Date of publication xxxx xx, 2025, Date of current version Feb 13, 2025.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

MTSU Basketball Data Tracker

CLIFFORD N JONES[1], JAELIN MCKETHAN[2], JONAH PITCHER[3], JUSTIN EUGENE[4] (Member, IEEE)

[1]Computer Science Department, Middle Tennessee State University, Murfreesboro, TN 37129 USA

Corresponding author: Dr. Khem Poudel (e-mail: Khem.Poudel@mtsu.edu).

ABSTRACT This work develops and deploys a relational database structure to track basketball team athletes' performance and health information. A relational design between players, their game performance and injuries, at its core, will track game involvement and injury occurrences through associative relationships. The system provides three main functions: automatic data consistency through foreign key constraints, query optimization tools for statistical data collection, and future framework capabilities that support advanced analytical features. This system enables performance tracking that provides long-term analytical abilities that help coaches make better decisions, recruit players, and develop players.

INDEX TERMS DBMS,

I. INTRODUCTION

MODERN advanced sports analytics solutions positively impact both professional basketball organizations and college basketball teams by monitoring player performance data. Sports are a widely studied topic, and as we can see from the research, data visualization benefits coaches' abilities to solve problems [2]. The combination of wearable solutions, database structures, and statistical software applications provides practical data to coaches for injury prevention, player development, and tactical strategy formation. An optimized database architecture supports reliable data storage, maintains consistency, and offers strong query functionalities to generate real-time dashboards and historical analysis.

Through this structured framework, sports scientists and coaching staff gain user-friendly applications to make informed decisions about player training programs and recovery protocols. Analyzing injury data can help with injury prevention [1]. Sports-related applications depend on the robust functionality of relational databases, enabling the tracking of player movements and game activities to improve performance analysis and strategic planning. A well-designed system that integrates data from multiple teams facilitates the evaluation of competitor performance and overall operational efficiency. The initiative focuses on developing a comprehensive database system to support the activities of the Middle Tennessee State University (MTSU) basketball team, incorporating relational modeling principles, key constraints, and multiple relationship structures to create a complete player metrics solution.

