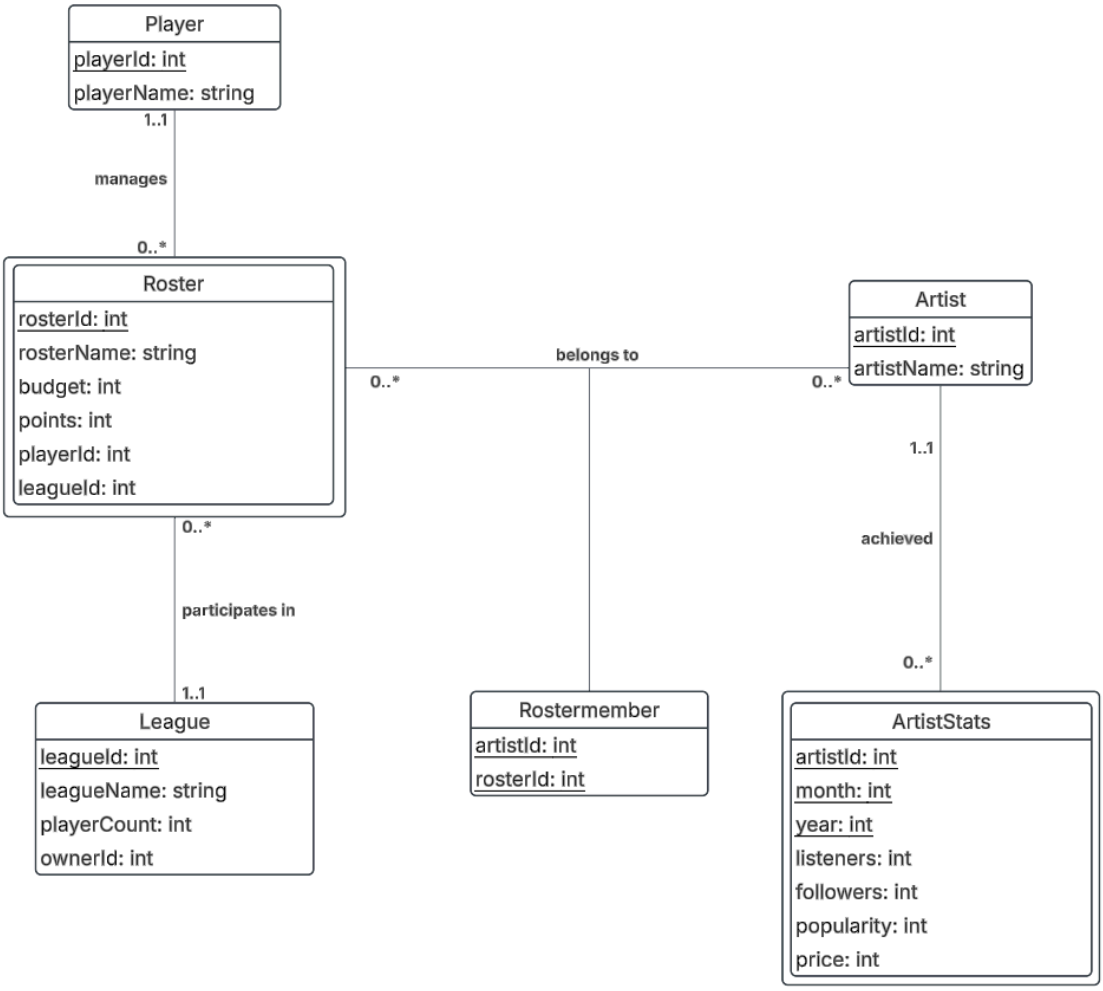
UML Diagram



Assumptions of Entities/Relationships

Entity

- Player:
    - Assumption: A player represents an individual user in the platform who participates in the game. Each player has a playerName and an unique playerID.
    - Reason : The player is modeled as an independent entity because it has distinct attributes and is responsible to manage the Roster. It is not part of another entity, as it is a central role in the system.
- Roster:
    - Assumption: Each player manages a roster. A roster has a rosterName and an unique rosterID, along with attributes: budget, points, playerID(the owner) and leagueID(the league it belongs to)
    - Reason: The roster is a separate entity because it acts as a bridge between Player, League, and Artist. Each Roster is linked to a League,

containing multiple Artists, while roster-specific attributes like budget and points ensure proper data organization.

- League:
    - Assumption: The platform allows multiple leagues, each has a unique leagueID, along with attributes leagueName,a playerCount (indicating the number of participants) and an owner ID.
    - Reason: It is an independent entity because multiple rosters participate in it, and it has its own attribute. If it is stored within the player or roster entity, it will have redundant data.
- Artist:
    - Assumption: Artists are selected by players for their rosters. Each artist has an artistName and a unique artistID.
    - Reason: Artists are not just attributes of a roster, since they exist independently and have their own characteristics. Also, they can be associated with multiple rosters. Model it as an independent entity allows flexibility in our game's design.
- ArtistStats
    - Assumption: each artist has their performances recorded(represented by listeners, followers, popularity), and has their price, which varies from month to month.
    - Reason: It needs to be tracked over time. Thus, if it is within an Artist entity, it would lead to redundant storage issues when recording multiple performances for the same artist. By creating a separate entity, we can track its history easier.
- RosterMember:
    - Assumption: The RosterMember entity represents relation between Roster and Artists, which specify which artists are included in a player's roster.
    - Reason: Since a roster can have multiple artist and an artist can be part of multiple rosters (many-to-many), it require a separate join entity.

Relationship

- Player-manages-Roster(one to many, 1..1 to 0..*)
    - A Player (1) can manage multiple Rosters (0..*), but each Roster must belong to exactly one Player (1).
- Roster-participates in-League(many to one, 0..* to 1..1)
    - Each Roster belongs to one League, while a League can have multiple Rosters.
- RosterMember- links- Roster and Artist (0..* to 0..*, many to many)

    - Each Roster can have multiple Artists, and each Artist can belong to multiple Rosters, forming a many-to-many relationship managed by RosterMember.
    - A Roster contains zero or more RosterMembers, and each RosterMember links to one Roster.（Roster to RosterMember is one to many)
    - Each Artist can appear in multiple RosterMembers, and each RosterMember is linked to exactly one Artist.(Artist to RosterMember is one to many)
- Artists-achieved-ArtistStats(1 to 0..*, one to many)
    - Each artist has zero or more ArtitsStats, but each ArtistStats belongs to one artist.

Normalization

Functional Dependencies

- playerId -> playerName
- rosterId -> rosterName, budget, points, playerId, leagueId
- leagueId -> leagueName, playerCount, ownerId
- artistId -> artistName
- artistId, month, year -> listeners, followers, popularity, price

It could be seen that the functional dependencies that exist are ones from primary keys to all other attributes. Therefore, our schema already adheres to 3NF.

Schema

League(leagueId: INT [PK], leagueName: VARCHAR(20), playerCount: INT, ownerId: INT)
Roster(rosterID: INT [PK], rosterName: VARCHAR(20), budget: INT, points: INT, playerId: INT [FK], leagueId: INT [FK])
Player(playerId: INT [PK], playerName: VARCHAR(20))
RosterMember(artistId: INT [PK] [FK], rosterId: INT [PK] [FK])
ArtistStats(artistId: INT [PK] [FK], month: INT [PK], year: INT [PK], listeners: INT, followers: INT, popularity: INT, price: INT)
Artist(artistId: INT [PK], artistName: VARCHAR(20))