

Contribution Report

Table of Contents

Contribution Report	1
Links	3
Github: https://github.com/ysumeet/CloudComputing	3
Application live url: https://s3797892-s3804879-covid.ts.r.appspot.com/	3
Summary	3
Introduction	4
Related work	4
Software Design/Architecture	4
Architecture Diagram:.....	5
Description:.....	5
Implementation:	6
Developer Guide:	7
Google App engine configuration:	7
S3 Storage configuration:.....	8
Lambda configuration:.....	9
API Gateway.....	9
CloudWatch configuration:.....	11
BigQuery Configuration:	11
Data studio Configuration:.....	12
Google Maps Configuration:	13
User Manual.....	15
Registration.....	15
Login.....	15
Change Username.....	16
Change Password	17
Update Location.....	18
Logout	18
Check Symptoms.....	18
View Statistics	19
References	19

Covid-19 symptom checker & test centre locator

S3797892 | S3804879

Student Name: Sumeet Yedula	Student Name: Sainath Reddy Kodiganti
Student ID: s3797892	Student ID: s3804879
Contributions: 1. S3 2. Lambda 3. Google Maps 4. App Engine 5. Version Control 6. Documentation	Contributions: 1. Big query 2. App Engine 3. Data Studio 4. Frontend 5. Deployment and dependency handling 6. Documentation
Contribution Percentage: 50	Contribution Percentage: 50
<i>By signing below, I certify all information is true and correct to the best of my knowledge.</i> Signature: Sumeet Yedula Date: 31-01-2021	<i>By signing below, I certify all information is true and correct to the best of my knowledge.</i> Signature: Sainath Reddy Kodiganti Date: 31-01-2021

Links

Github: <https://github.com/ysumeet/CloudComputing>

Application live url: <https://s3797892-s3804879-covid.ts.r.appspot.com/>

Summary

Cloud Computing has drastically changed one's approach towards the way of accessing data and processing power. It has brought a disruptive view for providing on demand information access and applications from anywhere at any time. Features of Cloud Computing include: Scalability, Resource Pooling, Easy Network Access, Availability, Easy Maintenance, Automation and Security. It is because of these features both the hosts and clients who are responsible for operations and management of their applications are viewing Cloud Computing as their ultimate resource. It is also very easy to use which alone makes its reach to the end users much more domineering than the available technologies.

Due to the current pandemic situation, there has been many discrepancies and delays in the information about the COVID symptoms that is being relayed to the masses. It is still not clear to the people what are the symptoms that they should be worried about, the extent up to which it affects the person and spread in the surroundings and where should they go to treat their symptoms.

It is these high reach, availability, and scalability features and services of Cloud Computing that we are leveraging upon to build and host the current web application which helps the common people to assess their symptoms and get to know the test centres and its details that are available in their location to treat the affected.

Introduction

2020- the year when world have realized the importance of health, hygiene, and self-care. All thanks to COVID-19. This year also help us realized the importance of having technology on our figure tips. So, the question that arises here is, can we combine the technology with heath care systems? The answer is YES. Here we propose an application which gathers data from the user, analyse if the users are symptomatic to COVID-19 and help one find the nearest testing centre. Also, provided the one stop analytical data of the COVID-19 statistics around the world, to generate awareness and keep our users updated.

This system provides users a questionnaire which will help us analyse if the users is symptomatic or non-symptomatic. If the users are found symptomatic, they are auto suggested the nearest location of COVID test centres from their current location. The next question here would be "Is it really necessary?". The answers would be YES again. The numbers of COVID-19 cases have increased dramatically. If we recall few months behind, we can observe that this is because of lack of awareness about the symptoms, like wearing masks, maintaining good hygiene, etc.,

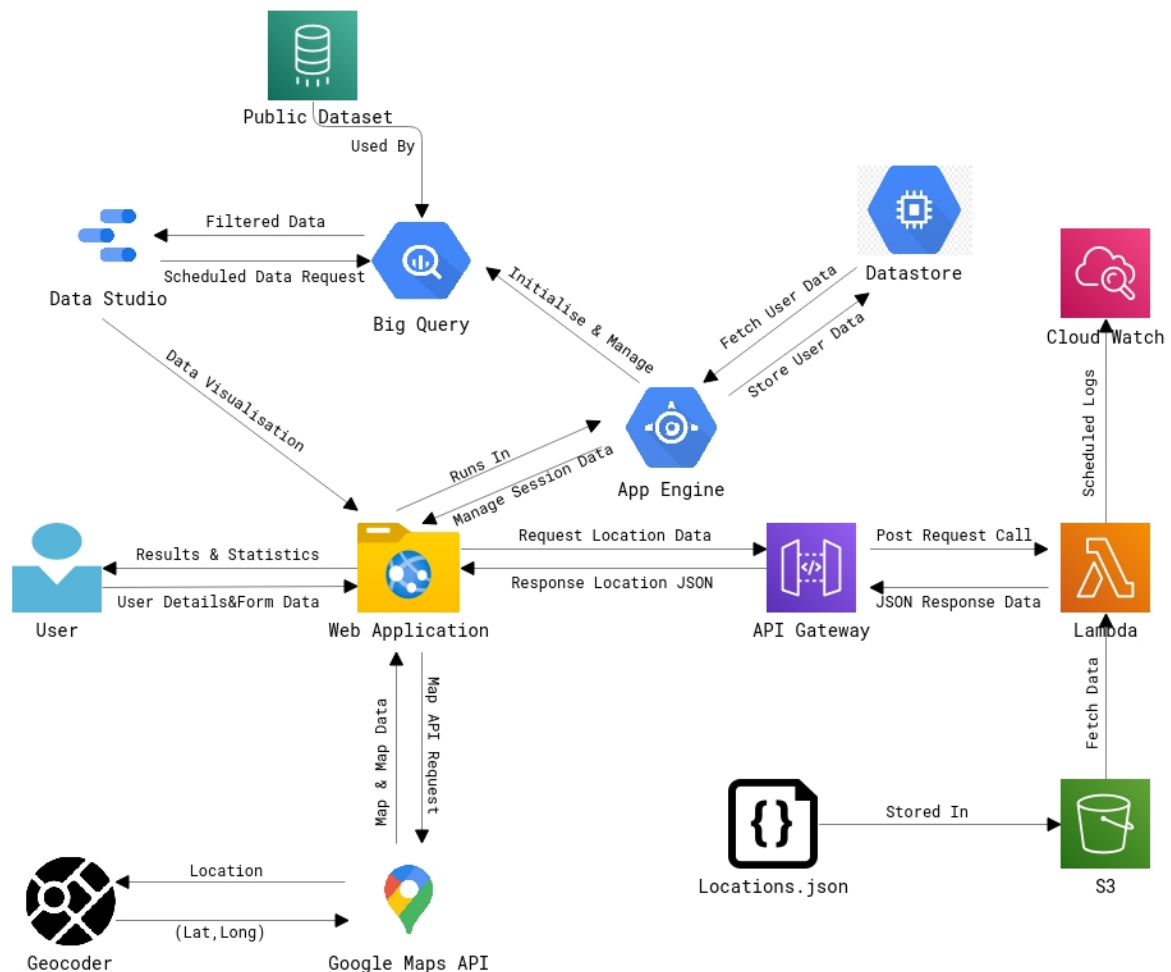
There is also an additional feature which show the real time statistics of the COVID cases around the world, and how the cases have increased in the different parts of the world.

