1. Running Instruction and Configurations

1. Backend Configuration Instructions

default Host: HTTP://127.0.0.1:8000 We must configure the database and Django properly to make the backend work as expected.

Database

Database Language Configuration

We use MongoDB as the language for our database. Therefore configuring Mongodb is necessary to run our prototype.

1. The following command is to download the official Homebrew formula for MongoDB and the Database Tools:

```
brew tap MongoDB/brew
```

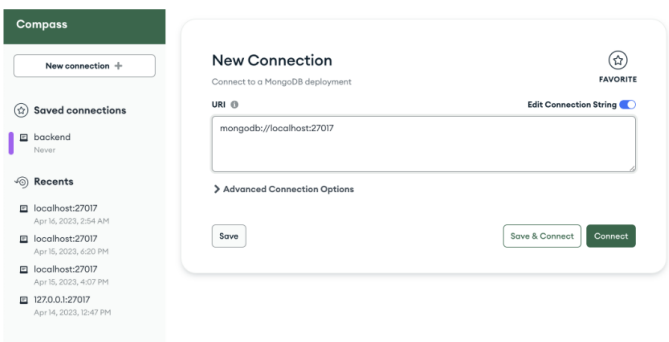
2. To install MongoDB, run the following command in your macOS Terminal application:

```
brew install mongodb-community@6.0
```

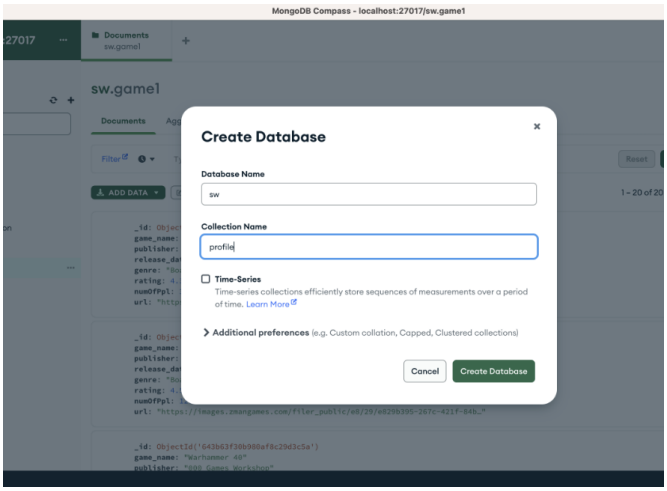
Build Database and Data Collection

We use MongoDB Compass to help us build the database. Please follow the link to download the MongoDB Compass https://www.mongodb.com/try/download/shell.

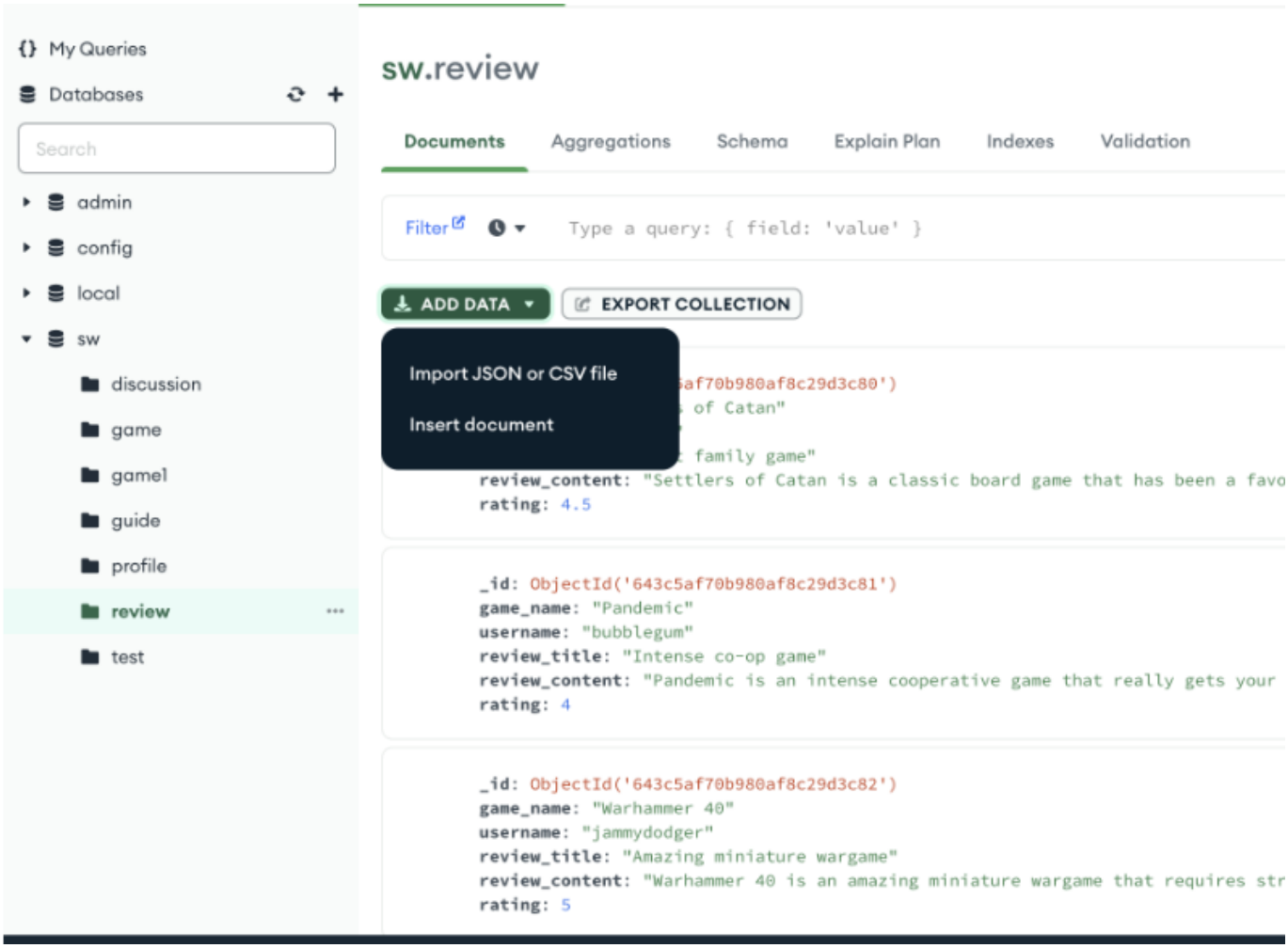
In MongoDB Compass, we need to connect with a MongoDB deployment, for example: get connected with port 27017 using mongodb://localhost:27017



