**Software Design Specifications**

**for**

**Online Auction Systems for the students of MU**

**Prepared by:**

**Dhriti Agarwal, Koganti Sirichandana, Krish Hindocha, Arun Shashank Varma Chekuri, Katepalli Priyadarshini, Akhilesh Sidda, Adla Deepanker Reddy**

**Document Information**

|  |  |
| --- | --- |
| **Title:** Software Design Specification |  |
| **Project Manager:** Dr.Raghu Kishore | **Document Version No:**3 |
|  | **Document Version Date:**28-3-2023 |
| **Prepared By:** Dhriti Agarwal, Koganti Sirichandana, Krish Hindocha, Arun Shashank Varma Chekuri, Katepalli Priyadarshini,Akhilesh Sidda,Adla Deepanker Reddy | **Preparation Date:**30-4-2023 |
|  |  |

**Version History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ver. No.** | **Ver. Date** | **Revised By** | **Description** | **Filename** |
| 1. | 12-2-2023 | Dr. Raghu Kishore | In this template, we are to edit all contents that are in italics. The "statement of work" is for us to decide the key features that we want to develop as part of your product. It helps us to stay focused, so that we don't deviate from our plan of action during subsequent phases of our project. | Statement of Work |
| 2 | 26-3-2023 | Dr. Raghu Kishore | SRS is a must to have document for every project and usually serves as a binding/legal document as to what the customer has requested for and what the developer is going to deliver. The purpose of this task is for you to get insight into things that you should have already thought of before you start design and coding of any project. | SRS |
|  |  |  |  |  |
|  |  |  |  |  |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **1 INTRODUCTION ............................................................................................................................................................** | | **4** |
| 1.1 | PURPOSE ...................................................................................................................................................................... | 4 |
| 1.2 | SCOPE .......................................................................................................................................................................... | 4 |
| 1.3 | DEFINITIONS, ACRONYMS, AND ABBREVIATIONS ........................................................................................................ | 4 |
| 1.4 | REFERENCES ................................................................................................................................................................ | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **USE CASE VIEW ............................................................................................................................................................** | | **4** |
|  | 2.1 | USE CASE ..................................................................................................................................................................... | 4 |
| **3** | **DESIGN OVERVIEW .....................................................................................................................................................** | | **4** |
|  | 3.1 | DESIGN GOALS AND CONSTRAINTS ............................................................................................................................. | 5 |
|  | 3.2 | DESIGN ASSUMPTIONS ................................................................................................................................................. | 5 |
|  | 3.3 | SIGNIFICANT DESIGN PACKAGES ................................................................................................................................. | 5 |
|  | 3.4 | DEPENDENT EXTERNAL INTERFACES ........................................................................................................................... | 5 |
|  | 3.5 | IMPLEMENTED APPLICATION EXTERNAL INTERFACES ................................................................................................. | 5 |
| **4** | **LOGICAL VIEW .............................................................................................................................................................** | | **5** |
|  | 4.1 | DESIGN MODEL ............................................................................................................................................................ | 6 |
|  | 4.2 | USE CASE REALIZATION .............................................................................................................................................. | 6 |
| **5** | **DATA VIEW ....................................................................................................................................................................** | | **6** |
|  | 5.1 | DOMAIN MODEL .......................................................................................................................................................... | 6 |
|  | 5.2 | D ATA MODEL (PERSISTENT DATA VIEW)...................................................................................................................... | 6 |
|  | *5.2.1 Data Dictionary....................................................................................................................................................* | | *6* |
| **6** | **EXCEPTION HANDLING .............................................................................................................................................** | | **6** |
| **7** | **CONFIGURABLE PARAMETERS ..............................................................................................................................** | | **6** |
| **8** | **QUALITY OF SERVICE ................................................................................................................................................** | | **7** |
|  | 8.1 | AVAILABILITY.............................................................................................................................................................. | 7 |
|  | 8.2 | SECURITY AND AUTHORIZATION ................................................................................................................................. | 7 |
|  | 8.3 | LOAD AND PERFORMANCE IMPLICATIONS ................................................................................................................... | 7 |
|  | 8.4 | MONITORING AND CONTROL ....................................................................................................................................... | 7 |

**1 Introduction**

*[The introduction of the Software Design Specifications should provide an overview of the entire Software Design. It should include the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the Software Design Specifications.]*

**1.1** **Purpose**

*[This section defines the purpose of the* ***Software Design Specifications****, in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document should be identified, with an indication of how they are expected to use the document.]*

The purpose of the Software Design Specifications (SDS) document is to provide a comprehensive and detailed overview of the design of the online auction system for college students. The document serves as a blueprint for the development team to understand the functionality and architecture of the system. It also helps to ensure that the system is developed according to the requirements and specifications defined in the project proposal. The SDS document typically includes several sections, such as an introduction, system overview, architectural design, user interface design, data design, and testing requirements. Each section provides specific details on the design and functionality of the system. The primary audience for the SDS document includes the project stakeholders, such as the project manager, development team, and quality assurance team. Other potential audiences may include clients, end-users, and system maintainers. The project manager uses the SDS document to monitor the progress of the development team, ensuring that the system is developed according to the design specifications. The development team uses the document to guide the implementation of the system, while the quality assurance team uses it to develop test plans and ensure that the system meets the required quality standards. The clients and end-users use the SDS document to understand the design and functionality of the system. Additionally, system maintainers use the document to understand the underlying architecture of the system and how it can be maintained and updated in the future.

