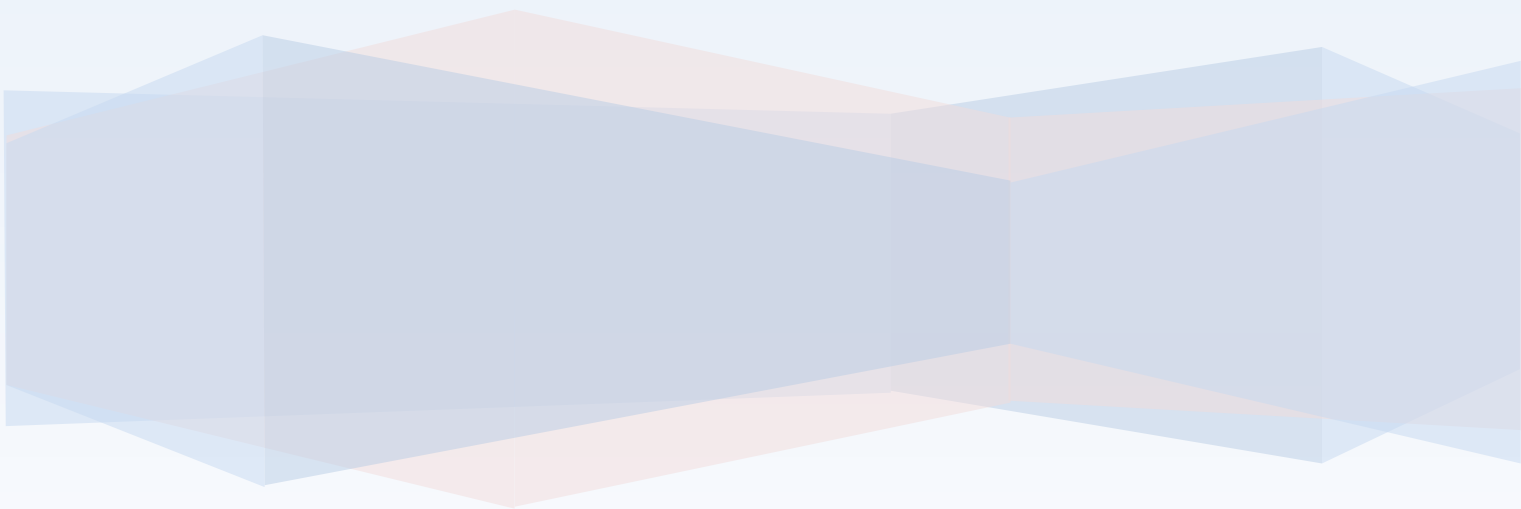


COS30043 – Interface Design and Development

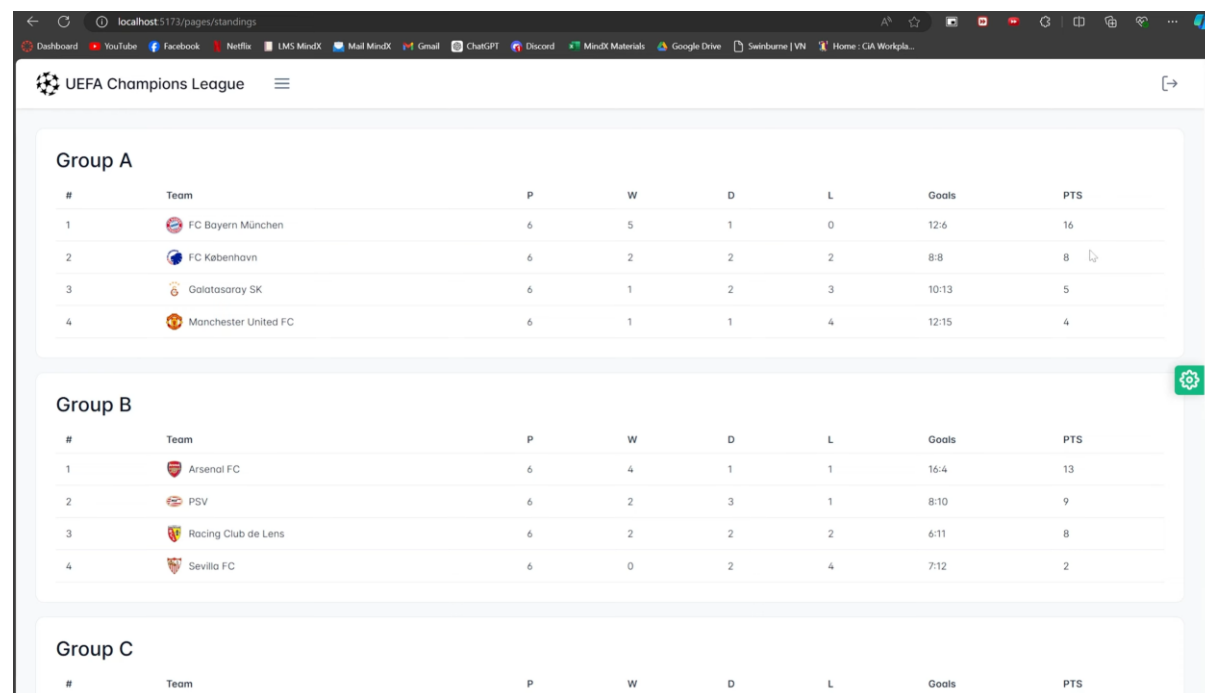
10.2HD – High Distinction Project

Huy Vu Tran - 104177995



1. Introduction

In this report, I will discuss the process of creating a single-page application (SPA) with a focus on the Standings Page of my custom-app project. Specifically, I will demonstrate the coding involved in implementing the Standings Page, which displays the group standings from A to H, along with additional relevant information.



UEFA Champions League							
Group A							
#	Team	P	W	D	L	Goals	PTS
1	FC Bayern München	6	5	1	0	12:6	16
2	FC København	6	2	2	2	8:8	8
3	Galatasaray SK	6	1	2	3	10:13	5
4	Manchester United FC	6	1	1	4	12:15	4

Group B							
#	Team	P	W	D	L	Goals	PTS
1	Arsenal FC	6	4	1	1	16:4	13
2	PSV	6	2	3	1	8:10	9
3	Racing Club de Lens	6	2	2	2	6:11	8
4	Sevilla FC	6	0	2	4	7:12	2

Group C							
#	Team	P	W	D	L	Goals	PTS

2. API service

Before delving into the coding details, I will provide some background information on the API I'm using for this project - the footballAPI. This API allows me to retrieve football data from some of the most well-known leagues in the world. I will explain the steps required to register for the API and obtain the necessary API key to access the data.

Available resources

See all available endpoints underneath. See Filtering table at the very bottom to see how to pass filters in an adequate format. You can also get an overview by running all available calls through Postman by importing this collection after download.

