Database Design
Project 4
Dr. Yicheng Tu

Game Shop: One Space

Company Executives:

- Davin Hill
- Karishma Jayaprakash
- Krishna Patel

I. Overview

Our group consists of three members, Davin Hill, Karishma Jayaprakash, and Krishna Patel. We proposed a new online game store named “One Space” where users can purchase games of their choice. The main purpose of our website is to allow users from all over the world to buy all types of games at one place with convenient checkout options.

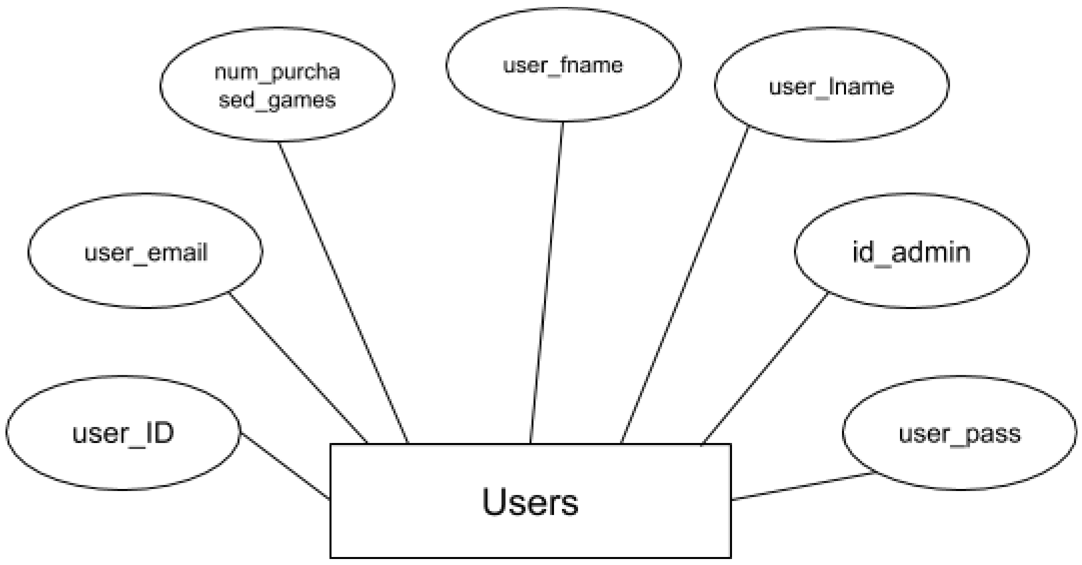
The website will be similar to GameStop, a well-known in-store and online game purchasing site, but we strive to offer games at a better price and an easy checkout option. We constructed the website with multiple views, where the admin view gets the opportunity to change and alter the games on the database conveniently through the website and thus our website gives the user a seemingly user-friendly environment.

II. Database Design

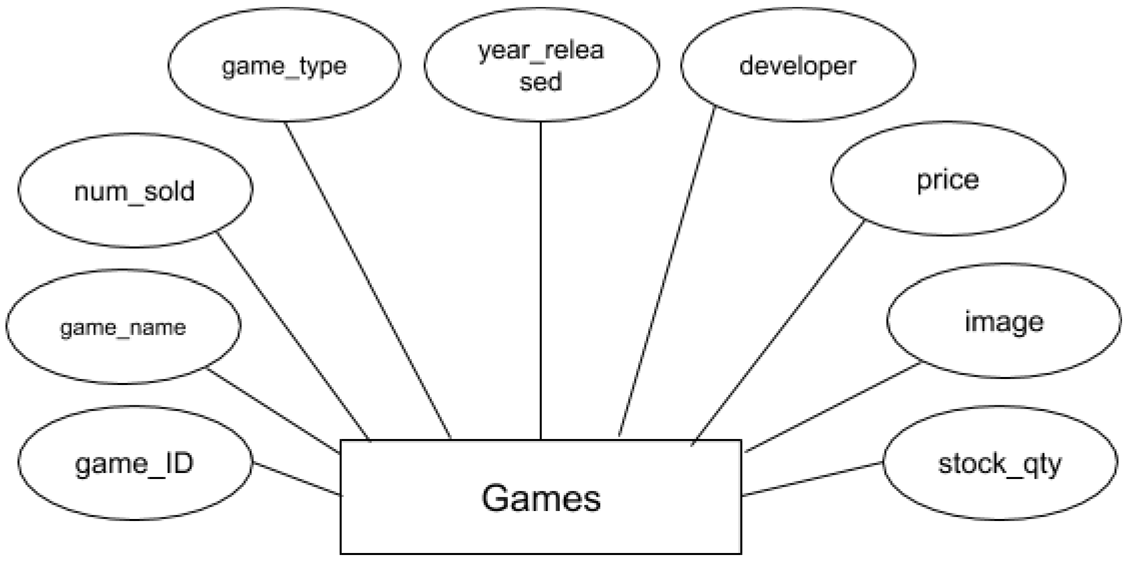
The following ER diagram provides a high-level introduction to our database. This will be helpful to understand the relation between the tables.

