Application: Gamer’s Log

Christian Tuton  
Elias Flores

Mark Harris

# 1. Background

Our theme was chosen because we feel that current game critiquing websites are dissatisfactory. We would like to make a website that allows users to track video games that they and their friends have played, and review and rate those games. We would like users to have access to a database of video game entries, but also be able to define their own games if they aren’t present in the database. Time permitting, we would also like users to be able to blog about past or future game releases.

# 2. Website Analysis

This section describes what functions we would like to implement in our website. It also describes existing websites and their functionality. Through this section we would like to show how our website differs from current websites and also demonstrate the important functions of our website.

## 2.a Existing Websites

Below we have listed 5 existing websites that are relevant to the theme and have described what functions the websites have.

1. IGN — a well known video game database critiquing website. The scores are a heterogeneous mix of critic and community reviews, with its main focus is centered around news. We would like to make a website that is similar to IGN, but is more community focused, and without the critic reviews (and a little less cluttered).
2. Metacritic — another well known video game database, but is more focused on score. The site also extends to other forms of media, such as movies. We take inspiration from metacritic, but want it to exclusively feature video games, and be less version dependant.
3. Gamespot — gamespot, like IGN, is focused more on news than reviews, but does have a vast database and a decent scoring system, which is where most of the similarities between our websites end. Like IGN, we want our website to be more community focused, and not have “paid off” reviews.
4. TheGamesDB — possibly the most similar to our project, TheGamesDB features a large database of games, with comments and reviews available, and the ability to add games, and search for them.
5. MyAnimeList — This website is not about video game critique, but it is similar in style for how we want the website to look/function, with the ability to make friends, change your profile, add items to your own personal list, an advanced sorting system, and a nice way to view the top movies/shows with relative ease.

## 2.b Website Function

Below we have listed the functions we would like our website to have, in different sections related to how much time we devote to the project. These sections are “Minimum” (the basic capabilities we will ensure are implemented), “Ideal” (capabilities we will prioritize and attempt to implement), and “Stretch Goals” (capabilities we probably won’t be able to implement, but would like to if possible).

### 2.b.i Minimum

* User Login — we would like users of our website to be able to login to their profiles. This information would be linked to any input they supply to the website. We predict we will need to use a database to hold usernames and passwords, as well as any information added by the users.
* Add games — we would like users to be able to add games to this website, given that the game they want to add is not already existing on the website. In order to implement this, we anticipate that we’ll need to store game titles and other game information in a database.
* Browse games — we would like to display a large amount of existing game information on our website for users to browse through. When a user selects a game from this list, the website would display the game rating, reviews, and general information.
* Rate games — we would like users to be able to rate existing games on the website. We hope to also have a general rating for each game, which would be an average of all user rates for that game.
* Review games — we would like users to be able to write reviews for games on our website. This would most likely take the form of comment feeds on games, and would include the user’s rating of that game.

### 2.b.ii Ideal

* Searching — we would like users to be able to search for a specific game stored on the website. We would probably implement game name searching first, and then possibly user searching as well so users could see what games other users like to play. If time allows, we may implement an advanced search system, which allows the user to filter certain data, such as genre or minimum rating.
* Sorting — we would like users to be able to sort the information on the website, so they can find and parse information easier.
* Social (friends) — we would like users to be able to add and communicate with friends through our website. This would allow users to see what their friends are playing and set up online game meetings.
* Profile management — we would like users to be able to manage information about themselves on their profile. This includes a list of games they like playing or have beaten, profile pictures that will appear next to their reviews of games, and player descriptions.

### 2.b.iii Stretch Goals

* News — if possible we would like to implement gamer news that keeps users in the loop of new developments in the game industry and release dates or prominent reviews for games.
* Forums — if possible we would like users to be able to communicate with each other over forum threads, which could be initiated by users about a specific game or topic.

## 2.c Comparison Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Proposed System** | **IGN** | **Metacritic** | **Gamespot** | **TheGamesDB** | **MyAnimeList** |
| **Login** | V | V | V | V | V | V |
| **Profile Management** | V | V | V | V | V | V |
| **User-Defined Games** | V | X | X | X | X | X |
| **User-Rated Games** | V | X | V | V | X | X |
| **User Reviews** | V | X | V | V | X | V |
| **Forums** | V | V | X | V | X | V |
| **Walkthroughs** | X | V | X | V | X | X |
| **Searching/Sorting** | V | V | V | V | V | V |
| **News** | V | V | V | V | X | V |
| **Trailers** | X | V | V | V | X | V |

# 3. Storyboard

This section contains descriptions of the users of our website, and how we hope to accommodate their needs. We first describe the needs of each user, and then include 3 sets of storyboards to show how each user would interact with the website (we have avoided repetition/overlap for each user, so even though some users have the same capabilities we may not have shown that in the storyboards). We have also included the differences between the computer monitor view and smartphone views.

## 3.a. Users

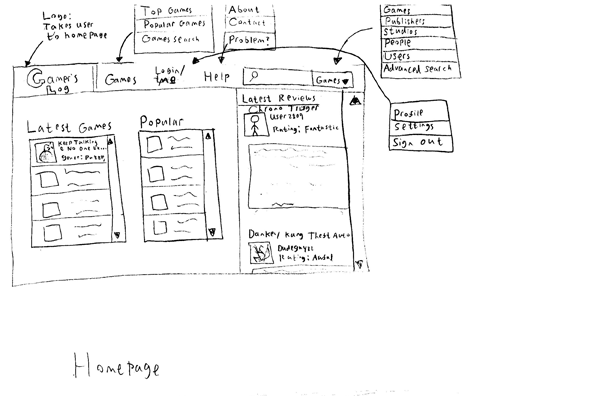
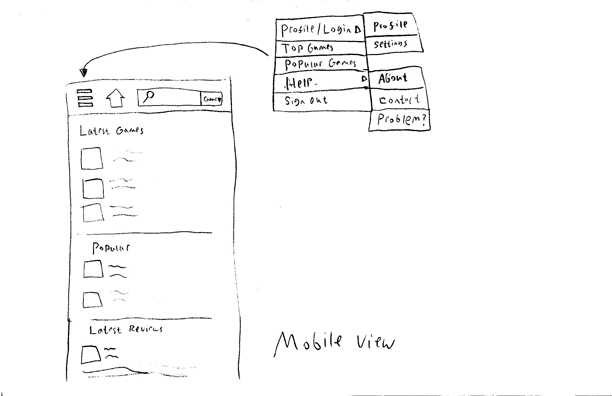
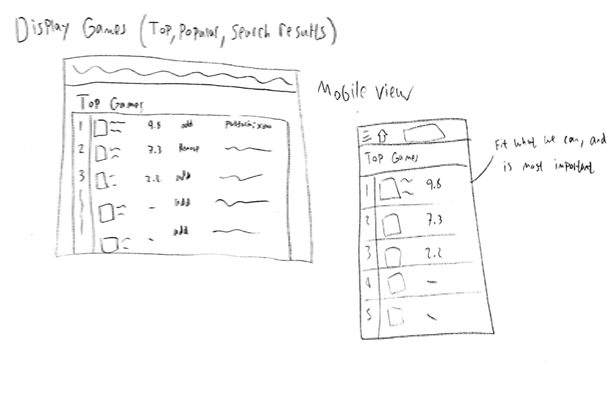
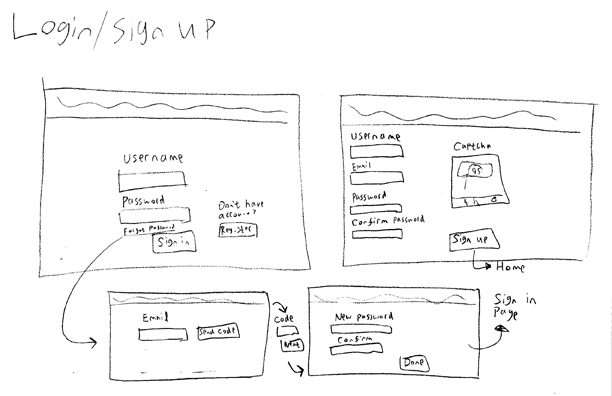
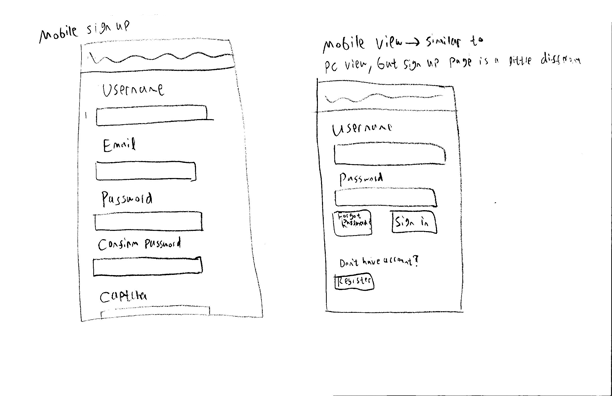
We have chosen three users to focus on for our website. These users are Guest, Registered, and Admin. The roles for each user are described below. The Admin user will probably be implemented last, or not implemented at all, since they aren’t the focus of our website.

* Guest — the Guest user is a user that does not have an account but is still using the website. We would like this user to be able to browse the data on the website, however not be able to interact with or otherwise add information to the website.
* Registered — the Registered user is a user that has an account, which includes a profile with data on what games they have played, what games they liked or disliked, reviews and ratings of games, and a list of people they are following (e.g. their “friends”). This user should have full access to add content to the website, meaning they can add games, ratings and reviews, and will be able to request other Registered users to be banned from the site (to be reviewed by an Admin user).
* Admin — the Admin user is a user with the capabilities to control all content in the website, including user information. They are able to remove games added by Registered users if they deem those games to be fraudulent or non-existent, and they are also able to ban users from the website (i.e. remove a user’s profile) if they feel the user is abusing the website.

