



Mini project report on

Auction Management System

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

Bachelor of Technology

in

Computer Science & Engineering

UE22CS351A – DBMS Project

Submitted by:

Tushar Swami

PES2UG22CS633

Yash Sinha

PES2UG22CS675

Under the guidance of

Dr. Geetha D.

Assistant Professor

PES University

AUG - DEC 2024

**DEPARTMENT OF COMPUTER SCIENCE AND 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

University Fest Management System

is a bonafide work carried out by

Tushar Swami

PES2UG22CS633

Yash Sinha

PES2UG22CS675

In partial fulfilment for the completion of fifth semester DBMS Project (UE22CS351A) 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. 2024 – DEC. 2024. 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. Geetha D.

Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Auction Management System** has been carried out by us under the guidance of **Dr. Geetha D, Associate 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 2024.

Tushar Swami

PES2UG22CS633

Yash Sinha

PES2UG22CS675

ABSTRACT

The Auction Management System is a desktop-based application designed to simplify the auction process by providing an intuitive, accessible platform for online bidding. Developed using Python's Tkinter library for the graphical user interface and MySQL as the backend database, this system enables secure, role-based access for administrators and bidders alike. Administrators can manage auction items, track user activities, and monitor bids in real-time, while bidders are able to view listings, place bids, and compete in active auctions.

This project addresses the limitations of traditional auction systems, such as the need for physical presence, by offering a streamlined digital experience. The local-server setup minimizes infrastructure costs, while the use of a password-based authentication system ensures user privacy and data integrity. The Auction Management System is a scalable solution, capable of handling essential functionalities in a structured, efficient manner. By bridging the gap between users and auctions, this application stands as a cost-effective, user-friendly approach to modernizing the auction experience.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	INTRODUCTION	6
2.	PROBLEM DEFINITION WITH USER REQUIREMENT SPECIFICATIONS	7
3.	LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED	10
4.	ER MODEL	12
5.	ER TO RELATIONAL MAPPING	13
6.	DDL STATEMENTS	14
7.	DML STATEMENTS (CRUD OPERATION SCREENSHOTS)	17
8.	QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)	19
9.	STORED PROCEDURE, FUNCTIONS AND TRIGGERS	21
10.	FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)	26
11.	CONCLUSION	34
12.	REFERENCES/BIBLIOGRAPHY	35
13.	APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS	37
14.	GITHUB REPOSITORY LINK	38

CHAPTER 1: INTRODUCTION

The Auction Management System is a software application designed to manage online auctions, providing a platform where users can list items for bidding, place bids, and track auction progress in real time. In traditional auction systems, participants are often required to be physically present, and the auction process itself can be slow and cumbersome. This project aims to modernize and simplify the auction process by creating a digital platform that allows users to interact seamlessly through a desktop application.

Built using Python's Tkinter library for the user interface and MySQL for backend data management, the Auction Management System offers a structured, easy-to-navigate environment for both administrators and bidders. Administrators can oversee auction events, manage listings, and monitor bidding activity, while bidders can log in securely, browse available items, and participate in active auctions. The platform incorporates role-based access control to ensure that users only access features relevant to their roles.

This project leverages the benefits of a local-server setup, which makes development and testing straightforward while keeping infrastructure costs low. It also emphasizes a password-based login system, providing a secure yet user-friendly approach to managing user accounts and ensuring data privacy. By digitizing the auction process, the Auction Management System provides a comprehensive solution for those seeking a streamlined and accessible bidding experience.

CHAPTER 2: PROBLEM DEFINITION WITH USER REQUIREMENT SPECIFICATION

➔ PROBLEM STATEMENT

Traditional auction systems are often limited by the need for participants to be physically present, making it challenging to engage a broader audience and creating logistical barriers to effective participation. This setup can be slow, inefficient, and inconvenient, as it requires manual tracking of bids, item listings, and user participation. Furthermore, managing auctions on paper or through fragmented processes increases the likelihood of errors, delays, and security vulnerabilities, as sensitive user information and bid data lack a centralized, protected storage solution.

In the digital age, there is a clear demand for a robust, secure, and efficient platform that enables users to conduct auctions online, eliminating the limitations of physical presence and manual processes. Such a system should ensure seamless user access, maintain data integrity, and offer intuitive functionality to meet the needs of both administrators (who manage items, bids, and user activity) and bidders (who engage in auctions and place bids).

The challenge, therefore, is to design and implement a comprehensive Auction Management System that enables real-time auction participation, simplifies user interactions, and provides secure data management. This system aims to facilitate a modern, accessible, and scalable solution to address the shortcomings of traditional auction models.

➔ USER REQUIREMENT SPECIFICATION

In the dynamic and ever-expanding realm of e-commerce, the Auction Management System emerges as a vital platform where sellers can efficiently list items, and bidders can engage in real-time auctions. With the global online auction market projected to grow significantly, driven by increasing consumer preference for diverse purchasing options and the thrill of bidding, this system is perfectly positioned to capitalize on these trends. It offers a transparent, user-friendly bidding mechanism that not only ensures the security and accuracy of payment transactions

but also adapts to the increasing demand for more interactive and engaging shopping experiences. Through its sophisticated design, the system facilitates seamless transactions while maintaining rigorous data integrity and security standards, catering to the modern user's expectations for reliability and transparency in dealings. The system starts by capturing the user's essential details. Each user is uniquely identified by a User ID and may serve as either a seller or bidder.

Users provide their full name, including first name, middle name, lastname, username, password, email, multiple phone number, and role (whether they are buyers or sellers). The system also records the user's address, which includes detailed information such as street name, city and pincode. A seller is one among the major pillars of the system as without them it is impossible to even create a word called as auction. If a user willing to be a seller has a unique seller_id, status of its product, its username & more than one phone-numbers if available.

