Music Teleportation

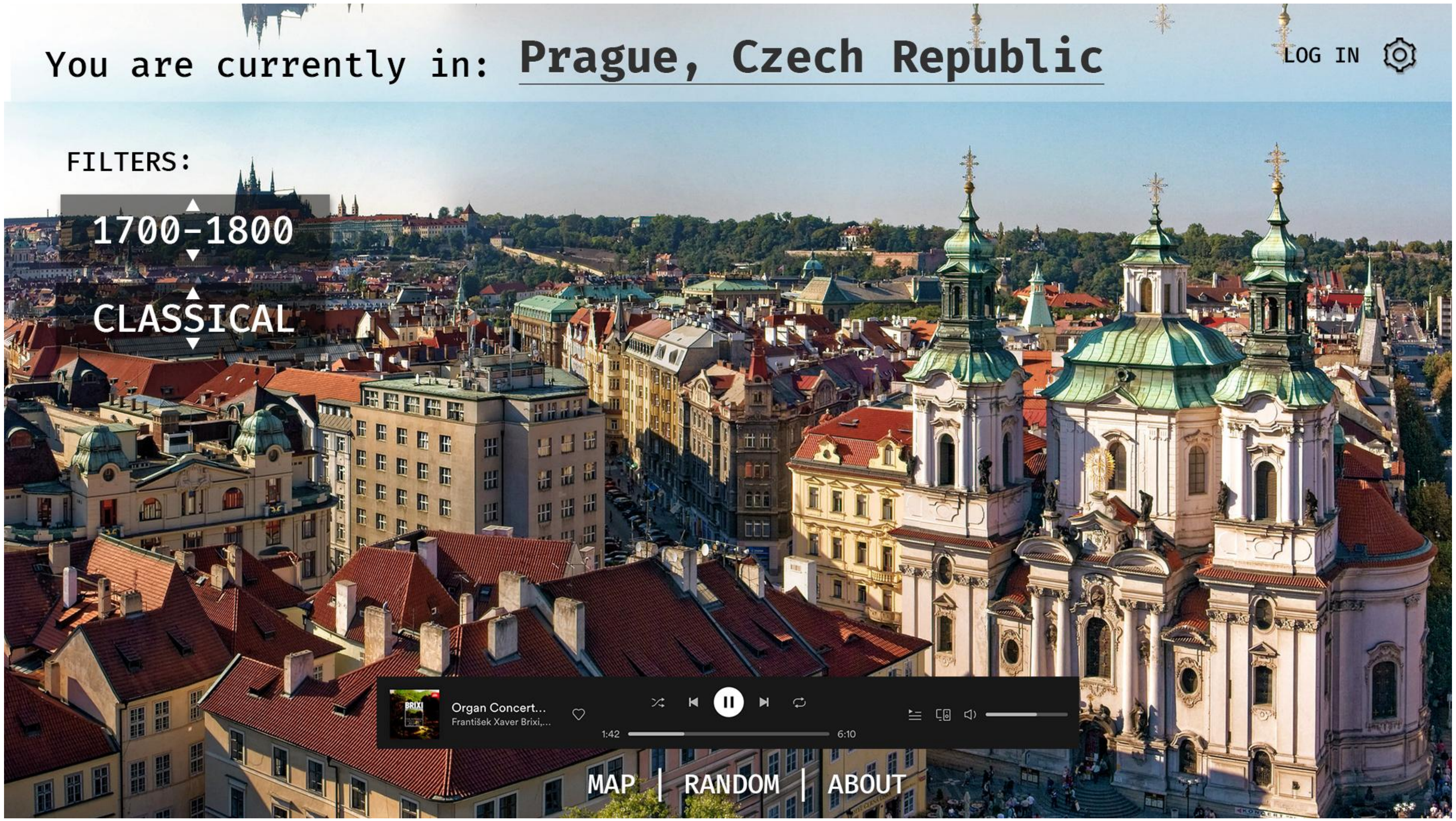
Alex Thomas

Joaquin Quintana

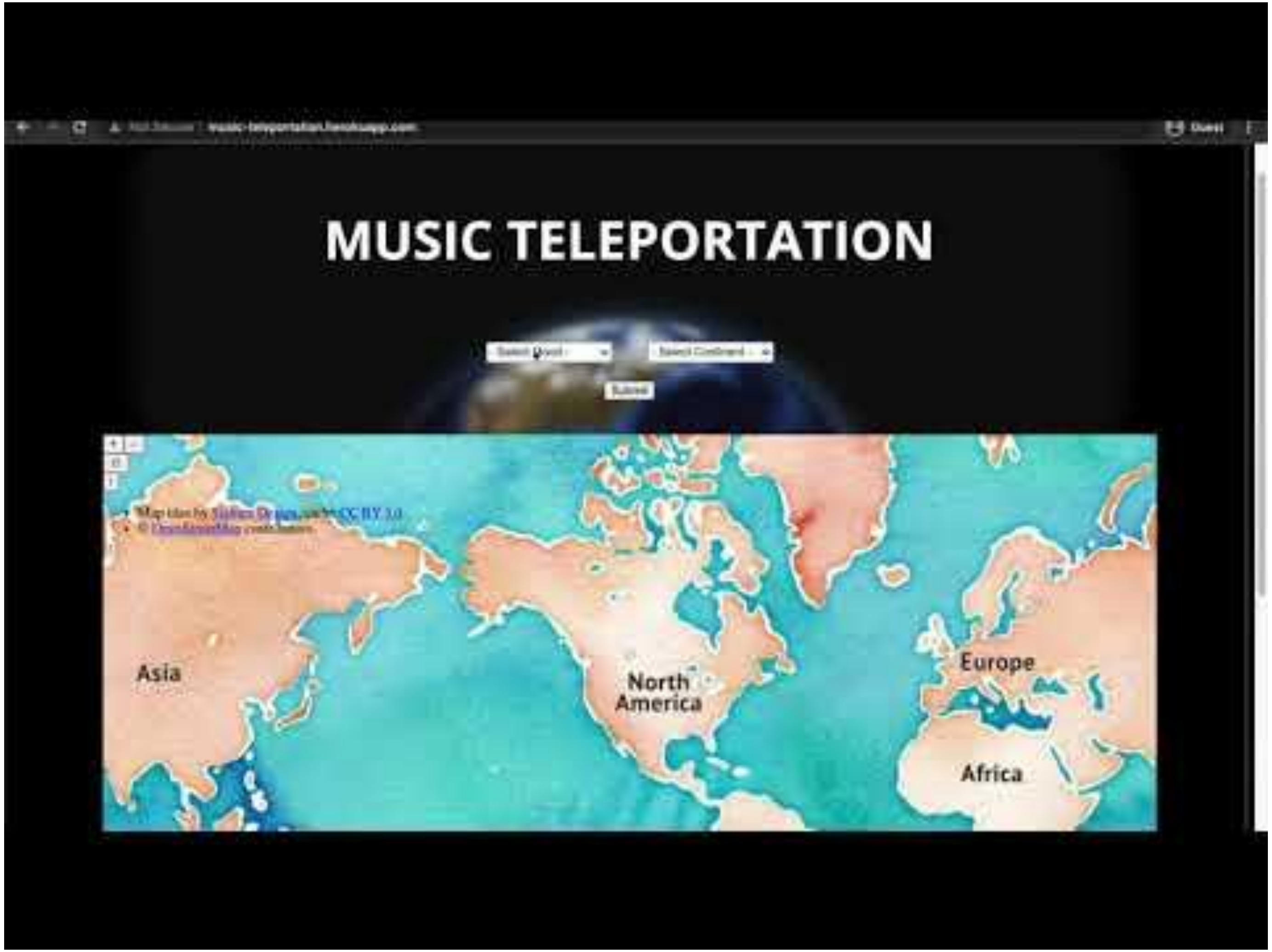
Timur Boskailo

Samuel Steingard

Concept


You are currently in: Prague, Czech Republic
LOG IN
FILTERS:
1700-1800
CLASSICAL
