

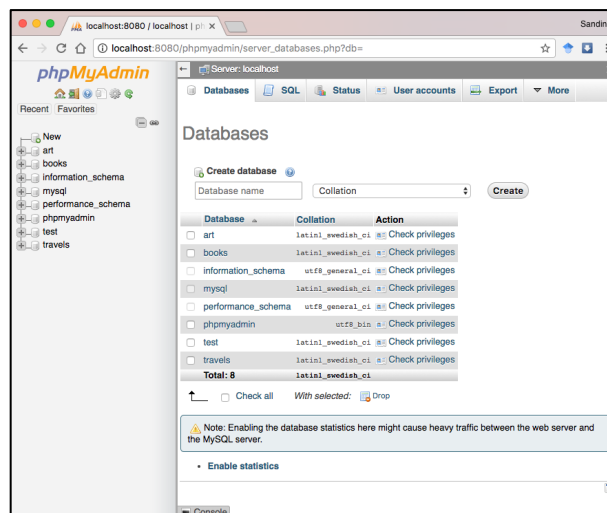
CS3500: Assignment 4 – PHP and Database [20 points + 5 Bonus Points]

Overview:

This assignment provides an opportunity for you to demonstrate your ability to generate dynamic web pages using PHP and Databases, as well as creating a session to keep a user logged in the website. In this assignment, you will be working with a Travel Journal webpage. You will be building upon a visual design provided by Bootstrap.

Database Instructions:

You have been provided with a SQL script named `travels.sql`. Open phpMyAdmin and select the database named “travels”. If you don’t have a database named “travels” then create one by clicking on the “Databases tab” shown in the figure below. Once you select the “travels” database, run the script by clicking on the “Import” tab (as shown in class) to create the tables and content for the website. Make sure it is working by performing simple selects.



While the database contains quite a few tables, this assignment will only make use of GeoCities, GeoCountries, TravelImage, TravelImageDetails, TravelImageRating, TravelPost, TravelPostImages, TravelUser, TravelUserDetails, TravelUserFollow. Explore these tables so you know which information they provide.

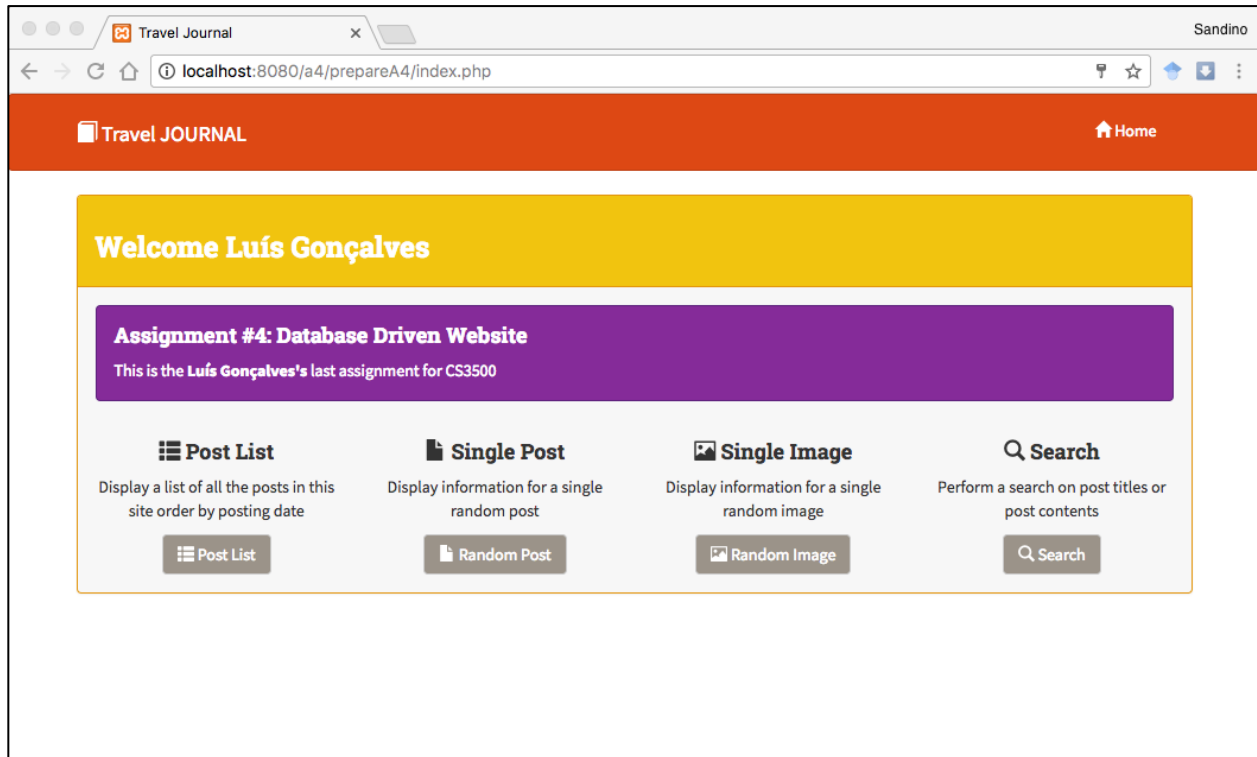
PHP Instructions:

1. You have been provided with 10 PHP files. These files are the templates you will use to show your content:
 - a. `side.php` and `header.php` contain the code to generate the header and the left side panel.
 - b. `error.php` is a page to redirect when an error has occurred.

