UNSW SYDNEY

COMP9900 Project Report

Project: E-Commerce Recommender System

Group Name: COMP9900-W18B-Whatever

Submission Date: 18 Nov 2021

Miao Ren	z5196879@ad.unsw.edu.au	z5196879	Scrum Master
Jiaqi Chen	z5241312@ad.unsw.edu.au	z5241312	Front-end Developer
Shifan Wang	z5243574@ad.unsw.edu.au	z5243574	Front-end Developer
Jiahao Ding	z5296418@ad.unsw.edu.au	z5296418	Back-end Developer
Hui Gu	z5201796@ad.unsw.edu.au	z5201796	Back-end Developer

1. Overview

1.1 System architecture

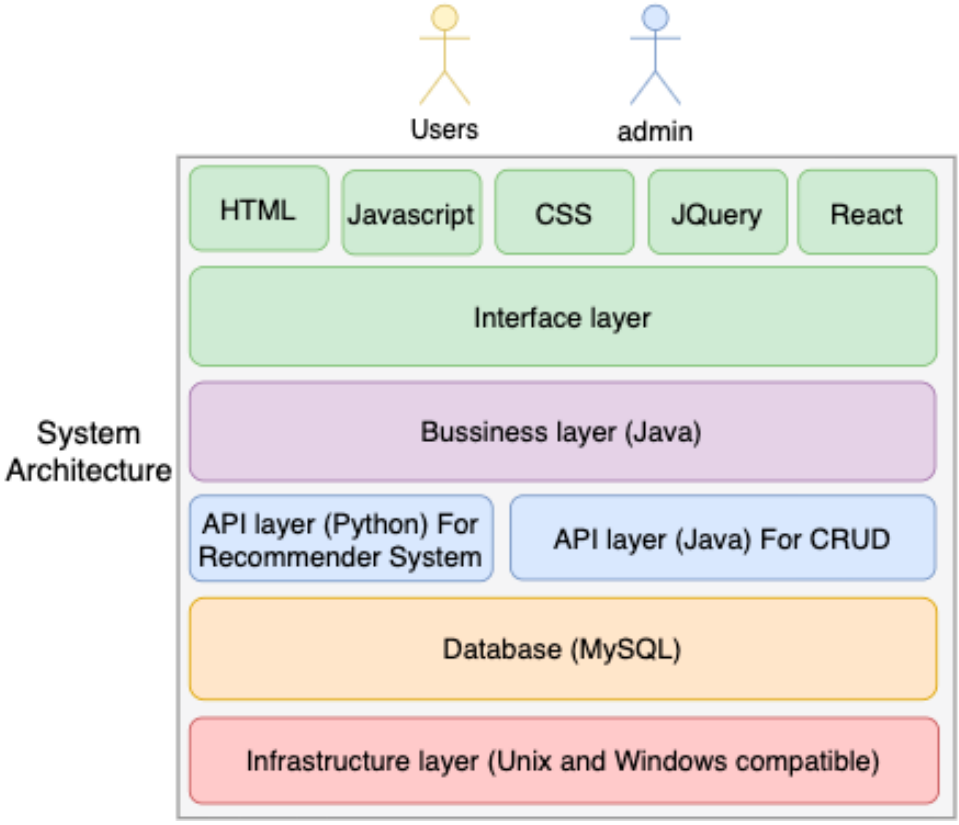


Figure 1.1 System Architecture

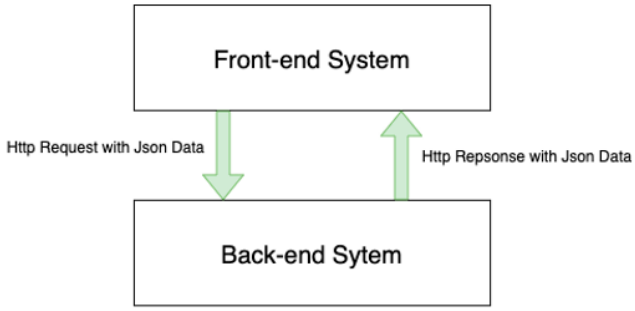
Our system architecture has a clear description, showing the presentation layer, business layer, and data layer in the system. Each layer contains a clear description of external participants and the way they interact with the system, as well as the planned technology and a clear description of language. The System architecture will be divided into two parts: front-end and back-end.

1.2 Front end

The **Front-end** will present a graphical interface for users and admins to manipulate the commodities and orders. The technologies that will be used in Front-end are HTML, CSS, JavaScript, React. The framework of the front-end system is **React**. Compared with using only HTML, CSS and JavaScript, it can improve our development efficiency.

1.3 Back end

The **Back end** is invisible to users and admins, but it is important to support the operation requested from the font-end. The communication between the front-end and back-end depends on the json data. The technologies used in the back end include Java, Spring Boot, MySQL, MyBatis, RabbitMQ, and Spring Security.



