Connecting React and Spring

👉 Updating the React Application

To connect our React frontend with our Spring Boot backend, we need to change the URL in our React code.

- Open the AllPosts.jsx file in your React project
- Find the useEffect hook that fetches data
- Change the URL from the fake JSON server to our Spring Boot API:

Example:

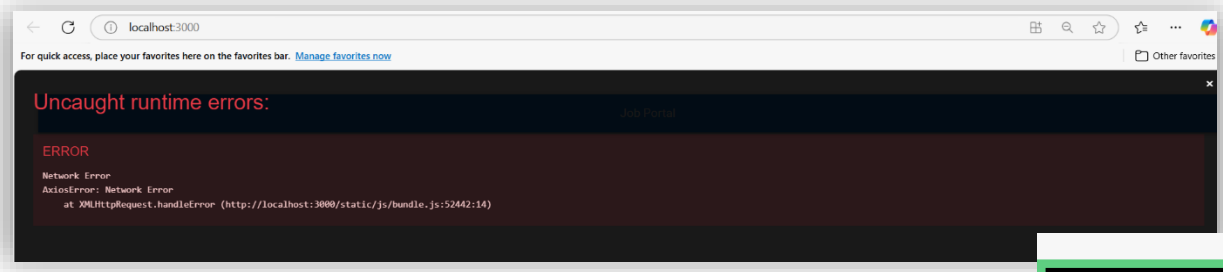
```
useEffect(() => {
    const fetchInitialPosts = async () => {
        const response = await axios.get(`http://localhost:8080/jobPosts`);
        console.log(response);
        setPost(response.data);
    }
     fetchInitialPosts();
  }, []);
```

👉 Understanding CORS Issues

When we try to run our React app now, we'll face a Network Error. This happens because of a security feature called CORS (Cross-Origin Resource Sharing).

What is CORS?

- It's a security mechanism built into browsers
- It prevents websites from requesting data from different domains/origins
- In our case, React runs on localhost:3000 while Spring Boot runs on localhost:8080
- Different ports count as different origins, so the browser blocks these requests



👉 Enabling CORS in Spring Boot

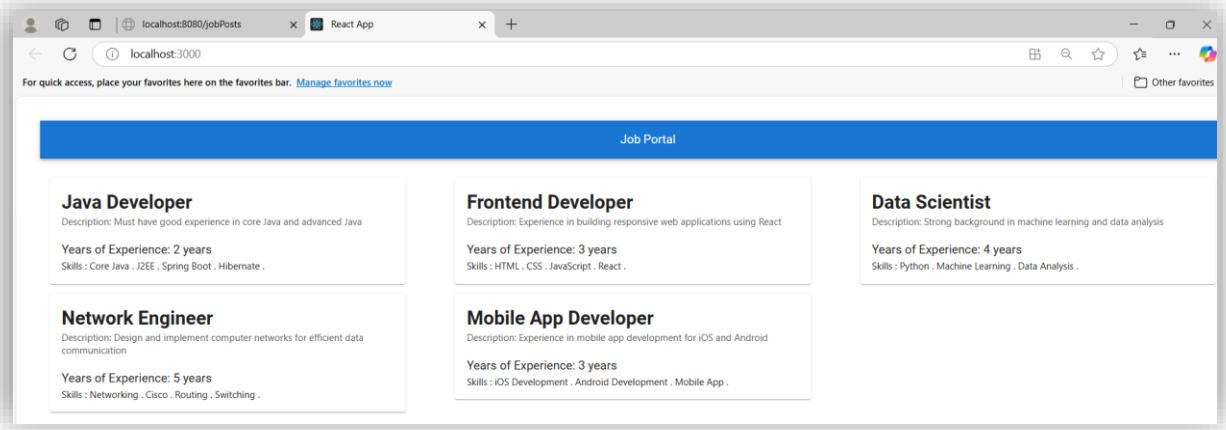
To fix this issue, we need to tell our Spring Boot application to accept requests from our React application:

- Add the @CrossOrigin annotation to our controller
- Specify which origin is allowed to access our API

Example:

```java
package com.telusko.spring_boot_rest.controller;

import java.util.List;

import com.telusko.spring_boot_rest.model.JobPost;
import com.telusko.spring_boot_rest.service.JobService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController  // Marks this class as a REST controller to handle HTTP requests
@CrossOrigin(origins = "http://localhost:3000") // Enables Cross-Origin requests
public class JobRestController {

    // Injecting JobService to handle business logic related to job posts
    @Autowired
    private JobService service;

    // Handles GET request to fetch all job posts
    @GetMapping("jobPosts")
    public List<JobPost> getAllJobs() {
        return service.getAllJobs(); // Delegates call to service layer to retrieve job listings
    }
}
```

Output:



The @CrossOrigin(origins = "http://localhost:3000") annotation tells Spring Boot to:

- Allow requests from http://localhost:3000 (our React app)
- Include the necessary CORS headers in responses
- Accept credentials and other details from this origin

Now your React frontend is successfully connected to your Spring Boot backend instead of the fake JSON server! Any changes you make to your data in Spring Boot will be reflected in your React application.