**SI 364 Midterm**

**Description**

A simple application utilizing a movie database.

Movie info -> Get information about a movie (above a 5 IMDB rating)

Creates two tables: Movie (Name, year, director)

Director (Name)

Compare with IMDB -> Test how close you can rate a movie with its actual IMDB rating

\*\*Check comments on canvas for api keys and code to input/modify

**List of routes with templates**

* http://localhost:5000/ -> base.html
* http://localhost:5000/movie -> form.html, movieresult.html (Form w/Post method to same page)
* http://localhost:5000/movieform -> movieform.html
* http://localhost:5000/movieinfo -> movieinfo.html
* http://localhost:5000/moviessearched -> moviessearched.html
* http://localhost:5000/directorssearched -> directorssearched.html

Code Requirements

* **Ensure that the SI364midterm.py file has all the setup (app.config values, import statements, code to run the app if that file is run, etc) necessary to run the Flask application, and the application runs correctly on http://localhost:5000 (and the other routes you set up)**
* **Add navigation in base.html with links (using a href tags) that lead to every other viewable page in the application. (e.g. in the lecture examples from the Feb 9 lecture, like this )**
* **Ensure that all templates in the application inherit (using template inheritance, with extends) from base.html and include at least one additional block.**
* **Include at least 2 additional template .html files we did not provide.**
* **At least one additional template with a Jinja template for loop and at least one additional template with a Jinja template conditional.**
  + **These could be in the same template, and could be 1 of the 2 additional template files.**
* **At least one errorhandler for a 404 error and a corresponding template.**
* **At least one request to a REST API that is based on data submitted in a WTForm.**
* **At least one additional (not provided) WTForm that sends data with a GET request to a new page.**
* **At least one additional (not provided) WTForm that sends data with a POST request to the same page.**
* **At least one custom validator for a field in a WTForm.**
* **At least 2 additional model classes.**
* Have a one:many relationship that works properly built between 2 of your models.
* **Successfully save data to each table.**
* **Successfully query data from each of your models (so query at least one column, or all data, from every database table you have a model for).**
* **Query data using an .all() method in at least one view function and send the results of that query to a template.**
* **Include at least one use of redirect. (HINT: This should probably happen in the view function where data is posted...)**
* **Include at least one use of url\_for. (HINT: This could happen where you render a form...)**
* **Have at least 3 view functions that are not included with the code we have provided. (But you may have more! Make sure you include ALL view functions in the app in the documentation and ALL pages in the app in the navigation links of base.html.)**
* **Additional Requirements -- an app with extra functionality!**
* Include an additional model class (to make at least 4 total in the application) with at least 3 columns. Save data to it AND query data from it; use the data you query in a view-function, and as a result of querying that data, something should show up in a view. (The data itself should show up, OR the result of a request made with the data should show up.)
* **Write code in your Python file that will allow a user to submit duplicate data to a form, but will not save duplicate data (like the same user should not be able to submit the exact same tweet text for HW3).**