UNIVERSITY OF INFORMATION TECHNOLOGY VNU-HCM FALCUTY OF INFORMATION SYSTEM

SOFWARE ENGINEERING FINAL PROJECT

CYBER GAMING

A black background with a black square

Description automatically generated with medium confidence

Presented to: Lec. Nguyen Tuan Nam, Ass. Tran Vinh Khiem

Presented by: Team 5-CS4273.P12.CTTT

Table of Contents

[**PREFACE** 3](#_Toc185263245)

[**FINAL PROJECT: GAY CYBER GAMING** 3](#_Toc185263246)

[**CHAPTER 1: INTRODUCTION.** 4](#_Toc185263247)

[**CHAPTER 2: PURPOSE OF CYBER GAMING SYSTEM.** 5](#_Toc185263248)

[**CHAPTER 3: SCOPE OF CYBER GAMING SYSTEM.** 5](#_Toc185263249)

[**CHAPTER 4: FUNCTIONAL REQUIREMENTOF RESTAURANT MANAGEMENT SYSTEM.** 6](#_Toc185263250)

[**4.1 Functions of CG** 6](#_Toc185263251)

[**4.1.1 User Authentication and Access Control:** 6](#_Toc185263252)

[**4.1.2 Items Management** 6](#_Toc185263253)

[**4.1.3 Menu Management** 6](#_Toc185263254)

[**4.1.4 Billing and Payment** 7](#_Toc185263255)

[**4.1.5 Order processing** 7](#_Toc185263256)

[**4.1.6 Forgot Password Mechanism** 7](#_Toc185263257)

[**4.2 Use case diagrams and specifications** 7](#_Toc185263258)

[**4.2.1 User Authentication and Access Control** 7](#_Toc185263259)

[**4.2.2 Items Management** 11](#_Toc185263260)

[**4.2.3 Manage Account** 14](#_Toc185263261)

[**4.2.4 Order Processing** 19](#_Toc185263262)

[**4.2.5 Manage Computer** 21](#_Toc185263263)

[**4.2.6 View menu** 25](#_Toc185263264)

[**4.3 Activity Diagram** 29](#_Toc185263265)

[**4.3.1. User** 29](#_Toc185263266)

[**4.3.2 Admin** 38](#_Toc185263267)

[**4.4 Sequence Diagram** 53](#_Toc185263268)

[**4.4.1 User** 53](#_Toc185263269)

[**4.4.2 Admin** 61](#_Toc185263270)

[**4.5 Class Diagram** 74](#_Toc185263271)

[**4.6 Entity Relationship Diagram** 75](#_Toc185263272)

[**CHAPTER 5: NON-FUNCTIONAL REQUIREMENT OF CYBER GAMING** 76](#_Toc185263273)

[**5.1 Usability** 76](#_Toc185263274)

[**5.2 Scalability** 76](#_Toc185263275)

[**5.3 Maintainability** 76](#_Toc185263276)

[**5.4 Performance** 77](#_Toc185263277)

[**5.5 Compatibility** 77](#_Toc185263278)

[**5.6 Reliability** 77](#_Toc185263279)

[**5.7 Security** 78](#_Toc185263280)

[**5.8 Availability** 78](#_Toc185263281)

[**CHAPTER 6: TEST CASE** 78](#_Toc185263282)

[**6.1 Login** 78](#_Toc185263283)

[**6.2 Order items** 79](#_Toc185263284)

[**6.3 Update item (game, food, drink, card) menu** 80](#_Toc185263285)

[**6.4 Update menu game** 81](#_Toc185263286)

[**6.5 Test case with account** 83](#_Toc185263287)

[**6.6 Payment** 84](#_Toc185263288)

[**CHAPTER 7: CONCLUSION** 85](#_Toc185263289)

[**7.1 Result** 85](#_Toc185263290)

[**7.2 Evaluation** 85](#_Toc185263291)

[**7.2.1 Advantage** 85](#_Toc185263292)

[**7.2.2 Disadvantage** 86](#_Toc185263293)

# **PREFACE**

The current period has seen an expansion in information technology, which has changed the nature and scope of operations for many enterprises, organizations, and people. The field of cyber management is one of the ways to show the use of technology applications not only helps optimize procedures but also opens up new prospects. Creating a modern cyber gaming management system is not only a trend, but also essential to improve service quality for the customers who enjoy their leisure time the most, especially in an era when people are becoming increasingly busy and have no time for themselves.

Our culminating project focused on constructing a robust, multifunctional cyber gaming management system. Throughout the implementation process, we received dedicated support from Assistant Lecturer Tran Vinh Khiem, while also benefiting from the specialized insights shared by Lecturer Nguyen Tuan Nam. The knowledge and skills we acquired not only deepened our understanding of software technology, but also empowered us to apply it effectively during the development of our cyber gaming management solution.

We wish to sincerely express our gratitude to Assistant Lecturer Tran Vinh Khiem for his unwavering commitment and extensive efforts in guiding us through this undertaking. Within this report, we will thoroughly explore the system's functionalities, outline the development process, and discuss the challenges we encountered. We trust that by the end of this document, readers will have gained a comprehensive insight into the work we have accomplished.

# **FINAL PROJECT: GAY CYBER GAMING**

Authors:

Ngo Thanh Trung

Doan Hoai Viet

Pham Duy Tuan

Nguyen Khanh Tuan

Ha Nhat Thai

Huynh Phan Nhat Thang

# **CHAPTER 1: INTRODUCTION.**

In Vietnam, computer games (PC) continue to attract a large number of players, particularly in the field of eSports. According to a report by Decision Lab and Vero, 44.9% of eSports players in Vietnam use the PC platform, highlighting the popularity of computers within the gaming community.

This preference stems from the immersive experience that PCs provide and the longstanding habit of playing games at internet cafes over the years. Additionally, PC games such as Than Trung by DUT Studio have achieved significant success, topping the trending charts on Steam on the day of release, surpassing even Dota 2 and GTA V.

In the fast-evolving digital era, the adoption of Cyber Gaming (CG) has become a pivotal trend. Confronted with the challenges of low-spec computers and increasing demand for operational efficiency, gaming businesses are embracing CG as an all-in-one solution. This strategic move is fueled by the need to secure a competitive edge in a technology-centric age and fulfill the growing expectations of both customers and industry benchmarks. By offering a variety of games and associated services, CG effectively tackles existing operational obstacles while promising to elevate the overall gaming experience.

# **CHAPTER 2: PURPOSE OF CYBER GAMING SYSTEM.**

The purpose of the Software Requirements Specification (SRS) document for a Cyber Gaming (CG) is to provide a comprehensive and detailed blueprint for the development and implementation of the system.

This essential document delineates both functional and non-functional requirements of the RMS, acting as a key channel of communication among all stakeholders, including developers, project managers, and end users.

The SRS clearly specifies the system’s features, functionalities, limitations, and performance benchmarks, ensuring a unified comprehension of the project’s scope and objectives. By detailing user expectations, system behavior, and technical parameters, the SRS lowers the risk of misinterpretation and guarantees that the resulting Cyber Gaming aligns with the intended business requirements.

At its core, the SRS functions as a pivotal guide, directing the development team towards producing an effective, fully operational, and customized cyber management solution.

# **CHAPTER 3: SCOPE OF CYBER GAMING SYSTEM.**

The Cyber Gaming (CG) solution is a specialized software platform that streamlines and enhances various operational facets of gaming activities. More specifically, it equips administrators and users with key features—ranging from order management to financial oversight and supplemental services—ultimately elevating the overall efficiency and profitability of the enterprise.

The scope of cyber gaming system encompasses the following key functionalities:

* **Order Management**: The system facilitates order processing, items management, and menu customization, ensuring swift and accurate order execution. It includes features for order placement and tracking to enhance the customer experience.
* **Seamless Billing Process**: The system will handle billing, payment processing, and receipt generation. It accommodates various payment methods and ensures accuracy in financial transactions.
* **Security:** Databases contain information of admins and users, can easy to check, add, delete account. Each role will be guaranteed to have limited access to the system
* **Forgot Password Mechanism:** Offering a secure and user-friendly method for users to reset their passwords through email verification.

# **CHAPTER 4: FUNCTIONAL REQUIREMENTOF RESTAURANT MANAGEMENT SYSTEM.**

## **4.1 Functions of CG**

### **4.1.1 User Authentication and Access Control:**

- Administrator: Unlimited privileges to utilize every available system feature.

- User: Access to order management and table-related functionalities.

- Access control limit certain functionalities based on user roles.

### **4.1.2 Items Management**

- The system allows staff to assign and track items include foods, drinks,s games… and manages the amount.

