Ivan Yeung, Vivian Graeber, Jeff Chen, Brian Chen (Team soup noodles)
Soft Dev
P01
Target ship date: 2022-12-23

To touch grass or to not touch grass?

Program Description

A site to determine if you should go outside today based on user preferences.

Program Components

A. Python Files
   a. database.py
      i. get_uid(username): Retrieves user id from username
      ii. get_password(username): Retrieves password from username
      iii. check_username(username): Returns whether or not user already exists
      iv. add_user(username, password): Add user credentials to table
      v. add_pref(uid, nba, anime, weather): Add how much the each cares about nba, anime, and weather
      vi. check_pref(uid): Checks whether the user has set their preferences or not
      vii. update_pref(uid, nba, anime, weather): Updates user's preferences of topics
      viii. get_weather_pref(user_id): Retrieves the if user cares about weather or not
      ix. get_nba_pref(user_id): Returns how much the user likes nba
      x. get_anime_pref(user_id): Returns how much the user likes anime
      xi. add_user_info(uid, city, favorite_anime, favorite_weather): Adds user's city, favorite anime, and favorite weather to user_info table
      xii. check_user_info(uid): Returns whether or not user has set their user_info table
      xiii. update_user_info(uid, city, favorite_anime, favorite_weather): Updates user's user_info table in database
      xiv. update_city(uid, city): updates just city column in user_info table
      xv. update_favorite_anime(uid, favorite_anime): updates just favorite_anime column in user_info table
      xvi. update_favorite_weather(uid, favorite_weather): updates just the favorite_weather column in user_info table
      xvii. get_anime(uid): Returns user's favorite anime
      xviii. get_favorite_weather(uid): Returns user's favorite weather
      xix. get_city(uid): return user's city
      xx. add_weather_info(city, temperature, humidity, rain_chance, aqi, sunrise, sunset): creates table for weather info
      xxi. get_temperature(city): gets temperature of city

xxii. get_humidity(city): retrieves humidity of city
xxiii. get_rain_chance(city): retrieves the chance of raining in city
xxiv. get_aqi(city): retrieves the air quality index of city
xxv. get_sunrise(city): retrieves the sunrise time of city
xxvi. get_sunset(city): retrieves the sunset time of city
xxvii. add_anime_algo(uid, statement): adds a statement based on the results of the algorithm about user's favorite anime
xxviii. get_anime_algo_statement(uid): gets the statement from anime_algo table
xxix. add_nba_algo(statement): adds a statement based on the availability of a nba basketball game
xxx. get_nba_algo_statement(): gets the statement from nba_algo table

b. api_info.py
    i. get_weather(user_location): Returns dictionary of weather, humidity, and rain chance, updates table with weather details
    ii. get_anime_date(id): Returns anime broadcast date and time, as well as info about the anime, updates table with anime details
    iii. search_anime(search): Returns dictionary of search results for anime with IDs and images as values
    iv. get_NBA(): Returns list of all games scheduled for this month
    v. replace_space(string): replaces whitespace with %20

c. algorithm.py
    i. calc_weather(city): Returns weight of weather on chance of going outside (0-1), chance gets higher as temp approaches 75F, humidity approaches 60%. rain chance approaches 0
    ii. NBA_today(data): Returns weight of NBA on chance of going outside (0-1), chance gets lower once you are within 30 min of an NBA game, and lowers to 0 once the game starts, and for 1 hour after the game starts.
    iii. weekday_to_integer(day): converts day of week to integer (Monday = 0 … Sunday = 6)
    iv. calc_anime_date(anime_date): Returns weight of anime on chance of going outside (0-1), chance decreases as you enter 30 min before anime airing time, down to 0% as anime airs and for 1 hour after it airs, or returns 1 if the anime is either finished airing or the anime is not airing within 30 minutes.
    v. algorithm(uid): Returns % chance of going outside based on user prefs and the calculations returned from above functions
    vi. grass(weight): Returns if should go outside using weight

d. app.py
    i. Flask
        1. @app.route("/"):

a. redirect to /login
2. @app.route("/login"):
    a. renders login.html
    b. login form: username & password
        i. check for existence of username and validity of password
3. @app.route("/register"):
    a. renders register.html
    b. register form: username & password
        i. check for availability of username
        ii. if account is successfully created, information is stored in database
4. @app.route("/home"):
    a. directs to a page that allows the user to go to the page where they can access other pages
5. @app.route("/preferences"):
    a. directs to a page that allows user to customize their preferences and interests
    b. is where users are redirected to after the very first login
6. @app.route("/grass"):
    a. runs the algorithm that determines if the user should go out on the particular day
        i. It is affected by how much the user is interested in the activity (1-10), changing the weight of the factor
        ii. Weights are used to decide if one should go out or not
    b. Returns page with results + activities
7. @app.route("/weather_details"):
    a. Serves weather details
8. @app.route("/nba_details"):
    a. Serves upcoming NBA games
9. @app.route("/anime_details"):
    a. Serves details about user's favorite anime

