WE WILL REFACTOR THE QUIZ APP TO USE REST AND A CUSTOM HOOK

`questions.json` (create a new file)

- *Located at the root of your project*, this will store your quiz questions in JSON format.
- Each question has an `id`, `questionText`, and `answerOptions`.
- `answerOptions` is an array of objects, each with `answerText` and `isCorrect`.

JSON Code

```
{
  "questions": [
    {
      "id": 1,
      "questionText": "What is the virtual DOM?",
      "answerOptions": [
        { "answerText": "An exact copy of the real DOM",
"isCorrect": false },
        { "answerText": "A lightweight representation of
the real DOM", "isCorrect": true },
        { "answerText": "A place where React components are
stored", "isCorrect": false },
        { "answerText": "A way to style React components",
"isCorrect": false }
      ]
    },
    // ... more questions here
  ]
}
.
```

2. `src/hooks/useQuestions.ts` (create a file for custom hooks)

- Defines your custom hook `useQuestions`.
- `useState` manages the state for `questions`, `loading`, and `error`.
- `useEffect` fetches the questions from the `json-server` API (`http://localhost:3000/questions`) when the component using the hook mounts.
- The `try...catch` block handles potential errors during the fetch.

- The `finally` block ensures `loading` is set to `false` after the fetch, regardless of success or failure.
- The hook returns an object with the `questions`, `loading` state, and any `error`.

TypeScript Code:

```
import { useState, useEffect } from 'react';
import { Question } from '../types';

const useQuestions = () => {
  const [questions, setQuestions] =
useState<Question[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<Error

 | null>(null);

  useEffect(() => {
    const fetchQuestions = async () => {
      try {
        const response = await fetch('http://
localhost:3000/questions');
        const data = await response.json();
        setQuestions(data);
      } catch (error:

 any) {
        setError(error);
      } finally {
        setLoading(false);
      }
    };

    fetchQuestions();
  }, []);

  return { questions, loading, error };
};
```

```
export default useQuestions;
```

.

3. `src/App.tsx` (Update)

- Import the `useQuestions` hook.
- Call `useQuestions()` to fetch the questions and get the `questions`, `loading`, and `error` values.
- Handle the loading and error states.
- Pass the `questions` data to the `Quiz` component as a prop.

TypeScript Code

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link }
from 'react-router-dom';
import Quiz from './components/Quiz';
import

 useQuestions from './hooks/useQuestions';
// Import the hook
import './App.scss';

const App: React.FC = () => {
  const { questions, loading, error } = useQuestions();
// Call the hook

  if (loading) {
    return <div>Loading...</div>;
  }

  if (error) {
    return <div>Error: {error.message}</div>;
  }

  return (
    <Router>
      <div className="app-container">
        <nav className="navbar">
          <Link to="/" className="nav-link">Home</Link>
          <Link to="/quiz" className="nav-link">Quiz</Link>
        </nav>

        <Routes>
          <Route path="/" element={<HomePage />} />
          <Route path="/quiz" element={<Quiz
questions={questions} />} />
        </Routes>
      </div>
    </Router>
  );
};

// ... HomePage component ...

export default App;
```
.