- Administrator:Can track all the items and easily to add/delete items if needed

- User: Friendly UI so that users can find any things they want easy and fast

### **4.1.3 Menu Management**

- Menu items categorized into logical groups or sections (e.g., foods, drinks, cart,…) to facilitate easy navigation and ordering.

- The system provides an intuitive interface for authorized manager to create new menu items, including details such as item name, category, pricing, and description.

- The system should facilitate the removal of menu items, ensuring that any outdated or discontinued offerings are no longer displayed.

- The system enables admin to establish and adjust menu item prices, taking into account various cost factors.

- The system permits the uploading and linking of images to each menu item, enhancing its visual appeal and assisting customers in making informed choices.

### **4.1.4 Billing and Payment**

- Generation of accurate bills.

- Integration with payment processors and receipt printing.

### **4.1.5 Order processing**

- The order creation process supports the addition of multiple items per order, including different quantities and various order types, such as food, drink, card...

- Order information, including item details and pricing, shall be synchronized.

- Restaurant staff have the capability to mark orders as complete, updating the order status accordingly.

### **4.1.6 Forgot Password Mechanism**

- Secure Citizen Identification Card-based to reset password

- Verification process to ensure the identity of the user

- Log of password reset activities for audit purposes

## **4.2 Use case diagrams and specifications**

### **4.2.1 User Authentication and Access Control**

This component of the cyber gaming system focuses on account verification and associated functionalities. It acts as the secure entry point for users to log into their accounts and provides mechanisms for account recovery in situations where passwords are forgotten.

A diagram of a diagram

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Name** | **Register** | **Login** |
| Scenario | A user or admin registers an account to access the CYBER GAMING management system. | A user or admin initiates the process to gain access to their account on the CYBER GAMING management software. |
| Description | This use case describes how a new user or admin creates an account by providing all required information and meeting system requirements. | This use case describes the process by which a user or admin logs into the CYBER GAMING management system to access functionalities specific to their roles. |
| Actor | User, Admin | User, Admin |
| Trigger | The actor selects the "Register" option on the login/registration interface. | The actor selects the "Login" option on the main interface. |
| Related Use Cases | Login, Forgot Password | Forgot Password, Exit |
| Pre-condition | - The system must be operational and accessible.  - The actor must provide a valid, unique email. | - The actor must have an active account with a username and password.  - The CYBER GAMING management system must be operational and accessible. |
| Post-condition | - The system saves the actor's information and redirects them to the login page. | - If the login is successful, the actor is granted access to the system with privileges according to their role.  - If the login fails, the system remains at the login screen, allowing retry or password recovery. |
| Flow of Events | 1. The actor selects the "Register" option.  2. Fills out the registration form.  3. Submits the form.  4. The system validates data.  5. If successful, data is saved.  6. If validation fails, error messages are shown. | 1. The actor enters the login screen and inputs their username and password.  2. Submits the information by selecting the "Login" button.  3. The system verifies credentials.  4. If credentials match, access is granted.  5. If credentials do not match, an error message is displayed. |
| Exception Conditions | If the email is already registered, the system suggests recovering the account. | If incorrect login information is entered multiple times, the account may be temporarily locked. |

|  |  |
| --- | --- |
| **Name** | **Forgot Password** |
| Scenario | A user or admin attempts to recover their account after forgetting their password. |
| Description | This use case describes the process by which a user or admin can recover their password if they have forgotten it. |
| Actor | User, Admin |
| Trigger | The actor selects the "Forgot Password" option on the login screen. |
| Related Use Cases | Login, Exit |
| Pre-condition | - The actor must have an active account with a registered email.  - The system must be operational and accessible. |
| Post-condition | - The actor receives a password recovery link via email.  - The system records the recovery action for security auditing. |
| Flow of Events | 1) User  1.1 Selects the "Forgot Password" option.  1.2 System check account and inform for admin  1.3 Admin receive notification and choose whatever password to reset 2) Admin  There is no way for admin to reset password themselves |
| Exception Conditions | -Admin forgot password so that the system can’t work, they must lend other account to continue the business  -Admin must check whether that account is really belonged to that user by manual method e.g. Citizen Identification Card |

|  |  |
| --- | --- |
| **Name** | **Exit** |
| Scenario | A user or admin wants to exit the application. |
| Description | This use case enables the user or admin to terminate their session and close the application. |
| Actor | User, Admin |
| Trigger | The actor selects the "Exit" option within the application. |
| Related Use Cases | Login |
| Pre-condition | - The actor must be logged into the system. |
| Post-condition | - The application is closed, and the actor’s session ends. |
| Flow of Events | 1. The actor selects the "Exit" option.  2. The system logs the actor out.  3. The application closes, returning the actor to their device's home screen or desktop. |

### **4.2.2 Items Management**

The "**Items Management**" module within the Cyber Gaming platform focuses on overseeing and updating information for the items featured on the menu. This component ensures that the menu remains current and accurately represents the offerings. Its core functionalities include adding, deleting, editing, and searching for items.

A diagram of a diagram

Description automatically generated

|  |  |
| --- | --- |
| **Name** | **Add New Items** |
| Scenario | Admin add new items include cards, new foods, drinks, combos for players into the menu |
| Description | This use case describes the progress of adding new items |
| Actor | Admin |
| Trigger | The need to expand the menu with new items |
| Pre-condition | The actor must be logged into the system |
| Post-condition | -That items not exist in the database  -Database update successfully the detail of that items  -Show on the menu for players to choose |
| Flow of events | 1. Admin log in their account for admin  2. Choose the sections they want to add  3. Enter the details for items, e.g. name, price, quantity, image  4. Click add  5. The system check if that items already exist  6. If that items are unique, then a pop-up inform add successfully, else it inform error |
| Exception conditions | - If that items existed, then system decline to add that items |

|  |  |
| --- | --- |
| **Name** | **Delete Existing Items** |
| Scenario | Admin delete existing items include cards, new foods, drinks, combos for players out of the menu |
| Description | This use case describes the progress of delete existing items |
| Actor | Admin |
| Trigger | The need to shrink the menu with new items |
| Pre-condition | The actor must be logged into the system |
| Post-condition | -That items already exist in the database  -Database update successfully the after deleting that items  -The menu will not display that items anymore |
| Flow of events | 1. Admin log in their account for admin  2. Choose the sections they want to delete  3. Click delete  4. A pop-up will ask again if admin want to delete that items  5. The system check if that items already exist  6. If that items existed, then a pop-up inform delete successfully, else it inform error |
| Exception conditions | - If that items not exist, then system decline to delete that items  - System will create a back-up storage to recover the action, will delete permanently after 30 days |

|  |  |
| --- | --- |
| **Name** | **Check the number of items** |
| Scenario | Admin or user want to see if the items are in stock |
| Description | This use case describes the progress of checking the number items |
| Actor | Admin, user |
| Trigger | The need to show the current quantity of that items |
| Pre-condition | -The actor must be logged into the system  -That item must exist in the database and display on the menu |
| Post-condition | -Always update the quantity of that items in database  -If they are out of stock, pop-up will display a error message, otherwise update the database |
| Flow of events | 1) User:  1.1 Log in their account  1.2 Choose the section they want to make order  1.3 If the items are out of stock, a notify will be displayed below that items and they can’t do anything with it  1.4 If the item is still in stock but the quantity ordered exceeds the available stock, reject the transaction.  1.5 If the item is still in stock and the quantity ordered matches the available stock, proceed with the transaction and display an 'Out of Stock' message below the item.  1.6 Database update correctly the numbers of items  2) Admin  2.1 Log in their account  2.2 Choose the section they want to check |
| Exception conditions | - Database makes mistake in calculating the quantity of items, e.g. quantity below 0,..  - Database don’t update the quantity of items |

### **4.2.3 Manage Account**

The "**MANAGE ACCOUNT**" section within the cyber gaming system is designed to handle user account administration and ensure the security and accuracy of user data. This section offers critical functionalities for account management, such as adding new accounts, deleting inactive or unnecessary accounts, and searching for existing accounts. By providing these features, the system helps maintain up-to-date and secure user information.

