PLAYER MANAGEMENT & ROSTER OPTIMIZATION FOR BOSTON BRUINS

Index

Introduction

Hockeyhive.com - Revolutionizing Player Management for the Boston Bruins

This report articulates the essentiality of the player management and optimization systems in sports leagues and how it is a viable business opportunity for anyone with the right knowledge and business plan. The segments of this report are stated in such a way to highlight the process of formation of the relational database for the player management and the way it has been created using database softwares which include MySQL and MongoDB. Player management and roster optimization are critical components of any sports team's success. They involve using data-driven methods to effectively manage player performance, fitness, and game-day readiness while selecting the best possible line-up for each match. With advancements in technology, we can now go beyond manual decisions and leverage data integration and predictive analytics to make informed choices. This not only improves team performance but also provides a competitive edge by identifying optimal player combinations for different scenarios. Our system aims to revolutionize this process by integrating player stats, game conditions, and opponent details into a centralized platform that ensures smarter, faster, and more strategic roster decisions.

Managing professional hockey teams requires integrating diverse data sets and making data-driven decisions. Current methods often fall short due to the fragmentation of data and the absence of predictive analytics. Hockeyhive.com addresses these challenges by offering a comprehensive system for player management that leverages historical and real-time data.

In simple terms, this player management software and interface helps in maintaining real-time data with accurate statistics of each player which helps the stakeholders involved to build an optimal roster and team for any given game.

Problem Statement

We are tackling the inefficiencies and absence of data-driven decision-making in professional hockey club management. Current techniques frequently rely on intuition and isolated data sets, resulting in inefficient player selection and strategy development.

Specifically, we want to overcome the following challenges:

Data Fragmentation: Critical information such as player statistics, fitness measures, game circumstances, and opponent data are dispersed across various sources, making thorough analysis difficult.

Lack of Predictive Insights: Without integrated predictive analytics tools, teams fail to foresee player performance and suggest appropriate lineups.

Ineffective Roster Optimization: Without using historical and real-time data, player selection and game plans are less effective, diminishing the team's competitive advantage.

Project Objectives:

Develop a relational database model using SQL to store critical team and game data, such as player profiles, performance statistics, team metrics, and opponent information.

Gather historical and real-time data on player performance, game outcomes, and opponent analysis. Integrate fitness and performance metrics to ensure that player condition (heart rate, endurance, etc.) is factored into line-up decisions.

Develop a predictive model that uses data from the database to recommend the most efficient roster for any given game.

Regularly update the model as new data becomes available to ensure accuracy and relevance.

Project Scope:

Data Sources and Collection: The system will integrate various data streams that influence roster selection decisions, few of them are as follows: -

> Player Data: Basic information about players, including demographics (age, height, weight), position, contract status, and historical performance metrics. –
>
> Performance Metrics: Statistics such as goals, assists, ice time, plus/minus ratings, power-play performance, penalty minutes, and other relevant hockey metrics. –
>
> Fitness Metrics: Data such as heart rate, endurance, calories burned, and other physical fitness measurements to monitor player readiness.

SQL Querying: The system will use advanced SQL queries to help us analyze historical and real-time data. Identifying patterns in player performance relative to different opponents and game conditions. Correlating fitness data with performance to determine when a player is at peak form. Comparing Bruins' performance metrics against opposing teams to inform tactical roster changes. We will also be leveraging NoSQL database integration to tackle unstructured data related to ‘Fitness Metrics’ for individual players.

Roster Optimization Model: Develop a model that uses the data insights to generate optimized player lineups based on performance metrics, fitness conditions, and opponent analysis. The model will prioritize players with the best fitness and performance stats for upcoming games. The system will generate recommendations for the most effective combinations of players (forwards, defensemen, goalies) and coaching strategies.

Reporting and Visualization: The project will include the development of SQL queries and reports to deliver insights through tables, charts, and visualizations. These reports will be designed for use by coaching staff and team management to: - Track player performance over time. - Visualize game results and trends in team metrics. - Compare Bruins' stats with opponent data for upcoming games.

Business Use Cases & Benefits:

The Hockeyhive.com Player Optimization Project offers several transformative benefits:

1. Improved Roster Optimization: By recommending players based on their historical performance, fitness levels, and venue-specific trends, the system maximizes the effectiveness of each roster.
2. Enhanced Strategic Gameplay: With deeper insights into opponent-specific and game-day conditions, coaches can develop strategies tailored to the strengths of their selected roster.
3. Boosted Team Performance: Integrating predictive analytics into player management ensures that the team is consistently fielding its best players, improving overall success rates.

This system empowers team managers, coaches, and medical staff to make better-informed decisions. By centralizing all data and enabling real-time analysis, the Hockeyhive.com system bridges the gap between raw data and actionable strategies. For the Boston Bruins, this means moving beyond traditional methods of player selection and embracing a future where data-driven insights lead to tangible results on the ice.

Business Plan:

The Hockeyhive.com Player Optimization Project is designed not just as a technological innovation but as a sustainable and scalable business model within the rapidly growing sports management software industry. According to projections, the market size for sports management software is expected to reach 15.57 billion USD by 2029, signalling an immense opportunity for solutions like Hockeyhive.com to capture value and revolutionize the way professional hockey teams manage their players, data, and strategies.

