Add asynchronous fetching of questions to quiz project - starter files are in `starterquiz` and solution is in `fetchquiz` folder:

- **`useFetch` Hook:**
  - Takes the API endpoint URL as input.
  - Uses `useEffect` to fetch data when the component mounts or the URL changes.
  - Manages `loading`, `error`, and `data` state.
- **`Quiz` Component:**
  - Uses the `useFetch` hook to get quiz questions from the API.
  - Handles the `loading` and `error` states gracefully.
  - Renders the questions dynamically from the fetched `questions` data.

Remember:

- Replace `'YOUR_API_ENDPOINT_HERE'` with the actual endpoint of your quiz data API. `json-server` will provide the endpoint after the next part where you move the questions and start the server.
- Make sure your API returns the questions in the same format as your hardcoded `questions` array (or adjust the code accordingly).

Additional Tips:

- **Error Handling:** Consider adding more robust error handling to the `useFetch` hook and the `Quiz` component to provide better user feedback.
- **Loading Indicators:** Enhance the user experience by adding loading indicators while the questions are being fetched.
- **Caching:** For performance, implement a caching mechanism to avoid fetching the same data repeatedly.

The code with a `useFetch` hook to fetch quiz questions from an external API, and modifications to the `Quiz` component to handle the fetched data:

```
import React, { useState, useEffect } from 'react';
import './index.css';

function Question({ questionText, options, correctAnswer,
selectedOption, onSelectOption }) {
  // ... (Question component remains the same)
}

// Custom useFetch hook
function useFetch(url) {
```

```
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    async function fetchData() {
      try {
        const response = await fetch(url);
        const json = await response.json();
        setData(json);
      } catch (err) {
        setError(err);
      } finally {
        setLoading(false);
      }
    }
    fetchData();
  }, [url]);

  return { data, loading, error };
}

function Quiz() {
  const [score, setScore] = useState(0);
  const [currentQuestionIndex, setCurrentQuestionIndex] =
useState(0);
  const [selectedOption, setSelectedOption] =
useState(null);
  const { data: questions, loading, error } =
useFetch('YOUR_API_ENDPOINT_HERE'); // Replace with your
API endpoint

  const handleNextQuestion = () => {
    if (questions[currentQuestionIndex].correctAnswer ===
selectedOption) {
      setScore(score + 1);
    }
    setCurrentQuestionIndex(currentQuestionIndex + 1);
    setSelectedOption(null);
  };
```

```
  if (loading) return <p>Loading questions...</p>;
  if (error) return <p>Error: {error.message}</p>;

  return (
    <div>
      {currentQuestionIndex < questions.length ? (
        <>
          <Question

questionText={questions[currentQuestionIndex].questionText}

options={questions[currentQuestionIndex].options}

correctAnswer={questions[currentQuestionIndex].correctAnswe
r}
            selectedOption={selectedOption}
            onSelectOption={setSelectedOption}
          />
          <button onClick={handleNextQuestion}>Next</
button>
        </>
      ) : (
        <p>You scored {score} out of {questions.length}</p>
      )}
    </div>
  );
}

export default Quiz;
```

Next: Set up a JSON Server and use your existing quiz questions with it. This is a quick and easy qua to test REST APIs. For production environments, use a more robust backend solution like Node.js, Python, or a cloud-based database, etc. By default, JSON Server uses a file-based database (`db.json`) that resets when the server restarts. If you need persistence, look into using a real database like SQLite or MongoDB with JSON Server.:

1. Install JSON Server

If you don't have it already, install `json-server` globally:

```
npm install -g json-server
```

2. Create a Data File

- Create a file named `db.json` (or any name you prefer).
- Copy your existing `questions` array from your React code and paste it into this `db.json` file.
- The JSON file should have questions in quotes, no stray commas, be in an object, and look like this:

```
{
  "questions": [
    {
      "questionText": "What is the name of the function
used to update the state of a component in React?",
      "options": ["useState", "useEffect", "useRef",
"useCallback"],
      "correctAnswer": "useState"
    },
    // ... rest of your questions
  ]
}
```

3. Start JSON Server

Run this command in your terminal. This assumes `db.json` is in the same directory where you run this command:

```
json-server --watch db.json
```

This will start the server on `http://localhost:3000` by default. Your questions will be accessible at `http://localhost:3000/questions`.

4. Modify Your `useFetch` Hook

Update the `url` parameter in your `useFetch` hook to point to the JSON Server endpoint:

JavaScript

```
const { data: questions, loading, error } =
useFetch('http://localhost:3000/questions');
```

Key Points

- JSON Server provides a quick and easy way to create a REST API for your quiz data.
- It's great for development and testing purposes.