After connecting, we created a new database called sw and a data collection called profile storing all the users' personal information, including their nickname, username, age, gender, email, password, favorite games, and friends.



In order to add data to the data collection, we can import profile.json prepared in the data repository.



we also created a game collection to store game information, guide collection, review collection, and discussion collection for storing guide, review, and discussion data. All of these data are in the data repository

Django

In order to install Django, please run the following command in the terminal:

```
pip install Django
```

some of the other Django packages are also necessary to run the backend

```
pip install djangorestframework
```

```
pip install django-cors-headers
```

```
python manage.py runserver
```

2. Frontend Running Instruction and Configuration

Default address: HTTP://localhost:3000 The Frontend is implemented by using the React.js library, the MUI UI Library, and the Antd UI Library.

2.1 Configuration

1. Download and configure the latest node.js (includes the npm)
2. Downloading necessary dependencies

```
cd /project/gmrs
npm install # Downloading necessary dependencies
```

2.2 Running Instruction

```
npm start
```

2. Usability Test Plan

Goal - This usability test's goal is to assess the game collection management and sourcing system's usability and find any usability problems that need be fixed before release.

Test Cohort

To ensure a successful usability testing study for our game collection management and sourcing system, it's essential to carefully consider the demographics of the test cohort. This guarantees

that the results are representative of the target user population and that any usability or user experience issues are identified.

We have determined that a sample size of 20 participants is appropriate for our study, as it aligns with industry standards and previous research studies. This sample size is sufficient to uncover usability problems and assess the user experience of our product effectively.

When selecting participants for our test cohort, we will take into account several crucial factors to ensure an accurate representation of the intended user demographic. These factors include:

- Age: We will consider participants from a wide age range, spanning from 18 to 60 years old, to assess the product's usability across different age groups.
- Experience with games: Participants with various gaming backgrounds, ranging from novice to experienced, will be sought to determine the product's intuitiveness and user-friendliness for different levels of expertise.
- Education level: We will consider participants with different levels of education, including high school graduates, college graduates, and postgraduate candidates, to ensure the product's accessibility to all users.
- Occupation: We will recruit participants from diverse occupational backgrounds, such as finance, healthcare, and education, to assess the product's usability in various contexts.
- Gender: We will strive for balanced gender representation in our test cohort, with both male and female participants considered.
- Ethnicity: We will consider participants from diverse ethnic backgrounds to ensure inclusivity and accessibility.
- Location: We will consider participants from different geographical regions to assess the product's usability in various parts of the world.
- Accessibility needs: To design an inclusive product, we will include participants with disabilities or different accessibility needs, such as color blindness or hearing impairments.
- Device preference: To assess the product's usability across different devices, we will include participants who use various types of devices, such as desktops, laptops, tablets, and mobile phones.
- Usage frequency: To assess the product's competitiveness in the market, we will include participants who frequently use similar products.

By considering these crucial parameters when selecting participants, our usability testing study results will accurately represent the target user population. This will enable us to identify any usability issues or user experience problems and make necessary improvements to enhance the product's usability and accessibility effectively.

Usability test procedure

Before-Test Survey:

The participants will be required to complete a pre-test questionnaire before the usability testing starts. The participant's background information, including their age, gender, and level of education, will be gathered using this questionnaire. Their gaming history, understanding of game mechanics, and current techniques of maintaining their game collections will also be gathered through the questionnaire. The team will be able to better understand how the participants' experiences and background may affect how well they perform throughout the usability testing with the use of this information.

- Example Before-Test Survey questions:
  - What age are you?
  - What gender are you?
  - What is the highest educational level you have attained?
  - How frequently do you play cards or board games?
  - Which sort of game do you prefer—board, card, miniature, or role-playing games?
  - How do you presently organize your game library, if one exists?

Task Scenarios

Using the prototype interface, participants will be given a set of task scenarios to complete. The purpose of these exercises is to assess the functioning and features of the prototype. For instance, the participants can be requested to add a new game to their library, look for new titles to buy, rate or review a game, or look for other gamers or gaming groups. The tasks will be given in a straightforward, natural language style for the participants.

- Example task scenarios questions:
  - Imagine that you have bought the board game Pandemic and want to add it to your collection. This is an example work situation. Please use the prototype's 'Add Game' feature to add 'Pandemic' to your collection.

