COMP 3512 Assignment #2

Due Wednesday April 14th at noon
Version 2, March 26, Changes in yellow

Overview

This assignment is a group project that expands your first assignment in functionality and scope. It is broken into several milestones with different dates. The milestones are in place to ensure your group is progressing appropriately.

Some of the specific details for some of the milestones and pages will develop over time; that is, this assignment specification is a living document that will grow over the next several weeks.

Composition

You can work in groups of three or four or five for this assignment. It is also possible to work individually or in a pair, but I do discourage it; please talk to me about this if you are planning on working by yourself or as a pair. Don't ask me for a group of six or more: the answer will be NO.

If working in a group, each member needs to take responsibility for and complete an appropriate amount of the project work. **Be sure to consult the instructor at least one week prior to any due date if your group is experiencing serious problems in this regard (but by then it's almost too late). If you are having problems two weeks before the due date with a group member, then there is probably a sufficient amount of time to rectify the problems!**

I feel foolish saying this in a third-year university course, but it is your responsibility to read all the assignment instructions thoroughly and carefully. If you are unclear about something, ask me. But before you do, read the material in question again!

Files

You will be able (eventually) to find the most recent version of the database scripts and all the images at the GitHub repo for the assignment:

`https://github.com/mru-comp3512-archive/w2021-assign2`

Note: github repo is now ready (March 17)

Periodically, I may upload a revised version of the database script on GitHub. If I do so, simply re-running the script file (e.g., using the MySQL command `source stocks-3rd.sql`) will remove your previous tables and re-create and re-populate them).

Source Code Approaches

You need to have *"a single source of truth"* when it comes to your source code. That is, there must be a "master" location that contains the definitive version of your source code. There are two approaches you can take for your source code in this assignment.

- **Team Leader**. In this case, the team leader will take responsibility for maintaining the most up-to-date code.
    - *Advantages*:
        - simplest approach (don't need to learn git)
    - *Disadvantages*:
        - Better hope team leader doesn't make a mistake,
        - If that person is not available, then where do you get the most recent code?
- **GitHub**. In this case, there will be a single github repo that will be the "owner" of the source code. Each member will be added as collaborators to the repo. Members will work locally using whatever technology they want. They will then push code to this single github repo when needed. Alternately, members will have to pull the current version of their branch before coding or each member will have to have different branches and then these branches will have to be merged. This is the workflow you will see in industry … it's complicated but employers today expect their prospective employers to already know it.
    - *Advantages*:
        - group members can use whatever technology they want for editing and running,
        - any github pushes will be "credited" to the person who did the push
        - gain very useful real-world experience working with git+github branches and merges.
        - Most hosting environments work with github
    - *Disadvantages*:
        - Everyone will have to learn and deal with git branching + merging (which can sometimes be frightening).

GitHub

Regardless of the source code approach you take, each group member will need their own Github account. You will also need to create a private repo on Github for the assignment. You have a couple of ways to do this. One way would be for one member in your group to create it (they would thus "own" the repo), and then add other members as collaborators.

The free GitHub account doesn't allow private repos; if you sign up for the Student Developer Pack (https://education.github.com/pack) you can have free private repos while a student. The other way is to email me, and I can create a private repo under our department's github organization (https://github.com/MountRoyalCSIS) for your group. You would need to supply the github names or emails for each member. You can also decide to use a public repo.

You will want to push out updates fairly frequently (1-2 times a day when working on it, or more frequently if lots of work is being done). I will examining the commits when marking. You can push your content to GitHub via the terminal, using the following commands (not the stuff in square brackets though, as those are comments):

```
git init     [only need to do this one command once for your assignment]
git add *
git commit -m "Fixed the rocket launcher"     [alter message and name as appropriate]
git remote rm origin [just in case your cloud9 workspace still linked to my github]
git remote add origin https:... your-repo-url.git      [specify URL of github repo … do this just once]
git push -u origin master   [login using your own individual github credentials]
```

For more information about Git and GitHub, read pages 571-577 of textbook (2nd Edition). There are many online guides to git (for instance, https://guides.github.com/introduction/git-handbook/).

