

E-sports Project Report

CSCI-665-M-2021FA-S-0737XL Software Engineering

Bingxiang Lin (Leader), Jiashu Chen, Hongyi Zhang

Supervised by Prof. Akhtar

New York Institute of Technology

December 13, 2021

Abstract

With the invention of E-sports, there is a growing need for games in people's spare time. Providing users with a platform for game information is a much-needed project. This project is used to collect information about users' game interests, and realize functions such as user search, comment, and recommend games for users. The administrator can add, delete, update and other functions to the game. In this project, we used CSS, HTML, etc. to construct the front end page, wrote the back end with Python, and used MySQL as the database. This project enables users to find the game information they want more quickly. We will continue to improve this project and find out the existing loopholes in our future work.

Contents

List of Figures

List of Tables

1	Introduction	1
1.1	Existing Systems	1
1.2	Proposed System	1
1.3	Advantage of the proposed system	1
1.4	Software Engineering Model	1
1.5	Purpose	1
1.6	Project Objective / Goals and Scope	2
1.7	Technologies & Tools	2
1.8	Users	2
1.9	Privilege or Access Level	2
1.10	Motivation	2
1.11	Timeline	2
2	Analysis of the System	2
2.1	Activity List	2
2.2	Functional and Non-functional Requirements	3
2.3	Context Diagram	3
2.4	Data Flow Diagram	3
3	Database Design	4
3.1	Table Schema	4
3.1.1	User Table	4
3.1.2	Admin Table	4
3.1.3	Game Table	5
3.1.4	Comment Table	5
3.1.5	Interest Table	5
3.2	Entity Relationship Diagram	6
4	Functionality and Implementation	6
4.1	Features	7
4.2	Security	7
4.2.1	Privilege Levels	7

4.3	File Structure	7
4.3.1	Home Page	8
4.3.2	More Games of Specific Type	9
4.3.3	Game Details	10
4.3.4	Search	11
4.3.5	Game List in the Admin Panel	12
5	Earned Value Analysis	12
6	Test	14
7	Conclusion	14
7.1	Limitations	14
7.2	Future Enhancements / Recommendations	15
7.3	Team Members	15
7.3.1	Learning Experience and Outcome	16
8	References	16

List of Figures

1	Context diagram	3
2	Data flow diagram	3
3	Entity relationship diagram	6
4	Home page view	8
5	Code snippet of home page	8
6	More games of a specific type	9
7	Code snippet of more games of a specific type	9
8	Game details	10
9	Code snippet of game details	10
10	Search	11
11	Code snippet of search	11
12	Game list in the admin panel	12
13	Code snippet of game list in the admin panel	12

List of Tables

1	User table schema	4
2	Admin table schema	4
3	Game table schema	5
4	Comment table schema	5
5	Interest table schema	5
6	Activities and cost	13
7	Project report at the end of day 45	13
8	Earned value analysis	14

1 Introduction

With the growth of the E-sports industry and the inclusion of E-sports in the Olympics, people are learning more and more about gaming. So we decided to develop a website about E-sports that would help users find games on the site, and collect users' interest.

1.1 Existing Systems

As is known to all, Steam is one of the largest game websites in the world. This website provides users with functions such as searching games and recommending games. Users can get recommended games from their friends and buy and download them. This web page is written by HTML, JS, CSS, and jQuery.

1.2 Proposed System

It is a game information platform with user and administrator pages. Users can log in to the page to find information about the game and give comments on the game, as well as find out what their friends like and recommend the game through the friends system. Administrators can log in to the administrator page to manage games, users and reviews.

1.3 Advantage of the proposed system

Users can quickly and conveniently find the game they want, and can put forward opinions and comments on the game in time. Administrators can collect information about users' interests.

1.4 Software Engineering Model

Concurrent Model—Each member is responsible for a portion of the function, all at the same time.

1.5 Purpose

Users can quickly find out what they like about the game and comment on it. Administrators can collect and manage users' game interests.

