

CA326

Technical Specification

Cian Sullivan, Leigh Reilly



0. Table of contents

1. Introduction

1.1 Overview

1.2 Glossary

2. System Architecture

2.1 Django Web Framework

2.2. Bootstrap

2.3 API's

2.4 BeautifulSoup

3. High-Level Design

4. Problems and Resolution

4.1 Android studio to Django

4.2 Upgrading API plan

4.3 Added BallDontLie Api

4.4 Added a get_player_id function

4.5 Added a get_team_logo function

4.6 Using cookies instead of a register and login function

5. Installation Guide



1. Introduction

1.1 Overview

ScoreCenter is a WebApplication that is designed for use on any desktop or laptop device. It allows users to stay up to date with the latest results, fixtures and standings of the NBA league, as well as providing statistics for the teams and players. Users can also compare statistics between specific teams and players of their choice. Users can also see news articles relating to any team of their choice in the NBA. Our website also features an option to choose a favorite team which stores the user's preference using cookies. Whenever a user loads back into this page they are given the option to see either news or fixtures based on their favorite team.

The ScoreCenter WebApp is built using Python and the Django web framework. The system interfaces with third party API's (API-Basketball, BallDontLie) to retrieve real time data such as the fixtures, results and statistics.

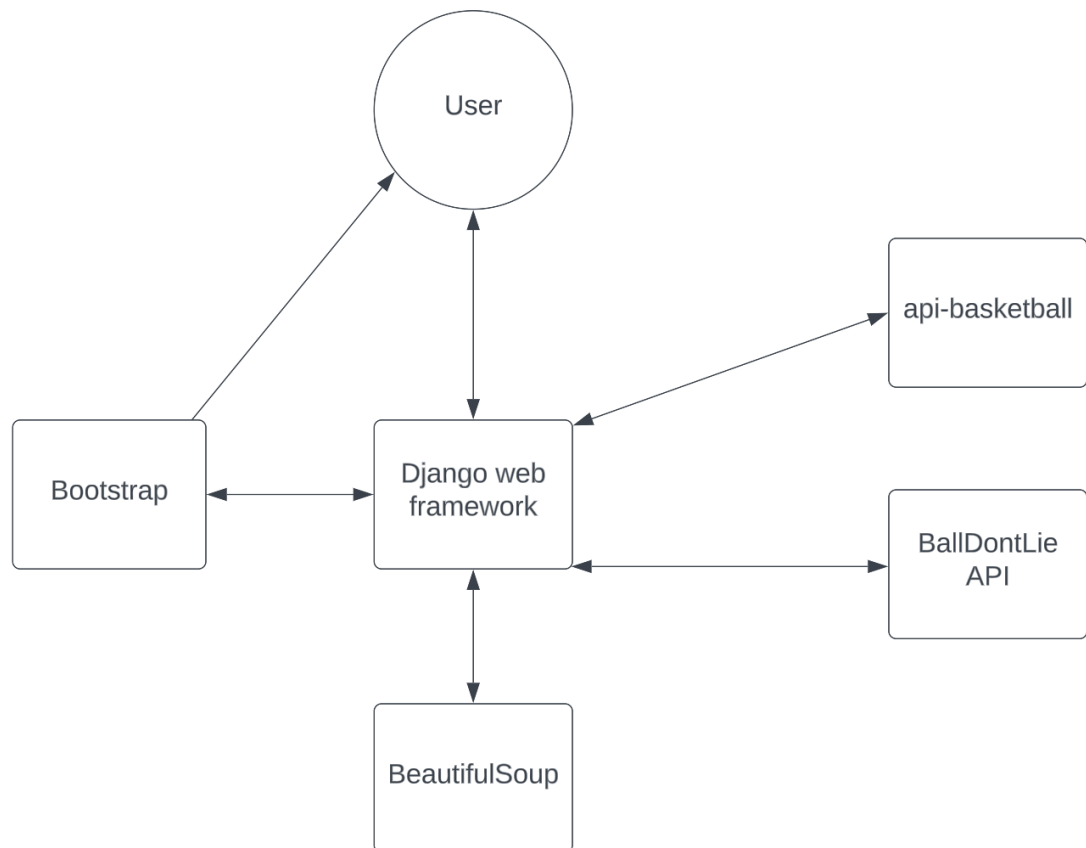
1.1 Glossary

API - Application Programming Interface

Django - a high-level Python web framework

Real-time - information that is delivered immediately after it is processed

2. System Architecture



2.1 Django Web Framework

The back-end of the system will be built using the Django web framework. Django will handle server-side logic and data processing, and will interact with the APIs used to get information on the NBA league. When a user makes a request like comparing players, Django will handle the request and route it to the appropriate view, in this case the `compare_players` view.



