1. Motivation for the idea/description of the problem the application solves
    - League of Legends is a competitive 5v5 video game where players form teams of five and battle against each other, aiming to destroy the Nexus, a core structure located within the enemy's base. Over the past decade, the game has surged in popularity, amassing millions of players worldwide. With a diverse pool of 164 champions to choose from in each match, players can create over 3 quadrillion unique combinations of 10 champions.
    - Victory hinges not only on the choice of champions but also on the synergy between them and the strategic selection of items. This has led players to ponder the ideal team composition that ensures a high win rate. However, the task is complex, as some champions demand a higher skill level than others. Consequently, a team composition that excels in higher ranks may underperform among lower-ranked players, and vice versa.
    - Our ultimate goal is to develop a user-friendly game simulator that leverages extensive data from past matches. By inputting a selected champion, players can query historical data to receive recommendations on optimal team compositions and item choices that maximize the chances of winning. Additionally, the simulator can display a list of enemy champions that have historically diminished the win rate of the selected champion, aiding players in making strategic bans before the picking phase.

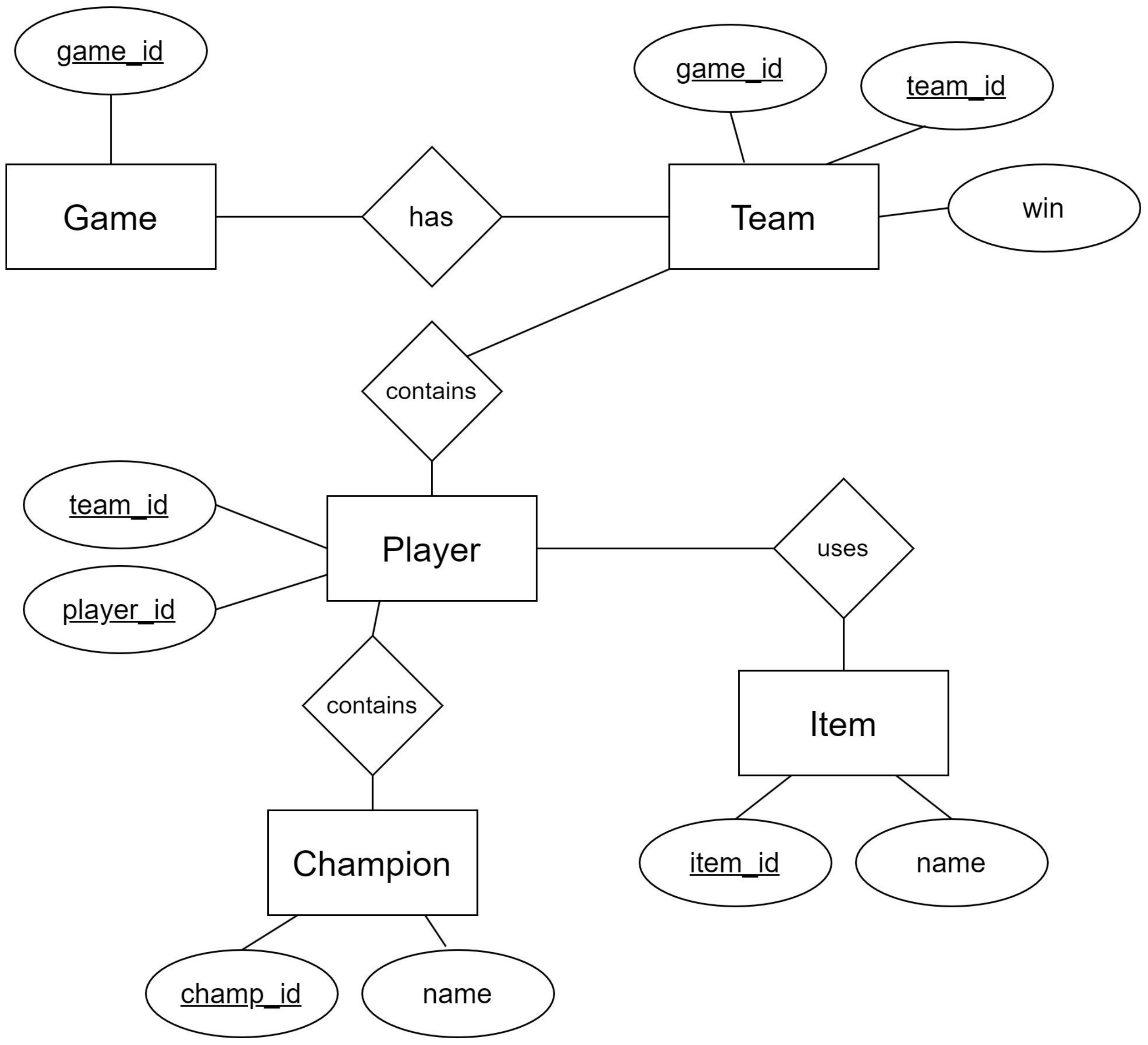
2. List of features you will definitely implement in the application
    - Upon selecting a champion, receive recommendations for a team composition that ensures the highest win rate.
    - Get suggestions for an opponent team composition that has historically exhibited the lowest win rate, aiding players during the ban and pick phase.
    - Obtain recommendations for the optimal item build for the chosen champion.