1.6 Project Objective / Goals and Scope

The project is aimed at all users who want to find and comment on games, and provides user reviews for various game companies.

1.7 Technologies & Tools

The front-end use HTML, CSS, JS. The back-end uses Python-Flask. The database uses MySQL.

1.8 Users

People who like to play games and are interested in them and administrators can use the system.

1.9 Privilege or Access Level

Users can access the home page but not the administrator page. The administrator can access the home page and administrator page.

1.10 Motivation

Gather data about users' interests and gaming conventions as well as provides them with a maintenance portal for new games and game features.

1.11 Timeline

In the first week, group members discuss project functions and assign tasks. In the second week, both the front end and the back end are written simultaneously. In the third week, finally, integrate the front end and the back end. Then test it.

2 Analysis of the System

2.1 Activity List

Users

Sigh in/Login — search — view games information — make comments — redirect to the game's website — sign up

Admin

Sign in/Login — manage users account — manage games information — manage users comments — sign up

2.2 Functional and Non-functional Requirements

Users can register and login, search for games, view game information, and add games to their like-list. Admin can add more games, view game-list, view user account, delete user account and comments.

2.3 Context Diagram

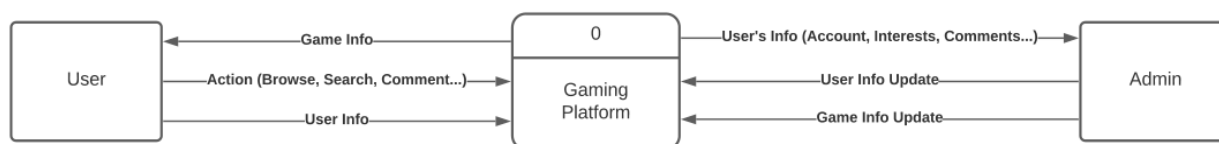


Figure 1: Context diagram

2.4 Data Flow Diagram

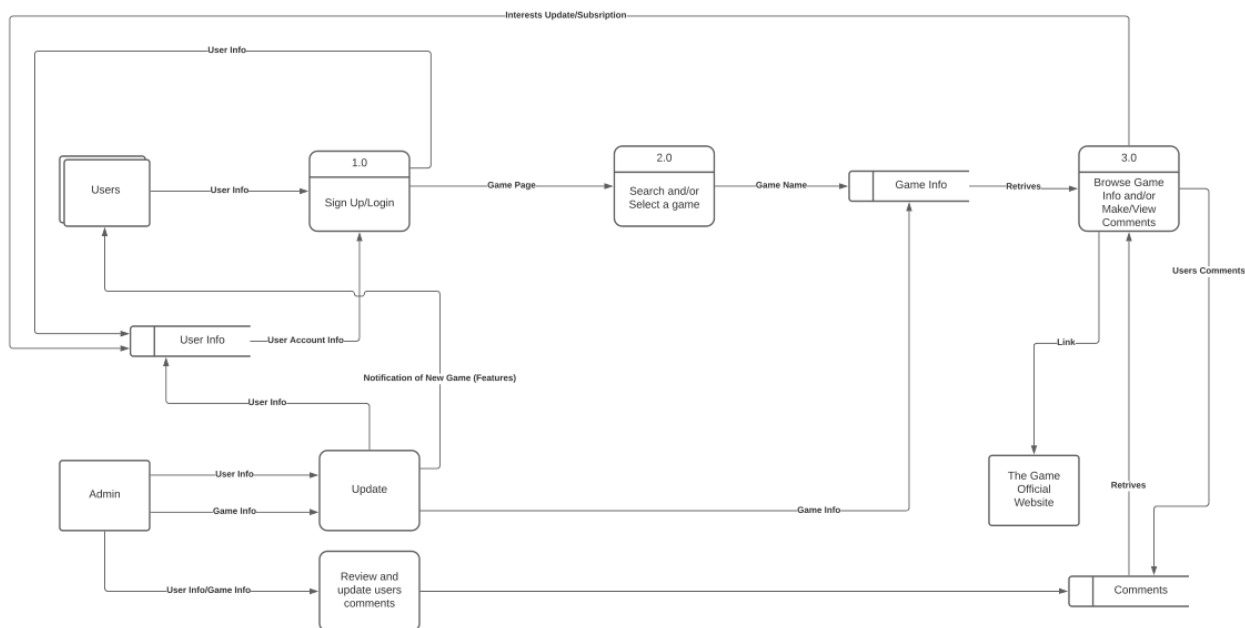


Figure 2: Data flow diagram

3 Database Design

This project was designed using MySQL database. One of the reason we used it is that it has all the features that we needed for this project. It is based on the SQL queries, and our member has the basic knowledge of SQL. So building and interacting with it would not be a big problem. It is a secure system that has a data security layer to protect credential information so that we don't need to worry much about our credential information being stolen by attackers. Also MySQL is a very fast database language, loading a page information from it will be fast too so that a user has good experience browsing our web without speeding too much time on page loading. Most importantly, our data that will be stored in the database are related to each other, which means that we need a relational database management system. MySQL fits the position. Even better, it is a free-to-use database!

