Cold Embers: Emily Ortiz, Gabriel Thompson, Sadi Nirloy, Thomas Zhang
Software Development
P01 Design Doc: ArRESTed Development
2022-12-05
time spent: 1.5 hours

Target Ship Date: 2022-12-21

GENERAL IDEA

- Input: Airport/Airport Code, Layover Start Time, Layover End Time
- Output: List of Businesses, Hotels, and Restaurants with Maps
  - Put coordinates of the airport into yelp, other apis?
  - Images?
  - Links to Yelp and google maps?

PARTS

Front End Framework: Foundation
The gridding of Foundation is nicer and less prone to hidden automatic shenanigans. Foundation also is optimized for mobile usage, which is where this kind of app would most likely be used.

APIs in Use:

- Airport API:
  - https://rapidapi.com/blog/airport-info-api-with-python-php-ruby-and-javascript-examples/
  - To get the coordinates of an airport from a name or a code
- Yelp
  - https://docs.developer.yelp.com/docs/fusion-intro
  - Using the coordinates of the airport, we find nearby businesses and restaurants.
- Hotels
  - https://rapidapi.com/davidtaoweiji/api/priceline-com
  - Using the coordinates of the airport, we find nearby hotels available for overnight/longer overlays
- Embedded Google Map
  - https://developers.google.com/maps/documentation/embed/get-started
  - Adding bits of google maps to show the location of the businesses on Google Maps

Database Storage:

- Users
  - Column 0: Username
  - Column 1: Password
- Business Feedback
  - Column 0: Name of Business
  - Column 1: Location
  - Column 2: Net_Rating
- Api conservation: Stores responses of the apis to a single airport, layover range of 12-ish hours (subject to change). When a request is repeated, we can pull the information from this table, and read through it to find the ones close enough and with appropriate hours.
  - Column 0: Airport Code
  - Column 1: Airport API response
  - Column 2: Yelp API answer
  - Column 3: Hotels API

    - Column 4: Embedded Google Maps

Python:

- One Flask App that acts as the middleman between the front end framework and the information from APIs and the database
- A program that holds a bunch of functions that interacts with the database directly
- A program that uses all of the APIs so that the users don't rapidly use up all of the API requests

PLANNING

Tasks:

- □ Make API cards
- □ Get and Process Feedback
- □ Database Logins and Registration
- □ Accessing the 4 APIs via Python
- □ Take in user requests and getting info from API
- □ Storing that API Info into API Conservation table
    - □ Checking if input is already in API table, and using stored info
- □ Embedding the Google Maps to show businesses and hotels
- □ Front End Stuff to beautify the site

Task Assignments:

Sadi: Database
Gabriel: Foundation
Emily: API
Thomas: flask

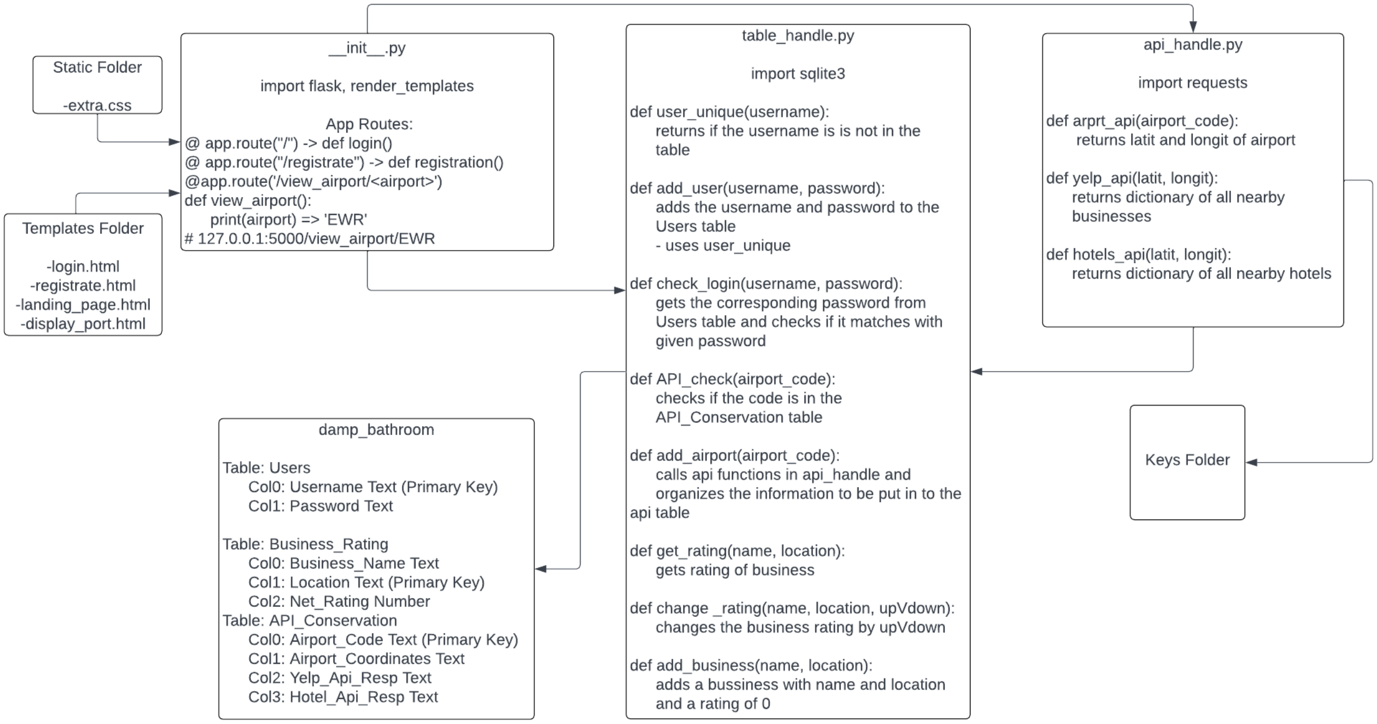
Backup:

