KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

Project: Telegram Bot

Submitted By – **Smriti (1906639)**
Abhay(1906647)

ACKNOWLEDGEMENT

30 March 2022

We would like to extend our sincerest gratitude to Mr. Ajay Anand, Assistant Professor, School of Computer Engineering, KIIT for giving us this opportunity to work on this project. We thank him for the constant guidance and support given wherever needed.

Moreover, we would like to thank our team members who made sure that this “team effort” was justified and a fruitful output was achieved.

Lastly, we would like to thank millions of anonymous computer scientists and enthusiasts around the world who help students like us through open-source platforms such as git.

TABLE OF CONTENTS

SECTION I: Project Background

We have made a telegram bot which updates the city-wise status of COVID-19 on a daily basis. This includes information about the number of affected people, number of active cases, number of recovered people, number of deaths etc.

What is Telegram?

Telegram is a freeware, cross-platform, cloud-based instant messaging (IM) service. The service also provides end-to-end encrypted video calling, VoIP, file sharing and several other features. It was launched for iOS on 14 August 2013 and Android in October 2013. The servers of Telegram are distributed worldwide to decrease frequent data load with five data centers in different regions, while the operational center is based in Dubai in the United Arab Emirates. Various client apps are available for desktop and mobile platforms including official apps for Android, iOS, Windows, macOS and Linux. All of Telegram's official components are open source,[20] with the exception of the server which is closed-sourced and proprietary.

Telegram provides end-to-end encrypted voice and video calls and optional end-to-end encrypted "secret" chats. Cloud chats and groups are encrypted between the app and the server, so that ISPs and other third parties on the network can't access the data, but the Telegram server can. Users can send text and voice messages, make voice, and video calls, and share an unlimited number of images, documents (maximum of 2 GB per file), user locations, animated stickers, contacts, and audio files.

In January 2021, Telegram surpassed 500 million monthly active users. It was the most downloaded app worldwide in January 2021 with 1 billion downloads globally.

Telegram Bot

A **telegram bot** is a telegram account operated by programs. It is a third-party application that runs inside Telegram. They can respond to messages or mentions, can be invited into groups, and can be integrated into other programs. It also accepts online payments with credit cards and Apple Pay. Users can interact with bots by sending them messages, commands and inline requests.

In June 2015, Telegram launched a platform for third-party developers to create bots. There are also inline bots, which can be used from any chat screen. In order to activate an inline bot, user needs to type in the message field a bot's username and query. The bot then will offer its content. User can choose from that content and send it within a chat.

These bots can also handle transactions provided by 'Paymentwall', 'Yandex.Money', 'Stripe', 'Ravepay', 'Razorpay', 'QiWi' and 'Google Pay' for different countries. They can also power Telegram's gaming platform, which utilizes HTML5, so games are loaded on-demand as needed, like ordinary webpages.

General functionalities of a telegram bot

Telegram bots can help users in a plethora of ways. Some of them include:

Can give customized notifications and news. A bot can act as a smart newspaper, sending you relevant content as soon as it's published.

Can create custom tools. A bot may provide you with alerts, weather forecasts, translations, formatting or other services.

Can build social services. A bot could connect people looking for conversation partners based on common interests or proximity.

Working of a telegram bot

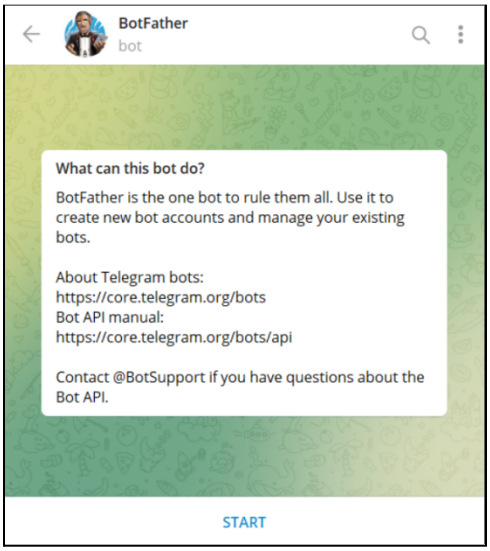
At the core, Telegram Bots are special accounts that do not require an additional phone number to set up. Users can interact with bots in two ways:

- Send messages and commands to bots by opening a chat with them or by adding them to groups.
- Send requests directly from the input field by typing the bot's @username and a query. This allows sending content from inline bots directly into any chat, group, or channel.

Messages, commands, and requests sent by users are passed to the software running on the developers' servers. Telegram's intermediary server handles all encryption and communication with the Telegram API for the developer. The developer can communicate with this server via a simple HTTPS-interface that offers a simplified version of the Telegram API. The interface is called Telegram Bot API.

Steps to create a Telegram Bot

Telegram bots are created with the help of "BotFather". "BotFather" helps one create new bots and change settings for existing ones. The detailed steps to create a new bot are:



Step 1: Use the /newbot command to create a new bot. The BotFather will ask the developer for a name and username, then generate an authentication token for your new bot.

Step 2: The name of the bot is displayed in contact details and elsewhere.

Step 3: The Username is a short name, to be used in mentions and t.me links. Usernames are 5-32 characters long and are case insensitive, but may only include Latin characters, numbers, and underscores. The bot's username must end in 'bot', such as, 'tetris_bot' or 'TetrisBot'.

Step 4: The token is a string that is required to authorize the bot and send requests to the Bot API. This token must be kept secure, as it can fully control one’s bot.