A bidder yet another major pillar of the system is responsible in buying the product he/she prefers via bidding. A bidder identified with a username, a unique bidder id, user_id, its phone & status of bidder whether the bidder is active or not. The heart of the system is the Item entity, representing the products available for auction aka status of product. Each item has a unique Item ID, and details such as the item's title, base price, description, start date, end date, and status (active or inactive) are recorded. Items also contains description of the product that helps in facilitating the buyer to know more about the product. The Bidding Process is managed through the Bid entity. A user can place multiple bids on different items, each recorded by a unique Bid ID. The system captures essential details such as the bid amount, and the corresponding bidder and item. Each bid links to both a bidder (user) and an item. Bidders can place bids on any number of items as long as the auction is still live, thus allowing flexibility in the process. Payments are an essential aspect of the system. Once the bidding process is complete, the payment entity ensures that users can settle their transactions using a variety of payment methods. Payments are identified by a Payment ID and can be made via UPI, credit, cash-on-delivery (COD), or net banking. Each payment is linked to a specific bid and item to ensure transparency and accountability. This entity securely stores the payment mode, details, and the associated bid information. The system incorporates a robust Review containing attributes like review id, item id, rating & overview of product mechanism, functioning as a weak entity, which is dependent on the Item for its identification. Each review is uniquely tied to both a specific item (via Item ID) and the bidder who has won the auction, ensuring accountability from both the buyer and the seller. As an identifying relationship, the Item ID serves as the key link that allows the Review entity to exist, emphasizing the connection between the feedback provided and the item it corresponds to. Ratings and reviews play a crucial role in

maintaining the transparency and integrity of the auction platform, fostering trust between users and enhancing the overall experience. The system ensures that multiple User Roles are managed efficiently. Administrators have full control over the system, managing users, items, bids, and payments. Sellers can only manage their own items, and bidders are allowed to participate in the bidding process and provide feedback on items they've interacted with.

CHAPTER 3: LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED

1. Python

- Description: Python is the primary programming language used for developing the Auction Management System. Known for its simplicity, readability, and extensive library support, Python is ideal for building desktop applications with a strong backend integration.

- Purpose: Python handles the application's core logic, user interface integration, and database operations, making it the backbone of the Auction Management System.

2. Tkinter

- Description: Tkinter is Python's standard GUI (Graphical User Interface) library, which provides a toolkit for creating user-friendly and interactive interfaces.

- Purpose: Tkinter is used to design the front-end interface of the Auction Management System, allowing users to interact with the system through graphical components like buttons, forms, and display windows.

3. MySQL

- Description: MySQL is a relational database management system known for its performance, reliability, and ease of use. It is a popular choice for backend data storage due to its ability to manage structured data efficiently.

- Purpose: MySQL serves as the database for storing all auction data, including user information, auction items, bid histories, and auction results. It ensures data integrity and supports efficient querying for real-time updates.

4. MySQL Workbench

- Description: MySQL Workbench is a graphical tool that provides functionalities for database design, management, and SQL development. It offers a visual interface for creating and managing MySQL databases.

- Purpose: MySQL Workbench is used for database modeling and management, allowing the team to create the initial database schema and manage tables, relationships, and other database components throughout development.

5. Python IDE (VS Code)

- Description: Visual Studio Code (VS Code) is a popular code editor known for its versatility, debugging tools, and extensive library of extensions.

- Purpose: VS Code is used for writing, editing, and debugging Python code, offering features like syntax highlighting, integrated terminal, and Git integration to streamline development for the Auction Management System.

6. SQL for Database Queries

- Description: SQL (Structured Query Language) is a standard language for managing and querying relational databases.

- Purpose: SQL is used within the Python application to interact with the MySQL database, including creating tables, inserting, updating, and retrieving data as needed for the auction operations.

7. Excel (for Testing Documentation)

- Description: Microsoft Excel or any other spreadsheet software is commonly used for documenting test cases, tracking testing results, and organizing testing data.

- Purpose: Excel serves as the format for test case management, where each test case is documented with pre-conditions, steps, expected results, actual results, and test status to ensure comprehensive testing coverage for the Auction Management System.

8. Git (for Version Control)

- Description: Git is a distributed version control system that helps track changes to code and collaborate with multiple team members.

