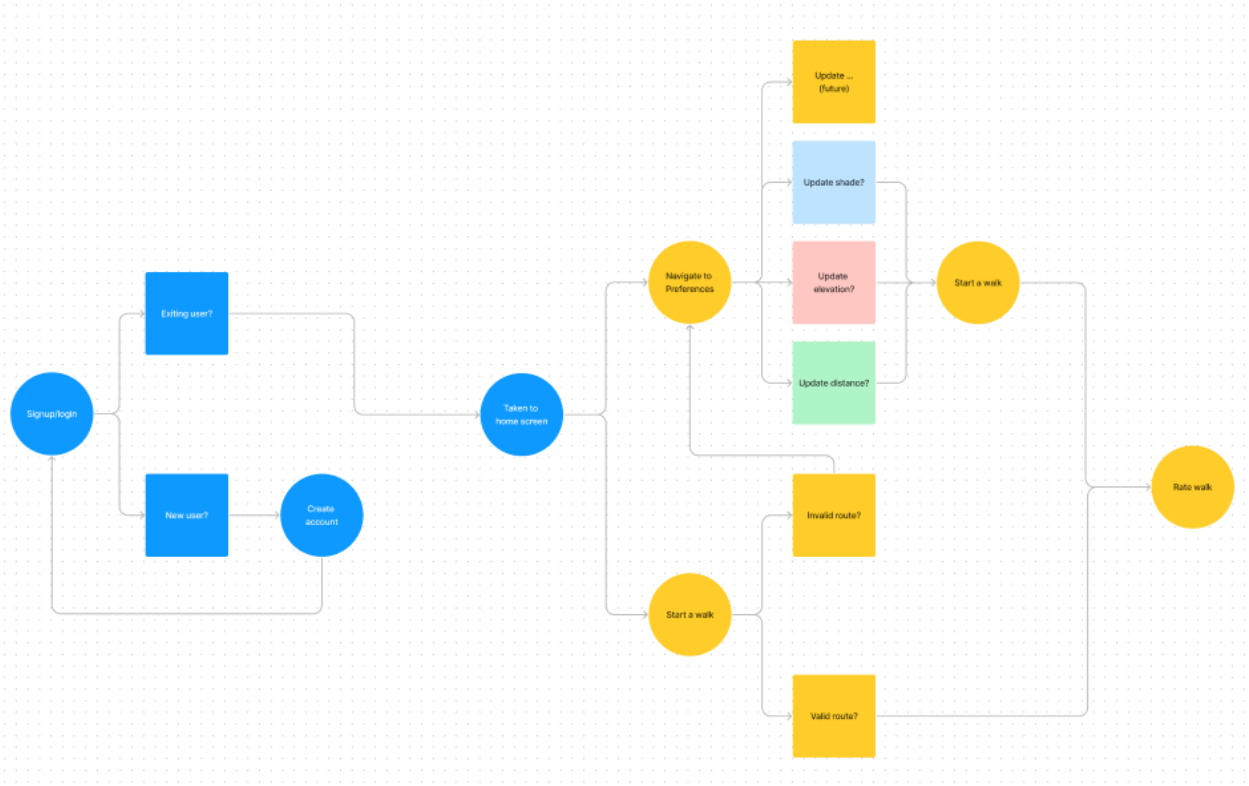
Writing 3: Project Description
Omega Battlebots: Noah, Luisa, Kyle, Sudhi

Product Specifications:

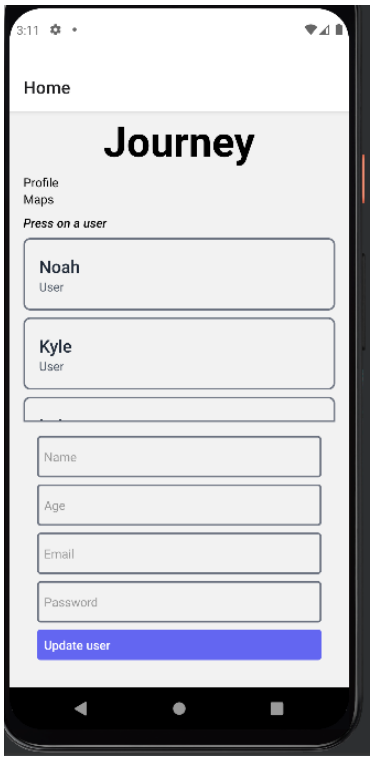
User Stories:

