
Scope (satisfying spec's requirements): ✓

Correctness (bug-free-ness): ✓

Design (elegance of implementation): ✓

Style (pretty-printed code): ✓+

Your design is quite good - I like how modular your code is. You could have improved it even more by using Decimal instead of Float for storing money, utilizing more functions in a function include file or in a class that provides an interface for handling validation, and setting up appropriate indexes and primary key for your History table.

C\$75 Finance Grading Survey

Scope

To what extent does your code implement the features required by our specification?
Generally speaking, think of ✓+ as "perfect", ✓ as "good," and ✓− as "not good".

Required Features

Your site must require that a user log in with a username and password in order to see or do anything (except, obviously, log in or register).

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must require that a user log in with a username and password in order to see or do anything (except, obviously, log in or register)."

The user should not be able to "log out" if they are not "logged in." You should eliminate the log out link on the home page if they do not have a session key.

Your site must allow a user to register for an account.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must allow a user to register for an account."

Your site successfully notifies the user when a registration has been successful, but does not provide them with a link to log in. After registration, your user should be redirected to a page where they can log in, with a message at the top of the page that indicates whether it has been successful or not. Optionally, you could log the user in if the registration was successful.

Your site must make sure a user's email is syntactically valid.

Assume that an email address must be of the form username@domain.tld or username@subdomain.domain.tld, where tld contains only alphabetical characters, subdomain and domain contain only alphanumeric characters, and username contains only alphanumeric characters, dots, underscores, hyphens, and/or pluses.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must make sure a user's email is syntactically valid."

You definitely have validation in place but there are a few problems.

Your client-side email validation incorrectly passes the following tests:
should reject: asdf@asdf.c0m
should reject: a=df@asdf.com

Your client-side email validation correctly passes the following tests:
should reject: asdf@as5df.com
should reject: asdfasdf@asdf.com
should reject: [blank]
should reject: asdf@asdf
should reject: asdf

Your server-side email validation incorrectly deals with the following tests:
should reject: asdf@asdf.c0m
should reject: asdf@as5df.com
should reject: a=df@asdf.com
should reject: asdf asdf@asdf.com
should reject: [blank]
should reject: asdf@asdf
should reject: asdf

Your server-side validation correctly passes the following tests:
should accept: asdf@asdf1.com
should accept: lJustHappenToHaveALongEmail@harvard.fas.edu
should accept: asdf1_asdf.asdf+asdf-asdf@asdf.com
should reject: duplicate usernames // vague "error" when server side validation is used

Your site must make sure a user's password is secure.

Assume that a user's password must be at least six characters, and it cannot be entirely alphabetic, or entirely numeric.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must make sure a user's password is secure."

Your server side password validation incorrectly deals with the following passwords:
should reject: as12

Your password validation does not follow these best practices:
— You should refuse a password that is the same as their username.
reject if username AND password: us3@us.us

Your server-side password validation correctly passes the following tests, and in several cases was enhanced by client-side validation (great!):
should accept: asdfgh1
should accept: l@#51AB
should reject: asdfghj // note warning message "Please make sure your password is alphanumerical" is kind of vague
should reject: 1234567 // note warning message "Please make sure your password is alphanumerical!" is kind of vague
should reject: as12

Your username/password validation correctly follows the following best practices:
— Do not give bland error messages that give no guidance to the user, such as "your entry is not valid"
— Users that want the added security of a longer password should be able to:
should accept: "1 should be able to have a 40 char ~~good~~"
— Warn your user if they have caps lock on:
warns: CAPS LOCK
Note - this is as simple in `jQuery` as
\$(`#example`).`keypress`(function(e) { var s = String.fromCharCode(e.which);
if (s.toLowerCase() === s && s.toLowerCase() !== s && !e.shiftKey) {alert("caps is on");
}); // http://goo.gl/uwHfx
// note - you actually use your password in the clear, which is kind of controversial but has a lot of advantages, one of which is that it avoids the caps lock problem.

Upon registering, a user should receive a free gift: \$10,000 in cash.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Upon registering, a user should receive a free gift: \$10,000 in cash."

Your site must allow a user to get a quote (i.e., look up Last Trade) for a stock by providing its symbol.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must allow a user to get a quote (i.e., look up Last Trade) for a stock by providing its symbol."

Your site must allow a user to buy shares of a stock by providing its symbol.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must allow a user to buy shares of a stock by providing its symbol."

This works quite well, however, the artificial limit of purchasing a maximum of 99 shares is completely arbitrary and not really feasible: there are plenty of penny stocks that you may want to purchase 1000 at a time.

Your site must allow a user to buy more shares of a stock that he or she already owns.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must allow a user to buy more shares of a stock that he or she already owns."

This works quite well

Your site must prevent purchasing a fraction of a share.

- ☐ ✓+
☒ ✓
☐ ✓−

Comments on "Your site must prevent purchasing a fraction of a share."

This kind of works because you have made the form only big enough to carry two digits. However, that is a `ctrl+shift+c` away from tampering in many browsers. I was easily able to get it to display 1.3 times the value of one share in the price column, and when I purchased, it rounded the number of stocks down to one, and charged my account exactly 1.3 times the price for the stock. This could be exploited for abuse. I did notice that the transaction value recorded in the database is the actual amount, not 1.3x the amount.

Your site must allow a user to sell shares of a stock that he or she already owns.

For simplicity, you may require that a user sell all or none rather than some.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must allow a user to sell shares of a stock that he or she already owns."

This works well.

Your site must prevent selling a fraction of a share.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must prevent selling a fraction of a share."

Works perfectly

Your site must allow a user to check the current value of his or her portfolio.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must allow a user to check the current value of his or her portfolio."

Account balance is displayed, but no net worth or portfolio current value is displayed as far as I can see. I do like how you can see how much the current value of the stocks you own differs from the price you bought them at

Your site must perform client-side validation, where possible, of user input related to a buy or a sell.

For instance, if some text field must contain a non-negative integer (e.g., number of shares to buy or sell), you must reject attempts to submit invalid input (as by admonishing the user with an alert) or prevent them from typing anything non-numeric at all.

- ☐ ✓+
☒ ✓
☐ ✓−

Comments on "Your site must perform client-side validation, where possible, of user input related to a buy or a sell."

This raises an interesting question - is an `html` intervention like setting the maximum length of an input field a client-side validation? Client-side validation is "a circumventable convenience to speed up server validation," and your max length use is just as circumventable as `javascript`, but it actually introduces an inconvenience.

On any page designed to take user input, you should give focus (via JavaScript) to the first field requiring the user's attention (e.g., the username field on your login page).

- ☐ ✓+
☐ ✓
☒ ✓−

Comments on "On any page designed to take user input, you should give focus (via JavaScript) to the first field requiring the user's attention (e.g., the username field on your login page)."

On `buystock.php`, the focus does not automatically go to the first and only input.

Your site must perform rigorous server-side error-checking.

Your site should validate input even if javascript is disabled.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must perform rigorous server-side error-checking."

You have some server side error-checking, but it is by no means rigorous. See comments above.

Scope: Grade

Scope: To what extent does the code implement the features required by our specification?

Assume a default score of ✓+. If code fails to satisfy one or some of the spec's requirements, downgrade to ✓. If code fails to satisfy most (8/15) of the spec's (feature) requirements, downgrade further to ✓−.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Scope: To what extent does the code implement the features required by our specification?"

This is a really solid submission. Way to go.

That said, you have incompletely implemented the following feature requirements:
1.) Your site must prevent purchasing a fraction of a share.
2.) Your site must perform client-side validation, where possible, of user input related to a buy or a sell.
3.) On any page designed to take user input, you should give focus (via JavaScript) to the first field requiring the user's attention
4.) Your site must perform rigorous server-side error-checking.

