Guitar Haven

Jacob DeBard	client | server | database
Jake Ford	client | server | database
Obaid Ameen	client | database
Taylor Childers	client | UI | counselor

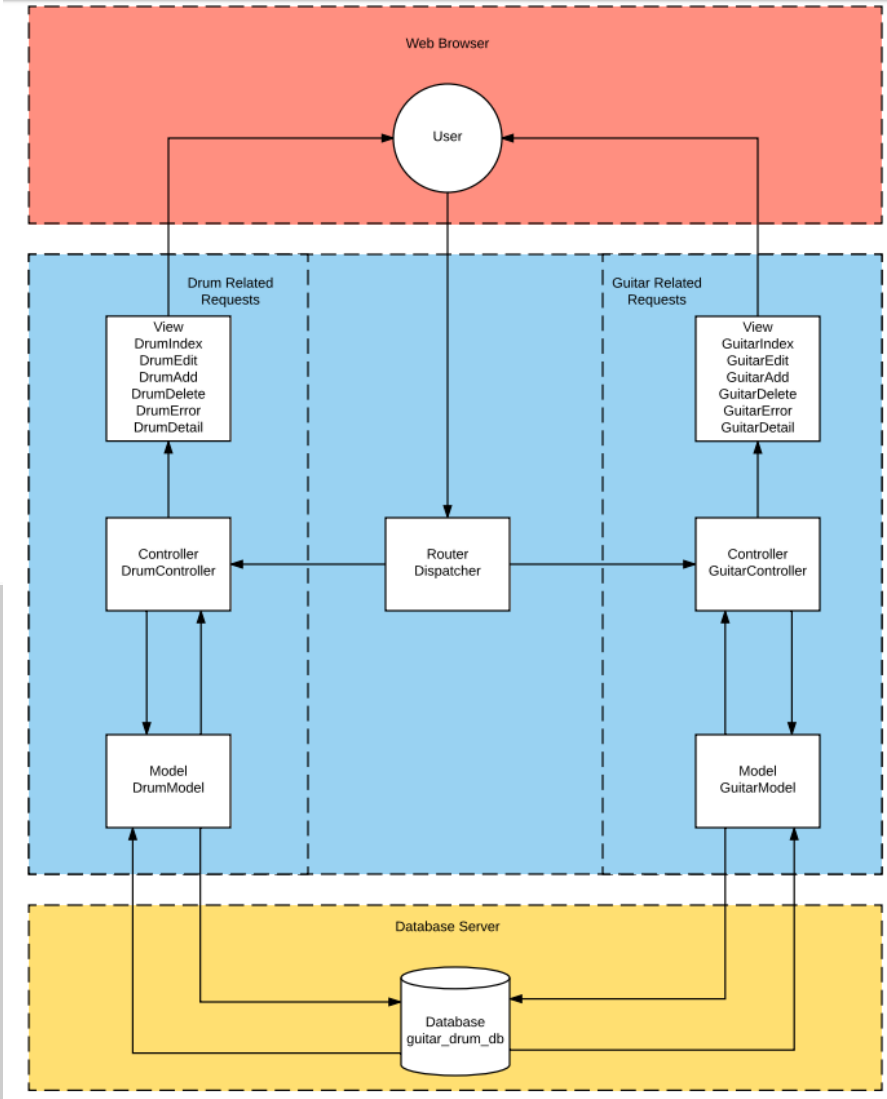
Table of Contents:

Introduction

Guitar Haven is the exclusive shop for musicians looking to purchase quality drum and guitar sets online. Using a dynamic database and an MVC file-framework the user experiences a seamless cohesive dynamic site with a responsive and interactive interface. When the user navigates to the homepage, they are greeted with the option to either shop for guitars or drums. As with every page, the home page also shows the header, which is extended to every view page, to provide consistency within the site. The user would have the option to direct to the Contact or About Page as well as navigate to the Guitar or Drum shopping pages. In addition the user has the ability to log in, which benefits a site admin who might need to do some updating of some items in the shop. When the user selects either drums or guitars to shop, they are given a list with all the items in the store. A search bar is present if the user knows what they are looking for, and can do a quick search to find their product. The site is able to make asynchronous AJAX requests while the user is searching to suggest possible results the user was looking for. If the user is a registered admin, they are also able to add a new instrument into the database from this page. (If they are looking at drums they can only enter a drum, and can only enter a new guitar if they are looking at guitars.) Selecting an instrument takes the user to the details page. There they can review the details of the instrument to decide if they want to buy. From this page, if a user is an admin, they can edit, or delete the current instrument from the database.

