Online Auction System

A database management system to manage bidding, auctions, and sales of goods and services on an online platform. This is a class project. The steps that I followed to design the database are included below.

Step 1: Identifying appropriate entities, relationships, attributes, domains, and keys

1. **User**
    - **UserID**: Integer, *Primary Key*
    - **Email**: String, Unique, Not Null
    - **Password**: String (Must have at least 8 letters or numbers), Not Null
    - **PhoneNumber**: String, Optional
    - **UserType**: Enum ('Buyer', 'Seller', 'Admin'), Not Null

2. **Item**
    - **ItemID**: Integer, *Primary Key*
    - **SellerID**: Integer, Not Null
    - **ItemName**: String, Not Null
    - **StartingBid**: Decimal, Not Null

3. **Auction**
    - **AuctionID**: Integer, *Primary Key*
    - **ItemID**: Integer, Not Null
    - **StartDate**: DateTime, Not Null
    - **EndDate**: DateTime, Not Null
    - **Status**: Enum ('Upcoming', 'Ongoing', 'Completed'), Not Null

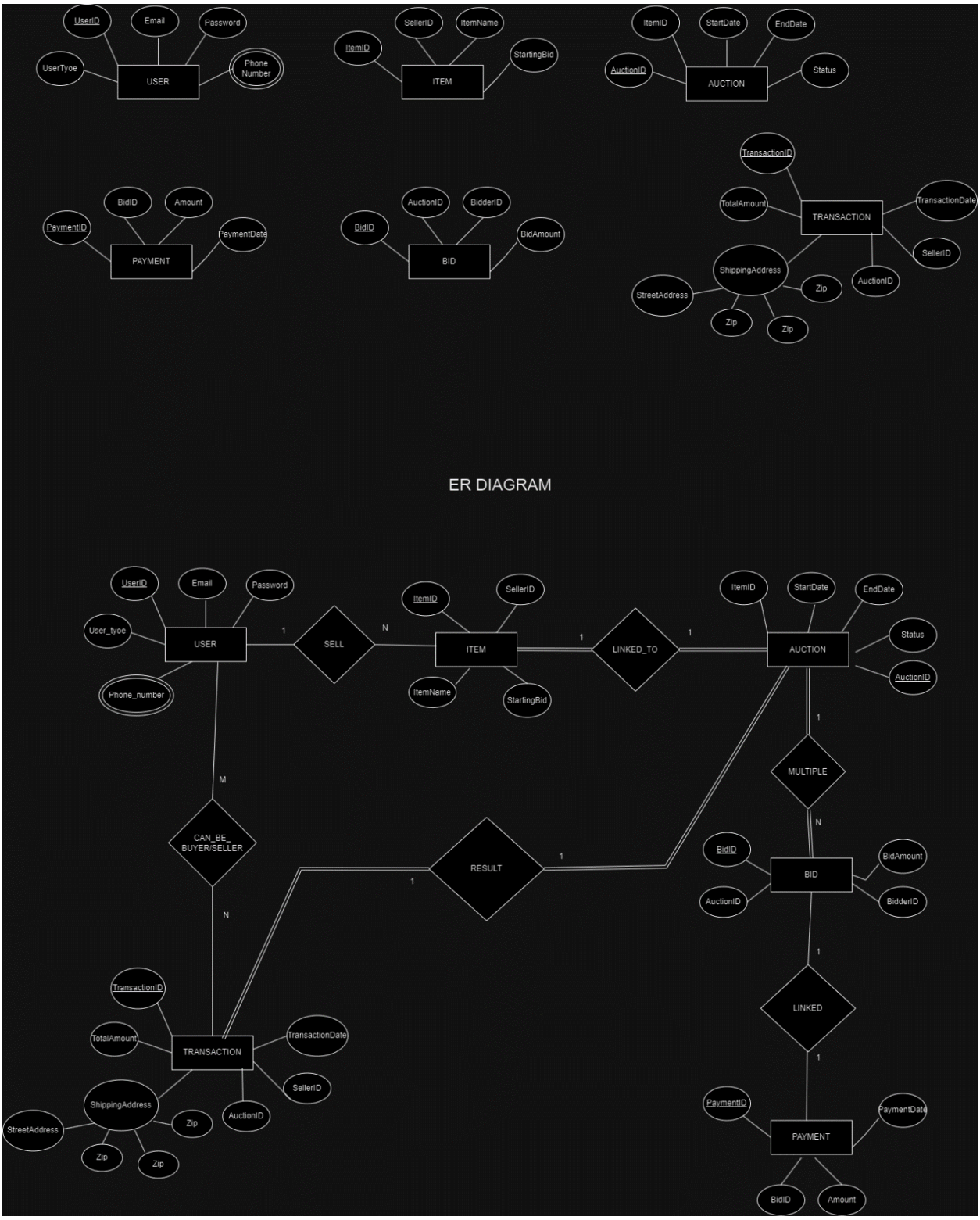
4. **Bid**
    - **BidID**: Integer, *Primary Key*
    - **AuctionID**: Integer, Not Null
    - **BidderID**: Integer, Not Null
    - **BidAmount**: Decimal, Not Null

