# CS 411: Final Project Report

TEAM 082

**1. Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**

Initially, our project aimed to combine historical performance, real-time updates, and social media sentiment into a single sports prediction tool. Over the semester, we pivoted away from real-time updates and sentiment analysis due to technical limitations with API reliability and time constraints. Instead, we focused our efforts on a backend for historical NBA data (covering the 2024-2025 regular season) analysis and advanced SQL functionalities, successfully connecting it to an interactive frontend.

**2.Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

Our application achieved its goal of being a useful tool for NBA fans and analysts to explore historical game and player data. Users can now:

- View team and player statistics.
- Compare performances.
- Add favorite teams/players to the user profile

While we didn't fully integrate live updates or sentiment analysis, the application still provides meaningful insights via interactive data visualizations and filtering tools.

**3.Discuss if you changed the schema or source of the data for your application**

We originally intended to rely heavily on Basketball Reference and The Sports DB APIs for data collection. Ultimately, we primarily used CSV datasets from Basketball Reference due to API call limits and data formatting inconsistencies with The Sports DB. We designed the schema to reflect these changes, emphasizing structured game, player, and team statistics with some predictive features manually calculated from static datasets. Furthermore, we had to do some data cleaning from this original CSV data as some of the varying datasets/CSV provided had some differences (Ex: Players dataset used “BOS” while Teams dataset used “CEL”) that we had to change for consistency.

**4.Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

Several changes were made from the initial ER model:

- User Table & Favorites: We added a user and user_favorite table to allow CRUD operations for user preferences, which weren’t in the original design.
- Normalization: Originally, we considered one large PLAYER table, but we broke up repeated values (e.g., team names) into foreign keys to the teams table for better normalization.
- GAME_STATS Table: Evolved significantly to include ranking metrics and composite performance values, enabling more sophisticated SQL queries and indexing.

- A more suitable design would split out Player Season Stats and Player Game Stats into separate tables, which we didn't fully implement but would help with future scalability.

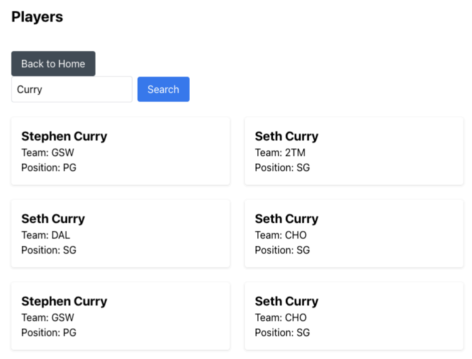
**5.Discuss what functionalities you added or removed. Why?**

Removed:

- Real-time updates
- Social media sentiment overlay (originally planned using Reddit + PRAW)

Added:

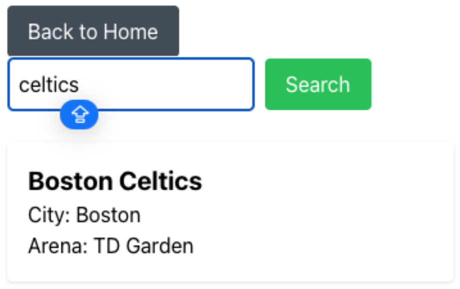
- Keyword search functionality: keyword search bars for Players and Teams. With these features, users can input a search term (partial match) and retrieve results. The frontend utilizes `fetch()` to send the keyword to the backend, which performs SQL queries using `LIKE` for flexible matching.


**Players**

Back to Home

Curry | Search

| | |
|---|---|
| **Stephen Curry**<br>Team: GSW<br>Position: PG | **Seth Curry**<br>Team: 2TM<br>Position: SG |
| **Seth Curry**<br>Team: DAL<br>Position: SG | **Seth Curry**<br>Team: CHO<br>Position: SG |
| **Stephen Curry**<br>Team: GSW<br>Position: PG | **Seth Curry**<br>Team: CHO<br>Position: SG |


Back to Home

celtics | Search

**Boston Celtics**
City: Boston
Arena: TD Garden

- A complete backend schema with foreign key constraints
- Index optimization on key performance metrics
- Favorite teams/players feature for users
- Advanced SQL queries

These choices prioritized a reliable MVP with analytics depth over external API dependencies.

**6.Explain how you think your advanced database programs complement your application.**

Our advanced SQL queries brought real value to the user experience:

- Top Performers: Identifying top teams and players based on statistical benchmarks.
- League-wide Comparisons: Queries that compare individual performance to league averages using nested subqueries.
- Filtering and Aggregation: Efficient data grouping and filtering using advanced JOINs and WHERE/HAVING clauses.

**7.Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.**

Ben Hesse (Backend & DB Integration): Translating messy CSVs into clean relational tables was challenging due to inconsistent schemas. Advice: automate preprocessing scripts using Pandas before SQL ingestion to maintain clean, validated rows.

Tina Ding: (DB Design): it was tricky to maintain referential integrity between user_favorite table and both users and players/teams because favorites could reference 2 different types of entities (players or teams) with various ID formats. Tip: design db early with clear foreign key relationships or enforce entity type validation manually.

Luca Listi (Data Collection & Analysis): Dealing with missing values and outliers in datasets—especially in scraped data—caused skewed results in predictive stats. Tip: use data profiling tools (like pandas-profiling or SQL aggregate checks) early on.

Chris Xie: One technical challenge was handling route handles and middleware. Initially, we did not have the correct ordering for the routes relating to the authorization, returning 404 errors or not reaching the correct routes. One piece of advice would be to track all the routes and register all of them and the middleware before calling listen() in order to properly maintain connection.

## 8.Are there other things that changed comparing the final application with the original proposal?

- Dropped real-time game updates and social sentiment overlays.
- Enhanced schema modularity to improve data consistency.
- Shifted predictive model implementation to a more basic statistical method due to time limits (e.g., use historical trends rather than ML).

## 9.Describe future work that you think, other than the interface, that the application can improve on

Areas for future improvement:

- Add real-time data pipelines (e.g., streaming game scores).
- Build out predictive ML models using historical datasets (integrate scikit-learn or TensorFlow).
- Reintroduce sentiment analysis via Reddit or Twitter APIs and merge it with game data insights.
- Add per-game player stats to support deeper match-level analytics.

## 10.Describe the final division of labor and how well you managed teamwork.

Ben Hesse: Database schema design, backend API development, SQL query writing.

Tina Ding:  Database advanced programs design and implementation, frontend design and development, crud operations implementation.

Luca Listi: Data collection, preprocessing, analysis, and evaluation of indexing strategies.

Chris Xie: Authorization routes, add and remove favorite team/players, advanced sql query writing

Despite having team switch-ups early on, we adapted by clearly dividing responsibilities and trying to meet regularly to integrate our work. As a team, we could have communicated better; however, we met project deadlines and supported each other during crunch periods before large deadlines.