Documentation

Nikhil Biyani 2020A3PS0776P Mitul Chadha 2020A7PS1510P

Live Auction Application

**April 9th, 2023**

# Instructions (Problem Statement)

The Live Auction Application is a web-based system that allows multiple users to participate in live auctions for various items simultaneously. The application has two primary views, one for the user and another for the administrator. The user view allows users to view and bid on the available items for auction. The administrator view allows the administrator to add new items for auction, set the starting bid, and monitor the bids placed by users.

The application requires users to register and log in to participate in the auction. Users can view all the available items for auction and the current bid amount for each item. They can place bids on any item they are interested in, and the system automatically updates the current bid amount for that item. Users can also view their bidding history and track the status of their current bids.

The administrator view of the application allows the administrator to add new items for auction. The administrator can enter the item details, including the starting bid amount, reserve price, and auction end time. Once the item is added, it becomes available for users to bid on. The administrator can also view the bidding history for each item and close the auction when the auction end time is reached.

To implement this system, the Entity Relationship (ER) diagram will include entities such as Users, Items, and Bids. The User entity will store user information, including name, email, and password. The Item entity will store the details of the items available for auction, such as item name, description, starting bid, and auction end time. The Bid entity will store the bid details of the users, including the user ID, item ID, and the bid amount. The ER diagram will also include relationships between these entities, such as a one-to-many relationship between the User and Bid entities, and a one-to-many relationship between the Item and Bid entities.

# ER Diagram

Based on the description provided, the entities and relationships in the Live Auction Application System are: **(Primary keys are in bold and underlined)**

## Entities:

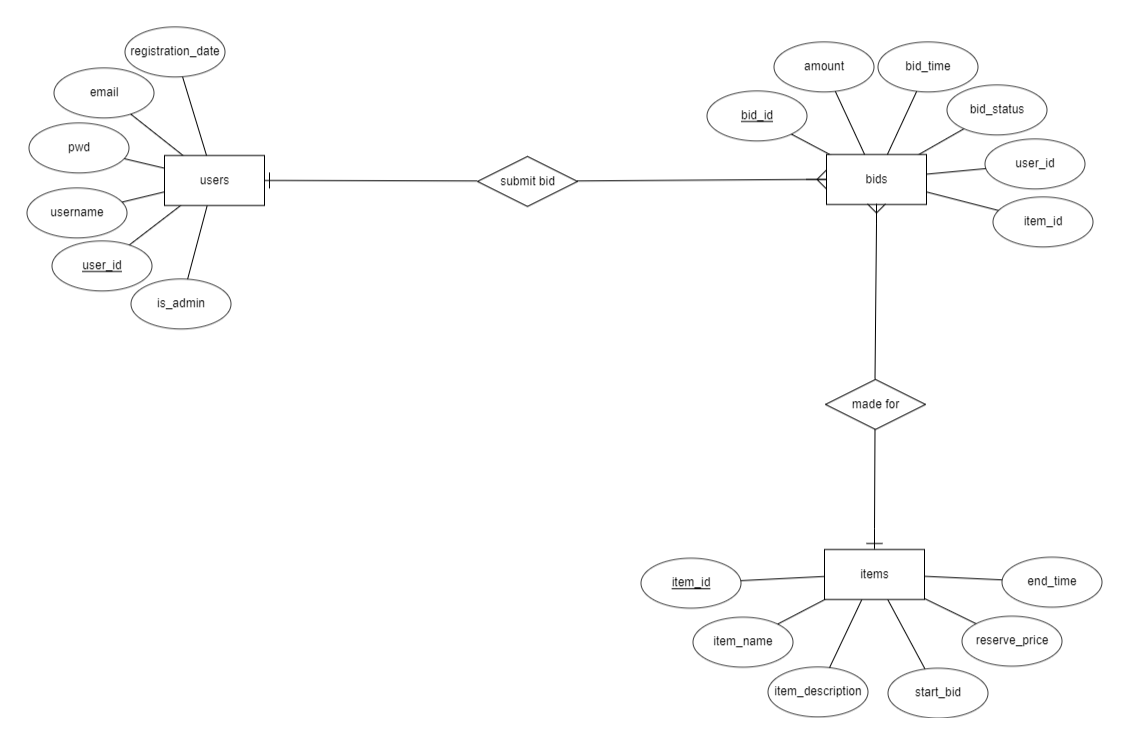
1. *Users*: Attributes – **user\_id**, username, pwd, email, registration\_date,is\_admin .
2. *Items*: Attributes – **item\_id**, item\_name, item\_description, start\_bid, reserve\_price, end\_time.
3. *Bids*: Attributes – **bid\_id**, user\_id, item\_id, amount, bid\_time, bid\_status(user\_id and item\_id are foreign keys in this table, which are being referenced from users and items table respectively).

## Relationships:

1. *Submit bid* : User will submit one or more bids.
2. *Made for:* Bids will be made for one item.

