

**ACCOUNT GAME ‘S STORE**

***Group 4***

**CE140386 Nguyen Son Hao**

CE140311 Nguyen Tran Quang Hien

CE130302 Thach Qui

CE130347 Pham Nhat Cuong

**Lecturer**

Luong Hoang Huong

Table content

[**I.** **Chapter 1: Introduction** 3](#_Toc46519358)

[**II.** **Schedule and role, Gantt diagram, Meeting and link Github** 10](#_Toc46519361)

[**III.** **Chapter 2: Theory** 16](#_Toc46519362)

[**a.** **What is JSP?** 16](#_Toc46519363)

[**b.** **Advantages of JSP over Servlet** 16](#_Toc46519364)

[**c.** **The Lifecycle of a JSP Page** 17](#_Toc46519366)

[**a.** **What is MVC?** 18](#_Toc46519370)

[**b.** **The advantages of MVC** 19](#_Toc46519371)

[**IV.** **Chapter 3: Architecture System** 21](#_Toc46519376)

[**1.** **Client-Server Architectures** 21](#_Toc46519377)

[**2.** **Use-case Diagram** 22](#_Toc46519378)

[**4.** **Entity relationship diagra** 25](#_Toc46519383)

[**5.** **Sequence diagram** 25](#_Toc46519383)

[**6.** **DFD** 28](#_Toc46519384)

[**V.** **Chapter 4: Functions, User Interfaces and Flow chart** 32](#_Toc46519394)

[**1.** **Functions** 32](#_Toc46519395)

[**2. Table structure** 32](#_Toc46519396)

[**Table 2.** **Account User** 33](#_Toc46519397)

[**Table 3.** **Product** 33](#_Toc46519398)

[**Table 4.** **Bill** 34](#_Toc46519399)

[**Table 5.** **Billdetail** 34](#_Toc46519400)

[**Table 6.** **Descriptiondetail** 34](#_Toc46519401)

[**Table 7.** **Role** 35](#_Toc46519402)

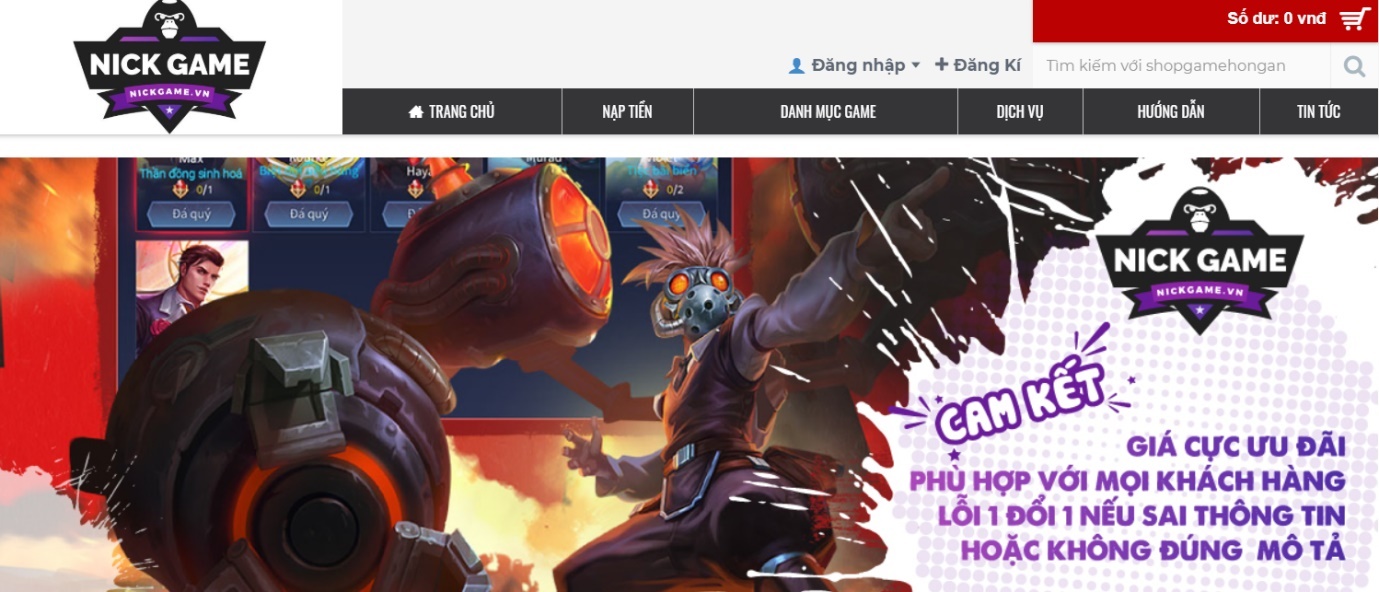
[**VI.** **Chapter 5: Conclusion** 40](#_Toc46519403)

# **Chapter 1: Introduction**

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. **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. **Hardware and Software Requirement**

**Hardware**

|  |
| --- |
| Minimum Hardware Configurations |
| Microsoft Windows Vista SP1/Windows 7 Professional:  Processor: 800MHz Intel Pentium III or equivalent  Memory: 512 MB  Disk space: 750 MB of free disk space  Ubuntu 9.10:  Processor: 800MHz Intel Pentium III or equivalent  Memory: 512 MB  Disk space: 650 MB of free disk space  Macintosh OS X 10.7 Intel:  Processor: Dual-Core Intel  Memory: 2 GB  Disk space: 650 MB of free disk space |

|  |
| --- |
| Recommended Hardware Configurations |
| Microsoft Windows 7 Professional/Windows 8/Windows 8.2:  Processor: Intel Core i5 or equivalent  Memory: 2 GB (32-bit), 4 GB (64-bit)  Disk space: 1.5 GB of free disk space  Ubuntu 15.04:  Processor: Intel Core i5 or equivalent  Memory: 2 GB (32-bit), 4 GB (64-bit)  Disk space: 1.5 GB of free disk space  OS X 10.10 Intel:  Processor: Dual-Core Intel  Memory: 4 GB  Disk space: 1.5 GB of free disk space |

**Required Software**

* NetBeans IDE runs on the Java SE Development Kit (JDK) which consists of the Java Runtime Environment and developer tools for compiling, debugging, and running applications written in the Java language.
* The tested JDK for this release is JDK 8u101 for Windows, Linux, and OS X. The 8.2 version of the IDE cannot be installed or run on the JDK older than JDK 8.
* Note:
* The PHP and C/C++ NetBeans bundles only require the Java Runtime Environment (JRE) 8 to be installed and run.
* Java features in the IDE and JavaFX 8 features require JDK 8.
* Download Tomcat for webserver, XAMPP for database.

