Sure! Here's a detailed **README.md** template for your React + TypeScript + .NET backend Scoreboard app with Redux, Bootstrap, and API integration.

markdown

CopyEdit

# Scoreboard App

A simple scoreboard application built with React (TypeScript), Redux, and Bootstrap on the frontend, and .NET 6 Web API with Entity Framework Core and SQL Server on the backend.

Users can add players, update scores, and delete players with data persisted via a REST API.

---

## Features

- Add, update, and remove players

- Persist player data on a .NET backend with SQL database

- Responsive UI built with Bootstrap

- State management with React hooks and Redux (optional)

- API communication with Axios

- HTTPS enabled with trusted developer certificate

- CORS configured for frontend-backend communication

---

## Prerequisites

- Node.js (v16 or above recommended)

- .NET 6 SDK or later

- SQL Server (local or remote)

- Visual Studio 2022 or Visual Studio Code

- Optional: Postman or curl for API testing

---

## Backend Setup (.NET API)

### 1. Clone backend repository (or create project)

```bash

git clone <your-backend-repo-url>

cd your-backend-folder

Or create a new Web API project:

bash

CopyEdit

dotnet new webapi -n Scoreboard\_api

cd Scoreboard\_api

**2. Configure database connection**

Edit appsettings.json to set your SQL Server connection string:

json

CopyEdit

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=ScoreboardDb;Trusted\_Connection=True;"

}

**3. Add Entity Framework Core models and DB context**

* Define Player model with Id, Name, Score properties.
* Create AppDbContext with DbSet<Player>.

**4. Add PlayersController.cs**

Implement REST endpoints for GET, POST, PUT, DELETE:

csharp

CopyEdit

[ApiController]

[Route("api/[controller]")]

public class PlayersController : ControllerBase

{

// Inject AppDbContext and implement CRUD endpoints

}

**5. Enable CORS in Program.cs**

csharp

CopyEdit

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowLocalhost3000", policy =>

{

policy.WithOrigins("http://localhost:3000")

.AllowAnyHeader()

.AllowAnyMethod();

});

});

var app = builder.Build();

app.UseCors("AllowLocalhost3000");

app.UseAuthorization();

app.MapControllers();

**6. Trust HTTPS dev certificate**

Run this once:

bash

CopyEdit

dotnet dev-certs https --trust

**7. Run the backend**

bash

CopyEdit

dotnet run

API will be available at https://localhost:44378/api/Players (adjust port accordingly).

**Frontend Setup (React + TypeScript)**

**1. Create React app with Vite**

bash

CopyEdit

npm create vite@latest scoreboard-frontend -- --template react-ts

cd scoreboard-frontend

**2. Install dependencies**

bash

CopyEdit

npm install axios bootstrap react-redux @reduxjs/toolkit

**3. Import Bootstrap CSS in main.tsx or index.tsx**

ts

CopyEdit

import 'bootstrap/dist/css/bootstrap.min.css';

**4. Create PlayerType interface in src/types/player.ts**

ts

CopyEdit

export interface PlayerType {

id: number;

name: string;

score: number;

}

**5. Setup Axios API service (src/services/api.ts)**

ts

CopyEdit

import axios from 'axios';

import { PlayerType } from '../types/player';

const API = axios.create({

baseURL: 'https://localhost:44378/api/Players',

headers: { 'Content-Type': 'application/json' },

});

export const fetchPlayers = () => API.get<PlayerType[]>('/');

export const addPlayer = (player: Omit<PlayerType, 'id'>) => API.post<PlayerType>('/', player);

export const updatePlayer = (player: PlayerType) => API.put(`/${player.id}`, player);

export const deletePlayer = (id: number) => API.delete(`/${id}`);

**6. Implement React components**

* Scoreboard.tsx — manage state, fetch players, handle add/update/delete
* Player.tsx — individual player UI with score buttons and delete
* AddPlayerForm.tsx — form to add a new player
* Header.tsx — display total players or scores

**7. Fetch players on mount, update state on API calls**

Use useEffect to fetch players on load and update via Axios service.

**8. Run the frontend**

bash

CopyEdit

npm run dev

App will be available at http://localhost:3000.

**Troubleshooting**

* **CORS errors**: Check backend CORS policy matches frontend origin
* **HTTPS trust issues**: Run dotnet dev-certs https --trust and restart browser
* **Network errors**: Use browser DevTools Network tab to inspect requests
* **Ports**: Ensure frontend and backend ports match your config
* **Database connection**: Confirm SQL Server is running and connection string is correct

**Next Steps / Improvements**

* Add Redux for centralized state management
* Add authentication (JWT)
* Add sorting/filtering on players
* Add player avatars or profile pictures
* Use SQL migrations and seed data
* Add unit and integration tests

**License**

MIT License © Your Name

**Enjoy your Scoreboard App! 🚀**

yaml

CopyEdit

---

If you want, I can help customize or add commands/scripts for deployment or tests!