Project 1: C$75 Finance

due Wed 7/25, noon ET

if you have already taken CS50 (or CSCI E-52), do not proceed; email David to discuss

Goals.

- Introduce you to SQL, MySQL, and phpMyAdmin.
- Introduce you to JavaScript.

Recommended Reading.

- `http://www.investopedia.com/university/stocks/`
- `http://www.php-editors.com/articles/sql_phpmyadmin.php`
- `http://www.w3schools.com/sql/`
- `http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html`
- `http://dev.mysql.com/books/mysqlpress/mysql-tutorial/ch10.html`

Academic Honesty

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed (*e.g.*, by some problem set or the final project). Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or accidentally shared in the course's virtual classroom) or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this course that you have submitted or will submit to another. Nor may you provide or make available your or other students' solutions to Project 0, Project 1, or Project 2 to individuals who take or may take this course (or CSCI E-75) in the future.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss problem sets with classmates, but you may not share code. You may also turn to the Web for instruction beyond the course's lectures and sections, for references, and for solutions to technical difficulties, but not for outright solutions to problems on projects. However, failure to cite (as with comments) the origin of any code or technique that you do discover outside of the course's lectures and sections (even while respecting these constraints) and then integrate into your own work may be considered academic dishonesty.

If in doubt as to the appropriateness of some discussion or action, contact the staff.

All forms of academic dishonesty are dealt with harshly. If the course refers some matter to the Administrative Board and the outcome for some student is disciplinary action, the course reserves the right to impose local sanctions on top of that outcome for that student that may include, but not be limited to, a failing grade for work submitted or for the course itself.

Grades.

Your code (CSS, HTML, PHP, *etc*.) will be evaluated along the following axes.

Scope. To what extent does your code implement the features required by our specification?
Correctness. To what extent is your code consistent with our specifications and free of bugs?
Design. To what extent is your code written well (*i.e.*, clearly, efficiently, elegantly, and/or logically)?
Style. To what extent is your code readable (*i.e.*, commented and indented with variables aptly named)?
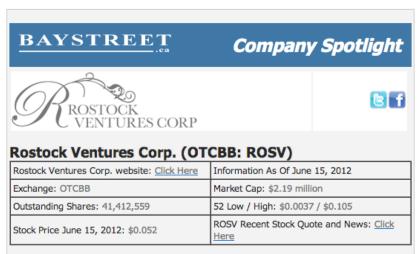
`investopedia.com.`

☐   If you're not quite sure what it means to buy and sell stocks (*i.e.*, shares of a company), surf on over to the URL below for a tutorial.

`http://www.investopedia.com/university/stocks/`

You're about to implement C$75 Finance, a Web-based tool with which you can manage portfolios of stocks! Not only will this tool allow you to check stocks' prices and portfolios' values, it will also let you buy (okay, "buy") and sell (fine, "sell") stocks!

`finance.yahoo.com.`

☐   Just the other day, I received the stock tip below in my inbox!

Let's get in on this opportunity now. Head on over to Yahoo! Finance at the URL below.

```
http://finance.yahoo.com/
```

Type the symbol for Rostock Ventures Corp, **ROSV**, into the text field in that page's top-left corner and click **Get Quotes**. Odds are you'll see a table like the below.

Wow, um, good thing I didn't buy. Anyhow, notice how Yahoo reports a stock's most recent (*i.e.*, "Last Trade") price ($0.03) and more. Notice, too, that Yahoo lets you download this data toward the bottom-right corner of that same page. Go ahead and click **Download Data (delayed)** to download a file in CSV format (*i.e.*, as comma-separated values). Open the file in Excel or any text editor (*e.g.*, Notepad or TextEdit), and you should see a "row" of values, all excerpted from that table. It turns out that the link you just clicked led to the URL below.

```
http://download.finance.yahoo.com/d/quotes.csv?s=ROSV&f=sl1d1t1c1ohgv&e=.csv
```

Notice how Rostock's symbol is embedded in this URL (as the value of the HTTP parameter called `s`); that's how Yahoo knows whose data to return. Notice also the value of the HTTP parameter called `f`; it's a bit cryptic (and undocumented), but the value of that parameter tells Yahoo which fields of data to return to you. For an unofficial list of the fields that can appear in that otherwise cryptic string, surf on over to the URL below.

```
http://www.gummy-stuff.org/Yahoo-data.htm
```

It's worth noting that a lot of websites that integrate data from other websites do so via "screen scraping," a process that requires writing programs that parse (or, really, search) HTML for data of interest (*e.g.*, air fares, stock prices apparently, *etc.*).[1] Writing a screen scraper for a site tends to be a nightmare, though, because a site's markup is often a mess, and if the site changes the format of its pages overnight, you need to re-write your scraper.