Related work

There is no previous existing application providing the exact functionality intended by us. Nevertheless, we have drawn inspiration from sources such as the questionnaire from spectrum health^[1] and testing location data from public data set COVID-19 Testing Locations^[2]. We have also got idea for visualisation from various search engine results that are displayed for COVID-19 stats

Software Design/Architecture

Architecture Diagram:



Description:

The following diagram represents the high-level architecture that we have implemented for deploying the web application using the services provided by both Google Cloud Platform and Amazon Web Services. The following are the elements that are presented in the architecture:

AWS Simple Storage Service: is a service provided by Amazon web services that is used to store and retrieve any amount of data at anytime from anywhere using simple web services interface. Here in this project, we have created a S3 bucket and placed the “Locations.json” file in it which is later used for providing test centre location details.

AWS Lambda: is a serverless computing platform offered by Amazon web services which runs based on events and allocates the resources automatically that are required by the code based on the events. Here in this project, we have written a Lambda function for fetching the locations data from the json file, restructure it and hand it over to the API gateway for providing further response.

S3797892 | S3804879

AWS Cloud Watch: is used for monitoring application an infrastructure used in the architecture. It is used for monitoring all the resources and services used in the AWS architecture. Here it is us for monitoring the API requests and Lambda function used for handling the API requests and JSON data.

AWS API Gateway: is an API management service provided by Amazon web services for monitoring, handling and securing API's at any scale. It places itself between a client and set of backend services and acts as a proxy to accept all the application programming calls and submit it to the corresponding handler and return the fetched results from the handlers to the appropriate applications.

Google App Engine: is a Platform as a Service for hosting web applications in data centres. It helps developers build scalable web applications in any programming language on serverless platform. It provides many services for web developers for building web applications and connecting it to storage services for storing its data.

Google Big Query: is a fully managed serverless data warehouse for managing large volumes of data. It uses SQL for querying over petabytes of data and has some machine learning capabilities embedded in it. In this project, we are using the public data sets provided by google, accessing them using big query and filtering out the needed data through the queries and providing it to the user.

Google Datastore: is a highly capable NoSQL based database provided by google on Google Cloud Platform for offering storage services to the end users on the client. It allows you to store, save files from any location at any point of time and lets you access it on demand. It is scalable and lets you manage high volumes of data without any need for you to know the SQL for accessing its data contents. Here in this project, we have used google cloud datastore to store the user credentials and the form data entered by the user.

Google Data Studio: is a data visualisation tools used for representing the data in a more readable and presentable format and turning it into fully sharable and customisable dashboards and reports which can be later embedded and used in any web applications for further use and relaying information.

Google Maps API: is a rich Java script API that we have used for representing the location data in a visual format on the maps that can help people easily locate them. We made use of the Geocoding concept for converting the location address to its latitudes and longitudes for representing them as markers on the maps making it visually more appealing and approachable to the users.

Implementation:

The user is presented with a rich client-side web application through which he can assess his symptoms and access its contents. When the user enters the credentials and form data for assessing his symptoms through the front end, they are fetched and saved in the google cloud data store using a request handler. The web application as a proxy to the user makes a post request call to the API gateway which in turn acts as a proxy between the Lambda function that handles the backend services and the client. Then the function fetches the location data from the S3 bucket and provides it to the API gateway which returns the data as a response json object to the client-side application where it will be visually presented to the end user. The data here is nothing but the location data which is sent to the Google maps API where it will be parsed, and the data is presented.

Here in this project, we are also using some public data sets provided by google to represent some statistical information visually to the user with the help of google big query tool and google data studio. The big query tool parses the huge volumes of data in the public data set and filters out the information needed to present it to the user and sends it to the google data studio where it is converted into the visual format and then later embedded into the client-side application. Cloud watch here plays the role of maintaining logs and records of the API calls made to the lambda function for the location data and filter out the exceptions and errors if any.

This entire application is then made available globally to all the end users by hosting it online with the help of Google app engine. The google app engine maintains and controls the application by providing it its rightful responses from other applications and services from time to time and keeps it running. Hence in this way the following high-level architecture is designed and developed to make the client-side web application working and presentable to the end users.

Developer Guide:

This guide is to replicate the setup of the Environment. It also outlines the procedures that are followed for the development and hosting of the various cloud services used in the *COVID-Detector*. The application is a web based responsive site and primarily uses Google App Engine to serve the application and draws support from various cloud service to enable the versatility and elasticity that is offered as an advantage by cloud.