SECTION II: CODE SNIPPETS

1) main.py

```
import requests
import json
import time


idi = -1001534803074
header = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36'}

message = "Hello! Welcome to Covid Platform!"
# read token file
with open('token.txt', 'r') as f:
   TOKEN = f.read()

base_url = "https://api.telegram.org/bot"+TOKEN
send_url = "https://api.telegram.org/bot"+TOKEN+"/sendMessage"
send_photo = "https://api.telegram.org/bot"+TOKEN+"/sendPhoto"

send_notification = False
"https://cdn-api.co-vin.in/api/v2/appointment/sessions/public/calendarByDistric
t?district_id="+i+"&date=31-03-2021"
sessions["available_capacity_dose1"],
sessions["available_capacity_dose2"]
"https://api.telegram.org/bot5261206107:AAFYjjoYoFpa0j3P_MSVwyuv7IgG8__zudo/sen
dMessage?chat_id=-1001534803074&text="+parse_data
urls = [
"https://drive.google.com/file/d/1KunYh-60sS-6UB7trWIklLBUgn3TkMMe/view?usp=sha
ring",
"https://drive.google.com/file/d/1mCz8GqMSECX4wwRUkPzQe0TCti1uLs83/view?usp=sha
ring",
"https://drive.google.com/file/d/1Ij1-FfFt2n03-838ViWo-oZ1TQtqA0W9/view?usp=sha
ring",
"https://drive.google.com/file/d/1Fa8jSh2eEBVBt617mT7QE9xqyrLp8--0/view?usp=sha
ring",
"https://drive.google.com/file/d/13ghoErKPmtAOpJDlOWVZD58e0mKGjOFx/view?usp=sha
ring",
"https://drive.google.com/file/d/1GPhR6yo1e0QhPBfQraLH2Lv_JN_W9rLp/view?usp=sha
ring",
"https://drive.google.com/file/d/1EoYL8ZrdzFpatKhKsZLV5zbBYRYvpY36/view?usp=sha
ring",
"https://drive.google.com/file/d/1NBgCm520RGSsKBfYi4BRw1tzpQJciBTi/view?usp=sha
ring",
"https://drive.google.com/file/d/1_5TuW4XUQiLV18nxvF-WIpfrWb_C5e2R/view?usp=sha
ring",
       ]
```

```
captions = [
   "State wise Analysis", "COVID 19 : Pandemic In India",
   "State wise Confirmed/Death/Recovered Stacked",
   "State wise Cases per 100 confirmed",
   "7 days Mean Confirmed",
   "Recovered",
   "Death Report",
   "Total Cases Statistics",
   "Top21 Confirmed/Death/Recovered Stacked"
]

def read_msg(offset) :
   parameters = {
       "offset": offset
   }

   resp = requests.get(base_url + "/getUpdates", data=parameters)
   data = resp.json()

   print(data)

   for result in data["result"]:
       send_msg(result)
       print(result)

   if data["result"]:
       return data["result"][-1]["update_id"] + 1



def auto_answer(message):
   if message == "Availability of vaccine in 512 on 31-03-2023":
       return "Sorry, I could not understand you !!! I am still learning and
try to get better in answering."

   elif message.startswith("Hello"):
       answer = "Welcome to COVID Platform !!!!! \n\n 1. Type 'Availability of
vaccine in DistrictCode on Date' \n\n 2. Type 'State wise Analysis' \n\n3.
Type 'COVID Stats' \n\n4.  Type 'State wise Confirmed/Death/Recovered'\n\n5.
Type 'State wise Cases' \n\n 6.  Type '7 days Mean' \n\n7. Type 'Top21
Confirmed/Death/Recovered Stacked'\n\n 8. Type 'Death Report'\n\n 9. Type
'Recovered Report'\n\n 10. Type 'Total Cases Statistics'"
       return answer

   elif message.startswith("Live Update vaccine"):
       pin=message.split()[3]
       dt=message.split()[4]
       global  send_notification
       send_notification=True
```

```
        while (send_notification):
            welcome = base_url + "/sendMessage?chat_id=-1001534803074&text=" +
message
            requests.get(welcome)
                        for i in range(0,100):
                    x =
"https://cdn-api.co-vin.in/api/v2/appointment/sessions/public/calendarByDistric
t?district_id=" +pin+ "&date="+dt
                    data = requests.get(x, headers=header)
                    results = json.loads(data.text)
                    counts = results["centers"]
                    if (len(counts) > 0):
                        msg = []
                        for centers in counts:
                            # msg=[]
                            msg.append({
                                "district_name": centers["district_name"],
                                # "district_name": centers["district_name"],
                                "name": centers["name"],
                                "fees": centers["fee_type"]
                            })
                            for sessions in centers["sessions"]:
                                msg.append({
                                    "min_age_limit": sessions["min_age_limit"],
                                    "vaccine": sessions["vaccine"],
                                    "slots": sessions["slots"],
                                    "available_capacity_dose1":
sessions["available_capacity_dose1"],
                                    "available_capacity_dose2":
sessions["available_capacity_dose2"]

                                })
                            parse_data = json.dumps(msg)
                            parse_data = parse_data.replace("{", "")
                            parse_data = parse_data.replace("}", "\n\n")
                            parse_data = parse_data.replace("[", "")
                            parse_data = parse_data.replace("]", "")
                            parse_data = parse_data.replace(",", "\n")
                            print(parse_data)
                            un_url =
"https://api.telegram.org/bot5261206107:AAFYjjoYoFpa0j3P_MSVwyuv7IgG8__zudo/sen
dMessage?chat_id=-1001534803074&text=" + parse_data
                            y = requests.get(un_url)
                            print(y)
                            time.sleep(20)
            time.sleep(600)

    elif message.lower()=="stop":
        print(message.lower())
```

```
        return "Covid Bot Stopped"

    elif message == "State wise Analysis":
        length = len(urls)
        for i in range(length):
            if captions[i] == message:
                # time.sleep(10)
                parameters = {
                    "chat_id": "-1001534803074",
                    "photo": urls[i],
                    "caption": captions[i]
                }
                resp = requests.get(send_photo, data=parameters)
                print(resp.text)
                break
            else:
                pass
    elif message == "COVID Stats":
            length = len(urls)
            for i in range(length):
                if captions[i] == "COVID 19 : Pandemic In India":
                    # time.sleep(10)
                    parameters = {
                        "chat_id": "-1001534803074",
                        "photo": urls[i],
                        "caption": captions[i]
                    }
                    resp = requests.get(send_photo, data=parameters)
                    print(resp.text)
                    break
                else:
                    pass
    elif message == "State wise Confirmed/Death/Recovered":
            length = len(urls)
            for i in range(length):
                if captions[i] == "State wise Confirmed/Death/Recovered
Stacked":
                    # time.sleep(10)
                    parameters = {
                        "chat_id": "-1001534803074",
                        "photo": urls[i],
                        "caption": captions[i]
                    }
                    resp = requests.get(send_photo, data=parameters)
                    print(resp.text)
                    break
                else:
                    pass
    elif message == "State wise Cases":
```

```
        length = len(urls)
        for i in range(length):
            if captions[i] == "State wise Cases per 100 confirmed":
                # time.sleep(10)
                parameters = {
                    "chat_id": "-1001534803074",
                    "photo": urls[i],
                    "caption": captions[i]
                }
                resp = requests.get(send_photo, data=parameters)
                print(resp.text)
                break
            else:
                pass
    elif message == "7 days Mean":
        length = len(urls)
        for i in range(length):
            if captions[i] == "7 days Mean Confirmed":
                # time.sleep(10)
                parameters = {
                    "chat_id": "-1001534803074",
                    "photo": urls[i],
                    "caption": captions[i]
                }
                resp = requests.get(send_photo, data=parameters)
                print(resp.text)
                break
            else:
                pass
    elif message == "Recovered Report":
        length = len(urls)
        for i in range(length):
            if captions[i] == "Recovered":
                # time.sleep(10)
                parameters = {
                    "chat_id": "-1001534803074",
                    "photo": urls[i],
                    "caption": captions[i]
                }
                resp = requests.get(send_photo, data=parameters)
                print(resp.text)
                break
            else:
                pass
    elif message == "Death Report":
        length = len(urls)
        for i in range(length):
            if captions[i] == "Death Report":
                # time.sleep(10)
```

```
            parameters = {
                "chat_id": "-1001534803074",
                "photo": urls[i],
                "caption": captions[i]
            }
            resp = requests.get(send_photo, data=parameters)
            print(resp.text)
            break
        else:
            pass


elif message == "Total Cases Statistics":
    length = len(urls)
    for i in range(length):
        if captions[i] == "Total Cases Statistics":
            # time.sleep(10)
            parameters = {
                "chat_id": "-1001534803074",
                "photo": urls[i],
                "caption": captions[i]
            }
            resp = requests.get(send_photo, data=parameters)
            print(resp.text)
            break
        else:
            pass
elif message == "Top21 Confirmed/Death/Recovered Stacked":
    length = len(urls)
    for i in range(length):
        if captions[i] == "Top21 Confirmed/Death/Recovered Stacked":
            # time.sleep(10)
            parameters = {
                "chat_id": "-1001534803074",
                "photo": urls[i],
                "caption": captions[i]
            }
            resp = requests.get(send_photo, data=parameters)
            print(resp.text)
            break
        else:
            pass
```

```
def send_msg(message):
   text = message["message"]["text"]
   message_id = message["message"]["message_id"]
   answer = auto_answer(text)
   parameters = {
       "chat_id": "-1001534803074",
       "text": answer,
       "reply_to_message_id": message_id
   }
   resp = requests.get(base_url + "/sendMessage", data=parameters)
   print(resp.text)
offset = 0
while True:
   offset = read_msg(offset)
```

