

An Industrial Oriented Mini Project [CS704PC]
on
“COWIN VACCINE SLOT TRACKER”

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

Bachelor of Technology
in
Computer Science and Engineering
by
MS. M PRATHYUSHA LAHARI
(18261A0539)

Under the guidance of
MR. R. SRINIVAS
(Sr. Assistant Professor)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

Kokapet (V), Gandipet (M), Hyderabad.

Telangana - 500 075. (India)

2021-2022

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

GANDIPET, HYDERABAD – 500 075. Telangana

CERTIFICATE



This is to certify that the Industrial Mini Project entitled “**Cowin Vaccine Slot Tracker**” is being submitted by **M Prathyusha Lahari** bearing **Roll No: 18261A0539** in partial fulfillment for the award of **B.Tech in Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out under the supervision of Mr. **R Srinivas, Sr. Assistant Professor, Department of CSE.**

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Project Guide

Mr. R. Srinivas

Sr Asst. Professor, Dept. of CSE

Head of the Department

Dr. C. R. K. Reddy

Professor, Dept. of CSE

External Examiner

DECLARATION

This is to certify that the work reported in this project titled “**COWIN VACCINE SLOT TRACKER**” is a record of work done by me in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

M PRATHYUSHA LAHARI
(18261A0539)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project.

I would like to express my sincere thanks to **Dr. K. Jaya Sankar, Principal MGIT**, for providing the work facilities in the college.

I am also thankful to **Dr. C.R.K Reddy, Professor & Head of Department**, Dept. of Computer Science and Engineering for providing excellent infrastructure and a conducive atmosphere for completing this project successfully.

I am also extremely thankful to my Project Coordinators, **Dr. C.R.K Reddy, Professor & Head of Department, Dept. of CSE, Ms. B Prashanthi, Sr. Assistant Professor, Dept. of CSE and Ms. D Deepika, Assistant Professor, Dept. of CSE**, for their valuable suggestions and interest throughout the course of this project.

I would like to express my sincere gratitude and indebtedness to my Project Guide, **Mr. R Srinivas, Sr. Assistant Professor, Dept. of CSE** who has supported me throughout the project with patience and knowledge.

I convey my heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Finally, I would like to take this opportunity to thank my family for their support throughout the work. I sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

M PRATHYUSHA LAHARI (18261A0539)

TABLE OF CONTENTS

Certificate	i
Declaration	ii
Acknowledgements	iii
List of Figures	vi
List of Tables	vii
Abstract	viii
1. Introduction	1
1.1 Problem Definition	2
1.2 Existing System	2
1.3 Proposed System	2
1.4 Requirements Specification	3
1.4.1 Hardware Requirements	3
1.4.2 Software Requirements	3
1.5 Technology	3
1.5.1 Python	3
1.5.2 Visual Studio	4
1.5.3 Co-WIN Public API	4
2. Literature Survey	5
3. Methodology	9
3.1 Architecture	9
3.1.1 Bot Creation	10
3.1.2 Adding District-wise Details	13
3.1.3 Adding Features & Commands	14
3.2 Modules	15

3.3 Diagrammatic Representation	16
3.3.1 Use Case Diagram	17
3.3.2 Activity Diagram	18
3.3.3 Sequence Diagram	19
4. Testing and Results	20
4.1 Evaluation	20
4.2 Results	20
4.2.1 Visual Studio IDE Environment	21
4.2.2 Final Model	21
4.2.3 Output	22
5. Conclusion and Future Scope	24
5.1 Conclusion	24
5.2 Future Scope	24
Bibliography	25
Appendix	26

LIST OF FIGURES

Figure 1.3	Subscription and push-notifications service	2
Figure 3.1	System Architecture	9
Figure 3.2	General Training Work-flow	9
Figure 3.3	Search Bot-father in telegram	10
Figure 3.4	Bot-father info	10
Figure 3.5	Start Bot-father	11
Figure 3.6	Creation of Bot	12
Figure 3.7	Token of Bot	12
Figure 3.8	Use Case Diagram	17
Figure 3.9	Activity Diagram	18
Figure 3.10	Sequence Diagram	19
Figure 4.1	Visual Studio IDE Environment	20
Figure 4.2	Final Bot Window	21
Figure 4.3	Result-1	22
Figure 4.4	Result-2	23
Figure 4.5	Result-3	23

LIST OF TABLES

Table 2.1	Literature Survey	8
-----------	-------------------	---

ABSTRACT

As in India, Vaccination as not yet completed and only less than 50% have completed their second-dose vaccine only for ages 18+. also, there are approximately 30 crores+ children below 18 years in India are still not vaccinated. Mainly in Tier-2 Cities, semi-urban and rural areas, Still the availability of vaccine and vaccine-slots is not accurate. Slots in COWIN generally get available for a few seconds (for 18+ age-group) i.e., the slots get booked very fast.

The objective of this project is to create all in one telegram- bot solution for finding vaccination slots for the COVID-19 virus. Using the Telegram companion bot for finding slot quickly with the help of COWIN API. It Checks and alert for available slots for covid vaccines for selected pin codes and districts.

1. INTRODUCTION

Digital revolution offers instant information on millions of web sites. In plenty of available data, customers are interested to find information in context of appearance of an object or a certain property as soon as it is published by a given site. Some sites offer push notification to customers if they are subscribed to such a service. However, not all sites implement push service, and in these cases, customers frequently visit the web site and wait for an occurrence, such as sport result, rank, list, etc. In this paper we present how an intelligent software agent can realise this task instead of users that are manually visiting the web site and send information as if the web site has implemented a push service.

This novel idea is good for both customers and web site providers. Customers would subscribe to get the information as soon as it is available, and web site providers would like to enable push service without changes in the web site software, that might be very costly. Provision of this kind of notification can be realised via sophisticated mobile phone alerts, text messages, twitter or Facebook notification, or even supported by new operating systems like Maverik's OS or web sites such as Google notifications.

Not all sites provide push services and enable notifications. In fact, there are only few sites, such as sport news or similar information sites that send push information to customers. The new digital era initiates situations where, a lot of people are waiting for a notification to appear on certain web pages. For example, one would like to find out if there is a change in a given property associated to a certain key phrase. In this case the user is frequently visiting all those web sites looking for a given text or keyword. This results time-consuming for user. So, we use telegram qpush alerts for any subscribed-service.

1.1 Problem Definition

As in India, only 18+ had their vaccine dose and only 21.9% have completed their second dose vaccine. Children are still not vaccinated. Mainly in Tier-2 Cities, semi-urban and rural areas, Still the. availability of vaccine and slots is not accurate. COWIN Slots generally get available for a few seconds (for 18+ age-group) i.e., the slots get booked very fast.

1.2 Existing System

CoWIN (Covid Vaccine Intelligence Network) is an Indian government web portal for COVID-19 vaccination registration, owned and operated by India's Ministry of Health and Family Welfare. It displays booking slots of COVID-19 vaccine available in the nearby areas and can be booked on the website. Registration for the vaccination slots can be booked on the same day or a few days prior. This vaccination booking slots, in cowin portal are not available every time. So, all people need to check Cowin Vaccine Portal frequently, which is very time-consuming task.

1.3 Proposed System

The objective of this project is to create Telegram-bot solution for finding vaccination slots for the COVID-19 Vaccination. Use the companion bot for finding slot quickly with the help of COWIN API in Python. Checks and alert for available slots for covid vaccines for available pin codes and districts. The work-flow of Subscription and Push-notifications service is shown in Fig 1.3.

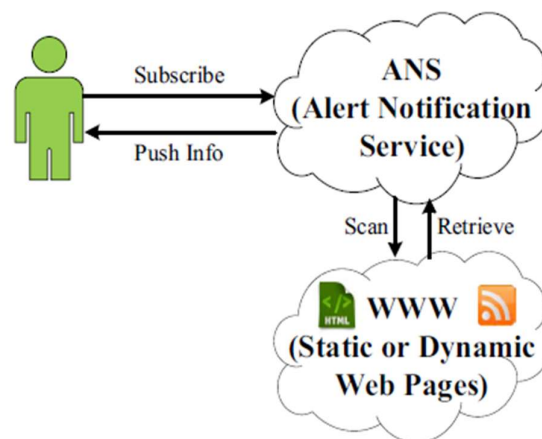


Fig 1.3. Subscription and push-notifications service

1.4 Requirements Specification

1.4.1 Hardware Requirements

- Processor : Intel core (i5 or higher)
- RAM : 4 GB
- ROM : 8 GB

1.4.2 Software Requirements

- Operating System : Windows/ MacOS
- Programming Language : Python
- Integrated Development Environment (IDE) : Visual Studio

1.5 Technology

1.5.1 Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

1.4.1.1 It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management

1.4.1.2 It supports multiple programming paradigms, including object- oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

1.4.1.3 Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

1.5.2 Visual Studio

1.5.2.1 Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, windows presentation foundation, Window Store and Microsoft Silverlight. It can produce both native code and managed code.

