DATABASE DESIGN

CS 6360.003

TITLE: Painting Auction Database System

Professor: Dr. Jalal Omer

Teaching Assistant : Maxwell Weinzierl

Project By:

Vivek Reddy Anthireddy (VXA200000)

Akshay Ramanujam Ranganathan (AXR200017)

Manikantan Arcot (MXA200072)

Chirag Rajesh Mukkatira (CRM210000)

Naveen Senthil Kumar (NXS200067)

Contents

Introduction

Over the years there has been significant growth in the field of art. With art receiving more attention, new forms and techniques of art have emerged leading to a growth in the number of artists. With the number of artists and art pieces increasing, the art connoisseurs and collectors need a site where they can purchase the artworks that they desire. To address this, we have decided to design and develop an online auction database system for auctioning paintings, each having different themes.

This system will enable any artist or seller to auction their paintings and buyers to bid on paintings that they are interested in.

The seller and buyer are identified by a unique member ID along with additional details such as name, password, phone number, home address which are all provided when they first register on the website.

Every buyer would need to add a shipping address to which the painting will be dispatched to on winning the auction which would be recorded in the database.

Similarly, every seller would need to add their respective bank details such as bank account number and routing number which would be recorded in the database, this would be used for sellers to receive the funds for their paintings from the winning bidder.

Each painting that is posted by the seller would come with a unique painting number assigned by the system, along with additional details such as date posted, title, description, theme, dimensions, starting bid price, bid increments, auction start date and end date.

The paintings are categorized by their respective themes. There are a total of three possible themes: Oil painting, Digital painting and sand painting which can be further extended if required.

The buyers can make bids on any number of paintings. When a buyer makes a valid bid, the bid price and time of bid is recorded. In addition, the history of bid price and time will be maintained. The buyer with the highest bid at the end of the auction would be announced as the winner and will be allowed to start a transaction with the seller to purchase the painting.

Once the transaction has been completed the buyer and seller can record feedback with a rating ranging from 1-10 of the other participating party in the transaction. Additionally, the buyer and seller can record feedback of their experience using our system which can be accessed only by the administrators, this can be used to then make any necessary enhancements or introduce new features in later updates.

The next section gives the system requirements including the context diagram along with the requirements to run the system. Section 3 provides the conceptual design of the database system where we can see the Entity-Relationship (ER) model as well as the constraints of the database system. In Section 4, we have information about the logical database schema along with the SQL statements used to construct the schema with the expected database operations and estimated data volumes. Section 5 contains the functional dependencies and data normalization that gives the normal form that the database is in along with mentioning all the dependencies present. A brief description about the database system is given in Section 6 where the steps to install and invoke the system are given. We talk about database tuning in Section 7. Sections 8 and 9 cover some additional queries and views and the user interface of the database system which the buyers and sellers mainly

interact with. Finally we conclude the topic of our database system project in Section 10 followed by the references and appendix.

1. System requirements

1.1. Context diagram

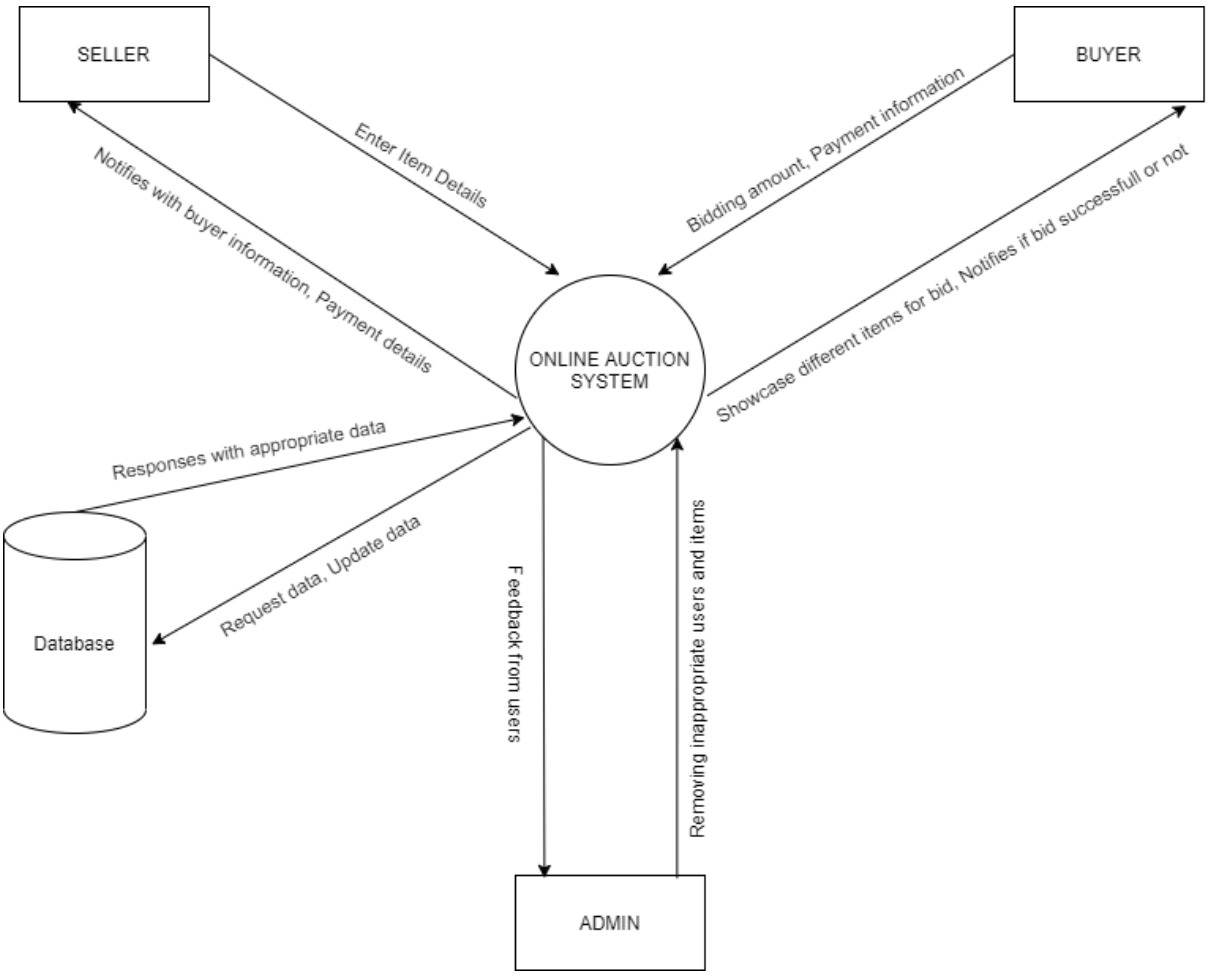


Fig 1: Context diagram of the painting auction database system

1.2. Interface Requirements

- User interacts with the login interface, on successfully entering login credentials user is given access to the auction site.
- Login interface interacts with the user, after the user enters login credentials login interface checks and verifies credentials and gives access to auction website.
- User interacts with the register interface, if the user is new then user can create an account by providing all required details.
- Register interface interacts with user, takes user details and stores in database and allows access to auction site as new user

- Register interface determines type of account of the user, i.e., either buyer or seller
- User interacts with the auction website, based on login credentials and type of account selected at registration, user can be buyer or seller.
- Buyer interacts with the auction site; buyer can view all items on auction and choose which to bid for.
- Seller interacts with the item page; seller can add details about item being sold and can also update and modify said items at any time.
- Buyers and sellers can view their profile details when logged into their account.
- Admin interacts with the database; admins can perform maintenance and manually make changes to the database as and when needed by adding new tables.
- Buyer interacts with the live auction page; buyer is able to place bids on the item and can view the status of bid.
- Auction site interacts with users, provides users with information regarding all available items and allows them to buy and sell their paintings.
- Users interact with the feedback page, buyers provide feedback to the sellers.
- Feedback page interacts with the database, Feedback is stored in the database and made available to the admin and seller.

1.3. Functional Requirements

1.3.1. Login Functional requirements

- System will have a login page where the user enters his/ her details to gain access to the auction site.
- Login page asks for the user's username and password.
- System will check for the correct username and password of each individual, both for buyer and seller.
- System will be able to store each user's username and password, so that it can check and allow them to pass.
- System will allow each user to create a buyer and a seller account by having them select the type of account at registration.
- Once the users are done with their job the system allows them to logout from the application.
- Sellers, with help of their accounts will be able to see all their sold paintings after the auction.
- System will help the buyers to see their purchases.
- System will ask for each and every buyer and seller feedback (Optional).
- System will store all their feedback.

1.3.2. Browsing Functional requirements

- The buyer will bid on the desired item after browsing through all the categories.
- With the help of the system the seller will post paintings of three mentioned categories to auction their product.
- System helps the buyer to see all his bids and go through them.
- System helps the buyer to see all his expired bids.
- System helps the buyer to see his purchase history.
- With help of the system the seller will be able to see all his products that went on auction.

- Sellers will be able to see their selling history.

1.3.3. Administrator Functional Requirements

- Admin is the one who has control over all products, he can view all the products.
- Admin has the authority to search and delete any product if needed.
- If anything goes wrong in the auction, the admin is the one who resolves it.
- If needed, the admin can add new categories to the database.
- Admin can view all buyers and sellers.
- Admin will have the authority to delete a buyer or a seller if he or she goes against the application policy
- Admin will be able to see all buyers and sellers contact messages.
- Admin will be able to see both buyer's purchase history and seller's selling history.
- Admin will be able to view both buyers and sellers feedback about the auction experience.

1.4. Non-Functional Requirements

- Performance: Loading of web pages is fast and search for the results based on filters. Fast querying and running smoothly.
- Scalability: The website is active even with 1000 users logged at once. It can be easily made to support a higher number of users by improving hardware.
- Portability and Compatibility: Accessing the website across all the web browsers and also all devices.
- Reliability: Data displayed is reliable and subjected to privacy conditions. System does not crash.
- Recoverability: Backup of data. No loss of data stored in the database.
- Security: Data collected and stored is secured.
- Data Integrity: The Relational Database design will have proper constraints to maintain correctness of the data.
- Usability: User friendly interface with minimalistic design.
- Interoperability:

  Data operated has the context of items put for auction with the user data.

  Data operated has the context of reviews from buyers with the seller data.

3. Conceptual design of the database

3.1. Entity Relationship Diagram

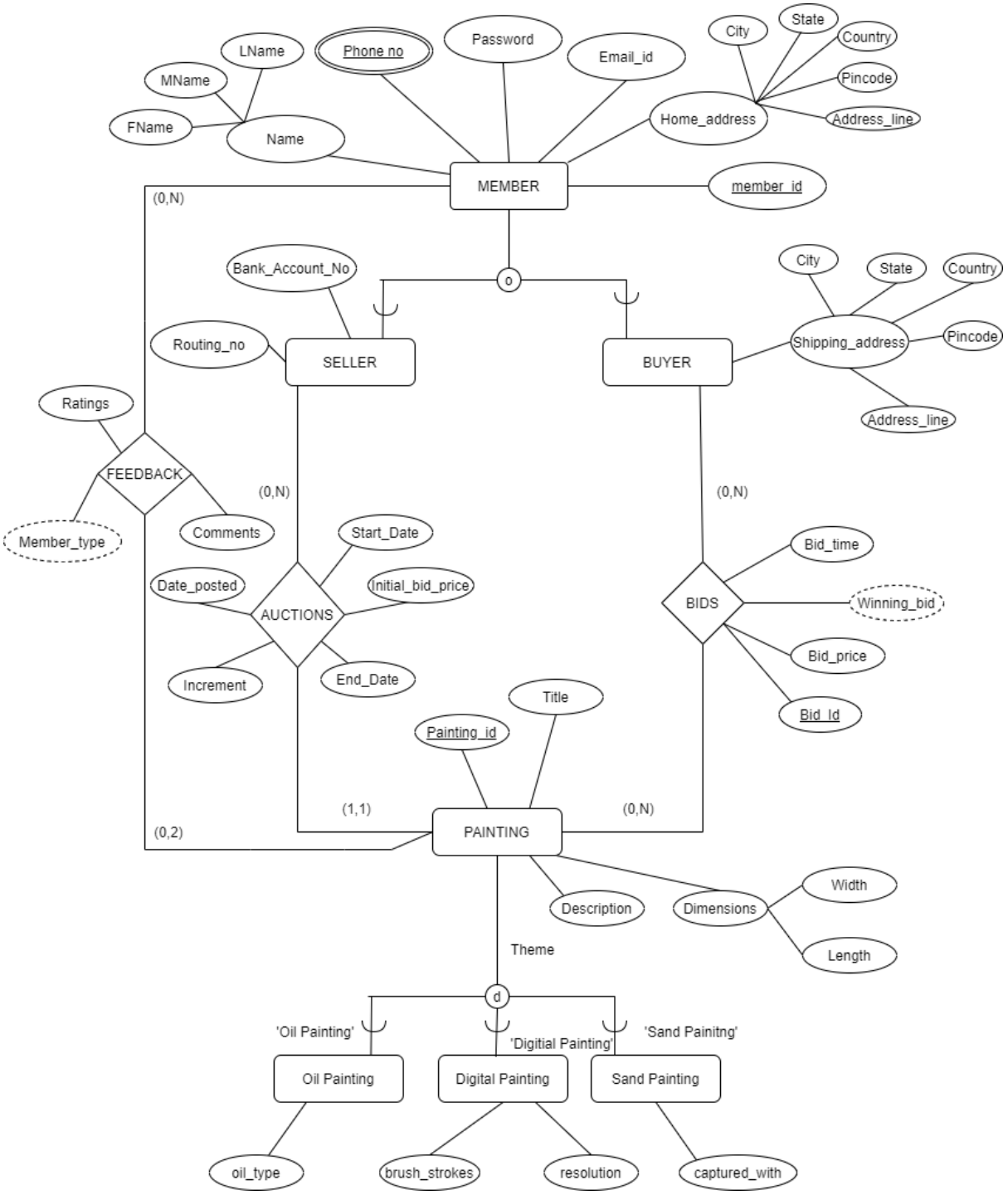


Fig 2: Entity Relational model of the painting auction database system

3.2. Business rules/ constraints

The following constraints have been implemented in the database system:

- **Key constraints**
  The keys are used to identify an entity from an entity set uniquely. The key constraints ensure that the key values are unique.
  This is implicitly taken care of by the DBMS when we define an attribute as a primary key.
  Example:
  Member Entity: member_id is the key
  Painting Entity: painitng_id is the key
- **Domain constraints**
  The domain is the valid set of values for an attribute. This constraint ensures that all attributes' values are within the domain.
  Example:
  title attribute in our Painting entity has VARCHAR(20) which ensures that the title does not exceed 20 characters.
  start_date attribute in our Painting entity has datetime which ensures that the attribute accepts only datetime datatype.
- **Entity integrity constraints**
  This constraint ensures that the primary key value is never NULL. This is implicitly taken care of by the DBMS when we define an attribute as a primary key.
- **Referential integrity constraints**
  This constraint is specified between two relations. In the Referential integrity constraints, if a foreign key in Table 2 refers to the Primary Key of Table 1, then every value of the Foreign Key in Table 2 must be null or be available in Table 2. This is taken care of by the DBMS by appropriately referencing the desired primary key.
  Example:
  Since our Painting entity has an attribute seller_id which references the Seller entities' member_id, the value in seller_id must be either null OR present in the Seller table.