**1.2 Scope**

*[A brief description of what the* ***Software Design Specifications*** *applies to; what is affected or influenced by this document.]*

The Software Design Specifications (SDS) document for the online auction system for college students applies to the design and architecture of the software system. It outlines the system's functionality, including the user interface design, data design, and implementation details. The SDS document is critical to the development process as it provides a blueprint for the development team to follow when creating the system. It influences how the system is designed, implemented, and tested, ensuring that it meets the project's requirements and specifications. The document is also influential in determining how the system will be maintained and updated in the future. The SDS document provides an overview of the system's architecture and design, making it easier for maintainers to understand how the system works and how it can be improved over time. Overall, the SDS document plays a vital role in the development and maintenance of the online auction system for college students. It influences how the system is designed, implemented, and tested, ensuring that it meets the project's requirements and specifications and is delivered within the proposed timeline and budget.

**1.3** **Definitions, Acronyms, and Abbreviations**

*[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the* ***Software Design Specifications****. This information may be provided by reference to the project Glossary.]*

The following are the definitions of terms, acronyms, and abbreviations used in the Software Design Specifications (SDS) document for the online auction system for college students:

SDS: Software Design Specifications document   
UI: User Interface   
API: Application Programming Interface   
DBMS: Database Management System   
SQL: Structured Query Language  
HTML: Hypertext Markup Language   
CSS: Cascading Style Sheets   
JS: JavaScript   
AJAX: Asynchronous JavaScript and XML   
MVC: Model-View-Controller   
CRUD: Create, Read, Update, Delete   
SSL: Secure Sockets Layer   
HTTP: Hypertext Transfer Protocol   
HTTPS: Hypertext Transfer Protocol Secure

The above definitions are not exhaustive, and additional terms, acronyms, and abbreviations may be included in the SDS document or the project's glossary.

**1.4** **References**

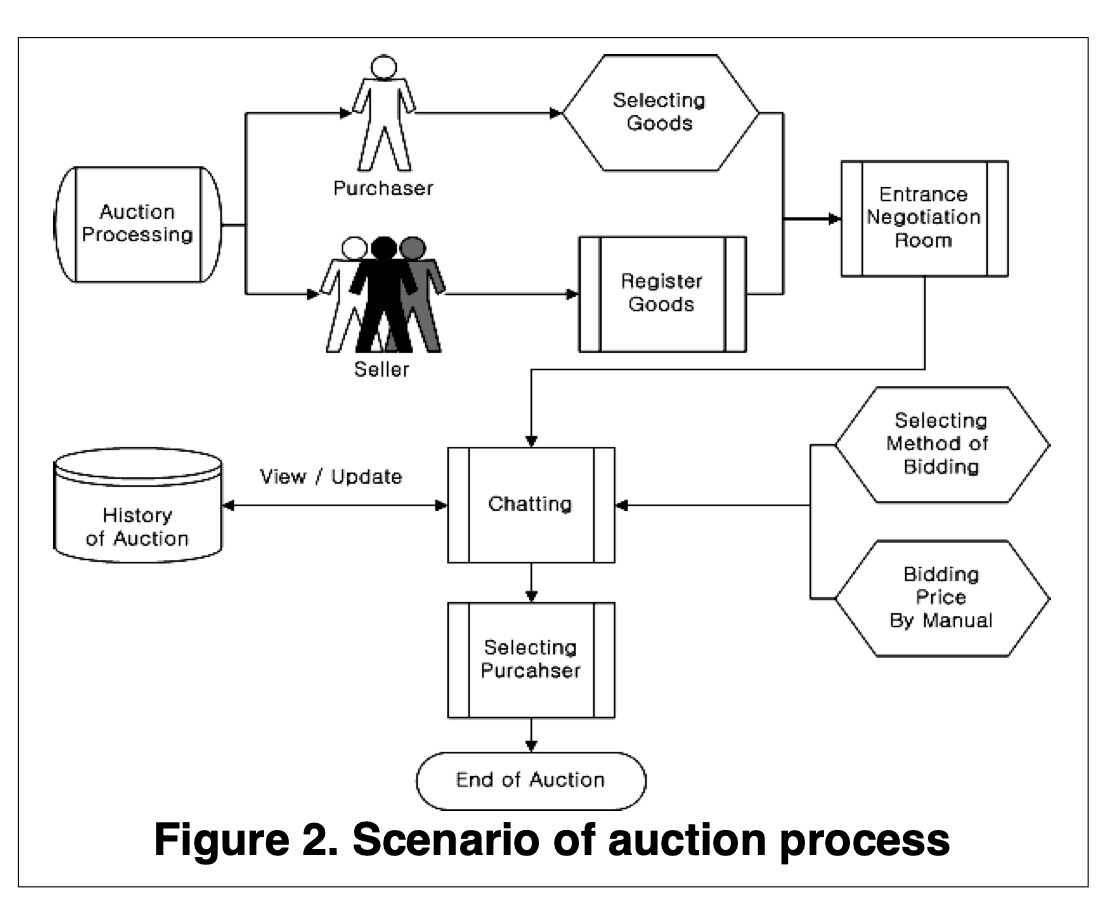
*[This subsection should provide a complete list of all documents referenced elsewhere in the* ***Software Design******Specifications****. Each document should be identified via an URL link to the source document library if available. If**not, specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