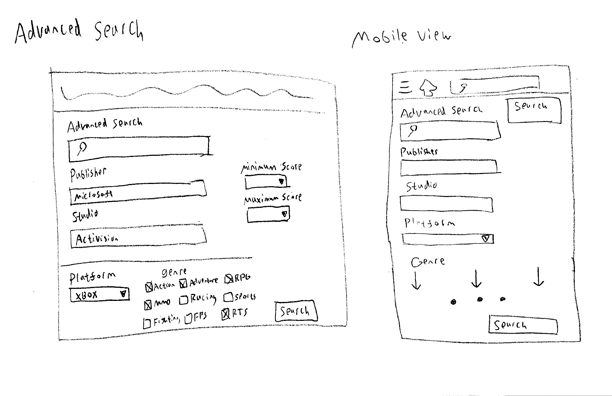
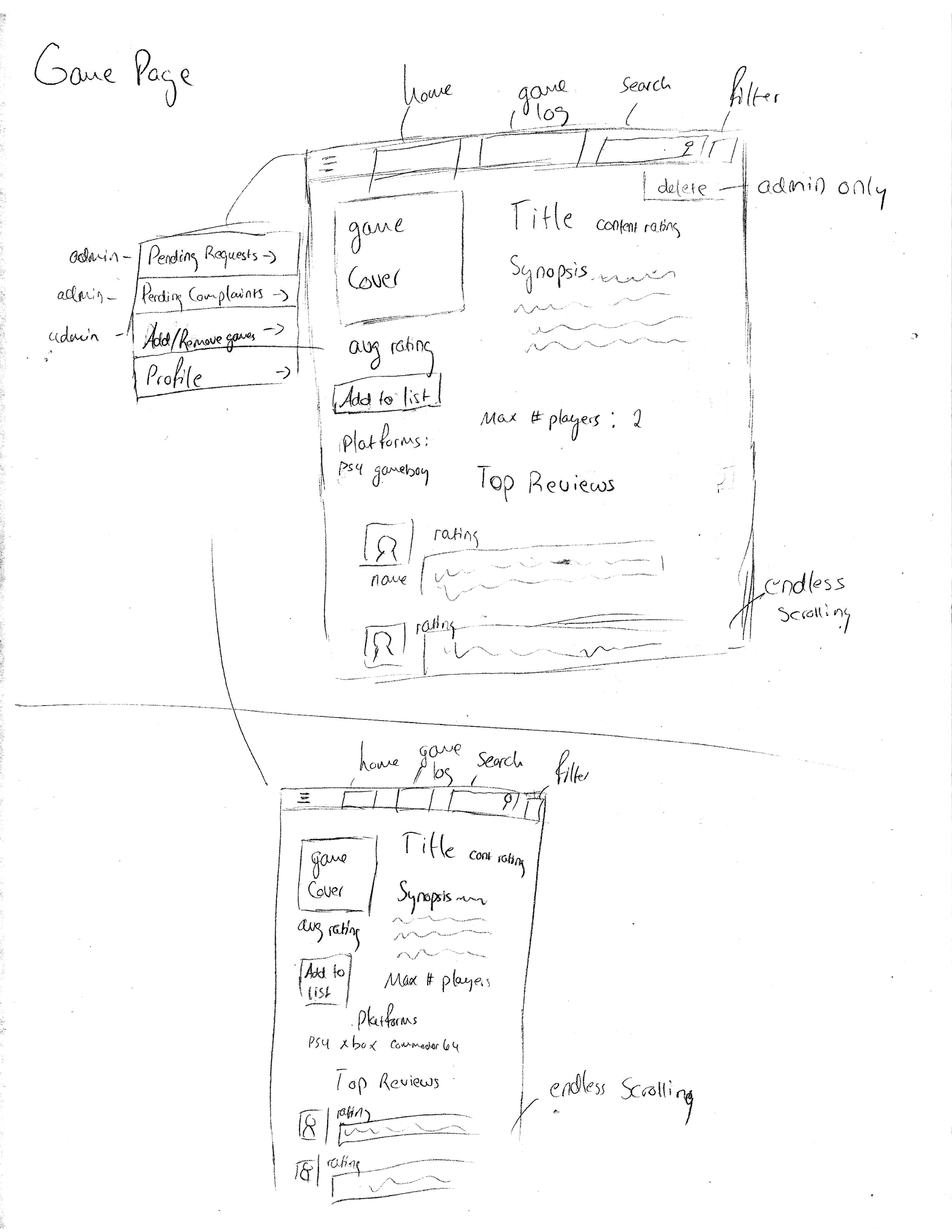
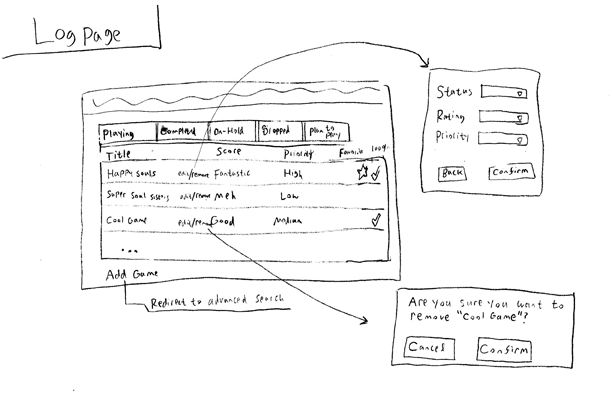
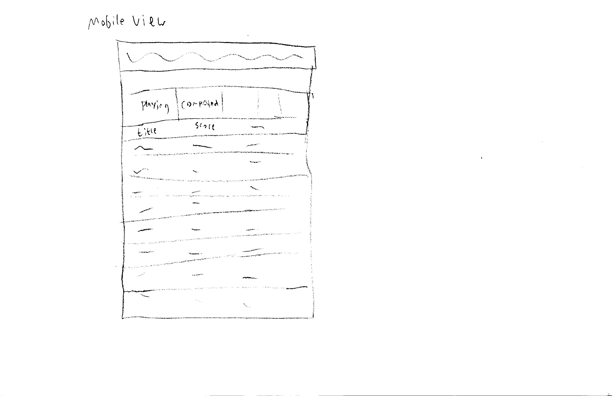
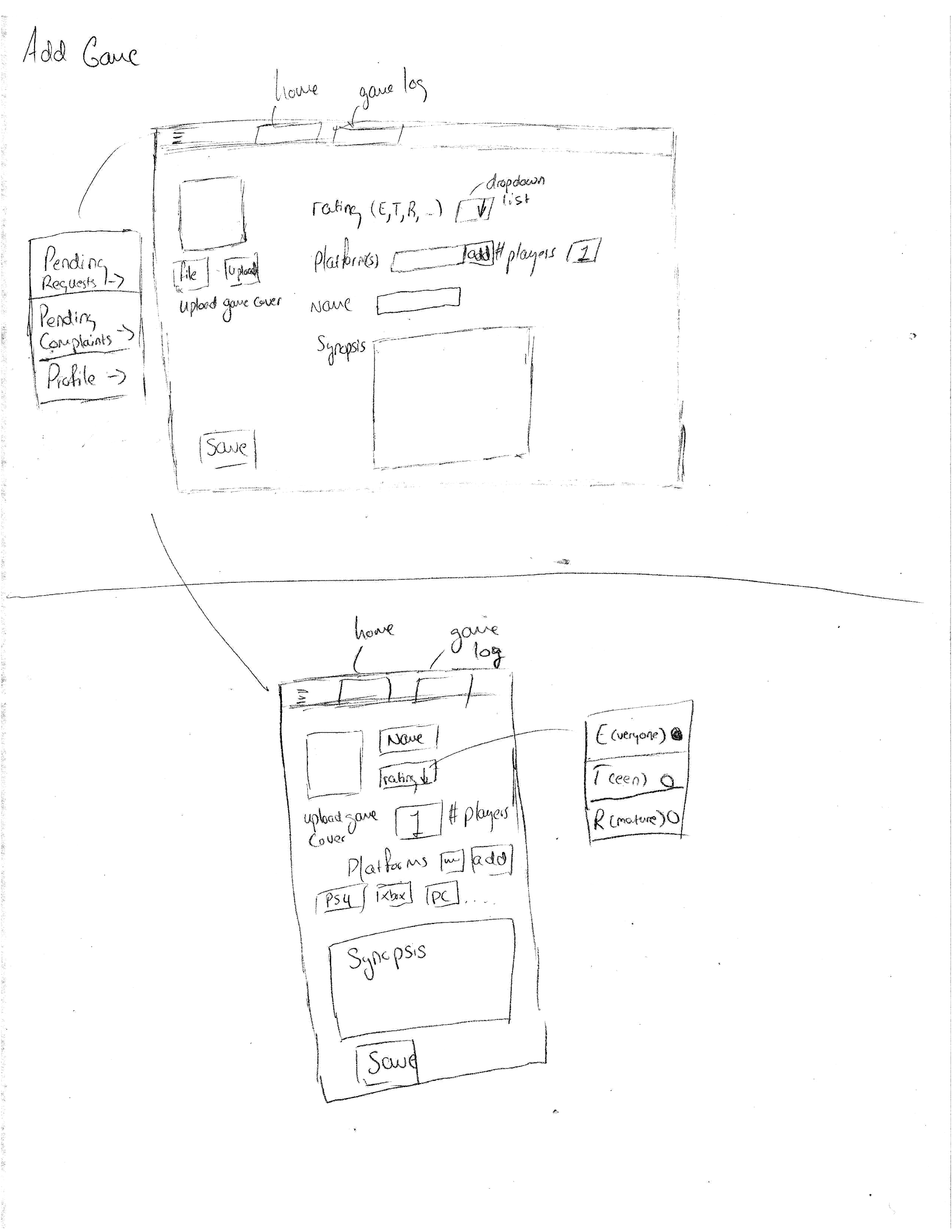
## 3.b. System Layouts by User

Below we demonstrate each user’s roles through storyboards. Each user has different capabilities, however some users have the same capabilities that we have chosen to not display. Essentially the Registered user will have all of the actions available as the Guest user plus extra functionality, and the Admin user will have all of the capabilities of the Registered user plus extra functionality as well.

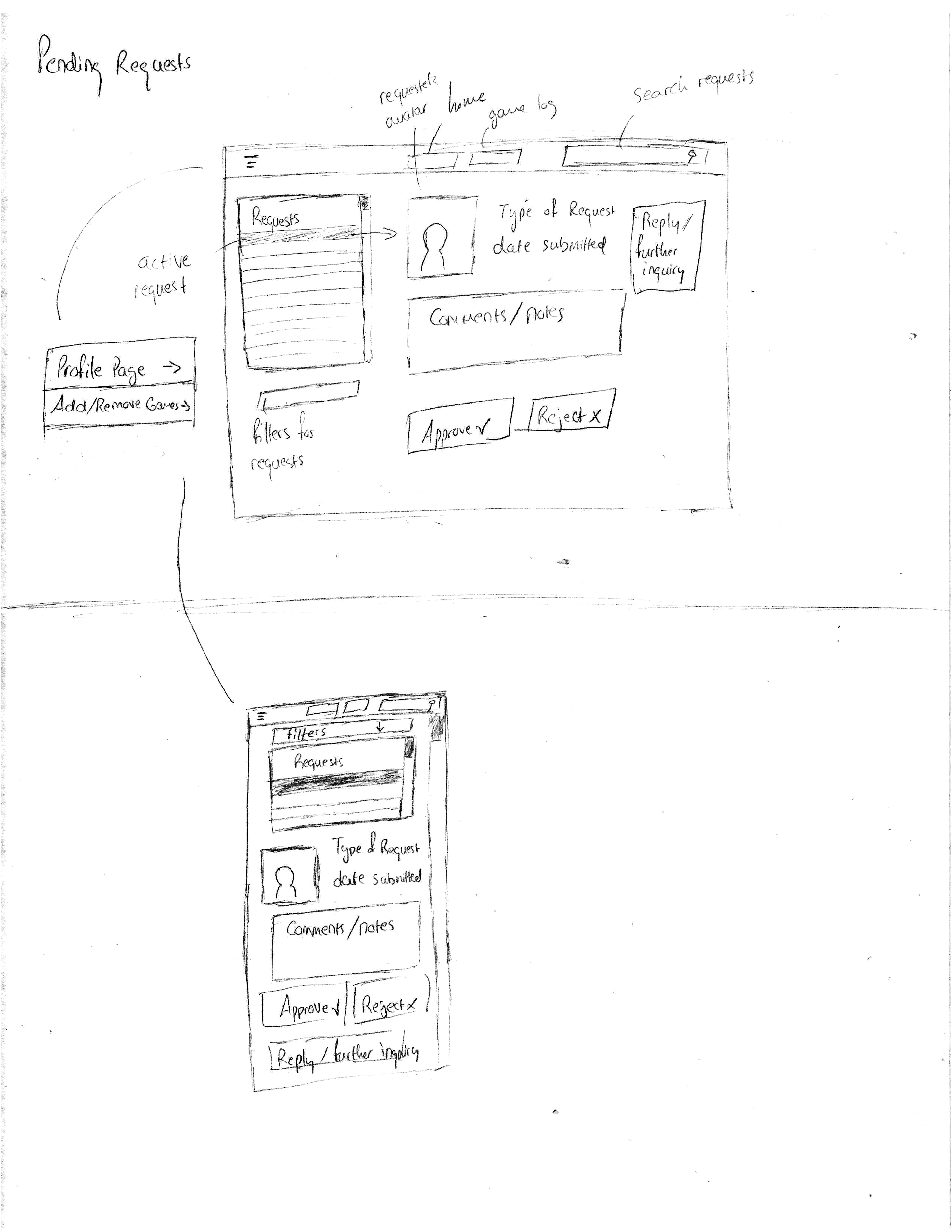
### 3.b.i Guest User

1. Home Page — shows news and other general information.  
     
     
     
     
   **Using the header menu the Guest user selects “Popular Games Page”.**
2. Popular Games — shows the site’s most popular games, ranked by how many users have added the games to their game list. Each game has a link to a “Game Page” that contains information on that game.  
     
     
     
   **The Guest user decides to create a user profile. Using the header menu the Guest user selects “Login Page”.**
3. Login Page — shows “Login” screen or a button to take you to the “Register” page. Guest clicks “Register”  
     
     
     
     
   **The Guest user selects the link to the “Register Page”.**
4. Register Page — shows information needed to populate the user profile, including username, password, email, etc. Displays a “Sign Up” button that will submit the information and create the profile.  
     
   **The Guest user selects “Sign Up” after filling in the information.**

### 3.b.ii Registered User

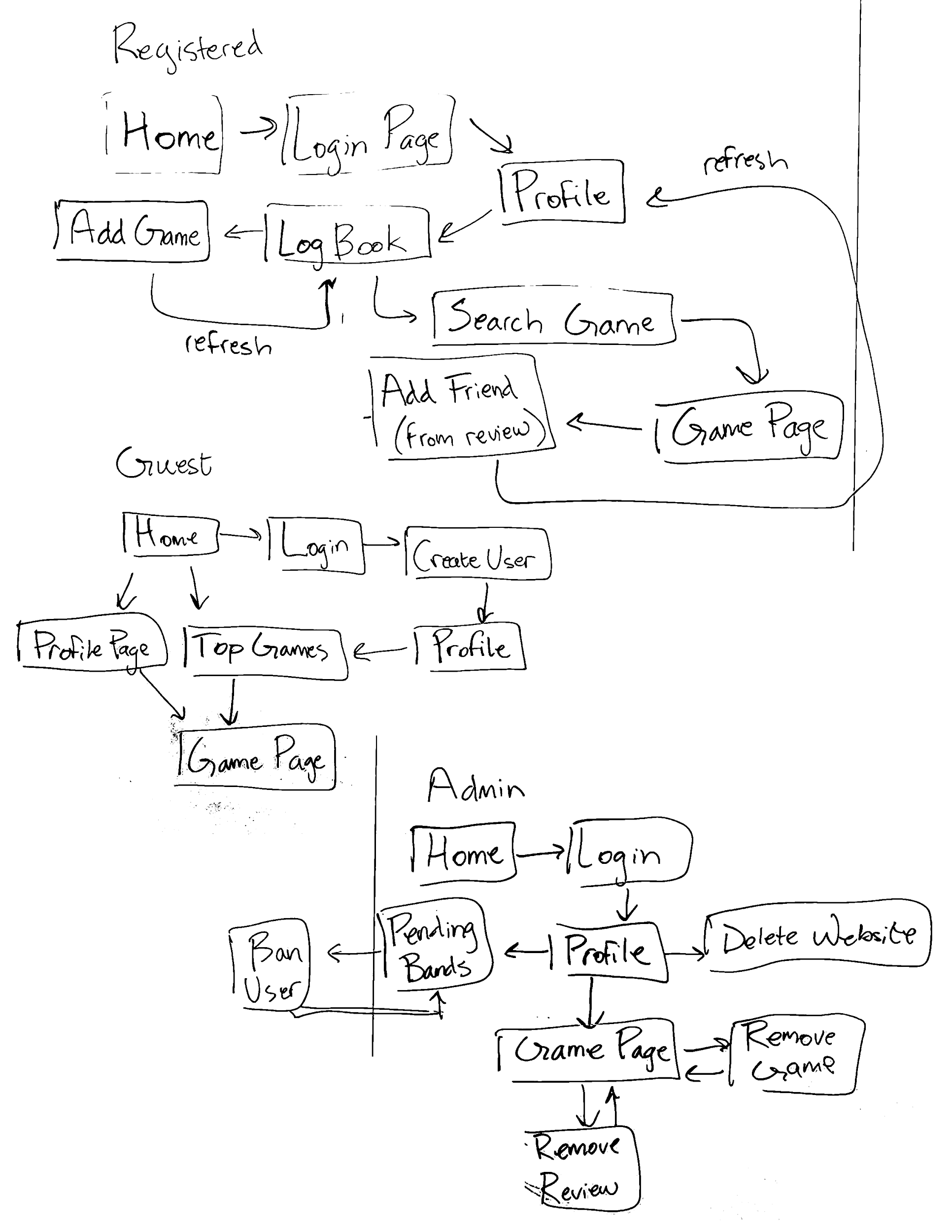
1. Profile Page — shows the user’s profile information, as well as the user’s recent activity on the website. This includes lists of the games the user has played, reviews the user has made, and people the user is following.  
     