Classes

Guitar Haven is programmed with the Object Oriented Approach (OOP), with the MVC framework, meaning each page is rendered using one or more classes that inherit and extend each other. Below is an MVC diagram as well as a breakout of every class that is used within the project.



Views

- IndexView: This class contains two functions which creates the header and footer which is extended to every other view class to display the same header and footer to each class. This is also where all CSS and JavaScript files are loaded.

- WelcomeIndex: This class extends IndexView, and is also the home page, which is called by the dispatcher as the default page to load whenever a specific page has not been requested.
- DrumIndexView and GuitarIndexView: These pages extend IndexView, to pass the header and footer to child classes. Here is where the searchbox is implemented. This is how the AJAX is able to distinguish between guitars and drums when searching the database.
- DrumIndex and GuitarIndex: These views extend DrumIndexView and GuitarIndexView respectively. Both have a single display function called display. Since the controller will pass in an array to the function, a single variable is passed in which contains an array of instruments to display. The purpose of this function is to display all of the instruments provided to it.
- DrumEdit and GuitarEdit: These are forms which extend DrumIndexView and GuitarIndexView respectively. These classes will present the user with the forms to input data to be inserted into the database. They store the values in a post variable and sends instructions to the controller to get the information added.
- DrumAdd and GuitarAdd: Simiar to the Edit classes, these retrieve information from the user and calls the controller.
- DrumDelete and GuitarDelete: These classes are called after the
- DrumError and GuitarError: These error classes extend DrumIndexView and GuitarIndexView respectively. Both have a single function with a single variable that is passed from the controller called display. If there is any error when information is being passed back and forth between the views, models, and controllers the controllers will call this class and pass some kind of error message to inform the user what happened.
- DrumDetail and GuitarDetail: These views each extend DrumIndexView and GuitarIndexView respectively. Both have a single function that is called by the controller called display. This time however the variable passed is a single instance of the Drum or Guitar Class, which contains all of the detailed information. This view will display this information to the user.

Models

- Drum and Guitar: These classes provide the framework for the Drum and Guitar objects. They contain an id, image, name, type, description, YouTube link, and price.
- DrumModel and GuitarModel: These models receive instructions from the controllers in order to pull specific information from the database. They can pull all the drums or guitars out of the database, pull a specific drum or guitar out of the database, or provide a more refined search of the database, using some criteria provided by the user. They can also add, edit and delete information in the database.

Controllers

- DrumController and GuitarController: These controllers have several methods which communicate to models and views to display information. Detailed below is the breakdown of each method:
    - index: this method does not have a variable pass in, but makes a request to the Drum or Guitar model to return all of the Drums or Guitars in the respective table. As long as there were no errors, this information is passed into the Drum Index view to display all of the Instruments.
    - detail: Once a user clicks on a specific instrument to view, this method will be called in order to rout to the correct one. The controller will pass the id to the model to look up that specific instrument and then when it gets the information back it will create a new instance of the Drum/GuitarDetail view class. This method will then pass the information provided by the model to the display method of the the detail object.
    - error: If for any reason the controller detects an error with information being passed from the views and models, it will stop and create a new instance of the error view class and call the display method. It will also pass an informative message into this method to explain what happened.
    - add: this method accepts SuperGlobal post variable information and sends this info to the model to create a new database object
    - edit: this method accepts SuperGlobal post variable information and sends this info to the model to edit an exiting database object
    - delete: this method accepts SuperGlobal post variable information and sends this info to the model to delete an exiting database object
    - search: This method finds the search terms stored in the SuperGlobal Get variable and sends them to the model in order to locate instruments meeting that search criteria. Once data is returned it creates a new instance of the Drum/GuitarSearch view and calls the display method, passing the array of instruments to the view.
    - suggest: this method utilizes the autoloader.js file to return an JSON array of values to be used in a view. Suggest receives instruction from the Guitar/DrumIndexView search form, and sends and receives information from the search method of the model
- WelcomeController: This controller has only one method which creates a new instance of the WelcomeIndex class and calls the display method.

Application

- Autoloader: The autoloader is called as part of the index.php file. This class looks throughout the project to find any classes to instantiate
- Database: Database set up all of the details for the database connection. It provides names for all the needed tables, connection strings, and credentials. This is used in the model classes.
- Dispatcher: This class has a single function that is responsible for dissecting the pieces of the requested URI and routing the request to the proper method of the matched controller. ie: \drum\detail\1 would be dissected into the drum controller, the detail method, and the variable 1 would be passed.