- ☐ ✓+
☒ ✓
☐ ✓−

Powered by [Google Docs](#)

[Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)

C\$75 Finance Grading Survey

Correctness

To what extent is your code consistent with our specifications and free of bugs? Generally speaking, think of ✓+ as "perfect", ✓ as "good," and ✓− as "not good".

Technical Requirements

Is there a readme in a file called README that lives in the same folder as the rest of your project?

- ☒ ✓+
☐ ✓
☐ ✓−

You must ensure that your application works within the CS50 Appliance at a URL of `http://localhost/~jharvard/project1/` when uploaded (as via SCP or SFTP) to `/home/jharvard/public_html/project1/`.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "You must ensure that your application works within the CS50 Appliance at a URL of `http://localhost/~jharvard/project1/` when uploaded (as via SCP or SFTP) to `/home/jharvard/public_html/project1/`."

You must store users' data in a MySQL database whose name starts with `jharvard_`.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "You must store users' data in a MySQL database whose name starts with `jharvard_`."

All PHP files must be `chmod'd 600`.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "All PHP files must be `chmod'd 600`."

Your site must pull its real-time data (prices, etc.) from Yahoo! Finance by parsing its CSVs.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Your site must pull its real-time data (prices, etc.) from Yahoo! Finance by parsing its CSVs."

Your database tables must have constraints and indexes defined where appropriate.

- ☐ ✓+
☒ ✓
☐ ✓−

Comments on "Your database tables must have constraints and indexes defined where appropriate."

Your users table does not have primary keys. Money should not be stored in floats because it rounds incorrectly.

You must avoid race conditions by using SQL transactions or locks.

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "You must avoid race conditions by using SQL transactions or locks."

Things to look for

Can you register for a new account only with a valid email address?

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Can you register for a new account only with a valid email address?"

Can the user only log in with valid account information?

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Can the user only log in with valid account information?"

Can the user log out?

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Can the user log out?"

Can the user perform any actions without logging in?

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Can the user perform any actions without logging in?"

When buying and selling stocks, is the proper amount added to or deducted from the user's balance?

- ☐ ✓+
☒ ✓
☐ ✓−

Comments on "When buying and selling stocks, is the proper amount added to or deducted from the user's balance?"

Can the user only buy valid stocks?

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Can the user only buy valid stocks?"

Can the user only sell stock that has already been purchased?

- ☒ ✓+
☐ ✓
☐ ✓−

Comments on "Can the user only sell stock that has already been purchased?"

Is all HTML and CSS valid?

- ☐ ✓+
☒ ✓
☐ ✓−

Comments on "Is all HTML and CSS valid?"

Mostly.

Is both client and server-side validation performed (i.e. error-checking), so crashing the app is impossible?

- ☐ ✓+
☒ ✓
☐ ✓−

Comments on "Is both client and server-side validation performed (i.e. error-checking), so crashing the app is impossible?"

Mostly.

Correctness: Grade

Correctness: To what extent is the code consistent with the specifications and free of bugs? Assume a default score of ✓. If code is truly 100% bug-free, upgrade to ✓+. If code manifests many bugs and really leaves lots of room for improvement, downgrade to ✓−.

- ☐ ✓+
☒ ✓
☐ ✓−

Comments on "Correctness: To what extent is the code consistent with the specifications and free of bugs?"

« Back

Continue »

Powered by [Google Docs](#)

[Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)


```
1. /home/jharvard/public_html/students/ already existed; chmodding to 711
2.
3.      TF: Peter Nore, Student: crystalhsieh7, Project: project1
4.      /home/jharvard/public_html/students/crystalhsieh7 already existed; chmodding to 711
5. /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/ already existed; chmodding to 711
6. downloading project1 by crystalhsieh7executing ... cd /home/jharvard/public_html/students/crystalhsieh7; git clone
   git@repo.cs50.net:crystalhsieh7/project1.git; git checkout submit;
7. Cloning into project1...
8.
9. /project1 dir, chmodding to 711; found project directory
10. /project1 dir, chmodding to 711 with permissions 0711
11. /project1/images dir, chmodding to 711 with permissions 0700
12. /project1/images/CS75FINANCE.png dir, chmodding to 711 public file, changing to 644
13. /project1/README.txt~ dir, chmodding to 711 public file, changing to 644
14. /project1/buystock.php dir, chmodding to 711 with permissions 0600
15. /project1/register.php~ dir, chmodding to 711 banishing to
   /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/register.php~
16. /project1/logout.php dir, chmodding to 711 with permissions 0600
17. /project1/footer.html dir, chmodding to 711 public file, changing to 644
18. /project1/browse.php~ dir, chmodding to 711 banishing to /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/browse.php~
19. /project1/header_validate.php dir, chmodding to 711 with permissions 0600
20. /project1/loggedin.php dir, chmodding to 711 with permissions 0600
21. /project1/login.php~ dir, chmodding to 711 banishing to /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/login.php~
22. /project1/addstocktotable.php dir, chmodding to 711 with permissions 0600
23. /project1/FINANCE_STYLE.css dir, chmodding to 711 public file, changing to 644
24. /project1/index.php~ dir, chmodding to 711 banishing to /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/index.php~
25. /project1/js dir, chmodding to 711 with permissions 0700
26. /project1/js/jquery-1.5.1.min.js dir, chmodding to 711 public file, changing to 644
27. /project1/js dir, chmodding to 711 with permissions 0711
28. /project1/js/jquery.validate.min.js dir, chmodding to 711 public file, changing to 644
29. /project1/input.php~ dir, chmodding to 711 banishing to /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/input.php~
30. /project1/input.php dir, chmodding to 711 with permissions 0600
31. /project1/input2.php~ dir, chmodding to 711 banishing to /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/input2.php~
32. /project1/README.txt dir, chmodding to 711 public file, changing to 644
33. /project1/browse.php dir, chmodding to 711 with permissions 0600
34. /project1/login.php dir, chmodding to 711 with permissions 0600
35. /project1/project1.sql dir, chmodding to 711 with permissions 0600
36. /project1 dir, chmodding to 711 with permissions 0711
37. /project1/loggedin.php~ dir, chmodding to 711 banishing to
   /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/loggedin.php~
38. /project1/addstocktotable.php~ dir, chmodding to 711 banishing to
   /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/addstocktotable.php~
39. /project1/index.php dir, chmodding to 711 with permissions 0700
40. /project1/FINANCE_STYLE.css~ dir, chmodding to 711 public file, changing to 644
41. /project1/priceofstock.php~ dir, chmodding to 711 banishing to
   /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/priceofstock.php~
42. /project1/blueprint dir, chmodding to 711 with permissions 0700
43. /project1/blueprint/plugins dir, chmodding to 711 with permissions 0700
```

```
44. /project1/blueprint/plugins/rtl dir, chmodding to 711 with permissions 0700
45. /project1/blueprint/plugins/rtl/readme.txt dir, chmodding to 711 public file, changing to 644
46. /project1/blueprint/plugins/rtl dir, chmodding to 711 with permissions 0711
47. /project1/blueprint/plugins/rtl/screen.css dir, chmodding to 711 public file, changing to 644
48. /project1/blueprint dir, chmodding to 711 with permissions 0711
49. /project1/blueprint/plugins/link-icons dir, chmodding to 711 with permissions 0700
50. /project1/blueprint/plugins/link-icons/readme.txt dir, chmodding to 711 public file, changing to 644
51. /project1/blueprint/plugins/link-icons/icons dir, chmodding to 711 with permissions 0700
52. /project1/blueprint/plugins/link-icons/icons/visited.png dir, chmodding to 711 public file, changing to 644
53. /project1/blueprint/plugins/link-icons/icons/email.png dir, chmodding to 711 public file, changing to 644
54. /project1/blueprint/plugins/link-icons/icons/xls.png dir, chmodding to 711 public file, changing to 644
55. /project1/blueprint/plugins/link-icons dir, chmodding to 711 with permissions 0711
56. /project1/blueprint/plugins/link-icons/icons/doc.png dir, chmodding to 711 public file, changing to 644
57. /project1/blueprint/plugins/link-icons/icons/external.png dir, chmodding to 711 public file, changing to 644
58. /project1/blueprint/plugins/link-icons/icons dir, chmodding to 711 with permissions 0711
59. /project1/blueprint/plugins/link-icons/icons/lock.png dir, chmodding to 711 public file, changing to 644
60. /project1/blueprint/plugins/link-icons/icons/pdf.png dir, chmodding to 711 public file, changing to 644
61. /project1/blueprint/plugins/link-icons/icons/feed.png dir, chmodding to 711 public file, changing to 644
62. /project1/blueprint/plugins/link-icons/icons/im.png dir, chmodding to 711 public file, changing to 644
63. /project1/blueprint/plugins/link-icons dir, chmodding to 711 with permissions 0711
64. /project1/blueprint/plugins/link-icons/screen.css dir, chmodding to 711 public file, changing to 644
65. /project1/blueprint/plugins/buttons dir, chmodding to 711 with permissions 0700
66. /project1/blueprint/plugins/buttons/readme.txt dir, chmodding to 711 public file, changing to 644
67. /project1/blueprint/plugins/buttons/icons dir, chmodding to 711 with permissions 0700
68. /project1/blueprint/plugins/buttons/icons/cross.png dir, chmodding to 711 public file, changing to 644
69. /project1/blueprint/plugins/buttons dir, chmodding to 711 with permissions 0711
70. /project1/blueprint/plugins/buttons/icons dir, chmodding to 711 with permissions 0711
71. /project1/blueprint/plugins/buttons/icons/key.png dir, chmodding to 711 public file, changing to 644
72. /project1/blueprint/plugins/buttons/icons/tick.png dir, chmodding to 711 public file, changing to 644
73. /project1/blueprint/plugins/buttons dir, chmodding to 711 with permissions 0711
74. /project1/blueprint/plugins/buttons/screen.css dir, chmodding to 711 public file, changing to 644
75. /project1/blueprint/plugins dir, chmodding to 711 with permissions 0711
76. /project1/blueprint/plugins/fancy-type dir, chmodding to 711 with permissions 0700
77. /project1/blueprint/plugins/fancy-type/readme.txt dir, chmodding to 711 public file, changing to 644
78. /project1/blueprint/plugins/fancy-type dir, chmodding to 711 with permissions 0711
79. /project1/blueprint/plugins/fancy-type/screen.css dir, chmodding to 711 public file, changing to 644
80. /project1/blueprint/src dir, chmodding to 711 with permissions 0700
81. /project1/blueprint/src/reset.css dir, chmodding to 711 public file, changing to 644
82. /project1/blueprint/src/ie.css dir, chmodding to 711 public file, changing to 644
83. /project1/blueprint/src/typography.css dir, chmodding to 711 public file, changing to 644
84. /project1/blueprint dir, chmodding to 711 with permissions 0711
85. /project1/blueprint/src/forms.css dir, chmodding to 711 public file, changing to 644
86. /project1/blueprint/src dir, chmodding to 711 with permissions 0711
87. /project1/blueprint/src/grid.css dir, chmodding to 711 public file, changing to 644
88. /project1/blueprint/src/print.css dir, chmodding to 711 public file, changing to 644
89. /project1/blueprint/src/grid.png dir, chmodding to 711 public file, changing to 644
90. /project1/blueprint/ie.css dir, chmodding to 711 public file, changing to 644
91. /project1 dir, chmodding to 711 with permissions 0711
```

```
92. /project1/blueprint dir, chmodding to 711 with permissions 0711
93. /project1/blueprint/print.css dir, chmodding to 711 public file, changing to 644
94. /project1/blueprint/screen.css dir, chmodding to 711 public file, changing to 644
95. /project1/mysql_connect.php dir, chmodding to 711 with permissions 0600
96. /project1/sellstock.php dir, chmodding to 711 with permissions 0600
97. /project1/logout.php~ dir, chmodding to 711 banishing to /home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/logout.php~
98. /project1/sellstock.php~ dir, chmodding to 711 banishing to
/home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/sellstock.php~
99. /project1/portfolio.php dir, chmodding to 711 with permissions 0600
100. /project1/priceofstock~ dir, chmodding to 711 banishing to
/home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/priceofstock~
101. /project1/buystock.php~ dir, chmodding to 711 banishing to
/home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/buystock.php~
102. /project1/mysql_connect.php~ dir, chmodding to 711 banishing to
/home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/mysql_connect.php~
103. /project1/register.php dir, chmodding to 711 with permissions 0600
104. /project1/header_validate.php~ dir, chmodding to 711 banishing to
/home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/header_validate.php~
105. /project1/portfolio.php~ dir, chmodding to 711 banishing to
/home/jharvard/public_html/students/crystalhsieh7/banished_temp_files/portfolio.php~
106. local path to dir: /home/jharvard/public_html/students/crystalhsieh7/project1
107. relative loc: /project1
108. url: http://192.168.56.50/~jharvard/students/crystalhsieh7/project1
109. http://192.168.56.50/~jharvard/students/crystalhsieh7/project1
110.
111. simulated local url: http://localhost/~jharvard/project1
112. summary: http://192.168.56.50/~jharvard/students/crystalhsieh7/project1/000_teaching_fellow_script_summary.txt;
113.
```

```
1. body{
2.     background-color: #003300;
3.     font-family: Lucida Sans Unicode, Lucida Grande, sans-serif;
4. }
5.
6. #header{
7.     padding-top: 140px;
8.     overflow: hidden;
9.     background-image: url(images/CS75FINANCE.png);
10.    /* used Paint on my computer to create the header image */
11.    background-repeat: no-repeat;
12.    height: 0px;
13.    margin: 5px 0 0 0;
14. }
15. #error{
16.     color: red;
17.     padding: 0px;
18.     font-size: 100%;
19. }
20.
21. th {
22.     font-size: 110%;
23. }
24.
25. h2{
26.     font-size: 110%;
27. }
28.
29. a:link{
30.     font-size: 110%;
31. }
32.
33.
34. /* create the look for the navigation bar with pure css */
35. #navigation{
36.     width: 12em;
37.     border-right: 1px solid;
38.     text-decoration: none;
39.     color: black;
40.     margin: 10px 0 0 0;
41.     list-style: none;
42.     margin: 0;
43.     padding: 0;
44. }
45.
46. #navigation li {
47.     margin: 0;
48.     border-top: 1px none;
```



```
49. }
50.
51. #navigation li a {
52.     display: block;
53.     background-color: #BCF5BC;
54.     border-top: 1px solid;
55.     font-size: 130%;
56.     border-bottom: 1px solid;
57.     text-decoration: none;
58.     padding: 2px 2px 2px 0.5em;
59.     color: #41383C;
60. }
61.
62. #navigation li a:hover{
63.     background-color: #CCFF99;
64. }
65.
66. }
67. .span-16{
68.     background-color: #FFFFFF;
69.     color: #2B60DE;
70.     margin: 20px 0px 0px 15px;
71.     padding: 25px 40px 50px;
72. }
73. p{
74.     font-size: 120%;
75.     margin: 0px;
76.     padding: 6px 0px;
77. }
78. .span-23{
79.     background-color: white;
80. }
81. .container{
82.     background-color: #FFFFFF;
83.     padding: 10px 0px 40px 40px;
84. }
85.
86. #company{
87.     font-size: 140%;
88.     padding: 10px 0px 0px 0px;
89. }
90.
91. h1{
92.     font-size: 180%;
93.     padding: 5px 0px 0px 0px;
94. }
95. table td, table th {
96.     border: thin solid black;
```



```
97.     margin: none;
98. }
99. table {
100.     border: thin solid black;
101. }
```