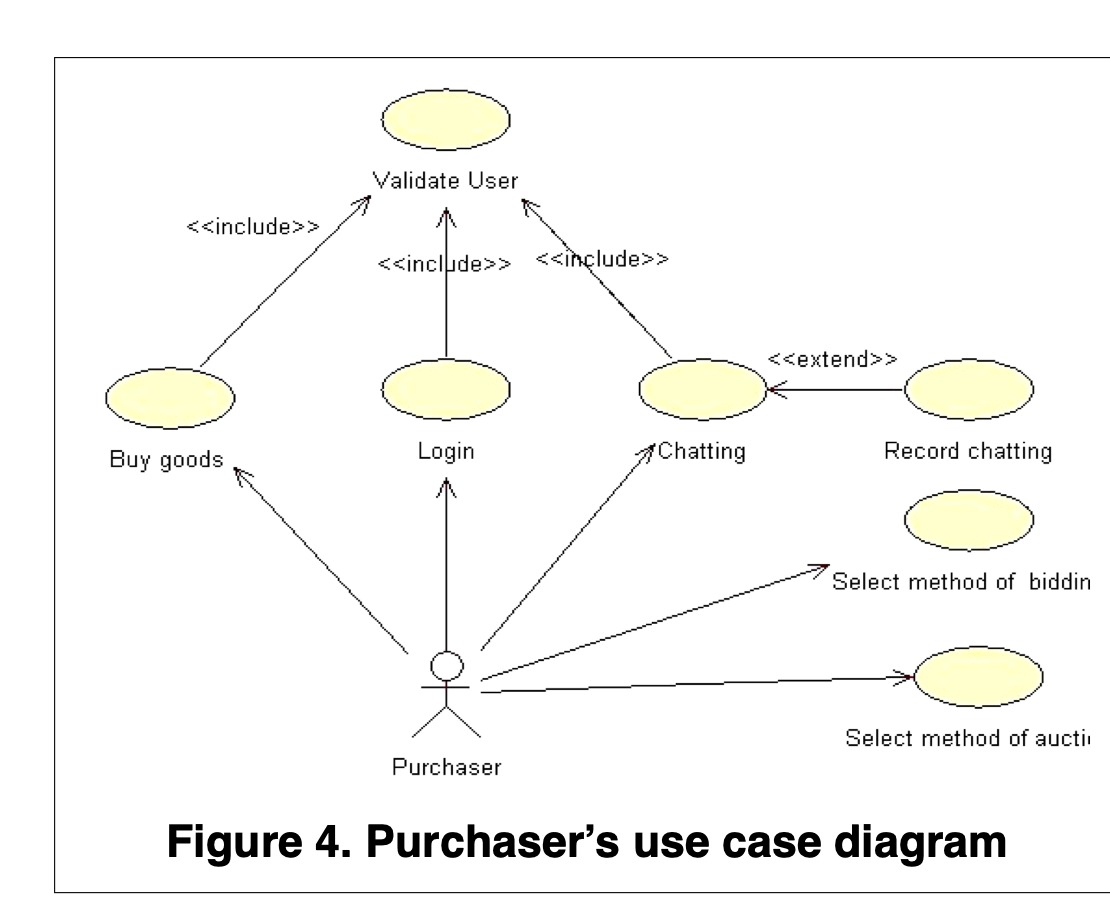
**2 Use Case View**

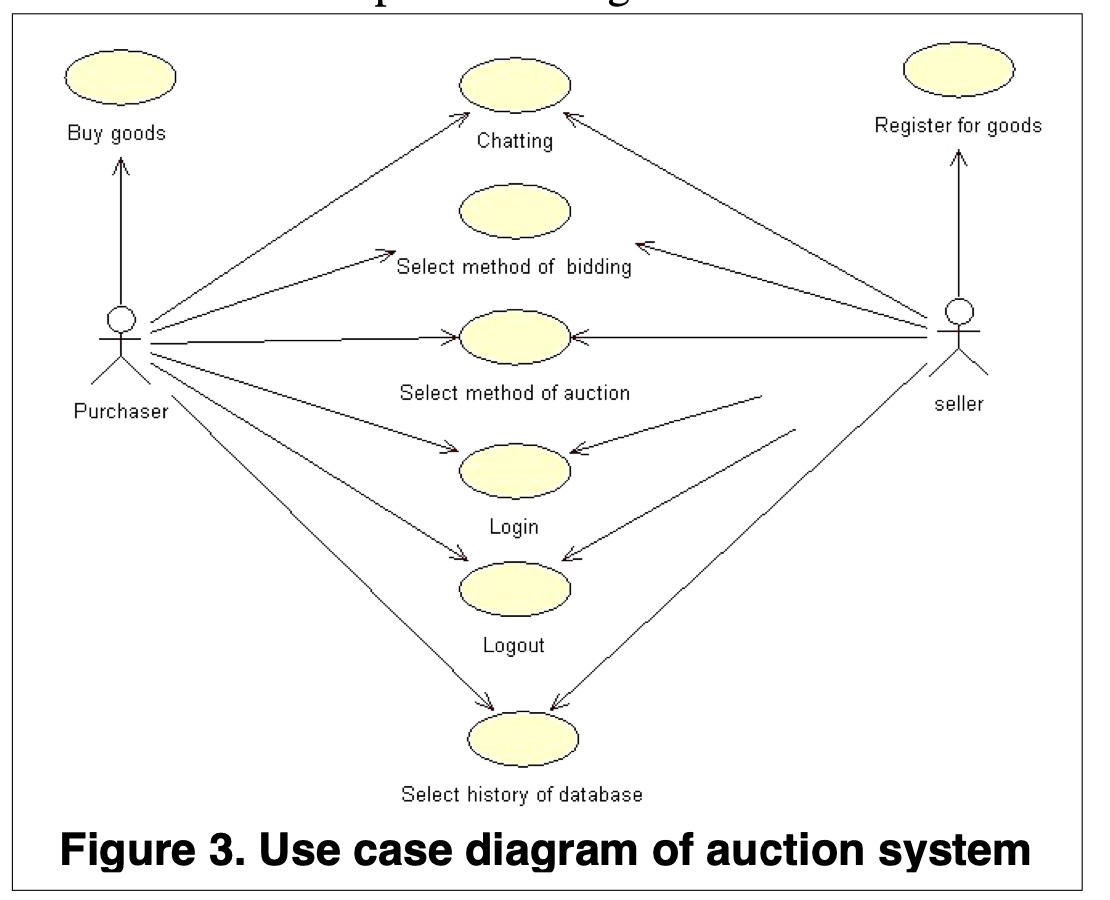
*[Based on the requirements document, this section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large design coverage - they exercise many design elements, or if they stress or illustrate a specific, delicate point of the design. Provide a use case diagram for use cases that pertains to the software design]*

**2.1** **Use Case**

*[If the functional requirements document does not contain sufficient information for a given use case, for each use case pertaining to the design, include a subsection with its name, its brief description, and usage steps or procedures for the particular use case.]*

**

**

**

The Use Case Diagram is a visualization of a use-case, i.e., the interaction between the auction system and the users. Figure 3 is the Use Case Diagram for the actions that the Users (Seller, Purchaser) can perform in an auction. Users, after Login, can select the method of auction (auction, reverse auction) and the method of bidding (Speed, Medium, Leisure). They can also express opinions about products or exchange information about products. Figure 4 is the Use Case. Diagram that represents the Purchaser’s behaviour. It defines the behaviour of the purchaser while participating in an auction after login.

