Interview App - Premier League Squads Finder
High-Level Document

Overview

The Premier League Squads Finder is a web application that provides users with a seamless way to search for 2024/25 Premier League football clubs and view detailed information about their current squad. The app leverages a modern tech stack and integrates with **API-Football** to fetch accurate and up-to-date data.

Goals and Objectives

1. **Primary Goal**: To create a user-friendly platform for football enthusiasts to quickly access up-to-date information about Premier League club squads.
2. **Key Objectives**:
    a. Provide a search interface for locating clubs.
    b. Display accurate squad details, including player age, position and even photo.
    c. Fetch reliable and current data from the **API-Football** service.
    d. Maintain a responsive design to ensure accessibility on various devices.

Target Audience

- Football fans, analysts, and enthusiasts.
- Individuals seeking quick squad details of Premier League clubs.
- Sports researchers looking for reliable squad data.

Features

- **Search Functionality**: Users can search for any Premier League club from 2024/25 season, by name or nickname.
- **Squad Information Display**: Showcases player details like:
    - First and Last Name
    - Position
    - Age
    - Current Photo
- **Integration with API-Football**: Utilizes this external API to fetch reliable data on clubs and players.
- **Responsive Design**: Optimized for both desktop and mobile devices.

- **Scalability**: Built to support future enhancements, such as match stats or historical data.

Architecture

React Frontend App (transferroominterviewapp.client):

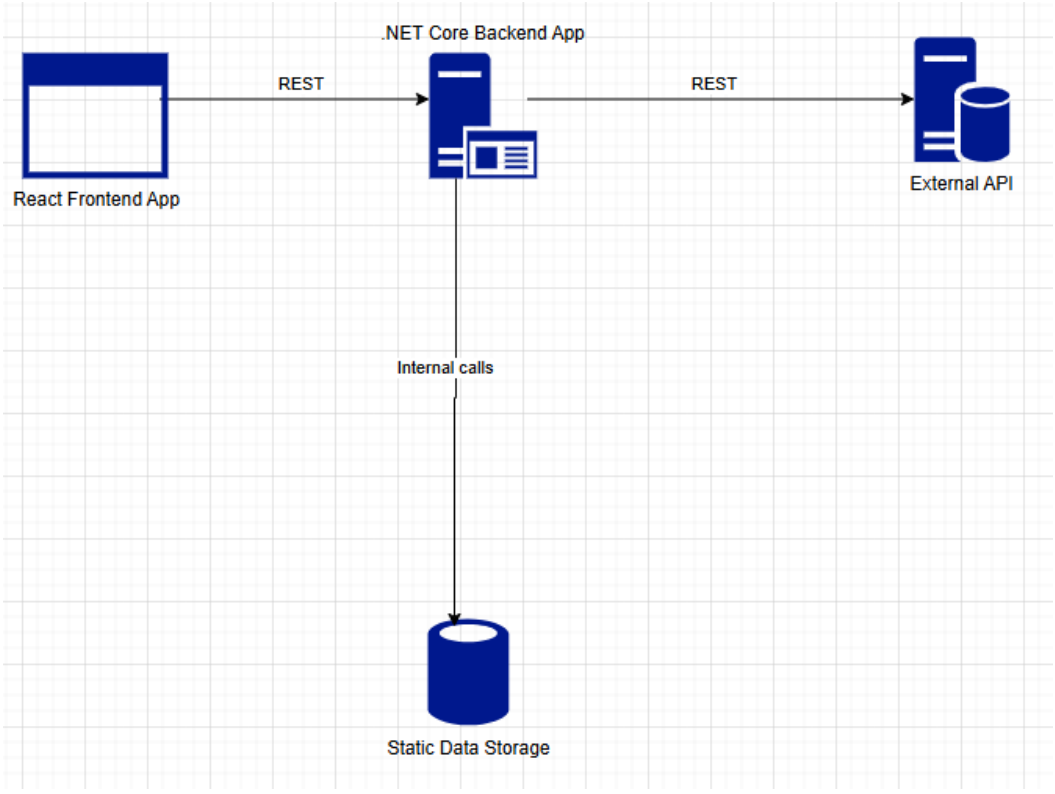
- Developed with React for an interactive and responsive user interface.
- Uses React Query for API communication with Backend app.

