# Premier League Squad Finder - High-Level Design Document

## 1. Overview

The Premier League Squad Finder is a full-stack web application that allows users to search for English Premier League teams and view their current squad information. The application integrates with a third-party API to retrieve data, and includes frontend, backend, and CI/CD pipeline components.

## 2. Architecture

### Components

- **Frontend**: React application created with Vite.
- **Backend**: ASP.NET Core Web API (.NET 8).
- **API Integration**: Uses API-Football https://www.api-football.com.
- **CI/CD**: GitHub Actions workflow.

### Folder Structure

```
SquadFinder/
├── backend/              # ASP.NET Core Web API
|   ├── SquadFinder.Api/     # Main Web API project
|   └── ...
├── frontend/             # React + Vite frontend
├── .github/workflows/       # GitHub Actions workflow file
```

## 3. API Selection

Requirements:

- Premier League squads per season (2024/5)
- Player details (including birthdate, position, nationality, etc.)
- Good documentation
- A free tier suitable for development and evaluation

## API Candidates:

**1. API-Football (by API-Sports)**

🔗 https://www.api-football.com/

Pros:

- Supports filtering squads by season and league.
- Well-documented Swagger UI and good developer portal.

Cons:

- Player birthdates not included in squad endpoint — requires extra API call per player.
- Team nicknames not included.
- Premium seasons (e.g., current ones) require paid plans.

**2. Football-Data.org**

🔗 https://www.football-data.org/

Pros:

- Free to use with basic tier.
- Supports filtering squads by season and league.

Cons:

- Teams endpoint doesn't include player photos. It will require extra api call per each player to get image.
- Team nicknames not included.

**3. SportMonks Soccer API**

🔗 https://sportmonks.com/soccer-api/

Pros:

- Rich player data including birthdates, nationality, images.
- Season and league filtering.

- Includes nicknames and team metadata.

Cons:

- No long-term free tier (14-day trial only).

**4. TheSportsDB**

🔗 https://www.thesportsdb.com/api.php

Pros:

- Community-curated data, includes images, nicknames, logos.
- Player bios and team descriptions available.

Cons:

- Data quality and consistency can vary (community-driven).
- Only older version of API available in free subscription

## Choice:

**API-Football** was chosen for the following reasons:

- Season-based squad queries supported
- Provides all necessary data instead of team nicknames (extra work, no mandatory)
- Good documentation
- **Free tier** – the most important aspect in terms of homework task

Issues:

- Squad endpoint returns squad data that doesn't include player birthdate that we need to have.

Endpoints:

- https://v3.football.api-sports.io/teams?league=legueId&season=season
  Returns the list of teams and seasons in which the player played during his career.
  parameters: league – 39 will be the premier league value, season – season f.e 2023

- https://v3.football.api-sports.io/squad/team=teamId
  Return the current squad of a team when the team parameter is used. When the player parameter is used the endpoint returns the set of teams associated with the player.

  Doesn't include birth date of players!

- https://v3.football.api-sports.io/players/profiles?playerId=playerId
  Return extra data for player. Needed to get Birth data

### Caching Strategy

To minimize dependency, cost of usage of external API and improve performance:

- Caching /teams endpoint response with In-memory cache by using key "teams:league:{FootballApiPremierLeagueId}:season:{season}"
- Caching /players/squads endpoint response with In-memory cache by using key "squad:{teamName}"
- Caching /players/profile endpoint response
- Default expiration: 1 hour

## 4. Backend Design

### Key Technologies

- ASP.NET Core 8
- FluentResults (error handling)
- In-memory caching via IMemoryCache

### Responsibilities

- **SquadController**: Accepts requests with a team name and season, validates parameters, and delegates work to a service.
  - **FootballApiService**: Handles API communication, caching, and transformation of raw API data to domain models.

