Jylah Bah, Bruk Abu, Mouhamadou Sissoko

#### Provide a README section for creating and running the project. We need complete specifications for building your project on our computer. Specify all libraries, software, etc. needed to run the application. Specify expected installation directories. If you use a specific technology for the project, the technology’s download page must be listed.

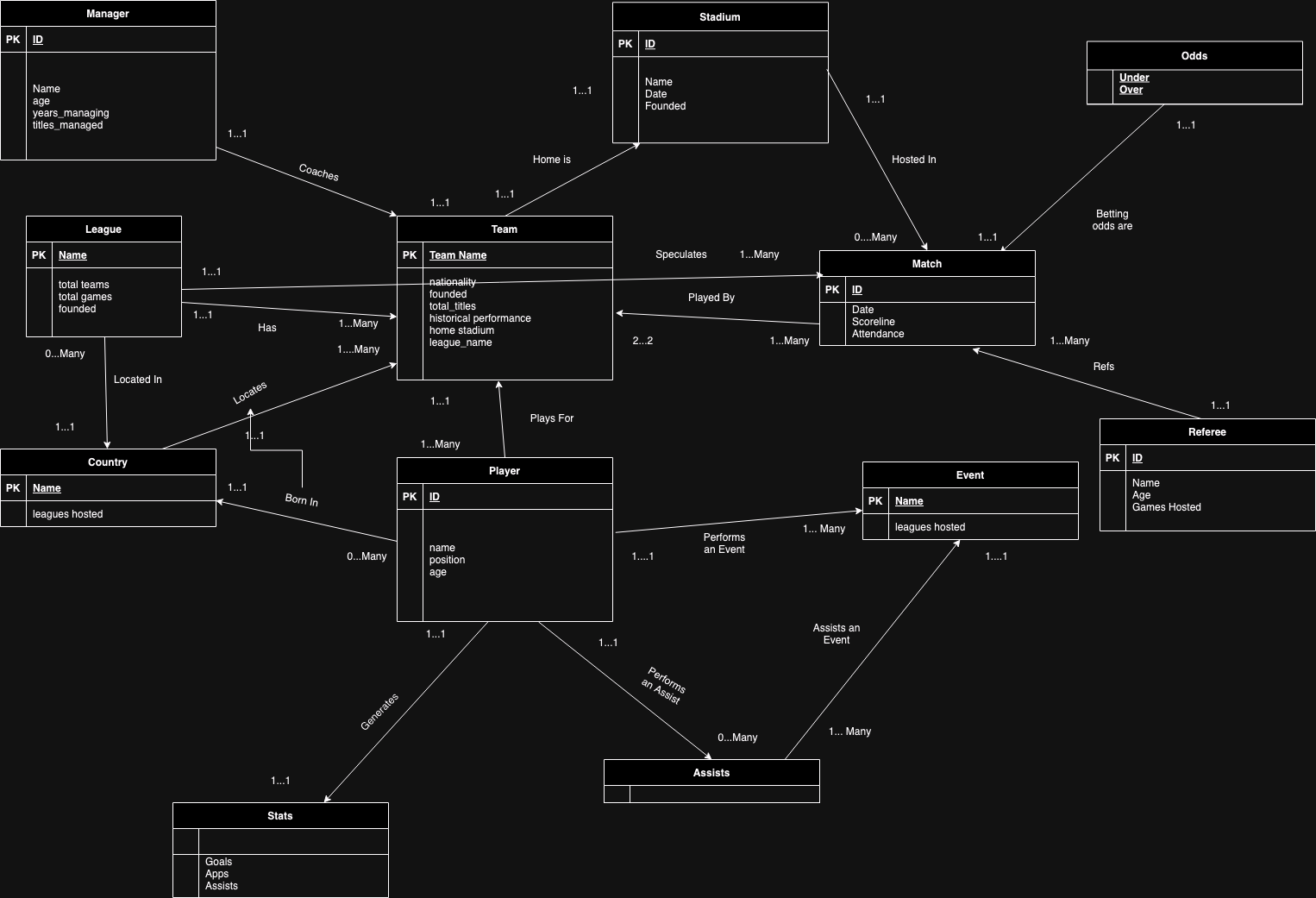
* 1. In GitHub

#### Provide the Technical Specifications for the project.

This project is developed using Python as the programming language, with Flask as the web framework. The data is stored in a MySQL database, and SQLAlchemy is used as the Object-Relational Mapping (ORM) tool to interact with the database. The frontend / templates for the web pages are created using Jinja2, a modern and designer-friendly templating language for Python. We also use bootstrap for some of the frontend element styling.

