Lebanese American University

COE 418 – Database Systems

Section 12

Dr. Helen Saad

Database Final Project Report

Moussa El Shami 202302471

Mohammad El Sinn 202302817

Hadi Soubra 202307632

20/12/2025

Table Of Contents

Table of Figures

Introduction

This document is the project report of the Online Auction Platform (The Final Bid), including explanations related to the website development, database definition and manipulation, and the interaction between both components. This project consists of developing a database and related software applications to manage the information requirements of an online auction system. An auction platform typically maintains a database containing details about listed items, hosts, bidders, bids, and orders. The system allows users to visualize items based on categories or keywords, place bids in real time, and automatically determine auction outcomes upon expiration. Additionally, the database system handles order creation, payment processing, and shipping information, while notifying users of auction results and status updates. The platform also incorporates real-time communication mechanisms to track bidding activity and manage items as they are listed, sold, or closed.

2. Background

2.1. Context and Problem Analysis

The analysis of the objective at hand was carried out by first determining the set of functionalities that the database system should be able to provide to the different types of users interacting with the online auction platform, namely hosts and bidders. Defining these roles was essential to clearly separate responsibilities and ensure proper access control within the system.

The host is:

1. Responsible for listing items for auction by providing detailed descriptions, setting initial prices, selecting appropriate categories, and defining auction end times.
2. Able to monitor the status of active and completed auctions.

The bidder can:

1. Browse for auction items based on various criteria such as category, keywords, or auction status.
2. Place bids on active auctions while respecting system constraints, including bid validity and auction timing.
3. Track their bidding activity by viewing ongoing auctions, previously placed bids, and completed auctions.
4. Complete the checkout process for won items by selecting a preferred payment method and shipping address.

The analysis and implementation of this project are based on the above-mentioned functionalities, which serve as the foundation of the system design and are further exploited and discussed in the following sections of this report.

2.2. Information Needs

The information needs of the auction platform include data related to auction items, mainly the item name, description, category, initial price, current price, auction end time, and item status. Additional information is required regarding users, including both hosts and bidders, such as their names, email addresses, login credentials, and contact details. The system must also store information related to bids, including bid amounts and timestamps, to accurately track auction activity. Furthermore, data concerning orders, payment methods, and shipping addresses is required to support transaction processing, order completion, and delivery management.

3. Proposal

3.1. Conceptual Data Model (CDM)

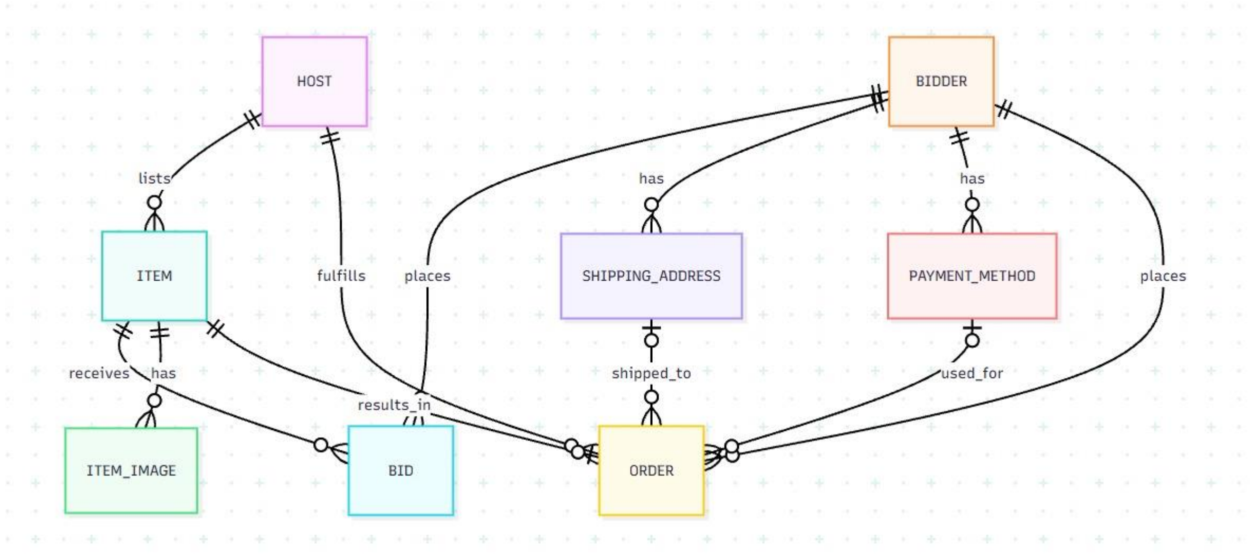

Figure 1 - Conceptual Data Model (CDM)