   ++++++++++++  
     
   **Using the header menu the newly Registered user selects “Top Games Page”.**
2. Advanced Search Page — allows the user to search the database off of specific inputs.  
     
     
     
   **The Registered user searches for a game, and selects the “Game Information Page” of a game from the displayed list.**
3. Game Information Page — shows specific information on a specific game. The page has an “Log Game” link to add it to the user’s profile.  
     
     
     
   **The Registered user decides to select “Log Game” to add the game to their “Log Book”. They are then directed to their “Log Book Page”.**
4. Log Book Page — shows a detailed, sectioned list of all game information related to the user. This allows users to manage the reviews and ratings they have given to games. From here they are also able to add games to the database using the “Add Game Page” link.  
     
     
     
     
   **Using the link the Registered user selects “Add Game Page”.**
5. Add Game Page — shows input fields for adding a game to the game database. This will notify the Admin users that a new game has been added, but will exist until an Admin decides to remove the game. It has a “Submit” link in order to finish adding the game.  
     
     
     
   **The Registered user fills out the information to add a new game. They then select the “Submit” link, and are redirected to the “Log Book Page”. Using the link the Registered user selects “Log Out”.**

### 3.b.iii Admin

1. Pending Requests Page — shows a list of users (selected by other users) who might should be banned from the website, based upon the website’s policies. The Admin user may chose to review the information by selected the “Complaint Page” link.  
     
     
     
   **The Admin user selects the “Request” link for one of the users on the ban list. Specific information on the request is displayed. The Admin user can approve or reject the request. They decide to think about it some more and come back to it later. The Admin user uses the header menu to “Log Out”.**

### 3.c Supplementary Website Flowchart

Below is the website flowchart we have tried to illustrate using our storyboards. Some functionality is included on the flowchart that is not illustrated in the diagrams; for the storyboards we focused on the main functions of our website.

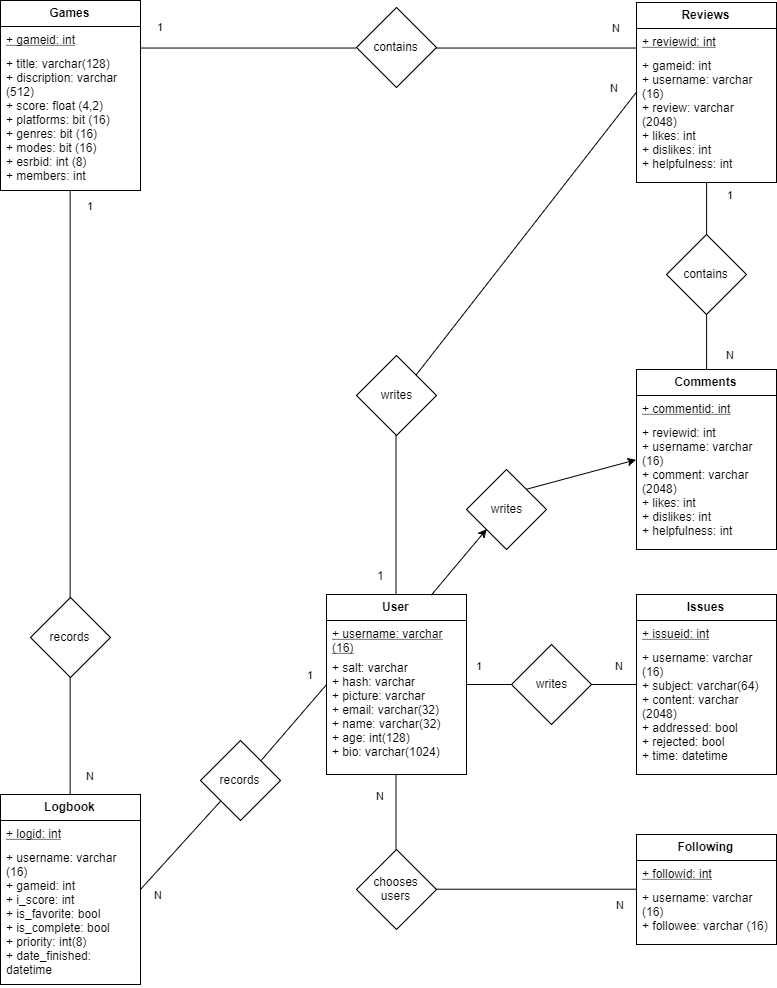


# 4. ER Diagram

This section contains representations of our databases. We will be using a MySQL database setup. In order to decide how to setup our database initially, we first looked at what game and user information we would need. Referencing our prototype, we enumerated all the fields we would need to store this data. Other data we decided to include was reviews, comments, issues, and following, which provides the rest of the functionality of our website.

## 4.a. ER Diagram

Below is the ER (Entity Relationship) diagram for our website. In order to make this diagram, we looked at the relationships between the tables we created. Essentially, each game should be able to be reviewed by any users, each review should be able to be commented on by any users, each issue should be written by a user, users should be able to follow each other, and each logbook should be written by a user. These relationships are expressed in the image below.



## 4.b. Table Design

Below are the complete descriptions of each table in our database. Following the title of each table is a description of the table contents and their relationship to other tables.

**User Table**: Stores information pertaining to the user. This table is referenced by the username. The user will have lots of one-to-many relationships with other tables, since their username is used as a foreign key when searching for correlations in other tables (Favorites, Reviews, Comments), and a many to many relationship with the Following table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Primary Key** | **Field Name** | **Data Type** | **Non-null** | **Unique** | **Binary** | **Foreign Key** | **Comments** |
| X | username | varchar(16) | X |  | X |  | Stores the user’s displayed username: it should be a string no more than 16 characters long |
|  | salt | varchar (16) | X |  | X |  | Stores a salt to be used during hashing |
|  | hash | varchar | X |  | X |  | Stores the hash to find the user’s original password |
|  | picture | varchar |  |  |  |  | Stores the user’s profile picture: the picture should be linked to as a URL (i.e. a string) |
|  | email | varchar(32) | X |  |  |  | Stores the user’s email: it should be a string no more than 32 characters long |
|  | name | varchar(32) | X |  |  |  | Stores the user’s name: it should be a string no more than 32 characters long |
|  | age | int(128) |  |  |  |  | Stores the user’s age: it should be an integer no greater than 128 |
|  | bio | varchar (1024) |  |  |  |  | Stores the user’s biography information: it should be a string no more than 1024 characters long |
|  | joined | datetime | X |  |  |  | Stores the date that the user joined the website. |

**Reviews Table**: Stores information about the reviews for all games. This table is referenced by the review ID. Each game’s reviews will be displayed by searching this table for the game’s ID and finding the corresponding review. If the result comes up as null, this means the game does not have any reviews. Therefore the gameid entry will be used as a foreign key to access game information. This table also can be searched by a username, in order to find a specific user who wrote a review.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Primary Key** | **Field Name** | **Data Type** | **Non-null** | **Unique** | **Binary** | **Foreign Key** | **Comments** |
| X | reviewid | int | X | X |  |  | Stores the review’s ID: should automatically increase or decrease |
|  | gameid | int | X |  |  | X | Stores the game’s ID: should be used as a foreign key |
|  | username | varchar(16) | X |  | X |  | Stores the user’s displayed username: it should be a string no more than 16 characters long |
|  | review | varchar (2048) | X |  |  |  | Stores the review: should be a string no more than 2048 characters long |
|  | likes | int | X |  |  |  | Stores the review’s number of likes: should be an integer initialized at 0 |
|  | dislikes | int | X |  |  |  | Stores the review’s number of dislikes: should be an integer initialized at 0 |
|  | helpfulness | int | X |  |  |  | Stores the review’s helpfulness calculated as the number of likes minus the number of dislikes: should be an integer initialized at 0 |

**Logbook Table**: Stores information about games that a user is playing, planning to play, or has completed. Each log can be referenced by the log ID. Information is displayed about the game by using the gameid as a foreign key to access game information. Entries can be sorted by Title or Score (using score\_i).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Primary Key** | **Field Name** | **Data Type** | **Non-null** | **Unique** | **Binary** | **Foreign Key** | **Comments** |
| X | logid | int | X | X |  |  | Stores the log ID: Incrementally stored |
|  | username | varchar(16) | X |  | X |  | Stores the user’s displayed username: it should be a string no more than 16 characters long |
|  | gameid | int | X |  |  | X | Stores the game ID: used as foreign key |
|  | score | int | X |  |  |  | Stores the score in int for easier sorting |
|  | is\_favorite | bool | X |  |  |  | Stores whether the game is a favorite |
|  | is\_completed | bool | X |  |  |  | Stores whether the game is completed by the user |
|  | priority | int | X |  |  |  | Stores the priority of the game |
|  | date\_finished | datetime |  |  |  |  | Stores the time game was completed: stored in seconds since the epoch |

**Following Table**: Stores information about which user is following which other user. This table is referenced by the follow ID. The username and followee can be used here as a foreign key to access each user’s information to be displayed on the website.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Primary Key** | **Field Name** | **Data Type** | **Non-null** | **Unique** | **Binary** | **Foreign Key** | **Comments** |
| X | followid | int | X | X |  |  | Stores the follow ID: for referencing this follow |
|  | username | varchar(16) | X |  | X | X | Stores the user’s displayed username: it should be a string no more than 16 characters long |
|  | followee | varchar(16) | X |  | X | X | Stores the username of the user being followed |

**Issues Table**: Stores issues and comments sent by users. This table is referenced by the issue ID. The username is used as a foreign key in order to access information about the user who wrote the issue.

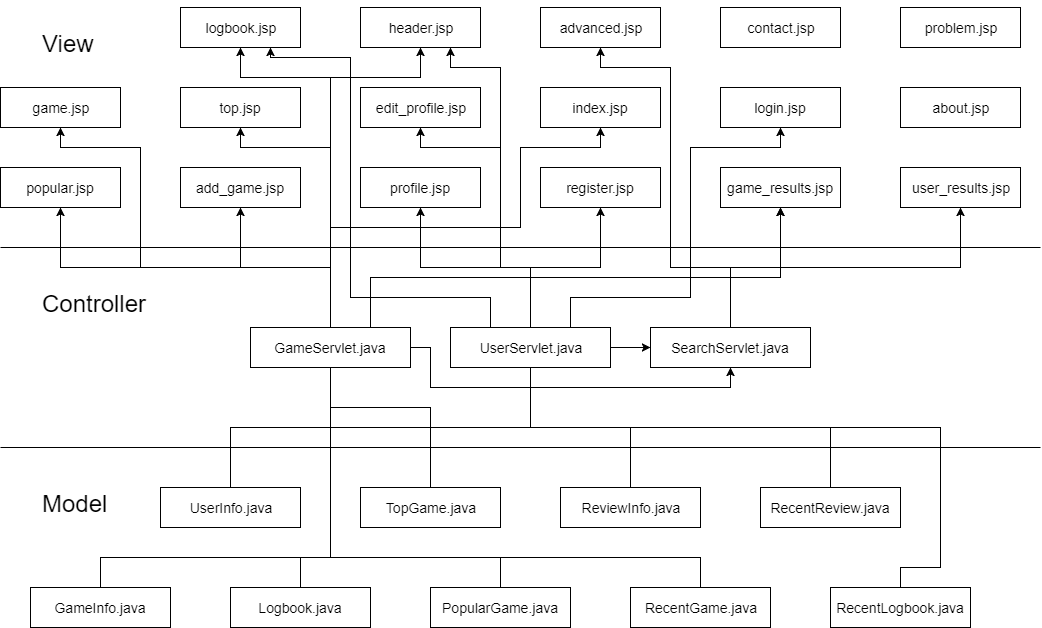
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Primary Key** | **Field Name** | **Data Type** | **Non-null** | **Unique** | **Binary** | **Foreign Key** | **Comments** |
| X | issueid | int | X | X |  |  | Stores the issue ID |
|  | username | varchar(16) | X |  | X | X | Stores the user’s displayed username: it should be a string no more than 16 characters long |
|  | subject | varchar (64) | X |  |  |  | Stores the subject of the message |
|  | content | varchar (2048) | X |  |  |  | Stores the body of the message |
|  | addressed | bool | X |  |  |  | Stores whether the issue has been addressed: default is false |
|  | rejected | bool | X |  |  |  | Stores whether the issue has been rejected for addressing: default value is false |
|  | time | datetime | X |  |  |  | Stores time of submission: stored in seconds since the epoch |