BRIXI
Organ Concert...
František Xaver Brixi,...
1:42
6:10
MAP | RANDOM | ABOUT

Demo

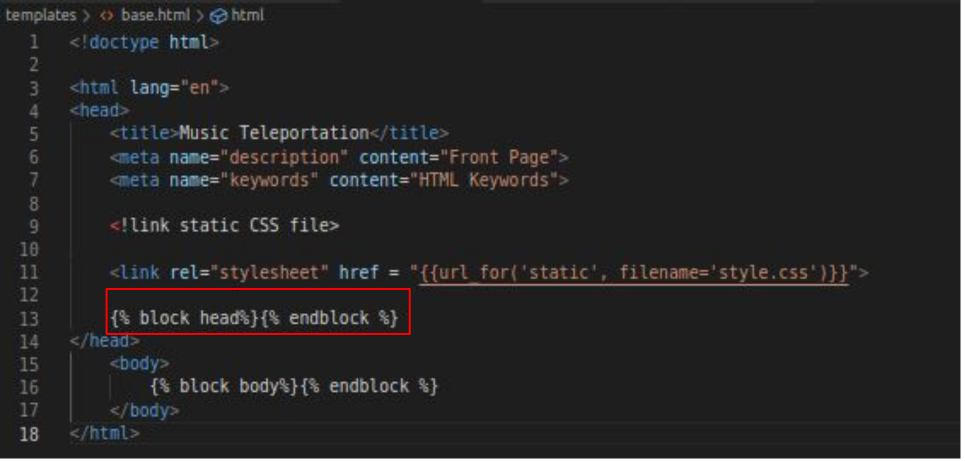

music-teleportation.herokuapp.com
MUSIC TELEPORTATION
Select Mood
Select Continent
Submit
Asia
North America
Europe
Africa