## 5. Frontend Design

### Key Technologies

- React

- TypeScript
- CSS modules for styles
- Vite
- Axios (HTTP client)

### Features

- Search bar for team name
- Dropdown to select season (2020-2025) (in free subscription football api only allow to query seasons older than 2023)
- Grid layout to display squad details with photo, position, and birthdate

### Component Breakdown

- **SquadSearcher.tsx**: Component for search, loading, and error state
- **SquadViewer.tsx**: Pure component to render squad data
- **squadService.ts**: Contains Axios call to backend

## 6. CI/CD Pipeline (GitHub Actions)

CI/CD configuration is stored in .github\workflows\deploy.yml file

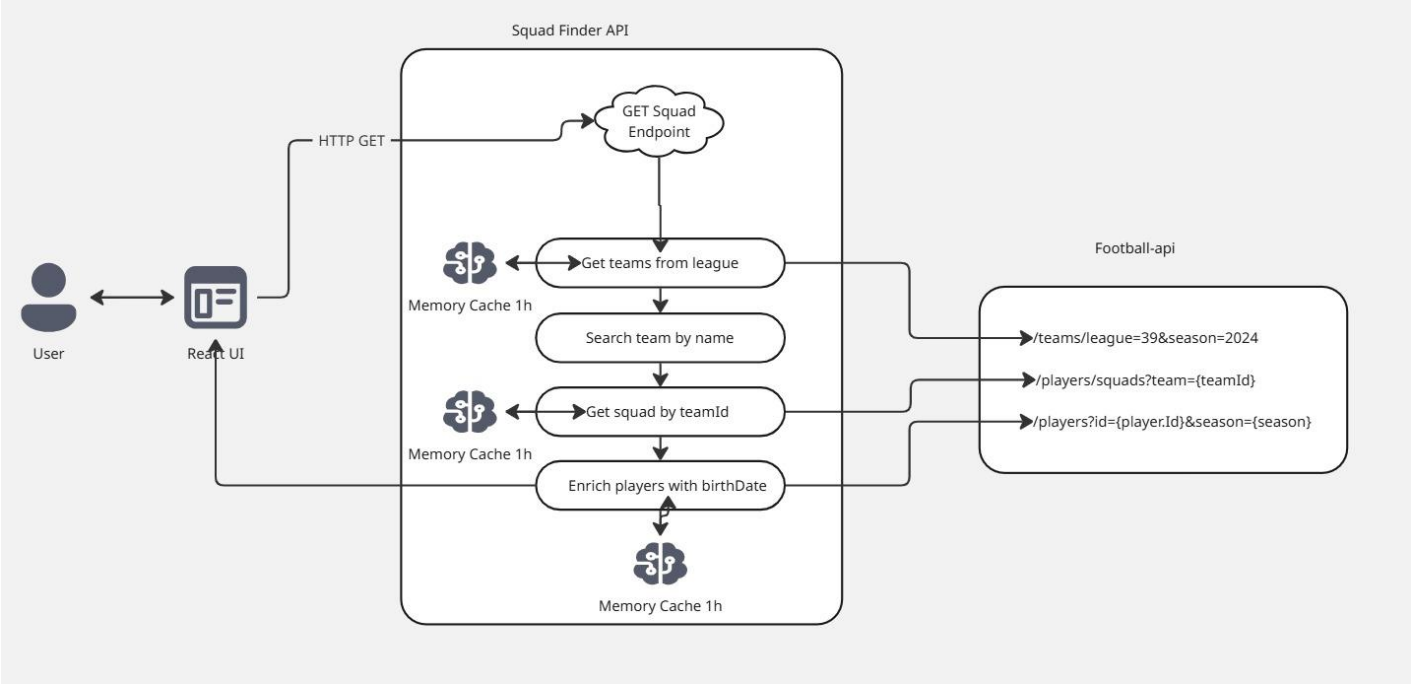
### Steps

- Read env variable and secrets from GitHub actions ApiFootball__ApiKey, ApiFootball__BaseUrl
- Checkout code
- Setup .NET and Node
- Build and test backend and frontend
- Publish backend artifacts
- Optionally deploy to a hosting environment (e.g., Azure), Not included here

### Configuration Notes

- Backend project path is backend/SquadFinder.Api
  - Output folder is /publish

## 7. Overall design




## 8. Summary

This application demonstrates a scalable, maintainable full-stack design that integrates external services with clean separation of concerns, caching, and extensibility. The CI/CD pipeline ensures smooth automated builds and deployment, and the structure supports future enhancements easily.