- c. `index.php` is the home page.
 - d. `post_list.php` is used to show a list of all the posts in the database.
 - e. `post_single.php` is used to display the content of a single post.
 - f. `image.php` is used to display a single image and its information.
 - g. `login.php` is used to display a login form.
 - h. `profile.php` is used to display information about a logged in user.
 - i. `search.php` is used to display a search form and the results from the search.
2. All links on all pages must be functional, meaning that they should point to the right page. For the most part, all provided files have the proper links pointing to the proper pages (but you need to double check).
 3. Make sure you use the PHP function `utf8_encode()` to deal with special accents and characters for text data obtained from the database.
 4. All the dates extracted from the database should be formatted using the PHP function `date('F d, Y', strtotime($row['PostTime']))`, where `$row['PostTime']` is the variable containing the date extracted from the database.

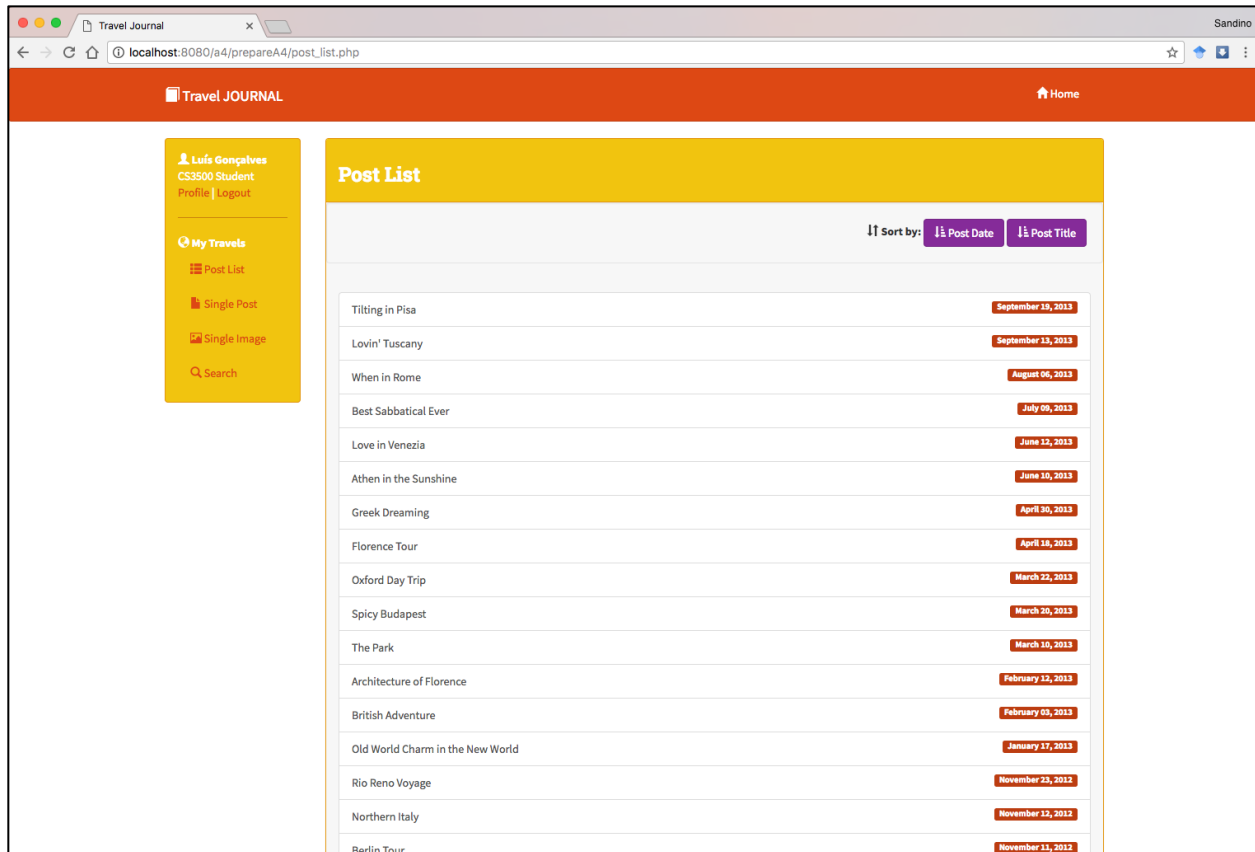
Specifications:

index.php: The home page is named `index.php`. This home page must contain additional links to the pages as shown below:



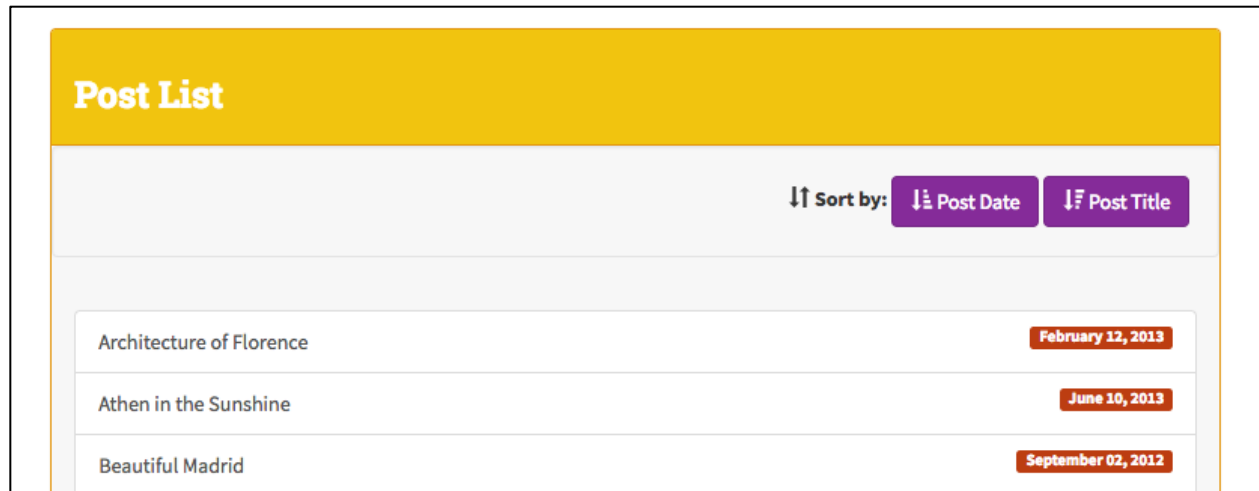
- This page is only accessible if a user is logged in (if you did the bonus).
- “Single Post” and “Single Image” should link to the respective PHP pages with an id to a random post or image attached to the buttons. You’ll search the `TravelPost` and `TravelImage` database and get `PostIDs` and `ImageIDs`. Then select one of each at random and add them to the id in the links (buttons), such that the `href` property of the link looks like: `href="post_single.php?id=12"` and `href="image.php?id=34"`. If the user reloads `index.php` a random image and post id will be attached to these links.

