CNG 352

Database Management System

Games Library Website (Gamebook)

Raed Alsheikh Amin 2528271 || Farnaz Rezaei Noey 2551406

DESCRIPTION:

This web application aims to allow the user to see all their own games in one click across all the different platforms and stores, such as Steam, EA, Epic Games, etc. In addition, it will allow the user to add friends, add games to preferences, and recommend similar games to the game that the user played before under some conditions. Gamebook application will allow the community of gamers to ask their questions and solve their problems by using Question Response section. The web application will be developed using PHP, and MySQL.

1.1 Data Requirements

User

GameBook refers to members of the public interested in managing their games across different platforms and stores. To become a User, a person should log in to the website. The data stored on User includes user ID, first name, last name, username, email, password, and date of birth. The user ID is unique for each member. A user can own many Owned Games (Games already bought in each store) and a game can be owned by many users(this does not mean that users can't buy the game together). A user can also have many games as WishList Games and a game can be in many users' wish list. User also has a recursive relation as Friendship. The data stored on Friendship includes chat and status (with pending or accepted constraints) which belongs to both users.

Owned Game

Owned Games are the games that a person has already bought. Owned Games are either installed or not installed. The data stored on Owned Games includes name, icon, description, type, genre, store, age rating, isFavorite, and hours played. Name is the name of the game which is unique across platforms. Age rating has the PEGI 3, PEGI 7, PEGI 12, PEGI 16, and PEGI 18 constraint (PEGI 3 means suitable for all ages, PEGI 7 means suitable for young children, PEGI 12 means suitable for children 12 and over, PEGI 16 means suitable for children 16 and over, and PEGI 18 means Only suitable for adults, these ratings are based on the contents of the game). Installed games are stored in the library with their date added.

Installed

Installed are the games that the user already has on the PC. The reason we are keeping track of installed or uninstalled games is that a game can be bought but not installed yet and we would like to check for redundancy for uninstalled games across different platforms.

Not Installed

Not Installed are the games that the user does not have on the PC but has already bought them.

WishList Game

WishList Games refers to the games that the user intends to buy and does not own in any store. The data stored on the WishList Games includes Price, name, icon, description, type, genre, store, and age rating. Name is the name of the game which is unique across platforms. Age rating has the PEGI3, PEGI7, PEGI12, PEGI16, and PEGI18 constraint (PEGI 3 means suitable for all ages, PEGI 7 means suitable for young children, PEGI 12 means suitable for children 12 and over, PEGI 16 means suitable for children 16 and over, and PEGI 18 means Only suitable for adults, these ratings are based on the contents of the game).

Library

The library is where the Owned Games will be stored with their date added. The data stored on Library will include sorting type and Library ID. Library ID is unique for each library. One library can store many Owned Games. Each user has only one library to manage, and that library belongs to only one user.

Platform

The data stored on the platform includes platform ID, name, and icon. Platform ID is unique across the platforms. Many Owned Games and many WishList Games belong to one or many platforms.

Feedback

The data stored on the Feedback includes Feedback ID, title, comment, timestamp, and rating (which has 0-5 constraint). Feedback ID is unique for each feedback. A user can make many feedback but a feedback belongs to one user, also an Owned Game can have multiple feedback but a feedback belongs to an Owned Game. We cannot give feedback to the games in the Wishlist since the user does not own those games yet which means the user hasn’t played the games yet so he shouldn’t be able to give feedback about it.

Question

Question is where the data related to the questions that the users asked will be stored. If a user has questions about a game the data will be stored in the question entity. No one can ask questions unless they are users. The data stored on Question includes QID and text. QID is unique for each question asked. A user can ask many questions, but one question is asked by one user. A question also can have many responses, but a respond belongs to one question.

Response

Response is where the data related to the responses to the questions will be stored. If a user replies to a question the data will be stored in the response entity. There can be no response if there are no questions. The data stored on Response includes ResponseID and text. ResponseID is unique for each response. A user can respond to many questions, but a respond belongs to one user.

1.2 Transaction Requirements

Data Entry (attributes are simplified but will be implemented fully)

Enter the details of a new user (such as name: Farnaz, last name: Rezaei, username: farnazrzn, email: farnazrzn@gmail.com, password:1234, DOB: 22/7/2001)

Enter the details of new owned game (such as Minecraft survival genre with PEGI7)

Enter the details of new wish list game (such as Tomb raider adventure genre with PEGI12 for 10$)

Enter the details of new feedback (such as Minecraft bug, sheep got stuck in a wall, with rating of 3)

Enter the details of new question (such as how to make a fishing rod in Minecraft?)