Google App engine configuration:

- To begin with proceed to <https://console.cloud.google.com/> and click on create new project



- Download and Install Google Cloud SDK <https://cloud.google.com/sdk>
- Download Install Git <https://git-scm.com/downloads>
- Install App engine extension for python using the command from command line/terminal:
`gcloud components install app-engine-python`
- Install additional python libraries using the command
`gcloud components install app-engine-python-extras`
- Run the following command to update all the installed Google Cloud SDK components, including the App Engine extension for Python
`gcloud components update`
- Install python 2.7 on the local OS and ensure the PATH variable is set correctly by executing
`python -version`
- Open the google cloud SDK shell and navigate to the Covid-Detector directory in the terminal
`dev_appserver.py app.yaml`
- Visit <http://localhost:8080/> in your web browser to view the app
- To deploy your app to App Engine, run the following command from within the root directory of your application where the app.yaml file is located:
`gcloud app deploy`
- To launch your browser and view the app at [http://\[YOUR_PROJECT_ID\].appspot.com](http://[YOUR_PROJECT_ID].appspot.com), run the following command
`gcloud app browse`

S3 Storage configuration:

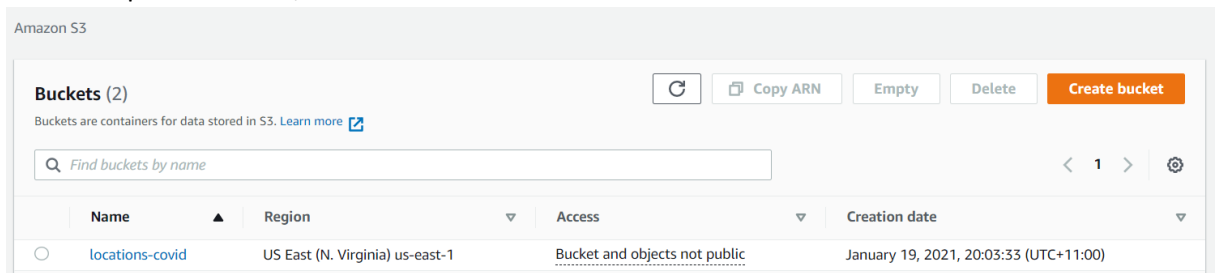
The Locations.json containing the data related to testing centres is stored in the amazon s3 bucket as an object for easier access and to be used by lambda further down.

- To begin with navigate to <https://s3.console.aws.amazon.com/s3> and select create a bucket
- Name the bucket *locations-covid*, enter any desired tags to help with better identification and select create bucket

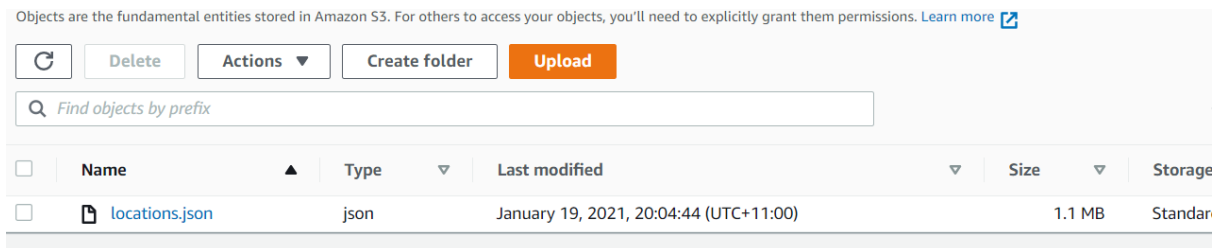
Covid-19 symptom checker & test centre locator

S3797892 | S3804879

- If the setup is successful, the console would look like this



- Select the *locations-covid* and select upload the json



Lambda configuration:

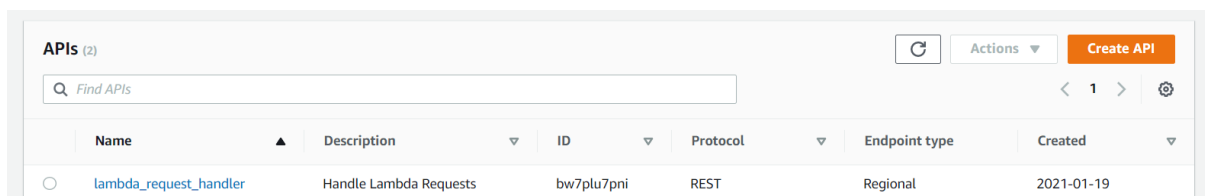
The locations from the json are served on REST call using the AWS Lambda service due to the benefits provided and deploy the code as a serverless function which be used by any service with simple REST call.

- Navigate to <https://console.aws.amazon.com/lambda/home?region=us-east-1#/discover>
- Select create a function and select Author from scratch enter the function name as *lambda_request_handler* and Runtime as Python 3.6 and select create a function
- Create a *lambda_function.py* and paste the code in the editor and use Test button to test the code

API Gateway

The AWS API Gateway sits in between the lambda function that webservices to expose the service as an access door for the functionality.

- Navigate to <https://console.aws.amazon.com/apigateway/main/apis?region=us-east-1>
- Select create API



- Select REST API and click build

S3797892 | S3804879

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

Import
Build

- Select New API and enter the API name and create API

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ New API
 ☐ Clone from existing API
 ☐ Import from Swagger or Open API 3
 ☐ Example API

Settings

Choose a friendly name and description for your API.

API name*

Description

Endpoint Type

Regional

* Required

Create API

- Select Action dropdown and choose Create Method and then select ANY
- Configure the method as below screenshot and choose correct lambda option

/ - ANY - Setup

Choose the integration point for your new method.