Think-Aloud procedure

Participants will be urged to utilize the think-aloud procedure while completing the task scenarios during the usability testing research. This entails talking out their reasoning as they work through the tasks. Participants will be invited to discuss their thoughts, feelings, and any challenges they encounter while carrying out the tasks.

Team can learn more about users' interactions with the prototype and spot areas for improvement by using the think-aloud procedure. Saying, "I'm going to click on the 'Add Game'

button and see what happens," is an example of what a participant may say. Oh, this page has a search bar. I'll enter "Pandemic" and check to see if anything comes up. Wonderful, it's here. The 'Add to Collection' button must now be clicked. Done!"

The think-aloud procedure will, in general, allow us to learn more about how players make decisions and what obstacles they run across while utilizing the game collection management and sourcing system. We can enhance the user experience and increase the usefulness of the product by detecting these problems.

Post-Examination Survey

Participants will be requested to answer a post-test questionnaire after completing the task scenarios. This survey will learn how satisfied they are overall with the prototype, how simple it is to use, and how useful the features are. There will be both closed-ended and open-ended questions on the survey. Open-ended questions will provide participants the opportunity to contribute more comments and recommendations, whereas closed-ended questions will be used to evaluate satisfaction and usefulness using a rating system.

- Question examples for a post-test questionnaire:
  - How pleased were you overall with the prototype, on a scale of 1 to 5?
  - How simple was it to perform the task situations, on a scale of 1 to 5?
  - Did you run across any problems utilizing the prototype? If so, could you explain?
  - What aspect of the prototype did you like best?
  - What further characteristics would you want to see in the prototype?

Analyzing the results

To give a thorough evaluation of the prototype, the usability testing findings will be analysed using both quantitative and qualitative methodologies.

Quantitative data analysis:

The quantitative information gathered from the usability test will be statistically analysed using methods like completion time, task success rate, and user satisfaction ratings. Each task's completion time will be noted, and an average will be determined. The amount of time each participant spends on activities involving discovering a game will be recorded. By calculating the percentage of participants who completed each activity, such as adding a game to their collection, the success rate will also be calculated.

The data will be compiled using descriptive statistics, such as mean, median, and standard deviation, to offer an overall performance of all participants across all activities. For instance, the

average completion time will show how long it usually takes for each participant to do a job.

To compare the performance of various participant groups and find statistically significant variations in task completion times and satisfaction ratings, inferential statistics, such as t-tests, will be used. To compare the performance of male and female participants in terms of task completion time or satisfaction ratings, for instance, we may use a t-test.

Qualitative data analysis:

To find recurring themes and patterns in the qualitative data, such as the think-aloud procedures and open-ended questionnaire responses, content analysis will be used. In order to determine the advantages and disadvantages of the prototype, the team will record and analyse the think-aloud procedure and the replies to the open-ended questionnaire. The team will also look for persistent problems and challenges that participants had when using the prototype. The qualitative data will be utilized to supplement the quantitative data and give a more thorough insight of how the participants used the prototype.

3. Usability Analysis

As described in the Usability Test Plan section, we invited 20 participants to join our test, and carefully considered the demographics of the test cohort when selecting these 20 samples, making the results of this test more representative .

We used the Quantitative data analysis method and Qualitative data analysis method described in the Usability Test Plan section to collect and analyze the usability test results. In the following paragraphs, the results collected by the Quantitative data analysis method and the Qualitative data analysis method will be presented one by one, and these results will be analyzed.

Quantitative data analysis Method

In the part of using the Quantitative data analysis method, we designed a questionnaire to obtain the results of the usability test. All questions in this questionnaire are quantitative rather than qualitative. In the first part of the questionnaire, we asked users to make an overall quantitative evaluation of the game collection management and game query system. The first quantitative question asked participants to rate how well the system met their needs on a scale of 1-5, representing very dissatisfied to very satisfied. A second quantitative question asked participants to rate how easy the system was to use on a scale of 1-5, representing very difficult to very easy. The results we get are as follows. For the first question, more than 80% of the participants rated the satisfaction of the system as 4 or above, and the average satisfaction score of the 20 participants was 4.7, which shows that our system has basically realized services that users want. For the second question, only 65% of the participants scored 4 and above, and the final average score of the ease of use of the system was 4.2, indicating that our system may not

be very easy to use, which requires us to improve some designs. For example, we should add some prompt icons to make it easier for users to understand what functions this part will achieve. Or add some documentation to help users understand how to use the system.

In the second part of the questionnaire, we asked 20 participants to perform the following 13 tasks, and recorded their ratings on the ease of completing each task, ranging from 1 to 5, indicating from very difficult to very easy. Compared with the second question in the first part of the questionnaire, that question is the evaluation of the overall usability of the system, and it is not easy for us to find out which specific tasks our design is not perfect, concise and intuitive. Record participants' ratings for each task, making it easier for us to make targeted improvements.

The 13 tasks the 20 participations took part in are described below:

- New users register by inputting their email address/ phone number, constructing a profile, and setting the password.
- Users log into the system by inputting their account number and their password.
- User reset their password.
- Users update their profile. Input basic info like - location, name, gender, age and display picture.
- Users navigate to the Library to find the collection of their games.
- Users search for a new game in the Store.
- Users add/remove/update games into the main list and favourite lists.
- Users find overview information of each game after clicking each game. And find the button to updatel or remove this game.
- Users navigate to a specific user community page from the community page after selecting community options.
- Users read and posted reviews in the review page.
- Users read and posted discussions in the discussions page.
- Users read and posted guides in the guides page.
- Users add new friends to their friend list by input user ID, region or game name.

