International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

AuctionOasis: A Scalable Web-Based Platform for Real-Time Live Auctions

Yash Sakhareliya
Dept. of Information Technology
Parul Institute of Engineering and Technology
Parul University, Vadodara, Gujarat, India
sakhareliyayash@gmail.com

Abstract— Auction systems have become increasingly popular as the uptake of e-commerce grows globally. However, conventional auction systems may be inflexible and unable to accommodate several bidders simultaneously. To address these issues, AuctionOasis provides a modular and comprehensive web platform that incorporates real-time bid processing, auction management, and secure participation of users. The platform is developed using Node.js, Express.js, MongoDB, and EJS for front-end rendering.Further, the system is planned to implement the Socket.io technology for conducting group live bidding and chatting. This document discusses the motivation behind developing AuctionOasis, its architectural framework, design aspects, implementation process, and future directions.

Keywords—Online auctions, Real-time bidding, Node.js, Express.js, MongoDB, Socket.io, WebSockets, Restful API, Scalability, Session authentication.

I. INTRODUCTION

The development of e-commerce led to an evolution of exchanges. Online auctions have emerged as effective tools to identify fair price levels. Such services give similar opportunities to buyers and sellers, encouraging their honest involvement in the market. However, the success of such mechanisms depends much on timeliness. Users rely on receiving updates on bids, correct counting down of time, and proper processing of several bids simultaneously without errors.

Existing sites like eBay and OLX demonstrate the potential of auctions in the virtual space, although they do not provide developers with enough options for customization. It is evident that there is a need for flexible auctioning systems that can be used in specific situations, for instance, sales within institutions, charity events, and trading rare items.

AuctionOasis acts as a full-stack web application, where users can create their own lists, join the auction process, and receive live information about bids. The application is based on Node.js, Express.js, MongoDB, and EJS as front-end, providing the opportunity to conduct safe auctions with session-based authentication. This paper introduces AuctionOasis both as an existing system and an architectural concept. Specific attention will be paid to issues of concurrent processing and possible directions for further improvement with the integration of Socket.io.

II. LITERATURE REVIEW

Online auction systems research is an interdisciplinary field involving distributed computing, human-computer interaction, and e-commerce. The pioneering works of David Lucking-Reiley provided fundamental economic aspects related to Internet auctions. He explained how the auction system implemented on the Internet could be as efficient, if not more so, than traditional auctions. Participation and fast information dissemination were critical to the operation of Internet auctions.

Technically speaking, the initial Internet auction systems relied on the mechanism of updating the webpage through page refreshing. This led to greater latencies and server utilization. The emergence of the WebSocket protocol helped overcome these issues as well as provide two-way communication between clients and servers. Studies conducted by Thiago Pimentel and Jeffrey V. Nickerson proved the potential of WebSocket as a platform for reducing communications latencies, thus making it suitable for real-time applications.

Node.js has proven to be a popular choice for running these applications due to its event-driven and non-blocking I/O design. The capacity of Node.js to handle thousands of simultaneous connections has been demonstrated through empirical studies conducted by Stefan Tilkov and Steve Vinoski. This characteristic is important during the peak bidding period. Additionally, MongoDB has an adaptive document-oriented database schema that suits the nature of auctions, as stated by Rick Cattell.

Based on the technologies used in the development of Socket.io, it forms a suitable foundation for providing real-time communication and fallback capabilities. It ensures the delivery of timely information despite variations in network performance. Earlier works have verified that Socket.io guarantees the timely delivery of messages to multiple clients. While there has been extensive literature on these topics, there have not been sufficient examples of open-source solutions incorporating these technologies in one place. AuctionOasis is a step forward toward achieving this goal.

III. MATERIALS AND METHODS

Development of AuctionOasis follows a requirement-based approach in an iterative process. The design of AuctionOasis has been based on the functional requirements, which have been identified after a thorough analysis of the workflow involved in an online auction process wherein the seller makes an offer and the bidders bid until the timer runs out.

A. Technology Stack and Data Sources

The following technologies form the backbone of AuctionOasis:
The primary technologies applied in the process of creating AuctionOasis are included in a solid full-stack architecture. On the backend side, Node.js is employed together with Express.js to create an application with routing capabilities. This combination is effective in implementing an application that creates RESTful APIs and allows adding various tools such as sessions, requests, and more through npm modules. MongoDB is used as a database to store the data in documents, which is flexible when working with auction-related information, and Mongoose facilitates the creation of schemas and data validation.
On the client side, EJS serves as a templating engine responsible for rendering dynamic HTML content on the server side. It allows generating personalized web pages such as listing auctions and dashboard. HTML, CSS, and JavaScript are employed on the client-side side for constructing a functional user interface.

B. Design and Implementation Steps

The first phase of system development was requirement analysis, wherein functionalities such as registering and logging into the account, creating auctions, viewing auctions, making bids on auctions, displaying real-time timers, and maintaining auction closure and bid history were recognized. The above-listed requirements served as the blueprint for designing the system architecture.

The MongoDB collections used in this system included Users, AuctionItems, and Bids, storing all required data about users, auction items, and bids made on them. All relationships between the entities have been maintained through Mongoose on the application level.