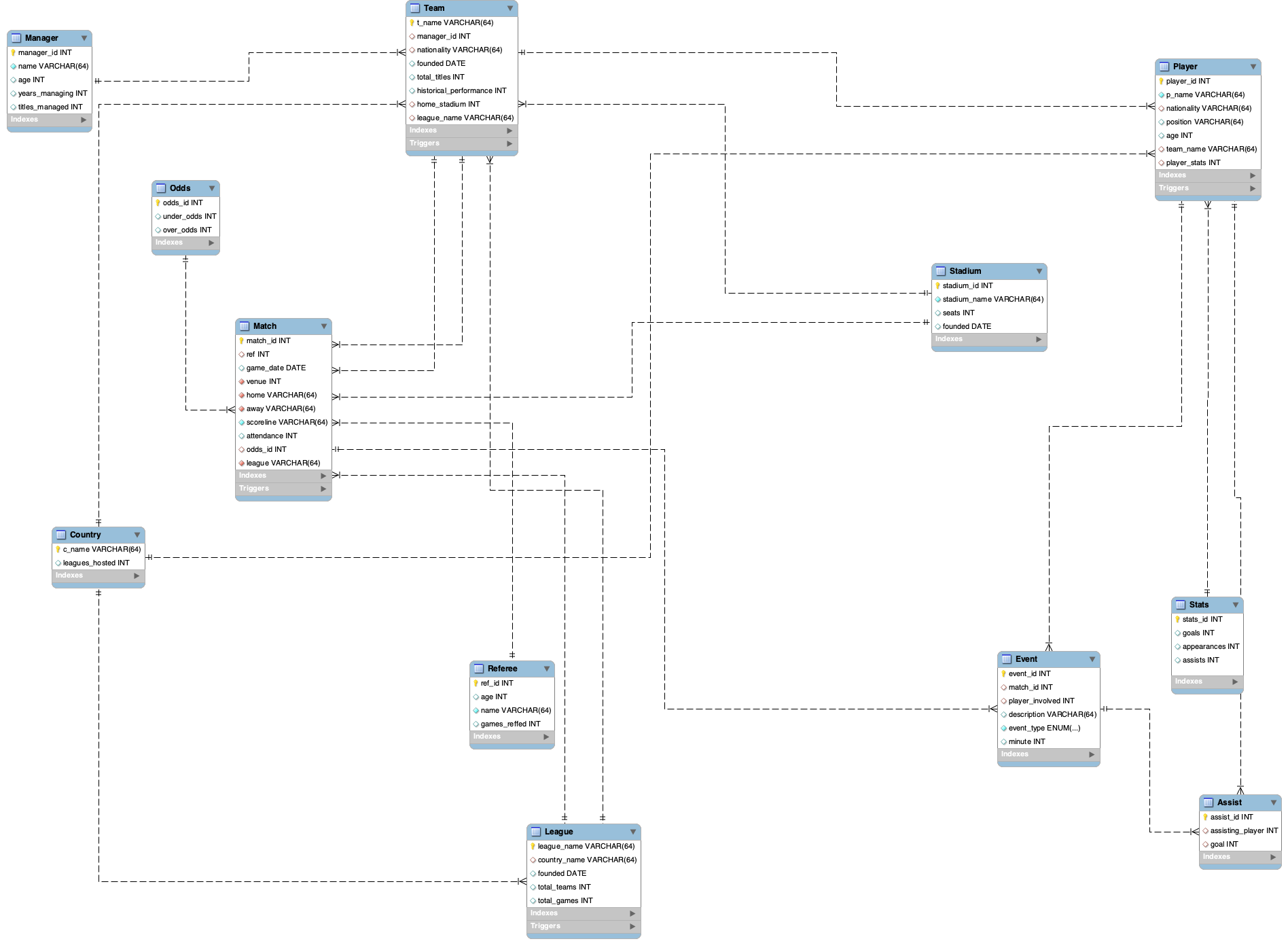
#### Provide a textual description of the current conceptual design as well as a UML diagram for the conceptual design of the database .