After collecting the ease of completion ratings for each task from 20 participants, we calculated the average score for each task's ease of completion. We will analyze these scores below. The average scores of the first task, the second task, the third task and the forth task are all above 4 points, indicating that these tasks are relatively easy to complete. Because in the prototype, we give very obvious text prompts on the input boxes and buttons. According to these prompts, users can quickly complete these four tasks.

The average score of the fifth task is also above 4 points, indicating that this task is relatively easy to complete. Because the word 'Library' can suggest to the user that this button will direct the user to their favorites. And the two drop-down lists on the Library interface often appear in other software, and users will be more familiar with them. It is easy for users to understand that as long as they click on the drop-down menu, they can get the list of favorite games.

The sixth task is similar to other search engines, and users can quickly search for the games they want. However, the search function is not perfect enough. For example, it is necessary to accurately search for a game name. If users accidentally type the wrong game name, they will not get the results they want, so the average score of this task is around 3.7. The average score of the seventh task is around 4.5, because there are very obvious add/remove/update buttons in the prototype to guide the user to complete the operation. The eighth task got the same high average score, because clicking the game in the drop-down list shows the overview is very in line with the habits of most users in using other softwares. The score of the ninth task is also very high, because there will be an 'Enter' prompt and an icon to open the door after each community icon in the game. It is easy for users to understand that clicking this icon will enter the corresponding community.

Tenth The eleventh and twelfth tasks all received an average score of 4 or more, because their interfaces are very common, similar to the SMS interface and the blog interface. The average score of the thirteenth task is only 3.8, which may only be because not all participants are familiar with the use of filters, perhaps older participants, or participants who do not usually surf the Internet often will feel this way confused.

Qualitative data analysis Method

In the part of qualitative analysis, we adopted the method of think-aloud. We asked 20 participants to use the system, and asked them to speak out loud about the difficulties they encountered when using the system or completing a task, and we recorded these thanks in real time. In the part of qualitative analysis, we adopted the method of think-aloud. We asked 20 participants to use the system, and asked them to speak out loud about the difficulties they encountered when using the system or completing a task, and we recorded these feedback in real time.

- When logging in, the login may fail due to the wrong account name or password entered, but there is not enough prompt message to explain whether the input user name does not exist or the password does not match the input user name, which causes the login failure, which is inconvenient for users to troubleshoot and resolve login failures.
- In the interface of searching communities and games, only typing input function is supported, and language input function is not supported, which may cause difficulty in

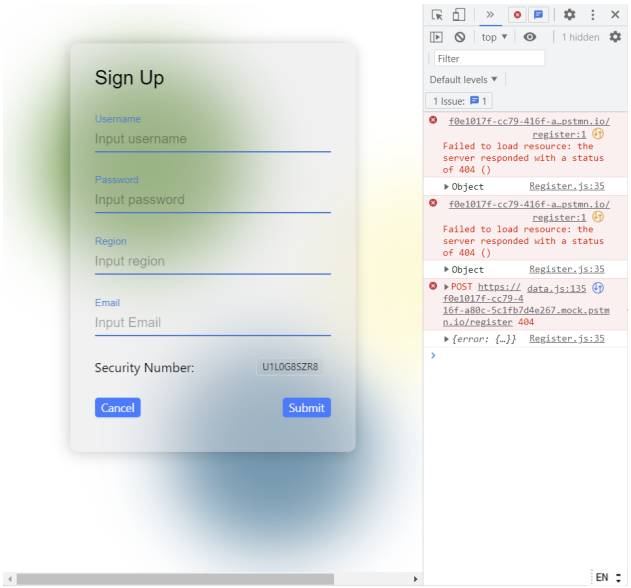
searching for people who are not familiar with typing or older people with degraded eyesight.

4. Code and Project Evaluation

4.1 Prototype Design Evaluation

We changed our architecture while developing the prototype. We switch to the B/S model (Browser/Server) from the MVC model. Our prototype is composed of two parts, the backend server, and the frontend user interface. Two developer implement the User Interface; And the other two people implements the server. In this section, we will introduce our product prototype by identifying the requirements prompted by the submission.

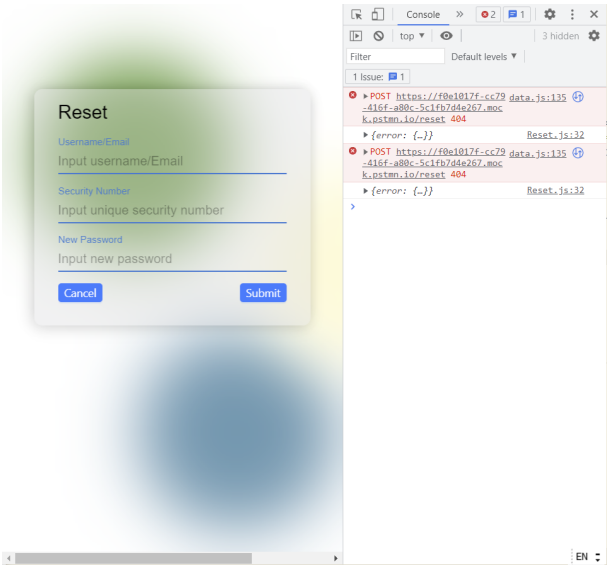
1. **User System** - Registration



Our product prototype fulfilled the registration functional requirement. The user can input a username, password, region, and email. In the prototype, we have also considered safety, we also add a security number randomly generated by an algorithm. After setting all data the system needed, then the user click the submit button to request the server create a new account. This request is a post Request.