**Games Table**: Stores information about each game entry, including both information on the game itself, and meta information on score and statistics. This table is referenced by the game ID. There is no need for a foreign key connection with this table.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Primary Key** | **Field Name** | **Data Type** | **Non-null** | **Unique** | **Binary** | **Foreign Key** | **Comments** |
| X | gameid | int | X | X |  |  | Stores the unique game ID associated with each game. For simplicity it should be the same as the gameDB id |
|  | title | varchar (128) | X |  |  |  | Stores the game title |
|  | description | varchar (512) | X |  |  |  | Stores the game description |
|  | score | float (4,2) | X |  |  |  | Stores the overall score for the game |
|  | platforms | Bit field | X |  |  |  | Stores a list of platforms in a string. An optimization would use a table of platform ids instead. With the current setup (collecting data from “thegameDB”), there will only be one platform per game |
|  | genres | Bit field | X |  |  |  | Stores a list of genres. An optimization would use a table of genre ids instead |
|  | modes | Bit field | X |  |  |  | Stores a list of modes (i.e. Co-op, Multiplayer, ect.). An optimization would use a table of mode ids instead |
|  | esrbid | int | X |  |  |  | Stores an id of the ESRB rating, which can be used in fetching the relevant image from the images folder |
|  | members | int | X |  |  |  | Stores the number of members the game has (i.e. the number of people who have added the game to their list) |

# 5. System Architecture

## 5.a MVC Architecture Diagram



## 5.b Architecture Explanation

Our architecture uses 17 individual JSP files, 3 servlets to handle requests to and from the JSP files, and 9 Java Beans that access the database. Notice that some JSP files aren’t connected to any servlets. These JSP files currently don’t need to send requests to the servlets, meaning either we didn’t have time to implement them or the page is a static information page.

Each file is described in the table below, which elaborates on its relationship to other files:

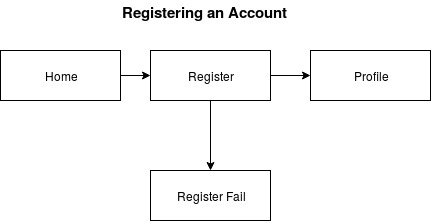
|  |  |
| --- | --- |
| **File** | **Description** |
| about.jsp | A static about page that contains information about the website and its creators. |
| add\_game.jsp | Used to add games to their logbook; the page sends a request to the GameServlet when the user submits the form. |
| advanced.jsp | Used to search the database for a number of different query parameters, including: game title, publisher, developer, minimum score, maximum score, platform, and username. Searching both by game and user forwards the request to the SearchServlet when the user submits the form. |
| contact.jsp | A static page that contains information used to contact us. |
| edit\_profile.jsp | Used to edit profile information; the page forwards the request to the UserServlet once the user submits the form. |
| game.jsp | Used to display information about a game. The user can choose to add the game to their logbook or they may choose to like or dislike reviews, in which case the request is forwarded to the GameServlet. |
| game\_results.jsp | Displays search results for games. The user can select to view more games, in which case a request to the SearchServlet is made. Otherwise they can choose to view the games’ information page (game.jsp), in which case a request is made to the GameServlet. |
| header.jsp | Used to store the header that is imported into all pages; this page accesses both the GameServlet and UserServlet, depending on the link the user decides to select. |
| index.jsp | Used as the homepage, which displays recently updated games, recently logged games, and recent reviews. |
| logbook.jsp | Used to display a user’s logged games. Each game has a link that the user can use to access the game information page, in which the request is forwarded to the GameServlet. |
| login.jsp | Used to login users; when a user logs in, the request is forwarded to the UserServlet. |
| popular.jsp | Displays the most popular games, ranked by how many members have added the game to their logbook. The user can choose to select a game to view its game information page, in which the request is forwarded to the GameServlet. |
| problem.jsp | A static page that we didn’t have time to connect to the database. It displays text fields for submitting an issue request, however these requests are not logged anywhere. |
| profile.jsp | Displays a user’s profile information and their favorited games. The user can choose to edit their profile information, in which the request is sent to the UserServlet. |
| register.jsp | Used to register users. Users must enter the requested information in the provided fields, and upon submission the request is forwarded to the UserServlet. |
| top.jsp | Displays the top games, ranked by the game’s score. A game’s score is updated each time the game is added to a user’s logbook. The user can choose to select a game to view its game information page, in which the request is forwarded to the GameServlet. |
| user\_results.jsp | Displays search results for users. |
| GameServlet.java | Handles requests related to games. |
| SearchServlet.java | Handles requests related to searching. |
| UserServlet.java | Handles requests related to users. |
| GameInfo.java | Stores the information from the database from the `Games` table in an ArrayList of GameInfo objects. The query parameter grabs all games from the database. |
| Logbook.java | Stores the information from the database from the `Logbook` table in an ArrayList of Logbook objects. The query parameter grabs all logbooks from the database. |
| PopularGame.java | Stores the information from the database from the `Games` table in an ArrayList of PopularGame objects. The query parameter sorts all games from the database by the game’s number of members. |
| RecentGame.java | Stores the information from the database from the `Games` table in an ArrayList of RecentGame objects. The query parameter sorts all games from the database by the game’s published date. |
| RecentLogbook.java | Stores the information from the database from the `Logbook` table in an ArrayList of RecentLogbook objects. The query parameter sorts all logbooks from the database by the date the logbook entry was added. |
| RecentReview.java | Stores the information from the database from the `Reviews` table in an ArrayList of RecentReview objects. The query parameter sorts all reviews from the database by the date the review was made. |
| ReviewInfo.java | Stores the information from the database from the `Reviews` table in an ArrayList of ReviewInfo objects. The query parameter grabs all reviews from the database. |
| TopGame.java | Stores the information from the database from the `Games` table in an ArrayList of TopGame objects. The query parameter sorts all games from the database by the game’s score. |
| UserInfo.java | Stores the information from the database from the `Users` table in an ArrayList of UserInfo objects. The query parameter grabs all users from the database. |

# 6. System Snapshot

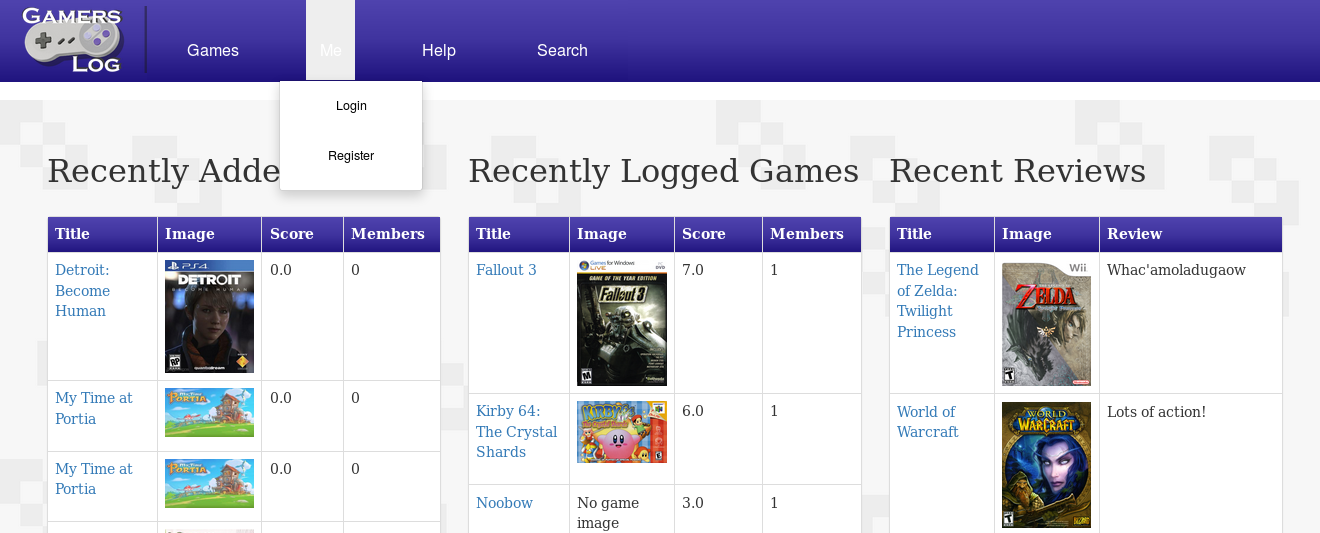
The main work flows that show most of the site’s usage are: registering a user, adding a game to the log book, reviewing a game, searching users, and viewing top and popular games.

Registering a User

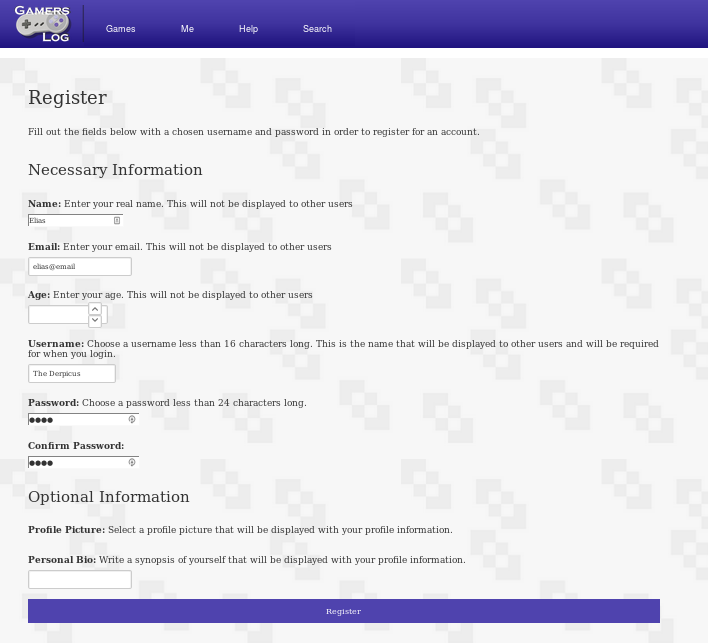
Registering a user has the following timeline where the user logs onto the homepage and clicks the register button on the bar. Then, after entering in the necessary information, the page will either accept their registration or reject it if the user already exists.



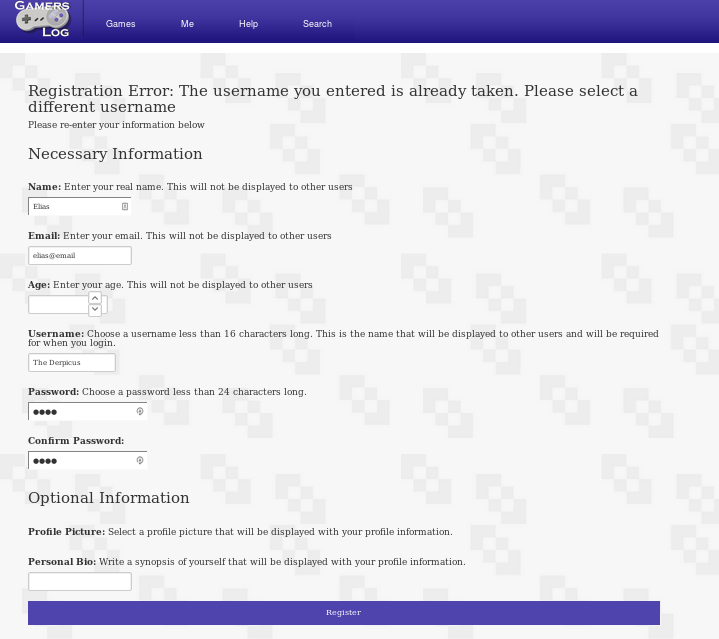
From the main page the user clicks on register