3.1 Table Schema

We have five tables for this project: user table, admin table, game table, comment table, and interest table.

3.1.1 User Table

The user table is to store user's information. The table schema is shown below:

user_id PRIMARY KEY	username	email UNIQUE	password

Table 1: User table schema

In this table, user_id is the primary key. It is automatically incremented by 1 every time a user account is added. The username is used to store user's username. The email is a unique key, only one distinct email is allowed when register. Lastly, password is, of course, used to store account password.

3.1.2 Admin Table

The schema of admin table is shown below:

user_id PRIMARY KEY	adminName	email UNIQUE	password

Table 2: Admin table schema

It has exactly the same attributes as user table. It is used to store admin information.

3.1.3 Game Table

The table shown below is game table:

game_id PRIMARY KEY	name	release_date	type	description	score	weblink

Table 3: Game table schema

It stores game's information. It has following attributes: game_id is the primary key that is automatically incremental by 1 each time when a new game is added; name column stores game name; release_date column stores the date when the game is released; type column stores game type; description column stores game's description; score column stores the score that the user give it to the game; and weblink column stores game image address.

3.1.4 Comment Table

The comment table schema is shown below:

comment	game_id FOREIGN KEY	user_id FOREIGN KEY	commentTime

Table 4: Comment table schema

It has four attributes: comment, game_id, user_id, commentTime. The comment column stores the comment that a user made in a game. The game_id and user_id columns are foreign keys since a game can has multiples comments and a user can make as many comments as he wants. The commentTime stores the time when it is made.

3.1.5 Interest Table

The interest table stores user's id and his interested game's id. So it has only two attributes, user_id and interest. They are foreign keys. The table schema is shown below:

user_id	interest

Table 5: Interest table schema

3.2 Entity Relationship Diagram

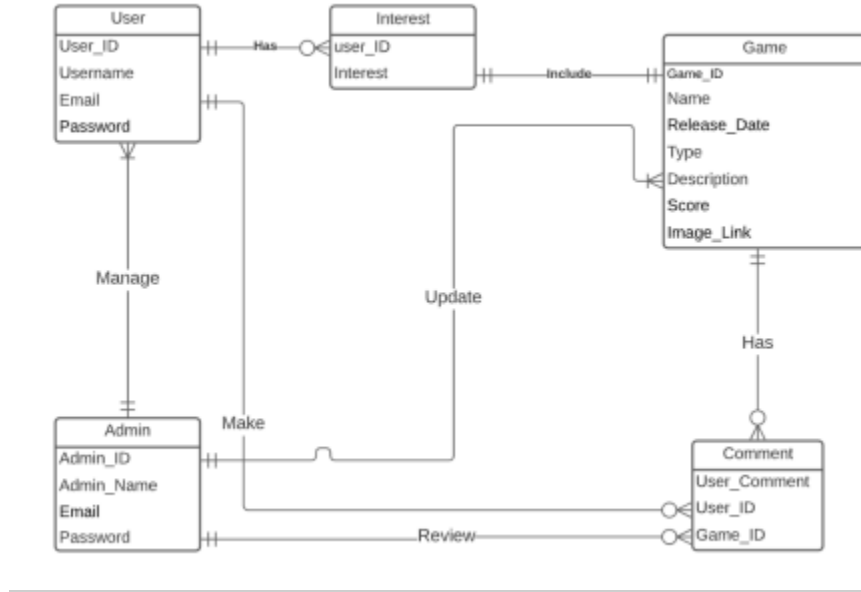


Figure 3: Entity relationship diagram

User table has one-to-many relationship to interest table and comment table since a user can have zero or multiple interests and can make zero or many comments in a game. Admin table has one to many relationship to user table, game table and comment table since an admin can maintain multiple users' accounts, games' features, and comments' contents. A game can has zero or many comments, so it has one-to-many relationship to comments table.

4 Functionality and Implementation

This project was implemented using HTML, CSS, and JavaScript for frontend development, and python for backend development. Since we were designing a web application, HTML, CSS and JavaScript are necessary. HTML and CSS are used to decorate our website, and JavaScript is used to handle some of user's input such as search request, button onclick and so on. Python has some dedicated web development frameworks such as Flask and Django. And it can easily connect to the database we selected, which is MySQL, so we selected it as our backend development. The reasons we used flask web framework instead of others are that it supports for API and secure cookies which allows us to protect users and admins information against attacker.

4.1 Features

Our system has several features for user area and admin panel. The home page of our website lists whole bunch of games with different type for users to select. If a user wants to see more of this type of game, he or she can always click more button to see more. Users can view game's information by clicking the game image. In the game information page, they can view game description and comments. They can leave comments if they have logged in. They can also add the game to their interested game list by click like button. We also provided a search function that allow users to search for their desired games. A list of relevant games will be shown to users as a search result. Users can check out the most popular games by going to ranking page. For admin panel, administrators have to login to gain access to the admin page. Once they'd logged in, they can maintain users' account and add more games. More features, we will talk about them in the conclusion section, will be added in the future.

4.2 Security

Security of users and admins information is top of our concern. To protect their account, their password will be encrypted using hash 256 algorithm that provided by flask framework. Without admin account logged in, the admin panel cannot be accessed to prevent attackers from accessing the data and make malicious change to the system. We will force users to have a strong password so that the account will be not easily stolen in the future.

4.2.1 Privilege Levels

Users can only access to the main page to view game's information. They are not allowed to change any information that are created by administrators. As an admin, he can access to admin panel to update user's account, user's comments and collect user's interests. He also can view game's info and add more games to the website. But he cannot directly interact with the database.

4.3 File Structure

The main.py file, where it is used to launch the project, is placed by itself with layout folder. Inside layout folder, there are several python files: __init__.py, auth.py, database.py and views.py. They are placed with templates folder where all html files are placed inside, and static folder where all javascript files that either in js or admin_js folder, images in images folder, css files in css folder, and font files in fonts folder are inside.