Entity-Relationship Diagrams:

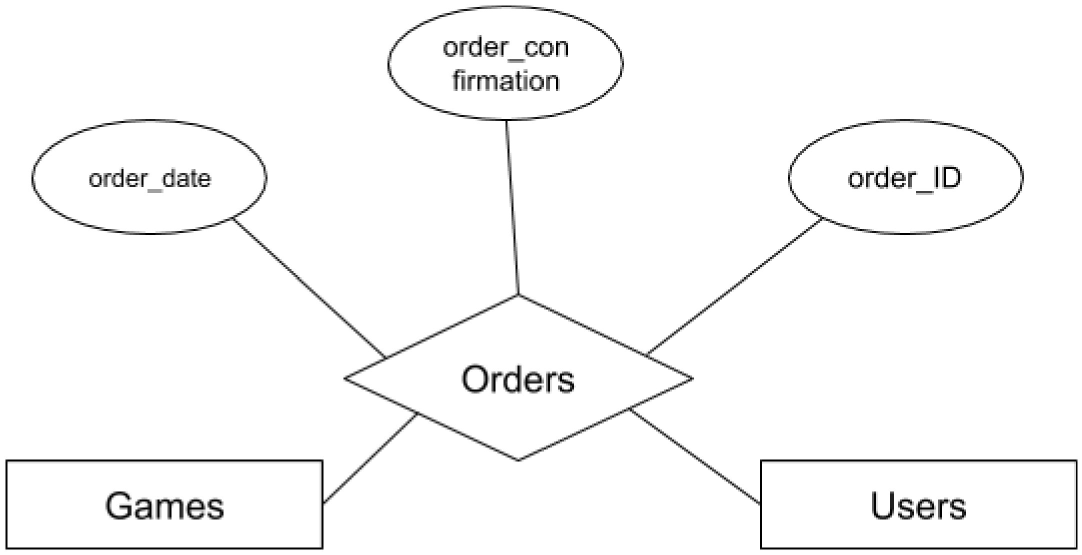
Users



Games



Orders



III. Client Functionality

This section talks about the different types of views depending on privileges assigned and how the user interface works. Basically, there will be three different of views for three different users.

1. **Anonymous user:**

- All the users who initially connect to the website are in this class and maintain in it until they log in with an account.
- The user will be able to view the home page that includes the log in section and the new user section.
- The user will also be able to know the trending game and also the top 8 best-selling games.
- Additionally, the user will be able to view all different types of games including Action, Fighting, Shooter, Simulation, and Sports.
- The user can view everything and will be able to purchase any game.

2. **Logged in user:**

- The user will be able to view all the things which can be viewed by the anonymous user.
- Additionally, the user will be able to see his/her first name once logged in.
- The user will have the privilege of purchasing different types games at the same time.
- The user will also be able to confirm the purchase order with the confirmation order number.
- Logged in users will have their table updated in the database for how many games they have purchased.

3. **Admin:**

- The admin will also be able to view top 8 best-selling game on the home page and also in the each game category section.
- The admin won't be able to purchase any game while being in the admin view.

- The admin will have privilege to add new games to the database, delete any existing game, changing price of the existing game, or updating the number of quantity of a particular game.

Common to all page

Every page will have a menu bar at the top of the website. This will help all type of user to move around the website with ease. There is also a section, at the bottom of each page, where the contact information, both phone number and email address is provided. Also credit is given to w3.css for providing the template for the website.

