**HCM University of Technology and Education**

**Facility of International Education**

**🕯✡🕮🕮✡🕯**

**A logo of hands holding a book

Description automatically generated**

**Final Report**

**SUBJECT: WEB PROGRAMMING**

**AUCTION WEBSITE**

**Instructor: Mai Anh Thơ**

**CLASS: [WEPR330479E\_23\_1\_01FIE](https://fhqx.hcmute.edu.vn/course/view.php?id=51418" \o "Web Programming (2+1)_ Nhom 01FIE)**

**STUDENT NAME MSSV**

**Đỗ Huỳnh Bảo Đăng 21110764**

**Bùi Chiến Thắng 21110798**

**Nguyễn Văn Anh Đồng 21110016**

**Ho Chi Minh city, December 2023**

**[Chapter 1: System Description 1](#_Toc13142)**

**[1.1. Application Overview 1](#_Toc28516)**

**[1.2. Key Features 1](#_Toc14184)**

**[1.3. Technical specifications 2](#_Toc18947)**

**[Chapter 2: System analysis and design 3](#_Toc29870)**

**[2.1. System Architecture and Component Interaction 3](#_Toc9698)**

**[2.2. BUSINESS LAYER 4](#_Toc13464)**

**[2.2.1. Overview 4](#_Toc16305)**

**[2.2.2. Class diagram 4](#_Toc24103)**

**[2.2.3. Example 4](#_Toc21300)**

**[2.3. Under layer Database 5](#_Toc5038)**

**[2.3.1. Purpose 5](#_Toc142)**

**[2.3.2. The use of DBUtil.java class 6](#_Toc18299)**

**[2.3.3. Example 6](#_Toc16570)**

**[2.4. Controller layer 8](#_Toc18137)**

**[2.4.1. Overview the use of Java servlets 8](#_Toc2836)**

**[2.4.2. Example 9](#_Toc30220)**

**[Chap 3: USER INTERFACE 11](#_Toc7133)**

**[Chapter 4 CONCLUSION 17](#_Toc3375)**

**[4.1. Result 17](#_Toc3097)**

**[4.2. Advantages and disadvantages 17](#_Toc25896)**

**[4.3. Development 18](#_Toc11588)**

**[Reference 19](#_Toc7431)**

**[Team Evaluation 20](#_Toc28581)**

# Chapter 1: System Description

## 1.1. Application Overview

The primary objective of the auction project is to create an online platform that facilitates efficient and transparent bidding processes The application serves as a virtual marketplace where users can buy and sell goods or services through a competitive bidding system. By leveraging the power of the web, the project aims to streamline traditional auction processes, making them more accessible, secure, and convenient for a diverse user base

## 1.2. Key Features

The auction project includes a range of features designed to enhance the user experience and ensure the seamless execution of auctions Some key features include:

- User Registration and Authentication:

+ Users can create accounts, fill in and adjust their personal information

+ Users can log in and log out

- Item Listing and Management:

+ Buyers can list/search items for auction, providing detailed descriptions of product

+ Cart to save Products for easy access and storing information

- Bidding System:

+ Users can place bids on items they are interested in, fostering a competitive environment

+ Real-time bidding updates and notifications keep participants informed of the current auction status

- Auction Monitoring and Reporting:

+ Notification inbox allow users to monitor ongoing and completed auctions

+ Reporting tools such as bid histories, winning bids prices, etc

1.3. Technical specifications

Front-End Technologies:

* HTML5, CSS, JSP for building the user interface
* Back-End Technologies:
* Java, Servlet for the server-side runtime environment
* JPA: an API for working with databases
* Database Management System:
* POSTSQL for efficient storage and retrieval of auction-related data

# Chapter 2: System analysis and design

## 2.1. System Architecture and Component Interaction

A diagram of a model

Description automatically generated

-View Layer:

Purpose: Display information and create an interactive environment for users

Technology: HTML5, CSS, JSP

Role: User Interface

-Controller Layer:

Technology: Java Servlets

Role: Intermediary between View and Model

+ Request Handling: Receives and validates requests from Views

+ Communication: Call model functions to process data

+ Data stream: Receives processed data from Model layers and displays it on the view layers

-Model Layers

Function: Processing at the Underlying Layer

Database Interaction: Manipulation Using JPA

Sublayers:

+ Business Class: Creating Database Classes with Attributes and Relationships

+ Lower DB Layer: Operations with Database Using Queries, Controller Layer (Servlet) Accessed this class via Static Functions

## 2.2. BUSINESS LAYER

### **2.2.1. Overview**

After a thorough analysis of the use case for this web application, it was decided that there would be at least seven business objects:

- Buyer (Stores buyer information)

- Seller (Stores seller information)

- Product (Stores product information)

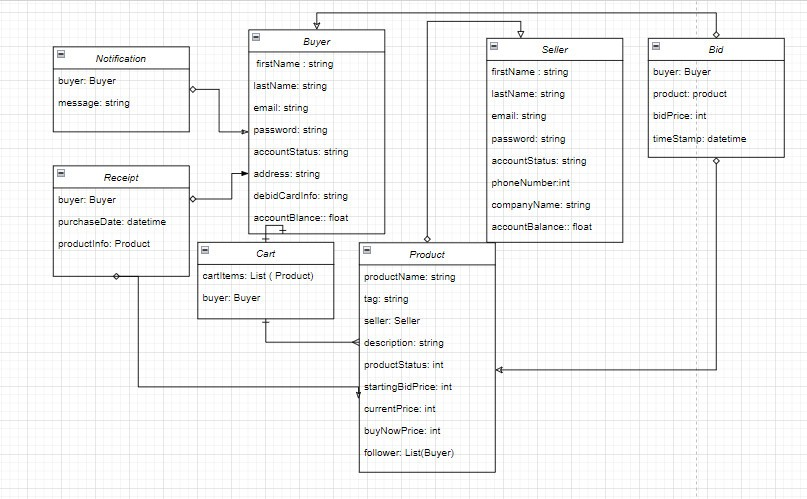
- Cart (Contains all the products that the customer has put in the cart )

- Bid (Save bid transaction information of customers)

- Recipt (Contains all invoice information)

- Notification (contains message information for customers (when the bid price changes, notification that you are the winner, etc))

### **2.2.2. Class diagram**



### **2.2.3. Example**

The example above is a typical example of a busisness class including attributes, attribute data types and relationships.

|  |
| --- |
| @Entity  public class Notification implements Serializable{    @ManyToOne  private Buyer user;    @Id  @GeneratedValue(strategy = GenerationType.AUTO)  private Long idNoti;  private String message;  public Long getIdNoti() {  return idNoti;  }  public void setIdNoti(Long idNoti) {  this.idNoti = idNoti;  }    public void setUser(Buyer user) {  this.user = user;  }  public Buyer getUser() {  return user;  }  public String getMessage() {  return message;  }  public void setMessage(String message) {  this.message = message;  }      } |

## 2.3. Under layer Database

### **2.3.1. Purpose**

Represents a Data Access Object (DAO) class responsible for handling database interactions for User entities. This class facilitates the communication between the application's business logic (User class) and the underlying database.

### **2.3.2. The use of DBUtil.java class**

The DBUtil class serves as a utility for managing the creation and retrieval of the JPA EntityManagerFactory, providing a centralized point for obtaining the necessary resources to interact with the database using JPA in a Java application.