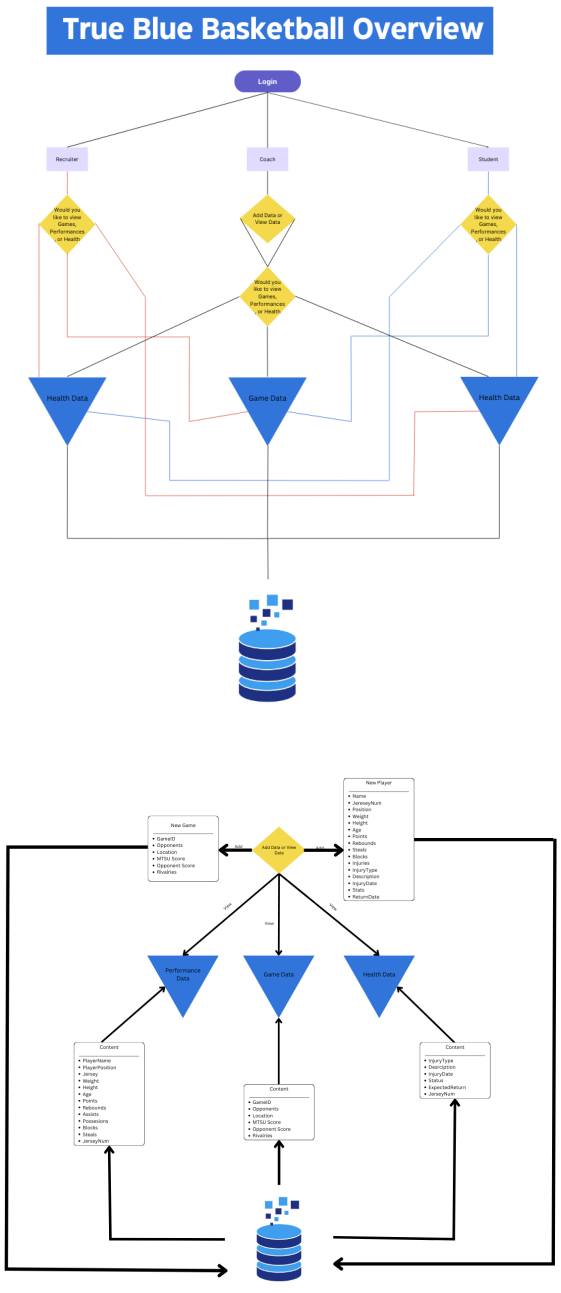
II. PROJECT OVERVIEW

In this project, user functionality is based on the type of role their account is. Coaches have the most abilities and functionality with the website as they are able to not only view the datasets, but add additional player, game, or injury-based data to the database for all accounts to see. The other accounts, such as student and recruiter accounts, are more focused on viewing the data and personalizing which players they want to keep track of and what kind of games they are interested in. The users can access whatever datasets they want and look up a specific player by their name or jersey to display all the relevant information they wish to see related to that player.

A. OBJECTIVES

True Blue Basketball addresses crucial needs of collegiate basketball teams by creating a full analytics solution built with mainstream web tools and specifically designed for college basketball teams. Key features include:

- The system integrates a vigorous multi-functional Database Management System (DBMS) which handles broad datasets that combine game statistics alongside player performance metrics with in-depth player health documentation.
- The platform incorporates separate user roles that permit coaches to write data but students and recruiters only read data and personalize their views.
- Users experience better interface usability than spreadsheets because the system integrates search capabilities along with pagination and favorite functions to improve user experience.



III. DATABASE SCHEMA CREATION

The backend system supports the interface operations using relational databases which have been established with structured SQL schemas. The database schema incorporates all major entities consisting of players, games, injuries, user accounts, favorites and participation records. The systematic approach creates normalized data relationships with the result of efficient querying as well as referential integrity throughout the system.

The **players table** includes name, position, jersey number, height, weight, age, and academic classification as fields which represent vital identity and physical characteristics of each athlete. Records regarding opponents and dates together with locations and scores are stored in the games table. The database requires these tables to conduct performance tracking alongside game analytics.) The **injuries table** tracks player-specific injuries of different types that displays their present status and their estimated return date to provide accurate coordination between medical staff and coaches.

The system depends on three essential supporting tables that deliver proper functionality to additional features. Authentication management via secured password hashes enables **user** management together with email uniqueness and viewer/coach role permissions located within the users table. Users establish their preferred line-up in the favorites table and this linkage enables custom dashboard functionality but the **player_game table** performs many-to-many recordkeeping during games to track playing time and starting status.

The database implementation includes full declaration for all its foreign key constraints. Within the **favorites table** a foreign key enables relationships between users and players while the **player_game table** connects both players with games. The database references enable both data integrity through consistency and the execution of JOIN queries which enhance the interactive User Interface features.

The implemented MySQL syntax relies on schema design principles that enhance scalability and extending capabilities. All Tables have been created with automatically incrementing primary keys when needed and contain **DATETIME** fields for monitoring timestamp creation times to enable chronological filtering and analyze data trends. The schema design acts as one cohesive structure which supports the development of an extensive front-end system.

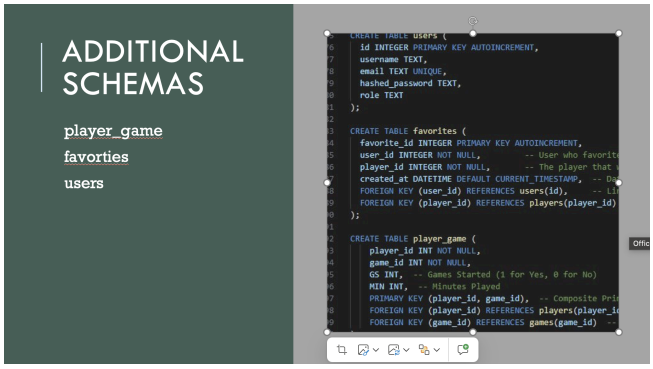


FIGURE 1: SQL schema snippet illustrating the auxiliary tables that support the True Blue Basketball DBMS: `users` for secure authentication and role management, `favorites` linking viewers to preferred players, and the many-to-many `player_game` table that logs individual participation metrics. Together with foreign-key constraints, these schemas enforce referential integrity and enable efficient, normalized queries that power the front-end experience.

