

**ACCOUNT GAME ‘S STORE**

***Group 4***

**CE140386 Nguyen Son Hao**

CE140311 Nguyen Tran Quang Hien

CE130302 Thach Qui

CE130347 Pham Nhat Cuong

**Lecturer**

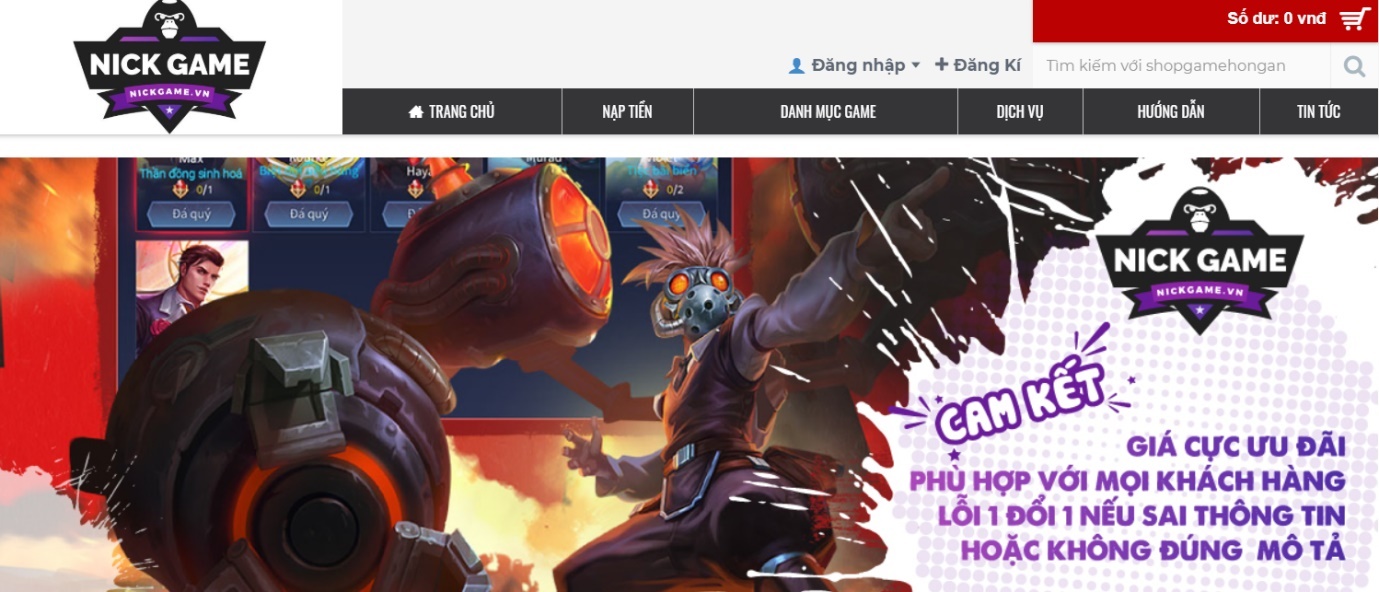
Huong Hoang Luong

# **Chapter 1: Introduction**

1.1 Problem definition

Today esports are becoming more and more popular. Helps reduce stress after working and studying hours.

Besides, the trading of game accounts between gamers is indispensable. So we decided to create a website that sells game accounts, so that gamers can conveniently buy and sell accounts, quickly and securely. The types of accounts our website sells include, League of legends, Arena of valor and fifa online 4. Because these games are very popular and are played by many people.



* + 1. *Demo Account game store*

1.2 Customer Requirement Specification

**Functions**

1. **For manager by role**

* Login, logout
* Change infomation account
* Manage account buy: search account buy, looking for bad seller account buy, looking for best seller account buy.
* Manage staff: search staff, producttivity charts are employee jobs.

