

# **School of Computer Science and Engineering**

## **RESTAURANT RECOMMENDATION VIA TELEGRAM CHATBOT**

A project submitted  
in partial fulfillment of the requirements for the degree of  
Bachelor of Technology in Computer Science and  
Engineering

**By**

Saharsh Satish (20BCE2426)

R. Shamini (20BDS0350)

Ashutosh Bashyal (20BCE2894)

Rishabh Saboo (19BCE0145)

**Course Instructor**

Dr. Ramesh Babu K

Professor

**April 2022**

## **UNDERTAKING**

This is to declare that the project entitled “RESTAURANT RECOMMENDATION VIA TELEGRAM CHATBOT” is an original work done by undersigned, in partial fulfillment of the requirements for the degree “Bachelor of Technology in Computer Science and Engineering” at School of Computer Science and Engineering, Vellore Institute of Technology (VIT), Vellore.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or University.

Saharsh Satish (20BCE2426)

R. Shamini (20BDS0350)

Ashutosh Bashyal (20BCE2894)

Rishabh Saboo (19BCE0145)

## **ABSTRACT**

The culture of eating food from restaurants has been increasing every day. Whether it be from tiffin service or restaurants, people are constantly searching for good places to eat. Even at this time of COVID, the demand of restaurant food has just decreased by 17% in 2021. The proposed system recommends restaurants that specializes on the food of the user, around their locality which has been top rated by the fellow visitors in your preferred messaging application. The system also connects the user, preferring to order the food, with the representative of the restaurant to clear any doubts if so. Chatbot-based recommendation service app can efficiently manage time and finances by allowing restaurant customers to easily access the information they need at any time and from any location. Chatbot is combined with the messaging platform and enables various services due to the text type interactive service.

**Keywords:** Chatbot, restaurant recommendation, messaging application, top rated

# TABLE OF CONTENTS

Chapter	
1	Abstract
1.1	Abstract & Keywords
1.2	Introduction
2	Literature Review
3	Proposed Methodology
4	Proposed architecture/Design
5	Description on Various Modules
6	Implementation (Sample Code)
7	. Testing (ALL Screen Snapshots)
8	Conclusion and Future Enhancements
9	References

## **Introduction:**

Chatbots have a wide variety of possible use cases and are of interest to both the industry and researchers. With the increasing number of people using messaging applications and the increasing number of chatbots being a major part in running a business from ground up, a chatbot can really help change the outcome of a small business. Chatbots have many possible fields of application and especially task-oriented chatbots have a clear utility in real applications as they are built to help users achieve some specific goal such as to make a restaurant reservation, order a pizza, book a flight, and so on. Using bots instead of human assistants could reduce the time and effort required to get help with the specific task at hand. This project helps to recommend restaurants based on their chosen food. It uses Telegram Bot API that is provided by the developer and retrieves its data from a database to recommend its user the best place to get their food. Moreover, it stores the users' feedback and constantly changes its recommendation based on the filtering the feedback.

# Literature Review:

Include Tables and figures wherever applicable

S.No	Title of the resource (journal paper/conference paper/ title of the web page)	Year	Journal name/ Conference title/ Website link
1	A Survey on Conversational Recommender Systems	2021	<a href="https://arxiv.org/pdf/2004.00646.pdf">A https://arxiv.org/pdf/2004.00646.pdf</a>
2	Introduction to AI Chatbots	2020	<a href="https://pdfs.semanticscholar.org/f5f4/746acffef08df37f184cb6acc0505362ea9b.pdf">s://pdfs.semanticscholar.org/f5f4/746acffef08df37f184cb6acc0505362ea9b.pdf</a>
3	Review of Chatbots Design Techniques	2018	<a href="https://www.researchgate.net/profile/Nahdatul-Akmanad/publication/327097910_Review_of_Chatbots_Design_Techniques/links/5b77cf3e4585151fd11cd905/Review-of-Chatbots-Design-Techniques.pdf">tps://www.researchgate.net/profile/Nahdatul-Akmanad/publication/327097910_Review_of_Chatbots_Design_Techniques/links/5b77cf3e4585151fd11cd905/Review-of-Chatbots-Design-Techniques.pdf</a>
4	End-to-End Trainable Chatbot for Restaurant Recommendations	2017	<a href="https://www.diva-al.org/smash/get/diva2:1139496/FULLTEXT01.pdf">ps://www.diva-al.org/smash/get/diva2:1139496/FULLTEXT01.pdf</a>