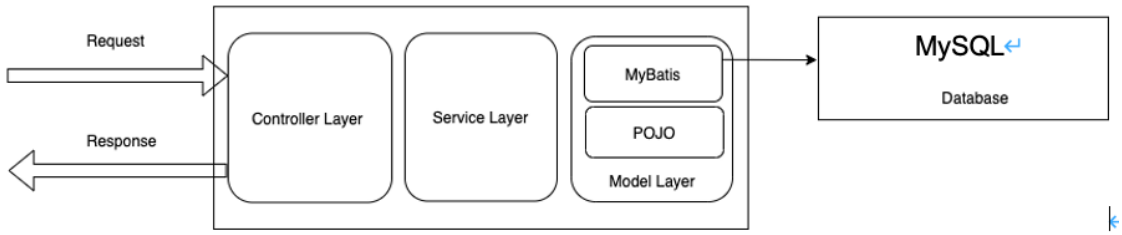


Figure 1.2 Back end

1.3.1 Basic Framework

Java is the language we will use to develop the system because we will use Framework **Spring Boot** to handle the entire backend. The advantage of using Spring Boot is that it can easily communicate with Spring Boot and other frameworks such as the tomcat server, which can improve development efficiency.

1.3.2 Back-end Architecture

We will use the MVC three-tier architecture, namely the model layer, the view layer and the controller layer.

Model layer is used to handle data persistence and processing requests. It can be divided into two main parts, the service layer deals with specific logical business, and the data access object is the entity in the database corresponding to the class object

View layer defines the presentation data. But in our system, a separate front end will be used instead of it.

Controller layer is responsible for processing the HTTP request received by the front end and sending the json data searched from the database back to the front end.

1.3.3 Authentication

There are two roles in our system, administrator and user. Both of them need to be authenticated when logging in to the web page. We will use a security framework called Spring Security to complete the authentication.

1.3.4 Data Access

In our system, we need to create, get, update and delete specific data in the database. We will use the ORM framework MyBatis to simplify data access and reduce development cycles. Moreover, RabbitMQ will use the asynchronous characteristics of RabbitMQ to improve response time and efficiency.

1.3.5 Recommender System

In our system, the recommendation algorithm is based on tags and collaborative filtering. In the beginning, the system will ask the user to select a label, and if a purchase occurs, the recommendation list will be updated. The recommendation of the game will be obtained according to our recommendation algorithm and sorted according to specific criteria. These functions will be completed by a Python script, and the controller layer will call the API from the script.

1.4 Design of functionalities

1.4.1 Mode

An intuitive and responsive Web interface is reflected in the following aspects:
The user registration process is relatively simple, users can register and log in without any hits.
The clear navigation bar and reasonable button hierarchy prompt users to intuitively distinguish the functions of the buttons.

There are two modes on the website including:
Tourist mode: Tourists can only browse the explore page and cannot do other operations.
User mode: Users who log in can enjoy all functionalities of this website.

1.4.2 Account management

Users can create their account.
Users can log in to the account.
Game producers can log in to the account.

1.4.3 Game management

Game producers can add a game.
Game producers can also view and modify the details of added games.

1.4.4 Game retrieval

Users can enter the game name in the search box to retrieve the game.
Display the detail page of this game.
Change the sorting method of the games in the dashboard.

1.4.5 Game purchase

Users can add games to their shopping cart.
Users can remove games from their shopping cart.
Users can purchase games in their shopping cart.
Users can view order history after checking out.

1.4.6 Game recommendation

The dashboard will make a personalized recommendation for Users.

Users can choose their preferred tags after registering accounts.

1.4.7 Review

Users can view other people's comments.

Users can comment on games they have purchased.

1.4.8 Donation

Game producers can make crowdfunding for their unfinished games.

1.5 Project objectives

PO-1: Users can create their account, and users and game producers can log in to the account by entering the correct username and password.

PO-2: Game producers can add a game with its name, thumbnail, description, tags, and so on. They can also view and modify the details of added games.

PO-3: Users can add games to their shopping cart, and the shopping cart will calculate the total price automatically. After checking out, users could view order history.

PO-4: Users can view other users' comments, and they can also comment on games they have purchased.

PO-5: Users can enter the game name in the search box to retrieve the game. After clicking the icon of a specific game, the detail page of this game will be displayed.

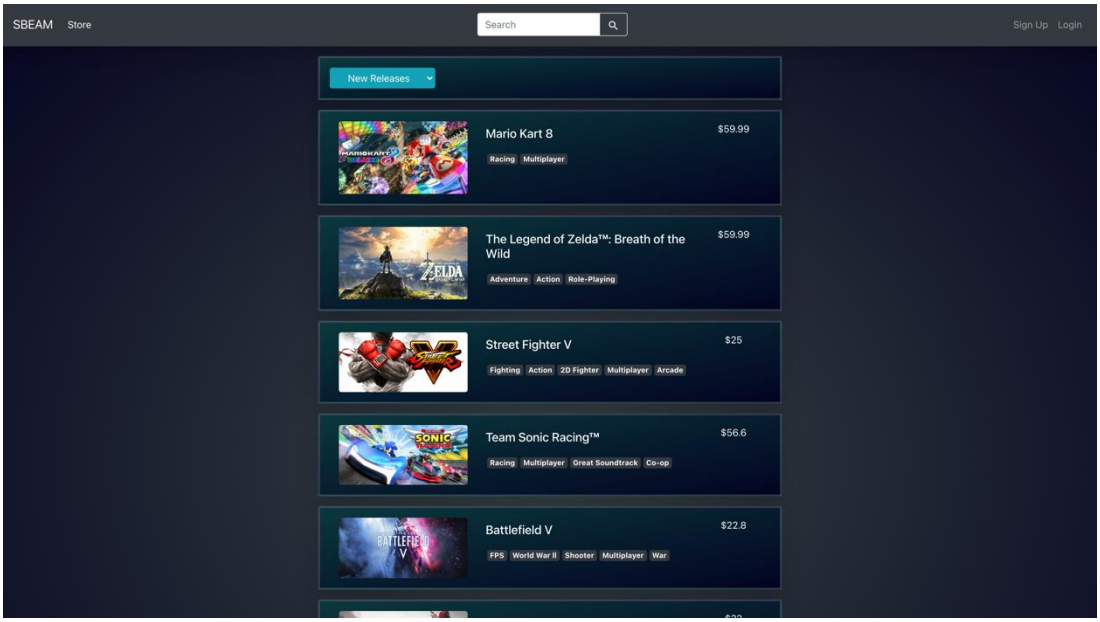
PO-6: The dashboard will make a personalized recommendation for users who are already logged in, based on multiple recommendation algorithms. And can also change the sorting method of the games in the dashboard.