1.5.2.2 Features of visual code studio are code editor, Debugger, Designer.

1.5.2.3 Visual Studio includes a host of visual designers to aid in the development of applications. These tools include:

1. Windows Forms Designer
2. WPF Designer
3. Web Designer/Development
4. Class Designer
5. Data designer
6. Mapping designer

1.5.3 Co-WIN Public API

Co-WIN Public APIs allow any third-party application to access certain un-restricted information, that can be shared with its users. This is limited only to read access in Co-WIN. The extent of access will be limited and in case of any misuse impacting the performance of Co-WIN solution will result in blocking any such application and entities as per the policies of MoHFW and taking any other appropriate action in accordance with law. The appointment availability data is cached and may be up to 5 minutes old. Further, these APIs are subject to a rate limit of 100 API calls per 5 minutes per IP. Please consider these points while using the APIs in your application.

2. LITERATURE SURVEY

Telegram launched a platform for third-party developers to create bots. Bots are Telegram accounts operated by programs. They can respond to messages or mentions, can be invited into groups and can be integrated into other programs. It 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.

Telegram also offers an API that allows developers to create bots, which are accounts controlled by programs. Such bots are used, among other things, to emulate and play old games in the app and inform users about vaccine availability for COVID-19.

1. Automated Motion Detection Security System Notifier using Raspberry Pi with Telegram-IEEE (Irina Mazwin Hazri, Mus'ab Sahrim, Wan Zakia) -2020.

Proposed Motion Detection Security System Notifier using Raspberry Pi with Telegram is as the name suggests. Using Raspberry Pi, a motion detection system is developed through the method of background subtraction, which works by subtracting the static from the background frame from the constantly changing foreground. The change of value in noise and pixel density between the background frame and the foreground indicates motion. Each time a motion is detected, the system will automatically send the frame to the user via Telegram to notify that there has been

2. Realtime monitoring of Bitcoin prices on several Cryptocurrency markets using Web API, Telegram Bot, MySQL Database, and PHP-Cronjob -IEEE (Rizky Parlika, Pratama Wirya Atmaja) – 2020.

The purpose of this research is to test hundreds of API data shared by hundreds of Cryptocurrency Markets and as a result we have succeeded in extracting the latest bitcoin price data from 20 Crypto Markets via shared APIs. This data will then be stored in a MySQL database on hosting automatically when a condition is reached. This data can then be mined continuously using Cronjob, and to facilitate the reading of data that has been collected, we use a telegram bot as well as using a web-based application.

3. Reporting Sleepy Driver into Channel Telegram via Telegram Bot -IEEE (Choirul Huda, Fitra Abdurrachman Bachtiar, Ahmad Afif Supianto) -2019

This study proposed a sleepy driver detection and information notification system implemented on smartphones. The system is capable to transmit personal information of a sleepy driver directly on a Telegram messaging app. It aims to identify a report regarding driver name, car type, and license plate and even current location. Some experiments that have been done prove that a system worthy to transmit information into Channel within Telegram Messenger precisely and obtain an accuracy of 89% using SVM.

4. Botanicum: a Telegram Bot for Tree Classification- Research Gate (Daria Korotaeva, Maksim Khlopotov, Anastasiia Makarenko)-2018.

The article presents Botanicum - a telegram bot for tree classification based on the image of a leaf. In this work, we used our previously developed classification model which underlies the bot. Botanicum can identify 20 different tree species typical for Russian flora. The process of user interaction includes a simple chat and a short instruction on how to take a photo. After image verification, the bot either outputs error description or the classification results.

5. Chatting with Arduino platform through Telegram Bot – IEEE(Juan Carlos de Oliveira; Danilo Henrique Santos) -2016.

This article seeks to introduce the integration between the smartphone messenger Telegram and the Arduino platform using Telegram Bots, allowing common people to create hardware prototypes and communicate with them using the same tool applied in the communication with other people.

Table 2.1: Literature survey

S.No	Name of Author	Title of Paper	Year	Method Used
1.	Irina Mazwin Hazri, Mus'ab Sahrim, Wan Zakia	Automated Motion Detection Security System Notifier using Raspberry Pi with Telegram	2020	Raspberry Pi senses and sends alerts using telegram.
2.	Rizky Parlika, Pratama Wirya Atmaja	Realtime monitoring of Bitcoin prices on several Cryptocurrency markets using Web API, Telegram Bot, MySQL Database, and PHP-Cronjob	2020	Shared-Web API from stock-exchange websites like Coin Gecko using telegram bot.
3.	Choirul Huda, Fitra Abdurrachman Bachtiar, Ahmad Afif Supianto	Reporting Sleepy Driver into Channel Telegram via Telegram Bot	2019	Linear SVM and Web API using telegram bot.
4.	Daria Korotaeva, MaksimKhlopoto v, Anastasiia Makarenko .	Botanicum: a Telegram Bot for Tree Classifications	2018	Classification and Feature engineering

5.	Santanu Halder; AbulHasnat; D.	Chatting with Arduino platform through Telegram Bot	2016	Arduino
----	-----------------------------------	---	------	---------

3. METHODOLOGY

3.1 Architecture

The objective is to alert via telegram notification of available slots from cowin portal. For this we create a telegram-bot and update token details and using different python libraries, we send API requests to cowin portal to retrieve slots. The System Architecture and general training work-flow can be seen in below figures 3.1 and 3.2 Respectively.

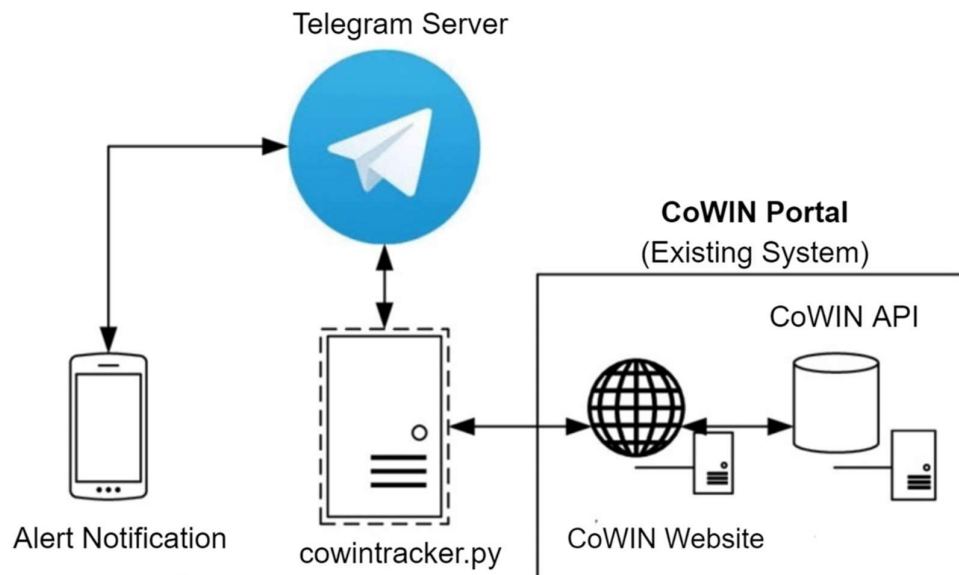


Fig. 3.1 System Architecture

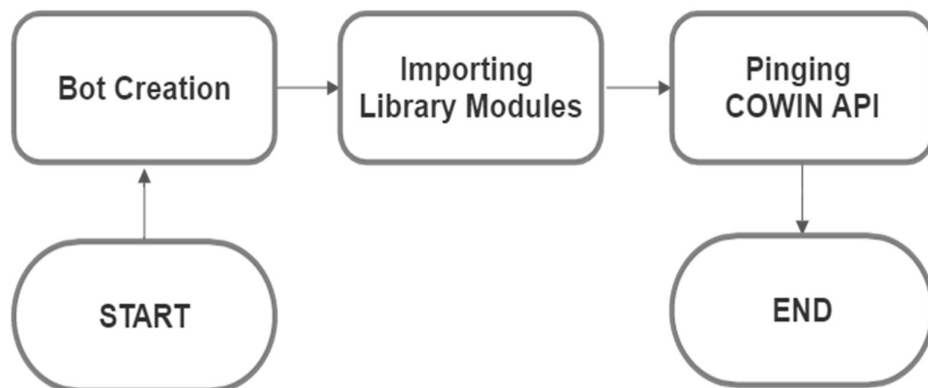


Fig. 3.2 General Training Work-flow

3.1.1 Bot Creation:

The first step is Creating Telegram bot. A chatbot is an automated multifunctional assistant, that can receive and send triggered messages, your bot can save information as variables for future usage. To set up a new bot, we need to register our bot first before using it. That way, we get the token to access the Telegram API.