Market Opportunity

Sports management is evolving into a data-centric field, where advanced tools and analytics are reshaping how decisions are made. However, existing systems often lack the depth and flexibility to address the unique needs of hockey teams, such as integrating structured and unstructured data, predicting player performance, and optimizing rosters for maximum efficiency. Hockeyhive.com fills this gap with a robust, AI-driven platform that leverages SQL and NoSQL databases, predictive modeling, and real-time insights to empower team managers and coaches. The projected growth of the sports management software market highlights the demand for solutions that deliver both technological and business value.

Revenue Model

To capitalize on this growing market, the Hockeyhive.com business plan includes the following revenue streams:

1. Subscription Model (SaaS):
    - The primary revenue stream is based on a subscription model, where teams pay a recurring fee to access the platform.
    - This model ensures consistent cash flow and allows for tiered pricing based on team size, league level, and additional features like advanced analytics or real-time updates.
2. Customization Fees:
    - Every team has unique requirements, and Hockeyhive.com offers customization services to tailor the platform to their specific needs.
    - These customization projects provide additional revenue while enhancing the platform's adaptability to different use cases.
3. Advertising and Sponsorships:
    - The platform's user interface provides opportunities for advertising and sponsorships, targeting brands aligned with sports and fitness industries.
    - This not only generates revenue but also adds value for advertisers by reaching a niche, high-value audience.
4. Vertical Integration:
    - Hockeyhive.com can expand its services to cover broader aspects of team management, such as ticket sales, venue management, and fan engagement.
    - Integrating these features into the platform creates a one-stop solution for teams and opens up new revenue opportunities.
5. Licensing Agreements:
    - The technology powering Hockeyhive.com can be licensed to other leagues, teams, or organizations outside of hockey, creating a scalable revenue model.
    - Licensing allows the platform to generate passive income while expanding its footprint into new markets.

Competitive Edge

The Hockeyhive.com business model is underpinned by the unique capabilities of the platform, which differentiate it from competitors in the sports management software industry:

- Comprehensive Data Integration: By combining structured player stats with unstructured fitness metrics, the platform provides a holistic view of player performance.
- Actionable Insights: Advanced predictive analytics and real-time updates ensure that teams have the information they need to make the best decisions.

- User-Friendly Interface: Designed with team managers and coaches in mind, the platform simplifies complex data and presents it in a way that's easy to understand and use.

These features not only make the platform indispensable for teams like the Boston Bruins but also create a compelling value proposition for teams across other leagues and sports.

Scalability and Growth Potential

As a scalable software-as-a-service (SaaS) platform, Hockeyhive.com has the potential to expand its reach far beyond the Boston Bruins. By targeting other hockey teams, leagues, and even other sports, the platform can grow its customer base significantly. The modular nature of the platform allows for seamless adaptation to new use cases, making it a versatile solution for sports management.

Additionally, the platform's data-driven approach aligns with industry trends toward greater reliance on analytics and technology. As more teams recognize the value of data in driving performance, Hockeyhive.com is well-positioned to become the go-to solution for player and roster optimization.

Impact

The robust business plan directly supports the goals of the Hockeyhive.com project by ensuring its sustainability and scalability. By leveraging the SaaS model, customization options, and additional revenue streams like advertising and licensing, the project creates a strong foundation for long-term success. This ensures that the platform can continue to innovate, offering new features and capabilities to meet the evolving needs of its users.

Furthermore, the business plan aligns with the core mission of the project: transforming hockey team management into a competitive edge. By providing actionable insights and optimizing player performance, the platform not only addresses current challenges but also sets the stage for future advancements in sports analytics.

Database Requirements:

In order to create a relational database for the player management and optimization. There were certain key database requirements useful for the same, which are explained further:

1. Players: From this table we need the inputs about the players which correlate with many different statistics. Few such variables are PlayerID, PositionID, Physical metrics.
2. Roster: This table consists of the data of historical roster formations for the previous matches. This can used to analyse the optimal roster for a given match in future. This table consists of data related to number of players in a given position and it also includes the data of coaches and captains.
3. Season Player Stats: This table consists of significant data related to the stats of a player in each season in the previous games. There are many statistics which helps in creating the best team for a given game. This table has data of Assists, Goals, Shots, Win%.

4. Team Stats: Team statistics are important for understanding the player formation in a team and their performances in a given team and position. This table has data of various variables like HomeWin%, AwayWin%, Points, OpponentStats.

These are few of the main inputs required for building this relational database and there are many other variables related to these tables and will help in making the management software better and efficient.

Database Structure:

Our Player management database is classified into key relational tables that are required for optimal and efficient database functioning:

Core Tables:

1. Teams: This stores the information about each team
2. Players: This table has the information of individual players and their IDs.
3. Games: The data represents individual game data with details such as GameID, results.
4. Venue: Stores information about each venue.

Operational Tables:

1. Roster: Tracks players assigned to a particular game or lineup (RosterID, PlayerID, GameID).
2. PlayerStats: Records performance statistics for players (e.g., Goals, Assists, Points).
3. SeasonPlayerStats: Tracks player stats over a season with breakdowns by PlayerID and Season.
4. SeasonGoalieStats: Similar to SeasonPlayerStats, but specific to goalies (Saves, Shutouts).