The conceptual data model as seen from Figure 1 illustrates the main entity types involved in the online auction platform, namely Bidder, Host, Item, Bid, Order, Payment Method, Shipping Address, and Item Image. These entities represent the core components of the auction system and capture the essential business concepts required to support the auction lifecycle.

The Bidder entity represents users who participate in auctions by placing bids and completing purchases, while the Host entity represents users responsible for listing items and fulfilling completed orders. The Item entity models auctioned products, including their auction status and lifecycle, whereas the Bid entity represents the bidding activity between bidders and items. The Order entity is used to represent completed auctions that result in a successful sale. The Payment Method and Shipping Address entities store the necessary information to support checkout and delivery, while the Item Image entity allows associating visual representations with auction items.

The relationships between these entities define the system's business rules, such as a host listing multiple items, bidders placing bids on items, items resulting in at most one order, and orders being associated with optional payment and shipping information. These relationships and their cardinalities are represented in the conceptual data model and serve as the basis for the logical database design.

3.2. Logical Data Model (LDM)

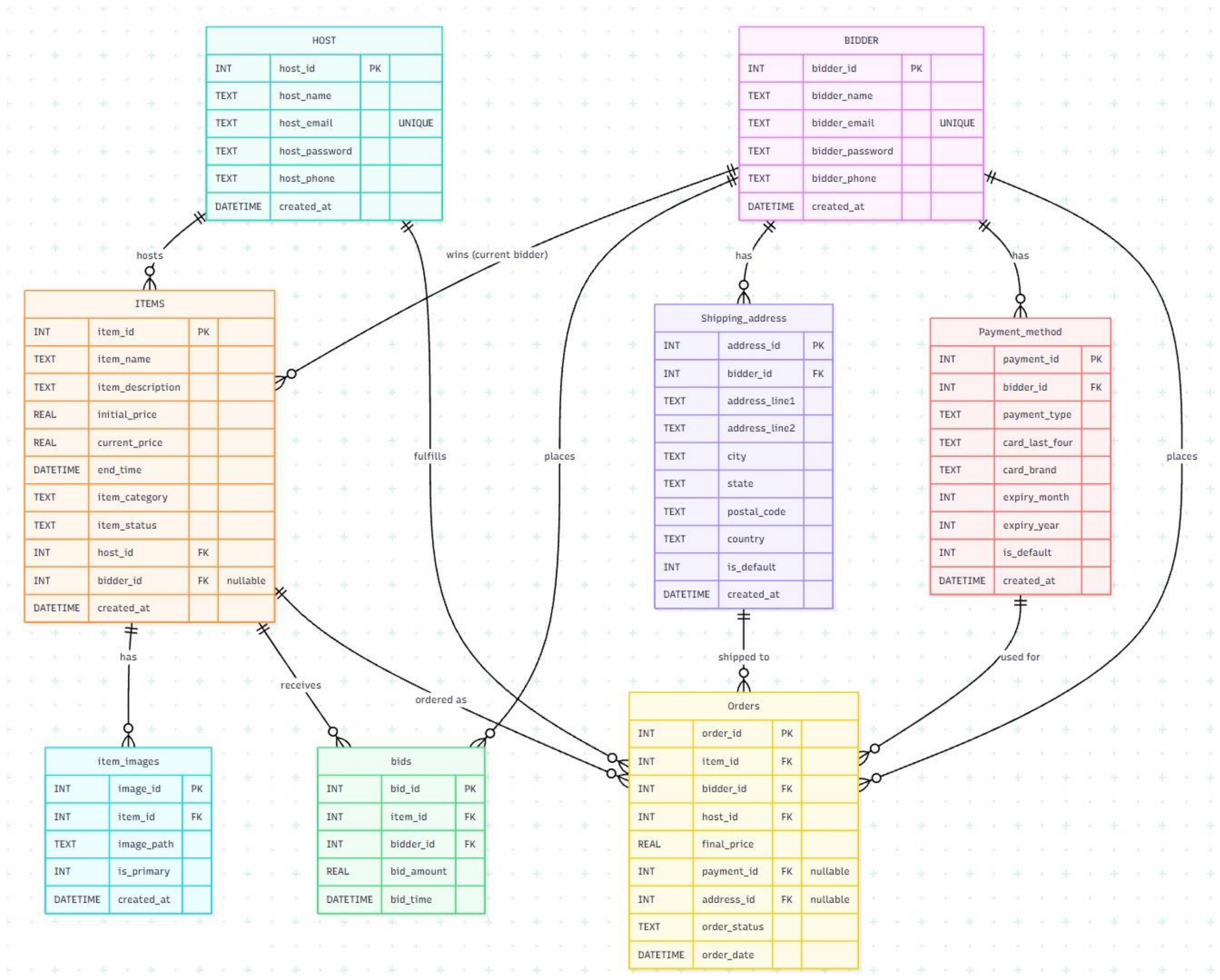

Figure 2 - Logical Data Model (LDM)

The logical data model was obtained by transforming the conceptual data model into a relational representation following the principles discussed during the course. This model introduces structured relations with clearly defined attributes and relationships while preserving the constraints and cardinalities defined in the conceptual model as seen from Fig 2.

In the logical data model, each entity from the conceptual model is mapped to a corresponding relation, with primary keys used to uniquely identify records and foreign keys used to establish relationships between tables. For example, relationships such as bidding activity, order placement, and item ownership are represented using foreign key associations. Nullable attributes are introduced where relationships are optional, such as payment methods or shipping addresses associated with an order. This logical representation ensures data consistency, minimizes redundancy, and supports efficient querying and transaction handling within the system.

3.3. Software Application and Database Design and Implementation

The tools that were used in this project are the following:

1. Node.js for server-side application development.
