Frontend–Backend Integration

[ASP.NET Core Web API + React]

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

1. Overview of Frontend–Backend Communication

REST API

- REST (Representational State Transfer) is the common pattern for communication between frontend (React) and backend (.NET).
- Backend exposes endpoints like:
  - `POST /api/auth/login`
  - `GET /api/products`
  - `POST /api/products`
- React calls these endpoints using HTTP requests (Axios or Fetch).

CORS (Cross-Origin Resource Sharing)

- Browser security feature that **blocks** frontend apps from calling APIs from another domain.
- Example:
    - React app: `http://localhost:3000`
    - .NET API: `http://localhost:5000`
- Must enable CORS in backend.

JWT (JSON Web Token)

- Token-based authentication.
- Backend issues JWT on login.
- Frontend stores the token and attaches it in `Authorization: Bearer <token>` header for protected API calls.

2. Enable CORS in ASP.NET Core Web API

Step 1: Add CORS service

```
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowReactApp",
        policy =>
        {
            policy.WithOrigins("http://localhost:3000")
                  .AllowAnyHeader()
                  .AllowAnyMethod();
        });
});
```

Step 2: Use CORS middleware

```
app.UseCors("AllowReactApp");
```

3. Axios Setup in React & Environment Variables

Install Axios

```
npm install axios
```

Create an Axios Instance (`src/api/axios.js`)

```
import axios from "axios";

const api = axios.create({
  baseURL: process.env.REACT_APP_API_URL
});

export default api;
```

Environment Variable (`.env`)

```
REACT_APP_API_URL=http://localhost:5000/api
```

> Restart React after editing `.env`.

4. Connecting Frontend Forms to Backend APIs

Login Form → `/auth/login`

```
const handleLogin = async () => {
  const response = await api.post("/auth/login", { email, password });
  console.log(response.data);
};
```

localStorage.setItem(“AUTH_TOKEN”, response.data.token);

Registration Form → `/auth/register`

```
await api.post("/auth/register", userDetails);
```

Product CRUD

- **GET**: `api.get("/products")`
- **POST**: `api.post("/products", newProduct)`
- **PUT**: `api.put("/products/1", updatedProduct)`
- **DELETE**: `api.delete("/products/1")`

5. Handling JWT in React

Login & Store Token

```
const response = await api.post("/auth/login", credentials);
localStorage.setItem("token", response.data.token);
```

Attach Token to Axios Requests

Modify Axios instance:

```
api.interceptors.request.use((config) => {
  const token = localStorage.getItem("token");
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

6. Fetch Protected Data with Token

```
const getProfile = async () => {
  const response = await api.get("/user/profile"); // Token auto-added
  setProfile(response.data);
};
```

If backend rejects unauthorized users, React must handle 401 errors.

7. Conditional Rendering Based on User Roles

Decode token to read role

Install:

```
npm install jwt-decode
```

Usage:

```
import jwtDecode from "jwt-decode";

const token = localStorage.getItem("token");
const payload = token ? jwtDecode(token) : null;

const role = payload?.role;
```

Conditional UI Rendering

```
{role === "Admin" && (
  <button onClick={addProduct}>Add Product</button>
)}

{role === "User" && (
  <p>Welcome regular user!</p>
)}
```

Protected Routes

Use React Router:

```
<Route
  path="/admin"
  element={role === "Admin" ? <AdminPage /> : <Navigate to="/login" />}
/>
```

Summary Checklist

Backend (ASP.NET Web API)

✓ Enable CORS

✓ Add JWT auth middleware

✓ Expose CRUD APIs

✓ Allow Authorization header

Frontend (React)

✓ Axios instance & Environment variables for API base URL

✓ Store JWT in `localStorage`

✓ Use interceptors to attach token

✓ Redirect unauthorized users

✓ Conditional rendering based on role

Q & A

Narasimha Rao T

tnrao.trainer@gmail.com