1. -Crystal Hsieh
2. CS-75
3. README FILE
- 4.
5. Hello, my name is Crystal Hsieh, and this is my C\$75 project.
- 6.
7. Please use Mozilla Firefox and Chrome to view my project.
- 8.
9. When beginning this project, I decided to focus on creating the login/register piece of the website before any of the other specifications for the project. First, I created the two tables for my database jharvard_crystalfinance called Users and History. Basically, Users would act as my data storage for any information about the user, like their name, their email, and their username. The History page would be used to remember transactions, and allow easy searches to find all the transactions from one user.
- 10.
11. I also tried to clean up the code by using require() to separate different parts of code into separate files. Besides the login and register page, the rest of the pages should not have been able to be accessed by someone who wasn't signed in. I created a loggedin.php so that I could just place a require() statement to check if the user was signed in. I also created the files header_validate and mysql_connect, so that I could call the files easily from any page. Header_validate would create the front part of every page of code, including the Javascript and PHP functions as needed. The mysql_connect page would set up the connection to the database for any page. I also included a footer.html page to close the open div tags from header_validate.
- 12.
13. When someone first accesses C\$75 Finance, they have to create an account on register.php. After going through the validation of creating an alphanumeric password and other requirements, the person can log in to their account via the login page. When they create an account, a new row is added to the Users table on my database. I also decided to use jQuery to help validate my form values, because the validation plugin also has a feature to validate emails, which made it a lot less confusing for me then using regular expressions. However, if I had more experience using Javascript, I would validate the form to check if the password was alphanumeric and alert the user before proceeding to the next page. Instead, the page processing the form prints that the user had an invalid password, and the user must go back to create a new one.
- 14.
15. Once the user goes to the homepage, they can click on the navigation bar for buying stock. I was originally going to have a navigation bar that had 6 different links (Home, Buy Stock, Sell Stock, Portfolio, Browse Stock Prices, Log Out). However, I realized that when a user is looking at their portfolio and they see their Money Gain/Money Loss, it would probably be convenient to be able to just click Sell Stock in their portfolio rather than at another link. Also, when a user is browsing different stock prices, it seems a lot more convenient for the user to quickly click Buy Stock when they see a good deal.
- 16.
17. When creating the table that shows up when a user searches a company symbol, I didn't want to include every part that was given by the CSV file. Rather, I tried to find the most compelling parts of the company's stock, like the Last Trade and the recent change in price. By fopen() and fgetcsv(), I was able to parse the csv file and retrieve the different parts of the array that I needed.
- 18.
19. When clicking the button to buy stock, the user is actually taken to another page called buystock.php, which is like a confirmation page for the user. It gives what the total price will be and it allows the user to change the desired quantity of the stock they wish to buy. When they click update, the page will reload and update the total price of the stocks. When they finally click Buy Stock, the php code checks to make sure the user actually has enough money to buy those stocks, then it subtracts the price of the stocks from the current amount of money the user has. As shown in class, I tried to implement transactions (commit and rollback) to prevent race conditions.
- 20.
21. I also tried to implement mysql_real_escape_string() on all my \$_POST values in order to prevent sql injection attacks. However, when I put mysql_real_escape_string() on my form values, my code would not allow me to new rows on my Users table. I thought it was more important that a user is able to create an account then it was to implement mysql_real_escape_string(), so I decided to take out mysql_real_escape_string() for those values. Also, I tried to confirm that the password was alphanumeric with alnum(), but it also made my code malfunction. It may be a syntax error, but I block comment where the code should have been.

- 22.
- 23. Lastly, as I tried to validate all my XHTML pages, sometimes it would give me the error that the document type does not allow tags like "input" or "form" to be outside of an "object" or "div" tag. After I place the input or form values inside an object tag, it continues to give me an error. This is only for pages "browse.php" and "buystock.php".
- 24.
- 25. I hope you like my site!
- 26.
- 27. Crystal Hsieh
- 28.
- 29.

Thanks!

```
1.  <?
2.      //gather all code from separate files
3.      //make sure user is logged in
4.      require("loggedin.php");
5.      //make sure there is a connection to the database
6.      require("mysql_connect.php");
7.      //recieve the overhead code
8.      require("header_validate.php");
9.
10.     //get the username of the current user
11.     $user = $_SESSION["username"];
12.     //find the row in the table Users that is owned by this user
13.     $sql = sprintf("SELECT * FROM Users WHERE user='$user'");
14.     //archive the symbol of the stock that wants to be bought
15.     $sym = $_SESSION["gsymbol"];
16.
17.     //query for that input
18.     $result = mysql_query($sql);
19.
20.     //check to make sure that it can query the database
21.     if ($result === FALSE){
22.         die("Could not query database");
23.     }
24.
25.     //if theres a row given, then get the info from it
26.     if (mysql_num_rows($result) == 1)
27.     {
28.         //fetch the array from the result
29.         $array = mysql_fetch_array($result);
30.         //gather the variables from the previous page
31.         //get the amount of money the user has
32.         $money = $array['moneyamt'];
33.         //these variables were calculated in the previous page and then stored as SESSION variables
34.         $price = $_SESSION["price"];
35.         $tprice = $_SESSION["tprice"];
36.         $quantity = $_SESSION["quantity"];
37.
38.         //if theres a result, find how much money they have
39.         if($money < $tprice){
40.             die("You do not own enough money to purchase this!");
41.         }else{
42.             //if they have enough money, subtract the amount from their account
43.             $money -= $tprice;
44.             //transaction begins
45.             @mysql_query("BEGIN");
46.             $sql = sprintf("UPDATE Users SET moneyamt = '$money' WHERE user = '$user'");
47.             $result2 = mysql_query($sql, $connect);
48.             //using TRANSACTIONS in mysql to prevent race conditions
```



```
49.         if(!$result2){
50.             @mysql_query("ROLLBACK");
51.         }else{
52.             @mysql_query("COMMIT");
53.         }
54.         //in case no connection is returned, return the error message
55.         if ($result2 === FALSE){
56.             die("Could not query database2");
57.         }
58.
59.         //check the history to see if this user has bought this stock before
60.         $sql = sprintf("SELECT * FROM History WHERE owner='$user' AND company='$sym'");
61.         $resulthis = mysql_query($sql, $connect);
62.
63.         //if no connection is returned, return error message
64.         if($resulthis === FALSE){
65.             die("Could not query database4");
66.         }
67.
68.         //if a row is found, fetch the array
69.         if (mysql_num_rows($resulthis) == 1){
70.             $array = mysql_fetch_array($resulthis);
71.
72.             //add the previous quantity to the new one
73.             $newquant = $array['quantity'] + $quantity;
74.
75.             //update the table with new quantity
76.             $sql = sprintf("UPDATE History SET quantity='$newquant' WHERE owner='$user' AND company='$sym'");
77.             $updated = mysql_query($sql, $connect);
78.
79.             //for debugging, error message will show if update doesn't work
80.             if($updated === FALSE){
81.                 die("updated didn't work");
82.             }
83.
84.         }else{
85.
86.             //add to the history table that the user has purchased something
87.             $sql = sprintf("INSERT INTO `History`(`company`, `owner`, `pricebought`, `quantity`) VALUES ('$sym','$user','$price', '
$quantity')");
88.
89.             //transaction begins
90.             @mysql_query("BEGIN");
91.             $result3 = mysql_query($sql);
92.
93.             //using TRANSACTIONS in mysql to prevent race conditions
94.             if(!$result3){
95.                 @mysql_query("ROLLBACK");
96.             }else{
```

```
96.         @mysql_query("COMMIT");
97.     }
98.
99.     //if error, send error message
100.    if ($result3 === FALSE){
101.        die("Could not query database3");
102.    }
103. }
104. print("You have successfully purchased stock!");
105. }
106. }
107. ?>
108. <h3>You currently have $<? print($money); ?> in your account.</h3>
109. <?
110. //reset the session variables to 0 for the next transaction
111. $_SESSION["tprice"] = 0;
112. $_SESSION["price"] = 0;
113. $_SESSION["gsymbol"] = null;
114.
115.
116. require("footer.html");
117. ?>
```



```

1. <?
2.     //retrieve necessary files
3.     //make sure user is logged in, connected to database, and basic formatting
4.     require("loggedin.php");
5.     require("mysql_connect.php");
6.     require("header_validate.php");
7.     $user = $_SESSION["username"];
8.
9. ?>
10. <!-- this is basically the search engine for stock prices -->
11. <h1>Browse for stock prices:</h1>
12. <h2>You currently have $<? print(getCurrentMoney($user));?> in your account!</h2>
13. <p>
14. Symbol:
15. <br/>
16. <object>
17. <p>
18. <!-- for some reason, when I validate this html document, it keeps giving me the error that I need to place the "form" and "input" tags within
    a div tag or object tag, but even when I do, it still gives me an error -->
19. <form action="<? $_SERVER["PHP_SELF"]; ?>" method="post">
20. <input class="first" type="text" name="gsymbol" id="gsymbol"/>
21. <input type="submit" name="submit" value="Search"/>
22. </form>
23. </p>
24. </object>
25. </p>
26. <?
27.
28.     if(isset($_POST["gsymbol"])){
29.         //real escape string to prevent sql injection attacks
30.         $gsymbol = mysql_real_escape_string($_POST["gsymbol"]);
31.         //I created functions that are in the header_validate.php file, like findInfo
32.         $info = findInfo($gsymbol);
33.         if($info === FALSE){
34.             //if no info is found, then return "there is no company"
35.             print("There is no company from this symbol.");
36.         }else{
37.             //if company is found, print out results in a table format
38. ?>
39.         <h2 id="company">
40.         Company Name:
41.         <?print($info[0]);?>
42.         </h2>
43.         <br/>
44.         <table>
45.             <tr>
46.                 <td>Last Trade: $<?print($info[1]);?></td>
47.                 <td>Change: <?print($info[4]);?></td>