1. To create a chatbot on Telegram, you need to contact the BotFather , which is essentially a bot used to create other bots. Click on the search icon in Telegram, then, type @botfather in the search bar.

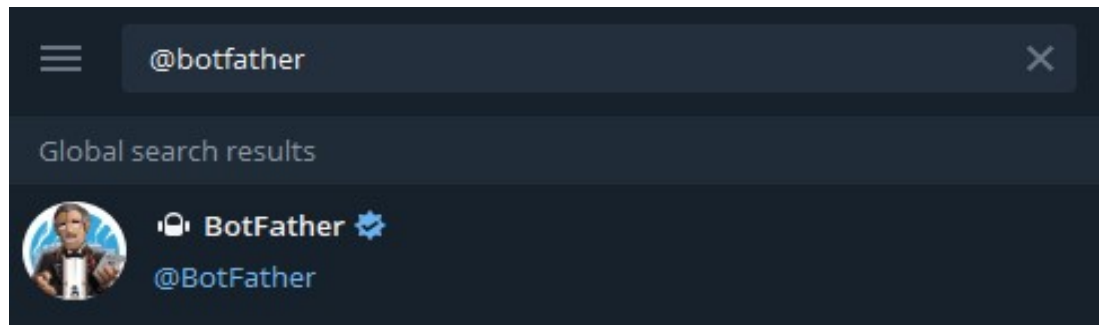


Fig. 3.3 Search Bot-father in telegram

2. Bot Father is the official bot created by Telegram to facilitate bot creation. Now, click on the start button to start a conversation.

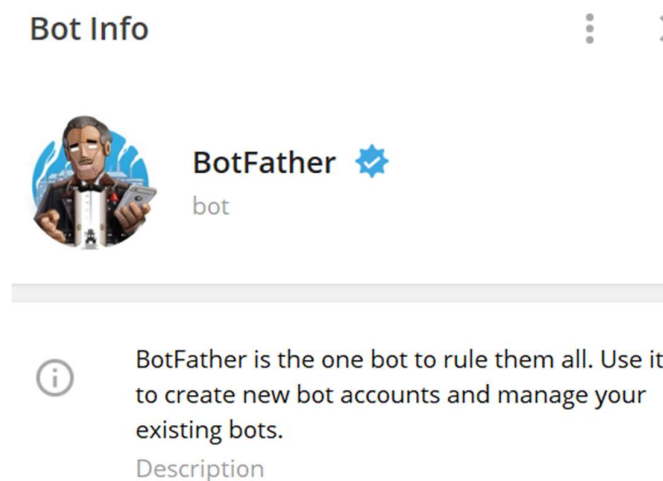


Fig. 3.4 Bot-father info

•

- Type /start to get started.

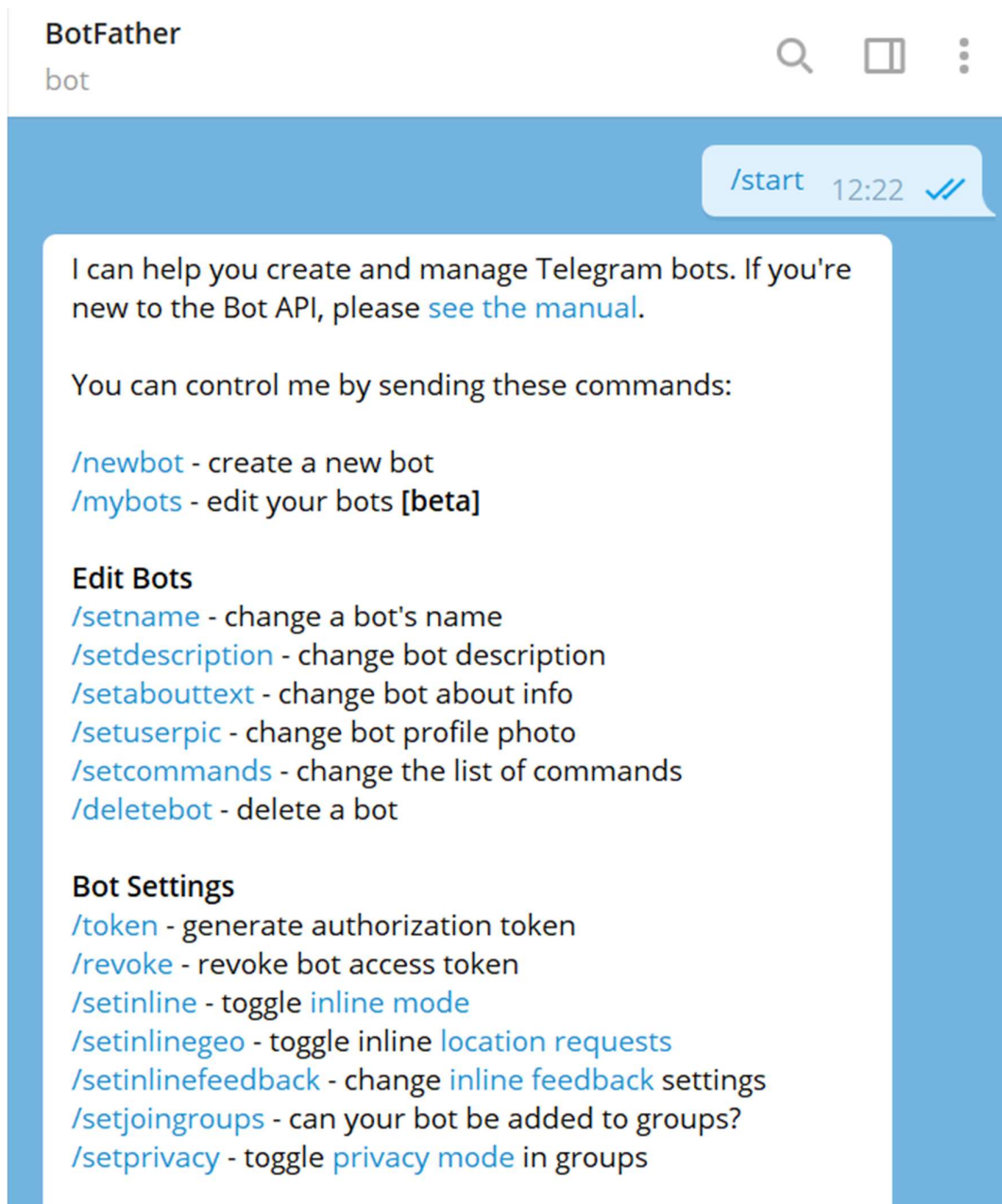


Fig. 3.5 Start Bot-father

3. Next, we create the bot by running `/new bot` command. we type in our preferred name and username for the bot.

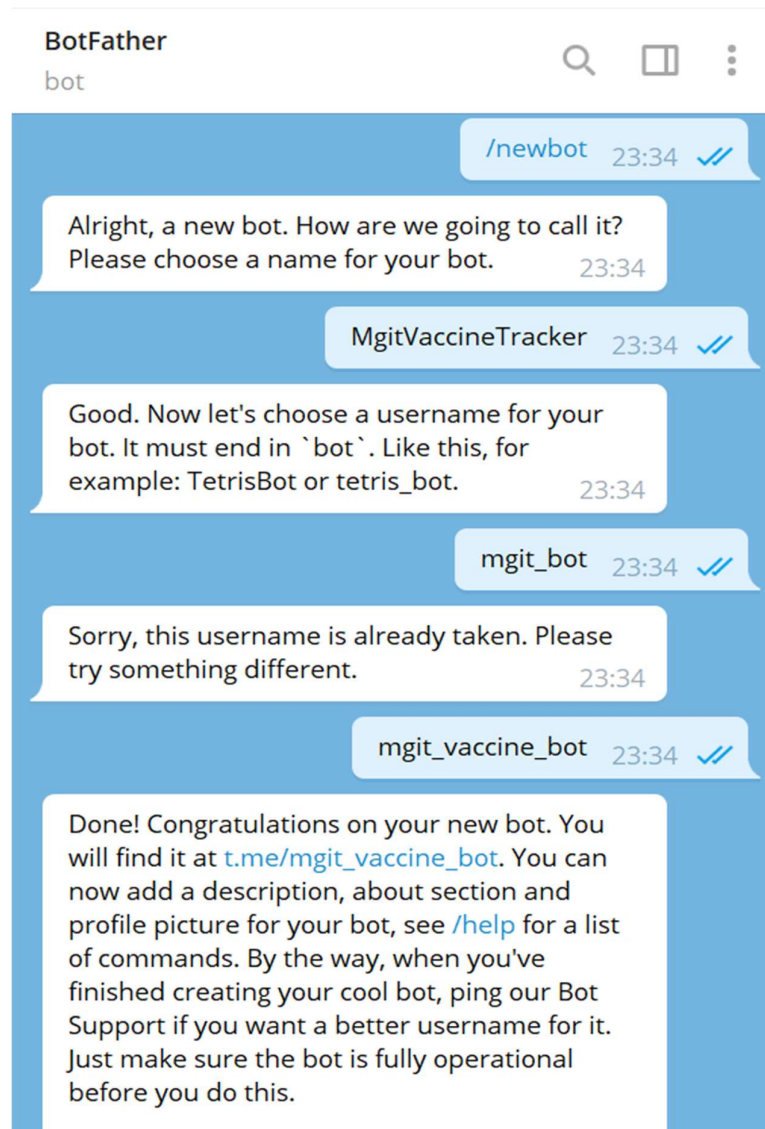


Fig. 3.6 Creation of Bot

4. Now, we copy our access token and save it somewhere. Remember to properly secure the access token, as it serves as your bot password.