2) COVID DATA ANALYSIS

In [20]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objs as go
from plotly.subplots import make_subplots
from urllib.request import urlopen
import json
import requests
import re
import math
import warnings
warnings.filterwarnings('ignore')
```

In [21]:
```
confirmed_color = 'red'
recovered_color = 'green'
death_color = 'yellow'
active_color = 'purple'
```

In [22]:
```
df1 = "https://api.covid19india.org/state_district_wise.json"
df2 = "https://api.covid19india.org/data.json"
```

In [23]:
```
def getting_data(url):
    response = requests.get(url)
    data = response.content.decode('utf-8')
    return data
```

In [24]:
```
df_state = json.loads(getting_data(df1))
df = json.loads(getting_data(df2))
```

In [6]:
```
lis = []
state_names = df_state.keys()
for state in state_names:
    district_names = df_state[state]['districtData'].keys() #Districts of Current State
    for district in district_names:
        temp = df_state[state]['districtData'][district]
        var_lis = [state,district,temp.get('confirmed'),temp.get('recovered'),
                   temp.get('active'),temp.get('deceased')]
        lis.append(var_lis)
    district_wise = pd.DataFrame(lis,columns=['State/UT','District','Confirmed',
                                              'Recovered','Active','Death'])
district_wise.head()
```

Out[6]:

	State/UT	District	Confirmed	Recovered	Active	Death
0	State Unassigned	Unassigned	0	0	0	0
1	Andaman and Nicobar Islands	Nicobars	0	0	0	0
2	Andaman and Nicobar Islands	North and Middle Andaman	1	1	0	0
3	Andaman and Nicobar Islands	South Andaman	51	32	19	0
4	Andaman and Nicobar Islands	Unknown	7496	7380	-13	129

In [25]:
```
temp = [[i['state'],i['confirmed'],i['recovered'],i['active'],i['deaths'],
         i['lastupdatedtime']] for i in df['statewise']]
statewise_total = pd.DataFrame(temp,columns=['State/UT','Confirmed','Recovered',