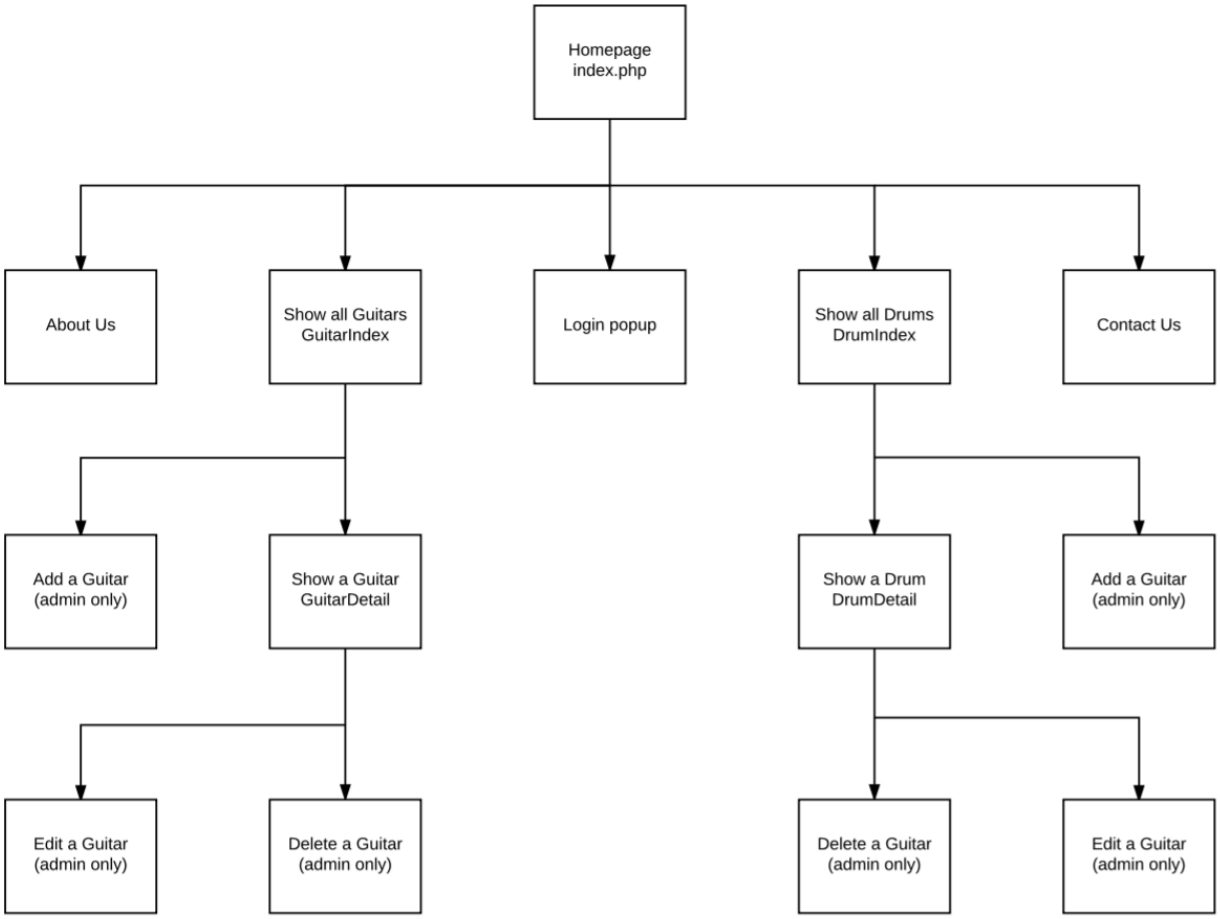
Features/Functionality

- Asynchronous AJAX request: Part of the search functionality includes AJAX –enabled asynchronous autocomplete that attempts to help the user find what they are looking for faster. Much like Google's autocomplete, this will attempt to finish the word a user starts typing.
- Add/Edit/Delete database objects: Users with admin privlages are able to add/edit/and delete information stored in the database from a friendly front-end UI.
- Users have two different catalogs to search from (guitars and drums) and can shop for the product they love the most.

Error and Exception Handling

The errors are handled within a class that is named with the specific error. These errors extend the error class. All errors are handled within the model during database altering or inserting. This way no inaccurate data is inserted. The site validates all fields that are inserted and even validates to make sure email addresses and YouTube links are valid. This level of error prevention is intended to provide a seamless user experience.

Site Map



Final Thoughts and Future State

Part of what we would want to implement as a future enhancement would be working cart. We focused primarily on the objects themselves and making all of the necessary functionality available. We would also want to have a way to better connect the users to the site with a way to upload videos using the instrument they bought so other users can see what the instrument sounds like and help musicians connect with each other.