1. **Design Overview**

*[This section provides an overview of the entire software design. It references and complies with the high-level design interface contracts, requirements and high-level module decomposition approach. ]*

**3.1** **Design Goals and Constraints**

*[This section describes the software requirement constraints and objectives that have some significant impact on the design. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]*

The following are the software requirement constraints and objectives that have a significant impact on the design of the online auction system for college students:

-Design and implementation strategy: The system must be designed and implemented using an agile development methodology to ensure that the system's requirements are met, and it is delivered within the proposed timeline and budget.  
-Development tools: The development team must use specific development tools, such as Visual Studio Code, Git, and GitHub, to ensure consistent and efficient development and version control.   
-Team structure: The development team must consist of at least one front-end developer, one back-end developer, and one database developer to ensure that the system is developed according to the required specifications.   
-Schedule: The system must be developed and delivered within the proposed timeline and budget, with milestones and deadlines set to ensure that the project stays on track.   
-Legacy code: The system must be designed and developed in such a way as to ensure compatibility with any existing legacy code.   
-Security: The system must implement secure authentication and authorization mechanisms to ensure that only authorized users have access to the system's data.   
-Scalability: The system must be designed to handle a large number of users and data without impacting its performance and functionality.   
-Usability: The system must be designed to provide an intuitive and user-friendly interface to ensure that users can easily navigate and use the system.

Overall, these constraints and objectives have a significant impact on the system's design and implementation, and they must be taken into account when creating the Software Design Specifications for the online auction system for college students.

**3.2** **Design Assumptions**

*[This section describes assumptions that have significant impacts on the software design.]*

The following are the assumptions that have significant impacts on the software design of the online auction system for college students:

-Users will have access to a modern web browser that supports HTML5, CSS3, and JavaScript.   
-The system will be accessed via a secure internet connection.   
-The system will be developed and deployed on a cloud-based server, such as AWS or Azure. -The system will be developed using popular web development frameworks, such as React.js for the front-end, Node.js for the back-end, and MongoDB for the database.   
-The system will support multiple payment methods, such as credit/debit card payments and PayPal.   
-The system will be designed to handle concurrent user sessions and transactions.   
-The system will be integrated with a third-party API for email notifications and messaging.   
-The system will not be responsible for shipping or delivery of items sold on the platform.  
-The system will not be responsible for verifying the accuracy or authenticity of user-provided information or item descriptions.   
-The system will comply with all relevant privacy and data protection regulations, such as GDPR and CCPA.

Overall, these assumptions have a significant impact on the software design of the online auction system for college students, and they must be considered when creating the Software Design Specifications to ensure that the system meets the project requirements and specifications.

**3.3** **Significant Design Packages**

*[This subsection describes the overall decomposition of the design model in terms of its package dependencies, hierarchy and layers]*

One possible way to organize the packages for an auction website could be:

1. User Interface Package: The components for the user interface of the auction website are all included in the user interface package. The HTML, CSS, and JavaScript files contained within it are in charge of rendering the website's pages on the user's browser.
2. Authentication Package: The website's authentication and authorisation processes are covered by the authentication package. It has elements like registration and login forms, user management, and security controls.
3. Product Management Package: This package includes every element necessary for managing the items up for sale. Products listing, product description, bidding possibilities, and auction administration are some of its constituent parts.
4. Payment Processing Package: This package oversees managing the website's auction's payment processing. Payment gateways, transaction management, and invoicing are some of its components.
5. Search and Filtering Package: All the parts necessary for finding and filtering the auction products are included in the search and filtering package. There are parts like a search bar, filters, and sorting choices.
6. Reporting and Analytics Package: Creating reports and analytics on the functionality of the auction website are the focus of the reporting and analytics package. Dashboards, data visualisation tools, and performance analysis tools are some of its components.
7. Database Package: All the elements necessary for the storage and retrieval of data for the auction website are included in the database package. It consists of elements like the query optimisation, data access layer, and database schema.

We can accomplish modularity, concern separation, and simplicity of maintenance by breaking the design model up into various packages. Changes made to one package should not have an impact on the others, and each package may be separately produced and evaluated. This design strategy also makes it possible for the auction system to be expanded upon and improved in the future.

**3.4** **Dependent External Interfaces**

*[The high level design identified external interfaces that are depended upon by the application. This section indicates where these interfaces are used at the internal module level.]*

The table below lists the public interfaces this design requires from other modules or applications.

|  |  |  |  |
| --- | --- | --- | --- |
| External Application | Interface Name | Module Using Interface | Functionality/ Description |
| Payment Gateway | Authorization API | Payment Processing Module | Initiates transactions and handles refunds |
| Shipping Carrier | Shipping API | Order Fulfillment Module | Generates shipping labels, tracks packages, and communicates with the shipping carrier |
| Identify Verification Service | Identify Verification API | User Authentication Module | Confirms the identity of users who register with the auction site |
| Email and SMS Notification Service | Notification API | Notification Module | Sends messages to users about bids, auctions, and other important events |
| Cloud Storage Provider | Storage API | Media Storage Module | Securely stores images and other files uploaded by users |