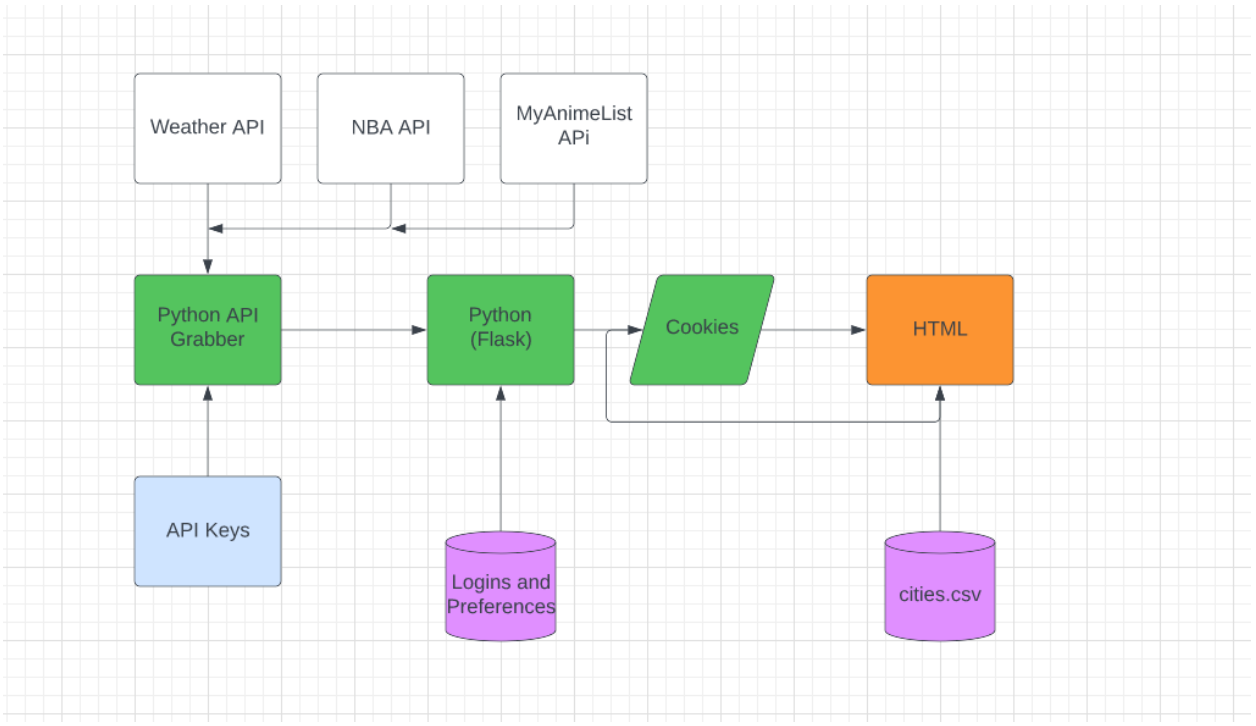
ii. Sessions
1. Session["username"]: Stores the username of the user that is logged in
2. Session["logged_in"]: Stores the logged in status of user

B. Html Files (Bootstrap)
    a. layout.html
        i. Stores the navbar that will be visible for each page

        ii. Sets up bootstrap for each page as well
    b. login.html
        i. form for username and password
        ii. Info about our site
    c. register.html
        i. form for username and password
    d. preferences.html
        i. Check boxes for different topics that user can show interest in
        ii. Sliders to show amount of interest for each supported topic
        iii. Form to enter city/region that user lives in
    e. grass.html
        i. Results of the the algorithm
        ii. Factors used to determine the result of the algorithm also available here
    f. anime.html
        i. Details for anime
    g. nba.html
        i. Upcoming NBA games
    h. weather.html
        i. Weather details

