Mini project report on

Online Auction System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE21CS351A – DBMS Project

Submitted by:

M.Keerthan Reddy	**PES2UG21CS258**
Mallikarjun Reddy	**PES2UG21CS274**

Under the guidance of

Dr.Mannar Mannan

Assistant Professor

Designation

PES University

AUG - DEC 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Online Auction System

is a bonafide work carried out by

M.Keerthan Reddy	**PES2UG21CS258**
Mallikarjun Reddy	**PES2UG21CS274**

In partial fulfilment for the completion of fifth semester DBMS Project (UE20CSS301) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2022 – DEC. 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature

Dr.Mannar Mannan

Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Online Auction System** has been carried out by us under the guidance of **Dr.Mannar Mannan Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2023.

M.Keerthan Reddy	**PES2UG21CS258**	**<Signature>**
Mallikarjun Reddy	**PES2UG21CS274**	**<Signature>**

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr.Mannar Mannan, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE21CS351A - DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor,ChairPerson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The Online Auction Platform is a web-based application designed to facilitate the buying and selling of items through an auction-style mechanism. The platform connects sellers and buyers, allowing sellers to list items, set starting prices, and define auction durations. Buyers can place bids on items they are interested in, and the highest bid at the end of the auction period determines the winner.

Key Features:

User Authentication and Authorization:

Users can register and log in to the platform, ensuring secure access to their accounts.

Differentiate between regular users and administrators for enhanced control and management.

Item Listing and Auction Management:

Sellers can create listings for items, providing details such as item name, description, starting price and auction duration.

Auctions automatically close at the specified end time, and the highest bidder is declared the winner.

Bidding System:

Buyers can place bids on items they are interested in, and the system updates the current highest bid in real-time.

Bid amounts are tracked, and the auction concludes at the scheduled end time.

Watchlist Functionality:

Users can add items to their watchlist to receive notifications and easily track items of interest.

User Dashboards:

Provide personalized dashboards for users to manage their listed items, bids, and watchlist.

Order History:

Users can view their order history, including successfully won items.

Seller and Admin Capabilities:

Sellers have access to tools for managing their listed items, and administrators have additional controls for user management and oversight.

Real-time Updates:

Implement real-time updates to keep users informed about bid changes, auction status, and other critical events.

Security Measures:

Employ secure user authentication and authorization methods to protect user accounts and sensitive information.

Database Integration:

Utilize a relational database to store user data, item details, bids, and other essential information.

The Online Auction Platform offers an engaging and secure environment for users to participate in auctions, fostering a dynamic marketplace for buying and selling various items. With a user-friendly interface and robust functionality, the platform aims to enhance the online auction experience for both buyers and sellers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Online Auction Platform

Our project, an Online Auction Platform, modernizes the auction landscape with a user-friendly digital interface. Seamlessly integrating React.js, Node.js, Express, MySQL, and WebSocket technologies, it provides a secure, responsive, and dynamic marketplace. Users experience streamlined authentication, efficient bid systems, and robust seller management. Key features include watchlist functionality, automatic item expiry updates, and secure transaction processes. Admins gain oversight with detailed analytics. The platform fosters a secure, interactive, and efficient auction experience for buyers and sellers alike.

Technology Stack:

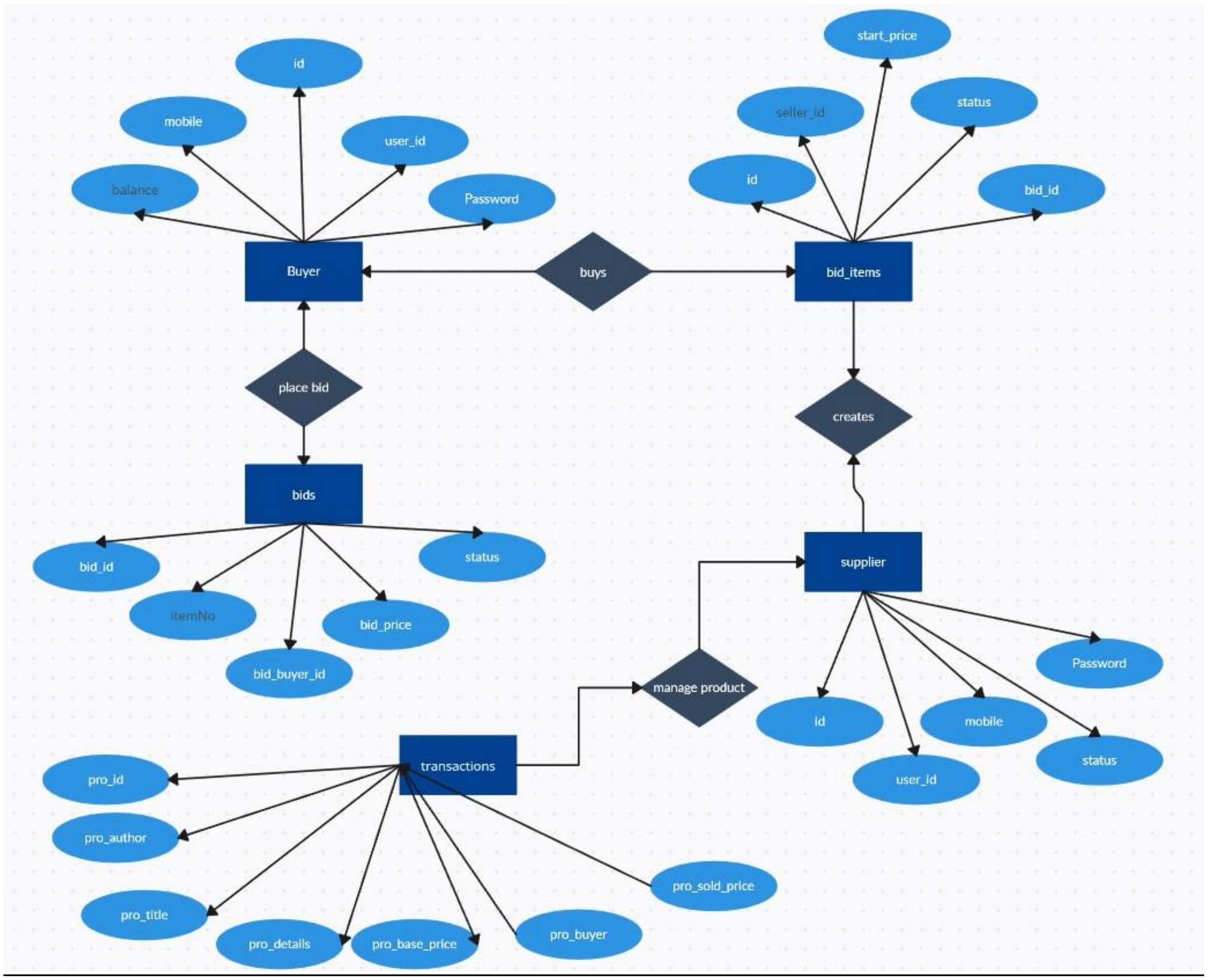
Frontend: React.js for a dynamic and responsive user interface.
Backend: Node.js and Express for server-side logic.
Database: MySQL for efficient data storage and retrieval.
Authentication: bcrypt for secure password hashing.

This online auction platform is designed to empower users with a reliable, secure, and feature-rich environment, fostering a vibrant online marketplace for various goods and services.

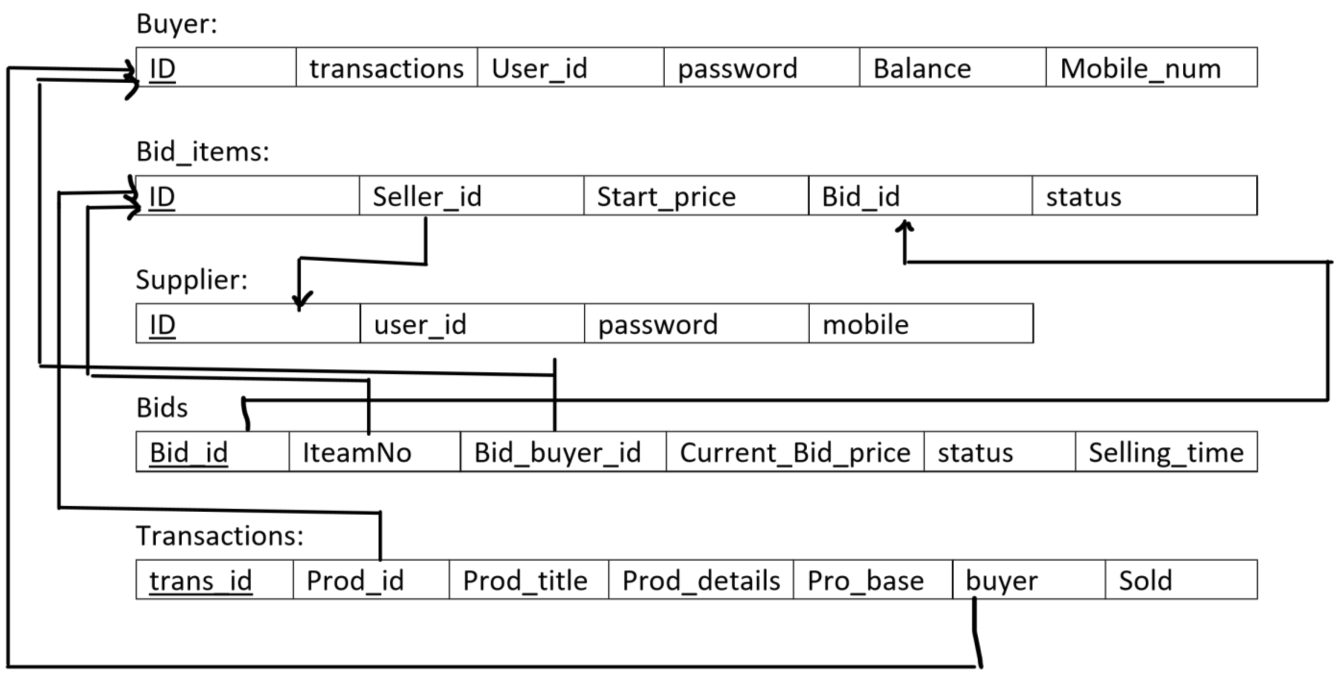
2. PROBLEM DEFINITION

The traditional auction model encounters various challenges in the contemporary digital landscape. Geographical constraints limit the reach of offline auctions, posing challenges for both sellers and buyers. Sellers face difficulties in efficiently managing and promoting their items, while buyers may find it cumbersome to monitor and participate in multiple auctions simultaneously. Security concerns, lack of real-time updates, and the absence of a centralized platform contribute to an outdated auction experience. To address these issues, our project focuses on the development of an Online Auction Platform. This platform aims to provide a seamless, accessible, and secure solution for conducting auctions in real-time, enhancing the overall experience for both sellers and buyers. Through this digital transformation, we strive to create a more inclusive, streamlined, and user-friendly environment for online auctions.