.NET Core Backend App (TransferRoomInterviewApp.Server):

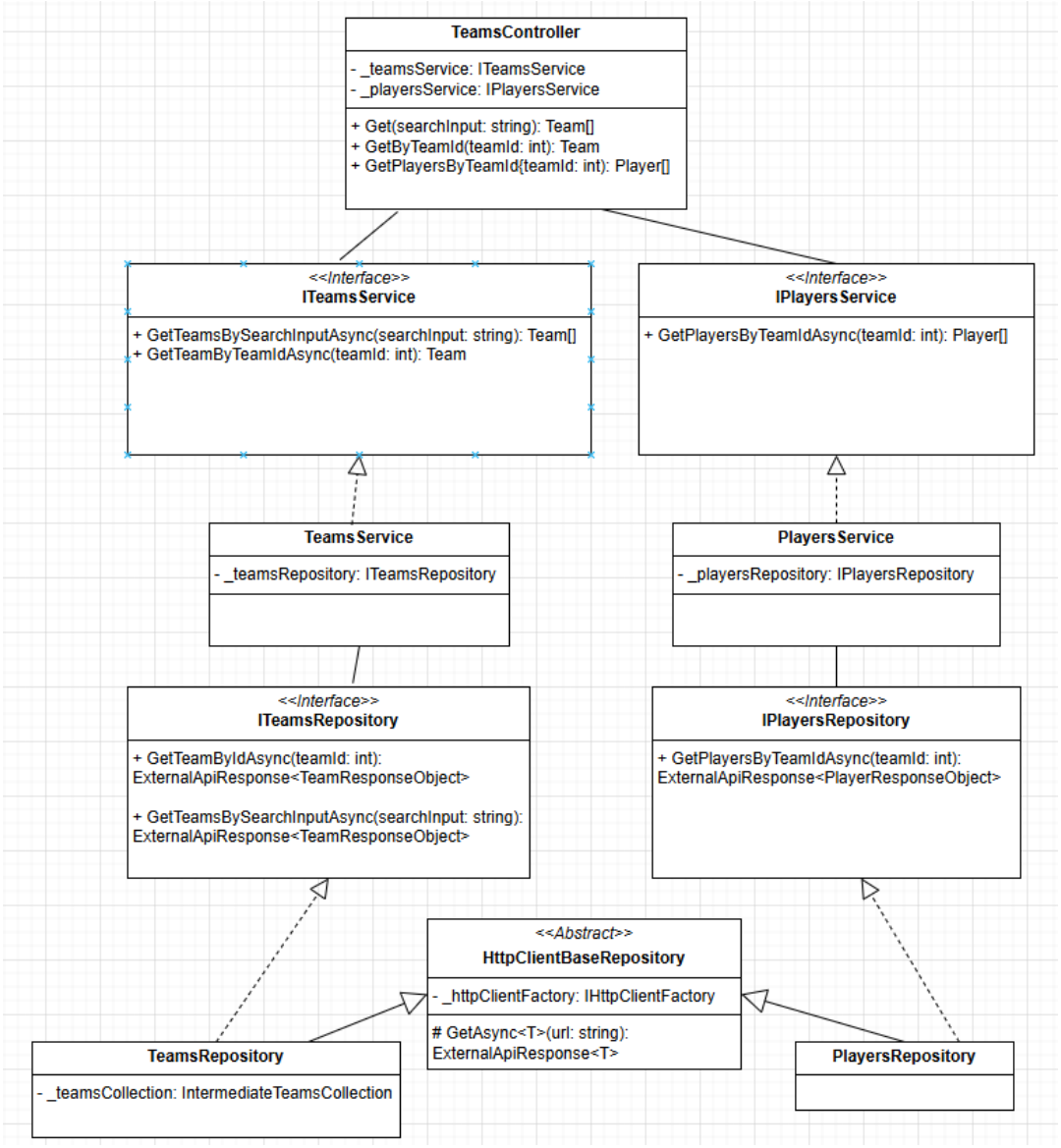
- Provides RESTful API for managing data requests.
- Uses HttpClient to communicate with **API-Football**.
- Maps the response from **API-Football** into domain models**.**
- Contains local data mentioned below, in further details.

Static Data Storage:

- Although, application communicates with external API, it still keeps some local data (as a singleton) to provide searching by nickname feature. Unfortunately, chosen API doesn't provide such feature out of the box. Also, due to the free tier limitations of the API, actual list of Premier League clubs from season 2024/25 is also stored inside these local data.