* 1. **Our conceptual design is based on the English Premier League.**
     1. We take in all of the factors that leagues would need to keep data on and reconstruct it into a mysql database. Attached below is a per table description of the database..
  2. **Country**
     1. Country Name
        1. Primary Key
     2. Keeps track of how many leagues are hosted.
        1. This is important as most soccer playing countries have multiple divisions and sometimes add more.
        2. Is updated by a trigger
  3. **Referee**
     1. Referee ID
        1. Every referee has a unique ID to identify them
        2. Primary Key
     2. The AGE of the referee
        1. Age is a factor to consider when picking a referee.
     3. Name
     4. Games Reffed
        1. The Number of Games a Referee has hosted. Is updated by a trigger.
  4. **Stadium**
     1. Stadium ID
        1. Every Stadium has a unique ID
        2. Primary Key
     2. Stadium name
     3. Maximum seating capacity
     4. Founding date.
  5. **Manager**
     1. Manager ID
        1. Every Manager has a unique ID
        2. Primary Key
     2. Manager Name
        1. Not Null
     3. Age
     4. Years managing
     5. Titles Managed
        1. Important for teams to consider if they want a new Coach
        2. Updated by a Trigger
  6. **League**
     1. League Name
        1. PK
     2. Founding Date
     3. Total Teams
        1. How many teams are in the League
        2. Updated by Trigger
     4. Total Games
        1. How many games have been played in this league
     5. Host Country
        1. Foreign Key
  7. **Team**
     1. Team Name
        1. PK
     2. Managed ID
        1. Foreign Key
     3. Nationality
        1. Foreign Key
     4. Home Stadium
        1. Foreign Key
     5. League
        1. Foreign Key
     6. Founding Date
     7. Total Titles
     8. Historical Performance
  8. **Odds**
     1. ID
        1. Primary Key
        2. Every ODDS has it’s unique ID
        3. When a match is deleted from the database it’s associated odds are deleted too.
     2. Over
        1. Betting Over
     3. Under
        1. Betting Under
  9. **Match**
     1. Match ID
        1. Primary Key
        2. Every Match has a unique ID
     2. Referee
        1. Foreign Key
     3. Game Date
     4. Venue
        1. Foreign Key
     5. Home Team
        1. Foreign Key
     6. Away Team
        1. Foreign Key
     7. Scoreline
     8. Attendance
     9. Odds
        1. Foreign Key
  10. **Stats**
      1. Stats ID
         1. Primary Key
         2. All Stats have a unique ID
      2. Goals
      3. Appearances
      4. Assists
  11. **Player**
      1. Player ID
         1. Primary Key
         2. All players have a unique ID
      2. Player Name
      3. Nationality
         1. Foreign Key
      4. Position
      5. Age
      6. Team
         1. Foreign Key
      7. Stats
         1. Foreign Key
  12. **Event**
      1. Event ID
         1. Primary Key
         2. All events have a unique ID
      2. Event Enum
         1. Goals
         2. Cards
         3. Substitutions.
      3. Match
         1. Foreign Key
      4. Involved Player
         1. Foreign Key
  13. **Assist**
      1. Assist ID
         1. Primary Key
         2. All assists have a unique ID
      2. Assisting Player
         1. Foreign Key
      3. Goal
         1. Foreign Key
            1. Event



#### 

#### Provide a logical design for the submitted database schema (Feel free to reverse Engineer your final schema in the MySQL workbench).



#### 

#### Provide the final user flow of the system. List the commands or methods the user performs to interact with the system.

1. Accessing the Application

- Install all the dependencies: pip install -r requirements.txt

- run the command: flask run.

- The website should be served on your localhost on port 5000 (Default port for flask).

2. Logging In

- Upon accessing the application, the user will be prompted to enter their username and password for database authentication.

3. Navigating through Navigation Bar Options

- After successful login, the user will be directed to the homepage (Home) of the application.

- The navigation bar will display 11 options: Home, Countries, Managers, Referees, Stadiums, Teams, Matches, Players, Events, Stadiums, and League.

4. Viewing Countries

- Click on Countries in the navigation bar.

- View a list of all countries stored in the database.

- Click on the eye icon next to a country to view specific details.

- Edit information regarding the country by clicking on the pencil icon next to the manager

5. Managing Managers

- Click on Managers in the navigation bar.

- Add a new manager by filling out the manager details form.

- View a list of all managers added so far.

- Click on a manager's name to view detailed information.

- Edit information regarding the manager by clicking on the pencil icon next to the manager

- Can delete managers that don't have a team by clicking on the trash icon

6. Handling Referees

- Click on Referees in the navigation bar.

- Add new referees using the provided form.

- View a list of all referees stored in the database.

- Click on a referee's name to view detailed information.

- Edit information regarding the referees by clicking on the pencil icon next to the referees

- Can delete referees by clicking on the trash icon

7. Managing Stadiums

- Click on Fields in the navigation bar.

- Add new stadiums used for matches.

