COVIDSCAPE: A Personalized COVID-19 Risk Prediction Service

Shuo Wang Hsu[1], Gabriel Oré[2], Wanwiset Peerapatanapokin[3], Aditya Sahu[4]
Viterbi School of Engineering
University of Southern California
Los Angeles, USA
{hsushuow[1], ore[2], peerapat[3], sahua[4]}@usc.edu

***Abstract*—Covidscape is a web-hosted service which provides a predicted risk score, of the possible exposure to the COVID-19, in the Greater Los Angeles Area. By utilizing Node-RED-based infrastructure to gather the GPS data from the users' devices, and PySpark to generate predictions using official daily testing data, the server then determines a risk score. Additionally, Covidscape leverages the data it obtains to further implement contact tracing on diagnosed users and notifies other users of potential contact. Furthermore, it provides a personalized solution to the users to give them a sense of security and alertness by being more informed and understanding of the risks and potential benefits of taking recommended precautions.**

***Index Terms*—COVID-19, Internet of Things, Machine Learning, Risk Estimation, Amazon Web Services, Node-RED, Contact Tracing**

I. Introduction

Covidscape is intended to be a user-friendly application. It allows the users to determine the COVID-19 risk levels of a certain "city" at any given time within Los Angeles County. Covidscape can predict the risk factors for each city for up to three days in advance.

With respect to Covidscape, we took a few assumptions at the user's end. First and foremost, we assumed that Covidscape machinery has access to the GPS location on a user's mobile device. Secondly, the client input relating to safety procedures are accurate and were performed in public. And finally, the users are voluntarily going to report if they have contracted the virus. With these assumptions, Covidscape generates a risk-level for each city in Los Angeles. Furthermore, these risk-level are calculated based off official data and reputable studies. Certain risk factors, such as *no mask, staying indoors or outdoors*, are considered "static", and hence, act as multipliers to the algorithm's result of the risk level.

The user's input will be used to generate the risk factor level for the algorithm to determine the risk level. The GPS log is also saved from the user's mobile device, which acts as a primary contributor in predicting the risk factors. When it comes to a possible case of exposure to the COVID, the user can report to the application about the incident. This allows the Covidscape to backtrack to other users that may have been in contact with an infected individual based off the GPS data that was gathered from the user. The users is then alerted through email and text messages.

II. Background and Motivation

Currently in regard to the COVID-19 pandemic, society faces endangerment on various levels. Due to some parts of normal life requiring to be maintained in the wake of the pandemic, i.e., grocery shopping, pharmacies etc., the number of cases continue to rise. Although COVID-19 restrictions were enforced in various places, the decision of authorities for easing of these restrictions on non-essential businesses and leisure events played a major role in recent resurgences of the cases. To counteract the virus, safety guidelines are enforced locally as recommended by the CDC, such as wearing mask, social distancing, and remaining outdoors when in public. As a result of the rising number of cases, different shutdown restriction levels, and safety precautions enforced, it is increasingly more difficult to determine the risk level that one may be exposed to at a specific location on a certain day. The Covidscape application intends to quantitate the risk for an individual to be at a city level region on a specified day tailored to the safety precautions the users practice. Covidscape will even be available to provide future predictions for up to three days based off modelling of the current trend and medical studies on the effects of safety precautions.

III. Covidscape Architecture

Covidscape is a Client-Server modeled web service which utilizes several different off-the-shelf frameworks and services to offer on-demand personalized COVID-19 risk prediction as well as additional features such as backwards contact tracing for registered users.

The main components in the Covidscape architecture include:

- **Node-RED-based Data Collection**: Receives user device's GPS sensor data and uses WebHookRelay and OwnTracks in addition to Node-RED flow to achieve automated data collection and storage into S3 bucket.
- **Server**: A Python Web Server program hosted on Amazon Elastic Compute Cloud (EC2) instance to handle Client requests.
- **Client**: A Web Application hosted on Amazon EC2 instance providing a gateway for user interaction and receives inputs via both manual user inputs and GPS sensor location using Google Map services.

- **Analytic Node**: PySpark running on Amazon EC2 Jupyter instance generating sub-county level machine learning risk prediction scores.
- **Other essential components**: User mobile devices' GPS sensor, Daily Official COVID-19 Testing Data, and several Amazon AWS Services including Elastic Compute Cloud (EC2), Simple Storage Service (S3), Simple Notification Service (SNS) and Simple Email Service (SES)

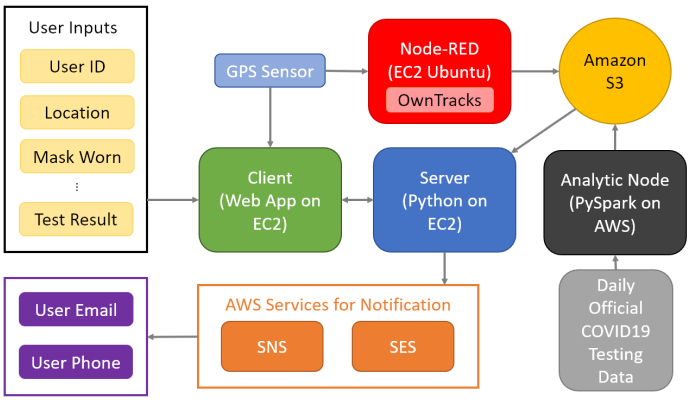

Figure 1: High-Level Architecture of Covidscape

Fig. 1 illustrates the general architecture of Covidscape. The detailed implementations of each main component is further discussed in detail in their respective sections.

IV. Node-RED-based Data Collection

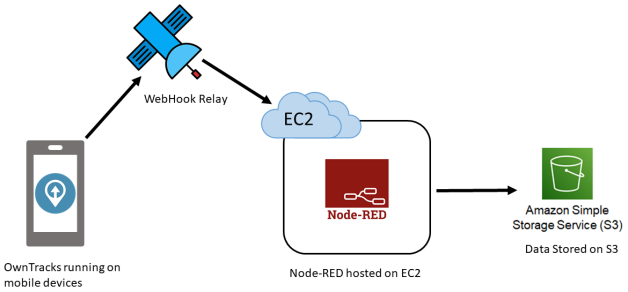

Figure 2: Architecture of Node-RED data collection mechanism

We used a few tools to devise an optimal method to collect GPS data for Covidscape.

