

COMP 3512 Assignment #2

Due Saturday April 6th at midnight

Version 1.1, March 6

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 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.

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.**

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 can 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/w2019-assign2>.

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 companies-small-2019.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.

- **Cloud9.** In this case, someone will create a Cloud9 workspace in their account, and then grant RW access to other members of their team. For backup purposes, members will push code changes to a GitHub repo.
 - *Advantages:*
 - simplest approach by far
 - members can edit the same file at same time and Cloud9 handles it.
 - *Disadvantages:*
 - all GitHub pushes will be credited to the owner of the cloud9 workspace regardless of who does it,
 - everyone in group has to use Cloud9.
- **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 or in their own cloud9 workspace. 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 often expect their prospective employers to know it.
 - *Advantages:*
 - group members can use whatever technology they want for editing and running (no cloud9 needed),
 - 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.
 - *Disadvantages:*
 - no simultaneous editing of files,
 - this is a much more complicated workflow,
 - everyone will have to learn and deal with git branching+merging.

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 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 examine the commits when marking. You can push your content from Cloud9 to GitHub via the terminal, using the following commands:

```
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	10%
Hosting	10%
Functionality (follows requirements)	65%

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 Cloud9 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.

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

- github + cloud9. This is certainly the easiest, but no one else but your group (and me) will be able to see your work. I will be able to mark your assignment but it will receive the minimum mark (1/10) for the hosting component of the assignment. Use this as a last resort if struggling with hosting or you run out of time.
- 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.
- 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 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. If interested in this route, let me know and I can request credits for you.