Management Tables:

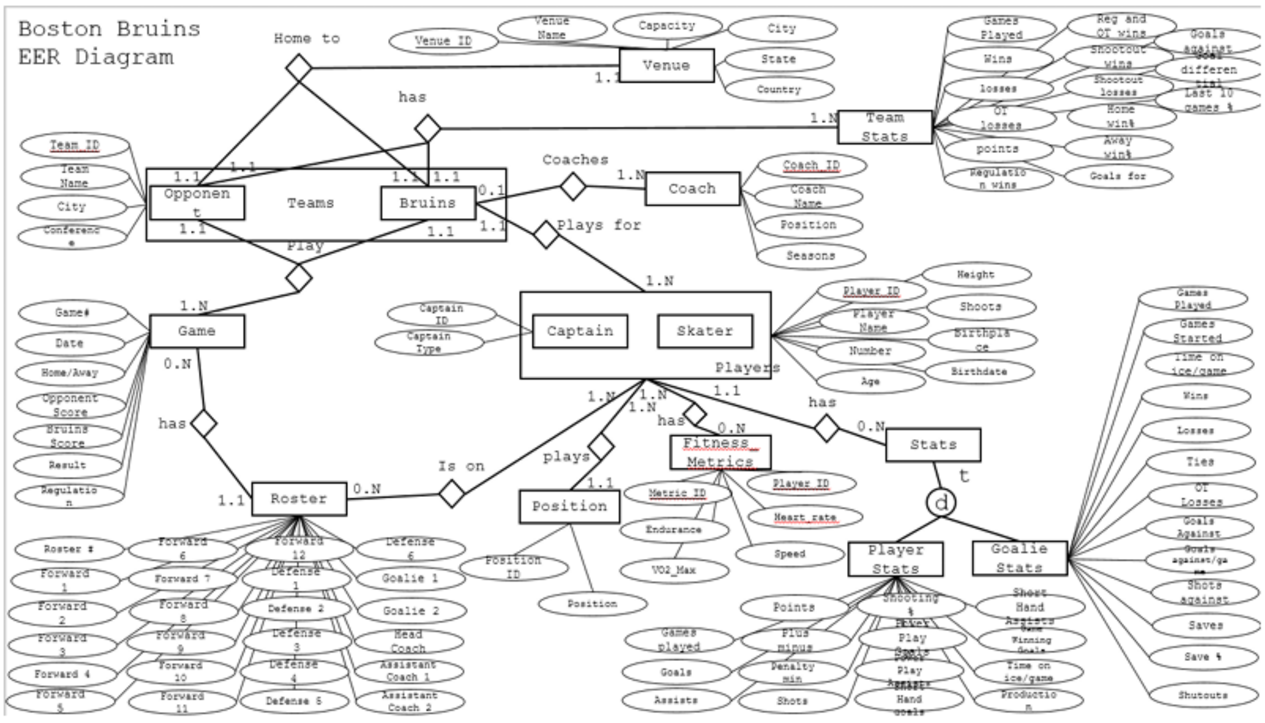
1. Captain: Tracks captaincy roles, such as captain and alternates (CaptainID, PlayerID, CaptainType).
2. Coach: Maintains data on coaches, including their positions and associated teams (CoachID, TeamID).
3. Position: Defines player positions (e.g., Forward, Défense, Goalie) with a PositionID.

Support Tables:

Support tables store supplementary data that enhances the functionality of the system or provides metadata.

1. Team Stats: Contains aggregated statistics for teams (TeamID, Wins, Losses, Points).
2. Fitness Metrics: Stores physical fitness data for players (MetricID, PlayerID, VO2_Max, Heartrate).

Entity Relationship Diagram:



The Enhanced Entity Relationship Diagram (EER) for the Boston Bruins serves as a robust data architecture framework for the Hockeyhive.com Player Management System. This diagram showcases the interconnected relationships among the various entities essential for managing players, games, venues, and performance metrics. It highlights how structured and unstructured data are seamlessly integrated to support roster optimization, performance analytics, and decision-making. There are 13 distinct tables in this EER diagram. Each table plays a critical role in managing the data relationships and supporting the Hockeyhive.com Player Management System.

Key Entities and Their Roles

1. Teams and Venues:

   - Team Entity: Represents the Boston Bruins and their opponents. Each team is uniquely identified by a Team_ID and includes attributes such as Team Name, City, and Conference. These attributes provide critical contextual information about each team's identity and regional affiliation.
   - Venue Entity: Tracks details about game locations, including Venue Name, Capacity, City, State, and Country. Each venue is uniquely identified by a Venue_ID.

- Relationship: Each team is associated with exactly one home venue (1:1 relationship). This relationship is critical for analyzing venue-specific performance, such as home versus away game dynamics, and for managing logistical details.

