Visualization of Sports Data

A Project Report

Presented to

The Faculty

of the Department of Computer & Electrical Engineering/Computer Science

California State University, Bakersfield

In Partial Fulfillment

of the Requirements for

Senior Project

in

Computer Science

By

Nathan Wardinsky | Aaron Pacheco | Jesse Garcia | Alejandra Villafan | Gustavo Jimenez

December 2024

ABSTRACT

Visualization of Sports Data

By

Nathan Wardinsky | Aaron Pacheco | Jesse Garcia | Alejandra Villafan | Gustavo Jimenez

Sportsball is an interactive baseball website that uses real-world baseball data to predict the outcome of batter vs pitcher matchups through a baseball simulation. The main objective is to provide an engaging experience for fans, allowing them to visualize baseball analytics through an intuitive and interactive platform.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TERMS

DBMS:	Database Management System
JSON:	JavaScript Object Notation
MLB:	Major League Baseball
HTML:	Hypertext Markup Language
XML:	Extensible Markup Language
SQL:	Structured Query Language
PDO:	PHP Data Objects

Chapter 1

Introduction

Baseball has captured the attention of communities worldwide for generations. With a rich history dating back to 1839, it has become one of the most statistically intricate sports in the world. The game involves numerous factors—such as pitches, at-bats, innings, and environmental conditions—that contribute to a vast array of individual statistics. This complexity can overwhelm fans, making it challenging to keep track of stats or engage in meaningful debates about their favorite teams.

This project aims to simplify these complexities by focusing on just two critical factors: batter and pitcher statistics. By condensing the data and presenting it visually, fans can participate in analyses and defend their teams without relying on dense mathematical arguments. The result is an accessible, engaging experience for any baseball enthusiast.

The project includes features like a media page and a player lookup column, but its core functionality revolves around five main components: the web scraper, the database, a player selection page, the "At Bat Logic" function, and the simulation page.

Web Scraper

The web scraper, developed in Python, leverages libraries such as Beautiful Soup and MySQL. Beautiful Soup enables the extraction of data from HTML and XML files, allowing us to read and scrape pages from the Baseball Reference website. This resource provides access to a comprehensive database of historical and current baseball statistics. Players are categorized as batters or pitchers, enabling precise targeting of their stats. Once identified, their statistics are saved into the database using the MySQL library to generate SQL queries.

Database Management

The project uses MariaDB as the database management system for storing player stats. PHP is utilized to access this database, particularly for retrieving data rather than saving it. With PHP’s PDO, efficient queries MariaDB can send data to the front end, where it is displayed using either PHP or JavaScript.

Player Selection Page

JavaScript powers most of the player selection page, allowing users to choose specific batters and pitchers. To ensure secure database interactions, JavaScript communicates with PHP scripts on the backend via HTML requests. These PHP scripts fetch and save player data for use in predictions generated by the "At Bat Logic" function.

"At Bat Logic" and Simulation Page

The "At Bat Logic" function works hand-in-hand with the simulation page to predict outcomes of batter-pitcher matchups. This function calculates possible results—such as outs, walks, or home runs—based on player statistics. The calculations are handled on the backend using JavaScript, keeping the process invisible to users. Instead, they see the final prediction displayed on the simulation page, making the analysis easy to interpret and engaging.

By streamlining complex baseball statistics and delivering user-friendly predictions, this project enhances the way fans interact with the game, fostering deeper engagement and informed discussions.

Chapter 2

Project Architecture

The architecture of this project can be described in 5 components. Web scraper pulling real life player data, both Present Day and Historical players, Database to facilitate pulled player data, user creation, and team lineups, Website for users to create team lineups, Backend At Bat Logic computes play outcomes based on Batter Vs. Pitcher stats and finally the Front end WebGL simulation visualizing the result of the play.