Integration type

☒ Lambda Function
 ☐ HTTP
 ☐ Mock
 ☐ AWS Service
 ☐ VPC Link

Use Lambda Proxy integration

☐

Lambda Region

us-east-1

Lambda Function

Use Default Timeout

☒

Save

- Use the Test button to test the API
- Select Stages from left navigation rail and select create, provide stage name and deployment type, and click on create
- Navigate to Logs and Enable cloud watch logs and select save changes

S3797892 | S3804879

- There will be a Invoke URL that will be displayed which can be used to invoke the designated Lambda function. Place this URL in locations.html at
lambda_request_handler Stage Editor

Delete Stage

Configure Tags

Invoke URL: https://bw7plu7pni.execute-api.us-east-1.amazonaws.com/lambda_request_handler

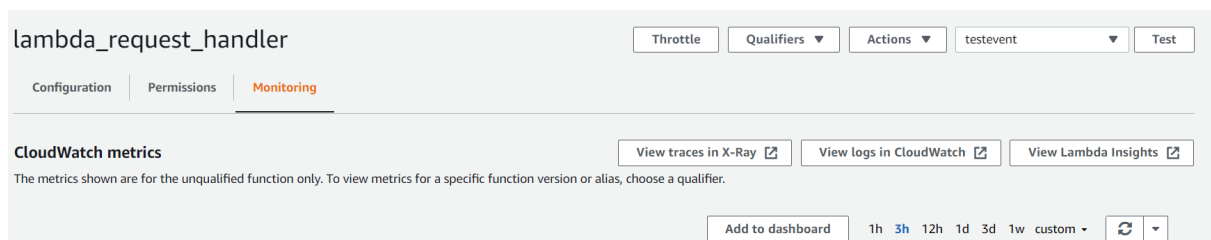
```
type: 'POST',  
url: "[URL]",  
data: JSON.stringify([ { state: state } ]),
```

- Expand the navigation menu on the left to view the individual URL's for specific REST URLs

CloudWatch configuration:

CloudWatch can be used to trace the logs from the lambda function that was crated earlier. This can help in troubleshooting the function when required.

- To view the logs, select monitoring from the lambda function dashboard that was created earlier



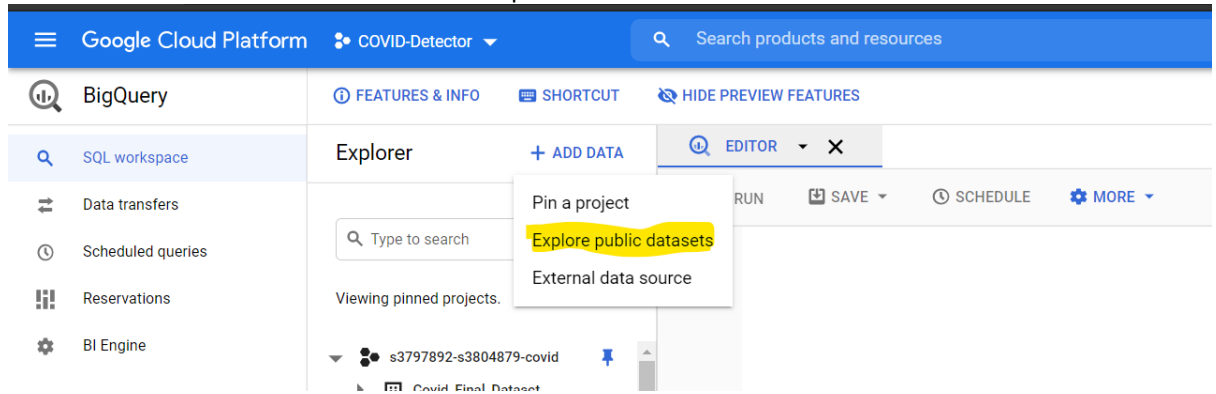
- Select the view logs in CloudWatch to redirect to the CloudWatch console
- The logs are displayed as per the timeline in log streams section.
- Select an item from the list to see the details with its timestamp.

BigQuery Configuration:

BigQuery from Google is serverless technology that is highly scalable service to manage any massive dataset and extract meaningful insights from the data. Besides, it also provides a large collection of public datasets with live data to explore

- To configure the BigQuery navigate to
https://console.cloud.google.com/bigquery?_ga=2.188234792.1472939372.1611970822-1702988852.1609721798&project=s3797892-s3804879-covid
- Create a dataset named Covid_Final_Dataset
- Select add data and then explore public datasets

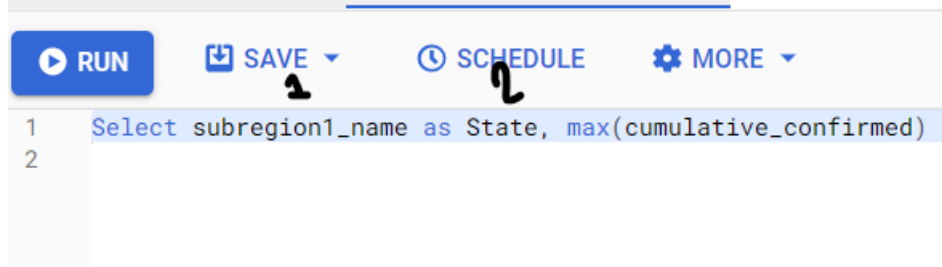
S3797892 | S3804879



- There will be a new project appearing in the workspace with name bigquery-public-data and select the *covid19_open_data*
- Select create a query and paste the following query

```
Select subregion1_name as State, max(cumulative_confirmed) as Confirmed_Cases from
bigquery-public-
data.covid19_open_data.covid19_open_data where country_code = 'US' and subregion1_n
ame != '' group by subregion1_name;
```

- Save the query and select schedule and create new schedule query



- Enter the name for schedule and repeats daily → 'start now' → 'End never' → Select destination table as *Covid-Detector* and *Covid_Final_Dataset*.
- Enter a valid table name → choose 'append to table' → click on 'schedule' button. This will create scheduled job to pull this specific data from public dataset to project's table.

Data studio Configuration:

Data studio offers a powerful cloud platform to visualise data as charts and tables with various connectors available out of the box and ability to create custom connectors.

