A Project/Dissertation ETE Report on

Covid-19 Status and Vaccine Slot Availability Finder

Submitted in partial fulfilment of the requirement for the award of the degree of

B.Tech (Computer Science and Engineering)

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

Mr. Padmanabhan P. (Assistant Professor)

Submitted By: - **1) Vikas Kumar Mishra (18SCSE1010507)**

(18021011735)

2) Nishant Kumar Singh (18SCSE1010543)

(18021011769)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA

MONTH, YEAR
AUG-DEC 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE’S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **“Covid-19 Status and Vaccine Slot Availability Finder”** in partial fulfillment of the requirements for the award of the B. Tech(CSE) -submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of Name… Designation, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Vikas Kumar Mishra(18SCSE1010507)
Nishant Kumar Singh(18SCSE1010543)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Padmanabhan P.
Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Vikas Kumar Mishra(18SCSE1010507) and Nishant Kumar Singh(18SCSE1010543) has been held on and his/her work is recommended for the award of B. Tech (CSE)

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December 2021
Place: Greater Noida

ACKNOWLEDGEMENT

In the accomplishment of completion of my project on **Covid-19 Status and Vaccine Slot Availability Finder** I would like to convey my special gratitude to **Mr. Padmanabhan P**. of Computer Science Department.
Your valuable guidance and suggestions helped me in various phases of the completion of this project. I will always be thankful to you in this regard.
I am ensuring that this project was finished by me and not copied.

Vikas Kumar Mishra (18SCSE1010507)
Nishant Kumar Singh (18SCSE1010543)

ABSTRACT

With its alarming surge of affected cases throughout the world, lockdown, and the World Health Organization's declaration, the Novel Coronavirus, Coronavirus, 19 has been declared pandemic throughout the world by the WHO. People are found to be able to prevent the spread of disease only by raising awareness social distancing, mask use, etc. They don't know which areas of popular culture are more affected by this pandemic. After the vaccine arrives, everyone can take it very fast. In front of the government, the biggest problem is how to manage this vaccine process. It is not possible to get vaccinated completely in 10 to 15 days. There is a panic situation between the people of a country. We make software to solve these problems. This project developed for tracking COVID-19 vaccine slots available in your pin-code and district for 18+ age category. Once available, it will send email notifications at periodic interval

to subscribers until slots are available. A android-based system is proposed for this business management page such as Automobile Service Center or Beauty Parlor / Saloons etc. to control the spread of social gatherings. The proposed system has features such as a single application for service providers and customers. Verification of real customers and service providers will be done through OTP and store-based photographs respectively. The owner can keep track of the details of his employees by registering and can track regular customers. It helps to book easily and cancel appointments. The customer can view non-working days with the event calendar and the services provided and their cost, time required etc. The system also offers the option to handle customer payments, invoice production, analytics reports help maintain the website and give the customer a reminder of appointments. So the system will do the right planning and reduce the effort and time of both the client and the owner.
Through this application we can view the details of covid patients and dead all over the world. It's difficult to schedule an appointment for Covid-Vaccine given that there are only a limited number of slots available each day, so this script automates the whole process by checking the availability of slots every 3 seconds as well as booking it once it becomes available. This script needs only one configuration - entering your preferences once **(pin code, district, centre preference, slot timing, etc.)** and then it takes care of everything else.

Keywords
Android, Covid19 Cases & Vaccination Statistics, Check Vaccine Availability, Firebase Cloud Storage, Authentication, Identification, etc.

CHAPTER 1

INTRODUCTION

1.1 Overview

The contagious coronavirus, or more technically known as COVID-19, has spread all over the world and is listed as a pandemic by the World Health Organization. It started surfacing in China in November 2019 and has been on the rise in all major parts of the world. On 27th July, 2020, more than 16.52M cases of COVID-19 were reported in about 187 countries and territories. On 30th January, 2020, the first case of coronavirus pandemic in India was reported, and the number of cases in India has now reached more than 1.48M .

Most people infected from COVID-19 experience mild to moderate respiratory illness. If a person is tested positive for coronavirus, every individual who has come in contact with the infected individual is advised to go for self-quarantine for two weeks, so that the infection chain can be broken and the disease does not spread further. At present, there is no specific treatment or vaccine available for COVID-19. Many countries are trying to develop contact-tracing techniques through which they can trace the person suspected of the infection. For example, in South Korea, the government is maintaining a database of known patients along with their details such as their age, gender, occupation, and so on. In Israel, the government has been allowed to track the mobile-phone data of people suspected with the infection. Singapore developed a mobile-based application which shows the number of cases of covid and also helps in finding covid vaccination slots.

We have developed a similar type of application named Covid-19 Status and Vaccine Slot Availability Finder.
This Application helps the user to keep a track on the covid-19 cases status in India. It shows the number of cases of covid positive patients and recovered cases and as well as the number of death cases each day.

This is all done in the Covid-19 Statistics section of the application. The Statistics section allows you to stay up to date regarding the number of cases, both locally and nationally. The accurate numbers can help you assess your risk further. Additionally, the availability of official updates prevents rumours and misinformation from spreading.

The Application also helps the users to find vaccine slots so that they can get vaccinated by finding vaccination centres nearby them and the slots available in the centres . Both the Statistics and the Vaccination Slot finder gets updated on a daily basis to provide up to date data to the users.

1.2 Features

- View Daily Covid19 Cases & Vaccination Statistics
- View State Wise Daily and Total Covid 19 Statistics
- Check Vaccine Availability in your area using your area Pin code
- Uses Firebase Cloud to store Data.
- Data Sync across different platforms and devices.

CHAPTER 2

LITERATURE REVIEW

2.1 Flutter

In this app we have using the Android studio with flutter mode. Basically, flutter is use to make the apps which they run in both of the famous mobile phone processor like Android and iOS. Flutter is a cross-platform UI toolkit designed to allow reuse of code across operating systems such as iOS and Android, while allowing apps to interact directly with basic platform services. The goal is to empower developers to bring the most efficient applications that feel natural to a wide range of platforms, covering the differences where they exist while sharing as much code as possible.

During the upgrade, Flutter applications work with a VM that provides hot reloading of changes without the need for full integration. For download, Flutter applications are directly integrated into the machine code, either in Intel x64 or ARM instructions, or in JavaScript if it directs the web. The framework is an open source, licensed BSD licensed, and has a thriving environment for third-party packages that enhance the functionality of a central library.

2.2 Dart

In this project we uses the Famous languages Dart. It is use to increase the productivity of you app. And it also provide the fast access and run of apps. Strong languages have been created by the translator, without producing a machine language code.

Of course, things eventually got worse. The concept of virtual machine (VM) became popular, which is actually an advanced translator that mimics hardware on software. The virtual machine makes it easy to send language to new hardware platforms. In this case, the input VM language is usually the intermediate language. For example, programming language (similar to Java) is integrated into intermediate language (bytecode) and rendered into VM (JVM).

In addition, there are now timely compilers (JIT). The JIT compiler launches during the process, compiling instantly. Real producers who used during the creation of the system (pre-working time) are now called pre-programmers (AOT).

2.2.1 Google Analytics

For storage we have to use the google analytics or firebase cloud. In this the main role the firebase cloud to store the state wish data into a json file and connected through app. Firebase Analytics is a tool that lets you do just that - it helps us learn how our Android and iOS users interact with our app. From the setup, it will automatically start tracking a set of defined events - which means we can start learning in the first step. And if that is not enough, we might add our own custom events that we can track. All these events then appear with the Firebase Dashboard within the Firebase Console - our central access point for statistics reports and other firebase services.
Once we have tracked and analyzed this data, we can make decisions about future changes in our app to better serve users. And if that were not enough, Firebase Analytics Incorporates Firebase Crash Reporting to create an audience of users who have experienced a follow-up crash, Firebase Notifications to send alerts to the audience and track events based on notification interaction, Firebase Remote Config to change the look / feel. and our application behavior based on Audience, Big Query performs improved data analysis on our tracked events and Google Tag Manager in our Firebase Analytics settings remotely from another web application.

