Cyber Captain Software Team Introduction Assignment
CRUD Registration and Login
09/29/23

Hello new software team Cyber Captains! We would love to have you help us work on our website with us, currently we are developing an internship tracking and distribution system. However, before we can do that, we'd like to test your skills and get you comfortable with PHP, Javascript, CSS, and SQL as well as some other software we use like Github, Visual Studio Code, Xampp and PHPMyAdmin. To accomplish this, we have simple task laid out in several parts below.

Your job will be to act as full stack developers, this means working on the front end (GUI – CSS/HTML), backend (Database – SQL) and the logic that connects the two (Javascript/PHP). You will be creating a registration/login system that lets certain users log in and delete certain user's accounts.

Part 0 – Setup

First, you must install the required software and make sure it is all working BEFORE you start programming. You may use other IDEs if you would like, though the recommended IDE is Visual Studio Code. This is because it is capable of doing everything we need it to all in one program. If you choose to use other IDEs that may be acceptable, however we can not promise we will be able to support any issues you have with them. If you have any questions or issues with any of the following steps please bring them to any of your Mentors and they will help you resolve them.

0.1 Installing your IDE, Visual Studio Code.

If you already have Visual Studio Code installed on your PC, you can skip this step. Otherwise, navigate to the below link and download/install Visual Studio Code.
Link: https://code.visualstudio.com/download

0.2 Familiarizing yourself with Github through Visual Studio Code.

If you already use Github with visual studio code, you can skip this step. Otherwise, ensure you have Git installed (It is separate from github) if you do not, navigate to this link and install Git.
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
Once you have done that, watch this tutorial (and possibly others if you need) and follow along using your VS Code application https://www.youtube.com/watch?v=i_23KUAEtUM.
If you encounter no issues, feel free to play around with it some more and try out other features of VScode/github then move on to the next step.

0.3 Installing XAMPP and modifying the configuration.

If you already have Xampp installed and functional you may skip the first half of this step. Otherwise, navigate to this link and install Xampp 8.1.17 (or similar 8.1.X) to your system.
https://www.apachefriends.org/download.html

0.3.1 Modifying your Xampp Config.

In case some students are using default xampp configuration for other classes, we use port 8080 (instead of the default port 80). To make this change you need to open xampp and under "Apache" you click "Config" and click on the first link (Apache httpd.conf) open that file. When you've done so, ctrl+f or scroll until you find something that says, "Listen: 80" and change it to

"Listen: 8080" then look for something called "ServerName localhost:80" and change that to "ServerName localhost:8080" Then save and close this file. If you've done this correctly, when you start your apache server it should go green and say the Port(s) are 443, 8080.
From now on, whenever you work on your project(s) you just need to ensure you have Apache running as well as MySQL.
If you did all this right you should be able to open your web browser now and navigate to 'localhost:8080' in your URL.

0.4 Familiarizing yourself with phpMyAdmin.
When you navigate to 'localhost:8080' after completing all the previous steps, you should be redirected to the 'dashboard' from here you can click 'phpMyAdmin' in the top right. This is a handy UI to make managing your mySQL database much easier! If you are already familiar with mySQL and prefer to use commands over the GUI, phpMyAdmin has a section for that as well. I suggest you familiarize yourself with it. Consider watching some tutorials online.

Part 1 – Registration

This will likely be the biggest hurdle for you of the whole project. Your first task will be to create a registration page. This will require you to do some front end, back end, and logic to connect the two. I suggest you do some planning before you write any code. Think about the things you will need to store in your backend database. For example, an email, a password, and some sort of ID field to identify the record with. (there may be other things you want to store as well that you will need in the future, now would be a convenient time to implement them if you can think that far ahead. However, it is not necessary at this step.

You will also need a front end, this will involve some HTML code, stored in .php files & some CSS code (called style and appropriately stored in a style.css file).

Lastly, you will need some logic in the middle to connect the two. Something to take the info entered into your front end forms that checks the information to be valid, not some sort of cyber attack, and not a duplicate, and then puts it into the backend. A comprehensive list of requires can be found below.

It is important to note, that that these do NOT have to be coded in any particular order and you might actually find it easier to start with one over another. You will likely find yourself bouncing back and forth between them during the process anyway.