Flask Infrastructure

- Flask - flask is a web development software which is integrated with Python, Jinja and Werkzeug.
    - Jinja - is a templating engine which has been developed to have a similar syntax to python
    - Werkzeug - is web server gateway interface (WSGI)

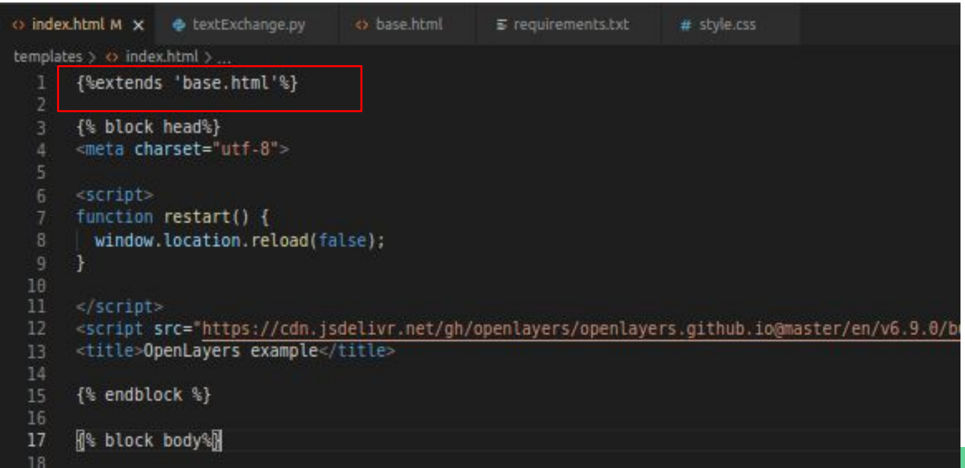
Using the Flask framework we were able to use python and a set of modules which made working with the backend and frontend a little cleaner.

Templates



templates > base.html > html

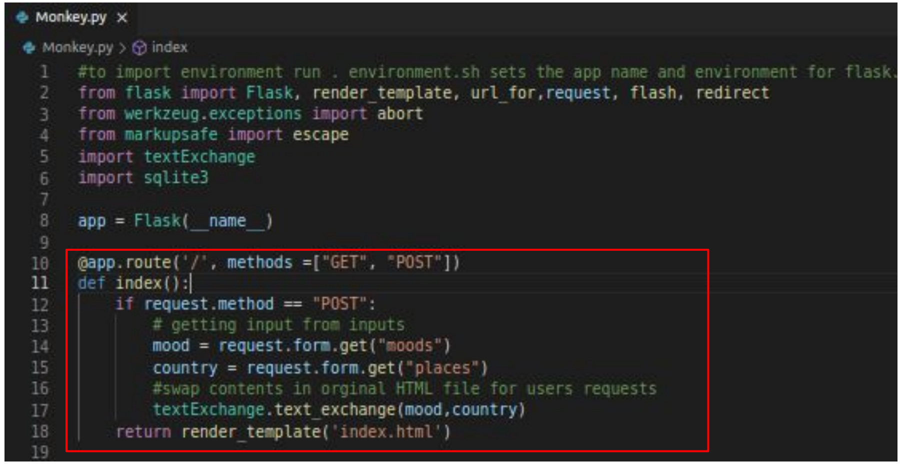
```html
<!doctype html>

<html lang="en">
<head>
    <title>Music Teleportation</title>
    <meta name="description" content="Front Page">
    <meta name="keywords" content="HTML Keywords">

    <!link static CSS file>

    <link rel="stylesheet" href = "{{url_for('static', filename='style.css')}}">

    {% block head%}{% endblock %}
</head>
    <body>
        {% block body%}{% endblock %}
    </body>
</html>
```



index.html M × | textExchange.py | base.html | requirements.txt | style.css

templates > index.html > ...

```html
{%extends 'base.html'%}

{% block head%}
<meta charset="utf-8">

<script>
function restart() {
  window.location.reload(false);
}

</script>
<script src="https://cdn.jsdelivr.net/gh/openlayers/openlayers.github.io@master/en/v6.9.0/b
<title>OpenLayers example</title>

{% endblock %}

{% block body%}
```



Monkey.py ×

Monkey.py > index

```python
#to import environment run . environment.sh sets the app name and environment for flask.
from flask import Flask, render_template, url_for,request, flash, redirect
from werkzeug.exceptions import abort
from markupsafe import escape
import textExchange
import sqlite3

app = Flask(__name__)

@app.route('/', methods =["GET", "POST"])
def index():
    if request.method == "POST":
        # getting input from inputs
        mood = request.form.get("moods")
        country = request.form.get("places")
        #swap contents in orginal HTML file for users requests
        textExchange.text_exchange(mood,country)
    return render_template('index.html')