# **Schedule and role, Gantt diagram, Meeting and link Github**

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 |

Figure 1: Role

1. **Schedule**

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

Figure 2: Schedule

1. **Diagram Gantt**

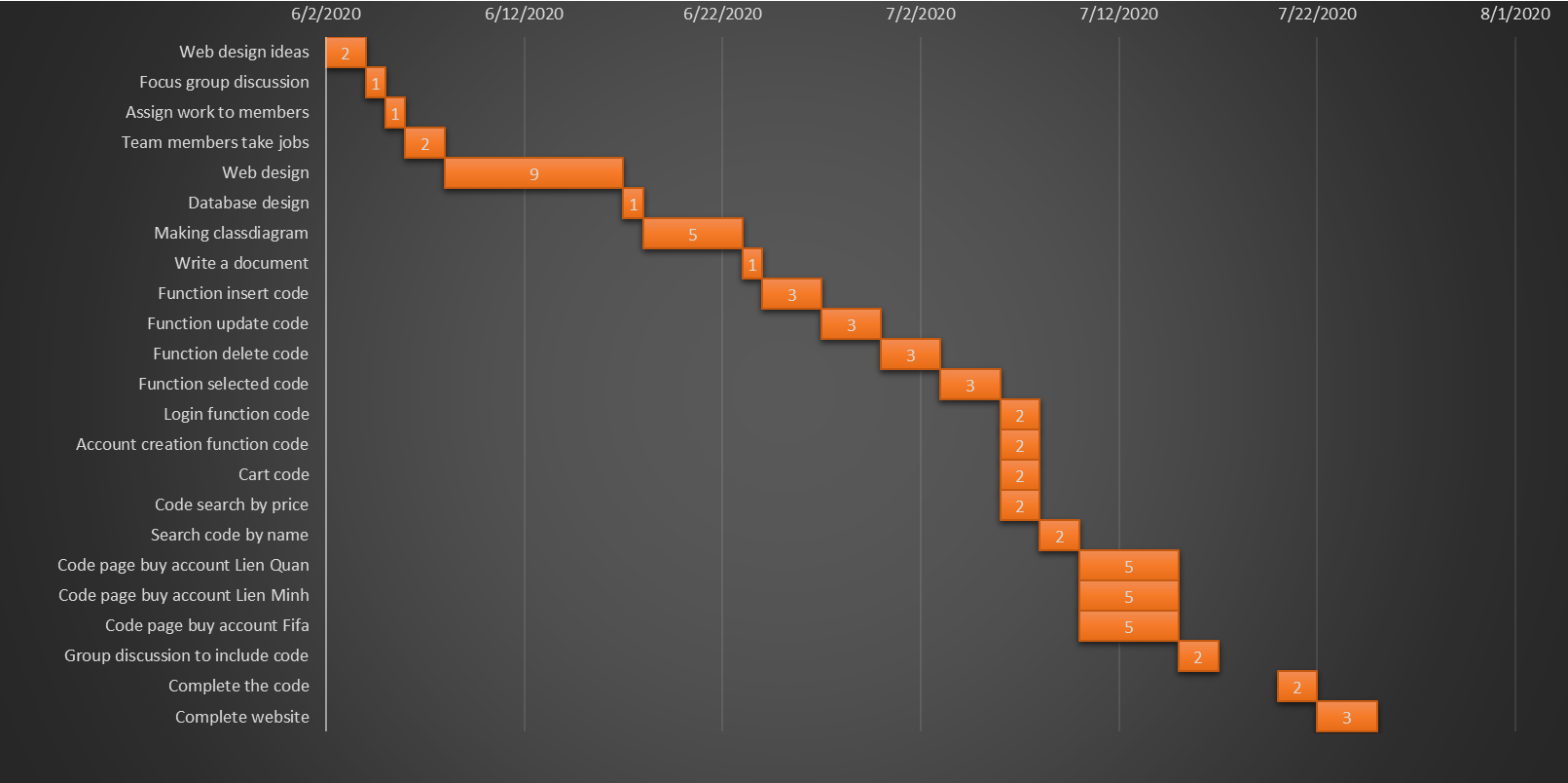


Figure 3: Gantt diagram

1. **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 |

Figure 4: Meeting schedule schedule

1. **Link GitHub**

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

# **Chapter 2: Theory**

1. Introduction to JSP

**What is 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.

**Advantages of JSP over Servlet**

* Extension to Servlet: JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy
* Easy to maintain: JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.
* Fast Development: No need to recompile and redeploy. If JSP

page is modified, we don't need to recompile and redeploy the

project. The Servlet code needs to be updated and recompiled have to change the look and feel of the application.

* Less code than Servlet: In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

**The Lifecycle of a JSP Page**

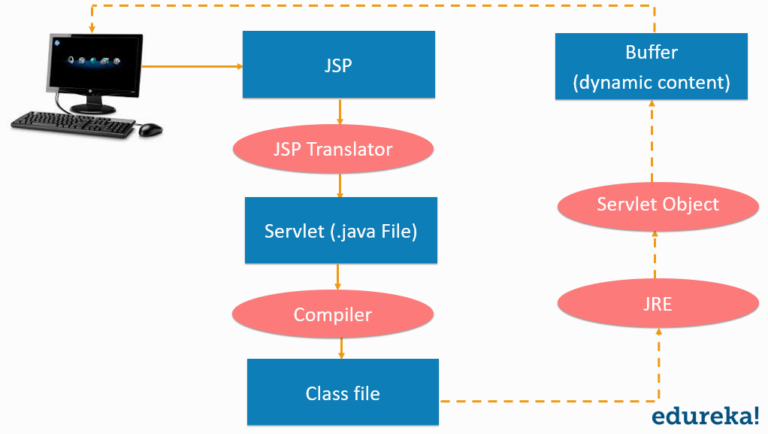
* Translation of JSP page
* Compilation JSP page
* Classloading
* Instantiation (Object of the Generated Servlet is created).
* Initialization (the container invokes jspInit() method).
* Request processing (the container invokes \_jspService() method).
* Destroy (the container invokes jspDestroy() method).

Figure 4: Introduction to JSP

1. Introduction to MVC in JSP

**What is MVC?**

* MVC is an architecture that separates business logic, presentation and data. In MVC, M stands for Model, V stands for View, C stands for controller.
* MVC is a systematic way to use the application where the flow starts from the view layer, where the request is raised and processed in controller layer and sent to model layer to insert data and get back the success or failure message.

**Model Layer:**

* This is the data layer which consists of the business logic of the system.
* It consists of all the data of the application
* It also represents the state of the application.
* It consists of classes which have the connection to the database.
* The controller connects with model and fetches the data and sends to the view layer.
* The model connects with the database as well and stores the data into a database which is connected to it.

**View Layer:**

* This is a presentation layer.
* It consists of HTML, JSP, etc. into it.
* It normally presents the UI of the application.
* It is used to display the data which is fetched from the controller which in turn fetching data from model layer classes.
* This view layer shows the data on UI of the application.

**Controller Layer:**

* It acts as an interface between View and Model.
* It intercepts all the requests which are coming from the view layer.
* It receives the requests from the view layer and processes the requests and does the necessary validation for the request.
* This requests is further sent to model layer for data processing, and once the request is processed, it sends back to the controller with required information and displayed according by the view.

**The advantages of MVC**

* Easy to maintain
* Easy to extend
* Easy to test
* Navigation control is centralized

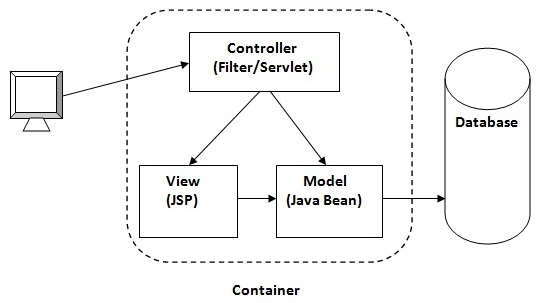


Figure 5: MVC Architecture

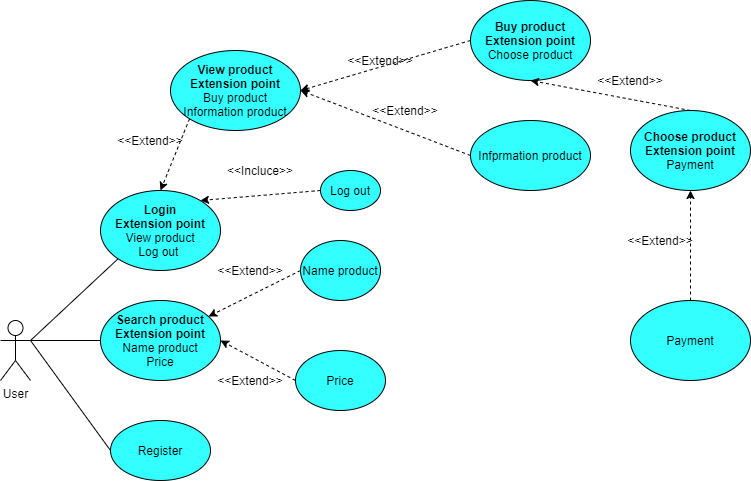
1. **Chapter 3: Architecture System**
2. **Client-Server Architectures**

A Client-Server Architecture consists of two types of components: clients and servers. A server component perpetually listens for requests from client components. When a request is received, the server processes the request, and then sends a response back to the client. Servers may be further classified as stateless or stateful. Clients of a stateful server may make composite requests that consist of multiple atomic requests. This enables more conversational or transactional interactions between client and server. To accomplish this, a stateful server keeps a record of the requests from each current client. This record is called a session.

