**A**

**PROJECT REPORT**

**ON**

**PrimeBid**

**(Bidding System)**

**AT**

**NAME OF THE COMPANY**

**PUNE**

**BY**

**Umesh Rajesh Mali**

**Pranav Sanjiv Rajurkar**

**SAVITRIBAI PHULE PUNE UNIVERSITY**

**In Partial Fulfillment of**

**MASTERS IN COMPUTER APPLICATION**

**SINHGAD INSTITUTE OF BUSINESS ADMINISTRATION AND RESEARCH**

**KONDHWA, PUNE-411048**

**2023-2025**

**CERTIFICATE OF APPROVAL**

This is to certify that the project report entitled…………………………………………………….………………………

submitted to the Department of Computer Application, Sinhgad Institute of Business Administration and Research in partial fulfillment of the requirement for the award of the Degree of MASTER OF COMPUTER APPLICATIONS (MCA Affiliated to Savitribai Phule Pune University) is an original work carried out by

Mr./Ms Exam No

Mr./Ms Exam No

The matter embodied in this project is a genuine work done by the student and has been certified by the following internal and external examiners deputed by Savitribai Phule Pune University.

Internal Examiner External Examiner

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Chapter No** |  | **Details** |
| **1** |  | **Introduction** |
|  | 1.1 | Company Profile / Institute Profile / Client Profile |
|  | 1.2 | Abstract |
|  | 1.3 | Existing System and Need for System |
|  | 1.4 | Scope of System |
|  | 1.5 | Operating Environment - Hardware and Software |
|  | 1.6 | Brief Description of Technology Used   * + 1. Operating systems used (Windows or Unix)     2. RDBMS/No Sql used to build database (mysql/ oracle, Teradata, etc.) |
| **2** |  | **Proposed System** |
|  | 2.1 | Study of Similar Systems ( If required research paper can be included) |
|  | 2.2 | Feasibility Study |
|  | 2.3 | Objectives of Proposed System |
|  | 2.4 | Users of System |
| **3** |  | **Analysis and Design** |
|  | 3.1 | System Requirements (Functional and Non-Functional requirements) |
|  | 3.2 | Entity Relationship Diagram (ERD) |
|  | 3.3 | Table Structure |
|  | 3.4 | Use Case Diagrams |
|  | 3.5 | Class Diagram |
|  | 3.6 | Activity Diagram |
|  | 3.7 | Deployment Diagram |
|  | 3.8 | Module Hierarchy Diagram |
|  | 3.9 | Sample Input and Output Screens (Screens must have valid data. All  reports must have at-least 5 valid records.) |
| **4** |  | **Coding** |
|  | 4.1 | Algorithms |
|  | 4.2 | Code snippets |
| **5** |  | **Testing** |
|  | 5.1 | Test Strategy |
|  | 5.2 | Unit Test Plan |
|  | 5.3 | Acceptance Test Plan |
|  | 5.4 | Test Case / Test Script |
|  | 5.5 | Defect report / Test Log |
| 6 |  | Limitations of Proposed System |
| 7 |  | Proposed Enhancements |
| 8 |  | Conclusion |
| 9 |  | Bibliography |
| 10 |  | Publication / Competition certificates |
| 11 |  | Appendix – Cost sheet , Data sheet |
| 12 |  | User Manual (All screens with proper description/purpose Details  about validations related to data to be entered.) |

**1.2 Abstract**

PrimeBid – Auction Platform is a web-based system that enables users to participate in auctions either as sellers or buyers. The platform is built using a modern technology stack — Node.js for the backend, React.js for the frontend, and MongoDB for the database. The aim is to offer a transparent, secure, and user-friendly auction experience.

Users can register, list items for auction, and place bids on available products. Auctions are time-bound, and once the bidding timer ends, the highest bidder wins. The system automatically processes this outcome and updates the item status. Admins have full control over the platform, including user and auction management. The project emphasizes real-time interaction, role-based access, and intuitive design.

**1.3 Existing System and Need for System**

Traditional auction systems, whether physical or online, often involve complex registration procedures, third-party involvement, or high service fees. Platforms like eBay are widely used but may not be beginner-friendly or suitable for localized or small-scale sellers.

The need for a dedicated, lightweight, and straightforward auction platform is significant. Many users prefer minimal interfaces with essential features and security. PrimeBid fills this gap by offering:

* Direct seller-to-buyer auctioning.
* Real-time bidding with countdown timers.
* Simplicity and security without unnecessary complexity.

This system is tailored for users who want to list or buy items in a secure, time-bound, and transparent digital environment.

**1.4 Scope of the System**

The system includes the following functional coverage:

* User Authentication: Secure login and signup using token-based authentication.
* Role Management: Admin, Seller, and Buyer roles with different permissions.
* Auction Management: Sellers can add auctions with descriptions, images, and start/end times.
* Bidding System: Buyers can view available auctions and place bids in real time.
* Real-Time Updates: Auction countdown and bid tracking are updated live.
* Winner Declaration: The system automatically finalizes the highest bidder once the timer ends.
* Admin Dashboard: Admins can monitor auctions, users, and manage reported issues.

**1.5 Operating Environment – Hardware and Software**

Hardware Requirements:

* Processor: Intel i5 or higher
* RAM: Minimum 8 GB
* Hard Disk: At least 100 GB free space
* Network: Reliable internet connection for real-time operations

Software Requirements:

* Frontend Development: ReactJS
* Backend Development: Node.js + Express.js
* Database: MongoDB (NoSQL)
* Code Editor: Visual Studio Code
* Version Control: Git & GitHub
* API Testing Tool: Postman
* Browser: Google Chrome (for testing and access)

**1.6 Brief Description of Technology Used**

**1.6.1 Operating Systems Used (Windows or Unix)**

* The system was developed and tested on both Windows 10 and macOS.
* It is platform-independent, as it is browser-based.
* The development environment used Node.js, which runs smoothly on any OS with proper configuration.

**1.6.2 RDBMS / NoSQL Used to Build Database (MySQL / Oracle / Teradata, etc.)**

* MongoDB was used as the NoSQL database for this project.
* MongoDB allows flexible schema design, making it ideal for storing data like auctions, bids, users, and payment proofs.
* The database is hosted on MongoDB Atlas for easy cloud access and scalability.

**2. Proposed System**

**2.1 Study of Similar Systems**

Several online auction platforms already exist, such as eBay, AuctionZip, HiBid, Proxibid, and Bidsquare, each offering users a way to participate in live or scheduled auctions. However, these systems come with notable limitations:

* eBay: Overwhelming for new users due to complexity, high seller fees, and ongoing issues with fraud and customer service.
* AuctionZip: Primarily focused on live, in-person auctions, limiting remote accessibility.
* HiBid: Offers online features but is often criticized for poor user experience and limited listings.
* Proxibid: Complicated registration, high transaction fees, and limited global accessibility.
* Bidsquare: Niche-focused (mostly art and antiques), smaller user base, and scalability concerns.