Use this token to access the HTTP API:
[2105638884:AAEKPoWvEACxp1XeCTFyp8B3NbjxjTS8ycA](#)
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

Fig. 3.7 Token of Bot

3.1.2 Adding District-wise Details

Every District name is Mapped with a Unique District Code. Based on District name, COWIN API pings web portal with corresponding district codes.

DistrictName	DistrictCode
Adilabad	582
Bhadradri Kothagudem	583
Hyderabad	581
Jagtial	584
Jangaon	585
Jayashankar Bhupalpally	586
Jogulamba Gadwal	587
Kamareddy	588
Karimnagar	589
Khammam	590
Mahabubabad	592
Mahabubnagar	593
Mancherial	594
Medak	595
Medchal	596
Mulugu	612
Nagarkurnool	597
Nalgonda	598
Narayanpet	613
Nirmal	599
Nizamabad	600
Peddapalli	601
Rangareddy	603
Siddipet	605
Suryapet	606
Rajanna Sircilla	602
Sangareddy	604
Kumuram Bheem	591
Vikarabad	607
Wanaparthi	608
Warangal(Rural)	609
Warangal(Urban)	610
Yadadri Bhuvanagiri	611

3.1.3 Adding Features & Commands.

To get vaccine slot details, we need to add some features to BoT, so that user can easily interact with application.

- **Required Features**

1. Checks in Pincode for slots.
2. Checks in District for slots.
3. Specify name of vaccine, address, date in messages
4. Checks for Age requirement
5. Checks for Dose type (I/II)

- **Commands**

So,Based on given Features,we give commands to bot,so user access slot details based on his preference of age,area,dose etc.,

1. Pincode - Check for slots in given pin code.
Command - **/pincode pin**
2. District - Check for slots in given district.
Command - **/district district_name**
3. BotPincode - Starts a bot for checking available slots in selected pincode and age and dose
Command - **/botpincode pin age dose**
4. BotDistrict - Starts a bot for checking available slots in selected district and age and dose
Command - **/botdistrict district_name age dose**

3.2 Modules

The first step is the importing of module from python packages. below are some of packages-

- **telegram.ext package**

1. **telegram.ext.Updater**

Its purpose is to receive the updates from Telegram and to deliver them to said dispatcher. It also runs in a separate thread, so the user can interact with the bot, for example on the command line. The dispatcher supports handlers for different kinds of data: Updates from Telegram, basic text commands and even arbitrary types. The updater can be started as a polling service or, for production, use a webhook to receive updates.

2. **telegram.ext.Dispatcher**

This class dispatches all kinds of updates to its registered handlers.

- i. **run_async** : Queue a function to be run asynchronously. Exceptions raised by the function will be handled by the error handlers.

- **datetime package**

The datetime module supplies classes for manipulating dates and times. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

- i. **timedelta** : A duration expressing the difference between two date, time, or datetime instances to microsecond resolution.

- **time package**

This module provides various time-related functions.

- i. **sleep** : Suspend execution of the calling thread for the given number of seconds. The argument may be a floating point number to indicate a more precise sleep time. The actual suspension time may be less than that requested because any caught signal will terminate the sleep() following execution of that signal's catching routine.

3.3 Diagrammatic Representation

UML Diagrams:

The Unified Modelling Language (UML) is a standard language for writing software blueprints. The UML is a language for:

- **Specifying:** It is just like a blueprint created by an architect prior to the construction.
- **Visualizing:** Visualizing is concerned with deep analysis of system to be constructed.
- **Constructing:** Modelling also provide us mechanism which are essential while constructing a system.
- **Documenting:** Finally, modelling justifies its importance by applying all its credentials to be bounded in a piece of paper referred as document.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modelling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system

3.3.1 Use Case Diagram

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed together its functionalities use cases are prepared and actors are identified.

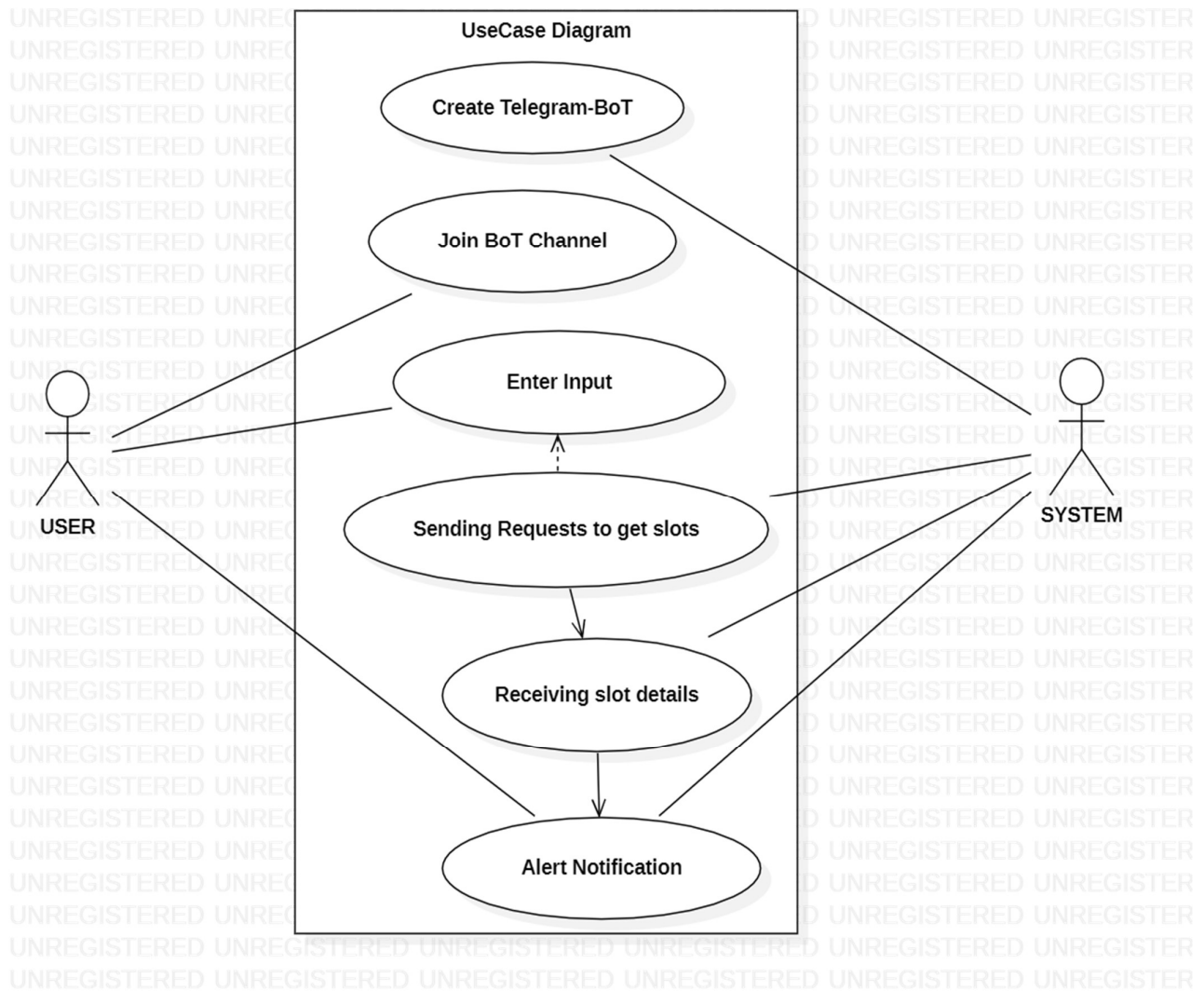


Fig. 3.8. Use Case Diagram

In the above figure Figure 3.8, the user joins the BoT Channel created by System. After Joining Bot Channel, user enters input. While, the system runs python script and sends API requests to COWIN Portal to receive slot details. This slot details are sent as telegram notification to user.

