**Team JET**
**Deliverable 2: The Case Study**

## Account (register, login, logout) [4] (Emmanuel)

**Card:** a user can register into JETrading
**Conversation:** the user will enter user name, password and their email address in the register form. The email address will be used to recover the user's account if he forgets his username or password.
**Conformity:** Once the user registered his account, we will use an SQL query to find his account in MySQL.
IMPORTANT: the user name cannot be a stock symbol. This will cause problem in the search box.
**Implementation:** **https://www.w3schools.com/php/php_forms.asp**
INSERT INTO user VALUES (username, password_hash, email);

**Card:** A user can press "forgot account?", the user will then receive an email to his email that is associated with his account.
**Conversation:** the email will contain a link that direct the user to the registered form that have his user name. The user will then reset his password.
**Conformity:** Once the user reset his password, we will use an SQL query to find his account in MySQL and then see if the password hash is different.
**Implementation:** Send Emails using PHP and Gmail.
https://www.youtube.com/watch?v=U13smZvdArI
SELECT email from user where email == $email;

**Card:** a user can login into JETrading
**Conversation:** the user will enter user name and password in the login form.
**Conformity:** Once the user login into his account, he will be able to access his own account information such as his wish list and current holding.
**Implementation:** **https://www.w3schools.com/php/php_forms.asp**

```
SELECT * from user
WHERE User_name       == $User_name
AND      password_hash == $password;
```

**Card:** a user can logout of JETrading
**Conversation:** the user will be logout if he clicked the logout button or if he stay on the page for more than 5 minutes. This ensures security.
**Conformity:** Once the user login into his account, he will be able to access his own account information such as his wish list and current holding.
**Implementation:** This is similar to session in ASP.Net
**https://stackoverflow.com/questions/10097887/using-sessions-session-variables-in-a-php-login-script**

```
session_destroy();
SELECT login_token from User
WHERE login_token is null
    AND user_id = SESSION["user_id"];
```

## User Profile Information (4) (Emmanuel)

**Card:** User can add his profile information.
**Conversation:** The profile information are: biography and privacy flag.
**Conformity:** We will use the query below and compare the user profile information.
**Implementation:**

```
SELECT * FROM user WHERE user_id = $_SESSION["user_id"]
```

**Card:** User can update his profile information.
**Conversation:** The profile information are: biography and privacy flag.
**Conformity:** We will use the query below and compare the user profile information.
**Implementation:**

```
SELECT * FROM user WHERE user_id = $_SESSION["user_id"]
```

**Card:** User can set his profile as private.
**Conversation:** When a user go to a private user profile page, he will only see the user name, follow button and message button. The user will also have to approve follower's request.
**Conformity:** We will use a private account and check whether the above statement is true.
**Implementation:**

```
SELECT * FROM user
WHERE   user_id = $_SESSION["user_id"]
AND privacy_flag = 1;
```

**Card:** A user can set his profile as public.
**Conversation:** When a user go to a public user profile page, he will see the user current holdings and his total return on investment. The follower's request is automatically approved.
**Conformity:** We will use a public account and check whether the above statement is true.
**Implementation:**

```
SELECT * FROM user
```

WHERE user_id = $_SESSION[“user_id”]
AND privacy_flag = 0;

**User interaction (follow) [6] (Botao)**

**Card:** A user can search for other users on the search textbox.
**Conversation:** Once a user is found, the user profile information will be displayed according to its privacy settings.
**Conformity:** We will use a query below to search for a user and compare the result from the search textbox.
**Implementation:**
$Search = data from the search textbox
SELECT * FROM user WHERE user_name = $SEARCH

**Card:** A user can follow another user after he successfully search a user.
**Conversation:** The followed user will then be added on the user’s followed list and on the other user’s follower list.
**Conformity:** We will use query below and compare it to the two user follower and following list.
**Implementation:**
$Search = data from the search textbox
SELECT user_id FROM user WHERE user_name = $SEARCH
//For the login user’s follower list
SELECT * FROM user_relationship WHERE follower_id = user_id
AND following_id = $_SESSION[“user_id”]

**Card:** A user can view his follower list.
**Conversation:** The ‘follower list’ will contain a list of users and their information. Their information will be shown depending on privacy settings.
**Conformity:**

1. We will use the query above and compare it to the user’s follower list.
2. We will make sure that the user’s privacy is respected by using the query below and compared to the user’s follower list.

**Implementation:**
SELECT * FROM user WHERE user_id = $follower_id;

**Card:** A user can view his following list.
**Conversation:** The ‘following list’ will contain a list of users and their information. Their information will be shown depending on privacy settings.
**Conformity:**

1. We will use the query above and compare it to the user’s following list.
2. We will make sure that the user’s privacy is respected by using the query below and compared to the user’s following list.

