Project Documentation

1. Project Code

   https://github.com/10258392511/SI507_Final_Project

   Please see README of the project to know more about requisite packages, how to supply API keys and a high-level introduction.

2. Data Sources

- https://www.planetware.com/michigan-tourism-vacations-usmi.htm

  This is a website so no official documentation is available. This data source is in HTML, so web crawling from main page to detail pages with BeautifulSoup is used. Caching is used, see a screenshot of cache file on the right. There are about 200 tourist sites recommended on the website; but due to duplicate and lack of information (e.g. an area too broad) I have retrieved 145 records. Each record is a block of presentation of one tourist site, where "place name" (used for JOIN of tables), "photo URL", "description", "other information URLs", "address" (used for retrieving geo info) are fields of interest.

Fig 2.1: cache_scraper.json

- https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets, https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/guides/standard-operators

  This is Twitter API. As an extension of HW 6 I also searched for user accounts and then search tweets by user. The data is in JSON and I accessed by GET method with caching. Not all toursite sites have active tweets within a week (per Twitter API), for some there are as many as 50, but some only 10. On average, there are around 1000 records in total for all tourist sites. I have retrieved around 200 records for 5 places. As this should be dynamic, saving to database is not necessary here. Each record is a tweet about the place, where "created time" (used for sorting) and "full_text" (full tweet) are fields of interest.

Fig 2.2: cache_twitter.json

- MapQuest and MapBox:

  https://developer.mapquest.com/documentation/geocoding-api/address/get/, https://docs.mapbox.com/mapbox-gl-js/api/

  The former is used in Project 2 and latter using longitude and latitude info from the former renders an embedded interactive map in HTML. Data is in JSON retrieved with API calls with caching. For each tourist site there is at least one record, so total number is over 145. But it turns out that some places are not available in the API, i.e. no record located in Michigan, so after data cleaning I've got around 140 records. Each record is geo info for the place, where fields of interest are "adminArea", "adminAreaType", "lat" and "lon". The former two verify whether the record is resonable, and the latter two are used for interactive map rendering.

Fig 2.3: cache_map.json

- https://rapidapi.com/community/api/open-weather-map?endpoint=apiendpoint_f719676c-072b-4a2d-ad2e-78f8375ea9c8

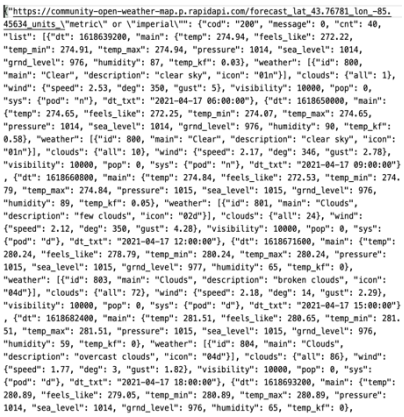

Figure 2.4: cache_weather.json

This API retrieves weather forecasting data for 5 next days every 3 hours (i.e. 40 data points). It's in JSON accessed be API calls with caching. For each place there are 40 records so in total there are 5800 records. I have retrieved data for 5 places thus 200 records. As this data should be dynamic, saving to database is unneccessary. Fields of interest are "temp" (for temperature), "wind speed", "description" (e.g. "clear sky"). The description will be written on each data point as text, and the former two will be displayed by Plotly line plots.

```
CREATE TABLE IF NOT EXISTS TouristSites (
    Id INTEGER NOT NULL,
    Name TEXT UNIQUE NOT NULL,
    PhotoURL TEXT, Desc TEXT NOT NULL,
    Address TEXT, InfoURL TEXT,
    PRIMARY KEY (Id AUTOINCREMENT))
```

```
CREATE TABLE IF NOT EXISTS Maps (
    Id INTEGER NOT NULL, Name TEXT UNIQUE NOT NULL,
    AdminArea6 TEXT,AdminArea6Type TEXT,
    AdminArea5 TEXT, AdminArea5Type TEXT,
    AdminArea4 TEXT, AdminArea4Type TEXT,
    AdminArea3 TEXT, AdminArea3Type TEXT,
    AdminArea1 TEXT, AdminArea1Type TEXT,
    Lat REAL, Lng REAL,
```

3. Database

- Schema is shown above.
- "Name" of "TouristSites" and "Name" of "Maps" form a foreign-key pair.
- Screenshots of record are shown below.

