**Project Report**

| Team Number: 20  Team Members:   * Bh Siva Ramya Sri * C Pradeep Reddy * D Srikarani   Branch: CSO |
| --- |

**Abstract:**

An online auction platform in a database management system (DBMS) is a web-based platform that allows users to bid on items and purchase them through an online auction process. The platform is designed to handle large volumes of data and traffic, allowing users to search for items, view product descriptions, place bids, and track their purchases. The DBMS manages the auction process, maintaining bids, transactions, user information records, and item reviews. It provides a secure and reliable platform for buyers and sellers to conduct business, with features such as payment processing, shipping options, and dispute resolution mechanisms. The platform also allows sellers to create and manage listings, set reserve prices, and monitor bidding activity. Overall, an online auction platform in a DBMS provides a convenient and efficient way for people to buy and sell goods in an online marketplace.

**Introduction:**

In the current digital age, online auctions have completely transformed the way we engage in buying and selling items. This innovative online auction project is designed to create a sophisticated and user-friendly platform that enables individuals to conveniently participate in auctions from their own homes using their electronic devices. The comprehensive project involves meticulous planning, user-centric design, rigorous implementation, and the deployment of a highly secure and interactive auction system, ensuring a seamless and engaging experience for all participants.

**Functional Requirements:**

Here are some common functional requirements for an online auction system project:

1. User Registration and Authentication

* Users can create accounts using email or social media.
* Users can log in using their credentials.

2. User Profiles

* Users can create and edit their profiles.
* Users can view their bidding history and won/lost auctions.

3. Auction Listings

* Sellers can create new auction listings with details like title, description, starting bid, and reserve price.
* Sellers can upload images for each auction item.
* Sellers can edit or delete their listings before any bids are placed.

4. Bidding Functionality

* Users can place bids on active auctions.
* Users receive real-time updates on the status of their bids (e.g., outbid notifications).

5. Automatic Bidding

* Users can set maximum bid amounts for automatic bidding (proxy bidding).
* The system automatically places incremental bids up to the user's maximum amount.

6. Search and Filter

* Users can search for auctions by keyword, category, seller, etc.
* Advanced filter options to narrow search results (e.g., by price range, auction end time).

7. Notifications and Alerts

* Push notifications for auction events (e.g., when outbid, auction ending soon, auction won).

8. Auction Management

* Sellers can view and manage their active and completed auctions.
* Sellers can contact winning bidders for payment and delivery arrangements.

9. Payment Integration

* Secure payment processing for auction winners.
* Integration with payment gateways (e.g., PayPal, credit card processors).
* Payment tracking and confirmation.