A diagram of a diagram

Description automatically generated

|  |  |
| --- | --- |
| **Name** | **Add account** |
| Scenario | Admin add new account into the database, as well as the money and play hour of that account |
| Description | This use case describes the progress of add new account |
| Actor | Admin |
| Trigger | The need to grant system access for user |
| Pre-condition | - Admin logged into their account  - User registered account |
| Post-condition | -Username must not contain “Admin”  -Database for user update successfully  -User can use all the service provided |
| Flow of events | 1. Admin log in their account for admin  2. User registered account  3. System check if the username contain “Admin”  4. If contain “Admin” show error message, else check the play hour they want and calculate the money they need to pay  5. Admin receive the notification  6. Choose add to add it into the database for user  7. System keep track of that account |
| Exception conditions | -If a user intentionally enters a username containing "Admin" more than three times, a notification should be sent to the admin, and a fine will be imposed on the user.  -If a user selects too few hours of gameplay, the admin has the right to deny the account addition. |

|  |  |
| --- | --- |
| **Name** | **Delete account** |
| Scenario | Admin delete existed account out of the database |
| Description | This use case describes the progress of delete existed account |
| Actor | Admin |
| Trigger | The need to revoke all access of user |
| Pre-condition | - Admin logged into their account  - User had a account |
| Post-condition | - Database don’t store any information about that account  - User don’t have access to log in by that account |
| Flow of events | 1. Admin log in their account for admin  2. The user registered a valid account  3. Admin check if that account wasn’t use for a long time or was use for illegal job  5. Choose that account and click delete  6. System send a ask again if admin want to delete  7. Choose OK and database will delete all information about that account |

|  |  |
| --- | --- |
| **Name** | **Edit account** |
| Scenario | Admin or user want to change information of account |
| Description | This use case describes the progress of editing a existing account |
| Actor | Admin, actor |
| Trigger | The need to edit the information like username, password if it was forgotten |
| Pre-condition | - Admin have all access in editing user account  - User can access the profile section to change whatever information they want |
| Post-condition | - System show new update information  - Database update successfully new information |
| Flow of events | Case 1: Admin edit  1. Admin choose user account section  2. Find and choose a specify section  3. System show that account  4. Admin choose things they want to change like username, password, money  5. Click OKAY  6. System verify new information  7.1. If success, system prompt a successful message and database will be updated  7.2. If fail, system prompt a error message and everything remain still  Case 2: User edit  1. User choose profile section  2. Choose the thing they want to change  3. System verify new information  4.1. If success, system prompt a successful message and database will be updated  4.2. If fail, system prompt a error message and everything remain still  5. System will give admin a notification and store history change |
| Exception conditions | - Only admins have higher access in editing account like money or play hour, so when user buy more hour play, admin need to edit carefully to avoid loss for user  - User may try to steal somebody account and change it, |
|  | **Find account** |
| Scenario | Admin find account in the database |
| Description | This use case describes the progress of finding existing account |
| Actor | Admin |
| Trigger | The need to check if that account exist in the database |
| Pre-condition | - Admin logged into their account  - User registered account |
| Post-condition | -Username must not contain “Admin”  -Search bar show correctly the account admin want to check |
| Flow of events | 1. Admin log in their account for admin  2. Admin input username in search bar  3. System check if that account exist  5. Display all related usernames if the username is not entered fully and belonged detail of that account |
| Exception conditions | -Admin may enter the wrong username so that system can’t find that account |

|  |  |
| --- | --- |
| **Name** | **Check account profile** |
| Scenario | Admin, user want to check detail of account e.g. name, money left, play hour,.. |
| Description | This use case describes the progress of checking account profile |
| Actor | Admin, user |
| Trigger | The need to verify the information of that account for security |
| Pre-condition | - Admin logged into their account  - User registered account |
| Post-condition | - The actor has accessed detailed information about user accounts.  - The system displays comprehensive information about each account. |
| Flow of events | 1) User:  1.1 Choose the profile on the web  1.2 System displays name, an area to check or change the password  2) Admin:  2.1 Admin choose the section contain the list account  2.2 Double click on the account they want to check  2.3 System display all the status, name, password, hour play of that account |
| Exception conditions | - Database has a copy account list just in case system failure,  so that information of user will not disappear |

### **4.2.4 Order Processing**

Within the Cyber Gaming system, the **Manage Order Information** section is crucial for monitoring and streamlining all order-related activities. This section includes several essential features, such as selecting combos, choosing cards, picking food and beverages, specifying the quantities of food and drinks, generating bills, applying discounts, and offering the option to cancel orders.

A diagram of a diagram

Description automatically generated

|  |  |
| --- | --- |
| **Name** | **Buy food, drink, cards or combos** |
| Scenario | User add what they like, admin receive order and service |
| Description | This use case describes the order processing |
| Actor | Admin, user |
| Trigger | The need to service the user what they want |
| Pre-condition | - Admin logged into their account  - User registered account  - Items are available on the menu |
| Post-condition | - Enough foods, drinks, cards  - Enough money  - Admin receive order and confirm  - Database update new quantity after a success order |
| Flow of events | 1) User  1.1. Log in the web  1.2. Choose sections they want  1.3. Choose specify items they want  1.4. Choose quantity  1.5. Confirm order  2) Admin  2.1. Log in the web  2.2. Receive order  2.3. Confirm order  2.4. Database update new quantity  2.5. Deliver the service and collect money |
| Exception conditions | - Not enough quantity in the database  - Not enough money of user  - Order things that don’t exist |

|  |  |
| --- | --- |
| **Name** | **Edit the order** |
| Scenario | Users want to add, subtract or cancel the order |
| Description | This use case describes the edit order processing |
| Actor | User |
| Trigger | The need to click add, subtract, delete button in the payment section |
| Pre-condition | - User registered account  - Items are available on the menu  - Items already in the cart |
| Post-condition | - Screen display exactly the quantity of that items  - Calculate price match with the order |
| Flow of events | 1. Check payment section  2. choose option needed to change e.g. add, subtract, delete,..  3. System calculate the price  4. Click OK to process the order |
| Exception conditions | - When quantity equal 0 if click more subtract may resulting in negative number  - System can calculate the wrong price |

|  |  |
| --- | --- |
| **Name** | **Make payment** |
| Scenario | User want to make payment for his order |
| Description | This use case describes the payment process |
| Actor | User |
| Trigger | The need to click “add to card” on the payment section |
| Pre-condition | - User registered account  - Items are in the payment section |
| Post-condition | - System confirm items in stock  - System confirm money  - Admin receive the notification |
| Flow of events | 1. Check payment section  2. Click “Add to card”  3. System check database to see if there are enough stock  4. System accept the order  5. System send admin a notification |
| Exception conditions | - When an order is made, database will update, so that if customer don’t have the money, admin must flexible to handle the situation |

### **4.2.5 Manage Computer**

The **Manage Computer** module within the cyber gaming platform is designed to oversee and organize all computer-related information. This section offers vital tools for effectively handling table assignments and monitoring availability. Its main features include the ability to shut down computers, locate specific machines, check their current status, and view detailed specifications.

**A diagram of a computer

Description automatically generated**

|  |  |
| --- | --- |
| **Name** | **Find computer** |
| Scenario | Admin want to find specific computers |
| Description | This use case describes the progress finding particular computers and detail belonged to them |
| Actor | Admin |
| Trigger | The need to find if is there any available computer, or computer is logged in but there is no one use, or computer is used for illegal thing |
| Pre-condition | - Admin logged into their account  - Database has information of all computer with unique id for each computer |
| Post-condition | - System must display all detail belong to that computer  - Admin have access to determine the activity of that computer |
| Flow of events | 1. Admin choose to computer section  2. System show a list of all computer with their summary detail  3. Admin search a specify computer in search bar or tick the common attribute of computer in the box  4. System filter and show the most suitable computer |
| Exception conditions | - If all detail were entered but nothing happen, there will be a message to suggest admin to reenter information |

|  |  |
| --- | --- |
| **Name** | **Shutdown computer** |
| Scenario | Admin want to find specific computers |
| Description | This use case describes the progress shutdown a computer that inactive or used for wrong purpose |
| Actor | Admin |
| Trigger | The need to prevent computer being inactive in a long time or using for illegal things that can cause harm for computer |
| Pre-condition | - Admin choose computer manage section  - That computer is in “online” status |
| Post-condition | - That computer is shut down  - The status of it is “offline” |
| Flow of events | 1. Admin choose to computer section  2. System show a list of all computer with their summary detail  3. Admin find a specific computer and choose the wanted one  4. Double click and choose shutdown  5. System prompt a confirm action form  6. Admin confirm then system immediately locate that computer and shutdown all running application on that computer  7. Update the status of that computer |
| Exception conditions | - User download virus that system can’t control that computer  - If admin computer suddenly turn off, there is no way to control user computer from far distance |