- To create a report on Data studio, select Blank report from add data to report section. Select BigQuery connect to data.
- Select My projects → COVID- Detector → Covid_Final_Dataset → Now select the table that was created earlier to store the scheduled query data.

← Add data to report



Make your BigQuery reports load even faster with BigQuery BI Engine. [Learn More](#)



BigQuery

By Google

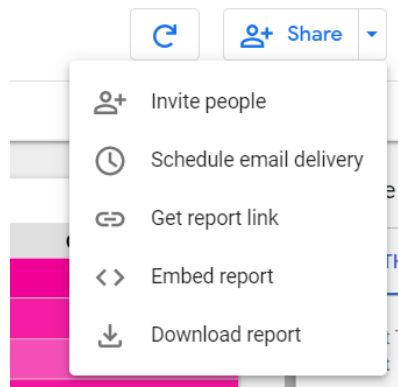
BigQuery is Google's fully managed, petabyte scale, low-cost analytics data warehouse. BigQuery charges for data. Those queries are charged to the credit card of the billing project.

[LEARN MORE](#)

[REPORT AN ISSUE](#)

MY PROJECTS	Project
SHARED PROJECTS	as1-task2
CUSTOM QUERY	COVID-Detector
PUBLIC DATASETS	My First Project
	My Google Assistant
	s3804879-my-client
	s3804879-myapi
	tute1cloud

- Select Add chart and insert table. Select state as Dimensions (Column) and Total_Tested, Confirmed_cases and Confirmed_Deaths as Metrics (Row)
- Now select dropdown from share and select Embed report



- Copy the Embed code from text field and place it in stats.html of the project as

```
<iframe class="studio" src="[EMBED CODE]"
frameborder="0" style="border:0" allowfullscreen></iframe>
```

Google Maps Configuration:

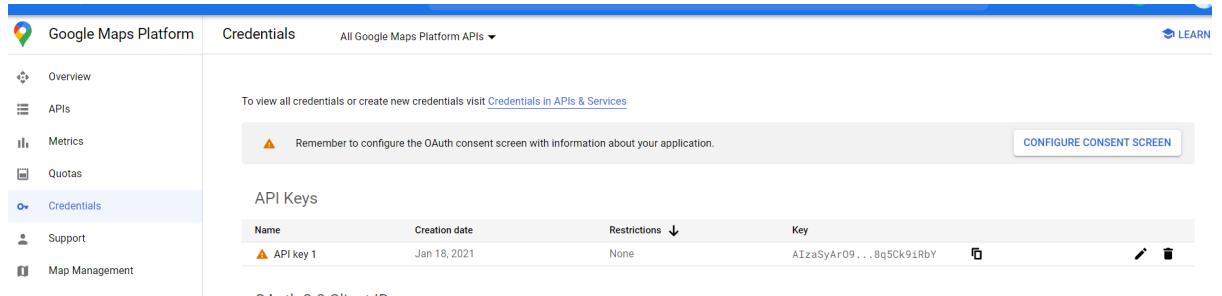
Google provides developers access to its Maps as a cloud product which can be used in project to display the testing centre locations to users.

- To utilise the google maps in the project the developer API keys needs to be created which be used to invoke the service via any of the available SDK. JS client in this case

Covid-19 symptom checker & test centre locator

S3797892 | S3804879

- Navigate to <https://console.cloud.google.com/google/maps-apis/overview?folder=&organizationId=&project=s3797892-maps-api>
- Select pick API's and select enable
Geocoding API
Maps JavaScript API
- Select Credentials and create a new API key and ensure to copy the key as it is required to access the service



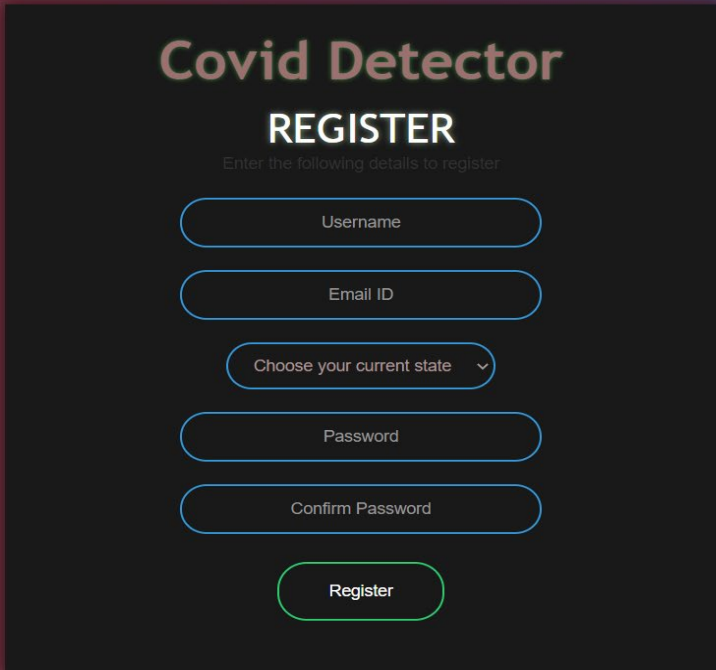
- Place the key at

```
<script async defer  
src="https://maps.googleapis.com/maps/api/js?key=[KEY]&callback=myMap"></script>
```