1. **For admin by role**

* Login, logout
* Manager account: customer, create account, change info, delete account, change pass, create status.
* Manager type of product: create type, edit type, delete type, create status.
* Manager account buy: create account buy, edit account buy, delete account buy, create status
* Manager account admin: create account, delete account, change info, change pass

1. **For employee by role**

* Register: input information, input password
* Login, logout
* Employee: view information, change information, save
* Search
* Search product: price product, information product, color
* Search infor customer: name, number phone, address, identity card
* Create bill
* Information customer: export bill
* Information product: export bill
* Date sell: export bill
* Code bill: export bill
* Information employee: export bill

1. **For customer by role**

* Login, logout
* View account infor: change account info, change password
* View product
* Choose account buy
* Buy product: choose payment type, choose transpot type, input name and address, bill
* Search products: Name account, price account.

1.3 Hardware and Software Requirement

**Hardware**

* ***For server***

|  |  |
| --- | --- |
| **Hardware requirement**  **(minimal configuration)** | **Hardware requirement**  **(recommended configuration)** |
| * CPU: 500 MHz processor * RAM: 2GB RAM * HDD: 500MB of free disk space | * CPU: 4GHZ Dual Core or 3GHZ processor * RAM: 4GB RAM * HDD: 1GB of free disk space |

* + - ***For client***

|  |  |
| --- | --- |
| **Hardware requirement**  **(minimal configuration)** | **Hardware requirement**  **(recommended configuration)** |
| * CPU: 500 MHz processor * RAM: 2GB RAM * HDD: 500MB of free disk space | * CPU: 4GHZ Dual Core or 3GHZ processor * RAM: 4GB RAM * HDD: 1GB of free disk space |

**Software**

* + - ***For server***

|  |
| --- |
| **Software requirement** |
| * Window 7 or higher * JRE 1.7 or higher |

* + - ***For client***

|  |
| --- |
| **Software requirement** |
| * Window 7 or higher * JRE 1.7 or higher |

# **Chapter 2: Theory**

2.1 Introduction to JSP

* It stands for **Java Server Pages**.
* It is a server side technology.
* It is used for creating web application.
* It is used to create dynamic web content.
* In this JSP tags are used to insert JAVA code into HTML pages.
* It is an advanced version of Servlet Technology.
* It is a Web based technology helps us to create dynamic and platform independent web pages.
* In this, Java code can be inserted in HTML/ XML pages or both.
* JSP is first converted into servlet by JSP container before processing the client’s request.

2.2 Introduction to MVC in JSP

* **MVC** stands for Model View and Controller. It is a **design pattern** that separates the business logic, presentation logic and data.
* **Controller** acts as an interface between View and Model. Controller intercepts all the incoming requests.
* **Model** represents the state of the application i.e. data. It can also have business logic.
* **View** represents the presentation i.e. UI (User Interface).

2.3 Introduction to Ajax

* AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
* AJAX is not a programming language.
* AJAX just uses a combination of:
* A browser built-in XMLHttpRequest object (to request data from a web server)
* JavaScript and HTML DOM (to display or use the data)

2.4 Introduction to JavaScript

* JavaScript was initially created to “make web pages alive”.
* The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads.
* Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

In this aspect, JavaScript is very different from another language called [Java](https://en.wikipedia.org/wiki/Java_(programming_language)).