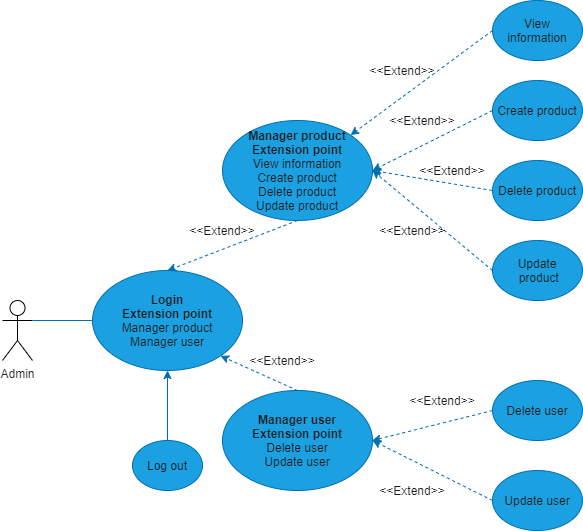
In order to simultaneously process requests from multiple clients, a server often uses the Master-Slave Pattern. In this case, the Master perpetually listens for client requests. When a request is received, the master creates a slave to processes the request and then resumes listening. Meanwhile, the slave performs all subsequent communication with the client.

Internally, the client component may consist of a ClientUI that forwards user requests to a controller component. The controller component forwards the request across a process or machine boundary to a RequestListener inside the server. The listener, which acts like a master, creates a RequestHandler slave and forwards the request to it:

1. **Use-case Diagram**

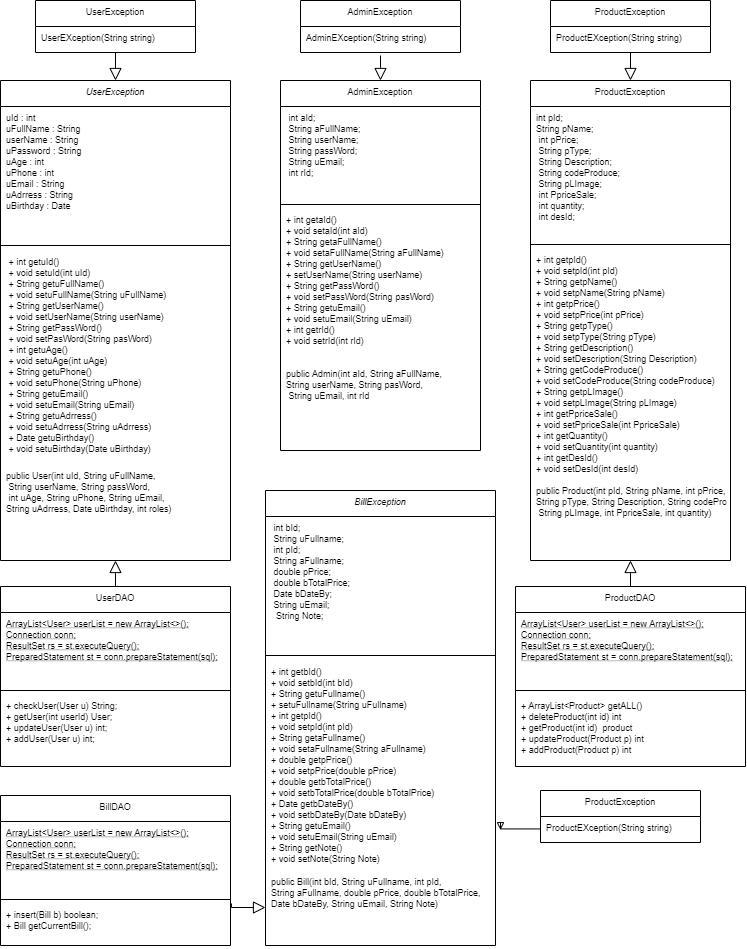
****

*Figure 6: Use case Shop account game of User*

****

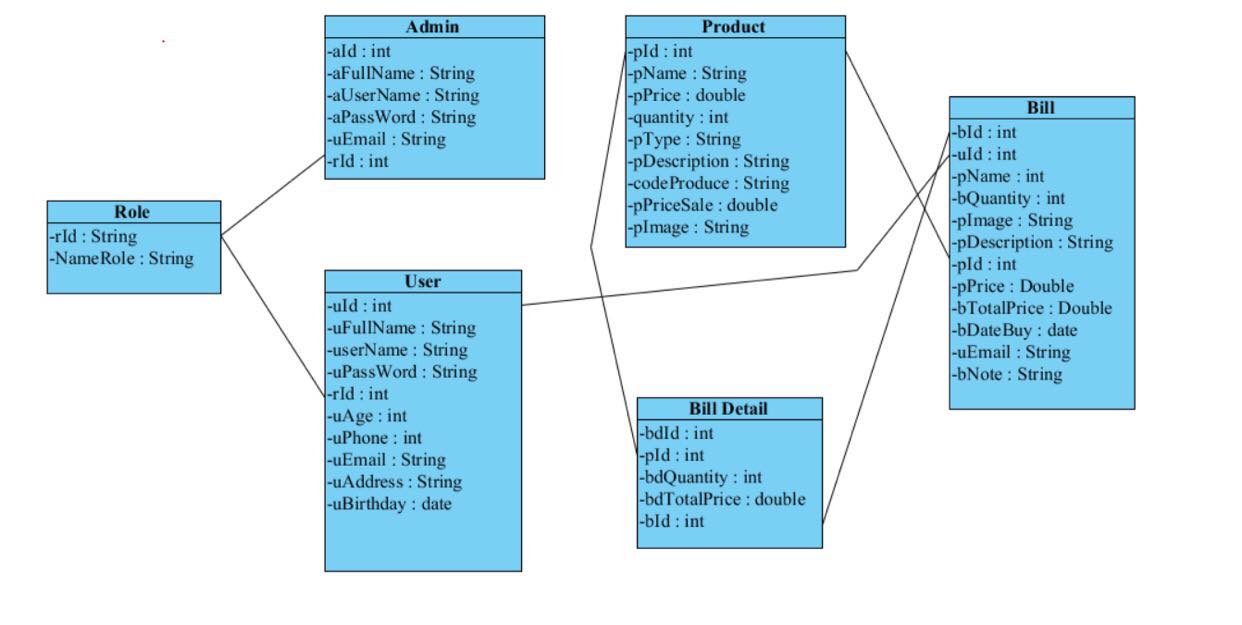
*Figure7: Use case Shop account game of Admin*