5. **Payment**
    - **PaymentID**: Integer, *Primary Key*
    - **BidID**: Integer, Not Null
    - **Amount**: Decimal, Not Null
    - **PaymentDate**: DateTime, Not Null
6. **Transaction**
    - **TransactionID**: Integer, *Primary Key*
    - **AuctionID**: Integer, Not Null
    - **BuyerID**: Integer, Not Null
    - **SellerID**: Integer, Not Null
    - **TransactionDate**: DateTime, Not Null
    - **TotalAmount**: Decimal, Not Null
    - **ShippingAddress**: String, Not Null

Relationships

- **User - Item**: One-to-Many. A user can sell many items, but each item is sold by one user.
- **Item - Auction**: One-to-One. Each item is linked to one auction, and each auction is for one item.
- **Auction - Bid**: One-to-Many. An auction can have multiple bids, but each bid is linked to one auction.
- **Bid - Payment**: One-to-One. Each bid can lead to one payment, and each payment is linked to one bid.
- **Auction - Transaction**: One-to-One. Each auction can result in one transaction, and each transaction is linked to one auction.
- **User - Transaction**: Many-to-Many. Users can be buyers or sellers in multiple transactions.

Step 2: Producing the ER diagram



Description

Entities: I chose Users, Items, Auctions, Bids, Payments, and Transactions as entities because they represent the main components of an online auction system. Users participate in auctions, either as buyers or sellers and Items are the products being auctioned. Auctions manage the process of bidding for these items. Bids capture the offers made by users, and Payments record the transactions from successful bids. Transactions finalize the sale of an item after an auction ends.

Attributes: Attributes like Email and Password are necessary for identifying and securing user accounts, while attributes like ItemName and StartingBid for Items describe the product being auctioned. Auction-related attributes like StartDate and EndDate define the time of the auction. BidAmount records the value of each bid, and PaymentDate is for when a payment was made. Transaction related attributes like TransactionDate and TotalAmount provide details about the final sale.

Keys: Primary keys like UserID and ItemID uniquely identify records in their respective tables, ensuring data integrity. This structure helps keep the database organized and allows for efficient data retrieval and management.

Relationships: Users can place multiple bids in different auctions, and they can also list multiple items for auction. Each item is associated with one auction, and each auction is for one specific item. An auction can receive multiple bids from various users. Each bid can result in a payment if it wins the auction. Each auction can result in a single transaction that records the sale of the item to the highest bidder, with the buyer and seller involved in the transaction.

Step 3: Mapping the ER diagram and normalizing the database

To map the ER diagram to a relational database model, I converted each entity from the ERD into a table.