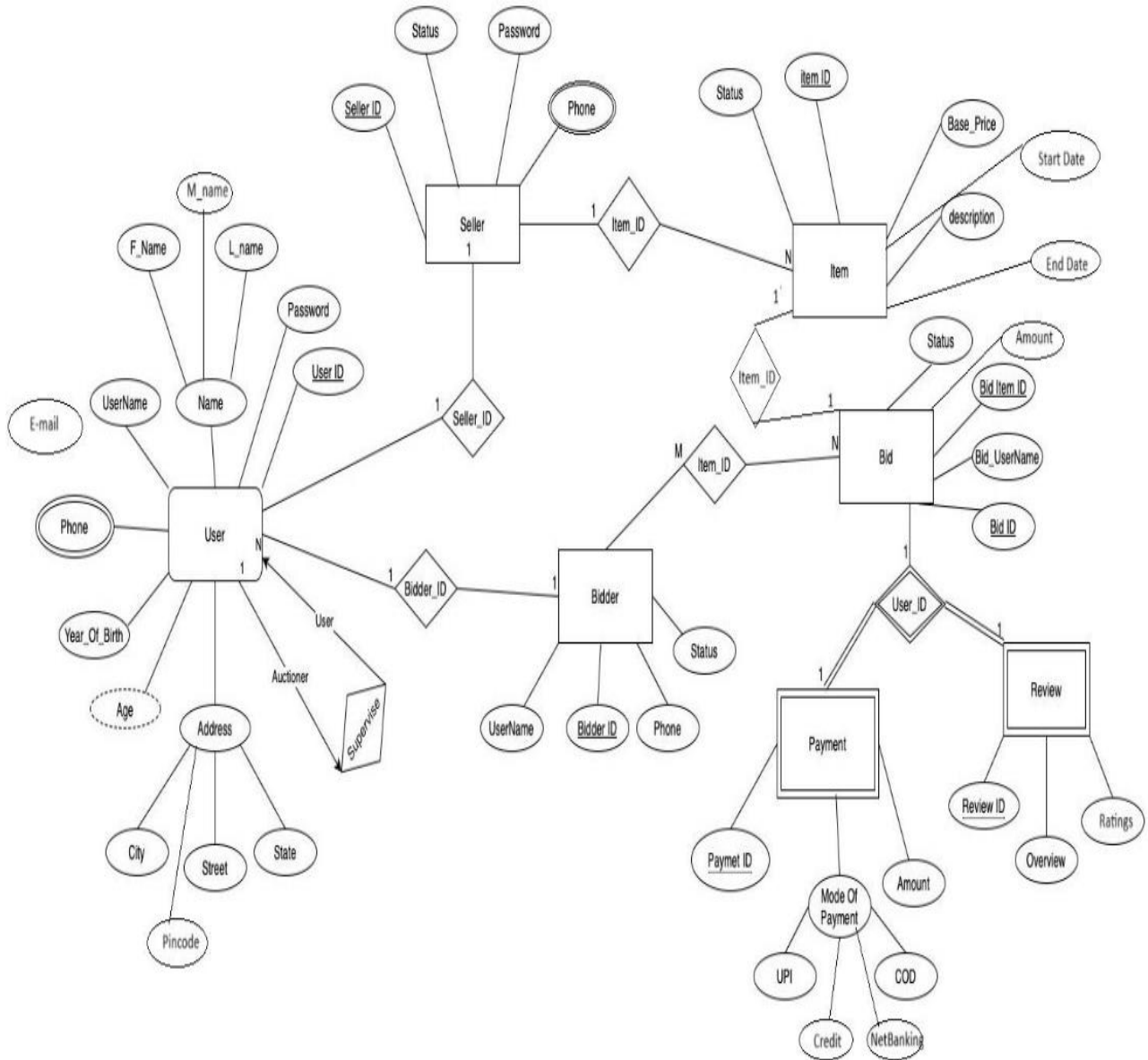
- Purpose: Git is used to manage the source code of the project, enabling collaboration, version tracking, and safe experimentation with new features or bug fixes through branching and merging.

9. Canva and Figma (for Design and Architecture Diagrams)

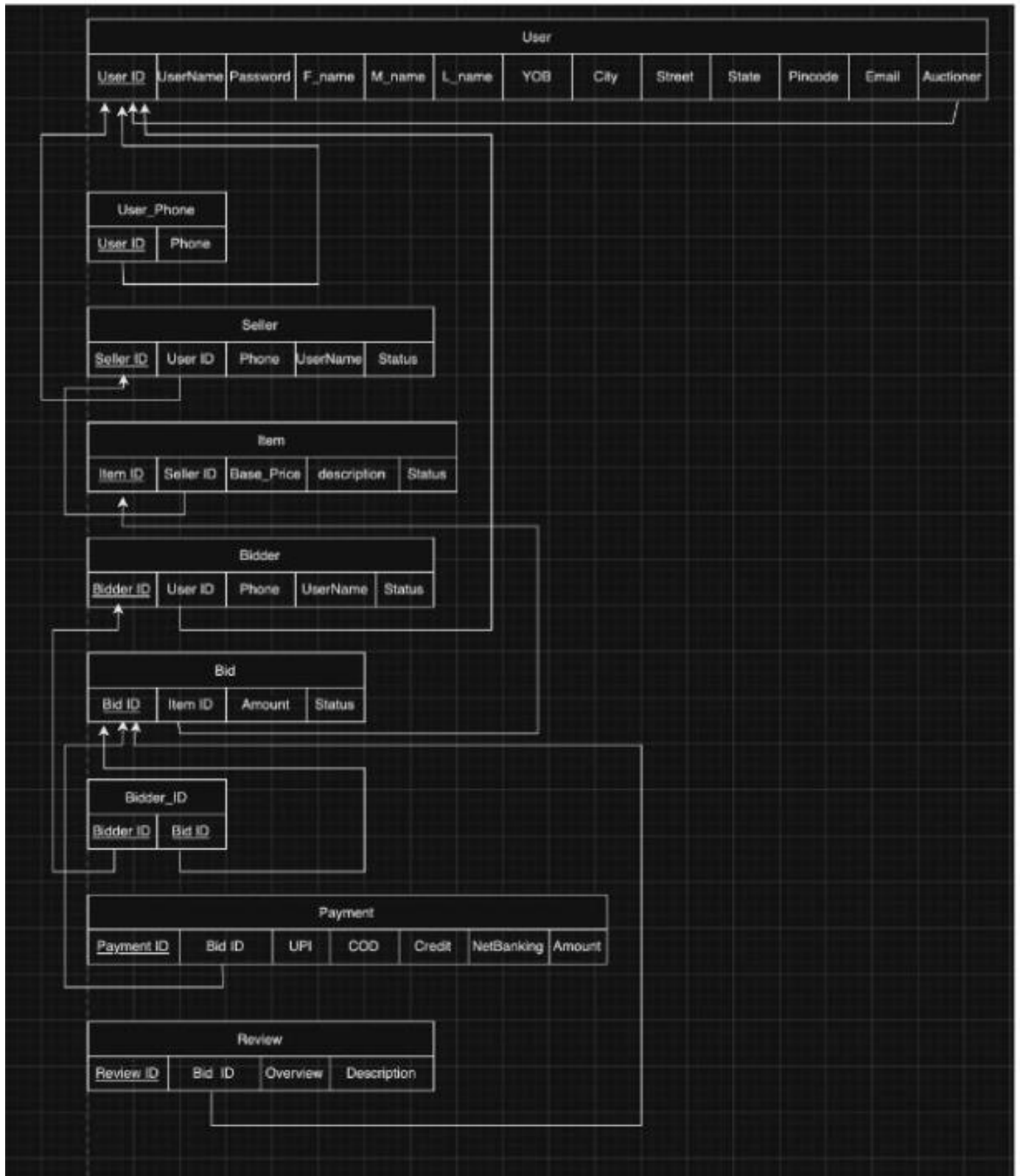
- Description: Canva and Figma are design tools used for creating high-quality flowcharts, ER diagrams, and UI mockups.

