MatchMatrix WebApp

Data Preprocessing Using Python:

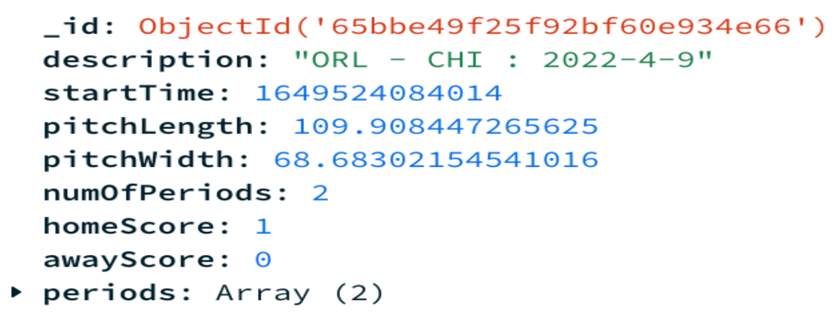
I utilized Python for preprocessing the 'tracking_data.csv' file, as Python seamlessly handles this type of data. The script comprises two functions: 'calculate_distance' and another function for dataset splitting. The 'calculate_distance' function calculates and returns the distance between two points. The second function is designed to collect data for a single player, sort it by 'frame_idx,' and then calculate the distance by summing the distances between consecutive frames until the total reaches a minimum distance of 10. Once this condition is met, it generates a JSON file containing the coordinates of the first and last positions, along with the frame range. This approach enhances the visualization of player runs, making them more apparent, especially when they occur within smaller frame ranges. To manage memory efficiently, I processed only the data for the first period**.** The Python script is dynamic and can handle data for all players and all periods efficiently

Database Creation:

After obtaining all the data in JSON format from the script, I employed a NoSQL database, specifically MongoDB. I designed schemas for both the processed data and match data. I made sure that the entire database setup was dynamic, capable of accommodating data from any soccer match, and avoided hardcoding any specific data.

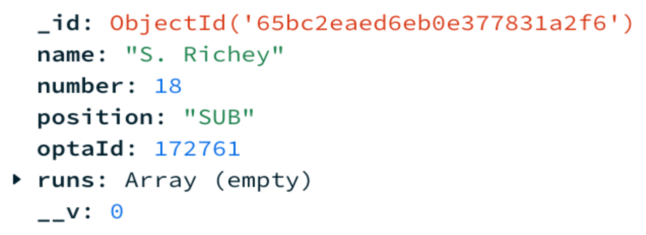
The database consists of five documents:

- MatchData: This document contains all the essential information for the current match.



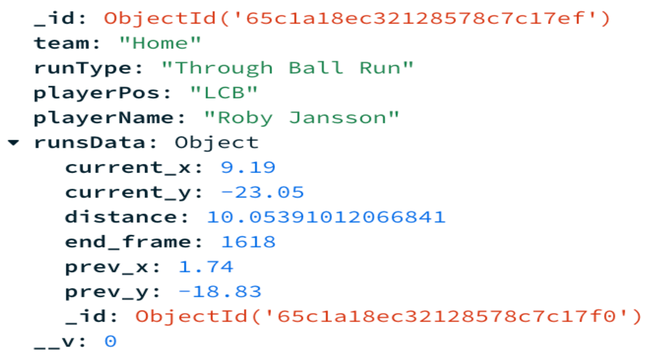
```
  _id: ObjectId('65bbe49f25f92bf60e934e66')
  description: "ORL - CHI : 2022-4-9"
  startTime: 1649524084014
  pitchLength: 109.908447265625
  pitchWidth: 68.68302154541016
  numOfPeriods: 2
  homeScore: 1
  awayScore: 0
▸ periods: Array (2)
```

- HomePlayer and AwayPlayer: These documents store JSON objects with information for each player, offering a flexible way to handle player data.



```
  _id: ObjectId('65bc2eaed6eb0e377831a2f6')
  name: "S. Richey"
  number: 18
  position: "SUB"
  optaId: 172761
▸ runs: Array (empty)
  __v: 0
```

- HomeDataRes and AwayDataRes: These documents contain JSON objects that capture user interactions with the website when categorizing player runs.



```
 _id: ObjectId('65c1a18ec32128578c7c17ef')
 team: "Home"
 runType: "Through Ball Run"
 playerPos: "LCB"
 playerName: "Roby Jansson"
▾ runsData: Object
   current_x: 9.19
   current_y: -23.05
   distance: 10.05391012066841
   end_frame: 1618
   prev_x: 1.74
   prev_y: -18.83
   _id: ObjectId('65c1a18ec32128578c7c17f0')
 __v: 0
```

<u>This approach ensures a scalable database structure, adaptable to various soccer matches without requiring fixed data entries</u>

Match Visualization:
This phase posed significant challenges due to differences in field dimensions compared to the website's view dimensions. Additionally, the coordinates in the dataset were based on real-world dimensions. To address this, I developed algorithms within the frontend, specifically in the 'controllers->scalePointsToPixels' module. These algorithms scale down all coordinates to match the view dimensions, making the visualization more realistic. To dynamically display the match lineup, I employed dynamic components, pulling data from the database.

Runs Visualization:
To provide users with a comprehensive experience for watching and categorizing runs, I implemented several features. Users have the flexibility to select the team and player position and initiate watching using controller buttons. I integrate the Framer Motion library to ensure runs visualization is both realistic and precise. Moreover, users can create new categories in addition to the provided ones, allowing them to categorize runs while actively watching. After clicking 'submit,' the database stores all user interactions for future reference and analysis.

Runs Analysis:
In this phase, all the data retrieved from the database is utilized to provide users with comprehensive insights. Users have the opportunity to view the analysis for the entire team. There are two main types of outputs available:

- **Clickable Cards**: These cards provide information on different run types along with relevant details about the runs and player information. Users can click on these cards to access more information.
- **Charts:** Data is visualized through two types of charts. The first chart displays all player names categorized by the distance covered, offering an overview of player performance. The second chart categorizes runs based on user-defined categories, providing a breakdown of run types as categorized by the user previously.

Areas for Future Improvement:

- Convert the Python Script into an API: Transforming the existing Python script into an API will enable easy access through endpoints in the web application. This would facilitate seamless data retrieval and interaction with the script's functionality.

- Integrate Three.js and React Fiber Libraries: Incorporating Three.js and React Fiber libraries can enhance the visualization capabilities by transitioning from 2D to 3D visualizations. This would provide a more immersive and realistic representation of the soccer match.

- Real-Time Data Processing and Visualization: Implementing real-time data processing and visualization capabilities would allow users to observe and analyze live match data as it happens. This could include real-time tracking of player movements, events, and statistics.

- Enhanced Player and Ball Movement Visualization: Introducing the ability to visualize the movement of all players and the ball simultaneously when observing a specific player's actions can improve the accuracy of run categorization. This feature would assist users in better understanding player interactions and team dynamics during a match