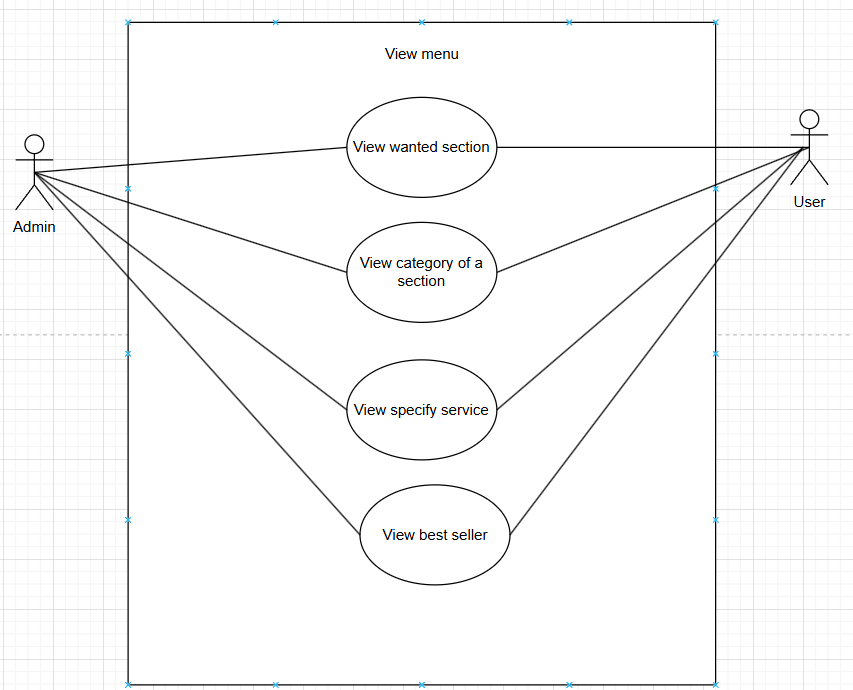
|  |  |
| --- | --- |
| **Name** | **Check status computer** |
| Scenario | Admin want to check detail information of specific computer |
| Description | This use case describes the progress of finding information like status, running applications of that computer |
| Actor | Admin |
| Trigger | The need to see if that computer is available, almost run out of money or doing illegal things |
| Pre-condition | - Admin choose computer manage section  - That computer exist in the database |
| Post-condition | - System display clearly information of that computer |
| Flow of events | 1. Admin choose computer management section  2. System show a list of computers with their summary information like money left, “online”, “offline”, “sleep” status  3. Admin choose a specify computer  4. System show all details information of that computer |
| Exception conditions | - If virus was downloaded, system may not detect the dangerous and lose control for that computer  - System may deep into the private information of user |

|  |  |
| --- | --- |
| **Name** | **Check specification of computer** |
| Scenario | Admin want to check Technical Specifications of the Computer. |
| Description | This use case describes the progress of check information about CPU, RAM, Storage, Card,… of that computer |
| Actor | Admin |
| Trigger | The need to see the ability of computer and what application that it can download and use |
| Pre-condition | - Admin choose computer manage section  - That computer specification exist in the database |
| Post-condition | - System display clearly information of that computer |
| Flow of events | 1. Admin choose computer management section  2. System show a list of computers with their summary information  3. Admin choose a specify computer  4. Admin change to page that show technical specification  5. System show all the details |
| Exception conditions | - User unaware of the ability of computer and may download some heavy application like high graphic game that can cause computer crash  - Admin unaware of the ability of computer so that they don’t know how to guide user |

### **4.2.6 View menu**

The **View menu** section of the cyber system is designed with a user-friendly interface, allowing customers to easily browse the available food and drink items, the games they can play, money with play hour left, check their profile. The main features include:

* View Food
* View Food Categories
* View Games
* View Game Categories
* View Drinks
* View Drink Categories
* View Combos
* View Best seller
* View Cards
* View profile
* View balance



|  |  |
| --- | --- |
| **Name** | **View section** |
| Scenario | User, admin want to check information of their account and the provided services |
| Description | This use case describes the progress of viewing services provided by cyber |
| Actor | User, Admin |
| Trigger | The need to check the sections in the main page |
| Pre-condition | - Sections, category is displayed on main page  - User had login in the system |
| Post-condition | - Show correctly what user, admin chose |
| Flow of events | 1. User log in successfully in the system  2. Find what he/she want, e.g. Game, food... section in the main page |
| Exception conditions | User, admin may not find the section that he/she want |

|  |  |
| --- | --- |
| **Name** | **View best seller** |
| Scenario | Users check detail what is the best service of cyber |
| Description | This use case describes the progress of viewing best seller service |
| Actor | User, admin |
| Trigger | The need to find quickly best option after logging in |
| Pre-condition | - Database has information of best seller service  - User logged in successfully |
| Post-condition | - All the best seller were shown on the main page  - Change different kind of services continuously |
| Flow of events | 1. User, admin log in successfully in the system  2. System displayed best seller directly in the main page |
| Exception conditions | - System don’t update information for best seller, so that it may display non-existing services, best seller in the past |

|  |  |
| --- | --- |
| **Name** | **View category** |
| Scenario | User , admin check different kind of a service |
| Description | This use case describes the progress of viewing categories of a section |
| Actor | User, admin |
| Trigger | The need to find quickly the item without checking all items |
| Pre-condition | - User, admin chose a section that has category  - Database store correctly item for each category |
| Post-condition | - System display correctly the item match with its category |
| Flow of events | 1. Users, admin find the section they want and choose  2. Check if that section contain category  3. User choose a specify category  4. System show all items belong to that category |
| Exception conditions | - Two different categories may include same item  - Item showed in category may be not belong to that category, make it harder for user to find |

|  |  |
| --- | --- |
| **Name** | **View profile account** |
| Scenario | User check his detail account |
| Description | This use case describes the progress of viewing user’s account |
| Actor | User |
| Trigger | The need to check again user’s information by selecting on the menu |
| Pre-condition | - User had a valid account  - Menu display profile section |
| Post-condition | - System display correctly information signed up by that user |
| Flow of events | 1. User log in successfully  2. Choose profile section on the main page  3. System show information of that user |
| Exception conditions | - User may mistake their log in resulting a different account  - In case system fail, data must be stored in other database so that users still have their information |

|  |  |
| --- | --- |
| **Name** | **View balance** |
| Scenario | Users check money left in their account |
| Description | This use case describes the progress of viewing money left of users |
| Actor | User, admin |
| Trigger | The need to check how many money they spent by choosing balance section |
| Pre-condition | - User had a valid account  - System keep track of that account and always update money |
| Post-condition | - System display correctly money left by time or after using a service |
| Flow of events | 1. User log in successfully  2. Choose balance section on the main page  3. System show money left, history transaction |
| Exception conditions | - User may forget what they order or don’t know the price per hour, so that they can report their balance is incorrect |