2. Express.js for building the RESTful API and handling HTTP requests.
3. SQLite as the relational database management system.
4. Socket.IO to enable real-time bidding and auction updates.
5. JSON Web Tokens (JWT) for user authentication and authorization.
6. HTML/CSS for structuring and styling the web interface.

Node.js provides a runtime environment that allows the development of scalable server-side applications [2], while Express.js simplifies routing, middleware handling, and request processing [3]. SQLite was chosen as the database management system due to its lightweight nature and ease of integration with Node.js [5].

The software application was designed to tightly integrate the database layer with the backend logic to support the full auction lifecycle. This includes user registration and authentication, item listing, bid placement, and order creation upon auction completion. Socket.IO was used to provide real-time communication between the server and clients, allowing bids to be broadcast instantly and auction results to be communicated without requiring page refreshes [4].

Additionally, JWT-based authentication was implemented to enforce role-based access control, ensuring that only hosts can list items and fulfill orders, while bidders can place bids and complete purchases [1]. The database is accessed through structured queries to store and retrieve data related to users, items, bids, orders, payment methods, and shipping addresses. This design ensures data consistency, security, and efficient interaction between the application and the database.

From a software design perspective, the application follows a modular architecture where the backend is divided into several logical components. The authentication component provides functions such as registerUser() and loginUser() to manage user access and role-based authorization. The item management component includes functions such as createItem() and getItems() to allow hosts to list auction items and users to browse them. Bidding functionality is handled through functions like placeBid() and getItemBids(), which validate bids and manage real-time updates. Finally, the order management component includes functions such as createOrder() and checkoutOrder() to handle auction completion, payment selection, and order fulfillment. This structure ensures a clear separation of concerns and an organized application design.

4. Experimental Evaluation

This section presents the experimental evaluation of the online auction platform by testing the system through both expert and non-expert user interactions. The evaluation aims to verify the correctness of database operations, the enforcement of business rules, and the overall performance and reliability of the application.