User Manual

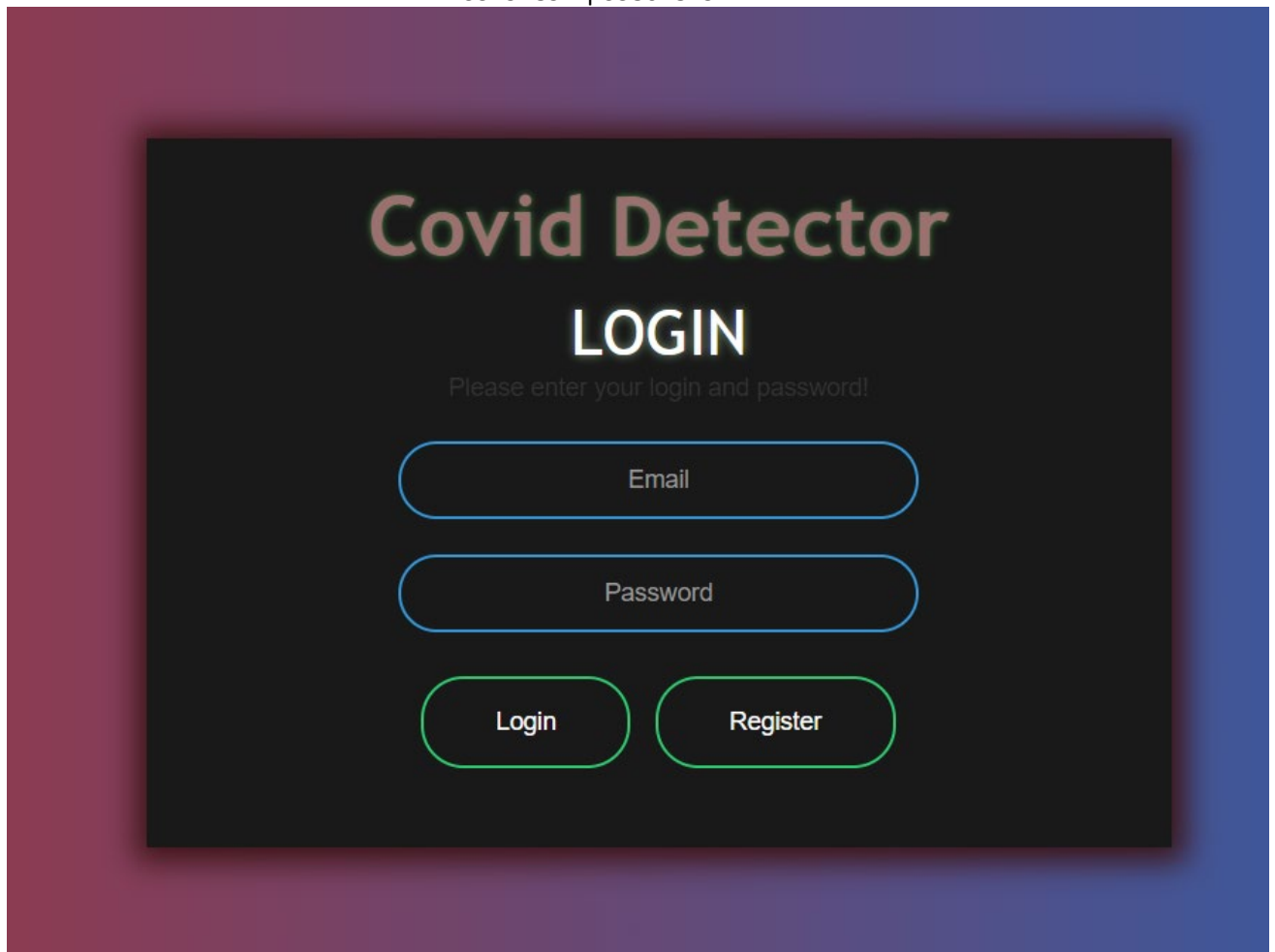
Registration

- Visit the following link: <https://s3797892-s3804879-covid.ts.r.appspot.com/>
- Once the page is loaded, click on the “Register” button. It will then redirect to the registration page.
- Enter the details as requested.
- Click on Register Button, it should redirect to the login page.
- Enter the Credentials to login.



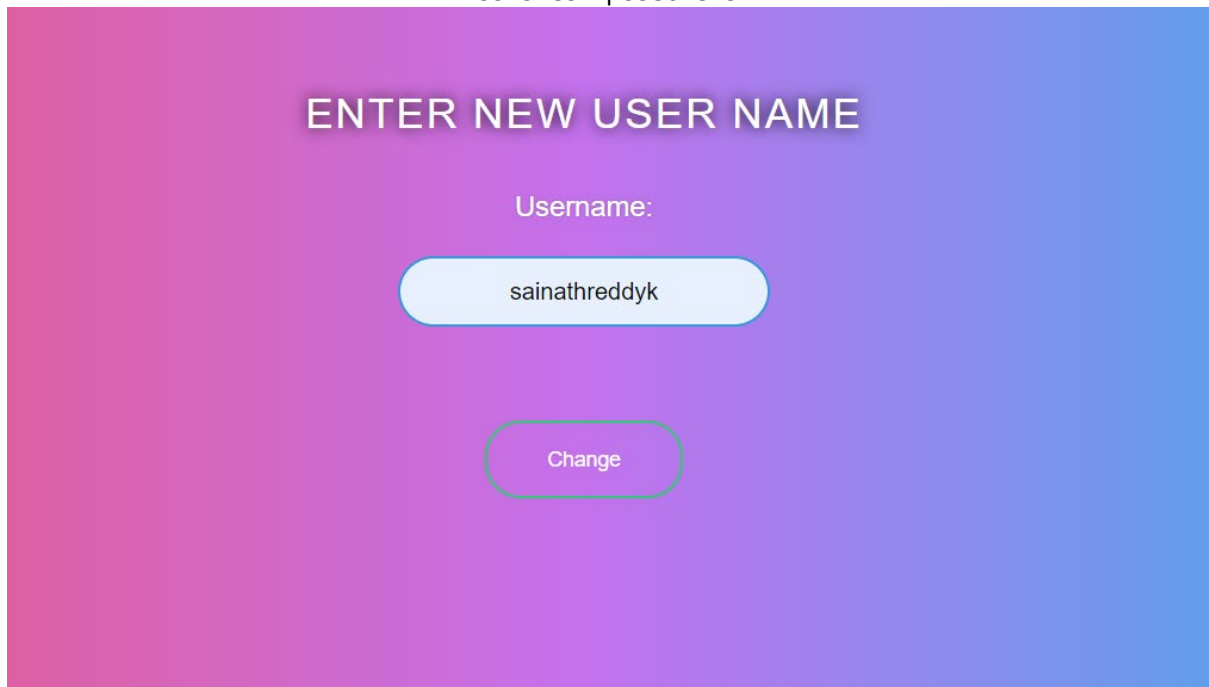
Login

- Visit the following link: <https://s3797892-s3804879-covid.ts.r.appspot.com/>
- Once the page is loaded, enter the requested credentials.
- Click on “Login” button.
- Upon successful login, user will be redirected to the main page.