## **4.3 Activity Diagram**

### **4.3.1. User**

#### **4.3.1.1 Log in/Sign up**

A screenshot of a diagram

Description automatically generated

#### **4.3.1.2 Log out**

A diagram of a process

Description automatically generated

#### **4.3.1.3 Buy items (Food, drink, combos, cards)**

A diagram of a product

Description automatically generated

#### **4.3.1.4 Edit order**

A diagram of a system

Description automatically generated

#### **4.3.1.5 Edit account**

A screenshot of a diagram

Description automatically generated

#### **4.3.1.6 Make payment**

A screenshot of a diagram

Description automatically generated

#### **4.3.1.7 Forgot password**

A diagram of a system

Description automatically generated

#### **4.3.1.8 View food menu**

A diagram of a system

Description automatically generated

#### **4.3.1.9 View drink menu**

A diagram of a system

Description automatically generated

#### **4.3.1.10 View game menu**

A diagram of a system

Description automatically generated

### **4.3.2 Admin**

#### **4.3.2.1 Log in/Sign up**

A screenshot of a diagram

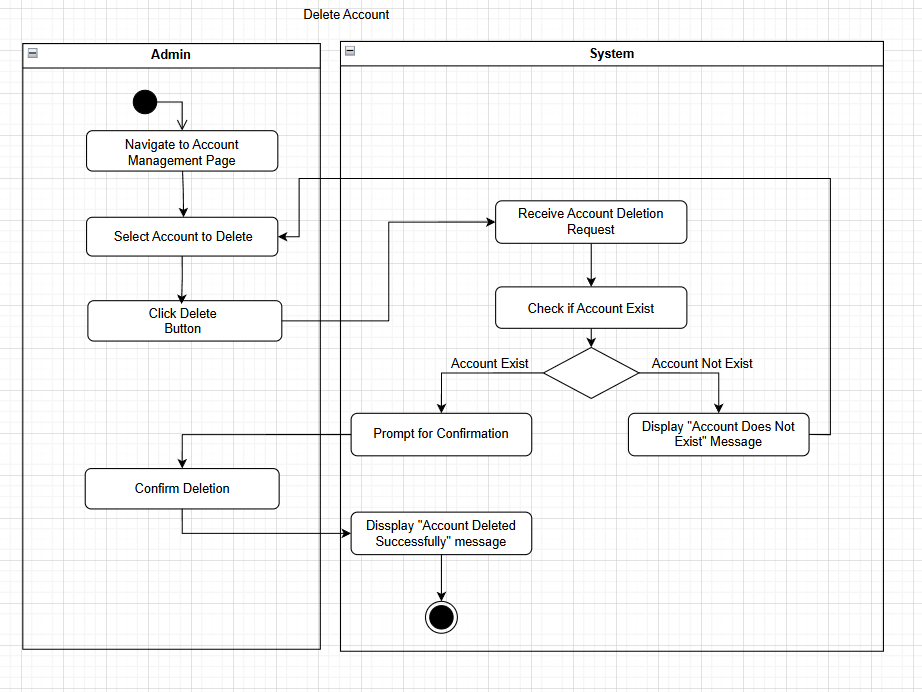
Description automatically generated

#### **4.3.2.2 Add account**

A screenshot of a diagram

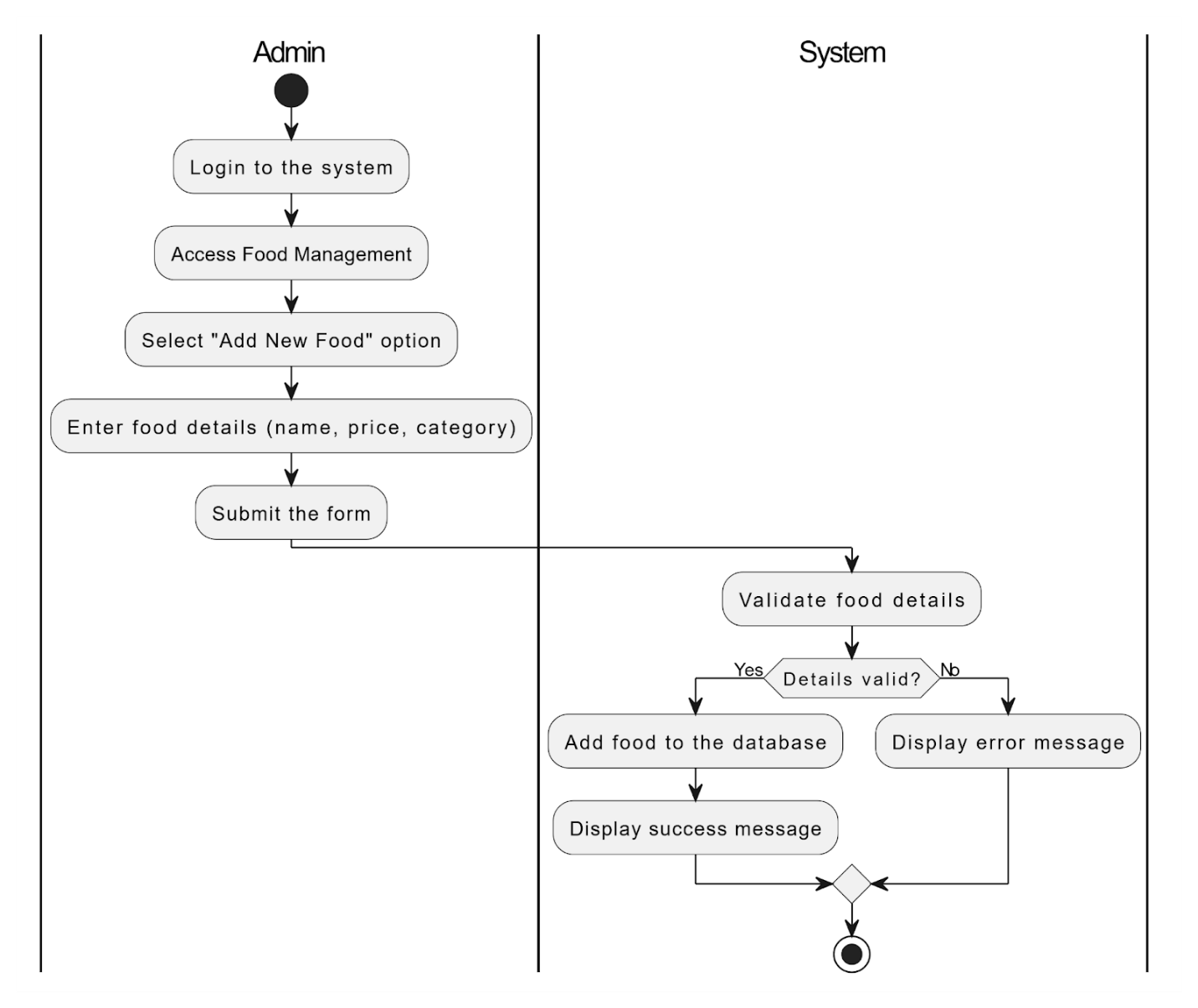
Description automatically generated

#### **4.3.2.3 Delete account**



#### **4.3.2.4 Add new items**

##### *4.3.2.4.1 Add food*



##### *4.3.2.4.2 Add drink*

A diagram of a system

Description automatically generated

##### *4.3.2.4.3 Add game*

A diagram of a system

Description automatically generated

#### **4.3.2.5 Update play hour**

A screenshot of a computer

Description automatically generated

#### **4.3.2.6 Delete existing items**

##### *4.3.2.6.1 Delete existing food*

A screenshot of a chat

Description automatically generated

##### *4.3.2.6.2 Delete existing game*

**A screenshot of a diagram

Description automatically generated**

##### *4.3.2.6.3 Delete existing drink*

A screenshot of a diagram

Description automatically generated

#### **4.3.2.7 Find Computer**

A diagram of a computer

Description automatically generated

#### **4.3.2.8 Shutdown Computer**

**A diagram of a computer system

Description automatically generated**

#### **4.3.2.9 Check status of computer**

**A diagram of a computer system

Description automatically generated**

#### **4.3.2.10 Check specification of computer**

**A diagram of a computer

Description automatically generated**

#### **4.3.2.11 View food menu**

A screenshot of a computer

Description automatically generated

#### **4.3.2.12 View drink menu**

A screenshot of a computer

Description automatically generated

#### **4.3.2.13 View game menu**

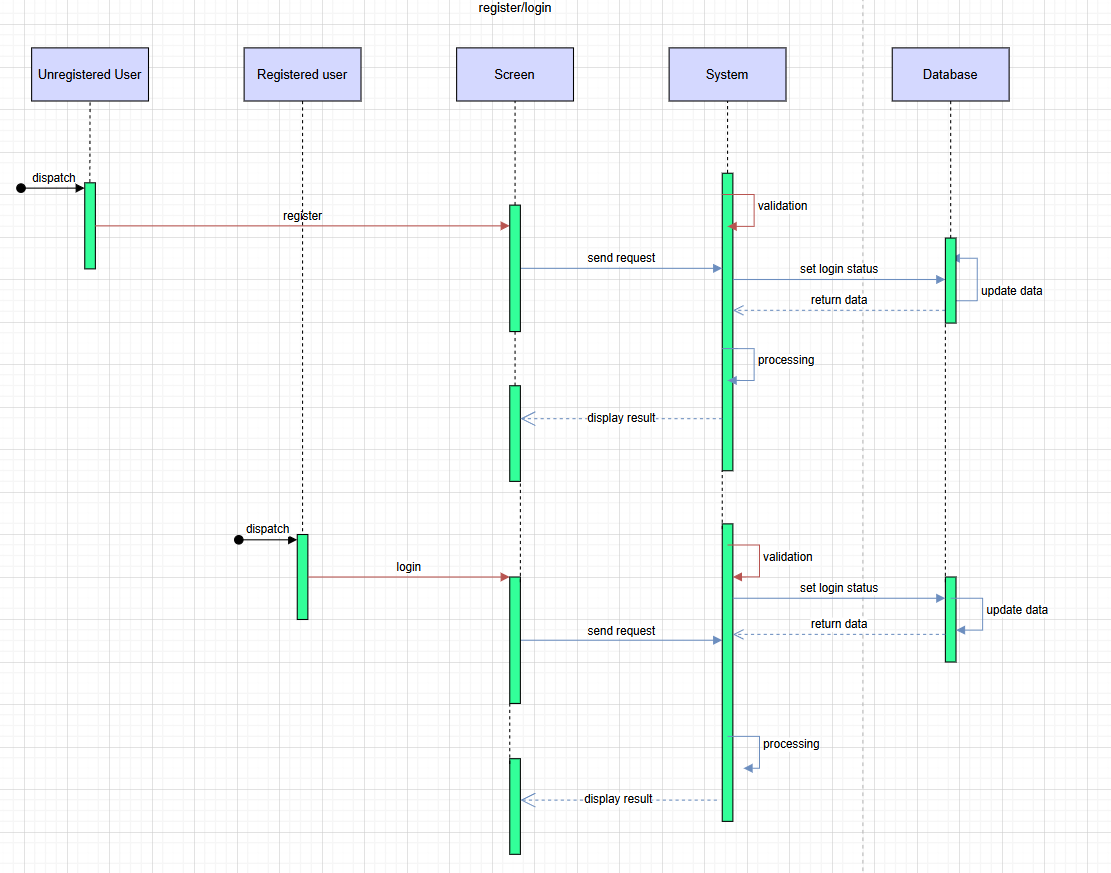
A screenshot of a computer program