4.1. Testing queries for expert user interface

The expert user interface refers to operations executed directly at the database or backend level without relying solely on the graphical user interface. These operations were tested to ensure that database manipulation, integrity constraints, and automated auction processing behave correctly. For example, when a host lists a new item, the backend inserts a corresponding record into the database. A simplified version of the database operation is shown below:

```
INSERT INTO ITEMS (item_name, initial_price, end_time, host_id)
VALUES ('Watch’, 100, '2025-01-15 18:00:00', 3);
```

To evaluate auction completion, the system periodically executes queries to detect expired auctions. When an auction ends, the highest bid is retrieved, and the item status is updated accordingly. If a valid bid exists, an order is created automatically. The following operations were verified during testing:

```
SELECT MAX(bid_amount)
FROM BIDS
WHERE item_id = 5;

UPDATE ITEMS
SET item_status = 'sold'
WHERE item_id = 5;

INSERT INTO ORDERS (item_id, bidder_id, final_price)
VALUES (5, 7, 350);
```

The correctness of these operations was evaluated by comparing the database state before and after execution. All automated operations completed successfully without violating foreign key constraints or data integrity rules.

4.2. Testing queries for non-expert user interface

The non-expert user interface corresponds to actions performed by hosts and bidders through the web application. To evaluate this interface, various real-world usage scenarios were tested.

For example, a bidder placing a bid on an active auction triggers backend validation to ensure that the bid amount is higher than the current price. If valid, the bid is inserted into the database and broadcast in real time. Conceptually, this process is translated into the following operation:

```
INSERT INTO BIDS (item_id, bidder_id, bid_amount)
VALUES (5, 7, 350);
```

Invalid bids, such as bids lower than the current price or bids submitted after auction expiration, were correctly rejected. During testing, the response time for bid submission and real-time update propagation was observed to be within a few seconds, ensuring a responsive bidding experience.

Another tested scenario involved completing the checkout process after winning an auction. When a bidder selects a payment method and shipping address, the order record is updated accordingly:

```
UPDATE ORDERS
SET payment_id = 2, address_id = 4, order_status = 'paid'
WHERE order_id = 10;
```

Attempts to complete checkout without valid payment or shipping information were rejected, confirming that business rules are properly enforced at the application and database levels.

4.3 Highlighting and Testing All System Functionalities

To evaluate the system, all major functionalities were tested under different scenarios, including concurrent bidding by multiple users, auction expiration handling, and repeated retrieval of active items and orders.

Performance-related metrics were observed during testing. The time required to detect auction expiration and generate an order was consistently within a few seconds after the auction end time. Query response times for retrieving active items, bids, and orders remained stable, and no data inconsistencies were observed during concurrent bidding sessions.

Based on the experimental results, the system demonstrated correct functionality, reliable performance, and consistent enforcement of business rules. The evaluation confirms that the online auction platform operates as expected and is capable of handling typical user interactions efficiently and accurately.

One significant challenge encountered during implementation was ensuring correct auction expiration handling while multiple bidders were actively placing bids. This was resolved by periodically checking auction end times on the server side and enforcing strict bid validation rules to reject late bids. This approach ensured data consistency and correct order generation even under concurrent bidding scenarios.

Conclusion

This was the report of the Database Systems course final project entitled “The Final Bid”. The different functionalities, along with possible improvements, were described throughout the report. From our personal experience perspectives, this project was both challenging and rewarding, as it required the application of database design concepts, backend development, and real-time system integration. Through this project, practical experience was gained in working with technologies such as Node.js, Express.js, SQLite, and Socket.IO, as well as in designing and implementing a complete database-driven web application.

References

[1] Auth0. (2024, November 30). *JSON Web Token Introduction - jwt.io*. JSON Web Tokens - Jwt.io. https://www.jwt.io/introduction

[2] *Node.js v20.2.0 Documentation*. (n.d.). Nodejs.org. https://nodejs.org/docs/latest/api/

[3] OpenJS Foundation. (2017). *Express - Node.js web application framework*. Expressjs.com. https://expressjs.com/

[4] Socket.IO. (n.d.). *Introduction | Socket.IO*. Socket.io. https://socket.io/docs/v4/

[5] SQLite. (2019). *SQLite Documentation*. Sqlite.org. https://www.sqlite.org/docs.html