Project Greed Programmer's Guide

1. Introduction:

This guide describes the development process involved in developing Project Greed. Project Greed is a web-based stock market simulator which is intended to simulate the buying and selling of real-world stocks utilizing an external API which collects real-time stock market data. In addition to buying and selling stocks, user's of this simulator will have the opportunity to participate in user generated competitions which will allow the user to compete against other users by comparing his/her stock purchases over a specified period of time with the winner being determined by which user has accumulated the most capital over the term of the competition. Along with purchasing stocks and selling stocks the user will have the ability to monitor stock trends and make queries based on historical information.

1.1 Notational Conventions:

The following conventions are used this document:

1.1.1

User: Refers to anybody interacting within the Project Greed environment.

You: Refers to the end user, System Administrator, or Programmer using this manual as a reference.

1.1.2

Anything in **bold** indicates a new heading (topic), chapter, sub-chapter or actual source code.

2. Implementation:

This simulator utilizes several programming languages to include PHP, HTML, JavaScript, SQL, and XML. In this section we will step through each module and explain which programming languages we chose to implement and the reasoning behind the decisions made to program the specific modules.

2.1. User Account (Log in/Log out):

This module allows the user to access his/her account. For this module we primarily used HTML and PHP languages to write this code. This particular module authenticates the user by matching user input with values maintained in database. If authentication is successful, user will be redirected to his/her registered account. If authentication is unsuccessful, the user will be redirected back to the log in page and asked to re-authenticate with the proper credentials.

```
session_start();
define('DB_HOST', 'pluto.hood.edu');
define('DB_NAME', 'htun');
define('DB_USER','htun');
define('DB_PASSWORD','jeiCau1a');

$con=mysql_connect(DB_HOST,DB_USER,DB_PASSWORD) or die("Failed to connect to MySQL: " . mysql_error());
$db=mysql_select_db(DB_NAME,$con) or die("Failed to connect to MySQL: " . mysql_error());
```

This portion of the code starts the user session with session_start(); method and connects to the database using the define() method, $con=mysql_connect and $db=mysql_select_db.

In the next section of this particular module, we have decided to create the function "SignIn" to handle the actual functionality of signing into his/her account as seen here:

```
function SignIn()
{
session_start();   //starting the session for user account page
if(!empty($_POST['username']))   //checking the 'user' name which is from Sign-In.html, is it empty or have some text
{
	$query = mysql_query("SELECT *  FROM account where username = '$_POST[username]' AND user_password = '$_POST[user_password]'") or die(mysql_error());
	$row = mysql_fetch_array($query);
	if(!empty($row['username']) AND !empty($row['user_password']))
	{
		$_SESSION['username'] = $row['username'];

		header("Location: http://pluto.hood.edu/~htun/projectgreed/indexmember.php");

	}
	else
	{
		header("Location: http://pluto.hood.edu/~htun/projectgreed/index.php?loginerror");
	}
}
}
```

We used session_start method again to begin the user's account accessing session. Using two if/else statement we check the user input then query the database using $query, which checks for matching information. There are also two header functions: one which redirects to an error message and one which redirects to the user's account.

This line executes the sign in function in the form of a button called submit. The HTML code for the submit button is located in a separate PHP file named signuplogin.php as seen below:

```
if(isset($_POST['submit']))
{
      SignIn();
}
```

The log out module simply destroys the user's session using session_unset() and session_destroy() and redirects back to the homepage using the header() function as seen below

```
session_start();
session_unset();
session_destroy();
header("Location:
http://pluto.hood.edu/~htun/projectgreed/index.php");
```

2.2. User Registration:

This module allows a user to create an account. The source code was written primarily in HTML and PHP. After authenticating through the log in/sign up page the user's account is created and the user will be redirected to the homepage displaying his/her account information.