2.2.2 Data-Parsing-Script

Most important library is use to forming this app. For using this library we collected the specific type of the data from the whole table. JSON Analysis is a very common function for applications that need to download data online.
Data Parshing is depend these thing:-

Write all JSON transfer code in person automatically perform the process by generating code This guide will focus on how to transfer JSON manually to Dart code, which includes: coding and coding on JSON describing safe model model classes analyzing JSON into a Dart code using a factory builder dealing with invalid / optional values data validation edits back to JSON to decrypt complex / nested JSON data selecting deep values by deep pick package

2.3 Http Protocol-

HTTP is a protocol for fetching resources. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. A complete document is reconstructed from the different sub-documents fetched, for instance, text, layout description, scripts, and more.

2.4 Shared Preferences-

Shared Preferences object points to a file containing key-value pairs and provides simple methods to read and write them. Each Shared Preferences file is managed by the framework and can be private or shared.

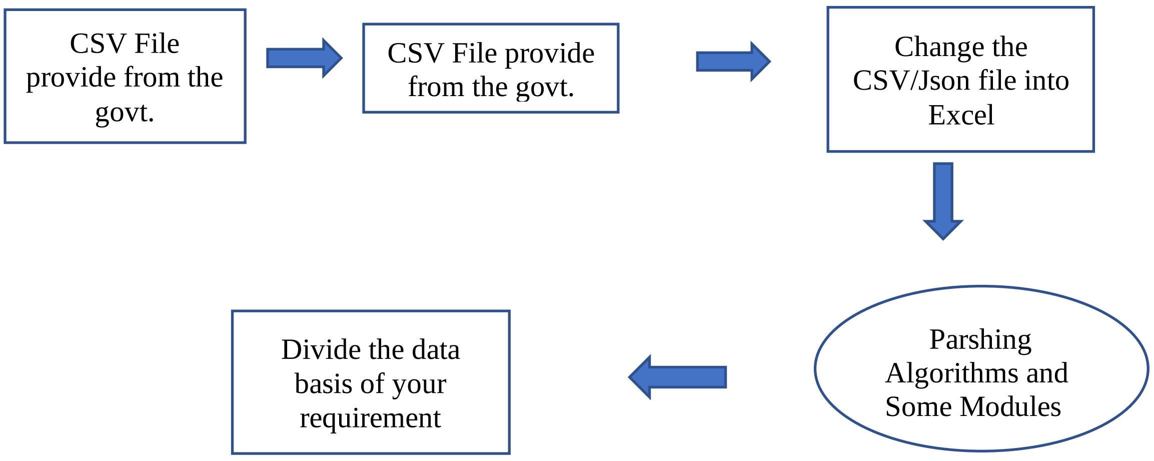
Data-Parsing-Script –
“Data parser” is a generic parsing script that handles a wide range of data formats: is files (as downloaded from the Web Of Science), Factiva datasets, PubMed, RIS files, batches of simple text files or any file formatted in csv format .

CHAPTER 3

SYSTEM DESIGN

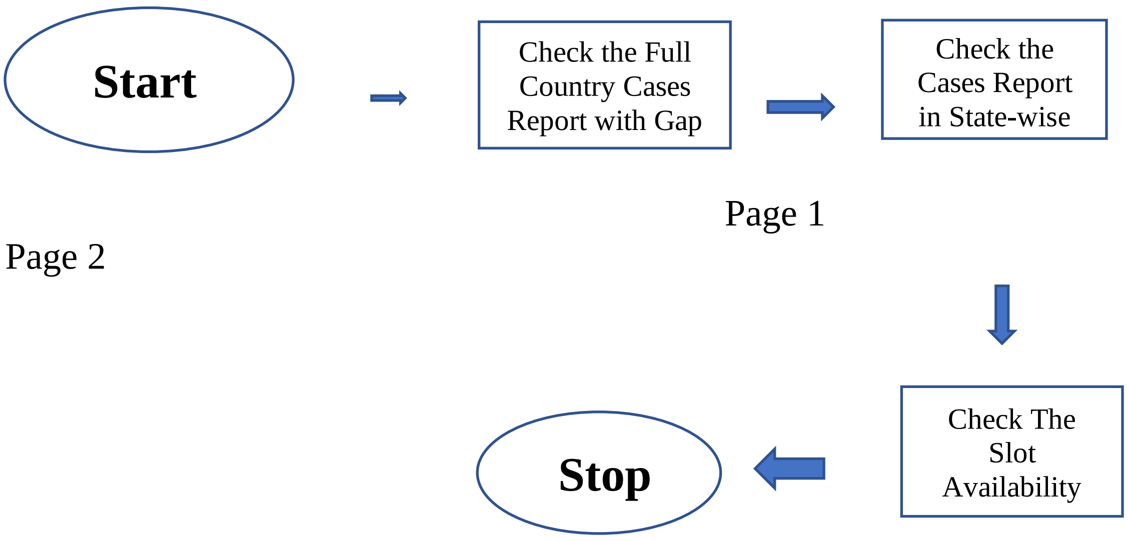
3.1 SYSTEM ARCHITECTURE

3.1.1 Data Parsing process:



CSV File provide from the govt.
CSV File provide from the govt.
Change the CSV/Json file into Excel
Parshing Algorithms and Some Modules
Divide the data basis of your requirement