3.3.2 Activity Diagram:

Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another.

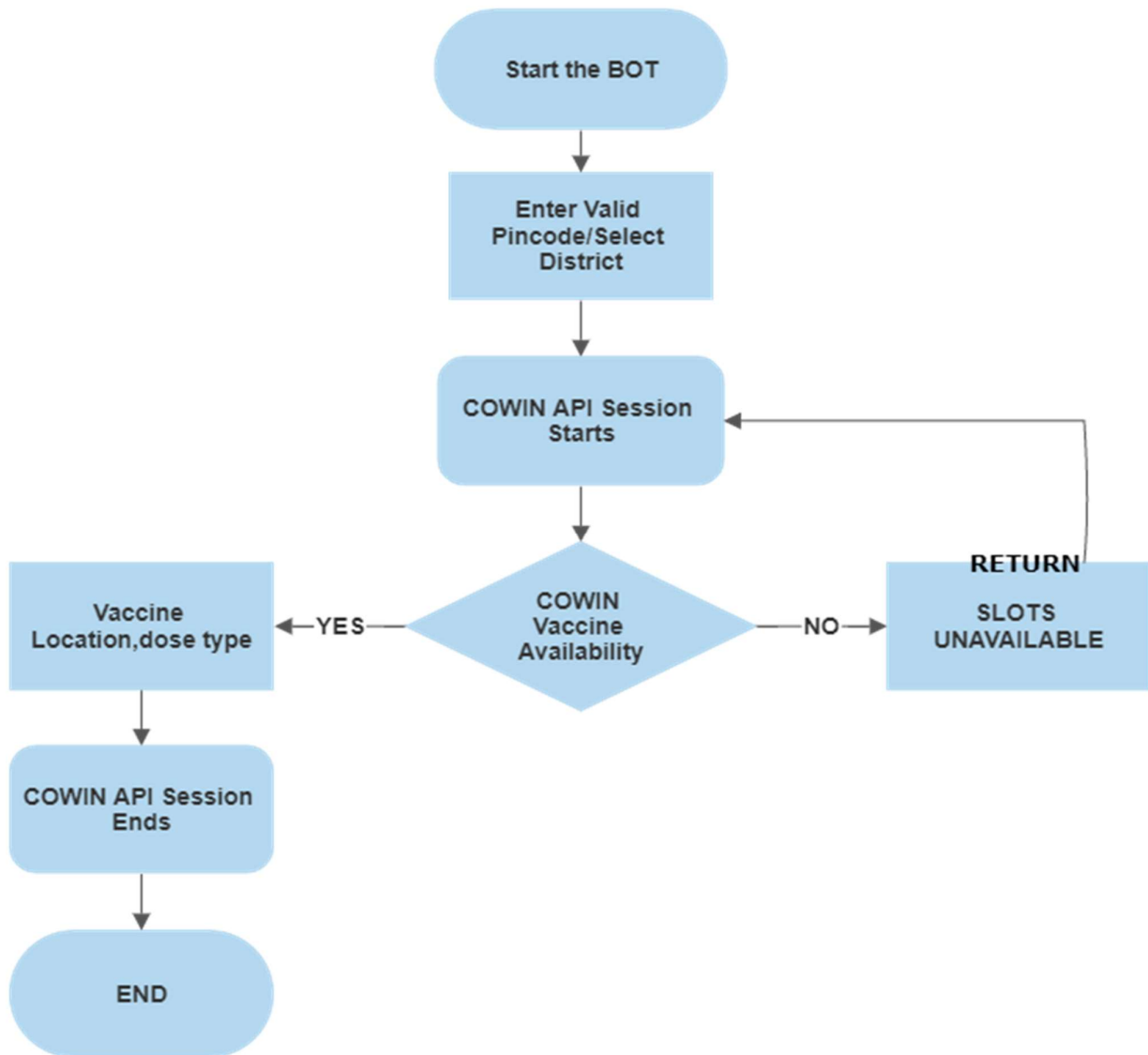


Fig. 3.9. Activity Diagram

The flow of control in this project can be observed in the Figure 3.9. Where bot started and COWIN API Session starts and based on vaccine slot availability, slot details are retrieved by COWIN API.

3.3.3 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

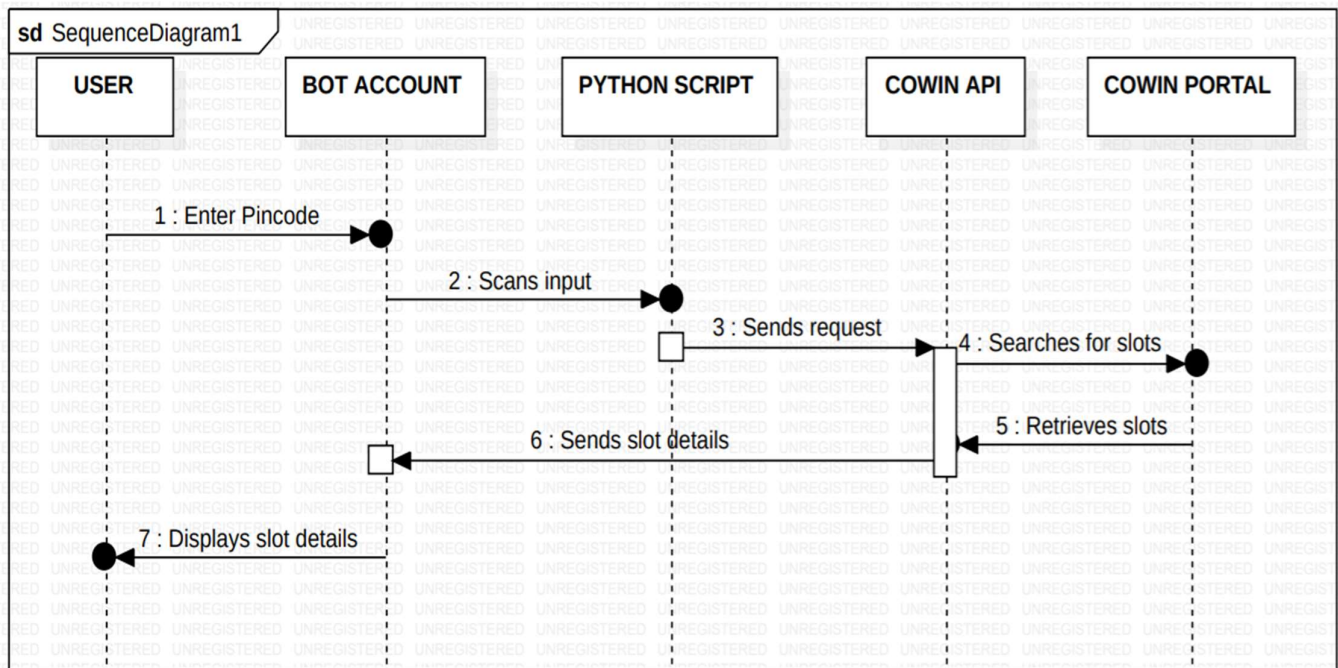


Fig. 3.10. Sequence Diagram

The Figure 3.10 shows the sequence followed in the project, where the user enters a pincode, then bot scans input. Based on input, server executing python script, sends API request to COWIN Portal to retrieve slots. This slot details are sent to bot via API and displayed to user.

