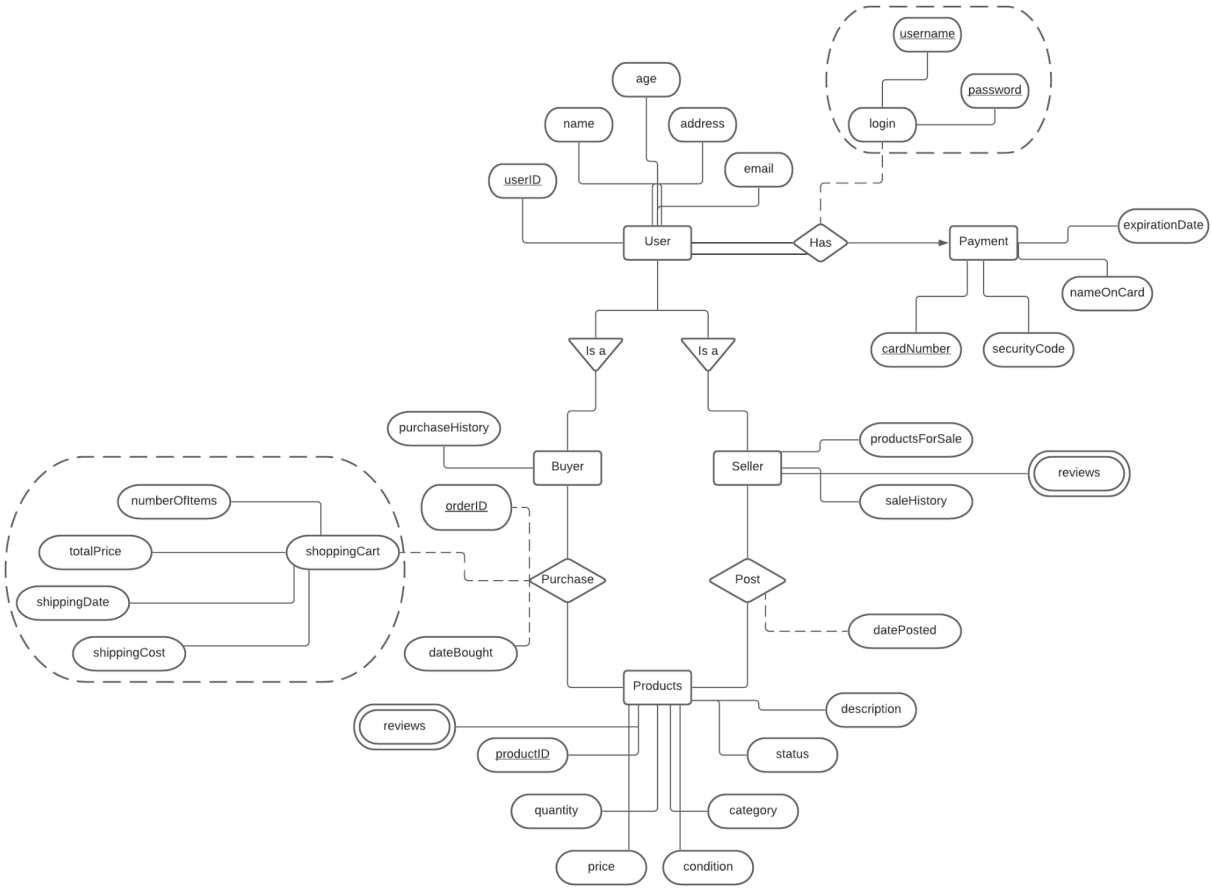
# Database Final Deliverable - TheMarket

Yuvraj Sreepathi (ys3mnq), Samara Elahi (se4hy), Vatsal Rami (vr9sjk), Mark Chitre (mnc8rj)

## Database Design

### ER Diagram




### Schema Statements

account_info(<u>username</u>, password)

payment(<u>cardNumber</u>, securityCode, nameOnCard, expirationDate)

post(<u>userID</u>, productID, datePosted)

products(<u>productID</u>, quantity, price, category, status)

product_review(<u>productID</u>, review)

purchase(orderID, dateBought, numberOfItems, totalPrice, shippingDate, shippingCost)

purchase_history(userID, productBought)

sellerHistory(userID, saleHistory)

sellerSelling(userID, productsForSale)

seller_review(userID, review)

user_account(userID, username, cardNumber)

user_basicInfo(userID, email)

user_info(email, age, name, address, cardNumber)

## Database Programming

We hosted both the database and the application locally. To run the application locally there are a set of steps that have to be followed so that the database can function correctly and properly interact with the application interface. To deploy and run the application, the user has to start the Apache Web Server and the MySQL database on the XAMPP Control Panel. Additionally, the source code for the web application has to be added to the 'C:\xampp\htdocs' path under a new folder of the user's choice. After these steps are completed, the application can be accessed by typing the following in the search bar of a web browser 'localhost/*name of folder in htdocs*/'. This should take the user to the login screen after which the user will be able to access the website.

