Requirements, Design and Team

for

Software Development

Version 1.0

Prepared by : 1. Meghna Raje
2. Logan Li
3.Xinmeng Chen
4.Yuchen Pu

March 1, 2023

Contents

1 Requirements engineering

The system is divided into 4 categories - User Management System, Library Management system, Social friends network system and Game collection expansion.

1. User Management System - The user management system allows users to create accounts, manage their collections, and access various features of the system.

2. Library Management system - The system should allow users to manage their game library by adding, updating, and deleting things. It should also be possible to search for and filter games based on characteristics such as game genre, mechanics, and themes.

3. Social friends network system - The system should allow users to search for prospective new gamers to play with and gaming clubs in a specific geographical search region.

4. Game collection expansion - The game collection expansion system should be scalable to accommodate a growing user base and increase in game collections.

The following sections contain functional and non-functional requirements, along with their priority, description, and other details.

1.1 Functional requirements

ID	FR01
Category	User System
Sub Category	Registration
Description	New users should be able to set up an account by entering their email address/phone number, creating a profile, and selecting a password.
Severity	High
Priority	High
Explanation	It has high priority since it affect the main function. The user system must support registering before logging in. Therefore, it is important.

ID	FR02
Category	User System
Sub Category	Succesful Login
Description	The system should provide the functionality for users to authenticate themselves using their email address or phone number, along with a corresponding password.
Severity	High
Priority	High
Explanation	A login system is required to verify users' identities before providing them access to a system. By ensuring that only authorised personnel have access, it helps to avoid illegal access, data breaches, and other security threats.

ID	FR03
Category	User System
Sub Category	Password Reset
Description	Users who have forgotten their password can recover access to the system without having to register a new account. This can enhance the user experience and reduce frustration.
Severity	Low
Priority	Low
Explanation	Although password reset may have a low impact on the daily use of the system, it is still a crucial security feature that should not be overlooked. Therefore, it has low priority and severity.

ID	FR04
Category	User System
Sub Category	Profile Update
Description	Users should be able to change their profile information, including basic details such as their name, location, age, gender, and display photo, through the system. This can enhance the user experience by allowing users to customise their profiles and ensure that their information is correct and up to date.
Severity	Medium
Priority	Low
Explanation	The profile update feature has a minor impact on overall system performance. As a result, its severity is modest, and it may be treated with lower priority than more essential features. .

ID	FR05
Category	Library Management system
Sub Category	Game collection
Description	The system should have a centralized list that allows users to compile and organize their games.
Severity	High
Priority	High
Explanation	The game collection has a major impact on overall system performance. As a result, its severity is high, and it may be treated with high priority than essential features. .

ID	FR06
Category	Library Management system
Sub Category	Custom List
Description	It should be possible for users to add, update, and remove games from their library. Users should be able to search for games in their collection using a variety of parameters such as name, game type and themes.
Severity	High
Priority	High
Explanation	The main function of this system is to display a collection of games, making it a high-priority feature. Additionally, multiple functions rely on this feature, making it crucial. Therefore, the severity of any issues with this function is high. .

ID	FR07
Category	Library Management system
Sub Category	Modify Custom List
Description	Users should be able to add/remove/update games into the main list and custom lists.
Severity	High
Priority	High
Explanation	The ability for users to manage the game lists is a critical function of the system as it directly impacts the user experience and satisfaction. .

ID	FR08
Category	Library Management system
Sub Category	Game Overview
Description	The system should display overview information of each game after clicking.
Severity	High
Priority	High
Explanation	The system should store information about each game in the collection, including the name, publisher, year of release, game type, mechanics, and other relevant details. These details should then be displayed to the user including reviews and rating for the game.

ID	FR09
Category	Library Management system
Sub Category	Game review and rating
Description	Users should be able to see and post their own game reviews and ratings from a user community and websites.
Severity	High
Priority	High
Explanation	The system should store information about each game in the collection, including the name, publisher, year of release, game type, mechanics, and other relevant details. These details should then be displayed to the user.

ID	FR010
Category	Library Management system
Sub Category	Game Search
Description	Users should be able to search the game shop for games that they do not have.
Severity	Medium
Priority	High
Explanation	The system should display to users each game's name, publisher, year of release, game type, mechanics, and other relevant details .

ID	FR11
Category	Social friends network system
Sub Category	Community feature
Description	To allow players to communicate and connect with other gamers, the system should have community features such as forums, chat rooms, and message systems.
Severity	High
Priority	High
Explanation	The system should allow users to form groups, organize events, and share game-related content .

ID	FR12
Category	Social friends network system
Sub Category	Game buddy suggestion
Description	The system should utilize location-based matching to suggest potential users with similar gaming preferences and located in the same region to the users.
Severity	High
Priority	High
Explanation	The system should allow users to form groups, organize events, and share game-related content .

ID	FR13
Category	Game Collection Expansion
Sub Category	Game Suggestions
Description	The system aims to enhance user experience by providing personalized game recommendations based on their collection, ratings, location, and reviews. The suggestions may also include potential gaming communities.
Severity	Low
Priority	Medium
Explanation	This requirement is optional and not required for the system's basic operation. Even if this functionality is not available, users can search for games depending on their likes. Implementing this feature, on the other hand, can improve the user experience and bring more value to the users. .

ID	FR14
Category	Library Management system
Sub Category	Rules and Clarifications
Description	Users should be able to add and see game rules and clarifications.
Severity	Low
Priority	Medium
Explanation	The system should allow users to access FAQs from game publishers, giving them more resources to answer any potential difficulties or inquiries. .

ID	FR15
Category	Library Management system
Sub Category	Cost and Inventorys
Description	The system should give cost and availability information for games.
Severity	Low
Priority	Medium
Explanation	Users should be able to examine and compare game pricing from various sellers. The system should provide hyperlinks to retailers from which users may buy games. .

1.2 Non-functional requirements

ID	NFR01
Category	User System and Library Management System
Sub Category	User Friendly UI
Description	The system should be user-friendly and easy to navigate. It should have clear and concise error messages and provide helpful feedback to users when they perform actions.
Severity	High
Priority	Low
Explanation	Due to the critical role of displaying the game collection, it is of high priority and significance within the system. Additionally, as several other functions depend on it, the severity of any issues with this function is also high. .

