**PharmGenius File Structure**

Here the file structure of the PharmGenius application is explained in general terms. Then, a short description of the contents of each file.

The file structure of the application contains of two primary folders – the includes folder, and the public folder. The includes folder contains PHP files that control backend functionality, including all primary functions used, database calls, AJAX calls, etc. The public folder contains PHP files for each page of the website. Also, the public folder contains a subfolder named static that houses all the Javascript, CSS, images, and other assets used by the site

NOTE: future project group may want to move the static folder up the file tree into the includes folder so that users are unable to navigate directly to these assets. This part of the structure is a holdover from how the previous site was set up.

NOTE: as of this writing, sections highlighted in yellow are not fully implemented.

This file structure is intended to be deployed such that the public folder is the home folder for the website. For example, if the live site were to be housed at “pharmacy.pitt.edu/pharmgenius”, then navigating to “pharmacy.pitt.edu/pharmgenius/index.php” should take them to the landing page. This way, the user will not be able to manually navigate to the files in the includes folder.

While working on the site using a WAMP/MAMP/LAMP stack, it may be easier to just install both folders in the localhost directory and navigate to localhost/public/index.php.

**The Includes Folder**

This folder houses files containing essential code for running the back end of the application, i.e. files that are not intended to serve as HTML pages.

*ajax\_calls.php*

This file contains an interface that handles all AJAX calls made for the entire site. The Javascript code (see public/static/js ) builds GET or POST data and sends it to this file. This file interprets the request from the Javascript and calls the appropriate PHP function(s) to perform the action requested. This is essentially the gatekeeper for any data exchanges that are not handled by page reloads or specific calls to another PHP file.

*db\_connections.php*

This is a simple PHP file that defines the database settings and creates a connection to the database. This file should be included in any file that makes calls to the database. Note: the settings in this file may need to be changed based on where the SQL server is deployed and the username/password that is used to connect to the database (see define calls at top of file).

*dropdown\_handler.php*

This file contains code that handles the updating the dropdown boxes based on what category has been selected.

*footer.php*

Contains a footer that should be included at the bottom of each page in the application. This includes the report modal that is used to report bugs. If this file is not included, a user will not be able to report a bug from the page they are on. Also includes some links to necessary Javascript. Note: this file contains the </body> and </html> tag, and as such, those tags do not need to be closed in each individual page. One simply needs to include this file.

*functions.php*

This is the backbone of the project. This file houses all of the code that handles the interactivity of the site. This includes, but is not limited to, making all calls to the database, building the navigation bar, login/password functions, etc. Basically, all code that does anything is stored in this file. Note: future groups may want to break this file into separate files that are logically organized, as the file kind of grew a lot during the updating of the application – should be discussed with the stakeholder as this type of refactor could eat a lot of time and be low in priority.

*get\_leaderboard.php*

Handles the updating of the leaderboard. This functionality should probably be wrapped into the ajax\_calls.php file, and could probably be done fairly easily, but time ran short on non-functional refactors.

*header.php*

Contains header data for all the pages in the application. This file should be included toward the beginning of each page on the site. Contains metadata and links to CSS/JQuery.

**The Public Folder**

This folder houses all the files for the pages to which a user can navigate.

*create\_account.php*

This page allows a user to create an account. A user of the site must be logged in to be able to do much of anything. Code that handles validation and submission of data via this page is handled by create\_account.js.

*index.php*

This is the main site landing page. Upon first visit, or if not logged in, this page will simply show you some information about the site. If a user is logged in, the arrow in the middle of the page will take them to the select\_quiz page instead.

*leaderboard.php*

Displays a leaderboard using the current data housed in the database. The user is able to filter by selecting different parameters in dropdowns. Code for this functionality is handled by leaderboard.js.

*login.php*

A page that will allow a user to login to the site. Also has a link to create\_account if the user does not yet have an account.

*logout.php*

A simple PHP script that logs the user out of the system by calling the logout function. This function essentially destroys the user’s session and returns the user to index.php.

*manage\_users.php*

A page viewable only by admins. Allows the administrator to mark users inactive, promote to admins or reset user scores. Backend is handled by manage\_users.js.

*profile.php*

Displays a user’s profile. Also allows a user to update profile information or change their password. Backend is handled by profile.js.

*review\_categories.php*

A page viewable only by admins. Allows the administrator to mark categories inactive or reactivate inactive categories. Backend is handled by review\_categories.js.

*review\_my\_questions.php*

A page that allows a user to view the questions that they themselves have submitted. Backend handled by review\_my\_questions.js.

*review\_new\_questions.php*

A page that allows a user to view questions that have recently been submitted i.e. questions that have not been marked as verified. Backend handled by review\_new\_questions.js.

*review\_old\_questions.php*

A page that allows a user to view all questions in the database. Backend handled by review\_old\_questions.js.

*select\_quiz.php*

Page that allows the user to specify which quiz they would like to take, from which category. Also, user can choose how many questions to get, whether or not to see only their own questions, and whether or not the questions should be timed. This file performs a GET request to take\_quiz.php upon submission.

*submit\_new.php*

This page allows a user to submit new categories/quizzes/questions to the database. Currently all submissions are marked as active, but it is the intent that when the site is live, the questions would be created and marked for review, so that admin users can decide whether the questions pass muster or not. Backend handled by submit\_new.js.

*take\_quiz.php*

This page handles the actual quiz taking. This page is accessed via GET request from select\_quiz.php. The GET parameters let take\_quiz.php know what quiz to select, how many questions, etc. All backend handled by take\_quiz.js. NOTE: it is important to note that the quiz is retrieved by AJAX, meaning that it is asynchronous. The call is done on page load, but if the webserver is slow, the quiz may not be immediately available. The user has to physically click a button in the modal that pops up to continue, but there is nothing preventing the user from doing so if the quiz data is not yet loaded. This may be a feature that needs to be implemented if it is deemed warranted, but local testing never produced a problem.

**The Static Folder**

This folder houses assets used by the site for varying capacities. These include images, Javascript, CSS, fonts, etc. For the most part, the CSS, Fonts, images, and such are all holdovers from the previous version of the application. As of this writing, the only major changes to the files in these folders are the addition of Javascript files to handle the backend of each page that requires such coding. These files are named the same as their “parent” page, aside from the file extension.

With a semi-working knowledge of Javascript, the code in these files should be relatively easy to understand. Most of them just handle some validation and use AJAX to update/retrieve data. Only take\_quiz.js is complication, and that is only due to the amount it needs to handle.