## **Proposed Methodology:**

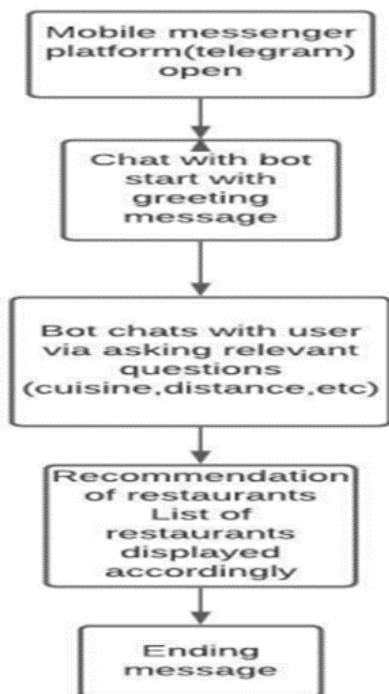
This is how we propose our bot will work

- User needs to use start command on telegram
- User has the option to choose cuisine/rate food/rate service
- If user chooses cuisine then bot provides options such as chinese,north indian,south indian
- On selection of cuisine ,bot provides 4-5 top restaurants from a dataset which we have got from kaggle .
- Then the user has the option of rate food and rate service with thumbs up or thumbs down.

## **Architecture/Design:**

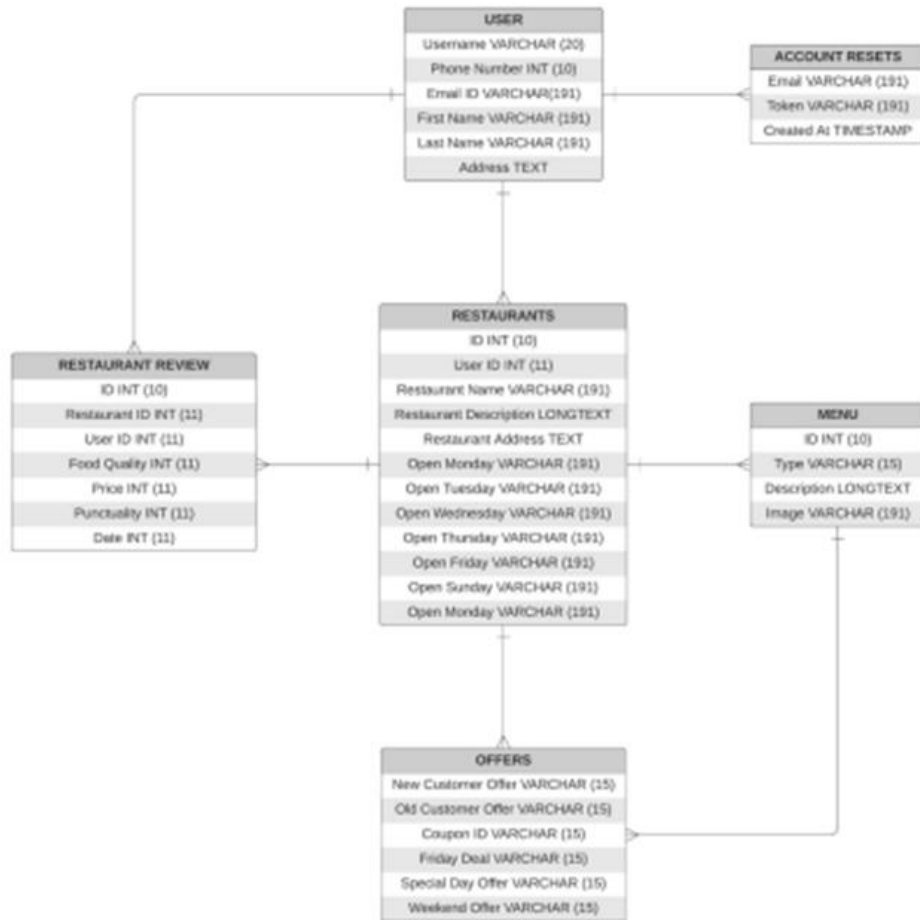
### **Design :**

1. The proposed chatbot can make it easier for users to search for real-time information.
2. Users also get more accurate information because the virtual personal assistant concept, combined with the mobile messenger platform, provides the relevant information.
3. Chatbot get restaurant recommendations while also receiving real-time information from the restaurant, such as reservation status, seat information, menu information, and so on.