IV. METHODOLOGY & SYSTEM ARCHITECTURE

The project is built around a classic three-tier architecture, integrating frontend, application logic, and database storage.The traditional three-tier approach sets the project structure by combining frontend elements with application logic along with database storage capabilities.

Presentation Layer (React): The user interface powered by React runs through localhost:3000 as part of the logical Presentation layer. The main application views consist of five key components named Home.js, Athletes.js, Games.js, Injuries.js, Favorites.js together with two reusable UI widgets SearchBar.js and AthleteCard.js that improve user experience and design consistency throughout the application.

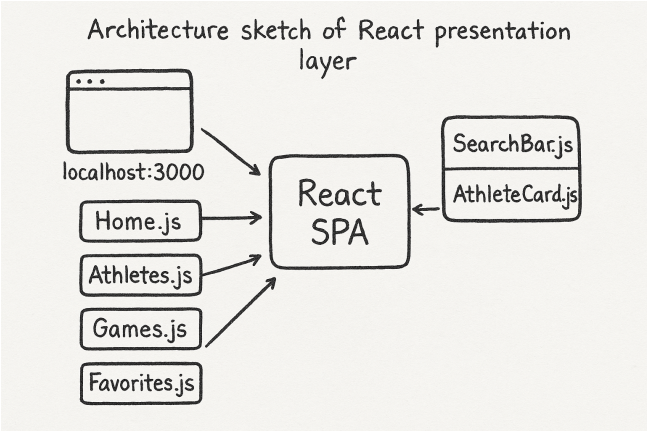


FIGURE 2: Architecture sketch of the React presentation layer: a single-page application (SPA) served at `localhost:3000` that routes to the main views (`Home.js`, `Athletes.js`, `Games.js`, `Injuries.js`, `Favorites.js`) and reuses shared widgets (`SearchBar.js`, `AthleteCard.js`).

Application Layer (FastAPI): The Application Layer utilizes FastAPI to deliver RESTful APIs that contain JWT-based secure authentication and validate all input data using Pydantic models while also handling CORS properly. The application implements role-based protection for different user types that separates coaches from general users.

Data Layer (SQLite): The application utilizes SQLite through the Data Layer to manage its comprehensive data records in structured tables including player, games injuries player_game, favorites and users. Foreign-key constraints create a mechanism to preserve data integrity within the system.

The basic request-response cycle operates through which React clients verify their authentication using JWT tokens to access specialization endpoints that deliver structured JSON data from FastAPI API endpoints.