1) **User Table**
    a. **Columns**: UserID (Primary Key), Email, Password, PhoneNumber, UserType
    b. **Explanation**: The User entity from the ER diagram is mapped directly to the User table. UserID is the primary key.
2) **Item Table**
    a. **Columns**: ItemID (Primary Key), SellerID (Foreign Key referencing UserID), ItemName, StartingBid
    b. **Explanation**: The Item entity is mapped to the Item table. The SellerID column is a foreign key referencing UserID in the User table, establishing the relationship between User and Item.
3) **Auction Table**
    a. **Columns**: AuctionID (Primary Key), ItemID (Foreign Key referencing ItemID), StartDate, EndDate, Status
    b. **Explanation**: The Auction entity is mapped to the Auction table. The ItemID column is a foreign key that links each auction to a specific item.

4) **Bid Table**

    a. **Columns**: BidID (Primary Key), AuctionID (Foreign Key referencing AuctionID), BidderID (Foreign Key referencing UserID), BidAmount

    b. **Explanation**: The Bid entity is mapped to the Bid table. It includes foreign keys to both the Auction and User tables to track which user placed a bid and which auction the bid belongs to.

5) **Payment Table**

    a. **Columns**: PaymentID (Primary Key), BidID (Foreign Key referencing BidID), Amount, PaymentDate

    b. **Explanation**: The Payment entity is mapped to the Payment table. The BidID column is a foreign key linking each payment to the corresponding bid.

6) **Transaction Table**

    a. **Columns**: TransactionID (Primary Key), AuctionID (Foreign Key referencing AuctionID), BuyerID (Foreign Key referencing UserID), SellerID (Foreign Key referencing UserID), TransactionDate, TotalAmount, ShippingAddress

    b. **Explanation**: The Transaction entity is mapped to the Transaction table. It includes foreign keys to link transactions to both the auction and the users involved (buyer and seller).

Normalization

To ensure the database is normalized, I checked that each table meets the requirements for the Third Normal Form

1) **First Normal Form**

    a. Each table has a primary key.

    b. All entries in each column are not repeating groups.

2) **Second Normal Form**

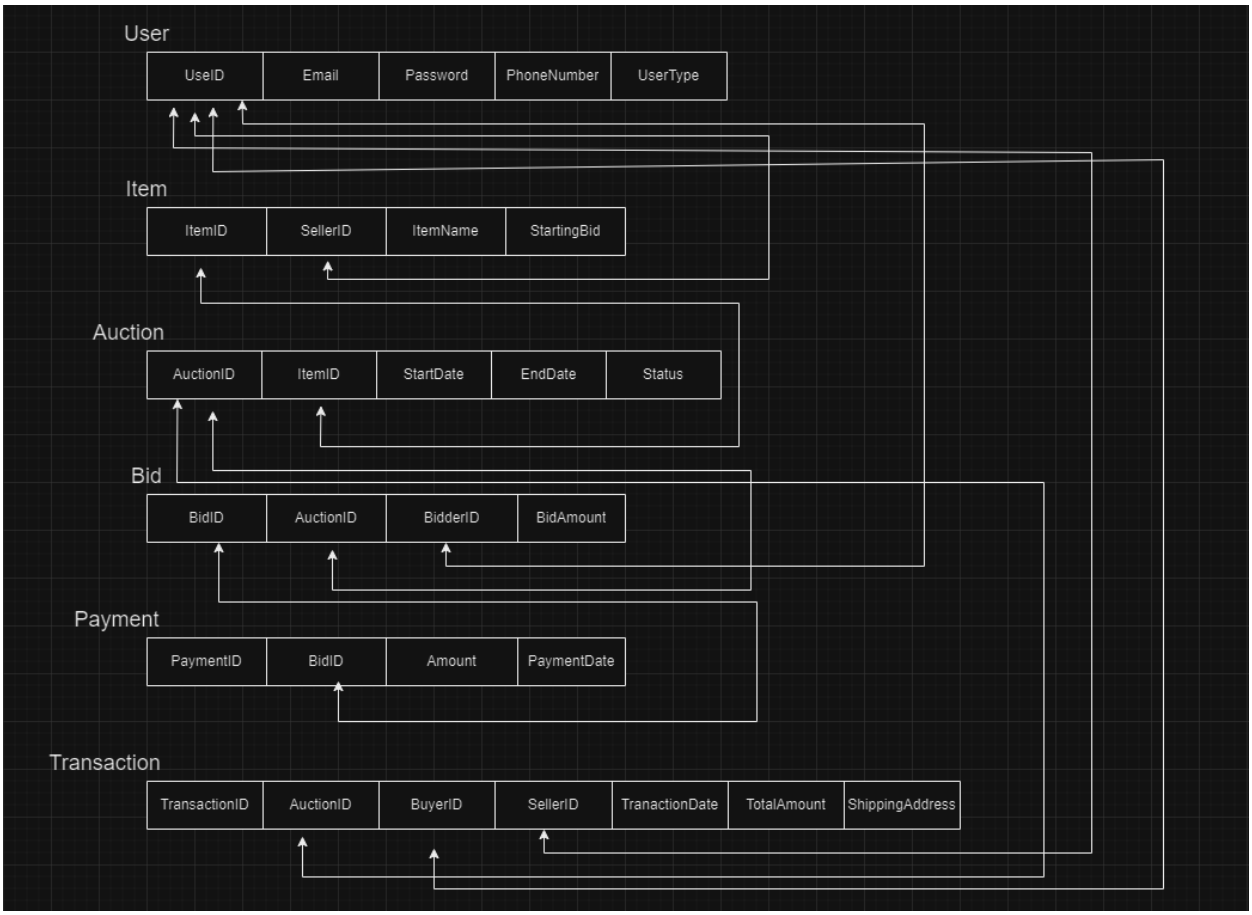
    a. The tables meet 1NF.