2.2. Bootstrap

The front-end of the system will include Bootstrap, the front-end framework. This provides a consistent and responsive user interface across the website. Bootstrap helps with styling the site with the built in styles they have.

2.3 API's

The system uses two third-party APIs - api-basketball and Balldontlie api - to get information on basketball.

Api-basketball is used for the results, the individual matches, the standings and the team statistics.

Balldontlie api is used for the fixtures and the player statistics.

Django will make HTTP requests to these APIs and parse the JSON responses to get the relevant data.

2.4 BeautifulSoup

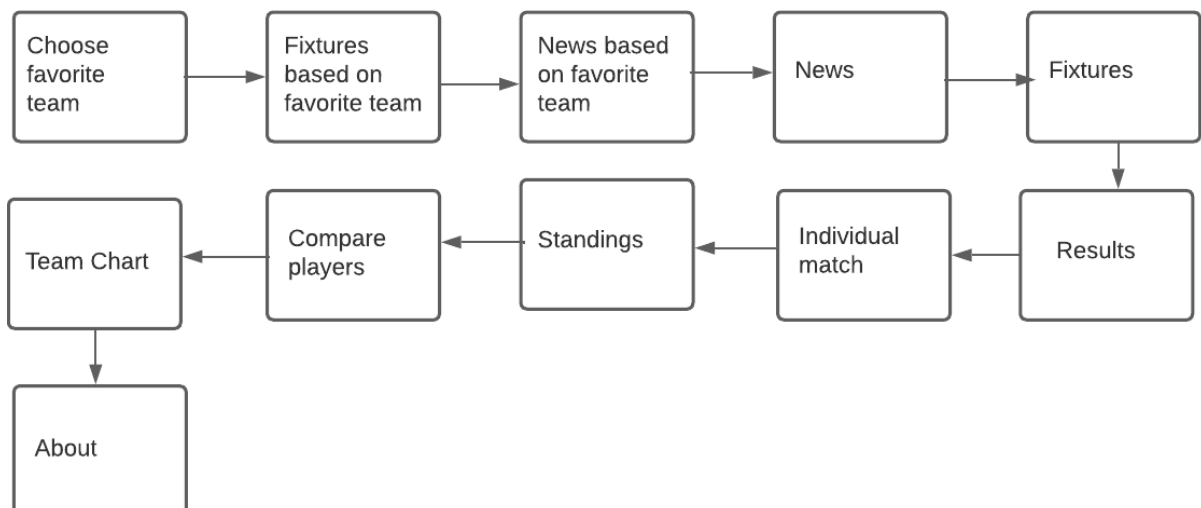
Beautifulsoup is used in our project to scrape ESPN for news articles on each team in the NBA. This made the news page possible as it was able to pull news articles which we then displayed on our website.



3. High-Level Design

- Step 1: Choose Favorite team
 - The User chooses a favorite team which is stored in the cookies.
- Step 2: Fixtures based on favorite team
 - The User is shown fixtures and results based on their favorite team.
- Step 3: News based on favorite team
 - The User is shown news related to their favorite team.
- Step 4: News
 - The User can choose any team in the NBA and see news based upon that team.
- Step 5: Fixtures
 - The user is shown today's fixtures in chronological order. They can also enter a start and end date and see fixtures based upon those dates.
- Step 6: Results
 - The user is shown the most recent results in chronological order. The user can click on the “view” section of any match to see the individual match details.
- Step 7: Individual Match
 - The user is shown points in each quarter and final score of each game. The page also displays the last 5 head to head matches between the two teams. There is also an embedded link to a video recap of the match, and an option to share this to facebook.