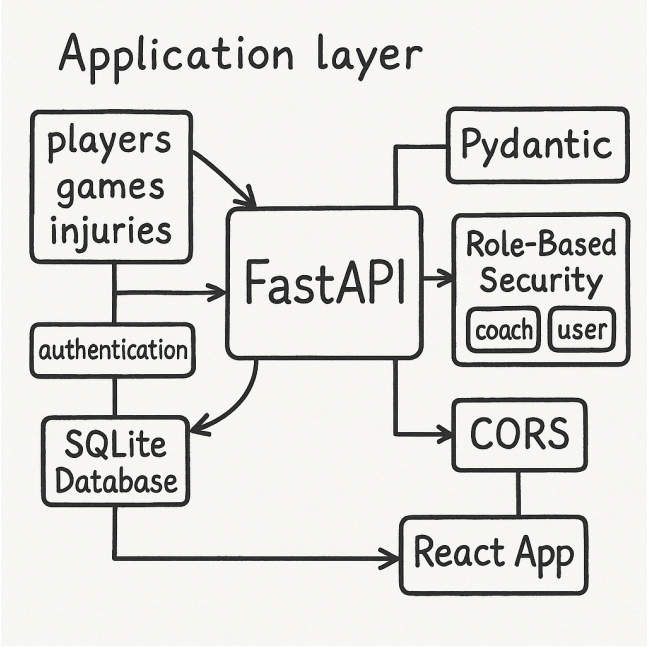


FIGURE 3: Architecture sketch of the FastAPI application layer: RESTful endpoints for players, games, and injuries are served through FastAPI, secured with JWT-based role-based access control, validated via Pydantic models, and exposed to the React client through CORS while persisting data in a SQLite database.

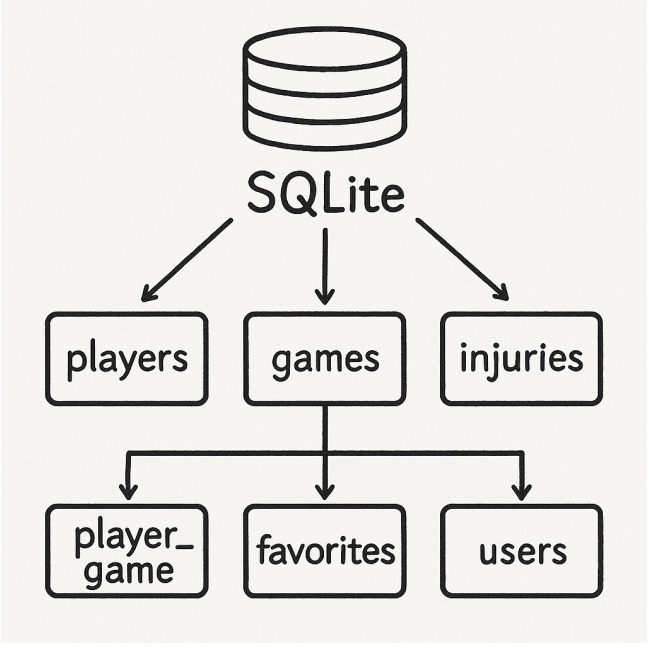


FIGURE 4: Architecture sketch of the SQLite data layer: a lightweight relational database storing `players`, `games`, and `injuries` tables, with associative tables `player_game`, `favorites`, and `users`; foreign-key constraints enforce data integrity across relationships.

DATABASE STRUCTURE

The database employs strategic normalization and relational schema design:

- All player information including personal data and in-depth statistical records resides in the Players Table.
- Games Table contains exhaustive data for each contest by keeping tabs on game dates alongside opponents and locations and score results.
- The database includes an Injuries Table which manages recorded injury data with details about player health conditions and the treatment methods and predicted medical recuperation durations.
- Player_Game Bridge Table: Facilitates a many-to-many relationship to accurately represent player participation in individual games.
- The Favorites Table enables users to manage their custom preferences for favorite players by maintaining distinct associations between users and the players they support.

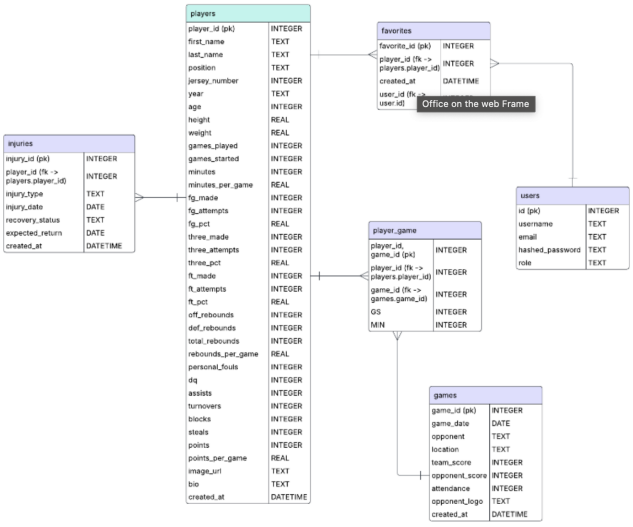


FIGURE 5: Entity–relationship diagram (ERD) for the SQLite data layer, illustrating core tables (`players`, `games`, `injuries`) and associative tables (`player_game`, `favorites`, `users`) linked by foreign-key constraints.