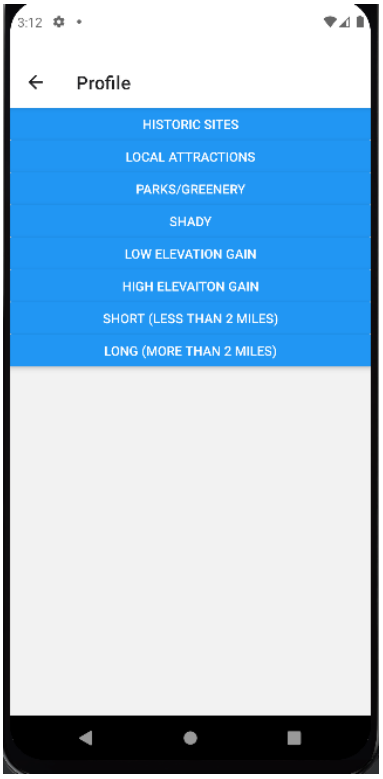
- As a tourist in Washington, DC, I would like to get updated recommendations of places to visit so that I can choose places to go to while in the city.
- As a resident in Washington, DC, I would like to walk in a shaded path through commercial areas so that I can enjoy the city how I want to
- As a tourist, I would like to get information about a location so that I can learn more about that place
- As a person in a wheelchair, I would like to go through a path with the least elevation to get to my destination
- As a dog parent, I would like to walk through green areas so that my dog can do his business
- As a kid parent, I would like to walk through parks so that my kids can enjoy the walk around the city
- As a person attending a birthday, I would like to walk through a commercial area so that I can buy a gift before arriving to my destination
- As a bored DC local, I would like to see events happening in my city so that I can enjoy an evening with my friends

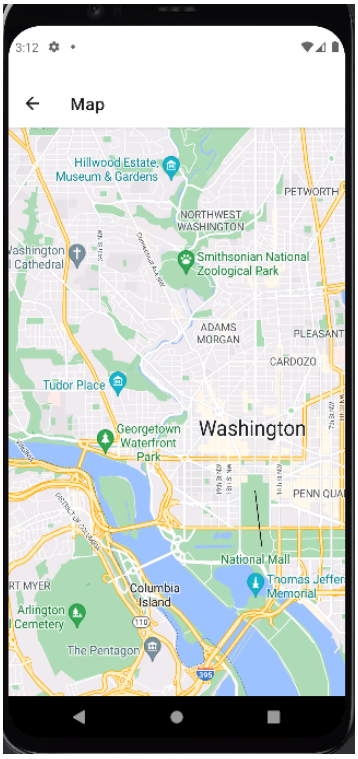
Flow Diagram:



Mockups:





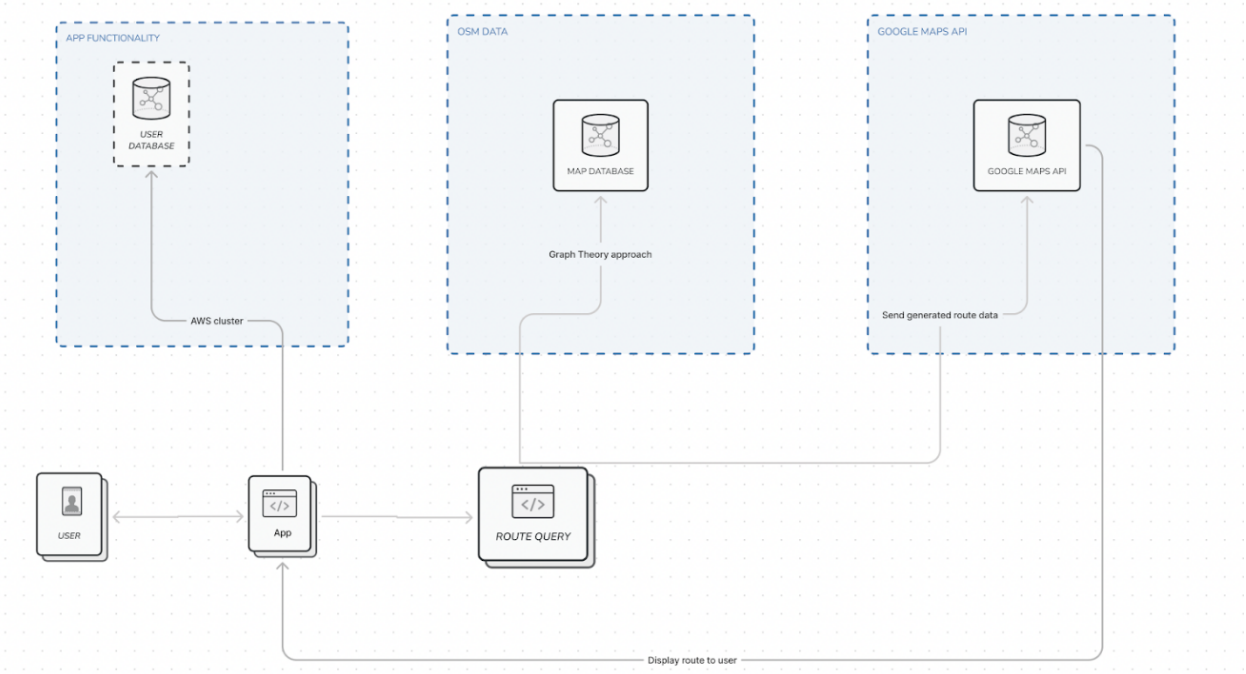


*The home screen will not showcase the database information in the final release, this is just for the alpha demo to showcase it is saving data.

Technical Specifications:

Architecture:

Overview of how the individual components interact with each other. The user interface takes the registration information of the user at sign up and adds it to the database. They then navigate to the profile screen where they can select the different options they want for a walk. These options are sent to the backend of our app where, using our routing algorithm to weigh the different features they want, a walk will be generated. This walk will then be returned to the app through the google maps api to showcase the generated route for the user.



External APIs and Frameworks:

Open Street Maps

Goal

- Retrieve data from the D.C. area and update the map

Description

- This is used in the .osm files in the code. It connects to mapping functionality and is called when a routine update to the map happens.

Google APIs:

- **Directions API**
  - Goal
    - Create a viewable route with the Google Maps UI to show to the user.
  - Description
    - We will send our routes (array of lat/lon coordinates) to this API and have it generate the corresponding route using Google Maps.
- **Places API**
  - Goal
    - Receive detailed information about certain places of interest and be able to display them to the users directly in our map.
  - Description
    - This endpoint uses information Google stores for over 200 million locations and allows us to query certain locations for helpful data (address, phone, details, photos) and use this to enrich the experience for our users.

SunCalc API:

Goal

- We will use the SunCalc API to create a model for the sun and retrieve information about the shadows at specific locations.

Description

- This API is used in the shadeCalc file in the code. It connects to updating our maps and is called routinely to update the map through different conditions set by different days through the year.

Endpoints

- GET /suncalc.org/#/{lat,lon,zoom/date/time/objectlevel/maptype}

Notable Python packages:

- **Osmium**
  - Goal
    - Read in map data, save additional map data, and create new maps
  - Description
    - This is used in the filter file in the code. It connects to the map functionality and is called when map data edits occur.
- **Osmnx**
  - Goal
    - Create routes
  - Description
    - This is used in the routing file in the code. It connects to routing functionality and is called when a user wants to generate a route. The current location and distance desired are generated by the app and passed into this package to get a route.

Algorithms:

Shade Calculation Algorithm

- Goal
    - Read in height data from the LiDAR data set of D.C. in order to generate expected shade values over certain latitude and longitude areas by use of the SunCalc API framework.
    - Success involves adding a tag to all of our nodes determining the times it will be shady (if we cannot find data for times in which it will be shady, we assume it will be sunny at that location at those times).
- Description
    - This algorithm works within the project by populating our maps database with shade values over time in D.C.

Routing Algorithm v1

- Goal
    - Use the Prize Collecting Traveling Salesman Problem (PCTSP) to generate a comparative structure to score routes and optimize the best route for an individual to take based on reward.
    - Success of running this algorithm will be the ability to score our routes and choices through routes in order to optimize the decisions as a route is generated.
- Description
    - This algorithm's part in the project is creating the routes for the users. By taking in data from the user (location and walk preference data), a route can be generated and scored to send back the optimal walk to the user.

Routing Algorithm v2

- Goal
    - Utilize a reinforcement learning model to generate a route based on choices it views as best through each stage in the route generation.
    - Success of this algorithm would involve the model generating "good" routes for a user ("good" being that an optimized score entails a walk traversing varied areas and areas of interest as opposed to looking to optimize scores in a boring fashing, for example, walking up and down a shady street)
- Description

- This algorithm’s part in the project is creating the routes for the users. By taking in data from the user (location and walk preference data), a route can be generated and scored to send back the optimal walk to the user.