ID	NFR02
Category	Library Management system
Sub Category	Reliability
Description	The overview page should be responsive and handle user requests quickly, even during peak usage times. Response time should be fast, and the system should be scalable to handle increasing numbers of users.
Severity	High
Priority	High
Explanation	The system should be very dependable, with little downtime and no data loss. To guarantee that the system remains functioning in the event of a breakdown, it should have robust error handling. .

ID	NFR03
Category	All
Sub Category	Security
Description	The system should be safe and protect user data from unauthorised access.
Severity	High
Priority	High
Explanation	To ensure the safety of users' personal information, the system should implement secure data storage and transfer technologies, including encryption and secure authentication.

ID	NFR04
Category	All
Sub Category	Scalable and Compatible
Description	The system should be scalable in order to handle an increasing user base and game collection. It should also support various devices and platforms, including laptops and mobile devices
Severity	High
Priority	High
Explanation	The system should be built to manage growing load while maintaining performance. Also, Web browsers and mobile applications should be able to access the system.

ID	NFR05
Category	All
Sub Category	Accessibility
Description	The system should be designed to be accessible for users with disabilities.
Severity	Low
Priority	Low
Explanation	The system should be designed to accommodate users with disabilities, ensuring that users with visual or hearing impairments can also use the system without difficulty.

ID	NFR06
Category	All
Sub Category	Backup and recovery
Description	The system should perform frequent data backups and have recovery procedures in place.
Severity	Low
Priority	Low
Explanation	If needed, the system should be able to restore data or revert to prior versions of the system.

ID	NFR07
Category	All
Sub Category	Local Language and currecy
Description	The system should handle different languages and curreny.
Severity	Low
Priority	Low
Explanation	The system should support several regional settings such as dates, times, and currencies.

1.3 Capture methodology

For our project, we have decided to use the Agile methodology for development. Agile methodology is flexible and iterative, allowing for continual improvement and adaption to new needs throughout the project. This means that we can respond to input rapidly and alter the project scope or features as needed, rather than having to start from zero. Agile also promotes cooperation, openness, and customer satisfaction, which aligns with our project aims of developing a user-friendly and efficient system.

Waterfall was not chosen for this project because of its rigid, sequential approach, which requires an upfront set of well-defined requirements. This method might be challenging in software development projects with changing requirements. If changes are required later in the development process, Waterfall's lack of flexibility can cause project delays or failure. Although Waterfall is better suited for projects with well-defined requirements and a definite timeline, Agile was found to be a better match for this project due to its dynamic nature, need for flexibility, and capacity to react to changes in requirements during the development process.

2 Software Design

2.1 Technology Breakdown

2.1.1 Front-end

React has been chosen as the library for developing the front-end of our applications, as it helps us to design and implement an interactive user interface. After considering the following advantages of React, we concluded that it is an excellent option for the front-end development of our application.

1. **Supports Cross-Platform Application Development**. By using React JS applications can be built for a variety of platforms, including iOS and Android.
2. **Offers Code Reusability**. React supports code reusability. It offers several advantages. For example, it allows the same code to be reused across multiple applications for similar functionality. As a result, it can significantly reduce development time.
3. **The efficiency, flexibility and intuitive nature**.

2.1.2 Back-end

Django is a popular choice for back-end development frameworks due to its standout features and capabilities.

At the start of the project, we evaluated both Python and Java for the back-end development of the application as they are both powerful back-end development languages. We considered Python for its outstanding simplicity and ease of use, and because it aligned more with the group's previous programming experience. While Java is generally faster than Python due to being compiled into bytecode, it is more suited to building complex or enterprise applications, whereas the application we are developing is data-intensive. Python has more expertise in this area, making it the clear choice for this project.

After deciding on Python as the programming language for the back-end of the application, we needed to choose a suitable framework. Django and Flask are two competitive frameworks for back-end development in Python. However, we ultimately chose Django for its key features, including an ORM for working with databases, an administrative interface for managing application data, and a templating engine for rendering HTML. Django is designed to help developers build web applications quickly with less boilerplate code and fewer decisions about application architecture. Flask, on the other hand, is a more lightweight framework that gives developers more control over their application structure. Flask is flexible and extensible, with a minimal core that can be extended with plugins and third-party libraries as needed. Flask provides a basic set of features for web development, but it lacks some of the built-in functionality of Django. So Django is much better at building large, complex content management systems with lots of built-in functionality than Flask, which fits right in with the Game Collection Management and Sourcing System we're developing.

2.1.3 Database

For the database, we decided to use MongoDB based on the fact that members of the group had more experience with MongoDB. Although Django does not provide built-in support for MongoDB due to it being a No-SQL database, there are many third-party libraries and extensions that allow Django to be used in conjunction with MongoDB, such as MongoClient. MongoClient can be used with djongo to provide a connection to a MongoDB server and allow Django to interact with the data in MongoDB. These packages provide an interface that allows Django to use the familiar ORM syntax to work with MongoDB collections, documents, and queries.

2.1.4 APIs

When using React as the front end and Django as the back end, we can use the Django REST framework (DRF) to build APIs that help the front end have access to all the data it needs including user data, user comments and community information.

2.1.5 Hosting

To allow scalability, high availability and secure data storage for applications, the system will be hosted on the Google Cloud Platform. We will create a container on GCP to act as our server. A container is a lightweight, standalone executable package that contains everything needed to run a software application. After the server has been successfully set up, the front-end and back-end of the application will be deployed to the server by using a containerisation tool such as Docker. Finally, once the front-end and back-end have been deployed and a connection has been established between the front-end and back-end containers to allow the front-end to send requests to and receive responses from the back-end, configuring security and access will be the final step in making the application available to the outside world. This includes configuring firewalls and access control.

2.1.6 Version Control

GitLab can help with version control as it is a powerful and flexible source code management tool that makes it easy to track changes to the codebase over time. With GitLab, we can create a centralised repository for the application's code and use Git to manage changes to the codebase.

