ImmediAlert

Table of contents

Introduction:

While contact tracing is an essential tool for managing a pandemic, its efficacy depends on the resources of a region's health department and how well a patient remembers where they've been during an incubation period. These aspects are variable and not uniform across the country, meaning many will never know whether they've been in close proximity with an exposed person. Unfortunately, when people *are* notified, affected individuals and businesses are often ostracized, bringing about undeserved humiliation. ImmediAlert (Immediate Medical Alert) offers a solution: vigilant, anonymous contact tracing.

Each user is given a unique token stored in a secure database, and when that token spends a significant time in one place, that location and potential time of exposure is logged via GPS. ImmediAlert then takes in data regarding local outbreaks and self-reported positive tests to determine if the user has been in close contact with an exposed person during their incubation period. There is no need for excessive panic or confusion regarding when you should test — ImmediAlert is there to help you.

In terms of functionality, the user is able to view a map (or list) of their previous locations and calculated risk of exposure during their stay. ImmediAlert also determines the rough addresses of the logged locations given their GPS coordinates. If a user is feeling uncertain or overwhelmed, they can take a questionnaire to determine their need for a COVID-19 test, through which they can be directed to nearby testing locations and information from the CDC regarding the common questions asked about the virus. A user can also self-report their own positive test, which allows ImmediAlert to notify anyone who has been in close proximity with an infected user to seek testing. The app additionally directs users to mental health resources specialized around the virus.

So all in all, this was our end goal:

Backend:

Backend requirements:
Django==3.1.2
django-cockroachdb==3.1.1
psycopg2-binary==2.8.6

Language used: Python
AWS Instance: EC2 Ubuntu
Web server: Nginx and gunicorn

Database host: CockroachCloud

API: https://www.getpostman.com/collections/8a8c086865cc8c4a7b2b

```
Post http://18.222.146.208/user/
{
"uuid":"<uuid4>"
}

Post http://18.222.146.208/gps/
{
    "uuid":"<uuid4>",
    "address":"<string>",
    "city":"<string>",
    "state":"<string>",
    "zip_code":"<string>"
}

Post http://18.222.146.208/gps/
{
    "uuid":"<uuid4>",
    "score":<int>,
    "close_contact":<boolean>
}

Post http://18.222.146.208/at_risk/
{
    "address":"<string>",
    "city":"<string>",
    "state":"<string>",
    "zip_code":"<string>",
    "risk_level":<int>
}

Get http://18.222.146.208/gps/<uuid4>/

Get http://18.222.146.208/symptoms/<uuid4>/

Get http://18.222.146.208/at_risk/<address>/<zip_code>
```

Database Schema:

user:

column uuid UUID4 UNIQUE NOT NULL

gps:

column uuid UUID4 NOT NULL,

column date DATE NOT NULL,

column address VARCHAR(50),

column city VARCHAR(25) NOT NULL,

column state VARCHAR(2) NOT NULL,

column zip VARCHAR(20) NOT NULL

symptoms:

column uuid UUID4 NOT NULL,

column date DATE NOT NULL,

column score INT8 NOT NULL,

column close_contact BOOLEAN NOT NULL

at_risk:

column date DATE NOT NULL,

column address VARCHAR(50),

column city VARCHAR(25) NOT NULL,

column state VARCHAR(2) NOT NULL,

column zip VARCHAR(20) NOT NULL,

column risk_level INT(3) NOT NULL;

Currently, we only have cleartext http:// requests. Going forward, this will definitely be a priority to secure the database.

The data collected from each user from the app will be tagged with a unique ID, and sent to the server for data processing. The idea being if a user has reported that they are positive, the app will compare the user travel history with everyone else in the system. If there is a match (same time and place), a notification will be sent out to the corresponding user. The most crucial thing is that the app collects no personal data, and requires no identification except an automatically generated ID to preserve the anonymity for user confidence in reporting the truth.

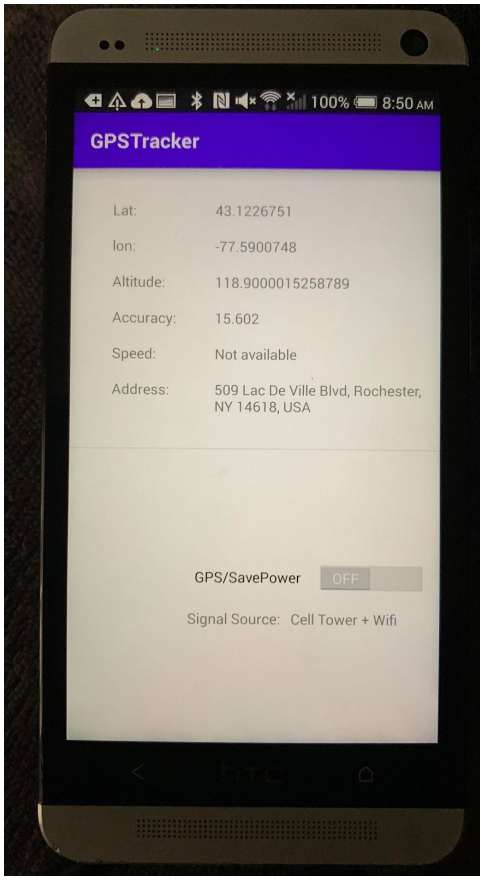
Middleware:

To facilitate the process between the backend data, and properly communicating the data to the user -- we needed our application to be able to handle this translation between the two. Using Java, which was our preferred language for this application. We acquire location data from the user who has the application installed, using this data we would be able to send it to our Django backend, and it would keep the information for each unique user. Whenever a user would click the “Test Positive?” button, it would trigger our system to alert other users who have also been to the same location as the infected user within the last 4 days. We also include a Covid Questionnaire that would allow for a self-check for symptoms of Covid-19, if the user scored an arbitrary amount of points on the question it would sway them to get tested, and if tested positive -- notify other users via the ‘Test Positive?’ button.

The questionnaire data was pulled from the CDC website. From reading on the CDC, certain symptoms are more characteristic of COVID, so they were given more points than others. The details of the questionnaire is below:

In the past 14 days, have you traveled internationally or returned to New York State from a restricted state/area within the US? If yes +1
Have you had a household or close contact with someone who is confirmed to be positive for COVID in the past 14 days? If yes notify to get tested In the past 24 hours, Have you had any of the symptoms below that are new or unusual for you? Temperature of 100 °F (37.8 °C) or higher +3
Chills +1
Muscle or body aches +1
Severe Fatigue +1
Headache +2
Congestion or runny nose (not due to allergies) +1
Sore Throat (not due to allergies) +2
Loss of taste or smell +1
Loss of appetite +1
Cough +1
Shortness of breath or difficulty breathing +3
Nausea, vomiting, or diarrhea +1
If score is more than 7 then notify person to test

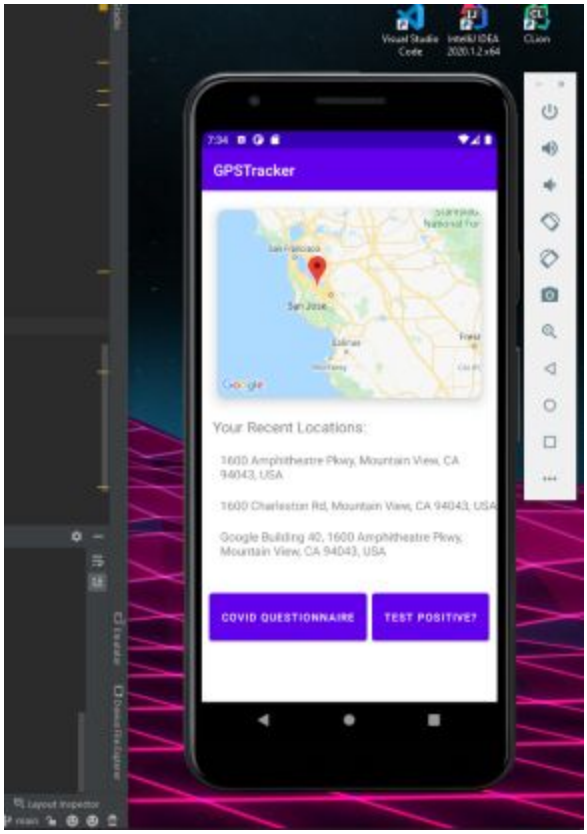
The behind the scenes working of the app is shown as below:



Showing the address from longitude and latitude, as well as address from scraping data from either wifi+cell tower or GPS signal, with a switch to change which option of getting GPS.

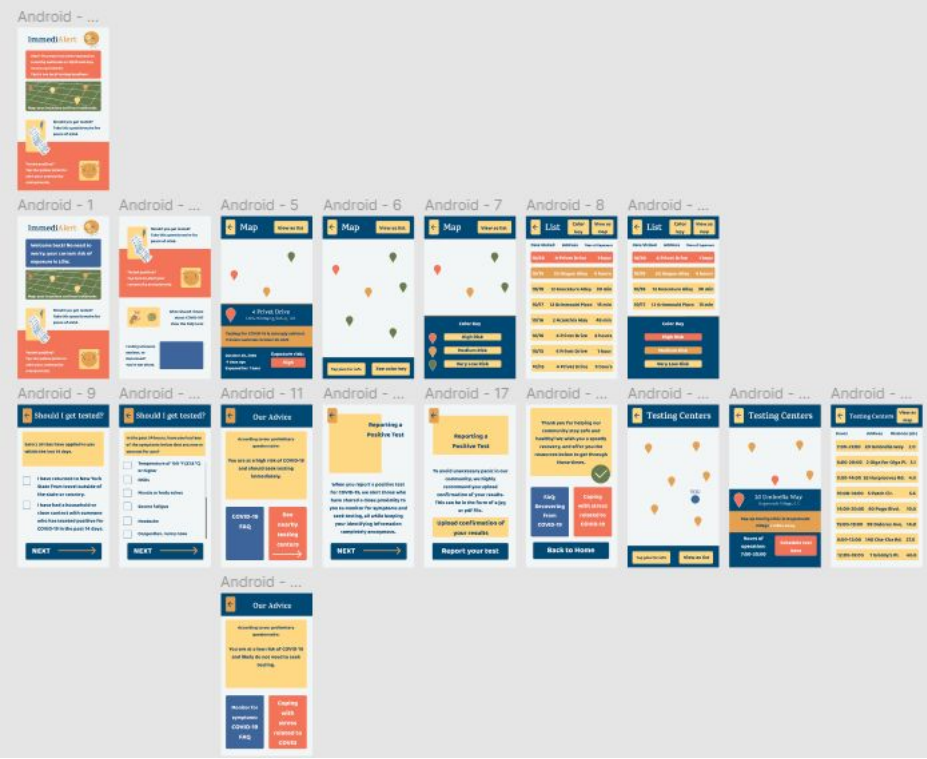
Front end:

Due to the limitations of the Hackathon, we were not able to implement all the UI elements that we had originally hoped for. We had only achieved thus far:



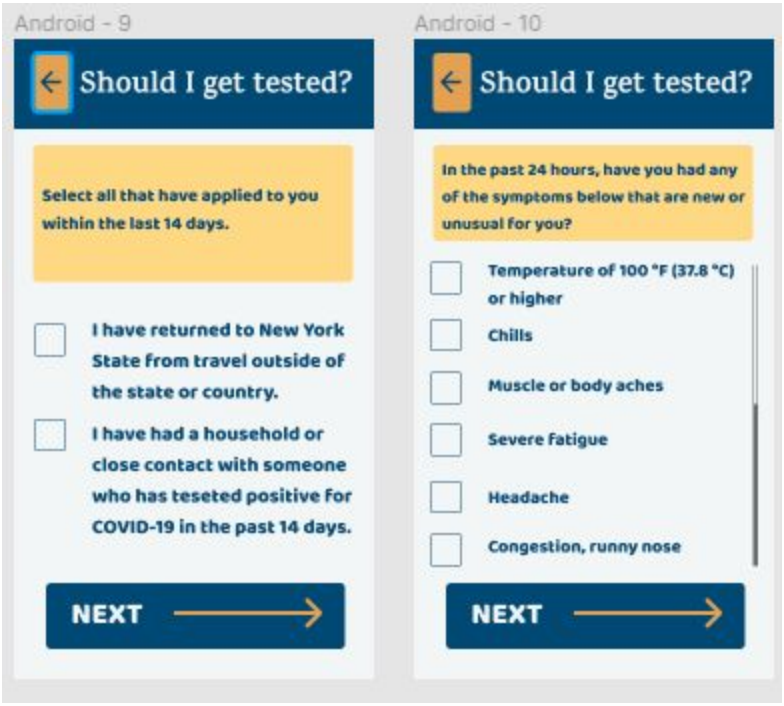
Current functionality of developed app

We had put in 3 test arbitrary locations to show the functionality as none of us had time to go out and log different addresses. However, this is our end goal for our project, which was made on Figma:
https://www.figma.com/file/sPqnYTxETdkHw45cuYrK9v/High-Fidelity?node-id=0%3A1



Figma overview

Sample home screen. Screenshot grabbed from
https://www.figma.com/file/sPqnYTxETdkHw45cuYrK9v/High-Fidelity?node-id=0%3A1



Sample questionnaire screen

To celebrate the app's roots at the University of Rochester (Built mostly by UR's students) and to offer a homely feel, the mockup was designed using the school's colors and beloved yellow jacket mascot.

Developer's grievances:

So all in all, this is the functionality we achieved in the 36 hours of this Hackathon:

1. Gather GPS information from Unique Users(Retrieved lat, long and converted to
2. Generate unique token for user to allow for association in the DB
3. Create unique UI that
    a. Display previous locations and last time there was an outbreak in that location
    b. Have a button for COVID symptom checking with scoring
    c. Have a button that allows the user to notify others that they have contracted COVID-19
        i. Notificate all users who were in locations with Patient Zero in the last 2 weeks 4. Add map, oriented on most recent location with pins to all recent locations

There was a VERY strong learning curve, but we were fortunate to have each others' base knowledge and resources offered by DANDYHACK's workshops, and good old Google. ImmediAlert required the use of a variety of tools: APIs, Android SDK (android studio), Django, Nginx, AWS, and CockroachDB, some of which none of us had experience working with. The app was coded mainly in Java, with a backend in Python. Most importantly, the team had no experience with Android dev despite starting out wanting to make an android app, so we had to teach ourselves with great help from M&T Tech office hours on Android development and Grace Ling's introductory workshop on UI/UX and Figma.

Of course, if we had more time, we would have implemented all the frontend we explored using our Figma mockup, such as hooking up our backend with our middleware. Right now, the project is in 3 separate pieces, and needs to be pieced together. We also want better security for our backend, as our end goal is anonymity, so users can confidently report data without worry, enhancing the overall user experience.