Prerequisites [1]

- Installing and setting up the **OwnTracks**.
- Configuring a **WebHook Relay** agent and setting up a bucket. Furthermore, we need to create a public endpoint for the data transfer to take place.
- Setting up an **AWS S3 bucket** [2].
- Creating an **AWS EC2 instance** [3], which is going to act as the "bones and muscles" for the Node-RED instance.
- And finally, implementing and deploying the **Node-RED instance**. [4]

The elements used in the implementation are as follows:

A. *OwnTracks*

An open-source application for iOS and Android, OwnTracks gathers GPS data from user's mobile devices and sends it to an endpoint of the user's choice in JSON format. It relies on the device's OS to learn whether the device has moved, whereupon, OwnTracks sends out message with its current coordinates. Additionally, OwnTracks can be configured to send data manually, periodically or to remain active at all times.

In Android devices it keeps running in background, whereas, in iOS based devices it is "killed off" by the kernel and is "woken up" at certain intervals of time to perform its operation. [5]

OwnTracks works in two modes:

1) **MQTT mode** [6]

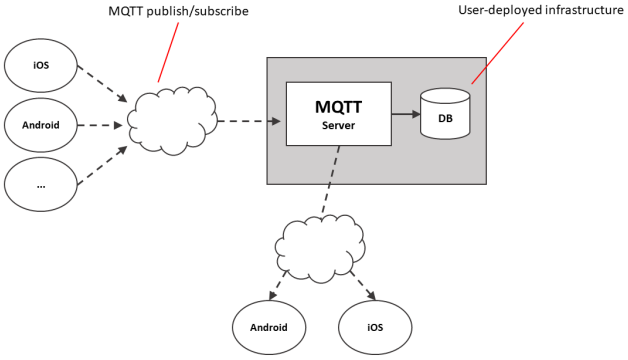

Figure 3: MQTT architecture

[5]

MQTT is a Machine-to-Machine, or the "Internet of Things", connectivity protocol. It is an extremely lightweight transport method, where one or more clients can *publish* or *subscribe* to *topics*. A topic addresses the type of messages that are being published by the client. The lightweight nature of MQTT broker can prove to be rather convenient if a client decides to setup his own infrastructure, for example, using Raspberry-Pi. The service to which an MQTT client connects, is called an *MQTT broker*.

In summary, a client publishes messages using MQTT broker, and another client subscribes to those messages.

2) **HTTP mode** [6]

OwnTracks uses HTTP mode as an "optional" feature. In this mode, HTTP is used to transmit all the generated JSON payload to the target endpoint. The user can configure an HTTP server with a public endpoint, which can then be used to POST requests over HTTP instead of publishing it using MQTT.

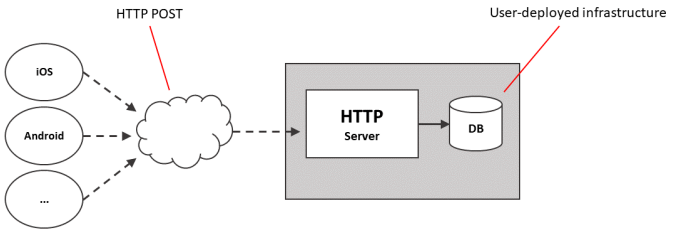

Figure 4: HTTP architecture

[5]

Usage : OwnTracks can send data to either an MQTT broker or to a standard HTTP endpoint. But for the sake of this project, we are constraining ourselves to an HTTP server, or more specifically, a WebHook Relay bucket.

B. WebHook Relay

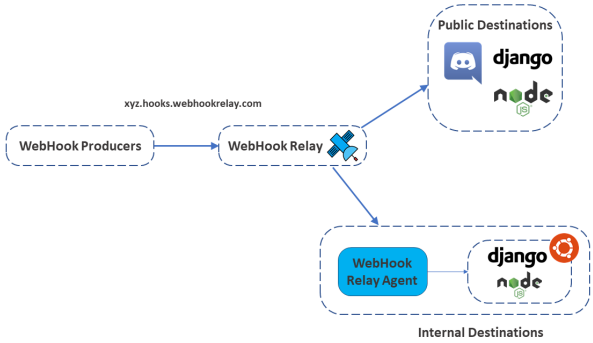

Figure 5: WebHook Relay Architecture

[7]

A webhook, also called HTTP push API, is an API driven tunneling solution. It is basically an **event-based output mechanism**. By default, transmissions in a WebHook relay is uni-directional. A WebHook Relay uses its internal network to relay the data. Therefore, The data is delivered to the target destination immediately, which makes them very efficient. Additionally, since it uses its internal network to pass on the data, it does not require any public IP or NAT configuration, which keeps Node-RED from being exposed to the internet.

Usage : A user can setup his own relay agent. However, in this scenario, we used a webhook service called WebHook Relay. For this service to come into play a user needs to follow these steps.

- Create a WebHook Relay account.
- Setup a WebHook Relay bucket. **Buckets** are grouping mechanisms for a user to configure input and output endpoints.
- After this, the user will get an input endpoint which are unique for every instance, such as **https://xyz.hooks.webhookrelay.com**.
- Configure the target endpoint. The user can either use WebHook's internal network with **http://localhost:8080**, or an HTTP server with **http://xyz.com**
- Use the endpoint to POST the data to the bucket, which will then immediately be relayed to the target endpoint.

C. AWS Simple Storage Service (S3)

[2]

AWS S3 is a cloud-based storage service developed by the Amazon Web Services, which was launched in the US in 2006. It provides *object storage* through a *web service* interface. The AWS describes it as a "service to make web-scale computing easier for developers". AWS uses the same scalable resources and infrastructure to support S3 that Amazon uses for its operations across the globe [8].