## Database Design:



**Fig: ER Diagram of entities used by the Chatbot**

## **Description on various modules:**

### **Developer Modules:**

The python libraries used for the development of the bot are:

1. telebot
2. telegram
3. telegram.ext
4. os
5. logging
6. uuid
7. requests

1. Telebot: It encapsulates all API calls in a single class. It provides functions such as send message, send document, etc. and several ways to listen for incoming messages.

2. Telegram: It is a module that is used to send notifications via telegram.

3. Telegram.ext: It is a library that provides a pure python interface to the Telegram Bot API.

4. OS: It provides functions for creating and removing a directory, fetching its contents, changing and identifying the current directory.

5. Logging: It is a code that is embedded into an application to create and manage log events. Logging libraries provide APIs for creating, structuring, formatting and transmitting log events in a consistent way. Like agents, they're used to send events from an application to a destination.

6. uuid: They are used for identifiers for documents, hosts, application clients, and other situations where a unique value is necessary.

7. Requests: It will allow you to send HTTP/1.1 requests. With it content like headers, form data, multipart files and parameters via simple Python libraries can be added. It also allows access to the response data of Python in the same way.

## **Implementation:**

```
import telebot as telebot
from telegram import *
from telegram import bot
from telegram.ext import *
from requests import *
import os
import logging
from uuid import uuid4

TOKEN = os.getenv("TOKEN")
bot = telebot.TeleBot(TOKEN)
#deserts="Desserts"
orderfood = "Type of food"
rateourservice = "Rate our service"
ratethefood = "Rate the food"
Vegetarian = "Vegetarian"
maincourse="Main course"
southIndian="South Indian"
icecreams="Ice creams"
american="American"
chettinad="Chettinad"
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)
# def put(update, context):
#     """Usage: /put value"""
#     # Generate ID and separate value from command
#     key = str(uuid4())
#     # We don't use context.args here, because the value may contain
whitespaces
#     value = update.message.text.partition(' ')[2]
#
#     # Store value
#     context.user_data[key] = value
#     # Send the key to the user
#     update.message.reply_text(key)
#
# def get(update, context):
#     """Usage: /get uuid"""
#     # Seperate ID from command
#     key = context.args[0]
```

```

#
#     # Load value and send it to the user
#     value = context.user_data.get(key, 'Not found')
#     update.message.reply_text(value)
logger = logging.getLogger(__name__)

# soup="Soups"
# starters="Starters"
# maincourse="Main Course"
# deserts="deserts"
# juices="Juices"

def start_handler(update, context):
    # update.message.reply_text("Hello", reply_markup=food())
    user = update.message.from_user
    logger.info("User %s started the conversation.", user.first_name)

    context.bot.send_message(chat_id=update.effective_chat.id,
text="Welcome, Please choose any one of the options",

reply_markup=ReplyKeyboardMarkup(choice()))

def choice():
    keyboard = [[KeyboardButton(orderfood)],
[KeyboardButton(rateourservice)], [KeyboardButton(ratethefood)]]
    return keyboard

def pref():
    buttons = [[KeyboardButton(Vegetarian,
callback_data="Vegetarian")],
[KeyboardButton("Nonveg", callback_data="Nonveg")]]
    return ReplyKeyboardMarkup(buttons)

def food():
    button = [[KeyboardButton(southIndian,
callback_data="southIndian")], [KeyboardButton(chettinad,
callback_data="chettinad")], [KeyboardButton(icecreams,
callback_data="icecreams")], [KeyboardButton(american,
callback_data="american")]]
    return button

```

```

# def favour():
#
#
keyboard=[[InlineKeyboardButton(soup,callback_data="soup")],[InlineKey
boardButton(starters,callback_data="starters")],[InlineKeyboardButton(
maincourse,"maincourse")],[InlineKeyboardButton(deserts,callback_data=
"deserts")],[InlineKeyboardButton(juices,callback_data="juices")]]
#     return keyboard

def message(update: Update, context: CallbackContext):
    # if orderfood in update.message.text:
    #     #update.message.reply_text(orderfood)
    #     button = [[InlineKeyboardButton("soup",
callback_data="soup")],
    #               [InlineKeyboardButton("starters",
callback_data="starters")],
    #               [InlineKeyboardButton("maincourse",
"maincourse")],
    #               [InlineKeyboardButton("deserts",
callback_data="deserts")],
    #               [InlineKeyboardButton("juices",
callback_data="juices")]]
    #
    #
context.bot.send_message(chat_id=update.effective_chat.id,reply_markup
=InlineKeyboardMarkup(button),text="What would you like to have
now??")

    if rateourservice in update.message.text:
        # update.message.reply_text(rateourservice)
        like="👍 "
        dislike="👎 "
        buttons = [[KeyboardButton(like, callback_data="like")],
                    [KeyboardButton(dislike, callback_data="dislike")]]
        context.bot.send_message(chat_id=update.effective_chat.id,
reply_markup=ReplyKeyboardMarkup(buttons),
                                text="Did you like our service?")

    if ratethefood in update.message.text:
        buttons = [[InlineKeyboardButton("👍", callback_data="like")],
                    [InlineKeyboardButton("👎",
callback_data="dislike")]]