```

Python file using the Flask infrastructure

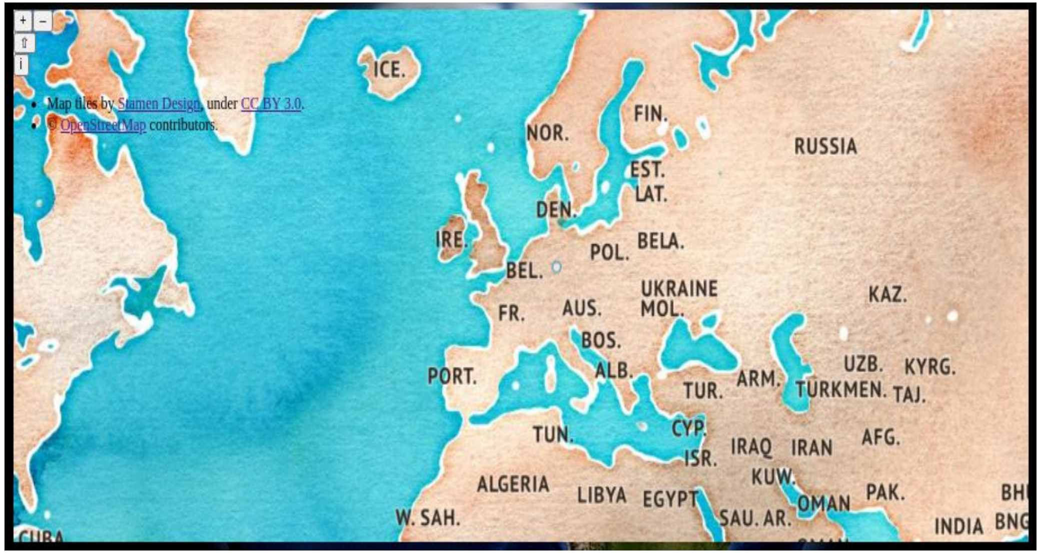
Static HTML is exchanged upon request using Pandas and Regex.

Challenge which I will talk about later

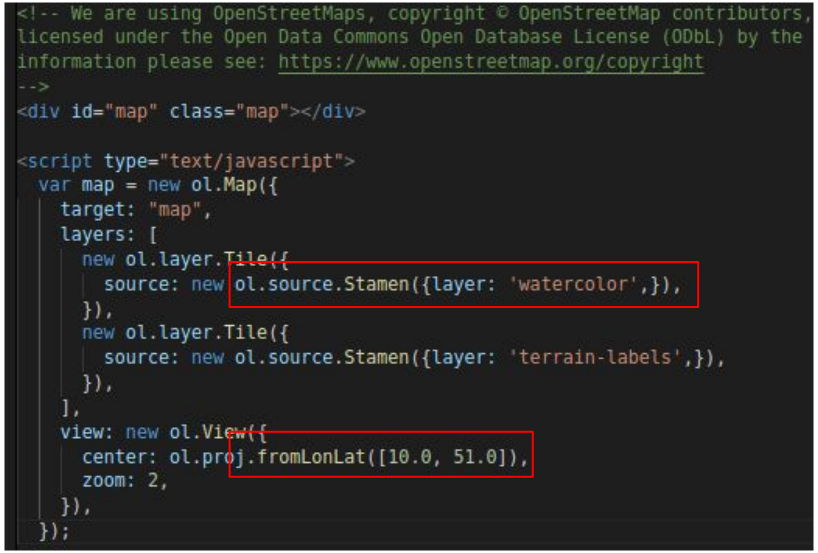
Web Applications

- OpenLayers - is an open source library primarily written in Javascript. The API provides an extensive environment for exploring dynamic maps.

- Spotify - We created a set of playlist which could then be stored as an iframe and exchanged dynamically upon users request.

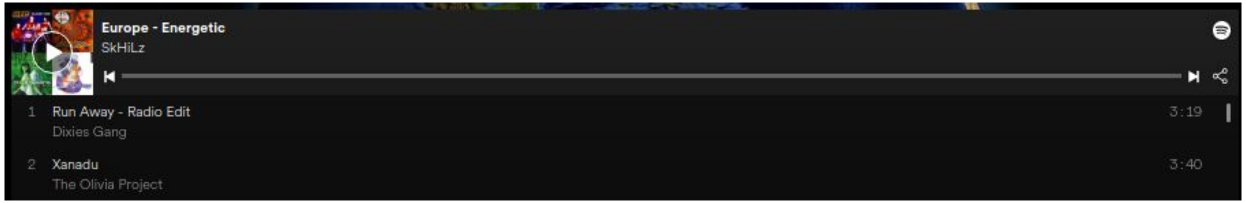


- Map tiles by Stamen Design, under CC BY 3.0.
- © OpenStreetMap contributors.



