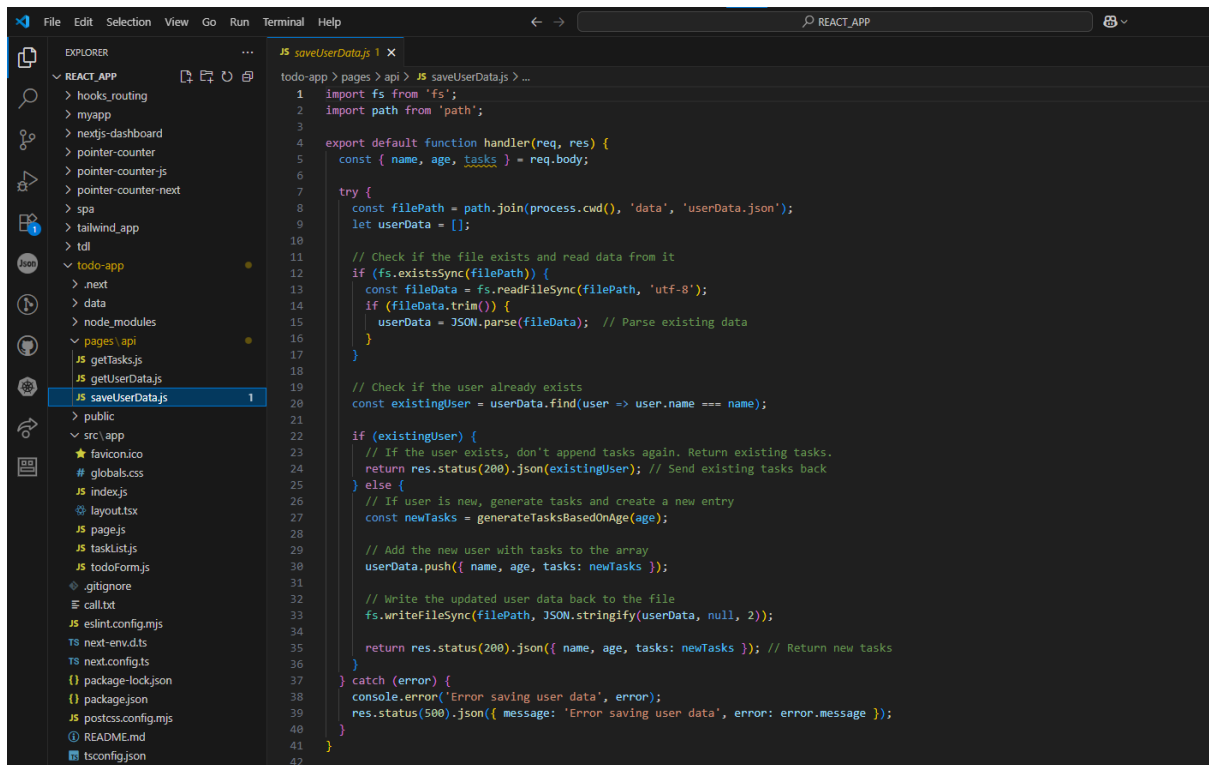


The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the file structure of a project named 'REACT_APP'. The file 'getTasks.js' is selected under the 'pages/api' directory. The main editor area shows the code for 'getTasks.js'.

```
1 import fs from 'fs';
2 import path from 'path';
3
4 export default function handler(req, res) {
5   try {
6     const filePath = path.join(process.cwd(), 'data', 'tasks.json');
7     const fileData = fs.readFileSync(filePath);
8     const tasks = JSON.parse(fileData);
9     res.status(200).json(tasks);
10  } catch (error) {
11    res.status(500).json({ message: "Error reading tasks data", error: error.message });
12  }
13 }
14
```

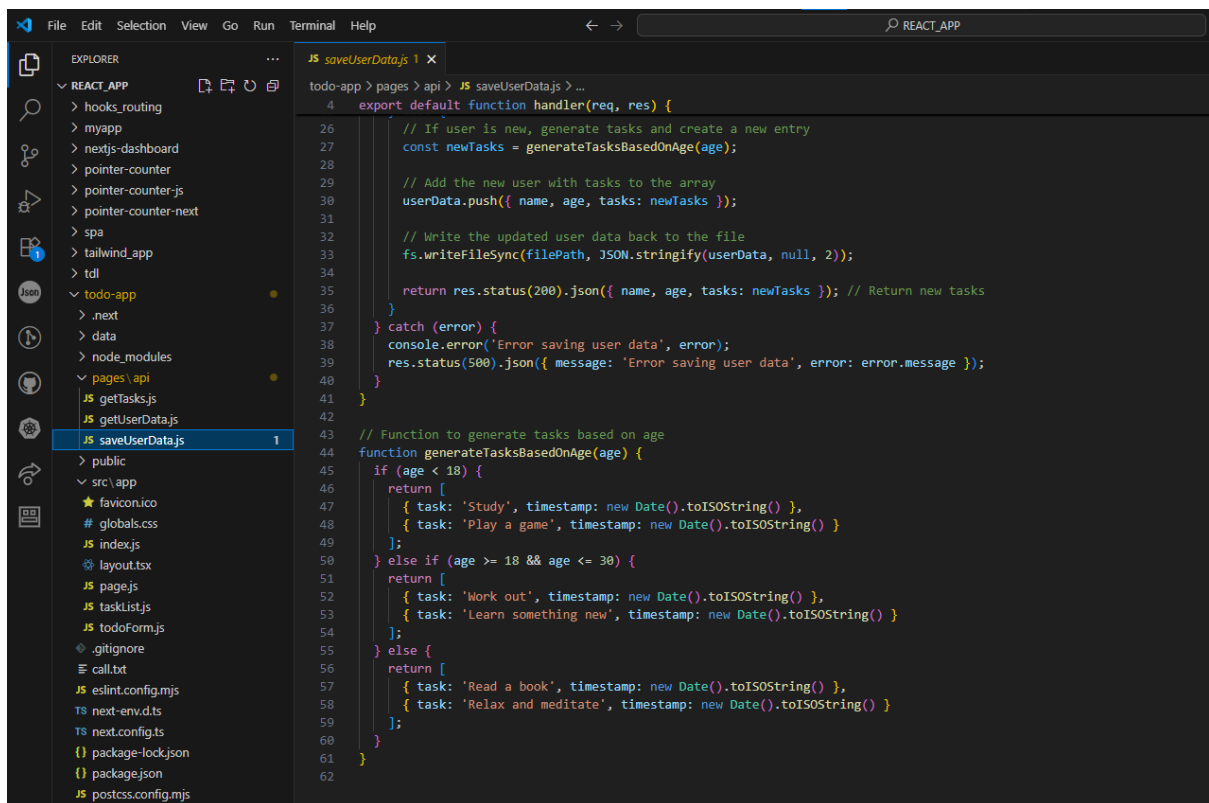
The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the file structure of a project named 'REACT_APP'. The file 'getUserData.js' is selected under the 'pages/api' directory. The main editor area shows the code for 'getUserData.js'.

```
1 import fs from 'fs';
2 import path from 'path';
3
4 export default function handler(req, res) {
5   const { name } = req.query;
6   console.log(`API hit with name: ${name}`); // Debugging statement
7
8   try {
9     const filePath = path.join(process.cwd(), 'data', 'userData.json');
10
11     // Check if the file exists
12     if (!fs.existsSync(filePath)) {
13       return res.status(500).json({ message: 'User data file not found' });
14     }
15
16     const fileData = fs.readFileSync(filePath, 'utf-8');
17
18     // Check if the file is empty
19     if (!fileData.trim()) {
20       return res.status(404).json({ message: "No user data available" });
21     }
22
23     // Parse the JSON content
24     let userData;
25     try {
26       userData = JSON.parse(fileData);
27     } catch (err) {
28       return res.status(500).json({ message: 'Error parsing user data JSON', error: err.message });
