Information system for playing different sports with strangers

(SportIsLife)

Lucas Jakin

Definition of the problem

Team sports are very advantageous for both athletes and nonathletes, they promote cooperation, good sportsmanship and help to put winning into perspective. Team sports teach you to appreciate the value of your teammates abilities and how they can contribute to a shared goal. Ultimately they are a great way to make friends and meet new people. Sports like football, basketball, volleyball can be easily played outdoors on purposely built surfaces.

Since these are all team sports, there is a minimal number of players that are needed for playing them in a successful way. Sometimes when you want to go outside in the fresh air to play some basketball, football, etc., it can happen that your friends might be busy, so you need to find other people to play with. Besides that you can meet the problem of not finding a place where to play, for example on finding a sports surface (basketball court, football pitch…) in a location you have never been before.

As a project in the course Systems III I want to develop a system that allows people to find sports surfaces near their location and match up with strangers for some quick matches/games.

Functional and nonfunctional requirements of new system

A functional requirement is a function or feature that must be included in an information system in order to satisfy the business need and be acceptable to the users.

A nonfunctional requirement is a description of the features, characteristics, and attributes of the system as well as any constraints that may limit the boundaries of the proposed solution.

Functional requirements

The system should enable the following functionalities:

1. The system should allow the users to create a personal profile in which their personal information can be stored. While creating a new account the user must insert basic information about him, such as name, email, username, password.

2. The system allows the user to properly log in if this already has an account. If the user is not logged in, then he will not be able to access any system service such as accessing teams, creating teams, creating surfaces etc.

3. The user must also hand in more information about himself by describing his skills in a certain sport, so that the users with the most similar skills can be grouped together and play a more fair game. Besides that the system also stores the user's current location.

4. The system offers the same services for different kinds of sports that can be played on an outside surface. The user should be allowed to select the sport that he wants to play. The possible sports that the user can choose are basketball, football, volleyball. Everytime they select a sport, the surfaces of the selected sport will be posted on the screen. The user can prioritize one sport.

5. While using the application the user shares his live location, so that the system knows where the user is located. With this information the system allows the user to check if there are courts, fields.. (depends on

selected sport) or some places to play near his location. Besides that the user can see if the court is free or if there are other people already playing.

6. The system allows the user to match up with other players, so that they can create teams. Every time a new team is created, it will be visible to all users and it can be entered by everyone. Each team has a maximum number of players. (15)

7. A user can also insert new courts or fields that are missing in the application. The newly inserted sport surface should exist and be playable.

8. Teams can organize matches and they can play against each other in whatever sport they want to. A specific match is played on a specific sport surface and it can be seen by other users, so that they know the surface is occupied.

Non-functional requirements

1. The system must be able to operate 24/7. Bigger updates will be made during winter time, since less people are willing to play in cold conditions (from November to March).

2. The system must be able to get the user's location. The user needs to give his permission on the system request.

3. The system must ensure a safe communication between users. Exchanged data flows just inside the system.

4. After a certain number of login attempts, the system must lock an account to protect a user's information from potential hackers.

5. External backups should be made every two weeks.

6. The system must be compatible for all operating systems and must offer the same features on all of them.

7. The users should be able to access the system wherever they want using a mobile phone or tablet connected to the internet.

Feasibility study

The solution to the problem is quite feasible, though from a technical point of view, the most technically challenging part of the project is the tracking of the user's current location. This feature is at the core of the system functionalities. Since it will be necessary to track multiple locations at once, an adequate web server unit will be needed to support a good performance. Selecting the right technologies is essential for GPS-based apps development, as for example Google Maps API for both Android and iOS apps development.

The review of legislation revealed there are certain restrictions regarding sharing the user's livetime location. Each user must give his approval for the application to know where he is located during the usage of the service. If the approval is not given, then the user is not able to use the complete service. Acceptance of location-tracking technologies has generally been identified as sufficient in other feasibility studies. All data has to flow in an encrypted way, most importantly the user's personal information.

The proposed information system can be realized as a mobile application suitable on both Android and iOS operating systems. The user is able to use the application for free without paying any subscription charges.