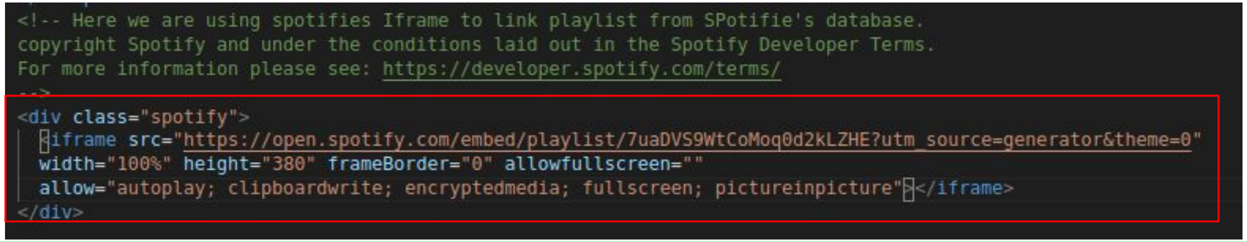
```
<!-- We are using OpenStreetMaps, copyright © OpenStreetMap contributors,
licensed under the Open Data Commons Open Database License (ODbL) by the
information please see: https://www.openstreetmap.org/copyright
-->
<div id="map" class="map"></div>

<script type="text/javascript">
  var map = new ol.Map({
    target: "map",
    layers: [
      new ol.layer.Tile({
        source: new ol.source.Stamen({layer: 'watercolor',}),
      }),
      new ol.layer.Tile({
        source: new ol.source.Stamen({layer: 'terrain-labels',}),
      }),
    ],
    view: new ol.View({
      center: ol.proj.fromLonLat([10.0, 51.0]),
      zoom: 2,
    }),
  });
```

Web map and Spotify Iframes



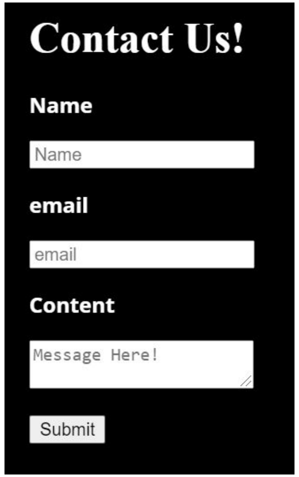
Europe - Energetic
SkHiLz

1	Run Away - Radio Edit Dixies Gang	3:19
2	Xanadu The Olivia Project	3:40



```
<!-- Here we are using spotifies Iframe to link playlist from SPotifie's database.
copyright Spotify and under the conditions laid out in the Spotify Developer Terms.
For more information please see: https://developer.spotify.com/terms/
-->
<div class="spotify">
  <iframe src="https://open.spotify.com/embed/playlist/7uaDVS9WtCoMoq0d2kLZHE?utm_source=generator&theme=0"
  width="100%" height="380" frameBorder="0" allowfullscreen=""
  allow="autoplay; clipboardwrite; encryptedmedia; fullscreen; pictureinpicture"></iframe>
</div>
```

Tool 2 - SQL/database


Contact Us!

Name

Name

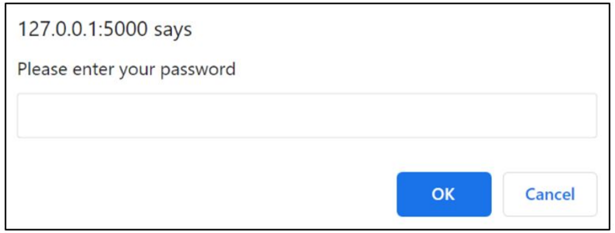
email

email

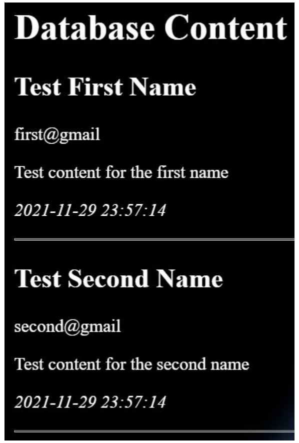
Content

Message Here!

Submit


127.0.0.1:5000 says

Please enter your password


Database Content

Test First Name

first@gmail

Test content for the first name

2021-11-29 23:57:14

Test Second Name

second@gmail

Test content for the second name

2021-11-29 23:57:14

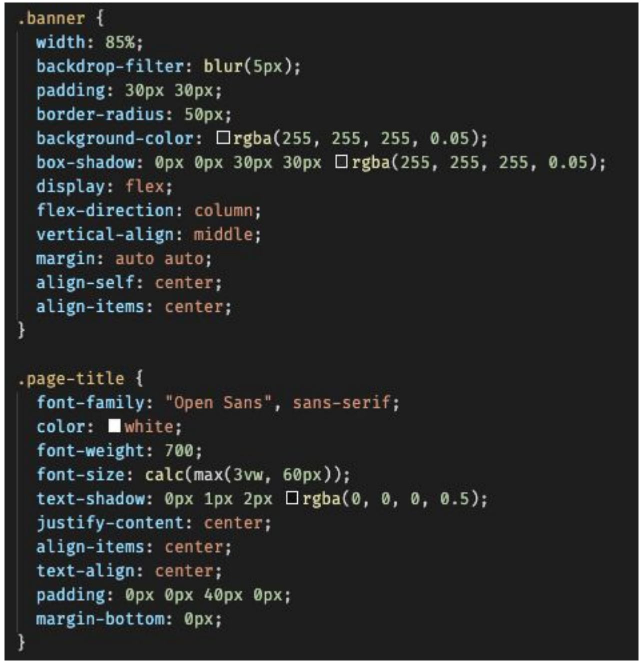
- method = post
  - INSERT INTO contacts
- database.db
  - schema.sql ➔ init_db.py

