# Full Stack Application Development

## CO-2: Week 6 : Exercise 2

**Aim:** Create a React component called CreatePost that allows users to create and submit a new post to a given API endpoint using Axios. Implement a form with input fields for the post title and body. On form submission, use Axios to send a POST request to the API endpoint with the entered data. Use postman for this exercise.

## Description:

The React Axios - CreatePost Component allows users to create and submit a new post using Axios and test it via Postman. The project setup includes installing dependencies (axios, json-server), setting up a mock API (db.json), and defining an api.js file to handle the POST request. The CreatePost.jsx component contains a form where users enter a title and body, which are sent to the API upon submission. The response updates the UI with a success or failure message. The App.js integrates CreatePost, and running npm start launches the app. Testing with Postman involves sending a POST request to http://localhost:8883/posts with JSON data, verifying that the post is stored in db.json. This setup enables seamless frontend-backend integration, making it easy to test API interactions in a React project.

## Step 1: Install Dependencies

```
npx create vite@latest week6b

cd week6b

npm install

npm install axios

npm install axios json-server
```

## Step 2: Set Up a Mock API (db.json)

1. **Create a file db.json in the root directory week6b to simulate an API:**

```
{
  "posts": []
}
```

## 2. Start the JSON server:

```
json-server --watch db.json --port 5000
```

## Postman:

Postman is a popular API testing tool that allows developers to send, test, and debug HTTP requests such as GET, POST, PUT, and DELETE without writing code. It provides a user-friendly interface to interact with APIs, making it easy to test backend services, monitor API responses, and automate API testing.

### Uses of Postman

✅ API Testing: Send requests to API endpoints and verify responses.
✅ Request Automation: Automate API calls using collections and test scripts.
✅ Debugging APIs: Inspect request headers, payload, and responses to troubleshoot issues.
✅ Collaboration: Share API requests and collections with a team.
✅ Mock Server: Simulate API responses without needing a real backend.
✅ Integration with CI/CD: Automate API testing in DevOps pipelines.

### Example Use Case

If you're building a React app with Axios, you can use Postman to manually test an API before integrating it. For instance, to create a new post in a JSON server, you send a POST request to http://localhost:5000/posts with the following JSON body:

json
CopyEdit
```
{
  "title": "Hello World",
  "body": "This is my first post using Postman!"
}
```
Postman will return a response confirming the post was added successfully.

🚀 It simplifies API development, testing, and debugging, making it an essential tool for developers! 😊

Download link: [https://www.postman.com/downloads/](https://www.postman.com/downloads/)

## Project Structure:

```
Week6b/
│── src/
│   │── components/
│   │   │── CreatePost.jsx
│   │── api.js
│   │── App.jsx
│── db.json (Mock API)
```

## Programs:

### CreatePost.jsx

```
import React, { useState } from "react";
import { createPost } from "../api";

const CreatePost = () => {
  const [title, setTitle] = useState("");
  const [body, setBody] = useState("");
  const [message, setMessage] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    const postData = { title, body };

    try {
      const newPost = await createPost(postData);
      setMessage(`Post created successfully! ID: ${newPost.id}`);
      setTitle("");
      setBody("");
    } catch (error) {
      setMessage("Failed to create post.");
    }
  };

  return (
    <div>
      <h2>Create a New Post</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Title:</label>
          <input
            type="text"
            value={title}
            onChange={(e) => setTitle(e.target.value)}
            required
          />
        </div>
        <div>
          <label>Body:</label>
          <textarea
            value={body}
            onChange={(e) => setBody(e.target.value)}
            required
          />
        </div>
        <button type="submit">Submit Post</button>
      </form>
      {message && <p>{message}</p>}
    </div>
  );
};

export default CreatePost;
```

## App.jsx:

```
import React from "react";
import CreatePost from "./components/CreatePost";

const App = () => {
 return (
  <div>
   <h1>React Axios - Create Post</h1>
   <CreatePost />
  </div>
 );
};

export default App;
```