Logical Design

Matrix User role / functions

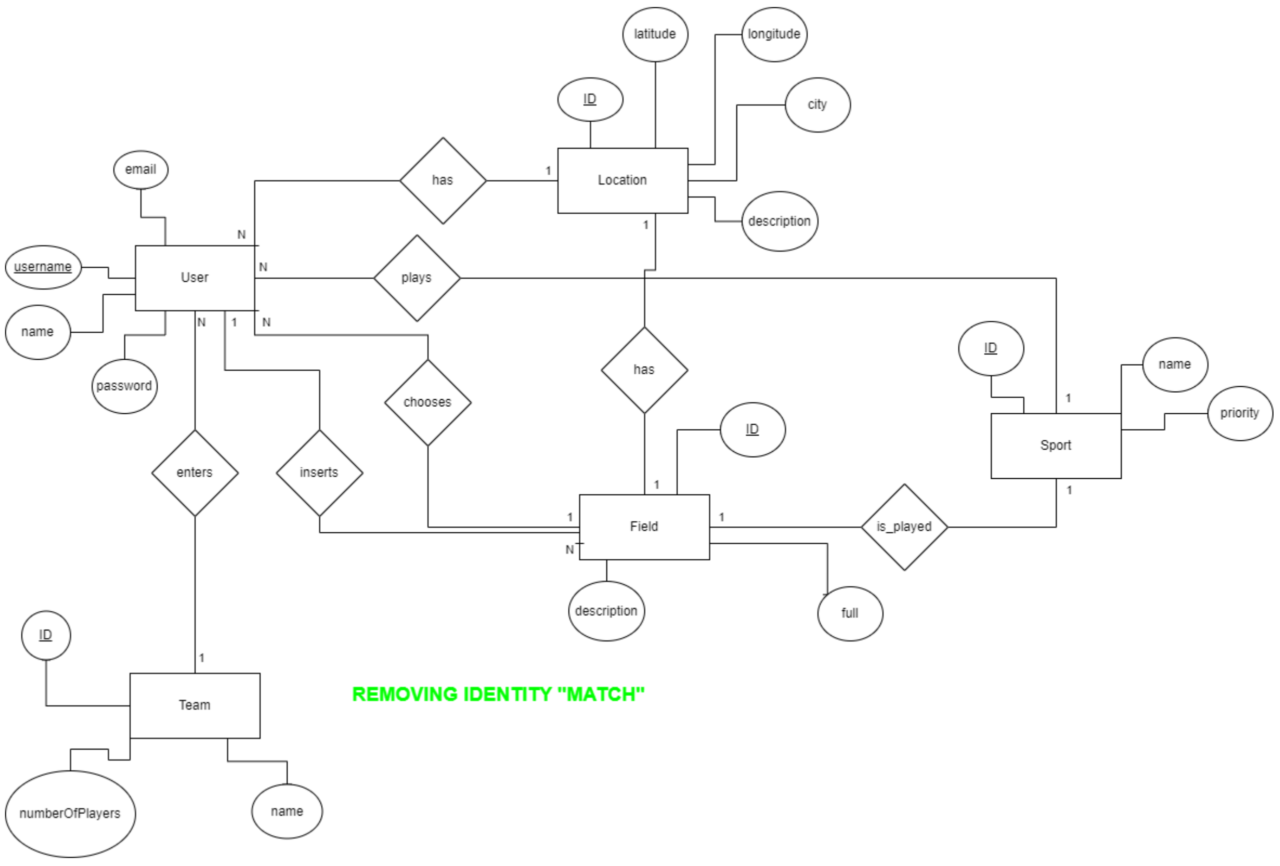
Table 1: Matrix user roles/functions

Functions	Sports Person (User)	Administrator
Account creation	Yes	Yes
Sport selection	Yes	No
Sharing location	Yes	No
Sport surface selection	Yes	No
Team match-up	Yes	No
Field insertion	Yes	Yes
Start match	Yes	No

