TNPG: aTriceratopy, Roster: Brian Yang, Jonathan Song, Prattay Dey, Verit Li
SoftDev pd7
P4 -- data visualization project design doc
2023-05-02
Time Spent: 3 hrs
Target Ship Date: 2023-05-23

Aeroplano Graveyardo

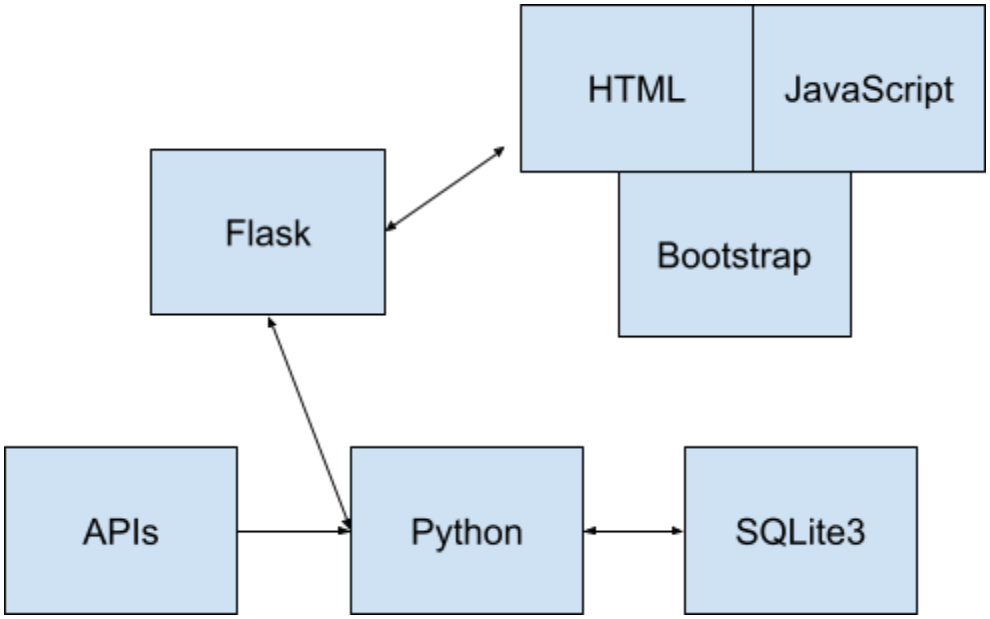
Abstract:
Aeroplano Graveyardo is a global map showing all civil aviation accidents from 1908 to 2019. Users will be able to see points on a globe of locations of crashes, select a crash and see the plane's intended path, interact with filters like aircraft type, country of crash, deadliness, and accident year. Users will also be able to see trends in the database file.

Program Components:

- Javascript
  - Allows for more user interactivity
  - Trends of airplane crashes using Google Charts library
    - https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_GoogleCharts.md
  - MaxboxGL for an interactive map
    - https://github.com/stuy-softdev/notes-and-code/blob/main/how-to/howto_MapboxGL.md
- HTML
  - Jinja syntax to collaborate with Flask
  - Templates to be rendered by Flask
- Python
  - Makes API calls
  - Uses database functions
- Flask
  - Our web server and delivery framework
  - Renders html templates to browser
- SQLite3
  - Storage of airplane crash data based off of:
    - https://www.kaggle.com/datasets/cgurkan/airplane-crash-data-since-1908
- **FEF** (Bootstrap)
  - Used to style website
  - Chosen because of more experience using within the team
  - Provides grid layout, navbar, dropdowns, range forms (a slider)

- APIs
    - Bing Maps API
        - To return coordinates when given the name of a location which can be then marked on mapbox map
        - https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_BingMaps.md
    - Mapbox API
        - Needed to get an access token for Mapbox GL
        - https://github.com/stuy-softdev/notes-and-code/blob/main/api_kb/411_on_Mapbox.md

Component Map:



Front End Site Map:

map.html

- Takes filtered data from the dataset and displays crashes it on an interactive map
- Clicking on a specific crash location will allow you to view more detail on the incident (exact time/date and summary of the incident)
- Slider to determine year interval of crashes shown

summary.html

- User is directed to this page when they click for more information on a specific flight

- A page that shows all the information provided by the dataset for a row (Date, Time, Airline/Operator, Flight number, Route, Aircraft Type, ICAO registration, total aboard, fatalities, summary)

trends.html

- Trend graphs for:
    - Crashes by Decade
    - Fatalities by Decade
    - Top 50 Most Crashed Planes

Middleware: Python

__init__.py

- The foundation of our flask app. Runs and displays web pages.