The function NewUser() takes the user's input and sets it equal to the variables that correspond to the input fields as seen below:

```
$username = $_POST['username'];
$user_password =  $_POST['user_password'];
$query = "INSERT INTO account (username,user_password) VALUES
('$username','$user_password')";

$data = mysql_query ($query)or die(mysql_error());
```

The function SignUp checks for user input and queries the database for the user's credentials. If the query succeeds it will return an error message because that username already exists. If the query fails it will the SignUp() function executes the NewUser() function:

```
function SignUp()
{
if(!empty($_POST['username']))   //checking the username which is from signuplogin.php, is it empty or have some text
{
      $query = mysql_query("SELECT * FROM account WHERE username = '$_POST[username]'") or die(mysql_error());

      if(!$row = mysql_fetch_array($query))
      {
            NewUser();
      }
      else
      {
            validation();
      }
}
}
```

2.3. User Portfolio:

This object acts as a users financial information center that is portable within the site for this simulation. A user is able to access portfolio features such as buying stock, selling stock, joining competitions and creating competitions. We used PHP as the primary source code language for this particular situation. In the code below we once again use session_start() function to invoke a new portfolio activation:

```
 $con=mysql_connect(DB_HOST,DB_USER,DB_PASSWORD) or die("Failed to connect to MySQL: " . mysql_error());
$db=mysql_select_db(DB_NAME,$con) or die("Failed to connect to MySQL: " . mysql_error());
```

As we see, our database is queried for log in credentials which is used to authenticate a specific user as we explained in section 2.1. The database then takes the user input, which in this case is a simple keystroke, and inserts a new row into the database.

```
$result = mysql_query("SELECT * FROM portfolio");
$portfolio_id = 1 + mysql_num_rows($result);
```

This new row will stand as a new user portfolio with portfolio features filled in with data from our database.

```
mysql_query("set foreign_key_checks=0;");
$query = "INSERT INTO portfolio (username,portfolio_id,capital,active)
VALUES ('$username','$portfolio_id',100000,false)";
$data = mysql_query ($query)or die(mysql_error());
header("Location:
http://pluto.hood.edu/~htun/projectgreed/indexmember.php?account"
);
$conn->close();
```

2.4. Transaction Engine:

The transaction module allows users to search for, buy, and sell stocks. The sell stock form is located on the home page called indexmember.php under "Stocks Purchased". The search and buy stocks forms are both located in the transaction page called transaction.php. There is a link to a list of all the stocks that can be used on this website which when clicked will open in a new tab:

```
<a href='stocklist.php' target="_blank">View All Stocks</a>
```

The module asks users to select the stock they want to buy and the number of shares of that stock using text in a form and uses a button to submit that input:

```
<div id="search">
<form method="POST" action="indexmember.php?transaction">
<td>Enter stock symbol</td><td> <input type="text" name="symbol"></td>
</tr>
<tr>
<td><input id="button" type="submit" name="submit" value="Select"></td>
```

The functionality for the button when clicked is shown below. :

```
if(isset($_POST['submit']))
{

$conn = new mysqli("pluto.hood.edu", "htun", "jeiCau1a", "htun");
if ($conn->connect_errno) {
   echo "Failed to connect to MySQL: (" . $conn->connect_errno . ") " .
$conn->connect_error;
   exit();
}

$result1 = $conn->query("SELECT * FROM stock WHERE stock_name =
'$_POST[symbol]'");
if ($result1){
    $row = $result1->fetch_assoc();
$_SESSION['stock_name'] = $row['stock_name'];
```

$_SESSION['stock_value'] = $row['stock_value'];

When the button is click and the database is queried, the result of the search is displayed under the button in a table. Its shows the symbol entered and current value:

$result1 = $conn->query("SELECT * FROM stock WHERE stock_name = '$_POST[symbol]'");

if ($result1){
$row = $result1->fetch_assoc();
$_SESSION['stock_name'] = $row['stock_name'];
$_SESSION['stock_value'] = $row['stock_value'];

echo "<form method='POST' action='indexmember.php?transaction'>
** <table border=1> \n**
<th>Stock Symbol   </th><th>Last Trading Value   </th><th>Number of Shares</th>\n