PO-7: Users can choose their preferred tags after registering accounts. This allows newly registered users to get personalized recommendations and find their potential favorite games. General game stores only provide recommendations based on trending for newly registered users.

PO-8: Game producers can make crowdfunding for their unfinished games. Users can support the game producers by making donations. Most game stores only provide pre-orders and do not accept donations. This functionality can help independent producers to make games.

2. Description of functionality

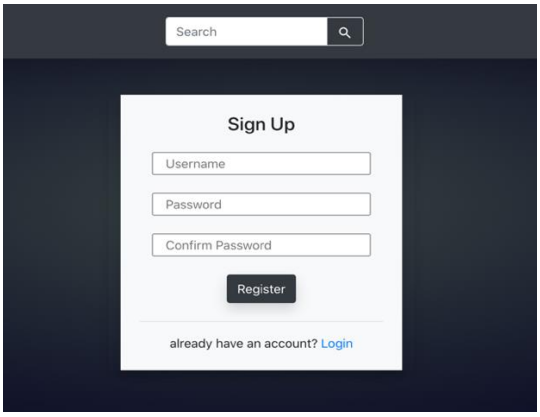
2.1 Tourist mode



Tourists can only browse the explore page and cannot do other operations.

2.2 Users

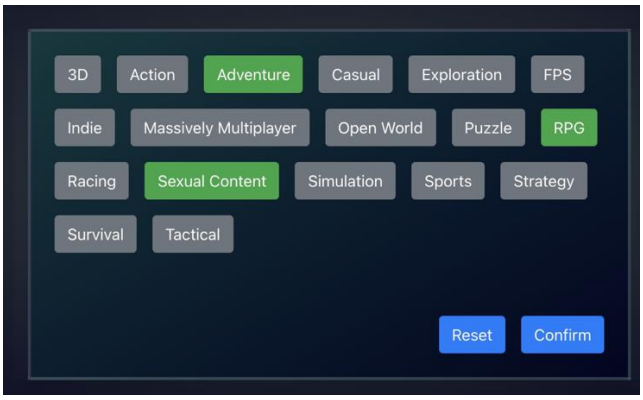
2.2.1 Sign-Up Page



The sign-up page is designed for users to register, once collected their Username, Password and Confirmed password, it will automatically jump to the tags-Selection Page. If

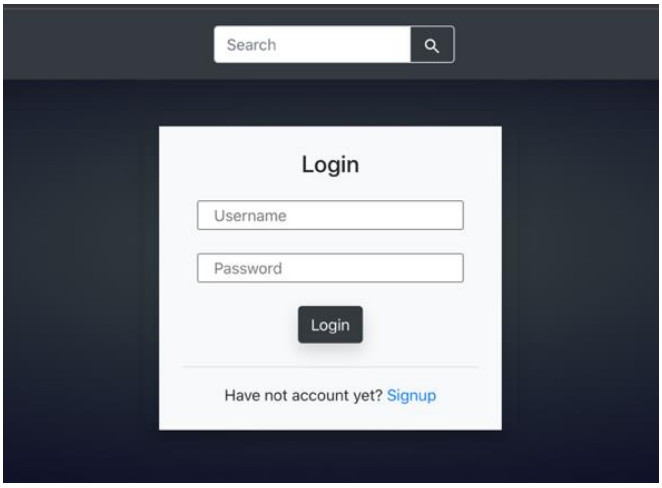
the password and confirm password are different, there will show an alert reminding users their password does not match. When registered user reregisters their account, an alert will pop up to remind the user that this account has been registered.

2.2.2 Tags Selection Page



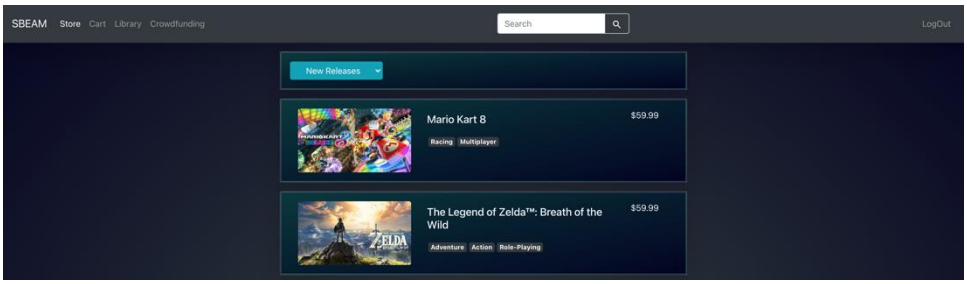
New registered users are asked to select tag they like so that the system can make recommendation list for them.