3. ER MODEL


id
mobile
user_id
balance
Password
Buyer
buys
bid_items
start_price
seller_id
status
id
bid_id
place bid
creates
bids
bid_id
itemNo
bid_buyer_id
bid_price
status
supplier
manage product
Password
id
mobile
status
user_id
transactions
pro_id
pro_author
pro_title
pro_details
pro_base_price
pro_buyer
pro_sold_price

4. ER TO RELATIONAL MAPPING



Buyer:

ID	transactions	User_id	password	Balance	Mobile_num

Bid_items:

ID	Seller_id	Start_price	Bid_id	status

Supplier:

ID	user_id	password	mobile

Bids

Bid_id	IteamNo	Bid_buyer_id	Current_Bid_price	status	Selling_time

Transactions:

trans_id	Prod_id	Prod_title	Prod_details	Pro_base	buyer	Sold

4.1 STEPS OF ALGORITHM FOR CHOOSEN PROBLEM

User Registration:

Users register on the platform providing necessary details like username, email, and password.
Seller Profile Creation:

Sellers create profiles, linking their accounts to verify their identity.
Item Listing:

Sellers list items for auction, including details like item name, description, starting price, start time, and end time.
Real-time Bidding:

Buyers place bids on items in real-time, with the system updating the current bid for each item.
Bid Validation:

The system validates bids, ensuring they exceed the current highest bid and comply with auction rules.
Auto Bid Feature:

Buyers can use an auto-bid feature, allowing the system to automatically place incremental bids on their behalf.
Watchlist Functionality:

Users can add items to their watchlist to monitor auctions of interest easily.
Item Expiry Handling:

A trigger updates the status of items to "sold" if the end time is reached, closing the auction.
User Authentication:

Secure authentication mechanisms, including encryption and cookie management, ensure user accounts are protected.
Admin Login:

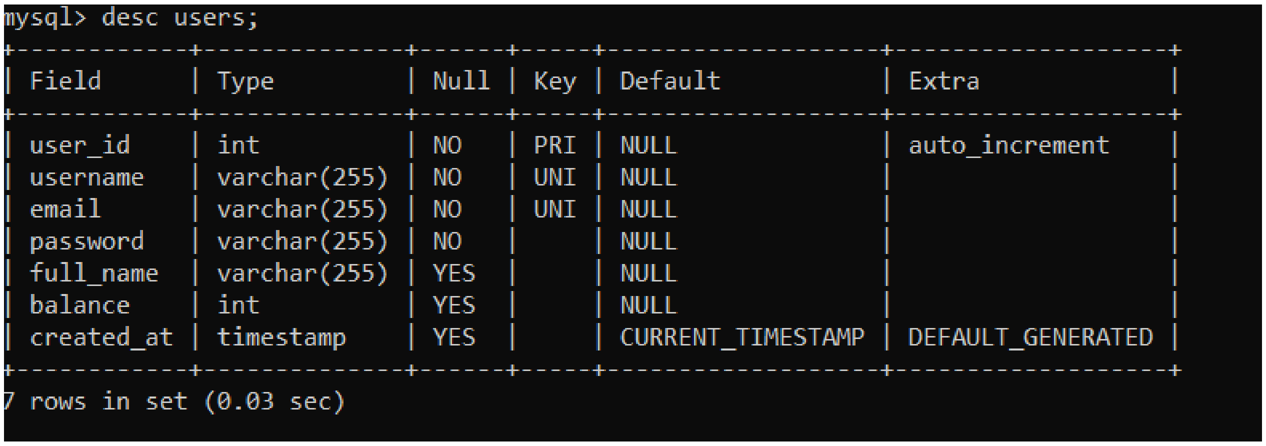
Admins log in to access additional functionalities for monitoring and managing the platform.
Order Placement for Sold Items:

Buyers can view and place orders for items they've won in the auction.

5. DDL STATEMENTS

STATEMENTS WITH SCREEN SHOTS OF THE TABLE CREATION

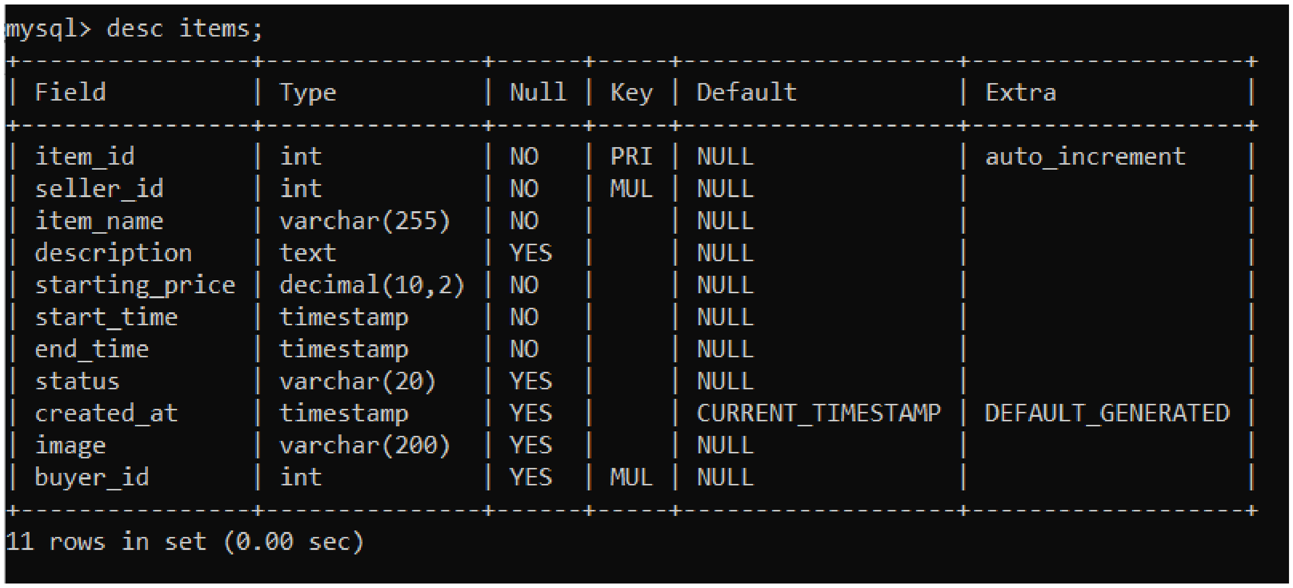
```
CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    full_name VARCHAR(255),
    balance INT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```



```
mysql> desc users;
+------------+--------------+------+-----+-------------------+-------------------+
| Field      | Type         | Null | Key | Default           | Extra             |
+------------+--------------+------+-----+-------------------+-------------------+
| user_id    | int          | NO   | PRI | NULL              | auto_increment    |
| username   | varchar(255) | NO   | UNI | NULL              |                   |
| email      | varchar(255) | NO   | UNI | NULL              |                   |
| password   | varchar(255) | NO   |     | NULL              |                   |
| full_name  | varchar(255) | YES  |     | NULL              |                   |
| balance    | int          | YES  |     | NULL              |                   |
| created_at | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+------------+--------------+------+-----+-------------------+-------------------+
7 rows in set (0.03 sec)
```

Items table:

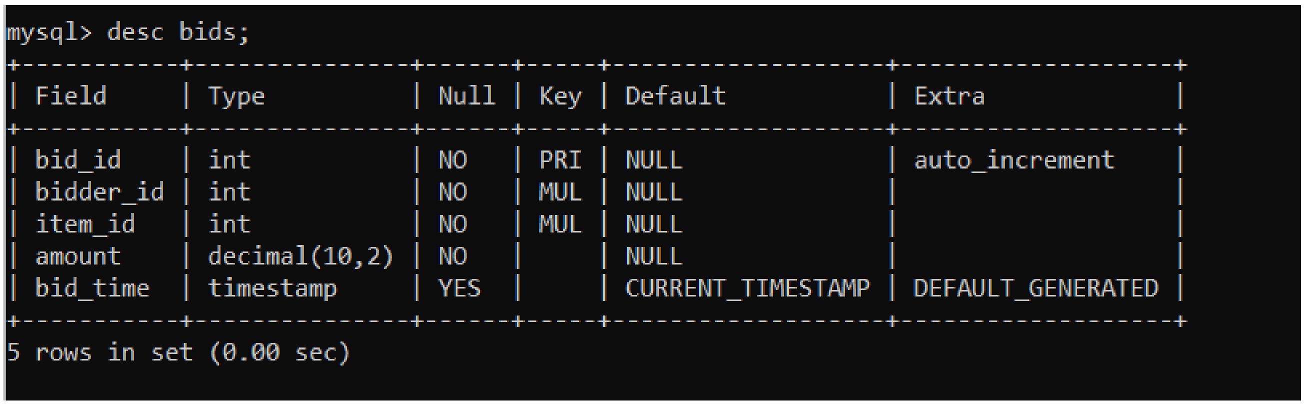
```
CREATE TABLE Items (
    item_id INT AUTO_INCREMENT PRIMARY KEY,
    seller_id INT NOT NULL,
    item_name VARCHAR(255) NOT NULL,
    description TEXT,
    starting_price DECIMAL(10, 2) NOT NULL,
    start_time TIMESTAMP NOT NULL,
    end_time TIMESTAMP NOT NULL,
    status ENUM('active', 'sold’, 'unsold') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    image varchar(200),
    buyer_id int,
    FOREIGN KEY (seller_id) REFERENCES sellers(seller_id),
    FOREIGN KEY (buyer_id) REFERENCES Users(user_id)
);
```



```
mysql> desc items;
+----------------+---------------+------+-----+-------------------+-------------------+
| Field          | Type          | Null | Key | Default           | Extra             |
+----------------+---------------+------+-----+-------------------+-------------------+
| item_id        | int           | NO   | PRI | NULL              | auto_increment    |
| seller_id      | int           | NO   | MUL | NULL              |                   |
| item_name      | varchar(255)  | NO   |     | NULL              |                   |
| description    | text          | YES  |     | NULL              |                   |
| starting_price | decimal(10,2) | NO   |     | NULL              |                   |
| start_time     | timestamp     | NO   |     | NULL              |                   |
| end_time       | timestamp     | NO   |     | NULL              |                   |
| status         | varchar(20)   | YES  |     | NULL              |                   |
| created_at     | timestamp     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| image          | varchar(200)  | YES  |     | NULL              |                   |
| buyer_id       | int           | YES  | MUL | NULL              |                   |
+----------------+---------------+------+-----+-------------------+-------------------+
11 rows in set (0.00 sec)
```

Bids table:

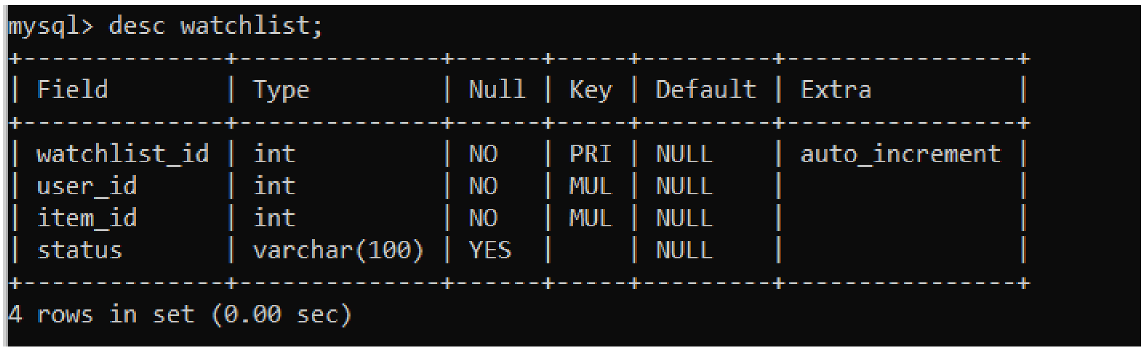
```
CREATE TABLE Bids (
    bid_id INT AUTO_INCREMENT PRIMARY KEY,
    bidder_id INT NOT NULL,
    item_id INT NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    bid_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (bidder_id) REFERENCES Users(user_id),
    FOREIGN KEY (item_id) REFERENCES Items(item_id),
    CHECK (amount > 0),
    CHECK (bid_time BETWEEN start_time AND end_time)
);
```