By comparing with the original design, the component have improvements on UI design, each text area has its own label. This design is useful to help the user to input correct information.

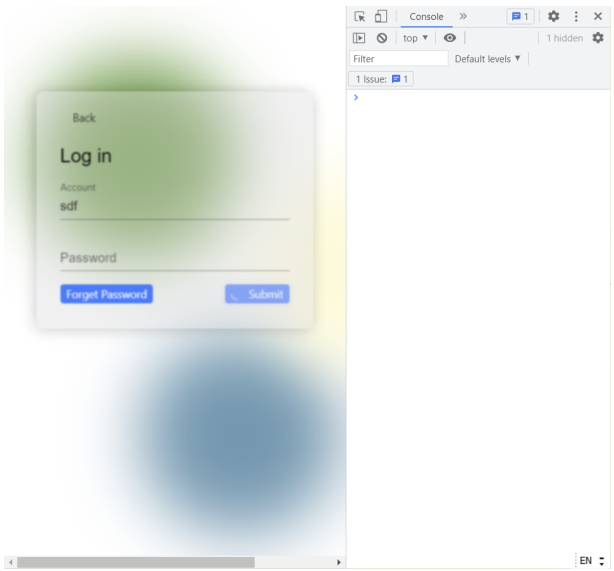
2. **User System** - Password Reset



This component implements the password reset functional requirement. The user can input their username or Email, unique security number, and new password. After setting all data, the user clicks the submit button to trigger the post request to the server. Then the server changes the password of the account.

By comparing with the original design, this component fully follows the original design. However, this design may be hard to use. The user may not be able to provide the unique security number due to an unexpected situation. The password reset function should support the email verification method. The user can verify themselves by email. This function needs improvement in future work.

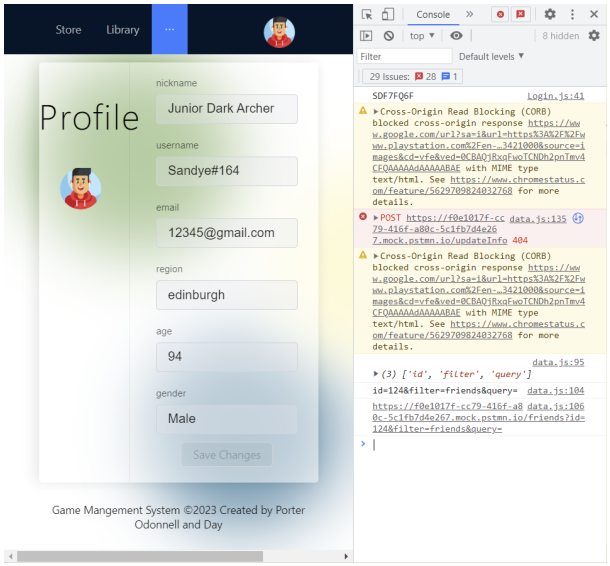
3. **User System** - Successful Login



This component implements login functional requirements. After the user inputs the correct account and password, the user clicks the submit button to request login service; Then the browser sends this request to the server.

The prototype does not provide error message when the user input wrong account or password. In a greater version, the login should provide proper feedback to tell the status of the system, such as login successful message or login message.

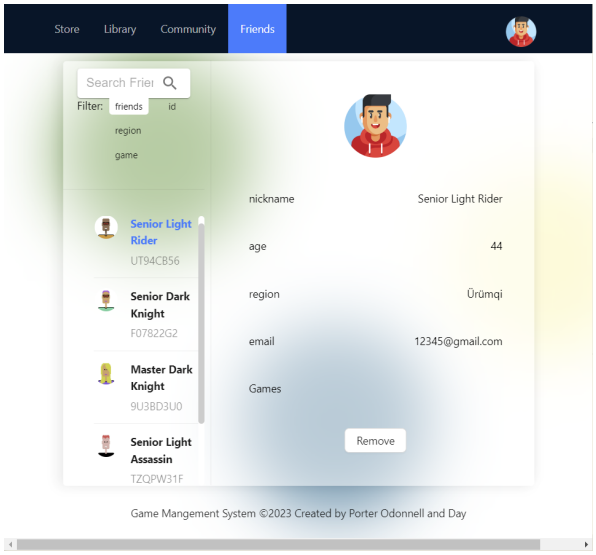
4. **User System** - Profile Update



Our prototype can update users' profile information. The use can change their personal information by inputting values to correspond input area. After that, the Save changes button will light up. It means that the user can save their changes.

This component not only displays the user's profile but also supports the user to modify their information. And the prototype follows the HCI heuristic principle that displays the status of the system. When the Save Changes button is grey, the user can't update their information. After the user inputs new information, the Save Changes button will change to blue; then, the user can request an update request to the server. This is a great improvement.