**3.5** **Implemented Application External Interfaces (and SOA web services)**

*[The high level design identified interfaces that are provided or owned by the application. This section indicates which internal modules are responsible for implementing these interfaces.]*

The table below lists the implementation of public interfaces this design makes available for other applications.

|  |  |  |
| --- | --- | --- |
| Interface Name | Module Implementing Interface | Functionality/Description |
| Payment Gateway API | Payment Processing Module | Implements the payment gateway API to initiate transactions and handle refunds |
| Shipping Carrier API | Order Fullfillment Module | Implements the shipping carrier API to generate shipping lables,track packages and communicate with the shipping carrier |
| Identify Verification API | User Authentication | Implements the identify verification API to confirm the identify of users who register with the auction site |
| Email and SMS Notification API | Notification Module | Implements the email and SMS notification API to send messages to users about bids, auctions, and other important events |
| Cloud Storage API | Media Storage Module | Implements the cloud storage API to securely store images and other files uploaded by users |

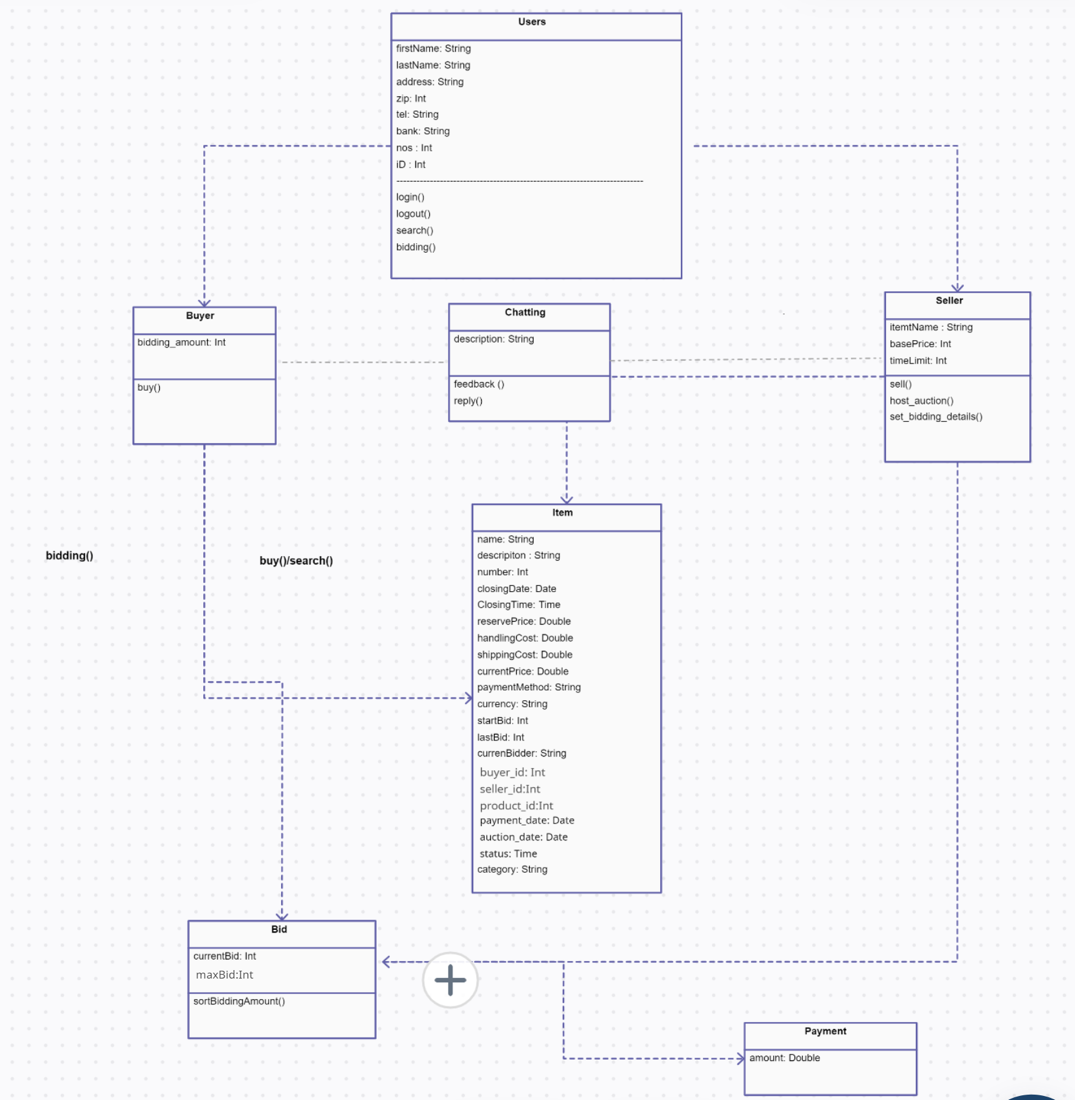
**4 Logical View**

*[This section describes the detailed design. The description should consist of a series of layers. The top most layers should show how the application modules interact to complete the key use cases from Section 2. The more detailed layers should expand each module’s separate interaction responsibility and show how the separate classes within the*

*module collaborate to implement the required behaviour. Depending on project needs, this decomposition may continue until each method in the layer can be described in a page or less of pseudo-code or text.]*

**4.1** **Design Model**

*[This section describes the software model from class design perspective. It decomposes the software into modules and then the significant classes that make up each module. The model should depict significant classes and class utilities that make up the system through one or more class diagrams. For each significant class, describe its responsibilities, as well as important relationships, operations, and attributes.]*



The design model consists of several modules, each of which is responsible for a specific aspect of the auction website. These modules are depicted as packages in the class diagram. Let's take a closer look at each package and the significant classes it contains:

1. User Interface Package: This package contains classes related to the user interface of the auction website, including pages, forms, and UI elements. The significant classes in this package are:

* HomePage: Responsible for displaying the home page of the website.
* ProductPage: Displays the details of a specific product.
* BidForm: Allows users to place bids on products.
* RegistrationForm: Handles user registration and account creation.
* LoginForm: Handles user login and authentication.