```
mysql> desc bids;
+-----------+---------------+------+-----+-------------------+-------------------+
| Field     | Type          | Null | Key | Default           | Extra             |
+-----------+---------------+------+-----+-------------------+-------------------+
| bid_id    | int           | NO   | PRI | NULL              | auto_increment    |
| bidder_id | int           | NO   | MUL | NULL              |                   |
| item_id   | int           | NO   | MUL | NULL              |                   |
| amount    | decimal(10,2) | NO   |     | NULL              |                   |
| bid_time  | timestamp     | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------+---------------+------+-----+-------------------+-------------------+
5 rows in set (0.00 sec)
```

Watchlist Table:-

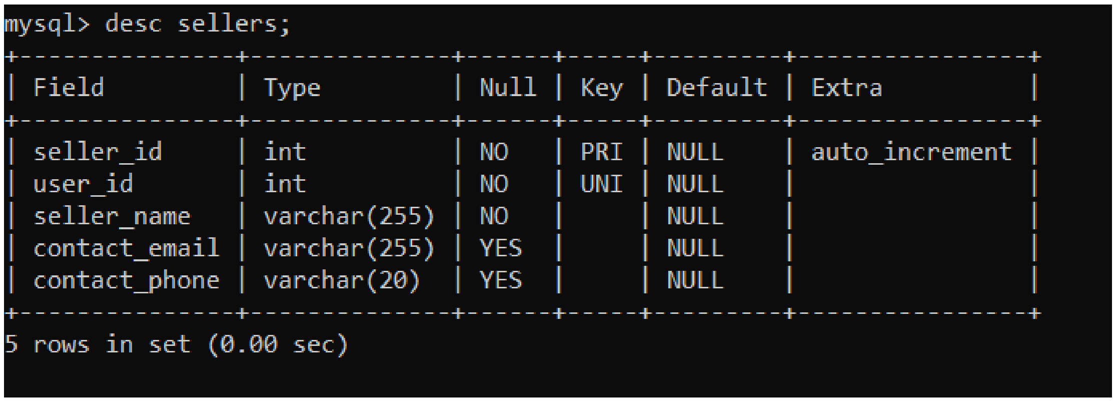
```
CREATE TABLE Watchlist (
    watchlist_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    item_id INT NOT NULL,
    status varchar(100),
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (item_id) REFERENCES Items(item_id)
);
```



```
mysql> desc watchlist;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| watchlist_id | int          | NO   | PRI | NULL    | auto_increment |
| user_id      | int          | NO   | MUL | NULL    |                |
| item_id      | int          | NO   | MUL | NULL    |                |
| status       | varchar(100) | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Sellers table:-

```
CREATE TABLE Sellers (
    seller_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL UNIQUE,
    seller_name VARCHAR(255) NOT NULL,
    contact_email VARCHAR(255),
    contact_phone VARCHAR(20),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

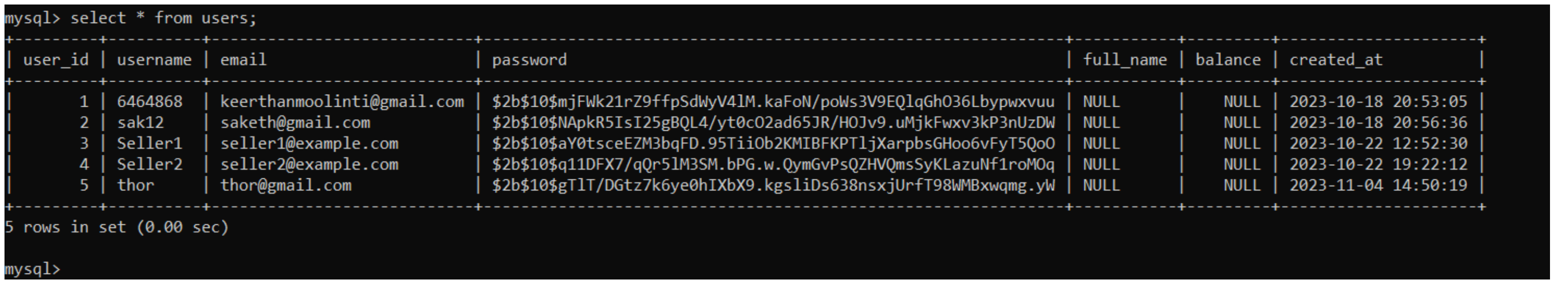


```
mysql> desc sellers;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| seller_id     | int          | NO   | PRI | NULL    | auto_increment |
| user_id       | int          | NO   | UNI | NULL    |                |
| seller_name   | varchar(255) | NO   |     | NULL    |                |
| contact_email | varchar(255) | YES  |     | NULL    |                |
| contact_phone | varchar(20)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

6. DML STATEMENTS

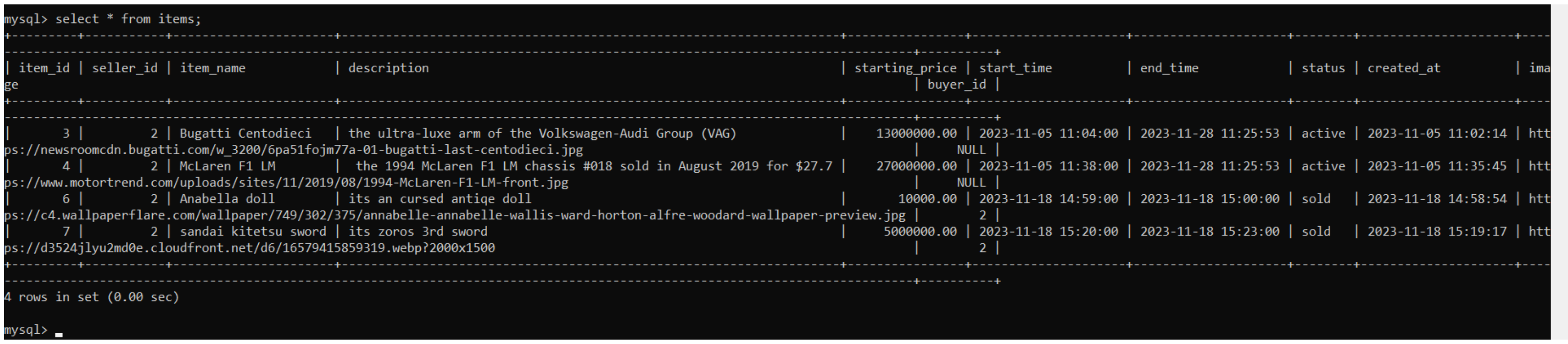
STATEMENTS WITH SCREEN SHOTS OF THE TABLE WITH INSERTED VALUES

```
INSERT INTO Users (username, email, password) VALUES (?, ?, ?)
```



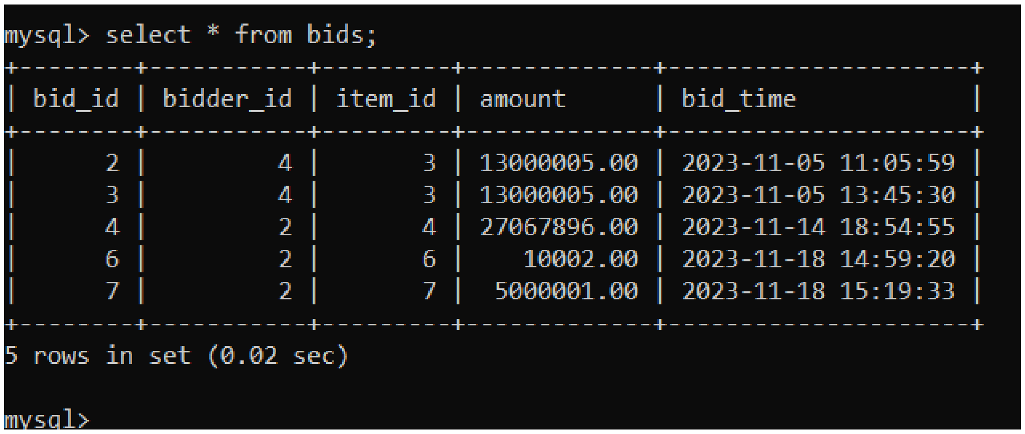
```
mysql> select * from users;
+---------+----------+---------------------------+--------------------------------------------------------------+-----------+---------+---------------------+
| user_id | username | email                     | password                                                     | full_name | balance | created_at          |
+---------+----------+---------------------------+--------------------------------------------------------------+-----------+---------+---------------------+
|       1 | 6464868  | keerthanmoolinti@gmail.com | $2b$10$mjFWk21rZ9ffpSdWyV41M.kaFoN/poWs3V9EQlqGhO36Lbypwxvuu | NULL      |    NULL | 2023-10-18 20:53:05 |
|       2 | sak12    | saketh@gmail.com          | $2b$10$NApkR5IsI25gBQL4/yt0cO2ad65JR/HOJv9.uMjkFwxv3kP3nUzDW | NULL      |    NULL | 2023-10-18 20:56:36 |
|       3 | Seller1  | seller1@example.com       | $2b$10$aY0tsceEZM3bqFD.95TiiOb2KMIBFKPTljXarpbsGHoo6vFyT5QoO | NULL      |    NULL | 2023-10-22 12:52:30 |
|       4 | Seller2  | seller2@example.com       | $2b$10$q11DFX7/qQr5lM3SM.bPG.w.QymGvPsQZHVQmsSyKLazuNf1roMOq | NULL      |    NULL | 2023-10-22 19:22:12 |
|       5 | thor     | thor@gmail.com            | $2b$10$gTlT/DGtz7k6ye0hIXbX9.kgsliDs638nsxjUrfT98WMBxwqmg.yW | NULL      |    NULL | 2023-11-04 14:50:19 |
+---------+----------+---------------------------+--------------------------------------------------------------+-----------+---------+---------------------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO Items (seller_id, item_name, description, starting_price, start_time, end_time, status, image)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)



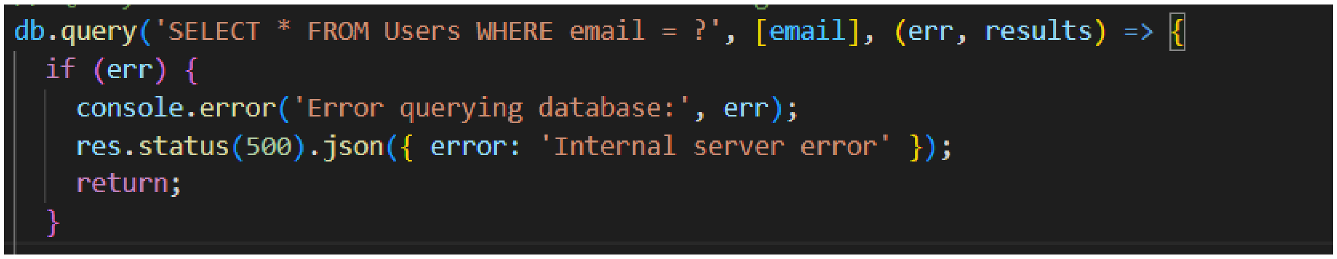
```
mysql> select * from items;
+---------+-----------+--------------------+------------------------------------------------------------+----------------+---------------------+---------------------+--------+---------------------+----
----------------------------------------------------------------------------------------------------+----------+
| item_id | seller_id | item_name          | description                                                | starting_price | start_time          | end_time            | status | created_at          | ima
ge                                                                                                   | buyer_id |
+---------+-----------+--------------------+------------------------------------------------------------+----------------+---------------------+---------------------+--------+---------------------+----
----------------------------------------------------------------------------------------------------+----------+
|       3 |         2 | Bugatti Centodieci | the ultra-luxe arm of the Volkswagen-Audi Group (VAG)      |    13000000.00 | 2023-11-05 11:04:00 | 2023-11-28 11:25:53 | active | 2023-11-05 11:02:14 | htt
ps://newsroomcdn.bugatti.com/w_3200/6pa51fojm77a-01-bugatti-last-centodieci.jpg                      |     NULL |
|       4 |         2 | McLaren F1 LM      |  the 1994 McLaren F1 LM chassis #018 sold in August 2019 for $27.7 |    27000000.00 | 2023-11-05 11:38:00 | 2023-11-28 11:25:53 | active | 2023-11-05 11:35:45 | htt
ps://www.motortrend.com/uploads/sites/11/2019/08/1994-McLaren-F1-LM-front.jpg                        |     NULL |
|       6 |         2 | Anabella doll      | its an cursed antiqe doll                                  |       10000.00 | 2023-11-18 14:59:00 | 2023-11-18 15:00:00 | sold   | 2023-11-18 14:58:54 | htt
ps://c4.wallpaperflare.com/wallpaper/749/302/375/annabelle-annabelle-wallis-ward-horton-alfre-woodard-wallpaper-preview.jpg |        2 |
|       7 |         2 | sandai kitetsu sword | its zoros 3rd sword                                      |     5000000.00 | 2023-11-18 15:20:00 | 2023-11-18 15:23:00 | sold   | 2023-11-18 15:19:17 | htt
ps://d3524jlyu2md0e.cloudfront.net/d6/16579415859319.webp?2000x1500                                  |        2 |
+---------+-----------+--------------------+------------------------------------------------------------+----------------+---------------------+---------------------+--------+---------------------+----
----------------------------------------------------------------------------------------------------+----------+
4 rows in set (0.00 sec)