1. Anonymous User:

If the user is not logged in, the menu bar will consist of four different options of “Home”, “Shopping cart”, “New user” and “Log in”.

a. Home Page

- The home page will be the first page users see when they connect to the website. There will be a message on the top left corner saying “Your Best Place to Shop Games”. The home page will have the most trending game’s name and it’s picture. Additionally most eight best selling games will also be displayed on the home page with their price. The eight games keep changing as the number of purchase for that game changes.
- The home page will also consist of the shop by Category section. This section will have six different type of option. Each option will take the user to the different page, where they will be able to view more games under that specific category. Clicking on the home page button, the user will brought back to the home page view.
- Anonymous users can see the option to purchase on the home page, and they will still be able to purchase games.The log out page is simply for the logged in user to log out the website once they finish visiting the website.
- The application layer will check if the user is logged in or not. If so, the section will get closed and user will be logged out and directly to the homepage of the anonymous user.
- The application layer is implemented by HTML, CSS, and PHP.

b. Shopping cart

- The shopping cart is a temporary state page that shows the anonymous user which items they have added to their order during their current visit to the website.
- In addition, anonymous users can change their shopping cart by both modifying the quantities of games in their cart when by add the game to shopping cart, as well as deleting games from their cart.
- The anonymous user will be able to view the total price of their order.
- Once the anonymous user hit purchase button, they will be directed to the order page.
- The anonymous user is suppose to enter their details, such as email, first name, last name, address and credit card number. Once hit submit they will be provided a confirmation number which will be unique for each purchase order.

c. New user

- The New user page will be for the users without account or those wishing to create new one. On this page, user is asked for their first name, last name, email, password and confirm password.
- If any of the required information is not inserted then an error message will be popped on the page. The user won't be allowed to make account unless they insert all the information.
- Once the user is successful making new account, they will be directed to the login page.
- The application layer will check the information the user enters thoroughly each time they submit. If all the data passes, the server will attempt to make the account and check for account creation errors. If no errors, then the database table of user will be updated with all the information.
- Additionally, the newuser page will invoke the login page php file.

d. Log in:

- The login page will be for the user with an existing account. Here the user will ask to input their user name in the form of email and password attached to that account.
- Similar to the New user page, if there is a missing information detected then the error message will pop up.

- The application layer will query the database with the information the user provides. If the information is valid for a user in the database then it will log that user in.
- Additionally, if user tries to login with username not present in the user table, the application layer will query the database and look for that username if not present in the table, the application layer will return the error message on the client side.

2. <u>Logged in user:</u>

If the user is logged in, the menu bar will consist of four different options of “Home”, “Shopping cart”,and “Log Out”.

a. Home Page

- The home page will function the same as mentioned above in the anonymous user section.
- The only difference for the home page when user is logged in is they can actually purchase the games they like. They can also input the number of games they would like to buy and add them to their shopping cart.
- The same function for adding the game to the shopping cart is applied to all the games which user see in the shop by category section.
- The application layer will invoke the shopping cart php file, which will query the database and grabs the game id which was selected by the client. Meanwhile the shopping cart will be updated and displayed on the client sides.

b. Shopping Cart

- The shopping cart is a temporary state page that shows the user which items they have added to their order during their current visit to the website.
- In addition, users can change their shopping cart by both modifying the quantities of games in their cart when by add the game to shopping cart, as well as deleting games from their cart.
- The user will be able to view the total price of their order.
- Once the user hit purchase button, they will be directed to the order page.
- There user is suppose to enter their details, such as email, first name, last name, address and credit card number. Once hit submit they will be

provided a confirmation number which will be unique for each purchase order.

c. Log Out

- The log out page is simply for the logged in user to log out the website once they finish visiting the website.
- The application layer will check if the user is logged in or not. If so, the section will get closed and user will be logged out and directly to the homepage of the anonymous user.