These platforms generally lack a simplified interface for beginners, real-time system feedback, and a flexible structure for smaller communities.

PrimeBid is designed to overcome these problems with:

* A simple, modern UI using ReactJS.
* Role-based access for Admins, Sellers, and Bidders.
* Real-time bidding logic handled by Node.js and Express.
* A document-oriented database (MongoDB) for flexibility and scalability.
* Transparent auction flow with minimal user friction.

**2.2 Feasibility Study**

**Technical Feasibility**

PrimeBid uses widely supported technologies:

* Frontend: ReactJS
* Backend: Node.js with Express.js
* Database: MongoDB (NoSQL)

These are all open-source, lightweight, and suitable for scalable applications.

**Operational Feasibility**

* The system is designed with clear user roles and intuitive workflows:
* Sellers can easily list products with images and auction rules.
* Buyers can register, bid in real-time, and receive notifications.
* Admins have full control through a separate dashboard.
* The system is platform-independent and works smoothly on any modern web browser.

**Economic Feasibility**

* All tech used is open-source, removing licensing costs.
* Hosting costs are minimal with deployment on local or cloud-based servers.
* The long-term cost of maintaining the system is low, making it viable even for individual use or small startups.

**2.3 Objectives of Proposed System**

The main objectives of PrimeBid are:

* To create a secure, responsive online platform for time-bound auctions.
* To support multiple user roles: Super Admin, Seller, and Bidder.
* To enable sellers to create and manage auction listings.
* To allow bidders to place bids with live feedback.
* To implement real-time auction updates and final winner announcements.
* To simplify the payment proof upload and verification process.
* To provide a full-featured Admin Dashboard for platform monitoring and management.

**2.4 Users of the System**

* Seller: Can create and manage auction listings with item descriptions, images, starting bids, and deadlines.
* Bidder: Can browse ongoing auctions and place bids in real-time. Receives notifications about being outbid or winning.
* Admin: Manages platform users, auctions, transactions, and can resolve disputes or monitor site activity.
* Super Admin (optional role depending on implementation): Higher-level control over Admin accounts and platform configurations.

**3. Analysis and Design**

**3.1 System Requirements**

Requirement analysis is the foundation for designing any system. For PrimeBid, both functional and non-functional requirements are identified based on user roles and auction workflows.

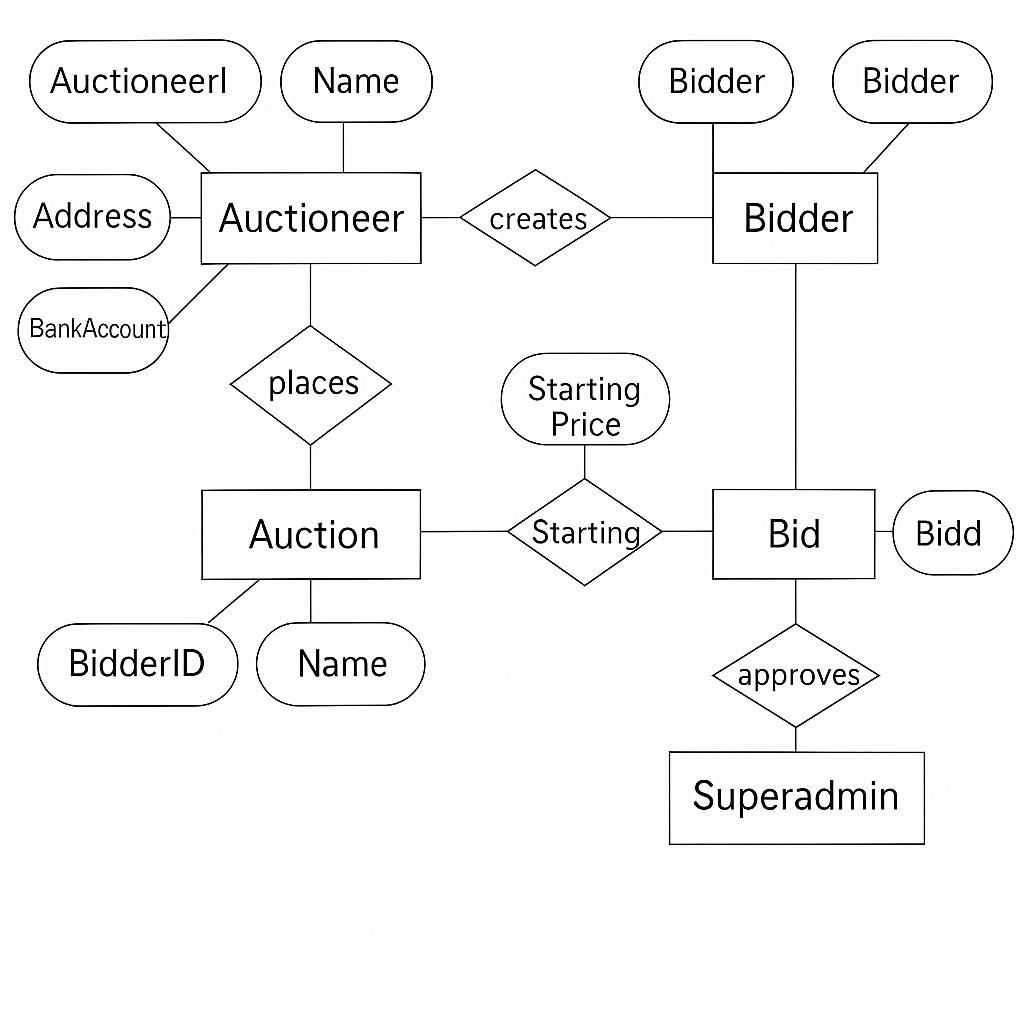
**Functional Requirements**

* User Registration and Login (Admin, Seller, Bidder)
* Auction listing creation by Seller
* Real-time bidding system
* Countdown timer for auction expiry
* Auto-selection of the highest bidder
* Upload proof of payment by winner
* Admin dashboard for managing auctions and users
* Role-based access control

**Non-Functional Requirements**

* Responsive and user-friendly UI
* Real-time performance
* Data consistency and integrity
* Security and authentication (JWT)
* Scalability with MongoDB