- 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 Docker LAMP container and then deploy container on any host environment that supports Docker.

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 can be arranged and tested (i.e., verify PHP and MYSQL work) well 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. If you decided to stick with the cloud9 option, then simply add me to the workspace with RW access.
- 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.

Milestones

You will need to implement certain functionality by specific dates. These won't be marked but you will incur penalties (deductions in marks) if the milestone is not completed on time. These will be periodically updated. You will show me your completed milestones in the lab on the dates shown below.

Milestone 1. You must decide on your group members, set-up your github accounts, and create your workspace in c9 (if you are using it) that will be used collectively by your group. Create a simple version of your About page (see description below). This page for this milestone needs no formatting. Simply indicate each the group member's name and github page (as a link). Also provide a link to the main github page for the assignment. *Due March 5-8.*

Milestone 2. Complete a preliminary design and implement it with CSS. Your design needs to be workable mainly at mobile size, and workable at desktop sizes. This means you will need to make use of media queries. See textbook pages 290-293 for more information. In Lab 12, look at the provided CSS for lab12-test01 to see simple media queries at work. *Due March 12.*

Milestone 3. Create the following API in PHP (place it in folder named `services`).

Due March 18.

`services/companies.php` – with no parameter, return JSON representation of all companies. If supplied with `symbol` parameter, then return just JSON data for single specified symbol.

Data Files

Images for the logos have been provided.

User, Company, and Portfolio data will come from MySQL. You have been provided with the import scripts for each of these.

External price data comes from <https://api.iextrading.com>.

Random user pictures come from <https://randomuser.me>.

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:

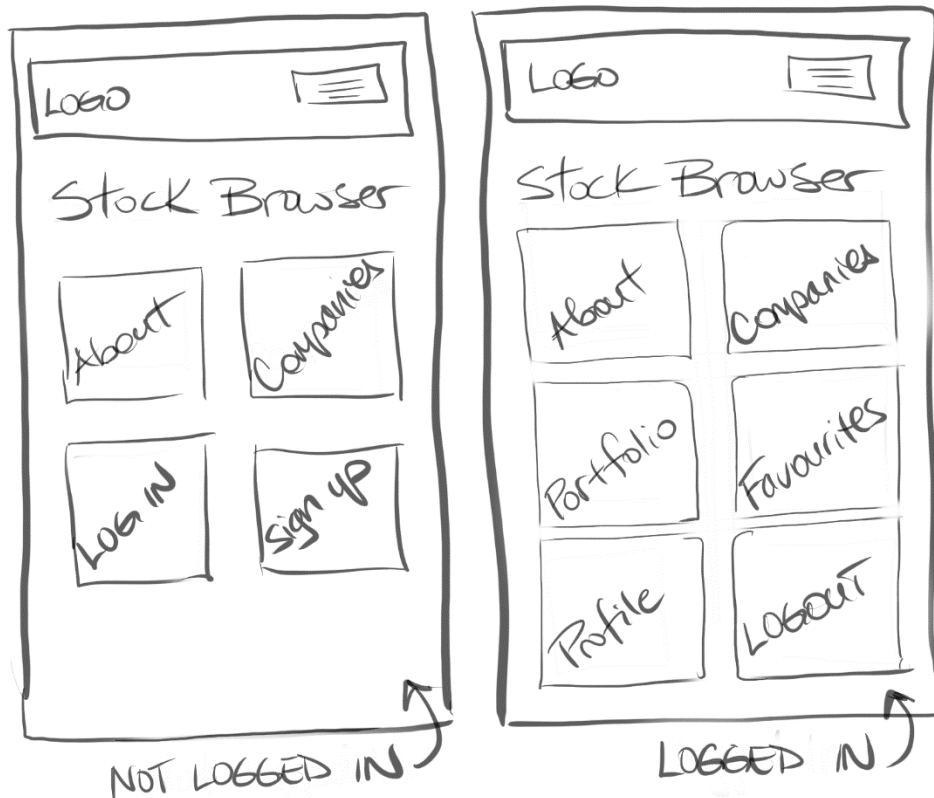
- Home
- About
- Companies
- Portfolio. Should only be available once user is logged in.
- Profile. Should only be available once user is logged in.
- Favorites. Should only be available once user is logged in.
- Login/Logout. If user is not logged in, then the option should be Login; if user is already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

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.

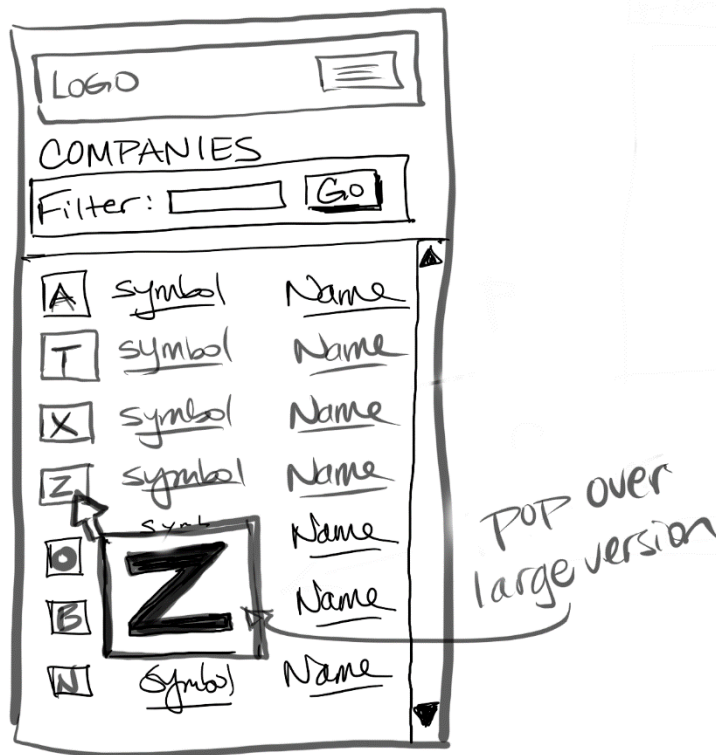