3. <u>Admin:</u>

a. Home page:

- In the admin page, the admin can see top 8 games and look around the website mostly the same as a user except the admin will have several options a normal user does not have.
- The admin will have access to addgame, change game price, delete game, and update game quantity.
- The application layer will query the database with the manager’s email and password. If successful the user will be directed to admin view
- Here the user won’t be able to buy any game.

b. Add new Game:

- The page requires the admin to insert the game name, game type, game year released, the name of the game developer, the game’s price and the stock quantity.
- Once the admin hits on add game the application layer will use the information inserted and update the game table with it.
- Once the update is successful the admin will be notified by the a message.

c. Change Price:

- This page will require the admin to input the game ID, game name and new price.

- The application layer will query the database with the inserted game ID and game name. If matched with any existing games in the game table then the price for that game will be changed in the database.

d. Change Stock:

- This page will require the admin to input the game ID, game name and new stock for that game.
- The application layer will query the database with the inserted game ID and game name. If matched with any existing games in the game table then the stock number entered will be added to the existing stock number in the database.

e. Delete Game:

- This page will require the admin to input the game ID and game name.
- The application layer will query the database with the inserted game ID and game name. If matched with any existing games in the game table then that game will be deleted from the database.

f. Log Out:

- The log out page is simply for the logged in user to log out the website once they finish visiting the website.
- The application layer will check if the user is logged in or not. If so, the section will get closed and user will be logged out and directly to the homepage of the anonymous user.

IV. Database Tables

The Database Tables section explains in detail the tables structure in the Database. This section breaks down the tables relationships, keys, table input types, and other details. Please refer to the E-R diagrams at the beginning of the document for a higher-level graphical description.

Users

Users (user_ID: INT(11), user_email: VARCHAR(40), user_fname: VARCHAR(20), user_lname: VARCHAR(20), user_pass: VARCHAR(20), num_purchased_games: INT(20), id_admin: TINYINT(1))

foreign keys: none
candidate key: user_fname, user_lname
primary key: user_ID
not null: user_ID, user_email, user_fname, user_lname, user_pass

This table stores all the information about the user on their id, email, first and last name, password, number of games purchased by the user and whether they are just a user or they have admin access. Using these, it helps the person to login and also keep track of the purchases made by them and helps with what view they are eligible to access, such as if they are an admin, they have a different webpage view. The primary key is the user_ID which is unique to every person.

Games

Games(game_ID: INT(11), game_name: VARCHAR(50), game_type: VARCHAR(20), year_released: DATE(YYYY-MM-DD), developer: VARCHAR(20), price: DOUBLE(5,2), stock_qty: INT(11), image: VARCHAR(40), num_sold: INT(5))

foreign keys: none
candidate key: game_name
primary key: game_ID
not null: game_ID, game_name, game_type, year_released, developer, price

This table stores the information on all the games that is on the database. It includes a unique game_ID which is the primary key. It has information on the game's id, name, type, the year it got released, the developer of the game, its price, quantity and the number of games that got sold so far. When a user is trying to purchase a game, information on its availability and the game type they belong to and other information such as its price are obtained from this table.

Orders

Orders(order_ID: INT(11), user_email: VARCHAR(50), game_ID: VARCHAR(20), order_date: DATE(YYYY-MM-DD), game_name: VARCHAR(20), order_confirmation: VARCHAR(40), user_ID: INT(11)

foreign keys: game_ID (Games.game_ID), user_ID (Users.user_ID)
candidate key: order_confirmation
primary key: game_ID
not null: game_ID, game_name, game_type, year_released, developer, price

This table stores the information on the orders made by the user. It consists of the user's email and their ID to identify the user and has information on the game's name and ID that they are purchasing. It further expands to the date of purchase, order id and a order confirmation number which acts as the candidate key as it generates unique combinations so that it does not repeat the same order confirmation number again. When a user makes a purchase, this table stores the information on the order that was placed by the user.

V. SQL Queries

Home Page Queries

Either when a signed in user or a guest user who is not signed in when visiting the home page they will see a top 8 games sold from our website store according to our database sales.

```
$product_array = $db_handle->runQuery("SELECT * FROM games
ORDER BY num_sold DESC");
```

Which calls the function in our dbController.php:

```
       function runQuery($query) {
        $result = mysqli_query($this->conn,$query);
        while($row=mysqli_fetch_assoc($result)) {
            $resultset[] = $row;
        }
        if(!empty($resultset))
            return $resultset;
    }
```

This collects the list of games that have sold the most and then sorts them in descending order. Therefor the front end developers can decide how many they want to display in what way.