**3.2 ERD**



**3.3 Table Structure:**

1. **User**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Description** |
| UserName | String | User Name |
| Password | String | Password |
| Email | String | Email |
| Address | String | Address |
| Phone | String | Phone |
| ProfileImage\_public\_id | Int | Profile Image public id |
| profileImage\_url | String | Profile Image url |
| paymentMethods\_bankAccountNumber | String | Payment Methods bank Account Number |
| paymentMethods\_bankAccountName | String | Payment Methods bank Account Name |
| paymentMethods\_bankName | String | Payment Methods bank Name |
| paymentMethods\_upi | String | Payment Methods UPI id |
| Role | String | role |
| unpaidCommission | Number | Unpaid Commission |
| auctionsWon | Number | Auctions Won |
| moneySpent | Number | Money Spent |
| createdAt | Date | Created At |

1. **Auction Table**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Description** |
| Title | String | title |
| Description | String | description |
| startingBid | Number | Starting Bid |
| Category | String | category |
| currentBid | Number | Current Bid |
| startTime | String | Start Time |
| endTime | String | End Time |
| image\_public\_id | String | Image public id |
| image\_url | String | Image url |
| createdBy |  | Created By |
| highestBidder |  | Highest Bidder |
| commissionCalculated | Boolean | Commission Calculated |
| createdAt | Date | Created At |

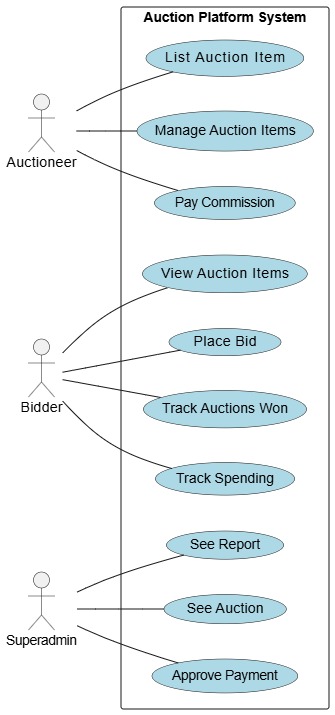
1. **Commision Table**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Description** |
| Amount | Number | amount |
| User | string | user |
| createdAt | Date | createdAt |

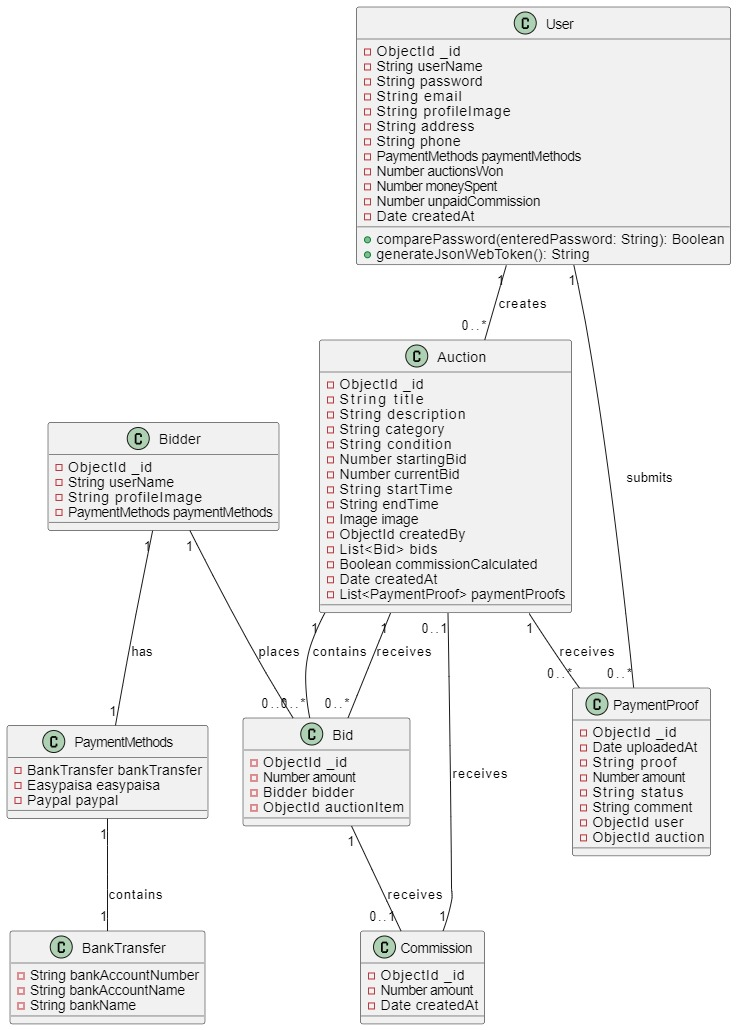
1. **Payment Proof**

|  |  |  |
| --- | --- | --- |
| **Field** | **Data Type** | **Description** |
| user\_id | String | User ID |
| proof\_public\_id | String | Proof |
| proof\_url | String | url |
| uploadedAt | Date | Uploaded date |
| Amount | Number | Amount |
| Comment | String | comment |