## api.js:

```
import axios from "axios";

const API_URL = "http://localhost:8883/posts";

// Function to create a new post
export const createPost = async (postData) => {
  try {
    const response = await axios.post(API_URL, postData);
    return response.data;
  } catch (error) {
    console.error("Error creating post:", error);
    throw error;
  }
};
```

## db.json

```
{
 "posts": []
}
```

## Test API in Postman

1. Open Postman.
2. Set Method to POST.
3. Enter URL: http://localhost:8883/posts.
4. Go to the Body tab → Select raw → Choose JSON format.
5. Enter the following data

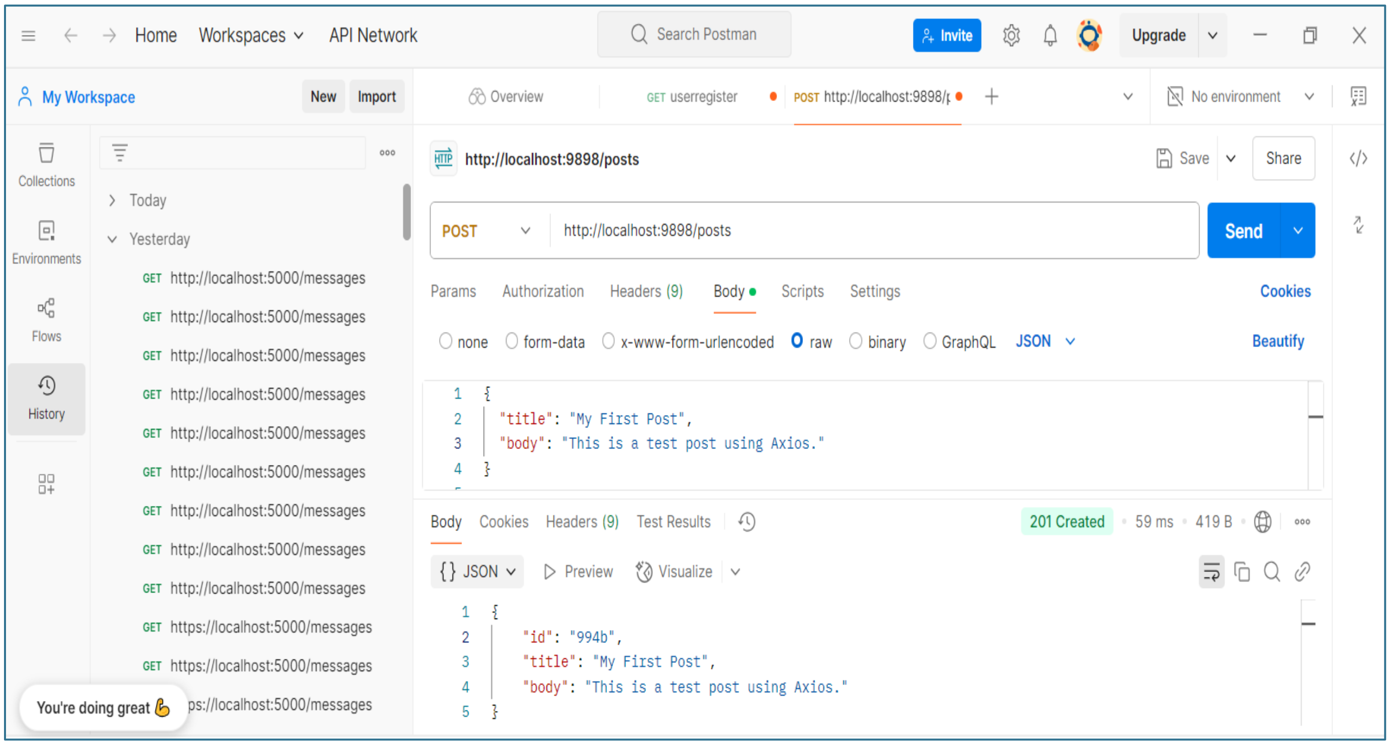
```
{
 "title": "My First Post",
 "body": "This is a test post using Axios."
}
```