```

```

        context.bot.send_message(chat_id=update.effective_chat.id,
                                reply_markup=InlineKeyboardMarkup(buttons),
                                text="Did you like the food?")

    if orderfood in update.message.text:
        # buttons = [[KeyboardButton(Vegetarian,
        callback_data="Veg")], [KeyboardButton("Non-Vegetarian",
        callback_data="Nonveg")]]
        #
    context.bot.send_message(chat_id=update.effective_chat.id, text="Choose
    any one?")
        update.message.reply_text("choose any one",
        reply_markup=pref())
        if Vegetarian in update.message.text:

context.bot.send_message(chat_id=update.effective_chat.id, reply_markup
=ReplyKeyboardMarkup(food()), text="Choose any one?")
        elif "Nonveg" in update.message.text:
            context.bot.send_message(chat_id=update.effective_chat.id,
            reply_markup=ReplyKeyboardMarkup(food()), text="Choose any one?")
            if southIndian in update.message.text:

context.bot.send_message(chat_id=update.effective_chat.id, text="https:
//www.zomato.com/chennai/mami-tiffen-stall-mylapore \n\n
https://www.zomato.com/chennai/nithya-amirtham-mylapore \n\n
https://www.zomato.com/chennai/prive-restaurant-mylapore \n\n
https://www.zomato.com/chennai/sukha-nivas-mylapore \n\n
https://www.zomato.com/chennai/hotel-chaitra-mylapore ")
            elif chettinad in update.message.text:

context.bot.send_message(chat_id=update.effective_chat.id, text="https:
//www.zomato.com/chennai/kaaraikudi-mylapore \n\n
https://www.zomato.com/chennai/anjappar-mylapore \n\n
https://www.zomato.com/chennai/viswanathan-chettinadu-hotel-mylapore
\n\n https://www.zomato.com/chennai/hotel-karur-alagar-mylapore \n\n
https://www.zomato.com/chennai/madurainaygam-chettinadu-mess-mylapore
\n\n https://www.zomato.com/chennai/hotel-select-mylapore")
            elif icecreams in update.message.text:

context.bot.send_message(chat_id=update.effective_chat.id, text="https:
//www.zomato.com/chennai/a2b-adyar-ananda-bhavan-mylapore \n\n
https://www.zomato.com/chennai/lassi-shop-mylapore \n\n
https://www.zomato.com/chennai/ibaco-mylapore \n\n
https://www.zomato.com/chennai/keventers-milkshakes-ice-creams-
mylapore \n\n https://www.zomato.com/chennai/milkyway-mylapore")

```

```

    # elif deserts in update.message.text:
        #context.bot.send_message(chat_id=update.effective_chat.id,
text="https://www.zomato.com/chennai/27-culinary-street-mylapore
")#\n\n https://www.zomato.com/chennai/zuka-mylapore \n\n
https://www.zomato.com/chennai/southern-street-mylapore \n\n
https://www.zomato.com/chennai/cake-affairs-mylapore \n\n
https://www.zomato.com/chennai/senthil-softy-zone-mylapore")
        elif american in update.message.text:
            context.bot.send_message(chat_id=update.effective_chat.id,
text="https://www.zomato.com/chennai/tovo-1-mylapore \n\n
https://www.zomato.com/chennai/bro-bistro-1986-mylapore \n\n
https://www.zomato.com/chennai/pedrenos-mylapore \n\n ")

    # update.message.reply_text("choose any one",
reply_markup=food())

    # else:
    #
context.bot.send_message(chat_id=update.effective_chat.id,text="Please
Choose the options below")

if __name__ == '__main__':

    updater = Updater(TOKEN, use_context=True)
    dp = updater.dispatcher

    # dp.add_handler(CommandHandler('put', put))
    # dp.add_handler(CommandHandler('get', get))
    dp.add_handler(CommandHandler("start", start_handler))
    dp.add_handler(MessageHandler(Filters.text, message))
    updater.bot.set_webhook()
    updater.start_polling()
    updater.idle()