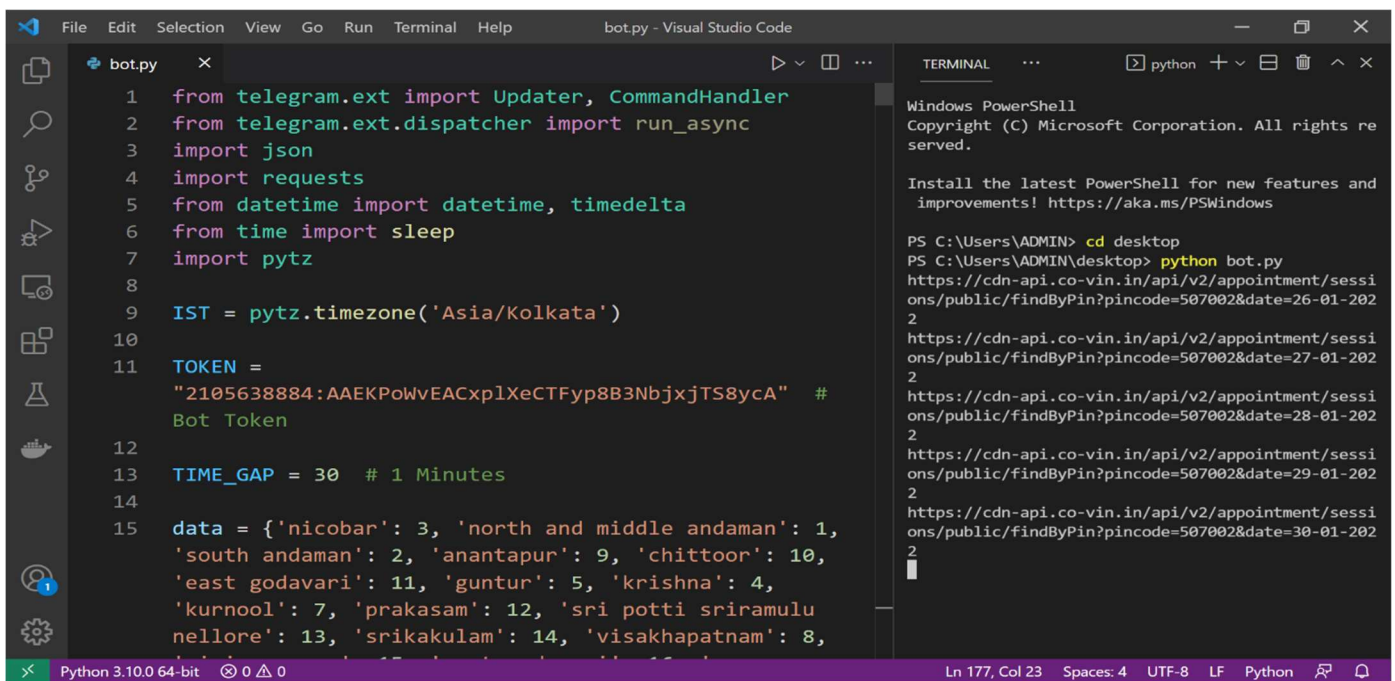
4. TESTING AND RESULTS

4.1 Evaluation

- First of all we need to import the required modules present in the python packages like datetime, telegram.ext .To use telegram-bot, we need to install some modules using 'pip command' in terminal.This module is useful to send alerts to bot by running python script in server
- .install python-telegram-bot with: `$ pip install python-telegram-bot`

4.2 Results

4.2.1 Visual Studio IDE Environment



The screenshot displays the Visual Studio IDE interface. The main editor window shows a Python script named `bot.py` with the following code:

```
1 from telegram.ext import Updater, CommandHandler
2 from telegram.ext.dispatcher import run_async
3 import json
4 import requests
5 from datetime import datetime, timedelta
6 from time import sleep
7 import pytz
8
9 IST = pytz.timezone('Asia/Kolkata')
10
11 TOKEN =
12 "2105638884:AAEKPoWVEACxplXeCTFyp8B3NbjxjTS8ycA" # Bot Token
13
14 TIME_GAP = 30 # 1 Minutes
15
16 data = {'nicobar': 3, 'north and middle andaman': 1,
17         'south andaman': 2, 'anantapur': 9, 'chittoor': 10,
18         'east godavari': 11, 'guntur': 5, 'krishna': 4,
19         'kurnool': 7, 'prakasam': 12, 'sri potti sriramulu
20         nellore': 13, 'srikakulam': 14, 'visakhapatnam': 8,
```

The TERMINAL window on the right shows the execution of the script in a Windows PowerShell environment. It displays the command `python bot.py` and the resulting output, which consists of a series of API calls to a service, each returning a JSON response with a status of 200.

Fig.4.1 Visual Studio IDE Environment

As shown in the above figure 4.1, it shows the environment for the implementation of the working code.

4.2.2 Final Model

After creating the Telegram-Bot Chat, we can see a text-box to enter the input where we have to enter the pincode, district name or any city name, for which we want to know the slot details of any given age, dose . The final model will be as shown in Fig 4.2,

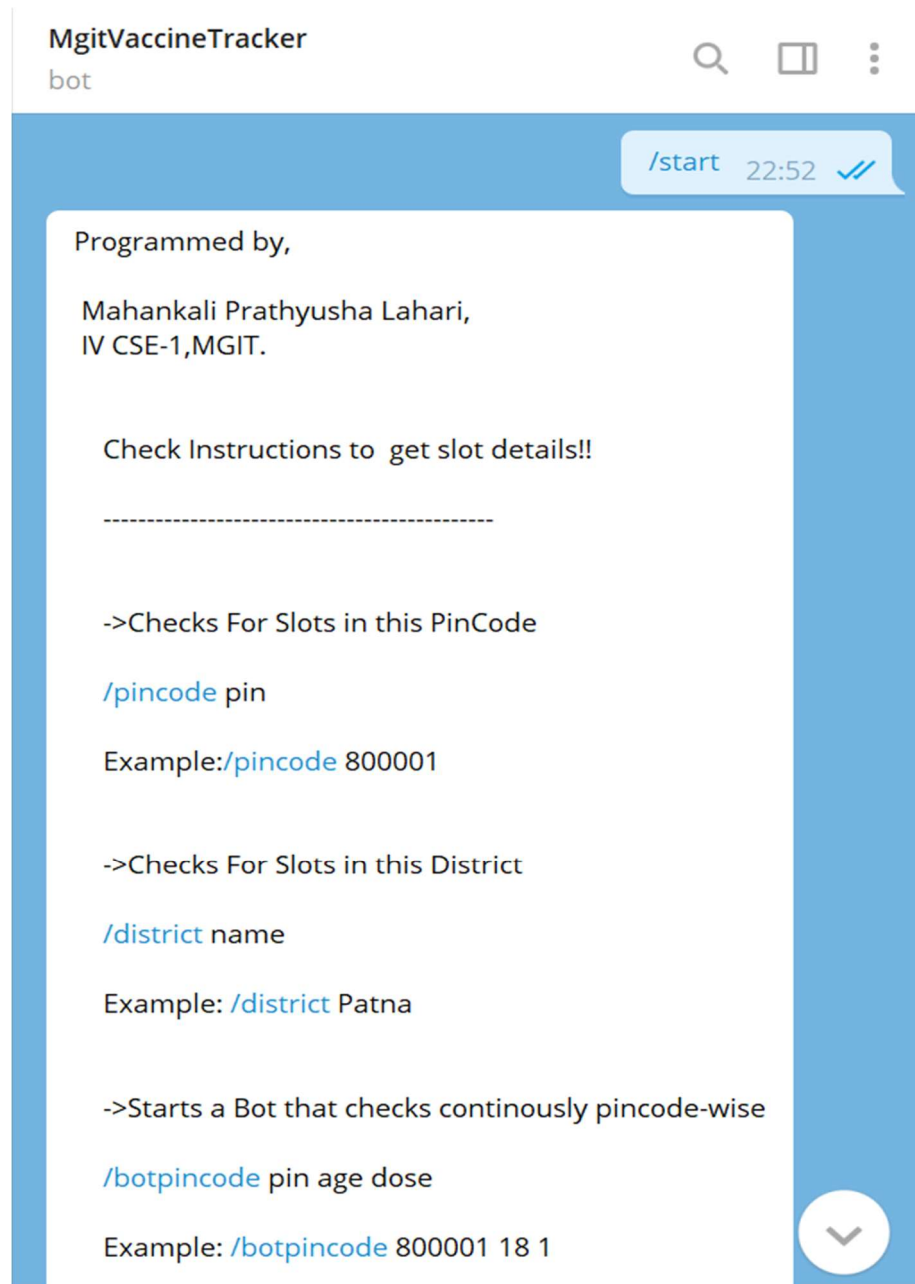


Fig 4.2 Final Bot Window

4.2.3 Output

After Entering the command in the input box, we can get the slot details at given pincode,district,city. Like this we can find the information of slots. The outputs are shown below

