Football Managing Site

Mark Recently joined a football club in his neighborhood, as a full stack web developer he got the idea of designing a website to manage players' details.

Backend Functionalities

A REST API will be developed using Node and Express and SQLite Database.

Players Model

Field	Data Type	Present in Request Body
id	**UUID**	**Yes**
firstName	**String**	**Yes**
lastName	**String**	**Yes**
position	**String**	**Yes**
number	**Integer**	**Yes**

Player-Matches Model

Field	Data Type	Present in Request Body
id	**UUID**	**No**
player_id	**UUID**	**No**
match_id	**UUID**	**No**

Matches Model

Field	Data Type	Present in Request Body
id	**UUID**	**Yes**
datetime	**DateTime**	**Yes**
place	**String**	**Yes**
score	**String**	**Yes**
result	**WIN | LOSE | DRAW String**	**Yes**

API Routes/Methods

Route	*Method*	*Description*
/players	***GET***	*Get all players.*
/players	***POST***	*Add a new player.*
/player/:id	***GET***	*Get single player details.*
/matches	***GET***	*Get all matches.*
/matches	***POST***	*Add a new match.*
/matches/:id	***GET***	*Get single match details.*
/matches/:id	***PATCH***	*Update match data.*

Player Object Response Example:

```
{
    "id": "some-uuid",
    "firstName": "Cristiano",
    "lastname": "Ronaldo",
    "position": "Goalkeeper",
    "number": "1"
}
```

Match Object Response Example:

```
{
    "id": "some-uuid",
    "place": "Delhi",
    "datetime": "2022-01-25T19:32:59.362Z",
    "score": "10-0",
    "result": "WIN",
    "players": [{
            "id": "some-uuid",
            "firstName": "Lionel",
            "lastname": "Messi",
            "position": "Attack",
            "number": "10"
        },
        {
            "id": "some-uuid",
            "firstName": "Cristiano",
            "lastname": "Ronaldo",
            "position": "Goalkeeper",
            "number": "1"
        }]
}
```

Notes:

- Use only SQLite database. And make proper tables/models and relations accordingly.
- Do not remove any existing files.

Frontend Components

An angular application will be developed with the mentioned components/pages and proper routing.

Home Page

- Create a home page component available at the **/** (base route).
- It will have a navbar at the top with all the links and club names.
- It will show all the players in a grid of cards with names, numbers, positions.
- Below that will be an **Add Player** button that will open the /register page.

Add Player Page

- Create a add player page component available at the **/register** route..
- It will have a reactive form with the title **Register** that allows the user to add a new player with all required input fields.
- Below that will be a **Add Player** button that will do a POST request on /players..

Matches Page

- Create a matches page component available at the **/matches** route.
- It will show all the matches in the table format. Clicking any one will open a new component on **/matches/:id** route that will show all details and players list.
- Below that will be a **Add Match** button that will open the **/add** page.

Add Match Page

- Create a add match page component available at the **/add** route.
- It will contain a reactive form to input match details such as place, datetime, score, dropdown to select result (Win, Lose, Draw). Below that will be a list of all players with a checkbox to select the players to add to the match.
- Below that will be a **Add Match** button that will do a POST request on /matches route.
- Add proper validation to all the fields. After a successful request, it should redirect to **/matches/:id** page to show the newly added match details.

Notes

- Use `npm start` to start the dev server.
- Use `npm test` to run the tests.
- Use proper state management and maintain the reactivity of components.
- Bootstrap and Font Awesome have been added for styling and icons.
- Do not remove or modify any **data-testid** attributes.