Change Username

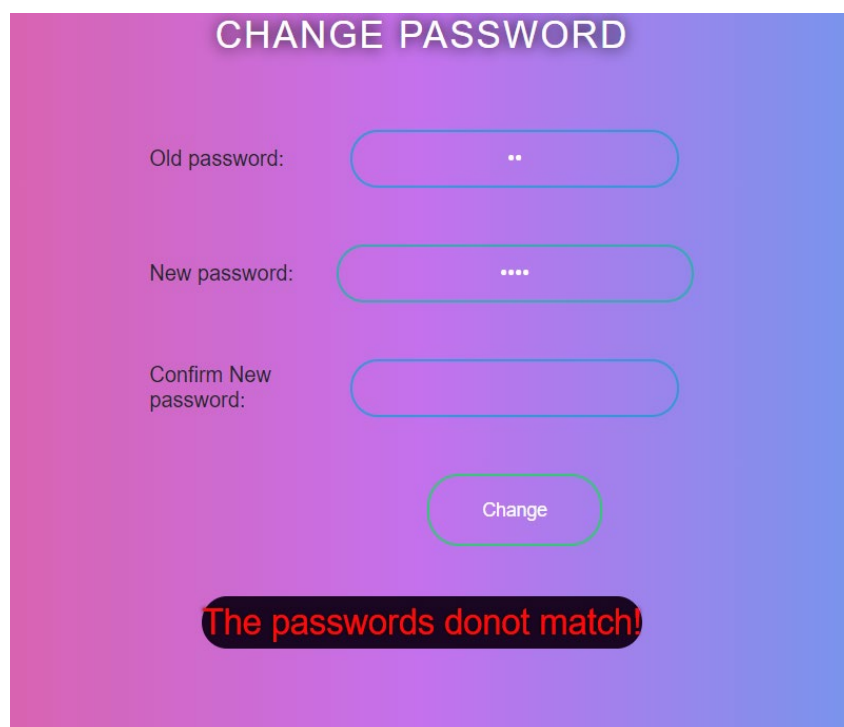
- Follow the Login Workflow.
- On the main page, click on the “Change Name” button in the navigation bar. It will redirect to the name page.
- Enter the new Username and click on “Change” button.
- The main page should be loaded with the new username on the top-left corner of the page.



A screenshot of a web form titled "ENTER NEW USER NAME" on a pink-to-blue gradient background. The form includes a label "Username:" followed by a light blue rounded rectangular input field containing the text "sainathreddyk". Below the input field is a rounded rectangular button with a green border and the text "Change".

Change Password

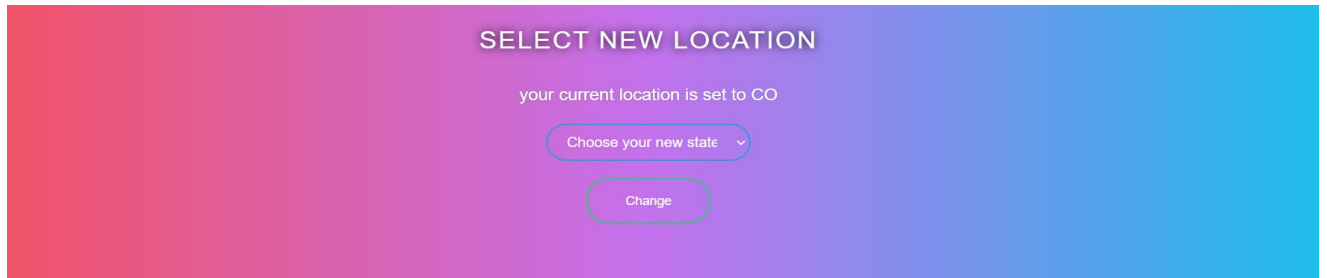
- Follow the Login Workflow.
- On the main page, click on the "Change Password" button in the navigation bar. It will redirect to the password page.
- Enter the new and old passwords and click on "Change" button.
- The login page should be loaded, enter the email id and new password and click on "Login" button.
- The main page should be loaded.



A screenshot of a web form titled "CHANGE PASSWORD" on a pink-to-blue gradient background. The form contains three labels with corresponding input fields: "Old password:" with a light blue rounded rectangular field containing two dots; "New password:" with a light blue rounded rectangular field containing four dots; and "Confirm New password:" with an empty light blue rounded rectangular field. Below these fields is a rounded rectangular button with a green border and the text "Change". At the bottom of the form, a black rounded rectangular box contains the text "The passwords donot match!" in red.

Update Location

- Follow the Login Workflow.
- On the main page, click on the “Update Location” button in the navigation bar. It will redirect to the state page.
- Choose the new state from the dropdown provided and click on the “Change” button.
- The main page should be loaded with the updated location.