10. Feedback and Rating System

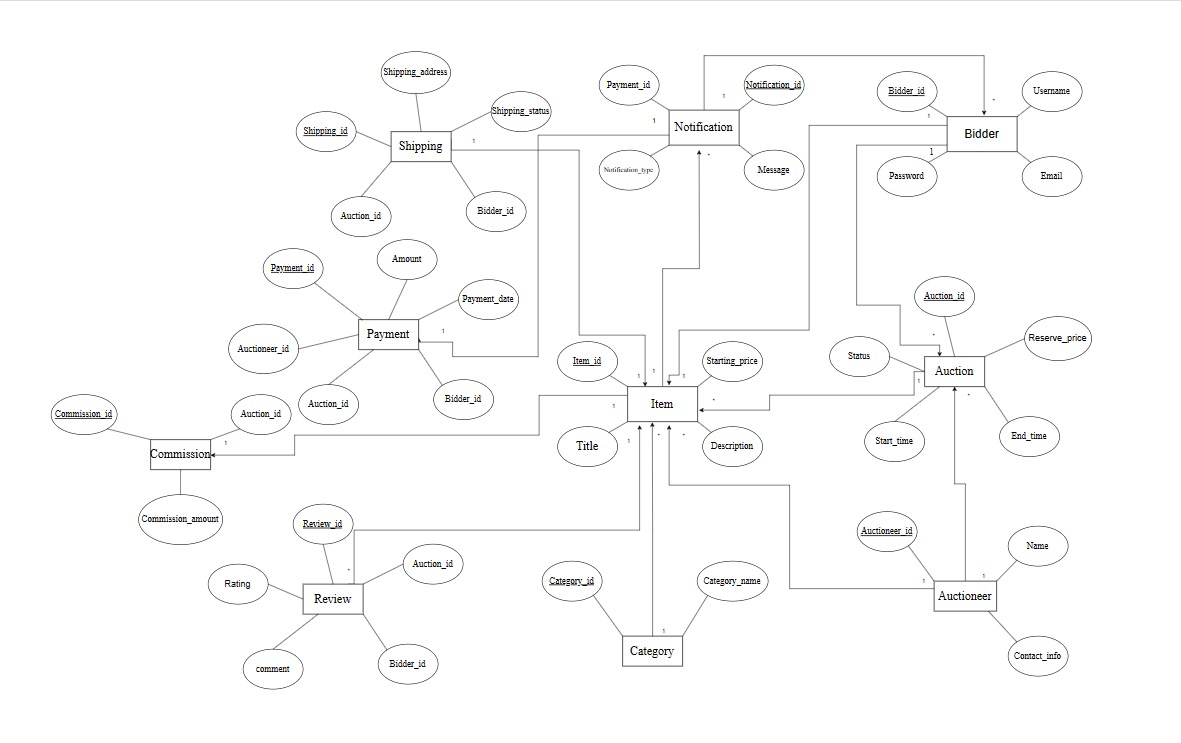
* Buyers and sellers can rate and leave feedback for each other after transactions.
* Aggregate ratings and feedback history visible on user profiles.

11. Reporting and Analytics

* Users can generate reports on their bidding and selling activities.
* Administrators have access to comprehensive analytics on system usage and performance.

12. Security and Safety

* User data protection and encryption.
* Compliance with relevant legal and regulatory requirements.
* Fraud detection and prevention mechanisms.

**ER Diagram :**

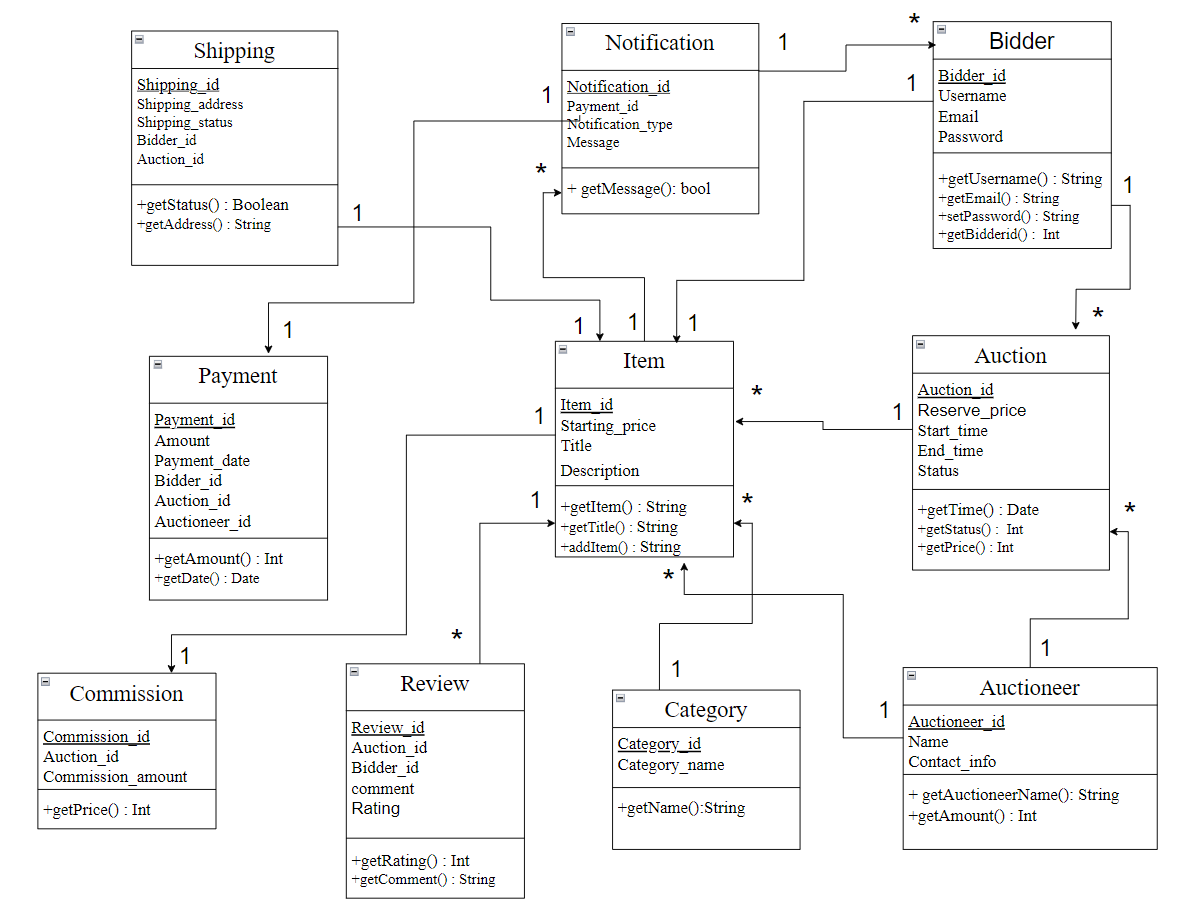
**Queries used to create the database:**