post_list.php: This page must display a list of posts (from the `TravelPost` table) that should look like the figure shown below. **Notice the sort order.** Each post title should also be a hyperlink to `post_single.php` with a querystring parameter `id` which is equal to the `id` of the post.



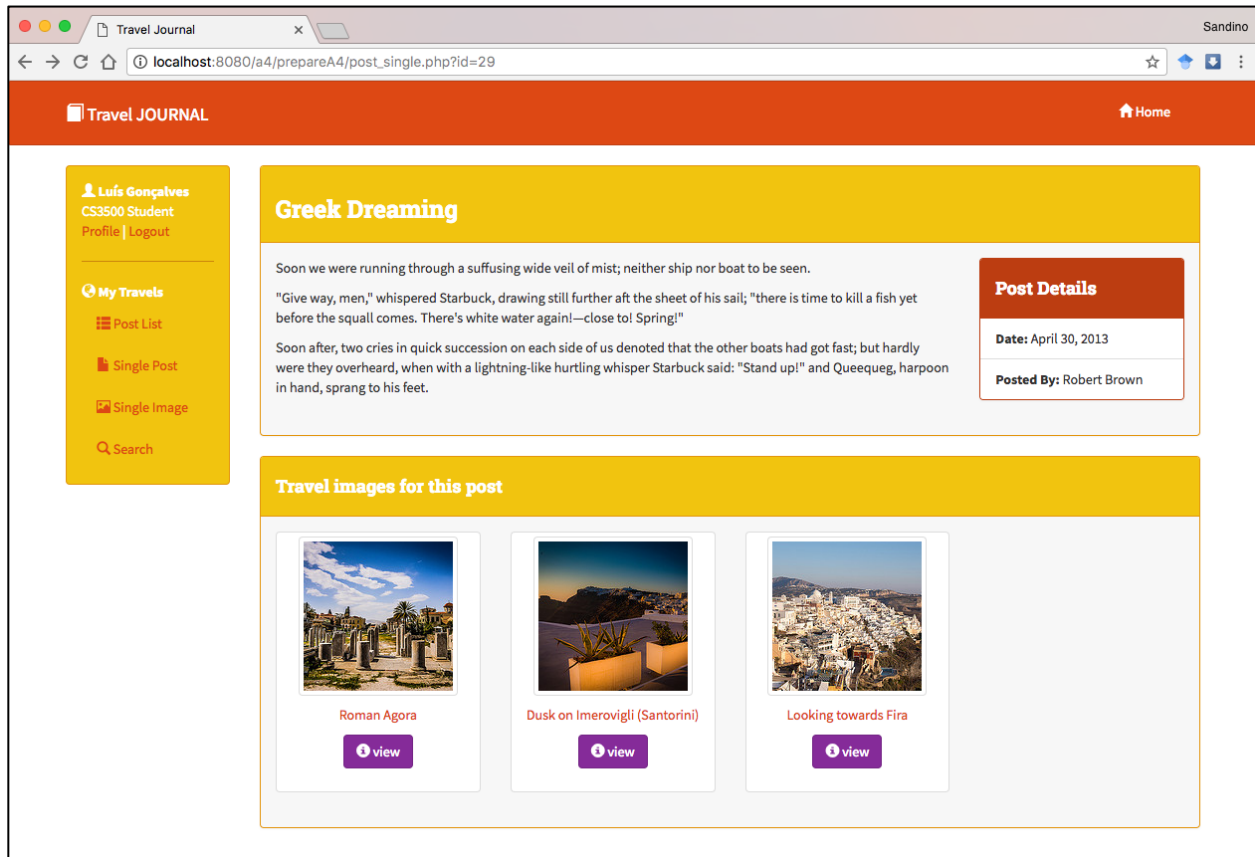
- The page contains all the HTML and Bootstrap classes, you only need to display the list using the result set from the database and adding each to a list element (``).
- The list by default show the post list order by `PostDate` descending. The sort by buttons allow the user to sort by:
 - Title (Post Title): Descending (Z to A) or Ascending (A to C).
 - Date (Post Date): Descending (Oldest to Recent) or Ascending (Recent to Older).
- **[BONUS]** The figure below shows how this sorting buttons work. When choosing to sort by date for example, the button's glyphicon class will change to "glyphicon glyphicon-sort-by-attributes-alt" to sort DESC and "glyphicon glyphicon-sort-by-attributes" to sort ASC. Also, the link will change to `order=PostTime&type=ASC` to sort by PostTime ASC or `order=PostTime&type=DESC` to sort by PostTime DESC, and `order=`

`order=Title&type=ASC` to sort by Title ASC or `order=order=Title&type=DESC` to sort by Title DESC.



