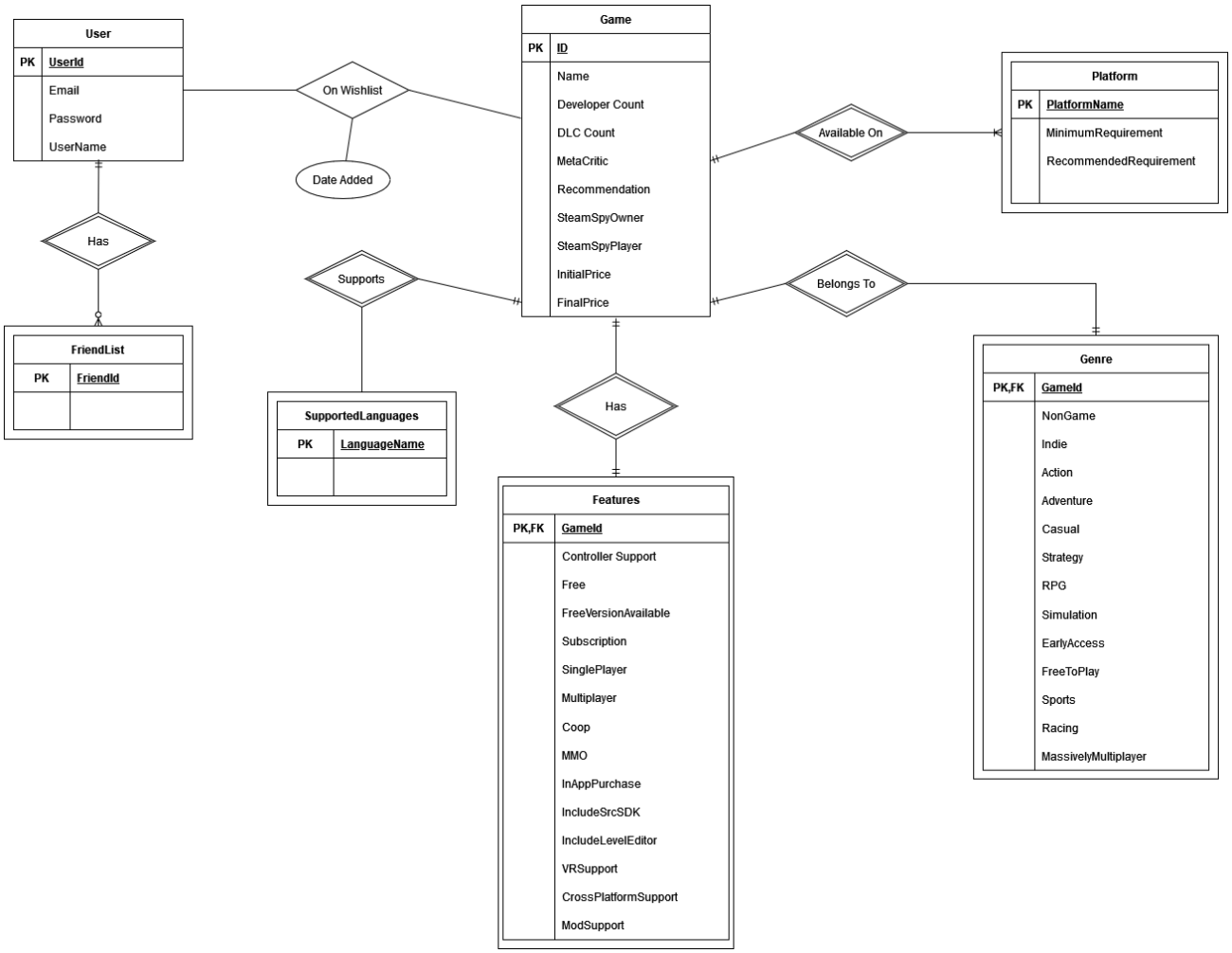
ER diagram:



Relational schema:

Game (ID:INT [PK],
NAME:VARCHAR(255),
DeveloperCount:INT,
DLCcount:INT,
MetaCritic:INT,
Recommendation:INT,
SteamSpyOwner:BIGINT,
SteamSpyPlayer:BIGINT,
InitialPrice:REAL,
FinalPrice:REAL
)

Platform (PlatformName:VARCHAR(255) [PK],
MiniumRequirement:VARCHAR(500),

RecommendedRequirement VARCHAR(500),
GameID:INT [PK,FK references to Game.ID]
)

Genre (NonGame: BOOLEAN,
Indie: BOOLEAN,
Action: BOOLEAN,
Adventure: BOOLEAN,
Casual: BOOLEAN,
Strategy: BOOLEAN,
RPG: BOOLEAN,
Simulation: BOOLEAN,
EarlyAccess: BOOLEAN,
FreeToPlay: BOOLEAN,
Sports: BOOLEAN,
Racing: BOOLEAN,
MassivelyMultiplayer: BOOLEAN,
GameID: INT [PK,FK references to Game.ID]
)