- javascript prompt
- OK (T) ➔ DB Admin
- OK (F) ➔ loop
- Cancel ➔ Home

SELECT * FROM contacts

Tool 3 - CSS

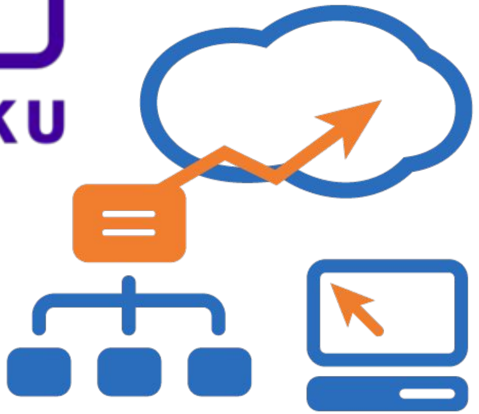
- CSS is a styling language that pairs with HTML
  - Natively understood by the browser - all webpages use CSS
- Sometimes hard to use
- Overall very useful and versatile



```css
.banner {
  width: 85%;
  backdrop-filter: blur(5px);
  padding: 30px 30px;
  border-radius: 50px;
  background-color: □rgba(255, 255, 255, 0.05);
  box-shadow: 0px 0px 30px 30px □rgba(255, 255, 255, 0.05);
  display: flex;
  flex-direction: column;
  vertical-align: middle;
  margin: auto auto;
  align-self: center;
  align-items: center;
}

.page-title {
  font-family: "Open Sans", sans-serif;
  color: ■white;
  font-weight: 700;
  font-size: calc(max(3vw, 60px));
  text-shadow: 0px 1px 2px □rgba(0, 0, 0, 0.5);
  justify-content: center;
  align-items: center;
  text-align: center;
  padding: 0px 0px 40px 0px;
  margin-bottom: 0px;
}
```

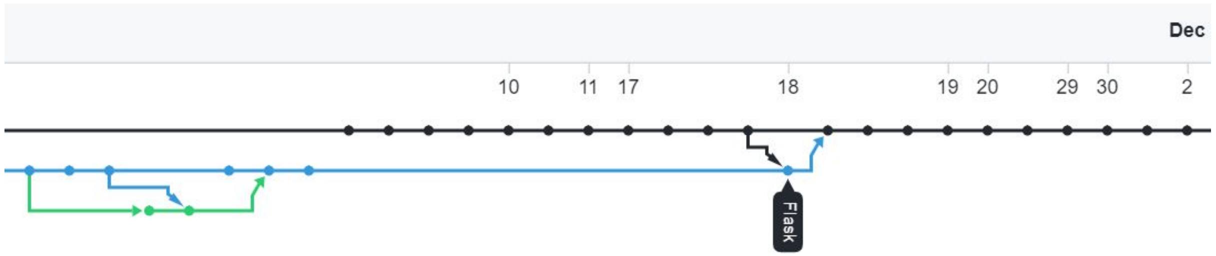
Tool 4 - Heroku

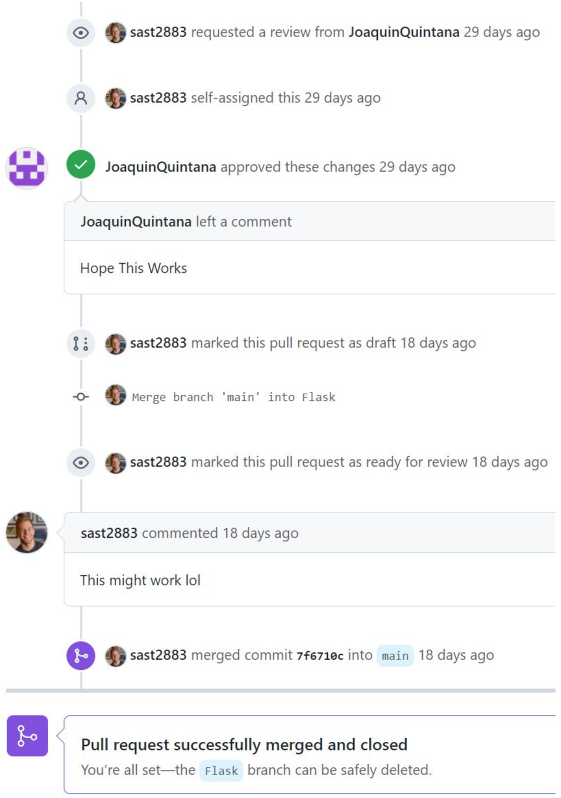
- Heroku is a PaaS (Platform as a Service)
- All wanted our app to be “live”
- Used in lab so implementation was easy
- Maintains version control for all users
- Very Useful!



