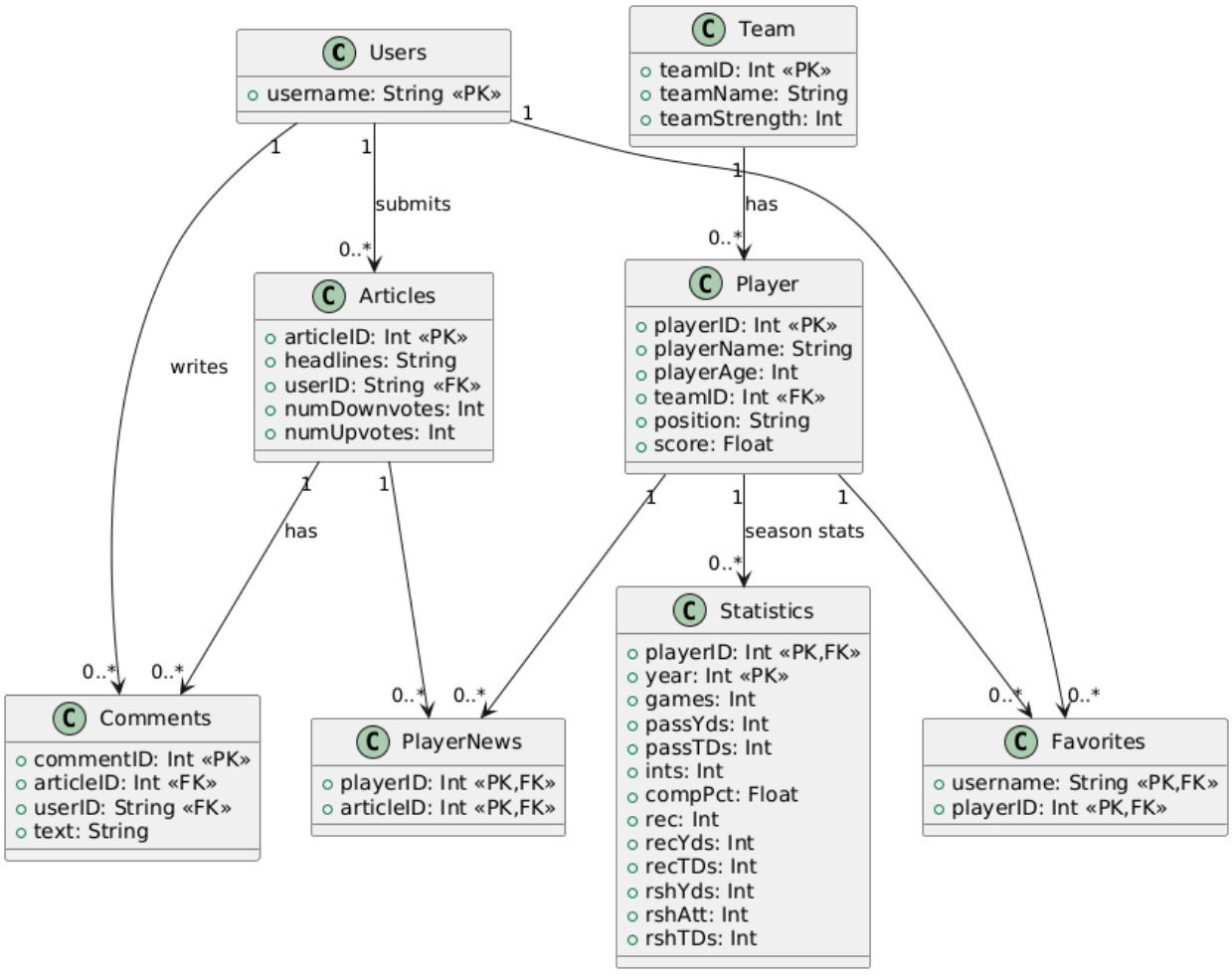
UML Diagram


Users
username: String «PK»
Team
teamID: Int «PK»
teamName: String
teamStrength: Int
submits
has
writes
Articles
articleID: Int «PK»
headlines: String
userID: String «FK»
numDownvotes: Int
numUpvotes: Int
Player
playerID: Int «PK»
playerName: String
playerAge: Int
teamID: Int «FK»
position: String
score: Float
has
season stats
Comments
commentID: Int «PK»
articleID: Int «FK»
userID: String «FK»
text: String
PlayerNews
playerID: Int «PK,FK»
articleID: Int «PK,FK»
Statistics
playerID: Int «PK,FK»
year: Int «PK»
games: Int
passYds: Int
passTDs: Int
ints: Int
compPct: Float
rec: Int
recYds: Int
recTDs: Int
rshYds: Int
rshAtt: Int
rshTDs: Int
Favorites
username: String «PK,FK»
playerID: Int «PK,FK»

