DOES**NOT**COMMUTE

BY **DNC**

GITHUB: wvenderbush, zzcnick, wostlund, brianlu

Delegation

Winston: Dark Sky API, Back-Front Liason

Zicheng: (PM): Google APIs, Flask App

Brian: Google APIs, Javascript help, MTA API

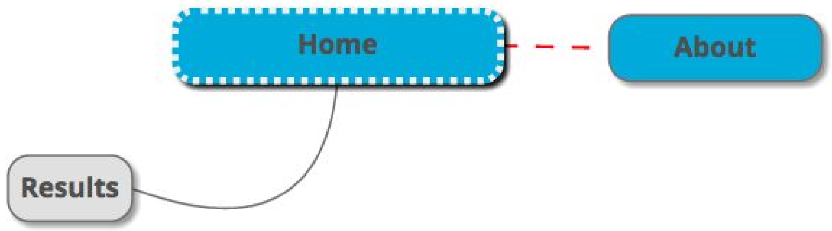
Will: Front End (Bootstrap)

Abstract: Using a location API, finds user location. User provides location to which they want to go. User gives them a time they want to arrive at that location. Then our website generates a way to get there and provides an estimated ETA, providing minute-to-minute weather, reasons for the ETA based on train and bus delays, possible traffic, and any other helpful information that the user may need.

Site Map:

- Homepage (Single Page)
    - Contains the DNC utility; elicits user input with various fields, such as departure time, preferred arrival time, destination, start location, and preferred mode of transportation.
    - Upon submission, will direct users to a generated results page with information about their route and other helpful tips.
- About Page
    - Contains a mission statement, biographies, future goals for the website, inspirations and aspirations, and the like.
- Result Page
    - If accessed via url (GET), will redirect to the homepage. If accessed from the homepage (POST), will generate another page with a map of the route, instructions on how to get there, transportation information and weather, along with other tips.

Site Map Image



APIs:

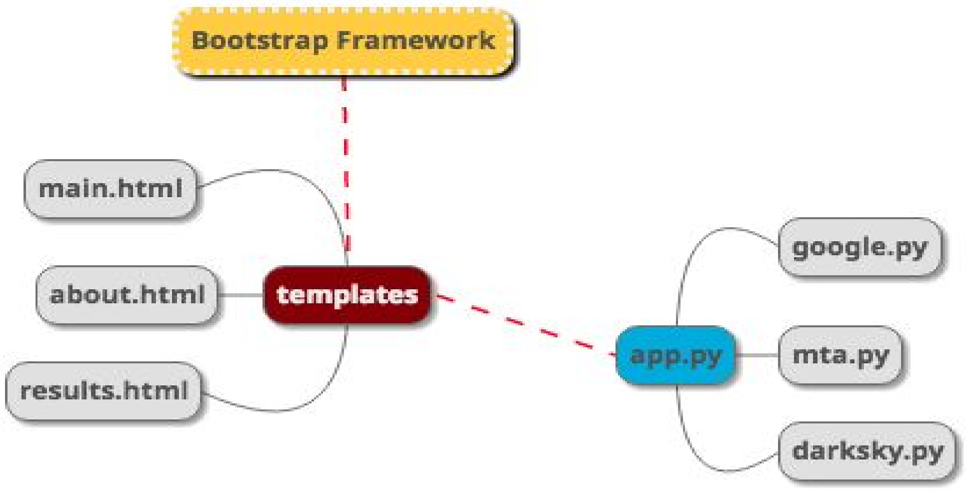
- Dark Sky
    - Used to retrieve accurate minute-by-minute weather information.
- Google Maps (Google Maps Distance Matrix API, Google Maps Geolocation API, Google Maps Directions API, Google Maps Geocoding API)
    - Use Directions API to provide the user with different routes to their destination.
    - Use Distance Matrix API to calculate the estimated time of arrival.
    - Use Geolocation and Geocoding API to determine location of user for on-the-go direction generation.
- MTA (Subway Times, Bus Times)
    - Used to provide the user with possible delays along their route and to enable accurate reasons for delays.

Component Descriptions:

- app.py
    - Contains flask app.
    - Handles redirection and user input.
    - Three main routes:
        - Home (GET, POST)
        - About (GET)
        - Results (POST)
- google.py
    - Handles Distance Matrix, Geolocation, Directions, and Geocoding APIs.

    - Geocoding APIs required to pass geo-location coordinates to DarkSky for weather data.
    - Distance Matrix handles distance and helps with time estimates.
    - Geolocation provides location information based on current location.
    - Directions helps map routes to location.
- mta.py
    - Handles pulling delay info (from subways and busses) from the MTA.
    - Returns necessary info based on path selected, trains/buses required on path, and weather.
- darksky.py
    - Gives up to date (minute-to-minute) weather info based on current location and goal location.
    - Also used to provide information about weather along the way, and weather within the given timeframe of travel.
    - Uses information from Geocoding/Geolocation API for location info.
- Templates
    - main.html -- used to generate homepage
    - about.html -- used to generate about page
    - results.html -- used to generate page based on inputs from homepage

Component Map:



Features:

- Ability to select location, or automatic location tracking based on Geolocation.
- Accurate and precise weather information based on location.
- Evaluation of ETA and analysis of whether you can make it on time.
    - Ability to input departure time and hopeful arrival time.
- Reminders based on weather, delays, and other conditions.
- Ability to provide multiple routes for a user to choose.

Application Layout:

1. Where do you want to go? (textbox input)
2. Where are you now? (textbox input, or option to use geolocation)
3. How do you want to get there? (checkboxes: car, bus/subway, bike, walk)
4. When do you want to get there? (Not necessary)
5. Submit the form, and have a results page generated for you.

Possible Future Features (Room For Future Development):

1. Ability for user to save frequent routes through cookies.
2. Implementation of a database to allow for users to create accounts and permalink routes and destinations.
3. Print output on same page as the homepage, eliminating the results page.
4. Use more APIs, such as Citibike API, to allow more user flexibility in route planning.

Dev Schedule:

- Friday - December 2
    - Set up devlog.
    - Set up skeleton files.
- Monday - December 5
    - Basic HTML templates (Will, Winston)
    - API calls and communications from utility files
        - Darksky API (Winston)
        - Google APIs (Brian, Zicheng)
        - MTA (Brian)
    - Basic routing in Flask app (Zicheng)
- Tuesday - December 6
    - Allow homepage to elicit user input to allow for API calls (Will, Winston)
    - Add interaction between Geolocation API and Darksky API (Winston)
    - Add interaction between Geolocation API and Geocoding API (Brian, Zicheng)

- Wednesday - December 7
    - Add ability to print API call data onto a results page (Will, Winston)
    - Add interaction between Directions API and MTA API (Brian)
    - Add interaction between Directions API and Geocoding API (Zicheng)
- Thursday - December 8
    - Flesh up the main page, add tabs in results page (Will, Winston)
    - Add interactions between Directions API and Direction Matrix API (Zicheng)
    - Add basic route analysis, help format information to make user-readable (Brian)
- Friday - December 9
    - Make page functional and working! (Will, Winston)
    - Brush up various API interactions, add documentation (Brian, Zicheng)
- Saturday / Sunday - December 10 / 11
    - Polish and publish! (Brian, Will, Winston, Zicheng)