4.3.1 Home Page

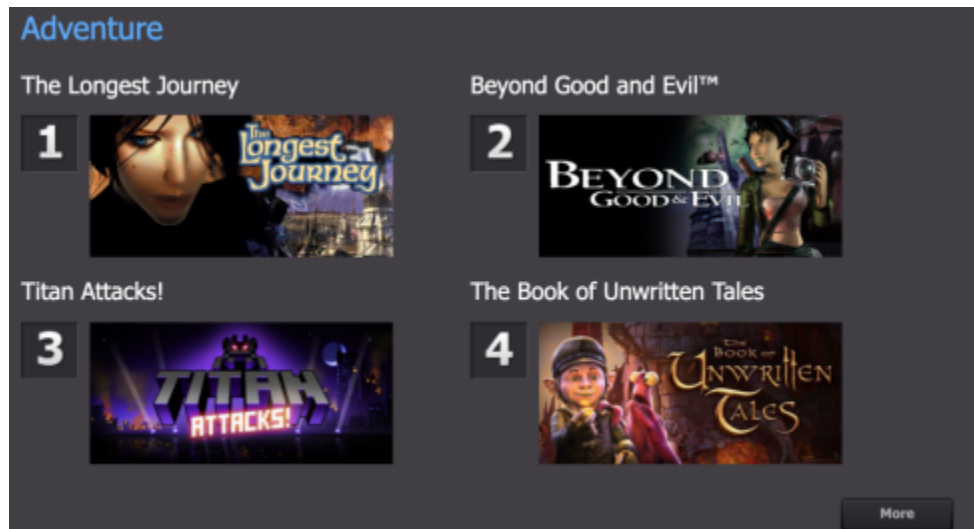


Figure 4: Home page view

```
<h2>Adventure</h2>

<div class="reasons">
  <h3>The Longest Journey</h3>
  
  <a href="{{ url_for('views.game', gameName='The Longest Journey') }}"></a>
</div>

<div class="col_w850">
  <div class="btn_more"><a href="{{ url_for('views.showMore', gameType='Adventure') }}">More</a></div>
</div>

@myViews.route('/')
def home():
    gameInfo = database.getAllGame()
    return render_template("index.html", games=gameInfo)
```

Figure 5: Code snippet of home page

When entering the website, python home function will be executed and direct user to the home page. With the code `render_template("index.html", games=gameInfo)`, index.html file will be opened with all the game information that obtained from `database.getAllGame()`. If a user wants to see detail of a specific game by clicking the game image, the a tag with the url to views.game will tell the system to execute game function in the views.py file with the parameter gameName. If a user wants to see more of a specific game type by clicking the more button, the a tag in the html code will tell the system to execute showMore function with the parameter gameType.

4.3.2 More Games of Specific Type

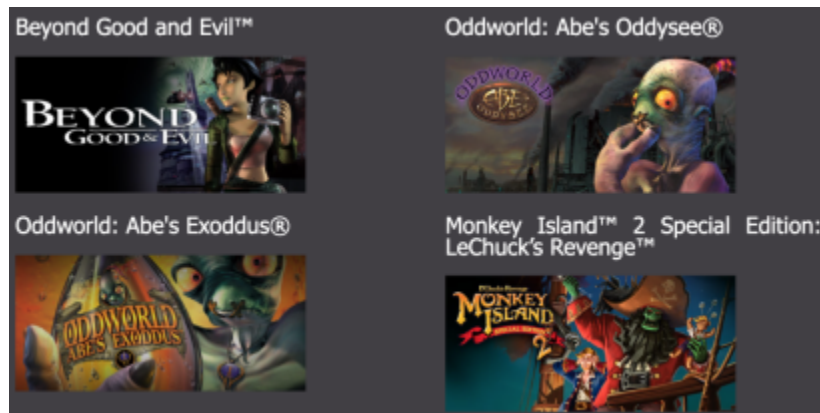


Figure 6: More games of a specific type

```
@myViews.route('/more/<gameType>')
def showMore(gameType):

    moreGame = database.getGameByType(gameType)
    return render_template("{gameType}.html", games=moreGame)

def getGameByType(gameType):
    mydb = mysql.connector.connect(
        host="127.0.0.1",
        user="root",
        password="Lbx-364881595",
        database="esportDB"
    )
    mycursor = mydb.cursor()
    mycursor.execute("SELECT * FROM gameInfo WHERE type = %s", (gameType,))
    result = mycursor.fetchall()
    mydb.close()
    return result

{% for game in games%}
<div class="reasons">
    <h3>{{game[1]}}</h3>

    <a href="{{ url_for('views.game', gameName=game[1]) }}"></a>
</div>
{% endfor %}
```

Figure 7: Code snippet of more games of a specific type

The `showMore` function will get more games from the database by the code `database.getGameByType(gameType)`. The `getGameByType` function in `database.py` will execute SQL command to get the games info. After that, users will be directed to a page, `gameType.html`, that has more of the type he is looking for by the return command. In the `gameType.html`, the for loop written in jinja2 language will iterate all the games that has the requested type and put them in a html div tag.