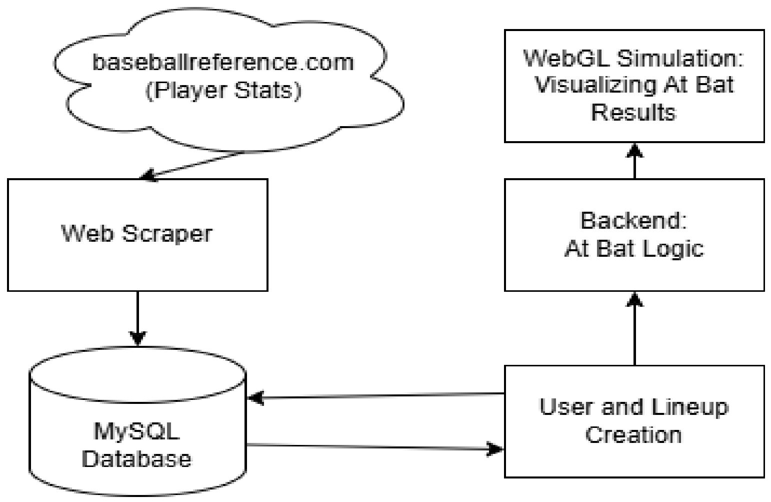

baseballreference.com (Player Stats)
WebGL Simulation: Visualizing At Bat Results
Web Scraper
Backend: At Bat Logic
MySQL Database
User and Lineup Creation

Figure 1. Flowchart of the main components of our project

First, the web scraper is in charge of pulling current MLB rosters from baseballreference.com where the individual player's stats are saved on our database. First, the web scraper pulls the full team roster before inserting each individual player who played for them in the current year. Batters and Pitchers account for the different stats at their positions.

The MySQL database plays a crucial role in storing player data, user accounts, and user-generated lineups. The database is structured to manage the relationships between users, their custom-created teams, and the players on those teams. It holds tables for players, individual batter and pitcher stats, user information, and the corresponding team lineups.

Once registered or logged in, users are directed to a custom news feed tailored to their preferred baseball team. Here, they can adjust their team preference, view relevant team news, or navigate to the team management page. On the team management page, users can create a team by entering a team name, which is stored in the database. Users then select one pitcher and five batters from the list of available players to form their team lineup. After saving the lineup, users can click the "Playing Field" button to proceed to the simulation.

In the backend of the simulation, Batter's and Pitcher's stats are compared against one another to arrive at the probability of each outcome. The logic first starts off with whether or not they put the ball in play, otherwise they walk or strikeout. If they put it in play it will then see if it is a hit and how many bases their hit would award them with. It cycles throughout each lineup until the user wishes to end the simulation.

The simulation uses the player's stats as inputs for the parameters. The simulation was built using WebGL and JavaScript. It also used special libraries such as Three.js and Cannon.js to create the 3D scene. Three.js was used to handle rendering (visual representation) and Cannon.js was used to handle the physics (movement, collisions). The user can move around the scene by using their scroll wheel or pinch and expand their keypad to zoom in and out .They also have the option to click and drag to move around the scene. The user may modify the parameters such as player stats using sliders provided on the right side of the screen in order to create their own unique game scenario. Once all the parameters are filled, the simulation combines everything and displays key moments from the simulated game. For example, the outcome of the simulated data will be displayed above the players at the middle of the field.

Chapter 3

Implementation

The project started with obtaining data from the Baseball Reference website and storing it. The programming language Python, a Python package named Beautiful Soup, and another Python package named MySQL Connector were used to scrape the required data for the sabermetric calculations and upload it to the database. MariaDB was used for the database management system; all of the players' data was stored on tables created here. A DBMS was a necessity because of the large amount of data needed for the website, web scraping for every user's request is less reliable, takes longer, and may create issues with the Baseball Reference website.

The web scraper takes a requested team's roster and goes to BaseballReference.com to pull the full list of their players who have made an appearance in the current baseball season. It pulls each player's unique ID and then goes to their individual stat page to populate their stats tables with their career totals. Our web scraper uses a three second pause between each pull to not get rejected by the website by flooding them with requests and continues until the end of their roster. Players are separated as batters and pitchers to account for the differences in their stats. Certain edge cases such as players with the same name are worked around with their unique IDs as they have a number associated with them depending on how many previous players have had the same name. Another edge case considering players Babe Ruth and Shohei Ohtani is that they qualify as both pitchers and batters. To simplify this whenever their IDs are read, they are just treated as a Batter to keep their hitting stats. The web scraper can go through multiple rosters depending what is inputted and inserts or updates each player on the given rosters.