4. Logical Database Schema

4.1. Schema

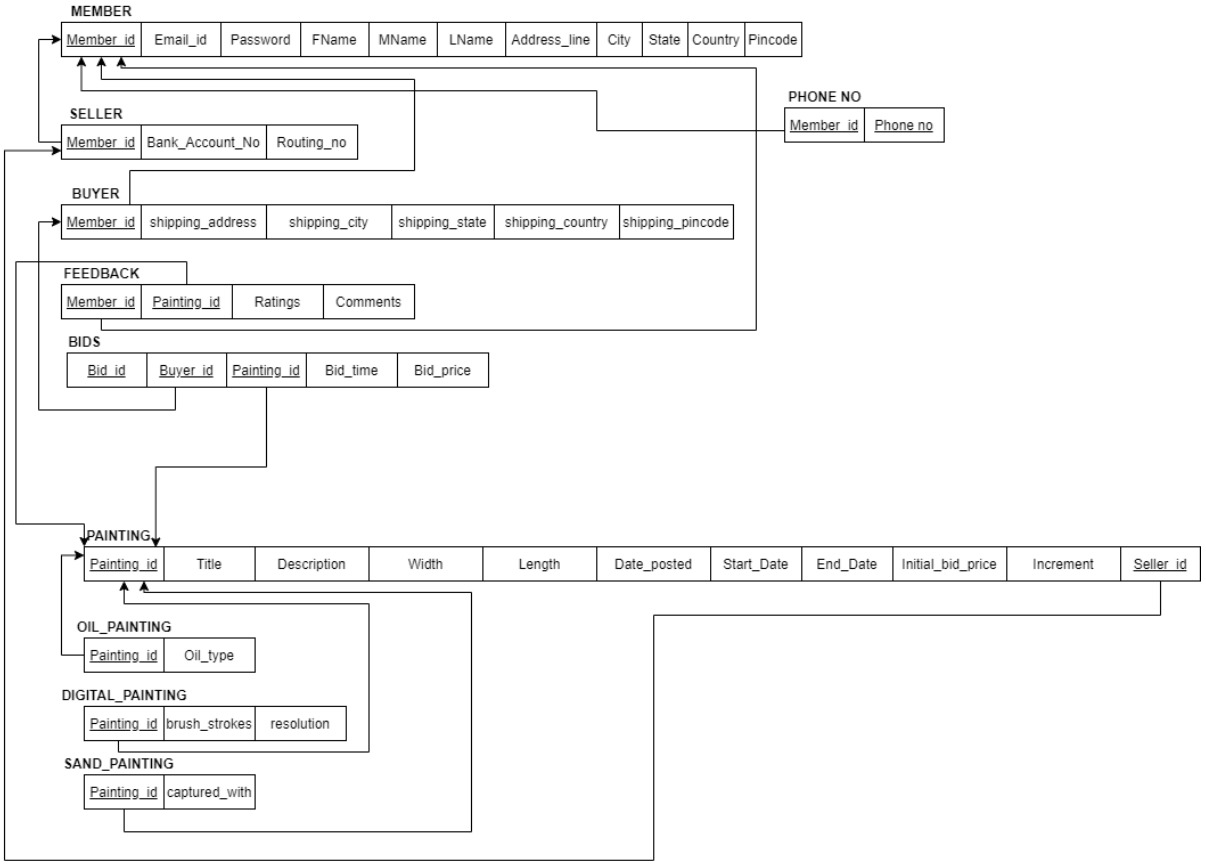


Fig 3: Relational database schema of the painting auction database system

4.2. SQL Statements and Database Operations

SQL Queries For Table Creation:

```
CREATE SCHEMA AuctionSystem;
USE AuctionSystem;

CREATE TABLE member (
member_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
email_id VARCHAR(50) UNIQUE,
```

```
password VARCHAR(20) NOT NULL,
fname VARCHAR(30) NOT NULL,
mname VARCHAR(10),
lname VARCHAR(30) NOT NULL,
address VARCHAR(70) NOT NULL,
city VARCHAR(20) NOT NULL,
state VARCHAR(10) NOT NULL,
country VARCHAR(10) NOT NULL,
pincode INT NOT NULL);

CREATE TABLE seller (
member_id INT NOT NULL PRIMARY KEY REFERENCES member(member_id),
bank_account_no INT NOT NULL UNIQUE,
routing_no INT NOT NULL UNIQUE);

CREATE TABLE buyer (
member_id INT NOT NULL PRIMARY KEY REFERENCES member(member_id),
shipping_address VARCHAR(70) NOT NULL,
shipping_city VARCHAR(20) NOT NULL,
shipping_state VARCHAR(10) NOT NULL,
shipping_country VARCHAR(10) NOT NULL,
shipping_pincode INT NOT NULL);

CREATE TABLE painting(
painting_id INT NOT NULL AUTO_INCREMENT,
seller_id INT NOT NULL,
title VARCHAR(20) NOT NULL,
description VARCHAR(70) NOT NULL,
width FLOAT,
length FLOAT,
date_posted DATETIME NOT NULL,
```

```
start_date DATETIME NOT NULL,
end_date DATETIME NOT NULL,
initial_bid_price FLOAT NOT NULL,
increment INT NOT NULL,
PRIMARY KEY(painting_id),
FOREIGN KEY(seller_id) REFERENCES seller(member_id));

CREATE TABLE feedback (
member_id INT NOT NULL,
painting_id INT NOT NULL,
ratings INT,
comment VARCHAR(280),
PRIMARY KEY(member_id, painting_id),
FOREIGN KEY(member_id) REFERENCES member(member_id),
FOREIGN KEY(painting_id) REFERENCES painting(painting_id));

CREATE TABLE bids (
bid_id INT NOT NULL AUTO_INCREMENT,
buyer_id INT NOT NULL,
painting_id INT NOT NULL,
bid_time DATETIME,
bid_price INT NOT NULL,
PRIMARY KEY(bid_id, buyer_id, painting_id),
FOREIGN KEY(painting_id) REFERENCES painting(painting_id),
FOREIGN KEY(buyer_id) REFERENCES buyer(member_id));

CREATE TABLE oilpainting (
painting_id INT NOT NULL
PRIMARY KEY REFERENCES painting(painting_id),
oil_type VARCHAR(50));
```

```
CREATE TABLE digitalpainting (
painting_id INT NOT NULL
PRIMARY KEY REFERENCES painting(painting_id),
brushstrokes INT NOT NULL,
resolution INT NOT NULL);

CREATE TABLE sandpainting (
painting_id INT NOT NULL
PRIMARY KEY REFERENCES painting(painting_id),
captured_with VARCHAR(10) NOT NULL);

CREATE TABLE phoneno (
member_id INT NOT NULL,
phoneno INT NOT NULL,
PRIMARY KEY(member_id, phoneno),
FOREIGN KEY(member_id) REFERENCES member(member_id));
```

4.2.1. INSERT STATEMENTS For Dummy Data:

```
INSERT INTO member (member_id, email_id, password, fname, mname,
lname, address,city, state,country, pincode)
VALUES
(1, "p@gmail.com", "a", "vivek", "a", "reddy", "courtyards","dallas",
"texas","US", 75248),
(2, "q@gmail.com", "aa", "mani", "a", "kumar", "eof","CA", "texas","US",
75249),
(4, "r@gmail.com", "aaa", "akshay", "a", "ram", "pearl","SA", "texas","US",
75250),
(5, "s@gmail.com", "aaaa", "naveen", "a", "shetty", "marquis","dallas",
"texas","US", 75243),
(6, "t@gmail.com", "ab", "chirag", "a", "S", "cityline","dallas", "texas","US",
75241),
```

(7, "u@gmail.com", "abc", "chirag", "a", "A", "ashwood","plano",

"texas","US", 75242),

(8, "v@gmail.com", "abcd", "vikky", "a", "B", "eof","frisco", "texas","US",

75246),

(9, "w@gmail.com", "ae", "manu", "a", "D", "cityline","plano", "texas","US",

75253);

Note:

In the dummy data, member id's 1,5,6 are sellers and 2,9 are buyers

INSERT INTO seller (member_id ,bank_account_no ,routing_no)

VALUES

(1,1234567,3451278),

(5,5678949,1239806),

(6,4098898,1287956);

INSERT INTO buyer (member_id
,shipping_address,shipping_city,shipping_state,shipping_country,shipping_

pincode)

VALUES

(2,"23103courtyards","dallas","texas","USA",12345),

(9,"1405cityline","charlotte","Carolina","USA",34567);

INSERT INTO painting (painting_id, title, description, width, length,

date_posted, start_date,end_date, initial_bid_price,increment, seller_id)

VALUES

(1, "The Desperate Man", "AAAA", 4, 5, "2021-10-2 03:14:07","2021-11-30

03:14:07", "2021-11-03 03:14:07",500, 50,1),

(2, " The Short Winter ", "BBBB", 3, 6, "2021-10-28 04:14:07", "2021-11-29

04:14:07","2021-10-31 04:14:07", 600,50, 6),

(3, "Thunder Bird", "CCCC", 6, 5, "2021-10-27 05:14:07", "2021-11-28 05:14:07","2021-10-31 05:14:07", 700,50, 5),
(4, "Sun and Eagle", "DDDD", 7, 8, "2021-10-26 06:14:07", "2021-10-27 06:14:07","2021-10-29 06:14:07", 200,50, 5),
(5, "Coyote Steeling Fire", "EEEE", 5, 6, "2021-10-25 07:14:07", "2021-10-26 07:14:07","2021-10-28 05:14:07", 600,50, 6),
(6, "The Long Winter", "FFFF", 9, 8, "2021-10-24 08:14:07", "2021-10-25 08:14:07","2021-10-29 07:14:07", 300,50, 1),
(7, "The Titanic", "GGGG", 5, 6, "2021-10-23 09:14:07", "2021-11-24 09:14:07","2021-11-28 05:14:07", 800,50, 1),
(8, "Stories in Snow", "HHHH", 5, 7, "2021-10-29 01:14:07", "2021-11-30 01:14:07","2021-12-05 05:14:07", 900,50, 5),
(9, "Golden Stillness", "IIII", 8, 9, "2021-10-25 02:14:07", "2021-11-26 02:14:07","2021-12-28 05:14:07", 1000,50, 6),
(10, "Another Day Gone", "JJJJ", 8, 9, "2021-10-24 03:14:07", "2021-11-25 03:14:07","2021-12-28 05:14:07", 500,50, 5);

```
INSERT INTO oilpainting (painting_id, oil_type)
VALUES
(5, 'sufffloweroil'),
(8, 'walnut oil'),
(10, 'poppy seed oil');

INSERT INTO digitalpainting (painting_id, brushstrokes, resolution)
VALUES
(1, 20,1080),
(3, 25,1440),
(6, 30,720);

INSERT INTO sandpainting (painting_id, captured_with)
VALUES
```

(2, 'photo'),

(7, 'video'),

(4, 'video'),

(9, 'photo');

INSERT INTO bids(bid_id,buyer_id,painting_id,bid_time,bid_price)

VALUES

(1,2,4,'2021-10-27 08:14:07',200),

(2,9,4,'2021-10-29 04:14:07',250),

(3,2,4,'2021-10-29 05:15:08',300),

(4,9,6,'2021-10-29 06:15:08',300);

Note: 2 won the bid on painting_id 4 and 9 won the bid on painting_id 6

INSERT INTO feedback(member_id,painting_id,ratings,comment)

VALUES

(2,4,4,"Yay won my first bid"),

(9,6,5,"Won without any competition");

INSERT INTO phoneno(member_id,phoneno)

VALUES

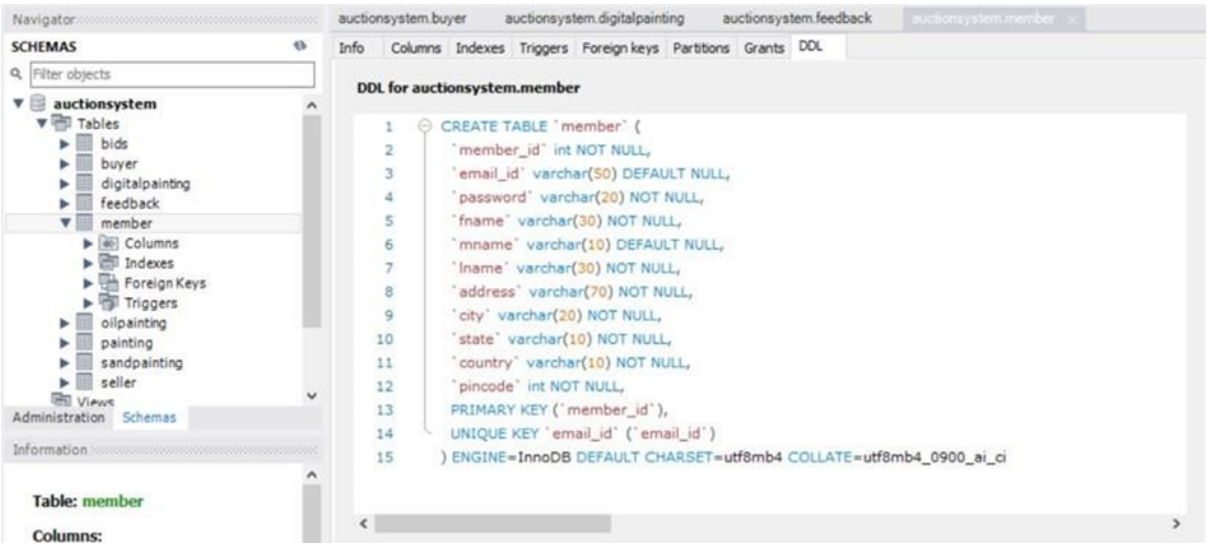
(1,99999999),

(2,98983419),

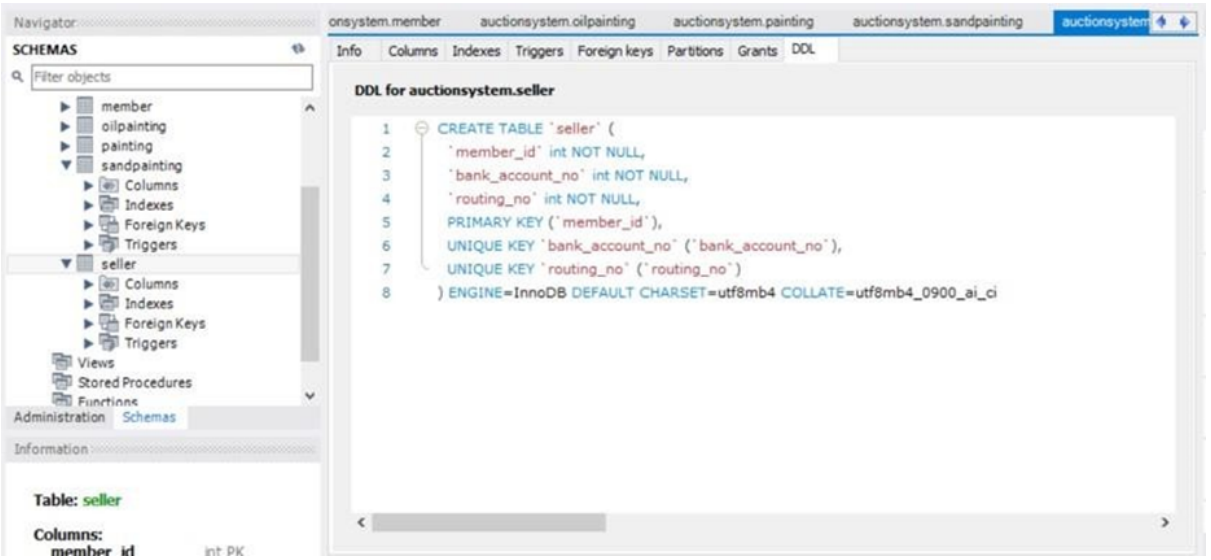
(2,12345678);

4.2.3. Table Creation Screenshots

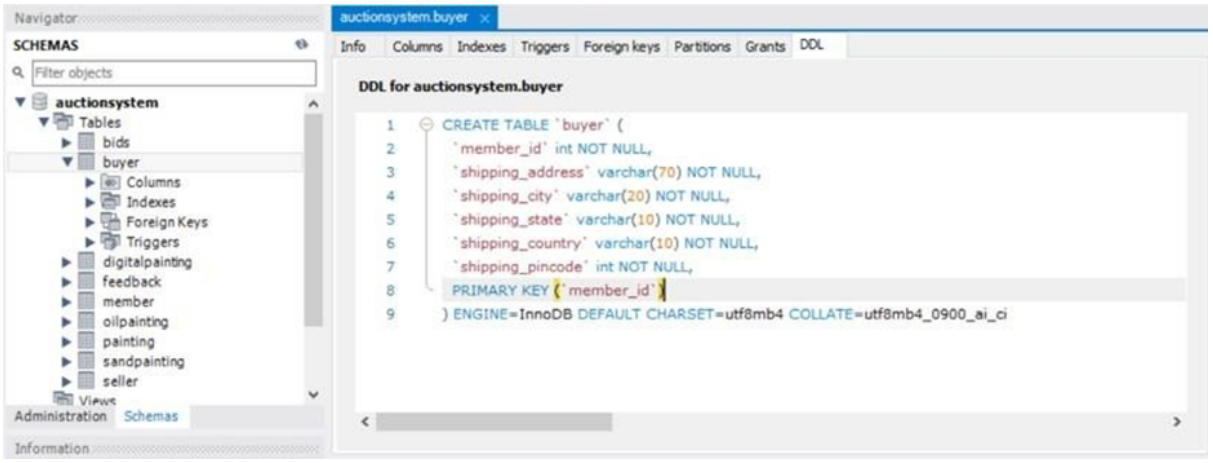
Member Table:



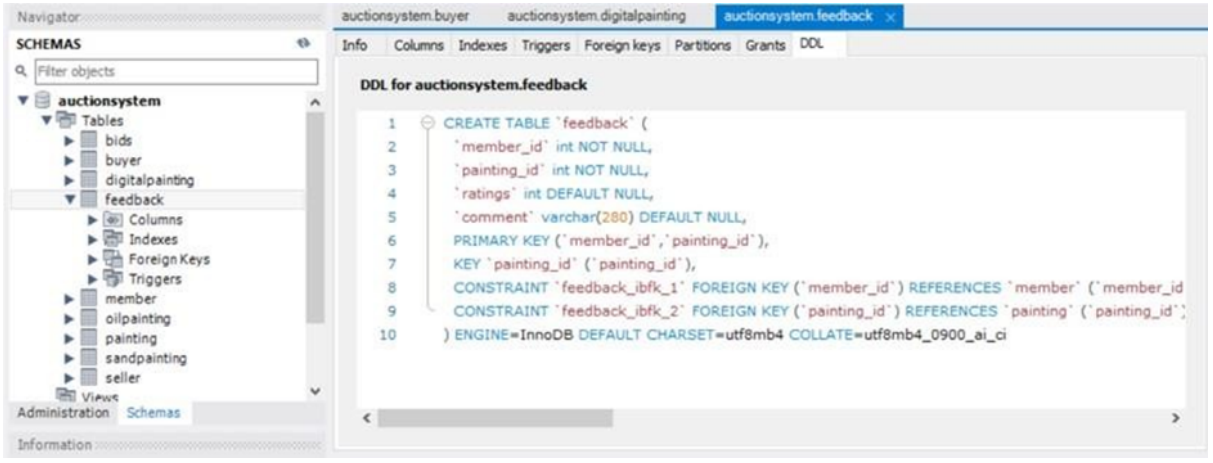
Seller Table



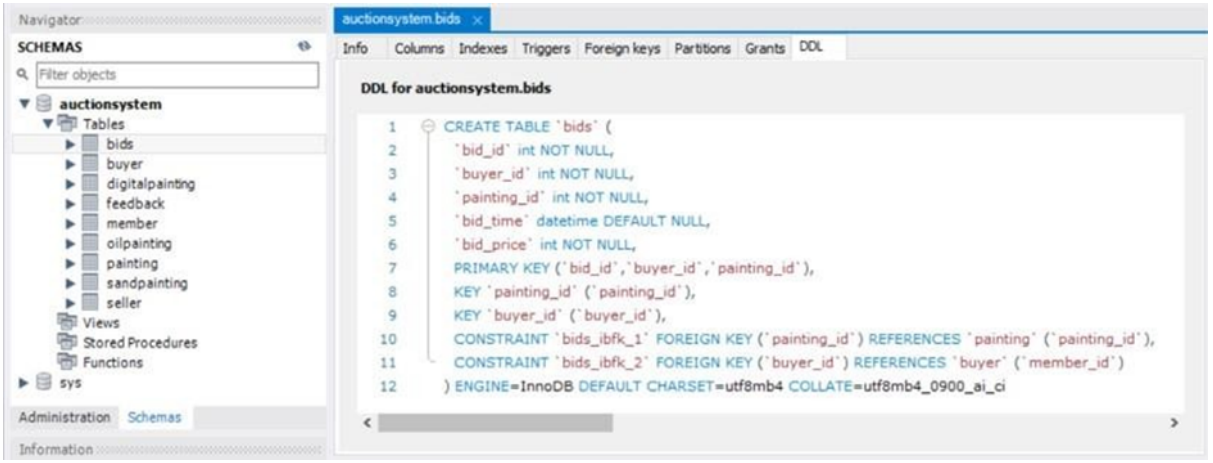
Buyer Table:



Feedback Table:

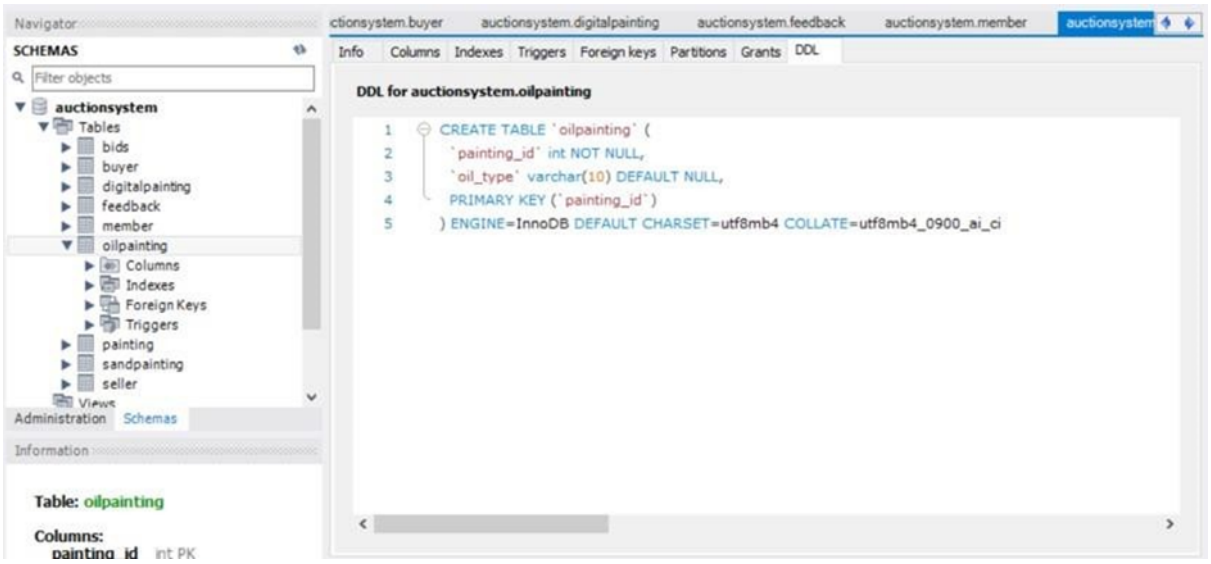


Bids Table:

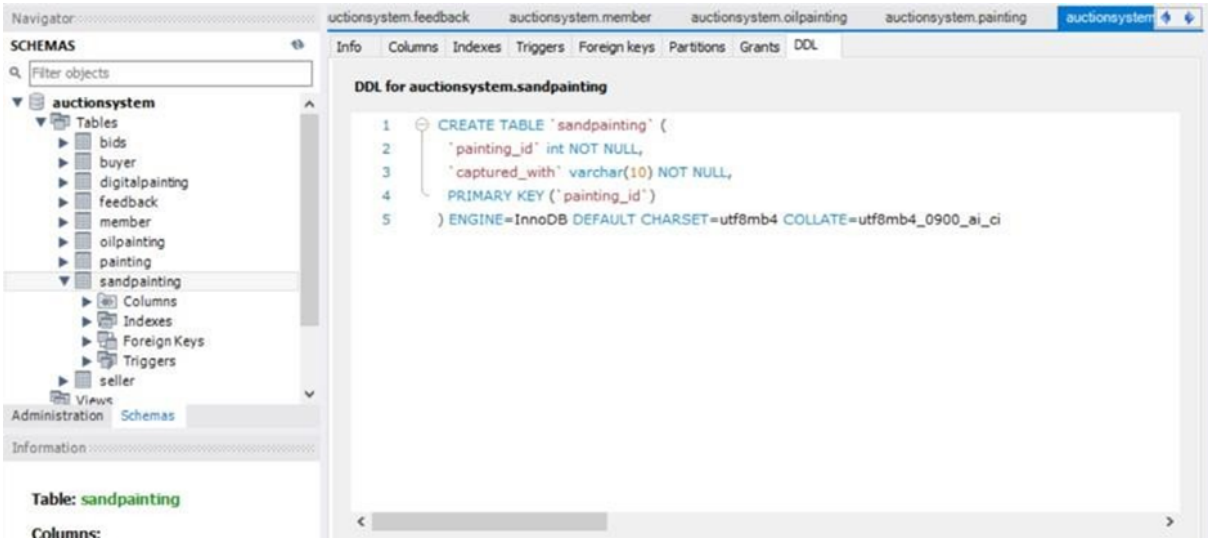


Painting Table:

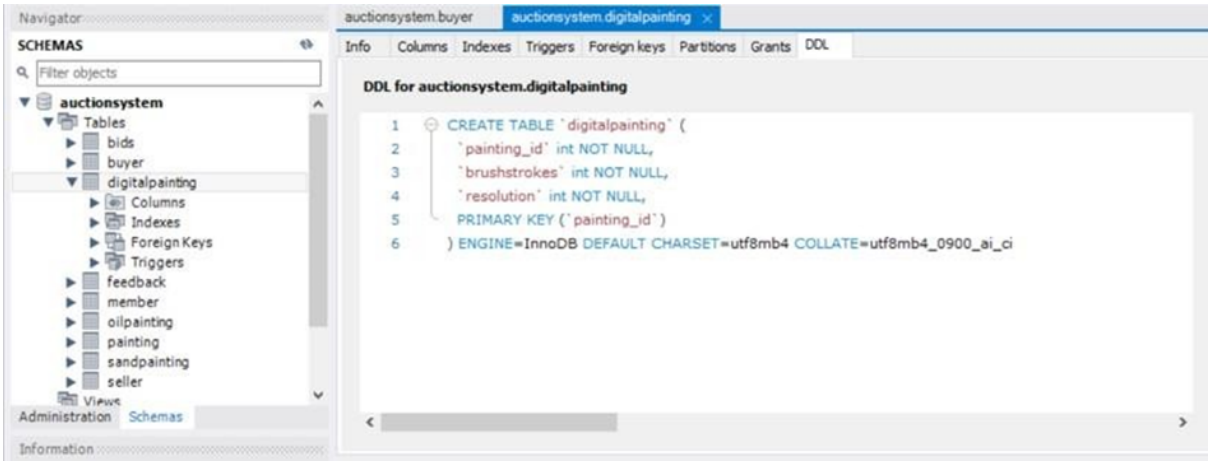
Oil Painting Table:



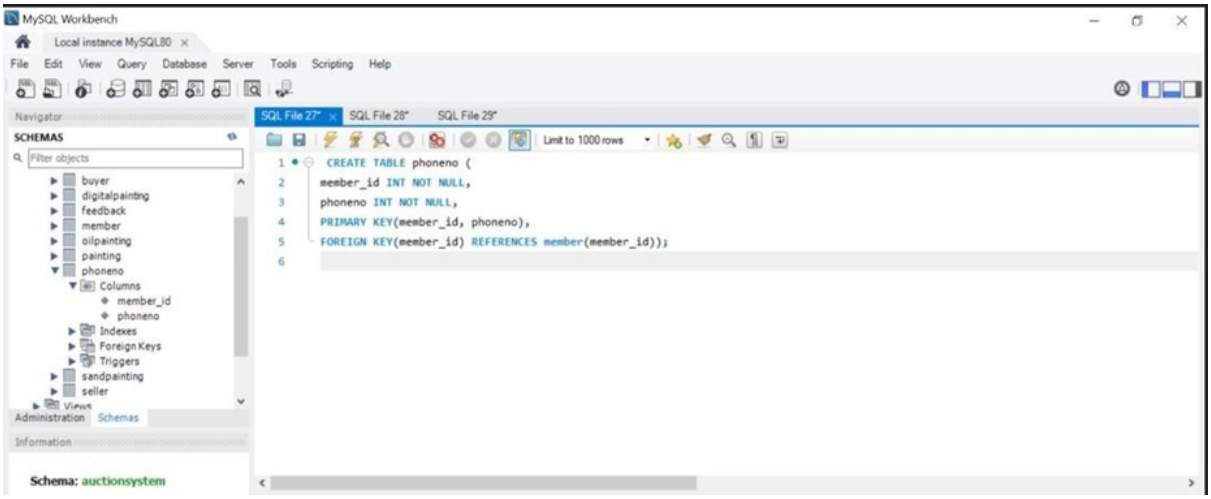
Sand Painting Table:



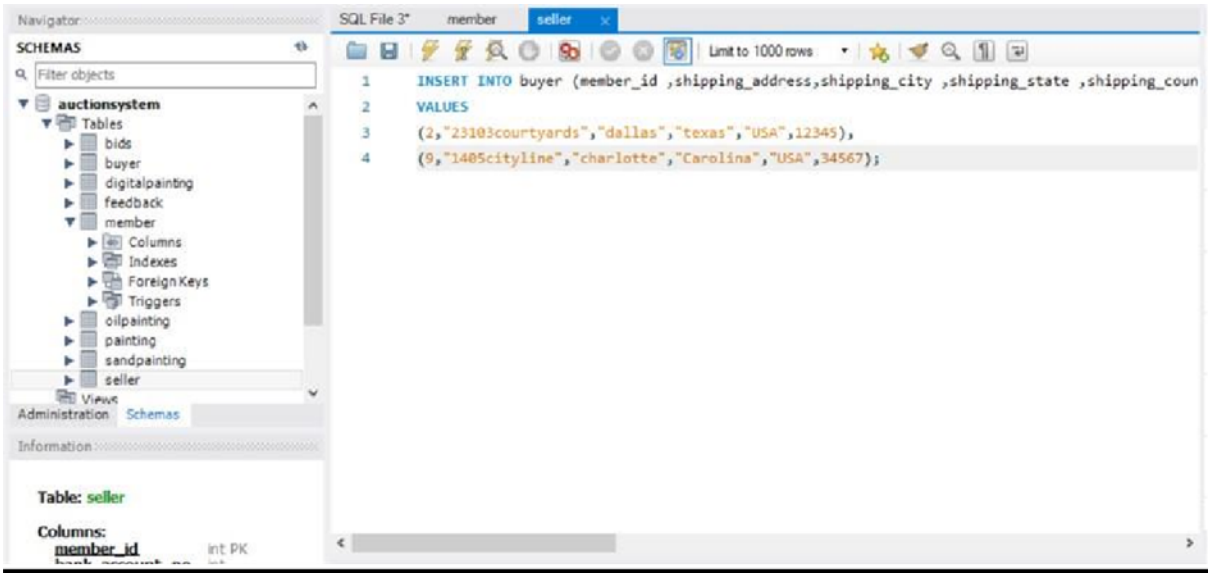
Digital Painting Table:



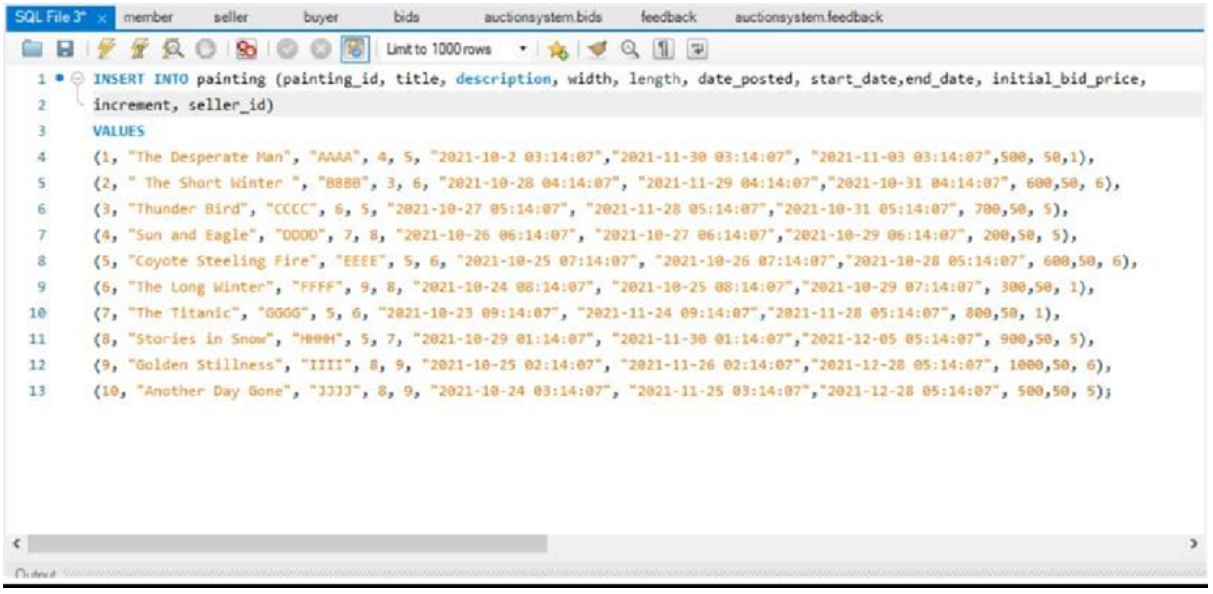
Phone number Table:



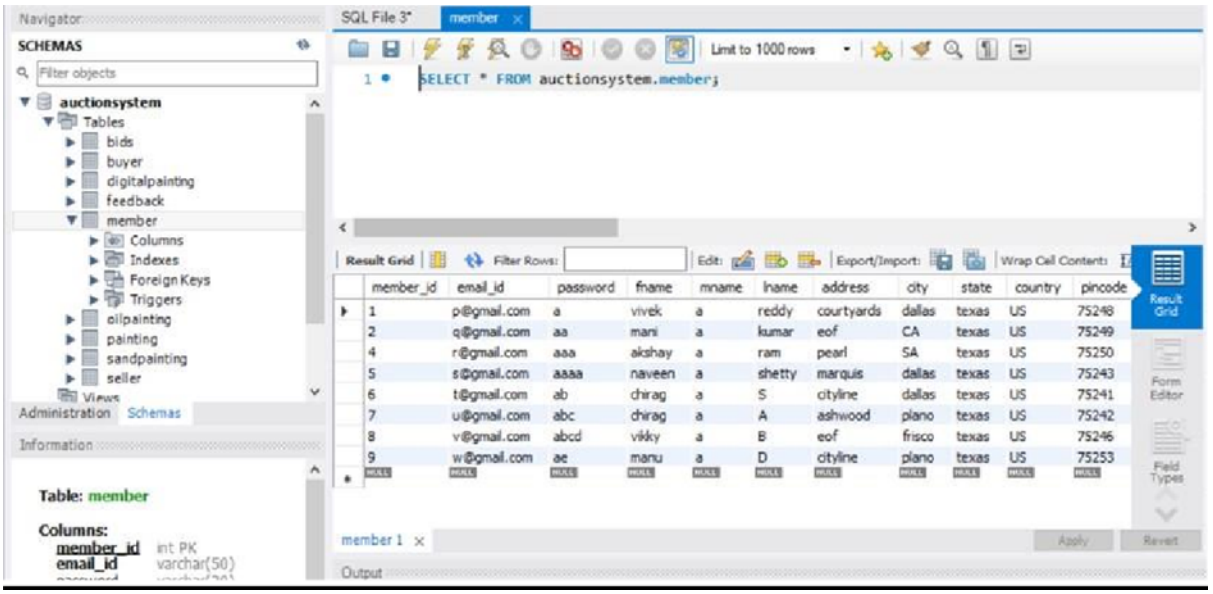
Insert into buyer



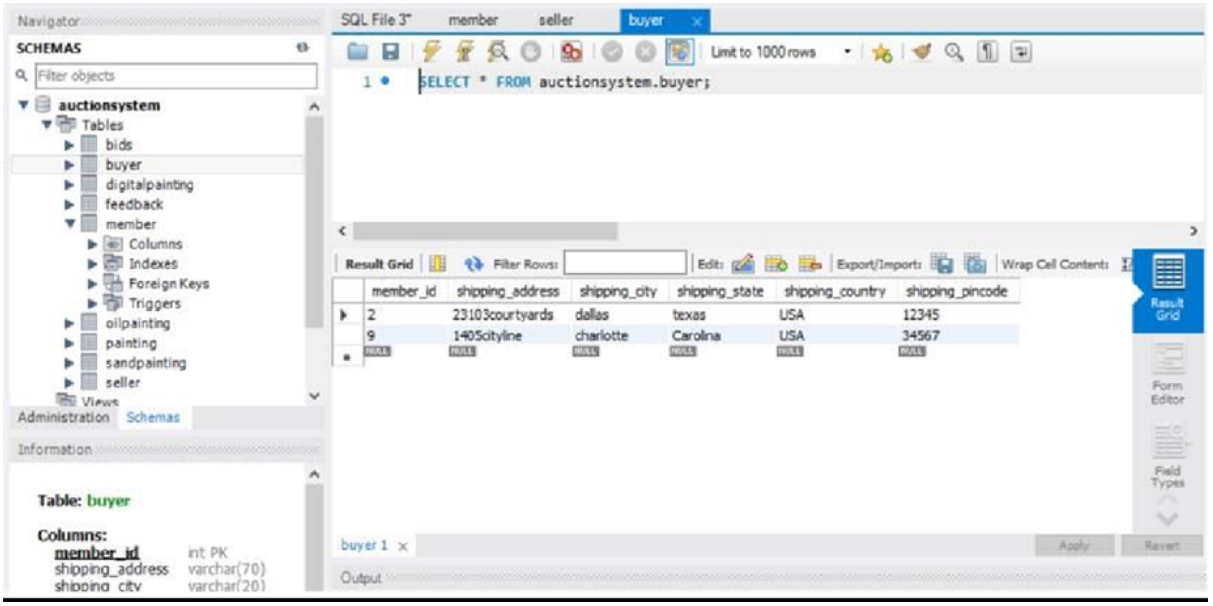
Insert into painting



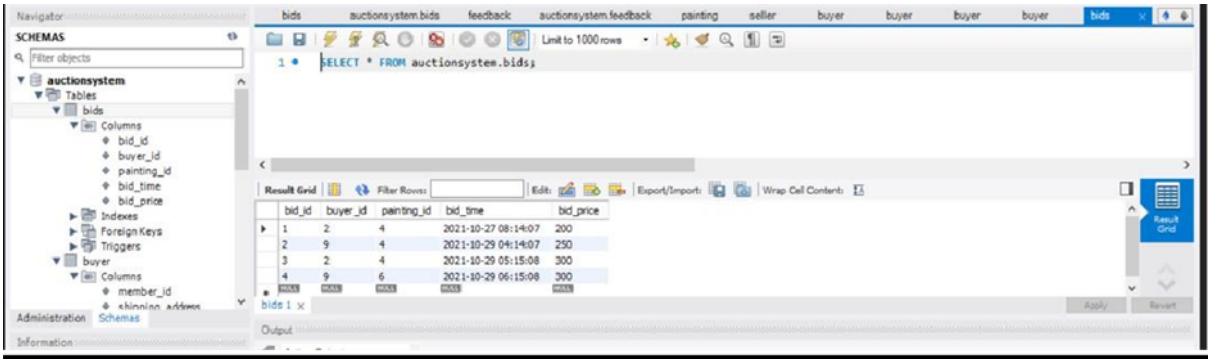
Select * from members



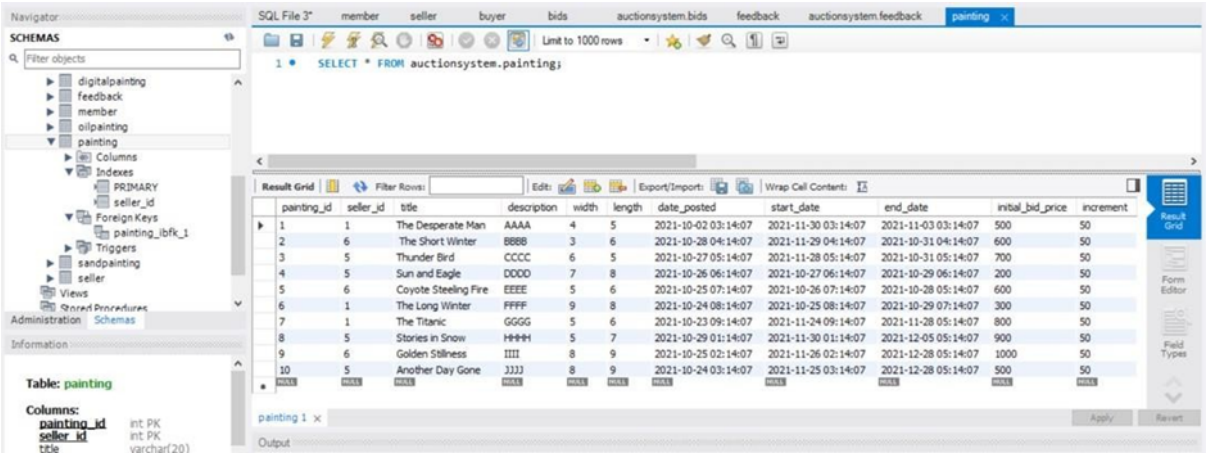
Select * from buyer



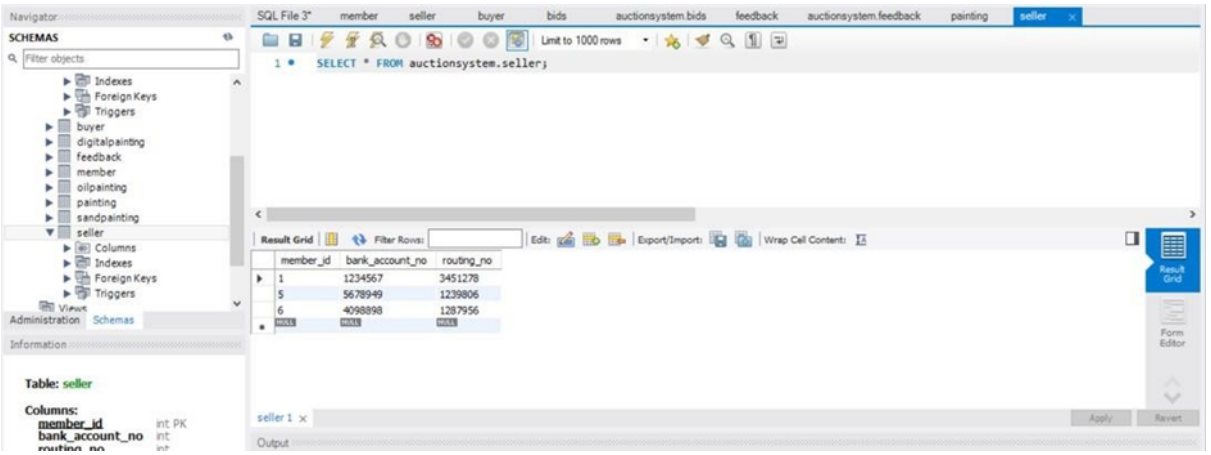
Select * from bids



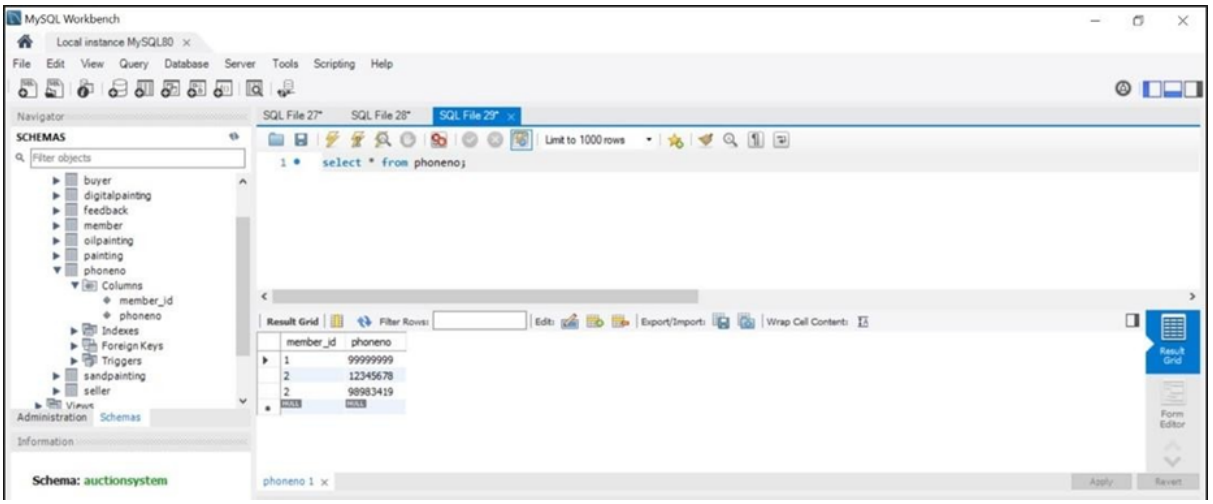
Select * from painting



Select * from seller



Select * from phoneno



5. Functional Dependency and Data Normalization

5.1. Functional Dependency

The functional dependency establishes a functional relationship between 2 sets of attributes X and Y where a value of X determines a Unique value of Y. it is denoted as,

X Y

Based on this assumption, below are the all possible functional dependencies that we have introduced in our auction system are as follows:

Member_ID --> Fname, Lname, Mname

Member_ID ⟶ Email_ID, Password

Member_ID⟶ Bank_Account_No, Routing_no

Member_ID ⟶ Address_line, Pincode, City, State , Country

Member_ID-> shipping_address, shipping_pincode, shipping_city, shipping_state,shipping_country

Member_ID,Painting_ID -> Ratings, Comments

Email_ID Password

Email_ID ->Member_id

Painting_ID Title

Painting_ID Description

Painting_ID Seller_id

Painting_ID⟶ Width, Length

Painting_ID⟶ Start_Date, End_Date, Date_Posted

Painting_Id⟶ Initial_bid_price, increment

Member_ID, Bid_ID,Painting_ID⟶ Bid_time, Bid_price

Painting_ID->Oil_type

Painting_ID->brush_strokes,resolution

Painting_ID->captured_with

5.2. Data Normalization

The construction of the database ensures that all the relations are in 1NF

All relations completely satisfy the property of 3NF where every non

prime attribute of Relation R meets the following conditions:

a) Fully Functional dependent on the key of R.

b) Non transitively dependent on every key of R.

Note: email is key for the Relation Member.

There are no nontrivial MVD's. So all relations even satisfy 4NF.

Our relation database was already normalized while designing the schema, thus there was no need of normalization of the tables.