The crucial role of this class in the web application stems from its responsibility for creating an EntityManager object. This object is configured with the settings provided in the persistence.xml file, establishing a connection and facilitating the extraction and insertion of data into the database.

|  |
| --- |
| package auction.data;  import javax.persistence.EntityManagerFactory;  import javax.persistence.Persistence;  public class DBUtil {  private static final EntityManagerFactory emf =  Persistence.createEntityManagerFactory("auctionPU");    public static EntityManagerFactory getEmFactory() {  return emf;  }  } |

### **2.3.3. Example**

The example below is of a class below the DB with static functions to add, delete, and edit data in the database as well as calling some data in accordance with user requirements.

|  |
| --- |
| package auction.data;  import javax.persistence.EntityManager;  import javax.persistence.EntityTransaction;  import javax.persistence.NoResultException;  import javax.persistence.TypedQuery;  import auction.business.Buyer;  public class BuyerDB {  public static void insert(Buyer buyer) {  EntityManager em = DBUtil.getEmFactory().createEntityManager();  EntityTransaction trans = em.getTransaction();  trans.begin();  try {  em.persist(buyer);  trans.commit();  } catch (Exception e) {  System.out.println(e);  trans.rollback();  } finally {  em.close();  }  }    public static void update(Buyer buyer) {  EntityManager em = DBUtil.getEmFactory().createEntityManager();  EntityTransaction trans = em.getTransaction();  trans.begin();  try {  em.merge(buyer);  trans.commit();  } catch (Exception e) {  System.out.println(e);  trans.rollback();  } finally {  em.close();  }  }      //Two function for check existed email  public static Buyer selectUser(String email) {  EntityManager em = DBUtil.getEmFactory().createEntityManager();  String qString = "SELECT u FROM Buyer u " +  "WHERE u.email = :email";  TypedQuery<Buyer> q = em.createQuery(qString, Buyer.class);  q.setParameter("email", email);  try {  Buyer user = q.getSingleResult();  return user;  } catch (NoResultException e) {  return null;  } finally {  em.close();  }  }  public static boolean emailExists(String email) {  Buyer u = selectUser(email);  return u != null;  }    //Check acount and password    public static boolean checkPassword(String email, String password) {  Buyer u = selectUser(email);    if(u!=null ){  if (u.getPassword().equals(password)) {  return true;  }  }  return false;  }        } |

## 2.4. Controller layer

### **2.4.1. Overview the use of Java servlets**

Java Persistence API, with its ORM capabilities, significantly streamlines database interactions in Java applications. The abstraction provided by ORM simplifies development, enhances code maintainability, and promotes a more productive and intuitive approach to working with relational databases.

As I mentioned in 2.1, the servlet take an intermediary between view layer and model layer. It receives and validates requests from views, call the function from model layer to processed data and send then back to view layers

### **2.4.2. Example**

The code below is a typical example of the server receiving data, validating it, calling functions from the model to process and returning further data to the view layer.

|  |
| --- |
| @WebServlet("/userLogin")  public class UsersServlet extends HttpServlet {  @Override  protected void doPost(HttpServletRequest request,  HttpServletResponse response)  throws ServletException, IOException {    HttpSession session = request.getSession();  String url = "/index.jsp";    // get current action  String action = request.getParameter("action");  if (action == null) {  action = "register"; // default action  }        else if (action.equals("register")) {    //Get imformation  String newEmail = request.getParameter("email");  String newPassword = request.getParameter("password");    //Create new buyer with information  Buyer buyer = new Buyer();  buyer.setEmail(newEmail);  buyer.setPassword(newPassword);    Seller seller = new Seller();  seller.setEmail(newEmail);  seller.setPassword(newPassword);  //save to database  String message;  if (BuyerDB.emailExists(newEmail)) {  message = "This email address already exists.<br>" +  "Please enter another email address.";  url = "/RegisteringForm.jsp";  }  else {  message = "Create new account succesfully, please login in";  BuyerDB.insert(buyer);  SellerDB.insert(seller);  url = "/LoginForm.jsp";  }        getServletContext()  .getRequestDispatcher(url)  .forward(request, response);  }    @Override  protected void doGet(HttpServletRequest request,  HttpServletResponse response)  throws ServletException, IOException {  doPost(request, response);  }  } |