2.2.3 Login Page



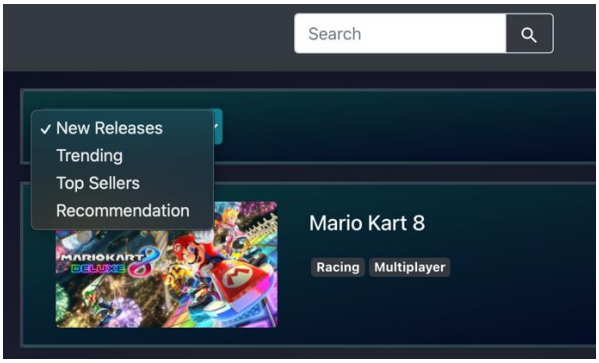
By simply clicking the ‘Login’ button, this page will slide out for gamers to log in and access their account by entering their Username and Password.

2.2.4 Explore Page



Click the picture can enter the detail page.

Customers can click the buttons from the navigation bar to jump to the store, cart, library, crowdfunding page respectively.

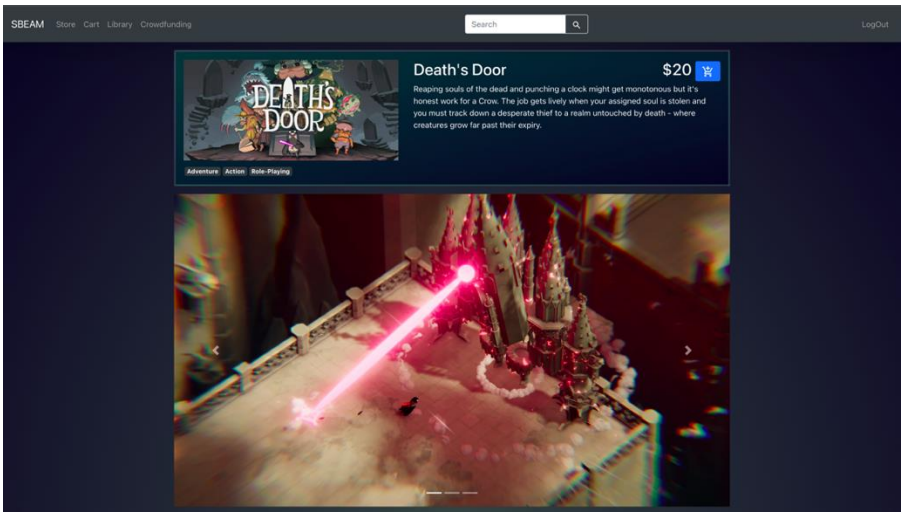


Users can change the game sort order.

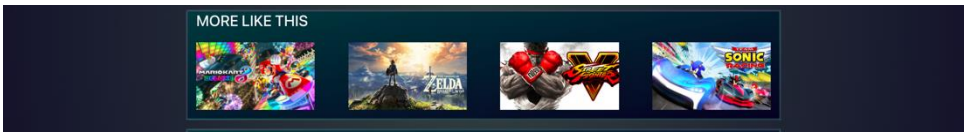
Users can enter the game name in the search bar to search for games.

Users can jump to the page to view more games.