S3 can be used to store and collect any type of data from anywhere. Additionally, it can also be used to "protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics". [2]

The objects stored in S3 are organized into buckets. These buckets can be managed by the AWS console, or by the AWS SDK, or with the S3 REST API. In addition to that, objects can be downloaded using HTTP GET interface or the BitTorrent protocol. [9]

Usage : The AWS S3 bucket used here acts as the primary storage for Covidscape. The user must configure and setup the bucket. Since S3 promises durability, redundancy and scalability, all the datasets used in this project were stored in S3.

D. Node-RED

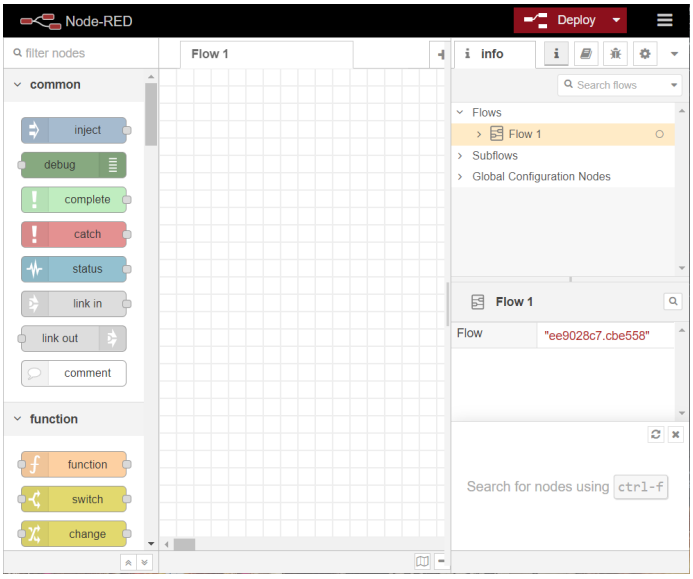

Figure 6: Node-RED API

Node-RED was originally created by the IBM Emerging Technology Services to integrate IoT hardware devices and

APIs. It is now an open-source tool, which works on the basic principle of the **flow-based programming**. It is a JavaScript-based development tool, which is built on Node.js platform. It also provides a browser-based flow editor.

The Data-flow programming in the Node-RED are called *Flows*, which consists of *Nodes*. These nodes are connected by wires, and each node is represented by an appropriate icon. Furthermore, Every individual node has a well-defined purpose; that is fed some data, and after performing a predefined operation, the node then passes the data to the next node.

Node-RED can operate in two ways: **First**, drag and drop method, where the nodes are placed on the panel and are connected by the wires; and **second**, where the flow can be imported as a JavaScript code. [10]

Node-RED is an extremely flexible tool, which can be used to create and deploy IoT-based applications quickly. Its visual representation makes it widely accessible to a large number of users.

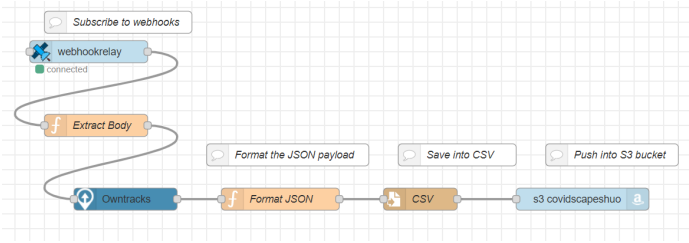

Figure 7: Flow of Node-RED data collection mechanism

Fig. 7 The flow is straightforward. Webhook relay node receives the data, and forwards it to function node to extract the body. After that, ownTracks node decrypts and returns the payload. Later, the updated payload is forwarded to a new function node, which extracts the necessary elements required for the Covidscape. Next, the File node creates a file locally, and appends the incoming data. Finally, each event causes the file to be updated and/or overwritten in S3 bucket.

Usage : The implementation of the Node-RED in Covidscape comprises of basic flows which are as follows :

1) **WebHook Relay Node**

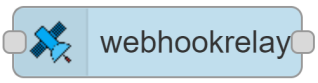

Figure 8: **node-red-contrib-webhookrelay**

WebHook Relay Node is a webhook tunnel-based receiver. A WebHook relay agent uses its *internal network* to relay the incoming traffic directly to the Node-RED instance.

2) **Function Node to extract the encrypted body from the payload.** This node returns the body of the payload and forwards it to the OwnTracks node as shown in figure.


```
{"_type":   "encrypted",
"data":     "qKjJgTyt\/YTqRdK5cGoCjbYcfzv9F8w
PgmwiyEUQFGNhcqn3d9LK1+ocP1UnIaCD7MfMSCB5e2y8
wHRXM8abzzCEC59Pm1mfVlKLX8+57vAJJ1pl3ahitcq4j
+mgoPsNEF\/slk0XpX2HN5AZoW5RIvjqPaBQhJlMUxV7D
gbuH9gtEQyC8Xt8vLXOcJhqxVxasBl8Ssh8MQzUeY6B9t
+YC\/tm48wMplWA6gUGgoQeNqye4\/p4aODuBpxro7M7X
6YeGR8+ujS3FlytcBrjsU1M+VLwy3ewXukMehskl3Z1Kt
gb\/7tsk2nQ2RrYP6wGU0xtkXtskgT\/sldeQvopMn\/P
MVJHzmNmfopE968hKobLddx3BXhcyf0HJjxXDNn3YBr48
jfkbTbs6ocxqX2iVbMYxiCjRKuiGBSglLR4ZvoSiYX5iy
9Y1YUt9NezvwmHWkt6MyVqIV04"}
```


Figure 9: Encrypted data from the OwnTracks

Figure 10: Function node to return the body

```
return {
        payload: msg.payload.body
    };
```

Listing 1: Source code to return the **body**

3) **OwnTracks Node for decrypting the body.**

Figure 11: **node-red-contrib-owntracks**

The data from OwnTracks can either be encrypted or in unecrypted JSON format. The access key on both the sides, the client and the Node-RED instance, must be the same for this Node to work. This node decrypts the data from the payload, and returns the data that was originally sent from OwnTracks application running on mobile device.


```
batt: 48
lon: -118.288294
acc: 65
p: 101.291
bs: 1
SSID: "Unknown Network"
BSSID: "f0:81:75:f7:e:8f"
lat: 34.030419
topic: "Aditya's Phone"
conn: "w"
tst: 1606725455
alt: 58
type: "location"
tid: "51"
```


Listing 2: Raw decrypted data from the OwnTracks node

4) **Function Node to extract the necessary component from the payload.** From the *earlier data*, the payload was observed to be rich with GPS-based information. However, latitude, longitude, user ID and timestamp are the only necessary elements required for this project.