Enter the details of new respond (such as you will need spider webs and wood to make the fishing rod)

Enter the details of new friend (such as Raed Amin with raedamin username and 432 user ID)

Data update/deletion

Delete the details of a friend from friends list.

Update/delete the details of a game in the wish list.

Update/delete the details of a game in the owned games list (whether it's installed or not).

Update/delete the details of the question.

Update/delete the details of the response.

Data queries

(a) List the details of all friends that the user has.

(b) List the details of the friend named Raed.

(c) List the details of the pending friends (pending friends are those requests that are neither accepted nor rejected). (status)

(d) List the details of all the owned games.

(e) List the details of the installed games.

(f) List the details of the first-person shooting games. (type)

(g) List the details of the favorite games.

(h) List the details of the adventure games. (genre)

(i) List the details of the games with PEGI12 rating.

(j) Group the games based on the belonging platform.

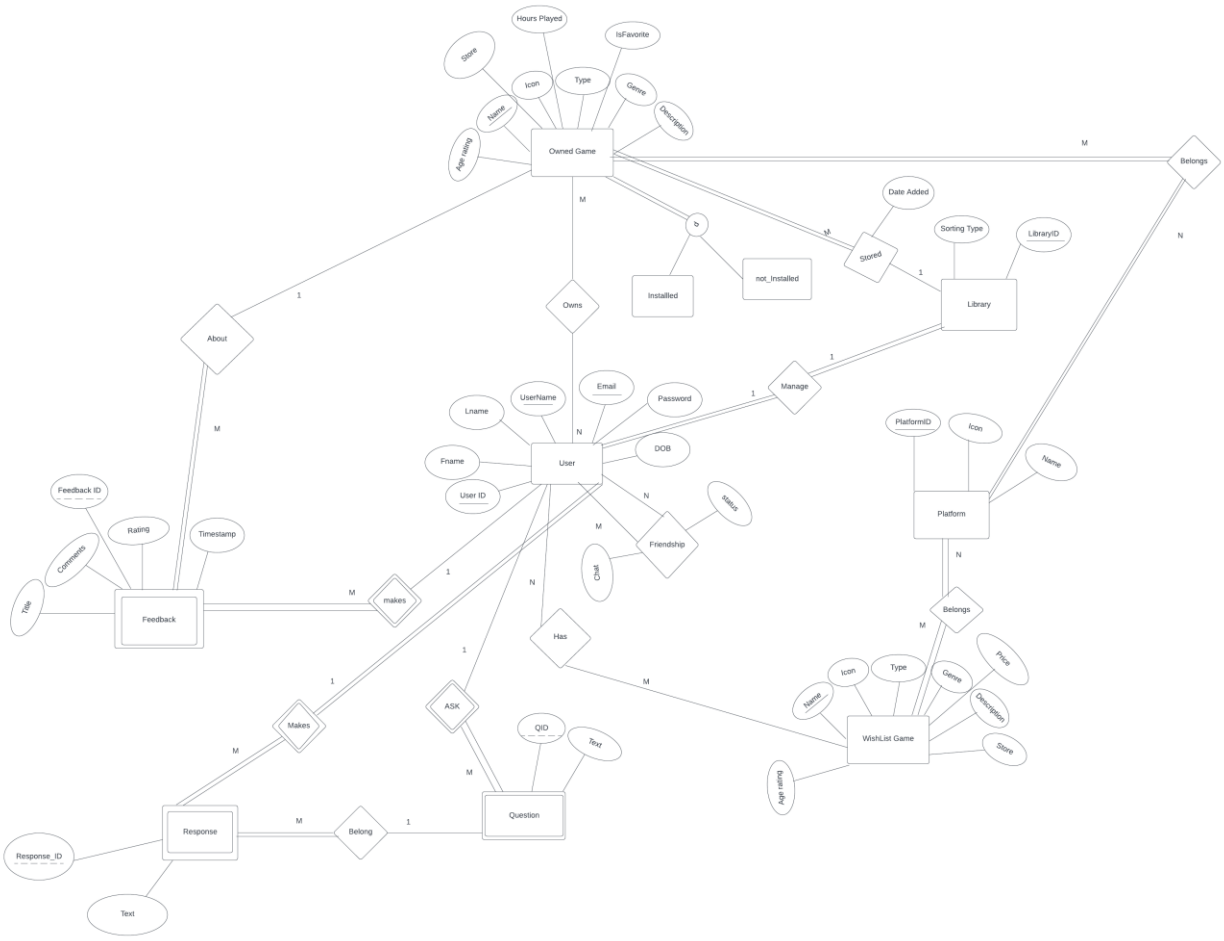
(k) List the details of all the games in the wish list.

(l) Display the cheapest game on the wish list.