Once player data is received and stored, a preview of this data can be seen on the website. Many of the web pages were created using standard website languages like PHP, HTML, and Javascript. The register, login, and create team page all use data cleaning methods to prevent SQL injections or otherwise invalid form data.

The team management page is designed to help users build their team by displaying player information and sending the selected players' stats to the simulation page. The page begins by presenting two lists: one showing all available players and another for the user's team, which is limited to one pitcher and five batters. Users can view each player's position and stats in the first list and select players to add to their team. When a player is selected, their stats are saved, and the user's team list is dynamically updated. Once the user completes their team and confirms their selections, the players' stats are sent to the simulation page.

This interactive page is powered by AJAX to ensure smooth and dynamic updates. HTML structures the elements visible to the user, while JavaScript handles the dynamic behavior by creating and manipulating HTML elements. For example, the player list is generated and populated by JavaScript functions.

To maintain security, JavaScript does not directly access the database. Instead, it uses PHP as an intermediary. When data is required, JavaScript sends an HTTP request to a PHP script using XMLHttpRequest. The PHP script processes this request, leveraging additional PHP functions to execute SQL queries and retrieve batter and pitcher data from the database. The retrieved data is then returned as a response, which JavaScript uses to populate the player list dynamically.

In addition to managing the player list, JavaScript also updates the user's team list as players are added or removed. Once the user finalizes their team, JavaScript and PHP work together to save the selected data, ensuring it is ready for the simulation page.

One of the main functions of the MySQL database was to facilitate player data, user info, and relational data between players and users.

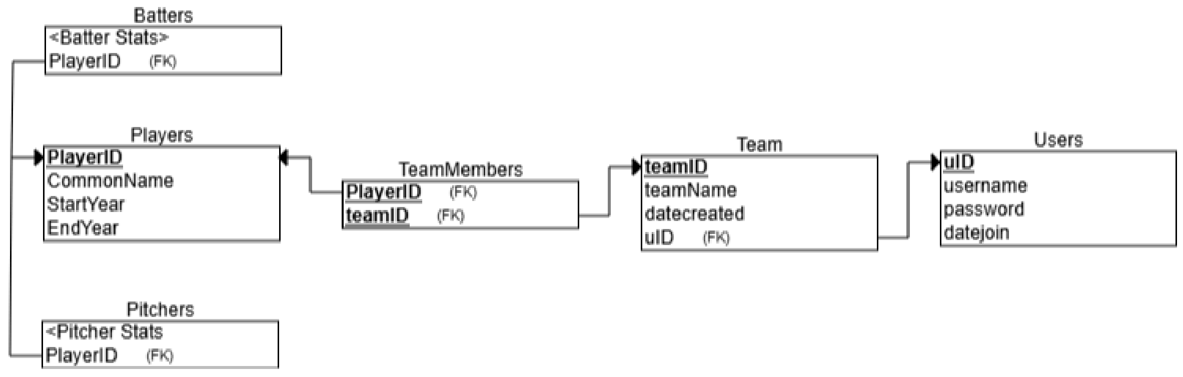
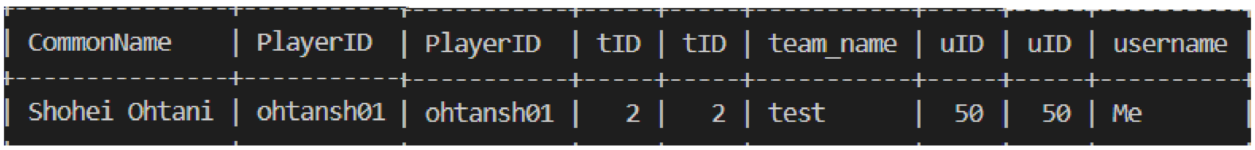


Figure 2: Entity Relation Diagram of the MySQL database

All baseball players are included in the Players table with additional data points shared amongst all players such as career start/end year and common name as displayed on baseball reference. Once the stats of the individual player is pulled, it is inserted into the correct table whether the player is a batter or a pitcher. The split tables were necessary due to both batters and pitchers having unique stats.