    b. All non-key attributes are fully functionally dependent on the primary key.

3) **Third Normal Form**

    a. The tables meet 2NF.

    b. Non-key attributes do not depend on other non-key attributes).



Constraints

1) **User Table**
    a. **Primary Key**: UserID
    b. **Unique Constraints**: Email
    c. **Not Null Constraints**: Email, Password, UserType
    d. **Foreign Keys**: N/A
2) **Item Table**
    a. **Primary Key**: ItemID
    b. **Foreign Key**: SellerID references UserID in User
    c. **Not Null Constraints**: SellerID, ItemName, StartingBid
3) **Auction Table**
    a. **Primary Key**: AuctionID
    b. **Foreign Key**: ItemID references ItemID in Item

c. **Not Null Constraints**: ItemID, StartDate, EndDate, Status

4) **Bid Table**

   a. **Primary Key**: BidID

   b. **Foreign Keys**: AuctionID references AuctionID in Auction, BidderID references UserID in User

   c. **Not Null Constraints**: AuctionID, BidderID, BidAmount

5) **Payment Table**

   a. **Primary Key**: PaymentID

   b. **Foreign Key**: BidID references BidID in Bid

   c. **Not Null Constraints**: BidID, Amount, PaymentDate

6) **Transaction Table**

   a. **Primary Key**: TransactionID

   b. **Foreign Keys**: AuctionID references AuctionID in Auction, BuyerID references UserID in User, SellerID references UserID in User

   c. **Not Null Constraints**: AuctionID, BuyerID, SellerID, TransactionDate, TotalAmount, ShippingAddress

Step 4: Defining the database in MySQL and inserting tuples

The .sql file is included in the github repository

The benefits of this design are clear and straightforward relationships between users, items, and auctions, which keep the data organized and easy to manage. By setting up these connections, I made sure the system maintains data integrity and avoids errors. In Step 3, I also made sure the database is normalized to Third Normal Form (3NF), which helps reduce redundancy and keeps the data clean. This setup allows for efficient data retrieval and supports complex queries. The design is also scalable, so it's easy to add new features or expand the system in the future. Overall, this approach makes sure the system is reliable, accurate, and flexible.