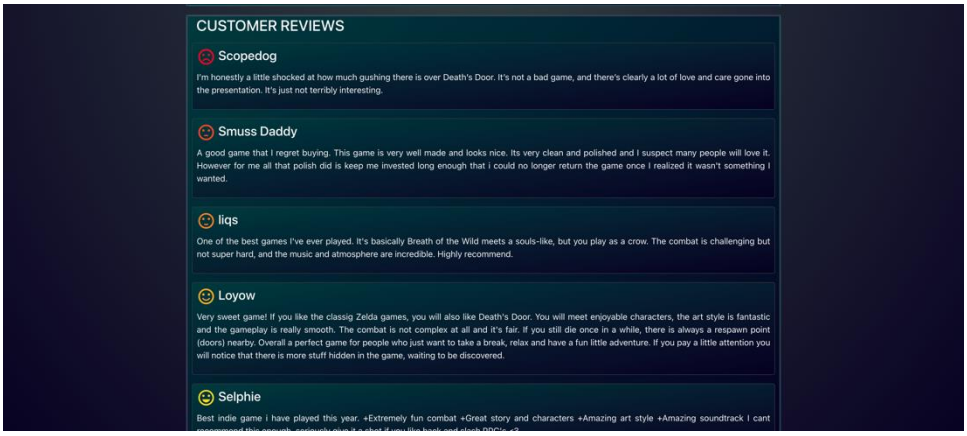
2.2.5 Game Detail Page



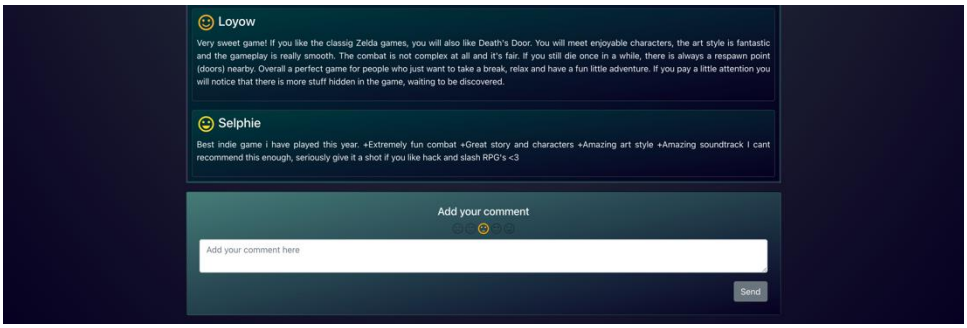
This is a page that provides more information for a game (Detailed description and more pictures). By clicking the button, users can add the game to the shopping cart.



According to the tag of this game, the system can recommend users several similar games.

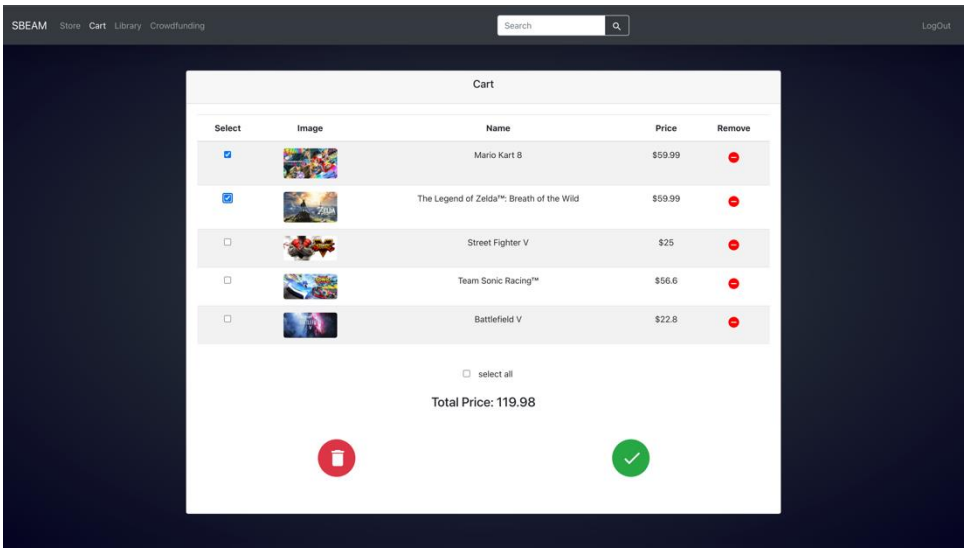


Users Can view other users' comments. The different expression shows the different attitude towards this game.



Users who have already purchased this game can share comments and rate this game.

2.2.6 Cart Page

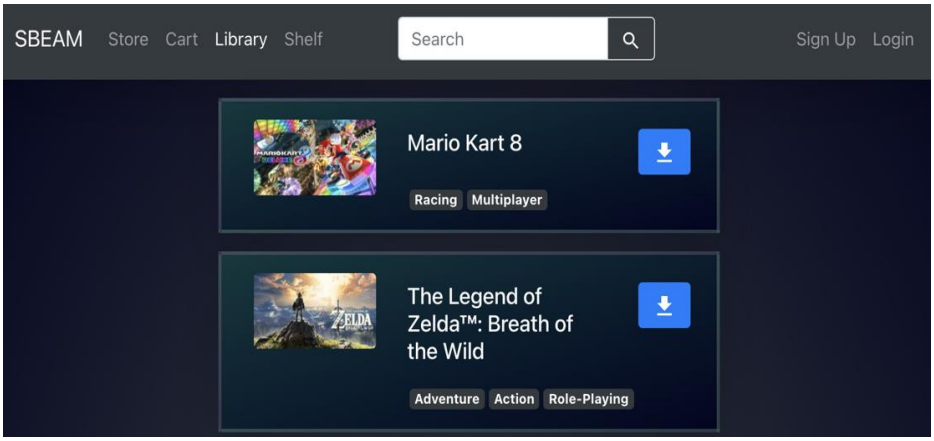


Click the remove button to remove the corresponding game from the shopping cart

Selected games could be removed from the cart by clicking the red trash button.

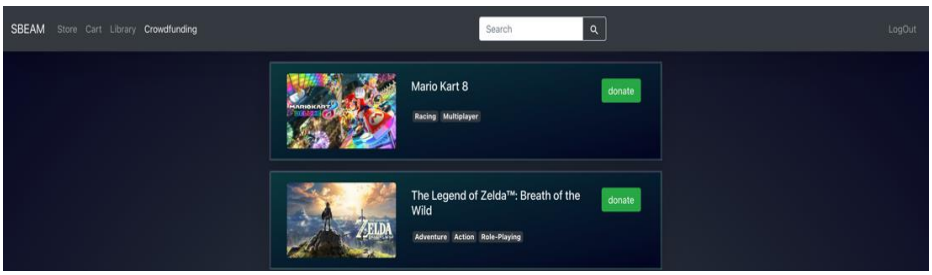
If customers want to buy the games they selected, they can click the green button.

