Maxwell Maia

St David’s Marist Inanda 2020

PAT Phase 2

Fire reviews

Table of Contents

[User Interface Design 3](#_Toc52028296)

[Sequencing 14](#_Toc52028297)

[Class design 16](#_Toc52028298)

[Persistent storage design 22](#_Toc52028299)

[Explanation of storage design 24](#_Toc52028300)

# User Interface Design

Login screen



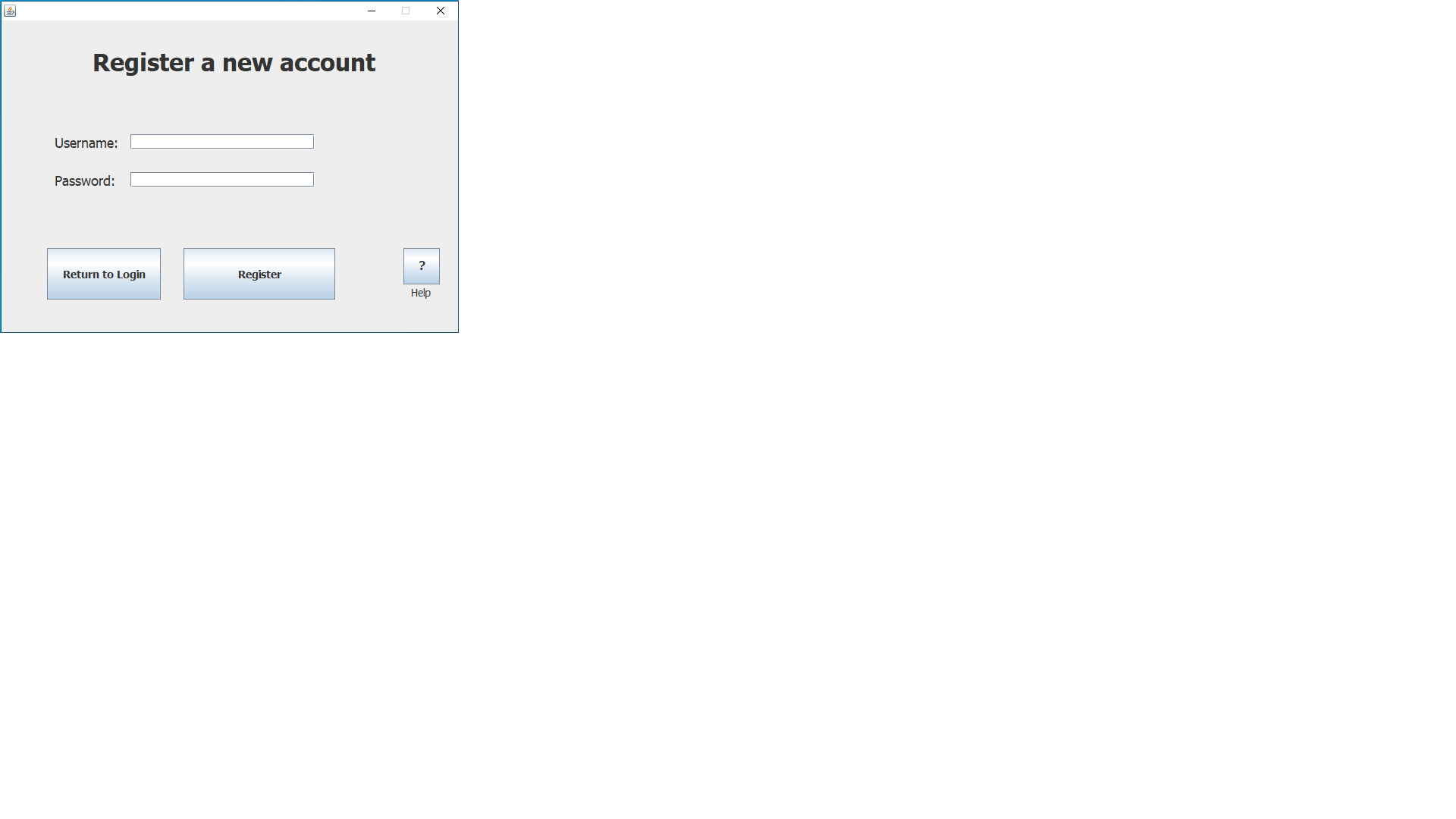
Security

All users have access to this screen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data entered | Data type |  | Data displayed | Data type |
| Username | String |  | NA | NA |
| Password | String |  | NA | NA |

|  |  |
| --- | --- |
| Action element | Action element function |
| Username - text field | The user enters their username. |
| Password field | The user enters their password. |
| Login - button | The program checks if the user’s entered credentials are valid. If they are, the user will “login” and the screen will change to the Main Application Screen. |
| Create new account - button | Changes screen to the Register Screen so that first time users can make their accounts. |
| Help – button | Opens the Help screen. This tells users to log in or create a new account. |