Table: Maps

	Id	Name	AdminArea6	AdminArea6Type	AdminArea5	AdminArea5Type
1	1	Detroit Institute of Arts		Neighborhood	Detroit	City
2	2	Motown Museum		Neighborhood	Detroit	City
3	3	The Ford Piquette Avenue Plant		Neighborhood	Detroit	City
4	4	Detroit Historical Museum		Neighborhood	Detroit	City
5	5	Charles H. Wright Museum of African-American History		Neighborhood	Detroit	City
6	6	Detroit Zoo		Neighborhood	Royal Oak	City
7	7	Historic Fort Wayne		Neighborhood	Detroit	City
8	8	Belle Isle		Neighborhood	Black Lake	City
9	9	The Henry Ford Museum & Greenfield Village		Neighborhood	Dearborn	City
10	10	Comerica Park		Neighborhood	Detroit	City

Table: TouristSites

	Id	Name	PhotoURL
1	1	Detroit Institute of Arts	https://www.planetware.com/photos-large/USMI/michigan...
2	2	Motown Museum	https://www.planetware.com/photos-large/USMI/michigan...
3	3	The Ford Piquette Avenue Plant	https://www.planetware.com/photos-large/USMI/michigan...
4	4	Detroit Historical Museum	https://www.planetware.com/photos-large/USMI/michigan...
5	5	Charles H. Wright Museum of African-American History	https://www.planetware.com/photos-large/USMI/michigan...
6	6	Detroit Zoo	https://www.planetware.com/wpimages/2021/03/michigan...
7	7	Historic Fort Wayne	https://www.planetware.com/photos-large/USMI/michigan...
8	8	Belle Isle	https://www.planetware.com/wpimages/2021/03/michigan...
9	9	The Henry Ford Museum & Greenfield Village	https://www.planetware.com/photos-large/USMI/michigan...
10	10	Comerica Park	https://www.planetware.com/wpimages/2021/03/michigan...

4. Interaction and Presentation Plans

- **Data Display Options and Usage**

  To start, a user has to create a "secrets.py" file and fill his / her API keys in. Please see README for detailed instruction. Then he / she can just run "run_app.py" file to establish database and launch the server.

  On index page a user can view a comprehensive list of Michigan tourist sites with a thumbnail. If the user wants to sort the results by distance to a certain place in MI, he / she can type the place in the text input just below the header, and hit "enter". To return to the default index page, he / she can just hit "enter" in the text input or just refresh the page.

The rest three options are for one tourist site. A user can click on any place listed on index page to navigate to a “place index” page where there are 3 links: description, map and weather forecast. The description page includes a photo and short description of the place, and if available, any tweets by possible Twitter accounts owned by the place and tweets by keyword searching of the place name. The map page displays an interactive map where a user can drag around, zoom in to see a street view & etc. Finally, the fourth view is the weather forecast page where 2 line plots are displayed for temperature and wind speed of next 5 days every 3 hours. These plots are interactive, so when a user hovers over each data point a short summary will pop up like “clear sky”. Note all pages are responsive, so a user can use the web app on monitors of any size including mobile phones.

- **Interactive and Presentation Technologies**
  This is a Flask web app. Interative parts uses Flask with Jinja2 templates; and responsive web design leverages Bootstrap5; and all interative plots are rendered by plotly.

5. Demo link:
   https://drive.google.com/file/d/1_gFmpL6cPxBnBkDgCY_Io-w9w8dCEtF9/view?usp=sharing