Software Design Specifications

for

Online Auction system for students to buy and sell their personal items

Prepared by:
ANVITHA SAJJA
SE22UARI147
LASYA PRIYA
SE22UARI186

Document Information

Title:Online auction system	
Project Manager: Anvitha	Document Version No:1.0
	Document Version Date:APRIL 8 2025
Prepared By:Lasya,Sharath,Nithin	Preparation Date:APRIL 8 2025

Table of Contents

1 Introduction

This document provides a comprehensive overview of the software design specifications for the Online Auction System. It outlines the system's architecture, use case realizations, data model, exception handling strategies, and quality of service metrics. It is intended for software developers, project managers, testers, and other stakeholders involved in the development of the system.

1.1 Purpose

The purpose of this document is to define the detailed software design for the Online Auction System. It serves as a guide for implementation and provides traceability back to the functional requirements. Target audiences include:

- Developers (for system implementation)
- Testers (for test planning and case creation)
- Project Managers (for tracking design progress)
- Maintainers (for system maintenance and future updates)

1.2 Scope

This document applies to all components of the Online Auction System including user registration, auction listing, bidding, notifications, payment, and item management. It influences the software development, testing, and deployment processes.

1.3 Definitions, Acronyms, and Abbreviations

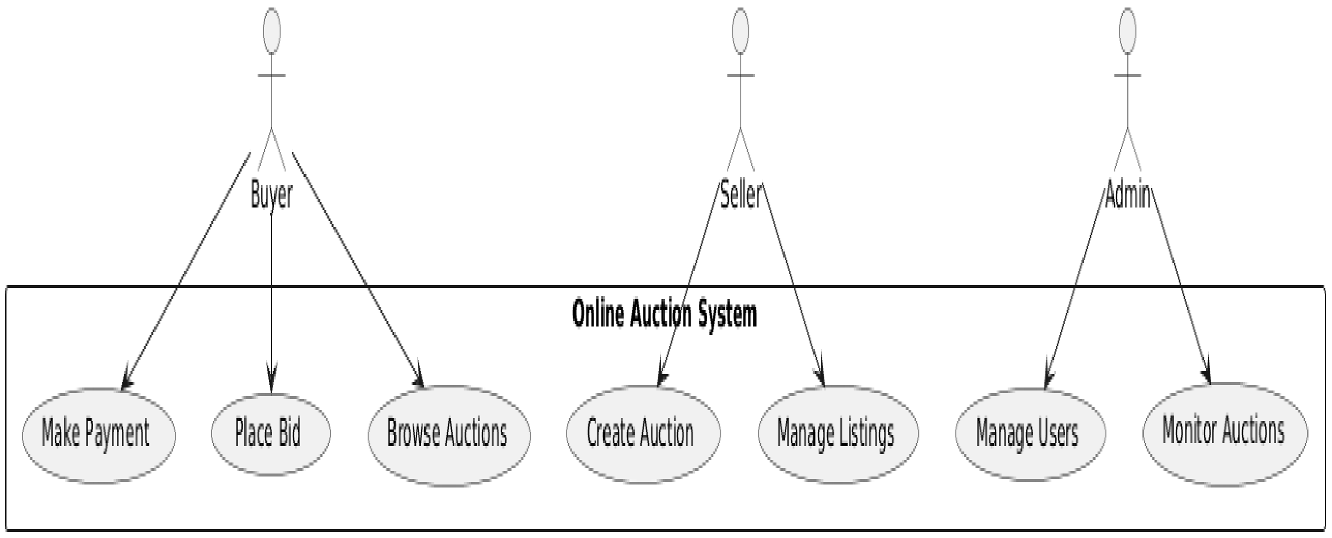
- OAS – Online Auction System
- UI – User Interface
- DB – Database
- API – Application Programming Interface
- UML – Unified Modeling Language
- JSON – JavaScript Object Notation

1.4 References

- Software Requirements Specification Document
- IEEE 1016-2009 Software Design Description Standard
- https://reactjs.org/
- https://expressjs.com/
- https://www.mongodb.com/

2 Use Case View

This section identifies and describes key use cases for the Online Auction System.



2.1 Use Case

Use Case 1: Register/Login

- User signs up or logs in with credentials

Use Case 2: Create Auction

- Seller lists a new item for auction

Use Case 3: Place Bid

- Buyer places a bid on an auction item

Use Case 4: Payment Processing

- Winning bidder completes payment

Use Case 5: Manage Items

- Users manage their listed or won items

3 Design Overview

This section outlines the architectural and modular design of the system.