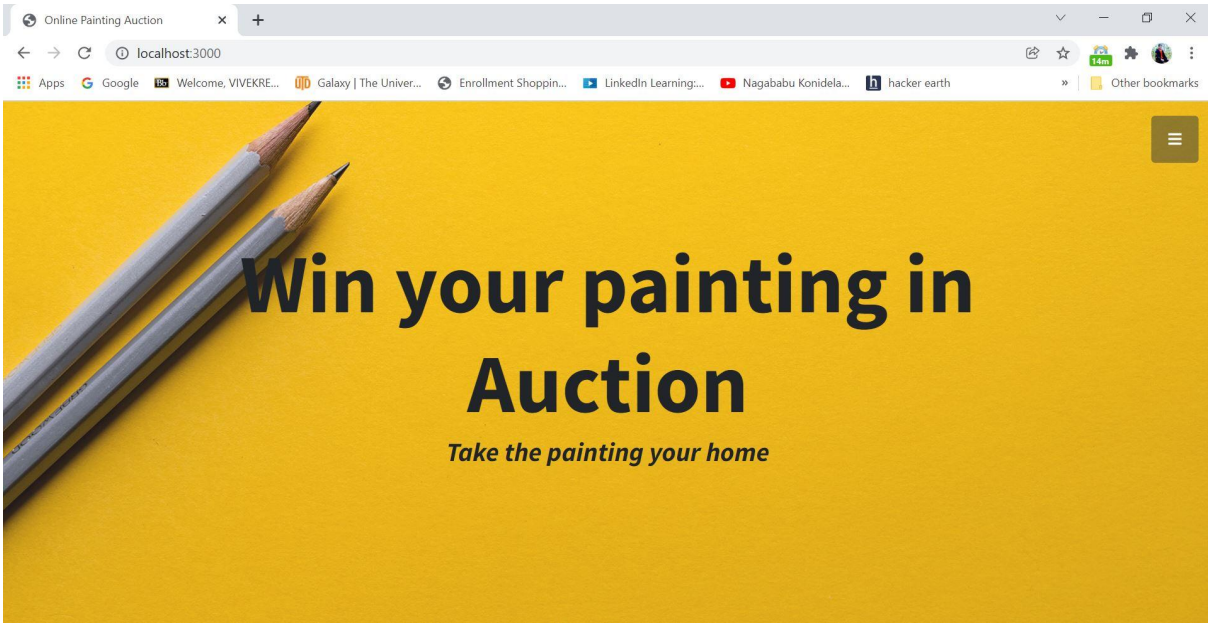
6. Database System

6.1. Installation and invoking of system

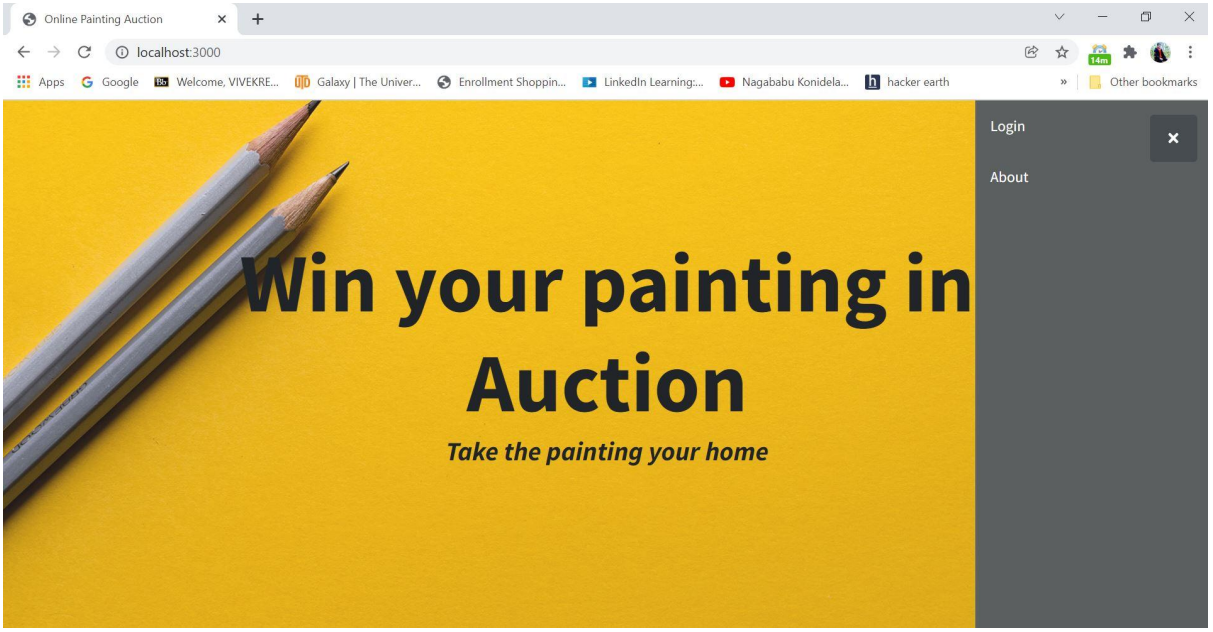
- Setup the Mysql database in the machine by executing the sql files shared in source code zip files.
- Install nodejs
- Download the source code (https://github.com/vrar02/online_auction_webapp).
- Run the command "npm install" in the project directory.
- Change the configuration of database connection information in config/index.js
- Run the command "nodemon start" to start the application.

6.2. Step-wise use of system

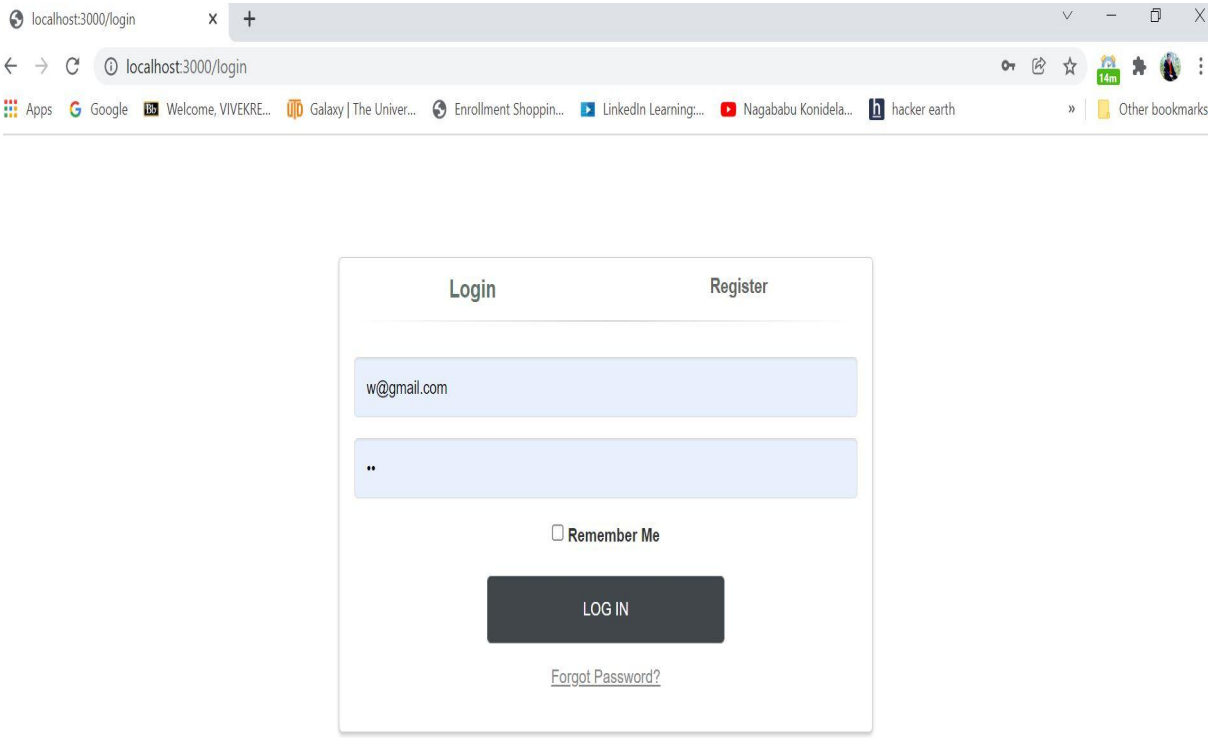
Design and working screenshots of the application below.



On opening the application, the user is greeted with the above screenshot. This screenshot displays the homepage for an auction system that focuses on paintings. The hamburger button on the top right lets the user login to their account.



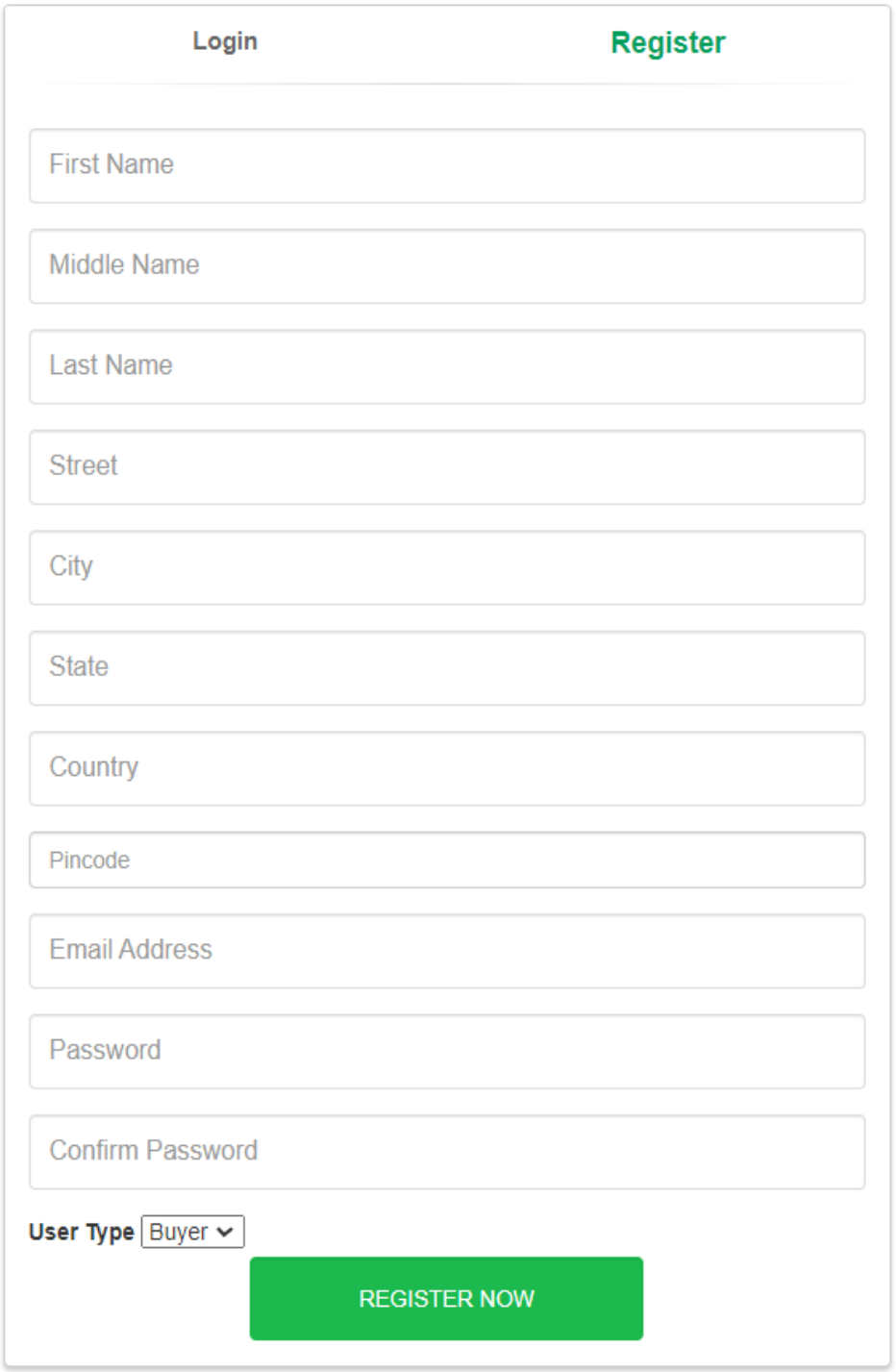
On clicking the hamburger button, the user has the option to login or to learn more about the website. Clicking login redirects the user to the login page.



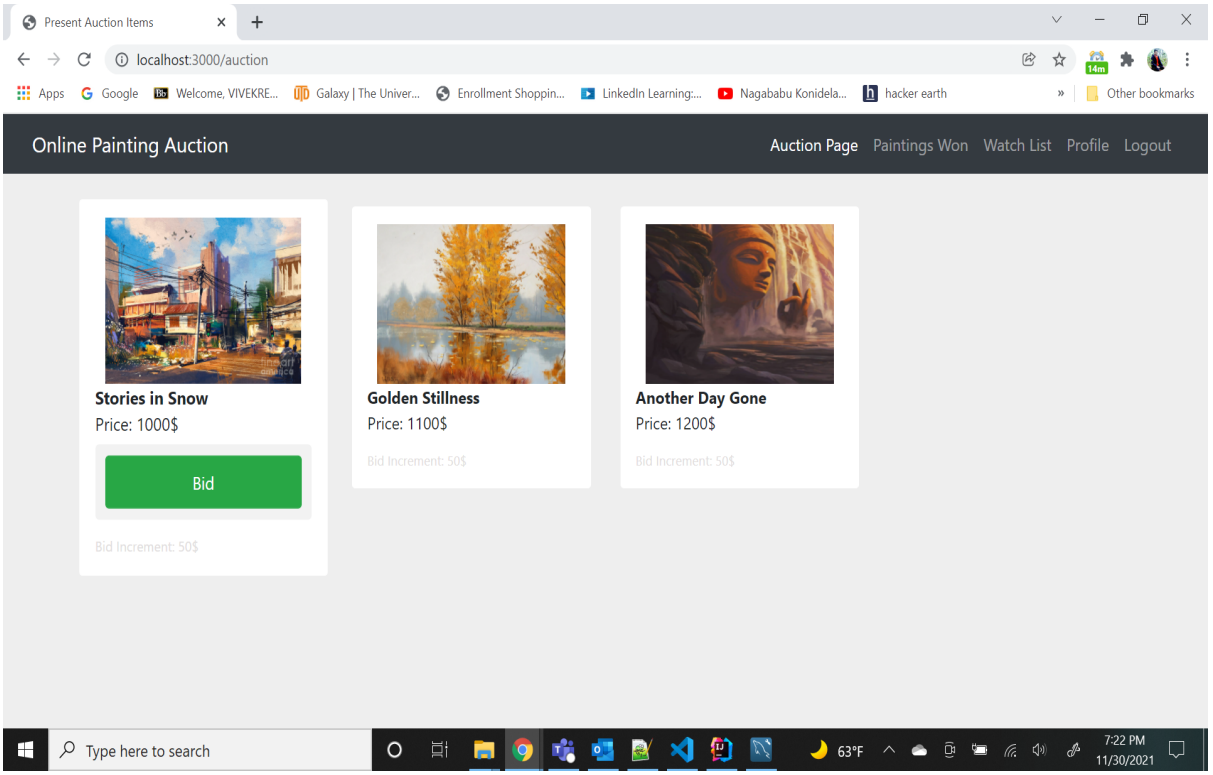
On the login page, if the user is an existing user, they just enter their login credentials. If the user is new, then they can register to the website through the register option. Upon entering the valid

credentials the user will be redirected to buyers/seller pages based on the account type given on registration.

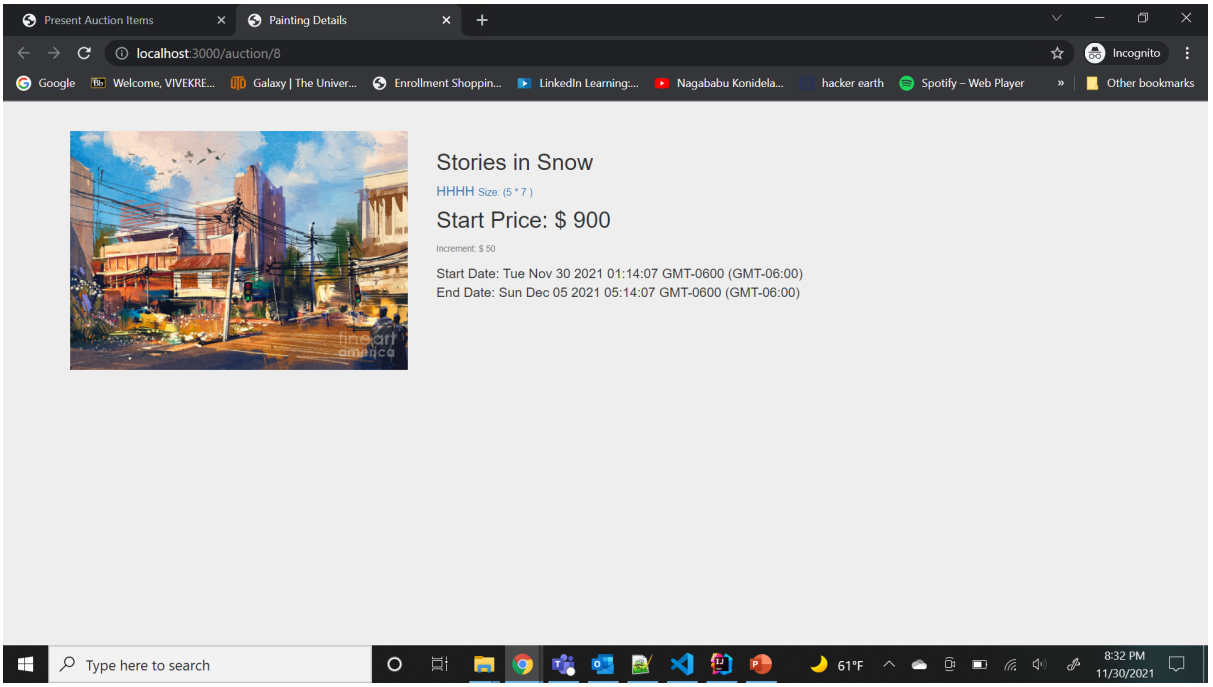
On going to the registration page, the user is asked to provide their details like name and address along with email and setting up their password. The user is asked to mention whether their account is to buy or sell based on which the functions of their account will vary.