If a user has anything in their cart signed in or not, the home page keeps the shopping cart while the user session is active. The users cart is saved to the session not to the database, but keeps a track of the game_id's and queries to keep the shopping cart updated for every page the user visits.

```
$productByCode = $db_handle->runQuery("SELECT * FROM games
WHERE game_ID = '" . $_GET["game_ID"] . "'");
```

Which again, calls the above function.

Action Page Queries

Similar to the home page, action page queries games by action type or similar type games for the user to browse. Also will collect all games and can be displayed however the front end developer wishes.

```
$product_array = $db_handle->runQuery("SELECT * FROM games
WHERE game_type = 'action' OR game_type = 'FPS' OR
game_type = 'Fighter' ORDER BY game_name");
```

Which calls the function in our dbController.php:

```
function runQuery($query) {
 $result = mysqli_query($this->conn,$query);
 while($row=mysqli_fetch_assoc($result)) {
     $resultset[] = $row;
 }
 if(!empty($resultset))
     return $resultset;
```

The returned results are sent back to the action page.php.

Fighting Page Queries

Similar to the home page, fighting page queries games by fighting type games for the user to browse. Also will collect all games and can be displayed however the front end developer wishes.

```
$product_array = $db_handle->runQuery("SELECT * FROM games
WHERE game_type = 'fighting'");
```

Which calls the function in our dbController.php:

```
function runQuery($query) {
 $result = mysqli_query($this->conn,$query);
 while($row=mysqli_fetch_assoc($result)) {
```

```
            $resultset[] = $row;
        }
        if(!empty($resultset))
            return $resultset;
```

The returned results are sent back to the fighter page.php.

Role-Playing Page Queries

Similar to the home page, role-playing page queries games by role playing type or similar type games for the user to browse. Also will collect all games and can be displayed however the front end developer wishes.

```
$product_array = $db_handle->runQuery("SELECT * FROM games
WHERE game_type = 'role-playing'");
```

Which calls the function in our dbController.php:

```
       function runQuery($query) {
        $result = mysqli_query($this->conn,$query);
        while($row=mysqli_fetch_assoc($result)) {
            $resultset[] = $row;
        }
        if(!empty($resultset))
            return $resultset;
```

The returned results are sent back to the roleplayingpage.php.

First Person Shooter Page Queries

Similar to the home page, FPS page queries games by FPS type or similar type games for the user to browse. Also will collect all games and can be displayed however the front end developer wishes.

```
$product_array = $db_handle->runQuery("SELECT * FROM games
WHERE game_type = 'FPS' OR game_type = 'action' OR
game_type = 'role-playing' ORDER BY game_name");
```

Which calls the function in our dbController.php:

```
        function runQuery($query) {
         $result = mysqli_query($this->conn,$query);
         while($row=mysqli_fetch_assoc($result)) {
             $resultset[] = $row;
         }
         if(!empty($resultset))
             return $resultset;
```

The returned results are sent back to the shooter.php.

Simulator Page Queries

Similar to the home page, simulator page queries games by simulator type or similar type games for the user to browse. Also will collect all games and can be displayed however the front end developer wishes.

```
$product_array = $db_handle->runQuery("SELECT * FROM games
WHERE game_type = 'simulator'");
```

Which calls the function in our dbController.php:

```
        function runQuery($query) {
         $result = mysqli_query($this->conn,$query);
         while($row=mysqli_fetch_assoc($result)) {
             $resultset[] = $row;
         }
         if(!empty($resultset))
             return $resultset;
     }
```

The returned results are sent back to the simulator.php.

Sports Page Queries

Similar to the home page, simulator page queries games by sports type games for the user to browse. Also will collect all games and can be displayed however the front end developer wishes.

```
$product_array = $db_handle->runQuery("SELECT * FROM games
WHERE game_type = 'sports'");
```

Which calls the function in our dbController.php:

```
        function runQuery($query) {
        $result = mysqli_query($this->conn,$query);
        while($row=mysqli_fetch_assoc($result)) {
            $resultset[] = $row;
        }
        if(!empty($resultset))
            return $resultset;
    }
```

The returned results are sent back to the simulator.php.