- Remove plane layover aspect of project
    - Just input a city you want to explore

Stretch Goals:

- Put an embedded google map into the website in which users can navigate to their airport and click on a pinpoint to select their airport
    - Would work for any location selected, not just an airport
- On results page, list top 3 places for each category
    - "See more" button for each category
    - put hotels, businesses, and restaurants on separate pages
- Have page not refresh when an upvote/downvote is submitted (like YouTube)

COMPONENT MAP



```
Static Folder

-extra.css
```

```
Templates Folder

-login.html
-registrate.html
-landing_page.html
-display_port.html
```

```
__init__.py

import flask, render_templates

App Routes:
@ app.route("/") -> def login()
@ app.route("/registrate") -> def registration()
@app.route('/view_airport/<airport>')
def view_airport():
      print(airport) => 'EWR'
# 127.0.0.1:5000/view_airport/EWR
```

```
table_handle.py

import sqlite3

def user_unique(username):
      returns if the username is is not in the
      table

def add_user(username, password):
      adds the username and password to the
      Users table
      - uses user_unique

def check_login(username, password):
      gets the corresponding password from
      Users table and checks if it matches with
      given password

def API_check(airport_code):
      checks if the code is in the
      API_Conservation table

def add_airport(airport_code):
      calls api functions in api_handle and
      organizes the information to be put in to the
api table

def get_rating(name, location):
      gets rating of business

def change _rating(name, location, upVdown):
      changes the business rating by upVdown

def add_business(name, location):
      adds a bussiness with name and location
      and a rating of 0
```

```
api_handle.py

import requests

def arprt_api(airport_code):
       returns latit and longit of airport

def yelp_api(latit, longit):
      returns dictionary of all nearby
      businesses

def hotels_api(latit, longit):
      returns dictionary of all nearby hotels
```

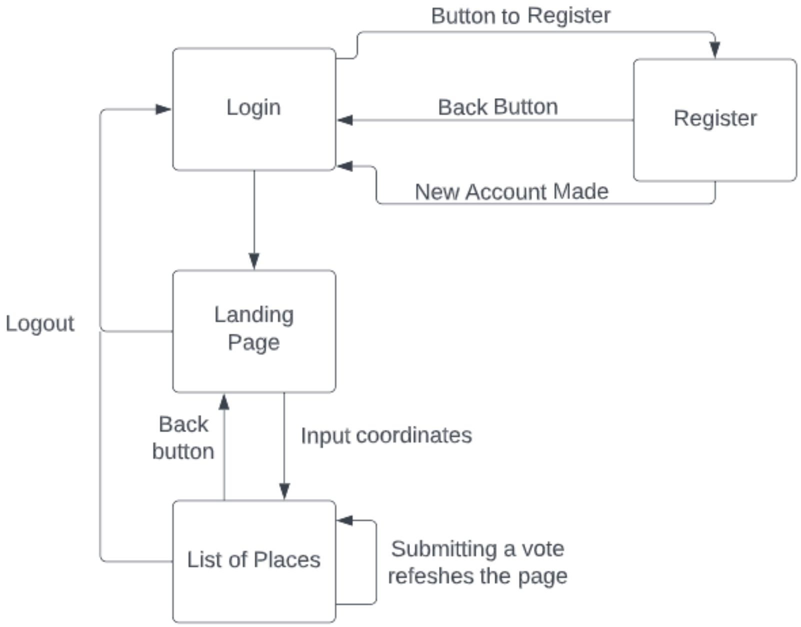
```
Keys Folder
```

```
damp_bathroom

Table: Users
      Col0: Username Text (Primary Key)
      Col1: Password Text

Table: Business_Rating
      Col0: Business_Name Text
      Col1: Location Text (Primary Key)
      Col2: Net_Rating Number
Table: API_Conservation
      Col0: Airport_Code Text (Primary Key)
      Col1: Airport_Coordinates Text
      Col2: Yelp_Api_Resp Text
      Col3: Hotel_Api_Resp Text
```

SITE MAP



Button to Register
Login
Back Button
Register
New Account Made
Logout
Landing Page
Back button
Input coordinates
List of Places
Submitting a vote refeshes the page