**Implementation:**

SELECT * FROM user WHERE user_id = $following_id;

**Card:** A user can view his approval list, this only applies if the user is private.
**Conversation:** The 'approval list' will contain a list of users and their information. Their information will be shown depending on privacy settings.
**Conformity:**

3. We will use the query above and compare it to the user's 'approval list' .
4. We will make sure that the user's privacy is respected by using the query below and compared to the user's 'approval list' .

**Implementation:**
SELECT * FROM user WHERE user_id = $following_id;

**Card:** A user can unfollow a user that he is currently following.
**Conversation:** This can be done from the following list and also from the user profile information page.
**Conformity:** We will use the query below and compare it the following list.
**Implementation:**
//this query should return null
SELECT * FROM user_relationship WHERE follower_id = user_id
AND following_id = $_SESSION["user_id"]

## User interaction (message) [8] (Emmanuel)

**Card:** A user can write a message by pressing the message button on another user profile info page.
**Conversation:** A UI will then appear and the user can enter the message there.
**Conformity:** A UI will appear directly on the user profile page. The UI will have a message box, a send button and a discard button.
**Implementation:**
-- select the user you want to send a message to
SELECT * FROM user WHERE user_id = receiver_id;

```
<form action="all_form.php" method="get" id="messageUI">
Message: <input type="text" name="message"><br>
<input type="submit">
<button type="button" onClick="removeMessageUI">discard</button>
</form>
```

**Card:** A user can send a message by pressing the submit button
**Conversation:** Empty message will not be sent. A time stamp will also be added at the beginning of the message.
**Conformity:** We will use a query to view the message in MySQL and compare to the message that is sent.
**Implementation:**

```
INSERT INTO ‘message’ (‘message_text’, ‘timestamp’)
        VALUES();
```

**Card:** A user can discard a message by pressing the discard button
**Conversation:** the messageUI will disappear
**Conformity:** the messageUI will disappear
**Implementation:**

```
SELECT * FROM messages WHERE sender_id = $_SESSION[“user_id”]; //the message text in the messageUI should not appear

//In javascript
Function removeMessageUI(){
   document.getElementById(“messageUI”).style.display = gone;
}
```

**Card:** A user can view all their messages.
**Conversation:** In the message list page, there will be a list of messages that the user send and receive. There will also be a reply button and a view message beside the message.
**Conformity:** We will use the query below and compare it to the message list page.
**Implementation:**

```
SELECT * FROM message WHERE sender_id     = $_SESSION[“user_id”]
                                OR  receiever_id = $_SESSION[“user_id”]
```

**Card:** A user can view a single message.
**Conversation:** In the message page, there will be a list of parent message and the message itself.
**Conformity:** We will use the query below and compare it to the message page.
**Implementation:**

```
Do{
SELECT parent_message_id FROM message
                            WHERE sender_id     = $_SESSION[“user_id”]
                                AND message_id = $message_id;
}while ($parent_message_id != null);
```

**Card:** A user can reply to a message.
**Conversation:** In the message list page, the user will click on a message that he wants to reply to.
**Conformity:** We will use the query below and compare it to the message list page.
**Implementation:**

```
SELECT * FROM message WHERE sender_id     = $_SESSION[“user_id”]
                                AND parent_message_id = $message_id;
```

**Card:** A user can block someone from messaging him.

**Conversation:** In the block list, the user can add a user that he wishes to block.
**Conformity:**

1. We will use the query below and compare it to the block list page.
2. We will use try to message the user using the blocked user's account. The message will simply not be added on to the database.

**Implementation:**

```
SELECT * FROM user_relationship
WHERE follower_id = $_SESSION["user_id"]
AND          blocked = 1;
```

**Card:** A user can unblock someone from messaging him.
**Conversation:** In the block list, the user can unblock a user.
**Conformity:**

1. We will use the query below and compare it to the block list page.
2. We will use try to message the user using the blocked user's account. The message should be added on to the database.

