FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

The functional and non-functional requirements for the GoTeam Application are:

* **Functional Requirements for Existing Functionality:**

1. **Retrieve All Players:**
2. **Description:** The system must provide an endpoint to retrieve a list of all players.
3. **Purpose:** Users need a comprehensive list of all players for various purposes, like scouting to identify potential talent for recruitment to the teams, and team management to efficiently manage team’s roster, substitutions, and game strategies.
4. **Acceptance Criteria:** Users can make a GET request to the ‘/players/getAllPlayers’ endpoint. The API returns a list of all players in the database. The response includes player details such as player name, jersey number, and team name. If no player exists, then it returns a null object.
5. **Retrieve Player by ID:**
6. **Description:** The system must provide an endpoint to retrieve a player’s information by their unique player ID.
7. **Purpose:** Users often need to access detailed information about a specific player, and player IDs provide a reliable means of identification for various cases like viewing customized player profiles that include career highlights, performance statistics, injury history.
8. **Acceptance Criteria:** Users can make a GET request to the ‘/players/getPlayerbyId/{playerId}’ endpoint, where ‘{playerId}’ represents the unique identifier of the player. If a player with the specified ID existes, the API returns the player’s details. If no player with the specified ID is found, the API returns an appropriate error message.
9. **Retrieve Players by Team:**
10. **Description:** The system must provide an endpoint to retrieve a list of players belonging to a specific sports team.
11. **Purpose:** Users often need to view the roster of a particular team, which includes the player associated with that team. This team roster can be used by coaches and team managers to prepare for upcoming games by knowing the availability of the players in that team. Access to the team roster allows fans to stay informed about the players representing their team, fostering a sense of connection and engagement.
12. **Acceptance Criteria:** Users can make a GET request to the ‘/players/getPlayersByTeam/{teamName}’ endpoint, where ‘{teamName}’ represents the name of the sports team. The API returns a list of players who belong to the specified team. If no players are found for the given team, the API returns an appropriate message indicating that no players are available for that team.
13. **Add Player:**
14. **Description:** The system must provide an endpoint to create a new player.
15. **Purpose:** Users need the ability to add new players to the database in order to expand the team’s roster, identifying young talents and brought into the system. When a player is injured or unavailable, teams may need to bring in replacement players.
16. **Acceptance Criteria:** Users can make a POST request to the ‘/players/createPlayer’ endpoint with a valid player object in the request body. The API creates a new player in the database with the provided information. If the request body is missing or invalid, the API returns an appropriate error message.
17. **Add Multiple Players:**
18. **Description:** The system must provide an endpoint to create multiple players simultaneously.
19. **Purpose:** Users may need to add multiple players in a single operation, saving time and effort in scenarios where sports teams are recruiting multiple players at once, or when multiple players are transitioning to a new sports management system from the old system.
20. **Acceptance Criteria:** Users can make a POST request to the ‘/players/createManyPlayers’ endpoint with a list of valid player objects in the request body. The API creates all the provided players in the database. If the request body is missing or contains invalid data, the API returns an appropriate error message.
21. **Update a Player by ID:**
22. **Description:** The system must allow users to update an existing player’s information.
23. **Purpose:** Users need the ability to correct or modify player data as needed. In sports management, player information such as names, positions, or contact details may change over time. Allowing updates ensures that the database reflects the most current and accurate information. It can happen in another case when a player transfers to another team within the same organization or to a different team altogether, their team affiliation and related information need to be updated.
24. **Acceptance Criteria:** Users can make a PUT request to the ‘/players/updatePlayer/{playerId}’ endpoint, where ‘{playerId}’ represents the unique identifier of the player to be updated. If the player with the specified ID exists, the API updates the player’s information with the provided data. If the player does not exist or the request body is missing or invalid, the API returns an appropriate error message.
25. **Delete All Players:**
26. **Description:** The system must provide a function to delete all players from the database.
27. **Purpose:** Users may need to perform bulk deletion of player records. It is useful when organizations want to start fresh with a clean slate. It allows them to remove all player records and begin anew, whether it’s a new season, a new team, or a new phase of operations.
28. **Acceptance Criteria:** Users can make a DELETE request to the ‘/players/deleteAllPlayers’ endpoint. The API deletes all player records in the database. If no players exist in the database, the API should return a message indication that there are no players to delete.
29. **Delete Player by ID:**
30. **Description:** The system must provide a function to delete player by their unique player ID from the database.
31. **Purpose:** Users need the ability to remove individual players from the database when necessary. Sports teams often undergo changes in their rosters due to factors such as transfers, retirements , or changes in player status. Removing individual players enables teams to keep their rosters accurate and up-to-date.
32. **Acceptance Criteria:** Users can make a DELETE request to the ‘/players/deletePlayerById/{playerId}’ where ‘{playerId}’ represents the unique identifier of the player to be deleted. If a player with the specified ID exists, the function deletes the player’s record from the database. If no player with the specified ID is found, the funciton should return an appropriate message that no player was deleted.
33. **Delete Players by Team:**
34. **Description:** The system must provide a function to delete all players belonging to a specific sports team from the database.
35. **Purpose:** Users may need to remove all players associated with a particular team for various reasons like team disbandment where all players associated with that team need to be removed from the database.
36. **Acceptance Criteria:** Users can make a DELETE request to the ‘/players/deletePlayersByTeam/{teamName}’. The function identifies all players in the database belonging to the specified team and deletes their records. If no players are found for the given team, the function should return an appropriate message that no players were deleted.