4.3.3 Game Details

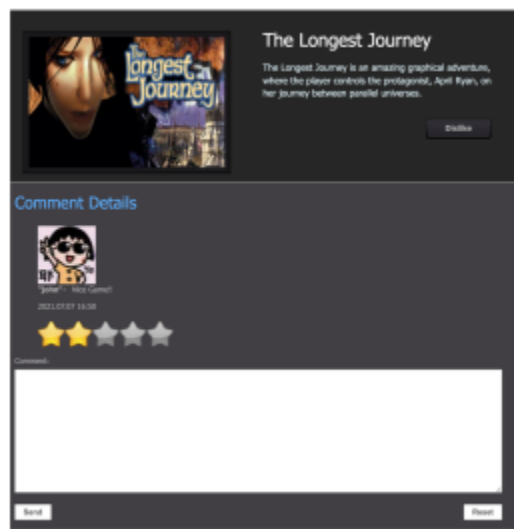


Figure 8: Game details

```
@myViews.route('/detail/<gameName>')
def game(gameName):
    gameInfo = database.getGameByName(gameName)
    comments = database.getComments(gameInfo[0][0])
    return render_template("{gameName}.html", game=gameInfo, comments=comments)

<div class="slider_content">
    <h2>{{game[0][1]}}</h2>
    <p>{{game[0][4]}}</p>
    {% if session.logged_in %}
    {% if game[0][0] is not in session.interests %}
    <div class="btn_more"><a href="/addInterest/{{ game[0][0] }}/{{ game[0][1] }}">Like</a></div>
    {% else %}
    <div class="btn_more"><a href="/removeInterest/{{ game[0][0] }}/{{ game[0][1] }}">Dislike</a></div>
    {% endif %}
    {% endif %}
</div>

<h2>Comment Details</h2>
<ul id="comment-list">
    {% for comment in comments %}
    <li>
        
        <p><strong>{{comment[1]}}: </strong> {{comment[0]}} </p>
        <p>{{comment[3]}}</p>
    </li>
    {% endfor %}
</ul>
```

Figure 9: Code snippet of game details

When a user is routed to `/detail` with a specific game name, `game(gameName)` function will be executed. It will get the details and comments of the game from the database by executing the `database.getGameByName(gameName)` and `database.getComments(gameID)` functions. Then it will go to `{gameName}.html` with the game info and game comments.

4.3.4 Search

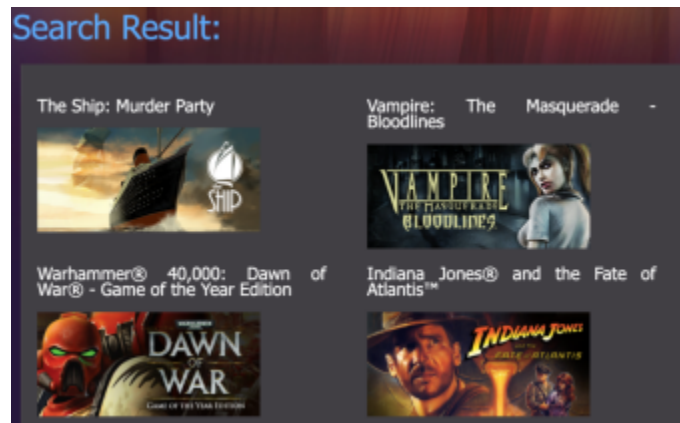


Figure 10: Search

```
@myViews.route('/search', methods=['Post', 'Get'])
def search():
    print(request.method)
    if request.method == 'POST':
        gameName = request.form.get('content')
        gameInfo = database.searchGame('%'+gameName+'%')
        return render_template("search.html", games = gameInfo)

    return redirect(url_for('views.fail'))

mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM gameInfo WHERE name LIKE %s", (name,))
result = mycursor.fetchall()
mydb.close()
return result

{% for game in games %}
<div class="results">
<div class="game">

</div>
</div>
{% endfor %}
```