# Chap 3: USER INTERFACE

|  |  |  |
| --- | --- | --- |
| **No.** | **Form** | **Funtion** |
| 1 | A screenshot of a website  Description automatically generated | This is the home page of our web. |
| 2 | A screenshot of a computer  Description automatically generated | This is where we are going to log in to the web page. It will check whether your account is stored in the database or not. |
| 3 | A screenshot of a computer  Description automatically generated | This is where you create an account. |
| 4 | A screenshot of a computer  Description automatically generated | This is where you add your personal profile to the website. |
| 5 | A screenshot of a computer  Description automatically generated | This is the page that shows your information, and you can adjust it. Additionally, you can view the products you have won and sell your own products. |
| 6 |  | This is the form where we initiate our auction. The seller will enter the starting price and set the end time. |
| 7 | A screenshot of a web page  Description automatically generated | After adding a product, you can view all of the products here, and you can update your bid, which the system will save. After the auction ends, the product will disappear. |
| 8 | A screenshot of a online auction  Description automatically generated | This is the cart section, where you can save the item you want to bid on and also set the bid price |
| 9 | A screenshot of a contact us  Description automatically generated | This is the contact section. If you have any problems, you can send your questions here, and we will respond later |
| 10 | A green and white background  Description automatically generated with medium confidence | This is the invoice page where you will be able to check the information of the product you have bought |
| 11 | A screenshot of a computer screen  Description automatically generated | This is the section where you can see all the products you have won. |
| 12 |  | This is the place where you can see all notifications, such as changes in prices during an auction |

# Chapter 4: CONCLUSION

## 4.1. Result

By researching, developing and perfecting web application for auction purpose,, our group has achieved a number of achievements.

Successfully applied MVC model and Java servlet and JPA technology

Complete interface and features for auctions. Some excellent features included

+ Automatically end the auction and determine the winner when the auction time exceeds

+ Automatically send a message to the buyer when the bid price is changed

+ Shopping cart and myWinProduct for users to manage and check their items

+ And some other features, etc

## 4.2. Advantages and disadvantages

\*Advantage

+ Easy-to-use, user-friendly software

+ Automatically bidding to reduce the error by human

+ Real-time auctions ensure fairness, transparency and minimize errors

+ Some functions such as transaction history, my winning product, notifications, etc. help customers trust in the company's transparency.

+ Login and authorization function to increase application security

\*Disadvantage

+

+ Some statistical features for seller are not available yet

+ Haven't tested all possible real cases

## 4.3. Development

Our team tried to complete the requirements of the project. Due to limited time, our group focuses mainly on the management Control bidding and increase experience, ensuring fairness and transparency for buyers

In the future, my team will develop and complete some more features such as

+ Develop statistical functions for seller to know the store's revenue situation

+ Test all possible real cases and fix bugs (if any)

+ Develop feature to automatically calculate salary based on employee's working time

## **Reference**

* *Java Servlet, JSP and Hibernate: Build eCommerce website*. (n.d.). Udemy. <https://www.udemy.com/course/java-servlet-jsp-and-hibernate-build-a-complete-website/>
* Team, D., Team, D., & Team, D. (2021, July 11). *Online Shopping System in Java using JSPs & Servlets*. DataFlair. <https://data-flair.training/blogs/online-shopping-system-java-jsp-servlets/>
* Baeldung, & Baeldung. (2022, July 4). *Guide to JavaServer Pages (JSP) | Baeldung*. Baeldung. <https://www.baeldung.com/jsp>
* Gervasio, A., & Gervasio, A. (2017, January 3). *Building a Web App with Java Servlets*. SitePoint. <https://www.sitepoint.com/tutorial-building-web-app-with-java-servlets/>
* *E-Commerce Website built using Java, JSP, Servlets and Hibernate with Database in MySQL | Abhinav123*. (n.d.). <https://coderspacket.com/e-commerce-website-built-using-java-jsp-servlets-and-hibernate-with-database-in-mysql>
* *Servlets and JSPs tutorial: Learn web applications with Java*. (n.d.). Udemy. <https://www.udemy.com/course/javawebtut/>

# Chapter 5: Fixing bug

During the development process we encountered quite a few bugs. We encountered many difficulties and spent a lot of time fixing them.However, that was because we were first exposed to java servlet and JPA technology and such a large project. Through this project, we are quite confident that we won't let those errors happen again or if they do, we will fix them quickly. Some typical bugs include

+Table cart did not have a buyer at first when we developed the class diagram. That makes it impossible for us to know who the cart belongs to. We fixed that simply by adding the buyer attribute to the Cart table. Although simple, it helps us realize the magic of JPA and code first. In the true object-oriented spirit, we only need to develop the class diagram and code. If it is wrong or missing in the database, we can simply edit and correct it without affecting the functions we have coded.