Note: the sketches in this assignment specification are meant to show functionality, not design. So in this case, your page needs a header, title, and some number of links/buttons. Here I've shown them as boxes, but you could do them as rectangles, circles, icons, simple links, etc. Make your pages look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

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 `` 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 symbol using JavaScript. In this one, you will filter via JavaScript on the name. 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** or **edit** the following fields from the Company table for the specified company: symbol, name, sector, subindustry, address, exchange, and website (not in diagrams only some fields are shown just for space considerations: your page needed to include all 7 fields).

When in view mode, the page will also contain three buttons/links: Edit, Favorite, and Month, which will have the functionality described below. Note, when in edit mode, the page will display a company information in a data entry form, but will not display these three buttons/links. There will be a Cancel button in the data entry form that will return to view mode without saving data.

Edit – display a data entry form populated with the current company’s data. Note this is **not** a separate page. The user **cannot** change the symbol. If the user saves/submits this form, then update the data in the database. After updating the data, return to the view mode but display a message somewhere on the page that the data has been updated. Though the sketches don’t illustrate this, when in edit mode, the input boxes should be populated with the current values.

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.

Month \$ – display the Stock Monthly Data page.

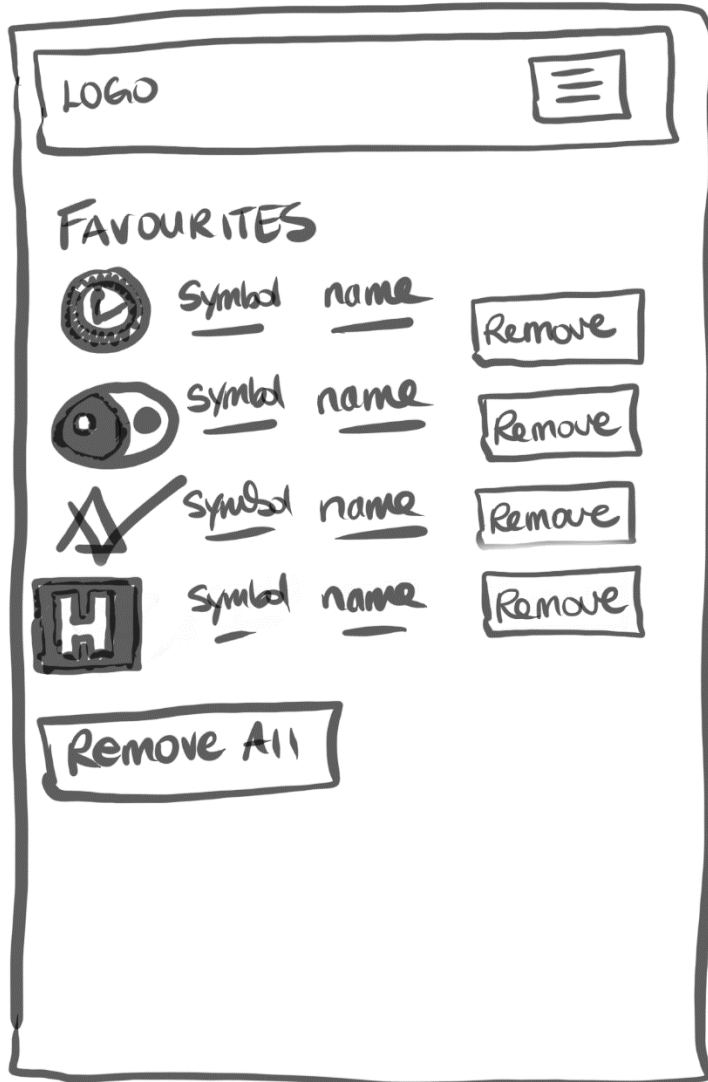
Note: I have shown the three buttons/links at the bottom of the page, but you can put them where you like. These sketches are just meant to show functionality not design.

The image contains two hand-drawn sketches of a web page for a single company, labeled 'view mode' and 'edit mode'.

view mode: The sketch shows a header with a 'Logo' and a menu icon. Below is the 'Company' section with an 'amazon' logo and 'AMZN' stock symbol, and the text 'Amazon.com Inc'. The 'Sector' is 'Consumer', 'Sub Industry' is 'Direct, Retail', and 'Address' is 'Seattle, Washington'. At the bottom are three buttons: 'edit', 'add to favourites', and '\$ month'.

edit mode: The sketch shows the same header and company information. The 'Sector', 'Sub Industry', and 'address' fields are now input boxes. At the bottom are 'save' and 'cancel' buttons. An arrow points from the 'edit mode' label to the 'cancel' button.