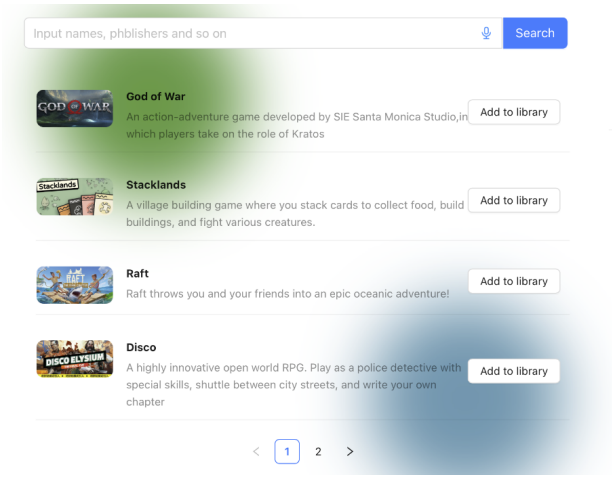
5. **Social Friends Network** - Search Friends



Compared with our original requirements, the friends component now supports the user to search for possible friends by id, region, and game. The user clicks the friend filter, the list will show all friends the user has. The user can click on one of their friends, and the left panel will show the profile information of the selected friend.

This part of the prototype does not fully follow the requirement. The prototype only supports the user to search for friends and add friends. There is no friend suggestion system. In submission two, we confirmed that we only implement a simple friend system. Since our main purpose is to implement a game management system. The complete functional social friend network may increase our additional development costs. Therefore, we only implement a simple friend network for demonstration use.

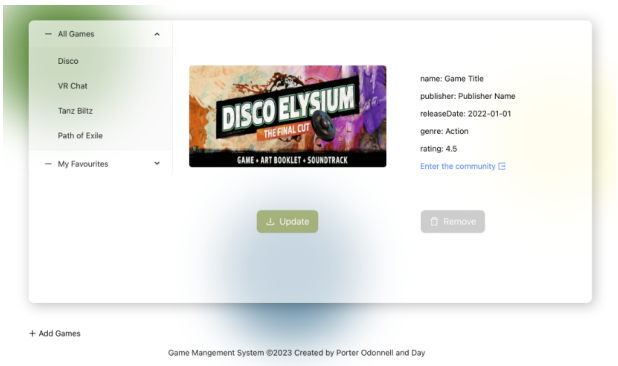
6. **Library Management System** - Game Search



In this component, we mainly implement the function of searching games. Users can get the corresponding search results by inputting the names, publishers and other information of the game in the input box and clicking the Search button. Compared with the original requirements, we did not filter the searched results by Region, Genre and other filter

conditions, because our database only stores 40 games, and the query generally returns only one result, so there is no need to further filter the query results. to filter. In order to improve this component, we need to optimize the query algorithm so that the query can not only support precise query, but also support fuzzy query.

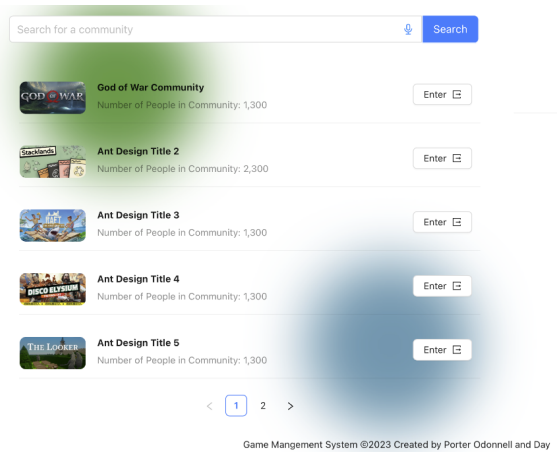
7. **Library Management System** - Game collection



In this component, we mainly realize the management of the user's game favorites. The user can click the 'All' drop-down list to get a list of all games in the favorites, or click the 'Favourites' list to get a list of the user's favorite games. At the same time, users can click the 'Add Games' button on the bottom left to jump to the store, which is convenient for users to search for more games. Click the game name in the list, and the introduction of the corresponding game will be displayed on the right, including name, publisher, released date, genre, rating and other information. And users can update this game or remove this game from favorites. Unlike the original requirements, this prototype will not provide a game shop, because our platform does not focus on the commercial sale of various games. In future work, we can improve the efficiency of favorites management, and we can remove or transfer favorites from the game directly in the drop-down menu.

8. **Social friends network system** - Community feature

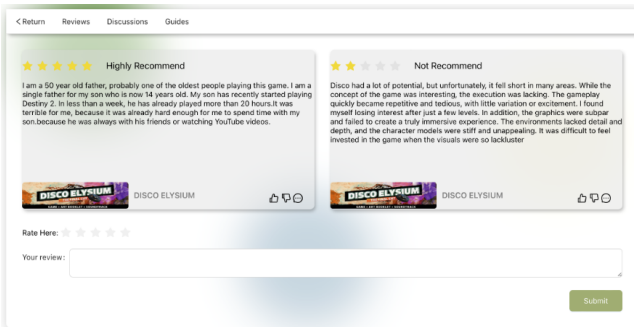
- Community Search



In this component, we mainly realize the user's search for the game community. The user enters the game name in the input box to get the information list of the corresponding

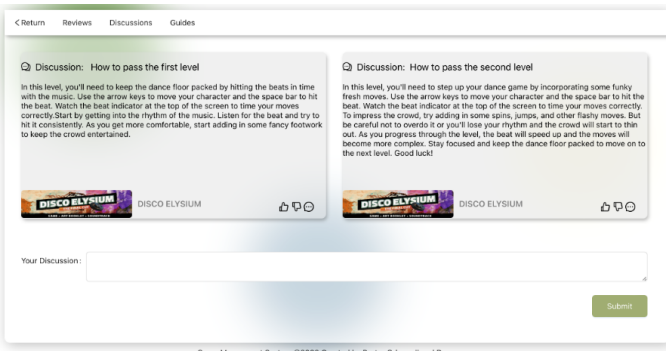
community, including which game the community belongs to and how many people the community currently has, and can enter the community through the Enter button. The design of this component is basically the same as the earliest requirements, and no major changes have taken place. In future work, we can optimize the query method. For example, our query now only supports text query, but this will cause great obstacles for people who are inconvenient to type, so we can add a voice query function to help special groups of people complete the query operation.