Create: The Create Table statement is used to create a new table in a database.

Syntax:CREATE TABLE table\_name (column1 datatype,column2 datatype,column3 datatype ....);

* create table auction(auction\_id int primary key,start\_time time not null,end\_time time not null,status tinyint,reserve\_price int not null);
* create table auctioneer(auctioneer\_id int primary key,name varchar(255),contact\_info int unique);
* create table item(item\_id int primary key,starting\_price int not null,description varchar(255),Title varchar(255) not null);
* create table bidder(bidder\_id int primary key,username varchar(255),email varchar(255) not null,password varchar(255) not null);
* create table payment(payment\_id int primary key,amount int not null,payment\_date datetime not null,bidder\_id int REFERENCES bidder(bidder\_id),auction\_id int REFERENCES auction(auction\_id),auctioneer\_id int REFERENCES auctioneer(auctioneer\_id));
* create table shipping(shipping\_id int primary key,auction\_id int REFERENCES auction(auction\_id),bidder\_id int REFERENCES bidder(bidder\_id),shipping\_status tinyint,shipping\_address varchar(255)not null);
* create table category(category\_id int primary key,category\_name varchar(255));
* create table review(review\_id int primary key,auction\_id int references auction(auction\_id),bidder\_id int references bidder(bidder\_id),comment varchar(255),rating int not null);
* create table commission(commission\_id int primary key,auction\_id int references auction(auction\_id),commission\_amount int not null);
* create table notification(notification\_id int primary key,message varchar(255),notification\_type varchar(255) not null,payment\_id int references payment(payment\_id));

**Class / UML attribute & methods diag:**



**All class java file codes:**

**Bidder class:-**

public class Bidder extends User {

public Bidder(String username, String password, String email) {

super(username, password, email);

}

}

**Auctioneer class:-**

public class Auctioneer extends User {

public Auctioneer(String username, String password, String email) {

super(username, password, email);

}

public void createAuction(Auction auction) {

}

}

**Category class:-**

public class Category {

private String name;

public Category(String name) {

this.name = name;

}

public String getName() {

return name;