Register screen



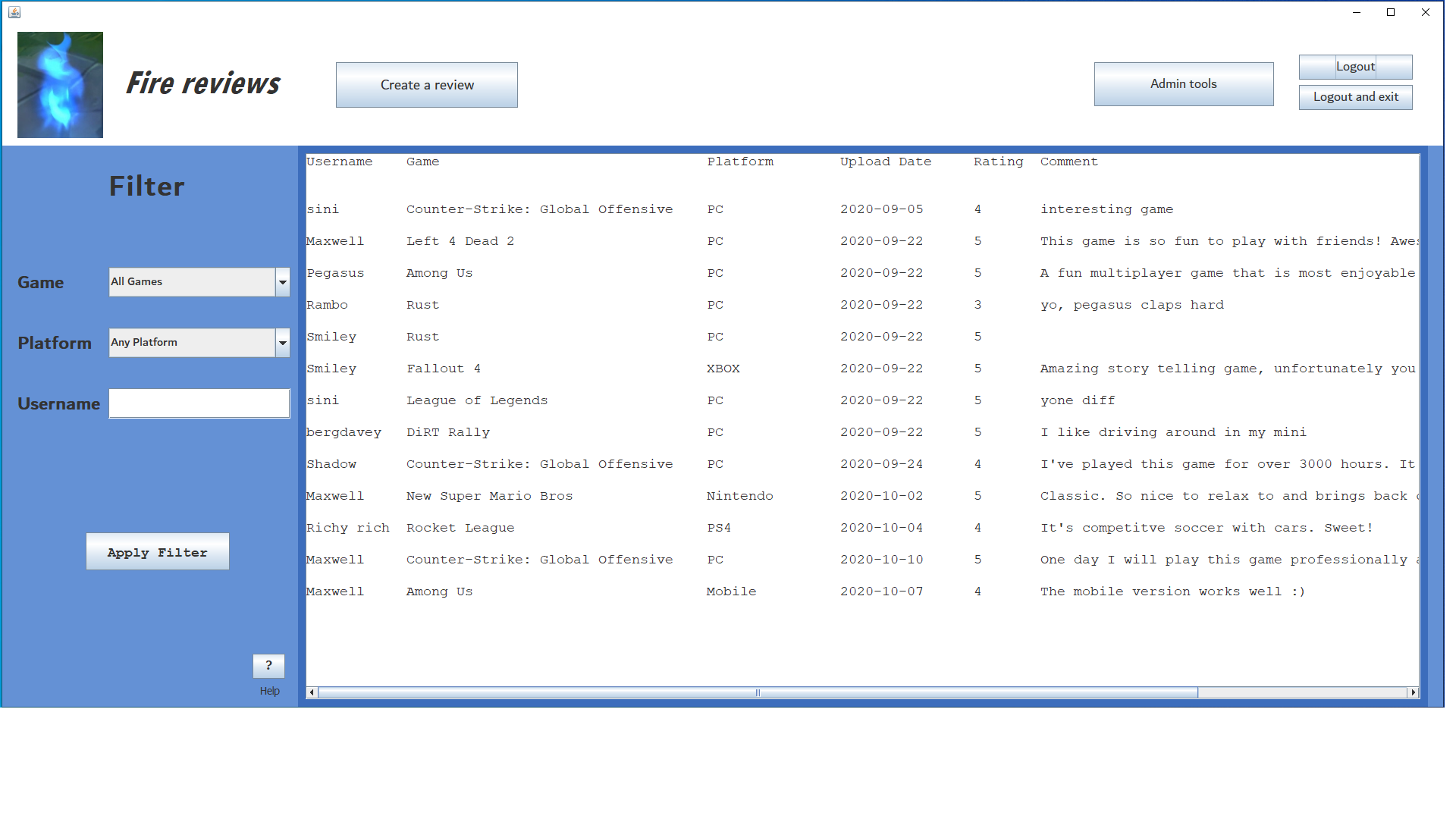
Security

All users have access to this screen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data entered | Data type |  | Data displayed | Data type |
| Username | String |  | NA | NA |
| Password | String |  | NA | NA |

|  |  |
| --- | --- |
| Action element | Action element function |
| Username - text field | The user enters their desired username. |
| Password field | The user enters their desired password. |
| Register - button | The program checks if the user’s entered credentials are valid. If they are, the user will “register”, generating a new user account. |
| Return to login - button | The user clicks this to return to the Login Screen. |
| Help – button | Opens the Help screen showing help information relating to this screen. |

Main app



Reviews

Security

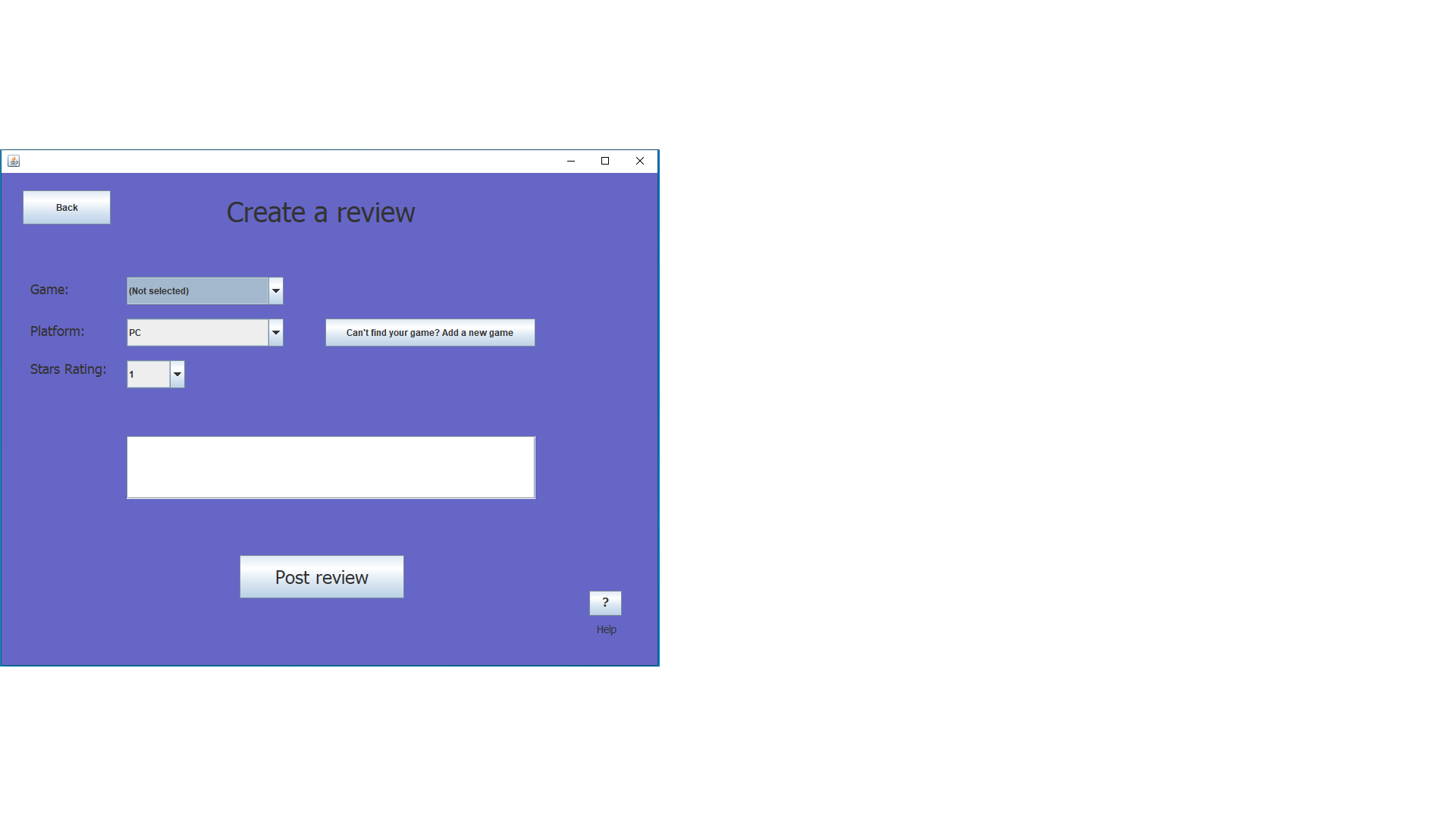
If a user has an account and they are logged in they have access to this screen.

