AssistRx PHP Coding Test

You can take as much time as you like to complete this test.

For New Developer Candidates

Welcome to the AssistRx PHP Coding Test! You solution will give us an understanding of where your skills will fit within our team. It covers the following topics:

- Coding Style (formatting, commenting)
- MYSQL/PDO (security for SQL injection)
- JavaScript functions and objects
- AJAX (using jQuery)
- CSS (you may use LESS also)
- Error Handling
- MVC Structure
- Native PHP functions
- Api Calls / JSON Result Sets
- Overall Understanding of Web Development

Folder Manifest

- config
  - database_sample.php
- models
  - my_model.php
  - patient_model.php
  - song_model.php
- public
  - arx_logo.png
  - jquery-1.9.1.min.js
  - normalize.css
  - script.js
  - song_search.js
  - styles.css

_database_export
ajax_controller
patients
README.md
README.pdf
report
songs

Getting Started:

1. On your local web server make a new database such as `arx_test`
2. Execute the `_database_export.php` file which will make two tables `patients` and `songs`
3. Duplicate the `config/database.sample.php` file as `config/database.php` and customize the MYSQL `host`, `user`, `pass`, `database` for your machine.
4. It helps if you make the project file your web root:

For example, on my machine I have XAMP. So in Terminal (on Mac) I would do:

```
$ cd /Applications/XAMPP/htdocs
$ sudo ln -s ~/Sites/arx_test arx_test
$ ls -la
```

The App Overview:

This application meets a single need. The Office Manager calls each patient and asks them which song is their favorite before the surgery. When coming out of surgery a patient will want to hear their favorite song as they are in recovery. The office has chosen to use the iTunes Music API for song selection results. The DBA has already set up your database and filled in the patient data. Your app must have three pages:

1. A list of patients
2. A page to assign a patient a favorite song
3. A report page

The website is mostly built, but the developer who started it got pulled off to work on another project. Not only did he not finish it, he left a few small bugs here and there. Furthermore, there was never a designer or UI person on the project at all.

Per Page Requirements:

1. Patient Listing Page

- *Done*: Must show a list of patients
- *Todo*: The page needs to be easier to navigate
  - So, either make it sorted by patient name and have it paginated, 10 per page
  - OR, make it use a JavaScript filter to search by name
  - If you use AJAX, try to use the `ajax_controller.php` in the manner `save_song_for_patient()` works.

2. Song Selection Page

- *Done*: This page takes a patient_id as a `$_GET` argument.
- *Done*: The page shows who the chosen patient is
- *Done*: The page allows the user to search for any song by name
- *Done*: When a song is selected, it is added to the database `songs` table
  - This `song_id` is then stored in the `patient` record under `favorite_song_id`
- *Done*: Song Data is stored as **JSON** in `songs.song_data`
- *Todo*: Song Data Cannot Be duplicated in the Database
  - You'll need to fix this - hint: use the hash (see the MYSQL trigger on that table)
  - If a user changes songs, make sure to delete the old song record if nobody else is sharing it
- *Optional*: the ability to preview the song (who wouldn't want to jam out at work?)

3. Report Page

- *Todo*: A list of all patients
- *Todo*: You can pick which report to make:
  - The song each patient chose, List the Artist, Album, Link to Buy the Song
  - Or, think of something interesting involving Genre and Age
- *Todo*: The client has specifically asked that we incorporate a Rainbow or Unicorn into the Design of this page
- *Optional*: add a play button so the user can preview songs
- *Optional*: show the Album Cover Image

Final Requirements

- *ADVICE*: If you aren't sure how to handle something - use your best judgment.
- *Todo*: Comment Each Function in the App Clearly
- *Todo*: Format your code as well as you can! (We prefer spaces not tabs)
- *Todo*: Make a file called `developer_notes.md` containing:
  - **What you did and why**
  - **What you would do given more time and a larger budget**
- *Optional*: Feel free to go as crazy as you like - there is no wrong answer.
  - You can add more JavaScript plugins
  - You can use tons of CSS, or even CSS frameworks
  - You can redo the whole thing…
- *Optional*: It's a good practice to avoid showing users the primary keys for your tables
  - Either implement a system to avoid doing this, or explain how you would do it.
- *Optional*: You can even use a full blown PHP MVC framework such as CodeIgniter if you like
- *Optional*: (ノ °□°) ノ ⁀ ┻━┻