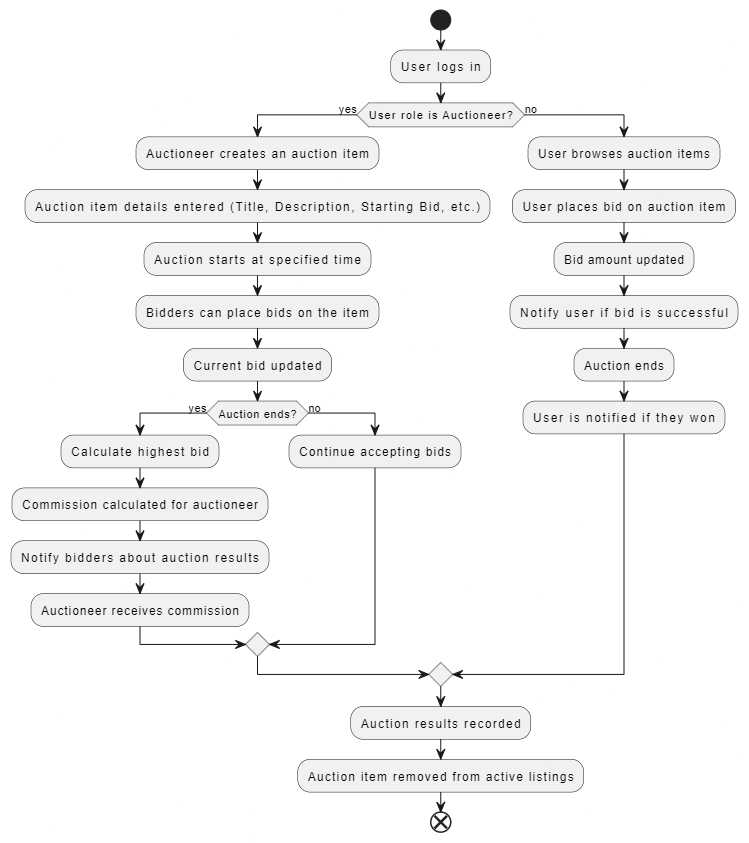
**3.4 Use Case Diagram:**

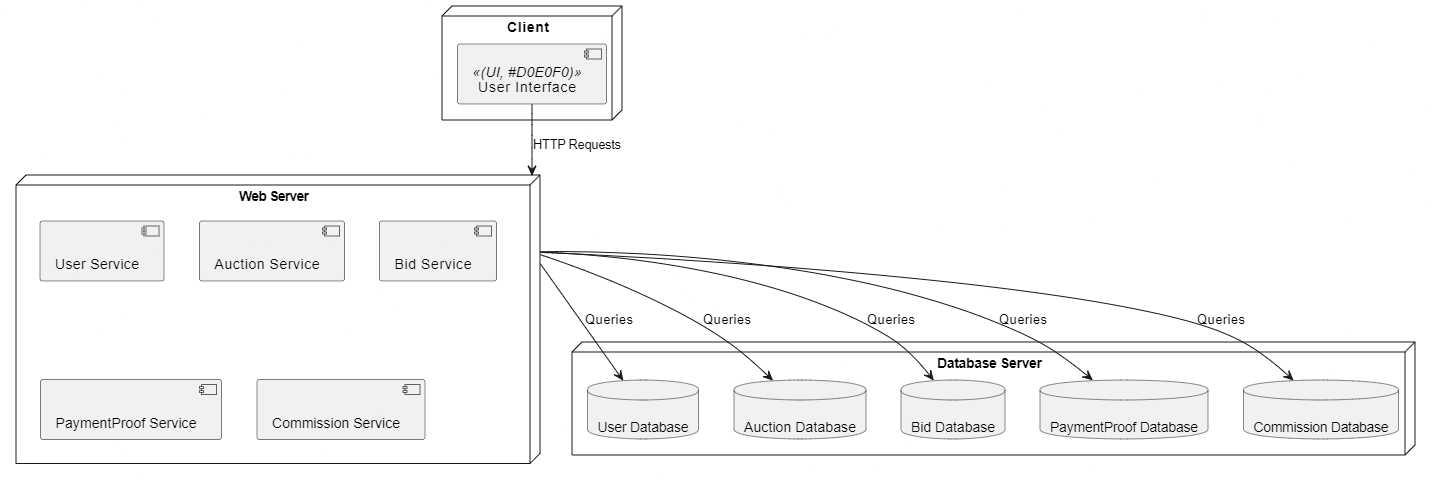


**3.5 Class Diagram:**



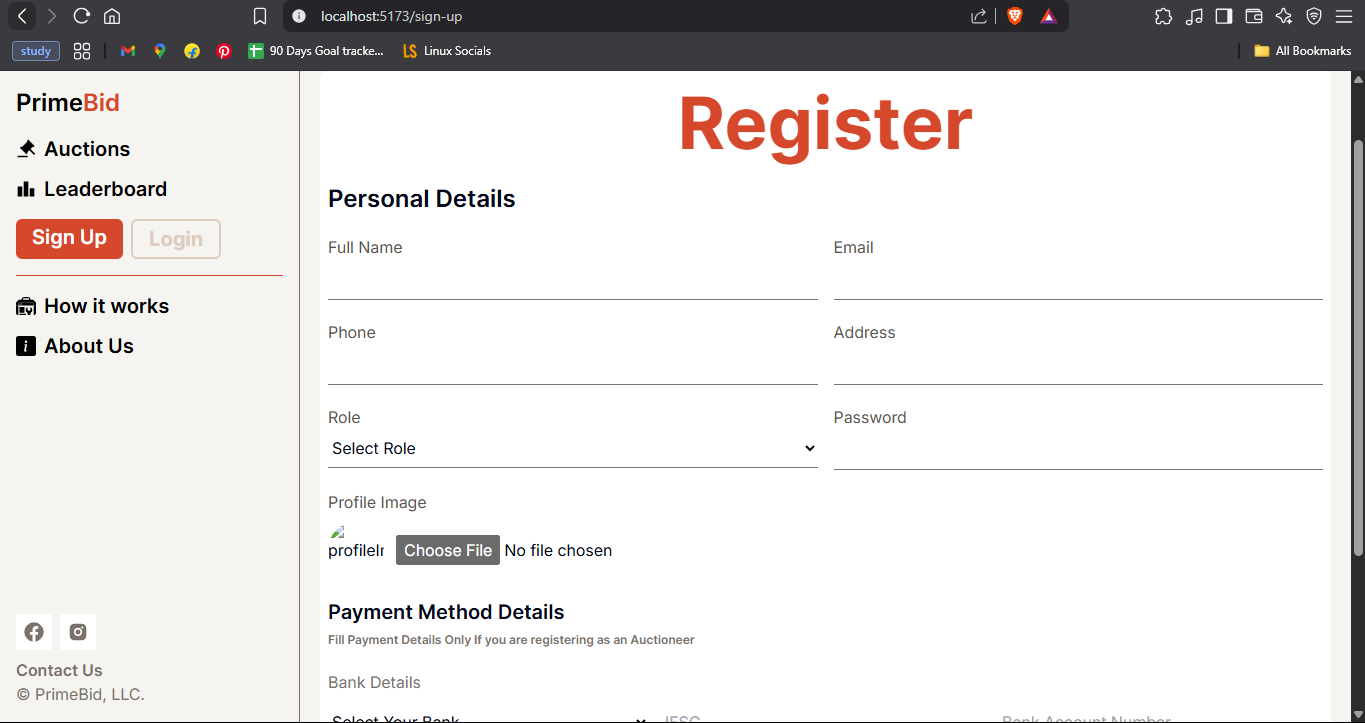