New User Queries

The new user page is used to create a new user and needs to check the database to see if there is exists a user already with the email the user provides as to not have duplicate email addresses. If not then the newly created user can be inserted into the database. The newly created user is required to have a first name, last name, email, and password. Otherwise the information will not be inserted into the database.

Our SQL prepared statement:
```
$sql = "SELECT user_ID FROM user WHERE user_email = ?";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql);
```

Our SQL execution statement:
```
mysqli_stmt_execute($stmt);
```

Then results are checked if the email exists, if not then we can proceed with the following insert.
```
$sql = "INSERT INTO user (user_email, user_fname,
user_lname, user_pass, num_purchased_games, is_admin)
VALUES (?, ?, ?, ?, 0, 0)";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql);
```

Executing the query:
```
$result = mysqli_query($conn, $stmt);
```

Once all statements are executed successfully, there will be a new user in the database and the user will be able to sign into their account.

Log In Queries

The login page is used to log user into the website, which needs to check if the user trying to sign in is actually a user. The query will check if what the user types in matches what is in the database, otherwise the user cannot sign into their account. The user will enter their email and password.

Our SQL prepared statement:
```
$query = "SELECT user_email, user_fname, user_pass,
is_admin FROM user WHERE user_pass =
'".$_POST["user_pass"]."' AND user_email
='".$_POST["user_email"]."'";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql);
```

Our SQL execution statement:

```
$result = mysqli_query($conn, $query);
```

Then results are checked if the email exists, if not then we can proceed with the following insert.

```
$query = "SELECT user_email, user_fname, user_pass, is_admin FROM user WHERE user_pass = '".$_POST["user_pass"]."' AND user_email ='".$_POST["user_email"]."'";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql);
```

Executing the query:

```
$result = mysqli_query($conn, $stmt);
```

Once all statements are executed successfully, there will be a new user in the database and the user will be able to sign into their account.

Manager Add Game Queries

The manager page has a add game selection for managers to add new game products they want to sell on the website easily. It accepts a game name, price, developer, year released, initial stock quantity, game type, and game image. Just like new user we need to see if this game is already in the database, so first we need to check if the game already exists before putting a duplicate game into the database.

Our SQL prepared statement:

```
$sql = "SELECT game_name FROM games WHERE game_name = ?";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

If the game is not in the database we can go ahead and insert:

```
$sql = "INSERT INTO games (game_name, game_type, year_released, developer, price, stock_qty, image, num_sold) VALUES (?, ?, ?, ?, ?, ?, ?, 0)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, there will be a new game in the database and users will be able to see a new game.

Manager Change Game Price Queries

The manager page has a change game price selection for managers to change current game product prices if they need to do a sale or just mark a game product down or up. It accepts a game name, game_id, and new price. If the game does not exist the manager will get an error that the game does not exist. Otherwise, the manager can execute the change price query.

Our SQL prepared statement:

```
$sql = "SELECT game_name FROM games WHERE game_name = ? && game_ID = ?";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

If the game exists, then it can continue to change the price of the game:

```
$sql = "UPDATE games SET price=? WHERE (game_ID = ? && game_name = ?)";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:
```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the game that the manager wanted to update its price to will updated.

Manager Update Game Quantities Queries

The manager page has a change game quantities for managers to change current game product quantities if they receive a shipment. Negative numbers can even be input to clear out current quantity if an error was made counting. It accepts a game name, game_id, and new quantity. If the game does not exist the manager will get an error that the game does not exist. Otherwise, the manager can execute the change price query.

Our SQL prepared statement:
```
$sql = "SELECT game_name, stock_qty FROM games WHERE game_name = '".$_POST["game_name"]."' && game_ID = '".$_POST["game_id"]."'";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:
```
$result = mysqli_query($conn, $stmt)
```

If the game exists, then it can continue to change the price of the game:
```
$sql = "UPDATE games SET stock_qty=? WHERE (game_ID = ? && game_name = ?)";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:
```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the game that the manager wanted to update the game stock quantity.

Manager Delete Queries

The manager page has a delete a selected game from the database if it exists. It accepts a game name, and game_id. If the game does not exist the manager will get an error that the game does not exist. Otherwise, the manager can execute the query and delete the game.