V. BACKEND SYSTEM IMPLEMENTATION: A COMPREHENSIVE TECHNICAL REPORT

The basketball team management application requires a backend system which offers solid and protected API capabilities to process user interactions while managing player data as well as game programs and player injuries. This system is designed around FastAPI as its main backend solution and uses Express.js for supporting endpoints through modern web technologies to deliver high security and simple maintenance features. The following guidelines provides complete

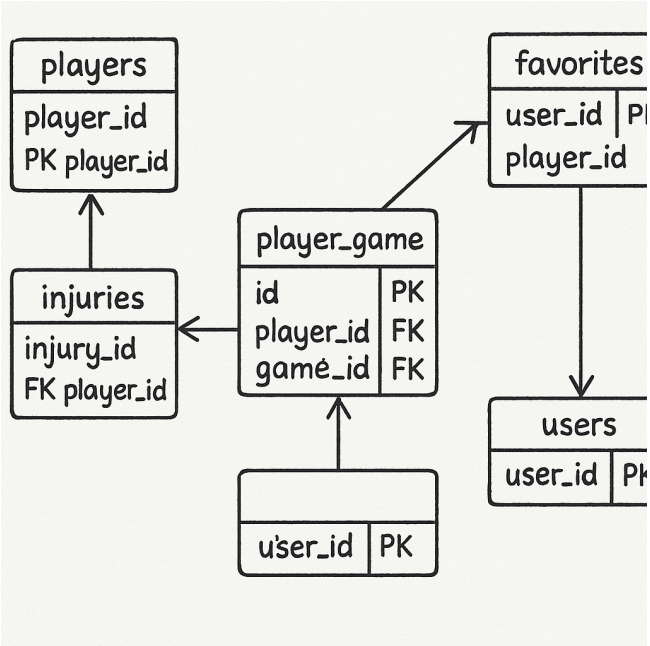


FIGURE 6: Hand-drawn entity–relationship diagram showing primary keys (PK) and foreign keys (FK) across the `players`, `games`, `injuries`, `player_game`, `favorites`, and `users` tables.

information about backend structure alongside explanations of its essential components and deployment details.

The FastAPI framework builds the main backend alongside two primary components:**FastAPI (Primary Backend):** This Python-based framework manages authentication systems as well as data management operations and implements role-based access control features. Data validation through Pydantic ensures dependable API interactions when working with the lightweight SQLite data storage. The **Express.js** server uses Node.js as its foundation to display athlete data through MySQL connections (Secondary Backend Server). Additional databases or services can be integrated into the system framework for improved flexibility but do not constitute its main purpose. Users can interact with the designed system through its RESTful APIs that enable data fetching while also permitting creation and update and deletion operations. This application implements essential features for managing players and games and tracking injuries and provides user login functions and controls permission levels for different users.Authentication and Security.

The backend implementation establishes security as its fundamental principle. Each request passes through authentication without server-side session assistance because the system uses JSON Web Tokens (JWT) for stateless authentication. Here's how it works:

When users register or login both endpoints support credential verification and bcrypt password hashing through a cost factor set at 12 followed by JWT token distribution. The system delivers role information through tokens which sup-

ports role-based access control (RBAC) for protected functionalities. The authentication system provides JWT tokens which receive their signature through SECRET_KEY and specify an expiration duration of 1 hour by default. Through the get_current_user dependency the system evaluates and authenticates access tokens for protected route endpoints which deny unauthorized requests.

```python
@app.delete("/delete-player/{player_id}")
def delete_player(..., current_user: UserInDB = Depends(get_current_user)):
    if current_user.role != "coach":
        raise HTTPException(403, "Only coaches can delete players.")
```

FIGURE 7: Code snippet showing role-based access control for player deletion in the FastAPI backend. The endpoint restricts player deletion to users with 'coach' role, raising a 403 Forbidden error for unauthorized attempts.

