COMP 3512 Assignment #2

Due Tuesday December 10th at midnight
Version 0.9, November 3

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/f2019-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 travel-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.

- **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 they are 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 (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*:
    - 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 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	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 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.

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 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.
- 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 and set-up your github accounts. 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 November 8.* **If you don't have a group by the morning of November 7, please email me and let me know; I will try to get you into a group in classs on November 7.**

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 November 15.*

Milestone 3. Create the following API in PHP.
Due November 22.

`api-countries.php` – with no parameter, return JSON representation of all countries. If supplied with `iso` parameter, then return just JSON data for single specified country. You will use the service that returns all countries on the Countries page.

`api-cities.php` – with no parameter, return JSON representation of all countries. If supplied with `iso` parameter, then return just JSON city data for single specified country. You will use this service in the Upload Photo form.

Data Files

Travel images are the same as the first assignment. **In the first assignment, the travel images resided in a GCP storage bucket. This time, the images will reside instead in a folder on your host/server.**

Data in this assignment will come from MySQL. You will be provided with the import scripts for them.

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.

Visual Design: Your pages will be marked on a desktop computer, but your site needs to be usable at mobile sizes. That is, I will be mainly testing and evaluating this assignment using a browser with a large browser width, say 1200 pixels wide. You should still use media queries as I will also test your pages with a smaller-width browser size to ensure it looks reasonable at the mobile sizes.

Header: Each page will have a header at the top that will contain some type of logo and a navigation menu. For smaller browser sizes, you will instead need 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
- Browse/Search
- Countries
- Cities
- Upload. 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.

Note: the sketches in this assignment specification are meant to show functionality, not design. Here I've shown content as boxes, but you could do them as rectangles, circles, icons, simple links, etc. Make your pages look nicer than these sketches!

Home (Not Logged In): The main page for the assignment. This file **must** be named `index.php`. This must have the functionality shown in the following sketches. The first shows the home page when user hasn't logged in; the second show the home page after a user has logged in.

A hero image is a large banner image: you can find attractive very large images on unsplash.com. The search box should take the user to the browse/search page (i.e., it should show results as if the user performed a photo title filter action).

Home (Logged In): Display the user info (first name, last name, city, country). Also display any photos that belong to them, and of their favorited photos (if none yet, be able to handle that).

Also display maybe 10-12 recommended photos the user "may" like. How to do this? Ideally, if you had several additional months to work on this assignment, you would create a recommendation engine using some type of Machine Learning algorithm. But given the time constraints, simply show: images with the same country and city to those they own or have favorited, or for favorited images by the same user. It's possible for a new user who hasn't uploaded or favorited any images, that you don't have anything to base this Like algorithm on. In that case, show the most recent (i.e., the last) 10-12 photos in the imagedetails table. If your Like algorithm has fewer than 10-12 images, fill it up with the most recent images.

Each of these photo images should be links to the `single-photo.php` page with the image id in the query string.
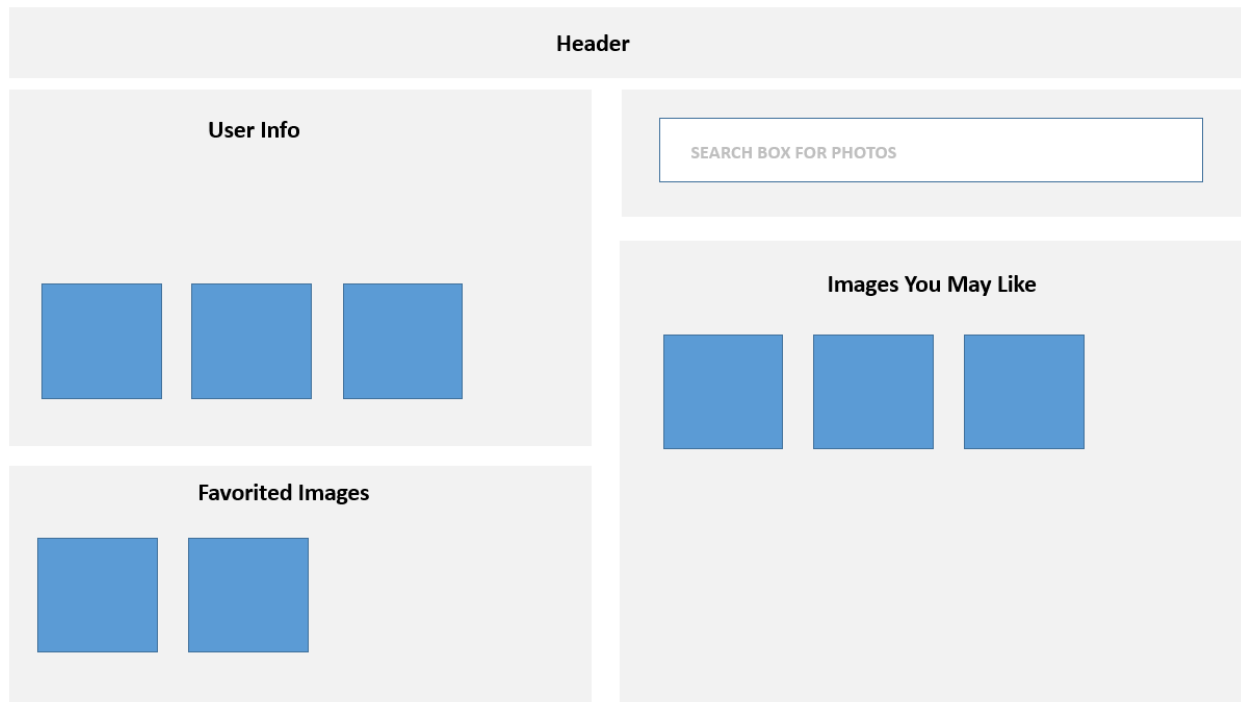
Country Page: will display a sorted list of all countries as well as provide details about a country. This file must be named `single-country.php`. This list must be populated in JavaScript from an API you created in your third milestone named `api-countries.php`. Each country in this list should be a hyperlink, with the URL being the same `single-country.php` page, but with a querystring parameter indicating which country to display (e.g., `single-country.php?iso=ca`).

