|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **VIETNAM NATIONAL UNIVERSITY – HOCHIMINH CITY**  **INTERNATIONAL UNIVERSITY**  **SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**  Logo | Trường Đại học Quốc Tế - International University  **WEB APPLICATION DEVELOPMENT**  **IT093IU**  **FINAL REPORT**  **Topic: EGAY AUCTION SITE**  By Group: 99% gambler – Members List   |  |  |  |  | | --- | --- | --- | --- | | **Number** | **Name** | **Student ID** | **Role** | | 1 | Đàm Nguyễn Trọng Lễ |  | Leader | | 2 | Nguyễn Quốc Trung | ITITIU22171 | Member | | 3 | Lê Hưng | ITCSIU22271 | Member | | 4 | Lê Nhật Anh | ITCSIU22254 | Member |   Instructor: Assoc. Prof. Nguyen Van Sinh |

TABLE OF CONTENTS

[I. INTRODUCTION 3](#_Toc199336668)

[1. ABOUT US 3](#_Toc199336669)

[2. THE PRODUCT’S INFORMATION 4](#_Toc199336670)

[3. WORK BREAKDOWN STRUCTURE 4](#_Toc199336671)

[4. DEVELOPMENT PROCESS: 7](#_Toc199336672)

[5. DEVELOPMENT ENVIRONMENT: 7](#_Toc199336673)

[II. REQUIREMENT ANALYSIS AND DESIGN 9](#_Toc199336674)

[1. REQUIREMENT ANALYSIS 9](#_Toc199336675)

[**A. FUNCTIONAL REQUIREMENTS:** 10](#_Toc199336676)

[**B. NON – FUCTIONAL REQUIREMENTS:** 27](#_Toc199336677)

[2. DESIGN 28](#_Toc199336678)

# **I. INTRODUCTION**

This section presents background information about the software development team - Gambler 99. It also introduces the core concept and basic details of the “Egay Auction Site” project. In addition, the main constraints encountered during the development process are outlined.

## **1. ABOUT US**

Gambler 99 is a team of four members formed as part of the Web Application Development course project. The primary objective is to design and implement a functional web application that demonstrates the practical application of web development concepts. This project also serves as valuable preparation for upcoming internships, allowing team members to gain hands-on experience with real-world tools, workflows, and problem-solving strategies. Throughout the development process, knowledge gained from lectures, textbooks, and online resources has been applied to build a platform that incorporates essential features of a modern website. The application supports dynamic user interaction, simulating the kind of technical and collaborative challenges commonly faced in professional software development environments.

Here is the information of team members and the tasks distribution:

|  |  |  |
| --- | --- | --- |
| **Name** | **Task** | **Contribution** |
| Đàm Nguyễn Trọng Lễ | Full - Stack | % |
| Nguyễn Quốc Trung | Front-end | % |
| Lê Hưng | Back-end | % |
| Lê Nhật Anh | Back-end | % |

Table 1. Task Distribution

## **2. THE PRODUCT’S INFORMATION**

Inspired by the success and influence of eBay - a pioneer in online auctions and consumer-to-consumer (C2C) sales—we chose to build a web like it to gain a deeper understanding of the unique architecture and functionality behind real-time bidding systems. The auction feature, which sets eBay apart from traditional e-commerce platforms, introduces dynamic pricing, time-sensitive competition, and live user interaction making it both technically challenging and intellectually engaging to implement. Through this project, we are able to explore essential concepts such as real-time auction mechanics, bid synchronization, user authentication, product management, and secure transactions.

## **3. WORK BREAKDOWN STRUCTURE**

The structure of this project can be expressed in the figure 1:

A diagram of a website

Description automatically generated with medium confidence

Figure 1. General structure of the project

This project contains three primary components: the Frontend, the Backend, and the Database. Each of these components is further organized into distinct substructures which describe the tasks needed to be accomplished of each teamembers. These diagrams below illustrate about the overview of the works of building database, backend and frontend. Each process of building them contains specification (identify all of things need to be constructed), implementation (implement, code base on the specification) and testing (check out the result of implementation whether meets the specification)

The Task for the Database Design can be expressed the the figure 2:

A diagram of a database

Description automatically generated

Figure 2. Database overview

* The tasks of the backend can be expressed in the figure 3:

A diagram of a system

Description automatically generated  
Figure 3. Backend overview.