---

[1] See `https://manual.cs50.net/Screen_Scraping` for an XPath-based approach.

Thankfully, because Yahoo provides data in CSV, C$75 Finance will avoid screen scraping altogether by downloading (effectively pretending to be a browser) and parsing CSV files instead. Of course, Yahoo's quotes aren't actually real-time but delayed (likely by a few minutes), but, hey, they're free.

BitBucket.

☐ So that you have a place to store revisions of this project privately, head to `https://bitbucket.org/` and create a **project1** repo!

Implementation.

☐ Your mission for this project is to implement C$75 Finance, a website that allows users to manage portfolios of stocks. The overall design and aesthetics of this site are ultimately up to you, but we require that your site meet some requirements. All other details are left to your own creativity and interpretation.

Feature Requirements.

☐ 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).

☐ Your site must allow a user to register for an account. A user's username must be a syntactically valid email address.[2] A user's password must be at least six characters, and it cannot be entirely alphabetic or entirely numeric. 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 its "Last Trade" price) for a stock by providing its symbol.

☐ Your site must allow a user to buy shares of a stock by providing its symbol; your site must allow a user to buy more shares of a stock that he or she already owns. A user may not buy fractions of shares.

☐ 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. A user may not sell fractions of shares.

☐ Your site must allow a user to check the current value of his or her portfolio (*i.e.*, his or her cash plus his or her stocks' value based on its "Last Trade" price).

☐ 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

---

[2] Rather than validate email addresses per RFCs 2821 and 2822, you may, for simplicity, 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.

admonishing the user with an alert) or prevent them from typing anything non-numeric at all.

☐ 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).

☐ Your site must perform rigorous server-side error-checking. Under no circumstances should we be able to crash your site or induce unreasonable behavior. Letting us sell more shares than we own is not, shall we say, reasonable. We will bang on your code and try to find faults; do not let us succeed.

Technical Requirements.

☐ You're welcome to develop your site on any computer using any IDE or text editor, even without using the CS50 Appliance, but you must ultimately ensure that it works within the CS50 Appliance at a URL of `http://project1/` when installed in `/home/jharvard/vhosts/project1/`.

☐ Only files that should be web-accessible should live in `project1/html/`; everything else should live in `project1/` or some (other) subdirectory therein.

☐ You must store users' data in a MySQL database (which you can create within the CS50 Appliance via phpMyAdmin).

☐ You must use PHP's PDO library to talk to MySQL (not PHP's `mysql_` functions). See `http://www.phpro.org/tutorials/Introduction-to-PHP-PDO.html` for reference.

☐ Your site must pull its real-time data (prices, *etc*.) from Yahoo! Finance by parsing its CSVs; you may find `http://us3.php.net/fgetcsv` of interest.

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

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

☐ You should avoid inefficiencies and redundancies in your database tables (*e.g.*, by keeping them in third normal form).

☐ You should avoid redundant HTML; factor out markup common to multiple pages using PHP functions or "templates" (*i.e.*, PHP files that you `require` in others).

☐ Your markup language should be valid (or "tentatively" valid) HTML5, as per `http://validator.w3.org/`, unless some feature of your site requires otherwise (for the sake of some browser); explain in HTML comments any intentional invalidities. Your HTML should also be as pretty-printed as possible. Your CSS need not be valid.

☐ Your PHP must be extensively commented and be as pretty-printed as possible.

☐ You may use a WYSIWYG editor to generate HTML and/or CSS that you would like to use in your site.

☐ If you integrate third-party CSS or JavaScript libraries into your project, cite their origin with comments.

☐ If you incorporate or adapt snippets of PHP code from the Web into your project (*e.g.*, examples from `php.net`), cite the code's origins with comments.

☐ If you incorporate images from the Web into your project, cite the images' with comments.

☐ Your website must appear and behave the same on the latest versions of at least two of these browsers:

    ☐ Chrome
    ☐ Firefox
    ☐ Internet Explorer
    ☐ Opera
    ☐ Safari

Exit Interview.

☐ Once done with your site, put together a readme in a file called `README` that lives in `project1/`.

Treat this readme as your opportunity not only to explain but to justify your design decisions. Tell us why you modeled your database tables as you did. Tell us why you chose, say, select menus over radio buttons for some feature. Tell us with which two (or more) browsers we should evaluate your site. And give us an overall sense of how your site works (*e.g.*, tell us which files do what). But still be succinct; keep this readme to just a few paragraphs in length.

How to Submit.

☐ Just prior to this project's deadline, instructions for submitting your work will be announced at `https://www.cs75.net/`. Be sure to look for those directions and then submit your work prior to this project's deadline.