Data dictionary

Entity	Description	Attribute	Type	Description of attribute
User	User of the system	username	varchar(25)	ID/Username of the user
		name	varchar(25)	Name of the user
		email	varchar(25)	Email of the user
		password	varchar(25)	Password of the user
Sport	Playable sport of the system	ID	int	Identifier of the sport
		name	enum	Type of the sport
		priority	boolean	User's priority on this sport
Field	Sport surfaces where users can play on	ID	int	Identifier of the surface
		full	boolean	State of field/court in sense of accessibility
		description	varchar(100)	Description of the surface
Team	Users of system grouped together	ID	int	Identifier of the team
		numberOfPlayers	int	Number of players on the team
		name	varchar(25)	Name of the team
Location	Current location on the map	ID	int	Identifier of the location
		latitude	varchar(25)	Latitude value of location
		longitude	varchar(25)	Longitude value of location
		city	varchar(25)	Name of the city in which the location points
		description	varchar(100)	Description of the location, where the user/surface is located

Entity relational diagram (ERD)



REMOVING IDENTITY "MATCH"

Figure 1: Entity relationship diagram

Relational model

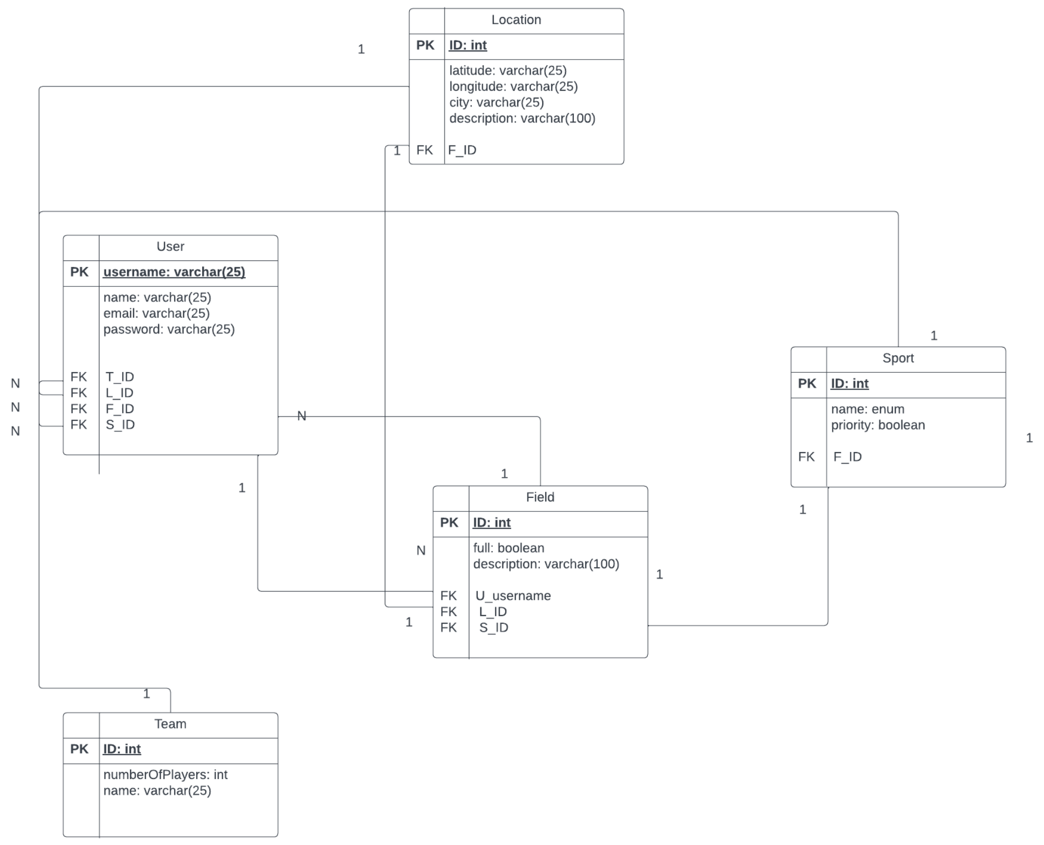


Figure 2: Relational model

“Match” identity has been removed

Process model

Functional decomposition diagram

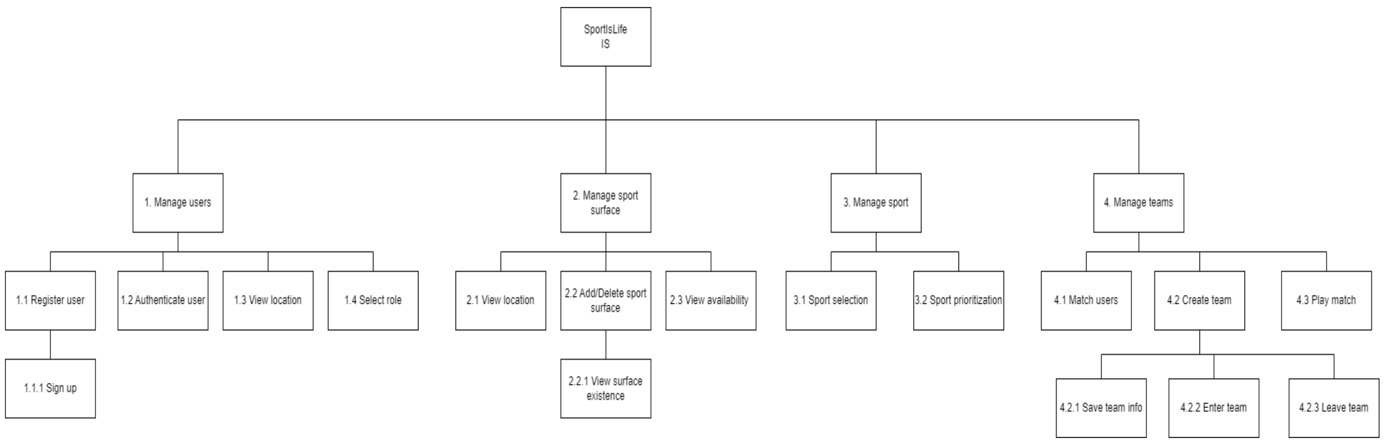


Figure 3: Functional decomposition diagram

Data Flow Diagram

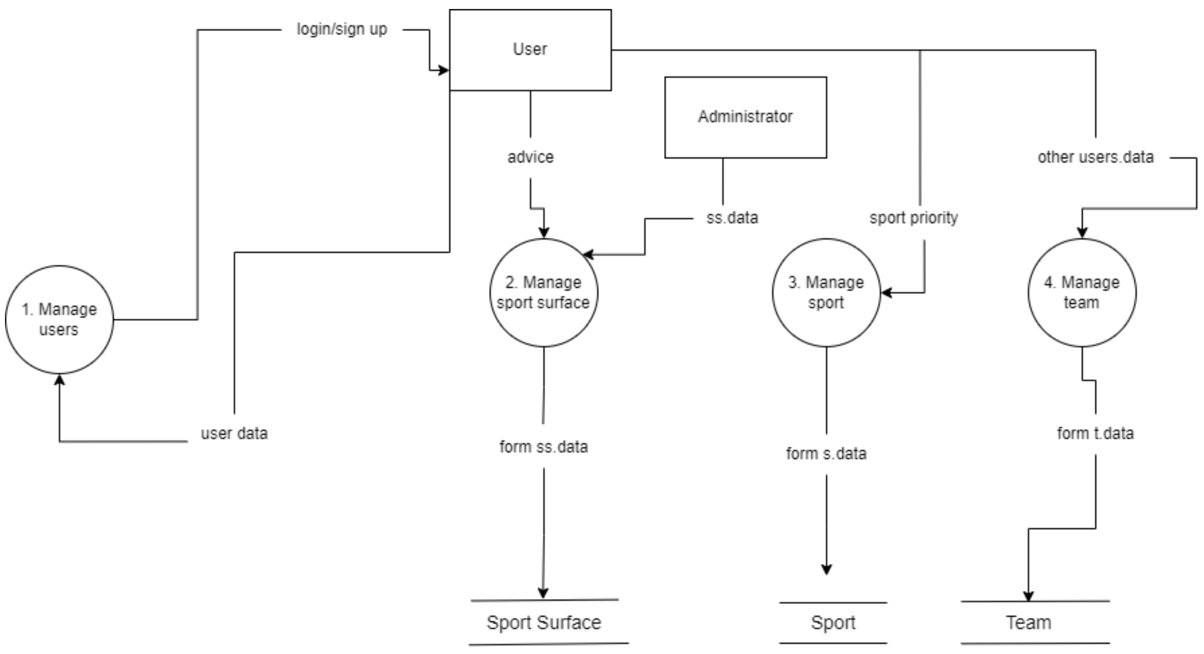


Figure 4: System level data flow diagram

Object-oriented Analysis

UML Class Diagram