29     }
30
31     const user = userData.find(user => user.name === name);
32
33     if (user) {
34       return res.status(200).json(user);
35     } else {
36       return res.status(404).json({ message: 'User not found' });
37     }
38   } catch (error) {
39     console.error('Error reading user data', error);
40     return res.status(500).json({ message: "Error reading user data", error: error.message });
41   }
42 }
43
```



```
File Edit Selection View Go Run Terminal Help
REACT_APP
> hooks_routing
> myapp
> nextjs-dashboard
> pointer-counter
> pointer-counter-js
> pointer-counter-next
> spa
> tailwind_app
> tdl
> todo-app
  > .next
  > data
  > node_modules
  > pages\api
    JS getTasks.js
    JS getUserData.js
    JS saveUserData.js 1
  > public
  > src\app
    ★ favicon.ico
    # globals.css
    JS index.js
    ✨ layout.tsx
    JS page.js
    JS taskList.js
    JS todoForm.js
    .gitignore
    call.txt
    JS eslint.config.mjs
    TS next-env.d.ts
    TS next.config.ts
    {} package-lock.json
    {} package.json
    JS postcss.config.mjs
    README.md
    tsconfig.json

todo-app > pages > api > JS saveUserData.js > ...
1 import fs from 'fs';
2 import path from 'path';
3
4 export default function handler(req, res) {
5   const { name, age, tasks } = req.body;
6
7   try {
8     const filePath = path.join(process.cwd(), 'data', 'userData.json');
9     let userData = [];
10
11     // Check if the file exists and read data from it
12     if (fs.existsSync(filePath)) {
13       const fileData = fs.readFileSync(filePath, 'utf-8');
14       if (fileData.trim()) {
15         userData = JSON.parse(fileData); // Parse existing data
16       }
17     }
18
19     // Check if the user already exists
20     const existingUser = userData.find(user => user.name === name);
21
22     if (existingUser) {
23       // If the user exists, don't append tasks again. Return existing tasks.