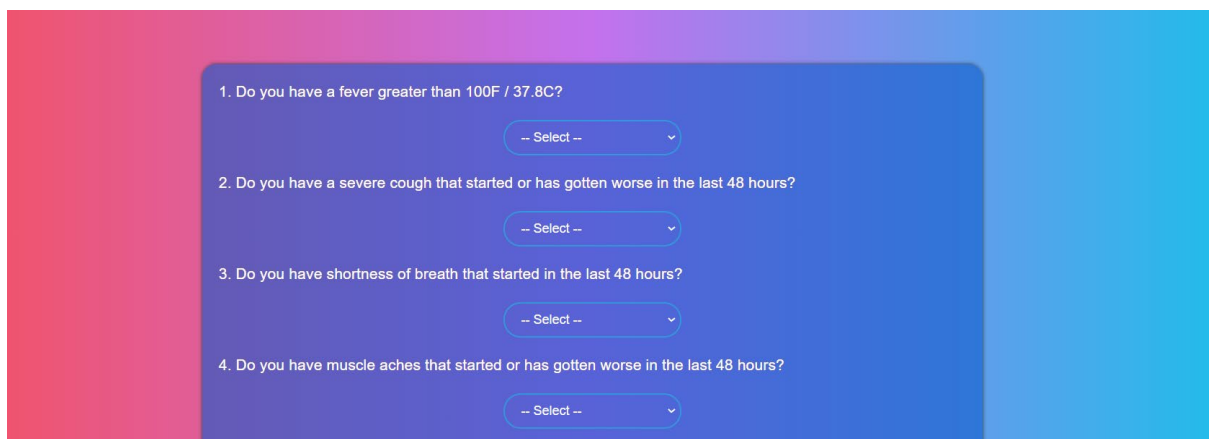


Logout

- On the main page click on “Logout” button.
- The user should be redirected to the login page after successful session termination.

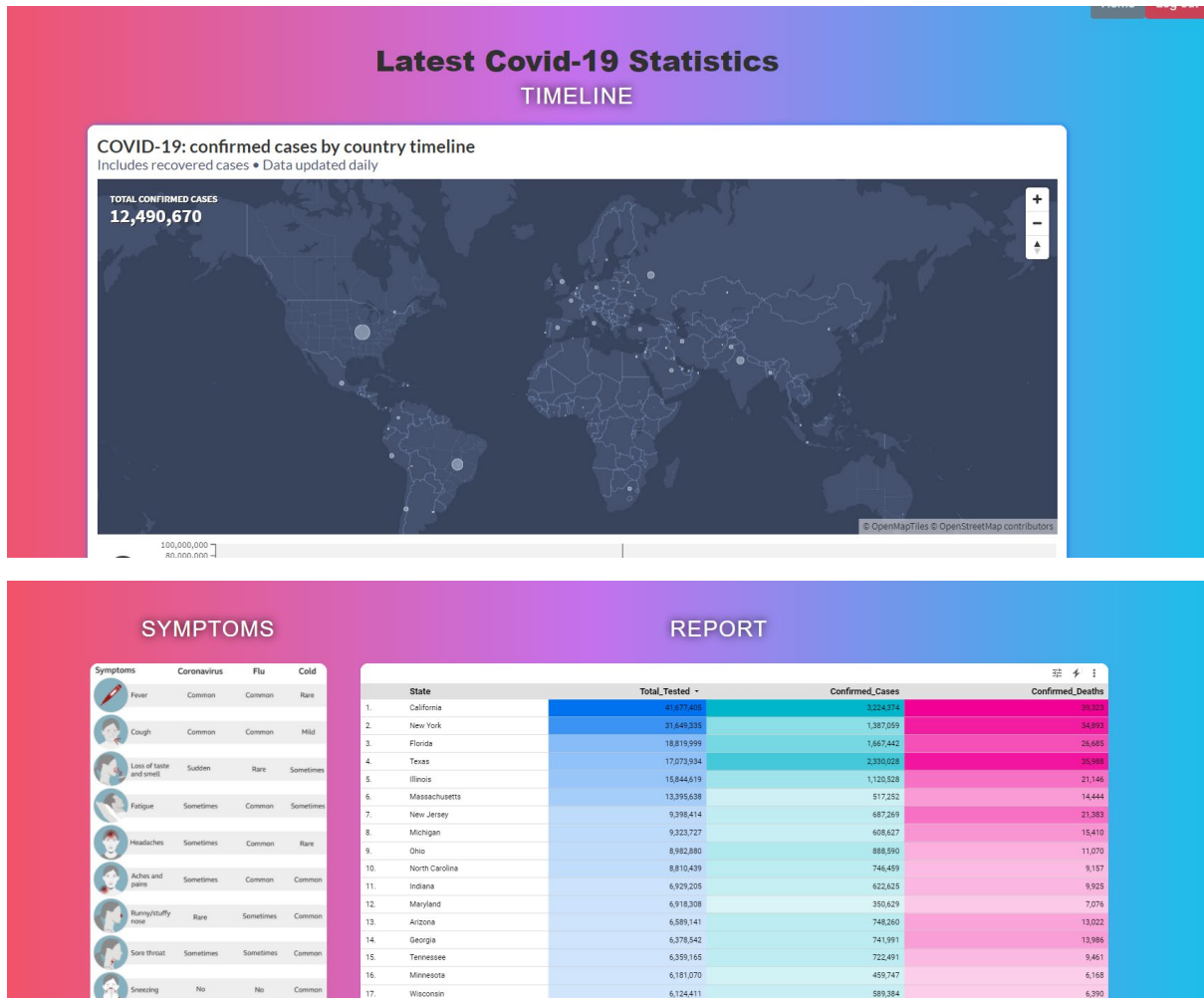
Check Symptoms

- Follow the Login Workflow.
- On the main page, click on the “Check Symptoms” button present on the centre of the page.
- A form should be visible to the user. Answer all the requested questions as per the knowledge of the user.
- Click the “Submit” button to submit all the answers.
 - If any of the submitted answers is “Yes”, then the page with the details regarding the test centres is displayed to the user.
 - If all the submitted answers are “No”, then a badge will be displayed to the user.
 - If any of the questions are unanswered then the user remains in the same page until all the questions are answered.



View Statistics

- Follow the Login Workflow.
- On the main page, click on the “View Statistics” button present on the centre of the page.
- The stats page will be displayed to the user showing all the live statistics related to the COVID-9 pandemic.



References

- [1] “COVID-19 Symptom Checker” - <https://covid19symptomchecker.spectrumhealth.org/home>
- [2] “COVID-19 Testing Locations” - <https://covid-19-apis.postman.com/covid-19-testing-locations/>