The “admin tools” button is only visible to admin accounts.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data entered  Data is entered to perform a filter sort. This will filter all of the reviews. | Data type |  | Data displayed  All reviews are displayed. | Data type |
| Game to filter by | String (Combo box) |  | Username | String |
| Platform to filter by | String (Combo box) |  | Game name | String |
| Username to filter by | String (text field) |  | Platform | String |
|  |  |  | Upload date | String |
|  |  |  | Rating | Integer |
|  |  |  | Comment | String |

|  |  |
| --- | --- |
| Action element | Action element function |
| “Create a review” – button | Opens the Create a review screen. Here the user posts their own review of a game. |
| Combo boxes  (platform, game) | All of the games in the database are loaded into the “game” combo box upon generating the main screen. The platform combo box is already configured. The user chooses a platform and/or game if they wish to filter the reviews.  “All games” and “Any platform” are the default values. If the user sorts by these values every review is displayed. |
| Username (filter) – textfield | Enter a username to filter reviews by.  The text field is empty by default. If the user filters with an empty text field then the reviews with made by every user is displayed. |
| Apply filter – button | Changes the displays reviews to the reviews that match the filter options the user has set. |
| Admin tools – button | Takes user to the Admin tools screen. This button is only visible to admin accounts. |
| “Logout and exit” – button | When the user clicks this the program is closed. |
| “Logout” – button | The user “logs out” and returns to the Login Screen. |
| “Help” – button | Opens the Help screen. This shows help information regarding the main application screen. |

Create a review



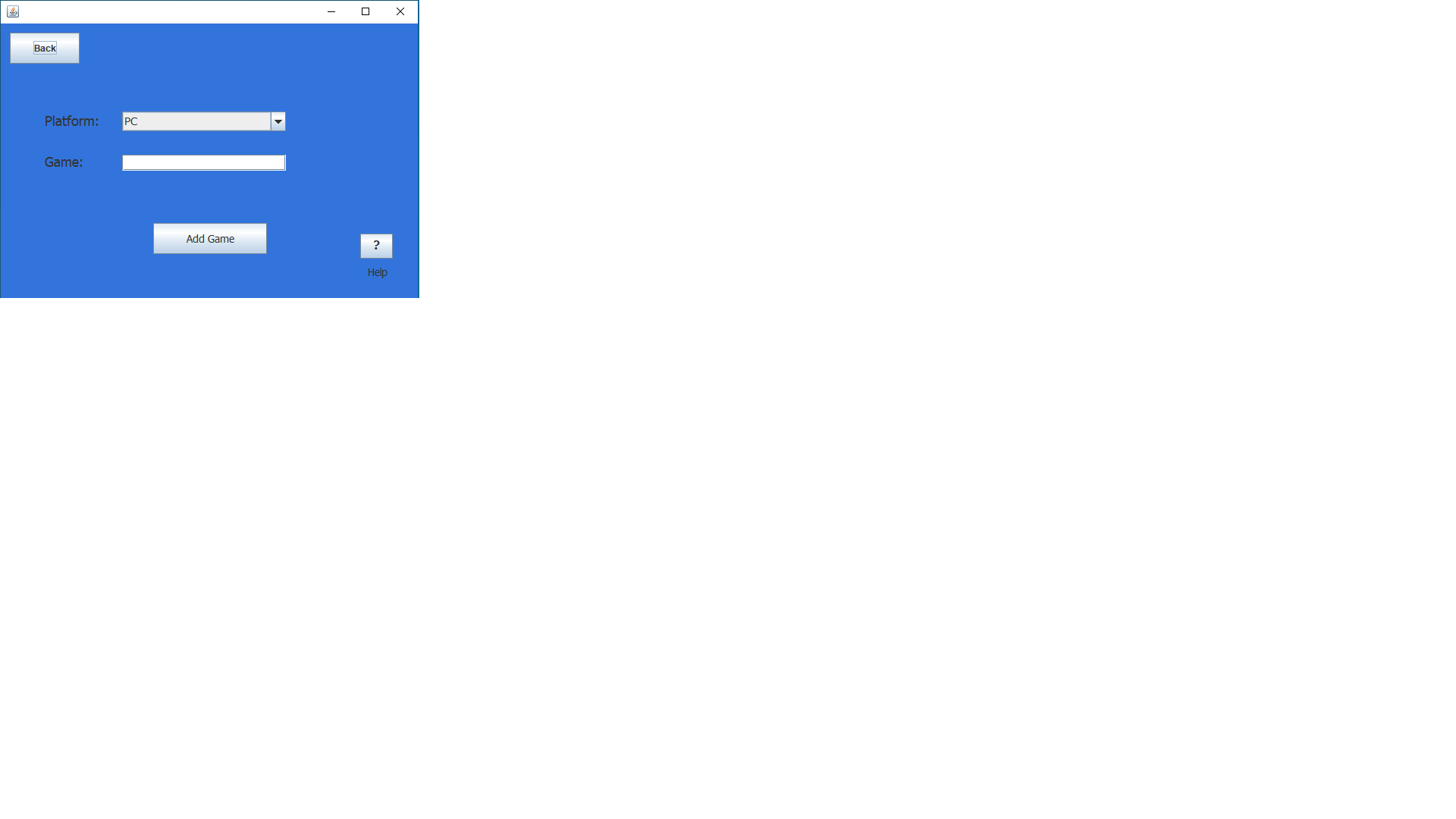
Comment:

Security

All users have access to this screen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data entered | Data type |  | Data displayed | Data type |
| Game name | Combo box - string |  | NA | NA |
| Platform name | Combo box - string |  | NA | NA |
| rating | Combo box - integer |  | NA | NA |
| comment | text area - string |  | NA | NA |

|  |  |
| --- | --- |
| Action element | Action element function |
| “Post review” – button | Once the user is ready to post their review, they click this button. This adds the review to the database if there are no errors with the inputs the user has given. |
| Combo boxes  (platform, game) | All of the games in the database will be loaded into the “game” combo box upon entering the screen. The “platform” combo box is already configured. The user chooses the platform and game they are making the review for.  “(Not selected)” is the default value for “Game”. If the user tries to make the review without selecting a game an error message is displayed. Additionally, if the game selected is not available on the platform selected an error message is displayed. |
| Add a new game – button | The user clicks this to add a new game to the database if the game they are looking for has not been added yet. This button takes the user to the Add game screen. |
| Rating – combo box | The user chooses the rating of the review of the game by selecting a number from the combo box (1-5). |
| Comment box – text area | The user types the comment for the review. |
| “Help” – button | Opens the Help screen. This shows help information regarding the “create a review” screen. |

Add game

Add a game

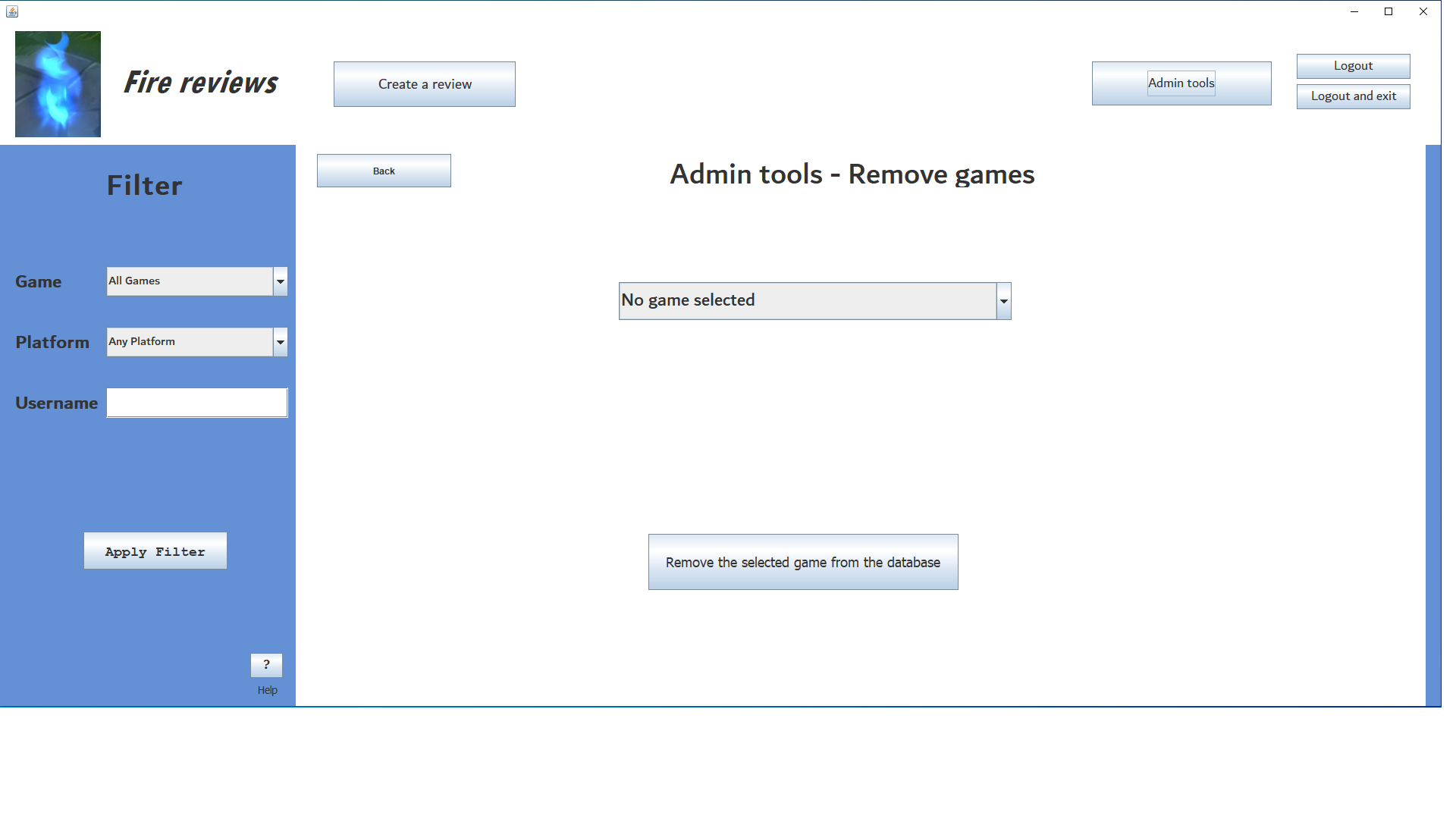
Security

All users have access to this screen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data entered | Data type |  | Data displayed | Data type |
| platform | Combo box - string |  | NA | NA |
| Game name | Text field - string |  | NA | NA |