To improve the performance of your assignment, you must store the list of countries in local storage after you fetch it. Your page should thus check if country data is already saved in local storage: if it is then use it, otherwise fetch it and store it in local storage.

The country filters have the exact same requirements as the first assignment, and use JavaScript to perform the filtering.

When the user clicks on a country (that is, the page is requested with a querystring parameter), display the details of the country (name, area, population, capital name, currency, domain, languages, neighbouring countries, and description), a list of photos for that country, and a list of cities in that country. Each photo image should be a link to `single-photo.php` with the appropriate query string parameter. Each city should be a link to `single-city.php` with the appropriate query string parameter.

City Page: This page must be named `single-city.php`. It will display information for a single city. Which city? The city whose id was passed as a query string to the page.

This page will allow the user to view the following city fields: name, area, population, capital name, currency, domain, languages, neighbouring countries, and description. The languages field is a comma-delimited list of ISO 639-2 codes. You must instead display the full name (e.g., English instead of en). In the first assignment, you used the languages service to transform the codes into country names. In this assignment, this transformation will have to happen on the server-side in PHP using the languages table.

You will also have to display a static country map that includes a marker for the city (you will use the Google Maps Static API). Like on the previous page, the city photos will be links to `single-photo.php`.

Notice that this page has the same country list+filter as the country page. Duplicate the functionality but don't duplicate the code+markup!

Header

Country Filters	City Details	City Photos
Country List	City Map	

Photo Page: This page should be named `single-photo.php`. It should display the single photo indicated by the passed querystring parameter. It should have the same functionality as the Single Photo View in the first assignment (tabs, map, hover photo, but no speaking or close buttons). Country and city fields should be appropriate links to `single-country.php` and `single-city.php`.

The Add to Favorites button will add the photo to a session-based list and provide feedback that it has been added. This can be achieved in two ways:

- Redirecting to a PHP page that adds the photo to the session list, and then redirects back to this page. This is easiest but old fashioned and inefficient; it results in two redirects (two request+response cycles).
- Do it asynchronously using JavaScript fetch. The fetched page would return simple JSON that indicates whether add to favorites was successful or not. This is a better approach and will get a slightly better mark.

This add to favorites button will only be visible if the user is logged in. If a photo is already favorited, you shouldn't be able to add it again. Ideally, you do this by hiding/disabling the button. Alternately, you display a message telling the user it's already been favorited.

Favorites Page: will display list of logged-in user's favorited photos (implemented via PHP sessions). If none yet, be able to handle that with a message. This page must be named `favorites.php`. The user should be able to remove photos singly or all at once from this list. This page will be entirely in PHP. The list should display a small version of the photo and its name. They both should be links to the appropriate single photo 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. Every year, for unknown reasons, students lose easy marks by not fulfilling these simple requirements.

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.

Browse/Search Page: display a list of photos with the ability to filter the list. Sort the list by title. In the filters, allow the user to select a city or country (using <select> lists, but only include cities and countries that have photos). Also provide a text box to allow the user to search for photos that contain the entered text anywhere in the title. Finally, the search/filtering will need some type of submit/filter button. This page doesn't use JavaScript but PHP which will require the programmatic construction of a SQL WHERE using this form data. To simplify your coding, assume that each filter is mutually exclusive (no ANDs or ORs needed). Provide a way to reset/remove the filters.

Header

Photo Filters	Browse/Search Results		
	Title	View	Add to Favorites
	Title	View	Add to Favorites

Login Page: will display a login screen with email and password fields, plus a login button and a link or button for signing up if the user doesn't have an account yet. 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.

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.

Upload Photo: allows logged-in user to upload a photo. Pages 561-568 provides example code for how to do this. This must be a user form with the following fields:

- title, description
- country (as a `<select>` list showing the country names, but using the appropriate id fields as value attribute). To simplify the user experience, make this list contain only the countries with photos.
- city (as a `<select>` list showing the city names, but using the appropriate id fields as value attribute). The city list ideally should contain only cities in the selected country. This will require using JavaScript and an asynchronous fetch.
- File Upload field and button.

The title field can't be blank.

On the server side, you will have to set the `ContinentCode`, `UserId` as well as the `Latitude` and `Longitude` fields for the photo: to do this, simply use the lat+long of the selected city. You will also have to set the `Path` field. You could simply use the filename of the uploaded file, but this potentially will cause errors if the same filename is uploaded. A safer approach would be to use something like the PHP `uniqid()` function to generate a unique filename for the uploaded photo (and the `Path` field).