(m) Display the game with the highest discount on the wish list.

(n) List the details of all the feedback/questions/responses the user made.

2.1 EER Diagram



Assumptions and Constraints

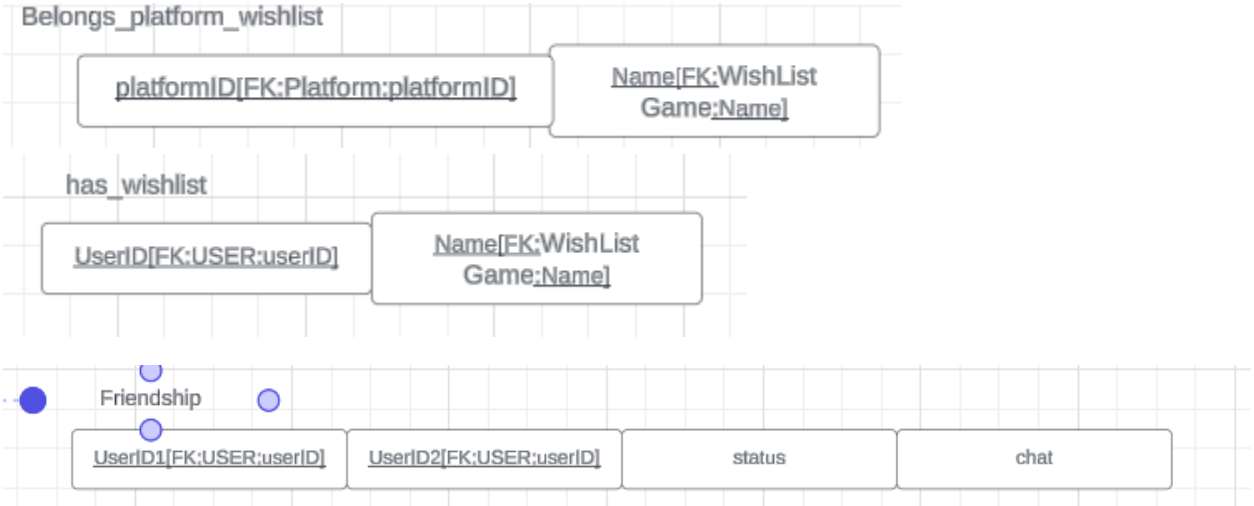
- Status in Friendship relation has pending or accepted constraints.
- A game can be owned by many users, but this does not mean that users can't buy the game together.
- Also, a game can belong to many platforms. This means different platforms can collaborate with each other to produce a game.

- Disjoint is used for owned game since a game can either be installed or uninstalled but not both.
- Age rating has the PEGI 3, PEGI 7, PEGI 12, PEGI 16, and PEGI 18 constraint (PEGI 3 means suitable for all ages, PEGI 7 means suitable for young children, PEGI 12 means suitable for children 12 and over, PEGI 16 means suitable for children 16 and over, and PEGI 18 means Only suitable for adults, these ratings are based on the contents of the game).
- Type and genre attribute in owned game are different from each other. "Type" typically refers to the fundamental gameplay mechanics and structure of a game like puzzle games, while "genre" refers to the overall theme, setting, and atmosphere of a game like action-adventure.
- Rating in Feedback entity has 0-5 constraint.
- Feedback is a weak entity because if a user does not make feedback there won't be any hence it has identifying relationship with user entity.
- Question is a weak entity since if a user does not ask a question, we won't have any question hence it has identifying relationship with user entity.
- Response is a weak entity since if a user does not reply to a question, we won't have any response hence it has identifying relationship with user entity.

3.1 Mapping

This is the first mapping of the EER diagram before normalization.

Updated tables: (corrected)



Assumptions

- isInstalled flag is the technique used to map the children of Owned games (installed, not installed) and the domain of it will be (0,1) (uninstalled, installed)
- Sorting Type has game name and platform constraints. This means the user can choose to see the games in order of the name of the games or in the order of the platforms.
- isFavorite flag has 0 or 1 constraint. If the game is favorite of the user it will be set to 1 else it will be 0 by default.
- Status in Friendship relation has pending or accepted constraints.
- Age rating has the PEGI 3, PEGI 7, PEGI 12, PEGI 16, and PEGI 18 constraint (PEGI 3 means suitable for all ages, PEGI 7 means suitable for young children, PEGI 12 means suitable for children 12 and over, PEGI 16 means suitable for children 16 and over, and PEGI 18 means Only suitable for adults, these ratings are based on the contents of the game).
- Rating in Feedback entity has 0-5 constraint.

3.2 Normalization

All the relations satisfy all Normalizations until BCNF => 1NF, 2NF,3NF,BCNF are all satisfied and there is no need for further split.