2. Games and Opponents:

    - Game Entity: Captures comprehensive game details, including the Game Date, Scores, Home/Away status, and Opponent. Each game is uniquely tied to a specific date and involves detailed performance tracking.
    - Opponent Entity: Represents opposing teams in games. This entity is linked to the Game entity to track the performance of the Boston Bruins against specific opponents.
    - Relationship: A team plays multiple games in a season (1:N relationship). The link between games and opponents enables in-depth opponent-specific analysis, such as identifying patterns in win/loss records, which helps coaches make strategic adjustments.

3. Players, Positions, and Roster Management:

    - Player Entity: Represents individuals on the team, categorized as Skaters or Goalies. Each player is uniquely identified by a Player_ID and includes attributes such as Name, Number, Height, Weight, Age, and Strength.
    - Roster Entity: Tracks player participation in specific games and positions. Positions such as Forwards, Defenders, and Goalies are tied to individual players.
    - Relationship: Players are linked to the roster entity (1:N relationship), and each roster is associated with specific games. This ensures that rosters are dynamically updated based on player fitness, performance, and game-day conditions.

4. Fitness Metrics:

    - Fitness Metrics Entity: Tracks unstructured data, such as VO2 Max, Endurance, Speed, and Heart Rate. These metrics are essential for assessing player physical readiness and are stored in a NoSQL database for flexibility in handling unstructured data.
    - Relationship: Each player is associated with multiple fitness metrics over time (1:N relationship). This enables real-time fitness monitoring and ensures that only players at their peak physical condition are included in rosters.

5. Player Stats and Game Performance:

   - Player Stats Entity: Captures detailed performance metrics for each player, including Points, Goals, Assists, and Penalty Minutes. These metrics provide an objective measure of a player's contribution during games.
   - Goalie Stats Entity: Focuses on goalie-specific metrics, such as Saves, Shutouts, and Goals Allowed. This distinction ensures tailored analytics for different player roles.
   - Relationship: Player stats are linked to individual players (1:N relationship). This allows historical performance data to be leveraged for predictive modeling, helping coaches identify top performers under various conditions.

6. Coaches and Captains:

   - Coach Entity: Represents team coaches and includes attributes such as Coach_ID, Name, Position, and Seasons. Coaches are critical decision-makers in roster and strategy planning.
   - Captain Entity: Represents team captains, who serve as leaders within the roster. The captain is linked to the team (1:1 relationship), emphasizing their unique role in team dynamics.
   - Relationship: Each coach is linked to a team (1:N relationship). The captain's relationship with the roster and players reinforces their role as an intermediary between coaches and players.

Primary and Foreign Keys of the database:

Primary Keys: TeamID, VenueID, PlayerID, GameID, RosterID.

Foreign Keys: HomeVenueID → References, Venue.VenueID, PlayerID → References Player.PlayerID, GameID → References Game.GameID, OpponentID → References Team.TeamID

Detailed Relationship Analysis

1. Team-to-Venue Relationship:

   - Each team has a specific home venue where most games are played. By analyzing this relationship, the system can provide insights into home versus away performance trends, which are crucial for optimizing team strategy.

2. Player-to-Fitness Metrics Relationship:

- Fitness metrics are dynamic and constantly updated. This relationship allows medical staff and coaches to track a player's physical condition over time and adjust training or rest schedules accordingly. Real-time integration of these metrics ensures that only fit players are included in game rosters.

3. Game-to-Roster Relationship:

   - This relationship connects individual games with player rosters. It ensures that roster changes, such as substitutions or exclusions, are accurately reflected for each game. This is especially important for compliance with league regulations and for analyzing game-specific player performance.

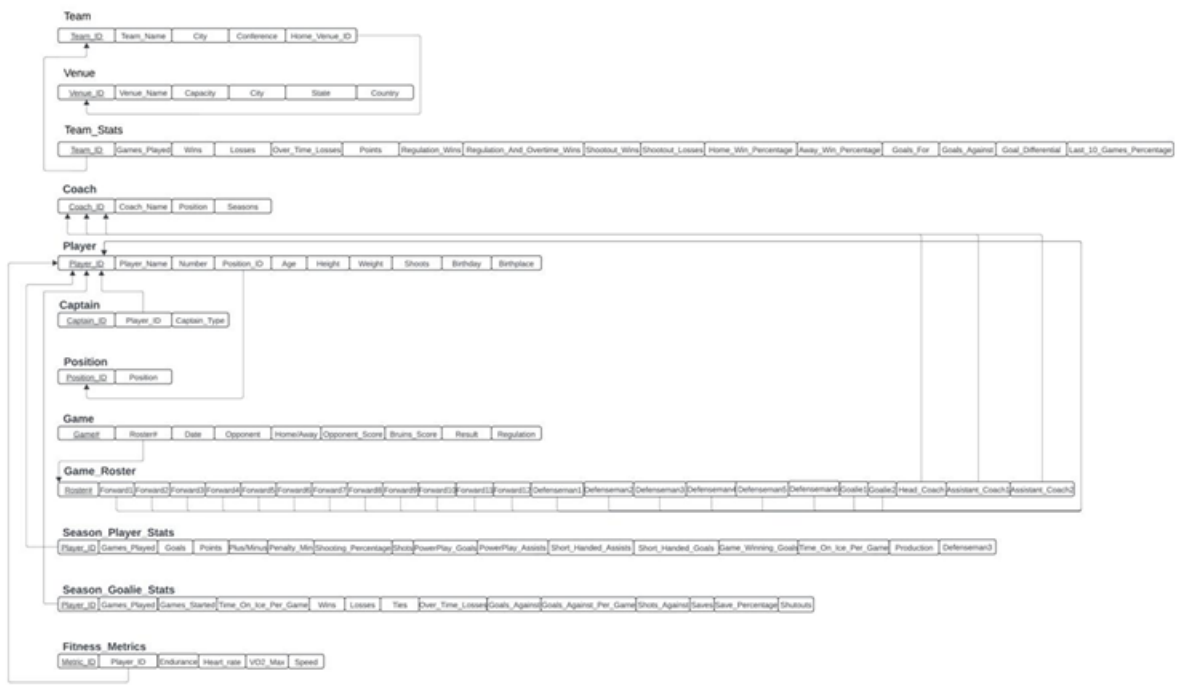
4. Player-to-Stats Relationship:

   - Historical stats linked to players provide the foundation for predictive analytics. For example, a player's average performance against a specific opponent or in a particular venue can inform roster decisions and game strategies.

5. Team-to-Opponent Relationship:

   - By linking teams to their opponents through games, the system enables opponent-specific analysis. This includes evaluating head-to-head records, identifying key rivalries, and tailoring strategies to exploit opponent weaknesses.

Relational Schema (UML):



Functional dependencies describe relationships between attributes in a table:

Player Table:

PlayerID → {PlayerName, Number, PositionID, Age, Height, Weight, Shoots, Birthdate}

This ensures that all player details are uniquely identified by PlayerID.

Game Table:

GameID → {Date, Home/Away, OpponentScore, BruinsScore, Result}

Composite key dependency: {GameID, TeamID} → {TeamStats, RegulationWins}

Team Table:

TeamID → {TeamName, City, Conference, HomeVenueID}

Functional dependency ensures that all attributes of a team are uniquely tied to TeamID.

Captain Table:

CaptainID → {PlayerID, CaptainType}

The functional dependency reflects the hierarchical structure, where each captain type (e.g., Captain, Alternate) is linked to a specific player.

Steps for Optimization:

Normalizing the Roster to remove repeating groups and align with 1NF:

We created a RosterPosition table where each forward/defenseman is listed in individual rows with references to PlayerID and RosterID. For avoiding storing derived attributes like percentages or aggregates (GoalsFor, GoalsAgainst) directly in tables.

Normalization Analysis:

1. First Normal Form (1NF)

Requirement: Ensure that all attributes contain atomic (indivisible) values, and there are no repeating groups or arrays. Tables like Player, Team, Venue, and Game follow 1NF by storing atomic values (e.g., PlayerName, TeamName, City).

Attributes such as Forward1, Forward2 in the Roster entity seem to break 1NF since they represent repeating groups. These should ideally be stored in a separate table with each forward or position represented as a separate row.

2. Second Normal Form (2NF)

Requirement: Ensure that all non-key attributes are fully functionally dependent on the primary key. Remove partial dependencies.

Example of 2NF:

In the Player table, attributes such as PlayerName, PositionID, and Height are functionally dependent on the primary key (PlayerID), adhering to 2NF.

3. Third Normal Form (3NF)

Requirement: Ensure that no non-prime attribute is transitively dependent on the primary key.

Example of 3NF:

In Venue, attributes like City, State, and Capacity directly depend on VenueID and are not transitively dependent through other attributes.

The provided relational schema is the backbone of the Player Management and Roster Optimization System.

1. Data Centralization:

   - The schema ensures that all relevant data about teams, players, games, and venues is interconnected, providing a centralized platform for data management.

2. Real-Time Analytics:

    - With tables like Game, Team Stats, and Season Player Stats, the system enables real-time performance tracking, helping managers make informed decisions.

3. Roster Optimization:

    - By analyzing relationships between Fitness Metrics, Game Roster, and Season Stats, the software can recommend the best player combinations for each match.

4. Automation:

    - Triggers and stored procedures in the database (as discussed earlier) use this schema to automatically update statistics, ensuring data consistency.

5. Scalability:

    - The relational structure is scalable, allowing easy integration of new teams, players, or metrics as the system grows.

Relational Schema Overview:

The relational schema for the Player Management and Roster Optimization System provides a structured approach to storing and analyzing data. By interconnecting entities such as teams, players, games, and venues, the database facilitates efficient management and decision-making.

Core Features:

1. Comprehensive Data Integration:

    - Centralizes data from diverse domains, including team performance, player statistics, and fitness metrics.

2. Real-Time Updates:

    - Automates updates to key statistics through triggers and procedural logic.

3. Performance Tracking:

    - Supports detailed tracking of player and team performance using season and game-specific metrics.

4. Decision Support:

   - Provides actionable insights for roster optimization by analyzing player fitness, roles, and game-day performance.