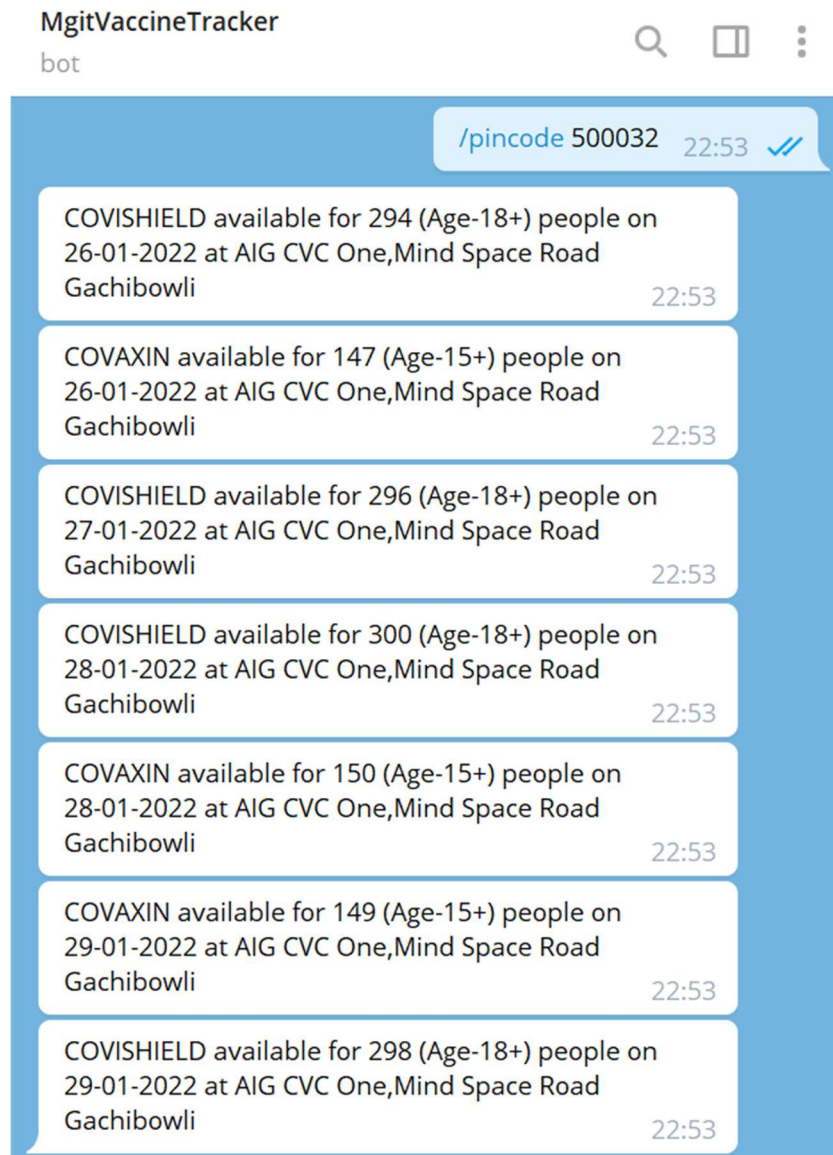


Fig.4.3 Result-1

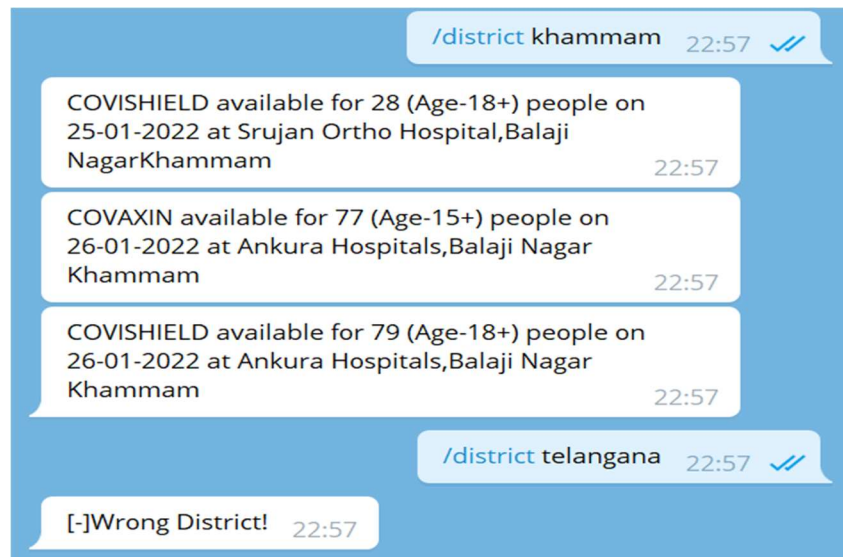


Fig.4.4 Result-2

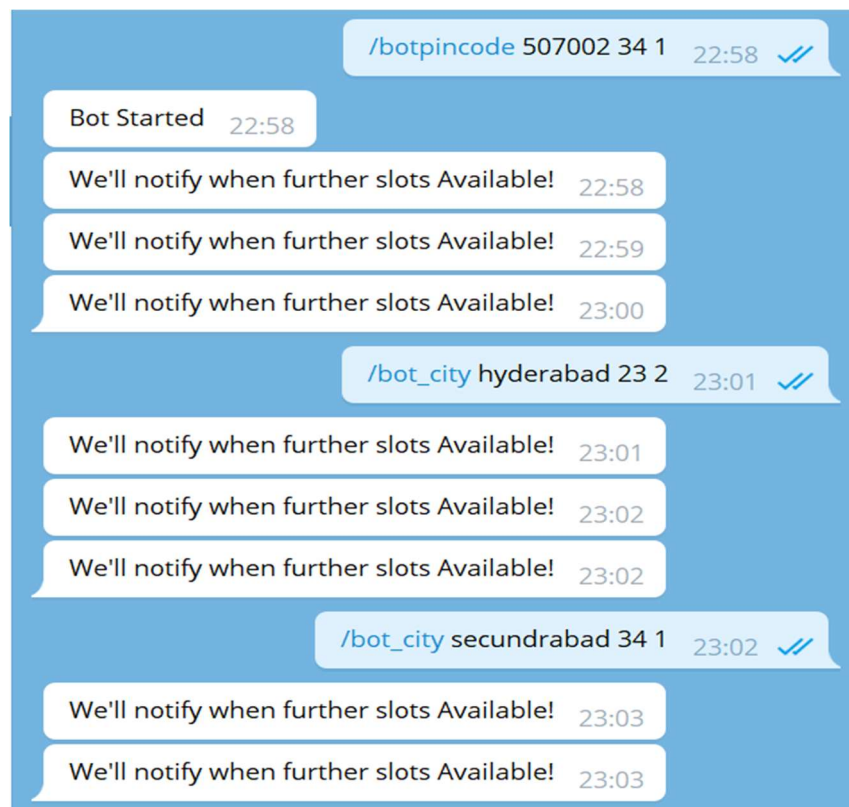


Fig.4.5 Result-3

Figures 4.3, 4.4, 4.5 display the final output on the different types of inputs. For Input of pincode , district name and city name with age and dose details, output is displayed with vaccine name, total no. of doses available for given age, hospital name and address.

5. CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION:

This project has demonstrated the efficacy and potential of using python libraries to create a BoT application for finding slots of vaccines which is very useful to the people who are not vaccinated and unable to find available slots for desired vaccine and based on their age, place, preference. This project is also very useful for the people who stays in rural areas who don't have proper idea on finding and booking slots in cowin portal, which is the major drawback. It's a time taking process which will waste our time in waiting for availability of vaccine slots. Registration for the vaccination slots can be booked on the same day or a few days prior. This vaccination booking slots, in cowin portal are not available every time. Slots in COWIN generally get available for a few seconds (for 18+ age-group) i.e., the slots get booked very fast. For this purpose this application is very useful by which we can easily find the vaccine slots for different age-groups at different pincodes, cities, districts.

5.2 FUTURE SCOPE:

There are many ways in which the performance of this model can be enhanced in the future. We can add more features like booking of slots after finding their availability. Booking of slots can be done by sending OTP to users and registering in cowin portal. This will makes the process further easier and convenient for users .OTP Registration is main application, which can be further implemented as extension of this application. The approach specified in this project can be also used not only for alert notifications, but also for any Internet-based transaction, allowing the cloud providers to upgrade conventional solutions without redeveloping the solution, but adding new functionalities by inserting an intelligent agent between the user and server. Those agents will act as a service, based on a scalable cloud solution.

BIBLIOGRAPHY

- [1] Rizky Parluka, “Realtime monitoring of Bitcoin prices on several Cryptocurrency markets using Web API, Telegram Bot, MySQL Database, and PHP-Cronjob”, October, 2020.
- [2] Irina Mazwin Hazri, “Automated Motion Detection Security System Notifier using Raspberry Pi with Telegram ”, July 2020.
- [3] Choirul Huda, “Reporting Sleepy Driver into Channel Telegram via Telegram Bot”, September, 2019.
- [4] Daria Korotaeva and Maksim Khlopotov , “Botanicum: a Telegram Bot for Tree Classification”, May 2018.
- [5] Juan Carlos de Oliveira and Danilo Henrique Santos , “Chatting with Arduino platform through Telegram Bot”, September, 2016.
- [6] Marjan Gusev and Sasko Ristov, “Alert Notification as a Service”, May 2014.