FDS:

User:

FD1: userID-> the rest

FD2: username-> the rest

FD3: Email -> the rest

Owned Game:

FD1: Name -> the rest

Library:

FD1: LibraryID-> the rest

Platform:

FD1: PlatformID-> the rest

FD2: Name -> the rest

WishList Game:

FD1: Name -> the rest

Feedback:

FD1: feedbackID-> the rest

Response:

FD1: responseID-> the rest

Question:

FD1: QID -> the rest

Belongs_game_platform:

Owns:

Belongs_platform_wishlist:

Friendship:

FD1: userid1, userid2 -> the rest

has_wishlist:

4.1 Create Tables (MYSQL)

Dropping the tables before each time running the database.

```
DROP TABLE Feedback;

DROP TABLE Response;

DROP TABLE Question;

DROP TABLE Belongs_game_platform;

DROP TABLE Belongs_platform_wishlist;

DROP TABLE Has_wishlist;

DROP TABLE Friendship;

DROP TABLE Owns;

DROP TABLE Users;
```

```
DROP TABLE OwnedGame;
DROP TABLE Library;
DROP TABLE Platform;
DROP TABLE WishListGame;

CREATE TABLE WishListGame(
Name CHAR(50) NOT NULL,
Icon BLOB DEFAULT NULL,
Typee CHAR(50) NOT NULL,
Genre CHAR(50) NOT NULL,
Age_rating INT NOT NULL CHECK (Age_rating IN (3, 7, 12, 16, 18)),
Store CHAR(50) NOT NULL,
Description TEXT NOT NULL,
Price INT NOT NULL,
PRIMARY KEY (Name));


CREATE TABLE Platform(
PlatformID INT NOT NULL,
Icon BLOB DEFAULT NULL,
Name CHAR(50) NOT NULL,
PRIMARY KEY (PlatformID));


CREATE TABLE Library(
LibraryID INT NOT NULL,
Sorting_Type CHAR(50) NOT NULL,
PRIMARY KEY (LibraryID));
```

```
CREATE TABLE Users(
UserID INT NOT NULL,
Fname CHAR(20) NOT NULL,
Lname CHAR(20) NOT NULL,
UserName CHAR(20) NOT NULL,
Email CHAR(50) NOT NULL,
Password char(10)NOT NULL,
DOB DATE NOT NULL,
Manage_LibraryID INT NOT NULL,
PRIMARY KEY (UserID),
FOREIGN KEY (Manage_LibraryID) REFERENCES Library (LibraryID));


CREATE TABLE OwnedGame(
Name CHAR(50) NOT NULL,
Icon BLOB DEFAULT NULL,
Typee CHAR(50) NOT NULL,
Genre CHAR(50) NOT NULL,
Age_rating INT NOT NULL CHECK (Age_rating IN (3, 7, 12, 16, 18)),
Store CHAR(50) NOT NULL,
Description TEXT NOT NULL,
HoursPlayed INT DEFAULT 0 NOT NULL,
isFavorite INT DEFAULT 0 CHECK (isFavorite IN (0,1)),
isInstalled INT NOT NULL CHECK (isInstalled IN (0,1)),
DateAdded DATE NOT NULL,
Stored_LibraryID INT NOT NULL,
```

```
PRIMARY KEY (Name),
FOREIGN KEY (Stored_LibraryID) REFERENCES Library (LibraryID));


CREATE TABLE Feedback(
FeedbackID INT NOT NULL,
Title CHAR(20) NOT NULL,
Comments TEXT NOT NULL,
Rating INT NOT NULL CHECK (Rating IN (0,1,2,3,4,5)),
time_stamp TIMESTAMP NOT NULL,
userid INT NOT NULL,
About_name CHAR(50) NOT NULL,
PRIMARY KEY (FeedbackID),
FOREIGN KEY (userid) REFERENCES Users(UserID),
FOREIGN KEY (About_name) REFERENCES OwnedGame(Name));


CREATE TABLE Question(
QID INT NOT NULL,
Textt TEXT NOT NULL,
userid INT NOT NULL,
PRIMARY KEY (QID),
FOREIGN KEY (userid) REFERENCES Users(UserID));


CREATE TABLE Response(
ResponseID INT NOT NULL,
Textt TEXT NOT NULL,
```

```
userid INT NOT NULL,
Belongs_QID INT NOT NULL,
PRIMARY KEY (ResponseID),
FOREIGN KEY (userid) REFERENCES Users(UserID),
FOREIGN KEY (Belongs_QID) REFERENCES Question(QID));

CREATE TABLE Belongs_game_platform(
Name CHAR(50) NOT NULL,
PlatformID INT NOT NULL,
PRIMARY KEY (Name,PlatformID),
FOREIGN KEY (Name) REFERENCES OwnedGame(Name),
FOREIGN KEY (PlatformID) REFERENCES Platform(PlatformID));

CREATE TABLE Owns(
userid INT NOT NULL,
Name CHAR(50) NOT NULL,
PRIMARY KEY (userid,Name),
FOREIGN KEY (userid) REFERENCES Users(UserID),
FOREIGN KEY (Name) REFERENCES OwnedGame(Name));

CREATE TABLE Belongs_platform_wishlist(
Name CHAR(50) NOT NULL,
PlatformID INT NOT NULL,
PRIMARY KEY (Name,PlatformID),
FOREIGN KEY (Name) REFERENCES WishListGame(Name),
```

```
FOREIGN KEY (PlatformID) REFERENCES Platform(PlatformID));


CREATE TABLE Friendship(
UserID1 INT NOT NULL,
UserID2 INT NOT NULL,
Status CHAR(50) DEFAULT 'Pending' NOT NULL CHECK (Status IN ('Pending','Accepted')),
PRIMARY KEY (UserID1, UserID2),
FOREIGN KEY (UserID1) REFERENCES Users(UserID),
FOREIGN KEY (UserID2) REFERENCES Users(UserID));


CREATE TABLE Has_wishlist(
userid INT NOT NULL,
Name CHAR(50) NOT NULL,
PRIMARY KEY (userid,Name),
FOREIGN KEY (userid) REFERENCES Users(UserID),
FOREIGN KEY (Name) REFERENCES WishListGame(Name));
```