Figure 12: Function Node to extract GPS elements

```
var data = msg.payload.topic + ","
+ msg.payload.lat + ","
+ msg.payload.lon + ","
+ msg.payload.tst;
msg.payload = data;
return msg;
```

Listing 3: Source code to extract name, lat, lon and timestamp from the payload.

Data returned from Function-II

```
"Aditya's Phone,34.030423,-118.288278,
1606672311"
```

Listing 4: Data returned from the payload

5) **File Node to create the file and/or to append the incoming data to the existing file.** Alternatively, it can be configured to delete the file based on the events.

Figure 13: File Node

6) **AWS S3 Node** to upload the file, which is stored locally, to the S3 bucket. Properties of the bucket are -
   - `msg.bucket` property is used to specify the name of the bucket in S3.
   - `msg.filename` property is used to name the input file in the bucket.
   - `msg.localFilename` property is used to take the content from the local file.

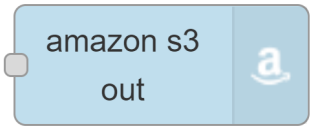

Figure 14: **node-red-node-aws** [11]

"Fig. 14 illustrates Amazon S3 out node. Uploads content to an Amazon S3 bucket. The bucket name can be specified in the node bucket property or in the `msg.bucket` property. The filename on Amazon S3 is taken from the node filename property or the `msg.filename` property. The content is taken from either the node `localFilename` property, the `msg.localFilename` property or the `msg.payload` property." [11]

V. Server Implementation

Covidscape's Server is implemented in Python using two main libraries: *Flask* [12] and *Boto3* [13]. These two libraries allows Covidscape to effectively and efficiently communicate with Client's Web Application and interact with Amazon Web Services (AWS).

Flask is an open-source micro web framework developed by Armin Ronacher of Pocoo, an international group of Python enthusiasts, in Python. *Flask* emerged as a natural choice of framework to use for Covidscape's need for a web server in Python due to its simplicity, popularity, and maturity as a framework. *Flask* provides a simple interface for the Server to route and handle different Client HTTP requests and respond accordingly in a straightforward fashion. Specifically, *Flask* offers the extension *Flask-CORS* to handle Cross Origin Resource Sharing (CORS), making cross-origin Asynchronous JavaScript And XML (AJAX) possible to facilitate interaction with Client Web Application.

Boto3 is Amazon AWS's own Software Development Kit (SDK) for Python. *Boto3* enables a central gateway to interacting with different Amazon Web Services. *Boto3* offers a straightforward interface to interact with all three of the services which Covidscape Server utilizes: S3, SNS, and SES.

- **S3**: *Boto3* provides straightfoward support for reading in files directly from S3 buckets which are then parsed by the Server code into Python dictionary structures using Python's *csv* and *codecs* library
- **SNS**: *Boto3* provides APIs for creating and subscribing users to SNS topics which supports Covidscape's functionality to send personalized notification SMS text messages to specific users.
- **SES**:While Amazon SNS also has the ability to send email notifications, the functionality was designed to send broadcast emails to a private group within an organization instead of customized notifications for public users. SNS requires the users to subscribe to a topic before emails can be sent to subscribers and it is not designed to send emails to a subset of subscribers, making it an undesirable choice for Covidscape's usage for sending notifications to specific users only. Therefore Covidscape uses SES for email notifications as that fits the purpose much better and enables users to verify email upon registering and simply receive the personalized notification emails later without any other setup.

In our initial prototype, Covidscape's Server program is hosted by an Amazon EC2 t2.micro Ubuntu 20.04 instance which has been sufficient for our testing and light usage. The Server program architecture has been designed to easily support redundancy which is slated for immediate future work to be able to handle larger loads and increase service reliability.

The specifics of Covidscape's server implementation can be separated into three main parts:

1) Client-Server Communication
2) Risk Score Calculation
3) Backward Contact-Tracing of Diagnosed Clients

A. Client-Server Communication

Client-Server communication is achieved by Client sending HTTP GET requests to Server, which responds to the Clients according to the type of request received. The Client generates and sends the HTTP requests through the hosted web application which will be discussed further in the User Interface section. The Server program uses *Flask* framework to host the web server in order to handle incoming HTTP requests from multiple users of the client application efficiently. The Server is able to differentiate different types of requests by the unique input data fields in the HTTP GET requests from the Client.

Three main types of Client requests are:

1) **Client Registration**: Server stores the user information and adds user email to be verified by Amazon SNS. Required user inputs include:
   - User ID
   - User Email
   - User phone number
2) **Risk Score Prediction**: Client submits a request for a total risk score prediction by entering user inputs of static risk factors and using specified GPS location or instructs the server to use the latest gps location collected by Node-RED from user history. Required user inputs include:
   - Static risk factors (whether client is wearing mask, location is indoors or outdoors, and social distancing is performed)
   - The day (today, tomorrow, or two days in advance) to generate predictions
   - Either an indication to use latest Node-RED GPS data with User ID or a specific Latitude and Longitude.
3) **Report Diagnosed User**: Users may voluntarily report when they have been diagnosed with COVID-19 through the Client web application. Server handles this report by storing the user information and initiating a backward contact-tracing on the diagnosed client. Required user inputs include:
   - User ID
   - User Test Result
   - Timestamp indicating the time the testing is performed

B. Total Risk Score Calculation

When a client requests for a total risk score, the server is responsible for calculating and returning the total risk score to the client. The total risk score is consisted of a base score augmented by several multipliers. The base score is the machine learning-generated sub-county level risk prediction score and a set of carefully defined static risk scores act as multipliers to the base score.