- Purpose: These tools are used to design system architecture, ER diagrams, and other visual representations required for planning, documentation, and presentation of the Auction Management System.

CHAPTER 4: ER MODEL



CHAPTER 5: ER TO RELATIONAL MAPPING



CHAPTER 6: DDL STATEMENTS

```
-- User Table
CREATE TABLE User (
    User_ID INT AUTO_INCREMENT Unique,
    Password VARCHAR(255) NOT NULL Unique,
    F_Name VARCHAR(255),
    M_Name VARCHAR(255),
    L_Name VARCHAR(255),
    Year_Of_Birth INT DEFAULT NULL,
    City VARCHAR(255) NOT NULL,
    Street VARCHAR(255) DEFAULT NULL,
    State VARCHAR(255) NOT NULL,
    Pincode INT NOT NULL,
    Email VARCHAR(255) NOT NULL CHECK (Email LIKE '%@%') Unique,
    PRIMARY KEY (User_ID)
);
```

```
-- Bidder Table
CREATE TABLE Bidder (
    Bidder_ID INT AUTO_INCREMENT Unique,
    User_ID INT UNIQUE,
    Username VARCHAR(255),
    Status VARCHAR(50) NOT NULL,
    PRIMARY KEY (Bidder_ID),
    FOREIGN KEY (User_ID) REFERENCES User(User_ID) ON DELETE CASCADE
);
```

```
-- Bid Table
CREATE TABLE Bid (
    Bid_ID INT AUTO_INCREMENT,
    Item_ID INT,
    Amount DECIMAL(10, 2) NOT NULL,
    Status VARCHAR(50) NOT NULL,
    PRIMARY KEY (Bid_ID),
    FOREIGN KEY (Item_ID) REFERENCES Item(Item_ID) ON DELETE CASCADE
);
```

-- Seller Table

```
CREATE TABLE Seller (  
    Seller_ID INT AUTO_INCREMENT Unique,  
    User_ID INT UNIQUE,  
    Username VARCHAR(255),  
    Status VARCHAR(50),  
    PRIMARY KEY (Seller_ID),  
    FOREIGN KEY (User_ID) REFERENCES User(User_ID) ON DELETE CASCADE  
);
```

-- Item Table

```
CREATE TABLE Item (  
    Item_ID INT AUTO_INCREMENT,  
    Seller_ID INT,  
    Base_Price DECIMAL(10, 2),  
    Description TEXT DEFAULT NULL,  
    Status VARCHAR(50) NOT NULL,  
    PRIMARY KEY (Item_ID),  
    FOREIGN KEY (Seller_ID) REFERENCES Seller(Seller_ID) ON DELETE CASCADE  
);
```

-- Bidder_Bid Table

```
CREATE TABLE Bidder_Bid (  
    Bidder_ID INT,  
    Bid_ID INT,  
    FOREIGN KEY (Bidder_ID) REFERENCES Bidder(Bidder_ID),  
    FOREIGN KEY (Bid_ID) REFERENCES Bid(Bid_ID) ON DELETE CASCADE  
);
```

-- User_Phone Table

```
CREATE TABLE User_Phone (  
    User_ID INT,  
    Phone VARCHAR(10) NOT NULL Unique,  
    FOREIGN KEY (User_ID) REFERENCES User(User_ID) ON DELETE CASCADE  
);
```



```
-- Review Table
CREATE TABLE Review (
    Review_ID INT AUTO_INCREMENT,
    Bid_ID INT,
    Overview TEXT NOT NULL,
    Description TEXT DEFAULT NULL,
    PRIMARY KEY (Review_ID),
    FOREIGN KEY (Bid_ID) REFERENCES Bid(Bid_ID) ON DELETE CASCADE
);
```


CHAPTER 7: DML STATEMENTS [CRUD OPERATIONS SCREENSHOT]

```
-- Inserting data into User Table
```

```
INSERT INTO User (Username, Password, F_Name, M_Name, L_Name, Year_Of_Birth, City, Street, State, Pincode, Email)
VALUES
('john_doe', 'pass123', 'John', NULL, 'Doe', 1985, 'New York', '5th Ave', 'NY', 10001, 'john_doe@gmail.com'),
('jane_smith', 'pass456', 'Jane', NULL, 'Smith', 1990, 'Los Angeles', 'Main St', 'CA', 90001, 'jane_smith@gmail.com'),
('alex_brown', 'pass789', 'Alex', 'Michael', 'Brown', 1992, 'San Francisco', 'Market St', 'CA', 94105, 'alex_brown@gmail.com'),
('emily_jones', 'pass101', 'Emily', NULL, 'Jones', 1995, 'Chicago', 'Lake St', 'IL', 60601, 'emily_jones@gmail.com'),
('michael_green', 'pass102', 'Michael', NULL, 'Green', 1988, 'Miami', 'Ocean Dr', 'FL', 33101, 'michael_green@gmail.com'),
('linda_white', 'pass103', 'Linda', NULL, 'White', 1993, 'Boston', 'Beacon St', 'MA', 02108, 'linda_white@gmail.com'),
('kevin_wilson', 'pass104', 'Kevin', NULL, 'Wilson', 1986, 'Dallas', 'Elm St', 'TX', 75201, 'kevin_wilson@gmail.com'),
('sarah_clark', 'pass105', 'Sarah', NULL, 'Clark', 1991, 'Seattle', '1st Ave', 'WA', 98101, 'sarah_clark@gmail.com'),
('jason_king', 'pass106', 'Jason', NULL, 'King', 1984, 'Houston', 'Main St', 'TX', 77002, 'jason_king@gmail.com'),
('olivia_walker', 'pass107', 'Olivia', NULL, 'Walker', 1994, 'Denver', 'Colfax Ave', 'CO', 80202, 'olivia_walker@gmail.com');
```

```
-- Inserting data into User_Phone Table
```

```
INSERT INTO User_Phone (User_ID, Phone)
VALUES
(1, '1234567890'),
(2, '0987654321'),
(3, '1122334455'),
(4, '2233445566'),
(5, '3344556677'),
(6, '4455667788'),
(7, '5566778899'),
(8, '6677889900'),
(9, '7788990011'),
(10, '8899001122');
```

```
-- Inserting data into Seller Table
```

```
INSERT INTO Seller (User_ID, Username, Status)
VALUES
(1, 'john_doe', 'Active'),
(2, 'jane_smith', 'Active'),
(4, 'emily_jones', 'Active'),
(5, 'michael_green', 'Inactive'),
(7, 'kevin_wilson', 'Active');
```

```
INSERT INTO Bidder (User_ID, Username, Status)
VALUES
(2, 'jane_smith', 'Active'),
(3, 'alex_brown', 'Active'),
(6, 'linda_white', 'Active'),
(8, 'sarah_clark', 'Inactive'),
(9, 'jason_king', 'Active'),
(10, 'olivia_walker', 'Active');
```