```



## Google script code:

### To store the user's information using the webhook

```
var token = "5208525951:AAffMbL1lKYtgNlWvDQfYhFBrDva_9PrXLM"; // FILL
IN YOUR OWN TOKEN

var telegramUrl = "https://api.telegram.org/bot" + token;

var webAppUrl =
"https://script.google.com/macros/s/AKfycbXpKaYar6nNlvXrVWqaKygWh-
_aPiw32jPZrc2qilzhGtmj9PpG_1lQV5F0FLYUGkng/exec"; // FILL IN YOUR
GOOGLE WEB APP ADDRESS

var ssId = "1n4McMf-cPNPa8gC0FbBNDNE268tf4wlujEsabLpxsv8"; // FILL IN
THE ID OF YOUR SPREADSHEET

..

// function getMe() {

//   var url = telegramUrl + "/getMe";

//   var response = UrlFetchApp.fetch(url);

//   Logger.log(response.getContentText());

// }

function setWebhook() {

  var url = telegramUrl + "/setWebhook?url=" + webAppUrl;

  var response = UrlFetchApp.fetch(url);

  Logger.log(response.getContentText());

}
```

```

function sendText(id,text) {

    var url = telegramUrl + "/sendMessage?chat_id=" + id + "&text=" +
text;

    var response = UrlFetchApp.fetch(url);

    Logger.log(response.getContentText());

}

//function doGet(e) {

    //return HtmlService.createHtmlOutput("Hi there");

//}

function doPost(e) {

    // this is where telegram works

    var data = JSON.parse(e.postData.contents);

    var text = data.message.text;

    var id = data.message.chat.id;

    var name = data.message.chat.first_name + " " +
data.message.chat.last_name ;

    // var answer = "Hi " + name + ", Please choose an option ";

    //  sendText(id,answer);


    // SpreadsheetApp.openById(ssId).getSheets()[0].appendRow([new
Date(),id,name,text,answer]);

```

```
SpreadsheetApp.openById(ssId).getSheets()[0].appendRow([new
Date(),id,name,text]);

if(/^@/.test(text)) {

    var sheetName = text.slice(1).split(" ")[0];

    var sheet =
SpreadsheetApp.openById(ssId).getSheetByName(sheetName) ?
SpreadsheetApp.openById(ssId).getSheetByName(sheetName) :
SpreadsheetApp.openById(ssId).insertSheet(sheetName);

    var comment = text.split(" ").slice(1).join(" ");

    // sheet.appendRow([new Date(),id,name,comment,answer]);

    sheet.appendRow([new Date(),id,name,comment]);

}

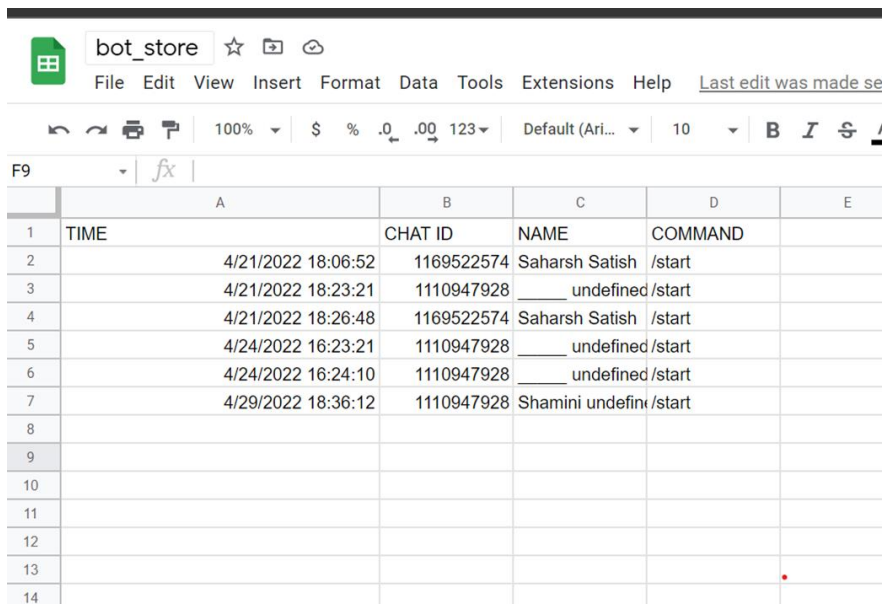
}
```