- **Base Score: Location-Based Risk Prediction Score** is obtained through the following steps:
  1) Analytic Node generates daily sub-county level risk prediction score and saves the result into an S3 bucket
  2) The Server performs a search for the specified GPS location's corresponding region or city by calculating the distance between each region's center to the user's input GPS location. Covidscape uses the Haversine formula as opposed to Vincenty formula. Haversine formula calculates the great-circle distance between two points on a sphere given their longitudes and latitudes. Haversine is less computationally intensive and more appropriate for recommendation based applications as discussed in [14]. Then the Server performs a linear search for the exact nearest city as there is no widely accepted or known algorithm that is generally more efficient as discussed in [15].
  3) The Server extracts the predicted score for the requested date of prediction of the closest region corresponding to the user location as the base score.
- **Multipliers: Static Risk Factor Scores** are carefully chosen risk factors backed by referenced scientific studies with their respective effect on the potential risk of contracting the COVID-19 pandemic. These static risk factors enable Covidscape to tackle risk prediction on a personal level as opposed to other efforts which have primarily focused on a certain location or certain groups of people. These personal static risk factors are obtained via user input:
  1) **Mask Worn vs. Not Worn**: UC Davis [16] reported the study [17] indicating that not wearing a facial mask increases the chance of contracting the COVID-19 virus by 2.86 times.
  2) **Indoor vs. Outdoor**: [18] and [19] have studied and discussed that being indoors increases the chance of contracting COVID-19 virus by an average of 18.6 times.
  3) **Strict Social Distancing vs. No Social Distancing**: John Hopkins [20] released the study [21] which concluded that not practicing social distancing increases the chance of contracting the COVID-19 virus by 4 times.

Even though the final combined total risk score is normalized to a value between 0 and 1 to better quantify the risk for clients to understand, there are several known challenges in defining, understanding, and utilizing the risk factor.

First, this risk factor should include additional scientific studies' results to back the chosen risk multiplier values and can also be further improved by introducing additional personal risk factors into the equation. The total risk score calculation also does not take into account the potential of related static risk factors affecting the efficacy of another e.g. the benefits of wearing a mask may be reduced if subject is

outdoors and practicing social distancing.

Second, the interpretation of the final risk score is also an interesting challenge. A reasonable choice is to split the risk into discrete sections, such as high and low risk. For demonstration purposes we have chosen to select 0.2 as the threshold where a total risk score smaller than 0.2 is considered low risk. This choice of final risk score threshold is completely based on experimented values and not backed with sufficient scientific studies.

Finally, Covidscape leaves the decision of what to do with the prediction score completely up to the users. A possible future work would be collaboration with health authorities and experts to provide possible recommendations and next steps according to the specific risk score. Covidscape could potentially recommend users to perform certain preventive measures due to their specific circumstances and situation to lower their risk of contracting the COVID-19 virus.

C. Backward Contact-Tracing Diagnosed Clients

Covidscape utilizes the available user data history to perform backward contact tracing on clients who have been diagnosed with COVID-19 and alert other clients who may have been in contact with a diagnosed client in the past fourteen days. The contact-tracing implementation can be best illustrated as the following flow of actions:

1) User reports test result via the Client web application. Server stores client input and initiates a backward contact-trace on the diagnosed user
2) Server obtains the latest set of user location history from Node-RED's user location database and builds up a complete user history map
3) Server performs a search and extracts the list of users who may have been in contact with the diagnosed client. Due to the periodic nature of user GPS location logging, the current Server implementation uses the following loose constraints on the tolerances of location and time to consider a potential contact:
   - **Latitude and Longitude**: 0.0001 degrees indicating ± 5.55 meters radius
   - **Time**: ± 30 minutes (within an hour of diagnosed client's timestamp)
4) Server sends email and phone SMS text notifications to the list of users to notify that they may have been in contact with a diagnosed client.

This functionality differs from Apple iOS and Google Android's attempts in several ways:

- Does not require Bluetooth functionality and consequently would not 'miss' devices due to Bluetooth being turned off, device being busy, too many devices in close proximity, signal interference, or other reason that may cause a failure to establish connections.
- Centralized approach where data storage and computation are handled in the cloud
  - No additional local CPU computation, storage needs, or battery drain on user devices
  - Data is retained even if device is lost or reset
  - Supports both mobile and desktop environments
- Not dependent on mobile OS version/updates or specific device hardware other than internet connectivity and access to a GPS sensor
- Does not require higher level approval from local health authorities of the specific country or region and does not need local health authorities to create their own applications.

The discussed backward contact tracing functionality in the current Covidscape implementation is an initial approach to the functionality. Several improvements on the current design being explored include:

- Providing additional information in the notification emails and text messages such as the exact time, location, and duration of the possible contact along with the user's predicted risk scores at the time of contact.
- Optimizing architecture and contact tracing algorithm to handle larger user history data. It may be possible that the current single server may be come overwhelmed with large amounts of user data to trace.
- Combining Covidscape's contact tracing method with Apple and Google's approach by utilizing their developer APIs to offer even better coverage and functionalities.

VI. Client - User Interface

The Covidscape client is a web application. It is made with standard web development tools like HTML, CSS and JavaScript. Additionally, we are using Google Maps API to represent our geographical data and provide localization feature. The client is also required to communicate with the server node to calculate risk scores and for the back trace functionality which is done using HTTP GET method as stated in section V-A.

The advantage of presenting our work with a web application is that it is highly flexible and can support both desktop and mobile usage with one implementation. Another advantage is that its development process is simple with resources being readily available online. In contrast an android application has more functionalities but the learning curve is steep and there are less resources available online. Covidscape is in its prototype stage and a web implementation can more efficiently showcase and explore our ideas.