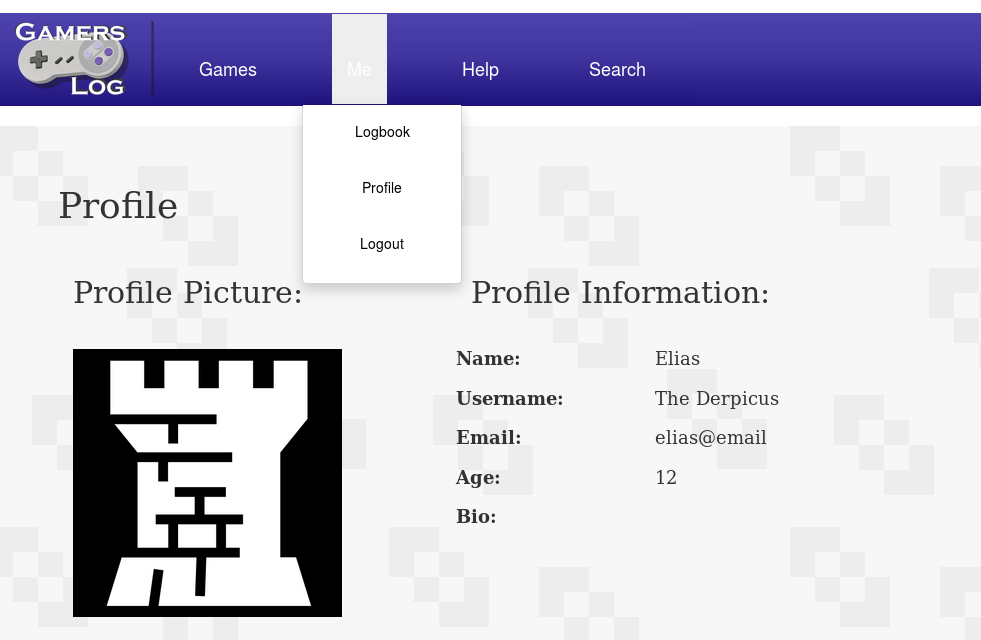
Then enters their information



If the user already exists the registration will fail

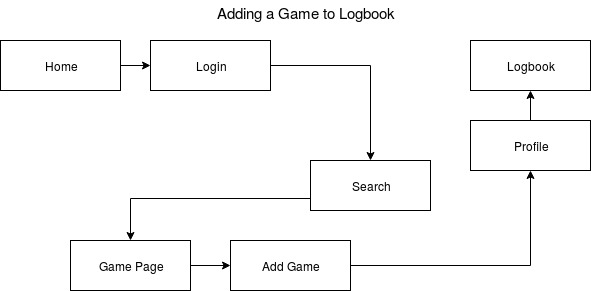


If it succeeds then it will take you to your profile and you can log out if you wish

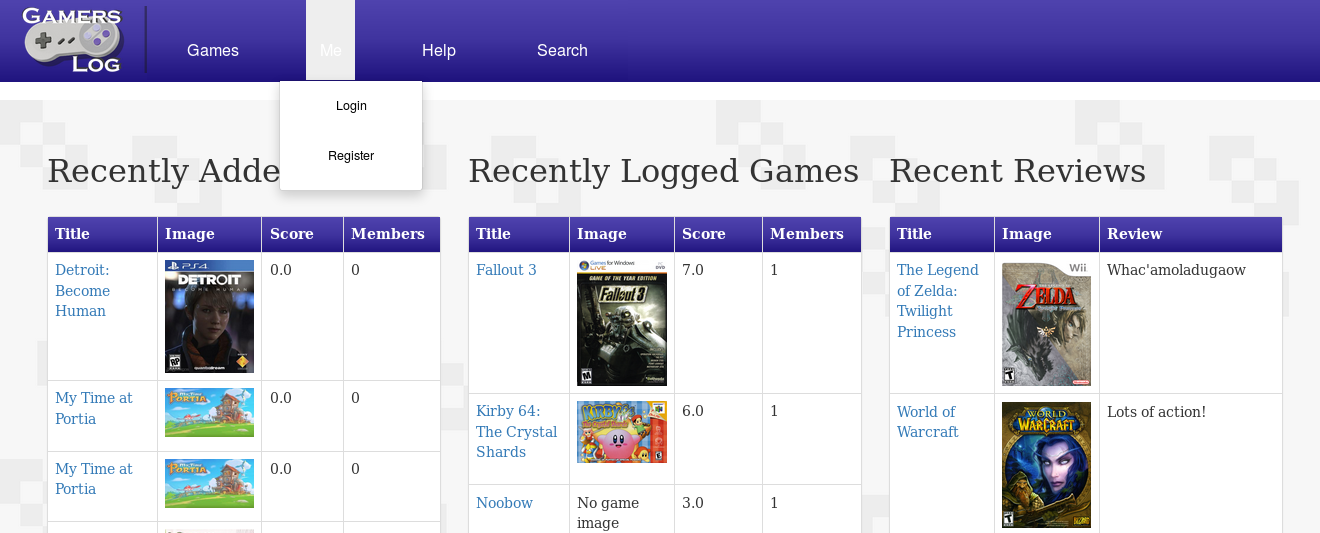


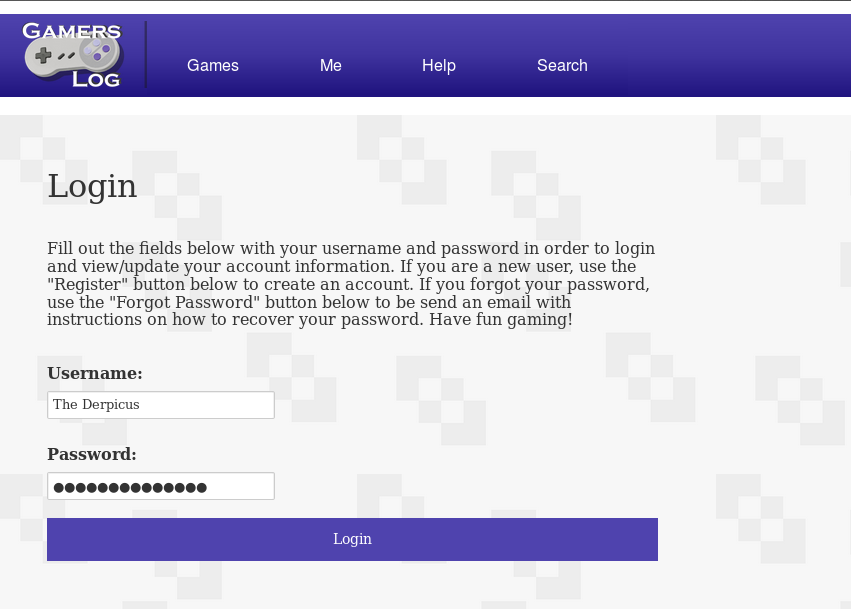
Adding a Game to Your Logbook

Adding a game to the logbook has the following timeline where the user logs onto the homepage and clicks the login button on the bar. When the user is back at the homepage they can go to a game’s game page by clicking on one on the homepage, the top and popular pages, or on a search results page. On the game’s page the user clicks the add to logbook button and get directed to the add game page where they enter a log entry. The user then gets redirected to the logbook page where their new entry is shown.