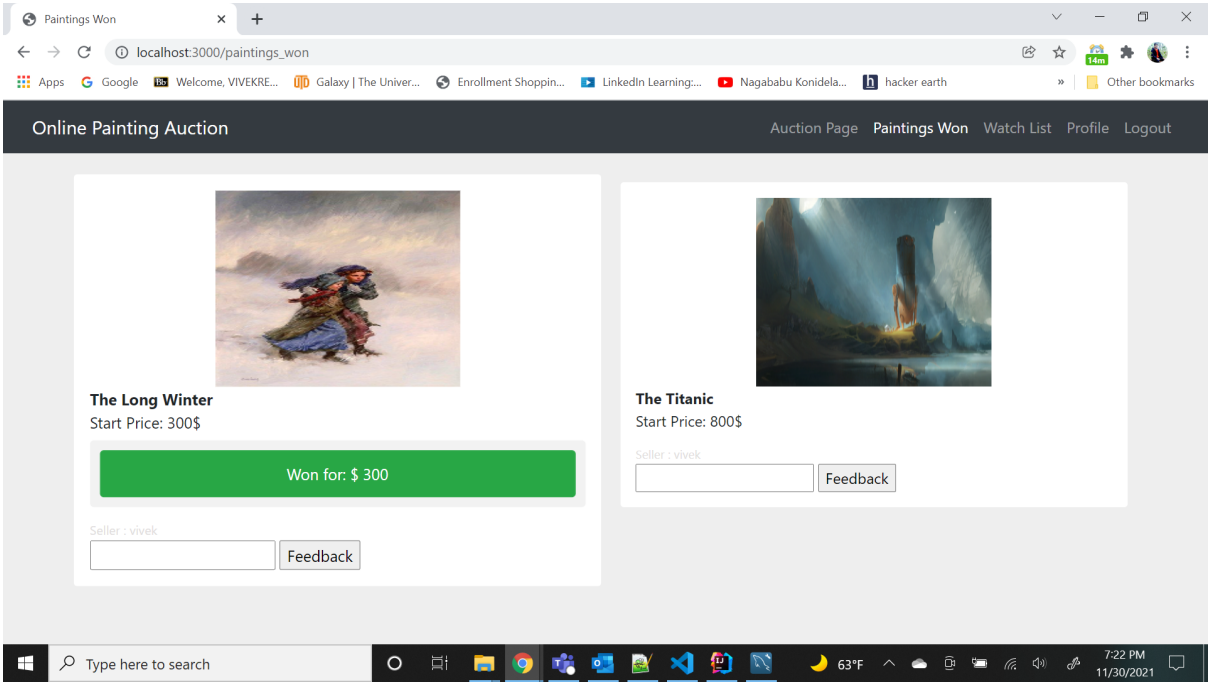
If the account type is a buyer account, the user will be redirected to the page shown below:



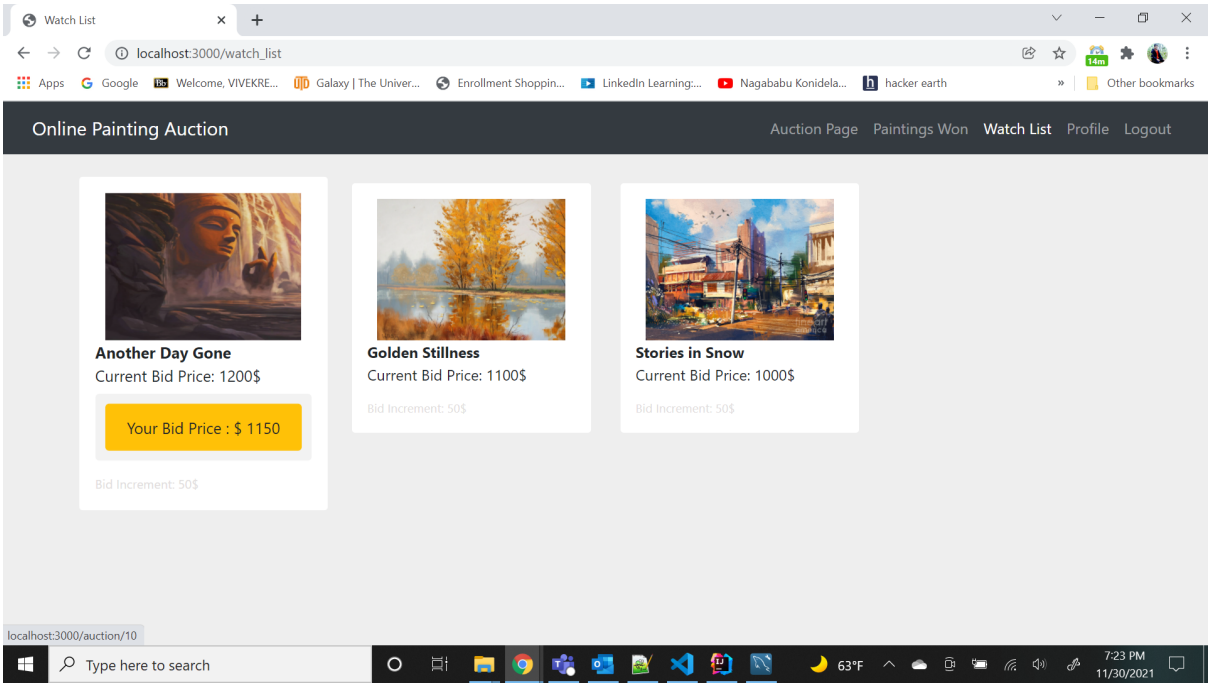
This page displays all the paintings that are currently being auctioned. It allows the buyer to bid on a painting of their choice. The buyer can bid on a painting by clicking the Bid button. On bidding,the bid value for the painting gets increased based on the increment value given by the seller. The buyer also has various other options on this page such as paintings won which displays all the paintings that the buyer has won, watch list which shows all the paintings that the buyer has bid on. The buyer can view his account details in profile and log out when his bidding is complete.



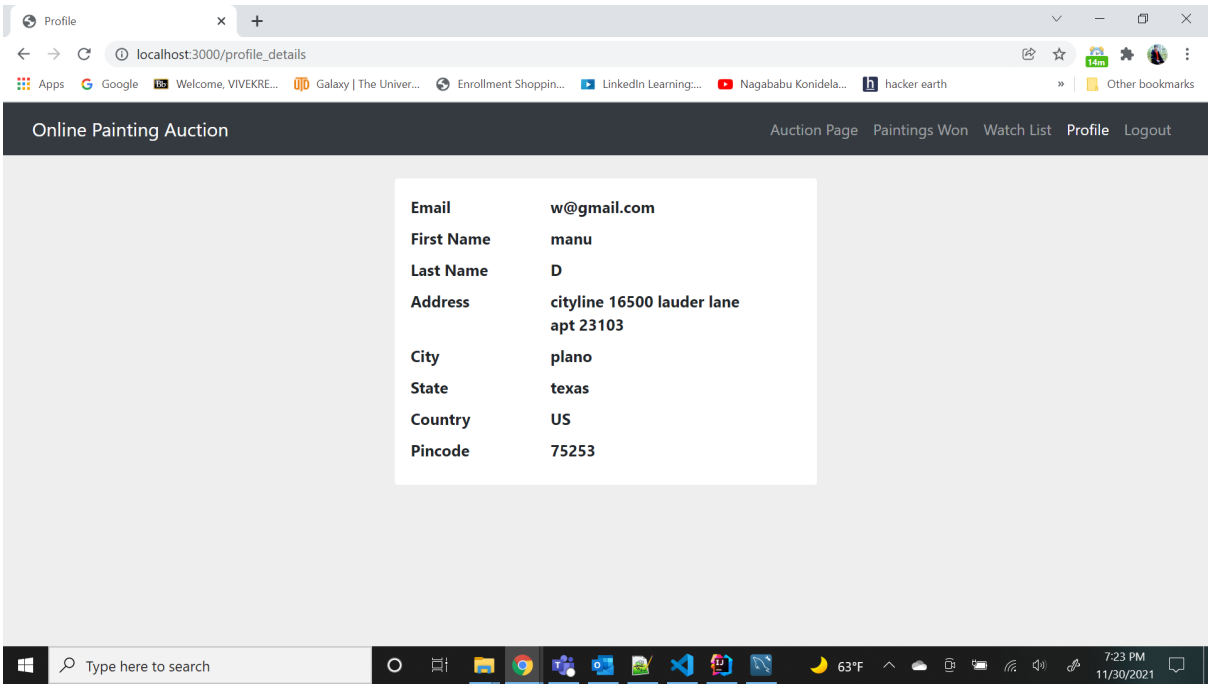
If the user wants to get more details about the paintings, they can just click on the painting which brings up its details. This page shows the information about the painting “Stories in Snow”. It displays the information like the start price,description, the start and the end date of the painting in the auction.



This page is used to display all the paintings that the buyer has won in the auction. It is used to display what is the start price and displays the winning amount. In this page, the buyer can provide feedback using the feedback input field.

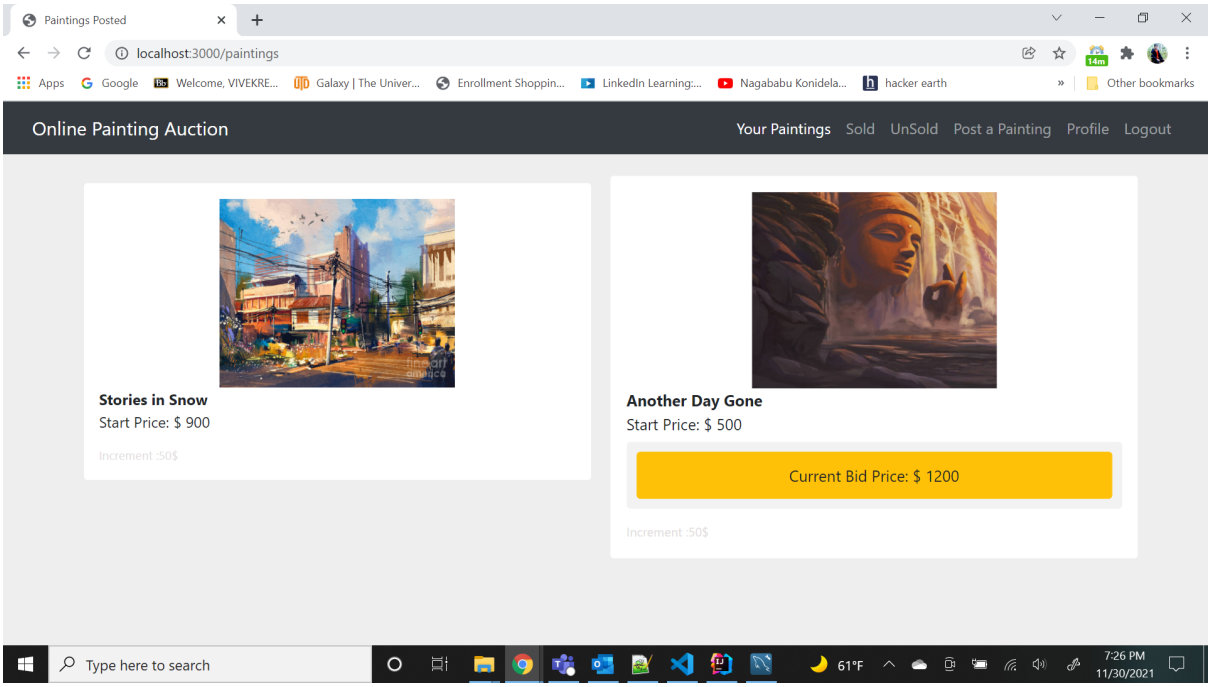


The above web page displays the paintings that the buyer participated in auctioning. This page also displays the current bid price for the particular painting and buyer's bid price on that particular painting

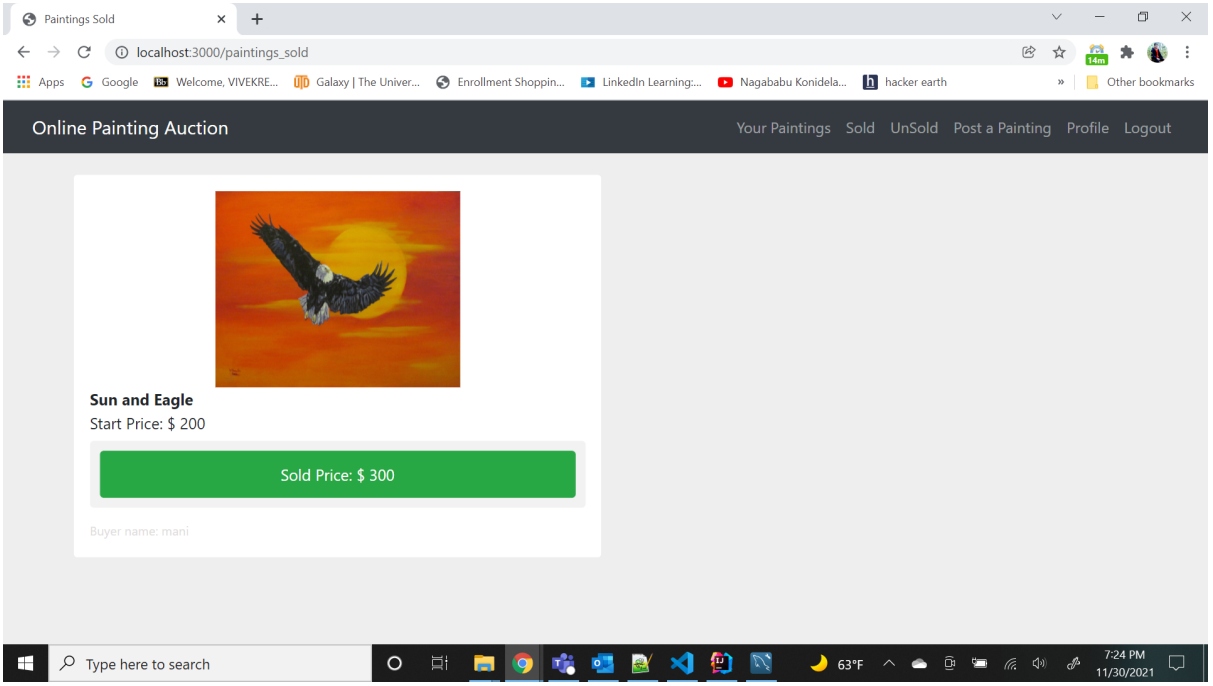


.