                                             'Active','Death','LastUpdateTime'])
statewise_total.head()
```

Out[25]:

	State/UT	Confirmed	Recovered	Active	Death	LastUpdateTime
0	Total	32249900	31441260	363849	432112	13/08/2021 23:27:22
1	Andaman and Nicobar Islands	7549	7419	1	129	13/08/2021 23:27:22
2	Andhra Pradesh	1994606	1963728	17218	13660	13/08/2021 23:27:22
3	Arunachal Pradesh	51513	49425	1836	252	13/08/2021 23:27:22
4	Assam	580657	566101	7707	5502	13/08/2021 23:27:22

In [26]:
```
statewise_total['Confirmed']=statewise_total['Confirmed'].astype('int')
statewise_total['Recovered']=statewise_total['Recovered'].astype('int')
statewise_total['Active']=statewise_total['Active'].astype('int')
statewise_total['Death']=statewise_total['Death'].astype('int')
statewise_total['RecoveryRate%'] = round(statewise_total['Recovered']/statewise_total['Confirmed']*100,2)
statewise_total['MortalityRate%'] = round(statewise_total['Death']/statewise_total['Confirmed']*100,2)
statewise_total['Active/100 Confirmed'] = round(statewise_total['Active']/statewise_total['Confirmed']*100,2)
for i,y in enumerate(statewise_total['LastUpdateTime']):
    statewise_total['LastUpdateTime'][i] = pd.to_datetime(y.split(' ')[0])
statewise_total['LastUpdateTime'] = pd.to_datetime(statewise_total['LastUpdateTime'])
statewise_total.head()
```

Out[26]:

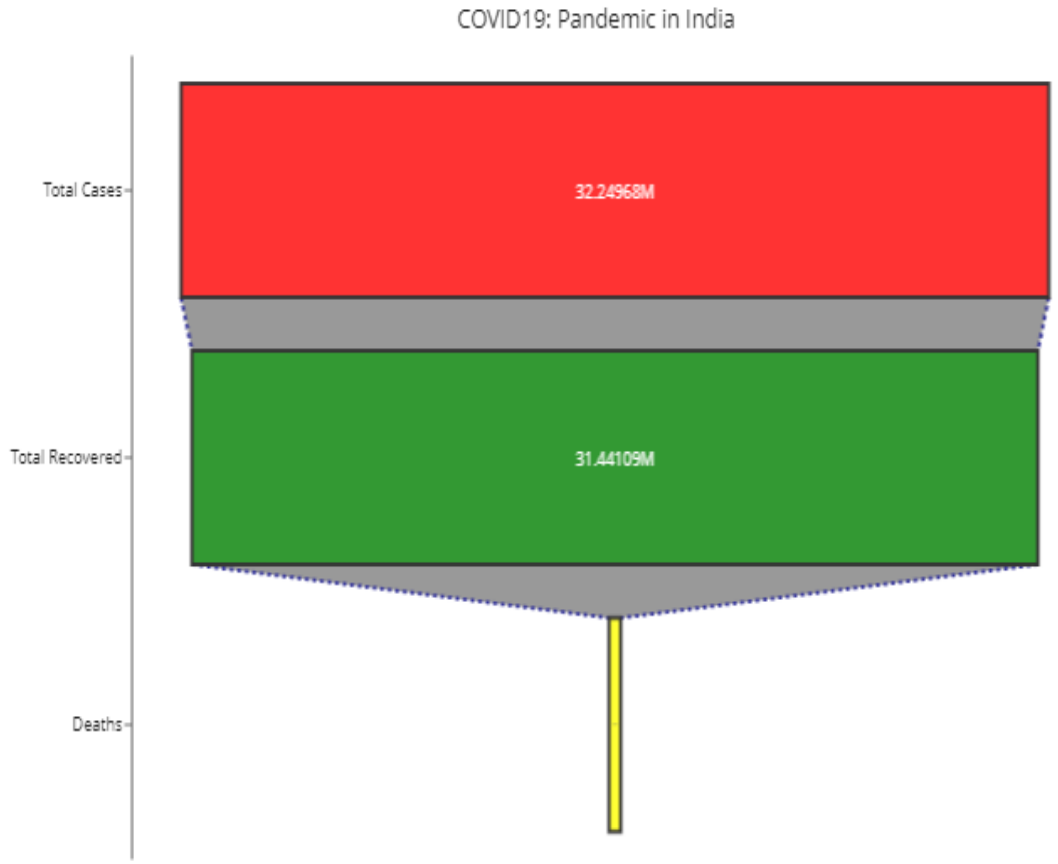
	State/UT	Confirmed	Recovered	Active	Death	LastUpdateTime	RecoveryRate%	MortalityRate%	Active/100 Confirmed
0	Total	32249900	31441260	363849	432112	2021-08-13	97.49	1.34	1.13
1	Andaman and Nicobar Islands	7549	7419	1	129	2021-08-13	98.28	1.71	0.01
2	Andhra Pradesh	1994606	1963728	17218	13660	2021-08-13	98.45	0.68	0.86
3	Arunachal Pradesh	51513	49425	1836	252	2021-08-13	95.95	0.49	3.56
4	Assam	580657	566101	7707	5502	2021-08-13	97.49	0.95	1.33

In [27]:
```
timeseries = [list(i.values()) for i in df['cases_time_series']]
timeseries = pd.DataFrame(timeseries,columns=df['cases_time_series'][0].keys())
timeseries
```

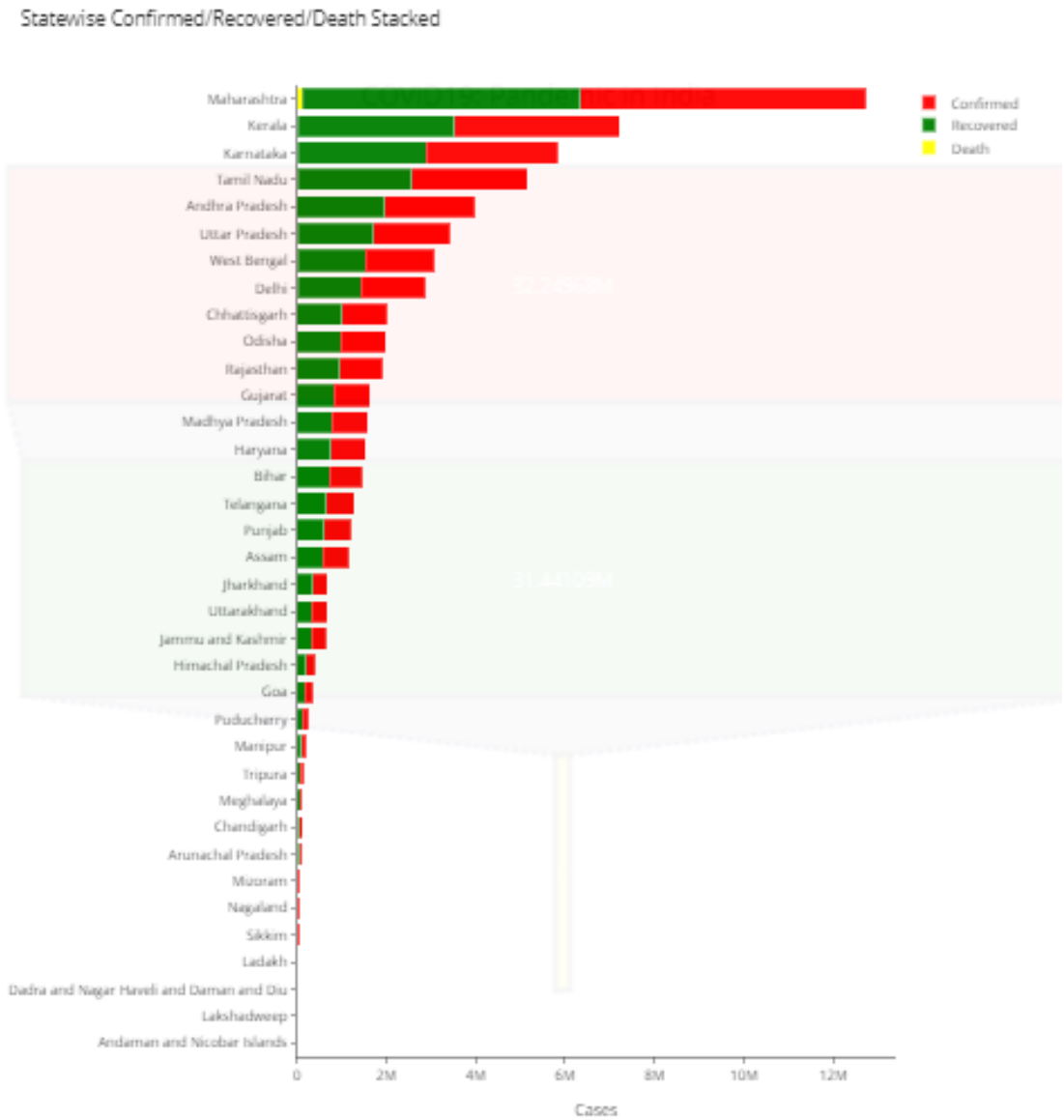
Out[27]:

	dailyconfirmed	dailydeceased	dailyrecovered	date	dateymd	totalconfirmed	totaldeceased	totalrecovered
0	1	0	0	30 January 2020	2020-01-30	1	0	0
1	0	0	0	31 January 2020	2020-01-31	1	0	0
2	0	0	0	1 February 2020	2020-02-01	1	0	0
3	1	0	0	2 February 2020	2020-02-02	2	0	0
4	1	0	0	3 February 2020	2020-02-03	3	0	0
...	...	...	...	...	...	...	...	...
560	40081	583	42156	12 August 2021	2021-08-12	32116848	429695	31294596
561	38761	477	35759	13 August 2021	2021-08-13	32155609	430172	31330355
562	36135	491	37936	14 August 2021	2021-08-14	32191744	430663	31368291
563	33245	421	35936	15 August 2021	2021-08-15	32224989	431084	31404227

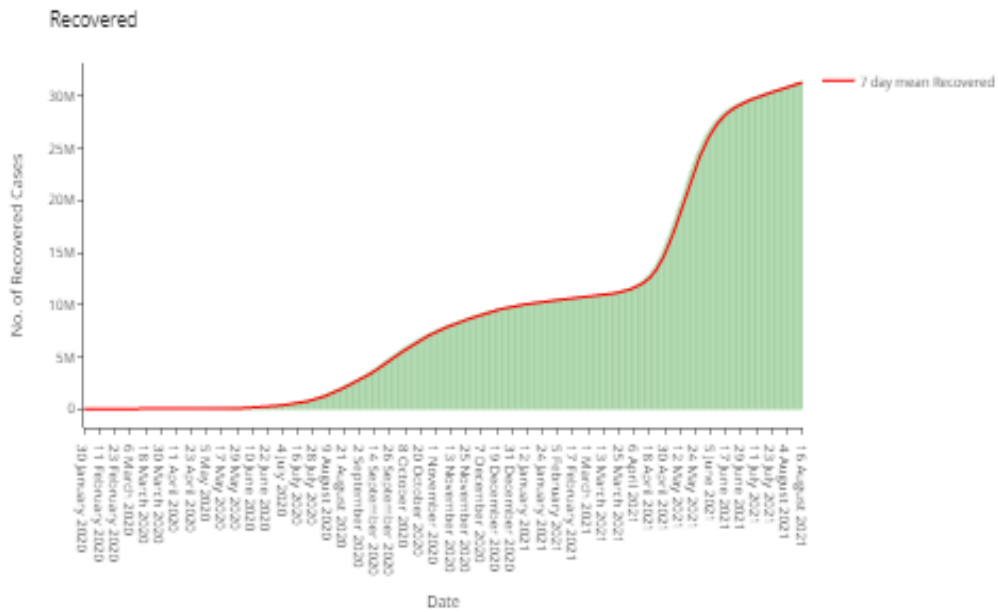
In [11]:
```
fig = go.Figure(go.Funnel(
    x = [timeseries['totalconfirmed'].iloc[-1],timeseries['totalrecovered'].iloc[-1],
         timeseries['totaldeceased'].iloc[-1]],
    y = ["Total Cases", "Total Recovered",  "Deaths"],
    textposition = "inside",
    textinfo = "value",
    opacity = 0.8,
    marker = {"color": [confirmed_color,recovered_color,death_color],
              "line": {"width": 2.5, "color": 'Black'}},
    connector = {"line": {"color": "navy", "dash": "dot", "width": 2.5}}))
fig.update_layout(
    template="simple_white",
    title={'text': "COVID19: Pandemic in India",'x':0.5,'y':0.9,
        'xanchor': 'center','yanchor': 'top'})
fig.update_layout(height=700)
fig.show()
```



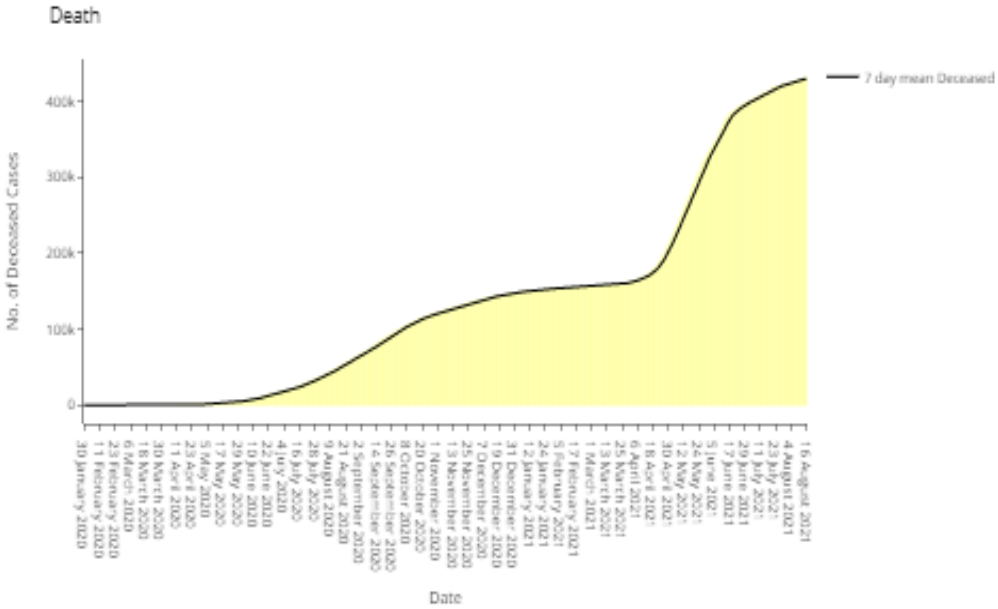
```
In [12]: temp = statewise_total[statewise_total['State/UT']!='Total']
         temp = temp[temp['State/UT']!='State Unassigned']
         fig = go.Figure(data=[
             go.Bar(name='Death', y=temp['State/UT'], x=temp['Death'],orientation='h',marker_color=death_color),