- Step 8: Standings
 - Displays the standings of the NBA league.
- Step 9: Compare Players
 - The user can enter as many players as they choose from the NBA league. They will then be displayed the average statistics of these players for the latest season. The user can also sort the stats by clicking on the attribute headings.
- Step 10: Team Chart
 - The user can enter as many teams as they like from the NBA league. They will then be displayed a graph with the number of games won.
- Step 11: About
 - The user can see information about the website, and lists its features.





4. Problems and Resolution

4.1 Android studio to Django

When using Android Studio we struggled to get the information from the API to display on our application.. This was a major problem as we needed the results and fixtures etc. for our application. To fix this, we decided to make a web app using the Django web framework. We had previous experience with Django which made the project a lot more achievable.

4.2 Upgrading API plan


When using API Basketball, it would only allow a free amount of 100 requests a day and then you were charged for every request after that. This made it very difficult for us as we were staying conscious of being charged for each request that was made. This was a problem as we would stop calling the api when we went over the limit, which slowed down the workflow for a number of days. To solve this problem, we decided to buy the pro plan for the api which was only 5 euro each a month. This plan has a hard limit of 7500 requests a day which we have not gone over, so we were never restricted after this change. We were also able to work freely, knowing that we would not get charged extra.

4.3 Added BallDontLie Api

We ran into another problem in relation to API's when we realized the API we were using (API Basketball) had no option to show statistics for each player. This had completely spoiled one of our ideas for comparing players based on their stats, which was a key part in our website. We decided to do extensive research into other Basketball API's which were within our price range. We eventually found another basketball API called BallDontLie. It turned out this API did not have as many features as we would have liked but it did allow us to get season averages of players which was exactly what we needed. We decided to use this in conjunction with API Basketball.

4.4 Added a `get_player_id` function

When using the two API's together we ran into some problems. Each API had a particular ID for each player or team respectively. We ran into problems when we were using API Basketball to retrieve the names of the players but using BallDontLie to retrieve the statistics based on the players names. The problem was that we needed the ID of the player to see the statistics of



that player. This would have been fine, except because we were using BallDontLie they had a different ID for each player so it did not return the correct results. To solve this we implemented a `get_player_id` function which returns the ID of a player in that API, based on their name. This solved our problem.

4.5 Added a `get_team_logo` function

We had a similar problem getting team logos, as only API-Basketball provided logos. To fix this problem implemented a `get_team_logo` function. This function makes a call to api-basketball using the team name as the search query to return the logo. We can then access logos when using the Balldontlie api.

4.6 Using cookies instead of a register and login function

We had originally planned to use a register and login functionality so the user could set a favorite team. However when it came time to implement this we realized what we wanted could be achieved much easier by using cookies to store a users favorite team. We then could show news and results based on this user's favorite team. We had now achieved the same end result we had previously wanted, but did not have to implement a register/login functionality. We also originally planned to use the Firebase database but with this new functionality, it was also unnecessary.



5. Installation Guide

Our WebApplication will currently be run on localhost rather than a server for the purposes of the demonstration. We have designed this project to be scalable so that in the future if we wanted we could turn it into a server and add more functionality.

To run the localhost on your computer, you must first clone the git repository (<https://gitlab.computing.dcu.ie/reilll38/2023-ca326-scorecentre>).

Install the necessary libraries as follows;

```
Pip install bs4
```

```
Pip install requests
```

```
Pip install matplotlib
```

Now on a command line, cd into the “2023-ca326-scorecentre/backend” folder.

Run the command “`python manage.py runserver`”.

Now paste the following into any web browser “`http://127.0.0.1:8000/`”.