Grading

The grade for this assignment will be broken down as follows:

Visual Design and Usability	15%
Programming Design and Documentation	08%
Hosting	07%
Functionality (follows requirements)	65%
Milestones	05%

Submitting and Hosting

You will be using JavaScript, PHP, and MySQL in this assignment. Eventually this will mean your assignment will need to reside on a working host server. In the labs, you made use of XAMPP on your local development machine as both host server and as a development environment. While easy, it means no one but you (or your group members and myself) can ever see your work. If you ever want to use your assignment as a portfolio piece, it needs to be on a working host server. Static hosts used in the first assignment will not work for this one.

For this assignment, these are your hosting/submitting options:

- Heroku. It has a free tier and is a popular hosting option that integrates nicely with github. It requires that at least one person register with heroku and install its CLI software on their computer. That person then uses a few command line instructions to copy software from github repo to the heroku servers. Some additional commands are needed to add mysql (or MariaDB which is the same thing) via third-party marketplace. Lots of online instructions available (try searching for "deploy PHP mysql heroku"). Will take some time to set up. I will provide a lab that illustrates this option.
- Digital Ocean. Similar to Heroku. You can also find free credits via Git Education program.
- Inexpensive Hosting. These usually won't be free (often about $5/month), but some are free. Once setup, your site can live forever. Possibilities include epizy, infinityfree and bluehost.
- Google Cloud Platform. In this case, you will setup a virtual LAMP stack on a Compute Engine (a virtualized server). Will be likely too expensive over time, but you can get free credits that will last for a few months. I will provide a lab that illustrates this option.
- Amazon Web Services (AWS). Similar to Google's offerings. Will be likely too expensive over time, but you can get free credits that will last for a few months. Probably easiest approach is launching LAMP stack on AWS Lightsail (first month for lightsail is free).
- Make use of a Docker LAMP container and then deploy container on any host environment that supports Docker. This will provide experience in the most important DevOps platform and will thus be excellent to put on your resume.

This step is going to take some time, so don't leave it till the very end. I would recommend assigning this task to someone in group who feels comfortable with operating systems and networking; because this takes time, you should ask that person to do a bit less programming. **The hosting should be arranged and tested (i.e., verify PHP and MYSQL work) a week before the assignment is completed!!!**

When your hosting is working and the assignment is ready to be marked, then send me an email with the following information:

- The URL of the home page of the site on your hosting platform.
- The names of the other people in the group
- The URL of the github repo so that I can mark the source code. If your repo is private, then add me as a collaborator.

Emergency Database Hosting

If you are having no luck at all getting your MySQL database working, you can try (with a small mark penalty), to use either the SQLite database, or connect to my MySQL database. If you chose the former, email me to request it.

If you chose the latter, email me at least a day before the deadline so I can create a temporary account for you and then send you the connection string and user name information.

Milestones

You will need to implement certain functionality by specific dates. You will show me your completed milestones in the lab on the dates shown below.

Milestone 1.
Due March 19.

You must decide on your group members and set-up your github accounts. Send me an email with each group member's name and the URL for each member's personal github page. Also provide URL for the main github page for the assignment.

If you don't have a group by the morning of March 19, please email me and let me know; I will try to get you into a group during the lecture.

Milestone 2.
Due March 31 (might change).

Create the following API in PHP.
`api-companies.php` – with no parameter, return a JSON representation of all companies. If supplied with `symbol` parameter, then return just JSON data for single specified company.

Data Files

Data in this assignment will come from MySQL. You will be provided with the import scripts for them. I will also provide a SQLite version in case you need it.

Functionality

Your second assignment will replicate some of the functionality from assignment 1, but is not a single-page application; instead multiple pages in PHP are needed. Some of these pages also use JavaScript but others don't.

Mobile-First Design: Your pages must be designed to be optimised for a mobile platform. That is, I will be mainly testing and evaluating this assignment using a browser with a small browser width, say 400-500 pixels wide, as if I am viewing your assignment on a cell phone browser. You should still use media queries as I will also test one or two of the pages briefly on a full-size browser to ensure it looks reasonable at the larger sizes.