<tr><td>" . $_SESSION['stock_name'] . "</td><td>" . '$' . number_format($_SESSION['stock_value'],2) . "</td>

A printed graph of that stock is also displayed with the table above. It shows the stocks progress over the past year:

echo "";
} else{echo "Cannot find stock. Try again";}
}

After the stock and stock value is displayed, the user is prompted to type in how many shares of that stock they want to buy with a text input form:

<td><input id='field' type='text' name='quantity'></td><td><input id='button' type='submit' name='buy' value='Buy'></td></tr>\n

**</table></form>
\n";**

The functionality for buying stocks is executed with the button called buy and the variables input by the user are assigned to their corresponding variables in the database:

$username = $_SESSION['username'];

$total = $_POST['quantity'] * $_SESSION['stock_value'];

$result = $conn->query("SELECT * FROM stock");
$stock_id=$result->num_rows + 1;

```
$result = $conn->query("SELECT * FROM portfolio WHERE username = '$username' AND
active = 1");
    $row = $result->fetch_assoc();
$capital = $row['capital'];

$balance = $capital - $total;
$portfolio_id = $row['portfolio_id'];
$quantity = $_POST['quantity'];
$stock_name = $_SESSION['stock_name'];
$stock_value = $_SESSION['stock_value'];
```

A if else statement queries the database and stores the user, queries the database and stores the result in a variable $total. $total is then displayed with echo statements that say you have bought that many shares of that stock at the value of that stock:

```
if($result1){ # Only deduct from the capital if the insertion/update was successful

$result = $conn->query("UPDATE portfolio SET capital = '$balance' WHERE portfolio_id =
'$portfolio_id'");

echo "<font color='green'>Bought " . $quantity . " shares of " . $stock_name . " at $" .
number_value($stock_value,2) . " per share. Total cost: $" . $total . "</font>";
```

2.5. What-if Engine:

The what-if module allows users to create hypothetical transactions which tell them how much money they would have today if they had bough a certain amount of shares a stock on a particular day in the past. This module allows users to input text for the fields asking for quantity of shares and stock symbol:

```
<form method="POST" action="indexmember.php?whatif">
<td>How much money would I have if I bought </td>
<td><input type="text" name="quantity"></td>
<td> shares of <td>
<td><input type="text" name="symbol"></td>
```

The module uses a drop down menu for the date, broken up into three sections, “day”, “month”, and “year” shown in the example for “day”:

```
<select name="day">

  <option value="1">1</option>
  <option value="2">2</option>
 <option value="3">3</option>

</select>
```

The simulation is executed through a button shown below:

<td><input id="button" type="submit" name="submit" value="GO!"></td>

In order to store the information input by the user into the database, there is a connection to the database as shown in previous modules and the functionality is created by the click of the button. When he button is clicked, the user input for "symbol", "quantity" "year", "month" and "day" is stored in the corresponding variables in the database :

```
if(isset($_POST['submit']))

{
   $stock_name = $_POST['symbol'];
   $quantity = $_POST['quantity'];
   $year = "&c=" . $_POST['year'] . "&f=" . $_POST['year'];
   $month = "&a=" . $_POST['month'] . "&d=" . $_POST['month'];
   $day = "&b=" . $_POST['day'] . "&e=" . $_POST['day'];
```

To get the value for the amount of shares of the stock the user entered, on the date the user entered,there is a URL generator that when executed creates the URL that gets stored in the variable "$url" from yahoo finance.

```
$url = "http://ichart.finance.yahoo.com/table.csv?s=" . $stock_name . $year . $month .
$day . "&g=d&ignore=.csv";
```