```

The <object> tag is supported in all major browsers, except Firefox.

Definition and Usage

The <object> tag is used to include objects such as images, audio, videos, Java applets, ActiveX, PDF, and Flash.

The object element allows you to specify the data and parameters for objects inserted into HTML documents, and the code that can be used to display/manipulate that data.

```
48.         <tr>
49.             <td>Trade Time: <?print($info[3]);?></td>
50.             <td>Trade Day: <?print($info[2]);?></td>
51.         </tr>
52.     </table>
53.     <div>
54.         <!-- for some reason, when I validate this html document, it keeps giving me the error that I need to place the "form" and "input"
tags within a div tag or other, but even when I do, it still gives me an error -->
55.         <form action="buystock.php" method="post">
56.             <? $_SESSION["gsymbol"] = $gsymbol; ?>
57.             <input type="submit" value="Buy Stock"/>
58.         </form>
59.     </div>
60. <?     }
61.     }
62.     require("footer.html");
63.
64.
65. ?>
```



```

1. <?
2.     //attach necessary documents
3.     require("loggedin.php");
4.     require("header_validate.php");
5.
6.     $quantity = 1;
7.
8.     //checks if there is a valid value for quantity
9.     if(isset($_POST["quantity"]) && $_POST["quantity"] > 0){
10.         $quantity = $_POST["quantity"];
11.     }
12.
13. ?>
14. <h1>Complete Your Transaction</h1>
15. <? if(isset($_SESSION["gsymbol"])){
16.     //store the given symbol
17.     $sym = $_SESSION["gsymbol"];
18.     //store the information on the symbol
19.     $array = findInfo($sym);
20.     //retrieve the current price via the getPrice() function
21.     $currentprice = (float)(getPrice($sym));
22.     //calculate the total
23.     $total = ($quantity)*($currentprice);
24.     //store the values in session variables for addstocktotable.php page
25.     $_SESSION["tprice"] = $total;
26.     $_SESSION["price"] = $currentprice;
27.     $_SESSION["quantity"] = $quantity;
28.
29.     //present table of the company stock and allow user to change quantity
30.
31. ?>
32. <table>
33.     <tr>
34.         <td>Company</td><td>Quantity</td><td>Price</td><td>Total</td>
35.     </tr>
36.     <tr>
37.         <td><?print $sym; ?></td>
38.         <td>
39.             <div>
40.                 <!-- this allows the user to change the quantity and see the page dynamically change with a new subtotal price -->
41.                 <form action="<? $_SERVER["PHP_SELF"]; ?>" method="post">
42.                     <input type="text" name="quantity" id="quantity" size="5" maxlength="2" value="<?print($quantity); ?>"/>
43.                     <!-- for some reason, when I validate this html document, it keeps giving me the error that I need to place the "form" and
"input" tags within a div tag or object tag, but even when I do, it still gives me an error -->
44.                     <input type="submit" value="Update"/>
45.                 </form>
46.             </div>
47.         </td>

```

See my example from section about the potential for XSS attacks with forms that use `$_SERVER["PHP_SELF"]`: <https://github.com/codekiln/S75-Sections/blob/master/opencyclemap/map08.php>

```
48.         <td><? printf("%0.2f", $currentprice);?></td>
49.         <td><? printf("%0.2f", $total);?></td>
50.     </tr>
51. </table>
52. <div>
53.     <form action="addstocktotable.php" method="post">
54.     <input type="submit" value="Buy Stock!"/>
55.     </form>
56. </div>
57. <? } ?>
58.
59. <?
60.     require("footer.html");
61. ?>
```



```
1. <!--closing tags to end the html documents -->
2.     </div>
3.     </div>
4.     </body>
5. </html>
```

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4.   <head>
5.     <title>C$75 Finance</title>
6.     <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
7.     <!-- Used CSS Framework from Blueprint -->
8.     <link rel="stylesheet" href="blueprint/screen.css" type="text/css" media="screen" />
9.     <link rel="stylesheet" href="blueprint/print.css" type="text/css" media="print" />
10.    <!-- Added my own stylesheet to the Framework -->
11.    <link href="FINANCE_STYLE.css" type="text/css" rel="stylesheet"/>
12.    <!-- Javascript here. jQuery Validate is downloaded to help with validation features -->
13.    <script type="text/javascript" src="js/jquery-1.5.1.min.js"></script>
14.    <script type="text/javascript" src="js/jquery.validate.min.js"></script>
15.    <!-- implement validation through functions below -->
16.    <script type="text/javascript">
17.      //
18.      $(document).ready(function(){
19.        $("input.first").focus();
20.        var validation = $("#login").validate(
21.          {
22.            rules: {
23.              user: { required: true },
24.              password: { required: true }
25.            },
26.          });
27.        var validation2 = $("#register").validate(
28.          {
29.            rules: {
30.              fname: {required: true, minlength: 2},
31.              lname: {required: true, minlength: 2},
32.              user: {required: true, minlength: 4, maxlength: 20},
33.              password: {required: true, minlength: 6},
34.              email: {required: true, email: true}
35.            }
36.          });
37.        var validation3 = $("#quantity").validate(
38.          {
39.            rules: {
40.              quantity: {number: true, digits: true, maxlength: 2}
41.            }
42.          });
43.
44.        var validation4 = $("#gsymbol").validate(
45.          {
46.            rules: {
47.              quantity: {maxlength: 5},
48.            }</pre></div>
```