Figure 11: Code snippet of search

When user makes search request, the search function will handle it. It will receive the key word from user by post method and execute `database.searchGame` to get the relevant games. The SQL command shown above will ask the database for the games that contains the key word. After all of those are done, `search.html` will be displayed with the games requested by user. In the HTML file, the for loop code in jinja2 is used to iterate through the game list and put it in the div tag.

4.3.5 Game List in the Admin Panel



Id	Name	Release time	Type	Thumbnail	Brief introduction	Action
1	Counter-Strike	2000-11-01	Action		Play the world's number 1 online action game. Engage in an incredibly realistic brand of terrorist warfare in this wildly popular team-based game. Ally with teammates to complete strategic missions. Take out enemy sites. Rescue hostages. Your role affects your team's success. Your team's success affects your role.	delete
2	Team Fortress Classic	1999-04-01	Action		One of the most popular online action games of all time, Team Fortress Classic features over nine character classes -- from Medic to Spy to Demolition Man -- enlisted in a unique style of online team warfare. Each character class possesses unique weapons, items, and abilities, as teams compete online in a variety of game play modes.	delete

Figure 12: Game list in the admin panel

```
@myViews.route('/game-list')
def gameList():
    games = database.getAllGame()
    return render_template("admin/game-list.html", games=games)

{% for game in games %}
<tr>
    <td>{{game[0]}}</td>
    <td>{{game[1]}}</td>
    <td>{{game[2]}}</td>
    <td>{{game[3]}}</td>
    <td><img class="game-Thumbnail" src={{game[6]}} alt=""></td>
    <td class="game-brief">{{game[4]}}</td>
    <td>
        <a href="{{url_for('views.deleteGames', id=game[0])}}"><button type="submit" class="btn
        btn-danger btn-sm">delete</button></a>
    </td>
</tr>
{% endfor %}
```

Figure 13: Code snippet of game list in the admin panel

The gameList function get all the games info from the database by executing `database.getAllGame` function. It then direct to game-list.html with the games info that just obtained from the database. In game-list.html, the for loop iterates all the game list and print it out, where `game[0]` is game id, `game[1]` is the game name, `game[2]` is the game type, `game[3]` is the release date, `game[4]` is the game description and `game[5]` is the image link. There is a delete button. When it is clicked, `deleteGames` function in views.py will be executed to delete the game info.

5 Earned Value Analysis

Our tasks are as follows:

1. Setup environment and learn about the theory about these tools
2. Implement basic website communication by post and delete
3. Implement login for admin and user
4. Complete UI and features combined
5. Test with some users' interests and gaming convention data
6. Write Project Report Paper

We measure the cost by *working hours per day* and perform earned value analysis as follows:

Activity	Predecessor	Duration (days)	Cost / Day	Total Cost
1	-	8	1h	8h
2	1	7	1h	7h
3	2	7	1.5h	10.5h
4	3	20	1h	20h
5	4	3	2h	6h
6	5	5	1h	5h

Table 6: Activities and cost

Activity	Actual % Completed	Incurred Cost
1	100	8h
2	100	9h
3	50	8h
4	80	22h
5	66	5h
6	0	0h

Table 7: Project report at the end of day 45

Activity	ACWP	BCWP	BCWS	CPI	CV	SPI	SV
1	8h	8h	8h				
2	9h	7h	7h				
3	8h	5.25h	10.5h				
4	22h	16h	20h				
5	5h	4h	6h				
6	0h	0	0				
Total to Date	52h	40.25h	51.5h	0.774	-11h	0.782	-11.25h

Table 8: Earned value analysis

6 Test

We downloaded real game data from Steam via the Python package SteamSpyPI to a CSV file. Then we wrote a Python program to parse this file and loaded the data to the database. We performed manual black box testing to check the following:

- Users can successfully login and logout.
- Users can view the games and comments loaded to the database.
- Admins can view the list of users, games, comments loaded to the database in the admin panel.
- When the user adds a comment, they get the correct feedback from the interface, and the data in the database is updated correctly.
- We also did some browser compatibility testing which includes safari and chrome.