Our SQL prepared statement:
```
$sql = "SELECT game_name FROM games WHERE game_name = ? &&
game_ID = ?";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:
```
$result = mysqli_query($conn, $stmt)
```

If the game exists, then it can continue delete the game:
```
$sql = "DELETE FROM games WHERE (game_ID = ? && game_name =
?)";
```

Combining to to our database connection:
```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:
```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the game that the manager wanted to deleted will be deleted from the database and no longer part of the game store available games.

Purchase Queries

Once the user signed in or not wants to complete a purchase on the website it takes the current shopping cart game IDs to use to query the database to collect those games to finalize the sale. There are several different queries in the purchase page. During a purchase we need to figure out if a purchase is being completed by a user with the given email address. Because we want to know what users purchase what games and or type later. Also, we want to later reward users who purchase games with discounts determined by how many games they purchase. So the purchase page is executed in two different directions, one records if the purchase is made by a user or if its by a non account user by email.

Additionally, the purchase page has to make sure the game quantity is greater than 0, meaning its in stock. If not the user is told it will be in stock soon and cancels the order.

Our SQL prepared statement from our shopping cart:

```
$sql = "SELECT game_name, stock_qty, num_sold FROM games
WHERE game_name = '$game_name' && game_ID = '$game_id'";
```

Used to see if the user purchasing is a account holder:

```
$sql5 = "SELECT * FROM user WHERE user_email =
'$user_email'";
```

Updating game stock after purchase:

```
$sql = "UPDATE games SET stock_qty=?, num_sold = ? WHERE
(game_ID = ? && game_name = ?)";
```

Inserting into the orders table:

```
$sql2 = "INSERT INTO orders (user_email, game_ID,
order_date, game_name) VALUES (?, ?, ?, ?)";
```

Used to update the user account if the user purchasing user is a account holder:

```
$sql4 = "UPDATE user SET num_purchased_games =? WHERE (user_email = ?)";
```

Used to update the final purchase confirmation code for the user:

```
$sql3 = "UPDATE orders SET order_confirmation=? WHERE (user_email = ? && order_date = ?)";
```

Combining to to our database connection:

```
$stmt = mysqli_prepare($link, $sql)
```

Executing the query:

```
$result = mysqli_query($conn, $stmt)
```

Once all statements are executed successfully, the game that the manager wanted to deleted will be deleted from the database and no longer part of the game store available games.

Overall, for implementing this project, a total of 27 SQL queries was used.

VI. Database Initialization Queries

The following four SQL queries are used to initialize the database tables:

```
     CREATE TABLE IF NOT EXISTS `games` (
       `game_name` varchar(50) NOT NULL,
       `game_ID` int(11) NOT NULL AUTO_INCREMENT,
       `game_type` varchar(20) NOT NULL,
       `year_released` date NOT NULL,
       `developer` varchar(20) NOT NULL,
       `price` double(5,2) NOT NULL,
       `stock_qty` int(11) NOT NULL,
       `image` varchar(40) NOT NULL,
       `num_sold` int(5) NOT NULL,
       PRIMARY KEY (`game_ID`)
)ENGINE=InnoDB AUTO_INCREMENT=41 DEFAULT CHARSET=latin1;

     CREATE TABLE IF NOT EXISTS `orders` (
       `order_id` int(11) NOT NULL AUTO_INCREMENT,
       `user_email` varchar(40) NOT NULL,
       `game_ID` varchar(20) NOT NULL,
       `order_date` date NOT NULL,
       `game_name` varchar(20) NOT NULL,
       `order_confirmation` varchar(40) DEFAULT NULL,
       PRIMARY KEY (`order_id`)
) ENGINE=InnoDB AUTO_INCREMENT=37 DEFAULT CHARSET=latin1;

     CREATE TABLE IF NOT EXISTS `user` (
       `user_ID` int(11) NOT NULL AUTO_INCREMENT,
       `user_email` varchar(40) NOT NULL,
       `user_fname` varchar(20) NOT NULL,
       `user_lname` varchar(20) NOT NULL,
       `user_pass` varchar(20) NOT NULL,
       `num_purchased_games` int(20) NOT NULL,
       `is_admin` tinyint(1) NOT NULL,
       PRIMARY KEY (`user_ID`)
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=latin1;
```