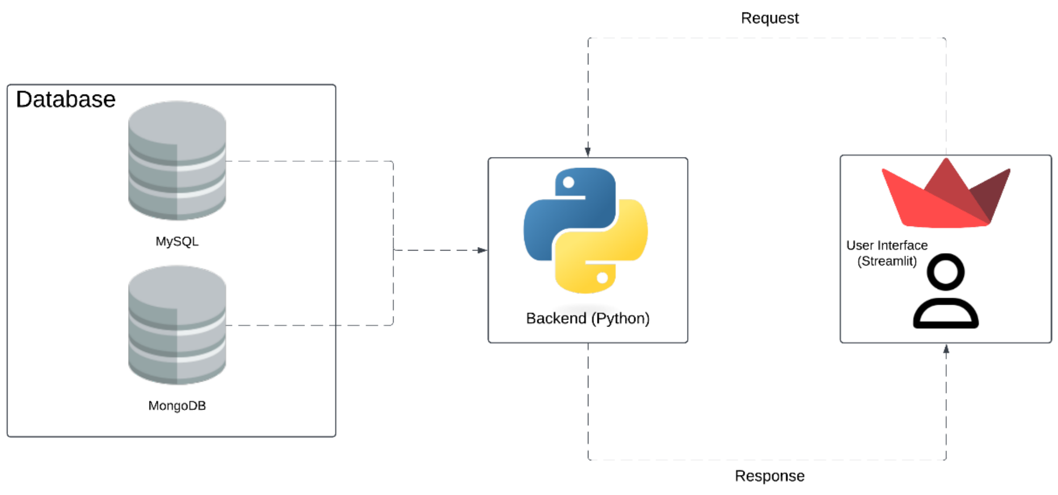
Relevance to Player Management Software

The schema serves as the foundation of the player management software, ensuring seamless data flow and supporting advanced analytics. By leveraging this schema, the software enables:

- Strategic Planning: Data-driven decisions for roster optimization.
- Scalability: Adding new teams, venues, or metrics as required.
- Efficiency: Minimizing manual data handling and enhancing operational accuracy.

This relational schema is a critical component of the Player Management System, enabling advanced features such as roster optimization, performance tracking, and real-time analytics. Its design ensures scalability and operational efficiency, aligning with the software's objective to revolutionize team management.

Application Structure:



<u>Database Layer:</u> We integrated the MySQL and MongoDB databases for managing the data of the application.

MySQL: -

Purpose: It stores all the structured data for the player management and roster analysis application that we have built. It is an efficient and optimal choice of these kind of data structures.

Usage: We have integrated it with PyCharm to get response for the front-end activities.

MongoDB:

Purpose: It stores all the unstructured data for the player management and roster analysis application that we have built. This comes in handy for integrating with datasets which has large variables for example, the fitness metrics component of our database.

Usage: Using this has led to better integration of all the important variables to the relatioal database and also helped in building a better connect with the PyCharm IDE compiler.

Python Back-end: This is one of the important components of our project, since it helps with bridging the UI and databases.

Functionality: We designed this in such a way to enable us in processing the front-end requests on the user interface. Then python plays the role of a arbitrator and processes this information received from the front-end and helps in retrieving the data from relational database which is in MySQL and MongoDB.

The data that it receives from the back-end databases, python further processes responses and sends it back to the UI.

Streamlit UI(User Interface):

Purpose: This is the front-end layer through which user can interact with the database.

Functionality: This is functional in conducting basic operations such as adding, updating the database records. Streamlit's powerful integration with Python empowers us to execute highly customizable and dynamic queries on our SQL database, delivering precise and actionable insights for comprehensive roster analysis.

Interaction framework:

User interaction

Users enter the details that they want to add using the forms available on the respective pages.

Python back-end operations

Once, the details are entered streamlit initializes the same & sends it over to the backend data integration.

Database operations

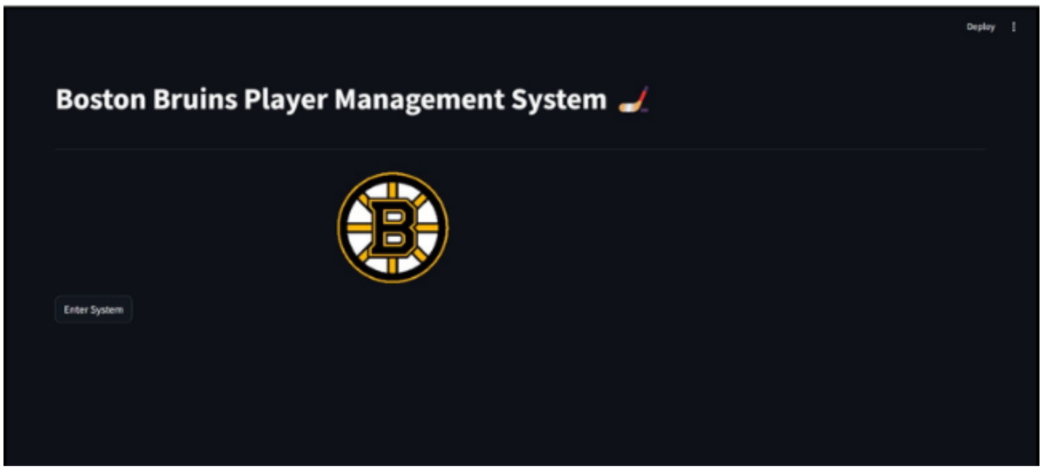
The input data is then added to the MongoDB database & then through the integration between MongoDB & MySQL, it is added to the MySQL database as well.