mysql>
```

INSERT INTO Bids (bidder_id, item_id, amount) VALUES (?, ?, ?)

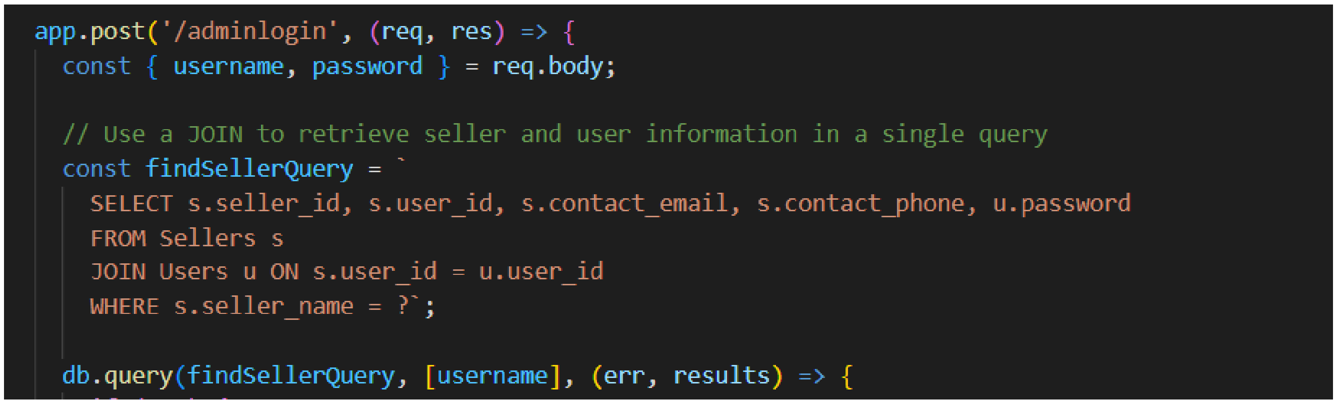


```
mysql> select * from bids;
+--------+-----------+---------+-------------+---------------------+
| bid_id | bidder_id | item_id | amount      | bid_time            |
+--------+-----------+---------+-------------+---------------------+
|      2 |         4 |       3 | 13000005.00 | 2023-11-05 11:05:59 |
|      3 |         4 |       3 | 13000005.00 | 2023-11-05 13:45:30 |
|      4 |         2 |       4 | 27067896.00 | 2023-11-14 18:54:55 |
|      6 |         2 |       6 |    10002.00 | 2023-11-18 14:59:20 |
|      7 |         2 |       7 |  5000001.00 | 2023-11-18 15:19:33 |
+--------+-----------+---------+-------------+---------------------+
5 rows in set (0.02 sec)

mysql>
```

Select statements:

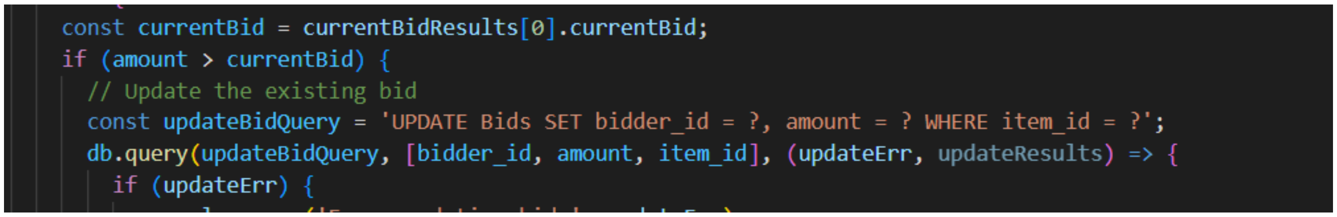


```
db.query('SELECT * FROM Users WHERE email = ?', [email], (err, results) => {
  if (err) {
    console.error('Error querying database:', err);
    res.status(500).json({ error: 'Internal server error' });
    return;
  }
```



```
app.post('/adminlogin', (req, res) => {
  const { username, password } = req.body;

  // Use a JOIN to retrieve seller and user information in a single query
  const findSellerQuery = `
    SELECT s.seller_id, s.user_id, s.contact_email, s.contact_phone, u.password
    FROM Sellers s
    JOIN Users u ON s.user_id = u.user_id
    WHERE s.seller_name = ?`;

  db.query(findSellerQuery, [username], (err, results) => {
```

Update:

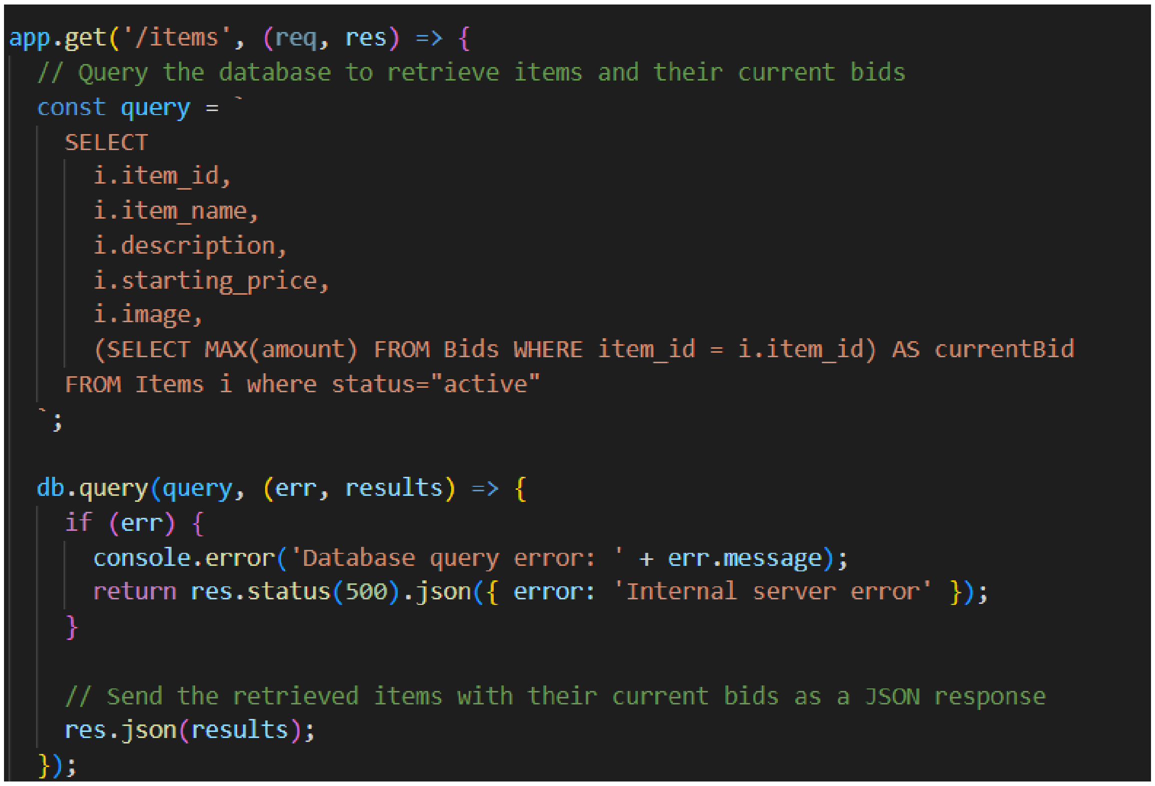


