**NBA Live Schedule**

**Diagram of the System**

Diagram

Description automatically generated

**Data Plan:**

1. Overall Data Structure for the GAMES:
2. The User input using which the program gets directions as to whether the user wants information on Today’s Matches or matches by a custom date.
3. The LIVE NBA data is pulled out from the API using the input of the user and the same is displayed to the user on the website.

Data that needs manipulation:

1. Convert the JSON data we get from the API using the following steps:
2. According to the data we get using the user’s input, using a for loop we go through all the desired data from the JSON data and store each detail in their respective variable, which is finally printed out to the user on the website.
3. If the user selects the Today’s Matches option, using the DateTime module in python, we get the live date using which we can get the information in the same manner.
4. But since the API server Is located on the eastern side of the world, I had to use concepts like string slicing, conditionals, and various python commands to fix this problem.
5. I was successfully able to fix this issue for New year’s, leap years, the month of February, and all other months as well.
6. Finally, if the user selects the option to view games as per custom date, then they are routed to a webpage that accepts the desired date.
7. Using this desired date, the process mentioned above is followed to get the details on the matches.
8. Visual Layout for Games:
9. Using an unordered list, we display the following:
10. Home Team VS Visitor Team
11. Their Score
12. The Quarter that is going on
13. The Clock time of the respected quarter
14. I have added borders and the HOVER feature after each list element so that the user can have a seamless experience.
15. I have also accessed the logo of each team using the API and printed the same for an aesthetic experience.
16. To avoid unnecessary elements, I have hidden the bullet points that are usually present in unordered lists which indeed adds to the look of the webpage.
17. I have used various CSS properties like box-shadow, padding, position, etc. to create an appealing webpage.
18. I have also used CSS properties like cursive, text-shadow, line height, margin, etc. to make the headers of the pages appealing.
19. Sample View of the Games Pages (In order of their occurrence)

A picture containing text

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, application, website

Description automatically generated

Graphical user interface, application, website

Description automatically generated

1. Overall Data Structure for Standings:
2. The user’s input is whether they want eastern conference standings or Western conference standings.
3. Using this respected JSON data is pulled from the API.
4. According to the data we get using the user’s input, using a for loop we go through all the desired data from the JSON data and store each detail in their respective variable, which is finally printed out to the user on the website.
5. Using a lambda function we access the rank of each team to sort them according to their respective ranks in the conference.
6. To get the streak of each team we access the streak of each team using the JSON data. But to to determine if it is a winning or a losing streak, we access the winStreak variable in the JSON data, which has a value of True or False. Using conditional statements, we assign the “W” or “L” to indicate if it is a winning or a losing streak.
7. Also, according to the conference selected by the user, the Title is also changed accordingly to show whether it is an Eastern or a Western conference standing.
8. Finally, according to the conference selected by the user, the following data is printed out:
9. Teams are sorted according to their rank (using a lamba function)
10. Team Name
11. Wins
12. Losses
13. Home Stats
14. Away Stats
15. Streak
16. Visual Layout for Standings:
17. Using a table element in HTML we display all the above data.
18. The Headings of each column are displayed using the <thead>, <tr>, and the <th> tags
19. The rows are displayed using the <tbody>, <tr>, and the <td> tags
20. For the background color, I used the linear-gradient property of CSS to have a multi-tone background.
21. To make the table look seamless, I removed the borders using the border-collapse and overflow properties.
22. Sample View of the Standings Pages (In order of their occurrence)

A picture containing text

Description automatically generated

**Text

Description automatically generated with medium confidence**

**Graphical user interface, text, chat or text message

Description automatically generated**

Calendar

Description automatically generated

Text

Description automatically generated with medium confidence

Calendar

Description automatically generated