|  |  |
| --- | --- |
| Action element | Action element function |
| Combo box  (platform) | The “platform” combo box is already configured. The user chooses the platform of the new game they are adding. |
| “Game” - textfield | The user enters the name of the new game. |
| “Add game” – button | The user clicks this to add a new game to the database if the game they are looking for has not been added yet. A message is displayed to tell the user whether the game was successfully added to the database or not.  This screen then closes. |
| “Help” – button | Opens the Help screen. This shows help information regarding the “Add game” screen. |

Admin tools



Security

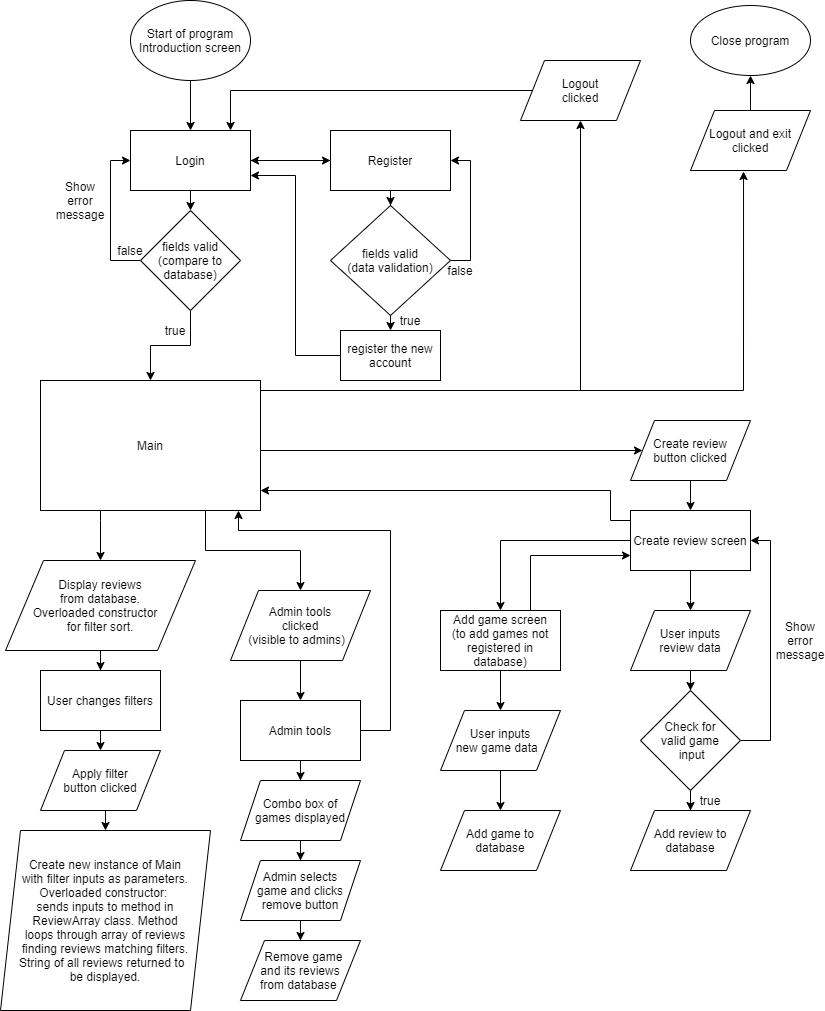
Only admins have access to this screen.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data entered | Data type |  | Data displayed | Data type |
| Game | Combo box - string |  | NA | NA |

|  |  |
| --- | --- |
| Action element | Action element function |
| Combo box  (game) | All of the games are loaded into this combo box from the database when the main screen is generated.  The admin chooses the game to delete from the combo box. |
| Remove game – button | The selected game is deleted from the database. This also deletes all reviews made on this game from the database. |

# Sequencing





# Class design

User

|  |  |
| --- | --- |
| Attributes | Description |
| * uID : integer * username : string * password : string * admin : boolean | User’s ID.  Username to login.  Password to login.  Whether the user is an admin or not. |
| Methods  + Constructor (uID : integer, username : string, password : string, admin : boolean)  + getuID() : integer  + getUsername() : string  + getPassword() : string  + getAdmin() : boolean  + toString() : string | Assign the values to the variables.    Get the User’s ID.  Get the username.  Get the password.  Get the boolean for admin. (Is user an admin?)  Returns a string showing all attributes. |

UserArray

|  |  |
| --- | --- |
| Attributes | Description |
| * uArray[] : User * size : integer * uID : integer | Array for user objects.  Amount of users in uArray.  Used to identify user when working with the database. |
| Methods  + Constructor ()  + Constructor(uIDInput : integer)  + testCredencials : boolean    + getAdminStatus()  + getUID()  + getuArray()  + getuSize() | Compiles array of users from database using “while loop.”  Same as default constructor but accepts UserID. UserID is used to get the admin boolean for user logged in.  Checks if username and password matches the username and password in the database.  Returns the admin boolean of user according to the UserID. (Parameterized constructor).  Returns the user ID.  Returns the array of users.  Returns size variable (size of uArray). |

Game

|  |  |
| --- | --- |
| Attributes | Description |
| * number: integer * name : string * platform : string | Essentially the game ID.  Name of the game.  Name of the platform the game is played on. |
| Methods  + Constructor (number : integer, name : string, platform : string)  + getNumber() : integer  + getName() : String  + getPlatform() : String  + toString() : String | Assign the values to the variables.  Get the number of the game.  Get the name.  Get the platform.  Returns a string showing all attributes. |