Here are some of the ways that GitLab can help with version control:

1. **Track changes to the code**: With GitLab, it's easy to track changes to your codebase over time. Every time a change is made to your code, GitLab creates a new commit to record the changes. Commit history also supports viewing who made each change and when.
2. **Collaborate with teammates**: GitLab makes it easy to collaborate with other group members in the codebase. Each person can create branches for different features or bug fixes and merge them into the master branch when they're ready. We can also use GitLab's merge request feature to review and approve changes made by others before merging them into our own master branch.
3. **Roll back changes**: We can easily roll back changes to our own codebase via gitlab, with support for restoring a commit or merge request to undo changes previously made.

2.1.7 Testing

We employ Pytest as our testing framework. Considering that Pytest has a cleaner and more flexible syntax than Unittest, and that Pytest has a wealth of plugins and extensions to provide additional information such as test coverage, test parameterisation and test reporting, Pytest would be a better choice for testing the application's code.

2.2 Component and System Design

2.2.1 High-Level Architecture

The model-view-controller (MVC) architecture is the best fit for the aforementioned problem among the available architectural styles. The programme is divided into three interrelated components by the MVC architecture as seen in 2.1: the model, the view, and the controller.

In this scenario, the model represents the data and business logic, the view the user interface for presenting the data, and the controller takes user input and regulates the data flow between the model and view. This design is ideal for online applications since it enables scalability, maintainability, and flexibility.

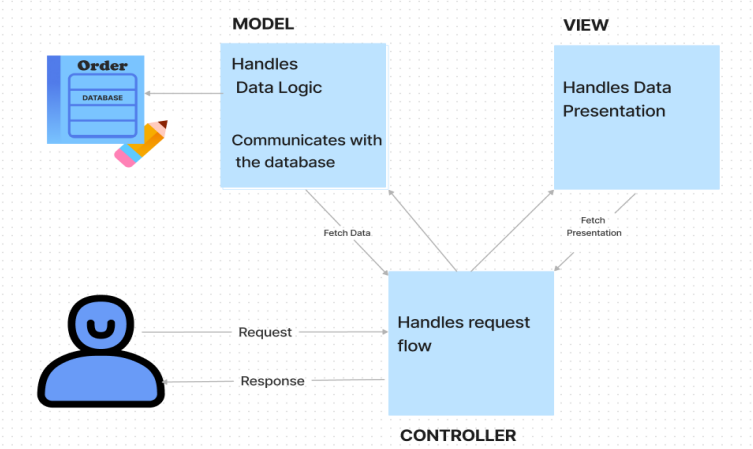


Figure 2.1: Overall system diagram

Layered structures, such as the n-tier architecture, may potentially be appropriate for the aforementioned situation. They are, however, more complex and inflexible than the MVC architecture.

Pipe and filter and shared-data designs are not well suited for this challenge because they lack a clear separation of concerns, making system maintenance and updating difficult.

Client-server architecture might be used to solve the above problem, but it may not be the best option because it is often employed when there is a clear distinction between the client and the server. In this situation, the system's interaction between multiple components is likely to be complicated, and the MVC design can give a better structure to handle this complexity.

In conclusion, the MVC design is best suited to the scenario since it allows for a clear separation of concerns.

2.2.2 Detailed Architecture

Here is a complete design using the Model-View-Controller (MVC) architecture. The following components are included in the design as seen in 2.2 :

1. Model - The Model represents the system's data and business logic. It is in charge of managing the database, storing user data, game data, user collections, ratings, and reviews, and interacting with external APIs.
    a) User model: includes user information such as email, phone number, profile information, password, and game collection.
    b) Game model: provides information on each game in the collection, such as the game's name, publisher, year of release, game type, game price, link to the seller, and other relevant information.
    c) Community model: includes information about game id, group name, group members, rules defined by particular community, rating given.
2. Controller- The Controller serves as a go-between for the Model and the View, taking user input from the View and connecting with the Model to perform data operations.
    a) User controller: manages user-related operations such as registration, login, reset password and profile updating.
    b) Game controller: is in charge of game-related tasks such as adding, updating, and deleting games from the user's library, searching for games, and showing game overviews.
    c) Community controller: manages community-related activities such as group formation, event planning, and exchanging game-related information.
    d) Social friend network controller: enable users to connect with one another, issue friend requests, accept or reject friend requests, browse friends' profiles, and send messages or chat with one another.

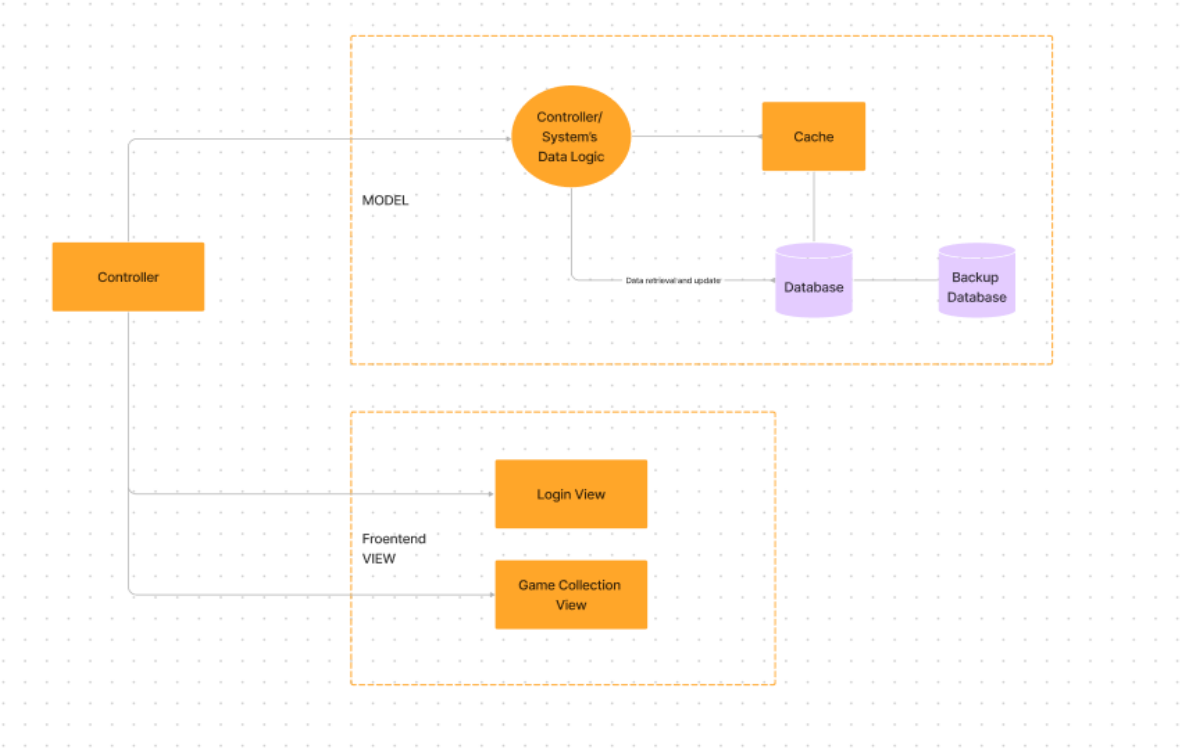


Figure 2.2: Detailed Architecture Diagram

e) Game suggestion controller: handles tailored game suggestions and prospective gaming community recommendations.