The code below gets the actual value from the URL generated above and stores it in a variable named value**:**

```
$filesize = 2000;
$handle = fopen($url, "r");
$raw_quote_data = fread($handle, $filesize);
fclose($handle);
$quote_array = explode("\n", $raw_quote_data);
foreach ($quote_array as $quote_value) {
$quote = explode(",", $quote_value);
$value = $quote[2];
$data1 = $conn->query($query);
```

Once the value is stored, the amount of money that you would have made having purchased that number of shares of that stock on that day in the past is stored in a variable called $balance and is calculated as:

```
$balance = $newValue - $value;
$balance = $quantity * $balance;
```

where **$newValue = $row['stock_value'];** and $balance is the users capital.

Lastly, there is an if else statement that displays a message which says if you bought that amount of shares of that stock you would have gained, lost, or had no change in money. An examples is below :

```
$stock_name . " value at selected date: $" . number_format($value,2) . ". Stock value now: $" . number_format($newValue,2) . ". If you purchased " . $quantity . " shares, you would have gained $" . number_format($balance,2) . ". So sorry!<br>";
```

2.6. Competition Engine:

The competition engine allows users to create, join, view and withdraw from competitions. A competition allows users buy and sell stocks in a portfolio and compare the value of their portfolio with the value of the portfolio's of the other users entered in the competition. To create a competition the user enters the information for “Entrance Fee”, “Duration”, and “Starts In” into text fields. A button called Create is used to submit the user's input:

```
<table border="0"> <tr>
<form method="POST" action="indexmember.php?competition">
<td>Entrance Fee</td><td> <input type="text" name="fee"></td>
</tr> <tr>
<td>Duration (days) </td><td> <input type="text"
name="duration"></td>
</tr> <tr>
<td><input id="button" type="submit" name="create"
value="Create"></td>
</tr> </form> </table>
```

The functionality for the button is executed when the button is clicked. The if else statement below sets the variables entered by the user to its corresponding variables in the database:, queries the database and inserts the information into the database.

```
if(isset($_POST['create']))
{
   $fee = $_POST['fee'];
   $duration = $_POST['duration'];
   $date_created = date('Y\-n\-d');
   $username = $_SESSION['username'];

$result = $conn->query("SELECT * FROM competition");
$competition_id=$result->num_rows + 1;

$query = "INSERT INTO competition
(competition_id,fee,duration,date_created,username,finished)
VALUES('$competition_id','$fee','$duration','$date_created','$username'
,'0')"; # if not, insert it into the database
```

The competition engine also allows users to join competitions that are upcoming by clicking a button called “Join”which is in another php file called

join_competition.php. The user's information such as portfolio id will be inserted into the database with their corresponding variables:

```
$username = $_SESSION['username'];
$result1 = $conn->query("SELECT * FROM portfolio WHERE username =
'$username' AND active = 1");
 $row1 = $result1->fetch_assoc();
 $portfolio_id = $row1['portfolio_id'];
 $capital = $row1['capital'];

 $result1 = $conn->query("SELECT fee FROM competition WHERE
competition_id = '$competition_id'");
 $row1 = $result1->fetch_assoc();
 $fee = $row1['fee'];

$balance = $capital - $fee;

$query = "UPDATE portfolio SET competition_id = '$competition_id'
WHERE portfolio_id = '$portfolio_id'";
$result2 = $conn->query($query);

$result3 = $conn->query("UPDATE portfolio SET capital = '$balance'
WHERE username = '$username' AND portfolio_id = '$portfolio_id'");
```