C. Misc.
    a. key_weather.txt
    b. key_MAL.txt

Component Interactions/Component Map



Database Organization

Logins

Username	UserID	Password

Preferences

UserID	NBA	Weather	Anime
	0-10	0-10	0-10

User Info

UserID	Location	Favorite Anime

Weather Info(Daily weather)

City	Temperature	Humidity	Rain chance

Anime_algo

User id	statement

Nba_algo

User id	statement

APIs

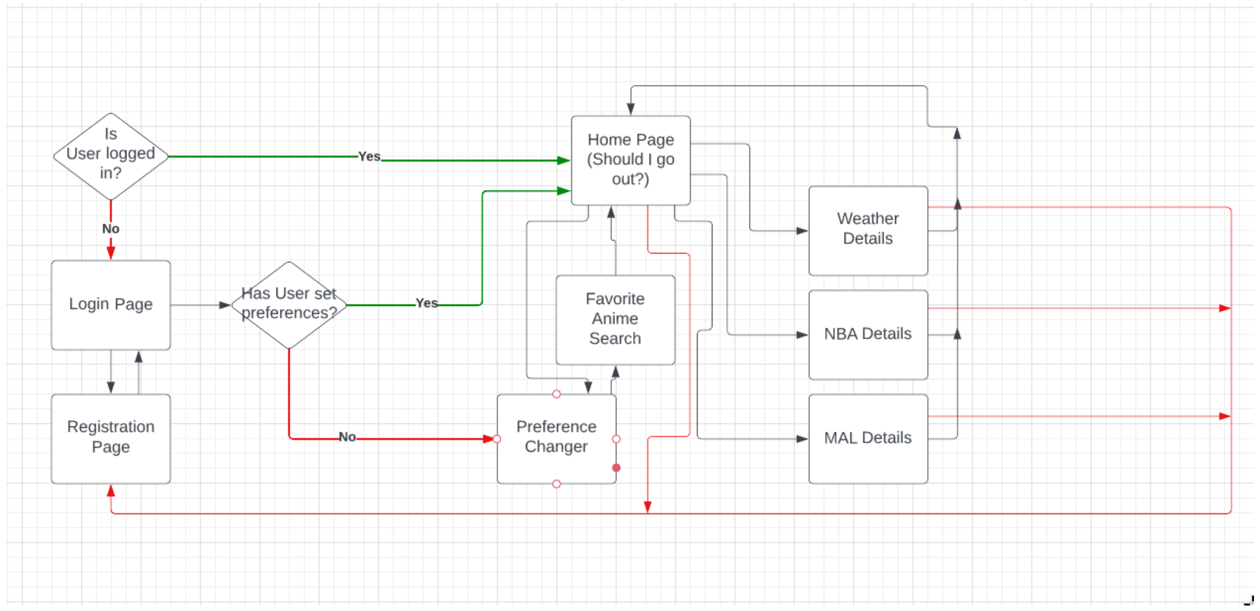
- Weather API: Used to see if the weather is good enough to go out
- Myanimelist api: Used to check if a new episode of anime is airing
- NBA schedule API: Used to see if a basketball game is going on

Bootstrap

We are using bootstrap because the style appeared more modern and clean.

- Navbar at the top of each page with links
    - Dropdowns for individual preferences on navbar
- Bootstrap forms to provide information
- General styling and information placement
- Checkboxes

Site Map



Task Breakdown (Strikethrough as we complete)

- ~~Create design doc~~
- ~~Revise design doc~~
- ~~Write Python to pull API data (*Jeff*)~~
    - ~~Confirm all APIs work~~
        - ~~Test by having all data from API put on a throwaway HTML file~~
    - ~~Functions to retrieve information from APIs~~
    - ~~Some sort of algorithm to determine whether user should touch grass or not~~
    - ~~Individual API pages~~
- ~~Write Python to serve the HTML (*Vivian*)~~
    - ~~Cookies to store user login status~~
    - ~~Login + registration~~
- ~~Create database (*Ivan*)~~
    - ~~Login storage~~
    - ~~Preferences storage~~
    - ~~Functions to retrieve data from database~~

- ~~Create HTML (*Brian*)~~
  - ~~Login Page~~
  - ~~Registration Page~~
  - ~~Preference Changer~~
  - ~~Pages that show relevant information about certain topics(based on the APIs we are using)~~
  - ~~Should I go out? page~~
    - ~~Have API update (constantly or set interval)~~
  - ~~CSS! (Bootstrap)~~
  - ~~Create API cards for APIs not already in database~~
- ~~TEST throughout the process!!!~~