Go Jose!
Gordon Mo, Joshua Liu, Selena Ho

Topic: Taylor Swift

Program Components:

- HTML:
  - Main Page
    - General taylor swift biography
    - Links to other pages within a table of contents
    - Includes links to other Taylor Swift related websites
  - Individual pages for different albums
  - Each page has a navigation bar linking it to all of the other pages
  - Within each page, there's an edit button that links to an edit page only if you're logged in
    - If not logged in → user is redirected to a login page
  - Edit page has a textarea from an HTML form and within the textarea there's the current HTML for the page that the user wants to edit
- Flask
  - Used for when users want to edit a page
  - Using templates, Flask puts the current page content within a textarea in an HTML form
  - a POST request is used to save and display the user's edits by updating the databases content data
- SQLite3
  - Used to create the tables that will store the information
    - Three Tables
      1. Username and Password
      2. Page Name and URL
      3. Content table for each page and its sections
         - Content data is the HTML of the pages
         - As the user edits the web page from the browser and submits their edits, the content data in the database will be updated to have the HTML that the user updated
- Multiple Python files
  1. One file dedicated for Flask related actions
  2. One file for database updating (when username/password added for example)

Tasks Required For Project Completion:

- Create HTML files for each of the pages (can start with just one or two albums and bare minimum content in each of the pages and as SQLite and Flask starts working with the minimum number of pages add information and pages after)
- Create python file for Flask
- Create python file for database updating

- Add in username and password stuff last so that it'll be easier to test out editing/updating the web pages

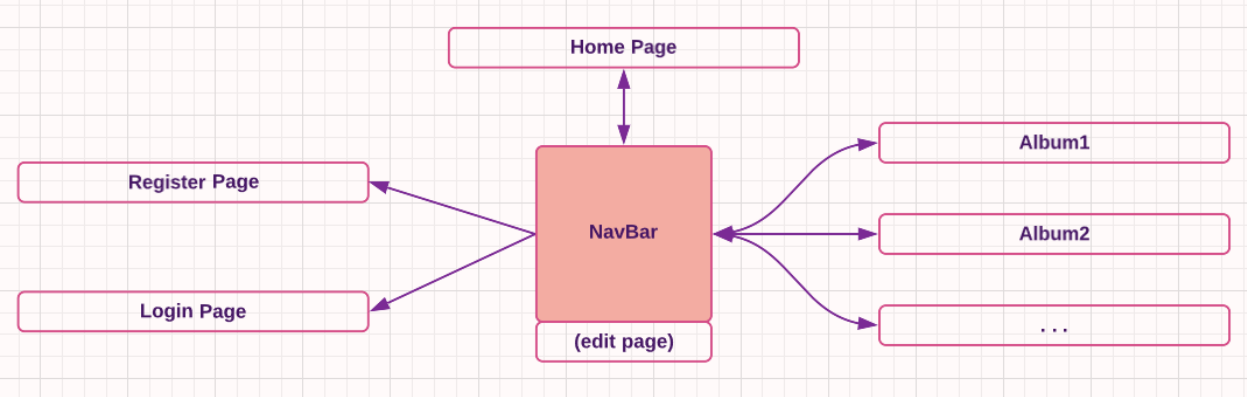
Breakdown of Tasks:

- Joshua does Flask
- Gordon does SQLite
- Selena does HTML
- But, each of the different tasks are connected anyways so everyone will be working on different aspects that aren't directly assigned to them