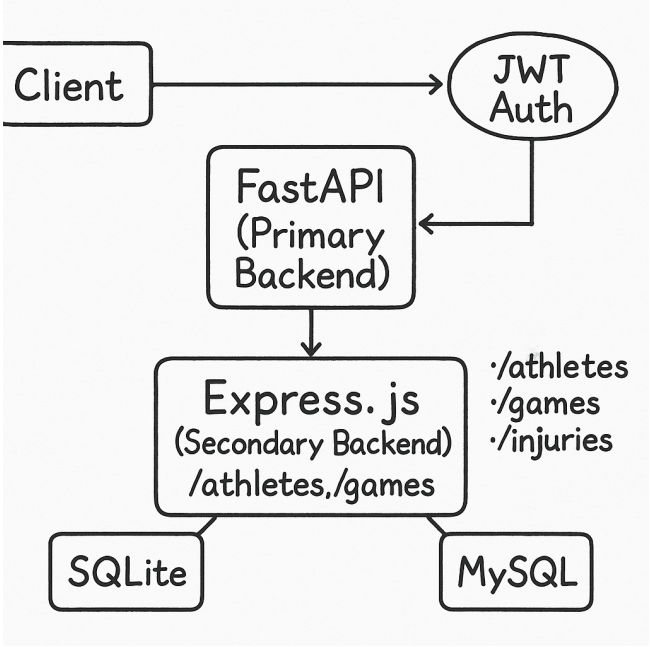


FIGURE 8: Overall backend architecture: client requests are authenticated via JWT, then handled by the primary FastAPI service (Python) which manages authentication, role-based access control, and SQLite persistence. Selected routes are proxied to a secondary Express.js service (Node.js) that interfaces with a MySQL database for athlete, game, and injury endpoints, providing extensibility for additional data sources.

VI. FRONT-END IMPLEMENTATION

True Blue Basketball Database Management System (DBMS) operates as a full-featured application to handle basketball team administration alongside player information management and scheduling and injury documentation for men's basketball team. This system aims to build an approachable user interface that connects through a reliable API while maintaining data efficiency and decision support capabilities for coaches and players and all involved stakeholders.

Modern web technologies, particularly React.js, were strategically used to create this application frontend which delivers responsive interactive and user-friendly experiences to the users. The project implements frontend design by modularly grouping major functions in separate components that enhance scalability as well as maintenance and future improvement possibilities.

Users first encounter an uncomplicated interface bar designed using React Router once they start their system session. The navigation component makes it simple for users to move between the Home page as well as Athletes, Injuries, Games, Favorites and Login pages of the application. The application provides consistent pseudo MTSU branding to all interfaces through a navigation system which enables smooth transitions among different application parts.

The **Home** page shows dynamic content by obtaining random player and game data from its API endpoints. Users engage right away with showcased content through the "spotlight" feature since it displays player stats alongside current game results prominently. The React useEffect hook enables instantaneous data fetching and display which delivers up-to-the-minute team highlight information to users.

Users within the **Athletes** section benefit from the management capabilities of the Athletes component which presents an extensive list of rosters. The Athlete data receives state management through React hook functions useState and useEffect that both retrieve and filter athletes while implementing pagination feature. Users can efficiently find specific players through the SearchBar component which provides an advanced search functionality. The system enables clear navigation across extensive data records through its paging controls. The system enables coaches to perform profile operations that include creation as well as removal of player profiles. The operations utilize secure API endpoints which need proper authorization tokens for the purpose of maintaining data security and integrity.