2.2.7 Library Page

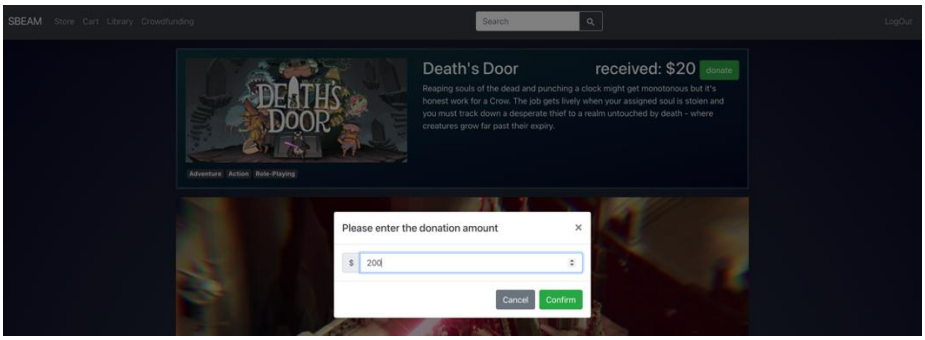


This page displays all purchased games for an account, click the download button to download the game to the local.

2.2.8 Crowdfunding Page



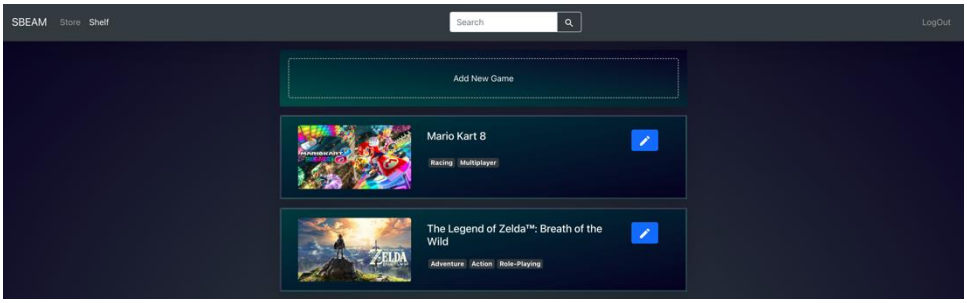
This page is designed to raise funds for games. If game producers have some financial problem when developing their games. They can use the crowdfunding function to release some game details to customers in advance. Click the game image or donate button to enter the details page.



Click the donate button, enter the donation amount in the pop-up window, then customers could input the number about how much they want to support. Finally, click confirm to donate to the game they like.

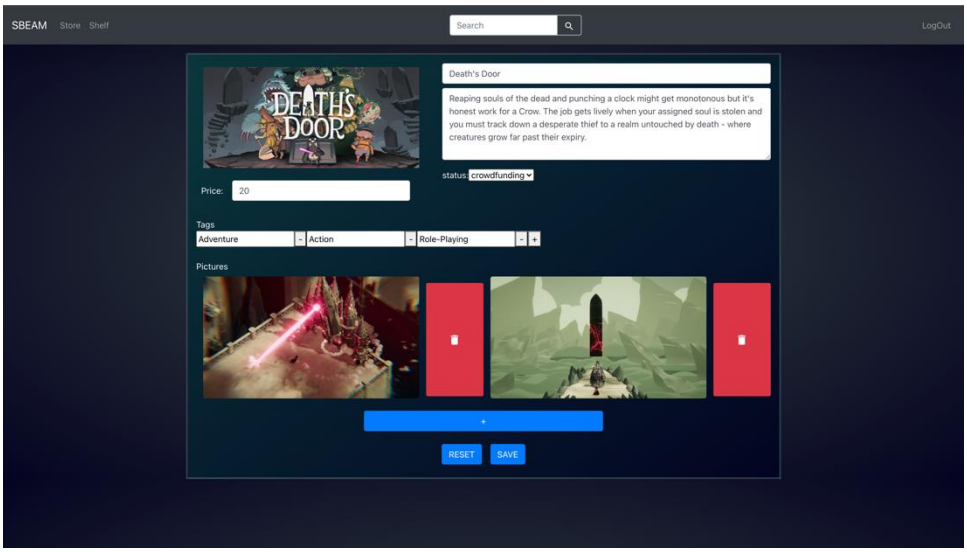
2.3 Admin

2.3.1 Game Shelf Page



This page is designed for admins. It contains all games made by them. By clicking the ‘add new game’ button, admins can create new games. Clicking the edit button can take the admin to the edit page.

2.3.2 Game Edit Page



Various details of the game can be modified.

Including the game name, game description, game amount, game tag, modify the cover, add, delete, modify the picture,

The game can be set to crowdfunding, allowing users to donate to unfinished games.

2.4 Functionality mapping to project objectives

	2.2.1	2.2.2	2.2.3	2.2.4	2.2.5	2.2.6	2.2.7	2.2.8	3.3.1	3.3.2
PO-1	■		■							
PO-2									■	■
PO-3						■	■			
PO-4					■					
PO-5				■	■					
PO-6					■					
PO-7		■			■					
PO-8								■		■

3. Third party functionalities

3.1 Front end

(1) react: front-end framework, making front-end page writing more convenient

(2) bootstrap: a library used to import common styles and well-designed components

(3) material-ui: a library used to import common icons

3.2 Back end

The libraries used in the e-commerce recommendation system are shown below.