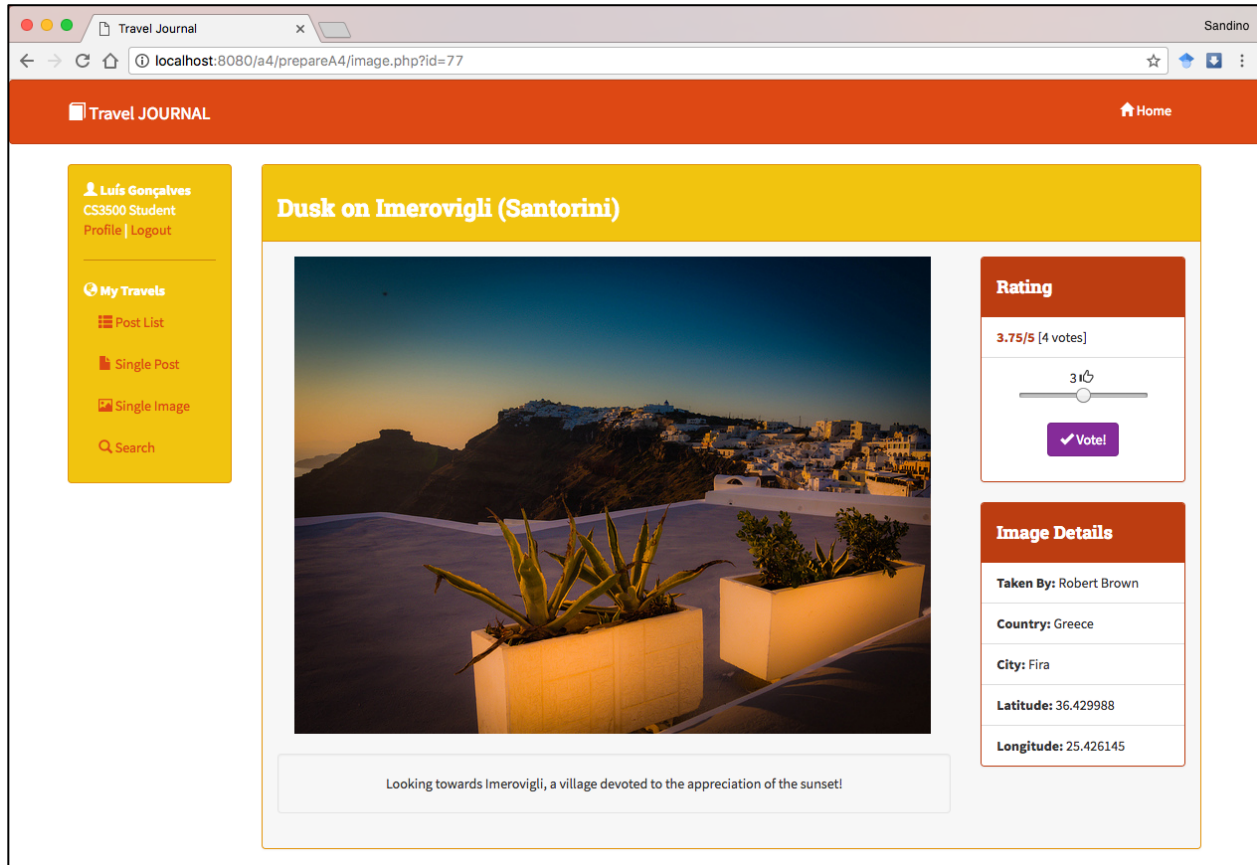
- Make sure to use `$_GET` to get the sorting options and change the sorting buttons accordingly.
- When a user clicks on a post in the list, it should take them to `post_single.php` where more details about the post will be shown.

post_single.php: This page displays information from the `TravelPost`, `TravelUserDetails`, and `TravelImage` tables for a single post (specified via the id passed in via a querystring parameter). This page should handle a missing or invalid querystring id parameters by redirecting to the page called `error.php`. The page is shown below:

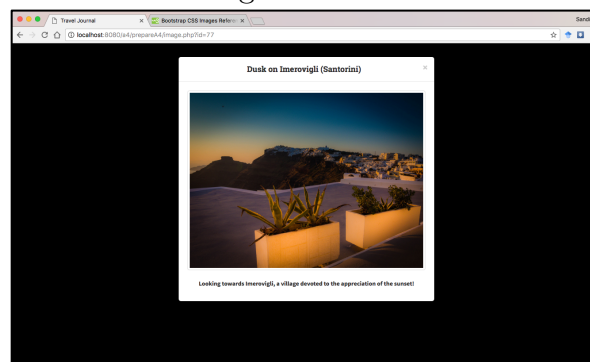


- In the “Travel images for this post” section all image’s thumbnails, titles, and the view button must be links for that image to `image.php`.

image.php: This page must display information from a variety of tables (TravelImageDetails, TravelImage, TravelImageRating, GeoCountries, GeoCities, TravelUserDetails) for a single image (specified via the image id passed in via a querystring parameter). You must display the image information as shown below:



- The image must be a link to the same large version using the Bootstrap Modal dialog to display it. The code for the modal class is provided to you, just make sure that you understand it and change the image path, title, and description accordingly, as shown in the image below:

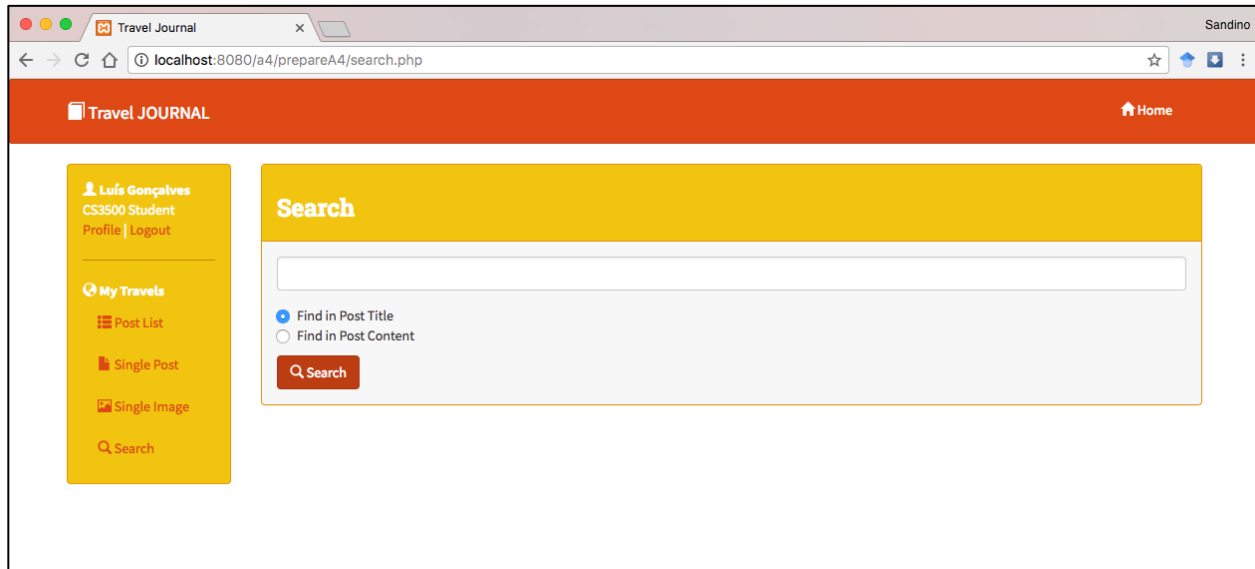


- In the rating panel you will display an average rating for the image using the `AVG()` operator in SQL (research that operator). Build a select statement that looks like: `"SELECT AVG(Rating) as RatingAvg, COUNT(Rating) as Votes..."`. You are going to display `RatingAvg/Votes` such that it looks like `3.75/5` [4 votes]. Use the PHP function `number_format($RatingAvg, 2, '.', '')` to display the average rating with two decimals after the period.
- The voting button should send the image id (using an input type hidden) and the selected value in the range input to `image.php` via `querystring`. You will then insert this vote into the table `TravelImageRating` using the field `ImageID`, and `Rating` in your `INSERT` statement. Remember to use binding values to avoid SQL injections and `commit` for this kind of transaction so the data can be permanently saved in the database:

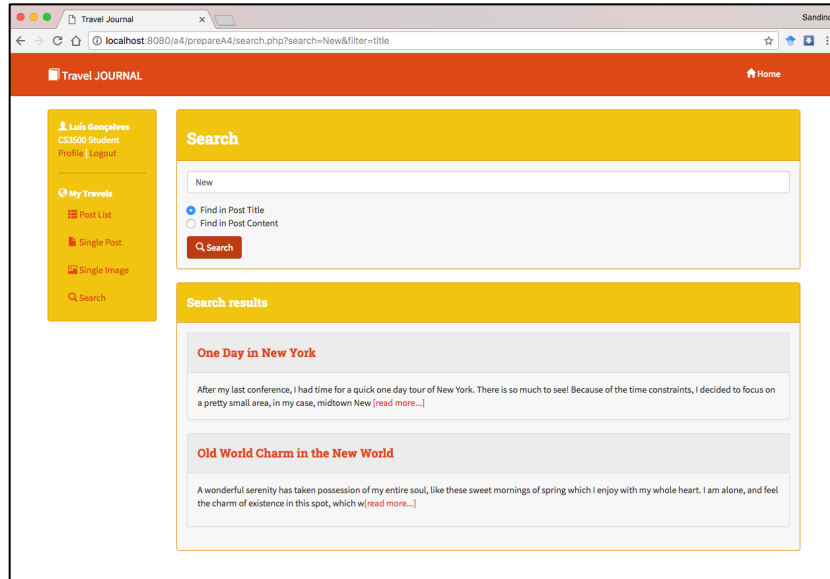
```
$pdo->beginTransaction();
$stmt = $pdo->prepare($sql);
$stmt->bindValue(1, $_GET['ImageID']);
$stmt->bindValue(2, $_GET['Rating']);
$stmt->execute();
$pdo->commit();
```

- After voting for the image and saving the results, the Rating panel should show this new information.
- Like with `post_single.php` this page should handle a missing or invalid `querystring id` parameter.