Features (ID:INT [PK,FK references to Game.ID],
ControllerSupport:BOOLEAN,
Free:BOOLEAN,
FreeVersionAvailable:BOOLEAN,
Subscription:BOOLEAN,
SinglePlayer:BOOLEAN,
MultiPlayer:BOOLEAN,
Coop: BOOLEAN,
MMO: BOOLEAN,
InAppPurchases: BOOLEAN,
IncludeSrcSDK: BOOLEAN,
IncludeLevelEditor: BOOL,
VRSupport: BOOL,
CrossPlatformSupport: BOOL,
ModSupport: BOOL,
)

User (UserID: INT [PK],
Email: VARCHAR(255),
Password: VARCHAR(255),
UserName: VARCHAR(255)
)

```
Wishlist ( UserID:INT [PK,FK references to User.UserID],
           GameID:INT [PK, FK references to Game.ID],
           DateAdded:VARCHAR(255)
)

FriendList (  FriendID: INT [PK],
          UserID: INT [PK, FK references to User.UserID]
          DateAdded: VARCHAR(255),
)

SupportedLanguages( LanguageName:VARCHAR(255) [PK],
                    GameID:INT [PK,FK references Game.ID]
)
```

Assumptions for entities:

For the main Game entity, we are assuming that the table contains mostly data that is more closely related to the game. That includes data for attributes like DLC count, producer Count, Metacritic scores, recommendations, player count, and prices for the game. For most of the attributes in this table other than the Name attribute, we are assuming that the data from our dataset for attributes like DLC count or Metacritic scores fit within the INT Type/Domain.

On the other hand, for the SteamSpyOwner count and SteamSpyPlayer count, we are afraid of the situation where there are more game player/owner count than the maximum value of INT, which is around 2 billion. As a result, we decided to use the BIGINT domain/type for the SteamSpyOwner and SteamSpyPlayer attributes. Although this will double the storage space for those attributes from 4 bytes (INT) to 8 bytes (BIGINT), this will definitely prevent any data overflowing problems for these two attributes, as it is impossible for a game to have user/player count bigger than the maximum of BIGINT, which is more than nine quintillion.

As for the Name of the game, we are using the common practice for names throughout this course, which is VARCHAR(255).

For the Features entity, we are assuming that all of the attributes have Type/Domain BOOLEAN, primarily because this is how the data is represented in the original dataset.This entity is a weak entity primarily because the game ID from the game table

determines the exact combinations of features for that game. It is possible for two games to have the same set of available features, so the combination of features cannot by themselves determine which game the table is referring to.

Since the Features-to-Game relation is an identifying relation, the relation of features-to-game has the cardinality of exactly one. On the other hand, based on the layout of the original dataset, we are assuming that every game has exactly one Features table associated with it. Even for games with no features, they would still have one linked features table with all the attribute values set to False. As a result, the game-to-features relation also has the cardinality of exactly one.

For the Genre entity, it behaves very similar to the Features table. All the attributes in the table have Type/Domain of Boolean, which is preserved from the original dataset. The Genre table is also a weak entity because two games could have the same genre combinations. So the genre attribute values cannot form an identifying relation, and the attribute values are solely determined by the gameID they are associated with.

The Genre-to-game relation is an Identifying relation, so the Genre-to-Game relation has a cardinality of exactly one. Similar to the Game-to-Features relation, the Game-to-Genre relation also has a cardinality of exactly one, because the original dataset gives every game exactly one combination of Boolean values to indicate what genre the game belongs to.

For the Platform entity, it is a weak entity primarily because we assume that both the minimum requirements and the recommended requirements are determined by a combination of GameID and platform name. The platform name alone cannot determine the requirements attributes since the requirements are platform-specific and game-specific. Since some of these requirements descriptions are quite long, we are using VARCHAR(500) instead of the normal VARCHAR(255) for these two attributes.

For the relation between Game and Platform, since the Platform-to-game relation is an identifying relation, the Platform-to-Game relation has cardinality of exactly one. On the other hand, we are assuming that a game will be available on at least one platform (otherwise no user can play the game), and one game on different platforms will have different hardware requirements. As a result, the Game-to-Platform relation has cardinality of at least 1 (Or Mandatory Many).

For the Language entity, it is a weak entity because the game ID, which is a foreign key, determines how many languages are available for a specific game. As a result, the

identifying relation means that the Language-to-Game relation has a cardinality of exactly one. On the other hand, some game entries in the dataset we are using do not have any supported language information, so we assume the Game-to-Language relation to be a general multiple/untitled relation.

We are assuming that a user can have 0 to many friends in his friends list, but a friends list only belongs to one user. Thus, the friends list entity is a weak entity, as it depends on the user id who owns this friend list. The Friend List-to-User relation has cardinality of exactly one, and the User-to-Friend-List relation has cardinality of 0 to many (Or Optional Many).

For the User-Game relation, we assume that a user can have multiple games on their wishlist, and a game can be on the wishlists of multiple users. Thus the User-Game relation is a many-to-many relation.