On the other side of the diagram, Users are able to create an account by entering a username and password. Once their account is registered, they are able to create a team. The team is attributed to the user by their hidden user id, 'uID'. When the user adds players to their team, they are added to a separate TeamMembers table. Players are attributed to the teams by the teamId. For example, the player Shohei Ohtani is added to a team named "test". The

TeamMembers table will add a row including Shohei Ohtani's playerID, "ohtansh01", and the teamID "2".



```
+---------------+-----------+-----------+-----+-----+-----------+-----+-----+----------
| CommonName    | PlayerID  | PlayerID  | tID | tID | team_name | uID | uID | username |
+---------------+-----------+-----------+-----+-----+-----------+-----+-----+----------
| Shohei Ohtani | ohtansh01 | ohtansh01 |   2 |   2 | test      |  50 |  50 | Me       |
```

Figure 3. Example of the Player/Team/User relationship in the MySQL database

The simulation is a JavaScript-driven baseball project designed to show the capabilities of interactive 3D rendering and physics simulation on the web to make baseball data easier to understand with visualization. It utilizes Three.js, a popular library for rendering 3D graphics, and CANNON.js, a physics engine, to create an immersive experience. Together, these libraries allow the simulation to realistically display baseball players, animations, and in-game physics such as player movement and collisions.

Here is a breakdown of some key components. The simulation employs model functions to load and manage 3D representations of players, including batters and pitchers. These functions assign different attributes to the models depending on whether they were a pitcher or batter, such as names, position keys, animations, and physics bodies, ensuring each player behaves as intended in the simulation. To load 3D assets, the simulation uses the GLTFLoader, a Three.js utility that reads .gltf files (a type of format for 3D models). These files are loaded from predefined paths and enhanced with animation mixers, enabling transitions between actions like stepping up to the plate when rendered, or pitching the ball.

The parameters for batters and pitchers are pulled from an array named Team_one. This array is dynamically populated using data provided by the user when creating their roster. The data is pulled from a PHP script that retrieves player statistics stored in a database and sends it to the JavaScript file in JSON format as previously explained. This integration ensures that the simulation can dynamically adjust to different team configurations based on user input.

When a roster is created, the data from the PHP script is echoed into JavaScript, allowing the simulation to incorporate predefined player attributes such as batting averages, on-base percentages, and slugging percentages. These attributes influence the outcomes of gameplay, making the simulation customizable and personalized.