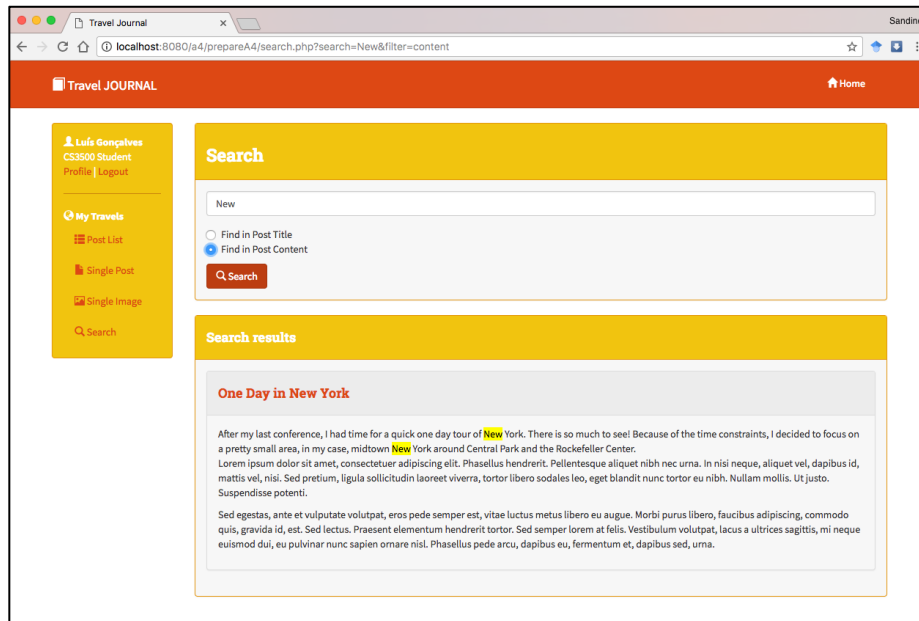
search.php: This page must display a list of posts (from the `TravelPost` table) according to search keywords, that should look like that shown below:



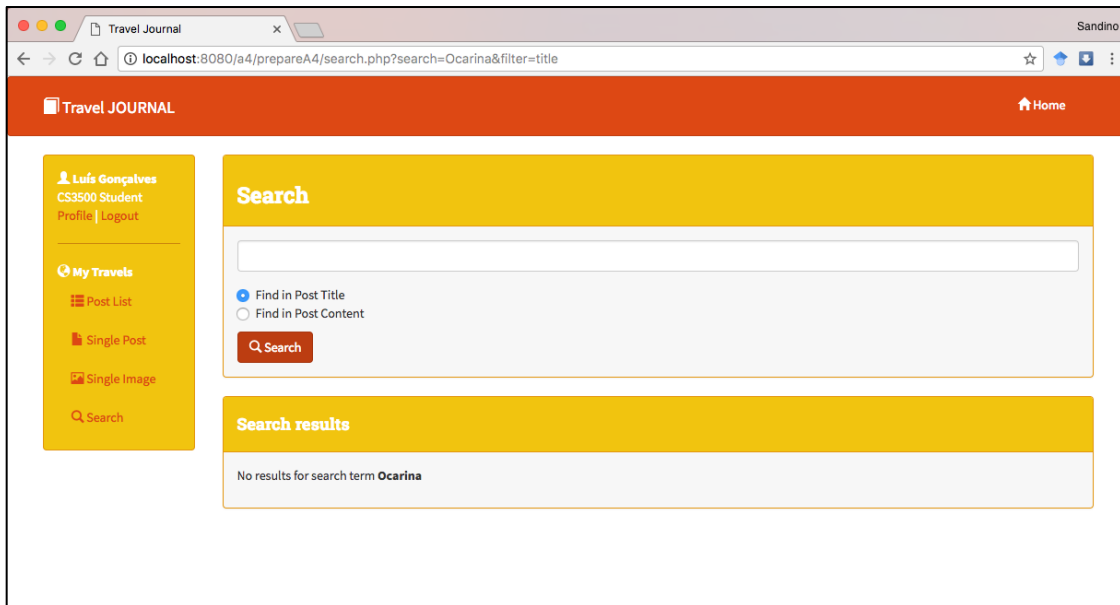
- Both “Find in Post Title” and “Find in Post Content” searches should use the SQL LIKE operator (research about it) with wildcards (%) so that it matches any occurrence of the search string. The LIKE operator is used in the WHERE clause and it might look like this: `SELECT * FROM TravelPost WHERE Message LIKE '%new%'`. This will match all the rows of the table `TravelPost` that contain the keyword “new” in the column `Message`.
- When the user presses the “Search” button, then display the title and the content for any post that matched. The title must be a link to `post_single.php` for that post (passing a querystring with the appropriate post id).
 - **[BONUS]** If the user performs a search by title, the results should look like the image below. Truncate the length of the message (content) to the first 200 characters and append a “[read more...]” link to `post_single.php` (like you did for the post title).