Header: Each page will have a header at the top that will contain some type of logo and a "hamburger" menu (that is, a responsive navigation bar). There are many examples online of the necessary CSS and JavaScript for this to work. If you make use of CSS+JavaScript you find online, please be sure to document this in the About page. The header/hamburger menu should have the following links/options:

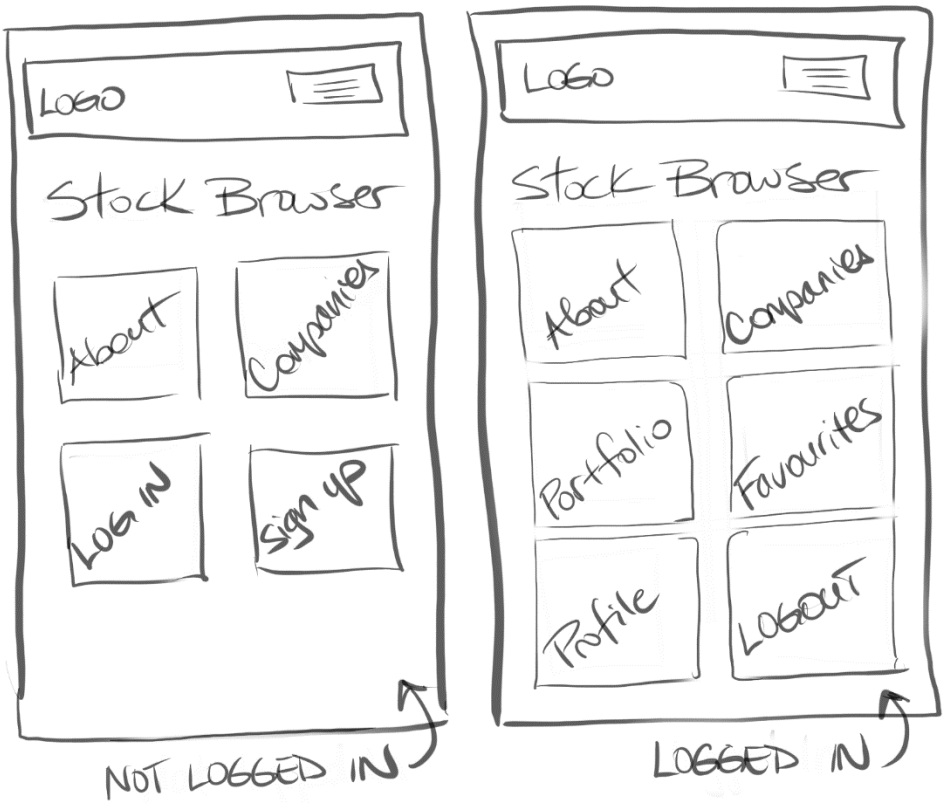
- Home
- About
- Companies
- Portfolio. Should only be available once user is logged in.
- Profile. Should only be available once user is logged in.
- Favorites. Should only be available once user is logged in.
- Login/Logout. If user is not logged in, then the option should be Login; if user is already logged in, then option should be Logout.

The same options and rules apply to the home page (except no home link there).

Home: The main page for the assignment. This file **must** be named `index.php`. This must have the functionality shown in the following sketch. It shows the options when user hasn't logged in and the options after a user has logged in.

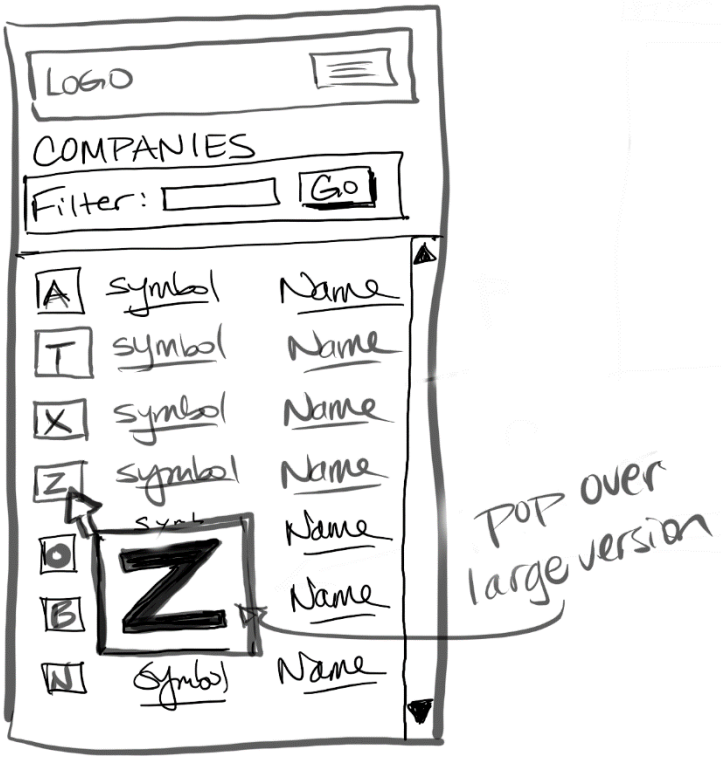
Note: the sketches is this assignment specification are meant to show functionality, not design. So in this case, your page needs a header, title, and some number of links/buttons. Here I've shown them as boxes, but you could do them as rectangles, circles, icons, simple links, etc. Make your pages look nicer than these sketches!