1. **Class diagram**



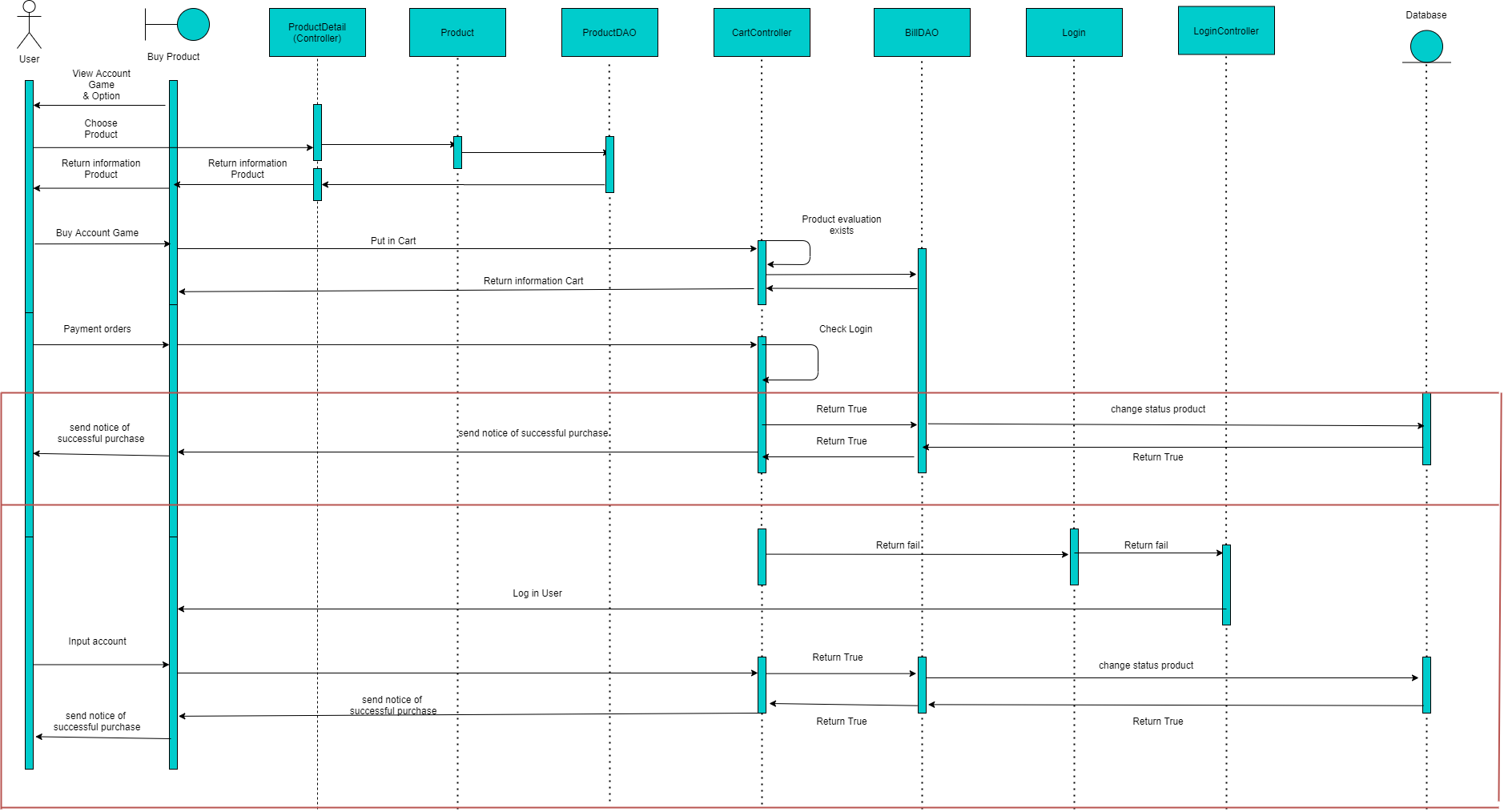
*Figure7: Class diagram*

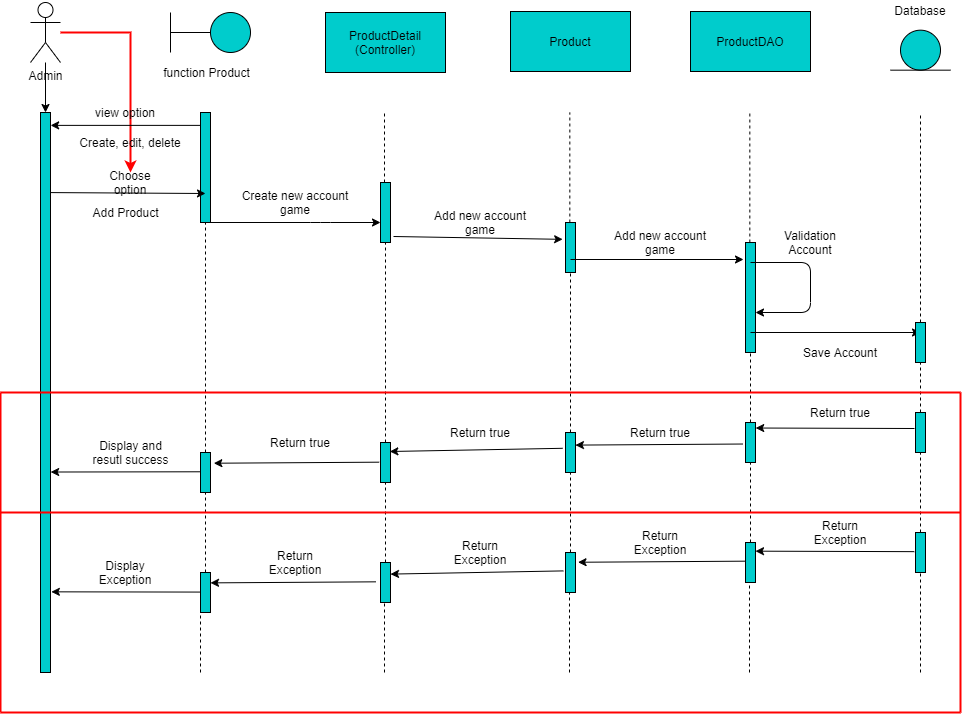
1. **Entity relationship diagram**

****

*Figure7: Database diagram of Shop account game*

1. **Sequence diagram**

****

****

*Figure 8: Sequence diagram*

1. **DFD**
2. **Data flow diagram symbol**

|  |  |
| --- | --- |
| Symbol | Description |
|  | **Data Flow:** Data flow are pipelines through the packets of information flow. |
|  | **Process:** A Process or task performed by the system. |
|  | **Entity:** Entity are object of the system. A source or destination data of a system. |
|  | **Data Store:** A place where data to be stored. |