7 Conclusion

After six weeks of hard work, our group finish the E-sport project a platform that users can view game information and make comments

7.1 Limitations

We have implemented most of the basic functionality, but there are still a few limitation and potential improvement.

- Games are currently only ranked by the number of likes.
- Users cannot add friends.
- Functionality of the admin panel is incomplete. Admin can only delete data and cannot update data from the admin panel.
- We did not have compatibility with mobile devices.
- Code is not very compact.

7.2 Future Enhancements / Recommendations

- Allow users to add friends to encourage interaction between users.
- Recommend games for users based on their interests and common games among their friends.
- Add more ways of ranking. For example, By release date and by number of download.
- Add more ways for the admin to manage the data.
- Improve the reusable of our code to make it more compact.
- Use a front-end framework by react.

7.3 Team Members

Bingxiang Lin

I currently know C, Java, and Python. But I also has the basic knowledge of HTML, CSS, and SQL. In this project, I was focus on the back-end development. I designed and managed the database and worked on the python flask.

Hongyi Zhang

At present, I have mastered the front-end technology of HTML, CSS and JS. In this project, In the completion of this project, how to make the overall layout was a difficult problem, and there were some technical difficulties in the overall page. I sought inspiration for page layout from various game websites and learned how to realize some functions by watching videos.

Jiashu Chen

I currently know Java, Python3, CSS, Java-script and HTML. In this project, I mainly did the following: 1 design the layout. 2 connect the front-end and the back-end of the admin panel. 3 write java-script, for example to search in list.

7.3.1 Learning Experience and Outcome

Bingxiang Lin

Through this project, I've learned a lot. For frontend, I got more knowledges on HTML, CSS and JavaScript. I am able to build a simple web. For backend, I'm so glad that I have the opportunity to apply the knowledge that I learned from Database Management course in this project. Now I have confidence to design and manage a database using MySQL on my own. I also mastered the python flask throughout this project. At first I got no knowledge of python flask and new to MySQL. It's quit challenging for me to start the project. Fortunately, there are many tutorials online where I can learn them from. As an outcome, I can totally develop a web application with the knowledge I've learned through this projects.

Hongyi Zhang

This project enabled me to learn a lot of new knowledge, such as using PS to create and modify the Logo, learning the overall layout of the page using the raster system of Bootstrap, and using the animate. CSS animation library, such as the fade in and out animation effect of the page. In the end, I completed the project through study and continuous attempts and realized most of the content.

Jiashu Chen

Through this project, I have a deeper understanding of team cooperation and collaboration capabilities needed for group projects. The project also gives me an opportunity to learn and use a lot of new technology and knowledge. I learned Python Flask along with the template engine. It has no database layer, so we used another library to connect to MySQL and wrote SQL in Python.

8 References

- [1] "Bootstrap · The most popular HTML, CSS, and JS library in the world." *Bootstrap*, <https://getbootstrap.com/>.

- [2] “jQuery.” *jQuery*, <https://jquery.com/>.
- [3] “MySQL Connector/Python Developer Guide.” *MySQL*, <https://dev.mysql.com/doc/connector-python/en/>.
- [4] “SteamSpyPI: an API for SteamSpy.”, *PyPI*, <https://pypi.org/project/steampypi/>.
- [5] “Welcome to Flask — Flask Documentation (2.0.x).” *Flask*, <https://flask.palletsprojects.com/en/2.0.x/>.