(Sub)Resource	Action	URI	Filters	Sample
Area	List one particular area.	/v4/areas/{id}	-	Open
Areas	List all available areas.	/v4/areas/	-	Open
Competition	List one particular competition.	/v4/competitions/{id}	-	Open
Competition	List all available competitions.	/v4/competitions/	areas={AREAS}	Open
Competition / Standings	Show Standings for a particular competition.	/v4/competitions/{id}/standings	matchday={MATCHDAY} season={YEAR} date={DATE}	Open
Competition / Match	List all matches for a particular competition.	/v4/competitions/{id}/matches	dateFrom={DATE} dateTo={DATE} stage={STAGE} status={STATUS} matchday={MATCHDAY} group={GROUP} season={YEAR}	Open
Competition / Teams	List all teams for a particular competition.	/v4/competitions/{id}/teams	season={YEAR}	Open
Competition / (Top)Scorers	List top scorers for a particular competition.	/v4/competitions/{id}/scorers	limit={LIMIT} season={YEAR}	Open
Team	Show one particular team.	/v4/teams/{id}	-	Open
Team	List teams.	/v4/teams/	limit={LIMIT} offset={OFFSET}	Open
Match	Show all matches for a particular team.	/v4/teams/{id}/matches/	dateFrom={DATE} dateTo={DATE} season={YEAR} competitions={competitionids} status={STATUS}	Open

To begin, I will show how the retrieved data from the footballAPI looks like in Postman. The API response contains an array of standings, as well as other pieces of information. I will focus on filtering out the unnecessary parts and concentrating solely on the standings data.