Description automatically generated

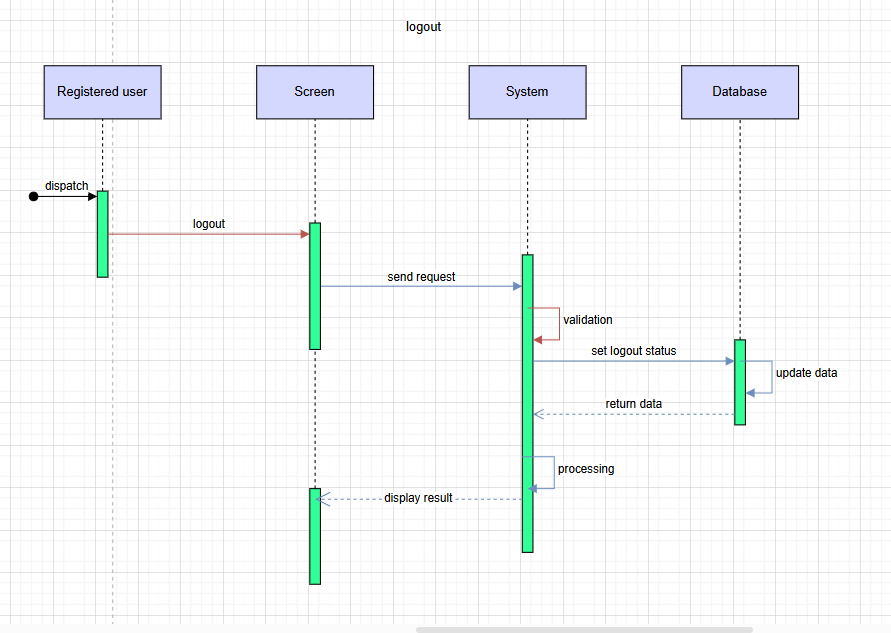
## **4.4 Sequence Diagram**

### **4.4.1 User**

#### **4.4.1.1 Log in/sign up**

****

#### **4.4.1.1 Log out**

****

#### **4.4.1.2 Buy items**

A diagram of a project

Description automatically generated

#### **4.4.1.3 Edit order**

A diagram of a screen

Description automatically generated

#### **4.4.1.4 Edit account**

A diagram of a diagram

Description automatically generated

#### **4.4.1.4 Make payment**

A diagram of a diagram

Description automatically generated

#### **4.4.1.5 Forgot password**

A diagram of a computer program

Description automatically generated

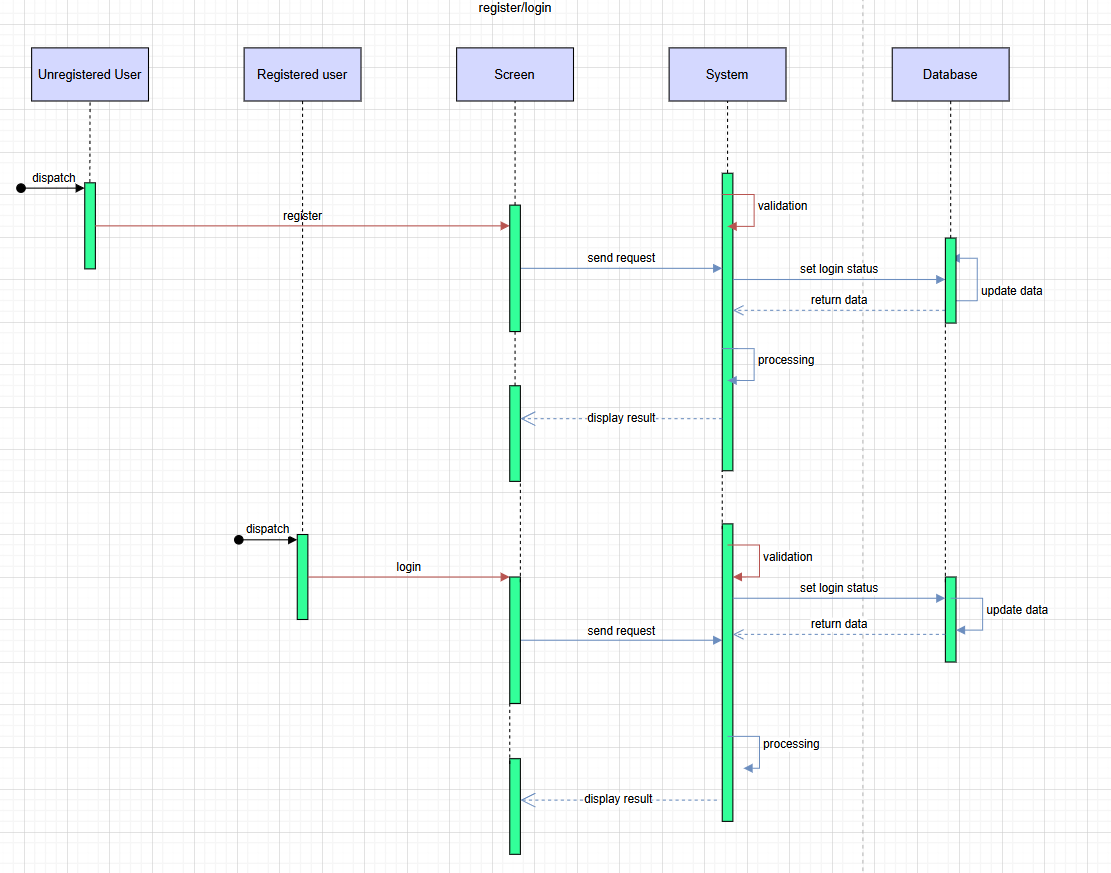
#### **4.4.1.6 View item menu**

A diagram of a diagram

Description automatically generated

### **4.4.2 Admin**

#### **4.4.2.1 Log in/Sign up**



#### **4.4.2.2 Add account**

A diagram of a system

Description automatically generated

#### **4.4.2.3 Delete account**

A diagram of a company

Description automatically generated

#### **4.4.2.4 Add new items**

##### *4.4.2.4.1 Add new food*

A screenshot of a computer

Description automatically generated

##### *4.4.2.4.2 Add new drink*

A screenshot of a computer

Description automatically generated

##### *4.4.2.4.2 Add new game*

A screenshot of a computer

Description automatically generated

#### **4.4.2.5 Update play hour**

A diagram of a project

Description automatically generated

#### **4.4.2.6 Delete existing items**

##### *4.4.2.6.1 Delete existing food*

A diagram of a project

Description automatically generated

##### *4.4.2.6.2 Delete existing drink*

A screenshot of a computer screen

Description automatically generated

##### *4.4.2.6.3 Delete existing game*

A screenshot of a computer screen

Description automatically generated

#### **4.4.2.7 Find computer**

A diagram of a computer process

Description automatically generated

#### **4.4.2.8 Shutdown computer**

A diagram of a software project

Description automatically generated

#### **4.4.2.9 Check status of computer**

A diagram of a computer system

Description automatically generated

#### **4.4.2.10 Check specification of computer**

A diagram of a diagram

Description automatically generated

#### **4.4.2.11 View items (food, drink, game)**

A diagram of a system

Description automatically generated

## **4.5 Class Diagram**

Our restaurant management software's class diagram outlines the system's foundational structure, detailing the various classes and their interrelationships that make up the system. Each class in the diagram is an abstraction of services and entities provided by the software, crafted to manage different aspects of cyber operations effectively.