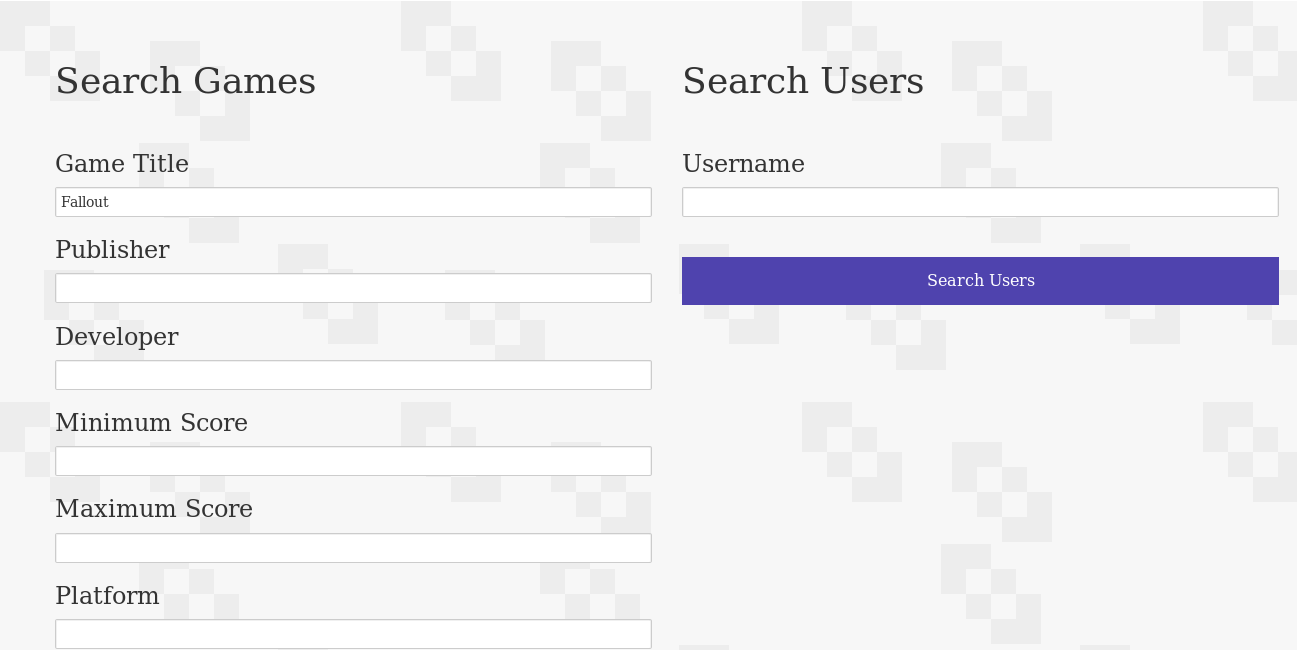


From the main page the user goes to the login screen

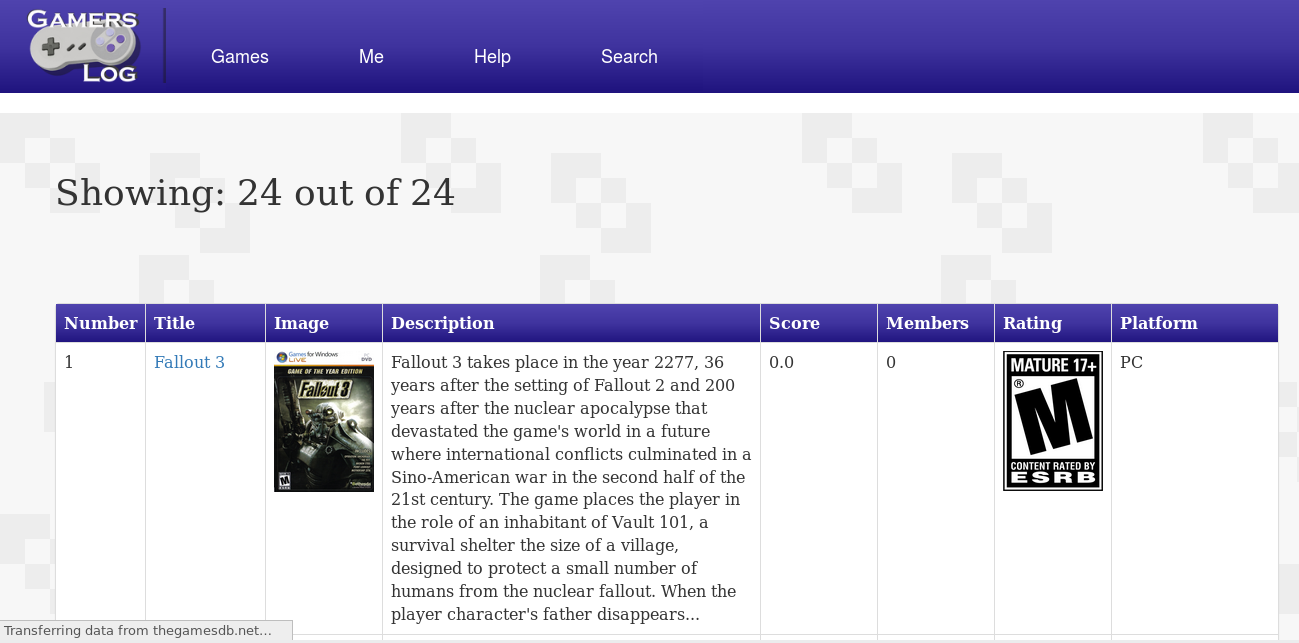




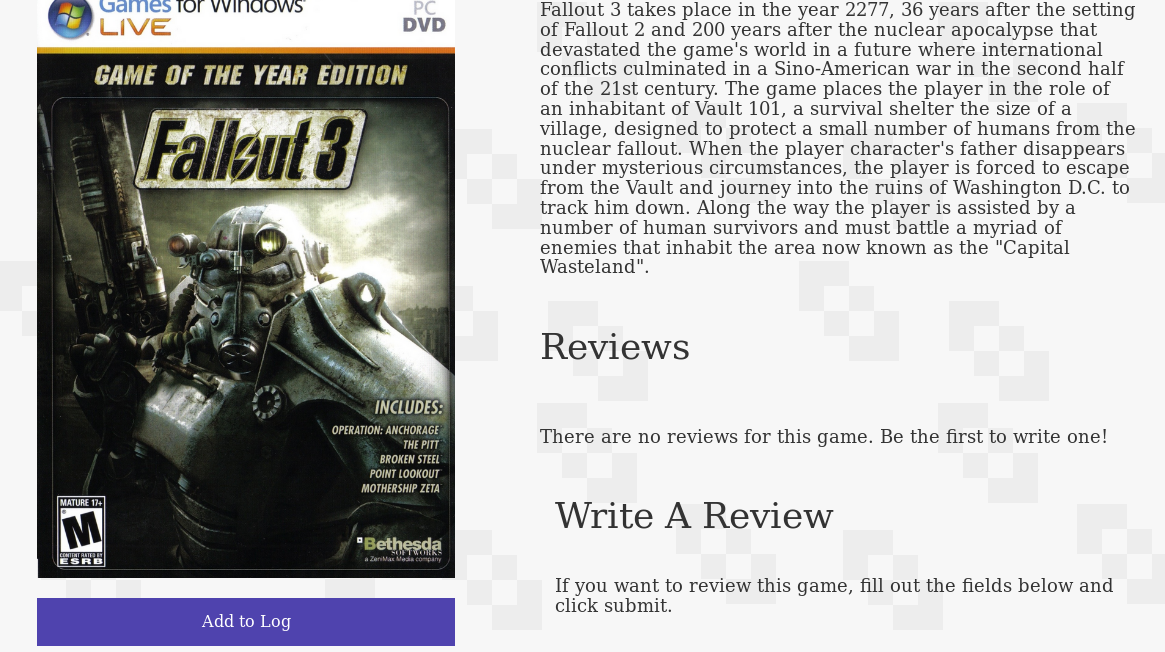
Then goes to the search page



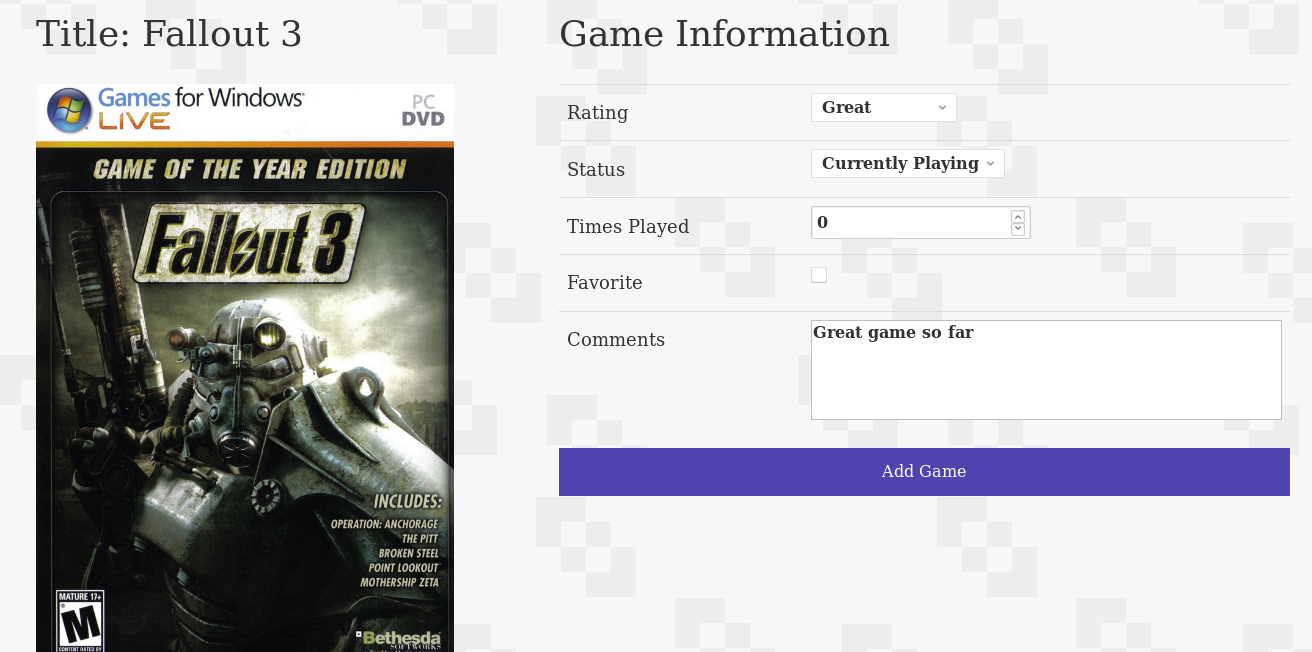
After submitting a query, the search results are shown and the user can click on a page to view



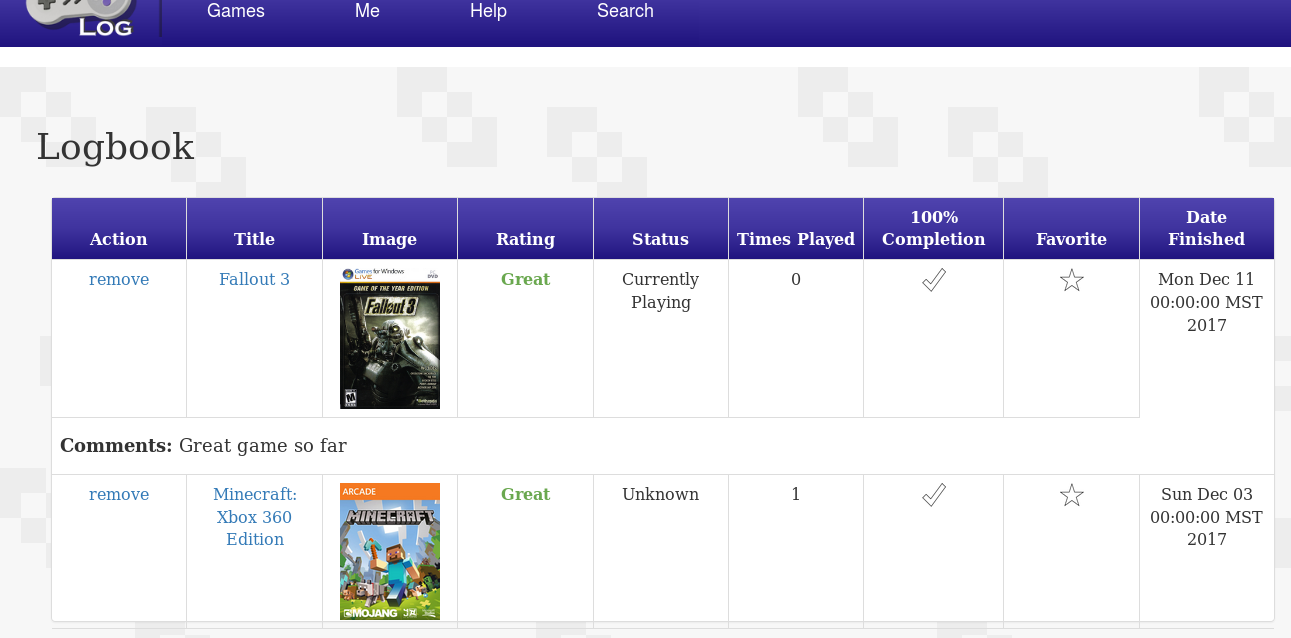
After clicking on a game the user will be redirected to the game’s game page



They then click on the add to log button to add it



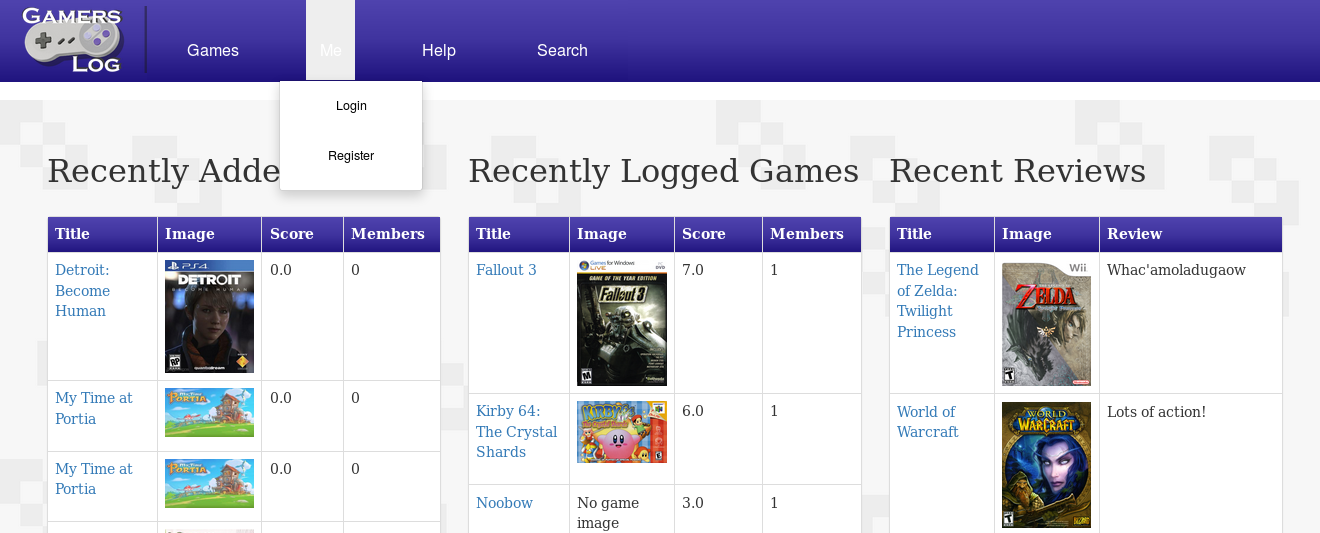
After entering the game information and clicking the add game button, it will show up in the user’s logbook

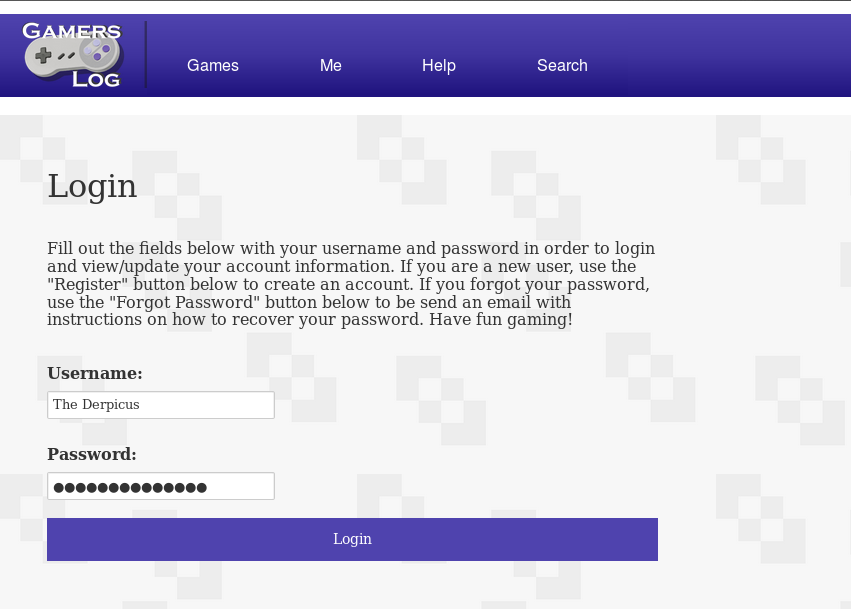


Reviewing a Game

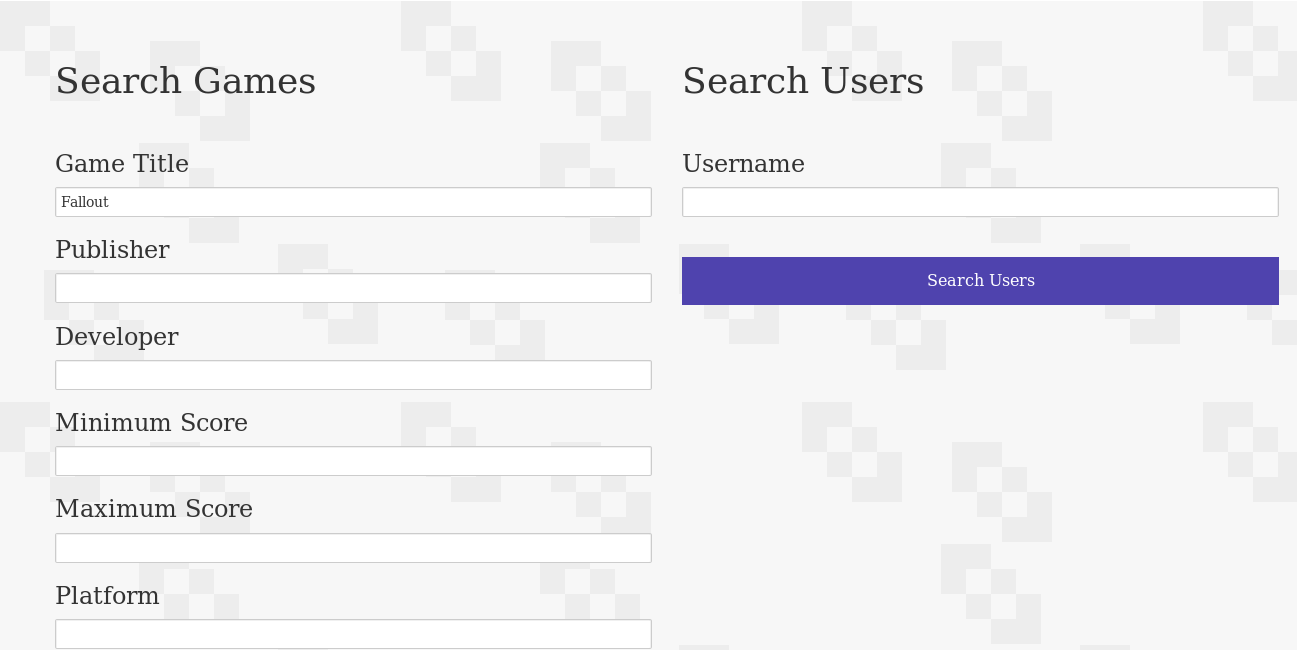
Reviewing a game has the following timeline where the user logs onto the homepage, logs in, searches for a game all as before. When at the game page the user can click the review button after adding a rating and writing a review.