Response

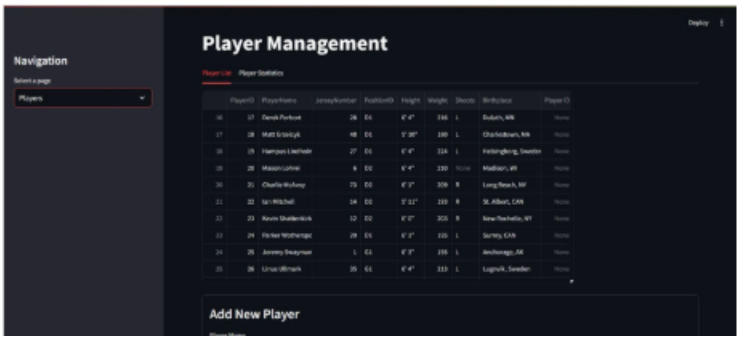
The data is then displayed in the front-end of the website in real-time.

User Interface:

The relational database we created till this point is for developing this front-end user interface. This is developed with the motive of extending this model to other sports leagues upon successful performance and feedback from the common users. This User Interface is developed for updating the information on a real-time basis for easy reference of the coaches and managers for deciding the optimal rosters for a particular game. Various parts of this interface and their purpose will be explained further:



The home screen is the starting point of the system. It welcomes users with the title "Boston Bruins Player Management System" and displays the Boston Bruins logo. A single button labelled "Enter System" guides users into the application. With this simple and elegant front page, we want to make it easy for the user to navigate to the desired statistic or the table which they want to the access the information from.

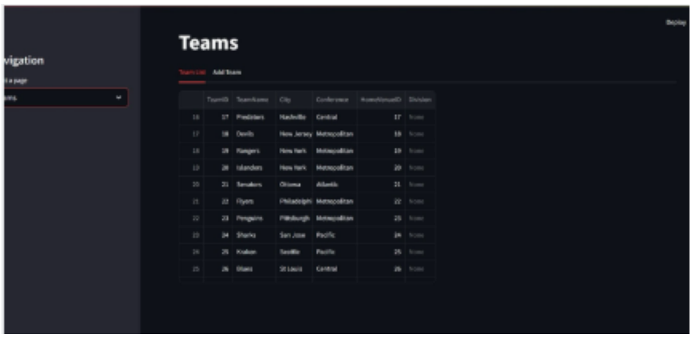


Player Management

Player List Player Statistics

	PlayerID	PlayerName	JerseyNumber	PositionID	Height	Weight	Shoots	Birthplace	Player ID
16	17	Derek Forbort	28	D1	6' 4"	216	L	Duluth, MN	None
17	18	Matt Grzelcyk	48	D1	5' 10"	180	L	Charlestown, MA	None
18	19	Hampus Lindholm	27	D1	6' 4"	214	L	Helsingborg, Sweden	None
19	20	Mason Lohrei	6	D2	6' 4"	210	None	Madison, WI	None
20	21	Charlie McAvoy	73	D2	6' 1"	209	R	Long Beach, NY	None
21	22	Ian Mitchell	14	D2	5' 11"	210	R	St. Albert, CAN	None
22	23	Kevin Shattenkirk	12	D2	6' 0"	205	R	New Rochelle, NY	None
23	24	Parker Wotherspoon	29	D1	6' 1"	195	L	Surrey, CAN	None
24	25	Jeremy Swayman	1	G1	6' 3"	195	L	Anchorage, AK	None
25	26	Linus Ullmark	35	G1	6' 4"	212	L	Lugnvik, Sweden	None

Add New Player


vigation

t a page

ams

Deploy

Teams

Teams List | Add Team

	TeamID	TeamName	City	Conference	HomeVenueID	Division
16	17	Predators	Nashville	Central	17	None
17	18	Devils	New Jersey	Metropolitan	18	None
18	19	Rangers	New York	Metropolitan	19	None
19	20	Islanders	New York	Metropolitan	20	None
20	21	Senators	Ottawa	Atlantic	21	None
21	22	Flyers	Philadelphi	Metropolitan	22	None
22	23	Penguins	Pittsburgh	Metropolitan	23	None
23	24	Sharks	San Jose	Pacific	24	None
24	25	Kraken	Seattle	Pacific	25	None
25	26	Blues	St Louis	Central	26	None

These snippets showcase examples of how data is displayed and managed within the system. The UI ensures that all relevant information is accessible, editable, and structured for ease of use.

Player Management Interface:

The Player Management page allows users to view and manage player-specific details.

- Key functionalities include:
- Displaying player data such as Player Name, Jersey Number, Position, Height, Weight, Shoots (Left/Right), and Birthplace.
- Options to add new players dynamically via a user-friendly form at the bottom. (This will be further explained in the report)
- This interface ensures that team managers can quickly access and modify player profiles, keeping the roster updated and ready for strategic decisions.

This page is just one example of the system's capabilities. Other player-related tables, such as fitness metrics, season statistics, and game performance, are also available in the software for deeper analysis.

