**Lab 1 - Laravel**

**Two-page DVD search with Laravel**

Build a two-page search module using the dvd database. You will be building search and results pages.

**Search**

Route: /dvds/search

This route should display a form with a text field, 2 drop-down menus, and a submit button. The drop-down menus need to be populated by database queries (i.e. list of Genres and Ratings directly from the database):

* Every SELECT menu should be populated by queries
* Text box for "title"
* Select menu for genre
* Select menu for rating
* Submit button

This form should make a GET request to the results page.

Create an "All" entry in the genre and rating drop-down menus.

**Results**

Route: /dvds

* Dynamic query builds a list of dvd results based on the three form fields from the search page. This should use [Laravel's Query builder](http://laravel.com/docs/queries)
* Results are listed out in formatted columns (using divs or tables), with display columns for:
* Title
* Rating
* Genre
* Label
* Sound
* Format
* Release Date (formatted) -- See <http://www.w3schools.com/sql/func_date_format.asp>

Use some conditional logic that only applies the genre and rating form data if the field is NOT "All".

**Models and Controllers**

* Create a model called *Dvd* to handle all your database code
* Create a Dvd controller for your 2 routes.

**DVD Pages with Eloquent**

**DVD Search**

Take your previous DVD Search page assignment, and modify the query to use Laravel's ORM instead of the Query Builder. Be sure to:

* define relationships for related models / lookups (rating, sound, genre, label, format)
* only display 30 records on the page

To keep things simple, you can display *dvds.release\_date* as the raw mysql datetime format.

**DVD Insert page**

Create the route **GET /dvds/create**. This should display a form to insert a new dvd into the database. Create dynamic drop downs for label, sound, genre, rating, and format. Use Eloquent to populate these select menus.

Next, create the route **POST /dvds**. This should add a new record in the dvds table and redirect the user back to **GET /dvds/create**.

Add a flash message and display on the page saying that the record was inserted successfully with a green background color.

**Optional: DVD Listing by Genre**

Make each record's genre a link to **GET /genres/{id}/dvds** where {id} corresponds to the genre ID. When the link is clicked, you will display a page that states that genre at the top and a list containing all DVDs of that genre.

1. First find the Genre model based on the ID.
2. Next, define a hasMany() relationship to display the many DVDs that a genre model has.

If you miss a point elsewhere in the assignment but do this optional part (and it works), I will give 1 point back.