**3.6 Activity Diagram:**



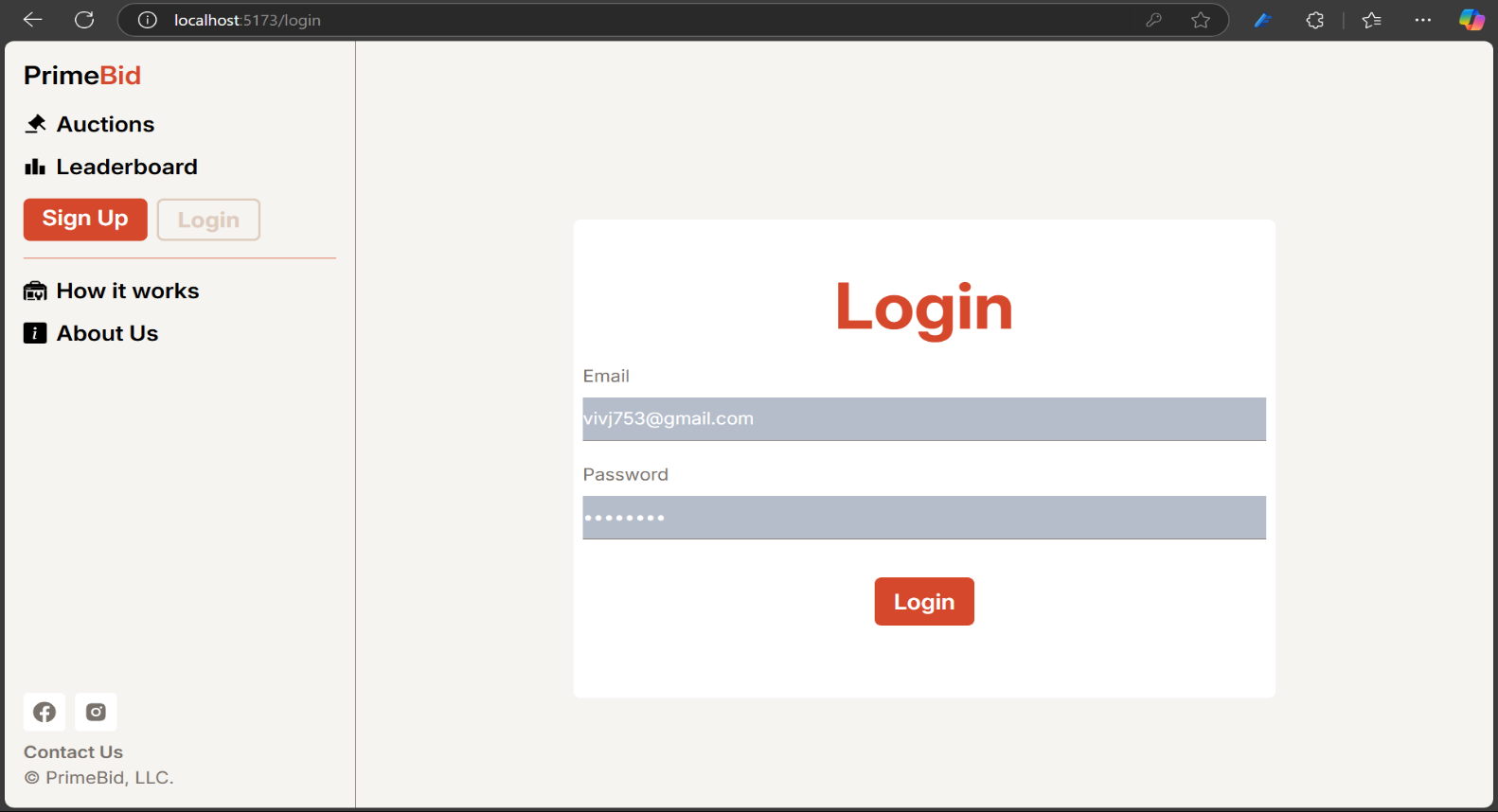
**3.7 Deployment Diagram:**

**3.9 Input / Output Screens**

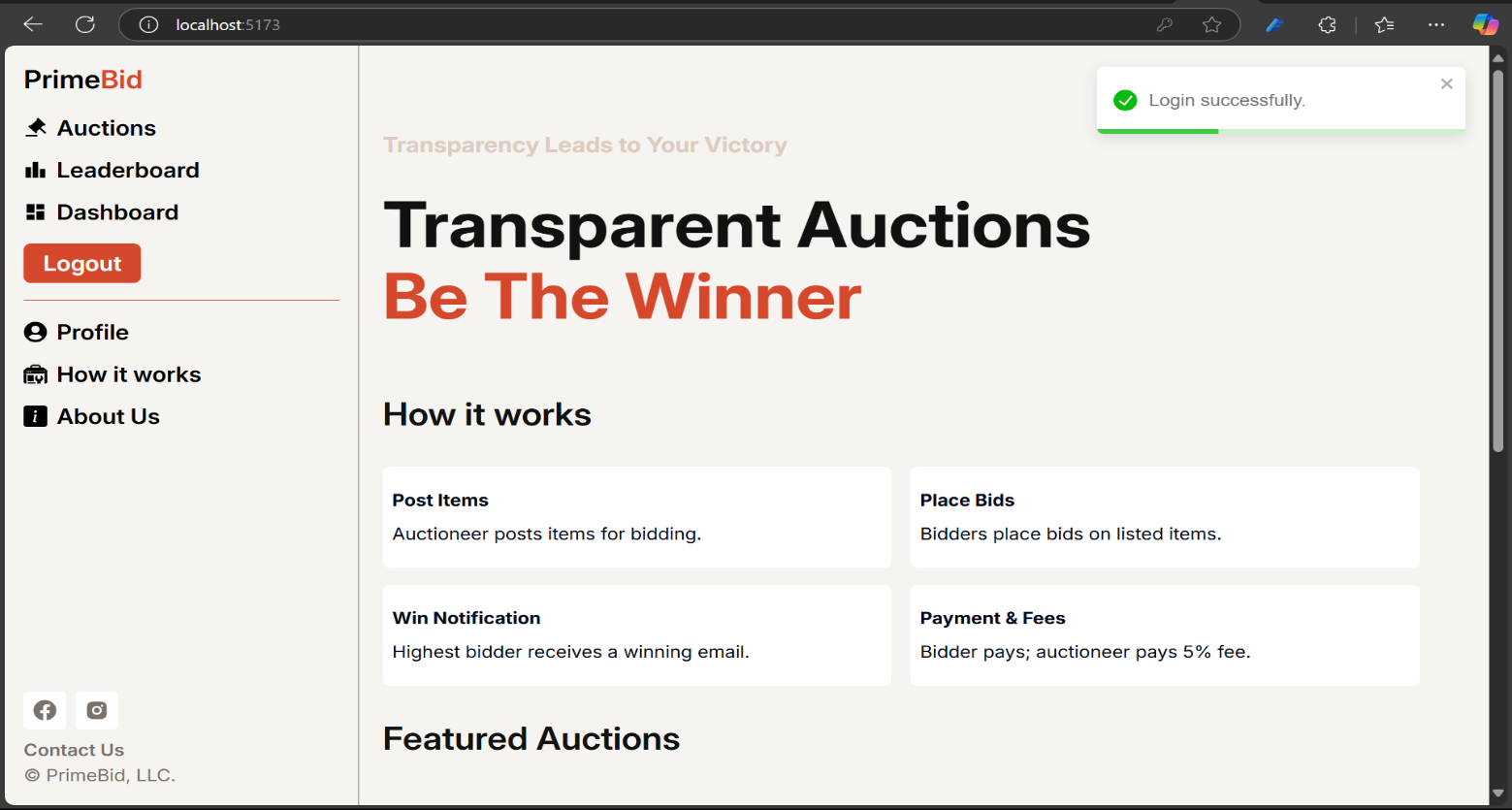
**Register Screen**

****

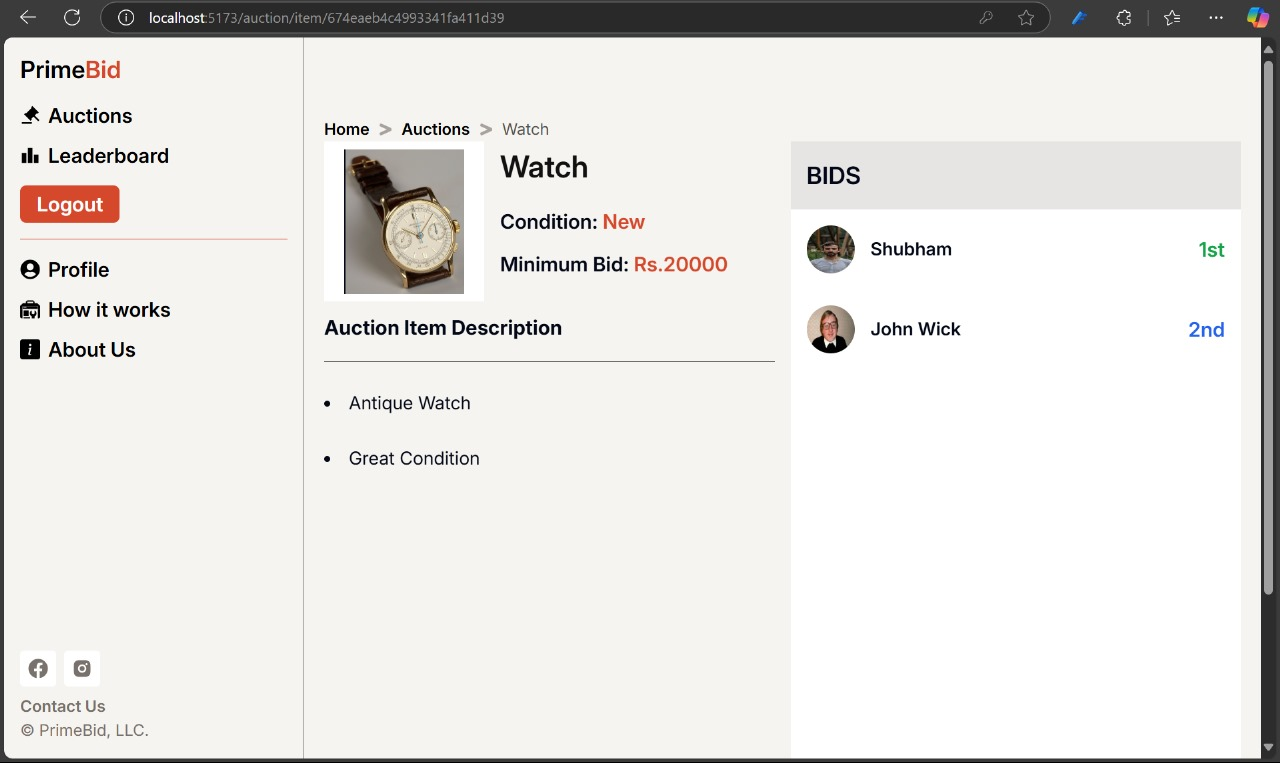