```
  const currentBid = currentBidResults[0].currentBid;
  if (amount > currentBid) {
    // Update the existing bid
    const updateBidQuery = 'UPDATE Bids SET bidder_id = ?, amount = ? WHERE item_id = ?';
    db.query(updateBidQuery, [bidder_id, amount, item_id], (updateErr, updateResults) => {
      if (updateErr) {
```

7. QUERIES

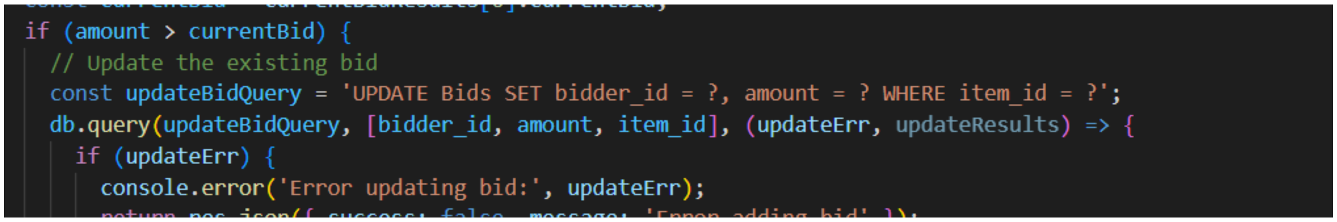
7.1 SIMPLE QUERY WITH GROUP BY, AGRREGATE

Max():



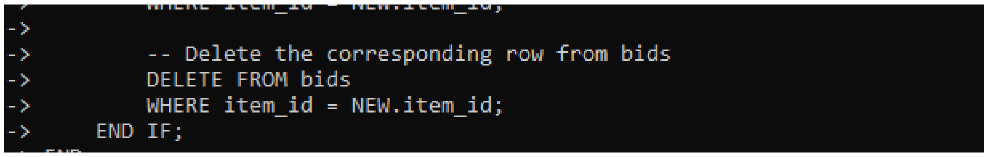
```
app.get('/items', (req, res) => {
  // Query the database to retrieve items and their current bids
  const query = `
    SELECT
      i.item_id,
      i.item_name,
      i.description,
      i.starting_price,
      i.image,
      (SELECT MAX(amount) FROM Bids WHERE item_id = i.item_id) AS currentBid
    FROM Items i where status="active"
  `;

  db.query(query, (err, results) => {
    if (err) {
      console.error('Database query error: ' + err.message);
      return res.status(500).json({ error: 'Internal server error' });
    }

    // Send the retrieved items with their current bids as a JSON response
    res.json(results);
  });
```

7.2 UPDATE OPERATION



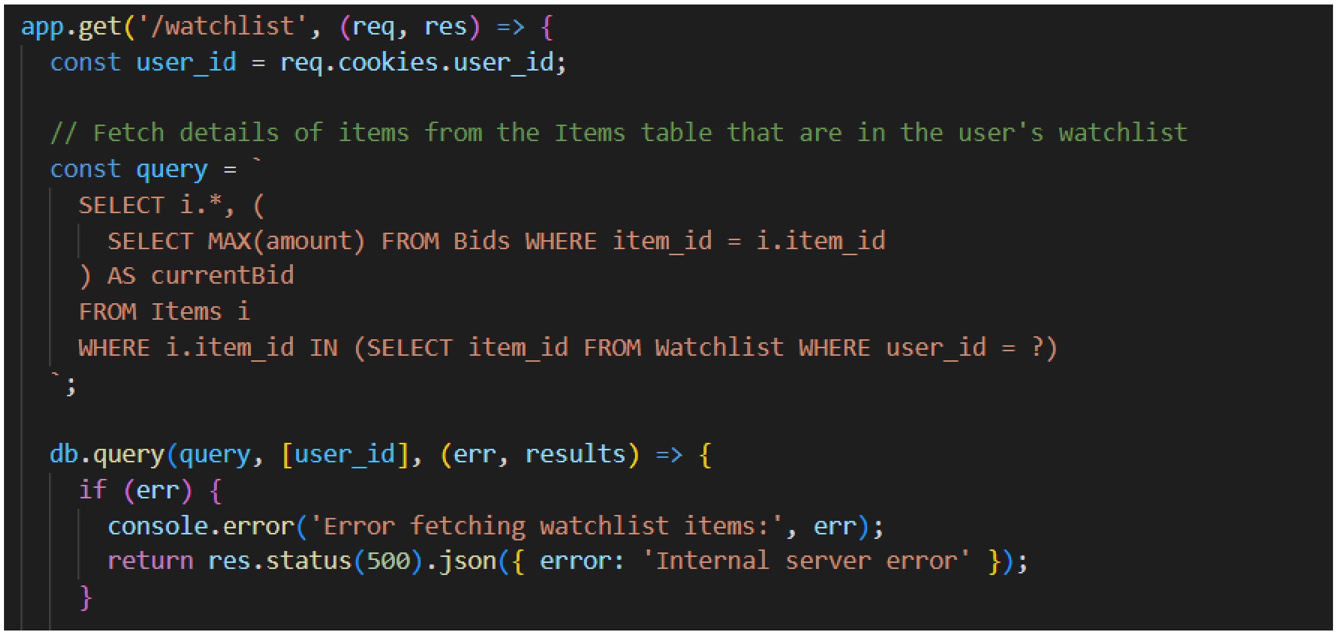
```
  const currentBid = currentBidResults[0].currentBid;
 if (amount > currentBid) {
   // Update the existing bid
   const updateBidQuery = 'UPDATE Bids SET bidder_id = ?, amount = ? WHERE item_id = ?';
   db.query(updateBidQuery, [bidder_id, amount, item_id], (updateErr, updateResults) => {
     if (updateErr) {
       console.error('Error updating bid:', updateErr);
       return res.json({ success: false, message: 'Error adding bid' });
```

7.3 DELETE OPERATION

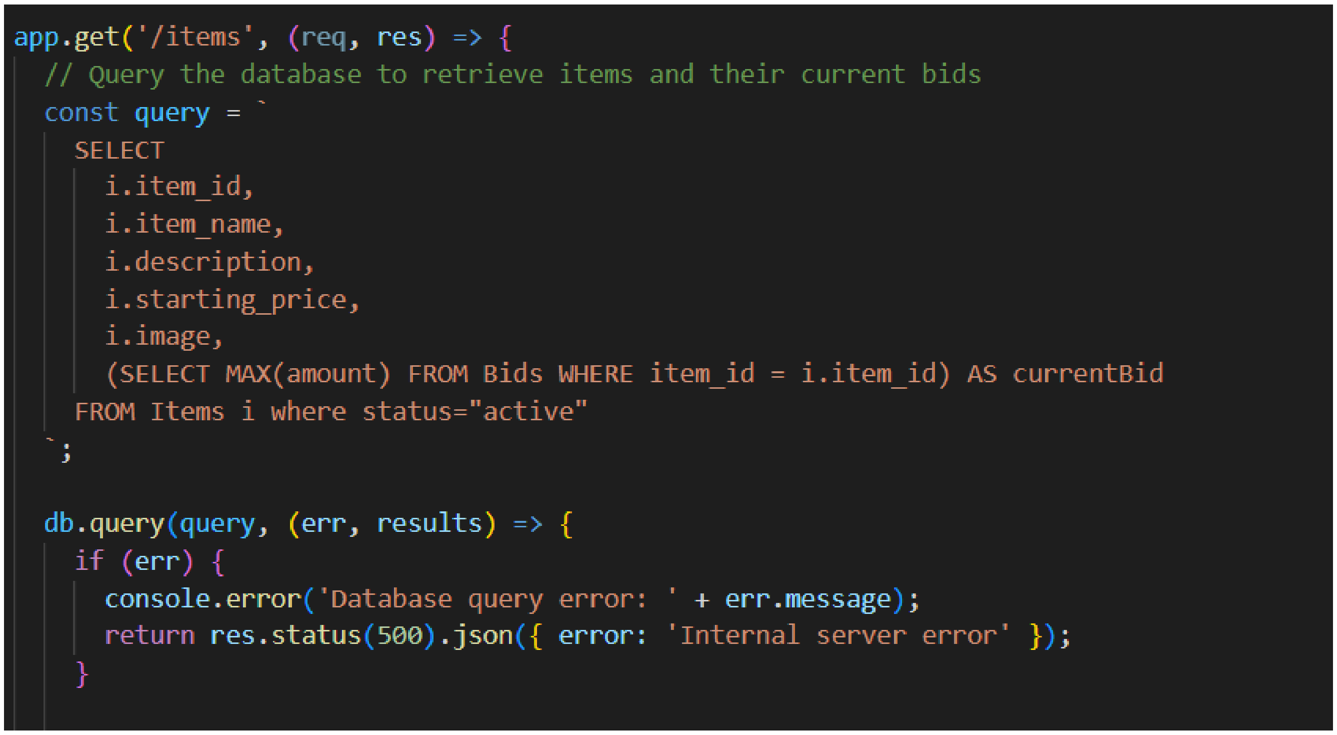


```
->          WHERE item_id = NEW.item_id;
->
->          -- Delete the corresponding row from bids
->          DELETE FROM bids
->          WHERE item_id = NEW.item_id;
->      END IF;
->  END
```

7.4 CORRELATED QUERY

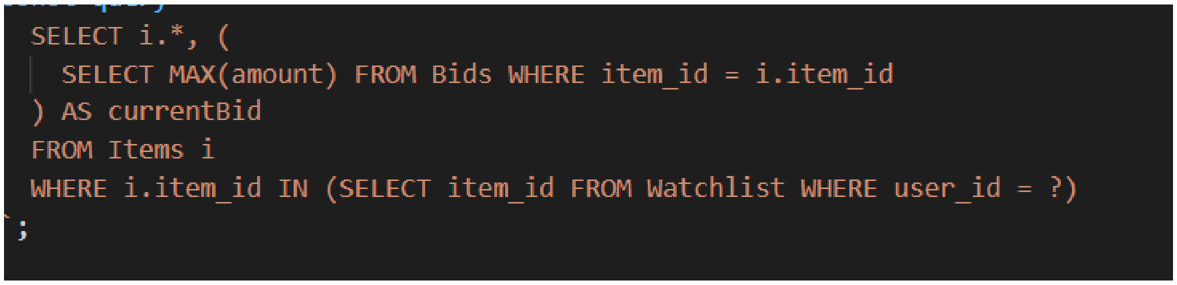


```
app.get('/watchlist', (req, res) => {
  const user_id = req.cookies.user_id;

  // Fetch details of items from the Items table that are in the user's watchlist
  const query = `
    SELECT i.*, (
      SELECT MAX(amount) FROM Bids WHERE item_id = i.item_id
    ) AS currentBid
    FROM Items i
    WHERE i.item_id IN (SELECT item_id FROM Watchlist WHERE user_id = ?)
  `;

  db.query(query, [user_id], (err, results) => {
    if (err) {
      console.error('Error fetching watchlist items:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
```