A diagram of a computer

Description automatically generated

## **4.6 Entity Relationship Diagram**

A diagram of a computer program

Description automatically generated

# **CHAPTER 5: NON-FUNCTIONAL REQUIREMENT OF CYBER GAMING**

## **5.1 Usability**

**User-Friendly Interface**: Ensure the user interface is user-friendly and easy to use.

**Learnability:** New users should be able to understand and use the primary functions of the system within three attempts.

**Error Prevention:** The system should prevent data entry errors by validating inputs in real-time and providing descriptive error messages.

## **5.2 Scalability**

**Anticipate Growth:** Estimate future growth in user base and data volume to ensure the system can scale accordingly.

**Flexible Infrastructure:** Use cloud-based services that offer flexible infrastructure options for easy scaling.

**Efficient Resource Utilization:** Optimize resource usage to ensure scalability does not lead to unnecessary costs.

**Testing for Scalability:** Conduct load testing and stress testing to validate that the system can scale as required under various conditions.

## **5.3 Maintainability**

**Version Control:** Utilize a version control system with clear branching and merging strategies to manage code changes effectively.

**Comprehensive Documentation:** Maintain detailed documentation for all modules, APIs, and system components to facilitate understanding and maintenance.

**Automated Testing:** Implement automated unit, integration, and system tests to ensure that changes do not introduce regressions.

**Code Modularity:** The codebase should be modular, allowing individual components to be updated or replaced without affecting other parts of the system.

## **5.4 Performance**

**Response Time:** The system must provide a response to user actions within 2 seconds for 95% of all interactions.

**Throughput:** The system should be able to handle at least 1,000 transactions per second without any noticeable degradation in performance.

**Scalability of Performance:** Performance must remain consistent under increased load, ensuring the system can scale to handle 10,000 concurrent users without significant performance loss.

**Latency:** Critical operations, such as database queries, should not exceed 100 milliseconds in latency.

**Load Time:** Web pages should load in under 3 seconds for users with standard broadband internet connections.

## **5.5 Compatibility**

**Cross-Browser Compatibility:** The system must function consistently across all major web browsers, including Chrome, Firefox, Safari, and Edge.

**API Compatibility:** The system should be compatible with existing RESTful or SOAP APIs to ensure seamless integration with external services.

**File Format Compatibility:** The system must support commonly used file formats, including CSV, XML, JSON, and PDF for data import and export.

## **5.6 Reliability**

**Regular Testing:** Perform reliability testing to validate system robustness under different failure scenarios and load conditions.

**Error Handling and Logging:** Implement comprehensive error-handling mechanisms and logging to ensure timely identification and resolution of issues.

**High Availability Setup:** Design the system with high availability in mind, using failover strategies, load balancing, and disaster recovery plans.

**Continuous Monitoring:** Use advanced monitoring tools to track system performance, detect anomalies, and trigger alerts for issues that may affect reliability.

## **5.7 Security**

**Risk Assessment:** Conduct regular risk assessments to identify and address potential security threats before they can be exploited.

**Security Updates:** Ensure the system is regularly updated with the latest security patches to prevent exploitation of known vulnerabilities.

**Incident Response Plan:** Develop and maintain an incident response plan to quickly identify, respond to, and recover from security breaches or attacks.

## **5.8 Availability**

**Failover Mechanism:** The system must have an automatic failover mechanism in place to switch to backup systems without manual intervention if a primary system fails.

**Uptime:** The system must have an uptime of 99.9% or higher, ensuring the service is available to users with minimal downtime.

**Disaster Recovery:** The system must include a disaster recovery plan that ensures full system restoration within 4 hours in case of a major failure or disaster.

# **CHAPTER 6: TEST CASE**

## **6.1 Login**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Test Scenario Description** | **Test Case Description** | **Pre-condition** | **Steps** | **Post-condition** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC01 | Verify successful login with valid credentials | User enters valid username and password | User has a registered account with valid credentials | 1. Open the web  2. Navigate to the login screen  3. Enter valid username and password  4. Click on the button login | User is logged into the web | The web should display the home screen | User is successfully logged in | Pass |
| TC02 | Verify login failure with in valid credentials | User enters invalid username and/or password | User has an account with invalid credentials | 1. Open the web  2. Navigate to the login screen  3. Enter valid username and/or password  4. Click on the button login | User should not be logged in | The web should display an error message indicating invalid credentials | The web should display an error message | Pass |
| TC03 | Verify login failure with blank fields | User leaves user and password fields blank | User has an account with valid credentials | 1. Open the web  2. Navigate to the login screen  3. Leave both username and password fields blank  4. Click on the button login | User should not be logged in | The web should display an error message indicating required fields | The web should display an error message | Pass |
| TC04 | Verify login failure with incorrect username | User enters a non-existent username | User does not have an account with the given username | 1. Open the web  2. Navigate to the login screen  3. Leave both username and password fields blank  4. Click on the button login | User should not be logged in | The web should display an error message indicating invalid username | The web should display an error message | Pass |
| TC05 | Verify login failure with incorrect password | User enters a valid username and an incorrect password | User does not have an account with the given username | 1. Open the web  2. Navigate to the login screen  3. Enter a non-existent username  4. Enter a valid password  5. Click on the button login | User should not be logged in | The web should display an error message indicating invalid password | The error massage is as expected | Pass |

## **6.2 Order items**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case Scenario ID** | **Test Scenario Description** | **Test Case Description** | **Pre-condition** | **Steps** | **Post-condition** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC06 | Verify successful placement of items order | User selects a menu item and completes the order process | User id logged into the web and has access to the menu | 1. Open the web  2. Navigate to the menu section  3. Select a food item  4. Specify any customizations (if weblicable)  5. Add the item to the cart | The order is placed successfully | The web display a confirmation message, and the order is reflected in the cyber’s order management system. | The order is successfully placed, and the confirmation message is displayed | Pass |
| TC07 | Verify that a user cannot proceed to payment with an empty cart | Attempt to proceed to payment with no items in the cart | User is logged in and no items are in the cart | 1. Open the web  2. Navigate to the food/drink menu section  3. Attempt to proceed to payment without selecting any items | System prevents user from proceeding | The system displays an error message, "Cart is empty. Please add items before proceeding." | Error message is displayed as expected | Pass |
| TC08 | Verify that a user can modify an order before making a payment | User updates the quantity or customizations of items in the cart | User has added items to the cart but not yet proceeded to payment | 1. Open the "Cart" section  2. Modify the quantity of an item  3. Update any customizations  4. Save changes | The cart is updated with the modifications | The system reflects the updated cart details | Modifications are saved successfully | Pass |
| TC09 | Verify that the order history reflects the newly placed order | Check the order in the "Order History" section | User has successfully placed an order | 1. Place a food order as described in Test Case 1  2. Navigate to the "Order History" section | The new order is recorded in the history | The system displays the new order in the "Order History" section with accurate details | The order in the "Order History" section with correct details | Pass |
| TC10 | Verify that a user can cancel an order before it is prepared | User cancels an order from the "Order History" or "Current Orders" section | User has placed an order and it has not yet been marked as prepared | 1. Navigate to the "Order History" or "Current Orders" section  3. Select an active order  4. Click "Cancel Order"  5. Confirm the cancellation | The order is canceled successfully | The system displays a cancellation confirmation message and updates the order status | The order is successfully canceled, and the status is updated | Pass |

## **6.3 Update item (game, food, drink, card) menu**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case Scenario ID** | **Test Scenario Description** | **Test Case Description** | **Pre-condition** | **Steps** | **Post-condition** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC11 | Verify that an admin can add a new item to the menu | Admin adds a new item with valid details | Admin is logged in with the necessary permissions | 1. Navigate to the "Menu Management" section  2. Click on "Add New Item"  3. Fill in the food item details (e.g., name, price, category)  4. Upload a food item image (if required)  5. Save the new item | The new item is added to the menu | The system displays a success message, and the new item webears in the menu list | New item successfully added to the menu | Pass |
| TC12 | Verify that an admin can edit the details of an existing item | Admin updates the price or details of an existing menu item | Admin is logged in, and at least one food item exists in the menu | 1. Navigate to the "Menu Management" section  2. Select an existing food item  3. Edit the details (e.g., update the price or change the name)  4. Save the changes | The changes are reflected in the menu | The updated item details are saved and displayed in the menu | Food details successfully updated | Pass |
| TC13 | Verify that an admin can delete a item from the menu | Admin removes an existing item from the menu | Admin is logged in, and at least one food item exists in the menu | Navigate to the "Menu Management" section  Select a food item to delete  Click on the "Delete" button  Confirm the deletion | The cart is updated with the modifications | The system reflects the updated cart details | Modifications are saved successfully | Pass |
| TC014 | Verify that the order history reflects the newly placed order | Admin attempts to save a new item without filling in mandatory fields (e.g., name or price) | Admin is logged in | 1. Navigate to the "Menu Management" section  2. Click on "Add New Item"  3. Leave mandatory fields (e.g., name or price) blank  4. Attempt to save the item | The system does not save the item and displays an error message | The system displays a validation error message, such as "Name is required" or "Price is required" | Validation error message displayed as expected | Pass |
| TC15 | Verify that an admin can upload an image for a item | Admin uploads a valid image for a food item in the menu | Admin is logged in, and the image file meets the system's requirements | 1. Navigate to the "Menu Management" section  2. Select an existing item or create a new one  3. Click "Upload Image" and select a valid image file  4. Save the changes | The item is updated with the uploaded image | The system displays the uploaded image alongside the item | Image uploaded and displayed successfully | Pass |