6. Click Send.
7. Response:

✓ The post is successfully stored in db.json

## Output:

## From postman



http://localhost:9898/posts

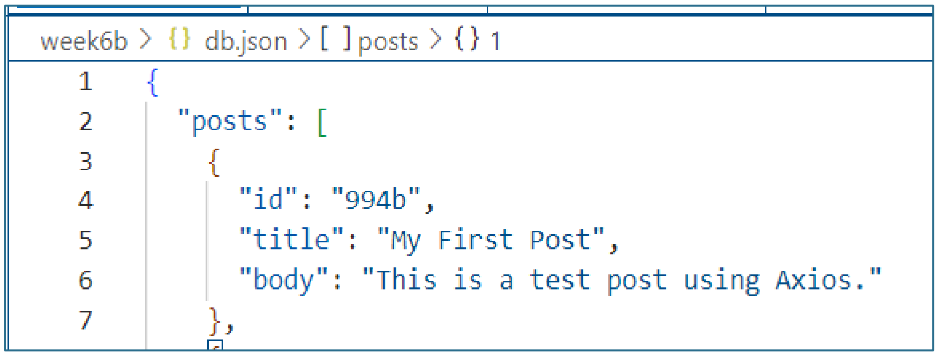
POST http://localhost:9898/posts

Body: raw [✓ JSON]

```
{
  "title": "My First Post",
  "body": "This is a test post using Axios."
}
```

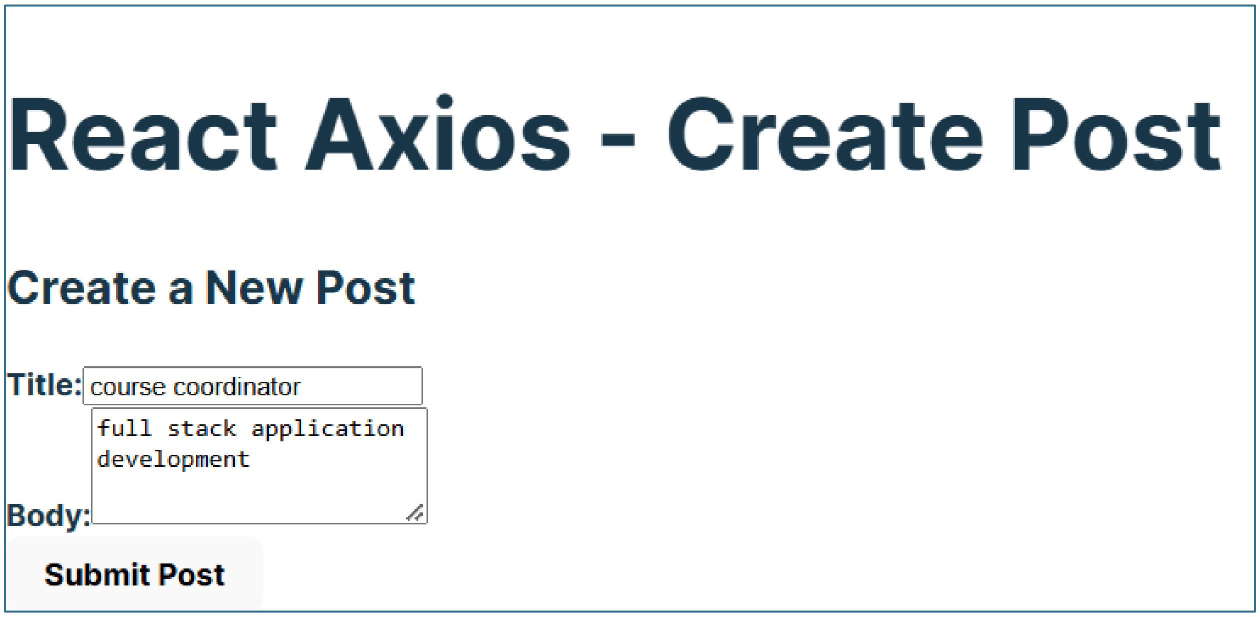
201 Created · 59 ms · 419 B

```
{
    "id": "994b",
    "title": "My First Post",
    "body": "This is a test post using Axios."
}
```



```
{
  "posts": [
    {
      "id": "994b",
      "title": "My First Post",
      "body": "This is a test post using Axios."
    },
```