24       return res.status(200).json(existingUser); // Send existing tasks back
25     } else {
26       // If user is new, generate tasks and create a new entry
27       const newTasks = generateTasksBasedOnAge(age);
28
29       // Add the new user with tasks to the array
30       userData.push({ name, age, tasks: newTasks });
31
32       // Write the updated user data back to the file
33       fs.writeFileSync(filePath, JSON.stringify(userData, null, 2));
34
35       return res.status(200).json({ name, age, tasks: newTasks }); // Return new tasks
36     }
37   } catch (error) {
38     console.error('Error saving user data', error);
39     res.status(500).json({ message: 'Error saving user data', error: error.message });
40   }
41 }
42
```



```
File Edit Selection View Go Run Terminal Help
REACT_APP
> hooks_routing
> myapp
> nextjs-dashboard
> pointer-counter
> pointer-counter-js
> pointer-counter-next
> spa
> tailwind_app
> tdl
> todo-app
  > .next
  > data
  > node_modules
  > pages\api
    JS getTasks.js
    JS getUserData.js
    JS saveUserData.js 1
  > public
  > src\app
    ★ favicon.ico
    # globals.css
    JS index.js
    ✨ layout.tsx
    JS page.js
    JS taskList.js
    JS todoForm.js
    .gitignore
    call.txt
    JS eslint.config.mjs
    TS next-env.d.ts
    TS next.config.ts
    {} package-lock.json
    {} package.json
    JS postcss.config.mjs

todo-app > pages > api > JS saveUserData.js > ...
4 export default function handler(req, res) {
26   // If user is new, generate tasks and create a new entry
27   const newTasks = generateTasksBasedOnAge(age);
28
29   // Add the new user with tasks to the array
30   userData.push({ name, age, tasks: newTasks });
31
32   // Write the updated user data back to the file
33   fs.writeFileSync(filePath, JSON.stringify(userData, null, 2));
34
35   return res.status(200).json({ name, age, tasks: newTasks }); // Return new tasks
36 }
37 } catch (error) {
38   console.error('Error saving user data', error);
39   res.status(500).json({ message: 'Error saving user data', error: error.message });
40 }
41
42 // Function to generate tasks based on age
43 function generateTasksBasedOnAge(age) {
44   if (age < 18) {
45     return [
46       { task: 'Study', timestamp: new Date().toISOString() },
47       { task: 'Play a game', timestamp: new Date().toISOString() }
48     ];
49   } else if (age >= 18 && age <= 30) {
50     return [
51       { task: 'Work out', timestamp: new Date().toISOString() },
52       { task: 'Learn something new', timestamp: new Date().toISOString() }
53     ];
54   } else {
55     return [
56       { task: 'Read a book', timestamp: new Date().toISOString() },
57       { task: 'Relax and meditate', timestamp: new Date().toISOString() }
58     ];
59   }
60 }
61 }
62
```

```
1  "use client"; // Client-side logic
2  import { useState } from 'react'; 4.0k (gzipped: 1.9k)
3  import TodoForm from './todoForm';
4  import TaskList from './taskList';
5
6  export default function Home() {
7    const [tasks, setTasks] = useState([]);
8    const [userDetails, setUserDetails] = useState(null);
9
10   const handleFormSubmit = async (user) => {
11     const { name, age } = user;
12
13     // Send request to save user data (name, age, and empty tasks)
14     const response = await fetch('/api/saveUserData', {
15       method: 'POST',
16       headers: {
17         'Content-Type': 'application/json',
18       },
19       body: JSON.stringify({ name, age, tasks: [] }), // Send empty array for tasks
20     });
21
22     const data = await response.json();
23
24     if (response.status === 200) {
25       // If tasks exist for the user, use them
26       setTasks(data.tasks);
27       setUserDetails({ name, age });
28     } else {
29       console.error('Error saving user data:', data.message);
30     }
31   };
32
33   return (
34     <div>
35       <h1>Todo Application</h1>
36       <TodoForm onSubmit={handleFormSubmit} />
37       {userDetails && <h2>Welcome back, {userDetails.name}!</h2>}
38       <TaskList tasks={tasks} />
39     </div>
40   );
41 }
42
```

```
1  "use client"; // Client-side logic
2
3  export default function TaskList({ tasks }) {
4    return (
5      <div>
6        <h3>Tasks:</h3>
7        {tasks.length === 0 ? (
8          <p>No tasks available. Add your name and age to get suggestions!</p>
9        ) : (
10          <ul>
11            {tasks.map((task, index) => (
12              <li key={index}>
13                {task.task} - {task.timestamp}
14              </li>
15            ))}
16          </ul>
17        )}
18      </div>
19    );
20 }
21
```