- View a list of all available fields.

- Click on the eye icon next to a stadium to view detailed information.

- Edit information regarding the stadium by clicking on the pencil icon next to the stadium

8. Managing Teams

- Click on Teams in the navigation bar.

- Add new sports teams using the team details form.

- View a list of all teams available.

- Click on a team's name to view detailed information.

- Edit information regarding the team by clicking on the pencil icon next to the team

9. Handling Matches

- Click on Matches in the navigation bar.

- Add new matches by entering match details.

- View a list of all scheduled matches.

- Click on a match to view detailed information

- Edit information regarding the match by clicking on the pencil icon next to the match

- Can delete matches that dont have an event or assist

10. Managing Players

- Click on Players in the navigation bar.

- Add new players to the database.

- View a list of all players.

- Click on a player's name to view detailed player information.

- Edit information regarding the player by clicking on the pencil icon next to the player

- Can delete player if it isn't tied to any events

11. Handling Events

- Click on Events in the navigation bar.

- Add new events by entering event details.

- View a list of all upcoming events.

- Click on an event to view detailed information.

- Edit information regarding the events by clicking on the pencil icon next to the event

- Can delete event by clicking trash icon

12. Managing Leagues

- Click on League in the navigation bar.

- Add new leagues by entering league details.

- View a list of all leagues stored in the database.

- Click on the eye icon next to a league next to the player to view detailed information.

- Edit information regarding the league by clicking on the pencil icon next to the league

- Can delete if it has no ties to other objects

13. Data extraction from a CSV

Enter the command:flask seed\_csv\_data

#### Provide a description of any code not working in this section, document any bugs you have encountered.

We failed to provide a trigger that could properly handle when a user has scored a goal, because the way we handle it is through an EVENT type which could either be a Goal, Card, or substitution. While we have many robust trigger, this is a glaring flaw that we struggled to implement due to our design.

#### Provide a “Lessons Learned” section

Throughout this project, we learned a lot about not only database design but also the proper approach to tackle this. We made the common mistake of only starting the project once the deadline was approaching, so we were very pressed for time. The time we had was marred with other projects so we had to be very smart on how we approached this. We decided to split up the project into three different portions, using a system very similar to the MVC pattern where one of us worked on the SQL, one worked on the back end, and the other on the front end. It is through this we could maximize our time and also our project. Lots of careful consideration came into the SQL database design, while tables are fairly straightforward after this class, when trying to mimic a premier league’s database triggers are VERY important. This is where we learned to pay close attention to the relationships we established and why it’s EXTREMELY important to have proper constraints. Many times our triggers would fail because we had the wrong constraint on them. Also, we learned it's incredibly important to make sure you have an established SQL before working on the back end for a program. Due to changes needing to be made, there were many times where we would have to adjust the back end as the SQL had to either be arranged or fields disappeared entirely. In the front end, we learned how to use Jinja2, a templating language for Python. Jinja2 is used in conjunction with the Flask web framework to dynamically generate HTML content. Finally, we learned how to include other templates in a template with the tag. This is used to include the 'Components/side\_table.html' template in the current template. This promotes modularity and reusability, as components can be defined once and included in multiple places. In the provided HTML excerpt, we can see the use of Jinja2's template inheritance feature**.** In the backend development process, we discovered the power of robust tools like SQLAlchemy for establishing efficient connections to SQL databases. SQLAlchemy not only simplifies database interactions but also provides an intuitive and powerful ORM (Object-Relational Mapping) framework, enabling seamless integration of database operations within our Flask application. Leveraging SQLAlchemy's capabilities, we were able to enhance productivity, optimize performance, and maintain scalability, ensuring a solid foundation for our database-driven application. While we did struggle, there was very little worry about the way we approached the project as we did extensive research on how other sports databases were established.

#### Provide a “Future work” section

In the future we plan to use our database to provide a more robust representation of the football world, which can handle multiple leagues and the teams that are attached. This would require for our database to be hosted on a web server so that anyone can freely use and interact with the webpage, which would require additional work on our end. This would additionally involve for us to protect our database because as it stands we do not have any prepared statements which makes the code vulnerable to such things as SQL injections through our front end. Additionally, we would like for the page to be as autonomous as people, this would require us to call upon an API which monitors the Premier League and whenever new data is handled our database handles it instantly and properly, through our use of triggers to properly handle any CRUD. We would also like to make it possible to differentiate teams and leagues by seasons, as this would greatly help users who want to get data from a certain version of the team rather than the current one. In the future, we also plan to add proper and interesting visualization.