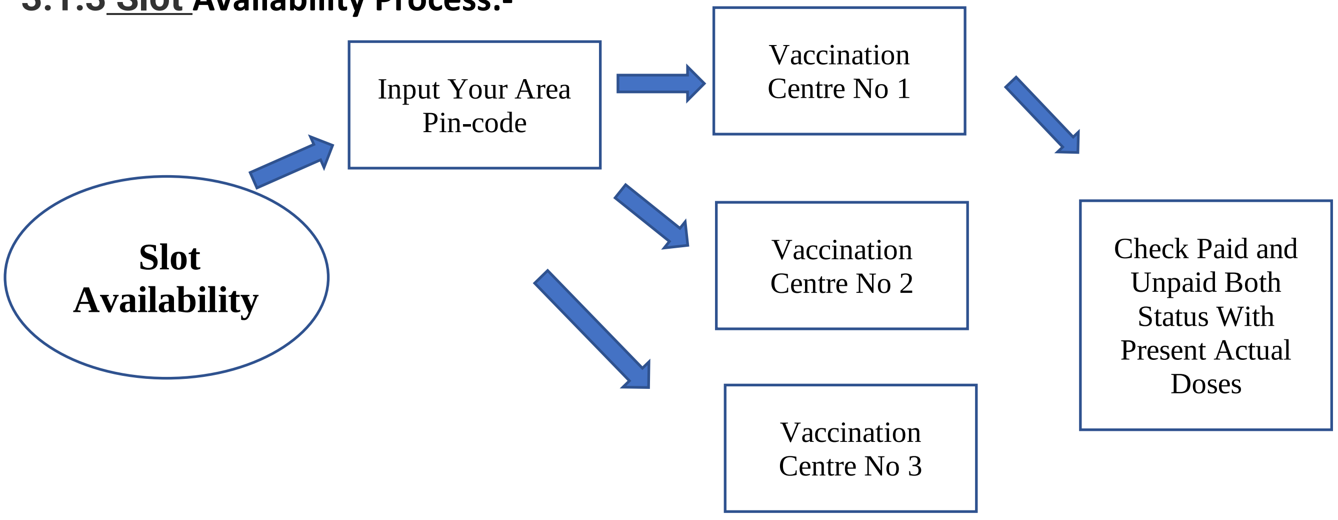
3.1.2 Process Model App



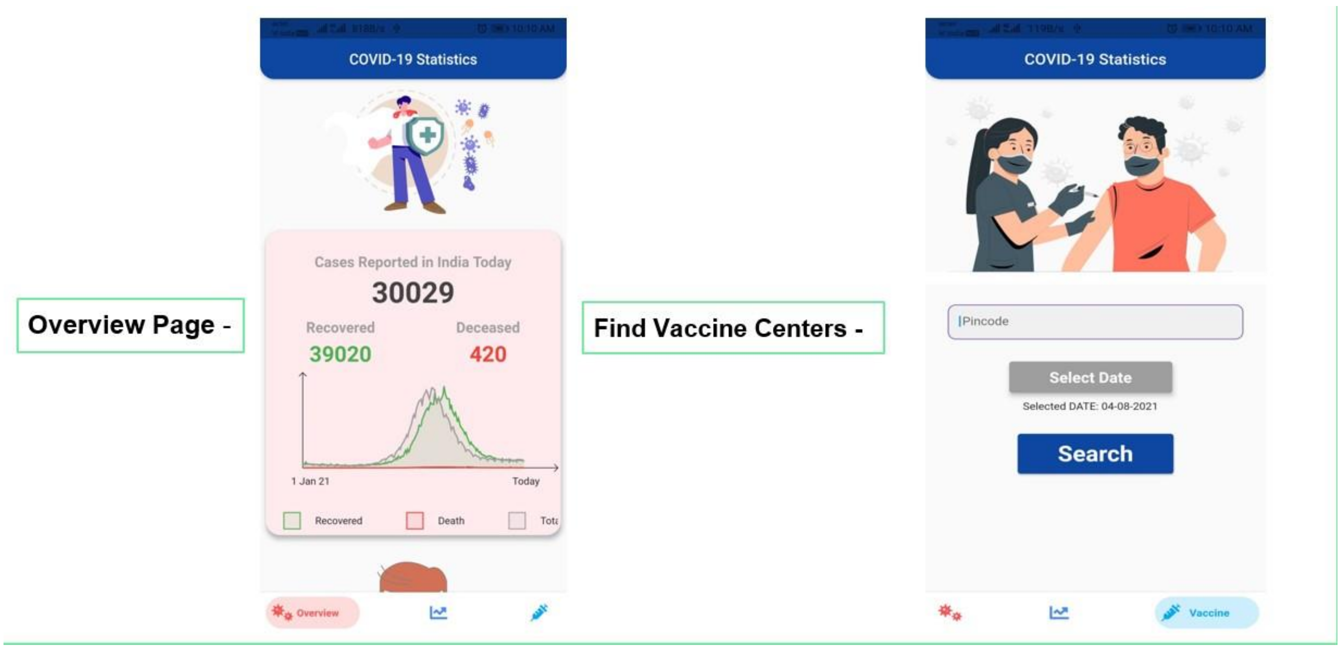
Start
Check the Full Country Cases Report with Gap
Check the Cases Report in State-wise
Page 1
Page 2
Check The Slot Availability
Stop

Page 3

3.1.3 Slot Availability Process:-

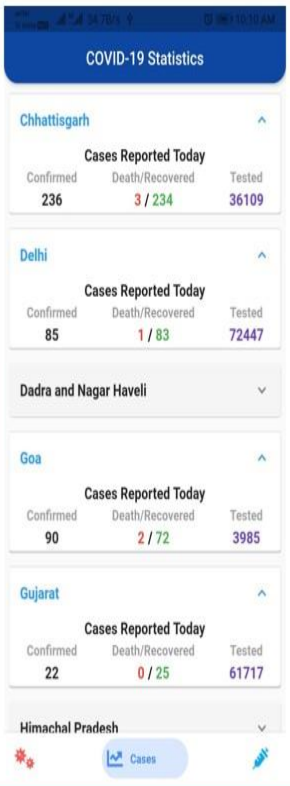


3.2 CONTEXT DIAGRAM



3.2.1 Overview and Slot Checking

Cases State wise -



COVID-19 Statistics

Chhattisgarh

Cases Reported Today

Confirmed	Death/Recovered	Tested
236	3 / 234	36109

Delhi

Cases Reported Today

Confirmed	Death/Recovered	Tested
85	1 / 83	72447

Dadra and Nagar Haveli

Goa

Cases Reported Today

Confirmed	Death/Recovered	Tested
90	2 / 72	3985

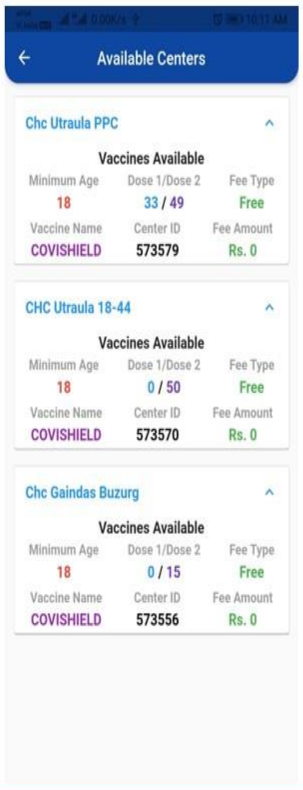
Gujarat

Cases Reported Today

Confirmed	Death/Recovered	Tested
22	0 / 25	61717

Himachal Pradesh

Available Centers -



Available Centers

Chc Utraula PPC

Vaccines Available

Minimum Age	Dose 1/Dose 2	Fee Type
18	33 / 49	Free
Vaccine Name	**Center ID**	**Fee Amount**
COVISHIELD	573579	Rs. 0

CHC Utraula 18-44

Vaccines Available

Minimum Age	Dose 1/Dose 2	Fee Type
18	0 / 50	Free
Vaccine Name	**Center ID**	**Fee Amount**
COVISHIELD	573570	Rs. 0

Chc Gaindas Buzurg

Vaccines Available

Minimum Age	Dose 1/Dose 2	Fee Type
18	0 / 15	Free
Vaccine Name	**Center ID**	**Fee Amount**
COVISHIELD	573556	Rs. 0

3.2.2 Cases Statistics with State wise

CHAPTER 4

FUNCTION MODULE

4.1 Data availability

4.1.1 Basic data

The data used for this project is taken from the following public sources available in the JSON format and commas comma sequences (csv) respectively:

1. As of 13 August, our API repository and api.covid19india.org have been withdrawn. Redirecting api.covid19india.org to data.covid19india.org

Modeling projections for cases and deaths

Modeling projections for cases and deaths
For the mathematical model, allow three points P1 = (x1, y1), P2 = (x2, y2), and P3 = (x3, y3) so that y1 corresponds to the number of events or deaths 30 days before the day. current date, x1 = 0, y2 corresponds to the number of events or deaths 15 days prior to today, x2 = 15, y3 corresponds to the number of events or deaths today, and x1 = 30. Based on the quadratic equation defined as f (x) = ax2 + bx + c,, let the equation system presented in (1), (2), and (3).
y1=ax21+bx1+c(1)y1=ax12+bx1+c (1)
y2=ax22+bx2+c(2)y2=ax22+bx2+c (2)
y3=ax23+bx+c(3)y3=ax32+bx+c (3)
Since x_1 = 0 from (1):
y1=c(4)y1=c (4)
Subtracting (2) from (4):
y1−y2=−ax22−bx2(5)y1−y2=−ax22−bx2 (5)
Isolating the variable *b* from (5):
b=y1−y2+ax22−x2(6)b=y1−y2+ax22−x2 (6)
Subtracting (2) from (3):
y3−y2=a(x23−x22)+b(x3−x2)(7)y3−y2=a(x32−x22)+b(x3−x2) (7)
Replacing *b* in (7):
y3−y2=a(x23−x22)+y1−y2+ax22−x2(x3−x2)y3−y2=a(x23−x22)+(y1−y2)(x3−x2)−x2−ax2(x3−x2)y3−y2=a((x23−x22)−x2(x3−x2))−(y1−y2)(x3−x2)x2(8)y3−y2=a(x32−x22)+y1−y2+ax22−x2(x3−x2)y3−y2=a(x32−x22)+(y1−y2)(x3−x2)−x2−ax2(x3−x2)y3−y2=a((x32−x22)−x2(x3−x2))−(y1−y2)(x3−x2)x2 (8)
Isolating the variable *a* from (8):
a=y3−y2((x23−x22)−x2(x3−x2))+(y1−y2)(x3−x2)x2((x23−x22)−x2(x3−x2))(9)
a=y3−y2((x32−x22)−x2(x3−x2))+(y1−y2)(x3−x2)x2((x32−x22)−x2(x3−x2))

(9)

Finally, isolating the variable *c* from (4):

c=y1(10)c=y1 (10)

Therefore, based on the a, b, and c used in quadratic calculations, the estimated cases and deaths are calculated. In addition, calculated quadratic calculations will be adjusted when new information is reported by WHO. In addition, the COVID-19 Dashboard provides a 90-day guess; however, in some countries, the quadratic equation can rise to a higher level; in these cases, the algorithms will stop counting and thus the estimated days will be less than 90 days.