(1) commons-io: this commons-io library is used to deal with the IO when a new game is created, a default picture of it will be read from the resource folder by commons-io and will be inserted into the database.

(2) spring-boot-starter-web: this spring-boot-starter-web library includes all the related libraries required to build the web application.

(3) spring-boot-starter-test: this spring-boot-starter-test library includes all the related libraries required to test the web application when the system is being developed.

(4) spring-boot-starter-security: this spring-boot-starter-security library is used to deal with the authentication for a token for the system.

(5) fastjson: this fastjson library is used to handle the json data that do does not correspond to the java entity received from the request.

(6) mybatis-spring-boot-starter: this mybatis-spring-boot-starter library is the library that enables us to use mybatis in the Spring Boot project. We can set the configuration of the datasource required by the mybatis in the application.xml file.

(7) mysql-connector-java: this mysql-connector-java library includes the driver for connecting the MySQL server to java.

(8) jjwt: this jjwt library is used to generate the token and valid the token received combined request.

(9) surprise: python library to implement collaborative filtering algorithms for recommendation system

(10) beautifulsoup: a python crawler library to get real data from steam

(11) pandas: a python library to read and write csv file, and make data processing more convenient.

4. Implementation Challenges

4.1 Front end

(1) If we only use vanilla js, the layout and components are time-consuming, and the front-end page styles written by us are messy, so we use several libraries including bootstrap, which contain many commonly used presets.

(2) When writing the front-end page, the browser on the PC side displays normally. But when visiting our website on a mobile phone, the layout could be messy. Therefore, we use bootstrap to make our website responsive. When the aspect ratio of the webpage exceeds the threshold, the layout will be changed automatically, so that the webpage layout can be displayed normally on the mobile phone.

4.2 Back end

(1) CRUD

The main part of the e-commerce system back end is CRUD. For example, it includes the creation of a user, the creation, update, retrieve of game, the creation, retrieve of information on the order, cart, and comment.

(2) Updating Game Tag

The admin can edit the detail of games such as the tag or the picture of these games. When updating these data, instead of updating it directly, it will first delete the original data, and then add new data

(3) Encapsulation of result

All the results returned from the back end to the front end are encapsulated into a Java Object called ResultEntity. If the request succeeds, it will return ResultEntity.successWithData(Data) or ResultEntity.successWithoutData(). If the request fails, it. Will return ResultEntity.failed(message) with corresponding error message.

(4) Authentication

In the e-commerce recommendation system, there are three kinds of roles for client user, admin, and visitor. The Back end can tell from different roles by the token sent from the front-end. Token is a string of character strings generated by the back end for identifying the identity of the client. The Token will be generated when the user or admin login to the system for the first time. Later, the user or admin will fetch specific with the request combined with the token. These authentications are accomplished by the JWTUtil and Spring Security. The Spring Security is based on the filter. Therefore, every time the client sends some request to the back end, the filter will first check the token combined with the request, if there is no token in the request header, the back-end server will reject this request.

4.3 Recommendation Algorithm

For the recommendation system, we used two different methods. One is a content-based recommendation algorithm, other is collaborative Filtering. A content-based recommendation system is an algorithm that works on the principle of similar content. If a customer has chosen some favorite tags when registering an account, then the recommendation system will compare the similarity between tags chosen by this customer and the tags of all games in-game library and recommend this customer some games sorted by similarity. To realize this algorithm, we need to compute distances between the tags of the customer and that of games. Here, we choose to use cosine similarity since tags are textual data. If vectors of A and B are close then the cosine value of these two vectors will be small, which means these two vectors are similar and vice versa. This kind of recommendation system is very efficient for a new user because we cannot get enough information from this user to predict the preference.

Cosine Similarity:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (1)$$

When a customer purchased some games, we design a feedback function for the user to write some comments and rate the game from 1 to 5. After giving feedback, we could know whether this customer like this game or not. Therefore, it is easy to analyze which users

have the same taste in games, and we could recommend a customer some games that his similar customer likes. It is the basic theory of user-based collaborative filter. In addition, there is another collaborative filter that is item-based. It is the method that if a customer has purchased a game, then the recommendation system will recommend some games which are like this game. Collaborative filtering is a very intelligent recommendation system that works on the similarity between different users and items. It is often used in e-commerce websites and streaming services such as Netflix and Disney+.

However, the complexity of common collaborative filters is very large. If the website requests recommendation list in real time, there will be a long time for the back end to send the result. Thus, it will cause a very bad user experience. So, we decide to use the funkSVD algorithm. We first get the SVD matrix and then find a similar group of users with the help of KNN, this dramatically decreases the running time. Furthermore, we asynchronously generate a recommendation list. We will store a unique recommendation list for every user, it will be returned instantly when website requests.

Cold start is a classical problem in the collaborative filter recommendation system. Specifically, if customers do not purchase any games, the persona will not be generated, so we could not recommend this customer any game. That is why we use a content-based recommendation algorithm for new users.
After getting the list of possible favorite games, it may contain games that have already been purchased. Therefore, after getting the gamers predicted rating of all games, the games that are already in the library must be removed and then be sorted.