4.2 Data Population

```
INSERT INTO WishListGame VALUES ('Finals',NULL,'FPS','Base Game',12,'Epic Games', 'FIrst Person Shooting Spike Rush Game with Heros Having Special Abilities',0);

INSERT INTO WishListGame VALUES ('The Witcher 3: Wild Hunt', NULL, 'RPG', 'Open World', 18, 'Steam', 'A story-driven, open world role-playing game set in a dark fantasy universe.', 29);

INSERT INTO WishListGame VALUES ('Minecraft', NULL, 'Adventure', 'Sandbox', 7, 'Mojang', 'A game about placing blocks and going on adventures.', 20);

INSERT INTO WishListGame VALUES ('FIFA 22', NULL, 'Sports', 'Soccer', 3, 'Origin', 'The latest installment in the FIFA series, featuring updated teams and gameplay.', 60);
```

```
INSERT INTO WishListGame VALUES ('The Legend of Zelda: Breath of the Wild', NULL, 'Action-Adventure', 'Open World', 12, 'Nintendo eShop', 'An action-adventure game set in an open world environment.', 50);

INSERT INTO WishListGame VALUES ('Among Us', NULL, 'Casual', 'Social Deduction', 7, 'Steam', 'A multiplayer game where crewmates try to complete tasks while impostors try to eliminate them.', 5);


INSERT INTO Platform VALUES (1111,NULL,'XBOX');

INSERT INTO Platform VALUES (2222, NULL, 'PlayStation');

INSERT INTO Platform VALUES (3333, NULL, 'Nintendo Switch');

INSERT INTO Platform VALUES (4444, NULL, 'Android');

INSERT INTO Platform VALUES (5555, NULL, 'IOS');

INSERT INTO Platform VALUES (6666, NULL, 'PSP');


INSERT INTO Library VALUES (123, 'Letter');

INSERT INTO Library VALUES (456, 'Platform');

INSERT INTO Library VALUES (789, 'Letter');

INSERT INTO Library VALUES (012, 'Platform');

INSERT INTO Library VALUES (023, 'Letter');

INSERT INTO Library VALUES (056, 'Platform');


INSERT INTO Users VALUES (1234, 'Farnaz', 'Rezaee', 'farnazrzn', 'farnaz@gmail.com', 'farnaz123', '2001-07-22',123);

INSERT INTO Users VALUES (5678, 'Raed', 'Amin', 'raedamin', 'raed.amin@yahoo.com', 'raed456', '2001-05-25',456);

INSERT INTO Users VALUES (91011, 'Alice', 'Smith', 'alicesmith', 'alice.smith@gmail.com', '988Alic*', '1988-11-28',789);

INSERT INTO Users VALUES (121314, 'Michael', 'Johnson', 'michaelj', 'michael.j@gmail.com', 'MichJ76', '1976-09-15',012);

INSERT INTO Users VALUES (151617, 'Emily', 'Brown', 'emilyb', 'emily.b@yahoo.com', 'EBrown502', '1992-05-02',023);
```