The screenshot shows a Postman interface with a GET request to `https://api.football-data.org/v4/competitions/2001/standings`. The request has an `X-Auth-Token` header with the value `686e01d4cc1c48b5a67d06f3c6700a3`. The response body is a JSON object:

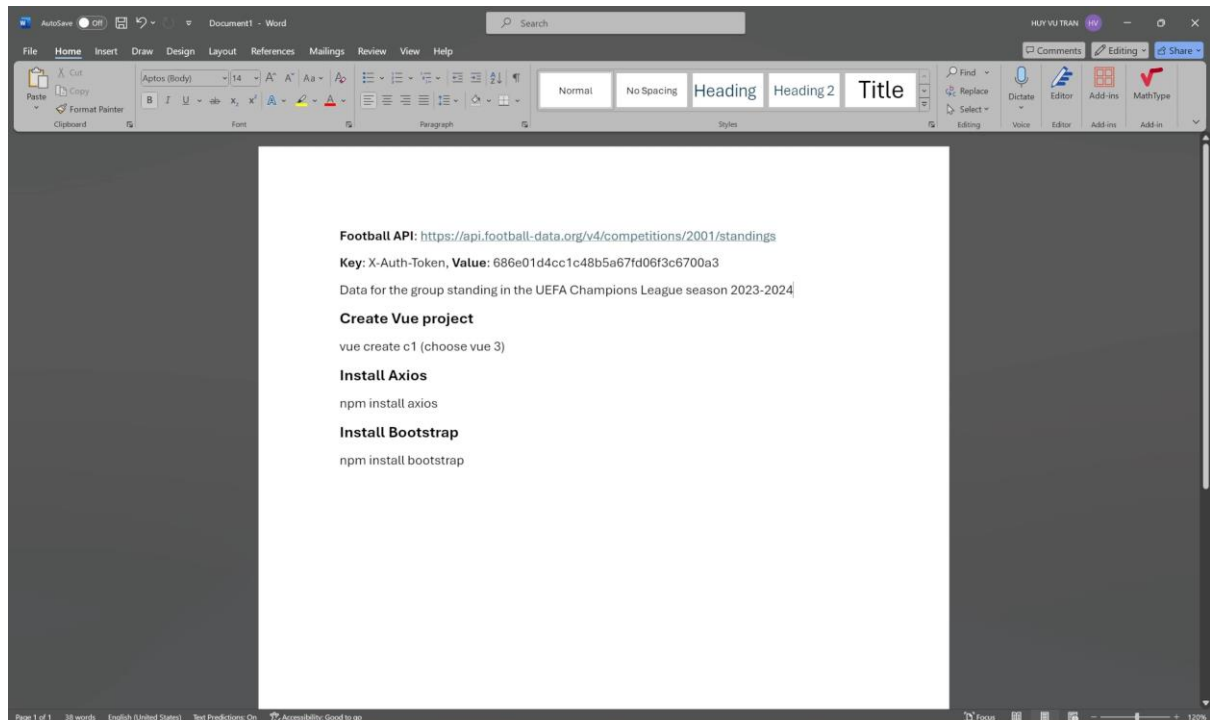
```

{
  "currentMatchday": 6,
  "winner": null,
  "standings": [
    {
      "stage": "GROUP_STAGE",
      "type": "TOTAL",
      "group": "Group A",
      "table": [
        {
          "position": 1,
          "team": {
            "id": 5,
            "name": "FC Bayern München",
            "shortName": "Bayern",
            "tla": "FCB",
            "crest": "https://crests.football-data.org/5.svg"
          }
        }
      ]
    }
  ]
}

```

3. Project configuration

Next, I will walk through the steps required to configure the project before coding. This includes creating a Vue project using the `vue create c1` command, installing Axios to make HTTP requests, and installing Bootstrap to style the website.