3.1 Design Goals and Constraints

- Responsive UI using React
- RESTful backend using Node.js/Express
- Secure authentication and authorization
- Scalable and flexible database schema using MongoDB

3.2 Design Assumptions

- Users will have access to internet
- System load will be moderate (100–500 concurrent users)

3.3 Significant Design Packages

- UI Package
- API Services
- Database Models
- Notification Services
- Authentication Services

3.4 Dependent External Interfaces

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

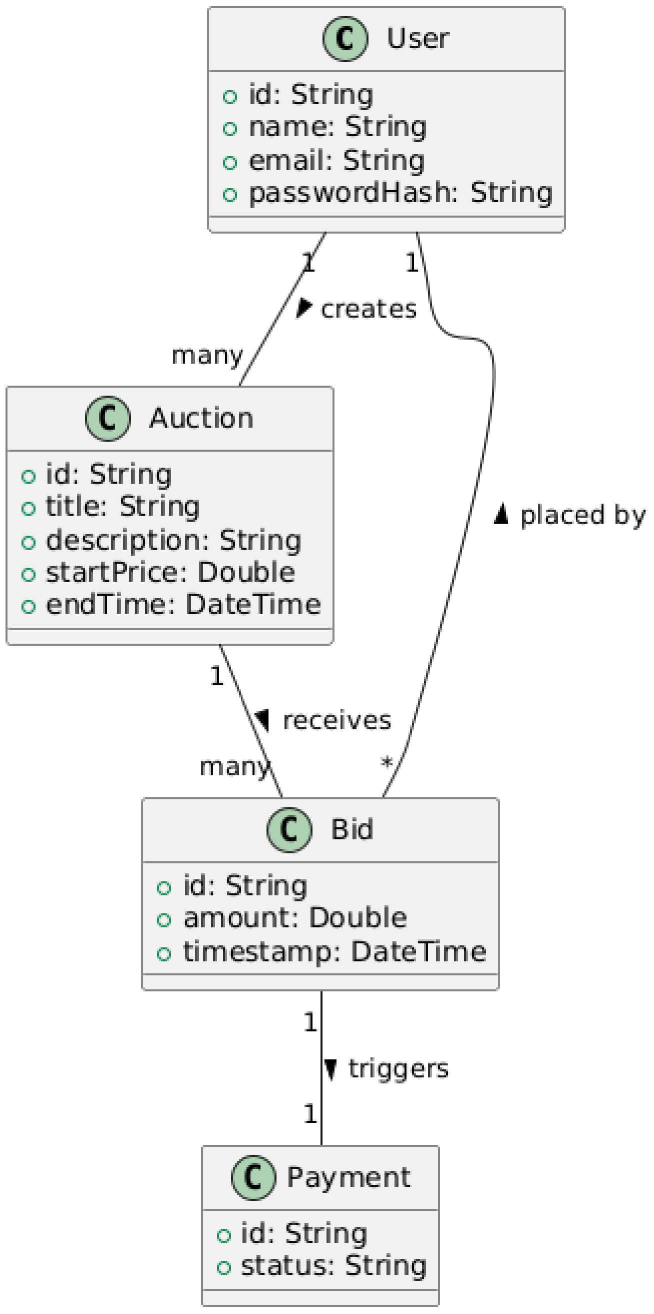
External Application and Interface Name	Module Using the Interface	Functionality/ Description
Payment Gateway	Payment API	Used for processing transactions
Email Service	SMTP Interface	Sends bid status updates

3.5 Implemented Application External Interfaces (and SOA web services)

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

Interface Name	Module Implementing the Interface	Functionality/ Description
User API	Auth Module	Implements registration and login
Auction API	Auction Module	Handles auction creation and bidding

4 Logical View


User
+ id: String
+ name: String
+ email: String
+ passwordHash: String
1
creates
many
1
placed by
Auction
+ id: String
+ title: String
+ description: String
+ startPrice: Double
+ endTime: DateTime
1
receives
many
*
Bid
+ id: String
+ amount: Double
+ timestamp: DateTime
1
triggers
1
Payment
+ id: String
+ status: String