I have called the site Stock Browser, but you are welcome to name your site anything you'd like.



Companies Page: will display list of all companies. This file must be named `list.php`. This list must be populated in JavaScript from an API you create named `api-companies.php`. Most of this functionality is similar to what you did in assign 1.

Your home page must display a loading animation while the API data is being retrieved. This simply requires using javascript to show a gif animation before the fetch, and use javascript to hide the element after the fetch receives its data.



In the list, it should display a small version of the logo, the company stock symbol, and the company name. The symbol and name should be links to `single-company.php` with a query string containing the relevant stock symbol.

When the user moves the mouse pointer over the square logo of a company, use JavaScript to dynamically display a larger pop-over version (say a width of about 200-400 pixels) of the logo; this pop-over version should move with the mouse and then disappear after you move the mouse outside of the thumbnail. This will require using JavaScript setting event handlers for the `mouseenter`, `mouseleave`, and `mousemove` events of each logo.

In the `mouseenter` event, simply unhide a <`div`> and dynamically add an <`img`> to that <`div`> with the larger version of the image. In the `mouseleave` event, re-hide the <`div`> with the image. For the `mousemove` event, the event argument can be used to retrieve the current x,y position of the mouse pointer; you can then programmatically set the CSS `left` and `top` properties of the <`div`> with the larger image.

Your list should be sorted ascending by symbol. You also need a way to filter the list by name. In the first assignment, you filtered the list on name using JavaScript. You can simply reuse your JavaScript filter from assign 1. Also provide way to easily “unfilter” the list.

Single Company Page: This page must be named `single-company.php`. It will display information for a single company. Which company? The company whose stock symbol was passed as a query string to the page.

This page will allow the user to **view** the following fields from the Company table for the specified company: symbol, name, sector, subindustry, address, exchange, website, description (not in diagrams only some fields are shown just for space considerations: your page needed to include all 8 fields).

NOTE: the illustrations below are a little inaccurate: I didn't want to redraw them!

The page contains two buttons/links: Favorite and History (shown as Month in diagram), which will have the functionality described below.

Favorite – add this company to the user's favorites. This will be a session-based list. When the user does click the Add to Favorites link/button, the site will update the PHP session value, and then redirect to the **Favorites** Page.