From the main page the user goes to the login screen

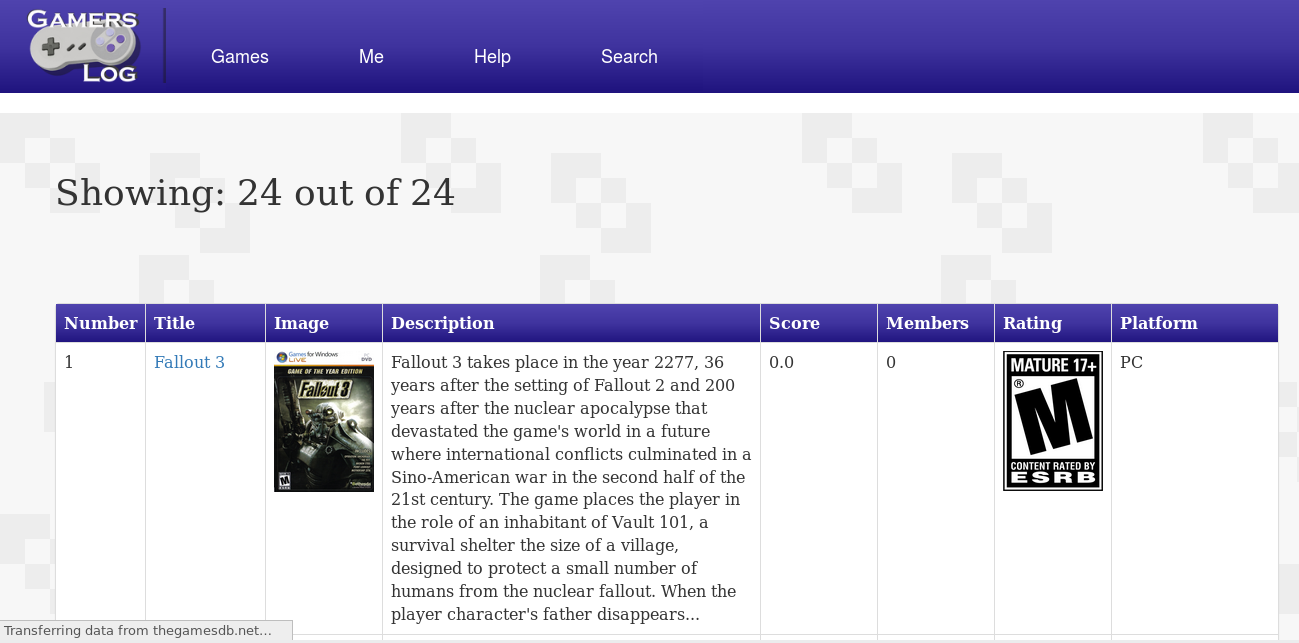




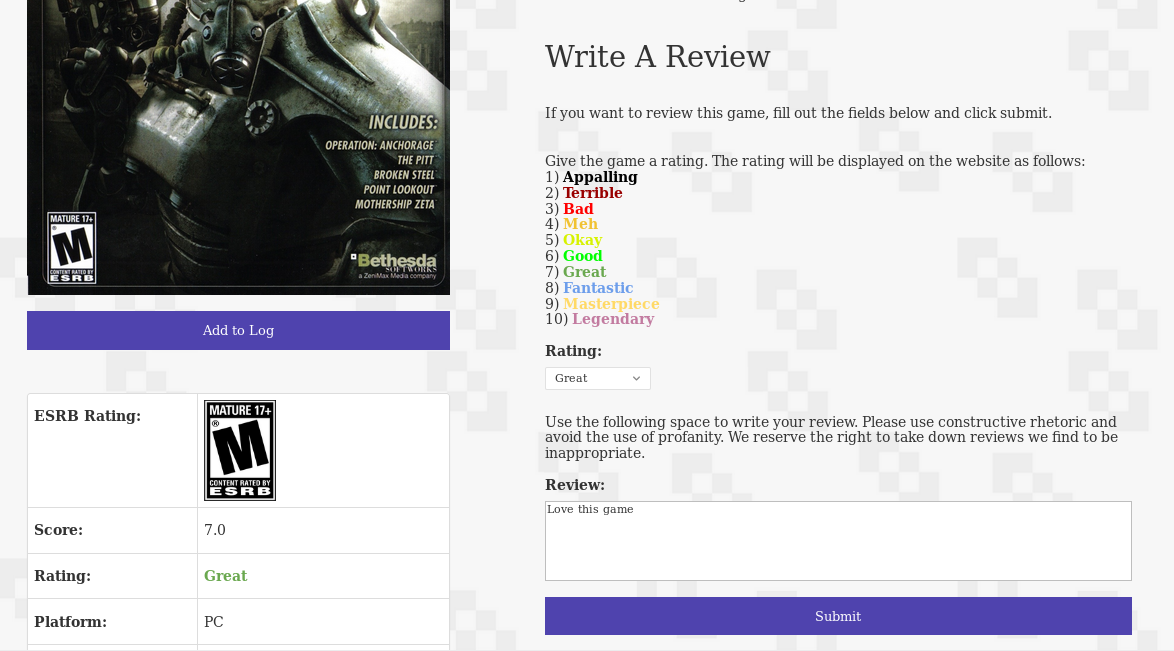
Then goes to the search page



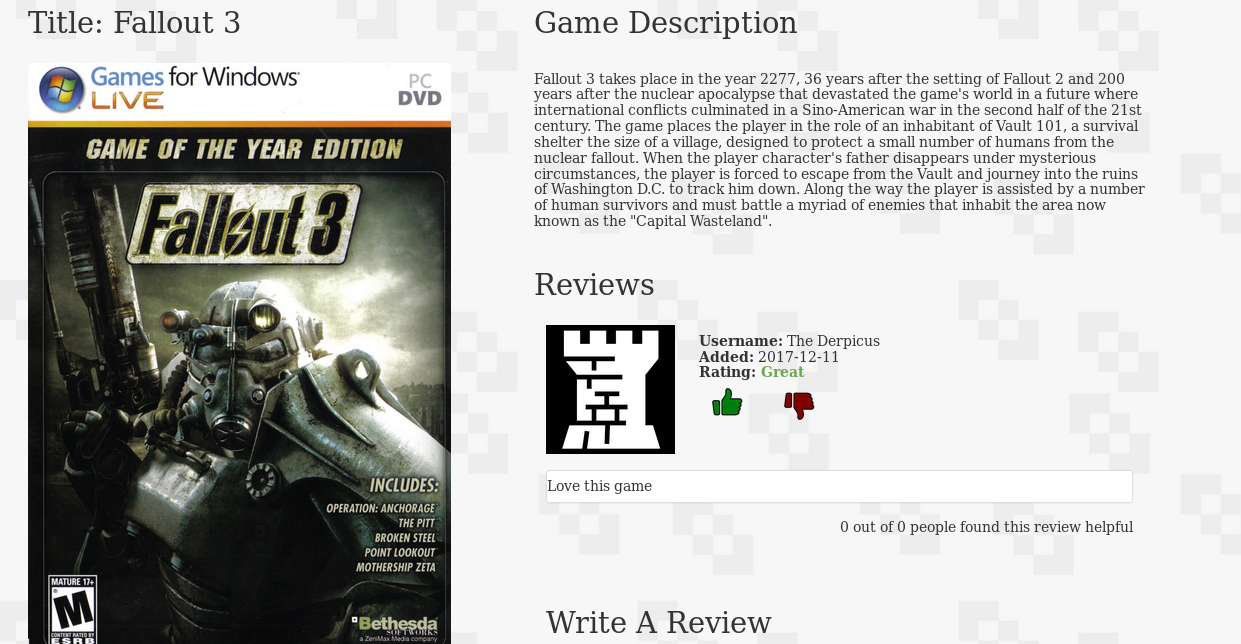
After submitting a query, the search results are shown and the user can click on a page to view



After clicking on a game the user will be redirected to the game’s game page where they can submit a review



The game page reflects the review

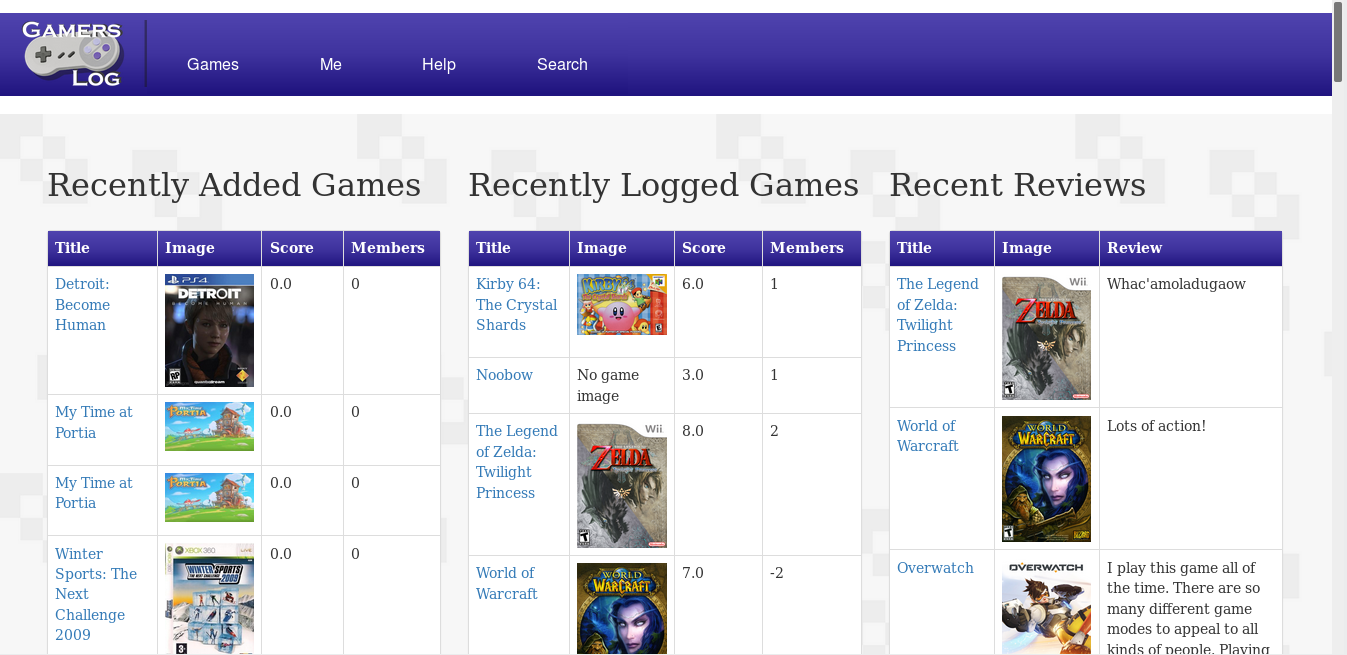


Searching a User

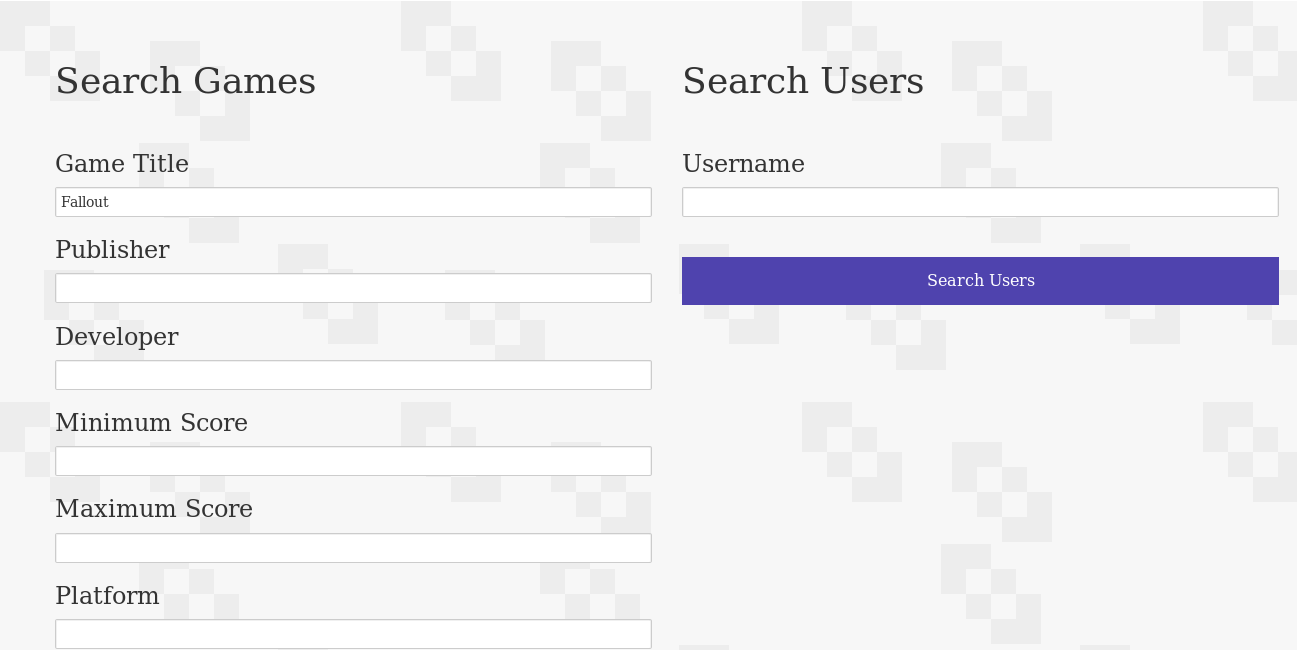
The timeline for searching a user is very similar to adding a game, but instead of clicking the game search button, they click the user search button. If the search field is empty the search results will include all users.