4.1 Design Model

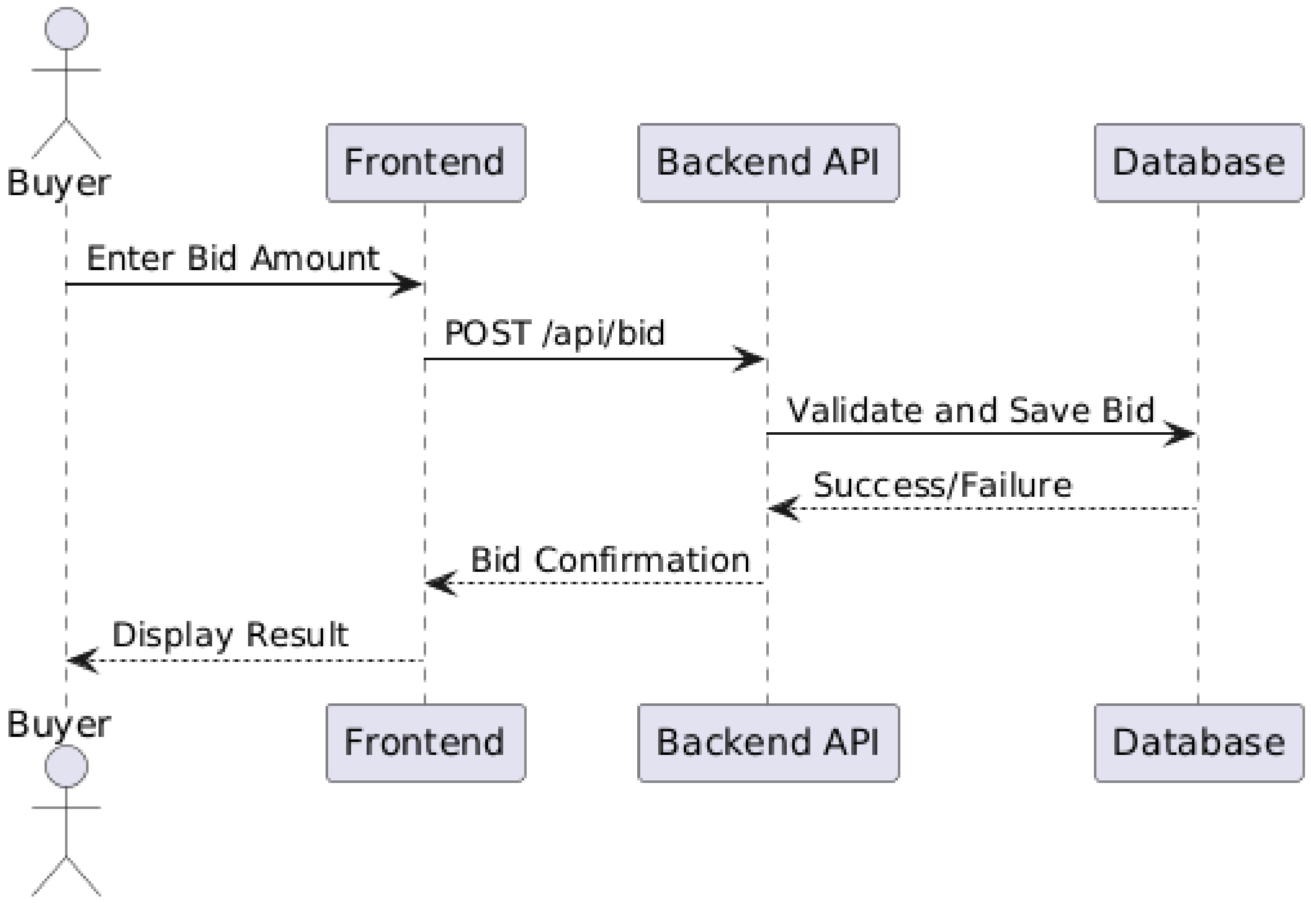
The system consists of the following classes:

- User – handles user data
- Auction – manages auction item details
- Bid – records bids placed
- Payment – manages transactions

4.2 Use Case Realization

Create Auction Use Case Realization:

- UI collects item data
- Sends POST to /api/auctions
- Server validates and saves to DB
- Returns auction ID



5 Data View

5.1 Domain Model

Entities:

- User(id, name, email, passwordHash)
- Auction(id, userId, title, description, startPrice, endTime)
- Bid(id, auctionId, userId, amount, timestamp)
- Payment(id, bidId, userId, status)

5.2 Data Model (persistent data view)

MongoDB collections:

- users
- auctions
- bids
- payments

5.2.1 Data Dictionary

userId	Unique identifier for the user
auctionId	Unique identifier for the auction item
amount	Bid amount

6 Exception Handling

Exceptions:

- Invalid LoginException
- BidTooLowException
- PaymentFailureException
- AuctionClosedException

All exceptions are logged using a centralized logger and user-friendly messages are returned.

7 Configurable Parameters

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

Configuration Parameter Name	**Definition and Usage**	**Dynamic?**
MAX_BID_AMOUNT	Limits maximum bid	Yes
SESSION_TIMEOUT	User session expiration time	No

8 Quality of Service

8.1 Availability

The system targets 99.5% uptime with periodic maintenance during off-peak hours.

8.2 Security and Authorization

- JWT-based authentication
- Role-based access control for admin and users

8.3 Load and Performance Implications

- Designed for 100-500 concurrent users
- Indexed fields in MongoDB for fast queries

8.4 Monitoring and Control

- Server logs monitored using PM2 and Loggly
- Real-time error notifications using webhook alerts