Therefore, the COVID-19 Dashboard allows for the analysis of confirmed cases, death rates, and death rates as well as speculative cases and the submission of death information: a) geographical charts to understand the spread of the disease in the country, b) bar charts , c) columns of charts to indicate the occurrence of the disease over time, and d) line charts to measure disease behavior.

4.2 Files are available

• Combined sheets provide integrated data at regional / regional level in csv format.

• V4 json end points. These are json apis used by the website to display all the statistics on the site. This can be used by engineers and analysts with knowledge of json separation (recommended method). All of our v4 conclusions have been improved and usable as this gives the impression of frontend Documents the same.

• The latest data from google sheet (10-20 minutes delay) is available in the latest archive. These are located under the green files section below. (Not recommended as the number of files is large and no additional information is available on these compared to the conclusions mentioned above.)

Status	**Link to API**	**Description**
💚	https://data.covid19india.org/v4/min/timeseries.min.json	Daily numbers across C,R,D and Tested per state (historical

Status	Link to API	Description
		data).
💚	https://data.covid19india.org/v4/min/data.min.json	Current day numbers across districts and states.

vaccine_doses_administered_statewise	http://data.covid19india.org/csv/latest/vaccine_doses_statewise_v2.csv

CHAPTER 5

OBJECTIVE AND METHODOLOGY

5.1 PROCESS

A android-based system is proposed for this business management page such as Automobile Service Center or Beauty Parlor / Saloons etc. to control the spread of social gatherings. The proposed system has features such as a single application for service providers and customers. Verification of real customers and service providers will be done through OTP and store-based photographs respectively. The owner can keep track of the details of his employees by registering and can track regular customers. It helps to book easily and cancel appointments. The customer can view non-working days with the event calendar and the services provided and their cost, time required etc. The system also offers the option to handle customer payments, invoice production, analytics reports help maintain the website and give the customer a reminder of appointments. So the system will do the right planning and reduce the effort and time of both the client and the owner.

India's vaccination campaign is about to fall as it battles the second wave of the COVID-19 epidemic. Although the government has finally agreed to increase the production capacity of the vaccine in the country. The Atmanirbhar Bharat 3.0 Mission has taken a significant step forward to promote COVID's indigenous policies by prohibiting COVID Suraksha from supporting the development and production of COVID's indigenous policies; three public sector functions (PSUs) have been established to increase policy capacity. As of 23 April 2021 approximately 13,41,80,854 of which 11,44,66,357 Volume 1 and 1,97,14,497 volume 2 vaccines have been given to the public. There are 68,395 vaccination centers out of which 61,836 are state and 6,559 are independent. There were 11,90,92,552 registrars and 1,38,82,605 were online, 8,10,50,255 logged in and 2,41,59,692 were FLW and HCW (senior staff and health workers). a total of 12,18,49,147 vaccine doses were given to Covishield and 1,23,31,706 were Covaxin. Following the same trends (1.38 crore doses per week) will take about 100 weeks to vaccinate nationwide in India, meaning that everyone in India will receive at least one dose of the vaccine by December 15th 2022. According to Johns Hopkins University of Medicine Coronavirus Resource Center, only about 1% of Indians in India are completely vaccinated. The COVID-19 vaccine is made in India by two companies. The Serum Institute of India in Pune is partnering with AstraZeneca in Covishield, and Bharat Biotech in Hyderabad has obtained a production license from the Indian Medical Research Council (ICMR) to produce Covaxin. According to Down To Earth, India is strong enough, with a panel of seven PSUs capable of producing goals. However, three production licenses from these PSUs - Central Research Center, Kasauli; BCG Policy Laboratory, Guindy; and the Pasteur Institute of India, Conoor - were withdrawn in 2008 because they did not comply with the good production procedures set out in the regulations (DownToEarth, 2021). Bharat Biotech also received funding from the Union government to develop its facilities. By May-June 2021, both measures are expected to double the current Covaxin production capacity, and by July-August 2021, it will have increased almost six to seven times. Vaccine production is expected to increase from one dose of crore per month in April 2021 to approximately seven doses of crore in July-August 2021, and approximately ten doses by September 2021.

Table 1. COVID-19 Specific Characteristics of Application, N=14, N (%)

Symptom management	4 (28%)
Symptom assessment	8 (57%)
Resource information in app	
Testing centres	6 (42%)
Preventative measures	5 (35%)
Regional/federal guidelines	2 (14%)
Those at higher risk	2 (14%)
Physical distancing	3 (21%)
Source of information supporting app	
Not specified	2 (14%)
Research evidence	0 (0%)
Professional experience	2 (14%)
Personal experiences or stories	0 (0%)
National guidelines	2 (14%)
Coronavirus tracking feature	4 (28%)
Live chat room	3 (21%)
Training resources for clinicians	2 (14%)
Frequently Asked Questions (FAQ) forum	**5 (35%)**

5.2 SUGGESTIONS

On the other hand, as the world struggles with the COVID-19 virus, its proliferation and flexibility lead to new challenges for governments and research communities. In this regard, the number of patients worldwide is growing by the

second wave of the COVID-19 outbreak and many countries are re-using closures and curfews to prevent infection. In addition, non-compliance with the rules of social isolation, security and privacy concerns, high level of anonymous cases, etc., suggests that COVID19 mobile applications should be integrated with new features, not limited to COVID-19 management, but also in providing information on health services such as diagnosis, consultation, treatment, procedures, procedures, and more. These apps can be very effective in raising awareness about prevention strategies such as social isolation, hand washing, and keeping information related to health issues and tracking contacts.
It is worth mentioning that most of the studies focused on the goals and approaches to improving mobile applications, quality, and technological advances were made in the early days of the COVID-19 epidemic. With the rapid development in the context of the epidemic, with new signs and cases, as well as new technologies emerging, there is a need for regular global updates of mobile applications. Although there are studies that focus on feature analysis and performance, their analysis is limited to common features of applications such as ease of use and ease of use, but did not include performance with specific features of COVID-19. Generally, studies have only examined mobile applications used to detect COVID-19 outbreaks infrequently.

COVID-19 brings new challenges, making it essential to review and evaluate these mobile applications so that these gaps can be closed. In this regard, the findings of this paper may have a positive impact on medical professionals, software developers, social organizations, science centres, and technology organizations. App developers can benefit from a comprehensive review of COVID-19 mobile applications, as they will be able to identify some of the barriers and utilize various functions and technologies to improve future applications. Clinicians can also review a variety of applications, and propose more efficient patient programs that can aid patients in improving their health management and adopting COVID-19 reduction strategies thanks to increased awareness of mobile appliances. With reference to a functional application, health care services such as diagnostics, consultation can be managed online, which not only saves time and money, but also helps prevent the spread of COVID-19 by reducing mobility.

CHAPTER 6

CONCLUSION AND FUTURE WORK

SWOT analysis allows us to understand more about this app since we can assess its strengths, weaknesses, opportunities, and threats. We believe that experimentation and creating such an app during a time like this is our greatest asset. The lack of awareness among people, especially in rural areas, and the inability to find solutions for this problem is certainly a drawback.

There is a chance to collaborate amongst newer minds to develop an app that will facilitate the management of vaccine programs; this may provide an opportunity to many new minds, as well as inspire people to head in a better direction. We took the first step toward a better future when we developed Covid 19 Status and Slot Finder.