+ Log out but other tabs are enabled to use that account. We fix it by deleting the entire user session not just, not just the navigation to the login page.. However, when deleting, pages loaded using users will have an error of not being able to find the current user. We fixed it by adding try catch to force the user to log in again

+ The lower layer class in the database cannot compare the where condition in the query if the data type being compared is a self-generated data type. When running that command, the system will report an error and say that the thing you are comparing is not found or does not exist. If the compared data type is string, int or other built-in types, the system will understand and automatically recognize it. However, it is a table type that we created in the JPA business class. We fixed it by set parameter directly "q.setParameter("currentUser", currentUser);"

+ The problem is finding a way to automatically end bidding. The problem is finding a way to automatically end bidding. This problem really causes quite a lot of difficulties for us, because time-related issues are quite complicated. There is quite a bit of documentation online so we used a trial and error approach. We selected some of the most likely successful methods such as ServletContextListener, ScheduledExecutorService, Schedule, etc and tested them one by one. Finally, we succeeded using scheduler.schedule(()) with the input being the delay time we calculated from the auction end time minus currentTimeNow

,etc

# Chapter 6: Team Evaluation

### Collaboration and Version Control

Our group uses github for version control as well as to collaborate with each other. More specifically, our team divides the work into different features, then divides it into separate branches for each person and develops the functions of each person in that branch.

The most annoying problem when using github is conflicts. Although we understand it very well on our part, we can fix it, but preventing it is much easier, so we have come up with some tips. General requirements of our team such as

+ Always pull the latest version of the main branch to your branch when preparing code to add new functions

+ Commit git regularly, usually after finishing a feature you have to commit, first commit to your branch then pull the request to main for others to approve, etc.

+ Avoid editing existing code unless you understand it very well

Those rules help us minimize conflicts to help work faster and more effectively

Besides the problem with conflict, it is really convenient and increases our working capacity, some obvious benefits include

+ Work independently, because we have divided the tasks and developed features independently, so we don't need to wait for anyone, github will automatically combine our work when done.

+ Never need to worry about making mistakes when trying new functions or features. With Github's version control, everything you commit has been saved, the old code is not lost, it has become part of the application timeline , you can restore at any time, to any version

+ Maximum cooperation, with the division of branches, although we work independently but are very connected. For example, when one person encounters a bug that cannot be resolved, another person simply needs to switch to the branch where the person is having the bug, and the code will automatically be taken from that person. They can easily discuss and resolve the bug together

### Work contributions and milestone

|  |  |
| --- | --- |
| **Tasks** | **Assignments** |
| Choose topic | All members |
| Choose Platforms and Deploys | All members |
| Design web features and ERD | All members |
| Design web interactions | Đăng, Đồng |
| Add backend servlets | All members |
| Customize front-end | Đăng, Đồng |
| Test and fix bugs | All members |
| Host web online | Thắng |
| Make report | All members |

### Backend servlet contributions

|  |  |  |
| --- | --- | --- |
| **STT** | **Student ‘s Name** | **General description of the job** |
| 1 | Bùi Chiến Thắng | Log in, log out function  Notification function  Bidding function |
| 2 | Nguyễn Văn Anh Đồng | Buyer function  Cart function  Load product function |
| 3 | Đỗ Huỳnh Bảo Đăng | Seller function  Add product function  Contact function |

Our source code is available here: <https://github.com/ThangBuiChien/Auction-Project>

This is our hosting url website: <http://auction2-env.eba-dtjkurmk.us-east-1.elasticbeanstalk.com/>