2.5 Introduction Spring

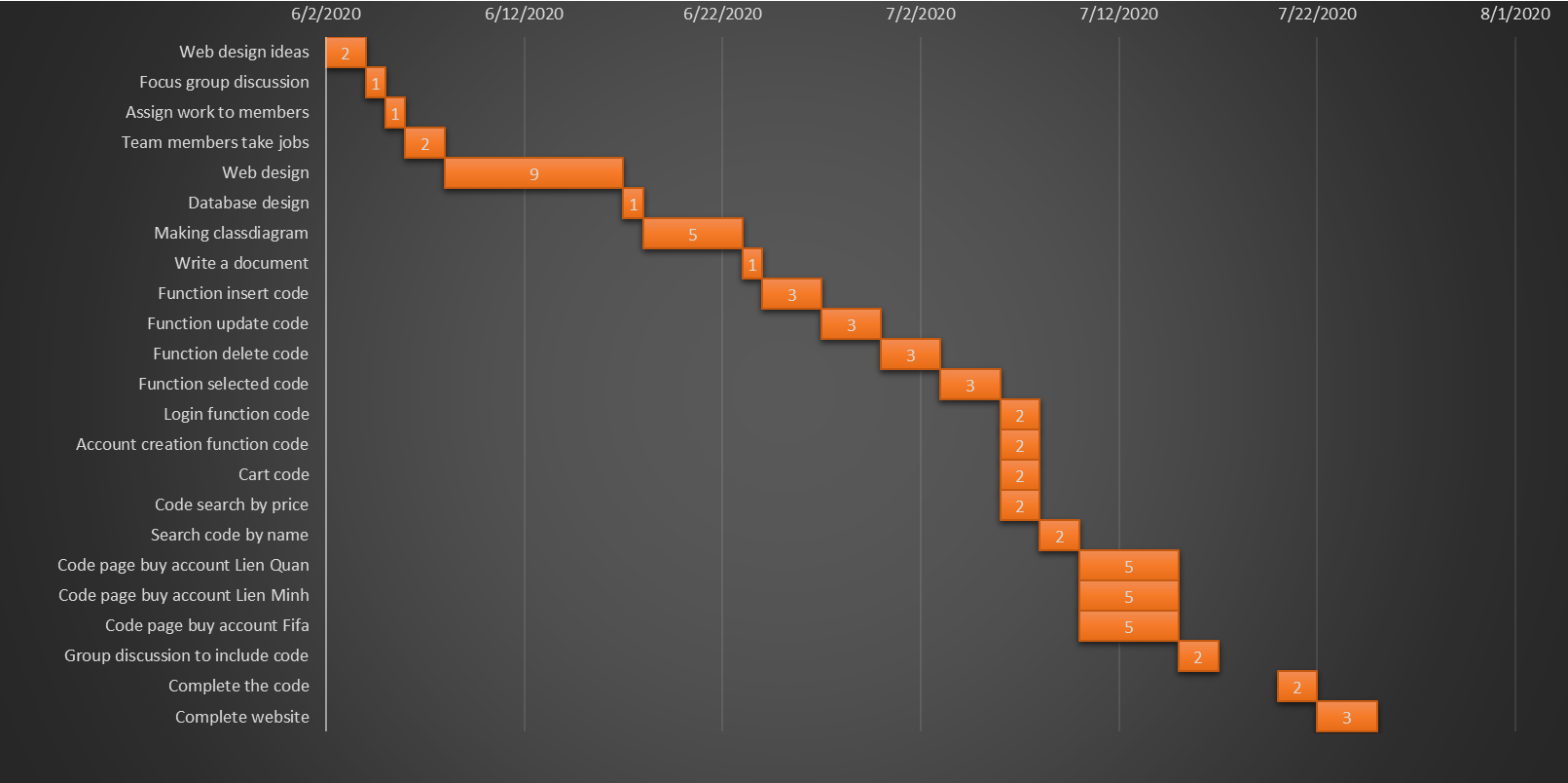
# **Schedule and role**

1. **Role**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Name of member** | **Role** | **Responsibility** |
| 1 | HaoNS | Leader | Assignment of tasks |
| 2 | CuongPN, HaoNS | Analyst | Define problem, input, output, process |
| 3 | HienNTQ, QuiT,CuongPN | Design | Design interface |
| 4 | HaoNS, HienNTQ, QuiT,CuongPN | Coder | Program all functions for project |
| 5 | HaoNS, HienNTQ, QuiT,CuongPN | Tester | Testing all functions for project |
| 6 | HaoNS, HienNTQ, QuiT,CuongPN | Maintenance | Check and backup data |