GameArray

|  |  |
| --- | --- |
| Attributes | Description |
| * gArray[] : Game * size : integer * distSize : integer     + data : DB | Array of games.  Amount of games in array.  Distinct amount of games (does not count duplicates that occur in database when there is a game available on multiple platforms).  Object to perform database queries. |
| Methods  + Constructor ()  + distinctNames() : Game  + getgArray() : Game  + getDistArray() : Game  + getSize() : integer  + getDistSize() : integer  + toString() : string | Compiles array of user’s from database using “while loop.”  Fill array with distinct game names only.  Get the gArray.  Get the distArray.  Gets size (size of gArray).  Gets distSize (size of distArray).  Returns a string showing all attributes of each element in the gArray. |

Review

|  |  |
| --- | --- |
| Attributes | Description |
| * rID : integer * uID : integer * gID : integer * rating : integer * df : DateFormat * dateObj : Date * comment : string | Review ID.  User ID.  Game ID.  Rating given by user 1-5 stars.  To format the dateObj (MM/dd/yyyy).  Stores the date the review is made.  The comment of the review. |
| Methods  + Constructor (rID : integer, uID : integer, gID : integer, dateObj : Date, rating : integer, comment : string)  + getrID() : integer  + getuID() : integer  + getgID() : integer  + getRating() : integer  + getDateObj() : Date  + getComment() : string  + toString() : String | Assign the values to the variables.  Returns review ID.  Returns user ID.  Returns game ID.  Returns rating.  Returns Date object.  Returns comment.  Returns a string showing all attributes. |

ReviewArray

|  |  |
| --- | --- |
| Attributes | Description |
| * rArray[] : Game * rSize : integer * dateObj : Date * df : DateFormat * ga : GameArray() * gSize : integer * fullgArray[] : Game * ua : UserArray() * uSize : integer * uArray[] : User * username : string * gameName : string * platform : string   + data : DB() | Array of reviews.  Amount of reviews in array.  Stores the date the review is made.  To format the dateObj (MM/dd/yyyy).  GameArray object used to get game array.  Amount of games in array.  Array of all games including duplicates.  UserArray object used to get user array.  Amount of user in array.  Array of all users.  Object to perform database queries. |
| Methods  + Constructor ()  + getSize() : integer  + getrArray() : Review  + addSpaces(input : string, desiredSpaces : integer)  + toString() : string  + toStringSort(gameIn : string, platforming : string, usernameIn : string) : string | Compiles array of reviews from database using “while loop.”  Get the gArray.  Get the review array.  Returns a string of spaces that will add to to the desiredSpaces character length (input + spaces = desiredSpaces). The point of this is to make the total length of a set of strings all the same.  Returns the string to be printed out showing the reviews on the main screen.  Returns string used to print out the SORTED reviews on the main screen. |

SQLSTatements

|  |  |
| --- | --- |
| Attributes | Description |
| * data : object of DB class * dateObj : Date * df : DateFormat | Stores the date the review is made.  To format the dateObj (MM/dd/yyyy). |
| Methods  + ReadToLogin(u : string, p : string)  + InsertReview(userID : integer, gameID : integer, rating : integer, comment : string)  + AddGame(gameName : string, platform : string)    + deleteGame(  gameToDeleteInput : string, gameToDeleteID : integer) | Inserts the username and password into database (registration).  Inserts data for new review into database  (create new review).  Inserts data for a new game and its platform into database (add game).  Deletes a game from Game table. Deletes all reviews that are linked to that game. |

DB

|  |  |
| --- | --- |
| Attributes | Description |
| * conn : connection * stmt : Statement |  |
| Methods  + Constructor ()  + queryTb(sqlStmt : string) : ResultSet  + updateTbl(update : string) | Locates and loads database.  To retrieve data from database (SELECT statement).  Changes something in the database (DELETE/UPDATE/INSERT). |

ErrorMsg

|  |  |
| --- | --- |
| Attributes | Description |
| * message : String | The message of the error. |
| Methods  + Constructor (m : String)  + showErrorMsg(m: string) | Sets the message to the error message in the constructor parameter.  Displays an information box with error message. |

Help

|  |  |
| --- | --- |
| Attributes | Description |
| * helpID : integer * message : String * data : object of DB class | Used to get the help message needed.  Stores the message. |
| Methods  + Constructor (hID : integer) | Fetch help message from database that relates to the hID. The hID is hardcoded for each help button.  Creates a help screen with the help message on it. |

# Persistent storage design

**User**

Field Type \_

UserID Autonumber (Primary key)

username Short text

password Short text

Admin Yes/No

**Games**

Field Type \_

GameID Autonumber (Primary key)

GameName Short text

Platform Short text

**Reviews**

Field Type \_

ReviewID Autonumber (Primary key)

UserID Number (Foreign key)

GameID Number (Foreign key)