INSERT INTO Users VALUES (181920, 'David', 'Wilson', 'davidw', 'david.w@gmail.com', '1218@Dwil', '2000-12-18',056);

INSERT INTO OwnedGame VALUES ('Valorant',NULL,'FPS','Base Game',12,'Epic Games', 'FIrst Person Shooting Spike Rush Game with Heros Having Special Abilities',12,1,1,'2024-05-14',123);

INSERT INTO OwnedGame VALUES ('Fortnite', NULL, 'Battle Royale', 'Action', 12, 'Epic Games', 'A battle royale game where you can build structures to defend yourself', 50, 0, 1, '2023-01-20',456);

INSERT INTO OwnedGame VALUES ('Counter-Strike: Global Offensive', NULL, 'FPS', 'Multiplayer', 16, 'Steam', 'A multiplayer first-person shooter game.', 500, 1, 1, '2022-08-12',789);

INSERT INTO OwnedGame VALUES ('Overwatch', NULL, 'FPS', 'Hero Shooter', 12, 'Blizzard', 'A team-based multiplayer hero shooter game.', 80, 1, 1, '2023-06-10',012);

INSERT INTO OwnedGame VALUES ('Red Dead Redemption 2', NULL, 'Action-Adventure', 'Open World', 18, 'Rockstar Games', 'An epic tale of life in America at the dawn of the modern age.', 120, 0, 1, '2023-09-25',023);

INSERT INTO OwnedGame VALUES ('Rocket League', NULL, 'Sports', 'Vehicular Soccer', 3, 'Epic Games', 'A high-powered hybrid of arcade-style soccer and vehicular mayhem.', 40, 1, 0, '2022-10-15',056);

INSERT INTO Feedback VALUES (1, 'Great Game', 'I really enjoyed playing this game. The graphics are amazing and the gameplay is smooth.', 5, NOW(), 1234, 'Valorant');

INSERT INTO Feedback VALUES (2, 'Needs Improvement', 'This game has potential but needs some bug fixes and balancing.', 3, NOW(), 5678, 'Fortnite');

INSERT INTO Feedback VALUES (3, 'Awesome Gameplay', 'Counter-Strike: Global Offensive is a classic. The gameplay is intense and keeps you coming back for more.', 5, NOW(), 91011, 'Counter-Strike: Global Offensive');

INSERT INTO Feedback VALUES (4, 'Fun with Friends', 'Overwatch is a blast to play with friends. The different heroes and abilities make each match unique.', 4, NOW(), 121314, 'Overwatch');

INSERT INTO Feedback VALUES (5, 'Amazing Story', 'Red Dead Redemption 2 has one of the best stories in a video game. The open world is beautifully crafted.', 5, NOW(), 151617, 'Red Dead Redemption 2');

INSERT INTO Feedback VALUES (6, 'Addictive Gameplay', 'Rocket League is so much fun. The fast-paced matches make it hard to put down.', 5, NOW(), 181920, 'Rocket League');

INSERT INTO Question VALUES (1, 'How do I unlock new agents in Valorant?', 1234);

INSERT INTO Question VALUES (2, 'What are some tips for building in Fortnite?', 5678);

INSERT INTO Question VALUES (3, 'How do I improve my aim in Counter-Strike: Global Offensive?', 91011);

INSERT INTO Question VALUES (4, 'What are the best heroes for beginners in Overwatch?', 121314);

INSERT INTO Question VALUES (5, 'How long is the main story in Red Dead Redemption 2?', 151617);

INSERT INTO Question VALUES (6, 'What are some advanced strategies in Rocket League?', 181920);

INSERT INTO Response VALUES (11, 'To unlock new agents in Valorant, you need to earn enough Valorant Points (VP) to unlock them from the in-game store.', 1234, 1);

INSERT INTO Response VALUES (22, 'In Fortnite, try to land in less populated areas at the start to gather resources and avoid early fights. Use the resources to build structures for defense later in the game.', 5678, 2);

INSERT INTO Response VALUES (33, 'To improve your aim in Counter-Strike: Global Offensive, practice aiming for headshots in deathmatch and use aim training maps to improve your reflexes.', 91011, 3);

INSERT INTO Response VALUES (44, 'For beginners in Overwatch, heroes like Soldier: 76, Mercy, and Reinhardt are good choices as they have straightforward abilities and are easy to learn.', 121314, 4);

INSERT INTO Response VALUES (55, 'The main story in Red Dead Redemption 2 can take around 50-60 hours to complete if you focus on the main missions and skip some of the side content.', 151617, 5);

INSERT INTO Response VALUES (66, 'In Rocket League, practice aerial shots and learn to use boost efficiently. Positioning and teamwork are also key to success in matches.', 181920, 6);