- Reviews



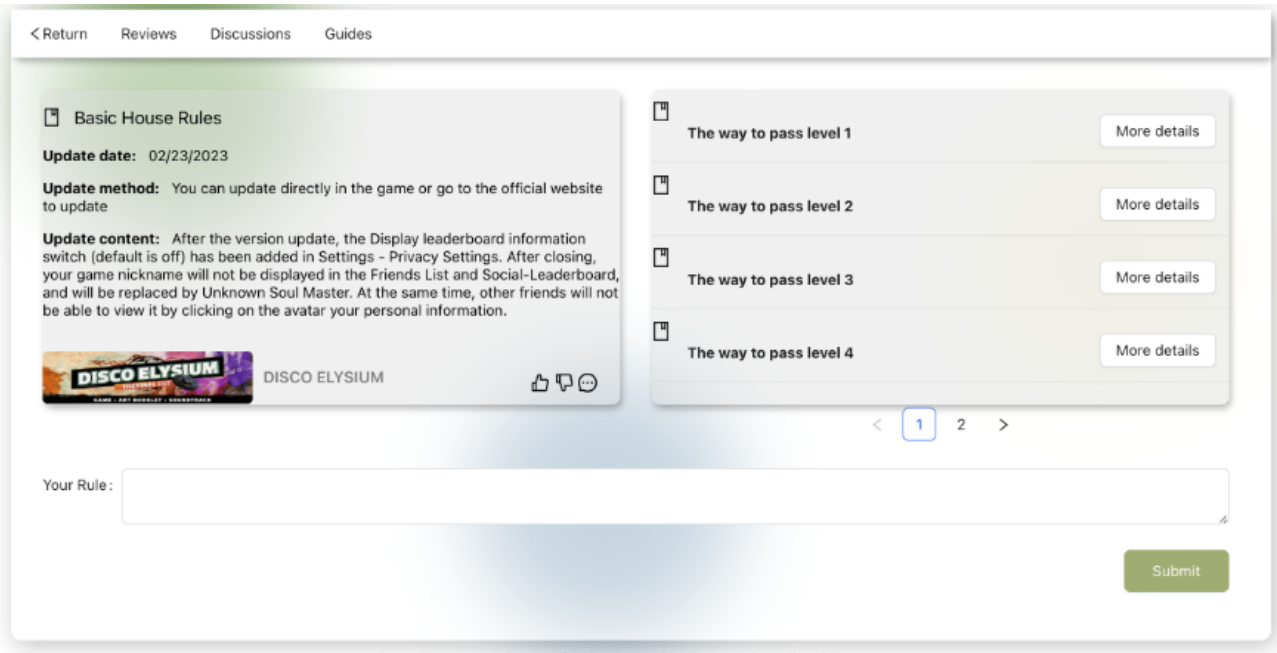
In this component, we mainly implement the function of users browsing and posting reviews to the community. Users can browse the reviews of this game, including ratings and comments on this game. Users can also add their own ratings and opinions about the game. This component is basically the same as described in the original requirements. In future improvements, we can filter these comments. For example, some comments that are not very helpful can be filtered out and do not need to be presented to users.

- Discussions



In this component, we mainly implement the function of users browsing and posting discussions to the community. Users can browse the discussions of this game. Users can also add their own discussions bout the game. This component is basically the same as described in the original requirements. In future improvements, we can filter these discussions . For example, some discussions that are not very helpful can be filtered out and do not need to be presented to users.

- Guides



In this component, we mainly implement the function of users browsing and posting guides to the community. Users can browse the guides of this game. Users can also add their own guides bout the game. This component is basically the same as described in the original requirements. In future improvements, we can filter these guides. For example, some guides that are not very helpful can be filtered out and do not need to be presented to users.

4.2 Training Evaluation

Each development team is composed of people with different levels of skills. Therefore, preparing necessary training sessions or training materials are useful to reduce the risk of technique. In our team, we use the B/S model. The team member should have strong front-end development skills and backend skills. In the original plan, we did not illustrate the training plan. In the actual developing process, we have pair programming sessions periodically. Firstly, the User interface structure is developed by the member who has strong react development skills. Then the raised issue will be solved by one of the frontend developers. The backend team uses the previous project's structure; they studied previous experience by looking at previous project documents.

4.3 Code Review

Code Review is an important stage during the process of software development. Frequent Code Review takes much time for the whole project. Our team did not plan the code review activity. In the actual development environment. The code is reviewed by the member who merges the code branch. In the following paragraph, we will discuss some advantages and issues we have found during the code review.

1. **The Encapsulation of request Functions**

In our source code, we need to request data from the server in many component. And all request methods' logic are quite same. Therefore, we encapsulate the GET request and POST request into two functions. Every time, we need to request the data, we can directly call two encapsulated function.