```
49.     });
50.
51.   });
52. //]]>
53. </script>
54. <?php
55.
56.     //function created to find the stock information for any company symbol given
57.     function findInfo($symbol){
58.         $link = "http://download.finance.yahoo.com/d/quotes.csv?s=".$symbol."&f=s1ld1t1clohgv&e=.csv";
59.         $yahprices = fopen($link, "r");
60.         while(($data = fgetcsv($yahprices, 1000, ",")) != FALSE) {
61.             if($data[1] <= 0.00){
62.                 return FALSE;
63.             }
64.             return $data;
65.         }
66.     }
67.     //function created to get the price of any specific company symbol (uses findInfo() with it)
68.     function getPrice($symbol){
69.         $info = findInfo($symbol);
70.         if($info === FALSE){
71.             print("There is no value for this symbol");
72.             break;
73.         }
74.         return $info[1];
75.     }
76.     //function created to get the current amount of money in the account of the user
77.     function getCurrentMoney($user){
78.         $sql = sprintf("SELECT * FROM Users WHERE user='$user'");
79.         $query = mysql_query($sql);
80.         if($query == FALSE){
81.             die("Could not query the database to access money amount");
82.         }
83.         while($array = mysql_fetch_array($query)){
84.             $money = $array['moneyamt'];
85.             return $money;
86.         }
87.     }
88.     //function to get the legal name of the user -> allows for greater personalization
89.     function getName($user){
90.         $sql = sprintf("SELECT * FROM Users WHERE user='$user'");
91.         $query = mysql_query($sql);
92.         if($query == FALSE){
93.             die("Could not query the database to access money amount");
94.         }
95.         while($array = mysql_fetch_array($query)){
96.             $fname = $array['first name'];
```

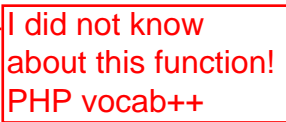
it would make more sense to put these functions into their own file

good modularity


```
97.         $lname = $array['last name'];
98.         $fullname = $fname. " ".$lname;
99.         print $fullname;
100.     }
101. }
102. ?>
103. </head>
104. <!-- This is the first part of every page in the site -->
105. <body>
106.     <div class="container">
107.         <div id="header" class="span-23">
108.             <h1>CS75 Finance</h1>
109.             <!-- the header here, which is implemented with CSS, was made by me with the program Paint -->
110.         </div>
111.         <div class="span-4">
112.             <ul id="navigation">
113.                 <!-- Navigation Bars -->
114.                 <li><a href="index.php" shape="rect">Home</a></li>
115.                 <li><a href="portfolio.php" shape="rect">Portfolio</a></li>
116.                 <li><a href="browse.php" shape="rect">Buy Stock</a></li>
117.                 <li><a href="logout.php" shape="rect">Log Out</a></li>
118.             </ul>
119.         </div>
120.         <div class="span-16">
121.             <!--main info-->
```

```
1. <!-- This is the Home Page for the site! -->
2. <?
3.     require("loggedin.php");
4.     require("mysql_connect.php");
5.     $user = $_SESSION["username"];
6. ?>
7.
8. <? require("header_validate.php"); ?>
9.
10.     <h1>Welcome to C$75 Finance!</h1>
11.     <h3>You currently have $<? print(getCurrentMoney($user));?> in your account!</h3>
12.     <p>
13.         Hi, <? echo getName($user); ?>!
14.     <br/>
15.     Check out your portfolio to see the status of your shares!
16. </p>
17.
18. <? require("footer.html"); ?>
```

```
1. <?
2.     require("header_validate.php");
3.     //start the session
4.     session_start();
5.
6.     $user = $_POST["user"];
7.     $pass = $_POST["password"];
8.     $lname = $_POST["lname"];
9.     $fname = $_POST["fname"];
10.    $email = $_POST["email"];
11.
12.    require("mysql_connect.php");
13.
14.    //should already be validated, but this is extra precaution
15.    if(isset($user) && isset($pass) && isset($lname) && isset($fname) && isset($email)){
16.        //check if the password is alphanumerical
17.        //i tried to implement this, but my code would malfunction when I uncommented
18.        // if(!ctype_alnum($pass)){
19.        //     die("Invalid Password!");
20.        // }
21.        //check if the password is only letters or only numbers
22.        if(ctype_alpha($pass) || is_numeric($pass)){
23.            $_SESSION['invalid'] = TRUE;
24.            header("Location: register.php");
25.        }
26.        $sql = sprintf("INSERT INTO `Users`(`user`, `password`, `last name`, `first name`, `moneyamt`, `email`) VALUES ('$user',AES_Encrypt('$pass', '$pass'), '$lname', '$fname', '10000', '$email')");
27.
28.
29.        //taken from w3schools.com/php/php_mysql_insert.asp
30.        //in case it does not insert a new table, it will show the error
31.        if(!mysql_query($sql,$connect)){
32.
33.            die("ERROR");
34.
35.        }
36.
37. ?>
38. <h2>
39. <?
40.     print("Your account has been created successfully!");
41. ?>
42. </h2>
43. <?
44.     }
45.     require("footer.html");
46. ?>
```



I did not know about this function! PHP vocab++


```
1. <?
2.     //this is a separate file to make it easier to check if the user is logged in on every page by linking all the pages to this page
3.     session_start();
4.     if($_SESSION["loggedin"] == FALSE){
5.         header("Location: login.php");
6.     }
7. ?>
```

```

1.  <?
2.      session_start();
3.      //connects to database
4.      require("mysql_connect.php");
5.
6.      $wrong = FALSE;
7.      //check username and password
8.      //if they inputted a user and password,
9.      if(isset($_POST["user"]) && isset($_POST["password"])){
10.         $user = mysql_real_escape_string($_POST["user"]);
11.         $pass = mysql_real_escape_string($_POST["password"]);
12.         //get sql
13.         $sql = sprintf("SELECT * FROM Users WHERE user='%s' AND password=AES_ENCRYPT('$pass', '$pass')");
14.
15.         //query for that input
16.         $result = mysql_query($sql);
17.         if ($result === FALSE){
18.             die("Could not query database");
19.         }
20.         //if theres a result, log the user in and redirect to home page
21.         if (mysql_num_rows($result) == 1)
22.         {
23.             //remember user is logged in
24.             $_SESSION["loggedin"] = TRUE;
25.
26.             //remember username
27.             $_SESSION["username"] = $user;
28.
29.             //bring user to home page
30.             header("Location: index.php");
31.         }else{
32.             $wrong = TRUE;
33.         }
34.     }
35. ?>
36. <? require("header_validate.php"); ?>
37.
38. <h1>Welcome to C$75 Finance!</h1>
39. <h2>Please log in to see your portfolio!</h2>
40.
41. <!-- if the user has tried to log in and their username or password is incorrect or the account does not exist, then a message of incorrect
    login is shown to notify the user -->
42. <? if($wrong === TRUE){ ?>
43.     <p id="incorrect">
44.         <? echo("Incorrect Login!"); ?>
45.     </p>
46. <? } ?>
47. <form id= "login" action= "<? echo $_SERVER["PHP_SELF"]; ?>" method="post">

```

good sanitization