Assumptions for Entities

- Users
    - Only the username is stored in the users entity, and this will be used for authentication. We do not need to store any other user data.
- PlayerInfo (& Favorites relation)
    - Each player can only be assigned to one team. In other words, player-to-team is a many-to-one relationship; a team can have many players but each player only has one team.
    - A user can store a list of favorite players. Each player can be favorited by many users. Thus, users-to-players is a many-to-many relationship, so we store it in its own table Favorites.
    - Lastly, we store the player's ML-generated score here which we will use as a comparison metric against other players.
- Statistics
    - The statistics entity uses a composite key of Player and Year. This is because a player has statistics from each season, thus we add in the year as a second attribute in our primary key to store multiple seasons for each player. This is also why we choose to store statistics in a separate entity than PlayerInfo. Thus, since a player can have multiple statistical seasons, statistics-to-players is a many-to-one relationship.
    - For the actual data stored in the table, not all columns will be filled for certain players. The statistical columns filled in will be determined by the player's position, as different positions have different stats associated with them. The logic of which columns to include when querying will be determined by the position associated with the PlayerID, stored in the PlayerInfo entity. This will be implemented programmatically in our backend.
- Articles (& PlayerNews relation)
    - An article can be about many players, and a player can have many articles talking about him. Thus, articles-to-players is a many-to-many relationship, so we store this in its own table PlayerNews.
    - Additionally each article is uploaded by one user. A user can submit many articles. Thus, article-to-user is a many-to-one relationship.
    - Lastly, we keep track of the number of upvotes and downvotes a certain article has. Once the downvotes reach a certain number (for instance 5), the article will be removed. We will implement this as a trigger such that after updating a news article's downvotes, we check if the downvotes surpasses the threshold and remove it if it does. We will implement this as a cascade such that all entries referencing this ArticleID in other relations will be removed.
- Comments
    - Comments are added by users for each article. Each comment can only be associated with one article, while one article can have many comments, so comments-to-articles is a many-to-one relationship.

    - Users can add many comments, but each comment is only added by one user. Thus users-to-comments is a one-to-many relationship.
    - Users can edit or remove comments that they posted, thus satisfying the CRUD requirements.
- Team
    - We simply store the team information here as well as its overall team strength, calculated based on the scoring of its players and the overall team record.

Normalization

We will perform 3CNF normalization on our table.

- The Users relation is already 3NF because there are no functional dependencies as the relation consists of only a singular primary key attribute (Username).

- The PlayerInfo relation is already normalized as the only functional dependency is PlayerID implies PlayerName, PlayerAge, TeamID, Position, and Score. Since PlayerID is a primary key that uniquely identifies all other attributes in the relation, this is 3NF.

- The Statistics relation is already normalized as the only functional dependency is (PlayerID, Year) implies Games, PassYds, PassTds, Ints, CompPct, Rec, RecYds, RecTds, RshYds, RshAtt, RshTds. Since (PlayerID, Year) is a superkey that uniquely identifies all other attributes in the relation, this is 3NF.

- The Articles relation is already normalized as the only function dependency is ArticleID implies Headline, UserID, NumUpvotes, NumDownvotes. Since ArticleID is a primary key that uniquely identifies all other attributes in the relation, this is 3NF.

- The Comments relation is already normalized as the only functional dependency is CommentID implies ArticleID, Text, and UserID. Since CommentID is a primary key that uniquely identifies all other attributes in the relation, this is 3NF.

- The Teams relation is already normalized as the only functional dependency is TeamID implies TeamName and TeamStrength, where TeamID is a primary key that uniquely identifies all other attributes in the relation.
- The Favorites relation is already normalized as there are no functional dependencies. (UserID, PlayerID) is a superkey, and this relation acts as a wrapper for the many to many relationship between the Players and Users relations.

- Finally, the PlayerNews relation is already in 3NF, as there are no functional dependencies. This relation acts as a wrapper for the many to many relationship between the Players and Articles relations.

As we can see, all relations in our schema satisfy 3NF, even without any modifications.

Relational Schema

Users(
  Username: VARCHAR (50) [PK]
)

PlayerInfo(
  PlayerID: INT [PK]
  PlayerName: VARCHAR (250)
  PlayerAge: INT
  TeamID: INT [FK to Team.TeamID]
  Position: VARCHAR (2)
  Score: FLOAT
)

Statistics(
  PlayerID: INT [FK to PlayerInfo.PlayerID]
  Year: INT
  Games: INT
  PassYds: INT
  PassTDs: INT
  Ints: INT
  CompPct: FLOAT
  Rec: INT
  RecYds: INT
  RecTDs: INT
  RshYds: INT
  RshAtt: INT
  RshTDs: INT
  PRIMARY KEY (PlayerID, Year)
)

Articles(
  ArticleID: INT [PK]
  Headlines: VARCHAR
  UserID: VARCHAR [FK to Users.Username]
  NumUpvotes: INT CHECK (NumUpvotes >= 0)
  NumDownvotes: INT CHECK (NumDownvotes >= 0)
)

Comments(
  CommentID: INT [PK]

```
  ArticleID: INT [FK to Articles.ArticleID]
  UserID: INT [FK to Users.UserID]
  Text: VARCHAR
)

Team(
  TeamID: INT [PK]
  TeamName: VARCHAR
  TeamStrength: FLOAT
)

Favorites(
  UserID: INT [FK to Users.UserID]
  PlayerID:  INT [FK to Players.PlayerID]
)

PlayerNews(
  PlayerID: INT [FK to Players.PlayerID]
  ArticleID: INT [FK to Articles.ArticleID]
)
```