**Login Screen:**

****

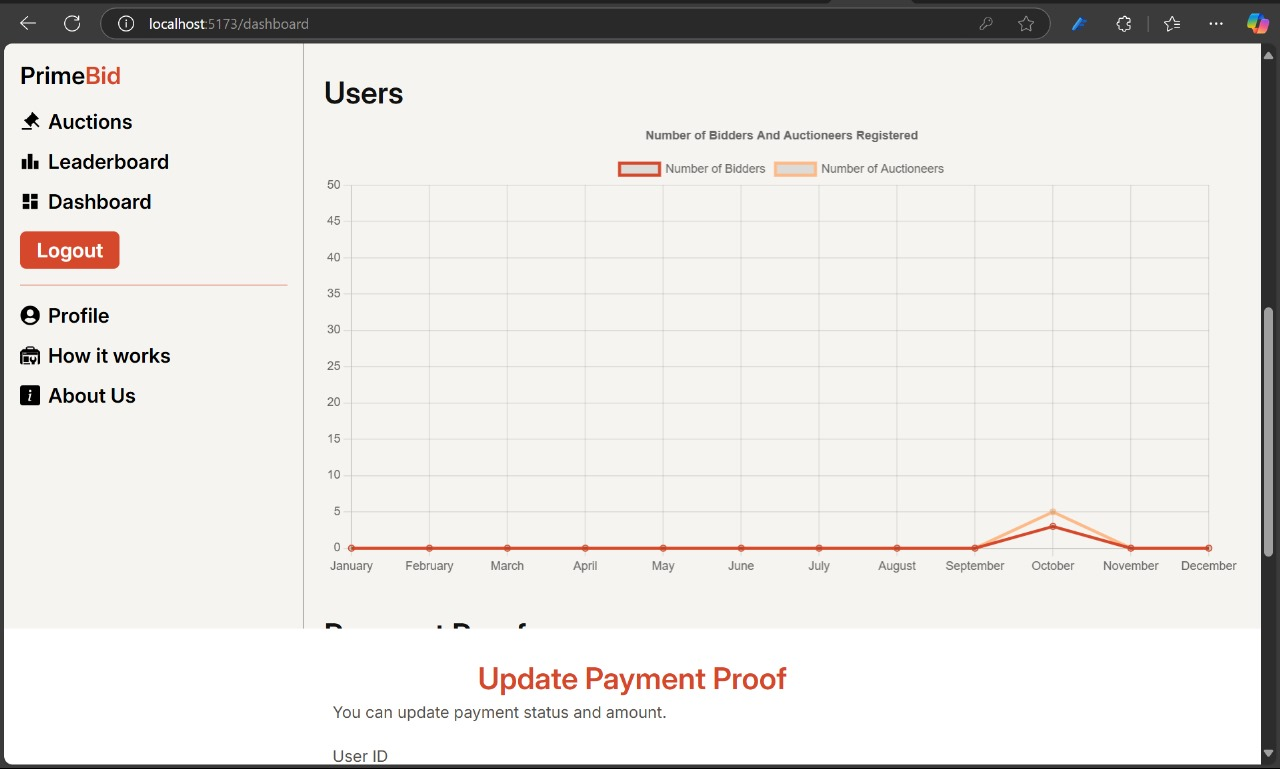
**Home Page:**



**Biding Page:**



**Admin Page :**



**4. Sample codes**

**4.1 Home.jsx**

import React from "react";