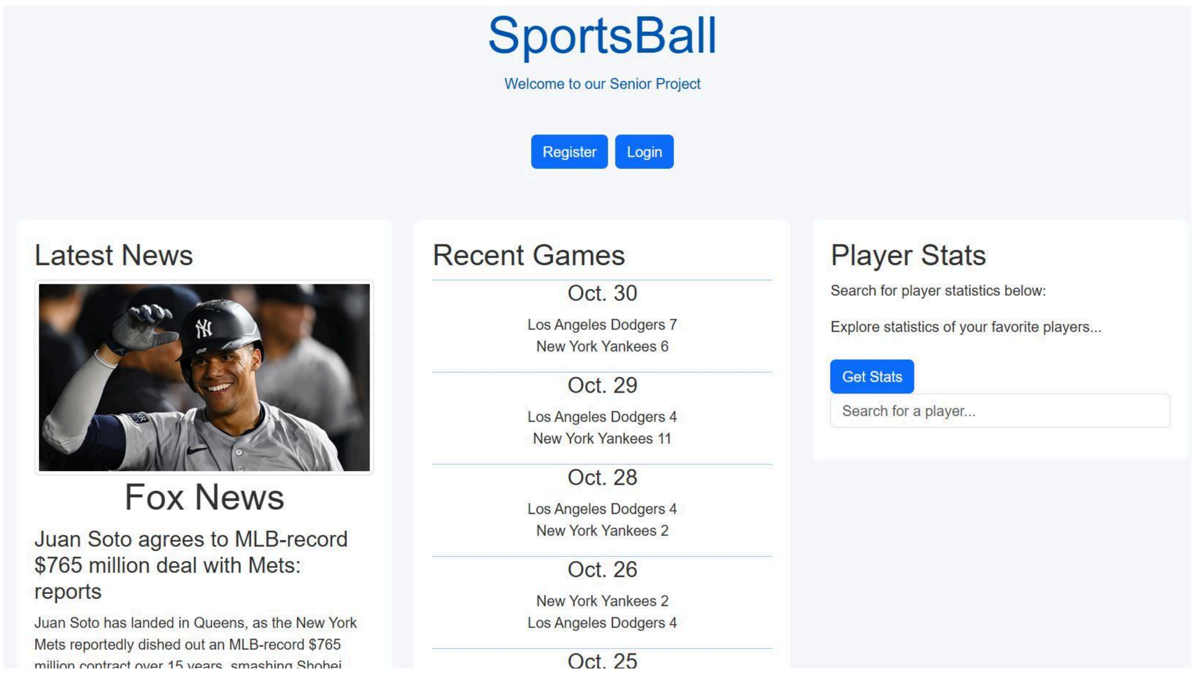
The simulation used an interactive graphical user interface (GUI) implemented with dat.GUI, a lightweight library for creating sliders, buttons, and other controls. These sliders are linked to a debugObject, a structure that stores batter and pitcher attributes like batting average, strikeout percentage, and walk percentage. When the "Create Batter" button is clicked, a function called fetches the stats of the next player from the Team_one array and updates the sliders. If the roster is incomplete or empty, the simulation falls back on predefined data. Users can manipulate the sliders to adjust player attributes dynamically, providing real-time control over gameplay variables. This feature allows users to experiment with different player configurations and see the immediate effects on gameplay.

Finally we combine batter and pitcher statistics from the sliders to simulate and display results such as Strikeout, Walk, or Hits based on the “at-bat logic” and at any point users can adjust stats, directly impacting the simulation outcomes.

Chapter 4

Results

After the implementation phase, we successfully integrated all the components of Sportsball into a cohesive and functional web application. The platform now allows users to visualize baseball statistics and make educated predictions based on simplified batter and pitcher data. Below are screenshots displaying key elements of our web application:



Juan Soto agrees to MLB-record $765 million deal with Mets: reports

Juan Soto has landed in Queens, as the New York Mets reportedly dished out an MLB-record $765 million contract over 15 years, smashing Shohei

Oct. 30
Los Angeles Dodgers 7
New York Yankees 6

Oct. 29
Los Angeles Dodgers 4
New York Yankees 11

Oct. 28
Los Angeles Dodgers 4
New York Yankees 2

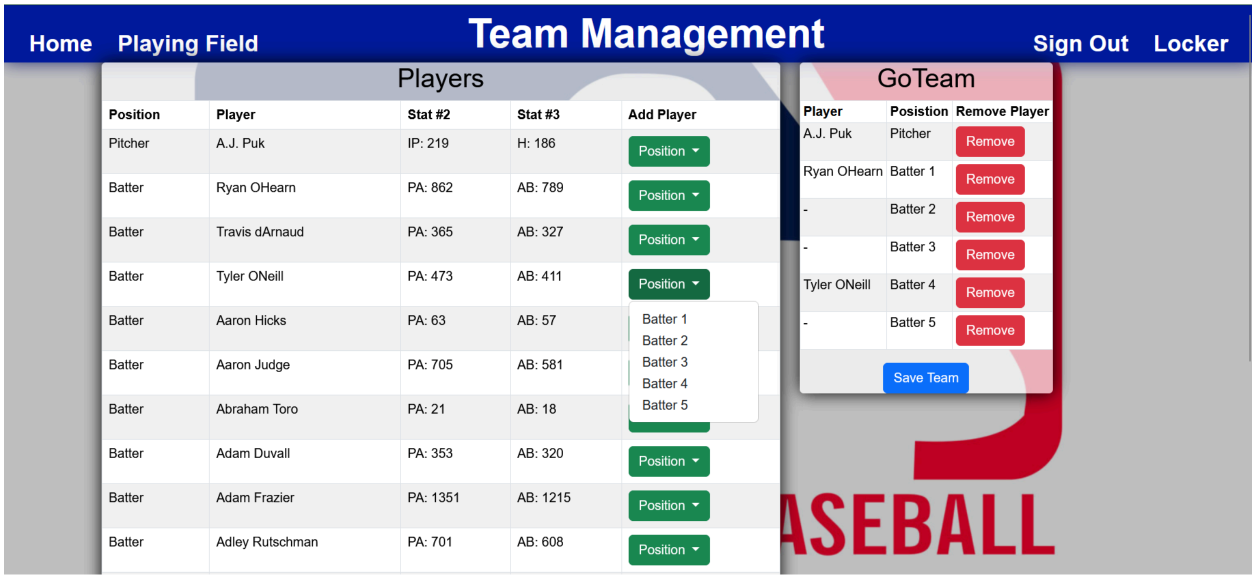
Oct. 26
New York Yankees 2
Los Angeles Dodgers 4