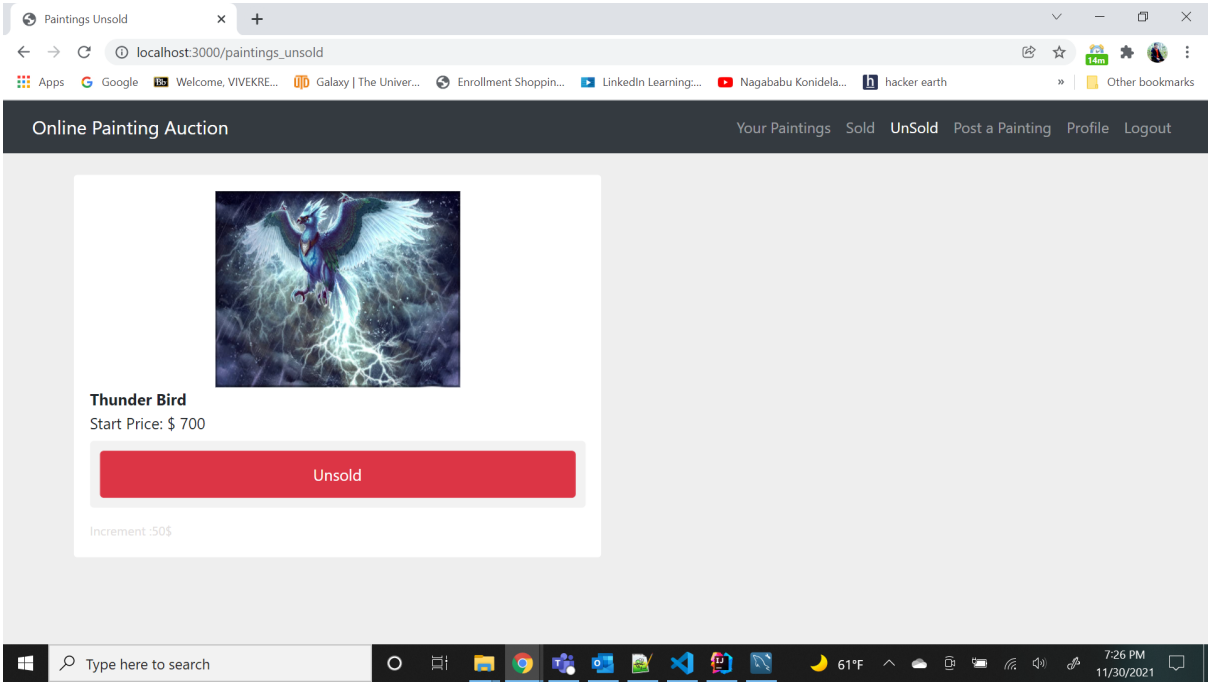
This web page is used to display the information about the buyer. It provides the email address, the first name, last name, address of the buyer. The information can be verified by the buyer.



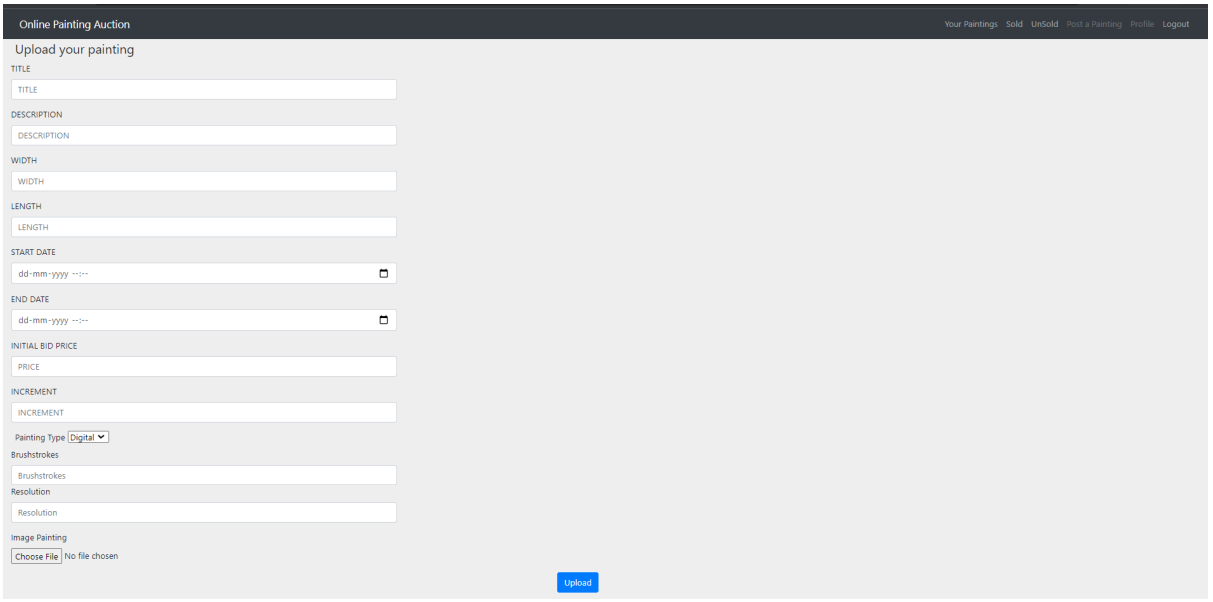
If the account is a seller type account, the options provided to the seller are different from those given to the buyer.
This web page is used to display the paintings that the seller is selling in the auction. It displays the start price, and provides the information about the price of the current bid for the particular painting.



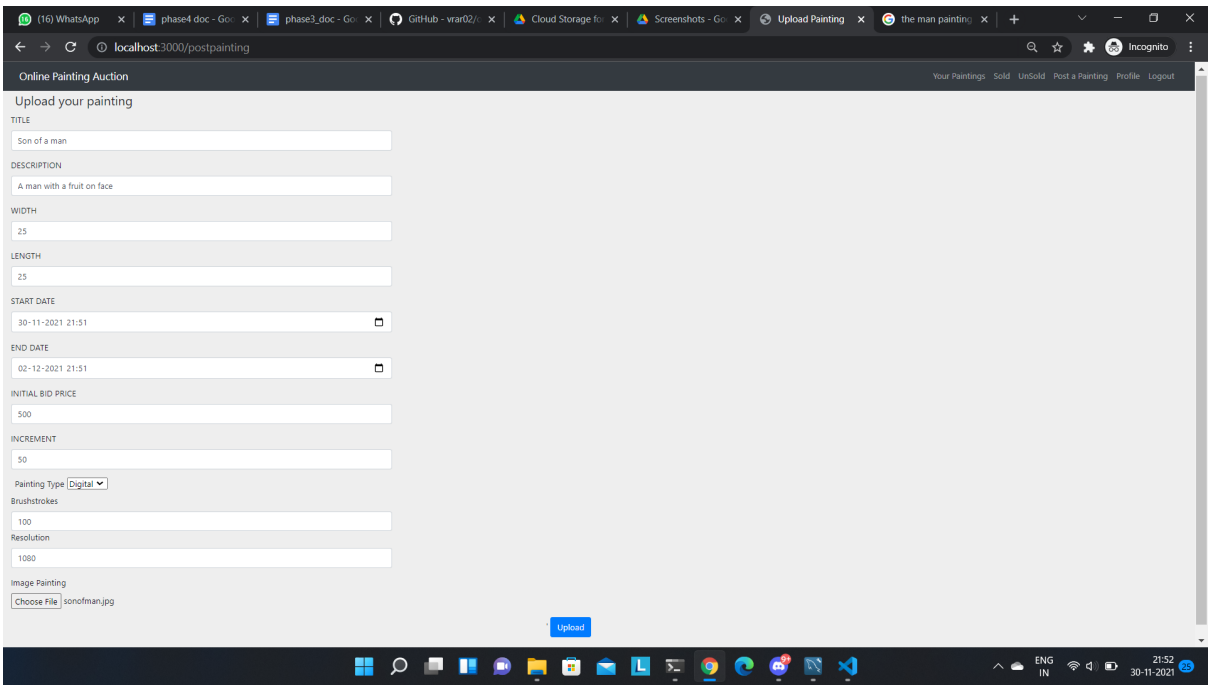
This web page is used to display the painting that has been successfully sold in the auction. It displays the price for which the painting was sold along with the buyer name who has bought the painting.



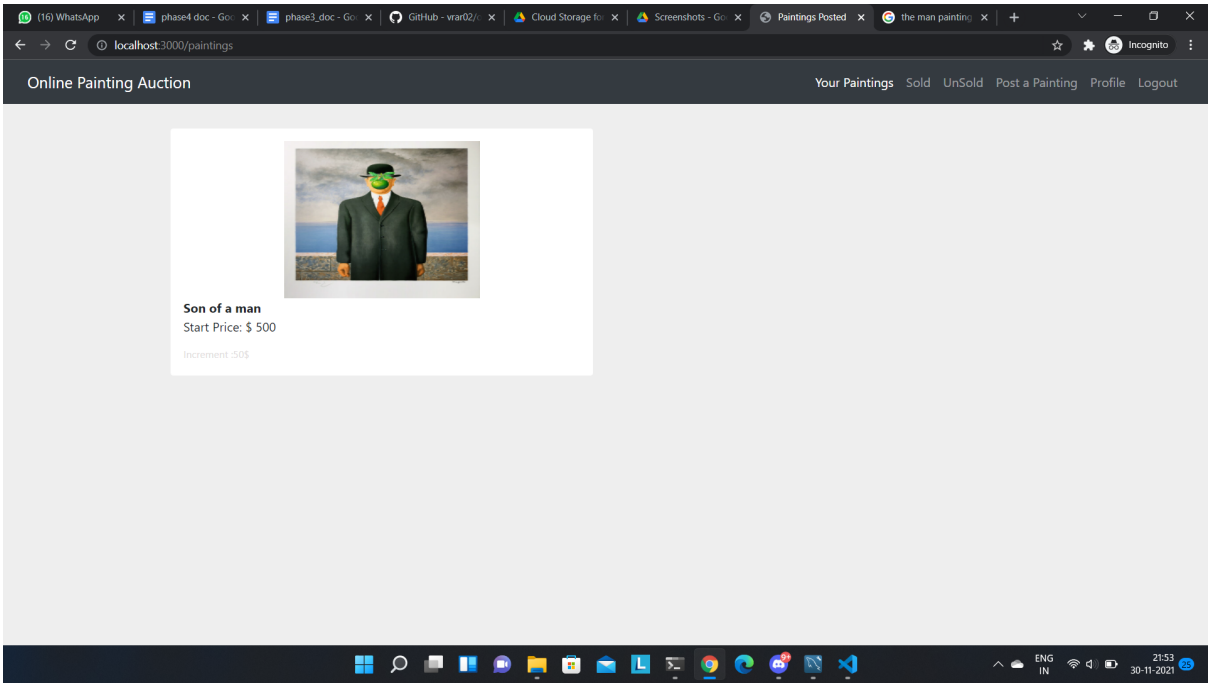
This web page is used to display the paintings that are unsold in the auction for the particular seller.



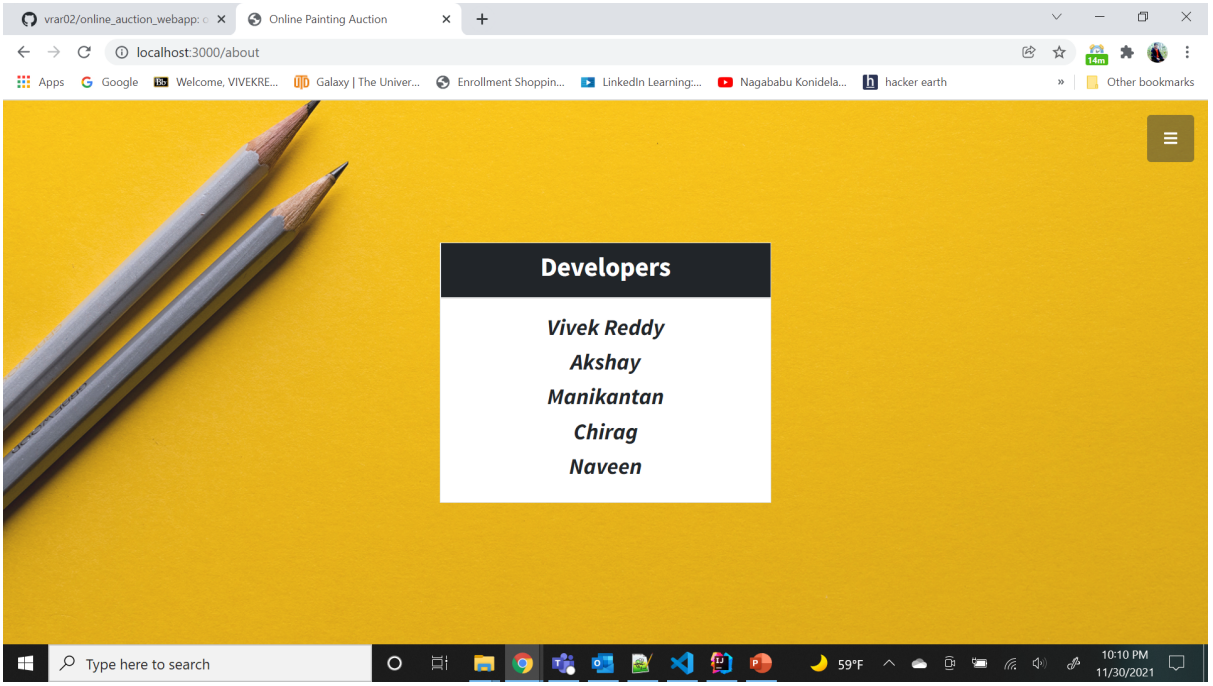
The above page helps the seller to add a new painting to the auction system. In this the seller can add the title, description, width, length, the start and end date of the auction, the initial bid price, the increment for the bid and the image of the painting.



The above screenshot is an example for a seller adding the information of painting in the web page.



The above screenshot is the result after adding the painting by the seller.



On selecting the about option in the homepage, it displays the information of the developers of the webpage:)

7. Database Tuning Suggestions

7.1. Indexing:

- create index email_index on member(email_id);
  Created the above index on the member table as email_id is unique and we fetch the member_id multiple times in our application to fetch the member_id by using the email_id.

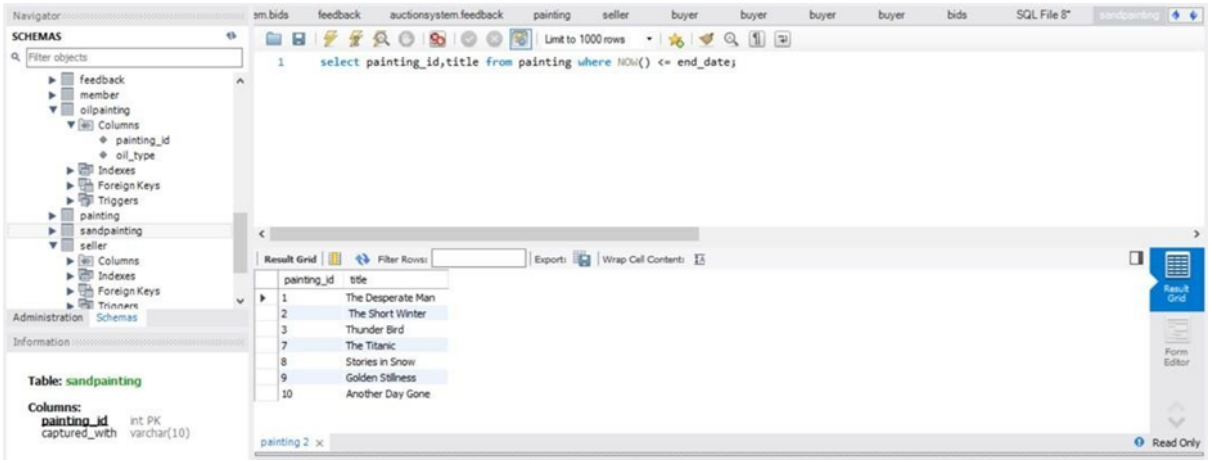
- create index bids_painting_index on painting(painting_id);
  Created above index because there are many views where the group is done based on painting_id.

8. Additional Queries and Views

8.1 Queries

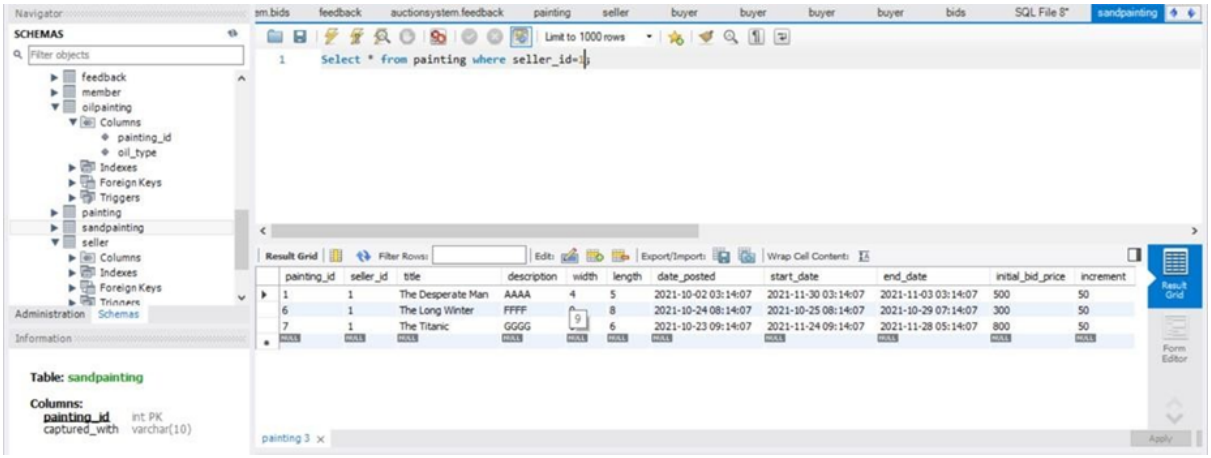
Query-1: Find all the current paintings which are in auction?