* The tasks of the Frontend can be expressed in the figure 4:

A diagram of a computer program

Description automatically generated

Figure 4. Frontend overview.

## **4. DEVELOPMENT PROCESS:**

This project is built step by step from the basic feature to the more complex. Each feature was designed base on the “Sofware Development Life Circle” rule: planning, Define requirement, coding, testing. In the testing step, if is there any bugs, the process will go back to the coding. This approach provide a alige model for building the project, which employs incremental and iterative development that are also called sprints. The figure 5 below will visulize the Alige Model:

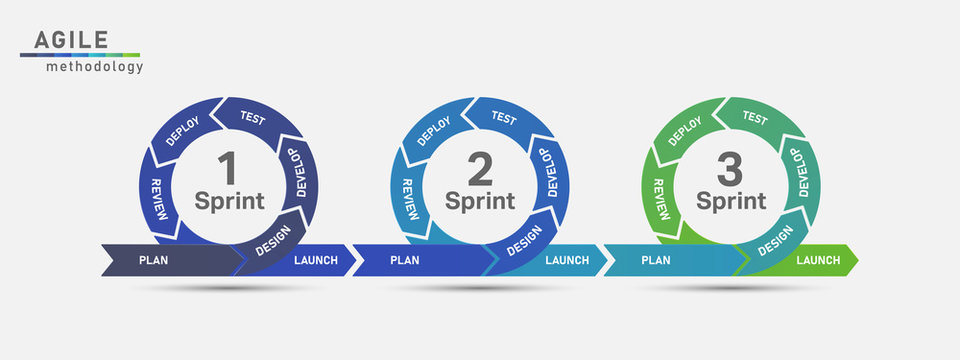


Figure 5. Alige Model.

## **5. DEVELOPMENT ENVIRONMENT:**

The project is an e-commerce platform, which requires a user-friendly, responsive, and secure implementation to meet modern web standards and user expectations. To achieve this, appropriate technologies were carefully selected to support both functionality and maintainability.

**Programming Languages:**

* Java was chosen for backend development due to its robustness, strong community support, and seamless integration with enterprise-level frameworks.
* JavaScript was used for frontend development, enabling dynamic user interfaces and smooth interaction with backend services.
* MySQL: A relational database management system used to store and manage structured data. It supports SQL for querying data and integrates smoothly with Java through Spring Data JPA, allowing efficient database access and manipulation.

**Frameworks:**

* Spring Boot was used to develop the backend, following the MVC architecture. It simplifies the creation of RESTful APIs and integrates well with:
* Spring Security for handling authentication and authorization using JWT (JSON Web Tokens).
* Spring Data JPA for database interaction and ORM mapping.
* Uses application.properties for centralized, flexible configuration.
* ReactJS Framework was used for building the frontend, providing a component-based architecture and efficient rendering. React allows for a responsive user experience and facilitates seamless integration with REST APIs.

**Tools:**

* Docker was used for hosting mySQL server: By using configure docker-compose file, and initialized SQL queries, it can easily install and running the database server, especially using docker to make the configuration setting (connection name, password, port) is consistent within each team members machine. Furthremore, mySQL workbench is also selected as a IDE tool for seeing and editing database.
* Intelliji and Vscode are two main IDEs for coding frontend and backend because they provide friendly UI and useful extensions, plugins for Java and Javascript (NodeJS).
* Postman was used to test all of the API services throughout the development process. It allowed the team to send HTTP requests (GET, POST, PUT, DELETE) to the backend, verify responses, debug issues, and ensure that each endpoint behaved as expected before connecting the frontend.

# **II. REQUIREMENT ANALYSIS AND DESIGN**

Requirement Analysis and Design is a critical phase in the software development lifecycle that focuses on understanding what the system should do and how it should be structured to meet those needs. During requirement analysis, stakeholders' needs are gathered, clarified, and documented to define functional and non-functional requirements. This ensures that the system will align with user expectations and business goals. Following this, the design phase translates these requirements into a blueprint for implementation, including system architecture, data models, interface designs, and component interactions. Together, these phases lay the foundation for building reliable, scalable, and user-centric software systems.