The **AthleteCard** component produces an elevated individual player representation through detailed statistical displays which include position information and jersey numbers and player class year along with average points and rebounds and assists and field goal percentage values. An interactive interface offers functionality to display bios and favorite status control and visual friend indicators using heart icons. Real-time updates of favorite statuses happen through backend communication so users experience a quick and customized viewing environment.

The **Games** section of the system presents users with

thorough details about basketball games that took place in the past and upcoming contests. The entry format for each game includes complete information about the opposing team together with game date and location and scores and team artwork which enhances visualization and consumer engagement. An interactive element with secure API functionality supports authorized coaching staff to make game entries and delete them from the system. Also the conditional rendering system displays win or loss outcomes.

Within the **Injuries** management component players can maintain their health records enable tracking of injuries together with data about types and dates as well as their recovery status and estimated return dates. The system provides essential functionality which enables better coaching decisions and healthcare management regarding player health. The application provides instant access to recent injuries because of the color-coded recovery status system which enhances both visual clarity and instant assessment capabilities. Coaches receive extended privileges for both creating new injury reports and handling their current injury records by utilizing API-based procedures.

The **Favorites** page collects customized user-demand through API endpoint retrieval to generate individual athlete listing pages. The system shows its ability to create personalized engagements with user data storage features which produces enhanced satisfaction levels among users.

Through a well-designed **JWT-based authentication system** users can achieve seamless access to the application during login or sign-up processes. The system uses secure user credential collection to obtain tokens which enable role authentication (viewer or coach) before effectively maintaining session storage for lasting authentication maintenance.

Aesthetic consistency along with pseudo MTSU branding standards guides the application design while that entails proper use of official colors, typography and logo placement. By using CSS styles to design the user interface the front-end looks attractive while good design improvements both readability and user interaction and system usability.

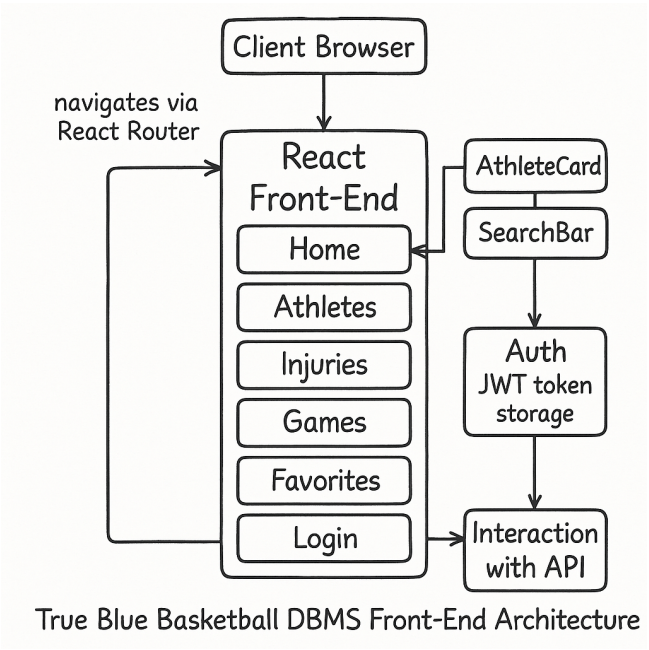


FIGURE 9: Front-end architecture of the True Blue Basketball DBMS. A client browser navigates the React single-page application (SPA) via React Router, accessing routed pages—`Home`, `Athletes`, `Injuries`, `Games`, `Favorites`, and `Login`. Reusable components (`AthleteCard`, `SearchBar`) enhance UI consistency, while JWT authentication is stored client-side and included in API calls for secure interactions with the back-end.

VII. SYSTEM DEMONSTRATION

Users can activate a walkthrough for the True Blue Basketball DBMS by starting from the user authentication interface. The application first displays a login interface with options to both sign in if already existing or create a new account for registration. The application changes permission settings according to the viewer or coach assignment of each user. The system allows coaches to modify database records but viewers experience restricted access to only reading information along with favorite organization abilities.