INSERT INTO Belongs_game_platform VALUES ('Valorant', 1111);

INSERT INTO Belongs_game_platform VALUES ('Fortnite', 2222);

INSERT INTO Belongs_game_platform VALUES ('Counter-Strike: Global Offensive', 3333);

INSERT INTO Belongs_game_platform VALUES ('Overwatch', 4444);

```
INSERT INTO Belongs_game_platform VALUES ('Red Dead Redemption 2', 5555);
INSERT INTO Belongs_game_platform VALUES ('Rocket League', 6666);

INSERT INTO Owns VALUES (1234, 'Valorant');
INSERT INTO Owns VALUES (5678, 'Fortnite');
INSERT INTO Owns VALUES (91011, 'Counter-Strike: Global Offensive');
INSERT INTO Owns VALUES (121314, 'Overwatch');
INSERT INTO Owns VALUES (151617, 'Red Dead Redemption 2');
INSERT INTO Owns VALUES (181920, 'Rocket League');

INSERT INTO Belongs_platform_wishlist VALUES ('Finals', 1111);
INSERT INTO Belongs_platform_wishlist VALUES ('The Witcher 3: Wild Hunt', 2222);
INSERT INTO Belongs_platform_wishlist VALUES ('Minecraft', 3333);
INSERT INTO Belongs_platform_wishlist VALUES ('FIFA 22', 4444);
INSERT INTO Belongs_platform_wishlist VALUES ('The Legend of Zelda: Breath of the Wild',
5555);
INSERT INTO Belongs_platform_wishlist VALUES ('Among Us', 6666);

INSERT INTO Friendship VALUES (1234, 5678,'Accepted');
INSERT INTO Friendship VALUES (1234, 91011,'Accepted');
INSERT INTO Friendship VALUES (5678, 91011,'Pending');
INSERT INTO Friendship VALUES (121314, 151617,'Accepted');
INSERT INTO Friendship VALUES (121314, 181920,'Pending');
INSERT INTO Friendship VALUES (151617, 181920,'Accepted');

INSERT INTO Has_wishlist VALUES (1234, 'Finals');
INSERT INTO Has_wishlist VALUES (5678, 'The Witcher 3: Wild Hunt');
INSERT INTO Has_wishlist VALUES (91011, 'Minecraft');
INSERT INTO Has_wishlist VALUES (121314, 'FIFA 22');
```

INSERT INTO Has_wishlist VALUES (151617, 'The Legend of Zelda: Breath of the Wild');

INSERT INTO Has_wishlist VALUES (181920, 'Among Us');

4.3 Data Manipulation

- Delete the details of a friend from friends list.

```
DELETE FROM Friendship
WHERE (UserID1 = 1234 AND UserID2 = 5678)
  OR (UserID1 = 5678 AND UserID2 = 1234);
```

- Update/delete the details of a game in the wish list.

```
UPDATE WishListGame
SET Price = 32
WHERE Name = 'FIFA 22';

DELETE FROM Has_wishlist
WHERE userid = 121314 AND Name = 'FIFA 22';
```

- Update/delete the details of a game in the owned games list (whether it's installed or not).

```
UPDATE OwnedGame
SET HoursPlayed = 40, isFavorite = 0, isInstalled = 0
WHERE Name = 'Valorant';

DELETE FROM Owns
WHERE userid = 1234 AND Name = 'Valorant';
```

- Update/delete the details of the question.

```
UPDATE Question
SET Textt = 'How do I BUY the new agent Glove in Valorant?'
WHERE QID = 1;
```

```
DELETE FROM Response
WHERE Belongs_QID = 1;
DELETE FROM Question
WHERE QID = 1;
```

- Update/delete the details of the response.

```
UPDATE Response
SET Textt = 'To BUY the new agent Glove in Valorant, you need to wait for the event to finish and after 11 days you can buy it with Valorant Points.'
WHERE ResponseID = 11;

DELETE FROM Response
WHERE ResponseID = 11;
```

4.4 Data queries

(a) List the details of all friends that the user has.

```
SELECT Users.UserID, Users.Fname, Users.Lname
FROM Friendship
JOIN Users ON (Friendship.UserID1 = Users.UserID OR Friendship.UserID2 = Users.UserID)
WHERE (Friendship.UserID1 = 1234 OR Friendship.UserID2 = 1234)
AND Users.UserID != 1234;
```

(b) List the details of the friend named Raed.

```
SELECT *
FROM Users
WHERE Fname = 'Raed';
```

