CMSC 388J Final Project Proposal

Group Members (1-4 members):

`Select your group members in gradescope!!!`

Directions:

Read the project specifications and fill out all the questions **in gradescope!**

> [!IMPORTANT] All the answers to the questions should be submitted to the gradescope submission with all of your group members selected

Logistics

There are 4 things you will submit for the final project (only 1 group member has to submit, but select all members names in gradescope):

1. Proposal
2. Project code
3. Writeup (submit when you submit your project's code)
4. Pretty Project (Extra credit due at the end of the semester)

Both the Proposal and Writeup use this document as a template (your writeup will likely be the same or similar to your proposal depending on how much has changed since your proposal document).

Due dates (unless specified elsewhere):

Proposal: April 12, 2024

- We **highly recommend** you complete this as early as possible so you have more time to work on the project. We will review your proposal within 1-2 days of submission.

Overview

The final project for this class is to create a Flask app in a group of 1-4. You have a lot of freedom for this project as long as you meet the requirements.

You're welcome to use Project 4 (or any other project from this course) as a base (though you're not required to–in fact, we encourage you to try creating something from scratch). *If you choose to use a course project, you need to make a "substantial" change. Examples of "substantial" changes include:*

- Using a different API (instead of the OMDB API) + minor feature to demonstrate knowledge of Flask
- It's no longer a "review site" but something else
- It's still a "review site" but you add a major feature
- Examples: ability to reply to reviews

Use your best judgment here but reach out to course instructors if you're unsure.

```
Description of your final project idea:

This project will revolve around reviewing, rating, and selecting favorite songs. Users can log in and l

Users can search for songs based on name, and will be presented with a list of results using a API like
and leave reviews / ratings for future users. When users select a song, that can see all the informatio
```

Requirements

Note that some of these requirements overlap with each other so some features may satisfy multiple requirements.

Registration and Login:

- There needs to be some sort of user control: logging in, registering, logging out.
- Certain features should only be available to logged-in users.

```
Describe what functionality will only be available to logged-in users:

Logged in users will get the chance to do the following
 --> Leave reviews for the songs that they have listened to.
 --> Save songs as "favorites", and be able to view their favorite songs
 --> View an account page where they can see their user information including their favorites
 --> Update their account information (update username, update email)
```

Forms:

- At least 4 forms (can include registration and login forms)
- Must be CSRF protected

```
List and describe at least 4 forms:

Form 1:
--> Registration Form
  allows users to insert an email, username, and password. Usernames and emails that are already taken w

--> Login Form
  will asks users for a username and password. If correct, they will be brought to an account page speci


--> Review Form
  Each review will have a title and a body. The form will ask for these two bits of information and wil
  present a submit button for users to send that review


--> Profile Update form
  Will allow logged-in users to update their profile information such as username and email address.


--> May implement some more forms, but cannot think of anything else at the moment...
```

Blueprints:

- Must have at least 2 blueprints
- Each blueprint should have at least 2 visible and accessible routes

```
List and describe your routes/blueprints (don't need to list all routes/blueprints you may have-just en

Blueprint 1
 - Profile / User blueprint.
  --> This blueprint will group together the user managment views on the website. For example
      some routes for this blueprint may include "register", "account", "update_account", "login".

      @users.route("/update_account", methods=...)
      @users.route("/account/<user-id>", methods=...)
      @users.route("/favorites", method=...)


- Songs blueprint
  --> This bluepring will group together the song management views on the website. For example some rou
```

```
      for this bluepring may include

      @song.route("/song_information", methods=...)
      @song.route("/reviews", methods=...)
```

Database:

- Must use MongoDB

```
Describe what will be stored/retrieved from MongoDB:

In my MongoDB I will be storing the following:
  --> User information such as username, email, passwords (hashed), favorite songs list
  --> Reviews (title, body, user_id, song_id)
```

Another Python Package or API:

- Find and use another Python package or API.
- Must be a package/API we haven't used in any of the projects (though anything mentioned in lecture material that wasn't used in a project is fair game).
- You can use a package/API we've already used if you're using it in a way that's *very* different from how we used it in the projects.
- Must affect the user experience in some way.

Examples (feel free to use these or come up with your own):

- Flask-Mail to send emails to users
- CalorieNinjas API with Requests package to access the API
- Spotify API
- Requests package to display data retrieved from an HTTP request
- BeautifulSoup4 to display data parsed from a website
- SciPy, NumPy, SymPy, etc
- Plotly

- Discord OAuth
- CAS

```
Describe what Python package or API you will use and how it will affect the user experience:

I will use the Spotify API to get songs and information about each song. I will be displaying the songs
name
```

Presentation:

- Doesn't have to be pretty but it needs to be usable.

  [!NOTE] Theres going to be a EC part of the project due on the last day of the semester where you can use tailwind/react/svelte/css to make your website more pretty, up to 25%!!!

Grading

Requirement

Points

Proposal submitted

100

Writeup submitted (same format as the proposal)

100

Registration and Login

75

Forms

50

Blueprints

50

Database

50

Another Python package or API

75

Total: 500 points