This study reviewed the best applications used during the outbreak of COVID-19 to provide health care services, contain the spread of the new coronavirus, and facilitate human migration during the return home to Saudi. Arabia, India, Italy, Singapore, United Kingdom, United States of America, and Australia. Analysis pointed out that different programs are designed various activities such as contact tracking, awareness, booking an appointment, online consultation, etc. However, only a few applications such as Arogya Setu as well Path Check has integrated a variety of functions and features
such as self-assessment, consultation, support and access information in a single app, which makes it easy for users access to services. Also, most apps are focused on it tracking a contact, while very few are dedicated to growth to raise awareness and share information about COVID-19, important to combat the spread of COVID-19.
Similarly, most apps rely on GPS as well Bluetooth technology for tracking a contact and other essentials job. There are no identified applications with the built-in community media features. In addition, one of the biggest challenges identified lack of integrated application with many features and functionality analysed in this read. In this sense, users rely on different programs medical care, mobility, diagnosis, follow-up, and awareness, etc. Therefore, an effective solution to solve this problem could be designing and improving integrated cell life application, which allows access to all of these functions. Using a single system can reduce costs and improve health data management, and decision-making

CHAPTER 7

SAMPLE CODE AND OUTPUT

APPENDIX 1: MAIN MODULE CODE

First Page XML:-

```
<manifest
xmlns:android="http
://schemas.android.c
om/apk/res/android"
                     package="com.example.covid19_stats">
                    <application
                        android:label="Covid-19 Stats"
                        android:icon="@mipmap/ic_launcher">
                        <activity
                            android:name=".MainActivity"
                            android:launchMode="singleTop"
                            android:theme="@style/LaunchTheme"

                android:configChanges="orientation|keyboardHidden
                |keyboard|screenSize|smallestScreenSize|locale|layout
                Direction|fontScale|screenLayout|density|uiMode"
                            android:hardwareAccelerated="true"

                android:windowSoftInputMode="adjustResize">
                            <!-- Specifies an Android theme to apply to
                this Activity as soon as
                                 the Android process has started. This
                theme is visible to the user
                                 while the Flutter UI initializes. After that,
                this theme continues
                                 to determine the Window background
                behind the Flutter UI. -->
                            <meta-data

                android:name="io.flutter.embedding.android.Normal
                Theme"
                              android:resource="@style/NormalTheme"
                              />
                            <!-- Displays an Android View that continues
```

```
showing the launch screen
                 Drawable until Flutter paints its first
frame, then this splash
                 screen fades out. A splash screen is useful
to avoid any visual
                 gap between the end of Android's launch
screen and the painting of
                 Flutter's first frame. -->
            <meta-data

android:name="io.flutter.embedding.android.SplashS
creenDrawable"

android:resource="@drawable/launch_background"
              />
            <intent-filter>
                <action
android:name="android.intent.action.MAIN"/>
                <category
android:name="android.intent.category.LAUNCHER
"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate
GeneratedPluginRegistrant.java -->
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
</manifest>
```

Second Page:-

```
<manifest
xmlns:android="http
://schemas.android.c
om/apk/res/android"
    package="com.example.covid19_stats">
   <application
        android:label="Covid-19 Stats"
    android:icon="@mipmap/ic_launcher">
    <activity
      android:name=".MainActivity"
      android:launchMode="singleTop"
      android:theme="@style/LaunchTheme"

android:configChanges="orientation|keyboardHidden
|keyboard|screenSize|smallestScreenSize|locale|layou
tDirection|fontScale|screenLayout|density|uiMode"
      android:hardwareAccelerated="true"

android:windowSoftInputMode="adjustResize">
      <!-- Specifies an Android theme to apply to
this Activity as soon as
          the Android process has started. This
theme is visible to the user
          while the Flutter UI initializes. After that,
this theme continues
          to determine the Window background
behind the Flutter UI. -->
      <meta-data

android:name="io.flutter.embedding.android.Normal
Theme"
        android:resource="@style/NormalTheme"
        />
      <!-- Displays an Android View that continues
showing the launch screen
          Drawable until Flutter paints its first
frame, then this splash
          screen fades out. A splash screen is useful
to avoid any visual
          gap between the end of Android's launch
screen and the painting of
          Flutter's first frame. -->
      <meta-data

android:name="io.flutter.embedding.android.SplashS
creenDrawable"

android:resource="@drawable/launch_background"
```

```
              />
            <intent-filter>
                <action
android:name="android.intent.action.MAIN"/>
                <category
android:name="android.intent.category.LAUNCHER
"/>
            </intent-filter>
        </activity>
        <!-- Don't delete the meta-data below.
             This is used by the Flutter tool to generate
GeneratedPluginRegistrant.java -->
        <meta-data
            android:name="flutterEmbedding"
            android:value="2" />
    </application>
</manifest>
```

Main Code:-

```
import
'package:flutter/material.da
rt';
import
'package:firebase_core/firebase_core.dart';
import
'package:covid19_stats/homepage/homePage.d
art';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}

class MyApp extends StatefulWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  _MyAppState createState() =>
_MyAppState();
}
```

```
class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: "COVID-19 Statistics",
      home: HomePage(),
    );
  }
}
```

Homepage :-

```
import
'package:covid19_stats/cases/cas
ePage.dart';
import
'package:covid19_stats/overview/overvie
wPage.dart';
import
'package:covid19_stats/vaccination/vacin
ationPage.dart';
import 'package:flutter/material.dart';
import
'package:bottom_navy_bar/bottom_navy_
bar.dart';
import
'package:fluttericon/font_awesome5_icon
s.dart';

class HomePage extends StatefulWidget
{
  const HomePage({Key? key}) :
super(key: key);

  @override
  _HomePageState createState() =>
_HomePageState();
}

class _HomePageState extends
```

```
State<HomePage> {
  int _currentIndex = 0;
  final List<Widget> _screens =
[OverviewPage(), CasePage(),
VaccinationPage()];

  void onItemTapped(int val) {
    setState(() {
      _currentIndex = val;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.only(
                bottomLeft:
Radius.circular(20),
                bottomRight:
Radius.circular(20))),
        title: Center(child: Text("COVID-19
Statistics")),
        backgroundColor: Colors.blue[900],
      ),
      body: _screens[_currentIndex],
      bottomNavigationBar: new
BottomNavyBar(
        // backgroundColor: Colors.grey,
        items: [
          BottomNavyBarItem(
              icon:
Icon(FontAwesome5.viruses),
              title: Text(
                "  Overview",
              ),
              activeColor: Colors.redAccent),
          BottomNavyBarItem(
              icon:
Icon(FontAwesome5.chart_line),
              title: Text("  Cases"),
              activeColor: Colors.blueAccent),
          BottomNavyBarItem(
              icon:
Icon(FontAwesome5.syringe),
              title: Text(
                "  Vaccine",
              ),
              activeColor: Colors.lightBlue)
        ],
        selectedIndex: _currentIndex,
        onItemSelected: onItemTapped,
      ),
    );
  }
}
```