```
app.get('/items', (req, res) => {
  // Query the database to retrieve items and their current bids
  const query = `
    SELECT
      i.item_id,
      i.item_name,
      i.description,
      i.starting_price,
      i.image,
      (SELECT MAX(amount) FROM Bids WHERE item_id = i.item_id) AS currentBid
    FROM Items i where status="active"
  `;

  db.query(query, (err, results) => {
    if (err) {
      console.error('Database query error: ' + err.message);
      return res.status(500).json({ error: 'Internal server error' });
    }
```

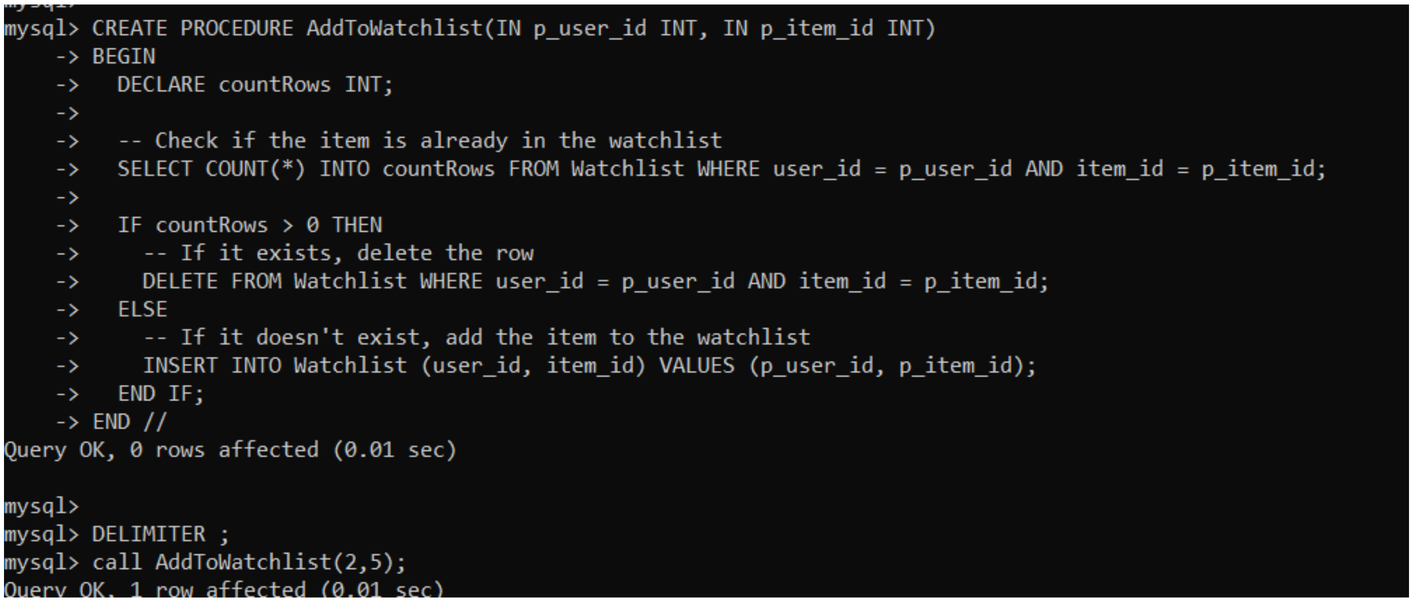
7.5 NESTED QUERY



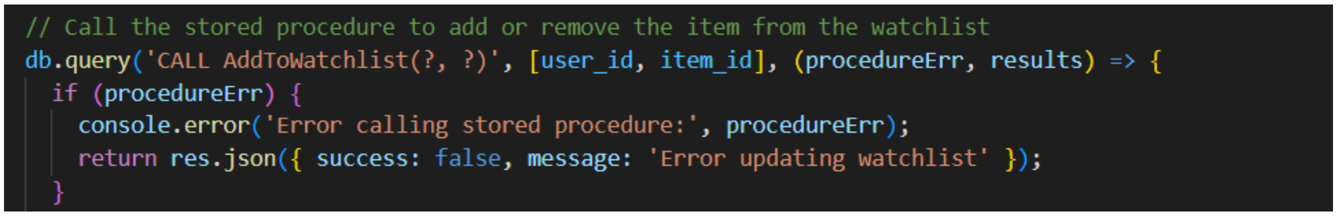
```
SELECT i.*, (
    SELECT MAX(amount) FROM Bids WHERE item_id = i.item_id
) AS currentBid
FROM Items i
WHERE i.item_id IN (SELECT item_id FROM Watchlist WHERE user_id = ?)
;
```

8. STORED PROCEDURES, FUCNTIONS AND TRIGGERS

8.1 STORED PROCEDURES OR FUNCTIONS

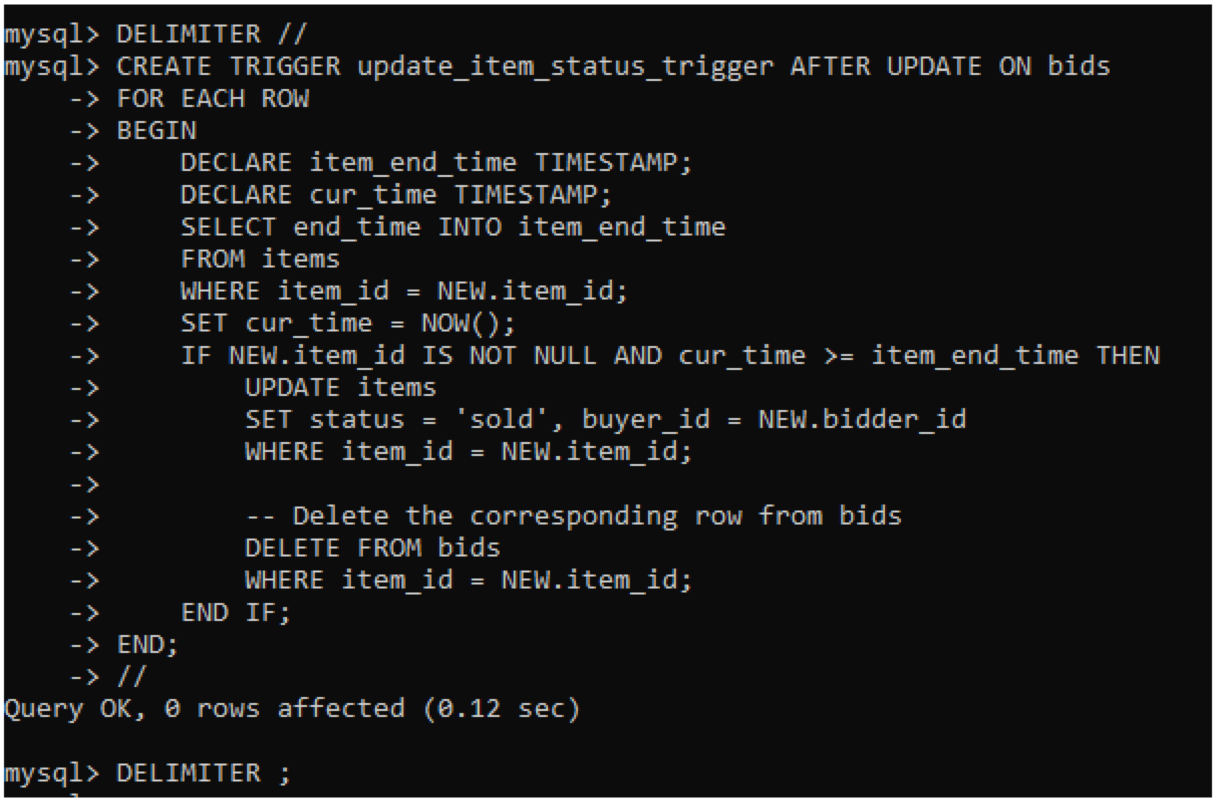


```
mysql> CREATE PROCEDURE AddToWatchlist(IN p_user_id INT, IN p_item_id INT)
    -> BEGIN
    ->   DECLARE countRows INT;
    ->
    ->   -- Check if the item is already in the watchlist
    ->   SELECT COUNT(*) INTO countRows FROM Watchlist WHERE user_id = p_user_id AND item_id = p_item_id;
    ->
    ->   IF countRows > 0 THEN
    ->     -- If it exists, delete the row
    ->     DELETE FROM Watchlist WHERE user_id = p_user_id AND item_id = p_item_id;
    ->   ELSE
    ->     -- If it doesn't exist, add the item to the watchlist
    ->     INSERT INTO Watchlist (user_id, item_id) VALUES (p_user_id, p_item_id);
    ->   END IF;
    -> END //
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> call AddToWatchlist(2,5);
Query OK, 1 row affected (0.01 sec)
```

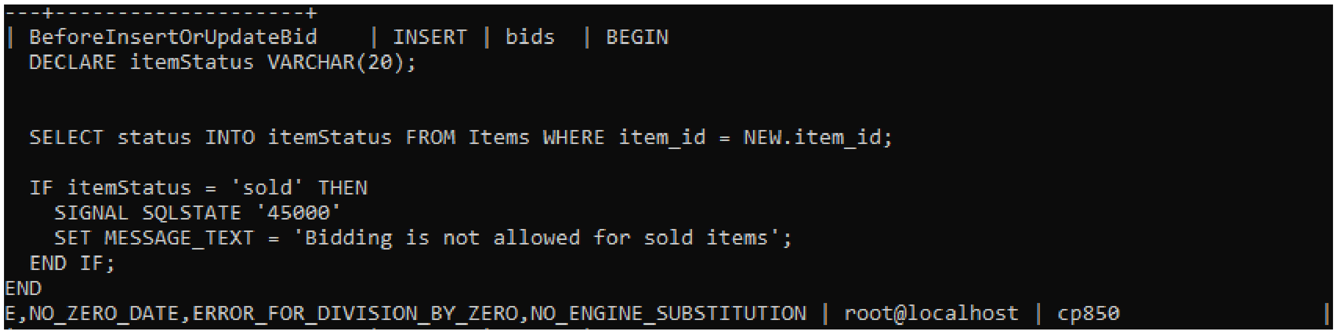