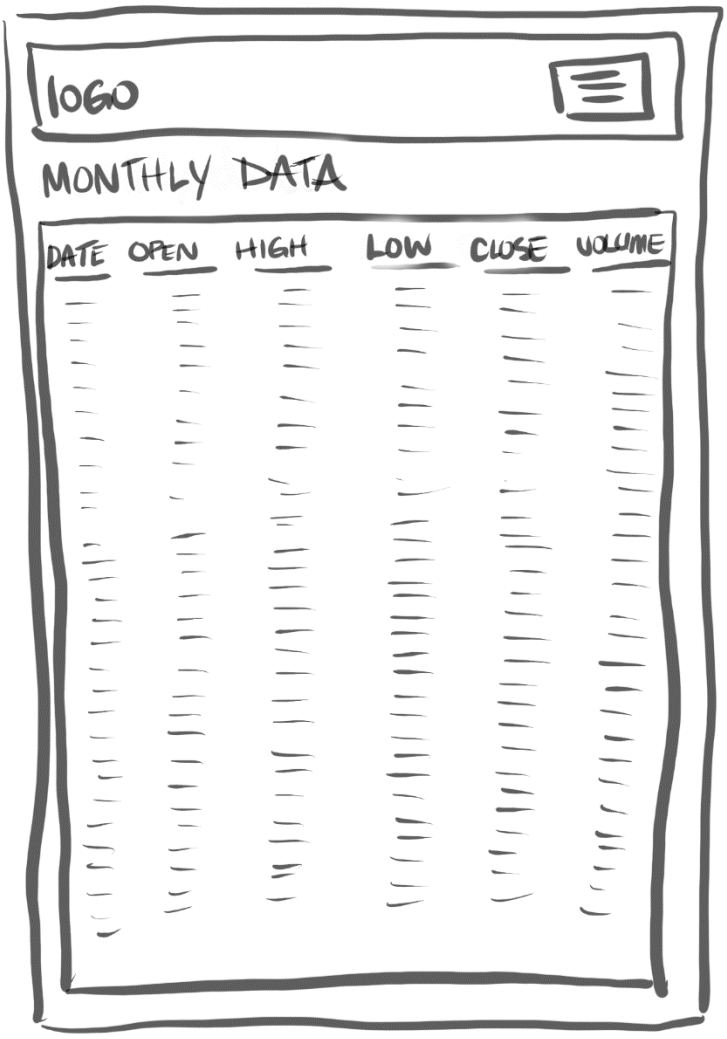
History – display the Stock History Data page.

Note: I have shown the buttons/links at the bottom of the page, but you can put them where you like. These sketches are just meant to show functionality not design.

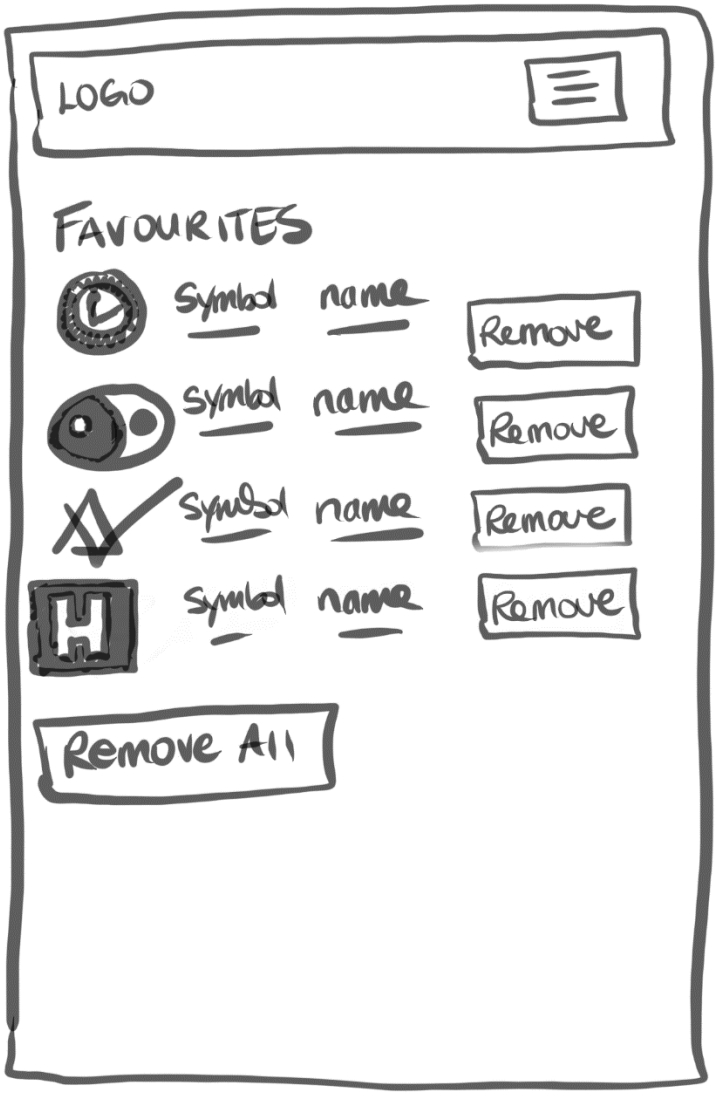
Stock History Page: This page should be named `history.php`. It should display the history data for the company/stock specified by a query string. This data is in the `history` table.