Cases Page:-

```
import
'package:covid19_stats/cases/di
strictPage.dart';
import 'package:flutter/material.dart';
import
'package:cloud_firestore/cloud_firestore.d
art';

class CasePage extends StatefulWidget {
  const CasePage({Key? key}) :
super(key: key);

  static final FirebaseFirestore _firebase =
FirebaseFirestore.instance;

  @override
  _CasePageState createState() =>
_CasePageState();
}

class _CasePageState extends
State<CasePage> {
  Map<String, String> stateCode = {
    "AN": "Andaman and Nicobar Islands",
    "AP": "Andhra Pradesh",
    "AR": "Arunachal Pradesh",
    "AS": "Assam",
    "BR": "Bihar",
    "CH": "Chandigarh",
    "CT": "Chhattisgarh",
    "DN": "Dadra and Nagar Haveli",
    "DL": "Delhi",
    "GA": "Goa",
    "GJ": "Gujarat",
    "HR": "Haryana",
    "HP": "Himachal Pradesh",
    "JK": "Jammu and Kashmir",
    "JH": "Jharkhand",
    "KA": "Karnataka",
    "KL": "Kerala",
    "LA": "Ladakh",
    "LD": "Lakshadweep",
    "MP": "Madhya Pradesh",
    "MH": "Maharashtra",
    "MN": "Manipur",
    "ML": "Meghalaya",
    "MZ": "Mizoram",
    "NL": "Nagaland",
    "OR": "Odisha",
    "PY": "Puducherry",
    "PB": "Punjab",
    "RJ": "Rajasthan",
    "SK": "Sikkim",
    "TN": "Tamil Nadu",
    "TG": "Telangana",
    "TR": "Tripura",
    "UP": "Uttar Pradesh",
    "UT": "Uttarakhand",
    "WB": "West Bengal",
  };
  @override
  Widget build(BuildContext context) {
    return Center(
      child: new
StreamBuilder<QuerySnapshot<Map<Stri
ng, dynamic>>>(
        stream:
```

```
CasePage._firebase.collection("stateDaily
Delta").snapshots(),
        builder: (context,

AsyncSnapshot<QuerySnapshot<Map<St
ring, dynamic>>> snapshot) {
          if (!snapshot.hasData) {
            return Center(child:
CircularProgressIndicator());
          }
          return ListView(
            children:
snapshot.data!.docs.map((document) {
              var stateName =
stateCode[document.id];
              return Padding(
                padding: const
EdgeInsets.all(4.0),
                child: Container(
                  child: InkWell(
                    child: new Card(
                      child: ExpansionTile(
                        // backgroundColor:
Colors.blue[100],

collapsedBackgroundColor:
Colors.grey[100],
                        title: Text(
                          stateName.toString(),
                          style: TextStyle(
                              fontSize: 18.0,
fontWeight: FontWeight.bold),
                        ),
                        // subtitle: ,
                        children: <Widget>[
                          Text(
                            "Cases Reported
Today",
                            style: TextStyle(
                                fontSize: 18,
                                fontWeight:
FontWeight.bold,
                                color:
```

```
Colors.black87),
                    ),
                    Row(
                      mainAxisAlignment:
MainAxisAlignment.spaceAround,
                      children: [
                        Column(
                          children: [
                            Padding(
                              padding: const
EdgeInsets.all(4.0),
                              child: Text(
                                "Confirmed",
                                style: TextStyle(
                                    fontSize: 16,
                                    fontWeight:
FontWeight.bold,
                                    color:
Colors.grey),
                              ),
                            ),
                            Padding(
                              padding:
                                  const
EdgeInsets.fromLTRB(0, 0, 0, 4),
                              child: Text(

(document.data()['confirmed'] ?? 0)
                                    .toString(),
                                style: TextStyle(
                                    fontSize: 18,
                                    fontWeight:
FontWeight.bold,
                                    color:
Colors.black87),
                              ),
                            ),
                          ],
                        ),
                        Column(
                          children: [
                            Padding(
                              padding: const
```

```
EdgeInsets.all(4.0),
                  child: Text(

"Death/Recovered",
                   style: TextStyle(
                     fontSize: 16,
                     fontWeight:
FontWeight.bold,
                     color:
Colors.grey),
                  ),
                 ),
                 Padding(
                  padding:
                    const
EdgeInsets.fromLTRB(0, 0, 0, 4),
                  child: Row(
                   children: [
                    Text(

(document.data()['deceased'] ?? 0)
                       .toString(),
                     style:
TextStyle(
                       fontSize: 18,
                       fontWeight:
FontWeight.bold,
                       color:
Colors.red),
                    ),
                    Text(
                     " / ",
                     style:
TextStyle(
                       fontSize: 18,
                       fontWeight:
FontWeight.bold,
                       color:
Colors.black),
                    ),
                    Text(

(document.data()['recovered'] ?? 0)
```

```
                                                     .toString(),
                                                 style:
TextStyle(
                                                     fontSize: 18,
                                                     fontWeight:
FontWeight.bold,
                                                     color:
Colors.green),
                                               ),
                                             ],
                                           ),
                                         ),
                                       ],
                                     ),
                                     Column(
                                       children: [
                                         Padding(
                                           padding: const
EdgeInsets.all(4.0),
                                           child: Text(
                                             "Tested",
                                             style: TextStyle(
                                                 fontSize: 16,
                                                 fontWeight:
FontWeight.bold,
                                                 color:
Colors.grey),
                                           ),
                                         ),
                                         Padding(
                                           padding:
                                               const
EdgeInsets.fromLTRB(0, 0, 0, 4),
                                           child: Text(

(document.data()['tested'] ?? 0)
                                                 .toString(),
                                             style: TextStyle(
                                                 fontSize: 18,
                                                 fontWeight:
FontWeight.bold,
                                                 color:
Colors.deepPurple),
                                                  ),
                                                ),
                                              ],
                                            ),
                                          ],
                                        ),
                                      ],
                                    ),
                                  ),
                                  onTap: () {
                                    DistrictPage.sCode =
                        document.id.toString();

                        Navigator.of(context).push(MaterialPage
                        Route(
                                       builder: (_) => new
                        DistrictPage()));
                                  },
                                ),
                              ),
                            );
                          }).toList(),
                        );
                      },
                    ),
                  );
                }
```