Based on the implementation of our web application, we attempted to create an instance of a trigger to insert into our user_account table, after a record has been inserted into the account_info table. The purpose of this trigger was to insert the newly created username in the user_account table and have an associated userID. This way, in the future, when we needed information about the current user, we would only have to use one table and avoid the need to cross-product or natural join in our queries.

## Database Level Security

Security at the database level is achieved through access control. All of our group members (admins) have special access privileges that allow us to edit/change any and all of the databases. Admins have the ability to grant privileges, shutdown, show, lock databases and create new users, in other words, they have all privileges. On the other hand, end-users can do a subset of the actions, the SQL command below lists which ones.

SQL Commands:

End User:

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, FILE, INDEX, ALTER, CREATE TEMPORARY TABLES, CREATE VIEW, EVENT, TRIGGER, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE ON *.* TO 'bevan'@'localhost' REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;

Admin:

GRANT ALL PRIVILEGES ON *.* TO 'vatsal'@'localhost' REQUIRE NONE WITH GRANT OPTION MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;

## Application Level Security

Security at the application level is ensured by the use of passwords, each account is associated with a unique username/email and password. Until the correct password is matched with each username, the account and its data will not be accessed. In addition, we have utilized password hashing. We use password hashing to encrypt the password, in the event that a user has unauthorized access to the database, they will not be able to see the password directly. They will see an encrypted version of the password.

We also tested our application by attempting SQL injection attacks when logging in. All of our attempts at an SQL injection attack failed, ensuring we have sufficient security for our database.

## SIGN UP

```php
// Validate username
if(empty(trim($_POST["username"]))){
    $username_err = "Please enter a username.";
} elseif(!preg_match('/^[a-zA-Z0-9_]+$/', trim($_POST["username"]))){
    $username_err = "Username can only contain letters, numbers, and underscores.";
} else{
    // Prepare a select statement
    $sql = "SELECT username FROM account_info WHERE username = ?";

    if($stmt = $con->prepare($sql)){
        // Bind variables to the prepared statement as parameters
        $stmt->bind_param("s", $param_username);

        // Set parameters
        $param_username = trim($_POST["username"]);

        // Attempt to execute the prepared statement
        if($stmt->execute()){
            // store result
            $stmt->store_result();

            if($stmt->num_rows == 1){
                $username_err = "This username is already taken.";
            } else{
                $username = trim($_POST["username"]);
            }
        } else{
            echo "Oops! Something went wrong. Please try again later.";
        }

        // Close statement
        $stmt->close();
    }
}
```

## LOGIN

```php
// Check if username is empty
if(empty(trim($_POST["username"]))){
    $username_err = "Please enter username.";
} else{
    $username = trim($_POST["username"]);
}

// Check if password is empty
if(empty(trim($_POST["password"]))){
    $password_err = "Please enter your password.";
} else{
    $password = trim($_POST["password"]);
}

// Validate credentials
if(empty($username_err) && empty($password_err)){
    // Prepare a select statement
    $sql = "SELECT username, `password` FROM account_info WHERE username = ?";
    if($stmt = $con->prepare($sql)){
        // Bind variables to the prepared statement as parameters
        $stmt->bind_param("s", $param_username);
        // Set parameters
        $param_username = $username;

        // Attempt to execute the prepared statement
        if($stmt->execute()){
            // Store result
            $stmt->store_result();
            // Check if username exists, if yes then verify password
            if($stmt->num_rows == 1){
                // Bind result variables
                $stmt->bind_result($username, $hashed_password);
                if($stmt->fetch()){
                    if(password_verify($password, $hashed_password)){
                        // Password is correct, so start a new session
                        session_start();

                        // Store data in session variables
                        $_SESSION["loggedin"] = true;
                        $_SESSION["username"] = $username;
                        $_SESSION["password"] = $hashed_password;

                        // Redirect user to welcome page
                        header("location: account.php");
                    } else{
                        // Password is not valid, display a generic error message
                        $login_err = "Invalid username or password.";
                    }
                }
            } else{
                // Username doesn't exist, display a generic error message
                $login_err = "Invalid username or password.";
            }
        } else{
            echo "Oops! Something went wrong. Please try again later.";
```