This page will have similar functionality as the **Stock Data** section from assignment one, except the data is being fetched and output entirely using PHP instead of JavaScript.

The headings at the top of each column should be clickable. When the user clicks on a column heading, the data will be redisplayed so that it is sorted on the value just clicked. When first displayed, sort by date. This sorting will happen in PHP not JavaScript, so these headings will have to be HTML links with querystrings indicating sort field.



Favorites Page: will display list of logged-in user's favorited companies (implemented via PHP sessions). This page must be named `favorites.php`. The user should be able to remove companies singly or all at once from this list. This page will be entirely in PHP. The symbol and company links should be to the appropriate single company page.



About Page: Provide brief description for the site, by mentioning class name, university, professor name, semester+year, and technologies used. Also display the names and github repos (as links) for each person in the group. Add a link to the main assignment github repo.

Just like in assignment 1, be sure to provide credit for any external/online CSS/JavaScript/PHP you have made use of in this assignment.

Profile Page: Not enough time for this one. Simply have a page that says Coming Soon or something like that.

~~allows logged-in user to view her (to simplify your coding, all the users have the same gender in this data set) personal information (first name, last name, city, country, email). Name this page `profile.php`. As well, display the user's profile picture. To do so, use a random image from the website randomuser.me. For instance:~~

~~`https://randomuser.me/api/portraits/women/60.jpg`~~

~~You can simply replace the number in the above sample URLs with the user`id` field value.~~

Portfolio Page: allows user to view her stock portfolio. The portfolio consists of multiple records from the `portfolio` table (though for a brand new user, there will be no portfolio records so be able to gracefully handle that case with an appropriate message). Each record contains a user id, stock symbol, and the amount of shares they own of that stock. You must display the logo, symbol, company name, # shares owned, current close price per share (use the close value from the last history record for that stock), and the value of those shares plus a grand total of their entire portfolio. Symbol and name should be links to the `single-company.php` page.

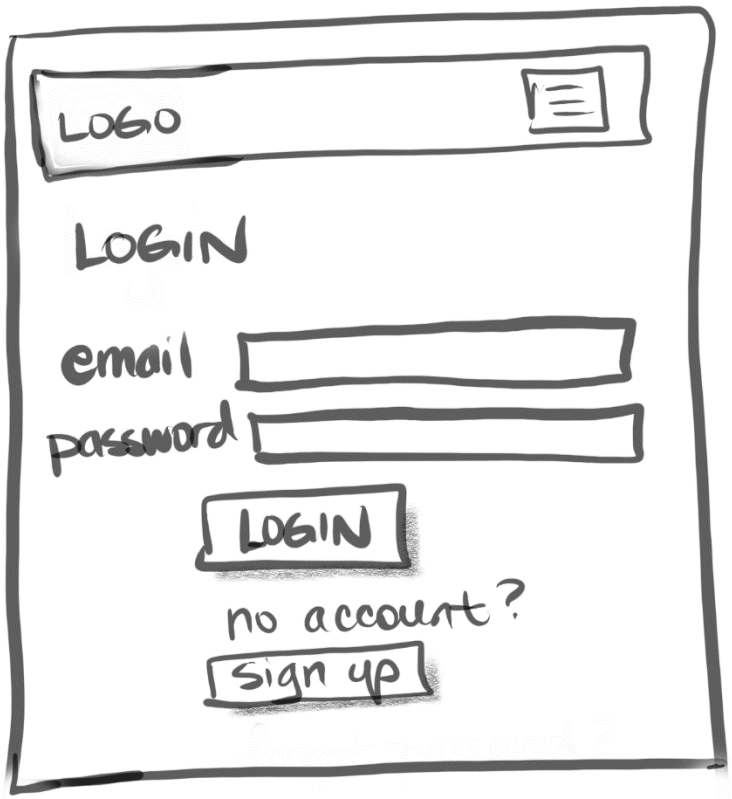
Login Page: will display a login screen. Some of the information below about hashing algorithms will be explained in class when we cover security.

Your database has a table named `users`. It contains the following fields: `id`, `firstname`, `lastname`, `city`, `country`, `email`, `password`, `salt`, `password_sha256`.