Challenge 1 - Git(Hub)

- Git = excellent version control, surprisingly nuanced
- Learning curve for the process:
  - Fetch → Pull
  - Add → Commit → Push
- Many merge attempts from “Flask” branch into main




sast2883 requested a review from JoaquinQuintana 29 days ago

sast2883 self-assigned this 29 days ago

JoaquinQuintana approved these changes 29 days ago

JoaquinQuintana left a comment

Hope This Works

sast2883 marked this pull request as draft 18 days ago

`Merge branch 'main' into Flask`

sast2883 marked this pull request as ready for review 18 days ago

sast2883 commented 18 days ago

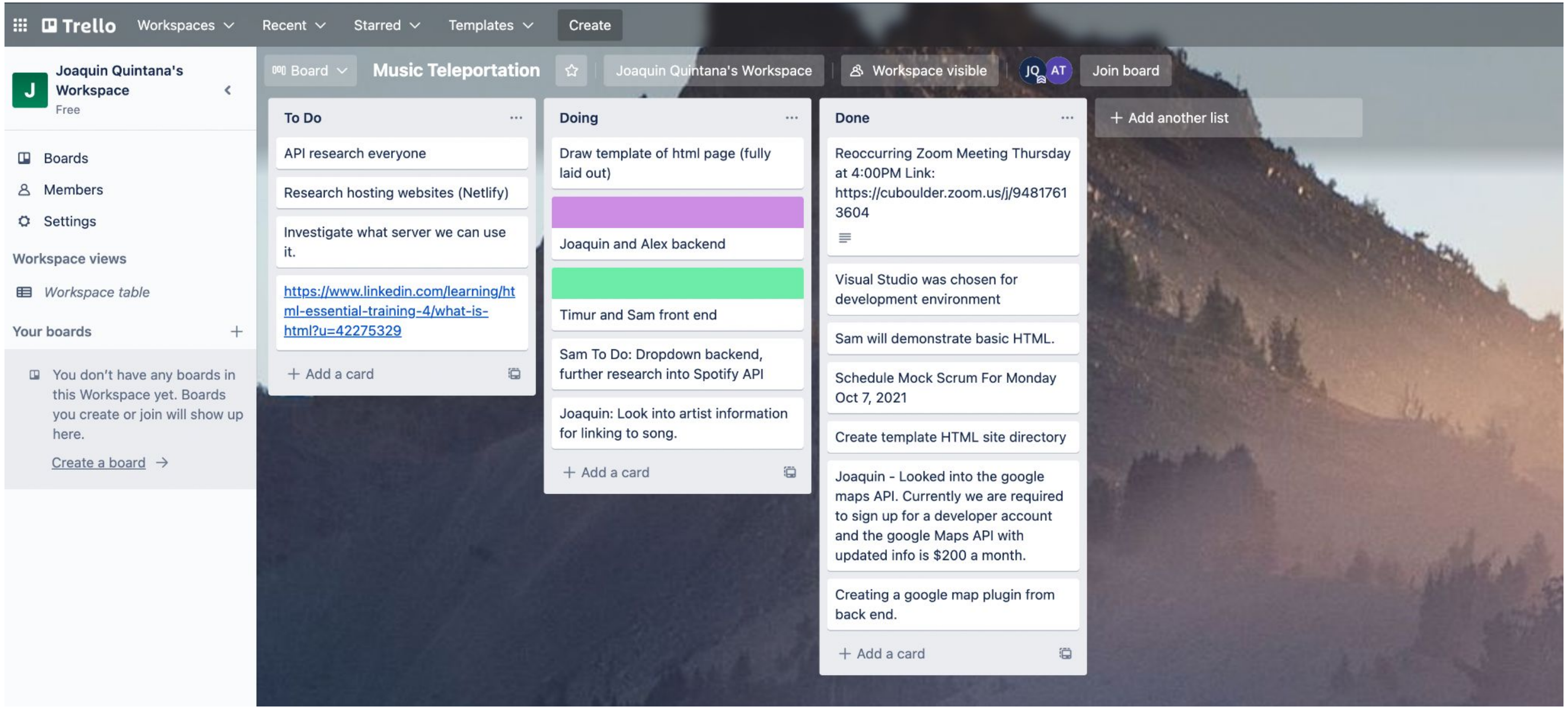
This might work lol

sast2883 merged commit `7f6710c` into `main` 18 days ago

Pull request successfully merged and closed

You're all set—the `Flask` branch can be safely deleted.

Challenge 2 - Project Tracking (Trello)



You don't have any boards in this Workspace yet. Boards you create or join will show up here.