3. View- The View is in charge of displaying data to users and receiving user inputs.
   a) Registration view: presents a form where new users may input their email address/phone number, create a profile, and choose a password.
   b) Login view: provides a form where users may authenticate themselves by entering their email address/phone number and password.
   c) Game collection view: shows a consolidated list of games in the user's library, as well as the ability to add, update, and delete games from the library.
   d) Game search view: allows users to search for games in the collection by name, game type, and themes.
   e) Game overview view: shows extensive information about each game in the collection, such as reviews and ratings.
   f) Community view: shows community features including forums, chat rooms, and messaging systems allowing players to communicate and connect with other gamers.
   g) Gaming buddy recommendation view: uses location-based matching to identify possible players in the same region who have similar gaming likes.
   h) Friend list view: contain a list of all the user's friends, with their profile pictures and usernames. You can also include options to send messages or invites to play games together.

4. Database Backup- A database backup is an essential part of any design, particularly for web applications. It includes making a backup of the database and storing it somewhere else, usually outside. This backup guarantees that the data can be restored to its prior condition in the case of a disaster, such as a server failure or a data breach.

5. Cache- A cache is a temporary storage location that caches frequently requested data to boost application speed. It decreases the amount of database requests and enhances the application's response time.

2.3 Implementation Paths

For the implementation plan, we have divided the design and implementation of the application into five main phases: requirements collection, design, development, testing and evaluation, and finally maintenance. Each phase involves multiple implementation goals, and the steps are carried out sequentially one after the other. The following are the specific tasks for each phase, along with the corresponding implementation goals.

- **Requirements collection**
  1. Elicitation: Identify and collect requirements.
  2. Analysis: Analyze requirements for completeness, consistency, and feasibility.
  3. Specification: Document requirements in a clear and concise manner.
- **Design**
  1. User Interface Design: Design an intuitive and accessible user interface.
  2. Architecture Design: Design the system architecture, including front-end and back-end components.
  3. Database Design: Design the database schema and establish relationships between entities.
- **Development**
  1. Front-end Development: Develop the user interface using React.
  2. Back-end Development: Develop the server-side logic using Django.
  3. Database Development: Develop the database, including schema creation, population, and data retrieval using MongoDB.
- **Testing and Evaluation**
  1. Unit Testing: Test individual components of the system for functionality and correctness.
  2. Integration Testing: Test the integration of front-end, back-end, and database components.
  3. Usability Analysis: Use the results of unit and integration testing to present evaluations and conclusions on usability.
  4. Project Evaluation: Evaluate the state of the application prototype and propose a plan for the potential further development of the application.
- **Maintenance**
  1. Updates and Bug Fixes: Address any bugs and make updates to the application as needed
  2. Scaling: Scale the application to accommodate growth and potential further development based on the project evaluation.

The implementation paths is visualised as shown in the figure 2.3.

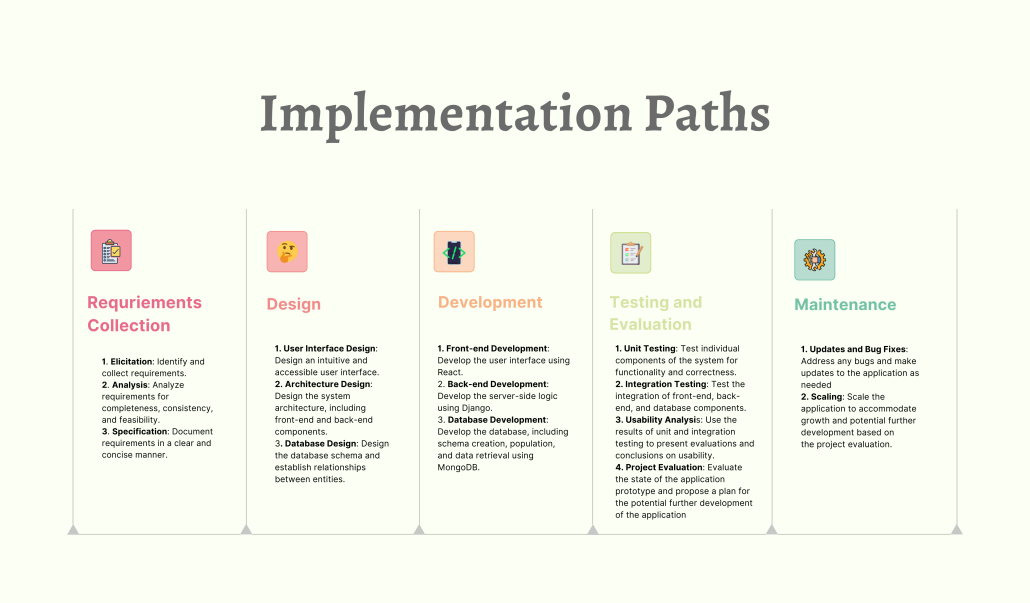


Figure 2.3: visualised Implementation Paths

3 Team structure and role assignment

3.1 Identified roles

The following are some frequent roles in software development, divided into management, development, quality assurance and risk management:

1. Management Roles
    a) Project Manager: In charge of project planning, organisation, and supervision.
    b) Product Owner: Represents the consumer and ensures the product satisfies their requirements.
    c) Scrum Master: A facilitator that ensures the Scrum team follows Agile principles and practises while also assisting in the removal of any barriers that may prevent the team from accomplishing their goals.
2. Development Roles
    a) Backend Developer: Creates server-side logic and APIs to guarantee the application's backend operates properly.
    b) Frontend Developer: In charge of creating the application's user interface and user experience, ensuring that it is responsive, intuitive, and visually appealing.
    c) Architect: Creates the overall system architecture.
    d) Database Administrator: Manages the system's database.
    e) End-user liaison: Included in the requirements engineering process. The major purpose of the end-user liaison function is to bridge the gap between the development team and end-users in order to guarantee that the end-users' requirements and expectations are satisfied throughout the project.
    f) DevOps Engineer: Assures that the system is correctly deployed, configured, and maintained.
3. Quality Assurance
    a) QA Analyst: Verifies that the system satisfies the requirements and is free of defects.
    b) Test automation engineer: creates, builds, and maintains automated tests for software applications.
4. Risk Management
    a) Risk manager: Identifies and analyses possible project hazards, as well as develop ways to avoid or mitigate such risks. They guarantee that the project is implemented in a way that minimises the possibility and effect of adverse outcomes.

3.2 Roles assigned in the team

As the team comprises only four members, some roles need to be merged, and each team member may have to take on multiple responsibilities.

1. Each team member is contributing to risk management by identifying and analysing risks relevant to their area of work.
2. We took into account each person's preferences and past experience when assigning roles and responsibilities.

Member Name	Roles Assigned	Reason
Meghna Raje	Project Manager, End-user Liaison, Architect, Backend Developer	Meghna's prior experience as a software developer makes her suitable for the role of backend developer in this project. She has experience in gathering requirements and ensuring the needs of end-users are met, making her a valuable end-user liaison. Additionally, her expertise in designing systems as an architect will be beneficial to the project. Meghna has also worked closely with project managers in the past and can provide support in project management.
Yuchen Pu	Product Owner, Frontend Developer, Test automation engineer	Yuchen's previous experience in software development and Human and computer Interaction given him the programming skill and design insight to develop front-end, back-end, database.
Logan Li	Backend Developer, DevOps Engineer, Database Administrator	Logan's previous experience in software development has given him the skills and insight to develop the back-end and implement the application's database. His experience in DevOps also gives him the ability to help teams deliver applications and services at a high level of organisation and speed.
Xinmeng Chen	Frontend Developer, End-user Liaison, QA Analyst	Xinmeng's experience in software development has equipped her with the skills and insights necessary to proficiently implement the front-end of and QA analysis the application. Through previous projects, she possesses the technical expertise to ensure the application's front-end is user-friendly and functional. Her experience in human computer interaction also helps her to figure out how to meet users' requirements, which is helpful for an End-user Liaison.

3.3 Communication Process

These are some communication methods that we follow:

1. Weekly standup meeting: As a team, we meet every Wednesday at 12 pm to provide updates on our progress, discuss any issues, and plan our work for the next week. This helps us stay on track and ensures everyone is on the same page.

2. Open Communication Channels: As a team, we established open communication channels for the team to easily reach out to each other for help or clarification, such as instant messaging on Microsoft teams or email.

3. Documented Communication: We made sure to document all the communication processes that we established for the project, which helped to ensure that everyone was aware of the processes and avoid any misunderstandings or miscommunications. To do this, we utilized Google Docs.

4. Collaboration Tools: We decided to use Overleaf and Figma as collaboration tools as a team. These tools assisted us in successfully coordinating our work, keeping everyone on the same page and ensuring that all team members were informed of any changes or updates.

5. Issue Resolution Calls: We also used team calls when we were coping with a particularly tough problem that required input from multiple team members. These sessions enabled us to analyze the problem in real time, share our screens to explain it, and coordinate to find a solution.

3.4 Work Assignment

To ensure effective task management and collaboration on the project, we will implement a structured approach to work assignment based on each team member's role and responsibilities. The GitLab issue tracker will serve as our primary tool for task management, enabling us to assign tasks with a clear start and end date, a detailed

description, and a specific team member responsible for its completion.

Tasks will be assigned based on the role of the team member, such as front-end developer, back-end developer, tester, or project manager. This will ensure that each team member works on tasks that align with their skills and responsibilities. When a new task is identified, it will be added to the issue tracker, and the team member responsible for that area of the project will be notified via email and can view the task on their personal dashboard within GitLab.

As work progresses, team members will update the status of their assigned tasks and add any necessary comments or notes. The project manager will regularly review the status of each task and make adjustments as needed to ensure that the project stays on track.

In addition to task assignment and management, we will also implement a code review process to ensure the quality of our code. Each team member will be responsible for submitting their code for review by at least one other team member, who will check for adherence to coding standards, functionality, and overall quality. Code reviews will be tracked within the issue tracker, with each review being assigned as a separate task.

By using the GitLab issue tracker for task assignment, management, and code review, we will be able to effectively collaborate as a team, prevent duplication of effort, and ensure that our code meets the highest standards of quality and functionality.

4 User Interface Design

This Prototype Link Figma Prototype

4.1 Control Flow

Register Request
Pre-request: The user must be on the home page.
Path: To sign up for an account, the user clicks the "No account? Click Here" link, which will navigate them to the Sign Up page. Once on the Sign Up page, the user will need to input their name, password, email address, and region. After all of the required fields have been filled out, a security number will be provided. It is important for the user to store this security number in a safe place. Finally, the user can choose to click the "cancel" button to return to the Home page, or the "submit" button to complete the registration process.
Sender: User

Login Request
Pre-request: The user must be on the home page
Path: The user clicks the Login Button on the home page. This button navigates to the Login Page. On the page of Login, the user inputs the name or email address. Next, the user can click the Submit button to log into the system or click the NeedHelp button to reset the password.
Sender: User

Reset Password Request
Pre-request: The user must be on the Login Page
Path: The user clicks the NeedHelp button on the Login Page. This button navigates to the reset password page. On the reset password page, the user inputs the user name/ email address, unique security number, and new password. The user can click the cancel button to go back to the login page or click the submit button to change the password and go back to the login page.
Sender: User

Check Profile Request
Pre-request: The user must be on the main system page.
Path: The user clicks the user icon on the right side of the navigation bar. The user icon navigates to the profile page. On the page of the profile, the user can check the nickname, username, email address, region, age, icon, and gender.
Sender: User

Update Profile Request
Pre-request: The user must be on the profile page.
Path: The user clicks the input field of the nickname, region, age, and gender. Next, the user can input updated information. Then, the user can click the save changes button to update basic information.
Sender: User

Search Friends Request
Pre-Request: The user must be on the Friends Page
Path: The User clicks the search bar to input the query. Then the user clicks the filter buttons. The user clicks the By friends button. The result will be friends that are most related to the query. The user selects the By region. the results will be players whose region is the query. The user selects the By Games filter. the results will be players who play the query game.