5. User Document

We use ubuntu 20.04.1 on VMWare Fusion.

5.1 Installing Java

By default, Ubuntu 20.04 includes Open JDK 11.
To install this version, first update the package index:

$ sudo apt update

To install the JDK, execute the following command

$ sudo apt install default-jdk

Check the version of java

$ java -version

5.2 Installing MySQL

To install the MySQL, execute the following command

$ sudo apt install mysql-server

Invoke MySQL with sudo

$ sudo mysql

Change the password for root to root

mysql >ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root';

Then invoke the mysql with user ‘root’ and password ‘root’

$ mysql -u root -p

5.3 Installing node.js

Run the following command to build yarn and nodejs environment.

$ sudo apt install curl

$ sudo apt remove cmdtest

$ sudo apt remove yarn

$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -

$ echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee

/etc/apt/sources.list.d/yarn.list

$ sudo apt-get update

$ sudo apt-get install yarn

$ sudo apt-get install -y nodejs

5.4 Installing Python3 and related package

Ubuntu 20.04 ships with Python 3 pre-installed. Upgrade the packages installed to make ensure the latest version is installed.

$ sudo apt -y upgrade

check the version of Python 3 that is installed in the system

$ python3 -V

install pip to manage software packages for Python

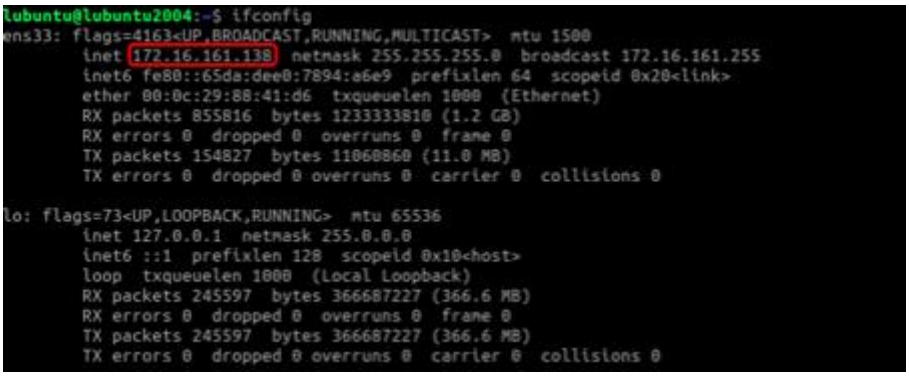
$ sudo apt install -y python3-pip

Install packages PyMySQL, pandas and surprise. This will take several minutes

$ pip install PyMySQL pandas surprise

5.5 Transferring file

Check the IP address for the server in the VMware Fusion.



Then transfer the compressed file **project.zip** to the server with scp

$ scp project.zip lubuntu@172.16.161.138:

```
johnny@192 ~/unsw/2021T3/COMP9900/final scp project.zip lubuntu@172.16.161.138:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                    LINUXVMIMAGES.COM
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                User Name: lubuntu
                Password:  lubuntu (sudo su -)
lubuntu@172.16.161.138's password:
project.zip                                   100%  196MB  35.1MB/s   00:05
```

Then unzip the project.zip and change the directory

$ unzip project.zip

$ cd project

First enter the MySQL console with user “root” and password “root”

$ mysql -u root – p

Then creating the schema in the MySQL, and exit

mysql> source ~/project/data/schema.sql

mysql> quit

```
mysql> source ~/project/data/schema.sql
```

```
mysql> quit
Bye
```

Change the directory to data Insert the data with python script

$ cd data

$ python3 insertdata.py

```
lubuntu@lubuntu2004:~/project$ cd data
lubuntu@lubuntu2004:~/project/data$ python3 insertdata.py
```

Insert recommend data with python script

$ python3 cf_udpate.py

```
lubuntu@lubuntu2004:~/project/data$ python3 cf_update.py
```

5.6 Starting up front-end and back-end

Change the directory with

$ cd ~/project/front-end

Run the shell script to start up the front-end

$ sh front-end-setup.sh

```
lubuntu@lubuntu2004:~/project/front-end$ sh front-end-setup.sh
```

Change the directory with

$ cd ~/project

Run the jar to start up the back end

$ java -jar project-1.0-SNAPSHOT.jar

```
lubuntu@lubuntu2004:~/project$ java -jar project-1.0-SNAPSHOT.jar
```

5.7 Visit localhost:3000 in the browser

6. Reference

GamesIndustry.biz. 2021. *Gaming will hit $91.5 billion this year - Newzoo*. [online] Available at: <https://www.gamesindustry.biz/articles/2015-04-22-gaming-will-hit-usd91-5-billion-this-year-newzoo> [Accessed 3 October 2021].

Kumar, S., De, K. and Roy, P., 2020. Movie Recommendation System Using Sentiment Analysis From Microblogging Data. *IEEE Transactions on Computational Social Systems*, 7(4), pp.915-923.

Ampatzoglou, A. and Stamelos, I., 2010. Software engineering research for computer games: A systematic review. *Information and Software Technology*, 52(9), pp.888-901.

Docs.python.org. 2021. *sqlite3 — DB-API 2.0 interface for SQLite databases — Python 3.9.5 documentation*. [online] Available at: <https://docs.python.org/3/library/sqlite3.html> [Accessed 20 June 2021].