```
48.     <p>
49.     Username
50.     <br/>
51.     <input class="first" type="text" name="user" id="user" size="35"/>
52.     <br/>
53.     Password
54.     <br/>
55.     <!-- type = password so that the characters cannot be seen on the screen for security reasons -->
56.     <input type="password" name="password" id="password" size="35"/>
57.     <br/>
58.     <input type="submit" value="Log In"/>
59.     </p>
60. </form>
61. <br/>
62. <!-- if they don't have an account, redirect them to register an account-->
63. <a href="register.php"><input type="submit" value="Register a New Account" id="register"/></a>
64.
65. <? require("footer.html") ?>
66.
```



```
1. <!-- LOG OUT -->
2.
3. <?
4.     require("header_validate.php");
5.
6.     //start the session
7.     session_start();
8.
9.     //reset all the session variables
10.    $_SESSION["loggedin"] = FALSE;
11.    $_SESSION["username"] = null;
12.    $_SESSION["tprice"] = 0;
13.    $_SESSION["price"] = 0;
14.    $_SESSION["gsymbol"] = null;
15.
16.    //delete any cookies that might have been created
17.    setcookie("user", "", time() - 3600);
18.    setcookie("pass", "", time() - 3600);
19.
20.    setcookie(session_name(), "", time() - 3600);
21.
22.    //destroy the session
23.    session_destroy();
24. ?>
25. <h1>You are now logged out!</h1>
26. <h3><a href="login.php">Home</a></h3>
27. <?
28.     require("footer.html");
29. ?>
30.
```

great - this works!

```
1. <?
2. //I made a separate file for connecting to MYSQL to reduce the amount of copy and paste on pages that needed to connect to MYSQL.
3.
4.
5. //taken from the login example from Malan in class, it checks if it can connect to the database server.
6. //if not, return error message
7. if(($connect = mysql_connect("192.168.56.50","jharvard","crimson")) == FALSE){
8.     die("Error! Unable to connect to the database!");
9. }
10.
11. // taken from login example from Malan, check if it can select database
12. if(mysql_select_db("jharvard_crystalfinance", $connect) == FALSE)
13.     die("Error! Unable to select the database!");
14.
15. ?>
```

great!



```
1.  <?
2.      //retrieve all necessary documents to make sure the user is logged in, connected to database, and has basic design needs
3.      require("loggedin.php");
4.      require("mysql_connect.php");
5.      require("header_validate.php");
6.
7.      //save the username of the current user
8.      $user = $_SESSION["username"];
9.  ?>
10.
11. <h1><? print($_SESSION["username"]);?>'s Portfolio</h1>
12. <h2>You currently have $<? print(getCurrentMoney($user));?> in your account!</h2>
13. <?
14.
15.      //find all history from that user
16.      $sqlselect = sprintf("SELECT * FROM History WHERE owner='$user'");
17.      $select = mysql_query($sqlselect, $connect);
18.      $array = mysql_fetch_array($select);
19.
20.      //because I need to test one of the arrays to see if the user has any stock, I need to save this array in case the user has bought stock,
    because then I can print it in the table later
21.      $arraysave = $array;
22.
23.      //if no history, then tell them they have not purchased any stock
24.      if($array === FALSE){
25.          echo "You have not purchased any stock yet!";
26.      }else{
27.          //if history, start creating a table of their stock history
28.          if($arraysave['quantity'] <= 0){
29.              $sym = $arraysave['company'];
30.              $sql = sprintf("DELETE FROM History WHERE owner='$user' AND company='$sym'");
31.              $result = mysql_query($sql);
32.
33.              if($result == FALSE){
34.                  die("Could not query the database");
35.              }
36.          }else{
37.  ?>
38.  <!-- inspired by http://www.w3schools.com/php/php_mysql_select.asp method of creating a table -->
39.  <table>
40.      <tr>
41.          <th>Company Name</th>
42.          <th>Price Bought</th>
43.          <th>Quantity</th>
44.          <th>Current Price</th>
45.          <th>Money Gain/Loss</th>
46.          <th>Sell All Stock</th>
47.      </tr>
```

```

48.     <tr>
49.
50. <?
51.         //because the user has bought a stock, I will display the array I archived with $arraysave
52.         echo "<tr>";
53.         echo "<td>".$arraysave['company']. "</td>";
54.         echo "<td>".$arraysave['pricebought']. "</td>";
55.         echo "<td>".$arraysave['quantity']. "</td>";
56.         $nowprice = (float)(getPrice($arraysave['company']));
57.         echo "<td>".$nowprice. "</td>";
58.         $change = $arraysave['quantity'] * ($nowprice - $arraysave['pricebought']);
59.         echo "<td>";
60.         printf("%0.2f", $change);
61.         echo "</td>";
62.         echo "<td>";
63.     }
64. ?>
65.     <form action="sellstock.php" method="post">
66.     <input type="hidden" name="company" id="company" value="<? echo $arraysave['company']; ?>"/>
67.     <input type="submit" name="sell" id="sell" value="Sell All Stock"/>
68.     </form>
69. </td>
70. <?     echo "</tr>";
71.
72.     //Print out all other arrays found that are identified with that user
73.     while($array = mysql_fetch_array($select)){
74.         if($array['quantity'] <= 0){
75.             $sym = $array['company'];
76.             $sql = sprintf("DELETE FROM History WHERE owner='$user' AND company='$sym'");
77.             $result = mysql_query($sql);
78.
79.             if($result == FALSE){
80.                 die("Could not query the database");
81.             }
82.         }else{
83.             echo "<tr>";
84.             echo "<td>".$array['company']. "</td>";
85.             echo "<td>".$array['pricebought']. "</td>";
86.             echo "<td>".$array['quantity']. "</td>";
87.             $nowprice = (float)(getPrice($array['company']));
88.             echo "<td>".$nowprice. "</td>";
89.             $change = $array['quantity'] * ($nowprice - $array['pricebought']);
90.             echo "<td>";
91.             printf("%0.2f", $change);
92.             echo "</td>";
93.             echo "<td>";
94.
95. ?>

```



```
96.         <!-- allow the user to decide to sell any of his stock -->
97.         <form action="sellstock.php" method="post">
98.         <input type="hidden" name="company" id="company" value="<? echo $array['company']; ?>"/>
99.         <input type="submit" name="sell" id="sell" value="Sell All Stock"/>
100.        </form>
101.    </td>
102. <?
103.     echo "</tr>";
104.
105.     }
106. }
107. echo "</table>";
108. }
109. ?>
110. <?
111.     require("footer.html");
112. ?>
```

```
1. -- MySQL dump 10.13  Distrib 5.5.14, for Linux (i686)
2. --
3. -- Host: localhost    Database: jharvard_crystalfinance
4. --
5. -- Server version  5.5.14
6.
7. /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8. /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9. /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10. /*!40101 SET NAMES utf8 */;
11. /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12. /*!40103 SET TIME_ZONE='+00:00' */;
13. /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14. /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15. /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16. /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17.
18. --
19. -- Table structure for table `History`
20. --
21.
22. DROP TABLE IF EXISTS `History`;
23. /*!40101 SET @saved_cs_client      = @@character_set_client */;
24. /*!40101 SET character_set_client = utf8 */;
25. CREATE TABLE `History` (
26.   `company` varchar(255) NOT NULL,
27.   `owner` varchar(255) NOT NULL,
28.   `pricebought` float NOT NULL,
29.   `quantity` int(11) NOT NULL
30. ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
31. /*!40101 SET character_set_client = @saved_cs_client */;
32.
33. --
34. -- Dumping data for table `History`
35. --
36.
37. LOCK TABLES `History` WRITE;
38. /*!40000 ALTER TABLE `History` DISABLE KEYS */;
39. INSERT INTO `History` VALUES ('LSI','chsieh',6.69,3);
40. /*!40000 ALTER TABLE `History` ENABLE KEYS */;
41. UNLOCK TABLES;
42.
43. --
44. -- Table structure for table `Users`
45. --
46.
47. DROP TABLE IF EXISTS `Users`;
48. /*!40101 SET @saved_cs_client      = @@character_set_client */;
```

this should be
decimal to avoid
rounding error

You should have
indexes and a
primary key set up.