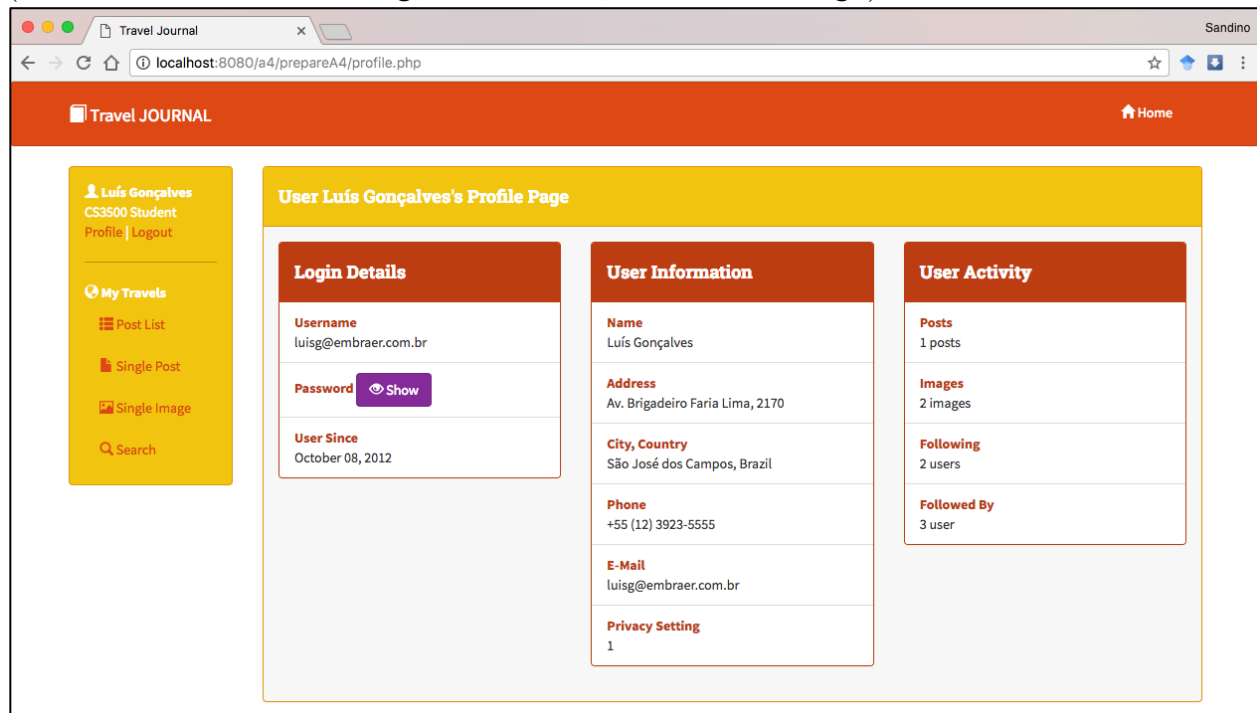
- **[BONUS]** If the user performs a search by the content, then highlight in yellow the search field in the resulting message content, as shown in the image below. You can achieve this effect by using the PHP function `str_ireplace()`, a **case insensitive** function that takes three parameters; The old word to be replaced, the new word that will replace the old word, and the string that contains the words. For the new word enclose it in the `<mark> </mark>` HTML5 element, which Bootstrap will automatically highlight with yellow. The post title of all the results, should be a link to `post_single.php`.



- If no results are returned, then show the message as shown in the figure below:



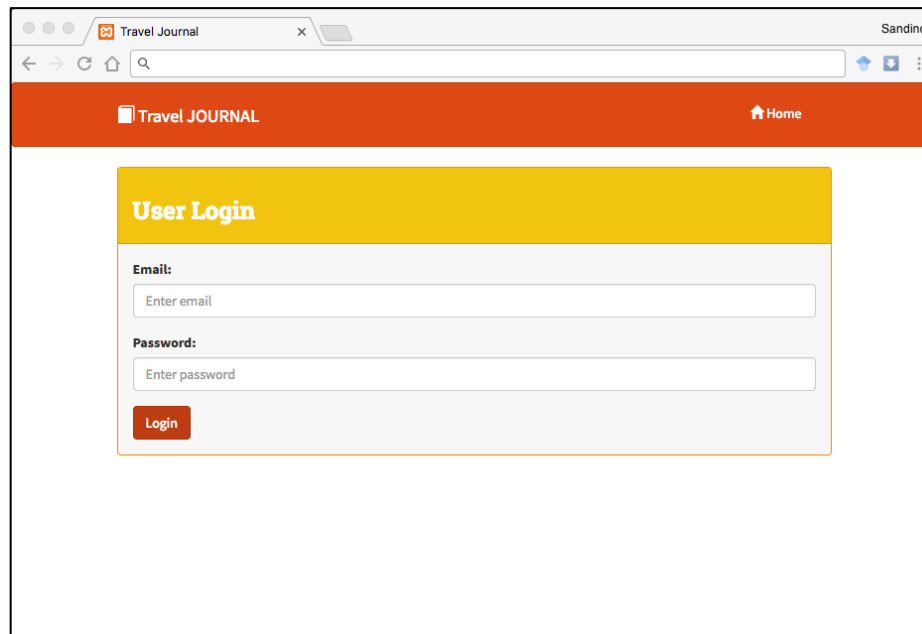
[BONUS] profile.php: This page must provide user information about the user that is logged in. This page will use the tables `TravelUser`, `TravelUserDetails`, `TravelUserFollowing`, `TravelPost`, and `TravelImage` and it must show login details (from `TravelUser`), user information (from `TravelUserDetails`), and user activity (from `TravelUserFollowing`, `TravelPost`, and `TravelImage`).



- Clicking the “show” button in the Password section of the “Login Details” panel should show/hide the password. This is implemented for you in the page. Just study the functionality and keep it in mind for the future.
- To get the numbers in the “User Activity” panel, the SQL SELECT should use the operator `COUNT()` and use the UID of the logged in user to filter those queries.

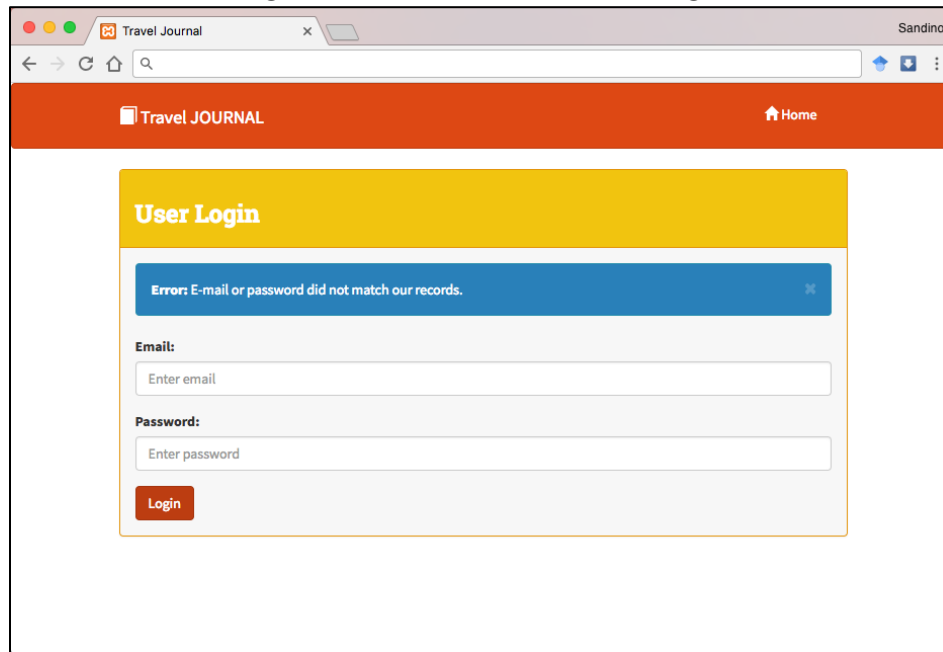
[BONUS] login.php: This page is the one the users will see if they are not logged in. So, each page (except for `side.php`, `header.php`, and `error.php`) in this assignment should redirect to `login.php` if the user is not logged in. You will be using the super global `$_SESSION` to set a username (first and last name of the user) and userid (UID) after creating the session using the PHP function `session_start()` (all the appropriate pages will start with this function as you will see when you open them in your editors).