1. Misc.
   - 1.1) All code must be neatly and consistently organized.
   - 1.2) Camelcase must be used for all names
   - 1.3) Comments must be inserted into code to allow other programmers (including yourself a month from now) easily decipher what code does what
2. Backend
   - 2.1) Must use phpMyAdmin & MySql for database
   - 2.2) Must store, at minimum, an email & password
     - 2.2.1. Email must be in the format ***@***.***
     - 2.2.2. Password must require at least 1 capital letter, 1 lowercase letter, 1 symbol (!@#$%^&*), 1 number, and be at least 8 characters in length.

2.3) Database table must contain a field called 'id' that auto increments as new rows are added.

3. Middle
   - 3.1) Must use at least some Javascript/PhP, other languages are welcome if they serve a purpose.
   - 3.2) Must insert records into the database using prepared SQL statements
   - 3.3) Password must never leave the client unencrypted. Before storing in the database encrypt it using a function like "Password_Hash"
4. Frontend
   - 4.1) Must use a form to a submit
   - 4.2) Form must use things like 'pattern' to ensure the user is entering valid data.
   - 4.3) Must collect and store atleast an email, password, first name, last name, as well as a phone number.

Part 2 – Logging in

Once you have fully completed Part 1 you should be intimately familiar with your toolbox. Now you will be required to utilize what you just practiced and implement a new feature/page to your website. You must create another form that allows users to enter pre-existing account info (an email & password) both must be checked and compared to the database to validate login. If they both pass, then the user is to be brought to another page informing them of a successful login attempt.

1. Misc.
   - 1.1) All code must be neatly and consistently organized.
   - 1.2) Camelcase must be used for all names
   - 1.3) Comments must be inserted into code to allow other programmers (including yourself a month from now) easily decipher what code does what
2. Front End
   - 2.1) Must have form with atleast two entry boxes and a button.
   - 2.2) Password field must contain toggleable feature to show/hide someone's password
   - 2.3) Username field must pattern match to ensure what the user enters is even possible to be in the database (use your registration pattern)
3. Middle
   - 3.1) Must check database efficiently for emails that match the user entered one.
     - 3.1.1) If one is not found, you must give generic login failure
     - 3.1.2) If one is found, you must then check the password associated with it. Remember, never let an encrypted password leave the client's side.
4. Backend
   - 4.1) No backend modifications should be necessary, though you may find opportunities to optimize or otherwise fix your backend from the previous part.

Part 3 – User Management

Finally, once you are able to login you must create a dashboard that the page redirects you to once you login. This page must be inaccessible to people who aren't logged in. On this page you must have at least two buttons. One that leads you to a page where you can manage users other than yourself and another that lets you manage yourself. From the page where you manage others you must be able to change people's personal information, such as their phone number or names. Email must be visible but not editable and password must not be shown. For your own page you must be able to edit all of the same information as above as well as have an option to change your password. This should require the previous password as well as the new password.

1. Misc.
   1.1) All code must be neatly and consistently organized.
   1.2) Camelcase must be used for all names
   1.3) Comments must be inserted into code to allow other programmers (including yourself a month from now) easily decipher what code does what
2. Front End
   2.1) several webpages must be created. Your previous 'login successful' page must now be turned into a dashboard containing several buttons.
      2.1.1) Logout button
      2.1.2) Manage Others
      2.1.3) Manage Self
   2.2) The logout button must log the user out and direct them to the login or landing page
   2.3) The Manage Others page must show a list of all users in the database and allow the user to modify their data
   2.4) The Manage Self page must allow the user to manage their own data as well as change their password at will.
3. Middle
   3.1) The code must never transfer an unencrypted password away from the client.
   3.2) You must achieve full CRUD (Create, Read, Update, Delete) functionality, this is achieved as such.
      3.2.1) Creation was accomplished during registration
      3.2.2) Reading was partially completed during login and will be further elaborated on when you create your 'manage users' pages
      3.2.3) Updating will happen when you allow users to change their information
      3.2.4) Deleting occurs when a user is removed from the database entirely.
4. Back end
   4.1) The backend must accurately reflect current users and all of their updated data in real time.