Figure 9: Data flow diagram symbol

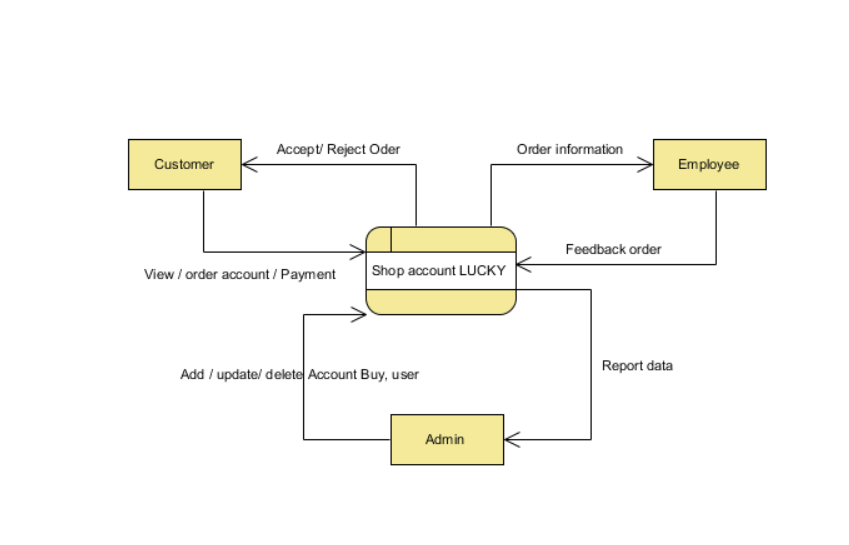
1. **Contextual Level 0**

Figure10: Contextual Level 0

1. **Level 1**

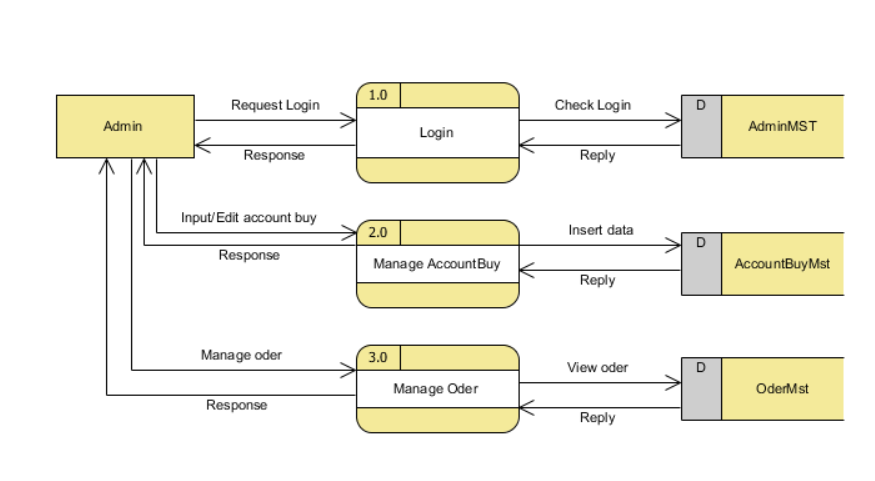


Figure 11: Admin Level 1

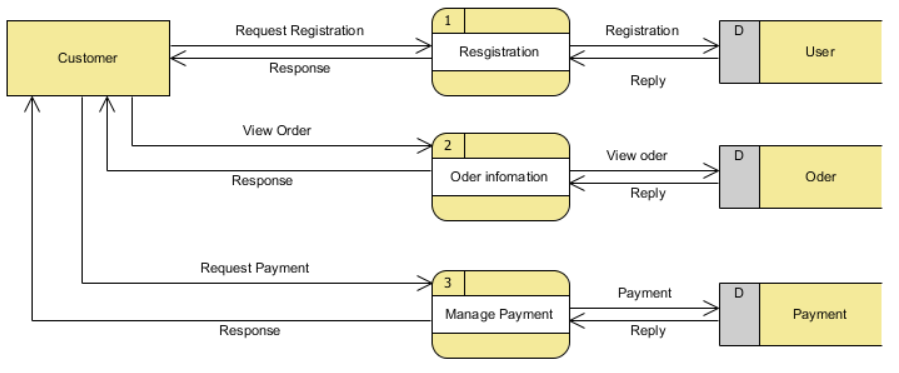
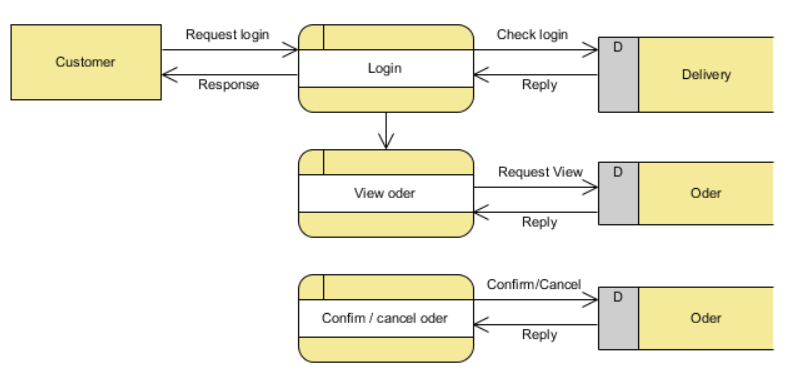
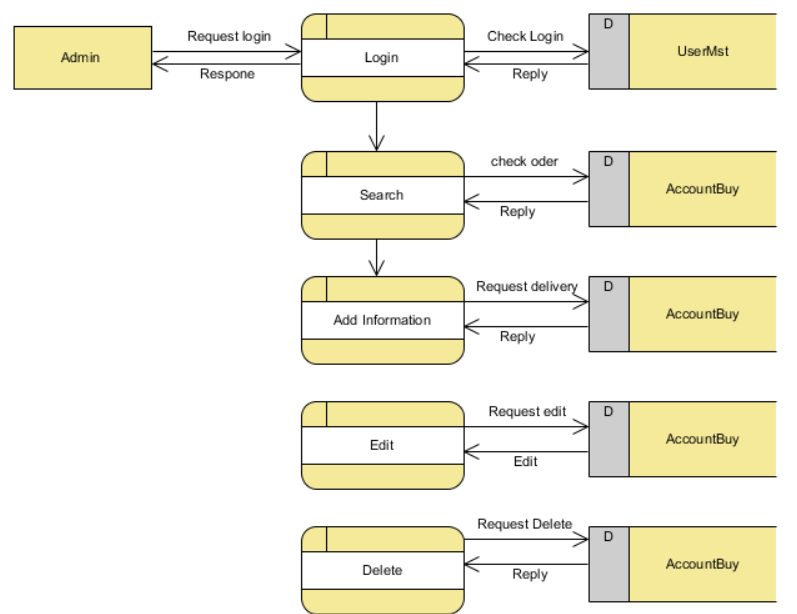


Figure 11: Customer Level 1

1. **Level 2**



*Figure 12: Customer level 2*



*Figure 12: Admin level 2*

1. **Chapter 4: Functions, User Interfaces and Flow chart**
2. **Functions**

* Login
* Register
* Logout
* Add account buy
* Update account buy
* Delete account buy
* Search account buy
* Add to cart
* Show account buy with the same price
* Buy product send mail.

1. **Table structure**

**Table 1. Account admin**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| aId | int | PRIMARY KEY | ID’s admin |
| aFullName | varchar(50) | NOT NUL | Full name’s admin |
| aUserName | varchar(50) | NOT NULL | Username’s admin |
| aPassWord | varchar(50) | NOT NULL | Password’s admin |
| uEmail | varchar(50) | NOT NULL | Email’s admin |
| rId | int | NOT NULL | Role’s admin |

**Table 2. Account User**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| uId | int | PRIMARY KEY | ID’s customer |
| uFullName | varchar(50) | NOT NUL | Full name’s customer |
| userName | varchar(50) | NOT NULL | Username’s customer |
| uPassWord | varchar(50) | NOT NULL | Password’s customer |
| rid | int | NOT NULL | Role’s customer |
| uAge | int | NOT NULL | Age’s customer |
| uPhone | int | NOT NULL | Phone’s customer |
| uEmail | varchar(50) | NOT NULL | Email’s customer |
| uAddress | varchar(50) | NOT NULL | Address’s customer |
| uBirthday | date | NOT NULL | Birthday’s customer |

**Table 3. Product**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| pId | int | PRIMARY KEY | ID’s product |
| pName | varchar(50) | NOT NULL | Name’s product |
| pPrice | int | NOT NULL | Price’s product |
| quantity | int | NOT NULL | Quantity’s product |
| pType | varchar(50) | NOT NULL | Type’s product |
| pDescription | varchar(500) | NOT NULL | Description’s product |
| codeProduce | varchar(100) | NOT NULL | code’s product |
| PpriceSale | int | NOT NULL | Sale’s product |
| pImage | varchar(150) | NOT NULL | Image’s product |
| dId | int | NOT NULL | id’s product |

**Table 4. Bill**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| bId | int | PRIMARY KEY | ID’s Bill |
| uId | int | NOT NULL | ID’s User |
| pName | varchar(50) | NOT NULL | Name’s Product |
| bQuantity | int | NOT NULL | Quantity’s Bill |
| pImage | varchar(200) | NOT NULL | Image’s Product |
| pDescription | varchar(200) | NOT NULL | Description’s Product |
| pId | int | NOT NULL | ID’s Product |
| pPrice | int | NOT NULL | Price’s Product |
| bTotalPrice | int | NOT NULL | TotalPrice’s Bill |
| bDateBuy | date | NOT NULL | DateBuy’s Bill |
| uEmail | varchar(50) | NOT NULL | Email’s User |
| bNote | varchar(100) | NOT NULL | Note’s Bill |

**Table 5. Billdetail**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| bdId | int | PRIMARY KEY | ID’s Bill |
| pId | int | NOT NULL | ID’s Product |
| bdQuantity | int | NOT NULL | Quantity’s Bill |
| bdTotalPrice | int | NOT NULL | TotalPrice’s Bill |

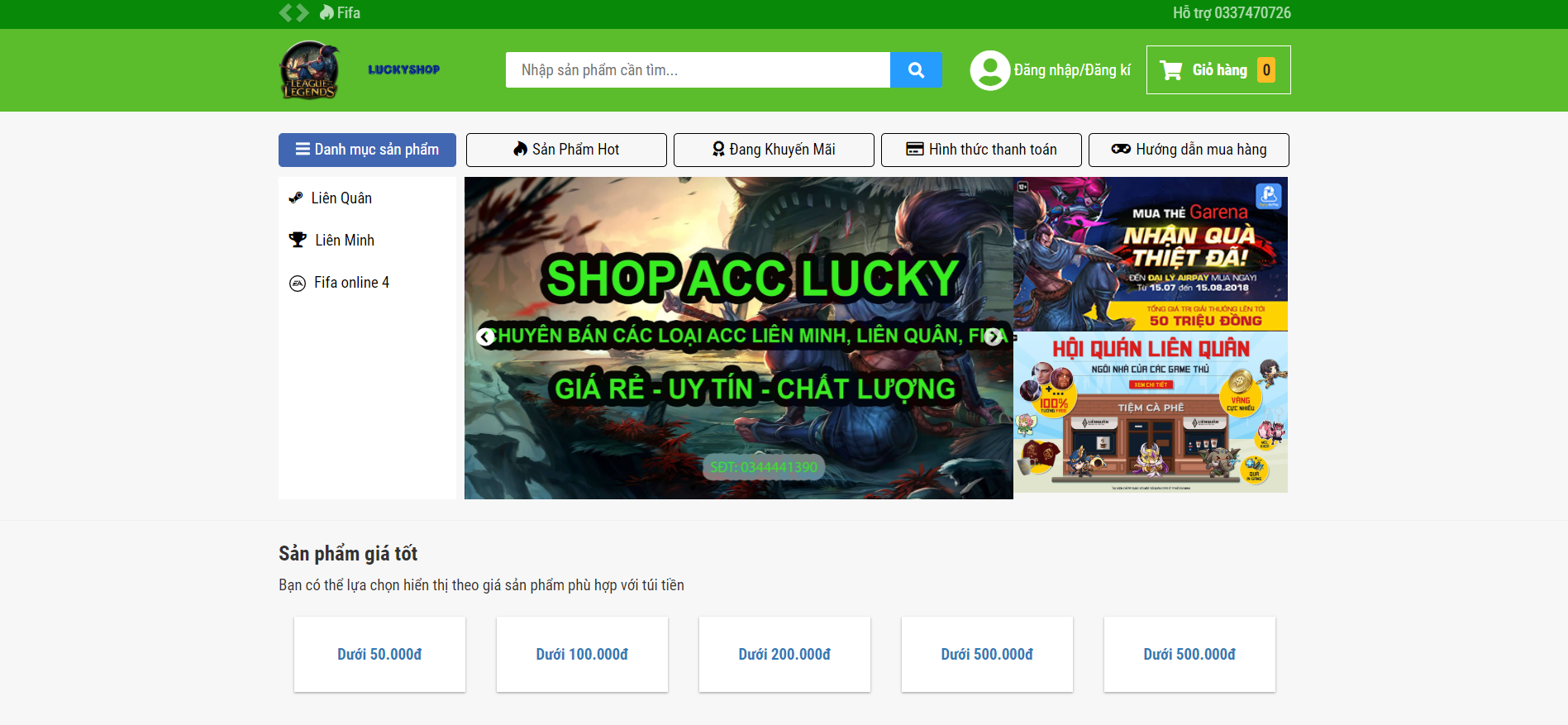
**Table 6. Descriptiondetail**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| **dId** | int | PRIMARY KEY | ID’s DesCription |
| **dDetail** | Text | NOT NULL | Detail’s Description |
| **dInfo** | Text | NOT NULL | Info’s Description |
| **dImage** | varchar(50) | NOT NULL | Image’s Description |