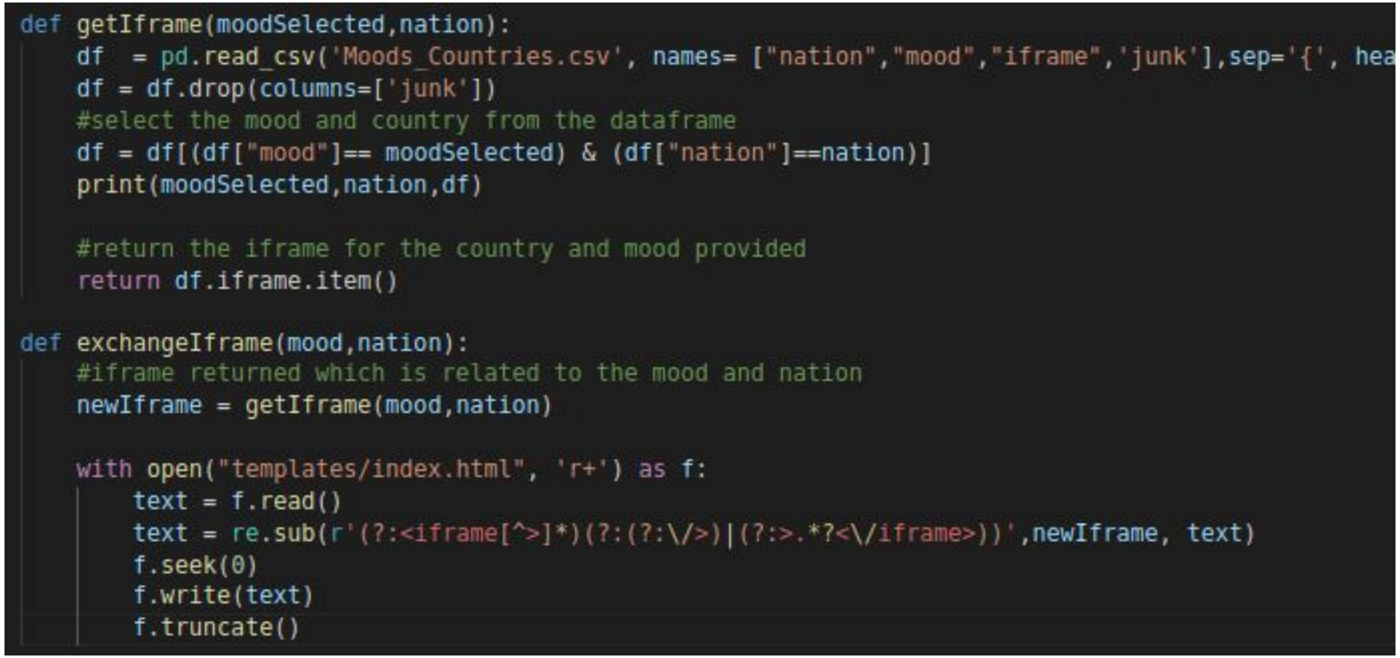
To Do
- API research everyone
- Research hosting websites (Netlify)
- Investigate what server we can use it.
- https://www.linkedin.com/learning/html-essential-training-4/what-is-html?u=42275329
- + Add a card

Doing
- Draw template of html page (fully laid out)
- Joaquin and Alex backend
- Timur and Sam front end
- Sam To Do: Dropdown backend, further research into Spotify API
- Joaquin: Look into artist information for linking to song.
- + Add a card

Done
- Reoccurring Zoom Meeting Thursday at 4:00PM Link: https://cuboulder.zoom.us/j/94817613604
- Visual Studio was chosen for development environment
- Sam will demonstrate basic HTML.
- Schedule Mock Scrum For Monday Oct 7, 2021
- Create template HTML site directory
- Joaquin - Looked into the google maps API. Currently we are required to sign up for a developer account and the google Maps API with updated info is $200 a month.
- Creating a google map plugin from back end.
- + Add a card

+ Add another list

Challenge 3 - Regex or Javascript



```
def getIframe(moodSelected,nation):
    df  = pd.read_csv('Moods_Countries.csv', names= ["nation","mood","iframe",'junk'],sep='{', hea
    df = df.drop(columns=['junk'])
    #select the mood and country from the dataframe
    df = df[(df["mood"]== moodSelected) & (df["nation"]==nation)]
    print(moodSelected,nation,df)

    #return the iframe for the country and mood provided
    return df.iframe.item()

def exchangeIframe(mood,nation):
    #iframe returned which is related to the mood and nation
    newIframe = getIframe(mood,nation)

    with open("templates/index.html", 'r+') as f:
        text = f.read()
        text = re.sub(r'(?:<iframe[^>]*)(?:(?:\/>)|(?:>.*?<\/iframe>))',newIframe, text)
        f.seek(0)
        f.write(text)
        f.truncate()
```

Questions