```
// Call the stored procedure to add or remove the item from the watchlist
db.query('CALL AddToWatchlist(?, ?)', [user_id, item_id], (procedureErr, results) => {
  if (procedureErr) {
    console.error('Error calling stored procedure:', procedureErr);
    return res.json({ success: false, message: 'Error updating watchlist' });
  }
```

8.2 TRIGGERS



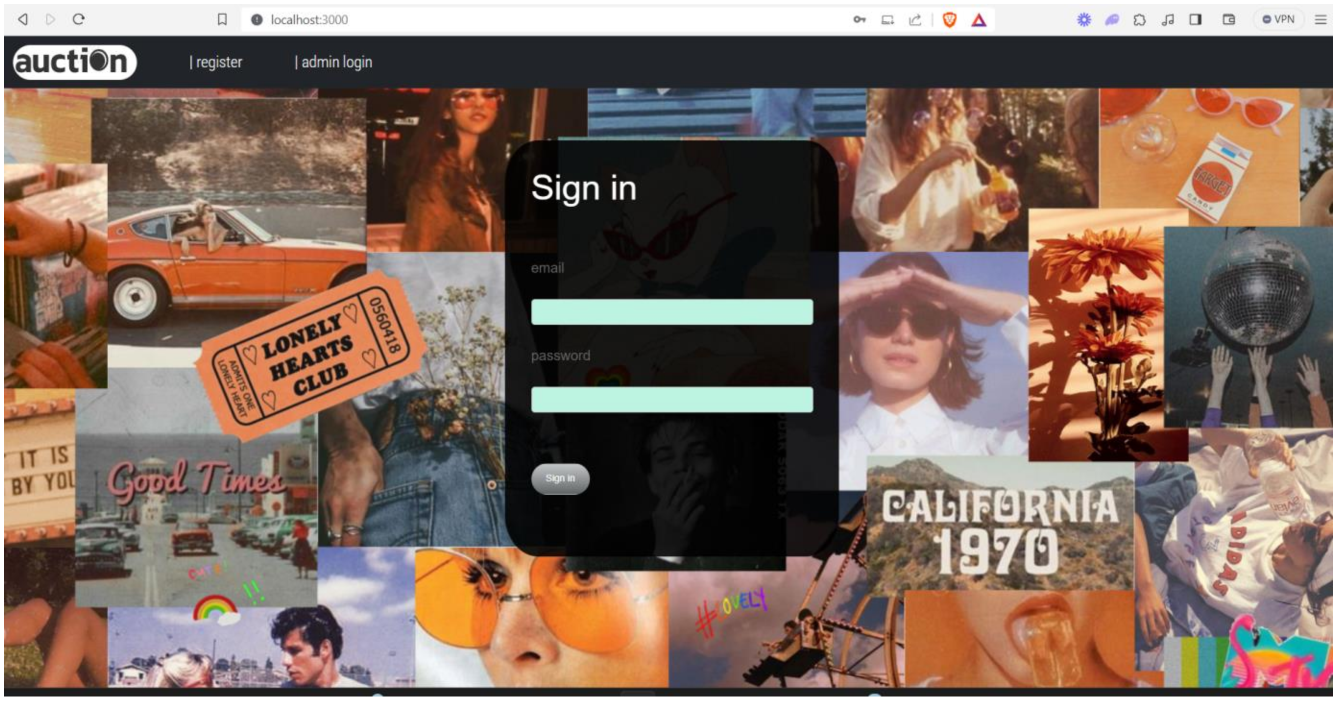
```
mysql> DELIMITER //
mysql> CREATE TRIGGER update_item_status_trigger AFTER UPDATE ON bids
    -> FOR EACH ROW
    -> BEGIN
    ->     DECLARE item_end_time TIMESTAMP;
    ->     DECLARE cur_time TIMESTAMP;
    ->     SELECT end_time INTO item_end_time
    ->     FROM items
    ->     WHERE item_id = NEW.item_id;
    ->     SET cur_time = NOW();
    ->     IF NEW.item_id IS NOT NULL AND cur_time >= item_end_time THEN
    ->         UPDATE items
    ->         SET status = 'sold', buyer_id = NEW.bidder_id
    ->         WHERE item_id = NEW.item_id;
    ->
    ->         -- Delete the corresponding row from bids
    ->         DELETE FROM bids
    ->         WHERE item_id = NEW.item_id;
    ->     END IF;
    -> END;
    -> //
Query OK, 0 rows affected (0.12 sec)

mysql> DELIMITER ;
```



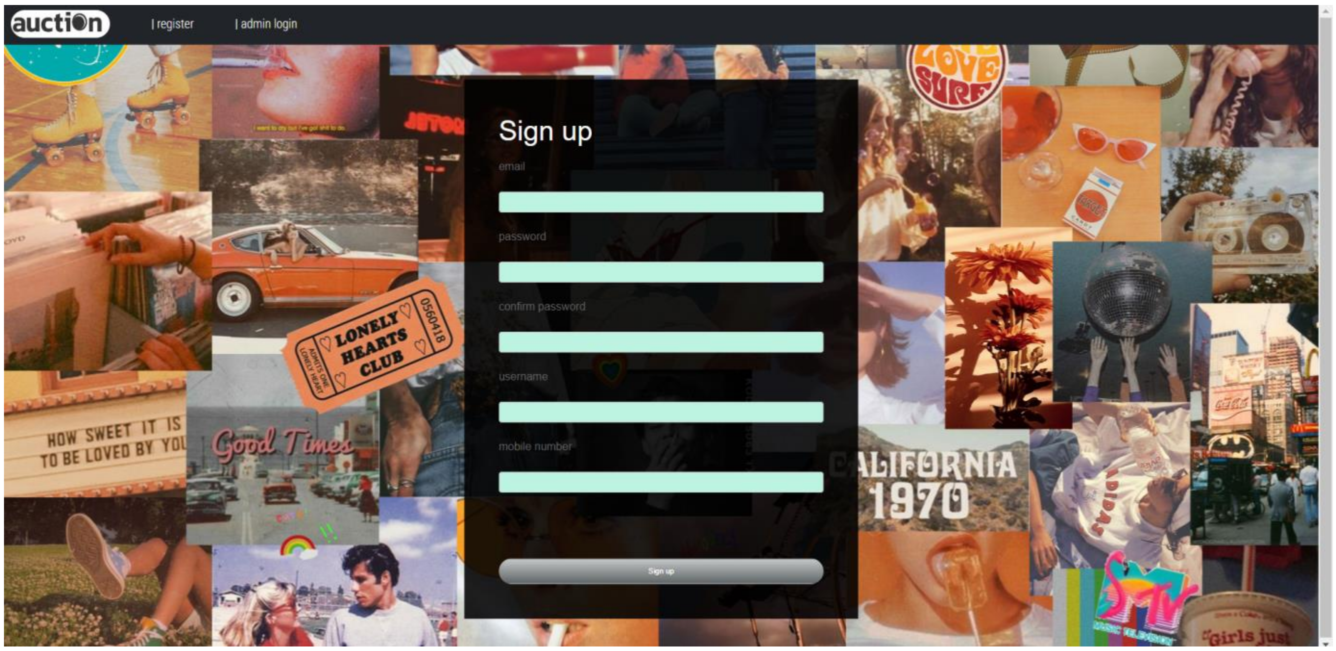
```
---+--------------------+
| BeforeInsertOrUpdateBid    | INSERT | bids  | BEGIN
  DECLARE itemStatus VARCHAR(20);


  SELECT status INTO itemStatus FROM Items WHERE item_id = NEW.item_id;