* **Functional Requirements for New Functionality:**

1. **Player Statistic Tracking:**
2. **Description:** The system shall automatically assign badges to players based on the player’s total score, calculated as the sum of goals and assists.
3. **Purpose:** It is used to recognize and reward players based on their performance in sports. Badges are a visual representation of a player’s achievements and skill level.
4. **Acceptance Criteria:** Users can make a GET request to the ‘/players/getAllPlayers’. When a player’s total score satisfies specific predefined conditions, the system shall assign the corresponding badge to the player(gold, silver or bronze). The badges awarded to each player shall be visible on their profiles and can be used for player comparison and recognition.
5. **Count Players by Team:**
6. **Description:** The system must provide a function to count the number of players associated with a specific sports team.
7. **Purpose:** By having access to the count of players on a particular team, they can effectively manage and maintain team rosters. It is also crucial for allocating resources such as uniforms, equipment, and playing time based on the size of the team.
8. **Acceptance Criteria:** Users can make a GET request to the ‘/players/countPlayersInATeam/{teamName}’ endpoint. The function identifies all players in the database belonging to the specified team, count the number of players associated and return it.
9. **Get the Top Goal Scorer:**
10. **Description:** The system must provide a function to retrieve the top goal scorer among a group of players who have the same badge category based on their total number of goals scored.
11. **Purpose:** This feature allows sports organizations, teams and fans to recognize and celebrate the outstanding achievements of players who excel in specific skill categories, as represented by their badges. It highlights top performers who have consistently demonstrated their goal-scoring prowess within a particular skill level.
12. **Acceptance Criteria:** Users can make a GET request to the ‘/players/topGoalScorer/{badgeAwarded}’ endpoint. The function identifies all players in the database belonging to the specified badge category, determines the player with the highest total number of goals scored among the players in the specified badge category and return that player.

* **Non-Functional Requirements:**

1. **Performance:**
2. **Response Time:** The system shall provide rapid response times for user interactions, ensuring that operations such as retrieving player statistics, assigning badges, and counting players execute within acceptable time frames.
3. Scalability: The application should be able to scale horizontally to accommodate an increasing number of players without a significant degradation in performance.
4. **Data Management:**
5. **Data Integrity:** The system shall maintain the integrity of player data, ensuring that updates and modifications are accurate and consistent.
6. **Documentation:**
7. **User documentation:** Clear and comprehensive user documentation shall be provided to assist users in understanding the application’s features and functionality.