1. Authentication Package: This package contains classes related to user authentication and authorization. The significant classes in this package are:

* User: Represents a user account in the system.
* UserManager: Provides functionality for managing user accounts, such as creating, deleting, and updating them.
* AuthenticationManager: Handles user authentication and authorization.

1. Product Management Package: This package contains classes related to managing the products that are being auctioned. The significant classes in this package are:

* Product: Represents a product that is being auctioned.
* ProductManager: Provides functionality for managing products, such as creating, deleting, and updating them.
* Auction: Represents an auction for a specific product.
* AuctionManager: Provides functionality for managing auctions, such as starting and ending them.

1. Payment Processing Package: This package contains classes related to payment processing for the auction website. The significant classes in this package are:

* Payment: Represents a payment transaction for a product.
* PaymentManager: Provides functionality for processing payments and managing payment transactions.

1. Search and Filtering Package: This package contains classes related to searching and filtering auction products. The significant classes in this package are:

* SearchBar: Allows users to search for specific products.
* Filter: Provides filtering options for auction products.
* Sorting: Allows users to sort auction products by different criteria.

1. Reporting and Analytics Package: This package contains classes related to generating reports and analytics on the performance of the auction website. The significant classes in this package are:

* Report: Represents a report on the website's performance.
* Analytics: Provides analytics on the website's usage and performance.
* ReportGenerator: Generates reports based on website data.

1. Database Package: This package contains classes related to the storage and retrieval of data for the auction website. The significant classes in this package are:

* Database: Represents the database for the auction website.
* DataAccessObject: Provides an interface for accessing and manipulating data in the database.
* QueryOptimizer: Optimizes database queries for performance.

Each significant class has its own set of responsibilities, attributes, and operations. For example, the Product class in the Product Management Package has attributes such as name, description, and price, and operations such as create, delete, and update. The User class in the Authentication Package has attributes such as username, password, and email, and operations such as authenticate and authorize.

The relationships between the significant classes are also important. For example, the Auction class in the Product Management Package has a one-to-one relationship with the Product class, indicating that each product can have only one auction. The User class in the Authentication Package has a one-to-many relationship with the Product class, indicating that a user can bid on many products, but a product can only have one highest bidder at any given time.

Overall, the design model for an auction website consists of several modules that work together to provide the functionality needed for the website.

**4.2** **Use Case Realization**

*[For each use case defined in section 2, provide detailed descriptions of how that use case scenario is realized within the design. At the highest level, show the interactions between modules using sequence or activity diagrams that implement each key use case (i.e. the business transaction implementation). For each interaction in the highest level, expand to a lower level sequence or activity diagram that shows how each class within the module collaborates to implement the required behaviour.]*

The Certification component is used to validate the user trying to log into the system.

• A seller enters products into the system by using the RegistrationGood component. At this time, the seller inputs an end date and time of auction, including the starting and end prices of products.

• Purchaser and Seller components manage information related to the auctions of the purchaser and the seller, as well as their private information.

• The Negotiation component manages the auction. If a bidder arrives at the time of the auction close or abidder who suggests the highest price exists, the auction will be closed. When an auction closes, the data record of the auction transfers to the ManagementHistoryAuction component.

• The ManagementHistoryAuction component shows the previous auction record of the auctioneer conducting the current auction.

• The DataBase component saves the relevant data pertaining to the current auction (e.g. the price of products and contents) separately in the database.

• According to the three kinds of bidding methods (Speed, Medium, Leisure), a purchaser decides the next bid after confirmation of the end price that has been suggested so far from the DataBase component using the Method of Bidding component.

Of the eight components developed, the Certification, Registration Good and Method of Bidding components are particularly useful and can be easily adapted for reuse in other systems.

**5 Data View**

*[This section describes the persistent data storage perspective of the system. This section is may not apply if there is little or no persistent data, or the translation between the Design Model and the Domain Model is trivial ]*

**5.1** **Domain Model**

*[Provide the entity model that represents the persistent data and relationships between them. The entities in this model are typically domain objects or data transfer objects.]*

Entities:

1. User - Represents a user of the auction site

- UserID (primary key)

- FirstName

- LastName

- Email

- Password

- Phone

2. Item - Represents an item being auctioned on the site

- ItemID (primary key)

- Name

- Description

- StartingBid

- MinimumBidIncrement

- BidEndTime

- SellerID (foreign key referencing User.UserID)

3. Bid - Represents a bid made on an item

- BidID (primary key)

- Amount

- Time

- BidderID (foreign key referencing User.UserID)

- ItemID (foreign key referencing Item.ItemID)

Relationships:

1. One user can sell many items (one-to-many relationship)

- User.UserID (one) to Item.SellerID (many)

2. One item can have many bids (one-to-many relationship)

- Item.ItemID (one) to Bid.ItemID (many)

3. One user can make many bids (one-to-many relationship)

- User.UserID (one) to Bid.BidderID (many)

**5.2** **Data Model (persistent data view)**

Answer: Sure, here's an example of a persistent data view for the entity model I provided for an online auction site for MU students:

Table: User

| UserID | FirstName | LastName | Email | Password | Phone |

|--------|-----------|----------|-------|----------|-------|

| 1 | Krish | Hindocha | krish.H@mu.edu | abc123 | 555-1234 |

| 2 | Dhriti | Agarwal | Dhriti.A@mu.edu | def456 | 555-5678 |

Table: Item

| ItemID | Name | Description | StartingBid | MinimumBidIncrement | BidEndTime | SellerID |

|--------|-------------|--------------------------------------------|-------------|---------------------|------------------|----------|

| 1 | Textbook | Introduction to Computer Science textbook | 20.00 | 1.00 | 2023-05-01 18:00 | 1 |

| 2 | Laptop | MacBook Pro M1 chip laptop | 800.00 | 50.00 | 2023-05-02 12:00 | 2 |