## **1. REQUIREMENT ANALYSIS**

**Use Case Diagram:**

A diagram of a product

Description automatically generated

### **A. FUNCTIONAL REQUIREMENTS:**

**Use case 1**: Registration into the system.

**Inputs:**

1. Username
2. Email
3. Password
4. Address

**Outputs:** navigate to login page to login the account just created

**Basic Course**

|  |  |
| --- | --- |
| Actor: New Customer | System |
| 1. Click register button | * 1. display the Sign-up Page. |
| 1. Enter username, email, address and password. |  |
| 1. Click Sign Up button. | * 1. validate input form and send to backend.   2. if success, return to the Login page.   3. if fail, display fail message. |

**Precondition**: none.

**Postcondition**: database update new account.

**User Story:**

As a new visitor to the e-commerce website, I want to register a new user account with my personal information, so that I can log in, browse products, make purchases, and manage my orders.

**Use case 2:** Login into the system

**Input**:

1. username
2. password

**outputs**:

1. Home page with user’s authorization [if success]
2. Login page [if fail]

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Click sign in button | * 1. display the Sign-in Page. |
| 1. Enter username and password. |  |
| 1. Click Sign In button. | * 1. validate input form and send to backend.   2. if success, return to the home page.   3. if fail, display fail message. |

**Precondition**: The customer must have an existing account.

**Postcondition**: The customer gains access to the website's features.

**User story:** As a registered customer, I want to log in using my username and password, so that I can securely access my personal account, view orders, and make purchases.

**Use case 3:** Login by google account

**Input**:

1. Gmail address
2. Password of the Gmail

**outputs**:

1. Home page with user’s authorization [if success]
2. Error Page [if fail]

**Basic Course**

|  |  |  |
| --- | --- | --- |
| Actor: Customer | System | Actor: Google Oauth2 |
| 1. Click sign in with google button | * 1. Direct to the google login page |  |
| 1. Enter Gmail address and password. |  | * 1. authenticate the Google account |
| 1. Click accept to access to the web | * 1. validate the google account input and redirect to home page |  |

**Precondition**:

1. Customer already have a google account.

**Postcondition**:

1. The customer gains access to the website's features.

**User story:** I want to quickly register and log in using my Google account, so that I can access the web without manually entering my information or remembering another password.

**Use case 4:** Browse product

**Input**: None

**Output**: See all active products

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| Click products button | display the product page. |

**Precondition**: none

**Postcondition**: none

**User story:** as a customer, I want to see all of the products to buy.

**Use case 5**: Search product by name, category

**Input**:

1. Searching keywords
2. Select category (optional)

**Output**:

1. Display all of products that have names match with the searching keywords and category.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Input keywords |  |
| 1. Select the category. | * 1. display category dropdown |
| 1. Click search button | * 1. Display products for the given condition. |

**Precondition**: none

**Postcondition**: none

**User story:** as a customer, I want to search products that I needed so that I can reduce my time for finding the appropriate products.

**Use case 6**: add product to cart

**Input**:

1. Select product to add
2. Choose the quantity to add

**Output**:

1. The selected products will be in the cart.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Choose a product to add | * 1. Display product details page |
| 1. Select quantity |  |
| 1. Click add to cart button | * 1. Add the chosen product and quantity to customer’s cart |

**Precondition**:

1. The product must be active

**Postcondition**:

1. Display success message that product was added to cart

**User Story**: As a customer, I want to see the details information of a specific product. If I like it, I can choose the quantity and add it into my cart.

**Use case 7:** Remove product from cart

**Input**:

1. Select product want to remove.
2. Input quantity to remove.
3. Customer can choose clear cart to remove all of items. (optional)

**Output**:

1. The product is deleted from cart or decrease the amount of quantity.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Click cart button | * 1. Display cart page |
| 1. Select quantity of a specific product to remove. | * 1. remove the cart item and update cart |

**Precondition**:

1. cart is not empty

**Postcondition**:

1. the cart item is deleted.

**User story**: as a customer, I want to remove products from my shopping cart or decrease the quantity of a product, so that I can update my order to only include items I intend to purchase.

