it.

00:35:24 induces you a lot, especially when you use the

00:35:29 online copilot.

00:35:32 So it induces you a lot, which ends up being disruptive.

00:35:36 Besides giving a lot of ready-made stuff. So, sometimes the person doesn't understand what's

00:35:41 happening, ends up accepting and it works.

00:35:45 It ends up being a bit disruptive in that regard.

00:35:49 **Researcher**
Cool.

00:35:52 **Participant**
It greatly increases productivity.

00:35:58 Effectiveness.

00:35:58 I agree that it improves.

00:36:03 We can deliver faster, do more things.

00:36:07 Less time.

00:36:08 **Researcher**
ok.

00:36:21 **Participant**
The study agrees.

00:36:34 Here, I think in the case of codeSPACE,

00:36:38 it worked very well, but there are several times

00:36:42 when you are using it, and it happens to have hallucination.

00:36:48 **Researcher**
OK.

00:36:49 **Participant**
Not doing what I want it to do.

00:36:51 so I somewhat agree.

00:36:57 Direction.

00:36:57 **Researcher**
Ready, ok.

00:37:06 **Participant**
Interaction could be clearer

00:37:12 That's what it would be. This interaction would be clearer and
00:37:14 understandable.
00:37:46 Flexibility I think is good for him to be able to read the entire code or just
00:37:57 **Researcher**
Cool.

00:38:07 **Participant**
And it's easy to use.
00:38:55 My God, no, I didn't feel insecure, nor unmotivated, nor anything
00:39:03 It helped me a lot.
00:39:06 Yeah, I didn't have to make an effort. He managed to meet all the requirements
00:39:10 that I sent to him.
00:39:14 and managed to do everything that was requested.
00:39:19 **Researcher**
Cool.

00:39:30 **Participant**
It wasn't physically demanding and mentally it wasn't very
00:39:35 **Researcher**
ok.

00:40:12 **Participant**
The tasks, I found the first 2 easy.
00:40:22 **Researcher**
OK
00:40:23 **Participant**
This one about saving a draft. I already had an idea of what to do,
00:40:26 which was simply to save it in the localstore or somewhere.
00:40:32 And the pagination that I wouldn't know how to answer right away,
00:40:36 would have to study the code a bit more.
00:40:51 **Researcher**
ok.

00:40:53 **Participant**
The pagination extremely realistic.
00:41:15 **Researcher**
ok
00:41:26 **Participant**
And persisting too, very realistic.
00:41:29 **Researcher**
Great, ok. Thanks for answering.
00:41:33 Now I'm going to ask a few more questions. Ok
00:41:48 How did the copilot influence your understanding of the code
00:41:59 **Participant**
I think it influenced me in how to solve the problem.
00:42:04 mainly in the persistence task. I have an idea of what had to be done,
00:42:10 which was to send to the localStore
00:42:14 Maybe in practice, doing the whole experiment without the help of the copilot
00:42:20 Would take more time, so it influenced me to do it in a
00:42:24 faster way
00:42:27 In the pagination task, the solution proposed by it worked the first time.
00:42:43 **Researcher**
Cool.

00:42:48 **Participant**
However, usually in the job market, these things are practically
00:42:52 ready.
00:42:55 So this induction sometimes ends up hindering
00:42:58 especially in projects you will work on,
00:43:02 that have some kind of structure, some kind of rule for the project.

00:43:08 Maybe the solution proposed by the copilot is not the most suitable for the standards the company uses.

00:43:13 **Researcher**
Understood.

00:43:13 **Participant**
I once worked on a project where basically everything you had to do.
00:43:20 There had to be a webhook involved in that and many times when I used the copilot
00:43:27 the response was not generated as I expected.

00:43:34 **Researcher**
Understood. Cool.

00:43:36 **Participant**
We saw voices, it ends up hindering.

00:43:43 **Researcher**
you managed to solve all
00:43:47 the tasks. The question is: How would you have solved the
00:43:51 tasks without using copilot?

00:43:56 **Participant**
Probably I would look at some existing code
00:43:59 to get an idea, especially in pagination.
00:44:04 And I would also look at codes on Stackoverflow.
00:44:11 and then I would apply my
00:44:14 knowledge that I already have in the area.

00:44:21 **Researcher**
Do you work with front-end application? react.

00:44:26 **Participant**
Yes, yes.

00:44:27 **Researcher**
Cool, did that make it easier?

00:44:30 **Participant**
It makes it a lot easier, especially understanding what copilot writes

00:44:44 **Researcher**
What aspects of copilot did you like and not like?

00:44:58 **Participant**
What I liked.
00:45:00 Was it being able to analyze the project as a whole,
00:45:05 so just by giving a command to it and sending a prompt
00:45:12 because I want it to analyze a project as a whole. It's really cool,
00:45:19 because you save time looking for these.
00:45:25 These modifications, where you need to modify.

00:45:29 **Researcher**
Cool.

00:45:33 **Participant**
And one thing, maybe I didn't like, is the way it responds separately.
00:45:40 So if it, because the first thing you will read
00:45:45 is the first part of the text and it already sends a code below.
00:45:51 The code is large, I think it confuses a bit to visualize the modifications
00:46:00 if it showed, for example:
00:46:04 follow the files below, instead of showing the entire code in the chat.
00:46:21 In the experiment at a certain point copilot showed the solution that altered 3 files. The manual application of each file can confuse

00:46:41 **Researcher**
Cool.
00:46:43 Did you change the way you interacted with copilot during the experiment?

00:46:58 **Participant**
I don't know, always follow the same line
00:47:01 always keep a direct prompt.

00:47:05 you have to give a general context for it to understand what you want.
00:47:10 But it has to be to the point.
00:47:13 **Researcher**
ok
00:47:18 Cool, you already had, you had already used the copilot before.
00:47:24 **Participant**
I use it.
00:47:26 the copilot inline.
00:47:33 Usually when I need some modification. Chat I use chatGPT
00:47:43 **Researcher**
ok, so you already have some
00:47:46 experience with these GPT chats, LLM?
00:47:54 **Participant**
Yes. It's quite normal for you to send something to it,
00:47:58 it to understand completely different things.
00:48:01 **Researcher**
Ok thank you very much for participating in our study.