Table: Bid

| BidID | Amount | Time | BidderID | ItemID |

|-------|--------|---------------------|----------|--------|

| 1 | 25.00 | 2023-04-25 12:00:00 | 2 | 1 |

| 2 | 30.00 | 2023-04-25 12:05:00 | 1 | 1 |

| 3 | 850.00 | 2023-04-26 09:00:00 | 1 | 2 |

| 4 | 900.00 | 2023-04-26 09:05:00 | 2 | 2 |

This persistent data view includes three tables (User, Item, and Bid), which correspond to the three entities in the entity model. The User table includes columns for UserID, FirstName, LastName, Email, Password, and Phone. The Item table includes columns for ItemID, Name, Description, StartingBid, MinimumBidIncrement, BidEndTime, and SellerID (which is a foreign key referencing the UserID column in the User table). The Bid table includes columns for BidID, Amount, Time, BidderID (which is a foreign key referencing the UserID column in the User table), and ItemID (which is a foreign key referencing the ItemID column in the Item table).

**5.2.1** **Data Dictionary**

**6 Exception Handling**

*[This section should describe exceptions that are defined within the application, the circumstances in which they can be thrown and handled, how the exceptions are logged, and the expected follow-up action needed.]*

Exception handling is an important part of any software development project. Exception handling is essential in an auction site since it ensures that the system is reliable and stable even in the face of unforeseen events. Exceptions that might require to be handled on an auction site include the following:

Connection errors: If the auction site depends on other databases or services, it's necessary to manage any issues that may arise when accessing these resources.

Errors in user input validation: In order to avoid storing incorrect data in the system or triggering unexpected behaviour, user input validation is crucial.

Payment processing errors: If the auction site accepts payments, it is important to handle any payment processing issues including failed payment gateways or incorrect payment information.

System failures: Due to hardware problems, network problems, or software defects, the auction site may experience unforeseen faults. The system's ability to handle exceptions will help prevent these mistakes from crashing or erasing data.

Developers incorporate strong error handling methods that can identify, report, and recover from problems in order to manage exceptions on an auction site properly. In order to aid users in understanding the issue and taking the proper action, they should also give them clear error messages.

To find and fix mistakes before they affect users, developers should also think about introducing automated testing and monitoring technologies. By doing so, you can lessen the effects of exceptions and guarantee that the auction site will continue to be dependable and stable.

Follow up actions that may be required to ensure that the system remains stable and reliable are:

Reporting and logging: This can assist developers in figuring out the source of the issue and taking the necessary steps to stop similar problems from happening in the future.

Retry or rollback: For instance, to avoid data inconsistencies, the system may try to retry the payment or rollback the transaction if a payment processing mistake occurs.

Alerting: If a system failure happens, developers may need to be informed so they may look into and quickly fix the problem. When such crucial exceptions take place, alerting methods like email or SMS notifications can be set up to notify the pertinent stakeholders.

Code review and refactoring: They are necessary since exceptions may be brought on by deeper problems with the code. To avoid making the same mistakes again, developers might need to examine the code and make modifications.

**7 Configurable Parameters**

*[This section may be more applicable to J2EE-based systems than Ruby on Rails applications. It should describe parameters used by the application that are configurable. For simple configuration, the name and definition/usage of each parameter should be provided. For complex configuration, the XML schema associated with the configuration should be provided. For each configurable parameter, indication should be made as to which parameters are dynamic (can be changed without restarting the application).]*

This table describes the simple configurable parameters (name / value pairs).

|  |  |  |
| --- | --- | --- |
| **Configuration Parameter Name** | **Definition and Usage** | **Dynamic?** |
| <Parameter name> | <Definition and Usage of Parameter> | <Yes/No> |
|  |
|  |  |  |
|  |  |  |

Sure, here are some additional configurable parameters that could be applicable for an auction website:

1. Bid increment: This parameter would allow the website administrator to configure the minimum amount that each new bid must exceed the current bid by. For example, the administrator could set the bid increment to $5, which means that each new bid must be at least $5 higher than the previous bid. This parameter is dynamic and can be changed without restarting the application.

2. Auction duration: This parameter would allow the administrator to configure the length of time that auctions last. For example, the administrator could set the duration to 7 days for standard auctions, and 24 hours for quick sales. This parameter is dynamic and can be changed without restarting the application.

3. Auction fee percentage: This parameter would allow the administrator to configure the percentage of the final sale price that the website charges as a fee for hosting the auction. For example, the administrator could set the fee percentage to 5%, which means that for an item that sells for $100, the website would charge a $5 fee. This parameter is dynamic and can be changed without restarting the application.

4. Payment methods: This parameter would allow the administrator to configure the payment methods that are accepted for auctions, such as credit cards, PayPal, or bank transfers. This parameter is dynamic and can be changed without restarting the application.

5. Shipping options: This parameter would allow the administrator to configure the shipping options that are available for auctions, such as local pickup, standard shipping, or expedited shipping. This parameter is dynamic and can be changed without restarting the application.

6. Auto-relist option: This parameter would allow the seller to configure whether an auction should be automatically relisted if it doesn't sell. For example, the seller could choose to have the auction automatically relisted up to three times if it doesn't sell. This parameter is dynamic and can be changed by the seller without restarting the application.

These parameters can help tailor the auction website to the needs of the administrator, sellers, and buyers, and improve the overall user experience.