Team Management

- The Teams page enables users to manage team-related data.
- Key functionalities include:
  - Displaying critical team information such as Team Name, City, Conference, Division, and Home Venue.
  - Providing a way to add new teams to the system, ensuring scalability for managing multiple leagues or tournaments.
- This interface helps ensure that team-related data is centralized, making it easier to associate teams with players, venues, and game schedules.

These are few examples of how team data is presented. The software also includes related tables for team statistics, venue details, and historical performance to provide a comprehensive overview.

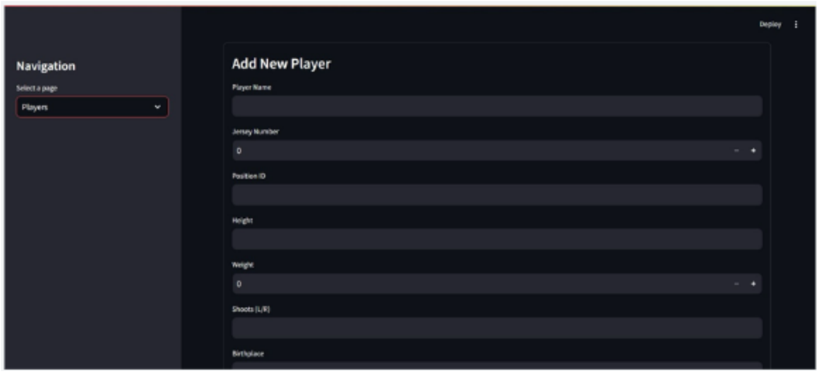
Comprehensive Data Integration

While these examples highlight the Player Management and Team Management pages, the system encompasses a wide range of additional tables and views, including:

- Game Statistics: Tracking game results, scores, and player performance during each match.
- Fitness Metrics: Monitoring player fitness levels, endurance, and readiness for games.
- Season Statistics: Providing cumulative performance insights for players and teams.

These additional tables enable the system to offer robust data integration and analytics, ensuring it meets the needs of modern sports management.

The user interface is a critical component of the software, providing intuitive navigation and efficient data management. These examples demonstrate the software's capability to handle complex player and team data while ensuring usability for end-users. The inclusion of other tables for game and performance stats further enhances the system's functionality, making it a comprehensive tool for player and roster management.



Add New Player

Player Name

Jersey Number
0

Position ID

Height

Weight
0

Shoots (L/R)

Birthplace

The Add New Player component is a crucial feature of the Player Management module, designed to provide users with a seamless method for adding new players to the database. This interface is user-friendly and ensures that all critical data about a player is collected and stored accurately in the system at the backend where the SQL & NoSQL are integrated. It is particularly useful for team managers and administrators who need to maintain up-to-date records of their rosters as new players join or existing players are reassigned. The form includes fields that capture essential details about the player, such as their name, jersey number, position, physical attributes (height and weight), shooting preference (left or right), and birthplace. These fields ensure that all relevant data is recorded systematically and aligns with the relational database structure. For example, the Jersey Number field is not just a display element but also a unique identifier for players within the team, making it easy to reference them across different modules, such as game rosters or statistics tracking. The Position ID dropdown links directly to the database's position table, ensuring that every player is assigned a role (e.g., Forward, Defenseman, or Goalie) that corresponds to the relational schema. This interface also promotes data accuracy by standardizing the input process. Each field is validated to prevent errors, such as duplicate entries or missing information. For instance, the Shoots (L/R) field requires a selection, ensuring clarity about the player's handedness, which is critical for game strategies and positional assignments.

Moreover, this interface is just one example of how the Player Management Software streamlines data entry and management. Beyond adding new players, the system provides additional functionality for editing player details, viewing statistics, and linking players to their respective teams and game rosters. This interconnected structure is made possible by the relational database underlying the software.

Instructions For Set-Up & Usage:

The instructions for successfully setting up the database – along with integration & front-end deployment are highlighted below –

1. Open the 'bruins_project' SQL text file using MySQL Workbench or any other MySQL compiler. This is our main SQL database where all of our data is stored.
2. Create an integration between the MySQL database & MongoDB by using the file – 'Data Integration.py'. Run this file on any Python IDE & make sure you have 'mysql-connector' & '' packages installed within the IDE.
3. Once the data integration is executed successfully, make sure you have 'streamlit', 'pandas' & 'plotly' packages installed in the IDE & move on to the front-end execution. Run the file 'Front-end Website.py' file.
4. Upon successful execution, you will receive a warning message in the console as mentioned here – 'Warning: to view this Streamlit app on a browser, run it with the following command: streamlit run C:\Users\SAURABH\PycharmProjects\pythonProject1\Front-end Website.py'
5. After this warning message is received, open the Python terminal in the IDE and run the command as mentioned in the warning message.
6. Running the 'streamlit run' command initializes the Streamlit package & the front-end website is deployed.
7. You should see the title page of our front end system with the 'Enter System' button

The instructions for using the front-end website to view and manipulate the data in the database are mentioned below –

1. Once the front-end website is opened, click on the 'Enter System' button.
2. You can select the table that is to be viewed from the Navigation tab.

3. To add any data to the database find the Add tab on the respective page, enter the details & click on Submit.