db.py

- Contains function to create and maintain a database. Including the storage of coordinates
- Contains function to retrieve data from database

api.py

- Intermediary module to allow getting information from api convenient and simple

reset.py

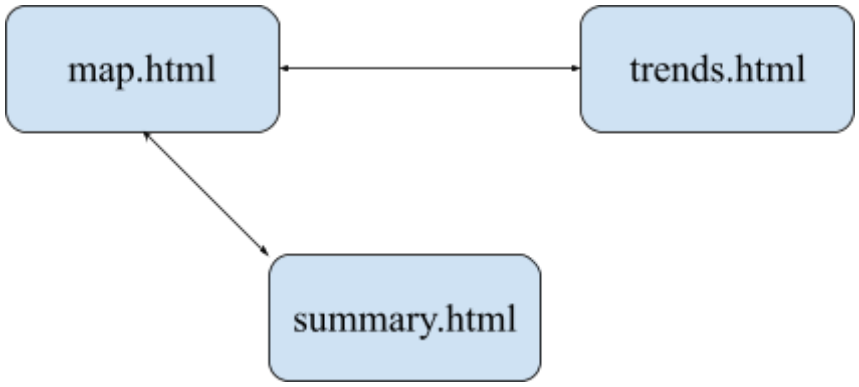
- Resets and populates the database, needs to be run before you run the app for the first time

Database Organization: SQLite3

- Airplane *crashes* table consisting of:
    - Flight id
        - generated starting at 0 and adding 1 for every next flight
    - Date
    - Time
    - Location
        - might need a filter to get rid of words that aren't the actual location like "near" in "near Los Angeles"
    - Operator (Airline or operator of the aircraft)
    - takeoff
    - destination
    - Aircraft type(model)
    - crew #
    - passengers #

    - fatalities #
    - ground (fatalities caused on the ground)
- Crash *coordinates* table consisting of:
    - Flight id
    - Latitude
    - Longitude
        - (Both longitude and latitude generated from Bing Maps API)

Site Map:



Task List:

- Database (Verit)
    - functions for populating the database
    - functions for accessing data
- Map (Brian)
    - map.html template
    - javascript for interactive map
- Summary (Jonathan)
    - summary.html template
    - flask such that every crash has a route to a summary page
- Trends (Prattay)
    - trends.html template
    - functions parse data to be fed to trend graphs

Useful Links:

https://www.kaggle.com/datasets/warcoder/civil-aviation-accidents
https://www.kaggle.com/datasets/cgurkan/airplane-crash-data-since-1908

https://www.ntsb.gov/safety/data/Pages/GeneralAviationDashboard.aspx#AVSpreadsheet

Task List

- [ ] Database
    - [ ] Function to set up tables
        - [ ] Crashes
            - [ ] Function to extract data from csv file at selected location
        - [ ] Coordinates
        - [ ] Rankings
    - [ ] Functions to access data in each of the tables
        - [ ] Crashes
        - [ ] Coordinates
        - [ ] Rankings
    - [ ] Functions to store data in each of the tables
        - [ ] Crashes
        - [ ] Coordinates
        - [ ] Rankings
- [ ] Flask
    - [ ] Flask route for root (map)
        - [ ] Link route to button on navbar
    - [ ] Flask route for trends
        - [ ] Link route to a button on navbar

- [ ] Flask route such that every crash has its own summary page
    - [ ] Link such that when a crash location is inspected and a crash selected to view summary, bring to summary page
- [ ] HTML/Bootstrap
    - [ ] Templates
        - [ ] map.html
        - [ ] summary.html
        - [ ] trends.html
    - [ ] navbar

- [ ] Javascript
    - [ ] Research D3.js
        - [ ] https://piazza.com/class/l7yp6eyrtj978b/post/651
        - [ ] https://d3-graph-gallery.com/
    - [ ] Graphs for showing data
    - [ ] Map
        - [ ] Clickable dots of all crashes
        - [ ] Zoom in on crash when dot clicked
        - [ ] Show pop-up with list of crashes when dot clicked
        - [ ] If a crash is selected and “show path” is toggled on, show the path of flight if possible
    - [ ] Dropdowns for filtering by certain categories and a submit button
        - [ ] Date
        - [ ] Number of casualties
        - [ ] Operator
        - [ ] Location
- [ ] APIs
    - [ ] Find best location api (pointstack might not be best)
    - [ ] Get API key
    - [ ] Function for getting coordinates of an inputted string location

- [ ] Droplet
    - [ ] Serve app on somebody's droplet

```
def get_all(category):
    db = sqlite3.connect(DB_FILE) #open if file exists, if not it will
create a new db
    c = db.cursor() #creates db cursor to execute and fetch

    c.execute(f"SELECT {category} FROM crashes")
    everything = c.fetchall()
    print(everything)

    return everything

reset_database()
populate_crashes()
get_all("date")
```