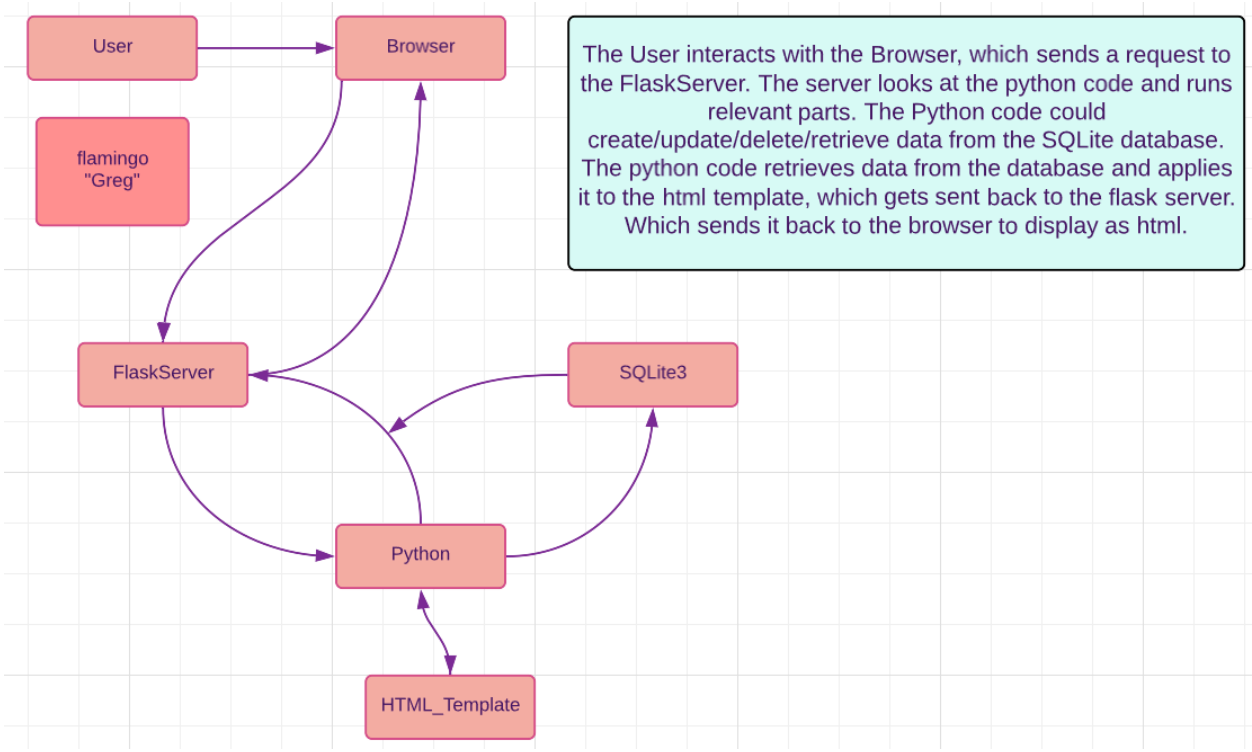
QCC:

- Can we python3 another python file through a python file? Import python file into main python file

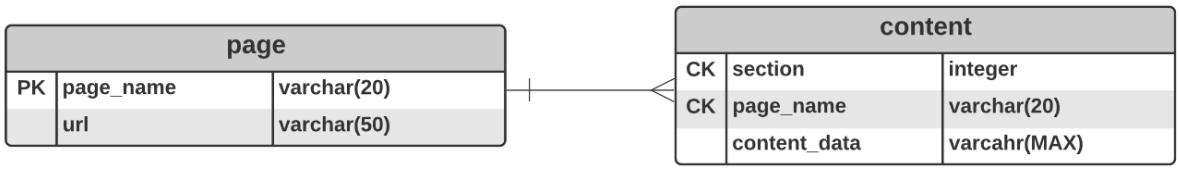
SITE MAP:



CONNECTION MAP:



- What if: (hopefully this makes sense)



Composite Key:

```
CREATE TABLE something (
  column1,
  column2,
  column3,
  PRIMARY KEY (column1, column2)
);
```

We have 2 tables,
one to store the ‘page’:
A wiki about animals would have a page for each animal:
wiki.com/dog
wiki.com/frog

And another to store the ‘content’ for the page:

Contents [hide]

1 Evolution and classification
- 1.1 Definition
- 1.2 Dinosaurs and the origin of birds
- 1.3 Early evolution
- 1.4 Early diversity of bird ancestors
- 1.5 Diversification of modern birds
- 1.6 Classification of bird orders
- 1.7 Genomics

2 Distribution
3 Anatomy and physiology

Much like how wikipedia has sections for each part of the content, we can have a table with the section corresponding to a certain page.

eg. Page “dog” has section 1 with certain content, and another section with more content.

To Update content:

Every section has a edit button next to it(if you’re logged in), that sends you to the endpoint /edit (eg. wiki.com/dog/edit?page_name=dog&section=1).

This page will give the user an html form with input type ‘textarea’. Flask will then put the current content within the textarea box. THe user can change this then send it back to flask, which will update the database.


```
<form action="/auth">
    <textarea name="username" type="text"></textarea>
```

To add and delete section:

To add a section, you do stuff. . .

Similar to changing a section, we could have a button to add a section to the db, which adds the section to the table and lets you put stuff in it at the same step.