(c) List the details of the pending friends (pending friends are those requests that are neither accepted nor rejected). (status)

```
SELECT Users.UserID, Users.Fname, Users.Lname
FROM Friendship
JOIN Users ON Friendship.UserID2 = Users.UserID
WHERE Friendship.UserID1 = 5678
```

```
AND Friendship.Status = 'Pending';
```

(d) List the details of all the owned games.

```
SELECT * FROM OwnedGame
WHERE Stored_LibraryID = (SELECT Manage_LibraryID FROM Users WHERE UserID
= 1234);
```

(e) List the details of the installed games.

```
SELECT * FROM OwnedGame
WHERE isInstalled = 1 AND Stored_LibraryID = (SELECT Manage_LibraryID FROM
Users WHERE UserID = 1234);
```

(f) List the details of the first-person shooting games. (type)

```
SELECT * FROM OwnedGame
WHERE Typee = 'FPS' AND Stored_LibraryID = (SELECT Manage_LibraryID FROM
Users WHERE UserID = 1234);
```

(g) List the details of the favorite games.

```
SELECT * FROM OwnedGame
WHERE isFavorite = 1 AND Stored_LibraryID = (SELECT Manage_LibraryID FROM
Users WHERE UserID = 1234);
```

(h) List the details of the adventure games. (genre)

```
SELECT * FROM OwnedGame
WHERE Genre = 'Adventure' AND Stored_LibraryID = (SELECT Manage_LibraryID
FROM Users WHERE UserID = 1234);
```

(i) List the details of the games with PEGI12 rating.

```
SELECT * FROM OwnedGame
WHERE Age_rating = 12 AND Stored_LibraryID = (SELECT Manage_LibraryID FROM
Users WHERE UserID = 1234);
```

(j) Group the games based on the belonging platform.

```
SELECT Platform.Name AS PlatformName, GROUP_CONCAT(OwnedGame.Name)
AS OwnedGames, GROUP_CONCAT(WishListGame.Name) AS WishListGames
FROM Platform
LEFT JOIN Belongs_game_platform ON Platform.PlatformID =
Belongs_game_platform.PlatformID
LEFT JOIN OwnedGame ON Belongs_game_platform.Name = OwnedGame.Name
LEFT JOIN Belongs_platform_wishlist ON Platform.PlatformID =
Belongs_platform_wishlist.PlatformID
LEFT JOIN WishListGame ON Belongs_platform_wishlist.Name =
WishListGame.Name
GROUP BY PlatformName;
```

(k) List the details of all the games in the wish list.

```
SELECT WishListGame.*
FROM Has_wishlist
JOIN WishListGame ON Has_wishlist.Name = WishListGame.Name
WHERE Has_wishlist.userid = 1234;
```

(l) Display the cheapest game on the wish list.

(m) Display the game with the highest discount on the wish list.

Since we don't keep track of the discount and we just update the price, l and m are basically the same.

```
SELECT WishListGame.*
FROM Has_wishlist
JOIN WishListGame ON Has_wishlist.Name = WishListGame.Name
WHERE Has_wishlist.userid = 1234
ORDER BY WishListGame.Price
LIMIT 1;
```

(n) List the details of all the feedback/questions/responses the user made.

```
SELECT FeedbackID AS ID, Title AS Text, Comments AS Detail, Rating AS Score,
time_stamp AS Timestamp
FROM Feedback
WHERE userid = 1234
UNION ALL
SELECT QID AS ID, Textt AS Text, NULL AS Detail, NULL AS Score, NULL AS
Timestamp
FROM Question
WHERE userid = 1234
UNION ALL
SELECT ResponseID AS ID, Textt AS Text, NULL AS Detail, NULL AS Score, NULL AS
Timestamp
FROM Response
WHERE userid = 1234;
```

4.5 discussion (index and workload)

Query Frequency:

Read queries: These might include displaying game details, showing wish lists, retrieving feedback, and listing owned games. These operations are likely to be frequent, especially for popular games or active users.

Write queries: Adding new games, providing feedback, asking questions, and managing friendships are write operations that might be less frequent but still important.

The number of records in each table can vary. For example, the Users table might have many records, while the Friendship table might have fewer records.

Game updates, price changes, and user interactions (e.g., adding friends, updating game libraries) may require frequent updates to the database.

primary and Foreign Keys indexes:

Foreign keys (e.g., Manage_LibraryID in Users) can also be indexed to speed up joins.

Primary keys can be indexed for sorting efficiency and retrieving data faster..

Conclusion:

while indexes can improve query performance, they also have overhead in terms of storage and maintenance. Therefore, indexes should be carefully chosen based on the specific workload and usage patterns of the application. Regular monitoring and tuning of indexes can help ensure optimal performance, and according to GameBook application, the indexes will complicate things since it doesn't have sufficient data to be stored.