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: 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 `userid` field value.

Portfolio Page: allows user to view her stock portfolio. The portfolio consists of multiple records (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 stock symbols and the number of shares they own of that stock. You must display the logo, symbol, company name, # shares owned, current close price per share, 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. You will need to look up the current price of the stocks in a portfolio via the IEX API. Instead of hitting the API multiple times (slow and inefficient), you can request multiple prices in one request as shown below (notice you supply a comma-delimited list of stocks):

`https://api.iextrading.com/1.0/stock/market/batch?symbols=aapl,fb,amzn&types=price`

This will require consuming the web service in PHP. Take a look at pages 904-910 of Chapter 19 of your textbook for examples of how to do this.

 symbol	name	#shares	\$Close	\$value
 AMZN	Amazon	10	\$100	\$1000
 GOOG	Google	5	\$12	\$60
 AAPL	Apple	10	\$500	\$5000

TOTAL PORTFOLIO VALUE \$6060

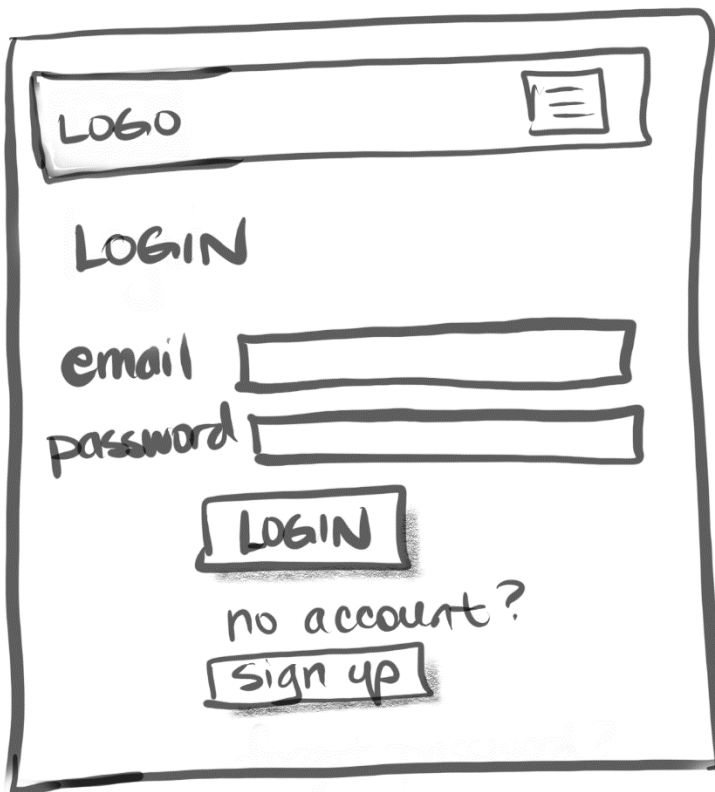
Login Page: will display a login screen. Some of the information below about hashing algorithms will be explained in class when we cover security.

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:

```
$digest = password_hash( $_POST['pass'], PASSWORD_BCRYPT, ['cost' => 12] );  
if ($digest == $password_field_from_database_table && emails also match) {  
    // we have a match, log the user in  
}
```

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: 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 hash with a cost value of 12.

A hand-drawn sketch of a registration form titled "SIGN UP". The form is enclosed in a rounded rectangle. At the top left is a "Logo" placeholder, and at the top right is a hamburger menu icon. Below the title, there are seven input fields stacked vertically, each with a label to its left: "First Name", "Last Name", "City", "Country", "email", "password", and "confirm password". A "SIGN UP" button is located at the bottom center of the form. To the right of the form, there are three handwritten annotations with arrows pointing to the input fields: a bracket spanning the first four fields with the text "can't be left blank", an arrow pointing to the "email" field with the text "proper format", and an arrow pointing to the "password" and "confirm password" fields with the text "must match".

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.