CHAPTER 8: QUERIES [JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY]

JOIN QUERY

```
query = """
SELECT
    Bid.Bid_ID, Item.Item_ID, Item.Description, Item.Base_Price,
    Bid.Amount AS Bid_Amount, Bid.Status,
    (SELECT MAX(B.Amount) FROM Bid B WHERE B.Item_ID = Item.Item_ID) AS Highest_Bid,
    CASE WHEN func1(Item.Item_ID) = 1 THEN 'On' ELSE 'Off' END AS Bid_Status
FROM Item
JOIN Bid ON Item.Item_ID = Bid.Item_ID
JOIN Bidder_Bid ON Bid.Bid_ID = Bidder_Bid.Bid_ID
WHERE Bidder_Bid.Bidder_ID = %s
"""
```

```
cursor.execute("""
    SELECT p.*
    FROM Payment p
    JOIN Bid b ON p.Bid_ID = b.Bid_ID
    WHERE b.Item_ID = %s
""", (item_id,))
```

AGGREGATE FUNCTION QUERIES

```
cursor.execute("""
SELECT Mode_Of_Payment, COUNT(*)
FROM Payment
GROUP BY Mode_Of_Payment
""")
payment_counts = cursor.fetchall()
```

NESTED QUERY

```
query = """
SELECT
    Bid.Bid_ID, Item.Item_ID, Item.Description, Item.Base_Price,
    Bid.Amount AS Bid_Amount, Bid.Status,
    (SELECT MAX(B.Amount) FROM Bid B WHERE B.Item_ID = Item.Item_ID) AS Highest_Bid,
    CASE WHEN func1(Item.Item_ID) = 1 THEN 'On' ELSE 'Off' END AS Bid_Status
FROM Item
JOIN Bid ON Item.Item_ID = Bid.Item_ID
JOIN Bidder_Bid ON Bid.Bid_ID = Bidder_Bid.Bid_ID
WHERE Bidder_Bid.Bidder_ID = %s
"""
```

```
cursor.execute("""
SELECT COUNT(*)
FROM Payment
WHERE Bid_ID IN
(SELECT Bid_ID FROM Bid WHERE Item_ID = %s)
""",
(item_id,))
```

CHAPTER 9: STORED PROCEDURE, FUNCTIONS AND TRIGGERS

STORED PROCEDURE

```
CREATE DEFINER='root'@'localhost' PROCEDURE `AddBidder`(  
    IN userId INT,  
    IN username VARCHAR(255)  
)  
BEGIN  
    DECLARE bidderExists INT;  
  
    -- Check if the bidder already exists  
    SELECT COUNT(*) INTO bidderExists  
    FROM Bidder  
    WHERE User_ID = userId;  
  
    -- If the bidder doesn't exist, insert a new bidder  
    IF bidderExists = 0 THEN  
        INSERT INTO Bidder (User_ID, Username, Status)  
        VALUES (userId, username, 'Active');  
    ELSE  
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Bidder already exists.';  
    END IF;  
END
```

```
CREATE DEFINER='root'@'localhost' PROCEDURE `AddNewUser`(  
    IN p_Username VARCHAR(255),  
    IN p_Password VARCHAR(255),  
    IN p_FName VARCHAR(255),  
    IN p_MName VARCHAR(255),  
    IN p_LName VARCHAR(255),  
    IN p_YearOfBirth INT,  
    IN p_City VARCHAR(255),  
    IN p_Street VARCHAR(255),  
    IN p_State VARCHAR(255),  
    IN p_Pincode INT,  
    IN p_Email VARCHAR(255)  
)  
BEGIN  
    INSERT INTO User (Username, Password, F_Name, M_Name, L_Name, Year_Of_Birth,  
    VALUES (p_Username, p_Password, p_FName, p_MName, p_LName, p_YearOfBirth, p  
END
```



```

CREATE DEFINER='root'@'localhost' PROCEDURE `AddSeller`(
    IN userId INT,
    IN username VARCHAR(255)
)
BEGIN
    DECLARE sellerExists INT;

    -- Check if the seller already exists
    SELECT COUNT(*) INTO sellerExists
    FROM Seller
    WHERE User_ID = userId;

    -- If the seller doesn't exist, insert a new seller
    IF sellerExists = 0 THEN
        INSERT INTO Seller (User_ID, Username, Status)
        VALUES (userId, username, 'Active');
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Seller already exists.';
    END IF;
END