From GUI:



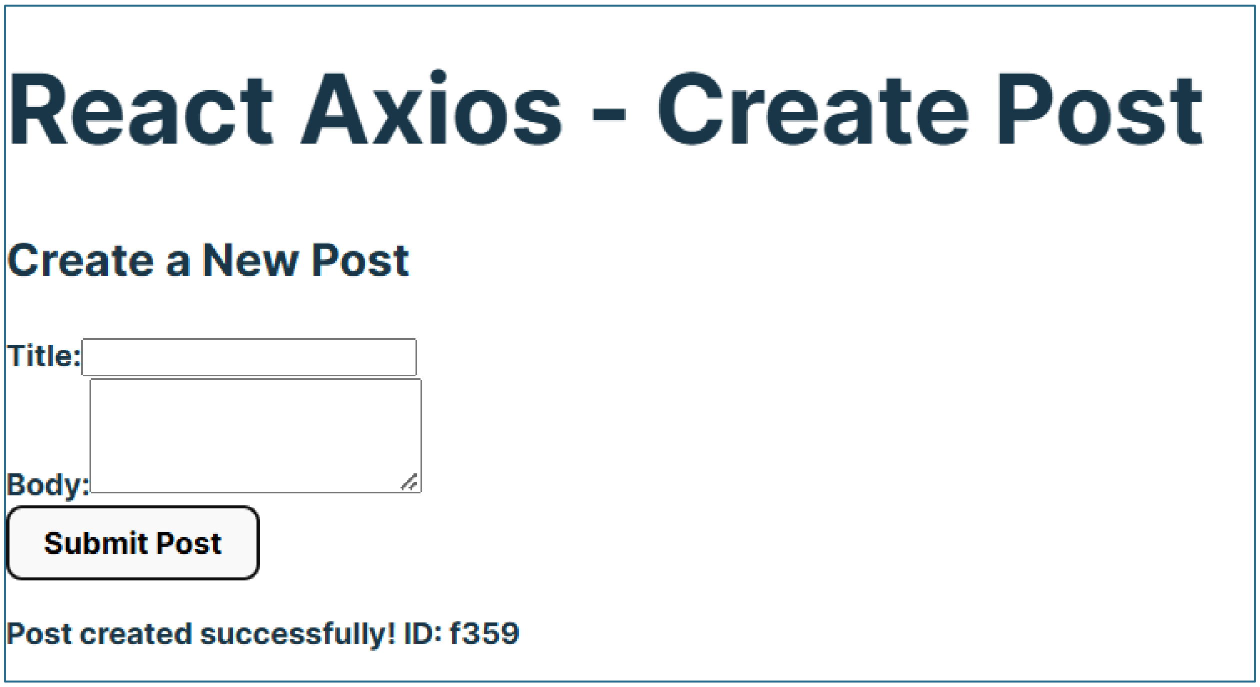
React Axios - Create Post

Create a New Post

Title: course coordinator

Body: full stack application development

Submit Post



React Axios - Create Post

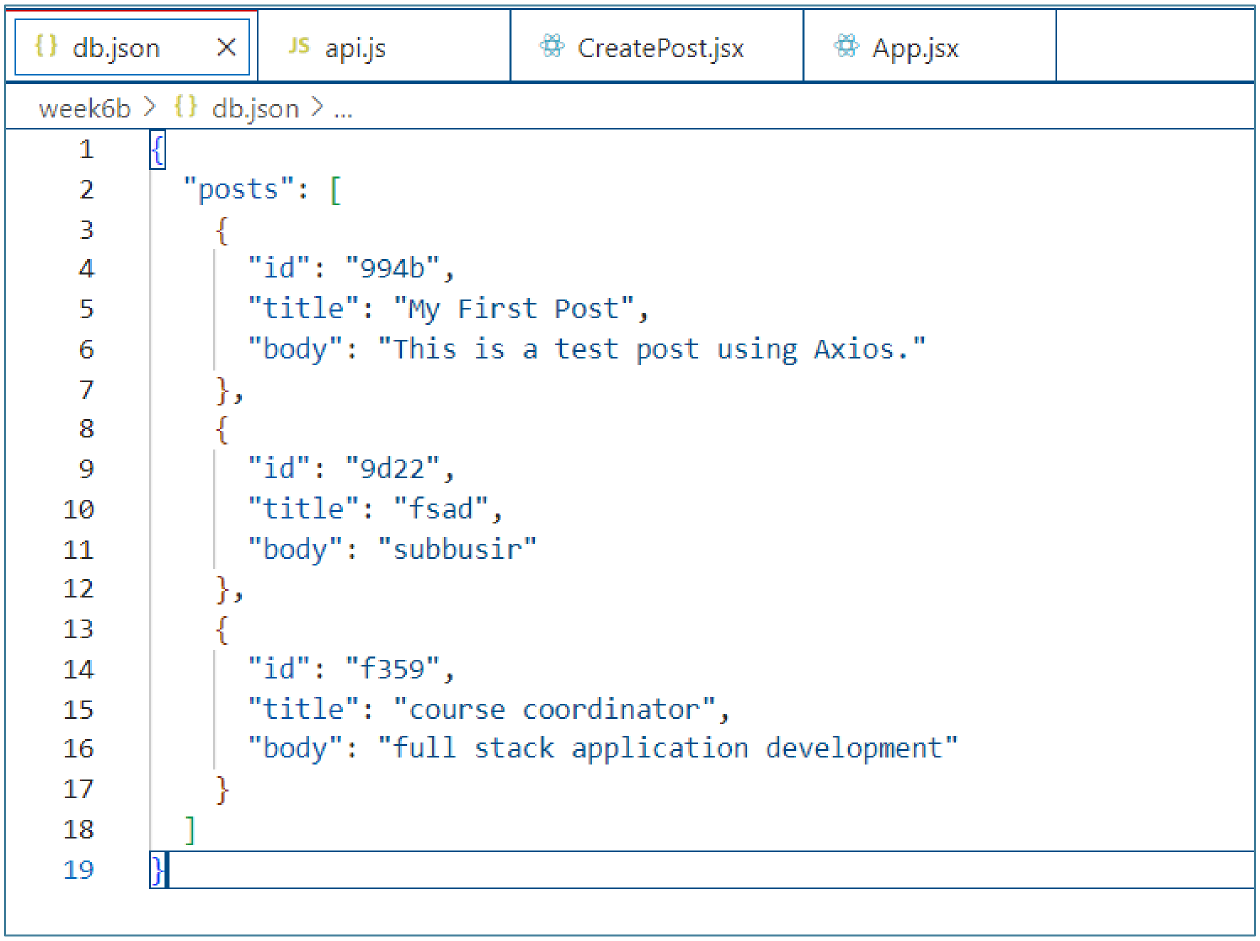
Create a New Post

Title:

Body:

Submit Post

Post created successfully! ID: f359



```json
{
  "posts": [
    {
      "id": "994b",
      "title": "My First Post",
      "body": "This is a test post using Axios."
    },
    {
      "id": "9d22",
      "title": "fsad",
      "body": "subbusir"
    },
    {
      "id": "f359",
      "title": "course coordinator",
      "body": "full stack application development"
    }
  ]
}
```

## Results:

Thus successfully executed postman API integration and GUI in react Axios and The React Axios - CreatePost Component allows users to create and submit a new post using Axios, with data stored in db.json from both the frontend GUI text fields and Postman. The project setup includes installing dependencies (axios, json-server), configuring a mock API (db.json), and defining api.js to handle POST requests. The CreatePost.jsx component provides a form where users enter a title and body, which are sent to the API upon submission. The response updates the UI with a success or failure message, and App.js integrates CreatePost. Running npm start launches the app, while Postman is used to send a POST request to http://localhost:8883/posts with JSON data, verifying that the post is successfully stored in db.json. This setup ensures smooth frontend-backend integration, allowing users to add posts through both the React UI and Postman API testing, making API interactions easy to develop and verify.