Here are some examples of simple configurable parameters that could be used in an auction website for students:

| Configuration Parameter Name | Definition and Usage of Parameter | Dynamic? |

|------------------------------|----------------------------------|----------|

| StartingBidAmount | The minimum starting bid amount for auctions | Yes |

| BidIncrementAmount | The minimum increment amount for each bid | Yes |

| AuctionDuration | The duration of the auction in days, hours, and minutes | No |

| MaxBidAmount | The maximum bid amount that a bidder can place | Yes |

| AuctionType | The type of auction (e.g. open or sealed bid, reserve or no reserve, timed or live) | No |

| CurrencyType | The type of currency used for bidding and payments | No |

The dynamic parameters in this list are StartingBidAmount, BidIncrementAmount, and MaxBidAmount. These can be changed without restarting the application, allowing for more flexibility in the auction process.

The auction website for students could also have some additional configurable parameters such as:

| Configuration Parameter Name | Definition and Usage of Parameter | Dynamic? |

|------------------------------|----------------------------------|----------|

| BidRestriction | The maximum number of bids a student can place in a single auction | Yes |

| ItemCondition | The condition of the item being auctioned (e.g. new, used, refurbished) | No |

| AuctionFeePercentage | The percentage fee charged on the final sale price of each item | No |

| AuctionHouseRules | The rules and policies for using the auction website | No |

These parameters can help provide more structure and guidance for the auction process, and help ensure a fair and consistent experience for all participants. The dynamic parameter in this list is BidRestriction, which can be adjusted without restarting the application.

**8 Quality of Service**

*[This section describes aspects of the design related to application availability, security, performance, and monitoring and control in production.]*

**8.1** **Availability**

*[This section should reference the availability business requirement for the application. It should highlight any aspects of the design specifically intended to support availability at the required level. It should describe any aspects of the design that can impact availability (mass update, data loading, housekeeping or periodic maintenance that requires downtime).]*

High availability should be ensured via the auction site's architecture, with little downtime required for maintenance or updates. To maintain continuous availability, this may entail the deployment of load balancers, redundant servers, and other strategies.

When traffic is at its highest, as it often is when popular things are up for sale or the bidding is near to end, the site should be able to manage it.

To guarantee that the site can be rapidly restored in the case of a failure or data loss, regular backups and data recovery methods should be in place.

In order to proactively identify and fix issues before they have an impact on users, the website should have monitoring and warning systems in place.

The website should be able to manage sudden surges in traffic or demand with grace.

**8.2** **Security and Authorization**

*[This section should reference the business requirements related to feature and data access security and authorization, and describe the authorization features and qualifier design intended to implement the requirements. This section should also describe any application specific design related to authorization outside of the Authorization framework, and how setup and management of the user access is exposed to those responsible for user administration.]*

Strong security measures should be incorporated into the architecture of the auction site to safeguard user information and stop unauthorised access. This may involve safe login methods, monitoring, and tracking of user behaviour, as well as the encryption of sensitive data.

Based on user roles and permissions, authorization features should be built to limit access to various site functions and data.

The website has to have explicit password regulations, user onboarding and offboarding procedures, and regular security audits for user access control.

High-value transactions or users with privileged access may be subject to additional security procedures, such as two-factor authentication.

It is important to conduct regular security audits and penetration tests to find and fix any possible flaws in the site's conception and execution.

The site should have a policy for regular password changes, with requirements for strong passwords.

The site should have strict access controls to ensure that only authorized personnel can access user data or make changes to the site.

**8.3** **Load and Performance Implications**

*[In the system requirements document for the application, load projections and performance requirements are provided. This section should describe the implications of load and performance requirements on the detailed design components (expected business transaction execution rate, message processing rate, database table growth projections, etc.). Sufficient detail should be provided to support the creation of load and performance test plans.]*

Here are some key load and performance implications to consider:

User experience: Users want websites to load quickly and without interruption and to be responsive and reliable. Unsatisfactory user experiences, and other negative emotions might result from slow loading times, errors, or other performance problems.

Revenue generating: Sites that are slow or unreliable can result in dropped sales, missed bids, and upset users, which can have an effect on income and profitability.

Competitivity: Users have a wide range of options and need quick and dependable websites in today's internet economy. Sites with speed problems or slow load times risk losing visitors to rivals who provide a better user experience.

Infrastructure requirements: Robust infrastructure, comprising servers, databases, and other hardware and software components, are needed for high traffic volumes and complicated functionality.

Scalability: A flexible and adaptive infrastructure that can grow or shrink to meet demand is necessary for scalability.

Testing and optimisation: Regular testing and optimisation can assist to find performance problems, improve site functionality, and make sure the site is quick, responsive, and dependable.

**8.4** **Monitoring and Control**

*[This section should describe controllable processes implemented by the application (message handlers, daemons). This section should also describe measurable values that the application will publish for monitoring from these processes. ]*

Here are some key considerations for monitoring and control in an auction site:

Server and network monitoring: It is necessary to guarantee that the auction site is functioning properly and to identify any problems that might impair it. This may involve keeping an eye on network traffic, CPU and memory consumption, and other performance indicators.

User activity monitoring: Monitoring of user behaviour is necessary to spot any suspicious activity, stop fraud, and make sure that the site is being used in accordance with the terms of service. Monitoring login attempts, bid activity, payment activity, and other user behaviour may fall under this category.

Error monitoring and logging: It is crucial for finding and fixing any problems that can impair the functionality or usability of the auction site. Monitoring for server problems, database errors, and application errors may be part of this.

Content monitoring: Necessary to make sure that it conforms with the terms of service and any relevant laws or regulations. The monitoring of user-generated content, such as product listings, descriptions, and photos, may fall under this category.

Access control: Crucial to ensuring that only authorised users have access to the auction site's features. Role-based access control, user authentication, and authorisation, as well as other security measures, may be included.

Performance tuning: This is for ensuring that the auction site operates as effectively as feasible. The optimisation of database queries, code optimisation, and other site performance improvements may fall under this category.

Disaster recovery: The auction site has a disaster recovery plan in place to ensure that it can resume operations as soon as a calamity, such as a server failure, a natural disaster, or a cyberattack, has occurred.