Sender: User

Friends suggestion service
Pre-Request: The user must be on the Friends page
Path: The system will display possible friends based on the regions and played games when the user does not input anything and select any search options.
Sender: System

Check/ Add / Remove Friends request
Pre-request: The user must be on the Friends page; The list of friends/players is not empty.
Path: The user clicks one of the players from the suggestion list, the friend list, or the result list. Detailed information about the selected player will be displayed on the right-side panel. If the player is the user's friend, the button will be the remove button, and otherwise, add button. The user can click the button to add the player or click the button to remove the player.
Sender: User.

Search Games Request
Pre-request: The user must be on the Store page.
Path: The user clicks on the search bar and then input games' names, publishers and so on. Then the user clicks on the magnifier icon at the end of the search bar. The System will return the search results for the users displayed below. The user can click the down arrow to get the drop-down lists of Genre, Region, Year and Sorted by to get more precise search results. The system also displays the suggested games to the user on the right of the page.
Sender: User.

Dark/Light Mode Exchange Request
Pre-request: The user must be on the Store page.
Path: When the interface is in the light mode, the user can click on the Open Dark Mode button to change to the Dark Mode. When the interface is in the dark mode, the user can click on the Open light mode button to change to the light mode.
Sender: User.

Install the Game Request
Pre-request: The user must be on the Store page.
Path: The user clicks on each game name to navigate to the overview page of the game, and then the user can click on the Install button to install the game.
Sender: User.

Add to Library Request
Pre-request: The user must be on the Store page.
Path: The user clicks on each game name to navigate to the overview page of the game, and then the user can click on the 'Add to Library' button to add the game into the library.
Sender: User.

Update Request
Pre-request: The user must be on the Library page.
Path: In the user's library, the user can click on the name of each game, and the overview of the game will be displayed on the right. Then the user can click on the Update button to update the game.
Sender: User.

Remove Request
Pre-request: The user must be on the Library page.
Path: In the user's library, the user can click on the name of each game, and the overview of the game will be displayed on the right. Then the user can click on the Remove button to remove the game from the library.
Sender: User.

Scanning Reviews Request
Pre-request: The user must be on the Community-reviews page.
Path: When the user is in the main page(the store page), he/she can click on the Library item on the navigation menu to navigate to the Library page. Next users click on each games' name to get the overview of the game on the right. Next, the user clicks on the Reviews item on the navigation menu of the overview to navigate to the Reviews page. Then in the Reviews Page, the user can scan the reviews posted by other users and add comments.
Sender: User.

Submitting Reviews Request
Pre-request: The user must be on the Community-reviews page.
Path: When the user is in the main page(the store page), he/she can click on the Library item on the navigation menu to navigate to the Library page. Next users click on each games' name to get the overview of the game on the right. Next, the user clicks on the Reviews item on the navigation menu of the overview to navigate to the Reviews page of this game. Then in the Reviews Page, the user can rate the game and submit his/her reviews about this game.
Sender: User.

Scanning Discussions Request
Pre-request: The user must be on the Community-discussions page.
Path: When the user is in the main page(the store page), he/she can click on the Library item on the navigation menu to navigate to the Library page. Next users click on each games' name to get the overview of the game on the right. Next, the user clicks on the Discussions item on the navigation menu of the overview of the game to navigate to the Discussions page of this game. Then in the Discussions Page, the user can scan the discussions posted by other users and add comments.
Sender: User.

Submitting Discussions Request
Pre-request: The user must be on the Community-discussions page.
Path: When the user is in the main page(the store page), he/she can click on the Library item on the navigation menu to navigate to the Library page. Next users click on each games' name to get the overview of the game on the right. Next, the user clicks on the Discussions item on the navigation menu of the overview of the game to navigate to the Discussions page of this game. Then in the Discussions Page, the user can submit his/her discussions about this game.
Sender: User.

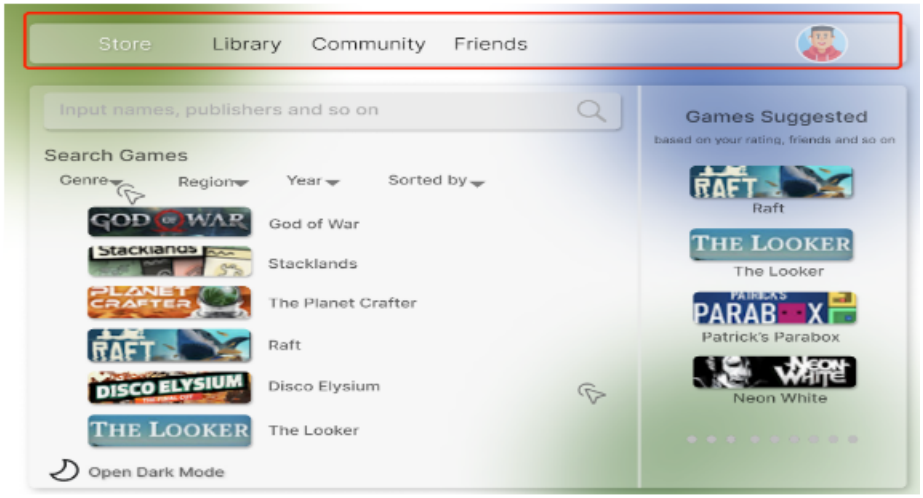
Scanning Guides Request
Pre-request: The user must be on the Community-guides page.
Path: When the user is in the main page(the store page), he/she can click on the Library item on the navigation menu to navigate to the Library page. Next users click on each games' name to get the overview of the game on the right. Next, the user clicks on the Guides item on the navigation menu of the overview of the game to navigate to the Guides page of this game. Then in the Guides Page, the user can scan the basic house rules created by the publisher including the FAQs and errata information. In addition, the user can also read the guides posted by other users.
Sender: User.