**Implementation:**

```
SELECT * FROM user_relationship
WHERE follower_id = $_SESSION["user_id"]
AND          blocked = 0;
```

**Stock (Wishlist, current_holding, transaction) [9] (Botao)**

- **Wishlist:** **stock_symbol, current_price.**
- **Current_holding: stock_symbol, current_price, bought_price**
- **Transaction:** **stock_symbol, current_price, bought_price, sold_price**

**Since these 3 tables are very similar, they can be in the same table. We can distinguish between them using default values. If stock.bough_price = 0, it is wishlist. If stock.sold_price = 0, it is current_holding. If all fields of stock is specified, then it is transaction.**

- **stock: stock_symbol, current_price, bought_price, sold_price**

**Card:** A user can search for a stock.
**Conversation:** The stocks and all their information will not be stored on the database, we will get their information through alpha vantage API. After the user enters the stock symbol, a query will be made to this API.
**Conformity:** If the user enter the correct stock symbol, the stock information page should appear.
**Implementation:**

```
SELECT * FROM stock where stock_id = $stock_id;
```

```
<form action="all_form.php" method="get">
Stock Symbol: <input type="text" name="stock_symbol"><br>
<input type="submit">
</form>
//In php form
$_GET["stock_symbol"] // use this info for api query
```

**Card:** The user can view the stock information page.
**Conversation:** On top of the stock information, there will also be buy button and add to wishlist button.
**Conformity:** When the user search for a stock with correct stock symbol, the stock data will populate the stock information page.
**Implementation:**
SELECT * FROM stock WHERE stock_id = $stock_id;

**Card:** The user can know more about a technical information by hovering a technical information.
**Conversation:** For example when the user hover over “PE ratio”, a tiny window will states “valuing a company that measures its current share price relative to its per-share earnings (EPS)”
**Conformity:** We will use the query below to compare to the information shown in the tiny window.
**Implementation:**
SELECT description FROM technical_info WHERE technical_name = $technical_name;

**Card:** A user can view the stock in his wishlist.
**Conversation:** All the stock on his wishlist will be stored on the database.
**Conformity:** We will login into a user account, and then make a comparison between the wishlist page and an SQL query on his wishlist in MySQL.
**Implementation:**
SELECT user_id FROM user WHERE user_name = $_SESSION[‘user_name”]
SELECT * FROM stock WHERE user_id = user_id AND bought_price = 0;

**Card:** A user can add the stock in his wishlist.
**Conversation:** Once this happens, the stock will be added to the database.
**Conformity:** We will login into a user account, and then make a comparison between the wishlist page and an SQL query on the stock table in MySQL.
**Implementation:**
SELECT * FROM ‘stock_wishlist WHERE user_id = $_SESSION[“user_id”];

**Card:** A user can remove the stock in his wishlist.
**Conversation:** Once this happens, the stock will be removed from the database too.

**Conformity:** We will login into a user account, and then make a comparison between the wishlist page and an SQL query on the stock table in MySQL.
**Implementation:**
Get $stock_id from the wishlist page.
DELETE FROM stock WHERE stock_id = $stock_id AND bought_price = 0;

**Card:** A user can view stocks that he bought on current holding page.
**Conversation:** this page will be populated from the stock table in MySQL.
**Conformity:** We will use the query below and compare it to the current holding page.
**Implementation:**
Get $stock_id from the wishlist page.
SELECT * FROM stock WHERE stock_id = $stock_id AND bought_price != 0;

**Card:** A user can buy stocks on the stock information page or from wishlist.
**Conversation:** Once this happens, the buy stock UI will appear directly on the page. The UI will allow the user to specify the number of stock he wishes to purchase. If the user can afford it the below statements will happen.

- The bought price will be added on stock table in MySQL.
- The stock will also be added on current holding page.

**Conformity:** We will make a query on the stock table to make sure the bought price is specify.
**Implementation:**
Get $stock_id from the wishlist page.
SELECT* FROM stock WHERE stock_id = $stock_id AND bought_price != 0;

**Card:** A user can sell the stocks that he bought on current holding page.
**Conversation:** Once this happens, the stock will appear on transaction page instead of the current holding page.
**Conformity:** We will use the query below and compare it to the transaction page.
**Implementation:**
Get $stock_id from the wishlist page.
SELECT * FROM stock WHERE stock_id = $stock_id AND bought_price != 0 AND sold_price != 0;

## Forum(comments, likes, unlikes) [15] (Timothy)

**Card:** A user can add a comment in the comment section on the stock information page.
**Conversation:** The user will specify the comments. The timestamp will be added at the bottom of his comment.
The SQL query below will be used to compare to the comment that the user posted on stock information page
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE user_id = $_SESSION[“user_id”];

**Card:** A user can delete a comment in the comment section on the stock information page.
**Conversation:** The comment will be deleted from the page.
The SQL query below will be used to compare to the comment that the comment has been deleted on stock information page.
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE user_id = $_SESSION[“user_id”];

**Card:** A user can tag his comment by typing #.
**Conversation:** Two of the most popular tags will be “Bullish” and “Bearish”. Therefore, these tags will be suggested on the add comment UI.
**Conformity:** The SQL query below will be used to compare to the tags on a comment.
**Implementation:**
SELECT * FROM ‘tags’ WHERE comment_id = $comment_id;