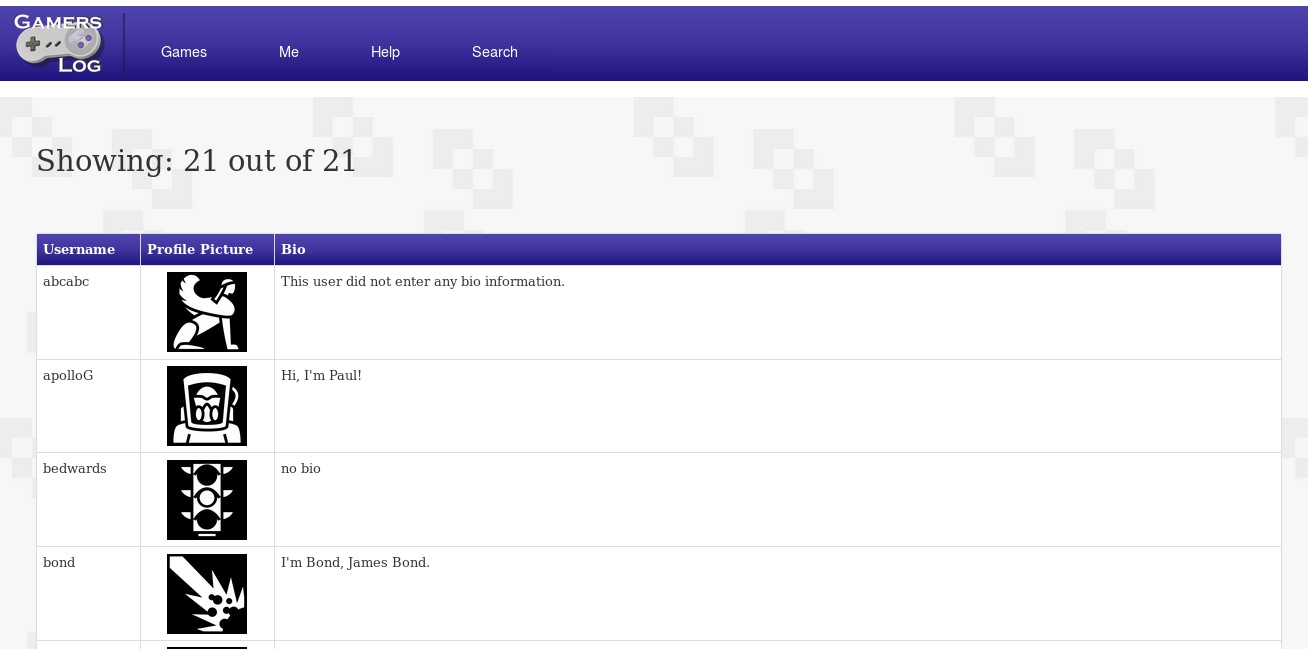
User starts at the homepage and clicks on the search button on the bar



The user refines the search if they wish, or leaves it blank to look at all users

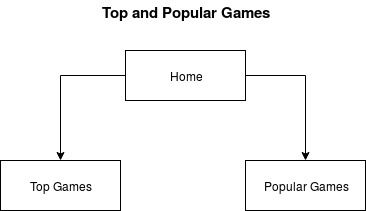


The search results show up based on the query

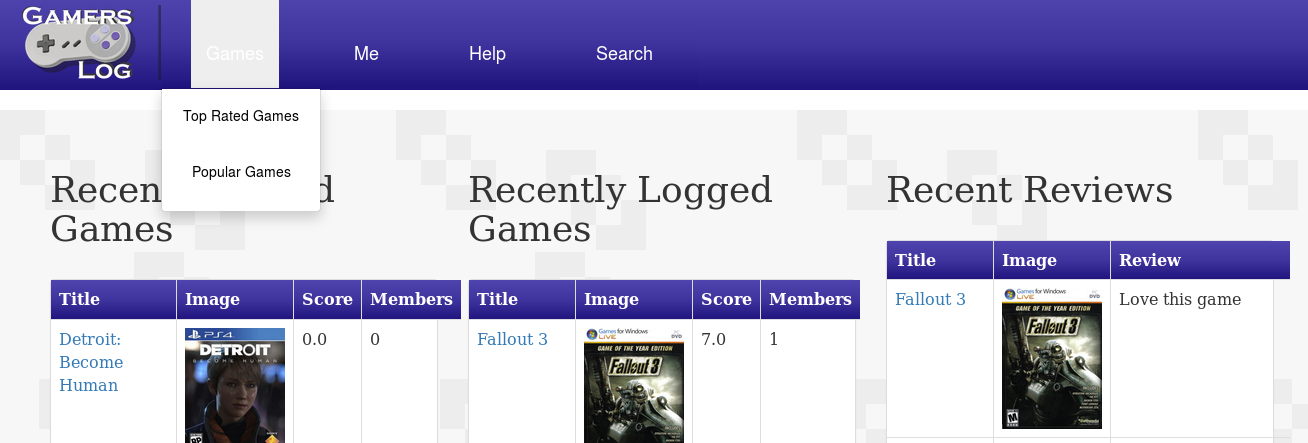


Viewing Top and Popular Games

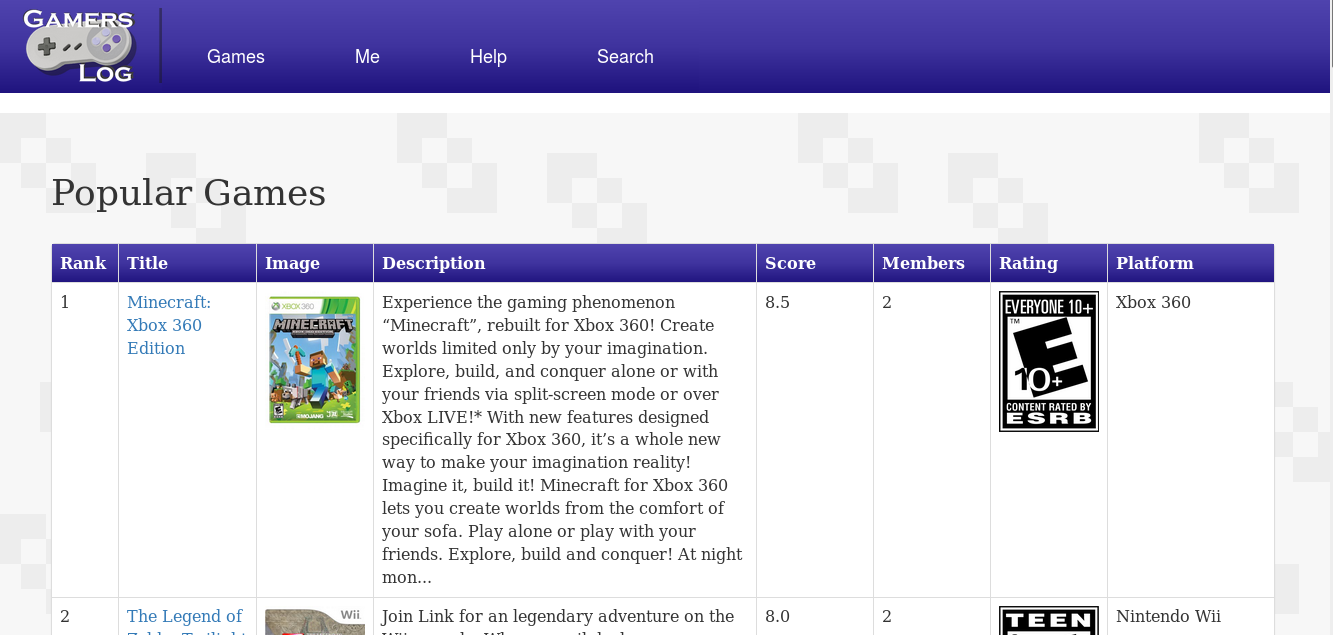
The timeline for viewing top and popular games is rather short, but is still an important feature of the site since it updates based on what people are rating certain games, or based on how many people are adding them to their logbooks.



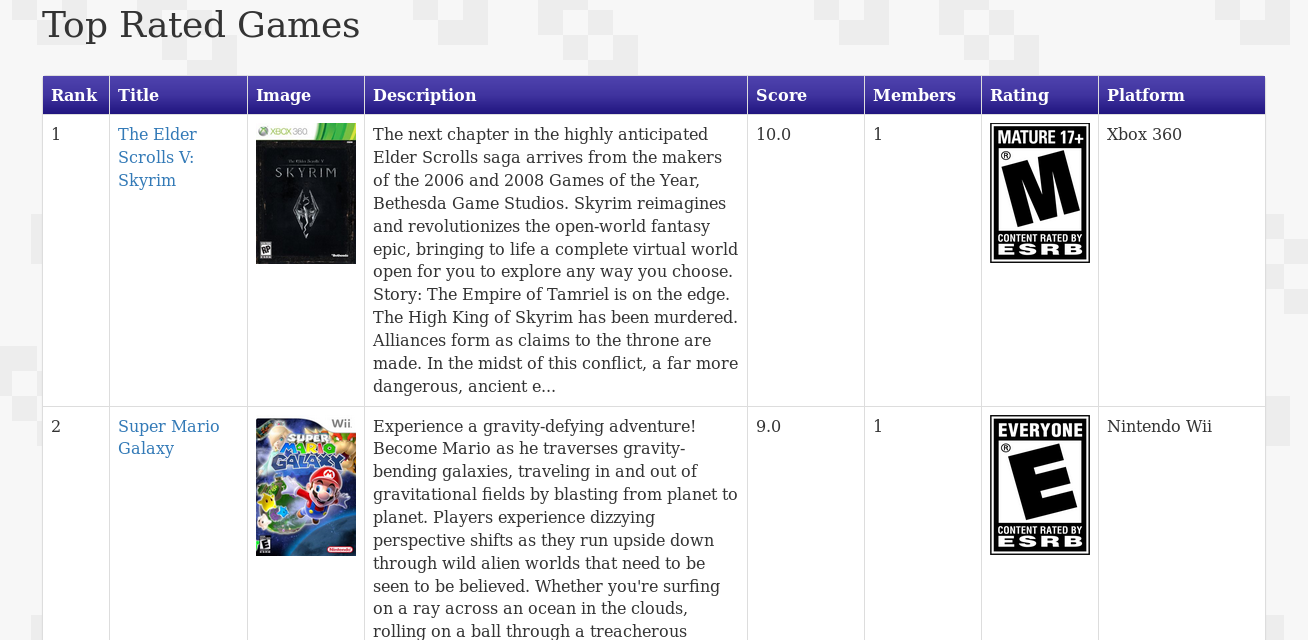
The user starts at the homepage and clicks on either Top or Popular games



Popular Games



Top games



# 7. Conclusion

In conclusion, Gamer’s Log provides a neat utility for keeping track of your games. The layout is designed to be intuitive, so that you can add games to your list with relative ease. The database is engineered to efficiently handle many types of data, while attempting to minimize redundancy.

The website offers a unique, community-driven experience, centered around videogames, and modeled after the design philosophy of myanimelist. Nearly all of the minimum criteria were met, and even most of the ideal features-- only some of the social features were missed.

## 7.a Summary

**Features Implemented:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Status** | **Notes** |
| User login system | Completed | Currently insecure (passwords stored in plaintext in database) |
| Add game to website | Not implemented | - |
| Display game information | Completed | - |
| Rate games | Completed | - |
| Review games | Completed | - |
| Search games | Completed + | Includes advanced search functionality, and the ability to search for users. |
| Sorting | Partial | Games can be sorted by using the advanced search mechanism, but logbook entries cannot be sorted, and there is no simple UI for sorting games. |
| Social (friends) | Not implemented | Can see other users on the website via reviews and search for other users. |
| Profile management | Partial | Users cannot upload their own profile pictures due to tomcat permissions in Takyon, but users can get a random profile picture from a pre-generated list. Furthermore, there is no option to edit/remove data from their personal logbooks, or take down reviews. Every other feature, such as being able to change information about themselves works just fine. |
| News (strech goal) | Not implemented | - |
| Forums (stretch goal) | Not implemented | - |

## 7.b Future works

There are many things planned for the future, the first of which includes patching existing bugs and completing all of the core functionality. News and forums will probably be the last features to be implemented, though there is more interest in having forums than having news. The following is a list of features we would like to implement in the future, outside of the core functionality and bug fixes:

* Implement searching directly into the navbar
* Display username directly in the navbar
* Add exploration page for finding games similar to your interests
* Add ability to search for publishers and developers
* Implement all console IDs (currently only most popular consoles are implemented)
* More complete advanced search page
* FAQ and legal pages (frequently asked questions, and information about privacy)
* All admin functionality
* Functionality for submitting a problem, and contacting the developers
* Admin view for all pages and special admin navbar
* Style fixes and improving website layout
* Users have the ability to upload their own profile pictures
* More secure login system (hashed passwords, email confirmation)
* Reset password functionality
* Images directly stored in database for faster loading times
* Automatically update games database
* Scoring system tweaks
* Store statistics about each game (e.g. average player age, date completed graph, ect.)
* More cowbell