}

}

**Item class:-**

public class Item {

private String itemName;

private String description;

private double startingBid;

private Category category;

public Item(String itemName, String description, double startingBid, Category category) {

this.itemName = itemName;

this.description = description;

this.startingBid = startingBid;

this.category = category;

}

public String getItemName() {

return itemName;

}

public double getStartingBid() {

return startingBid;

}

public Category getCategory() {

return category;

}

}

**Auction class:-**

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class Auction {

private Item item;

private Date startTime;

private Date endTime;

private Auctioneer auctioneer;

private List<Bid> bids = new ArrayList<>();

private double highestBid;

private Bidder highestBidder;

public Auction(Item item, Date startTime, Date endTime, Auctioneer auctioneer) {

this.item = item;

this.startTime = startTime;

this.endTime = endTime;

this.auctioneer = auctioneer;

this.highestBid = item.getStartingBid();

}

public boolean placeBid(Bid bid) {

if (new Date().before(endTime) && bid.getAmount() > highestBid) {

highestBid = bid.getAmount();

highestBidder = bid.getBidder();

bids.add(bid);

return true;

}

return false;

}

public double getHighestBid() {

return highestBid;

}

public Bidder getHighestBidder() {

return highestBidder;

}

}

**Payment class:-**

public class Payment {

private Auction auction;

private double amount;

private String paymentMethod;

public Payment(Auction auction, double amount, String paymentMethod) {

this.auction = auction;

this.amount = amount;

this.paymentMethod = paymentMethod;

}

public Auction getAuction() {

return auction;

}

public double getAmount() {

return amount;

}

public String getPaymentMethod() {

return paymentMethod;

}

}

**Shipping class:-**

public class Shipping {

private Item item;

private String address;

private String trackingNumber;

public Shipping(Item item, String address) {

this.item = item;

this.address = address;

}

public void setTrackingNumber(String trackingNumber) {

this.trackingNumber = trackingNumber;

}

public String getTrackingNumber() {

return trackingNumber;

}

}

**Review class:-**

public class Review {

private User reviewer;

private User reviewee;

private String comments;

private int rating; // Rating out of 5

public Review(User reviewer, User reviewee, String comments, int rating) {

this.reviewer = reviewer;

this.reviewee = reviewee;

this.comments = comments;

this.rating = rating;

}

public String getComments() {

return comments;

}

public int getRating() {

return rating;

}

}

**Notification class:-**

public class Notification

{

private User user;

private String message;

public Notification(User user, String message)

{

this.user = user;

this.message = message;

}

public String getMessage()

{

return message;

}

}

**Commission class:-**

public class Commission

{

private Auction auction;

private double commissionRate;

public Commission(Auction auction, double commissionRate)

{

this.auction = auction;

this.commissionRate = commissionRate;

}

public double calculateCommission()

{

return auction.getHighestBid() \* commissionRate;

}

}

**Main class:-**

import java.util.Date;

import java.util.Scanner;

public class OnlineAuctionSystem {

public static void main(String[] args) {

// Sample data and interaction logic

Auctioneer auctioneer = new Auctioneer("auctioneer1", "pass", "auctioneer@example.com");

Bidder bidder1 = new Bidder("bidder1", "pass", "bidder1@example.com");

Bidder bidder2 = new Bidder("bidder2", "pass", "bidder2@example.com");

Category electronics = new Category("Electronics");

Item laptop = new Item("Laptop", "High-performance laptop", 500, electronics);

Auction auction = new Auction(laptop, new Date(), new Date(System.currentTimeMillis() + 3600 \* 1000), auctioneer);

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("Enter your username:");

String username = scanner.nextLine();

System.out.println("Enter your password:");

String password = scanner.nextLine();

User user;

if (username.equals(auctioneer.getUsername()) && auctioneer.authenticate(password)) {

user = auctioneer;

} else if (username.equals(bidder1.getUsername()) && bidder1.authenticate(password)) {

user = bidder1;

} else if (username.equals(bidder2.getUsername()) && bidder2.authenticate(password)) {

user = bidder2;