```

```

CREATE DEFINER='root'@'localhost' PROCEDURE `AddSellerItem`(
    IN p_user_id INT,
    IN p_username VARCHAR(255),
    IN p_status VARCHAR(50),
    IN p_base_price DECIMAL(10, 2),
    IN p_description TEXT,
    IN p_item_status VARCHAR(50)
)
BEGIN
    DECLARE v_seller_id INT;

    -- Check if the User_ID already exists
    IF NOT EXISTS (SELECT 1 FROM Seller WHERE User_ID = p_user_id) THEN
        -- Insert into Seller table if User_ID does not exist
        INSERT INTO Seller (User_ID, Username, Status)
        VALUES (p_user_id, p_username, p_status);

        -- Get the last inserted Seller_ID
        SET v_seller_id = LAST_INSERT_ID();
    ELSE
        -- Retrieve existing Seller_ID for existing User_ID
        SELECT Seller_ID INTO v_seller_id FROM Seller WHERE User_ID = p_user_id;
    END IF;

    -- Insert into Item table
    INSERT INTO Item (Seller_ID, Base_Price, Description, Status)
    VALUES (v_seller_id, p_base_price, p_description, p_item_status);
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_bidder_bid`(
  IN p_Bidder_ID INT,
  IN p_Bid_ID INT
)
BEGIN
  -- Insert a new record into Bidder_Bid table
  INSERT INTO Bidder_Bid (Bidder_ID, Bid_ID)
  VALUES (p_Bidder_ID, p_Bid_ID);
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_payment`(
  IN p_bid_id INT,
  IN p_amount DECIMAL(10, 2),
  IN p_method ENUM('UPI', 'COD', 'Net_Banking', 'Credit_Card')
)
BEGIN
  INSERT INTO Payment (Bid_ID, Amount, Mode_Of_Payment)
  VALUES (p_bid_id, p_amount, p_method);
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `submit_bid`(IN p_item_id INT, IN p_bid_amount DECIMAL(10, 2),
BEGIN
  INSERT INTO Bid (Item_ID, Amount, Status)
  VALUES (p_item_id, p_bid_amount, p_status);
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `submit_or_update_bid`(
  IN p_Item_ID INT,
  IN p_Amount DECIMAL(10, 2),
  IN p_Status VARCHAR(50)
)
BEGIN

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `update_bid`(IN bid_id INT, IN new_amount DECIMAL(10, 2),
BEGIN
  UPDATE Bid
  SET Amount = new_amount, Status = new_status
  WHERE Bid_ID = bid_id;
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `insert_bid_and_link_bidder`(
  IN p_bidder_id INT,
  IN p_item_id INT,
  IN p_amount DECIMAL(10, 2),
  IN p_status VARCHAR(50)
)
BEGIN
  DECLARE new_bid_id INT;

  -- Insert into the Bid table
  INSERT INTO Bid (Item_ID, Amount, Status)
  VALUES (p_item_id, p_amount, p_status);

  -- Get the last inserted Bid_ID
  SET new_bid_id = LAST_INSERT_ID();

  -- Insert into the Bidder_Bid table to link the bidder with the bid
  INSERT INTO Bidder_Bid (Bidder_ID, Bid_ID)
  VALUES (p_bidder_id, new_bid_id);
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `SubmitReview`(
  IN p_Bid_ID INT,
  IN p_Overview TEXT,
  IN p_Description TEXT
)
BEGIN
  INSERT INTO Review (Bid_ID, Overview, Description)
  VALUES (p_Bid_ID, p_Overview, p_Description);
END

```

```

CREATE DEFINER=`root`@`localhost` PROCEDURE `update_bidder_bid`(IN bidder_id INT, IN bid_id INT)
BEGIN
  IF EXISTS (SELECT * FROM Bidder_Bid WHERE Bidder_ID = bidder_id AND Bid_ID = bid_id) THEN
    -- If entry exists, update it
    UPDATE Bidder_Bid
    SET Bid_ID = bid_id
    WHERE Bidder_ID = bidder_id;
  ELSE
    -- If entry does not exist, insert a new one
    INSERT INTO Bidder_Bid (Bidder_ID, Bid_ID) VALUES (bidder_id, bid_id);
  END IF;
END

```


FUNCTIONS

```
CREATE DEFINER='root'@'localhost' FUNCTION `func1`(item_id INT) RETURNS int
    DETERMINISTIC
BEGIN
    DECLARE bid_status INT;

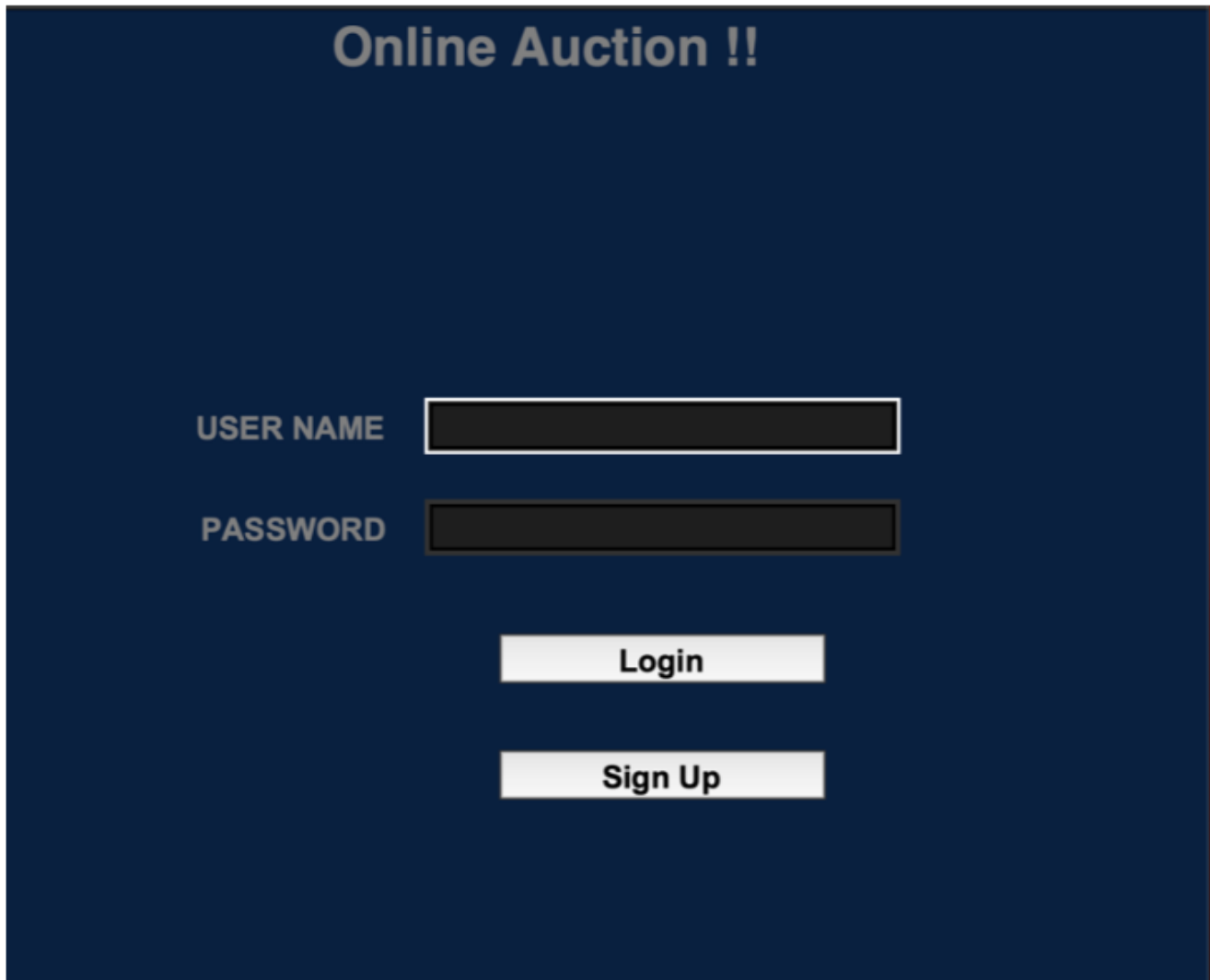
    -- Check if there is any payment associated with bids on the item
    SELECT CASE
        WHEN EXISTS (SELECT 1 FROM Payment
                     JOIN Bid ON Payment.Bid_ID = Bid.Bid_ID
                     WHERE Bid.Item_ID = item_id)
        THEN 0
        ELSE 1
    END INTO bid_status;

    RETURN bid_status;
END
```