**Use case 8**: review a product

**Input**:

1. Write the review for a product.
2. Choose the rating.

**Output**:

1. Display success message
2. Display comment and rating in the product detail page.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Access to a product | * 1. Display product details page |
| 1. Input review |  |
| 1. Choose the rating (from 1 to 5) | * 1. Add the chosen product and quantity to customer’s cart |
| 1. Submit review | * 1. Validate review and update |

**Precondition**: none

**Postcondition**:

1. The review is added

**User story:** As a customer, I want to see the reviews of a product and also give my review for a product.

**Use case 9**: make purchase (payment)

**Input**:

1. Complete check cart and proceed to payment.
2. Input card number [if use Stripe method]

**Output**:

1. Update order and payment.
2. Display transaction details
3. Display fails message [if process payment fails]

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Access to cart page | * 1. Display product cart page |
| 1. Click checkout button | * 1. display payment methods |
| 1. Choose a payment method to purchase |  |
| 1. Submit payment information | * 1. Process payment   2. Display transaction information |
| 1. Confirm and view order | * 1. Display order details |

**Precondition**:

1. The cart is not empty
2. Customer would have a valid card number [if using Stripe payment]

**Postcondition**:

1. The order is created from cart.
2. The payment succeeds and update for order and payment.

**User story**:

1. As a customer, I want to make purchase directly on the website using my card number.
2. As a customer, I want to pay by cash when the products are delivered.

**Use case 10**: Bid an auction

**Input**:

1. Bidders submit the amount of money that they want to buy a product on auction.

**Output**:

1. The new bid is accepted and display immediately on the auction page.
2. Display fails if the auction is due, or the amount of bid is not larger than the current price.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Bidder | System |
| 1. Access to auction page | * 1. Display product auction page |
| 1. Select an active auction to bid | * 1. Display the auction detail page |
| 1. Place an amount of money to bid | * 1. Update the current price   2. Display the new bid |

**Precondition**:

1. The auction must be active.
2. The amount of bid must be larger than the newest bid.
3. The bidder is not owner of this product.

**Postcondition**:

1. Update current price for the auction.
2. Display the newest bid in real time.

**User story**: As a customer, I want to place bids on a unique product that is listed for auction, so that I have a chance to win and purchase the product if I offer the highest bid before the auction ends.

**Use case 11**: Update profile

**Input**:

1. Enter the feature that users want to change

**Output**:

1. The account information is update

**Basic Course**

|  |  |
| --- | --- |
| Actor: User | System |
| 1. Access to edit account page | * 1. Display account information |
| 1. Change some information |  |
| 1. Place an amount of money to bid | 3.1 Display update success |

**Precondition**:

1. User login into the web

**Postcondition**:

1. The account information is update

**User story:** As a customer, I want to update my personal information such as address and email,

So that my shipping and delivery details are always accurate.

**Use case 12**: Reset password.

**Input**:

1. Users enter the username

**Output**:

1. The password is reset to “123456” (user can change it later)

**Basic Course**

|  |  |
| --- | --- |
| Actor: User | System |
| 1. Click forgot password | * 1. Display the reset password |

**Precondition**:

1. User input correct username

**Postcondition**:

1. A new password is released

**User story**: As a customer, I want to reset and change my password when I forget it, So that I can regain access to my account securely.

**Use case 13**: Create product

**Input**:

1. Seller fill all of information of a product

**Output**:

1. A new product is created for that seller

**Basic Course**

|  |  |
| --- | --- |
| Actor: Seller | System |
| 1. Access to seller page | * 1. Display seller page |
| 1. Click create new product |  |
| 1. Fill all information of new product | 3.1 The new product is created |

**Precondition**:

1. User must be a seller

**Postcondition**: none

**User Story:** As a customer, I also want to sell my own products on this website.

**Use case 14:** Delete product

**Input**:

1. The Seller selects a product to delete

**Output**:

1. The product is deleted

**Basic Course**

|  |  |
| --- | --- |
| Actor: Seller | System |
| 1. Access to seller page | * 1. Display seller page |
| 1. Click delete product | 2.1 the product is deleted |

**Precondition**:

1. User must be a seller
2. The seller already has at least one product.

**Postcondition**:

1. Buyer cannot see that product because it is deleted

**User Story:** As a seller, there are some products that I cannot supply more, so I need to delete them on the web to avoid some customers buy them, but I cannot offer.