Oct. 25

Search for player statistics below:

Explore statistics of your favorite players...

Figure 4. Project Homepage



Players

Position	Player	Stat #2	Stat #3	Add Player
Pitcher	A.J. Puk	IP: 219	H: 186	Position
Batter	Ryan OHearn	PA: 862	AB: 789	Position
Batter	Travis dArnaud	PA: 365	AB: 327	Position
Batter	Tyler ONeill	PA: 473	AB: 411	Position
Batter	Aaron Hicks	PA: 63	AB: 57	
Batter	Aaron Judge	PA: 705	AB: 581	
Batter	Abraham Toro	PA: 21	AB: 18	
Batter	Adam Duvall	PA: 353	AB: 320	Position
Batter	Adam Frazier	PA: 1351	AB: 1215	Position
Batter	Adley Rutschman	PA: 701	AB: 608	Position

GoTeam

Player	Posistion	Remove Player
A.J. Puk	Pitcher	Remove
Ryan OHearn	Batter 1	Remove
-	Batter 2	Remove
-	Batter 3	Remove
Tyler ONeill	Batter 4	Remove
-	Batter 5	Remove

Figure 5. Team Management Page

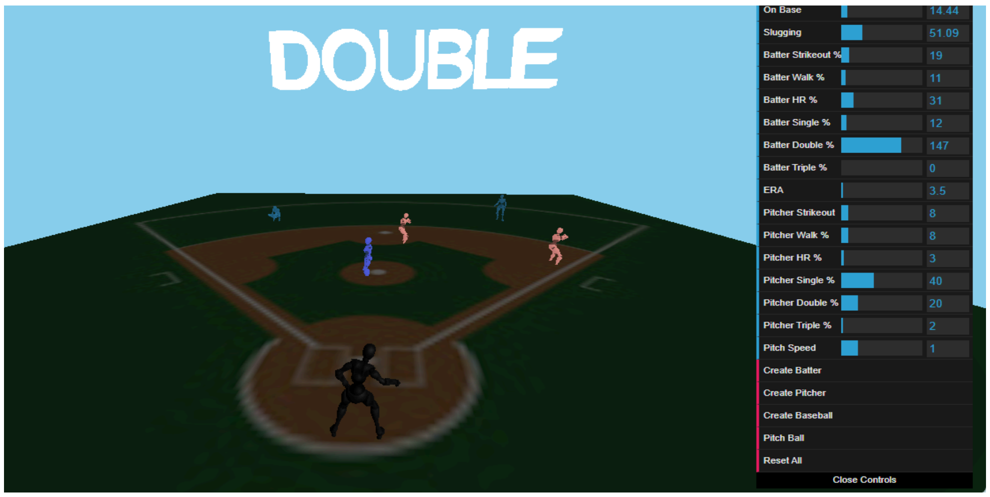
The Team Management Page provides users with an intuitive interface for assembling and managing their team. The page is divided into two distinct sections: a Players List on the left and a Team List on the right.

The Players List serves as the pool of available players, showcasing essential information to aid decision-making. Each player entry includes their position and key stats that users can evaluate when selecting team members. To streamline the selection process, a dropdown menu is available for each player, allowing users to assign them to specific positions on their team.