TRIGGERS

```
• CREATE DEFINER = CURRENT_USER TRIGGER `AUCTION`.`Review_BEFORE`
  BEFORE INSERT ON `Review`
  FOR EACH ROW
  BEGIN
    -- Check if a review already exists for the same Bid_ID
    IF EXISTS (
        SELECT 1 FROM Review WHERE Bid_ID = NEW.Bid_ID
    ) THEN
        -- Raise an error if a duplicate is found
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Review already exists for this item';
    END IF;
  END;
```

CHAPTER 10: FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)



The image shows a web form titled "Online Auction !!". It features two input fields: "USER NAME" and "PASSWORD". Below these fields are two buttons: "Login" and "Sign Up". The form is set against a dark blue background.

Online Auction !!

USER NAME

PASSWORD

Login

Sign Up

Online Auction !!

Username :

Password :

First Name :

Middle Name :

Last Name :

Year of Birth :

City :

Street :

State :

Pincode :

Email :

Register

Welcome to the Main Page!

Sell

Bid

Enter Item Details

Description:

Base Price:

Submit

Available Items for Bidding

Item ID	Item Description	Base Price	Bid Status
11	rolex	\$12345.00	On
13	buggati	\$200.00	Off
90	casio	\$99.00	Off
91	wwe	\$123.00	Off
92	mona_lisa painting	\$1000.00	On
93	lexus 500	\$10000.00	On

Apply for Bid

Applied Bids

Bid ID	Item ID	Description	Base Price	Your Bid	Status	Highest Bid
25	13	buggati	200.00	400.00	Not Won	900.00
28	13	buggati	200.00	550.00	Not Won	900.00
32	13	buggati	200.00	800.00	Not Won	900.00
34	90	casio	99.00	110.00	Not Won	1000.00
36	90	casio	99.00	1000.00	Won	1000.00
40	91	wwe	123.00	4000.00	Won	4000.00
41	92	mona_lisa painting	1000.00	2000.00	Not Won	99999.00
42	11	rolex	12345.00	12346.00	Won	12346.00

Preview

Enter bid for Item ID: 92

Base Price: \$1000.00

Submit

WON

Item ID: 11
Description: rolex
Base Price: \$12345.00
Bid Amount: \$12346.00
Status: Won

Payment

Want to give Review on Item?

Overview:

Description:

Submit Review

Make Payment

COD

UPI

Card

You have to pay: 12346.00

Admin Dashboard

Bid

Bidder

Bidder_Bid

Item

Payment

Review

Seller

User

User_Phone

User_ID	Username	Password	F_Name	M_Name	L_Name
1	john_doe	pass123	John	None	Doe
2	jane_smith	pass456	Jane	None	Smith
5	michael_green	pass102	Michael	None	Green
6	linda_white	pass103	Linda	None	White
7	kevin_wilson	pass104	Kevin	None	Wilson
8	sarah_clark	pass105	Sarah	None	Clark
9	jason_king	pass106	Jason	None	King
10	olivia_walker	pass107	Olivia	None	Walker
11	Tushar123	password	Tushar		Swami
12	yash99	yash	yash		sinha
13	sudhanshu123	sh	shudhanshu		sinha
15	vipul123	vipul	vipul		khandekar
16	vedant	vedant	vedant		sinha

Add

Update

Delete

Payment Counts: COD: 2 | UPI: 1

Total Users: 13

Add New Entry to User

User_ID	<input type="text"/>
Username	<input type="text"/>
Password	<input type="password"/>
F_Name	<input type="text"/>
M_Name	<input type="text"/>
L_Name	<input type="text"/>
Year_Of_Birth	<input type="text"/>
City	<input type="text"/>
Street	<input type="text"/>
State	<input type="text"/>
Pincode	<input type="text"/>
Email	<input type="text"/>
Role	<input type="text"/>

Submit

Update Entry in User

User_ID	12
Username	yash99
Password	yash
F_Name	yash
M_Name	
L_Name	sinha
Year_Of_Birth	2004
City	banglore
Street	kormanglea
State	mumbai
Pincode	12345
Email	yash@gmail.com
Role	user

CHAPTER 11: CONCLUSION

This Auction Management System project has provided an in-depth understanding of database management principles and their application in building a functional and efficient system. The project emphasized designing and implementing a robust database structure to support key auction functionalities, such as user registration, item listing, bidding, and auction closure. Through this process, we gained valuable experience in relational database design, data integrity, and the management of client-server interactions using MySQL as the backend.