In total, there are 3 main pages to the web UI:

1) Home page
2) Report page
3) Login page

A. Home page

The home page is where the user can conveniently find the main functionalities of Covidscape. Fig. 15 shows a sample of the home page on desktop and Fig. 16 shows the mobile version. The map from Google Maps API covers most of the UI. The map centers at the University of Southern California by default and serves to visualize COVID-19 risk areas in

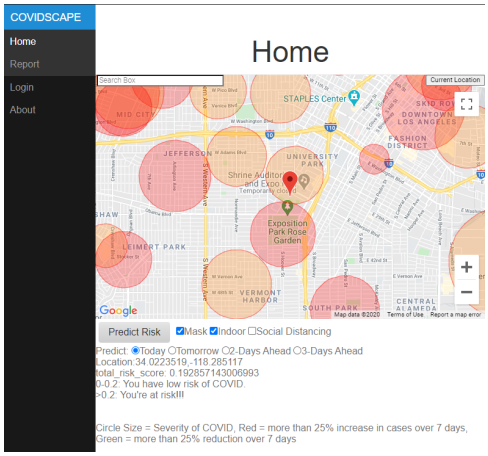

Figure 15: Desktop Home Page

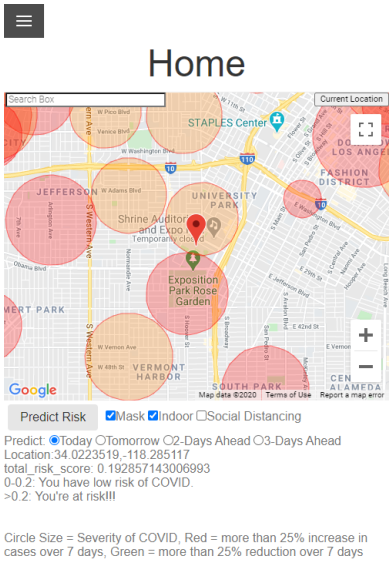

Figure 16: Mobile Home Page

LA. A circle is drawn at the center of each city. We then represent the risk score of each city in LA county by the size of the circle. The bigger the circle, the higher the concentration of population with COVID-19. The details of how the geographical risk score is calculated is covered in VII Analytic Node - Machine Learning. The color of the circle is another visual representation. It represents the 'growth' rate of COVID-19 in an area. Red circle means there is more than 25 percent increase in COVID-19 cases over 7 days. Green circle means there is more than 25 percent reduction in COVID-19 cases over 7 days. Yellow circle means COVID-19 growth is anywhere between the green and red boundary.

The localization functionalities are located at the top-left and top-right of the Google Maps UI. On the top-left shows a search box with auto complete and drop down feature. On the top-right shows a 'Current Location' button that selects the user location with permission. Once a location is selected, the map re-centers at that location and a red pin is dropped to specify the exact place.

Once the user specifies a location, whether it is their current location or some other interested location, they can click on 'Predict Risk' to calculate the COVID-19 risk score of being in that location. The button is located to the bottom-left of the Google Maps UI. To the right of the 'Predict Risk' button, there are check boxes for the user to indicate their COVID-19 precaution including mask wearing, indoor or outdoor, and social distancing. How these option affect the final risk scores is covered in section V-B. Under the 'Predict Risk' button, a radio-button selector can be selected to calculate the current score or the future score of visiting the interested location. We provide risk scores of up to three days ahead. Once the user clicks on 'Predict Risk', the client communicates with the server node through HTTP GET and displays the requested risk score below the button. Our final score ranges from 0 to 1, we have categorized a score of 0-0.2 to have low COVID-19 risk and a score of more than 0.2 to have high risk of COVID-19.

B. Report page

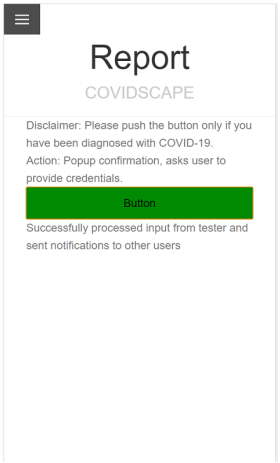

Figure 17: Report Page

Fig. 17 shows the report page UI. The report page design is plain and simple, it aims to grab the user attention to the big green button in the middle of the page. The user shall press the report button only if they have been diagnosed with COVID-19. To prevent clicking the button by mistake, the user is asked to provide their username before proceeding. Then, the client will tell the server to notify all other users who have been in close proximity with the infected user in the past 14 days. The notified users will then know to take higher precautions as they might be infected with COVID-19 and could spread the disease. For the user who pressed the button, a message will appear under the button informing whether the report was successful or not. Note that in the current scope of Covidscape we have not address the privacy and bad actor concerns of this functionality. However, it is a priority in the next stages of development.

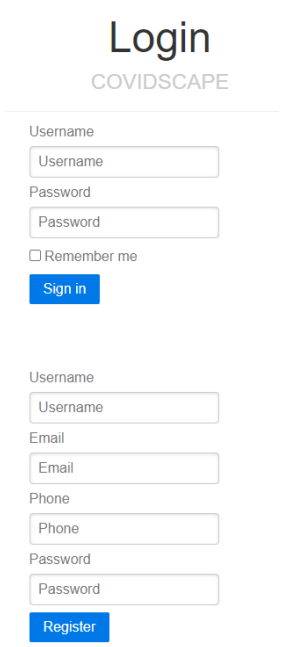

Figure 18: Login Page

C. Login page

Fig. 18 shows the login page UI. The login page is quite standard of what might be found on other websites. It consists of a Sign in feature on the top and a Register feature on the bottom for first time users. To register, the user must provide a username, email, and password. Once the register button is pressed, the client communicates with the server and a message is shown whether the registration was successful.

D. Web hosting

The client site is also hosted on an Amazon EC2 instance. EC2 allows us to easily create a virtual machine that has a public IP interface on Amazon servers. This way, the instance will be readily available at all times for usage. We chose a free tier instance with low computing power as it is sufficient for our current scope and user base. Furthermore, the computing power of this instance is not utilized, only the network performance is important for the user to download web pages. We then host the web using Apache HTTP server. Now, Covidscape can be easily accessed through the EC2 instance's public IP Address.