## Cardinality:

1. A User can make zero or more bids. A Bid must belong to one User.
2. An Item can have zero or more bids. A bid must belong to an item.



# Relational Schema

Based on the description provided, the relational schema along with constraints, candidate keys and foreign keys.

SQL code contains data type constraints for each attribute.

**Raw**:

1. *Users* (**user\_id**, username, pwd, email, registration\_date,is\_admin)
2. *Items* (**item\_id**, item\_name, item\_description, start\_bid, reserve\_price, end\_time)
3. *Bids* (**bid\_id**, user\_id, item\_id, amount, bid\_time, bid\_status)

## Normalization:

To check for 3NF form, we must show that there is **no transitive dependency** present.

*1NF Check:* Tables should not contain any multivalued attribute, we do not have any multivalued attributes. ☑

*2NF Check:* All the non-prime attributes should be fully functional dependent on Candidate Key (No Partial dependency), since we do not have any subsets for the primary keys, there is no partial dependency. Calculating closure and checking, the model satisfies 2NF. ☑

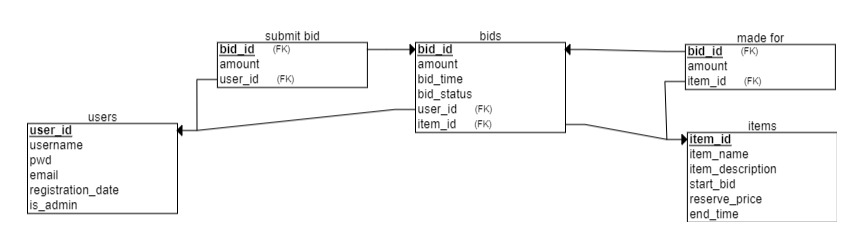
*3NF Check:* To ensure no transitive dependency, each Functional Dependency should have LHS as a Candidate Key OR RHS as a Prime Attribute, our functional dependencies have LHS as candidate keys, therefore it is in 3NF.

We need to check whether any attribute depends on another apart from the Candidate key.

There is no such functional dependency. ☑

## ­­­Constraints:

|  |  |  |
| --- | --- | --- |
| **Table** | **Primary Key** | **Foreign Key** |
| users | user\_id | - |
| items | item\_id | - |
| bids | bid\_id | user\_id, item\_id |



*Users* (**user\_id**, username, pwd, email, registration\_date)

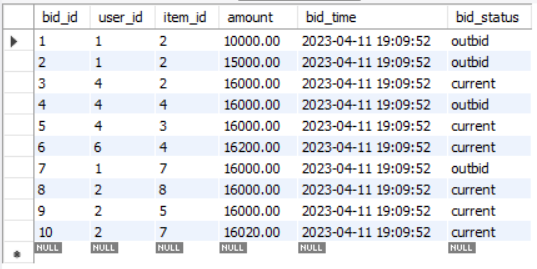
*Items* (**item\_id**, item\_name, item\_description, start\_bid, reserve\_price, end\_time)

*Bids* (**bid\_id**, user\_id, item\_id, amount, bid\_time, bid\_status)

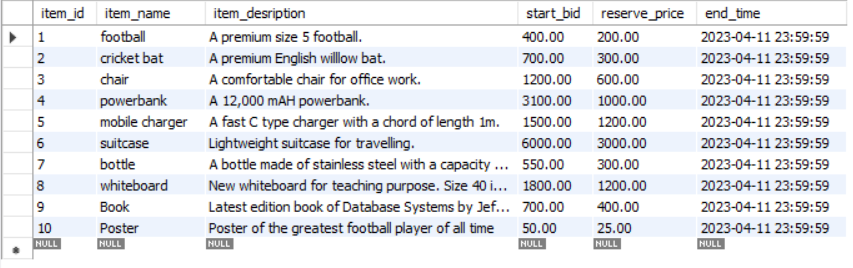
This is the database schema. Create frontend to show all the relevant information.

SQL queries and their output screenshots:-

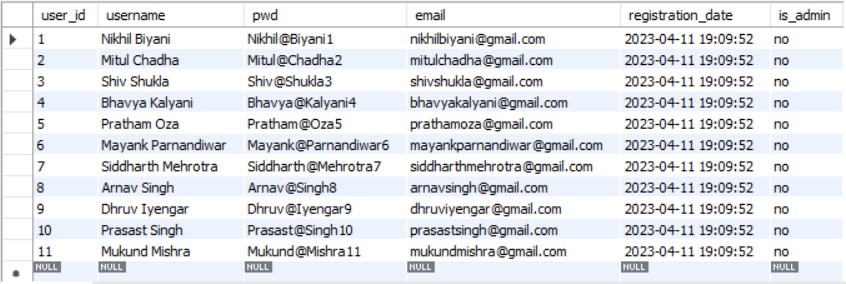
BID TABLE(for reference)