## Testing:

**To track the users who have pressed start button:**

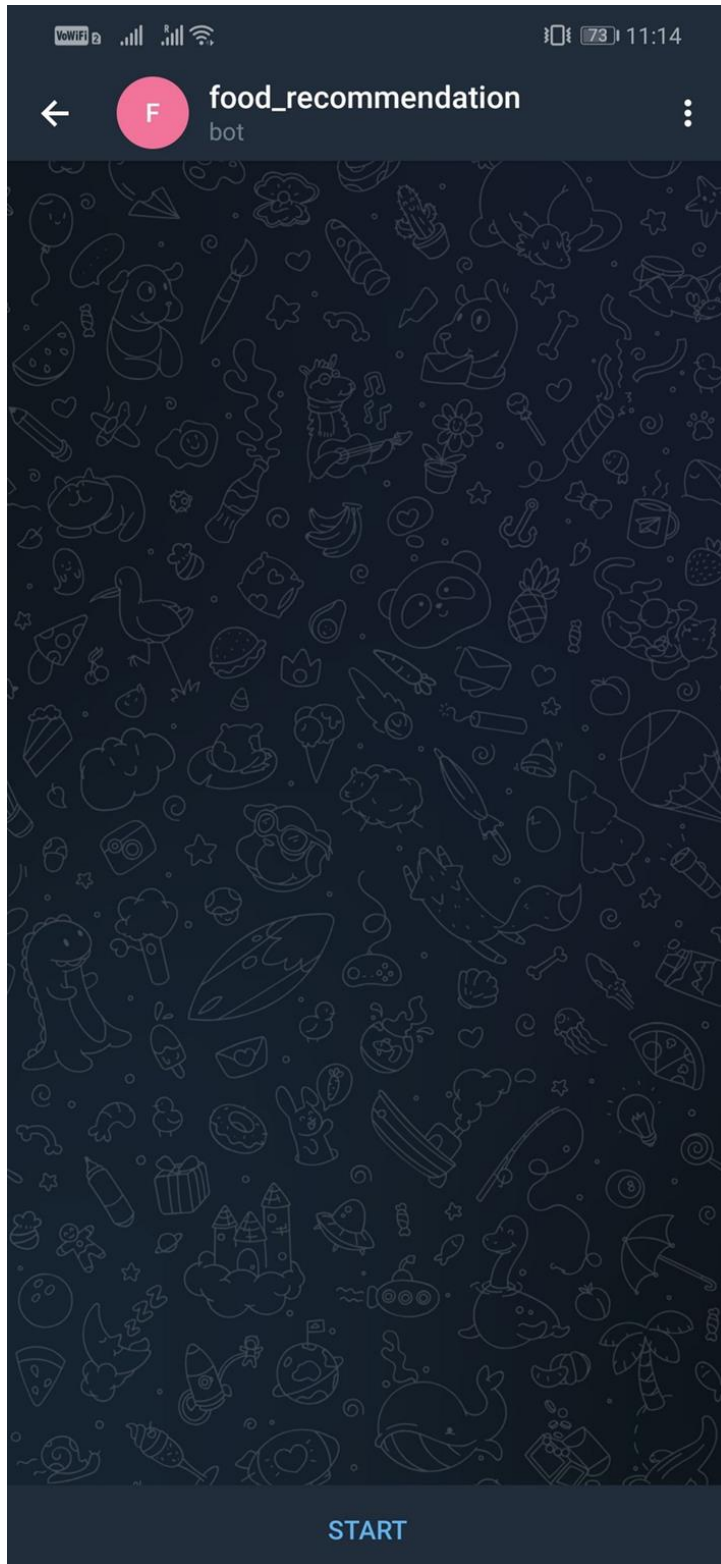
```
main x
C:\Users\Shamini\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:/Users/Shamini/Pych
2022-04-29 18:30:01,335 - apscheduler.scheduler - INFO - Scheduler started
2022-04-29 18:30:10,370 - __main__ - INFO - User ____ started the conversation.
2022-04-29 18:30:36,677 - __main__ - INFO - User Shamini started the conversation.
```