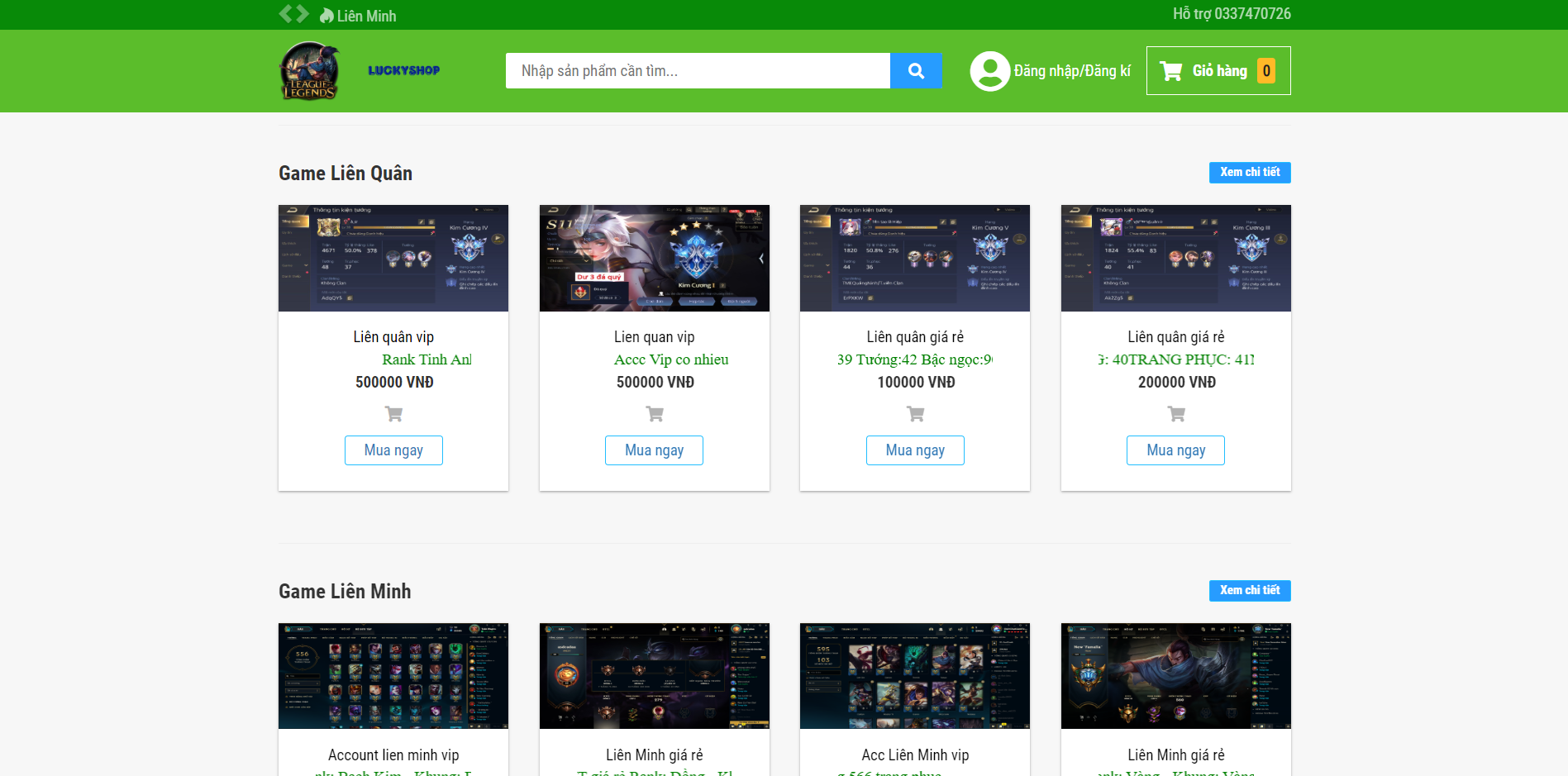
**Table 7. Role**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraint** | **Description** |
| **rId** | int | NOT NULL | ID’s Role |
| **NameRole** | varchar(50) | NOT NULL | Name’s Role |