Submitting Guides Request
Pre-request: The user must be on the Community-guides page.
Path: When the user is in the main page(the store page), he/she can click on the Library item on the navigation menu to navigate to the Library page. Next users click on each games' name to get the overview of the game on the right. Next, the user clicks on the Guides item on the navigation menu of the overview of the game to navigate to the Guides page of this game. Then in the Guides Page, the user can submit his/her guides about this game.
Sender: User.

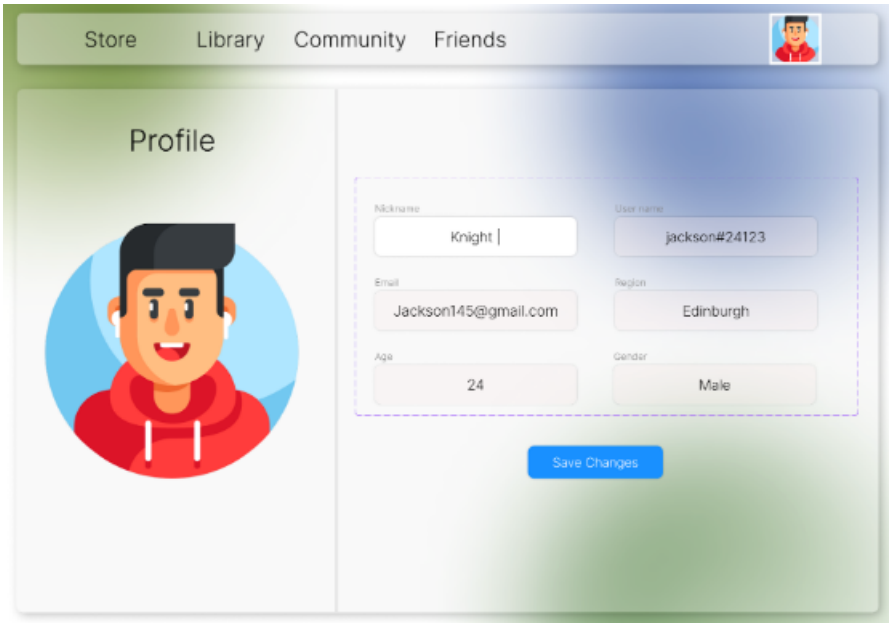
Search Community Request
Pre-request: The user must be on the Community page.
Path: The user clicks on the search bar and then inputs the name of the game community. Then the user clicks on the magnifier icon at the end of the search bar. The System will return the search results for the users displayed below. The system also displays the suggested communities to the user on the right of the page.
Sender: User.

Enter Community Request
Pre-request: The user must be on the Community page.
Path: After searching, search results of specific communities will be displayed below and the suggested communities will also be displayed on the right of the page. The user can click on the Enter the Community item of each game to navigate to the game's community, where they can scan or post reviews, discussion and guides of the game.
Sender: User.

4.2 Explanation Of Design

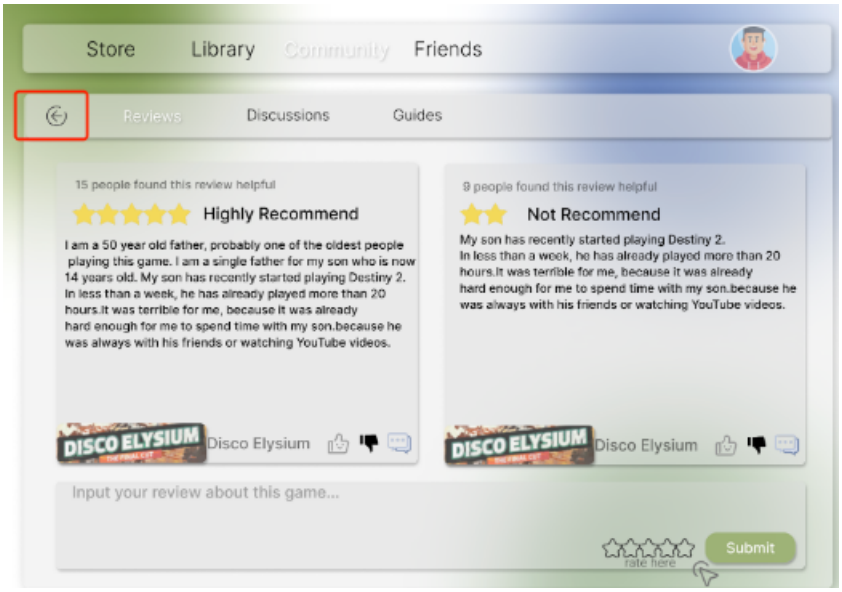


1. The Heuristics satisfied: Visibility of System Status
Reason: The navigation menu above will show users which page they are in. The user can understand clearly the main content of this page. The demonstration image is shown below and the navigation menu is in the red frame.