APPENDIX

Import Required Library

```
from telegram.ext import Updater, CommandHandler
from telegram.ext.dispatcher import run_async
import json
import requests
from datetime import datetime, timedelta
from time import sleep
import pytz
```

setting timezone to Indian Standard time

```
IST = pytz.timezone('Asia/Kolkata')
```

telegram-bot token

```
TOKEN = "2105638884:AAEKPoWvEACxplXeCTFyp8B3NbxxjTS8ycA" # Bot Token
```

#sleep timer

```
TIME_GAP = 60 # 1 Minute
```

#data of district codes

```
data = {'anantapur': 9, 'chittoor': 10, 'east godavari': 11, 'guntur': 5, 'krishna': 4, 'kurnool': 7, 'prakasam': 12, 'sri
potti sriramulu nellore': 13, 'srikakulam': 14, 'visakhapatnam': 8, 'vizianagaram': 15, 'west godavari': 16, 'ysr
district, kadapa (cuddapah)': 6, 'adilabad': 582, 'bhadradri kothagudem': 583, 'hyderabad': 581, 'jagtial': 584,
'jangaon': 585, 'jayashankar bhupalpally': 586, 'jogulamba gadwal': 587, 'kamareddy': 588, 'karimnagar': 589,
'khammam': 590, 'kumuram bheem': 591, 'mahabubabad': 592, 'mahabubnagar': 593, 'mancherial': 594,
'medak': 595, 'medchal': 596, 'mulugu': 612, 'nagarkurnool': 597, 'nalgonda': 598, 'narayanpet': 613, 'nirmal':
599, 'nizamabad': 600, 'peddapalli': 601, 'rajanna sircilla': 602, 'rangareddy': 603, 'sangareddy': 604, 'siddipet':
605, 'suryapet': 606, 'vikarabad': 607, 'wanaparthy': 608, 'warangal(rural)': 609, 'warangal(urban)': 610, 'yadadri
bhuvanagiri': 611}
```

```
headers = {"Accept": "application/json", "Accept-Language": "hi_IN",
           "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/70.0.3538.77 Safari/537.36"}
```

```

def district(update, context):
    flag = 0
    da = str(' '.join(context.args[:])).lower()

    for day in range(0, 5):
        d3 = (datetime.now(IST) + timedelta(days=day)).strftime("%d-%m-%Y")
        d4 = data.get(da)
        if(d4 == None):
            update.message.reply_text("[-]Wrong District!\n")
            return
        url = f"https://cdn-api.co-
vin.in/api/v2/appointment/sessions/public/findByDistrict?district_id={d4}&date={d3}"
        print(url)
        try:
            response = requests.get(url, headers=headers)
        except:
            update.message.reply_text("API Error!\n")
            return
        if response.status_code == 200:
            json_data = json.loads(response.text)
            if len(json_data["sessions"]):
                for session in json_data["sessions"]:
                    if session["available_capacity"] > 0:
                        msg = f'{session["vaccine"]} available for {session["available_capacity"]} (Age-
{session["min_age_limit"]}+) people on {session["date"]} at {session["name"]}, {session["address"]}\n'
                        update.message.reply_text(msg)
                        flag = 1
            if flag == 0:
                update.message.reply_text("No Slots found check later!")

def pincode(update, context):
    flag = 0
    if(len(context.args) == 1):
        pin = str(context.args[0])
    else:
        update.message.reply_text(

```

```

        "Please provide pin or type Command properly!\n")
    return
for day in range(0, 5):
    d3 = (datetime.now(IST) + timedelta(days=day)).strftime("%d-%m-%Y")
    url = f"https://cdn-api.co-
vin.in/api/v2/appointment/sessions/public/findByPin?pincode={pin}&date={d3}"
    print(url)
    try:
        response = requests.get(url, headers=headers)
    except:
        update.message.reply_text("API Error!\n")
        return
    if response.status_code == 200:
        json_data = json.loads(response.text)
        if len(json_data["sessions"]):
            for session in json_data["sessions"]:
                # print(session)
                if session["available_capacity"] > 0:
                    msg = f'{session["vaccine"]} available for {session["available_capacity"]} (Age-
{session["min_age_limit"]}+) people on {session["date"]} at {session["name"]}, {session["address"]}\n'
                    update.message.reply_text(msg)
                    flag = 1
        if flag == 0:
            update.message.reply_text("No slots found check later!")

```

```

def botpincode(update, context):

```

```

    if(len(context.args) == 3):
        pin = str(context.args[0])
        dosec = str(context.args[2])
        dose = 'available_capacity_dose'+dosec
        age = int(context.args[1])
    else:
        update.message.reply_text("Invalid Example- /botpincode 800001 18 1")
        return
    update.message.reply_text("Bot Started\n")

```

```

while True:
    for day in range(0, 5):
        d3 = (datetime.now(IST) + timedelta(days=day)).strftime("%d-%m-%Y")

        url = f"https://cdn-api.co-
vin.in/api/v2/appointment/sessions/public/findByPin?pincode={pin}&date={d3}"
        print(url)
        try:
            response = requests.get(url, headers=headers)
        except:
            update.message.reply_text("API Error!\n")
            return
        if response.status_code == 200:
            # print(response.text)
            json_data = json.loads(response.text)
            if len(json_data["sessions"]):
                for session in json_data["sessions"]:
                    if session[dose] > 0 and session["min_age_limit"] <= age:
                        msg = f"{session['vaccine']} available for {session[dose]} people for Dose {dosec} on
{session['date']} at {session['name']}, {session['address']}\n"
                        update.message.reply_text(msg)
                        update.message.reply_text(
                            "\nLink to Book Slots : https://www.cowin.gov.in/")
                        return
            update.message.reply_text(
                "We'll notify further when any Slots Available!\n")
            sleep(TIME_GAP)

```

```

def botdistrict(update, context):
    da = str(' '.join(context.args[:-2])).lower()
    dosec = str(context.args[-1])
    dose = 'available_capacity_dose'+dosec
    age = int(context.args[-2])
    update.message.reply_text("[+]Bot Started\n")
    while True:
        for day in range(0, 5):

```

```

d3 = (datetime.now(IST) + timedelta(days=day)).strftime("%d-%m-%Y")
d4 = data.get(da)
if(d4 == None):
    update.message.reply_text("[-]Wrong District!\n")
    return
url = f"https://cdn-api.co-
vin.in/api/v2/appointment/sessions/public/findByDistrict?district_id={d4}&date={d3}"
print(url)
try:
    response = requests.get(url, headers=headers)
except:
    update.message.reply_text("API Error!\n")
    return
if response.status_code == 200:

    json_data = json.loads(response.text)
    if len(json_data["sessions"]):
        for session in json_data["sessions"]:
            if session[dose] > 0 and session["min_age_limit"] <= age:
                msg = f"{session['vaccine']} available for {session[dose]} people for Dose {dose} on
{session['date']} at {session['name']}, {session['address']}\n"
                update.message.reply_text(msg)
                update.message.reply_text(
"/nLink to Book Slots : https://www.cowin.gov.in/")
                return
    update.message.reply_text(
        "We'll notify further when any slots Available!\n")
    sleep(TIME_GAP)

```

```

def help(update, context):
    msg = ""

```

Programmed by,

**Mahankali Prathyusha Lahari,
IV CSE-1,MGIT.**

Check Instructions to get slot details!!\n

-----\n

->Checks For Slots in this PinCode\n

/pincode pin\n

Example:/pincode 507002\n

->Checks For Slots in this District \n

/district name\n

Example: /district khammam\n

->Starts a Bot that checks continously pincode-wise \n

/botpincode pin age dose\n

Example: /botpincode 507002 18 1\n

->Starts a Bot that checks continously city-wise \n

/bot_city city age dose\n

Example: /bot_city hyderabad 18 2\n

Link to Book Slots : <https://www.cowin.gov.in/>

-----\n

'''

update.message.reply_text(msg)

def main():

updater = Updater(token=TOKEN, use_context=True, workers=10)

dp = updater.dispatcher

dp.add_handler(CommandHandler("district", district, run_async=True))

dp.add_handler(CommandHandler("pincode", pincode, run_async=True))

dp.add_handler(CommandHandler("botdistrict", botdistrict, run_async=True))

dp.add_handler(CommandHandler("botpincode", botpincode, run_async=True))


```
dp.add_handler(CommandHandler("help", help, run_async=True))
dp.add_handler(CommandHandler("start", help, run_async=True))
updater.start_polling()
updater.idle()
```

```
if __name__ == "__main__":
    main()
```