- You will use the information about users defined in the `TravelUser` table to login users. All passwords in this table are `'abcd1234'`. You can use the username `'luisg@embraer.com.br'` for your testing (this is the user that appears in all the images in this document).
- This page provides a login form (using the method POST) as shown in the image below:

A screenshot of a web browser window. The browser's address bar shows 'Travel Journal' and the user's name 'Sandino'. The page has an orange header with 'Travel JOURNAL' on the left and a 'Home' link on the right. The main content area features a yellow box with the title 'User Login'. Inside this box, there are two input fields: 'Email:' with a placeholder 'Enter email' and 'Password:' with a placeholder 'Enter password'. Below these fields is a red 'Login' button.

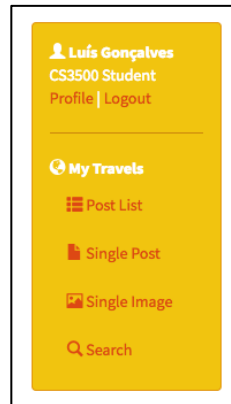
- After the user clicks the “Login” button you will check the email and password by performing a query to the `TravelUser` table, using the email and password values provided in `$_POST`. This query should return the UID. If a result is returned (you can check this by using `$result->rowCount()` on the result set if using PDO), then add the UID to the `$_SESSION['userid']`. Then, using the UID, select `FirstName` and `LastName` from the `TravelUserDetails`, concatenate them and added to `$_SESSION['username']`. As mentioned before, you might have to use the PHP function `utf8_encode()` to deal with special accents and characters.

- If no results were returned by checking the email and password in the `TravelUser` table, then do not set `$_SESSION` values, and show the form again, but now with a message like the one shown in the figure below:



- After the session has been established, the page should redirect to `index.php`.
- Each of the appropriate pages (`index.php`, `post_list.php`, `post_single.php`, `image.php`, `profile.php`, and `search.php`) must check if the `$_SESSION['username']` and `$_SESSION['userid']` are set, if not, then they should redirect to `login.php` so the user can login.
- When `login.php` receives the querystring `login.php?logout=1`, you will destroy the session by using the PHP function `unset()` on `$_SESSION['username']` and `$_SESSION['userid']`. This will effectively eliminate the session, so the user will have to login again in order to access the other pages. This querystring will be passed to `login.php` by clicking the "Logout" option in the side panel (more details below)

[BONUS] side.php: This page is included in the majority of the other pages and it implements the code for a side panel menu. As shown in the figure below:



- The user name should be extracted from the `$_SESSION['username']` since you have that name there already. Note: *Whenever you see the user name you should echo the `$_SESSION['username']` value.*
- “Profile” option should be a link to `profile.php`, and “Logout” option a link to `login.php?logout=1`, so `login.php` will take care of destroying the session, effectively login the user out.
- The options “Single Post” and “Single Image” will have a link to a random post or image (like in `index.php`). You’ll search the `TravelPost` and `TravelImage` database and get `PostIDs` and `ImageIDs`. Then select one of each at random and add them to the id in the links, such that the href property of the links looks like: `href="post_single.php?id=12"` and `href="image.php?id=34"`

Testing:

1. First, test your page by seeing if the pages and links work.
2. Make sure the random image and post are attached to the links in the `side.php` (if you did the BONUS) and the `index.php`, such that when clicked, it takes you to a random image or post in `image.php` or `post_single.php`, respectively.
3. Test the search functionality and make sure that the different searches behave in the way expected. Also make sure that if no results are returned, the appropriate message is post.
4. (if you did the BONUS) Make sure that the session is correctly implemented, and that the only page accessible to users that are not logged in is `login.php`
5. Test the vote button in `image.php` and make sure the votes are being recorded and display accordingly.
6. Try resizing all the pages and observe how with bootstrap, the layout adapts to the new display size.

Evaluation:

Below is the evaluation for Assignment 4.

Points	Item
8	image.php: correct data, hyperlink with proper querystring id, handle querystring errors, modal to show picture, and voting functionality.
BONUS 2	login.php: user login, handle users that are not logged in, redirect accordingly, logout users.
2	index.php: correct data, and random links.
2	post_list.php: correct data, sorting functionality, and appropriate links to post_single.php.
5	post_single.php: correct data, all images for post
BONUS 2	profile.php: show the correct data for the logged in user.
3	search.php: show the correct result for the different type of searches, link in post titles to the appropriate post in post_single.php
BONUS 1	side.php: show the correct data for the logged in user, random links, and logout link