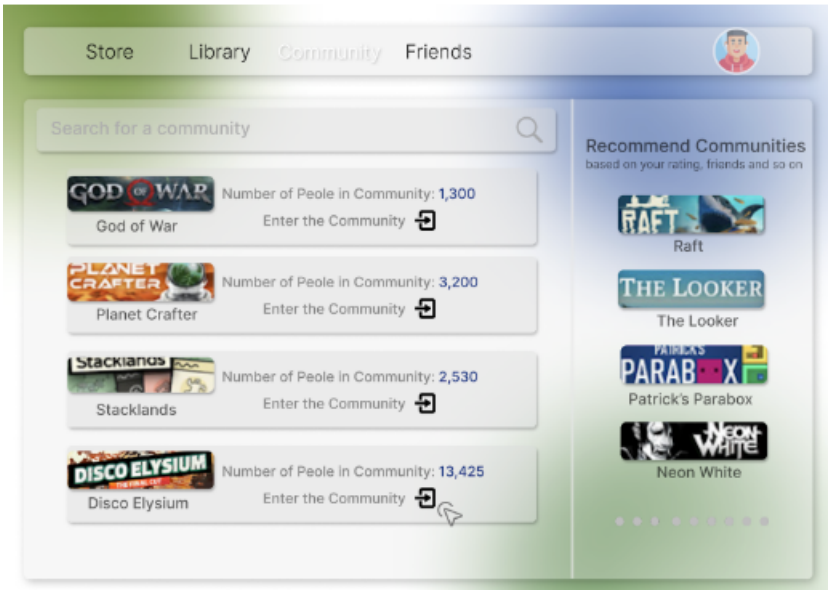


2. The Heuristics satisfied: Match between the system and the real world

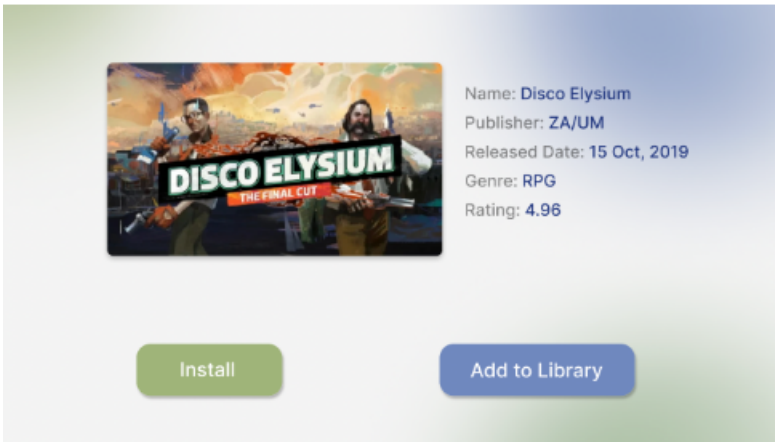
Reason: The Update Profile looks similar to the business card we have seen in our real world with a face image on the left and the detailed personal information on the right. When the user navigates to the Profile page, he/she will get an intuitive idea of this page. The Profile page is shown below.



3. The Heuristics satisfied: User Control and Freedom
Reason: When navigating to the Community Hub, which is a sub page of the Community page, the user can find the return button easily and return to the community page. The demonstration image is shown below and the return button is in the red frame.

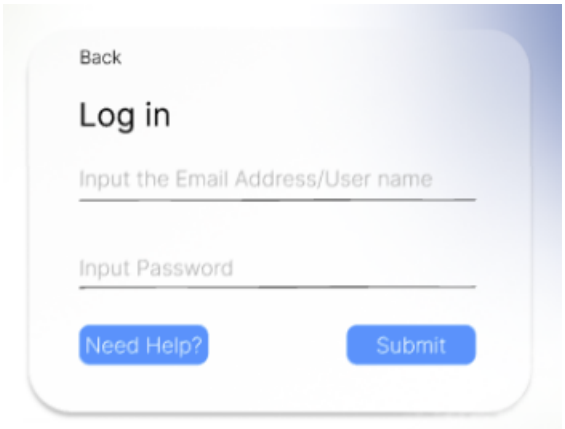


4. The Heuristics satisfied: Consistency and Standards
Reason: In the Community page, the communities and their corresponding information, including the name and the number of people in the community are displayed in boxes in the same style. It's easy for users to figure out they stand for the same type of thing. In addition, the door icon can recall users of the real door, indicating that when the user clicks on the door icon, he/she will enter the community. The image of the Community page is shown below.

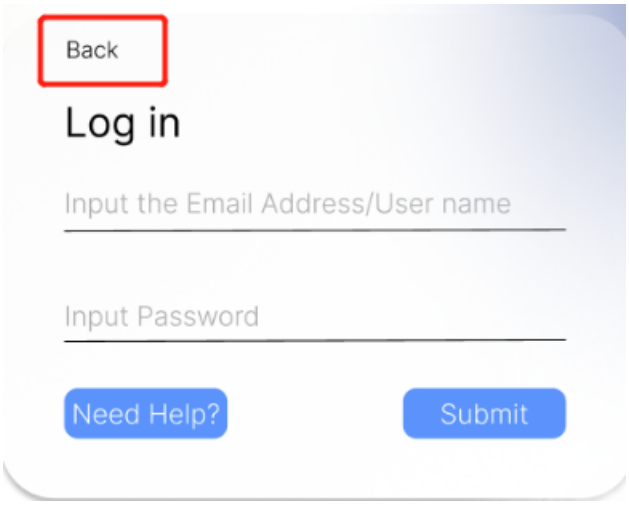


5. The Heuristics satisfied: Aesthetics and Minimalist Design
Reason: Take the game's overview page as an example and the elements designed on this page are clear and distinguishable. The title of this game is displayed in the form of a picture and occupies a large space, which helps users intuitively understand which game this is an overview of. The detailed information about the game, such as publisher and release date, is marked in blue for easy reference by users. The Install button and Add to Library buttons are displayed in different colors to facilitate users to distinguish them.

6. This design satisfies the heuristic principle: Help and documentation.
Reason: Four filters below the search bar are used to help search. We add a tag to tell the user that these four filters are used to help search. This design will help the user quickly use the friend network subsystem.



7. This design satisfies the heuristic principle: Aesthetic and minimalist design
The log-in box only contains five components. The back link, Need Help button, the submit button, the account input field, and the password input field. It keeps the content and visual design of UI focused on the essentials. The user won't be distracted by anything else.



8. This design satisfies the heuristic principle: Help users recognize, diagnose, and recover from errors
The user must have an account to support them to use our game collection system. This backlink helps the user to back to the home page. So, the user can click the register option on the home page. It gives a chance to the user to register an account before logging into the system.

4.3 Accessibility

Eyes Healthy: For users with low vision, our interface is designed in high-contrast colors, which means that the colors used on your page have a high contrast ratio. In this way, users with low vision can clearly and easily read the information in our interface. What's more, the interface also works for people who are sensitive to intensive light at night. The user can turn the Light Mode to Dark Mode at night to protect their eyes. Besides, Steam has Dark Mode, which is not helpful for users who have eye problems. In this case, our design overweight the Steam

Easy Understanding: The text on the interface will be given a head title to make users understand clearly and the head titles are very clear and catchy. Using the Cathay items on the navigation menu, users can easily navigate to other pages.

Robustness: The interface is responsive to various kinds of devices. Users can visit the interface with their smartphones and laptops of various kinds. It's helpful for people who don't bring their laptops with them because laptops are heavier.