Overview :-

```
import
'dart:m
ath';
        import 'package:intl/intl.dart';
        import 'package:flutter/material.dart';
        import 'package:cloud_firestore/cloud_firestore.dart';
        import 'package:draw_graph/draw_graph.dart';
        import 'package:draw_graph/models/feature.dart';
        import 'package:lottie/lottie.dart';

        class OverviewPage extends StatefulWidget {
```

```
  const OverviewPage({Key? key}) : super(key: key);

  static final FirebaseFirestore _firebase =
FirebaseFirestore.instance;
  @override
  _OverviewPageState createState() => _OverviewPageState();
}

class _OverviewPageState extends State<OverviewPage> {
  int maxNum(confirmed, recovered) {
    int max1 = 0;
    int max2 = 0;
    int maximum = 0;
    for (int i = 0; i < confirmed.length; i++) {
      if (int.parse(confirmed[i].toString()) > max1) {
        max1 = int.parse(confirmed[i].toString());
      }
      if (int.parse(recovered[i].toString()) > max2) {
        max2 = int.parse(recovered[i].toString());
      }
    }
    maximum = max(max1, max2);
    print(maximum);
    return maximum;
  }

  List<double> parseSeries(List<dynamic> data, total) {
    // int value = int.parse(total.toString());
    List<int> parsedData = [];
    for (int i = 0; i < data.length; i++) {
      parsedData.add(data[i]);
    }
    List<double> series = [];
    for (int i = 309; i < parsedData.length; i++) {
      series.add(parsedData[i] / int.parse(total.toString()));
    }
    // print(parsedData[2]);
    return series;
  }

  List<String> getSpaces(List<dynamic> data) {
    int i;
    List<String> space = [];
  for (i = 309; i < data.length; i++) {
    space.add('');
  }
  space[0] = "1 Jan 21";
  space[i - 310] = "Today";
  return space;
}

var time = '';
@override
void initState() {
  getTime();
  super.initState();
}

getTime() async {
  DocumentSnapshot variable = await OverviewPage._firebase
      .collection('countryDailyDelta')
      .doc('lastUpdated')
      .get();
  time =
      "Last Updated: ${DateFormat('dd/MM/yy hh:mm
a').format(DateTime.fromMicrosecondsSinceEpoch(variable['time'
].microsecondsSinceEpoch))}";
}

@override
Widget build(BuildContext context) {
  return SingleChildScrollView(
    scrollDirection: Axis.vertical,
    child: Container(
      child: Column(
        children: [
          Container(
            height: 200,
            child: Lottie.asset("assets/26428-covid-19-protect.json"),
          ),
          Container(
            child: Center(
                child: new StreamBuilder<DocumentSnapshot>(
                    stream: OverviewPage._firebase
                        .collection('countryDailyDelta')
                        .doc('TT')
```

```
                  .snapshots(),
                builder:
                  (context, AsyncSnapshot<DocumentSnapshot>
snapshot) {
                 if (!snapshot.hasData) {
                   return Center(child: CircularProgressIndicator());
                 }
                 var info2 = snapshot.data!;
                 return Padding(
                   padding: const EdgeInsets.all(4.0),
                   child: Card(
                     shape: RoundedRectangleBorder(
                         borderRadius: BorderRadius.circular(20)),
                     color: Colors.red[50],
                     elevation: 5,
                     child: Column(
                       children: [
                         Padding(
                           padding:
                               const EdgeInsets.fromLTRB(0, 8, 0, 0),
                           child: Text(time.toString()),
                         ),
                         Padding(
                           padding: const EdgeInsets.all(8.0),
                           child: Text(
                             "Cases Reported in India Today",
                             style: TextStyle(
                                 fontSize: 20,
                                 fontWeight: FontWeight.bold,
                                 color: Colors.grey),
                           ),
                         ),
                         Padding(
                           padding:
                               const EdgeInsets.fromLTRB(0, 0, 0, 4),
                           child: Text(
                             info2['confirmed'].toString(),
                             style: TextStyle(
                                 fontSize: 40,
                                 fontWeight: FontWeight.bold,
                                 color: Colors.grey[800]),
                           ),
                         ),
Row(
  mainAxisAlignment:
      MainAxisAlignment.spaceAround,
  children: [
    Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(
            "Recovered",
            style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Colors.grey),
          ),
        ),
        Padding(
          padding: const EdgeInsets.fromLTRB(
              0, 0, 0, 4),
          child: Text(
            info2['recovered'].toString(),
            style: TextStyle(
                fontSize: 30,
                fontWeight: FontWeight.bold,
                color: Colors.green),
          ),
        ),
      ],
    ),
    Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(
            "Deceased",
            style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Colors.grey),
          ),
        ),
        Padding(
          padding: const EdgeInsets.fromLTRB(
                                                0, 0, 0, 4),
                                        child: Text(
                                          info2['deceased'].toString(),
                                          style: TextStyle(
                                              fontSize: 30,
                                              fontWeight: FontWeight.bold,
                                              color: Colors.red),
                                        ),
                                      ),
                                    ],
                                  ),
                                ],
                              ),
                              Center(
                                child: new
StreamBuilder<DocumentSnapshot>(
                                  stream: OverviewPage._firebase
                                      .collection('countryTimeSeries')
                                      .doc('TT')
                                      .snapshots(),
                                  builder: (context,
                                      AsyncSnapshot<DocumentSnapshot>
                                          snapshot) {
                                    if (!snapshot.hasData) {
                                      return Center(
                                          child: CircularProgressIndicator());
                                    }
                                    var infoSeries1 = snapshot.data!;
                                    var maximum = maxNum(
                                        infoSeries1['deltaConfirmed'],
                                        infoSeries1['deltaRecovered']);
                                    final List<Feature> features = [
                                      Feature(
                                          title: "Recovered",
                                          color: Colors.green,
                                          data: parseSeries(
                                            infoSeries1['deltaRecovered'],
                                            maximum,
                                          )),
                                      Feature(
                                        title: "Death",
                                        color: Colors.red,
                                        data: parseSeries(
                                  infoSeries1['deltaDeceased'],
                                  maximum),
                            ),
                            Feature(
                              title: "Total Cases",
                              color: Colors.grey,
                              data: parseSeries(
                                  infoSeries1['deltaConfirmed'],
                                  maximum),
                            ),
                          ];
                          return Container(
                            child: Padding(
                              padding: const EdgeInsets.all(4.0),
                              child: LineGraph(
                                features: features,
                                size: Size(
                                    MediaQuery.of(context)
                                        .size
                                        .width,
                                    220),
                                labelX:
                                    getSpaces(infoSeries1['dates']),
                                labelY: [],
                                graphOpacity: 0.1,
                                showDescription: true,
                                graphColor: Colors.black87,
                              ),
                            ),
                          );
                        },
                      ),
                    ),
                  ],
                ),
              ),
            );
          })),
),
Container(
  height: 200,
  child: Lottie.asset("assets/41479-covid19-test.json"),
),
          Container(
            child: Center(
              child: new StreamBuilder<DocumentSnapshot>(
                stream: OverviewPage._firebase
                    .collection('countryWiseRecord')
                    .doc('TT')
                    .snapshots(),
                builder: (context, AsyncSnapshot<DocumentSnapshot>
snapshot) {
                  if (!snapshot.hasData) {
                    return Center(child: CircularProgressIndicator());
                  }
                  var info = snapshot.data!;
                  return Column(
                    children: [
                      Padding(
                        padding: const EdgeInsets.all(4.0),
                        child: Card(
                          shape: RoundedRectangleBorder(
                              borderRadius: BorderRadius.circular(20)),
                          color: Colors.yellow[50],
                          elevation: 5,
                          child: Container(
                            width: MediaQuery.of(context).size.width,
                            child: Column(
                              children: [
                                Padding(
                                  padding: const EdgeInsets.all(8.0),
                                  child: Text(
                                    "Total Cases Reported in India",
                                    style: TextStyle(
                                        fontSize: 20,
                                        fontWeight: FontWeight.bold,
                                        color: Colors.grey),
                                  ),
                                ),
                                Padding(
                                  padding:
                                      const EdgeInsets.fromLTRB(0, 0, 0, 4),
                                  child: Text(
                                    info['confirmed'].toString(),
                                    style: TextStyle(
                                        fontSize: 40,
        fontWeight: FontWeight.bold,
        color: Colors.grey[800]),
  ),
),
Row(
  mainAxisAlignment:
      MainAxisAlignment.spaceAround,
  children: [
    Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(
            "Recovered",
            style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Colors.grey),
          ),
        ),
        Padding(
          padding: const EdgeInsets.fromLTRB(
              0, 0, 0, 4),
          child: Text(
            info['recovered'].toString(),
            style: TextStyle(
                fontSize: 30,
                fontWeight: FontWeight.bold,
                color: Colors.green),
          ),
        ),
      ],
    ),
    Column(
      children: [
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(
            "Deceased",
            style: TextStyle(
                fontSize: 20,
                fontWeight: FontWeight.bold,
                color: Colors.grey),
                                          ),
                                        ),
                                        Padding(
                                          padding: const EdgeInsets.fromLTRB(
                                              0, 0, 0, 4),
                                          child: Text(
                                            info['deceased'].toString(),
                                            style: TextStyle(
                                                fontSize: 30,
                                                fontWeight: FontWeight.bold,
                                                color: Colors.red),
                                          ),
                                        ),
                                      ],
                                    ),
                                  ],
                                ),
                                Center(
                                  child: new
StreamBuilder<DocumentSnapshot>(
                                    stream: OverviewPage._firebase
                                        .collection('countryTimeSeries')
                                        .doc('TT')
                                        .snapshots(),
                                    builder: (context,
                                        AsyncSnapshot<DocumentSnapshot>
                                            snapshot) {
                                      if (!snapshot.hasData) {
                                        return Center(
                                            child:
                                                CircularProgressIndicator());
                                      }
                                      var infoSeries = snapshot.data!;

                                      final List<Feature> features = [
                                        Feature(
                                          title: "Recovered",
                                          color: Colors.green,
                                          data: parseSeries(
                                              infoSeries['recovered'],
                                              info['confirmed']),
                                        ),
                                        Feature(
              title: "Deceased",
              color: Colors.red,
              data: parseSeries(
                  infoSeries['deceased'],
                  info['confirmed']),
            ),
            Feature(
              title: "Total Cases",
              color: Colors.grey,
              data: parseSeries(
                  infoSeries['confirmed'],
                  info['confirmed']),
            ),
          ];
          return Container(
            child: Padding(
              padding: const EdgeInsets.all(4.0),
              child: LineGraph(
                features: features,
                size: Size(
                    MediaQuery.of(context)
                        .size
                        .width,
                    220),
                labelX: getSpaces(
                    infoSeries['dates']),
                labelY: [],
                graphOpacity: 0.1,
                showDescription: true,
                graphColor: Colors.black87,
              ),
            ),
          );
        },
      ),
    ),
    Row(
      mainAxisAlignment:
          MainAxisAlignment.spaceAround,
      children: [],
    )
  ],
),
```

```dart
    ),
  ),
),
Container(
  height: 200,
  child: Lottie.asset("assets/39099-sanitizer.json"),
),
Padding(
  padding: const EdgeInsets.all(4.0),
  child: Card(
    shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(20)),
    color: Colors.blue[50],
    elevation: 5,
    child: Container(
      width: MediaQuery.of(context).size.width,
      child: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: Text(
              "Total Tested",
              style: TextStyle(
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                  color: Colors.grey),
            ),
          ),
          Padding(
            padding:
                const EdgeInsets.fromLTRB(0, 0, 0, 4),
            child: Text(
              info['tested'].toString(),
              style: TextStyle(
                  fontSize: 40,
                  fontWeight: FontWeight.bold,
                  color: Colors.deepPurple),
            ),
          ),
          Row(
            mainAxisAlignment:
                MainAxisAlignment.spaceAround,
            children: [
Column(
  children: [
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Text(
        "Vaccine Dose 1",
        style: TextStyle(
            fontSize: 20,
            fontWeight: FontWeight.bold,
            color: Colors.grey),
      ),
    ),
    Padding(
      padding: const EdgeInsets.fromLTRB(
          0, 0, 0, 4),
      child: Text(
        info['vaccinated1'].toString(),
        style: TextStyle(
            fontSize: 30,
            fontWeight: FontWeight.bold,
            color: Colors.blue),
      ),
    ),
  ],
),
Column(
  children: [
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Text(
        "Vaccine Dose 2",
        style: TextStyle(
            fontSize: 20,
            fontWeight: FontWeight.bold,
            color: Colors.grey),
      ),
    ),
    Padding(
      padding: const EdgeInsets.fromLTRB(
          0, 0, 0, 4),
      child: Text(
        info['vaccinated2'].toString(),
        style: TextStyle(
                                fontSize: 30,
                                fontWeight: FontWeight.bold,
                                color: Colors.blue[900]),
                            ),
                          ),
                        ],
                      ),
                    ],
                  ),
                  Center(
                    child: new
StreamBuilder<DocumentSnapshot>(
                      stream: OverviewPage._firebase
                          .collection('countryTimeSeries')
                          .doc('TT')
                          .snapshots(),
                      builder: (context,
                          AsyncSnapshot<DocumentSnapshot>
                              snapshot) {
                        if (!snapshot.hasData) {
                          return Center(
                              child:
                                  CircularProgressIndicator());
                        }
                        var infoSeries1 = snapshot.data!;

                        final List<Feature> features = [
                          // Feature(
                          //   title: "Total Tested",
                          //   color: Colors.purple,
                          //   data: parseSeries(
                          //       infoSeries['tested'],
                          //       info['']),
                          // ),
                          Feature(
                            title: "Vaccine Dose 1",
                            color: Colors.blue,
                            data: parseSeries(
                                infoSeries1['vaccinated1'],
                                info['vaccinated1']),
                          ),
                          Feature(
                            title: "Vaccine Dose 2",
                                      color: Colors.deepPurple,
                                      data: parseSeries(
                                          infoSeries1['vaccinated2'],
                                          info['vaccinated1']),
                                    ),
                                  ];
                                  // print(infoSeries1['vaccinated1']);
                                  return Container(
                                    child: Padding(
                                      padding: const EdgeInsets.all(4.0),
                                      child: LineGraph(
                                        features: features,
                                        size: Size(
                                            MediaQuery.of(context)
                                                .size
                                                .width,
                                            220),
                                        labelX: getSpaces(
                                            infoSeries1['dates']),
                                        labelY: [],
                                        graphOpacity: 0.1,
                                        showDescription: true,
                                        graphColor: Colors.black87,
                                      ),
                                    ),
                                  );
                                },
                              ),
                            ),
                          ],
                        ),
                      ),
                    ),
                  ),
                ],
              );
            },
          ),
        ),
      ),
    ],
  ),
),
    );
  }
}
```