The user reaches the Home page directly after finishing their login procedure. Random highlights from the selected player along with recently played games appear prominently on this screen. The real-time spotlight shows regularly updated critical stats along with the most memorable match highlights. All athletes appear in the Athletes section of the application after navigation. Users benefit from the search tool to find specific players by their names before accessing detailed information which consists of bios and performance stats through a click action. Coaches benefit from an interface which lets them both create new player entries through a structured form and instantly delete any outdated or incorrect entries directly from the interface.

Users in the Games section can review a sequential list of matches that they can page through. The match card presents

team scores and locations together with interactive visual indicators that show win or loss results. Through the interface coaches gain access to record new games and manage scores together with the removal of previous events.

Through the Injuries module coaches obtain the ability to record physical setback progress by each team member. The injury record creation process allows users to choose players from available options then enter details after selecting the player. Returning dates serve as an optional feature during the process. Medical staff and training personnel can obtain instant recovery status information through the date-sorted records which are color coded for easy observation.

The favorite section in the app enables user customization by providing a way to save their preferred athletes for later review. A smooth user interface supports the button functions for favoriting and unfavoriting players alongside a simple interface design.

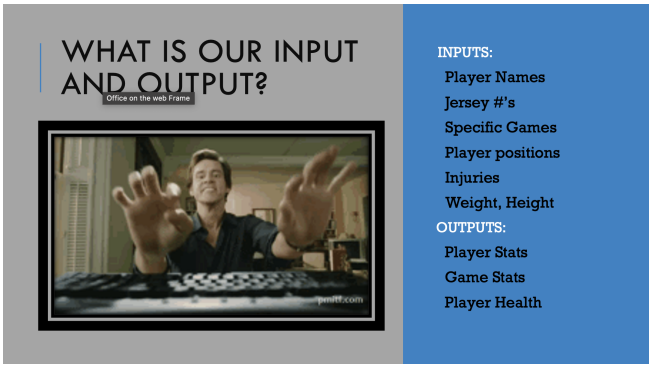


FIGURE 10: System demonstration of the True Blue Basketball DBMS. **(a)** Secure login/registration screen adjusts permissions based on role (coach vs. viewer). **(b)** Home page spotlight shows a random player and a recent game highlight. **(c)** Athletes page with search, pagination, and detailed player cards; coaches can add or delete records. **(d)** Games list displaying scores, locations, and win/loss indicators, with CRUD controls for coaches. **(e)** Injuries module for tracking player health, colour-coded by recovery status. **(f)** Favorites page where viewers curate and manage their preferred athletes.

VIII. CONCLUSION

The True Blue Basketball DBMS frontend implementation effectively combines React.js advanced features for secure API usage together with dynamic content display features and comprehensive authentication and authorization features. The complete implementation at pseudo MTSU basketball delivers stakeholders a dependable platform which delivers a successful experience and efficient operational performance with improved team management capabilities and player development systems.

REFERENCES

[1] S. Schneider, "Sports injuries: population based representative data on incidence, diagnosis, sequelae, and high risk groups * Commentary," British Journal of Sports Medicine, vol. 40, no. 4, pp. 334–339, Apr. 2006, doi: https://doi.org/10.1136/bjsm.2005.022889.

[2] C. Perin, R. Vuillemot, C. D. Stolper, J. T. Stasko, J. Wood, and S. Carpendale, "State of the Art of Sports Data Visualization," Computer Graphics Forum, vol. 37, no. 3, pp. 663–686, Jun. 2018, doi: https://doi.org/10.1111/cgf.13447.

[3] FastAPI Documentation. (2023). FastAPI. https://fastapi.tiangolo.com/.

[4] OWASP. (2021). Authentication Cheat Sheet. https://owasp.org/www-project-cheat-sheets/

[5] Provos, N., & Mazières, D. (1999). A Future-Adaptable Password Scheme. USENIX.

[6] OWASP. (2021). Authentication Cheat Sheet. https://owasp.org/www-project-cheat-sheets/

[7]