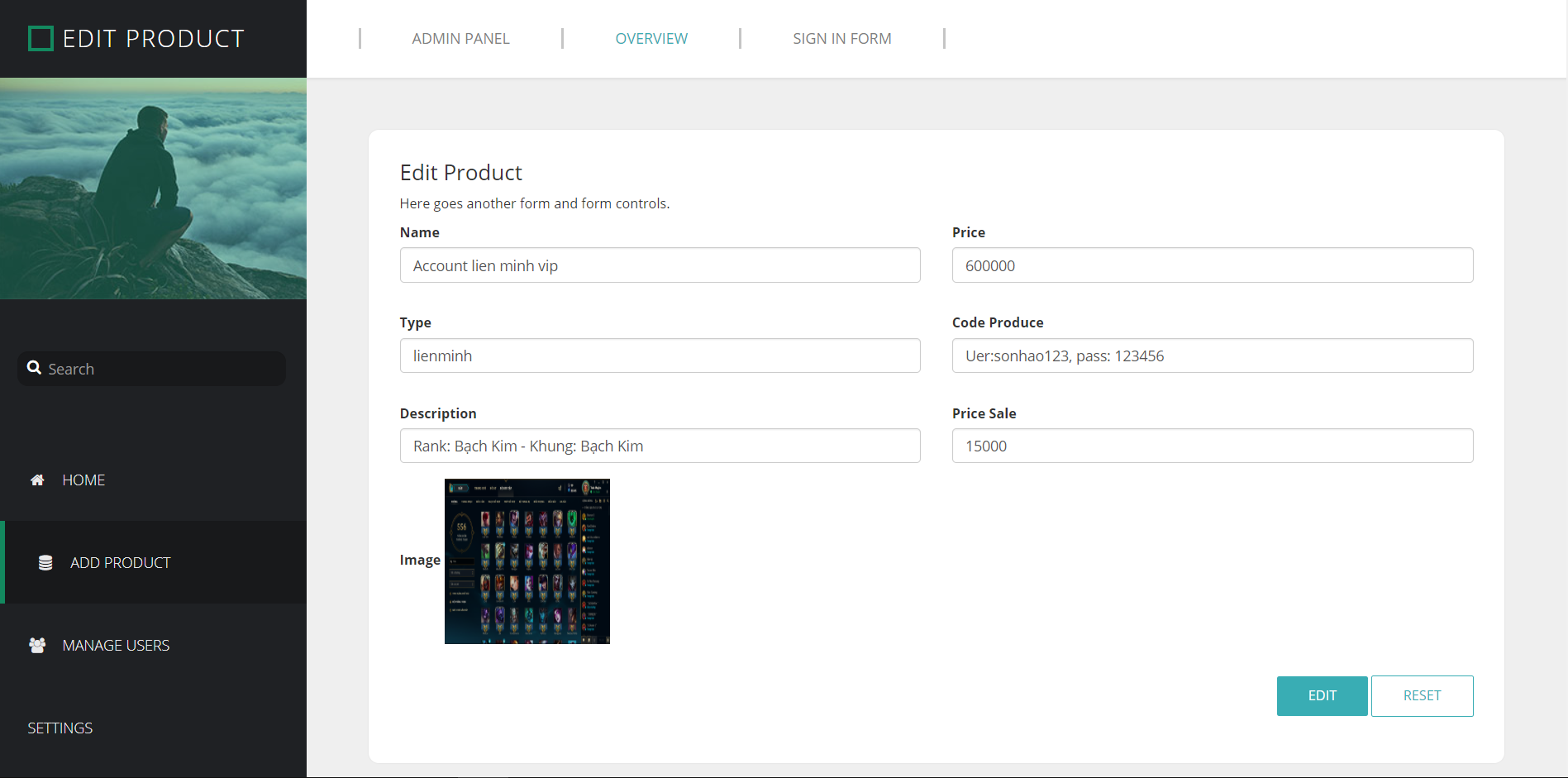
1. **User Interfaces**



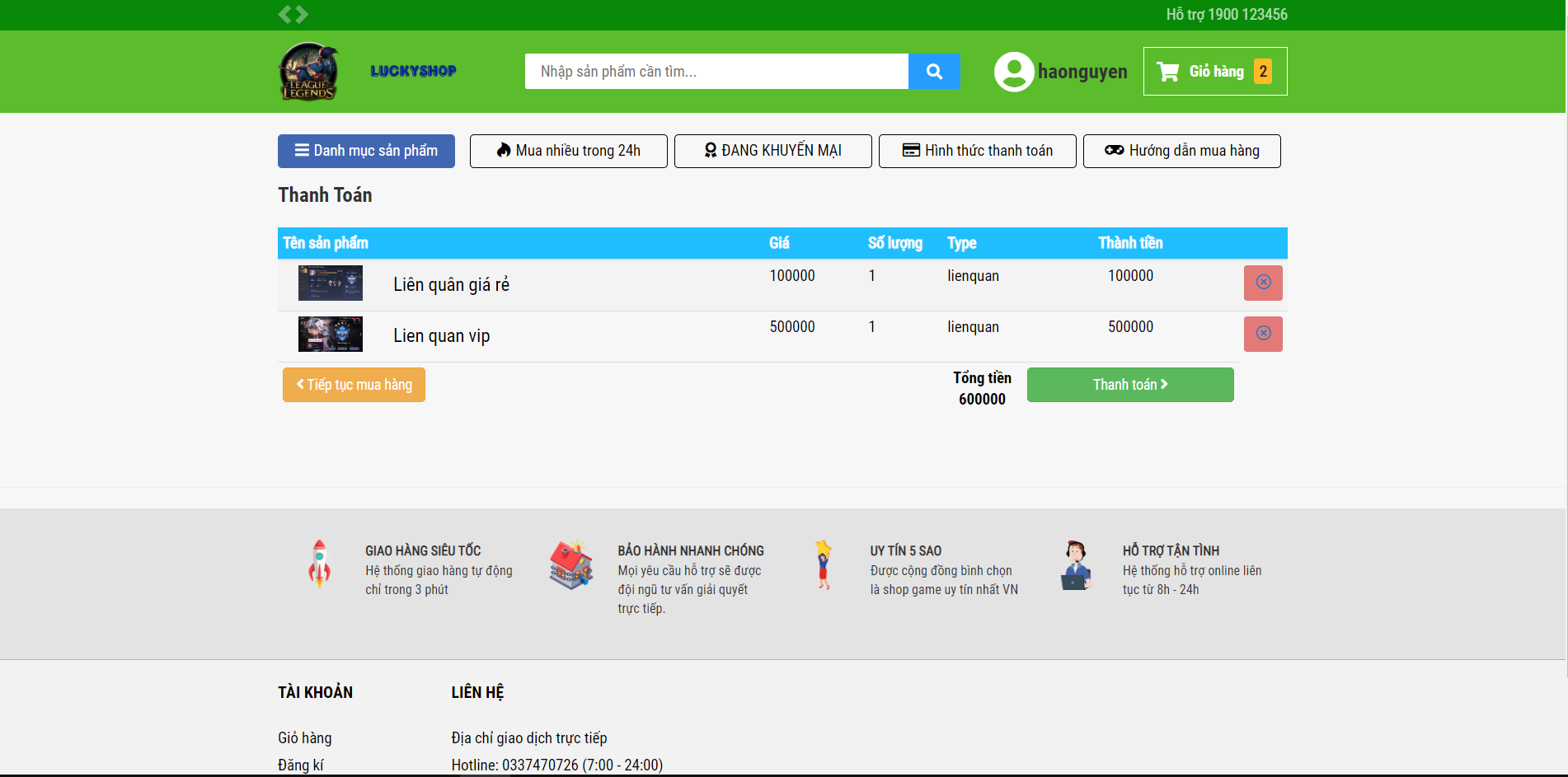
*Figure 13: Home page*



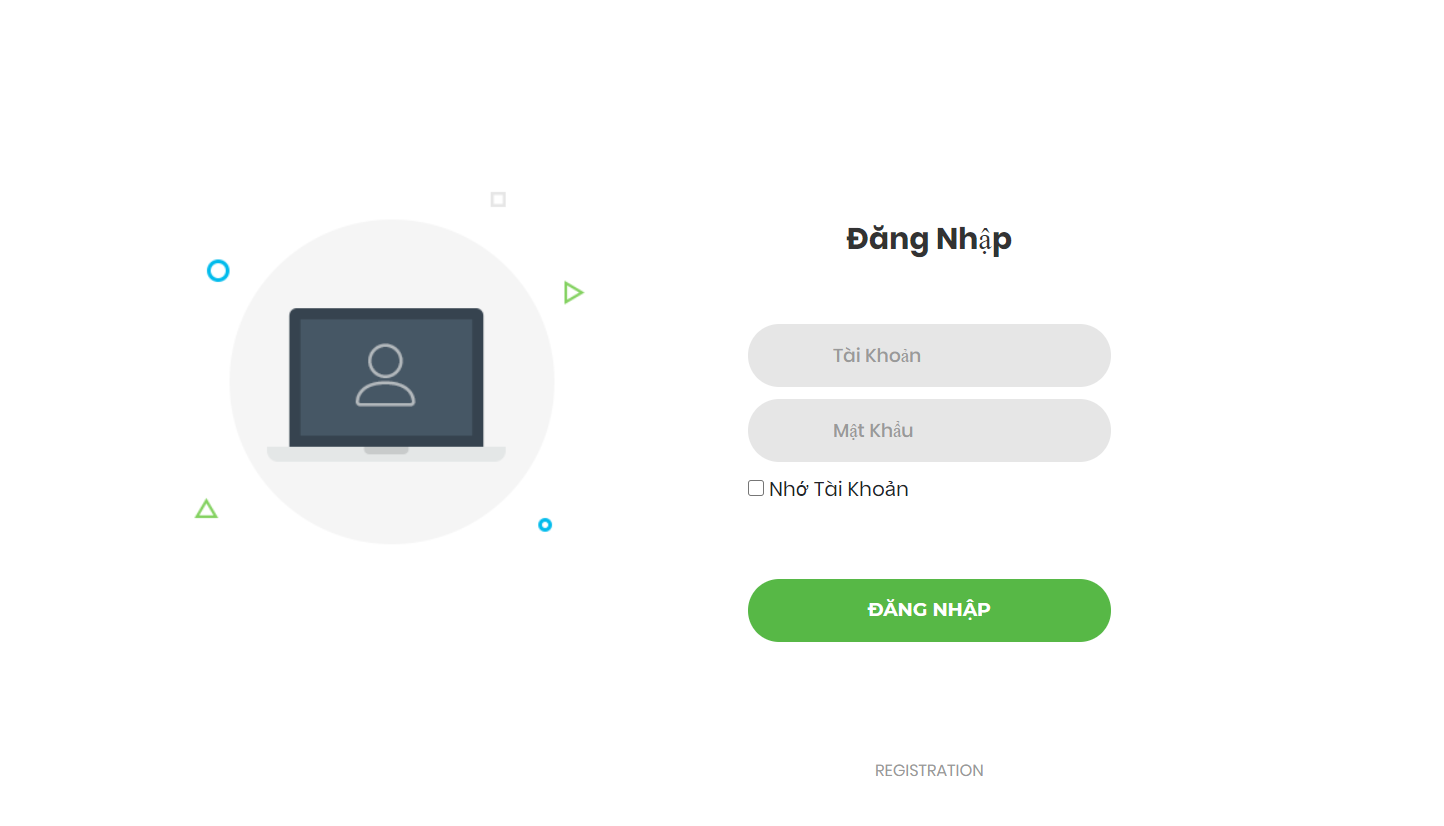
*Figure 14: Shop page*



*Figure 15: Update info account page*



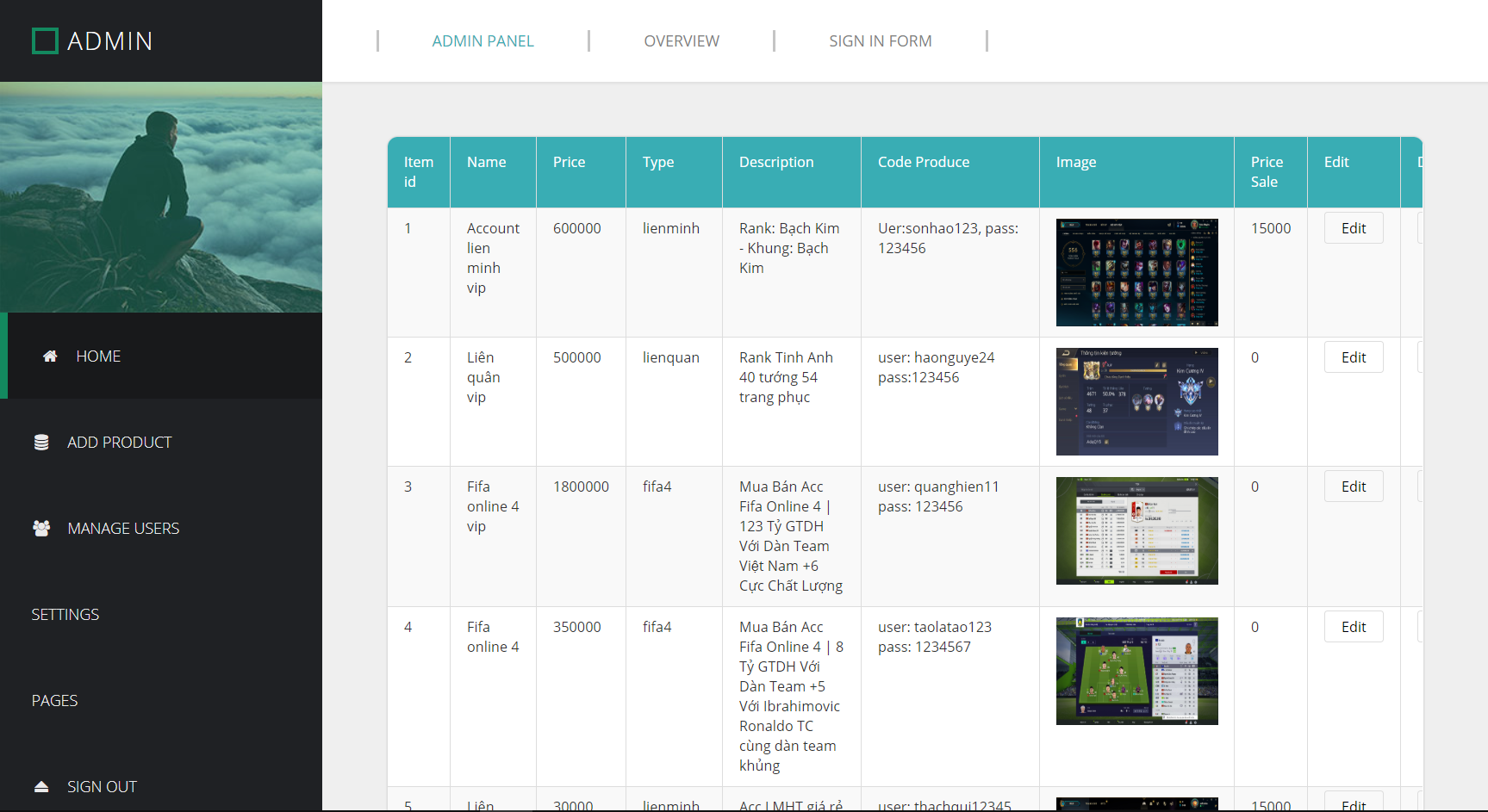
*Figure 16: Cart page*



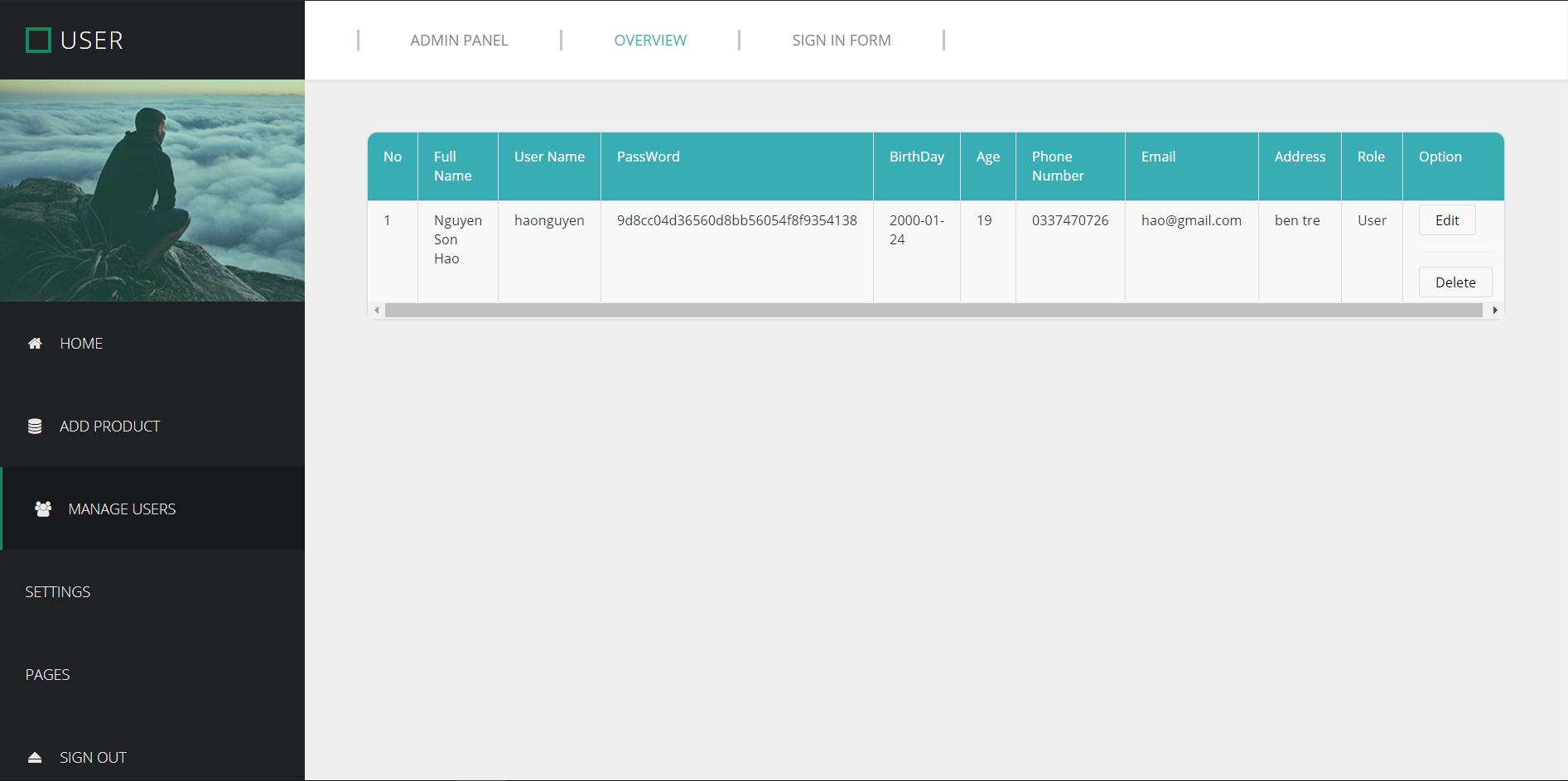
*Figure 17: Login*



*Figure 17: Registration*



*Figure 18: Admin home page*



*Figure 19: Admin manager users*

# **Chapter 5: Conclusion**

Thank mentor Luong Hoang Luong for helping us complete this project. The project is still at an early stage of lack of control, in the future we will develop more to make it more and more complete.