uploadDate Date/Time

Rating Number

Comment Long text

**Help**

Field Type \_

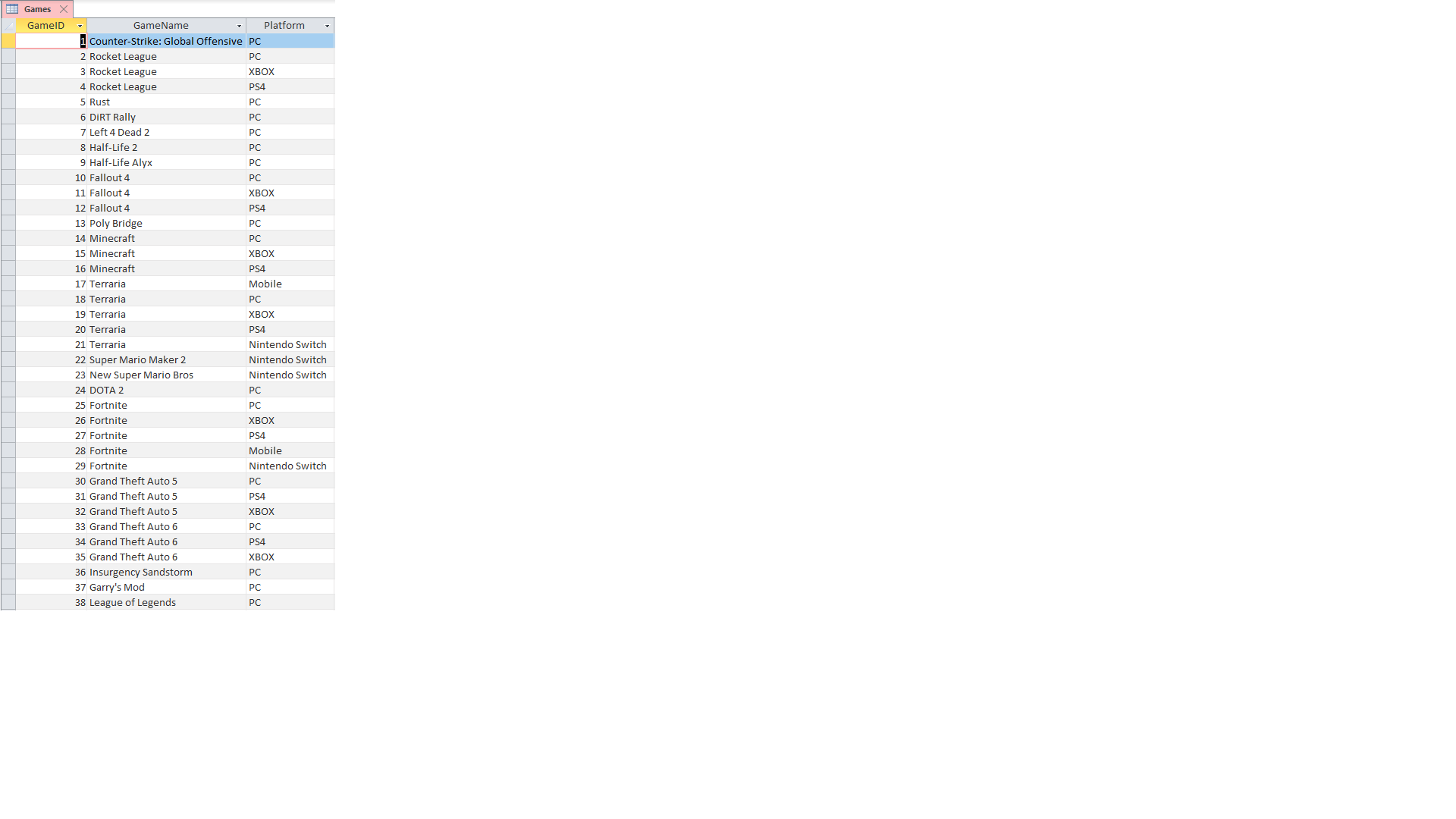
HelpID Autonumber (Primary key)