This module also allows users to see the competitions they are currently in and have been in, in the past. This information is displayed in a table and is printing from the database.

```
echo '<div id="comp">';
 echo "<center><table border=1></center>\n";
 echo "<th>Competition Number</th><th>Created
by</th><th>Entrance Fee</th><th>Lasts until</th><th>Entered
portfolio</th>\n";

 $i=0;
 while ($i < $num) {
    $row1 = $result1->fetch_assoc();
    $competition_id = $row1['competition_id'];
    $portfolio_id = $row1['portfolio_id'];

$result2 = $conn->query("SELECT * FROM competition WHERE
competition_id = '$competition_id'");
    $row2 = $result2->fetch_assoc();
    $date_created = $row2['date_created'];
    $duration = $row2['duration'];
    $expiration = strtotime($date_created . "+ " . $duration . " days");
    $expiration = date("Y-m-d",$expiration);
```

```
    $fee = "$" . $row2['fee'];

echo "<form method='POST'
action='indexmember.php?competition'><tr><td>$competition_id</td
><td>$username</td><td>$fee</td><td>$expiration</td><td>$port
folio_id</td><td><input type='submit' name='withdraw'
value='Withdraw'/></td><td><input type='hidden'
name='competition_id'
value='$competition_id'/></td></tr></form>\n";
   $i++;
}
echo "</table>\n";
```

Lastly, users can withdraw from a portfolio by clicking a button called “Withdraw” and the user's money is returned to their portfolio:

```
$competition_id = $_POST['competition_id'];
   $result2 = $conn->query("SELECT fee FROM competition WHERE
competition_id = '$competition_id'");
    $row2 = $result2->fetch_assoc();
   $fee = $row2['fee'];

$username = $_SESSION['username'];
   $result1 = $conn->query("SELECT * FROM portfolio WHERE username
= '$username' AND active = 1");
    $row1 = $result1->fetch_assoc();
   $portfolio_id = $row1['portfolio_id'];
   $capital = $row1['capital'];
  $balance = $fee + $capital;

   $result1 = $conn->query("UPDATE portfolio SET capital = '$balance'
WHERE portfolio_id = $portfolio_id");
   $result1 = $conn->query("UPDATE portfolio SET competition_id =
NULL WHERE portfolio_id = $portfolio_id");

   echo "<font color='green'>Successfully withdrew from competition " .
$competition_id . "</font><br>";
```

2.7. Home Page:

The homepage module is used to display pertinent information to include account information, news feeds, and ticker feeds. This particular module utilizes HTML and PHP as the primary programming languages for the creation of this module's source code.

We use the include command to call several files; repository.php, show_stock_quotes(), account.php, newsfeed.php and top_ten.php. The use of these include files complete the homepage.

3. Database Management:

Back-end data in Project Greed is maintained in a series of MySql tables kept on the home server. Each table represents an entity that interacts with the user interface and stores information related to it. These tables are: 'stock', 'portfolio', 'competition', and 'account'. Accounts are identified by the primary key username, and it is in this table that passwords are also stored. For portfolios, competitions, and stock, it is imperative that each of them are identified by unique primary keys, namely portfolio_id, competition_id, and stock_id respectively. Each portfolio is associated with an account and competition via foreign keys. Aside from this, this table records each portfolio's current capital as a float value, and an activation status as a boolean value. The stock table includes all ambient stocks updated via the repository engine, as well as any stocks purchased by users. Each of the latter is associated with a portfolio. This table also records the stocks' current value and, if owned by a portfolio, the number of shares it makes up. Each competition is associated with the user that created it, but also includes other information such as an entrance fee (float), duration in days (integer), and date created (date). The date and is stored in MySql standard format (YYYY-MM-DD).

4**.** Error and Exception Handling:

Error and exception handling provides the developers of this software a valuable tool in providing our customers with a quality product. Throughout this program, we have implemented many error checking measures to ensure proper input is executed and improper input executes an error or exception with instructions on how to correct the error. Additionally, we did not find any major user errors that could cause fatal functionality.

5**. Conclusion:**

This was a challenging project to say the least. This project certainly tested our resolve both individually and as a team. Working together with our group from the start of our project up until the presentation of our project has ultimately been our saving grace. Had we procrastinated in any way this project would have been doomed from the start. The group performed exceptionally under all circumstances. All in all, this project proved to be fun, a good educational experience and has set us up for future success.