             go.Bar(name='Recovered', y=temp['State/UT'], x=temp['Recovered'],orientation='h',marker_color=recovered_color),
             go.Bar(name='Confirmed', y=temp['State/UT'], x=temp['Confirmed'],orientation='h',marker_color=confirmed_color)
         ])
         fig.update_layout(barmode='stack',title='Statewise Confirmed/Recovered/Death Stacked', xaxis_title="Cases", yaxis_title="Sta
                           yaxis_categoryorder = 'total ascending', height = 1000,
                           template='simple_white')
         fig.show()
```



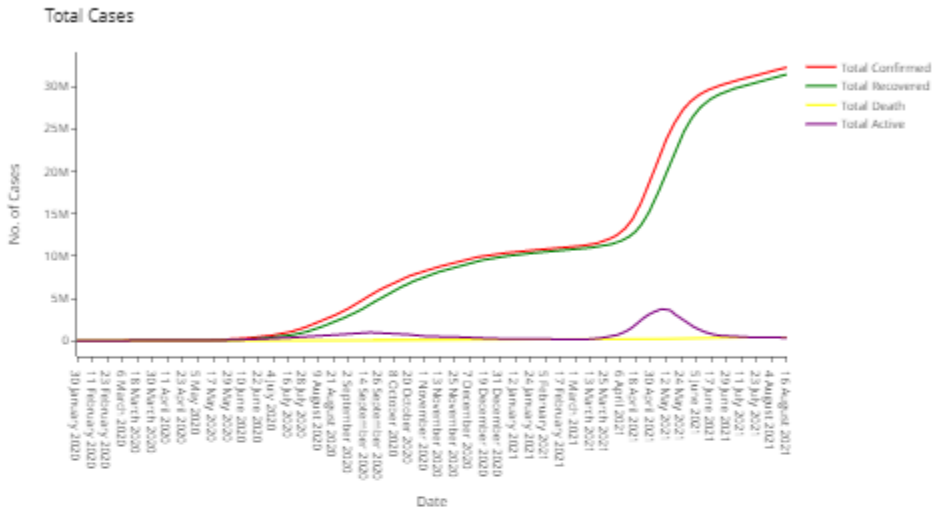
In [17]:
```
fig = px.bar(timeseries, x='date', y='totalrecovered',
             color_discrete_sequence=[recovered_color],template='simple_white')
fig.update_layout(title='Recovered', xaxis_title="Date", yaxis_title="No. of Recovered Cases")
fig.add_scatter(x=timeseries['date'],y=timeseries['7dyMnRecovered'],name='7 day mean Recovered',
                marker={'color': 'red','opacity': 0.6,'colorscale': 'Viridis'},)
fig.show()
```



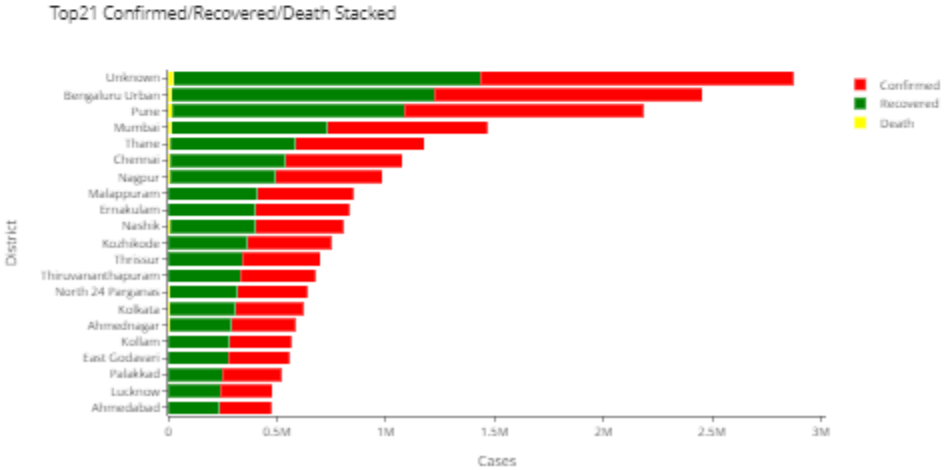
In [19]:
```
fig = px.bar(timeseries, x='date', y='totaldeceased',
             color_discrete_sequence=[death_color],template='simple_white')
fig.update_layout(title='Death', xaxis_title="Date", yaxis_title="No. of Deceased Cases")
fig.add_scatter(x=timeseries['date'],y=timeseries['7dyMnDeceased'],name='7 day mean Deceased',
                marker={'color': 'black','opacity': 0.6,'colorscale': 'Viridis'},)
fig.show()
```



In [21]:
```
timeseries['totalactive'] = timeseries.totalconfirmed-timeseries.totalrecovered-timeseries.totaldeceased
fig = px.line(color_discrete_sequence=[confirmed_color],template='simple_white')
fig.add_scatter(x=timeseries['date'],y=timeseries['totalconfirmed'],name='Total Confirmed',marker={'color': confirmed_color,