```

49. /*!40101 SET character_set_client = utf8 */;
50. CREATE TABLE `Users` (
51.   `user` varchar(255) NOT NULL,
52.   `password` varchar(255) NOT NULL,
53.   `last name` varchar(255) NOT NULL,
54.   `first name` varchar(255) NOT NULL,
55.   `moneyamt` float NOT NULL,
56.   `email` varchar(255) NOT NULL,
57.   UNIQUE KEY `user` (`user`),
58.   KEY `email` (`email`)
59. ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
60. /*!40101 SET character_set_client = @saved_cs_client */;
61.
62. --
63. -- Dumping data for table `Users`
64. --
65.
66. LOCK TABLES `Users` WRITE;
67. /*!40000 ALTER TABLE `Users` DISABLE KEYS */;
68. INSERT INTO `Users` VALUES ('hihihi','! 3~&Â¸ ! Â³â€°Â°d ','hihi','hihihi',10000,'hihihi@hi.com'),('chsieh','ÃÃ³Â°Â±b+oÃÂ§&7M â€šÃ°3',
'Hsieh','Crystal',288.08,''),('hehel23',' Ã¸Â²2Ã-ÃÂ²â€œÃâ€ Ã-a6Ã \0 ','hsieh','crystal',10000,'hi@hi.com'),('nom123','Ãâ€ Ã...Ãµ\"
Ã@ Ã.Ãâ€œÃ.ÃÂ²','nom','nom',10000,'nom@nom.com'),('',' CÃ>cÃfÃÃ.Ã¸iâ€vâ, - ','',',',',',10000,''),('hamster','ÃÃ³Â°Â±b+oÃÂ§&7M â€šÃ°3',
'Hsieh','Crystal',7316.63,'hamster'),('SHEEP','7jÃ«mvÃ, GÃGÃ.â€Ãâ€ÃÃÃ','baa','baaa',10000,'sheep@sheep.com'),('broski',' â€ÃÃÃÃ\Z
Ã|IÃ;cÃ-]Ã.Ã,Æ','kk','kk',10000,'bro@go.com'),('pamp','Ã; Â²Â°@ a&Ãf Ã*ÃÃ.Ãâ€Ã','pamp','pamp',10000,'pamp@pamp.com'),('lamb','\\
Ã.Ã|ÃÃÃ-â€\nÃ.EiÃ¸ÃSOÃÃ%', 'lamb','lamb',10000,'lamb@lamb.com'),('nomnom','Ã½)p v_\0%Ã½hJaÃµDÃ- ','Hsieh','Crystal',10000,
'ljlljkljlkj@lkdj.com'),('kjlkljlkjl','UÃ> Ã;@~QÃ+G `YÃ+k~','jklkljlkjl','kljlkj',10000,'ljlljkljlkj@lkdj.com'),('awesome',
'Ã-< ÃÃÃÃ.Iâ€. |Ã±Ã.Ã³','hsieh','Crystal',10000,'awe@awe.com'),('hamham','ÃÃ+jÃ¼<syÃ²Ã- -ÃÃ ÃÃÃ_','hamham','hamham',10000,
'hamham@ham.com');
69. /*!40000 ALTER TABLE `Users` ENABLE KEYS */;
70. UNLOCK TABLES;
71. /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
72.
73. /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
74. /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
75. /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
76. /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
77. /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
78. /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
79. /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
80.
81. -- Dump completed on 2011-07-27 11:12:02

```

```
1. <?
2.     //start the session
3.     session_start();
4.     //connect to database
5.     require("mysql_connect.php");
6. ?>
7.
8. <? require("header_validate.php");
9.     session_start();
10. ?>
11.
12. <h1>Welcome to CS75 Finance!</h1>
13. <h2>Fill out the form to create an account!</h2>
14.
15. <!-- I created a simple form for the user to create an account, and it is being validated by jQuery. Unless all the required fields are filled,
    the form will not submit -->
16. <form id= "register" action= "input.php" method="post">
17.     <p>
18.         First Name<br/>
19.         <input class="first" type="text" name="fname" id="fname" size="35"/>
20.     </p>
21.     <p>
22.         Last Name<br/>
23.         <input type="text" name="lname" id="lname" size="35"/>
24.     </p>
25.     <p>
26.         Desired Username<br/>
27.         <input type="text" name="user" id="user" size="35"/>
28.     </p>
29.     <p>
30.         Desired Password<br/>
31.         <input type="text" name="password" id="password" size="35"/>
32.     </p>
33.     <!-- if user messes up, remind them of their mistake -->
34.     <p id="error">
35.         <? if($_SESSION['invalid'] == TRUE){
36.             print("Please make sure your password is alphanumeric!");
37.         } ?>
38.     </p>
39.     <p>
40.         Email<br/>
41.         <input type="text" name="email" id="email" size="35"/>
42.     </p>
43.     <p>
44.         <input type="submit" value="Register"/>
45.     </p>
46. </form>
47. <br/>
```

48.

49. `<? require("footer.html") ?>`


```
1. <?
2.     //retrieve all necessary documents to make sure the user is logged in, connected to the database, and has basic design needs
3.     require("loggedin.php");
4.     require("mysql_connect.php");
5.     require("header_validate.php");
6.
7.     //archive the username
8.     $user = $_SESSION["username"];
9.
10.    //if a user has clicked to sell one of their stock, check if the company was chosen
11.    if(isset($_POST["company"])){
12.
13.        //use real_escape_string to prevent mysql attacks
14.        $sym = mysql_real_escape_string($_POST["company"]);
15.
16.        //look through history to see if the owner had bought that stock before
17.        $sql = sprintf("SELECT * FROM History WHERE owner='$user' AND company='$sym'");
18.        $selectresult = mysql_query($sql);
19.
20.        //if there was no connection, send error message
21.        if($selectresult === FALSE){
22.            die("Could not query the database");
23.        }
24.
25.        //if a result is found, access the quantity of the stock the user owne
26.        //also find the current price of the stock
27.        while($array = mysql_fetch_array($selectresult)){
28.            $quantity = $array['quantity'];
29.            $currentprice = (float)(getPrice($sym));
30.            $subtotal = $quantity*$currentprice;
31.
32.            //add the subtotal of the sold stock to the current money amount of the user
33.            $total = $subtotal + getCurrentMoney($user);
34.
35.            //update the user's money account by how much they earned from selling stock
36.            $sql = sprintf("UPDATE Users SET moneyamt='$total' WHERE user='$user'");
37.            $usertable = mysql_query($sql);
38.
39.            if($usertable == FALSE){
40.                die("Could not query the database to change money amount");
41.            }
42.        }
43.
44.        //find the company row and delete it from the history table!
45.        $sql = sprintf("DELETE FROM History WHERE owner='$user' AND company='$sym'");
46.        $result = mysql_query($sql);
47.
48.        //if no connection, send error message
```

```
49.         if($result == FALSE){
50.             die("Could not query the database");
51.         }
52.         //else send message that the transaction was complete
53.         echo "The stock has been sold!";
54.     ?>
55.         <!-- remind the user how much money they currently own now -->
56.         <p>You currently have $<? print(getCurrentMoney($user));?> in your account!</p>
57.     <?
58.         }else{
59.             //in case a user gets to this page somehow, nothing will happen unless a company name is submitted.
60.             print("No company name was selected!");
61.         }
62.     require("footer.html");
63. ?>
```