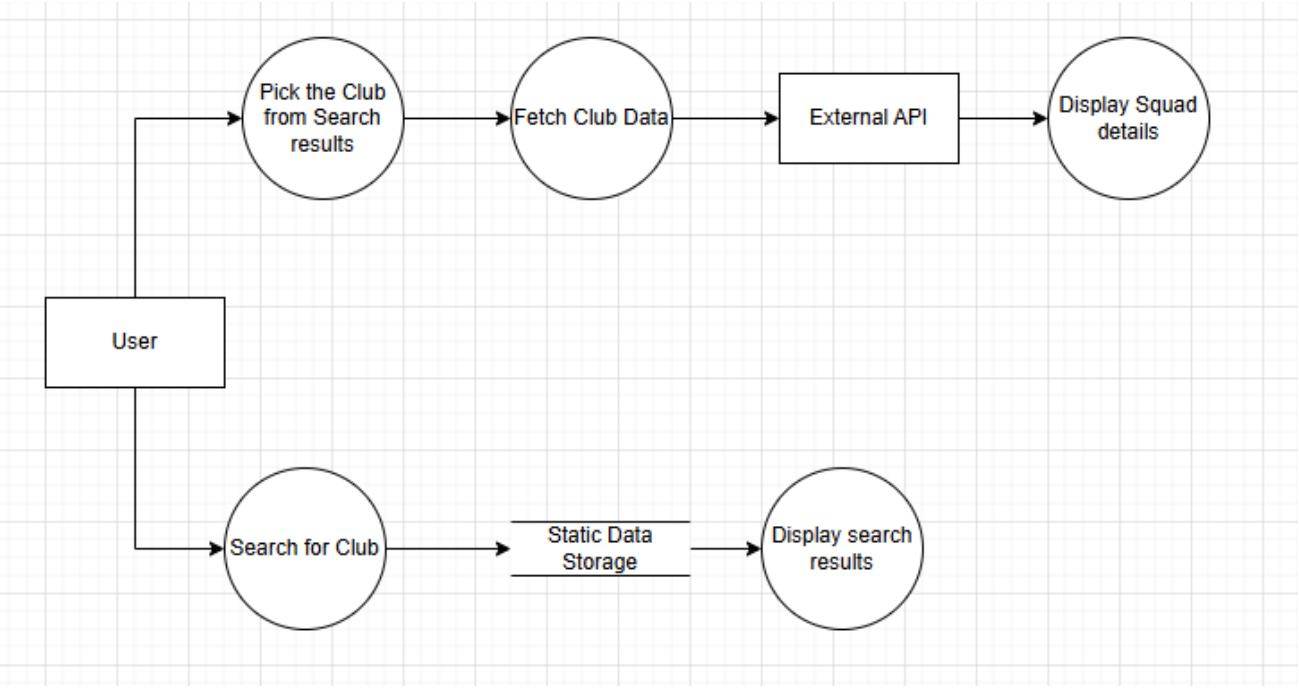
High-level architecture of the application



UML Diagram for Backend application

Data Flow:

1. **User** searches for a club using a Frontend app.
2. Request reaches the Backend app that performs search using data from the **Static Data Storage**.
3. Once the matching teams are found, data are displayed on the Frontend.
4. **User** picks the club from the displayed list.
5. Request for club details is sent to the Backend, which sends the request to **API-Football** for latest details and squad.
6. To reduce the payload of single request, details and squad requests are split into two separates.
7. **API-Football** returns the data that are later displayed using the Frontend app.



Data Flow Design for the main use cases of the app.

Technology Stack

- **Frontend**: React, React Router, React Query, Bootstrap, Vitest
- **Backend**: .NET Core, Swagger
- **External API**: **API-Football**

Deployment

The Premier League Squads Finder is being deployed as a single service using **Azure App Service**, with continuous deployment set up via **GitHub Actions** connected to the repository's main branch.

Every push to main branch, triggers the workflow defined in *.github/workflows/main_app-interviewapp.yml* file.

Build:

1. Initial setup with branch checkout, as well as installation of necessary packages and dependencies
2. Actual build of the application
3. Backend project unit tests execution
4. Frontend project ESLint check
5. Frontend project unit tests execution using Vitest
6. Preparation of the published code as an output artifact

Deploy:

1. Download of the artifact outputted at the end of **Build** step
2. Login to Azure
3. Deployment of the application into chosen Azure resource

Assumptions

- **API-Football** is the primary source for Premier League squad data.
- Developer contains free subscription to access additional features of **API-Football,** which is limited to some historical data (hence the usage of static data for storing teams from 2024/25 season), lack of nicknames support, and limitation of 100 requests per day.
- Users will primarily search for clubs using exact or partial names or one of existing nicknames chosen based on club's Wikipedia pages.

Future Enhancements

- **Advanced Search**: Enable filtering by player positions or nationalities.
- **Live Match Stats**: Extend integration with **API-Football** for real-time match updates.
- **Club History**: Add historical data about clubs, such as past seasons' performance.
- **User Accounts**: Allow users to save favorite clubs or players.
- **Multiple Deployment Environments:** Split the workflows between dev and production pipelines to reduce the risk of incorrect application behavior on production level.

Technical Challenges and Considerations

- **Dealing with External API limitations:** Using free version I had access to a lot of data but some of the more crucial for this task, were behind the paywall. Especially, filtering

teams in leagues by seasons was reduced only to some of the three historical ones. Hence the decision to prepare a Static Data Storage as a bridge between search feature and fetching for squad data.

- **Using GitHub Actions for the first time:** As a challenge for myself, I decided to pick GitHub Actions as a CI/CD solution. Main idea behind it was to keep the infrastructure as thin as possible, without reaching out to Azure DevOps or Jenkins, since my code is already on the GitHub repository. I was positively surprised that Azure contains very seamless way to connect with GitHub and utilize their Actions feature.