import { useSelector } from "react-redux";

import { Link } from "react-router-dom";

import FeaturedAuctions from "./home-sub-components/FeaturedAuctions";

import UpcomingAuctions from "./home-sub-components/UpcomingAuctions";

import Leaderboard from "./home-sub-components/Leaderboard";

import Spinner from "@/custom-components/Spinner";

const Home = () => {

  const howItWorks = [

    { title: "Post Items", description: "Auctioneer posts items for bidding." },

    { title: "Place Bids", description: "Bidders place bids on listed items." },

    {

      title: "Win Notification",

      description: "Highest bidder receives a winning email.",

    },

    {

      title: "Payment & Fees",

      description: "Bidder pays; auctioneer pays 5% fee.",

    },

  ];

  const { isAuthenticated } = useSelector((state) => state.user);

  return (

    <>

      <section className="w-full ml-0 m-0 h-fit px-5 pt-20 lg:pl-[320px] flex flex-col min-h-screen py-4 justify-center">

        <div>

          <p className="text-[#DECCBE] font-bold text-xl mb-8">

            Transparency Leads to Your Victory

          </p>

          <h1

            className={`text-[#111] text-2xl font-bold mb-2 min-[480px]:text-4xl md:text-6xl xl:text-7xl 2xl:text-8xl`}

          >

            Transparent Auctions

          </h1>

          <h1

            className={`text-[#d6482b] text-2xl font-bold mb-2 min-[480px]:text-4xl md:text-6xl xl:text-7xl 2xl:text-8xl`}

          >

            Be The Winner

          </h1>

          <div className="flex gap-4 my-8">

            {!isAuthenticated && (

              <>

                <Link

                  to="/sign-up"

                  className="bg-[#d6482b] font-semibold hover:bg-[#b8381e] rounded-md px-8 flex items-center py-2 text-white  transition-all duration-300"

                >

                  Sign Up

                </Link>

                <Link

                  to={"/login"}

                  className="text-[#DECCBE] bg-transparent border-2 border-[#DECCBE] hover:bg-[#fff3fd] hover:text-[#fdba88] font-bold text-xl  rounded-md px-8 flex items-center py-2 transition-all duration-300"

                >

                  Login

                </Link>

              </>

            )}

          </div>

        </div>

        <div className="flex flex-col gap-6">

          <h3 className="text-[#111] text-xl font-semibold mb-2 min-[480px]:text-xl md:text-2xl lg:text-3xl">How it works</h3>

          <div className="flex flex-col gap-4 md:flex-row md:flex-wrap w-full">

            {howItWorks.map((element) => {

              return (

                <div

                  key={element.title}

                  className="bg-white flex flex-col gap-2 p-2 rounded-md h-[96px] justify-center md:w-[48%] lg:w-[47%] 2xl:w-[24%] hover:shadow-md transition-all duration-300"

                >

                  <h5 className="font-bold">{element.title}</h5>

                  <p>{element.description}</p>

                </div>

              );

            })}

          </div>

        </div>

        <FeaturedAuctions />

        <UpcomingAuctions />

        <Leaderboard />

      </section>

    </>

  );

};

export default Home;

**4.2 ViewMyAuction.jsx**

import CardTwo from "@/custom-components/CardTwo";

import Spinner from "@/custom-components/Spinner";

import { getMyAuctionItems } from "@/store/slices/auctionSlice";

import React, { useEffect } from "react";

import { useDispatch, useSelector } from "react-redux";

import { useNavigate } from "react-router-dom";

const ViewMyAuctions = () => {

  const { myAuctions, loading } = useSelector((state) => state.auction);

  const { user, isAuthenticated } = useSelector((state) => state.user);

  const dispatch = useDispatch();

  const navigateTo = useNavigate();

  useEffect(() => {

    if (!isAuthenticated || user.role !== "Auctioneer") {

      navigateTo("/");

    }

    dispatch(getMyAuctionItems());

  }, [dispatch, isAuthenticated]);

  return (

    <>

      <div className="w-full ml-0 m-0 h-fit px-5 pt-20 lg:pl-[320px] flex flex-col">

        <h1

          className={`text-[#d6482b] text-2xl font-bold mb-2 min-[480px]:text-4xl md:text-6xl xl:text-7xl 2xl:text-8xl`}

        >

          My Auctions

        </h1>

        {loading ? (

          <Spinner />

        ) : (

          <div

            className={`${

              myAuctions.length > 2 && "flex-grow"

            } flex flex-wrap gap-6`}

          >

            {myAuctions.length > 0 ? (

              myAuctions.map((element) => {

                return (

                  <CardTwo

                    title={element.title}

                    startingBid={element.startingBid}

                    endTime={element.endTime}

                    startTime={element.startTime}

                    imgSrc={element.image?.url}

                    id={element.\_id}

                    key={element.\_id}

                  />

                );

              })

            ) : (

              <h3 className="text-[#666] text-xl font-semibold mb-2 min-[480px]:text-xl md:text-2xl lg:text-3xl mt-5">

                You have not posted any auction.

              </h3>

            )}{" "}

            :

          </div>

        )}

      </div>

    </>

  );

};

export default ViewMyAuctions;

**5 Testing**

**5.1 Test Strategy**

The PrimeBid Auction Platform's test strategy focuses on ensuring its functionality, security, and stability. Both manual and automated testing approaches were used.

Types of Testing Used:

* Unit Testing: To test individual functions like login, bid placement, etc.
* Integration Testing: Ensures communication between frontend, backend, and database is smooth.
* System Testing: To verify the whole platform works as expected.
* Security Testing: Ensures proper authentication and data protection.