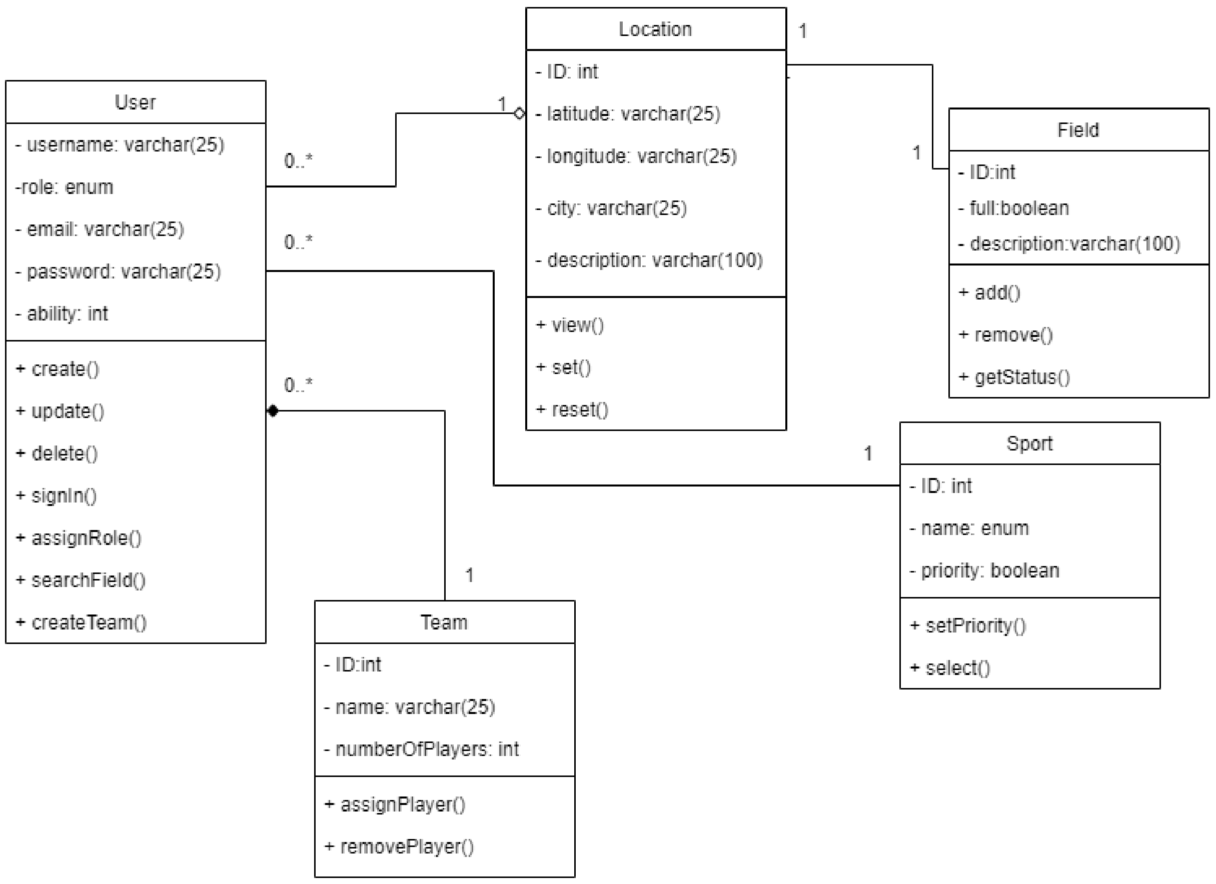


Figure 5: UML Class Diagram

UML Sequence Diagrams

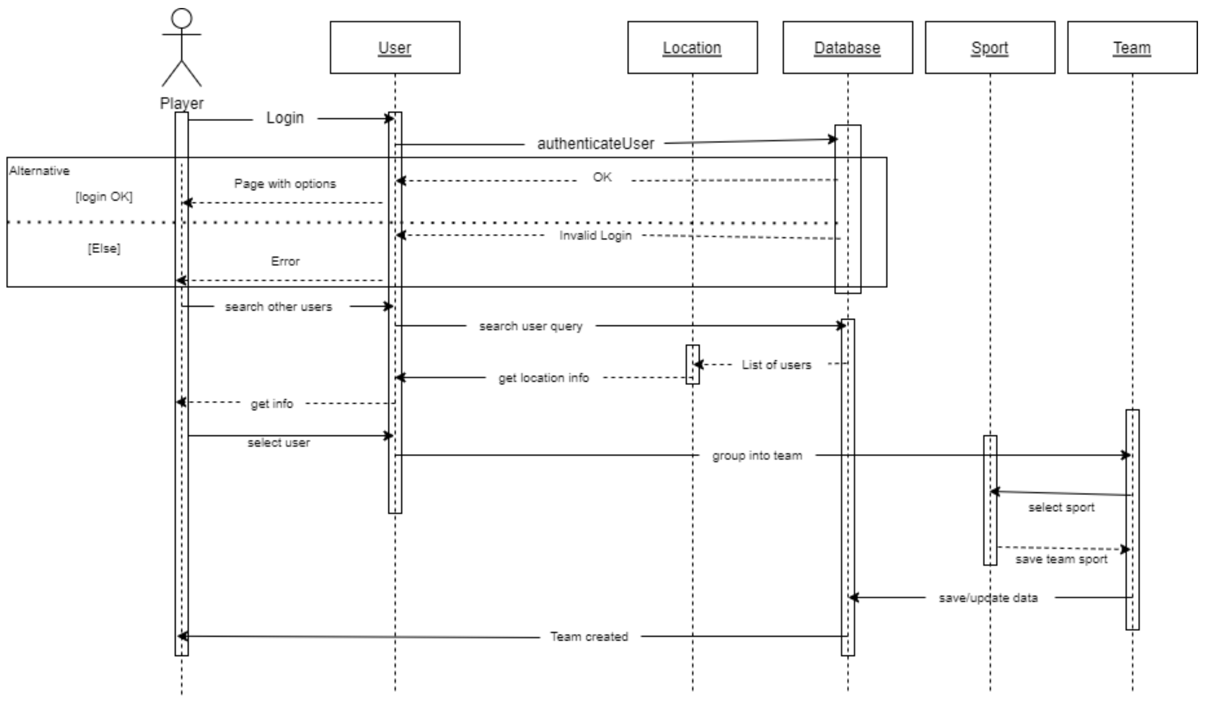


Figure 6: UML Sequence diagram 1 for grouping users into a team

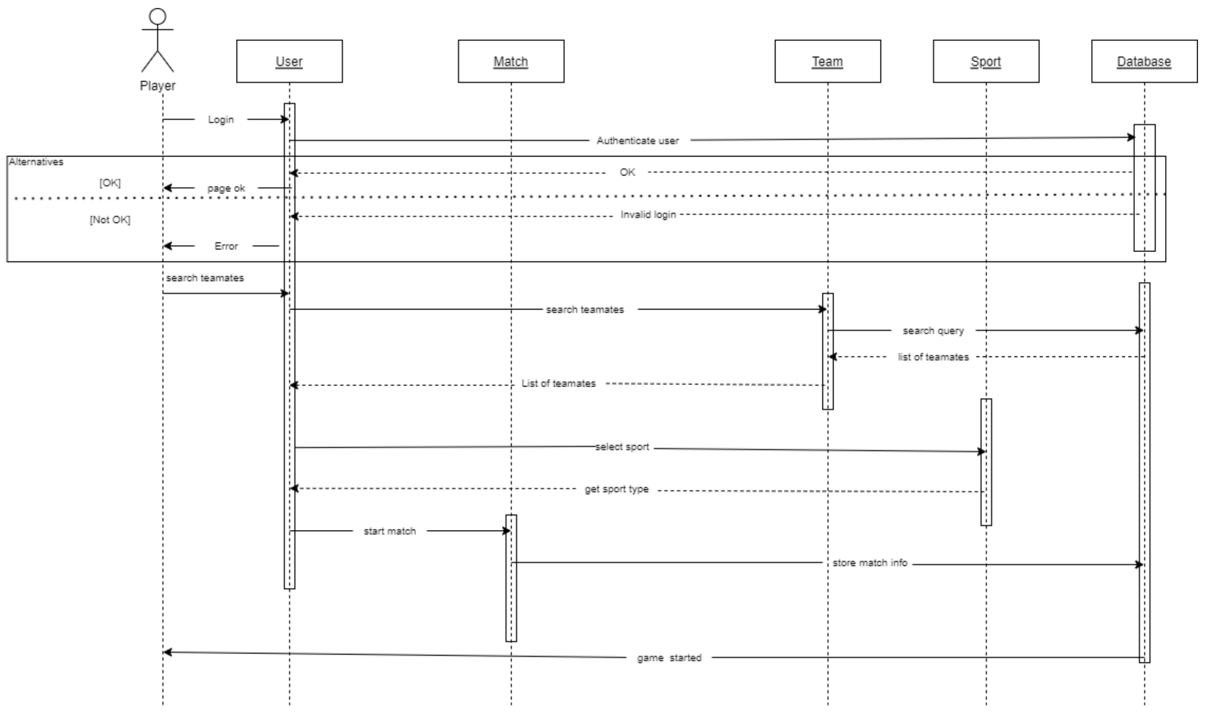


Figure 7: UML Sequence diagram 2 for starting a match

Physical Design Phase

Physical Data Model

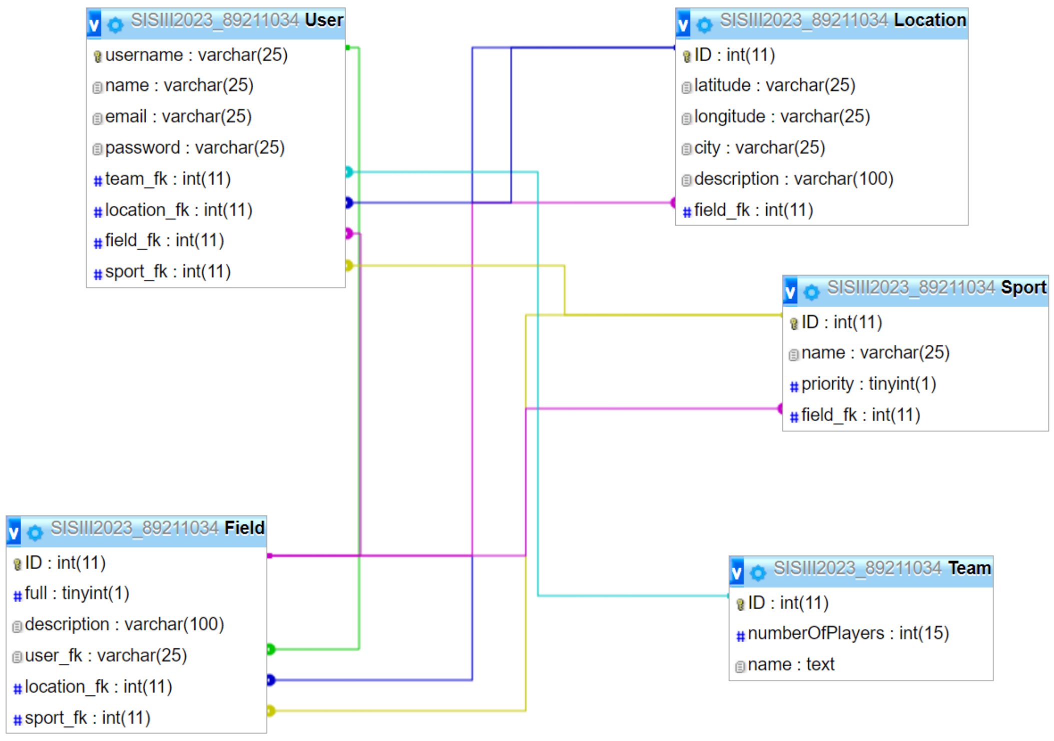


Figure 5: Physical database model