The Team List on the right displays the players currently added to the user's team. Each player in this list is accompanied by a convenient remove button, enabling users to make quick adjustments to their lineup. The page leverages real-time updates powered by AJAX, ensuring that any changes—whether adding or removing players—are immediately reflected without requiring a page refresh.

Once users are satisfied with their selections, they can finalize their team by clicking the Save Team button. This action securely saves the team name along with the full roster of selected players into the database, preserving all changes for future use.

This streamlined yet powerful interface ensures an efficient and user-friendly experience, enabling users to manage their teams with ease and confidence.



On Base	14.44
Slugging	51.09
Batter Strikeout %	19
Batter Walk %	11
Batter HR %	31
Batter Single %	12
Batter Double %	147
Batter Triple %	0
ERA	3.5
Pitcher Strikeout	8
Pitcher Walk %	8
Pitcher HR %	3
Pitcher Single %	40
Pitcher Double %	20
Pitcher Triple %	2
Pitch Speed	1

Figure 6. WebGL/JavaScript simulation, play resulted in a double with statistics on right

On the server side, we implemented a robust system to retrieve and manage baseball data. By web scraping information from Baseball-Reference.com, we populated a MariaDB database with extensive player statistics, both current and historical. This database enables users to select players for their team roster and analyze data relevant to specific matchups. Queries to the database are processed efficiently, ensuring a smooth user experience when interacting with large datasets. The server also integrates “at-bat logic” calculations, which take game-state factors like outs and players on base into account, to produce real-time probability predictions. Using JavaScript on the backend, we implemented calculations to process data dynamically. Data retrieved from the database is formatted automatically into structures suitable for analysis and visualization. The probability functions utilize user-selected factors to calculate win probabilities, which are then rendered in real-time on the client side. This ensures that users can make informed predictions based on their chosen parameters without having to navigate complex calculations themselves.

In addition, MariaDB's relational database model allowed us to efficiently store and filter the scraped data. This structure supports user queries and ensures that only the most relevant information is delivered to the front end. The database seamlessly integrates with our web application, enabling interactive features like roster selection and scenario-based probability analysis. Overall, the results of the Sportsball project exceeded our expectations. We created a functional and engaging platform that simplifies the process of analyzing baseball statistics while maintaining statistical accuracy. Each team member played a vital role, contributing to the design, development, and implementation of key features. The final product combines elements of web scraping, database management, JavaScript programming, and data visualization into a streamlined application. We believe that Sportsball offers a unique way for fans to connect with the sport, enhancing their understanding and enjoyment of baseball while fostering a deeper appreciation for its strategic complexities.

Chapter 5

Conclusion

After a year of working on Sportsball, we have gained valuable experience in creating web applications that combine complex data management with interactive user experiences. Each team member contributed to critical parts of the project, such as data scraping, probability calculations, and user interface design, ensuring all components worked seamlessly together. Maintaining strong integration between these modules was crucial to delivering a smooth and engaging experience for users.

This project offers a user-friendly platform that makes exploring baseball statistics both engaging and accessible to fans of all backgrounds. By simplifying the process of analyzing key player statistics and providing predictions based on carefully calculated probabilities using the “at bat logic”, we make it possible for fans to experience the game in a more interactive way. Whether users are casual observers or long-time enthusiasts, this platform offers tools that allow them to dive deeper into the game without feeling overwhelmed by its complex numerical aspects. The goal is to make baseball statistics approachable and enjoyable, providing a more immersive way to connect with the sport.

Additionally, by encouraging active participation and offering visual tools that simplify complex data, we hope to foster a greater appreciation for baseball’s rich statistical history. As fans gain a deeper understanding of the game through user-friendly insights and predictions, they are inspired to explore the intricacies of baseball further. This project not only aims to enhance the fan experience but also serves as a bridge between casual enjoyment and the more advanced statistical analysis that characterizes modern baseball. Through this, we believe fans can develop a stronger connection to the game and its ongoing narratives.

References

- Baseball Reference. (n.d.). *Baseball statistics and history*. (2024). Retrieved from https://www.baseball-reference.com/