## **6.4 Update menu game**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case Scenario ID** | **Test Scenario Description** | **Test Case Description** | **Pre-condition** | **Steps** | **Post-condition** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC16 | Verify that an admin can add a new game to the menu | Admin adds a new game with valid details | Admin is logged in with the necessary permissions | 1. Navigate to the "Game Management" section  2. Click on "Add New Game"  3. Enter the game details (e.g., name, genre, description)  3. Save the new game | The new game is added to the menu | The system displays a success message, and the new game webears in the game menu | The game is successfully added to the menu | Pass |
| TC17 | Verify that an admin can edit the details of an existing game | Admin edits the details of an existing game in the menu | Admin is logged in, and at least one game exists in the menu | 1. Navigate to the "Menu Management" section  2. Select an existing food item  3. Edit the details (e.g., update the price or change the name)  4. Save the changes | The updated game details are saved and displayed in the menu | The system displays the updated game details | Game details successfully updated in the menu | Pass |
| TC18 | Verify that an admin can remove a game from the menu | Admin removes a game from the menu | Admin is logged in, and at least one game exists in the menu | 1. Navigate to the "Game Management" section  2. Select a game to delete  3. Click on the "Delete" button  4. Confirm the deletion | The game is removed from the menu | The system removes the game and displays a success message | Game successfully deleted from the menu | Pass |
| TC19 | Verify that the system prompts the admin to fill in mandatory fields when adding a new game | Verify that the system prompts the admin to fill in mandatory fields when adding a new game | Admin is logged in | 1. Navigate to the "Game Management" section  2. Click on "Add New Game"  3. Leave mandatory fields (e.g., name or genre) blank  4.Attempt to save the game | The system does not save the game and displays an error message | The system displays a validation error message, such as "Name is required" | Validation error message displayed as expected | Pass |
| TC20 | Verify that an admin can upload a cover image for a game | Admin uploads a valid image for a game in the menu | Admin is logged in, and the image file meets the system's requirements | 1. Navigate to the "Game Management" section  2. Select an existing game or create a new one  3. Click "Upload Image" and select a valid image file  4. Save the changes | The game is updated with the uploaded image | The system displays the uploaded image alongside the game details | Image uploaded and displayed successfully | Pass |

## **6.5 Test case with account**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case Scenario ID** | **Test Scenario Description** | **Test Case Description** | **Pre-condition** | **Steps** | **Post-condition** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC21 | Verify that an admin can add a new user account with valid details | Admin adds a new user account with required information | Admin is logged in with necessary permissions | 1. Navigate to the "User Management" section  2. Click on "Add New Account"  3. Fill in the required user details  4. Set a default password  5. Save the new account | The new user account is added successfully | The system displays a success message, and the new user account webears in the user list | New user account successfully added | Pass |
| TC22 | Verify that the system enforces mandatory fields when adding a new user account | Admin attempts to save a new user account without filling in mandatory fields | Admin is logged in | 1. Navigate to the "User Management" section  2. Click on "Add New Account"  3. Leave mandatory fields blank  4. Attempt to save the account | The system does not save the account and displays an error message | The system displays a validation error message, such as "Username is required" | Validation error message displayed as expected | Pass |
| TC23 | Verify that the system prevents the addition of duplicate user accounts with the same username or email | Admin attempts to create a user account with a username or email already in use | Admin is logged in, and a user with the same username or email already exists | 1. Navigate to the "User Management" section  2. Click on "Add New Account"  3. Enter a username or email already in use  4. Attempt to save the account | The system prevents duplicate accounts from being created | The system displays an error message, such as “Username already exists" | Error message displayed as expected | Pass |
| TC24 | Verify that an admin can assign a specific role when adding a user account | Admin selects a role for the new user account during creation | Admin is logged in | 1. Navigate to the "User Management" section  2. Click on "Add New Account"  3. Fill in the user details  4. Select a role from the dropdown menu (e.g., gamer, staff)  5. Save the account | The user account is created with the assigned role | The new account is displayed in the user list with the correct role assigned | User account created with the correct role | Pass |
| TC25 | Verify that the system enforces password policy (e.g., minimum length, special characters) when creating a new user account | Admin sets a password for the new user account that does not meet the system's password policy | Admin is logged in | 1. Navigate to the "User Management" section  2. Click on "Add New Account"  3. Fill in the user details.  4. Enter a password that does not meet the policy requirements  5. Attempt to save the account | The system rejects the account creation and displays a password policy error | The system displays an error message, such as "Password must be at least 8 characters long and contain a special character” | Password policy error message displayed as expected | Pass |

## **6.6 Payment**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test case Scenario ID** | **Test Scenario Description** | **Test Case Description** | **Pre-condition** | **Steps** | **Post-condition** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| TC26 | Verify that a user can successfully make a payment | User completes a payment using a valid payment method | User has an active session or order requiring payment | 1. Navigate to the "Payment" section  2. Select a valid payment method  3. Enter the required payment details  4. Confirm the payment | Payment is processed successfully | The system displays a success message and updates the order/payment status to "Paid" | Payment successfully completed, and confirmation message displayed | Pass |
| TC27 | Verify that the system rejects payment if the amount in user account is insufficient | User attempts payment with amount less than required | User has a pending payment | 1. Navigate to the "Payment" section  2. Select the payment method  3. Enter invalid details  4. Attempt to complete the payment | Payment is rejected | The system displays an error message, such as "Invalid details. Please try again" | Payment failed as expected with an error message | Pass |
| TC028 | Verify that the payment history reflects a successful transaction | Check that the payment webears in the "Payment History" section after completion | User has successfully completed a payment | 1. Complete a payment as described in Test Case 1  2. Navigate to the "Payment History" section | The payment is recorded in the history | The system lists the payment in the history with correct details | Payment webears in the "Payment History" section with accurate details | Pass |
| TC29 | Verify that the admin can successfully cancel a pending payment | Admin cancels a pending payment for a user | A pending payment exists in the system | 1. Navigate to the 'Payment Management' section  2. Locate the pending payment  3. Select 'Cancel Payment'  4. Confirm the cancellation | The payment is cancelled successfully | The system displays a confirmation message: 'Payment successfully cancelled' and updates the payment status to 'Cancelled' | Payment successfully cancelled, and status updated to 'Cancelled' | Pass |

# **CHAPTER 7: CONCLUSION**

## **7.1 Result**

**Gay Station Cyber Gaming System** is a game management software with an aesthetically pleasing and user-friendly interface. With this software, players can fully enjoy their leisure time without encountering any obstacles, while the business owner can easily manage the entire operation of the cybercafe with just a few simple actions. By combining multiple architectures, modern technologies, and a well-planned process, this software is not only easy to use in the present but also offers great potential for future development at an affordable cost.

## **7.2 Evaluation**

### **7.2.1 Advantage**

* Friendly interface for both admin and user, keep up with the trend so that users can get excited
* Admin easily manage the system without much knowledge about technology,
* Access rights are clearly defined, allowing players to have significant control over their accounts while ensuring that their actions do not affect the overall system.
* The system for storing food and beverages is organized clearly, making it easy to update with new items that align with current trends among young people.
* The payment system is diverse, allowing for multiple payment methods. The business owner can track the cybercafe's revenue using statistical algorithms, with access restricted exclusively to the admin for viewing.

### **7.2.2 Disadvantage**

* Some basic features are still missing, which, though minor, could impact the overall user experience.
* The user information section lacks adequate security measures, making it vulnerable to potential data breaches.
* Some functions are still manual, and with the rapidly changing trends, certain areas may quickly become outdated, making updates complex and challenging.