The actual password for each user is **mypassword**, but that's not what is stored in the database. Instead, the actual password has been subjected to a bcrypt hash function with a cost value of 12. The resulting digest from that hash function is what has been stored in the password field. That means to check for a correct login, you will have to perform something equivalent to the following:

```
// 1. check if email exists
...
// 2. check if entered password matches password digest saved in database
if ( password_verify( $_POST['password in form'], $data_from_table['password field' ) )
{
   // 3. you have a match, log the user in
} else {
   // no match set error message
}
```

For successful matches, you will need to preserve in PHP session state the log-in status of the user and the user's id. As well, after logging in, redirect to the **Home** page.



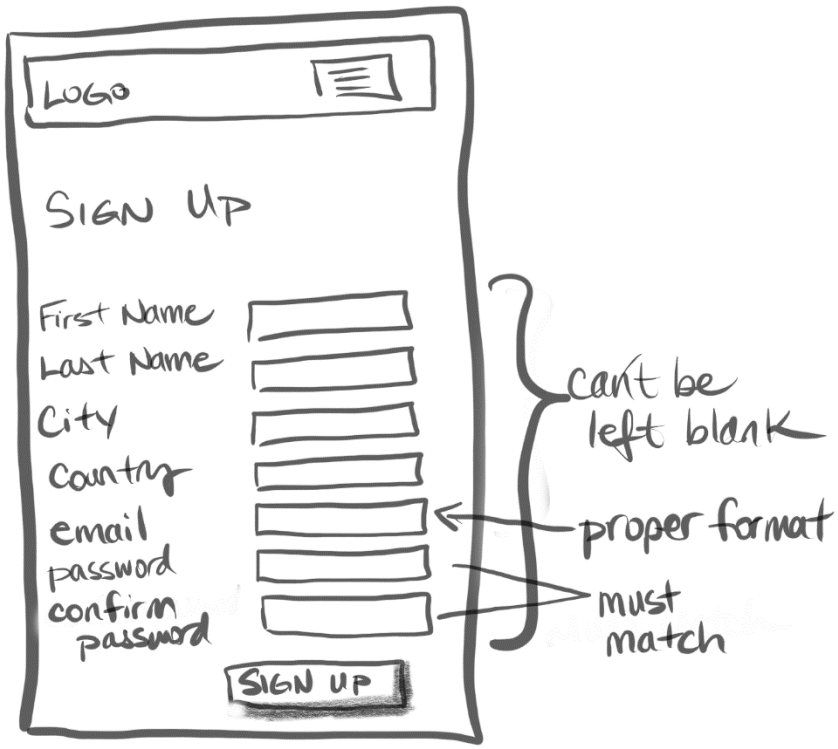
Registration/Signup Page: Not enough time for this one. Simply have a page that says Coming Soon or something like that.

~~will display a registration form. Perform the following client-side validation checks using JavaScript: first name, last name, city, and country can't be blank, email must be in proper format (use regular expressions), and the two passwords must match and be at least 8 characters long. Be sure to display any error messages nicely.~~

~~Once the user clicks Sign Up/Register button, you will have to check to ensure that the email doesn't already exist in the `users` table. If it does, return to form (with the user's data still there) and display appropriate error message. If email is new, then add new record to the Users table, log them in, and redirect to **Home** page.~~

~~You will **not** store the password as plain text in the database. Instead, you will save the bcrypt digest with a cost value of 12:~~

```
$digest = password_hash( $_POST['pass'], PASSWORD_BCRYPT, ['cost' => 12] );
```



Logout. If the user is logged in, then modify your session state information so your program knows a user is no longer logged in. After logging out, redirect to **Home** page.

Recommendations

1. Start by completing Lab14a.
2. This is a data-driven site, so I would recommend beginning by constructing the table gateway classes in PHP: use the ones in Lab14a as your model.
3. Implement the second milestone.
4. Implement the basic design and styling. Design and usability is worth 15% so don't ignore this!
5. Implement basic functionality on list page (add in JavaScript popover later).
6. Implement single company page and history.
7. Get hosting working.
8. Implement favorites.
9. Implement the login.
10. ~~Implement the registration page.~~
11. Implement home page and about us.

Steps 2-4 and 5-11 can be done asynchronously and worked on by different team members.