**Use case 15**: Update product

**Input**:

1. the sellers submit the updating information of a product

**Output**:

1. The product information is updated

**Basic Course**

|  |  |
| --- | --- |
| Actor: Seller | System |
| 1. Access to seller page | * 1. Display seller page |
| 1. Select a product and click update |  |
| 1. Submit updating information | 3.1 The product is updated |

**Precondition**:

1. User must be a seller.
2. Seller has at least one product.

**Postcondition**:

1. The product is updated

**User Story:** As a seller, I want to update my product information (such as name, description, price, and stock), so that customers see the most accurate and up-to-date details when browsing or purchasing.

**Use case 16**: Create auction

**Input**:

1. Seller selects an inactive product to create auction.
2. Seller input start and end time, starting price and reserve price if needed.

**Output**:

1. An auction is created for the selected product.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Seller | System |
| 1. Access to seller page | * 1. Display seller page |
| 1. Select a product and click auction | * 1. Display add auction page |
| 1. Submit auction setup | 3.1 The auction is created for that product |

**Precondition**:

1. The user must be a seller.
2. The seller has at least one product.
3. The product selected is not an active product.

**Postcondition**:

1. Seller who owns the product in this auction cannot place any bid.

**User story:** As a seller, I want to create auctions for my unique products, so that I can sell my products with the prices as high as possible.

**Use case 17**: Place bid for an auction

**Input**:

1. Customers select an auction and submit bid amount.

**Output**:

1. The system displays the newest bid.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Access to auction page | * 1. Display auction page |
| 1. Select an auction to place a bid | * 1. Display auction details page |
| 1. Submit bid | 3.1 Display the newest bid. |

**Precondition**:

1. The bidder does not own the product in the selected auction.
2. The auction must be active.

**Postcondition**:

1. The current price of the auction is update to the amount of newest bid.

**User story:** As a customer, I want to buy unique products through an auction feature, so that I can place bids and potentially win items at competitive prices.

**Use case 18**: win the auction

**Input**: none

**Output**:

1. Customer will have an order with the product in the auction they won.
2. Customer will receive email notification about winning an auction.

**Basic Course**

|  |  |
| --- | --- |
| Actor: Customer | System |
| 1. Placed the highest bids | * 1. Check the winners and create orders for them   2. Send email notification |

**Precondition**:

1. The customer’s bid must be the highest in the auction.
2. The auction is ended.

**Postcondition**:

1. An order for product in the winning auction is created.
2. Other bidders cannot place any bid on the ended auction.

**User Story:** I want the system automatically to create an order and send email if I win an auction.

### **B. NON – FUCTIONAL REQUIREMENTS:**

**1. Performance Requirements:**

* Response Time: The system should respond to user actions (e.g., bidding, page loading) within 2 seconds under normal load.
* Concurrent Users: The system should support at least 10,000 concurrent users during peak bidding hours.
* Real-Time Updates: Bid updates on auction items should reflect on all active user sessions within at least 1 second.

**2. Availability & Reliability**

* Uptime: The system should have an availability of 99.9% uptime over a 12-month period.
* Backup and Recovery: Daily automated backups must be performed, with the ability to recover data within 30 minutes after a failure.

**3. Scalability**

* The system should be able to scale horizontally to handle increased traffic during promotional or festive periods without performance degradation.

**4. Security**

* Authentication: Secure user login with OAuth 2.0 support (e.g., Google login).
* Authorization: Different roles (customer, seller, admin) must have proper access control.
* Data Protection: All sensitive user data must be encrypted in transit (HTTPS) and at rest.
* Audit Logs: All bidding and transaction activities should be logged for dispute resolution and monitoring.
* Allow some pages (list of products, auctions, …) can be accessed without login.

**5. Usability**

* The user interface should be intuitive and responsive, working seamlessly across desktop, tablet, and mobile devices.
* The learning curve for a new user should not exceed 5 minutes.

**6. Maintainability**

* The codebase should follow modular architecture (e.g., microservices or layered architecture) to allow easier updates and debugging.
* New features should be able to be integrated with minimal downtime.

## **2. DESIGN**