1. **Schedule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task No** | **Task Description** | **Expected Completion Date** | **Expected Time Needed(hrs)** | **Members in charge** |
| **1** | Web design ideas | 02/06/2020 | 1 | HaoNS, HienNTQ, CuongPN, QuiT |
| **2** | Focus group discussion | 04/06/2020 | 1 | HaoNS, HienNTQ, CuongPN, QuiT |
| **3** | Assign work to members | 05/06/2020 | 1 | HaoNS |
| **4** | Team members take jobs | 06/06/2020 | 1 | HaoNS, HienNTQ, CuongPN, QuiT |
| **5** | Web design | 08/06/2020 | 168 | HaoNS, HienNTQ, CuongPN, QuiT |
| **6** | Database design | 17/06/2020 | 3 | HaoNS, HienNTQ, CuongPN, QuiT |
| **7** | Making classdiagram | 18/06/2020 | 4 | HienNTQ |
| **8** | Write a document | 23/06/2020 | 4 | HaoNS, HienNTQ, CuongPN, QuiT |
| **9** | Function insert account buy code | 24/06/2020 | 3 | HaoNS |
| **10** | Function insert user code | 24/06/2020 | 3 | HienNTQ |
| **11** | Function insert bill code | 24/06/2020 | 3 | CuongPN |
| **12** | Function update account buy code | 27/06/2020 | 3 | QuiT |
| **13** | Function update user code | 27/06/2020 | 3 | HaoNS |
| **14** | Function update bill buy code | 27/06/2020 | 3 | HienNTQ |
| **15** | Function delete account buy code | 30/06/2020 | 3 | CuongPN |
| **16** | Function delete user buy code | 30/06/2020 | 3 | QuiT |
| **17** | Function delete bill buy code | 30/06/2020 | 3 | HaoNS |
| **18** | Function selected account buy code | 03/07/2020 | 3 | HienNTQ |
| **19** | Function selected user code | 03/07/2020 | 3 | QuiT |
| **20** | Function selected bill code | 03/07/2020 |  | HaoNS |
| **20** | Login function code | 06/07/2020 | 3 | CuongPN |
| **21** | Account creation function code | 06/07/2020 | 3 | CuongPN |
| **22** | Cart code | 06/07/2020 | 3 | HaoNS |
| **23** | Code search by price | 06/07/2020 | 3 | QuiT |
| **24** | Search code by name | 08/07/2020 | 3 | HienNTQ |
| **25** | Code page buy account Lien Quan | 10/07/2020 | 3 | HaoNS |
| **26** | Code page buy account Lien Minh | 10/07/2020 | 3 | HienNTQ |
| **27** | Code page buy account Fifa | 10/07/2020 | 3 | QuiT |
| **28** | Group discussion to include code | 15/07/2020 | 5 | HaoNS, HienNTQ, CuongPN, QuiT |
| **29** | Testing and fixcode | 17/07/2020 | 4 | HaoNS, HienNTQ, CuongPN, QuiT |
| **30** | Complete the code | 20/07/2020 | 5 | HaoNS, HienNTQ, CuongPN, QuiT |
| **31** | Complete website | 22/07/2020 | 5 | HaoNS, HienNTQ, CuongPN, QuiT |
| **32** | Deadline | 25/07/2020 | 1 |  |

1. **Diagram Gantt**
2. **Meeting schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **Meeting schedule** | | | |
| **Date** | **Content** | **Form** | **Time** |
| **22/06/2020** | Discuss the function of insert, update account buy, user, bill | Offline | 19:00 H |
| **25/06/2020** | Discuss delete and select product, user, bill, progress report | Offline | 19:00 H |
| **06/07/2020** | Discuss login, create account, cart, search by price, search by name. | Offline | 19:00 H |
| **09/07/2020** | Discuss created some page by account. | Offline | 19:00 H |
| **14/07/2020** | Discuss interface design, fix bugs. | Offline | 19:00 H |
| **17/07/2020** | Edit document and do PowerPoint | Offline | 19:00 H |

1. **Link GitHub**

<https://github.com/nguyensonhao/Prj321_SE1403_Group4_WebsiteBanAccountGame>

# **Design pattern**

1. **System Organization**
2. **Use-case diagram**
3. **Class diagram**
4. **Sequence diagram**
5. **Interface**

**DB Diagram**

**Table structure**

1. **Interface**

**For Admin**

**For Client**

# **Execution flow**

# **Maintenance**

Everyday, we always upload all data in Google Drive and send that our program to each member in group. By this way, we don’t worry that our program will be destroyed or lost and we are sure each member in group can be understand program.

****

# **Check list**