  IF itemStatus = 'sold' THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Bidding is not allowed for sold items';
  END IF;
END
E,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | cp850                    |
```

10. FRONT END DEVELOPEMENT

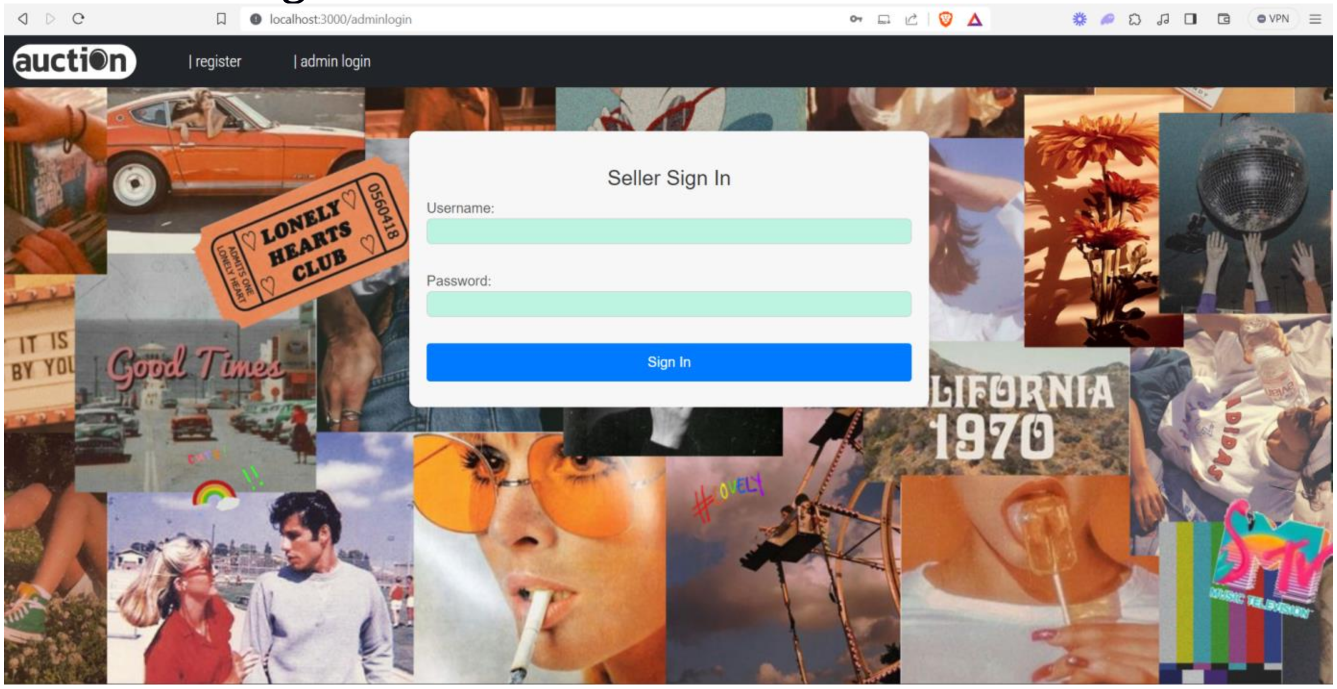
10.1 “/” or Signin:



10.2 /register



10.3 /adminlogin

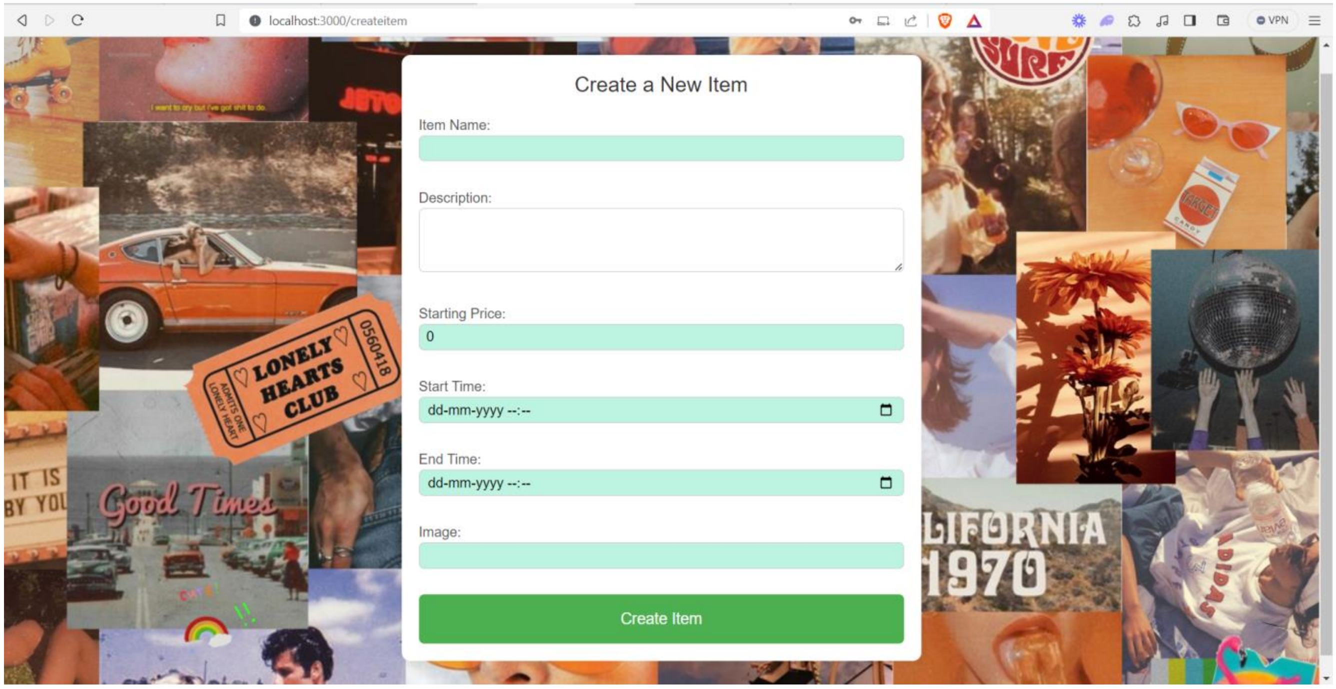


Seller Sign In

Username:

Password:

10.4 /createitem



Create a New Item

Item Name:

Description:

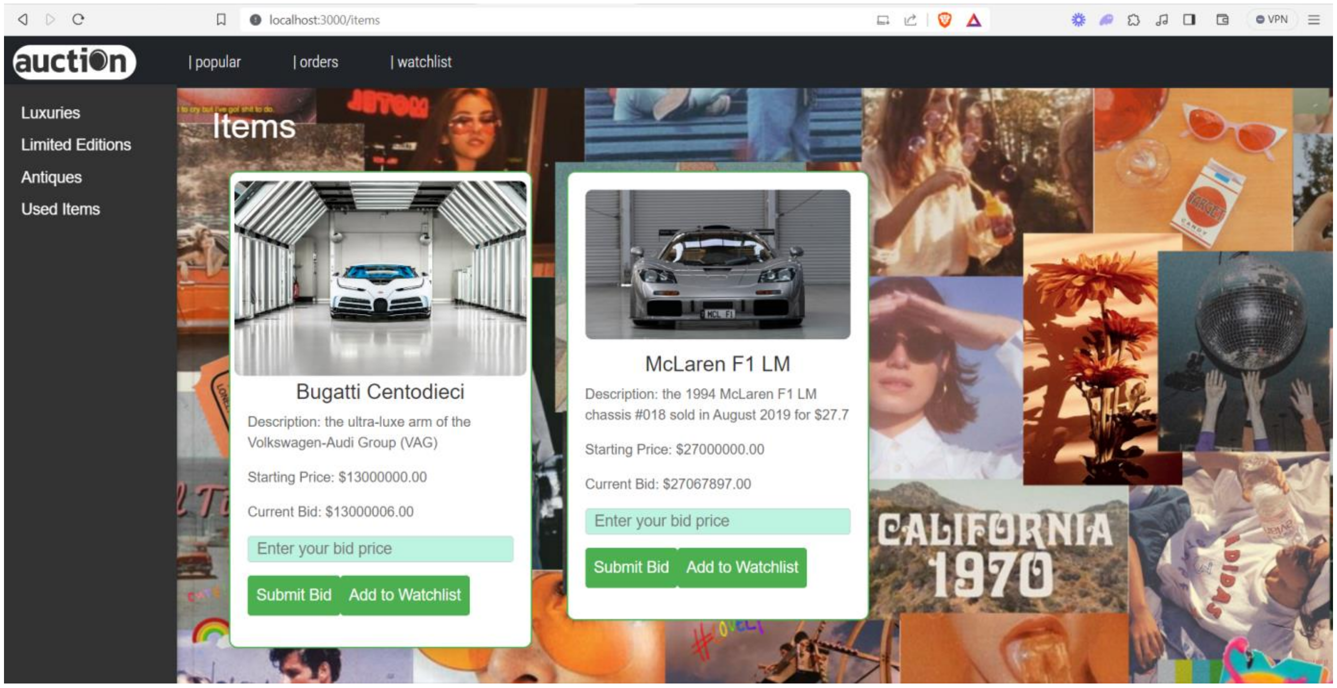
Starting Price: 0

Start Time: dd-mm-yyyy --:--

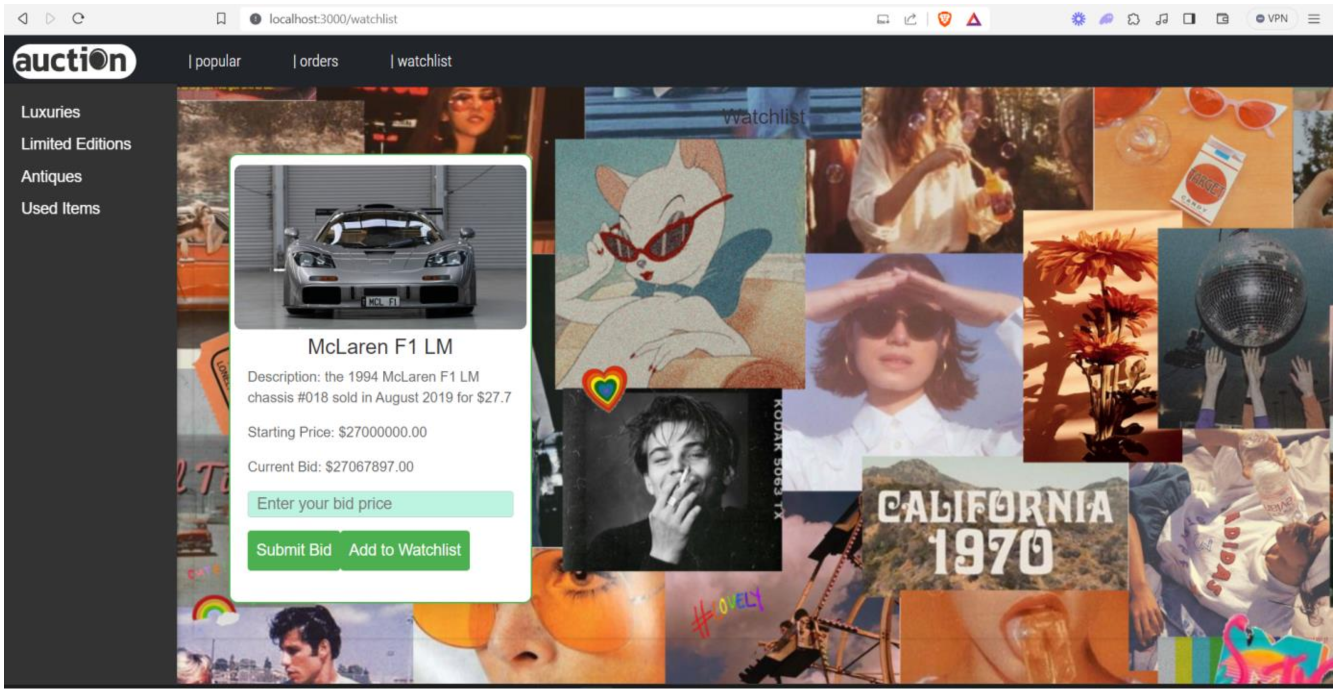
End Time: dd-mm-yyyy --:--

Image:

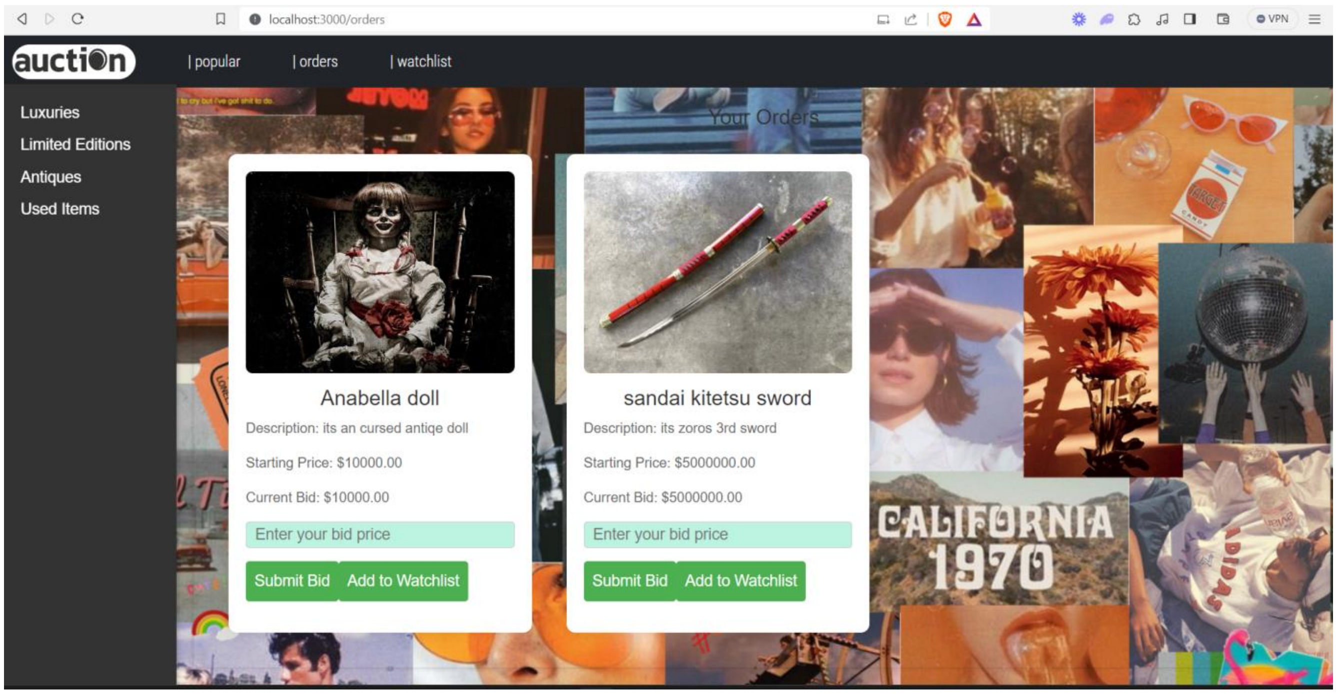
10.5 /items


localhost:3000/items
auction
| popular
| orders
| watchlist
Luxuries
Limited Editions
Antiques
Used Items
Items
Bugatti Centodieci
Description: the ultra-luxe arm of the Volkswagen-Audi Group (VAG)
Starting Price: $13000000.00
Current Bid: $13000006.00
Enter your bid price
Submit Bid
Add to Watchlist
McLaren F1 LM
Description: the 1994 McLaren F1 LM chassis #018 sold in August 2019 for $27.7
Starting Price: $27000000.00
Current Bid: $27067897.00
Enter your bid price
Submit Bid
Add to Watchlist

10.6 /watchlist



10.7 /orders



Your Orders

Anabella doll

Description: its an cursed antiqe doll

Starting Price: $10000.00

Current Bid: $10000.00

Enter your bid price

Submit Bid Add to Watchlist

sandai kitetsu sword

Description: its zoros 3rd sword

Starting Price: $5000000.00

Current Bid: $5000000.00

Enter your bid price

Submit Bid Add to Watchlist

REFERENCES

[1]simplelearn.com

[2]geeksforgeeks.com

[3]radixweb.com