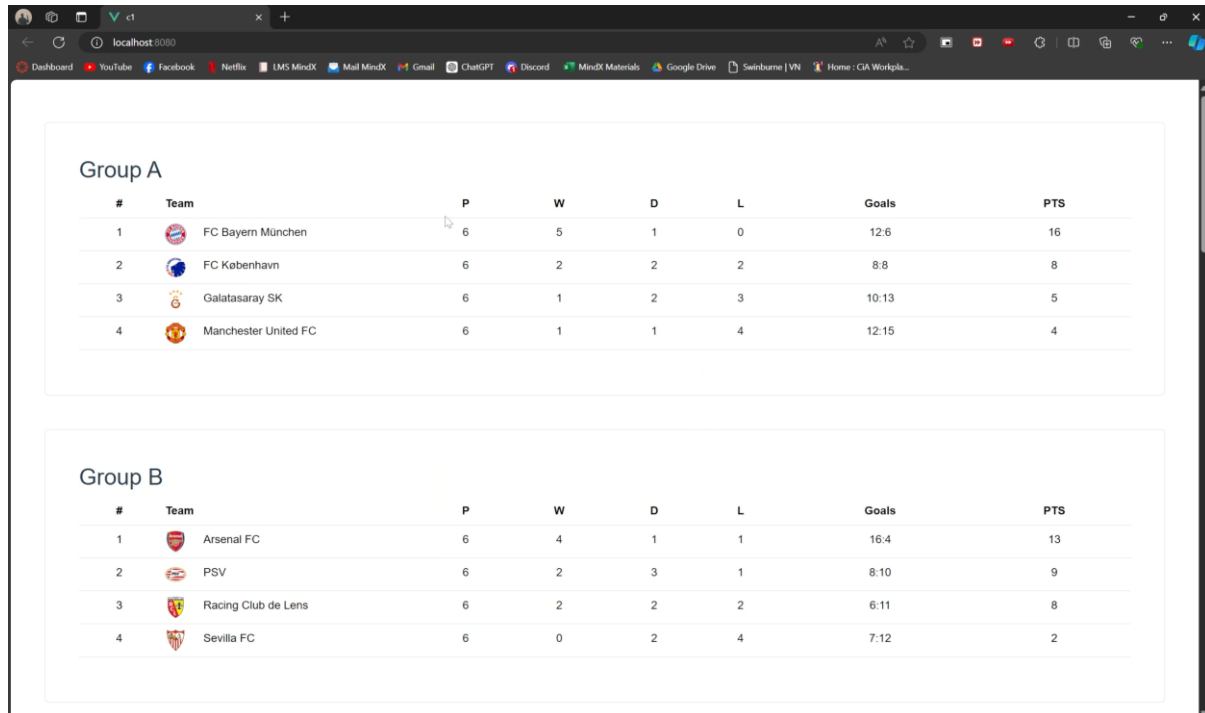
4. Coding process

One of the challenges I encountered during the development process was the CORS (Cross-Origin Resource Sharing) error. This error occurs when the web application tries to make a request to a different domain, and the browser enforces a security feature known as the Same-Origin Policy. To bypass these restrictions, I will explain how to set up a proxy server in the `vite.config.js` file, which will act as an intermediary, making the API requests on behalf of the application.

Moving on, I will dive into the code in the `api.js` file, where I can fetch the desired data from the API. I will demonstrate the use of Axios to create an instance with the base URL pointing to the proxy endpoint, and the configuration of the necessary headers.

Finally, I will provide a detailed explanation of the code in the `HelloWorld.vue` file, where the Standings Page is implemented. I will cover the process of fetching the

data, handling any potential errors, and the structure of the template, including the use of v-for directives to display the group standings and team information.



The screenshot shows a web application running on localhost:3000. It displays two tables representing football group standings. The first table is for Group A, and the second is for Group B. Each table lists four teams with their respective statistics: matches played (P), wins (W), draws (D), losses (L), goals scored/conceded (Goals), and total points (PTS).

Group A

#	Team	P	W	D	L	Goals	PTS
1	FC Bayern München	6	5	1	0	12:6	16
2	FC København	6	2	2	2	8:8	8
3	Galatasaray SK	6	1	2	3	10:13	5
4	Manchester United FC	6	1	1	4	12:15	4

Group B

#	Team	P	W	D	L	Goals	PTS
1	Arsenal FC	6	4	1	1	16:4	13
2	PSV	6	2	3	1	8:10	9
3	Racing Club de Lens	6	2	2	2	6:11	8
4	Sevilla FC	6	0	2	4	7:12	2

5. Conclusion

In conclusion, I have covered the process of creating a single-page application with a focus on the Standings Page in my custom-app project. I have provided an overview of the footballAPI, explained how to configure the project, and demonstrated the coding involved in fetching and displaying the data. By addressing the CORS issue and setting up a proxy server, I was able to seamlessly integrate with the external API and deliver a functional Standings Page. This report should serve as a comprehensive guide for anyone interested in building a similar type of SPA with a Standings Page feature. Here's the video demonstration for this project: <https://www.youtube.com/watch?v=BV4uUZRxi-A>