} else {

System.out.println("Invalid credentials. Try again.");

continue;

}

if (user instanceof Bidder) {

System.out.println("Enter your bid amount:");

double bidAmount = scanner.nextDouble();

scanner.nextLine(); // Consume newline

Bid bid = new Bid((Bidder) user, bidAmount);

if (auction.placeBid(bid)) {

System.out.println("Bid placed successfully!");

} else {

System.out.println("Bid too low or auction ended. Current highest bid is: " + auction.getHighestBid());

}

}

System.out.println("Do you want to continue? (yes/no)");

String response = scanner.nextLine();

if (!response.equalsIgnoreCase("yes")) {

break;

}

}

scanner.close();

System.out.println("Thank you for using the Online Auction System!");

}

}

**Challenges list:**

Creating an online auction system comes with its own set several technical and operational challenges of challenges.Some of them are like:

Technical Challenges

1. Scalability

- Challenge: The system must handle a large number of simultaneous users, especially during popular auctions.

- Solution: Implement load balancing, use scalable cloud services, and optimise the database for high read/write operations.

2. Real-Time Bidding

- Challenge: Ensuring real-time updates for bids and auction statuses across all users.

- Solution: Use WebSockets or other real-time communication protocols to push updates to clients instantly.

3. Data Consistency

- Challenge: Maintaining consistent data across the system, especially for bids placed at nearly the same time.

- Solution: Implement distributed transaction management and ensure ACID (Atomicity, Consistency, Isolation, Durability) properties in the database.

4. Security

- Challenge: Protecting user data, preventing fraudulent activities, and ensuring secure transactions.

- Solution: Use encryption for data storage and transmission, implement multi-factor authentication, and regularly conduct security audits.

5. User Authentication and Authorization

- Challenge: Managing user accounts securely and ensuring only authorized users can perform certain actions.

- Solution: Implement robust authentication mechanisms, role-based access control, and session management.

6. Payment Integration

- Challenge: Integrating with payment gateways to handle transactions securely and efficiently.

- Solution: Use well-established payment gateways with comprehensive APIs, and ensure PCI DSS compliance.

7. Bid Sniping Prevention

- Challenge: Preventing last-second bids that may frustrate users.

- Solution: Implement anti-sniping features such as extending the auction end time by a few minutes if a bid is placed in the last moments.

Operational Challenges

1. Customer Support

- Challenge: Providing timely support to users experiencing issues.

- Solution: Set up a dedicated support team and use helpdesk software to manage and track user queries.

2. User Experience

- Challenge: Ensuring the platform is user-friendly and accessible.

- Solution: Conduct user testing, gather feedback, and continuously improve the UI/UX design.

3. Fraud Detection

- Challenge: Detecting and preventing fraudulent activities such as shill bidding or fake listings.

- Solution: Implement machine learning algorithms to detect suspicious patterns, and set up manual review processes for high-risk activities.

4. Compliance and Legal Issues

- Challenge: Adhering to laws and regulations across different jurisdictions.

- Solution: Consult legal experts to ensure compliance with relevant laws, and implement policies for dispute resolution and user protection.

5. Shipping and Logistics

- Challenge: Coordinating the shipping of items post-auction.

- Solution: Partner with reliable shipping services, and integrate tracking systems to keep users informed.

6. Performance Monitoring and Maintenance

- Challenge: Ensuring the system remains operational and performance at all times.

- Solution: Use monitoring tools to track system health, set up alerts for critical issues, and have a robust maintenance plan.

7. Marketing and User Acquisition

- Challenge: Attracting and retaining users in a competitive market.

- Solution: Develop and execute a comprehensive marketing strategy, including SEO, social media marketing, and promotional campaigns.

Addressing these challenges requires a combination of technical expertise, strategic planning, thoughtful design, robust security measures, and continuous monitoring. By anticipating these issues and preparing solutions, you can build a robust and reliable online auction system.