VII. Analytic Node - Machine Learning Sub-County Level Risk Prediction

The Analytic Node is responsible for the sub-county level prediction score calculation. It takes input from a reputable data source in the form of COVID-19 density per area and uses machine learning to predict scores up to 3 days in the future. The score is then returned to a user with a value between 0 to 1 for presentation purposes. As the data is updated daily, the analytic node re-computes the geographical prediction everyday.

A. Data Source

Our data source for the sub-county level COVID-19 data is given by the County of Los Angeles Department of Public Health [22]. This data is also referred to as city level cases data. The data from the Department of Public Health is published as a daily article and is publicly available online. For each city in LA two numerical data is given, total case count and cases per 100,000 population. We selected to use the rate data per 100,000 population, as it represents the COVID-19 density which allows us to compare cities of different sizes. There are numerous sources of public COVID-19 data, but other than data from the County of Los Angeles Department of Public Health there is no other sources we are aware of that shares a such a geographically detailed COVID-19 data.

B. Data Manipulation

The data from LA's Department of Public Health represents the rate from the beginning of COVID-19. It is likely that most of the COVID-19 patients have since recovered and are no longer contagious. A better representation of the contagious COVID-19 population would be the concurrent infected population. We subtracted the current data to the data 14 days ago to get the number of concurrent patients, this data reflects the contagious population rate/density. We chose 14 days as window as the Center of Disease Control(CDC) [23] estimates that the patient is contagious during incubation period which lasts 4-5 days and 10 days from symptom onset. We use the historical data of this value as the input to the machine learning algorithm. The machine learning algorithm then make predictions of the future values of concurrent COVID-19 patients in the same area. The values are then normalized from 0 to 1 as a base prediction score before storing on the S3 bucket. Further calculation is made with user risk factors in the server node, then the final score is represented to the user. Section V-B describes in detail how the server node manipulates the base score.

C. Machine Learning Algorithm

Now, what we have is a history data of concurrent COVID-19 patients per city, and we want to know the prediction of 1, 2, or 3 days in the future. The nature of this data is that it shows a trend, whether stable, increasing or decreasing and does not fluctuate significantly. The prediction is also represented as a continuous value, rather than a cate gorical outcome. We choose linear regression [24] [25] as our machine learning implementation. Linear regression is great at predicting trends from historical data as it constructs a linear curve that best fits our historical data (which has a trend of increasing or decreasing COVID-19 patients). The prediction then extrapolates the linear curve to future values of 1, 2, or 3 days in the future.

One question that comes to mind is how much historical data should we use as the input to the machine learning algorithm? If we include data since the beginning of COVID, the output trend will represent an average of over many months instead of recent trend. This information is unlikely to represent the current state of COVID in the area. Another way

to state this is the linear regression algorithm gives the same priority for every data point. But, in the case of historical data, recent values should be given more weight. However, using shorter history data will result in fewer data points and the outcome might overshoot due to high variance of the input. We also considered that there is unequal COVID-19 testing in the seven day week. It is might be that less COVID-19 testing is happening on weekends than weekdays. Therefore, we should take multiple of 7 day historical data points as input to the machine learning system. In the next section, we show the accuracy of our prediction as well as a comparison between 7-days and 14-days input of historical data.

We selected PySpark as our machine learning framework. PySpark is suitable for our implementation because of its map-reduce architecture. The map-reduce programming model allows us to perform filtering, sorting and counting the large data set with ease. This will allow us to analyse aggregate data such as calculating the average risk score or finding the top ten COVID-19 hot spots in the Greater Los Angeles Area.

D. Machine Learning Performance Evaluation

To measure the accuracy of our predictions, we compare the predicted value to the actual value given by Los Angeles County Department of Public Health on the day of prediction. First, the error of prediction is calculated as:

$$error = \left| \frac{(predictedValue - actualValue)}{actualValue} \right| \quad (1)$$

Then the accuracy is the inverse of the error:

$$accuracy = 1 - error \quad (2)$$

This is calculated for every city and for the prediction of current day, next day, next 2 days and next 3 days. The calculation is then repeated over ten sets of predictions. We take the average of the accuracy values and present them in the table shown at Fig. 19. In the table, two cases of input data is shown, this is to compare 7-days and 14-days input of historical data. We observe that 7-days input data has a 1.2 percent improvement in average accuracy over 14-days input data. And so, we selected 7-days historical input as the standard for Covidscape where it achieved 93 percent average prediction accuracy.

	0 Day	1 Day	2 Day	3 Day	Average
Accuracy (7 days input)	0.944419	0.919411	0.907682	0.882769	0.931915
Accuracy (14 days input)	0.930589	0.911223	0.893337	0.869249	0.920906

Figure 19: Accuracy of Machine Learning Prediction

E. Geographical Mapping

The machine learning output of the base risks scores per area is also used in Google Maps UI of the homepage. We require the coordinates of each city in LA to correctly show the scores on the map. We have to thank COVID-19 risk estimation project [26] of ANRGUSC research group for providing the coordinates of cities in LA. There are a total of 232 cities where we predicted scores and display on the map. [26] and [27] shows some risk estimation work similar to Covidscape, however they do not offer personalized risk estimation and real-time interactive user interface.

VIII. Conclusion and Future Work

The Covidscape's architecture has been described in detail with respect to usage of Node-RED, client-user interface, server implementation, and machine learning risk prediction. The design takes in user inputs to be sent to the server that will then provide a risk score back to the user. The risk prediction score is based on the machine learning-generated local risk score based on the reputable data that is collected based of the city-level current case counts augmented by static risk factors as multipliers to the score. With the friendly and intuitive user interface, users can determine their risk score in a quick and simple manner. It was also described that the users are able to easily report that they have been diagnosed to contract the virus, which then allows the server to notify the other users that may have been in contact with the diagnosed user using SMS text and email.