**Card:** The user can use any of the technical information as tag.
**Conversation:** This tag will then behave the same way as the technical information on the stock information page.
**Conformity:** The comment will be inserted into the table and the tags in the comment will be available to view.
**Implementation:**
INSERT INTO comments(user_id, comment_id, comment)
VALUES();

SELECT * FROM tags WHERE tag_name = $tag_name;

**Card:** A user can view the comment section on the stock information page.
**Conversation:** Both public and private users comments will be shown.
**Conformity:** The SQL query below will be used to compare to the comments on stock information page
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE stock_symbol = $stock_symbol;

**Card:** A user can view a comment from a private user.
**Conversation:** The user will only see the user name and comment of the private user.
**Conformity:** The SQL query below will be used to compare to the comments on stock information page
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE stock_symbol = $stock_symbol;

**Card:** A user can view a comment from a public user.
**Conversation:** The user will these information: user name, comment, the current holding of this stock.

**Conformity:** The SQL query below will be used to compare to the comments on stock information page
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE stock_symbol = $stock_symbol;

**Card:** A user can check the “only public user” checkbox if he just wants to see public user’s comment.
**Conversation:** This also means, all private users’ comments will be hidden.
**Conformity:** The SQL query below will be used to compare to the comments on stock information page
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE stock_symbol = $stock_symbol;

**Card:** A user can sort the comment section by the user that hold the biggest amount of shares.
**Conversation:** The “only public user” checkbox will be automatically checked. If there is no holder, then all public users’ comments will be shown.
**Conformity:** The SQL query below will be used to compare to the comments on stock information page
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE stock_symbol = $stock_symbol;

**Card:** A user can sort the comment section by the user that have the highest return on investment.
**Conversation:** The “only public user” checkbox will be automatically checked. If there is no holder, then all public users’ comments will be shown.
**Conformity:** The SQL query below will be used to compare to the comments on stock information page
**Implementation:**
SELECT * FROM ‘stock_comments’ WHERE stock_symbol = $stock_symbol;

**Card:** A user can view all the tags that are used by the comments.
**Conversation:**
**Conformity:** If there is a tag on any comment in the comment section. Then a tag will appear on top of the comment section.
**Implementation:**
SELECT comment_id FROM ‘stock_comments’ WHERE stock_symbol = $stock_symbol;
SELECT tag_name FROM ‘tags’ WHERE comment_id = $comment_id;

**Card:** The user can click on one of the tags, and the comment section will only show the comment that have that tag.
**Conversation:**
**Conformity:** The query below will be compared to the comment that have the tag.
**Implementation:**

SELECT tag_name FROM ‘tags’ WHERE comment_id = $comment_id;

**Card:** The user can like a comment.
**Conversation:** By liking a comment, the number of likes on the comment will increase. The like button will also change to unlike.
**Conformity:** The query below will be compared to the number of likes that a comment have.
**Implementation:**
SELECT number_of_likes FROM ‘stock_comments’ WHERE comment_id = $comment_id;

**Card:** The user can unlike a comment after liking it.
**Conversation:** By unliking a comment, the number of likes on the comment will decrease. The unlike button will also change.
**Conformity:** The query below will be compared to the number of likes that a comment have.
**Implementation:**
SELECT number_of_likes FROM ‘stock_comments’ WHERE comment_id = $comment_id;

**Card:** A user can sort the comment section by the comment that have the highest number of likes.
**Conversation:** If there is no likes on any comment, then a text that says “no likes on any comments”
**Conformity:** The SQL query below will be used to compare to the comments on stock information page
**Implementation:**
```
SELECT * FROM ‘stock_comments’
        WHERE stock_symbol = $stock_symbol
        ORDER BY number_of_likes ASC;
```

**Forum**
stock_id
stock_information
comments

**Comments**
user_id
comment_id
comment
privacy_flag
likes

**Likes**
user_id
comment_id

**User**
user_id
User_name
Password_hash
Email
Login_token
Biography
Privacy_flag

**User_relationship**
Follower_id
Following_id
Approved
Blocked

**Message**
Message_id
parent_message_id
Sender_id
Receiver_id
Message
Timestamp
read

**Stock**
stock_id
stock_name
stock_symbol
stock_information
stock_price

**Stock_held**
stock_id
stock_quantity
stock_price
user_id
account_id
bought_price
sold_price

**Technical_info**
Technical_info_id
Technical_name
Technical_definition

//TODO
Make Diagram

https://www.lucidchart.com/invitations/accept/66c2e3f6-801b-4c00-be31-63943585ce37