**5.2 unit Test Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Description | Expected Result | Actual Result | Status |
| Login Test | Enter valid credentials | User should be redirected to dashboard | Working as expected | Pass |
| Bid Test | Place a valid bid on an active auction | Bid should be accepted | Working as expected | Pass |
| Role Check | Try accessing admin route as bidder | Access should be denied | Working as expected | Pass |
| Auction Timer | Wait till auction end time | Bidding should be disabled | Working as expected | Pass |
| Upload Proof | Upload image after winning bid | Image should be stored and shown to admin | Working as expected | Pass |

**5.3 Acceptance Test Plan**

This plan was used to verify the system meets all user requirements:

|  |  |  |
| --- | --- | --- |
| Scenario | Expected Outcome | Status |
| Admin can manage users and auctions | Admin dashboard loads, actions are successful | Pass |
| Seller can post a new auction | Auction should be created and visible | Pass |
| Bidder can join and bid live | Real-time bid updates and win logic works | Pass |
| System blocks expired auctions | No new bids accepted after end time | Pass |
| Secure login and logout | JWT-based token authentication works | Pass |

**5.4 Test cases:**

Future enhancements for Auction website could include:

**Test Cases for Auction Platform**

**1. User Authentication**

* **Registration:** Valid details → Account created; Invalid details → Error shown.
* **Login:** Correct credentials → Access granted; Incorrect → Error shown.
* **Logout:** User logs out → Redirect to login page.

**2. Item Listings**

* **Add Item:** Valid details → Item listed; Missing details → Error message.
* **Edit Item:** Modify item details → Changes saved successfully.
* **Delete Item:** Delete an item → Item removed from the listing.

**3. Bidding**

* **Place Bid:** Higher bid → Accepted; Lower bid → Rejected with error.
* **Auto Increment:** Bidder increases max bid → Updated highest bid.
* **Auction Close:** Bids stop after auction ends → Final winner declared.

**4. Notifications**

* **Bid Notification:** Notify users when outbid or won.

**5. Security**

* **SQL Injection:** Malicious input → Input sanitized, no database issues.
* **Unauthorized Access:** Access pages without login → Redirect to login.

**6. Reports**

* **Activity History:** User views their bids and winnings → Correct data shown.
* **Auction Summary:** Admin views auction stats → Data displayed accurately.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Steps | Test Data | Expected Result | Actual Result | Status |
| TC-01 | User Registration - Valid Input | 1. Navigate to registration page.  2. Enter valid details. 3. Click "Register". | Username: user1 Email: user1@example.com Password: P@ssword1 | Account created successfully, and confirmation displayed. | Account created successfully. | Pass |
| TC-02 | User Registration - Invalid Email | 1. Navigate to registration page.  2. Enter invalid email.  3. Click "Register". | Username: user2 Email: invalidEmail Password: P@ss1 | Error message "Invalid email format" displayed. | Invalid email format  displayed. | Pass |
| TC-03 | Add Item - Valid Input | 1. Log in.  2. Navigate to "Add Item".  3. Enter valid details and submit. | Name: Phone Price: $100 Description: New phone | Item successfully added and visible in auction listing. | Item successfully added and visible in auction listing. | Pass |
| TC-04 | Add Item - Missing Fields | 1. Log in.  2. Navigate to "Add Item".  3. Submit with missing name field. | Price: $100 Description: New phone | Error message "Name is required" displayed. | Name is required displayed. | Pass |
| TC-05 | Place Bid - Higher Bid | 1. Log in.  2. Select an auction item. 3. Enter a higher bid and submit. | Current Bid: $100 User Bid: $120 | Bid accepted, user becomes the highest bidder. | Bid accepted | Pass |
| TC-06 | Place Bid - Lower Bid | 1. Log in.  2. Select an auction item. 3. Enter a lower bid and submit. | Current Bid: $100 User Bid: $90 | Error message "Bid must be higher than current bid" displayed. | Bid must be higher than current bid displayed. | Pass |
| TC-07 | Notifications - Auction Won | 1. Auction ends with user3 as highest bidder. | Highest Bidder: user3 | User3 receives notification: "You won the auction!" | Get the notification-You won the auction | Pass |

**5.5 Defect Report**

**This defect report summarizes any issues encountered during the testing phase of the Auction Platform.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Defect ID** | **Test Case ID** | **Module** | **Description** | **Severity** | **Status** |
| **–** | **TC-01 to TC-07** | **Multiple** | **No defects encountered in current test cases.** | **–** | **Pass** |

**6. Limitations of the Proposed System**

**Technical Problems:**

Website downtime, slow page loads, and bugs can disrupt auctions, especially during high-stakes moments, damaging the user experience and trust.

**Limited User Accessibility:**

Users in different time zones or those with slow internet connections may have difficulty engaging in real-time auctions, which could limit the potential audience.

**Security Concerns:**

Auction websites can be vulnerable to hacking and fraud, especially in financial transactions and personal information storage.

Protection of payment details and user privacy is essential but can be costly and complex.

**7. Proposed Enhancement :**

* **AI and Machine Learning**: Implement AI for personalized recommendations, fraud detection, and bidding predictions.
* **Blockchain Technology**: Use blockchain for transparent, secure, and tamper-proof transaction records.
* **Augmented Reality (AR)**: Enable users to visualize products in 3D for a better understanding of item quality.
* **Improved Mobile Experience**: Optimize apps for faster navigation, notifications, and a user-friendly interface.
* **Global Expansion**: Introduce multilingual support, regional payment gateways, and international shipping options.

**8. Conclusion :**

The **PrimeBid – Online Auction Platform** successfully demonstrates the integration of modern web technologies to deliver a seamless and real-time bidding experience. This project showcases effective use of the **MERN stack (MongoDB, Express.js, React.js, Node.js)** along with **Socket.IO** for real-time bid updates and **role-based access control** for enhanced security and usability.

From designing a clean and responsive frontend to building a robust backend API and implementing live auction functionalities, this project provided a comprehensive full-stack development experience. The clear separation of user roles (Admin, Seller, Bidder) ensures secure and intuitive interactions, while features like winner declaration and auction management add real-world practicality.

Overall, PrimeBid is a scalable and production-ready application that reflects both technical proficiency and a strong understanding of user experience. This project has not only strengthened my backend and frontend development skills but also deepened my understanding of real-time web applications and full-stack architecture.

**9. Bibliography:**

* [**https://developer.ebay.com/**](https://developer.ebay.com/)
* [**https://www.biddingowl.com/**](https://www.biddingowl.com/)
* [**https://www.bidjs.com/**](https://www.bidjs.com/)
* [**https://woocommerce.com/products/woocommerce-auction/**](https://woocommerce.com/products/woocommerce-auction/)
* [**https://chatgpt.com/c/674ec957-1944-800a-8d1a-dea308fa2d08**](https://chatgpt.com/c/674ec957-1944-800a-8d1a-dea308fa2d08)
* [**www.youtube.com**](http://www.youtube.com)
* [**www.google.com**](http://www.google.com)