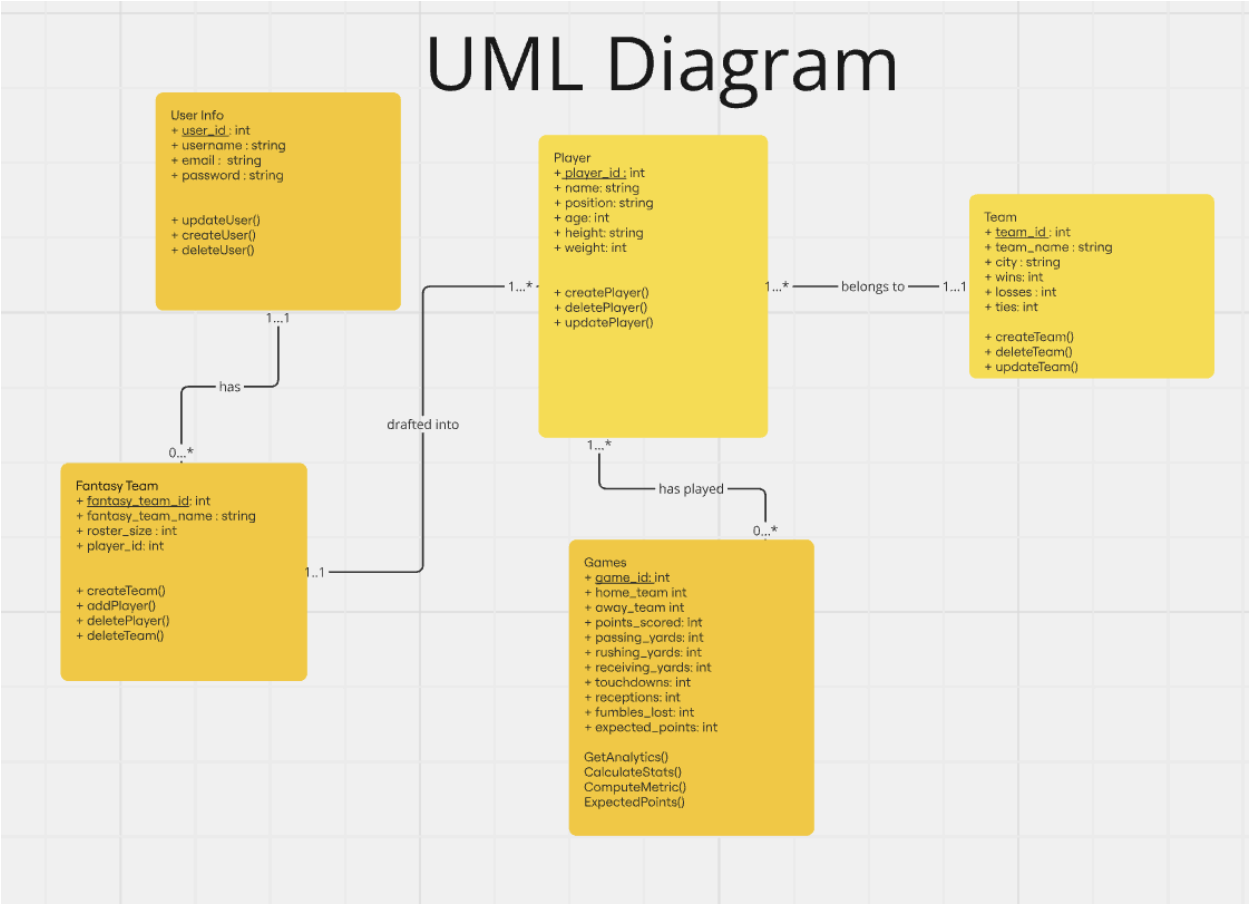
Stage 2 Database Design

1 Draw ER/EML Diagram



2 Assumptions

Our project consists of 5 tables: User, FantasyTeam, Games, Player, and Team. We have Many to One relations between User and Fantasy Team, Player to Team, and Player to Fantasy Team. User to Fantasy Team is to ensure that one user can have multiple fantasy teams. Player to Team is so that one team can store many players. Player to Fantasy Team is to ensure that one player cannot appear on multiple teams from a User. We also have Many to Many relationships between Player and Games. This is to allow games to have multiple players and to allow players to play in multiple games.

The reason we have Games store our statistics and not just apply them to every player is to help do future modeling to allow more flexibility and variance in our predictions for player outcomes. Having player performance split up between games helps us understand more information about how that player performs week to week and not just have all their aggregated stats as a whole.

4 Normalizing database

We begin finding all the functional dependencies and then creating a mapping with a letter to simplify the calculation

Player.team_id = A
Team.team_id = D
FantasyTeam.user_id = B
Userinfo.user_id = E
Game.game_id = C
FantasyTeam.fantasy_team_id = F

R (A, B, C, D, E, F)
FD = {A ->BCD; E -> BF}

Finding the candidate key:

L	M	R	N
AE		BCDF	

Candidate key = *{AE}

Normalizing into relations

1. Making the right side singleton
    a. FD = {A -> B; A -> C; A -> D; E -> B; E -> F}
2. Getting rid of unnecessary LHS
    a. FD = {A -> B; A -> C; A -> D; E -> B; E -> F}
3. Getting rid of unnecessary dependencies
    a. FD = {A -> B; A -> C; A -> D; E -> B; E -> F}

Final FD = {A -> B; A -> C; A -> D; E -> B; E -> F}
Relations: X(A,B,C,D); Y(E,B,F); Z(A,E)

After we have obtained the functional dependencies we calculate the 3nf and found our database to already be in 3nf form. Therefore, our schema is 3nf.

5 Converting to conceptual database design

UserInfo(
user_id: INT [PK],
username: VARCHAR(100),
email: VARCHAR(100),
password: VARCHAR(100)
)
Team(
team_id: INT [PK],
team_name: VARCHAR(100),
city: VARCHAR(100),
wins: INT,
losses: INT,
ties: INT
)
Player(
player_id: INT [PK],
name: VARCHAR(100),
team_id: INT [FK to Team.team_id],
fantasy_team_id: INT [FK to FantasyTeam.fantasy_team_id]
position: VARCHAR(50), age: INT,
height: DECIMAL(5,2),
weight: DECIMAL(5,2),

)
FantasyTeam(
fantasy_team_id: INT [PK],
user_id: INT [FK to UserInfo.user_id],
fantasy_team_name: VARCHAR(100),
roster_size: INT
)
Games(
game_id: INT [PK],
home_team: int
away_team: int
player_id: INT [FK to Player.player_id],
points_scored: INT,
passing_yards: INT,
rushing_yards: INT,
receiving_yards: INT,
touchdowns: INT,
receptions: INT,
fumbles_lost: INT,

expected_points: INT
)

Fix for suggestions from previous stage from canvas:

“The project is interesting. I think you may want to give more details about the creative component. Currently, the description for the creative component is not enough and I cannot understand your plan to implement this component, like how you will analyze the player performance and what data and technical methods you will use to support the analysis. You also need to point out the detailed technically challenging points to implement this component.”

More details for creative component:

For the creative component we want to provide performance metrics and ranking players. We wanted to use linear regression to create models to show the user how their player is expected to perform in future games based on their past performance, and use nearest neighbor technique learned in cs 441 to find players who have similar attributes which a user could use to make informed decisions in their draft. The way we will quantify performance is based on their prior performance using linear regression and computing totals on their stats from the Player Table and populating it inside the Analytics table.