The RESTful API architecture was adopted to implement the system. The routes for /auth, /auctions, and /bids were created, and middleware to authenticate sessions was implemented. In order to address concurrent bidding, server-side validation has been implemented such that only those bids that are greater than the previous highest bid will be accepted. This can be achieved using an atomic update query in the database, which will avoid race condition. Another background process will regularly check for expired auctions, close them, and assign them a winner automatically.

IV. SYSTEM DESIGN AND ARCHITECTURE

The three-tier architecture used by AuctionOasis includes Presentation Tier, Application Tier, and Database Tier. Each tier plays a specific role in this model, making the system maintainable and upgradable separately.

A. Use Case Analysis

AuctionOasis participants include:
The Bidder: who sets up their account, browses through the various auctions, places bids on products they wish to purchase, monitors the remaining time and top bid placed for each auction, and checks their bid history.
The Seller: who begins auctions by entering data for their product, sets the initial price and how long the auction will run, watches bids placed on their products, and gets alerts when their auction ends.

B. System Architecture

For the Presentation Layer, there are the EJS templates provided via Express.js, augmented by JavaScript for the dynamic parts like bid forms and countdown timers.
The Application Layer holds the Express.js route handlers and controllers. The route handlers check the request validity, call the corresponding actions on Mongoose models, and render views or return JSON data. Authentication middleware will intercept the calls to the restricted routes and redirect unauthorized users back to the login page.

C. Data Models

Entity Relationships for AuctionOasis are summarized below:
User – id, username, email (Unique), passwordHash, createdAt.
AuctionItem – id, sellerId (References User), title, description, imageUrl, startingPrice, currentHighestBid, endTime, status(active | closed), createdAt.
Bid – id, auctionItemId (References AuctionItem), bidderId (References User), amount, placedAt.

V. RESULTS AND DISCUSSION

AuctionOasis was evaluated in an experimental setup resembling actual auction conditions involving fast multiple bid submissions, timer synchronization, and auction closing validation. The following findings were noted.

A. Results (Informal)

Bid Consistency During Concurrent Bids: Multiple bid submissions by concurrent users using different web browsers validated that the atomic transaction approach in MongoDB effectively protected against double acceptance of bid submissions. In all the experiments, only a single bid from the concurrent pair was processed, and the user who submitted the rejected bid was informed about the rejection because another user had posted a better bid.

Accuracy in Auction Closing: The automated auction closing process identified and closed all auctions within one scheduler period (one minute) after the completion of their expected time. Winners were accurately marked by linking them with the highest bid document of the auction that was closed.

User Experience: Test users found the interface easy to navigate and use. They commented positively on the countdown timer on the auction details page.

Session Fixation and Session Data Leakage: There was no evidence of session fixation or session data leakage during the test process. Session termination upon logout proved successful, and further access attempts using the same browser redirected users to the login page.

B. Discussion

The findings confirm the appropriateness of several major architectural choices of AuctionOasis with respect to bid manipulation and system stability. Firstly, the choice of employing server-side session handling along with atomic MongoDB operations allows creating an adequate basis for working with concurrent bids without issues. Moreover, even though rendering via EJS is not that dynamic compared to single-page applications, it is still effective for this particular case and has many benefits, including simplicity and good server-side SEO.

Nonetheless, there are certain drawbacks of AuctionOasis that should be considered. For example, bids need refreshing or polling with AJAX calls, which means delays for customers; this issue will be addressed via adding support for Socket.io. Another drawback is related to slightly delayed auction closing implemented through the scheduler; it would be better to optimize the process with the help of event-driven timers or triggers within MongoDB. Finally, AuctionOasis does not feature any sophisticated capabilities, such as recommendation engines. In the future, the application might include embedding-based similarity searches via vector databases.

VI. CONCLUSION

AuctionOasis demonstrates that a modular, full- Through this project, it has been proven that a full stack web application developed using open source software technology can fulfill all of the basic requirements of an online auction website such as safe participation of users, effective bidding system, automatic management of auctions and responsive design. The architecture of the system ensures proper segregation between its presentation layer, logical and database layers which make it much easier to understand and improve upon.
The most important point that comes out of this project is that even real-time auction features can be implemented without having to deploy a complex distributed system. It has been proved that a good monolithic architecture along

with the use of atomic transactions and intelligent background processing in MongoDB is enough for a decently scaled auction system. Future enhancements will be focused on adding the real-time capabilities of socket.io to update bids instantly and add chat feature to the system.

Acknowledgment
The author would like to acknowledge the efforts of faculty members and colleagues from the Department of Information Technology, Parul Institute of Engineering and Technology, who encouraged him and gave valuable comments while designing AuctionOasis. He is particularly grateful to the developers of the open source libraries used in the design of this auctioning site and to the initial users, who provided feedback on the user interface and the process of bidding.

REFERENCES

1. D. Lucking-Reiley, "Auctions on the Internet: What's Being Auctioned, and How?" Journal of Industrial Economics, vol. 48, no. 3, pp. 227–252, 2000.
2. V. Pimentel and B. G. Nickerson, "Communicating and Displaying Real-Time Data with WebSocket," IEEE Internet Computing, vol. 16, no. 4, pp. 45–53, 2012.
3. S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," IEEE Internet Computing, vol. 14, no. 6, pp. 80–83, 2010.
4. R. Cattell, "Scalable SQL and NoSQL Data Stores," ACM SIGMOD Record, vol. 39, no. 4, pp. 12–27, 2010.