In the initial stages, we developed a comprehensive database schema and utilized entity-relationship diagrams to map out the interactions between various entities, ensuring data normalization and optimized query performance. The integration of Python (Tkinter) for the frontend further highlighted the importance of seamless interaction between the application layer and the database layer, with SQL queries effectively managing data retrieval, insertion, and updates for auction transactions.

The project's primary goals were met, with the system efficiently handling core auction operations and managing user interactions with accuracy and consistency. This experience has solidified our understanding of database principles and the significance of designing efficient schemas for real-time applications.

In conclusion, the Auction Management System project has been a valuable exercise in applying DBMS principles, enhancing our knowledge in database design, query optimization, and transactional integrity, and preparing us to work on more complex database-driven applications.

CHAPTER 12: REFERENCES/BIBLIOGRAPHY

1. Books and Academic Articles

- Sommerville, I. (2011). *Software Engineering* (9th Edition). Addison-Wesley. A comprehensive guide to software engineering principles and methodologies, covering requirements analysis, design, development, and testing in software projects.
- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th Edition). McGraw-Hill.
This text explores practical approaches to software engineering, including lifecycle models, project planning, and quality assurance processes.
- Date, C. J., & Darwen, H. (2006). *Databases, Types, and the Relational Model* (3rd Edition). Addison-Wesley.
A foundational text on database design and the relational model, essential for understanding database concepts relevant to systems like Auction Management.

2. Official Documentation and Standards

- Tkinter Documentation. Python Software Foundation. Available at: <https://docs.python.org/3/library/tkinter.html>
This documentation details the Tkinter library's GUI functionalities, components, and customization options used in Python applications.
- MySQL Documentation. Oracle Corporation. Available at: <https://dev.mysql.com/doc/>
Comprehensive reference for MySQL database management, with specifics on query optimization, database schema design, and SQL commands.
- SQL Reference Manual, W3Schools. Available at: <https://www.w3schools.com/sql/>
An introductory SQL guide covering essential SQL commands, functions, and syntax. Useful for database management and integration in backend systems.
- Git Version Control Documentation. Git. Available at: <https://git-scm.com/doc>
Detailed documentation on Git version control, covering repository management, branching, merging, and collaboration tools essential for multi-developer environments.
- Canva User Guides, Canva, Inc. Available at: <https://www.canva.com/learn/>
This guide provides information on creating design and diagramming assets, useful for generating flowcharts and interface designs in software documentation.
- Figma Documentation, Figma, Inc. Available at: <https://help.figma.com/>
A detailed resource for using Figma for prototyping and creating user interfaces, utilized for the interface design of the Auction Management System.

3. Software and Tools Used

- Visual Studio Code. Microsoft Corporation. Available at:

<https://code.visualstudio.com/>

Visual Studio Code (VS Code) is a source-code editor with features including syntax highlighting, debugging, Git integration, and extensions support, facilitating development in Python and SQL.

- MySQL Workbench. Oracle Corporation. Available at:

<https://www.mysql.com/products/workbench/>

MySQL Workbench is a tool for database administration, design, and management, providing a GUI interface for SQL query execution, schema design, and performance tuning.

CHAPTER 13: APPENDIX A: DEFINITIONS, ACRONYMS & ABBREVIATION

Definitions

- **Auction:** A competitive process in which participants place bids on an item, with the highest bid securing the item once the auction ends.
- **User Authentication:** A process for verifying the identity of users based on their credentials (e.g., username and password) to ensure secure access to the system.
- **Bid:** A financial offer made by a participant to purchase an item in an auction. Bids are usually competitive, with each bid potentially increasing the item's final price.
- **Backend:** The part of a software application responsible for data processing, business logic, and database interactions. It operates behind the scenes to manage the data flow and responses for the frontend.
- **Frontend:** The part of a software application that the user interacts with directly. This includes graphical interfaces, forms, and display elements created using Tkinter.
- **Database:** A structured collection of data stored electronically, managed by a DBMS like MySQL in this project, enabling CRUD (Create, Read, Update, Delete) operations.

Acronyms

- **SQL:** Structured Query Language - A domain-specific language used for managing and querying data in relational databases.
- **GUI:** Graphical User Interface - A visual way for users to interact with electronic devices through graphical elements like icons and buttons.
- **DBMS:** Database Management System - Software that provides a systematic way to create, retrieve, update, and manage data in databases.

Abbreviations

- **ERD:** Entity-Relationship Diagram - A graphical representation of entities and their relationships, used in database modelling.
- **IDE:** Integrated Development Environment - A software suite that provides tools for coding, debugging, and testing applications (e.g., VS Code for Python development).
- **CRUD:** Create, Read, Update, Delete - The four basic operations used in databases to manage records.
- **PPT:** PowerPoint - A presentation software often used to prepare and display project-related information.
- **OS:** Operating System - The software that supports a computer's basic functions, such as task scheduling, application management, and device control.

CHAPTER 14: GITHUB REPOSITORY LINK

1. NAME: YASH SINHA
SRN: PES2UG22CS675

GITHUB LINK

[**https://github.com/sinhayash3011/AUCTION-MANAGEMENT-SYSTEM**](https://github.com/sinhayash3011/AUCTION-MANAGEMENT-SYSTEM)

2. NAME: TUSHAR SWAMI
SRN: PES2UG22CS633

GITHUB LINK

[**https://github.com/tusharswmi121**](https://github.com/tusharswmi121)

END

OF

REPORT