fig.add_scatter(x=timeseries['date'],y=timeseries['totalrecovered'],name='Total Recovered',marker={'color': recovered_color,
fig.add_scatter(x=timeseries['date'],y=timeseries['totaldeceased'],name='Total Death',marker={'color': death_color,'opacity'
fig.add_scatter(x=timeseries['date'],y=timeseries['totalactive'],name='Total Active',marker={'color': active_color,'opacity'
fig.update_layout(title='Total Cases', xaxis_title="Date", yaxis_title="No. of Cases")
fig.show()
```



In [22]:
```
temp = district_wise.sort_values('Confirmed').tail(21)
fig = go.Figure(data=[
    go.Bar(name='Death', y=temp['District'], x=temp['Death'].head(21),orientation='h',marker_color=death_color),
    go.Bar(name='Recovered', y=temp['District'], x=temp['Recovered'].head(21),orientation='h',marker_color=recovered_color),
    go.Bar(name='Confirmed', y=temp['District'], x=temp['Confirmed'].head(21),orientation='h',marker_color=confirmed_color)
])
fig.update_layout(barmode='stack',title='Top21 Confirmed/Recovered/Death Stacked', xaxis_title="Cases", yaxis_title="Distric
                  yaxis_categoryorder = 'total ascending',
                  template='simple_white')
fig.show()
```

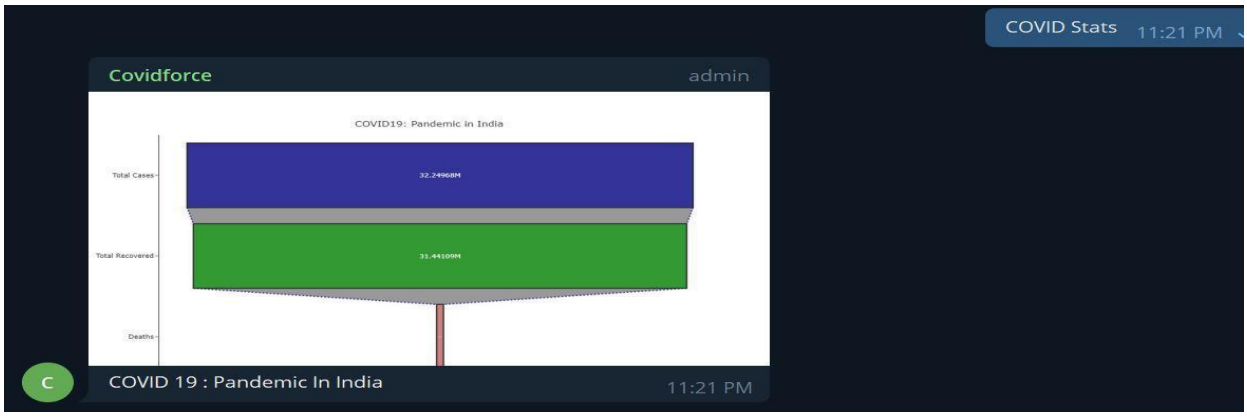


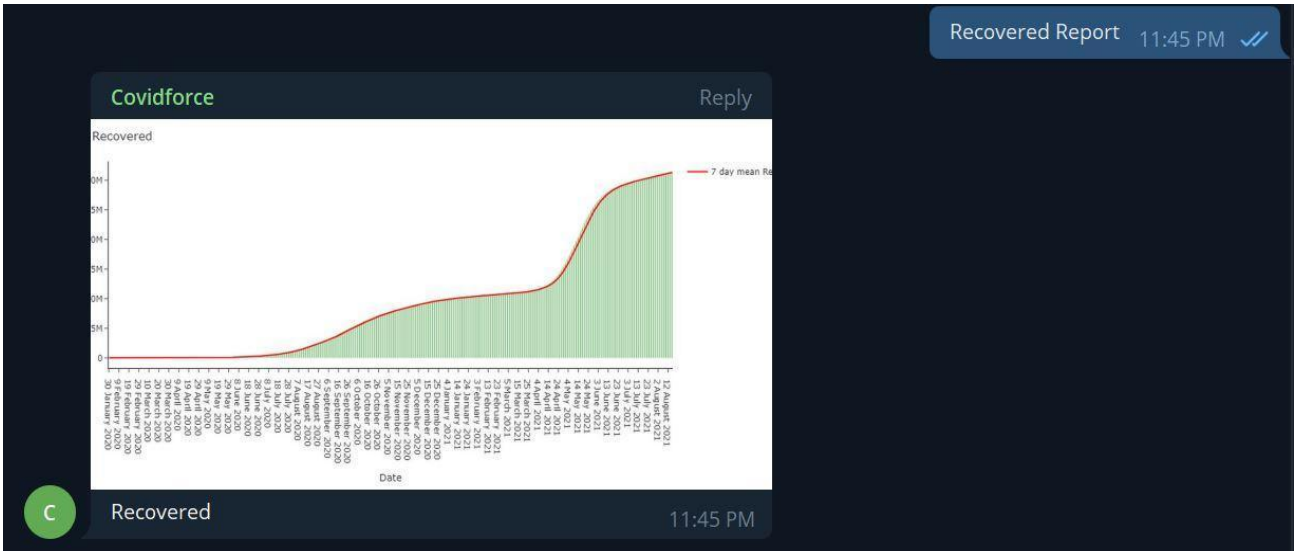
In [23]:
```
district_wise = district_wise[district_wise['State/UT']!='State Unassigned']
district_wise.sort_values('Confirmed', ascending= False).head(30).fillna(0).style\
                        .background_gradient(cmap='Blues_r',subset=["Confirmed"])\
                        .background_gradient(cmap='Greens_r',subset=["Recovered"])\
                        .background_gradient(cmap='Reds_r',subset=["Death"])\
                        .background_gradient(cmap='pink_r',subset=["Active"])
```

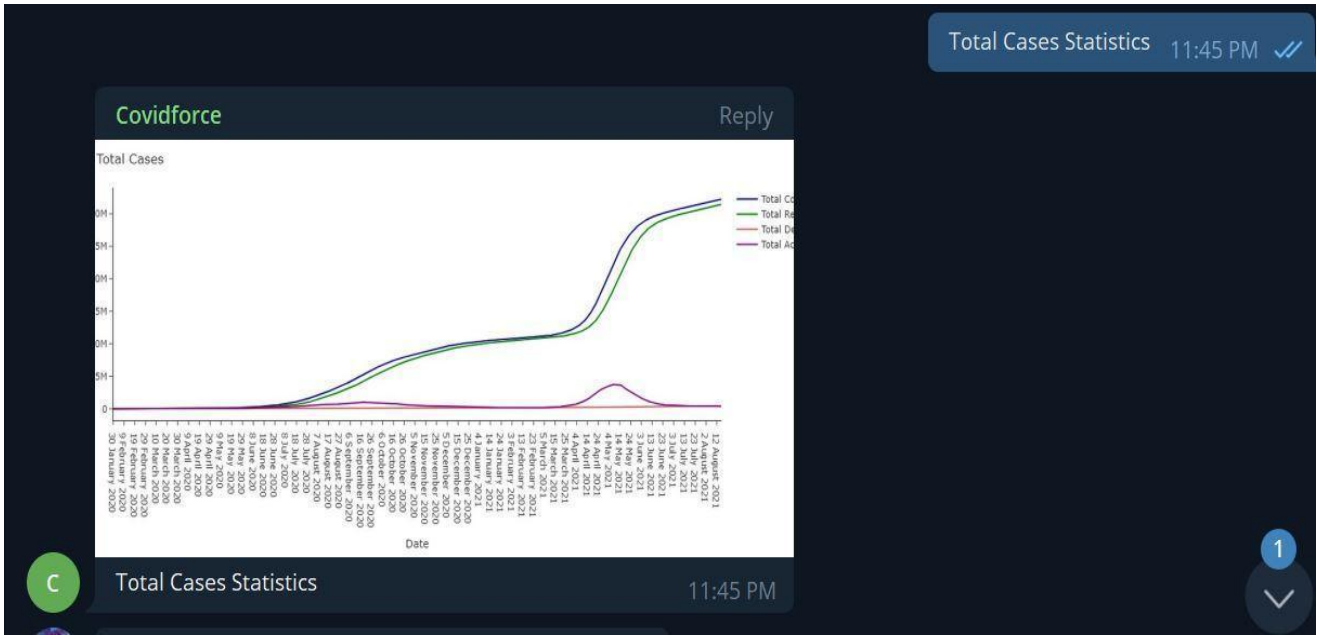
Out[23]:

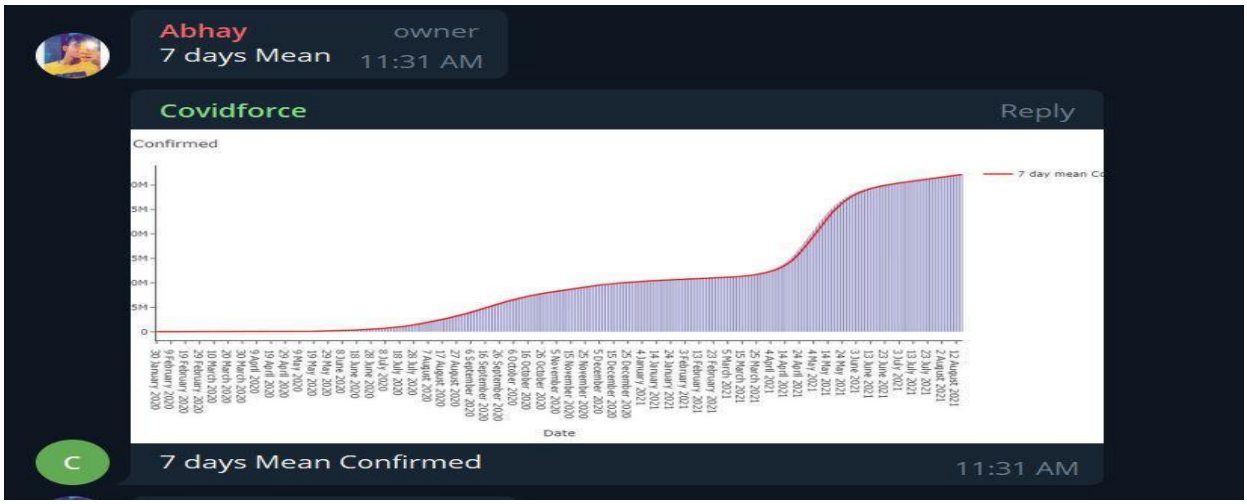