**To store the user information in excel:**



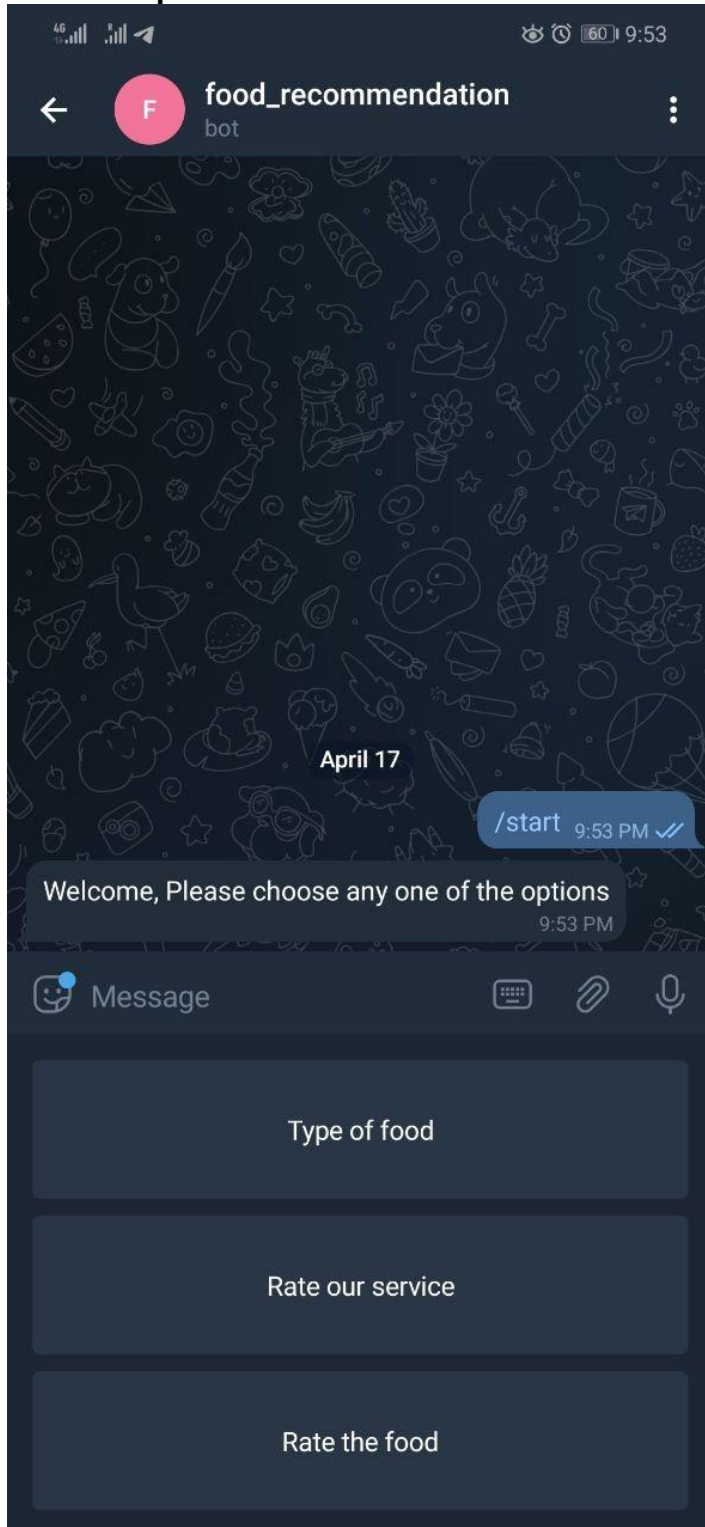
	A	B	C	D	E
1	TIME	CHAT ID	NAME	COMMAND	
2	4/21/2022 18:06:52	1169522574	Saharsh Satish	/start	
3	4/21/2022 18:23:21	1110947928	____ undefined	/start	
4	4/21/2022 18:26:48	1169522574	Saharsh Satish	/start	
5	4/24/2022 16:23:21	1110947928	____ undefined	/start	
6	4/24/2022 16:24:10	1110947928	____ undefined	/start	
7	4/29/2022 18:36:12	1110947928	Shamini undefin	/start	
8					
9					
10					
11					
12					
13					
14					

## 1. Start Page:



This is the start page of the bot. When a user searches for the bot or is taken via link they are taken to this interface where they can initiate the bot by pressing or clicking “START” at the bottom which sends a command to the server and the bot responds accordingly.