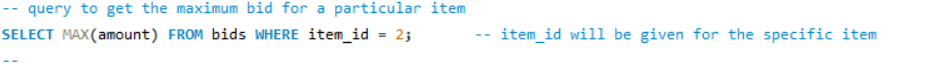


ITEMS TABLE(for reference)

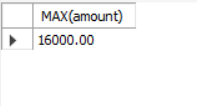


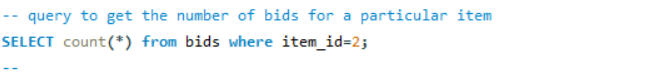
USERS TABLE(for reference)



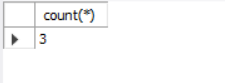


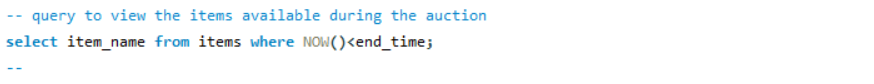
Output:



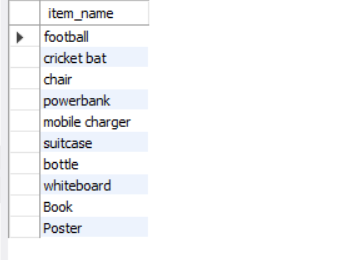


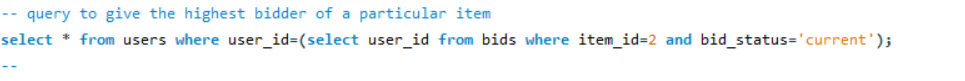
Output:



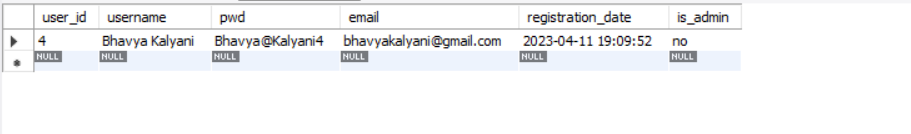


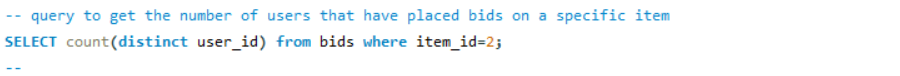
Output:





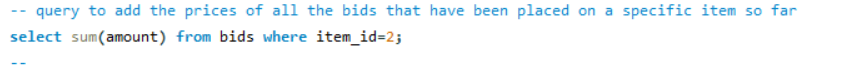
Output:





Output:



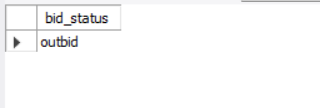


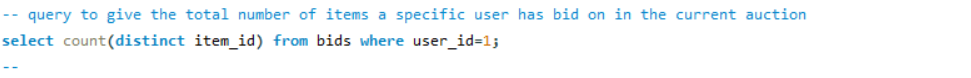
Output:





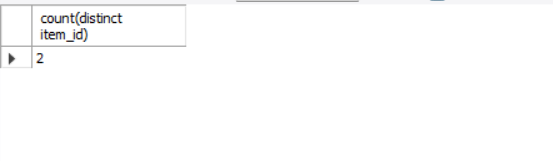
Output:

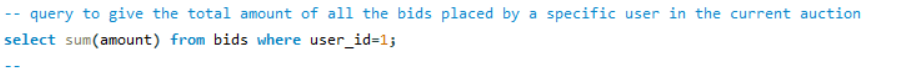




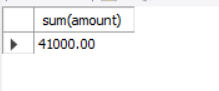
:

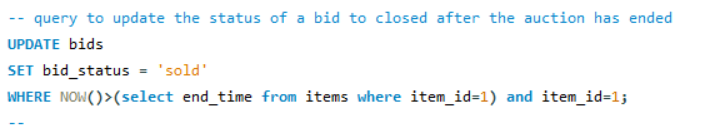
Output:

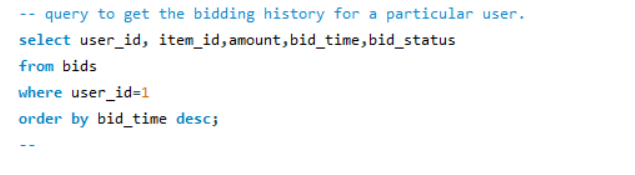




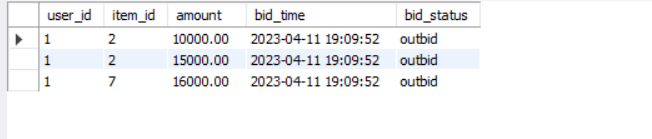
Output:







Output:



**LINK FOR THE VIDEO DOCUMENTATION:**

<https://drive.google.com/drive/folders/1tR1LKS-X86kwb9eBBXV6MwRoUpnd5tH_?usp=share_link>