REFERENCES

[1] D. Pao, X. Wang, X. Wang, C. Cao, Y. Zhu, “String Searching Engine for Virus Scanning”, Computer, IEEE Transactions on Computers, pp.1 -1, 2010

[2] Symantec Corporation, Understanding Heuristics: Symantec’s Bloodhoud Technology, Symantec White Paper Series, 1997.

[3] A. E. Stepan, “Defeating Polymorphicsm: Beyond Emulation”, Virus Bulletin Conference October 2005, pp. 40-48, Oct 2005

[4] Wing Wong, Mark Stamp, Hunting for Metamorphic Engines, SpringerVerlag France, 2006

[5] Burak Bayoglu, Ibrahim Sogukpinar, Polymorphic Worm Detection Using Token-Pair Signature, Proceedings of the 4th International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, p.p. 7-12

[6] Jean-Marie Borello, Ludovic Mé, Code Obfuscation Techniques for Metamorphic Viruses, Springer-Verlag France, 2008

[7] Magnus O.Myreen. Verified Just-In-Time Compiler on x86. Principles of Programming Languages (POPL), 2010

[8] Tom Brosch and Maik Morgenstern. Runtime Packers: The Hidden Problem. Proc. BlackHat USA, Black Hat, www.blackhat.com/presentations/bh-usa-06 BH-US-o6-Morgenstern.pdf; 2006. [accessed 02.04.11].

[9] Kang, M.G., Poosankam, P., Yin, H.: Renovo: A hidden code extractor for packed executable. In: Proceedings of the 5th ACM Workshop on Recurring Malcode (MORM) (2007)

[10] Paul Royal, Mitch Halpin, David Dagon, Robert Edmonds, Wenke Lee, PolyUnpack: Automating the Hidden-Code Extraction of UnpackExecuting Malware

[11] S. TreadWell and M. Zhou, “A Heuristic Approach for Detection of Obfuscated Malware”, Intelligence and Security Informatics, 2009. ISI’ 09, pp. 291-299, June 2009

[12] A. H. Sung, J. Xu, P. Chavez, S. Mukkamala, “Static Analyzer of Vicious Executables (SAVE),” Proceeding ACSAC ’04 Proceedings of the 20th Annual Computer Security Applications Conference, pp. 326-

334, 2004
[13] M. Schultz, E. Eskin, and E. Zadok. Data mining methods for detection of new malicious executables. In Proceedings of IEEE International Conference on Data Mining, pp. 38-49, May 2001.
[14] James M. Aquilina, Eoghan Casey, Cameron H. Malin, Malware Forensics: Investigating and Analyzing Malicious Code, Cahpter 7, p.p 340
[15] Matt Pietrek, An In-Depth Look Into the Win32 Portable Executable File Format, MSDN Magazine, February 2002
[16] Microsoft MSDN, http://msdn.microsoft.com/enus/library/windows/desktop/ms683156(v=vs.85).aspx
[17] Windows, Dev Center – Desktop, http://msdn.microsoft.com/enus/library/windows/desktop/aa364934(v=vs.85).aspx
[18] Windows, Dev Center – Desktop, http://msdn.microsoft.com/enus/library/windows/desktop/aa364418(v=vs.85).aspx
[19] Windows, Dev Center – Desktop, http://msdn.microsoft.com/enus/library/windows/desktop/aa364428(v=vs.85).aspx
[20] Hennessy, John A.; Goldberg, David (1996). Computer Architecture: A Quantitative Approach. Morgan Kaufmann Publishers. ISBN 1-55860-329-8
[21] B. Schwarz, S. Debray, and G. Andrews, "Disassembly of Executable Code Revisited", Proc. of 9th Working Conference on Reverse Engineering (WCRE), pp. 45–54, 2002.
[22] Reverend Bill Blunden, The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System, p.p 54-56, Chapter 2