3. List of features you might implement in the application, given enough time
    - Receive general strategic guidance, including recommendations on the optimal timing for securing neutral objectives (such as Baron Nashor or Herald) and attacking enemy turrets to maximize your chances of victory.
    - A trivia quiz feature that assesses players' knowledge of champions, items, and various aspects of the game. (As well as a leaderboard)

4. List of pages the application will have and a 1-2 sentence description of each page.
We expect that the functionality of each page will be meaningfully different from the functionality of the other pages.
    - Main page
        - Contains a brief description of the main functionality, as well as buttons leading to each page
    - Ban and pick simulator
        - Creates a ban pick simulation based on the in-game ban pick simulation, with percentages of winning being updated for each side as the ban pick simulation progresses

- Detailed Champion stats, item simulator
    - Shows the winrate, pick-rate, and other interesting statistics about champions, and shows the winrate for champions based on their items they choose

5. Relational schema as an ER diagram



6. SQL DDL for creating the database

Game(game_id, gameCreation, gameDuration, gameId, gameMode, gameType, gameVersion, mapId, participantIdentities, participants, platformId, queueId, seasonId, status_message, status_status_code)

Team(game_id, team_id, win, firstBlood, firstTower, firstInhibitor, firstBaron, firstDragon, firstRiftHerald, towerKills, inhibitorKills, baronKills, dragonKills, vilemawKills, riftHeraldKills ,dominionVictoryScore, bans)

Player(team_id, player_id, championId, spellId, highestAchievedSeasonTier, win, item_id, kills, deaths, assists, largestKillingSpree, largestMultiKill, killingSprees, longestTimeSpentLiving, doubleKills, totalDamageDealt, magicDamageDealt, physicalDamageDealt, trueDamageDealt, largestCriticalStrike, totalDamageDealtToChampions, magicDamageDealtToChampions, physicalDamageDealtToChampions, trueDamageDealtToChampions, totalHeal, totalUnitsHealed, damageSelfMitigated, damageDealtToObjectives, damageDealtToTurrets, visionScore,role, lane

Champion(champion_id, name)

Item(item_id, name)

7. Explanation of how you will clean and pre-process the data. This tutorial demonstrates how to do simple pre-processing in Python.

- There are some columns where the data is stored in a JSON format. Since the data isn't atomic we need to reformat the JSON data into a valid csv format
- In terms of cleaning data, when we encounter null values in a column, we will remove the entirety of the game that that column is associated with
- We will also use pandas to preprocess the data in Kaggle so that it fits the format we want in our SQL tables, and then download that data from Kaggle into a local folder from a csv, and then upload that csv to our AWS server

8. List of technologies you will use. You must use some kind of SQL database. We recommend using MySQL or Oracle specifically because you will use MySQL in HW2, and we will provide guidance for setting up a MySQL database. Some groups in the past have had issues with MySQL, but Oracle is another option.

- We will be using MySQL for the construction of our database.
- We will be using Python and Pandas for the cleaning and pre-processing of the csv datasets.

9. Description of what each group member will be responsible for
Izzy: React.js frontend and web app design
Matthew: Node.js backend + routing/queries/backend operations
Jinseo: SQL queries + frontend
Hwisang: AWS setup, SQL queries, and integration with cloud