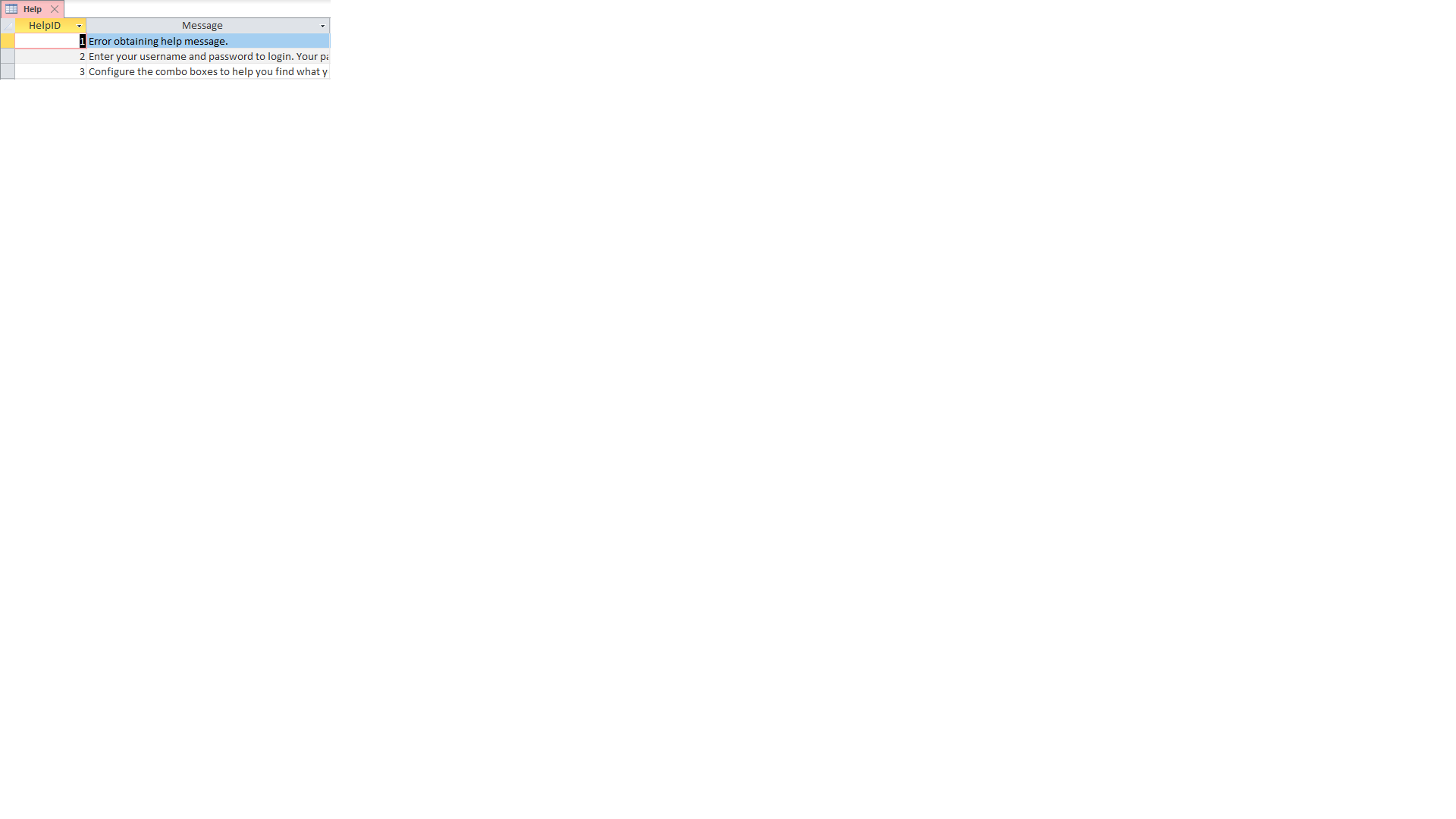
Message Short text

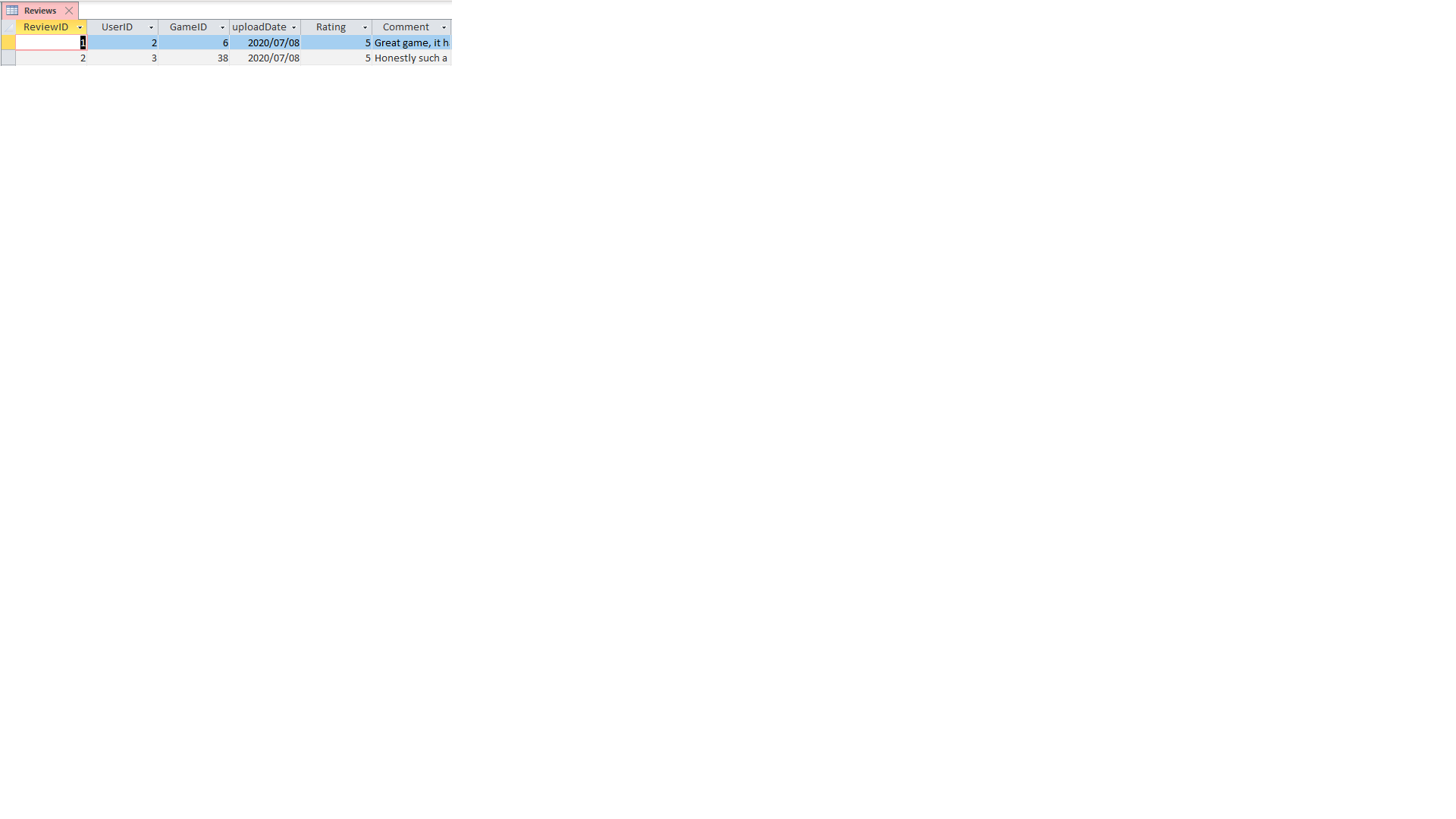


A close up of a logo

Description automatically generatedSample data







# Explanation of storage design

No text files were used because everything that had to be done could easily be accomplished using a database as the storage medium.

Game id’s, names and their platforms are stored permanently in the database. The names of games that are available on multiple platforms are duplicate data. Because of this, the program puts the data from the Games table into 2 arrays.

- One array includes the duplicate names. This is used to filter by platform.

- The second array contains only unique names. This second array is used to provide a list of games in the combo box that sorts reviews.

Reviews are also stored permanently in the database. This data is taken into primary memory in an array. During the execution of the program the reviews are then displayed. The array of reviews can now also be filtered and then displayed.

The userID is stored in primary memory so that it can be used to identify the creator of a new review. This is also used to determine if the user is an admin or not.

Help information is stored permanently in the database. This data will not change throughout the course of the program’s execution. It is only read from when the user clicks a help button. This also means that if help information needs to be changed for any reason, only the database would need to be updated. This is particularly useful if the database is stored online because then the programmer would not need to release an updated version of the program to change the help information.