Currently, Covidscape has much potential to grow with several expansions in its domain. Currently the one server being used may instead be multiple servers to provide better reliability and attend to multiple clients at once for when the number of users increases. Expanding the number of users may also be possible through going beyond the scope of Los Angeles County to include other counties, or even expand to state or national level. To be reliably available, the data size would be large and would need to be recent, so an improvement to the server would be to access the data directly from a database instead of the csv files as a median that constantly needs to be updated. The users also already voluntarily share their GPS location on their mobile device, which may be useful to alert them when they enter a location that would be high risk. Another useful addition is to automate the Analytic node to gather data, upload data to the server, and run the machine learning algorithm daily.

Covidscape is still a work in progress with much flexibility for further enhancements and is a convenient and user-friendly tool that will allow the users to have a better idea on how to determine their risk factor to the COVID-19 virus. By providing a predicted risk score, Covidscape aims to eliminate uncertainties of where and when would be a higher risk location to contract the virus and allow individuals to be more conscious when they are at a higher risk location. With these uncertainties the virus lessen, this would allow users to make more sound decisions with the current health safety mandates that they would take part in.

References

[1] R. Karolis. (2019) Fast & simple location tracking with node-red and owntracks. [Online]. Available: https://dev.to/webhookrelay/fast–simple-location-tracking-with-node-red-and-owntracks-195o/

[2] (2020) Getting started with amazon simple storage service. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/gsg-/GetStartedWithS3.html/

[3] (2020) Tutorial: Getting started with amazon ec2 linux instances. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest-/UserGuide/EC2_GetStarted.html/

[4] (2020) Running on amazon web services. [Online]. Available: https://nodered.org/docs/getting-started/aws/

[5] (2020) What owntracks does. [Online]. Available: https://owntracks.org/booklet/guide/whathow/

[6] (2020) Owntracks guide : Scenarios. [Online]. Available: https://owntracks.org/booklet/guide/scenarios/

[7] R. Karolis. (2020) Introduction to webhook relay. [Online]. Available: https://webhookrelay.com/intro/index.html

[8] V. Persico, A. Montieri, and A. Pescapè, "On the network performance of amazon s3 cloud-storage service," in *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, 2016, pp. 113–118.

[9] A. Gupta, A. Mehta, L. Daver, and P. Banga, "Implementation of storage in virtual private cloud using simple storage service on aws," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020, pp. 213–217.

[10] M. Lekić and G. Gardašević, "Iot sensor integration to node-red platform," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018, pp. 1–5.

[11] N.-R. Community. (2020) node-red-node-aws. [Online]. Available: https://flows.nodered.org/node/node-red-node-aws

[12] (2020) Welcome to flask — flask documentation (1.1.x). [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/

[13] (2020) Boto3 documentation. [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

[14] H. Mahmoud and N. Akkari, "Shortest path calculation: A comparative study for location-based recommender system," in *2016 World Symposium on Computer Applications Research (WSCAR)*, 2016, pp. 1–5.

[15] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[16] R. Kushman. (2020) Your mask cuts own risk by 65 percent. [Online]. Available: https://www.ucdavis.edu/coronavirus/news/your-mask-cuts-own-risk-65-percent/

[17] D. K. Chu, E. A. Akl, S. Duda, K. Solo, S. Yaacoub, and H. J. Schünemann, "Physical distancing, face masks, and eye protection to prevent person-to-person transmission of SARS-CoV-2 and COVID-19: a systematic review and meta-analysis," *The Lancet*, 06 2020. [Online]. Available: https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(20)31142-9/fulltext

[18] H. Nishiura, H. Oshitani, T. Kobayashi, T. Saito, T. Sunagawa, T. Matsui, T. Wakita, and M. Suzuki, "Closed environments facilitate secondary transmission of coronavirus disease 2019 (covid-19)," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/04/16/2020.02.28.20029272

[19] K. Lyell. (2020) As activities move indoors this fall, risk of catching covid-19 increases, researchers say. [Online]. Available: https://www.coloradoan.com/story/news/local/2020/09/18/risk-catching-covid-19-increases-activities-move-indoors-researchers-colorado-state-university-say/5817487002/

[20] J. Eichberger and C. Kempler. (2020) Covid-19 study links strict social distancing to much lower chance of infection. [Online]. Available: https://www.jhsph.edu/news/news-releases/2020/covid-19-study-links-strict-social-distancing-to-much-lower-chance-of-infection.html

[21] S. J. Clipman, A. P. Wesolowski, D. G. Gibson, S. Agarwal, A. S. Lambrou, G. D. Kirk, A. B. Labrique, S. H. Mehta, and S. S. Solomon, "Rapid Real-time Tracking of Nonpharmaceutical Interventions and Their Association With Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) Positivity: The Coronavirus Disease 2019 (COVID-19) Pandemic Pulse Study," *Clinical Infectious Diseases*, 09 2020, ciaa1313. [Online]. Available: https://doi.org/10.1093/cid/ciaa1313

[22] L. A. C. D. of Public Health. (2020) Public health reports 16 new deaths and 5,014 new positive cases of confirmed covid-19 in los angeles county. [Online]. Available: http://www.publichealth.lacounty.gov/phcommon/public/media-/mediapubdetail.cfm?unit=media&ou=ph&prog=media&prid=2832

[23] C. for Disease Control and Prevention. (2020) Interim clinical guidance for management of patients with confirmed coronavirus disease (covid-19). [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/hcp/clinical-guidance-management-patients.html

[24] A. Spark. (2020) Classification and regression. [Online]. Available: https://spark.apache.org/docs/latest/ml-classification-regression.html#linear-regression

[25] K. T. K. Wang. (2020) Guide to implementing linear regression in pyspark and r. [Online]. Available: https://towardsdatascience.com/guide-to-implement-linear-regression-in-pyspark-and-r-26a94fe938a3

[26] B. Krishnamachari. (2020) Covid-19 risk estimation. [Online]. Available: https://github.com/ANRGUSC/covid19_risk_estimation

[27] S. Bodapati, H. Bandarupally, and M. Trupthi, "Covid-19 time series forecasting of daily cases, deaths caused and recovered cases using long short term memory networks," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, 2020, pp. 525–530.