Ans. select painting_id,title from painting where NOW() <= end_date;



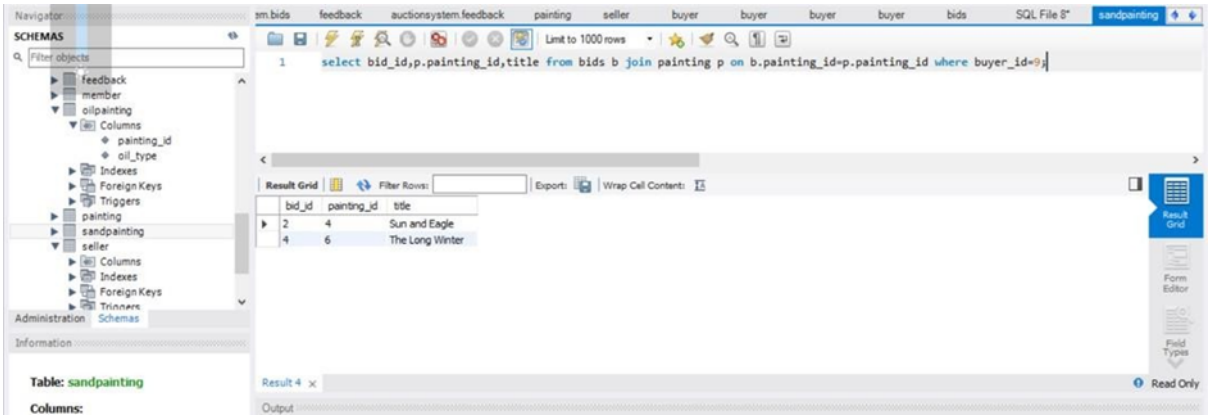
Query-2: Find all the paintings that were posted by a particular seller with seller_id=1?

Ans. Select * from painting where seller_id=1;



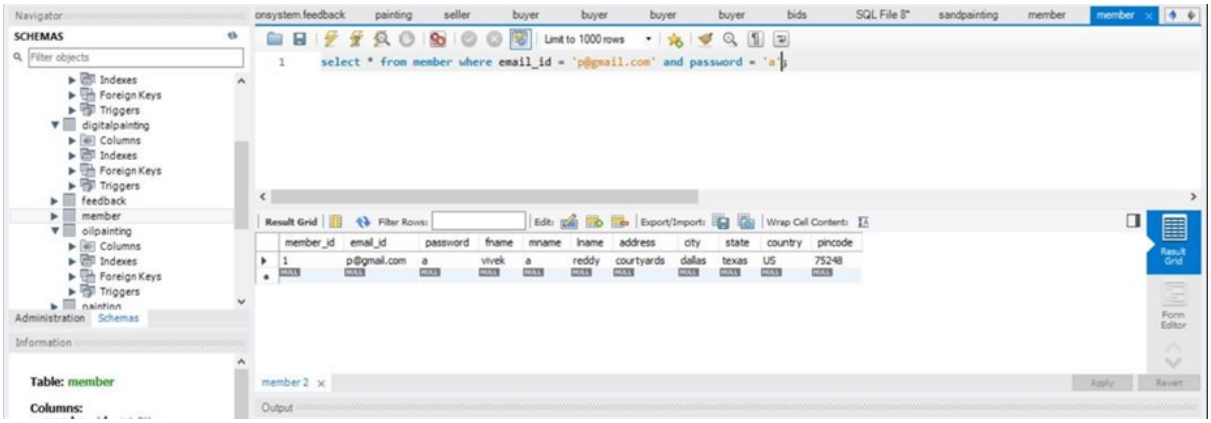
Query-3: Find all the bids made by a particular buyer with buyer_id=9?

Ans. select bid_id,p.painting_id,title from bids b join painting p on b.painting_id=p.painting_id where buyer_id=9;



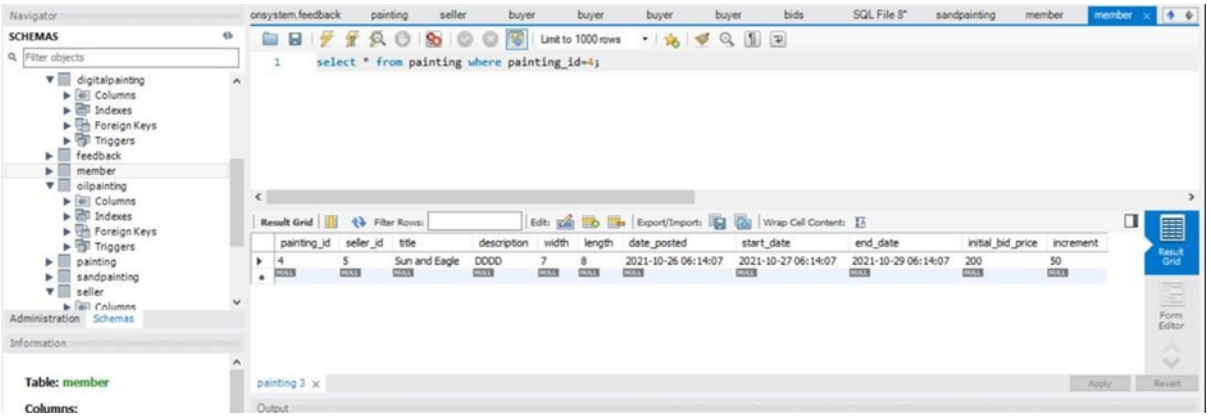
Query-4: Show the details of the member with following login credentials with email id='a@gmail.com' and password='a'?

Ans. select * from member where email_id = 'p@gmail.com' and password = 'a';



Query-5 : Display all the details of a particular painting?

Ans. select * from painting where painting_id=4;



8.2. Views

View -1: Create a view which displays the seller, the painting id and the price it sold for?

Create view sold_price_painting as
select bids.painting_id as painting, painting.title as title, painting.initial_bid_price as start_price, max(bids.bid_price) as sold_price, painting.seller_id as seller, member.fname from painting,bids,member where bids.painting_id=painting.painting_id and member.member_id =painting.seller_id and now()>painting.end_date group by painting;

```
1 •  SELECT * FROM auctionsystem.sold_price_painting;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

painting	title	start_price	sold_price	seller	fname
4	Sun and Eagle	200	300	5	naveen
6	The Long Winter	300	300	1	vivek
7	The Titanic	800	1450	1	vivek

View 2: Create a view which displays who won the auction for paintings?

Create view winner_painting as select * from sold_price_painting, bids where sold_price_painting.sold_price=bids.bid_price and sold_price_painting.painting=bids.painting_id;

```
1 •  SELECT * FROM auctionsystem.winner_painting;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

painting	title	start_price	sold_price	seller	fname	bid_id	buyer_id	painting_id	bid_time	bid_price
4	Sun and Eagle	200	300	5	naveen	3	2	4	2021-10-29 05:15:08	300
6	The Long Winter	300	300	1	vivek	4	9	6	2021-10-29 06:15:08	300
7	The Titanic	800	1450	1	vivek	54	9	7	2021-11-27 06:56:27	1450

View 3: Create a view which displays the paintings currently in auction?
Create view paintings_in_auction as Select * from painting where now()<=end_date and now()>=start_date;

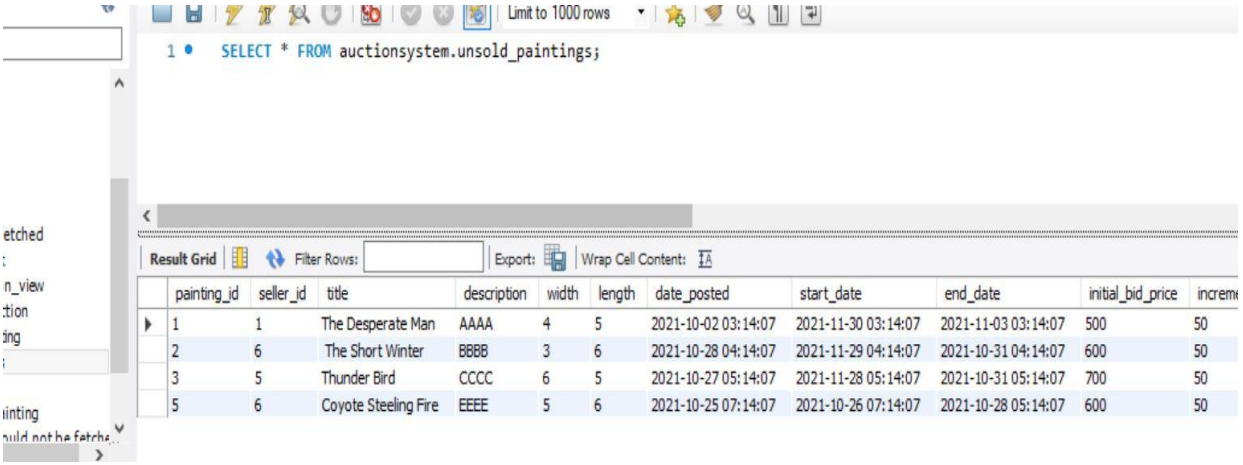
create view paintings_auction_view as select pa.painting_id as picture, pa.title as title ,pa.increment as increment,pa.initial_bid_price as start_price, COALESCE(max(bids.bid_price),0) as current_bid_price, pa.seller_id as seller from paintings_in_auction as pa left join bids on pa.painting_id=bids.painting_id group by picture ;

```
1 •  SELECT * FROM auctionsystem.paintings_auction_view;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

picture	title	increment	start_price	current_bid_price	seller
8	Stories in Snow	50	900	1100	5
9	Golden Stillness	50	1000	1150	6
10	Another Day Gone	50	500	1200	5

View4: Create a view to display all the unsold paintings?
CREATE VIEW `unsold_paintings` AS SELECT * FROM `painting` WHERE ((NOW() > `painting`.`end_date`) AND `painting`.`painting_id` IN (SELECT `bids`.`painting_id` FROM `bids`) IS FALSE)



```
1 •  SELECT * FROM auctionsystem.unsold_paintings;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

painting_id	seller_id	title	description	width	length	date_posted	start_date	end_date	initial_bid_price	increme
1	1	The Desperate Man	AAAA	4	5	2021-10-02 03:14:07	2021-11-30 03:14:07	2021-11-03 03:14:07	500	50
2	6	The Short Winter	BBBB	3	6	2021-10-28 04:14:07	2021-11-29 04:14:07	2021-10-31 04:14:07	600	50
3	5	Thunder Bird	CCCC	6	5	2021-10-27 05:14:07	2021-11-28 05:14:07	2021-10-31 05:14:07	700	50
5	6	Coyote Steeling Fire	EEEE	5	6	2021-10-25 07:14:07	2021-10-26 07:14:07	2021-10-28 05:14:07	600	50

9. User Application Interface

The application uses the mysql database for the backend, nodejs for implementation of the middleware where the data from db is processed and sent for displaying to the user based on his request. The middleware takes the http request from the user and performs some operation on the database and gives the appropriate response back to the user. We have used ExpressJS, Bootstrap, HTML and CSS for displaying the data and for interacting with users.

Some of the code snippets below will explain how the request and response flow happens from user interface to the database with the help of middleware.

Users will be able to register as a buyer or a seller. Once the user registers, the user can login with the valid credentials in /login webpage where the user details will be posted to the server and the request will be processed by verifying the credentials in verifyLoginDetails function shown below in screenshot.

```
app.use("/login", async (request, response) => {
    if (!request.session.isAuth) {
      const { email, password } = request.body;

      var verify_login_obj = new VerifyLogin();

      var login_details = await verify_login_obj.verifyLoginDetails(
        email,
        password
      );
```

```
  async verifyLoginDetails(email, password) {
    const mysql_obj = new MySQLBackend();
    try {
      const connect_obj = mysql_obj.connect();
      if (connect_obj != null) {
        console.log("connected to db");
        const sql = `select * from member where email_id=? and
password=?`;
        const query =
util.promisify(connect_obj.query).bind(connect_obj);
        var items = await query(sql, [email, password]);
        return items;
```

```
      }
    } catch (err) {
      throw new Error(err);
    } finally {
      console.log("Disconnected from db");
      mysql_obj.disconnect()}}
```

The screenshot shows the verifyLoginDetails function takes the credentials as input verifies with the details in the database and response is sent back accordingly.

To Display all the items currently in auction:

Whenever a buyer clicks on the auction page the above route receives the request and processes it to fetch the current auction items and render the page current_auction_items template where all the paintings are displayed.(shown in below screenshot)

```
router.get("/auction", isBuyer, async (request, response) => {
  const oa = new OnlineAction();
  const items = await oa.fetchCurrentAuctionItems();
  console.log("current_auction_items", items);
  response.render("current_auction_items", {
    pageTitle: "Present Auction Items",
    auction_items: items,
  });
});
```

Whenever a buyer bids on a particular painting the request places a new bid into the bids table and sends a broadcast to all the clients connected to a socket named “bid_done” such that the auction page is updated in a responsive way.

```
router.post("/biditem", isBuyer, async (request, response) => {
  var { bid_details } = request.body;
  console.log("bid_details", bid_details);
  const bid_obj = JSON.parse(bid_details);
  const oa = new OnlineAction();
  var buyer_email = request.session.userInfo;
  var buyer_ids = await oa.fetchBuyerId(buyer_email);
```

```
    var buyer_id = buyer_ids[0].member_id;
    var date_obj = new Date().toISOString().slice(0, 19).replace("T", " ");
    console.log(date_obj);
    var price = 0;
    if (bid_obj.current_bid_price == 0) {
      price = bid_obj.initial_bid_price + bid_obj.increment;
    } else {
      price = bid_obj.current_bid_price + bid_obj.increment;
    }
    console.log("pirce", price);
    await oa.insertIntoBids(buyer_id, bid_obj.painting_id, date_obj, price);
    var data = {
      painting_id: bid_obj.painting_id,
      current_bid_price: price,
      start_price: bid_obj.initial_bid_price,
      increment: bid_obj.increment,
    };
    request.io.on("connection", (socket) => {
      socket.broadcast.emit("bid_done", data);
    });
    response.redirect("/auction");
});
```

10. Conclusion and Future Work

Conclusion

With the popularity of artworks rising in current times, a lot of artists are getting the incentive to produce works and sell them. Similarly, a lot of people have taken an interest in art and search for places where they can buy them. With everything becoming digital and stores moving their businesses online, having a trustworthy site where artists and buyers can buy and sell artwork can ease the burden and process significantly. While undertaking this project we learnt a lot about not just databases but also designing UI's and other technologies. It was a very fun and enriching experience where we could put into practice everything we have learnt during the lectures and also more exposure in working with others and also as a team.

Future Work

This database system is very scalable and can be scaled up to increase the number of users that are actively bidding at any given time. At the moment it is acting only as an auction site for painting but in the future it can be modified to sell all kinds of art and also increase its scope from just art to include many other types of items getting auctioned, all while getting updated with new features using the latest advances made in database technology.

11. References

- https://nodejs.org/en/
- https://getbootstrap.com/
- https://www.w3schools.com/sql
- https://bootsnipp.com/
- https://ejs.co/
- https://dev.mysql.com/doc/

12. Appendix

Final Project Report.zip

Final Report Name: Project Report (Dec 3rd 2021)

/doc:
Phase 1 (Sept 17th 2021).pdf
Phase 2 (Oct 8th 2021).pdf
Phase 3 (Oct 29th 2021).pdf
Phase 4 (Nov 30th 2021).pdf
Project Report (Dec 3rd 2021).pdf

Source code :
/online_aution_webapp
- /bin - contains code for starting the node server
- /client - contains static files such as images, javascript and css
- /config - contains database config
- /routes - contains code for all middleware operations
- /services - contains code to fetch from database or push to database
- /sql - contains sql scripts to create the database schema
- /view - contains ejs files for frontend
- README file