2. **Modular**

   In our source code, the prototype is composed of a server component. Each component has its own layout. We strictly create an independent folder to store each component. This design helps us to quickly locate the bugs. It greatly speeds up our development.

3. **Failed State Management**

   In the front part, the code is finished by two team members. In the react structure, each component can have its data attributes. During the code review, we noticed that team members do not follow the structure, they abuse the state variable during the implementation of each component. All variables should be managed by single one modular.

4. **Duplicate pieces of code**

   In the server part, API are implemented by two team members. Due to lack of communication, there are many API have similar code. A better solution is writting API doucmentation to the developer team. Therefore the developer can avoid writting duplicate code by looking at the documentations

4.4 Project Teams Review

In the following section, we will discuss the review of Project Teams. We will compare the original planning and the actual development. We will check what we have followed in the original plan and what we did not follow.

4.5 Risk Mangement

1. In the original plan, our solution to unexpected occurrence is making preparation for them. In the actual plan, the team have faced scheduling problem. two team mebmber are not in edinburgh during the development duraton. It is difficult to process cooperate work. Some detailed work are not suitable solved by online meeting. Our actual solution is transfering the work to available team member who is able to take offline meeting.

2. In the original plan, our solution to Technical issue is providing training session to support team memebers. In the actual development, we use pair programming to solve technical issue. Team member who have strong skills takes time to support team member who has technical problem to finish implementation.

4.6 Time Estimates

In the original plan, we confirmed that we have three weeks to implement the prototype. Every task is assigned a target deadline. If there is no any risk appear during the cycle of development, we can implement the core function of the prototype. In the actual development, each team member is not able to fully follow the schdule since they have other works, assessment, and planned extracurricular activities. Therefore, our project has acutally exceeded the scheduled deadline. We overlooked personal activity when we made the development schdule.

4.7 Tasks Management

In the original plan, our project development was divided into six parts that were backend, frontend, database, architect, end-user liaison, and DevOps. There are too many parts. It is hard to assign tasks to the team member. In the actual development, the development was roughly divided into two parts. two people were responsible for implementing the frontend, two people were responsible for implementing server. Two same team works separately. After they finished their work. They started to integrate the front end and the server. However, the tasks were not appropriate since we faced so many technical issues. Our team did not finish all the confirmed tasks.

4.8 Interaction

According to our original plan, we intended to create individual branches on GitLab to enable team members to develop specific system features and address any bugs. We believed that leveraging GitLab for this process would result in a reduction of both time and development costs. During the actual development phase, we adhered closely to the original plan. Each team member created a new branch using their name as the branch title, branching off from the master branch. All team members then proceeded to implement their assigned tasks in their own branch. After all the features had been successfully implemented, the team convened a coding meeting to merge all of the branches back into the master branch. By closely monitoring the commit history, we were able to effectively manage the development stage. The commit history helped us to ensure that each team member implemented the relative UI or API.

4.9 Post Project Review

1. **What has been done**

   In this project, we developed a comprehensive project plan to guide our team's progress, although we did not strictly adhere to it as the project unfolded. Upon realizing that the original plan would require a longer implementation timeline, we abandoned the initial architecture and instead adopted a B/S (Browser, Server) development model. This model proved to be simpler and faster than the MVC model we had originally intended to use, allowing team members to focus on their respective areas of expertise. Ultimately, we

successfully integrated the server and frontend Web user interface, resulting in a functional web application prototype that can perform basic functions.

2. **Strengths**

   Our team have faced some unexpected situation, such as technique issue, inconvenient communication, time mangement issue. However, Due to the choice of an easier development solution and and use of gitlab, our project did a greater implementation in three weeks from the coding to integration and submitting. On the other hand, in the actual development, we will peridoically evaluate each planed function. If the function is not necessary to the application, we will give up immediately. Our project is very flexible.

3. **weaknesses**

   We did not give sufficient consideration to risk management in our project, which resulted in inadequate solutions to address the issues that arose. As a consequence, our development timeline was extended due to unresolved problems. In order to enhance the success of future projects, it is crucial to carefully consider potential risks and develop detailed and appropriate solutions to mitigate them.

4. **Important Lessons**

   During the development process, our primary challenge was technical issues for which we lacked proper solutions in the short term. This experience has taught us the importance of regularly allocating time for team members to undergo training and improve their development capabilities. Strengthening the team's technical expertise is crucial for accelerating project development.

   Additionally, it is crucial to select a suitable development model and promptly abandon unrealistic plans in order to effectively develop a project. The right development model can save time and reduce technical issues, while giving up unrealistic plans can help to lower project costs.

   In addition, it is imperative to take into account the project timeline in a more comprehensive manner. In our previous project, we failed to consider variables such as weekends and the amount of time each team member was willing and able to allocate to the project. As a result, our actual development time fell short of the planned total time. To avoid encountering similar issues in the future, it is essential to create a more detailed and thorough schedule.

   One approach is to request that each team member provide a clear accounting of their available time at the outset of the project. This can include a breakdown of their work hours on weekdays, weekends, and any upcoming vacations or personal commitments. By

collecting this information early on, we can take into account each team member's unique schedule and availability when creating the project timeline.

Furthermore, it is crucial to ensure that the schedule is realistic and achievable. This involves carefully balancing the available resources with the project requirements, taking into account potential delays, and allowing sufficient time for testing and debugging. By creating a detailed and well-considered schedule, we can avoid overburdening team members and ultimately achieve project success within the planned timeline.