	State/UT	District	Confirmed	Recovered	Active	Death
160	Delhi	Unknown	1436101	1411327	-293	25067
289	Karnataka	Bengaluru Urban	1231474	1207260	8290	15923
361	Maharashtra	Pune	1099851	1066447	14419	18679
351	Maharashtra	Mumbai	738239	715660	4212	15968
368	Maharashtra	Thane	592025	574998	5901	11092
619	Tamil Nadu	Chennai	540300	529907	2048	8345
353	Maharashtra	Nagpur	493063	483591	264	9137
325	Kerala	Malappuram	440734	410437	28759	1508
318	Kerala	Ernakulam	430396	401576	26921	1830
356	Maharashtra	Nashik	403658	394257	839	8561
324	Kerala	Kozhikode	387573	362390	23271	1869
329	Kerala	Thrissur	355009	341779	11239	1965
328	Kerala	Thiruvananthapuram	343093	330216	9386	3376
766	West Bengal	North 24 Parganas	320162	314358	1213	4591
762	West Bengal	Kolkata	311183	305367	835	4981
335	Maharashtra	Ahmednagar	294758	283222	5231	6304
322	Kerala	Kollam	284264	280404	2412	1389
8	Andhra Pradesh	East Godavari	281384	277087	3065	1232
326	Kerala	Palakkad	269048	249691	17683	1652
711	Uttar Pradesh	Lucknow	238644	235939	54	2651
170	Gujarat	Ahmedabad	238022	234550	61	3411
317	Kerala	Alappuzha	237133	227678	8265	1150
7	Andhra Pradesh	Chittoor	234198	229637	2778	1783
323	Kerala	Kottayam	231995	223230	7989	762
620	Tamil Nadu	Coimbatore	231863	227403	2259	2201
365	Maharashtra	Satara	226151	214414	6239	5471
349	Maharashtra	Kolhapur	199490	189411	4427	5647
320	Kerala	Kannur	199236	189191	8847	1147
367	Maharashtra	Solapur	194066	183777	5207	4986
364	Maharashtra	Sangli	192215	180025	6964	5217

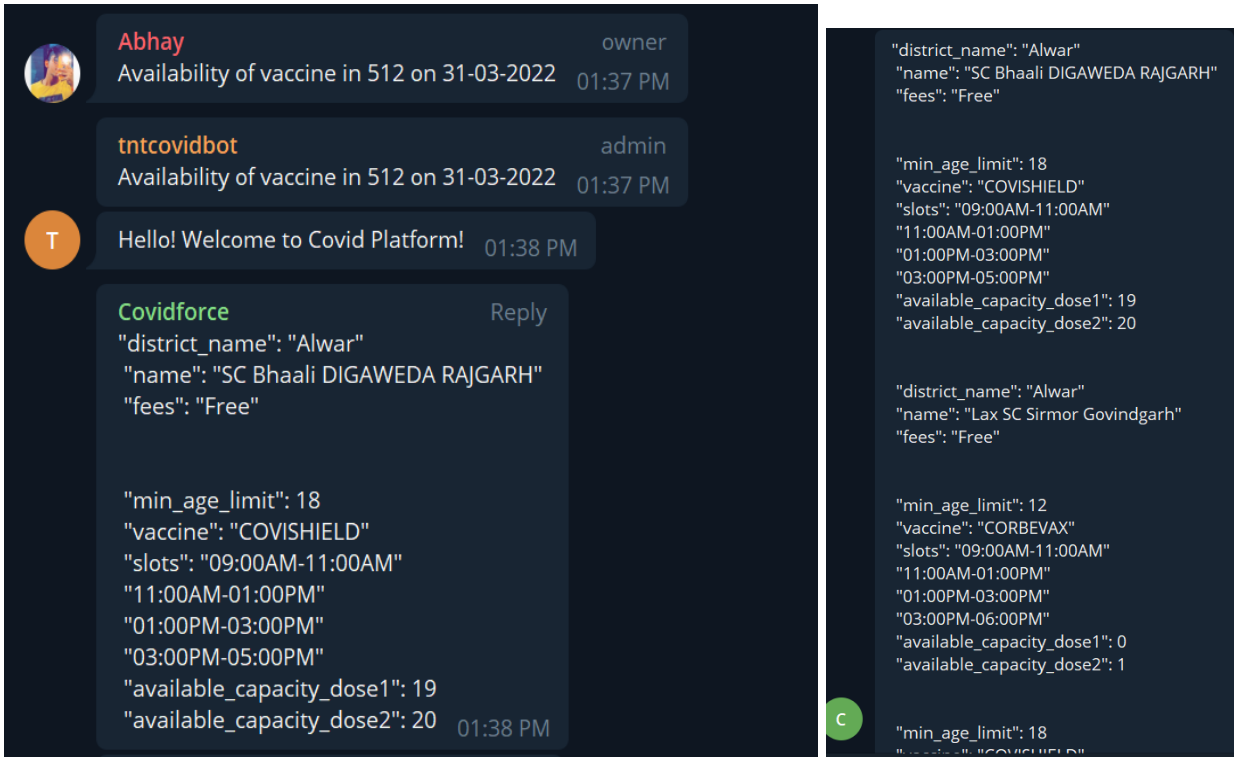
SECTION III: OUTPUT




Recovered Report 11:45 PM
Covidforce
Reply
Recovered
Date
Recovered
11:45 PM


Total Cases Statistics 11:45 PM
Covidforce
Reply
Total Cases
Date
Total Cases Statistics
11:45 PM


Abhay
owner
7 days Mean
11:31 AM
Covidforce
Reply
Confirmed
Date
7 days Mean Confirmed
11:31 AM



Demo video: [Link]

Github link : [Covid19IndiaTracker_bo]