## 2. Initial Options:

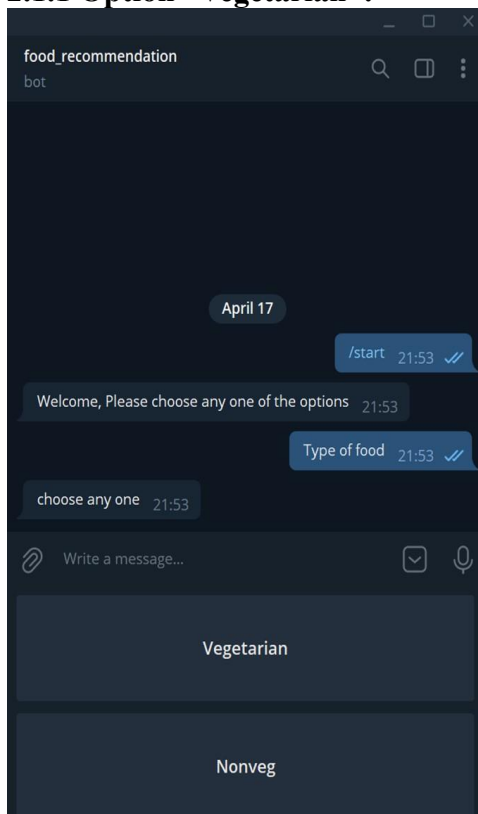


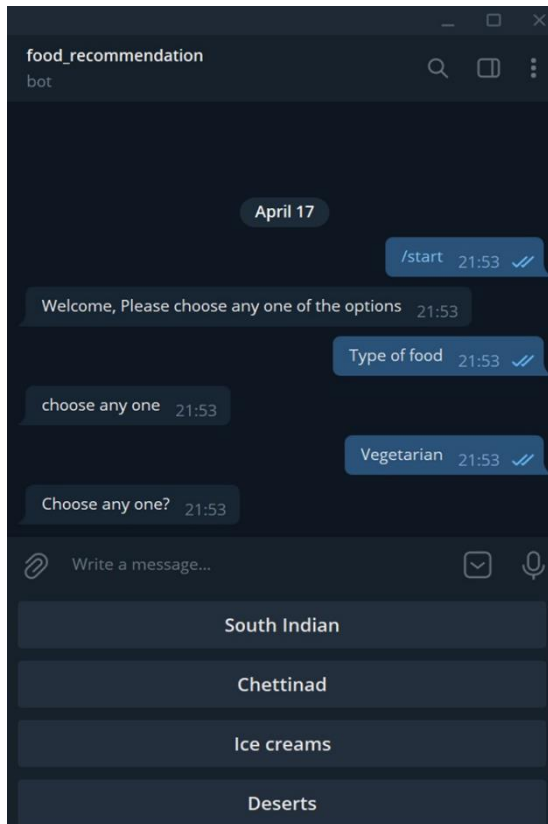
Once the user presses “START” they are given an option to choose the usability of the bot namely “Type of food”, “Rate our Service” and “Rate the food”. The user can select “Type of food” to choose the main aim of the bot. “Rate our service” to rate the service provided by the bot and “Rate the food” to rate the food once they are done with the service and delivery.

## 2.1 Option “Type of Food”:

On clicking “Type of food” from the previous option the user is asked to choose from Vegetarian food and Non-vegetarian food. Based on the selection the user is taken into further functionality of the bot.

### 2.1.1 Option “Vegetarian”:





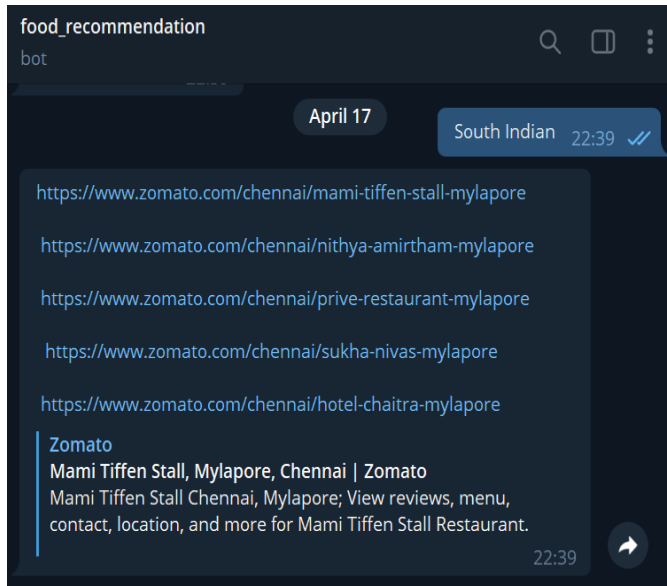
Once the user chooses Vegetarian they are then asked to select from list of cuisines that ranges from providing Breakfast to Lunch to Dinner and also serve a three-course meal consisting of an appetizer, main dish and dessert.

### **Results of selection of cuisine by the user:**

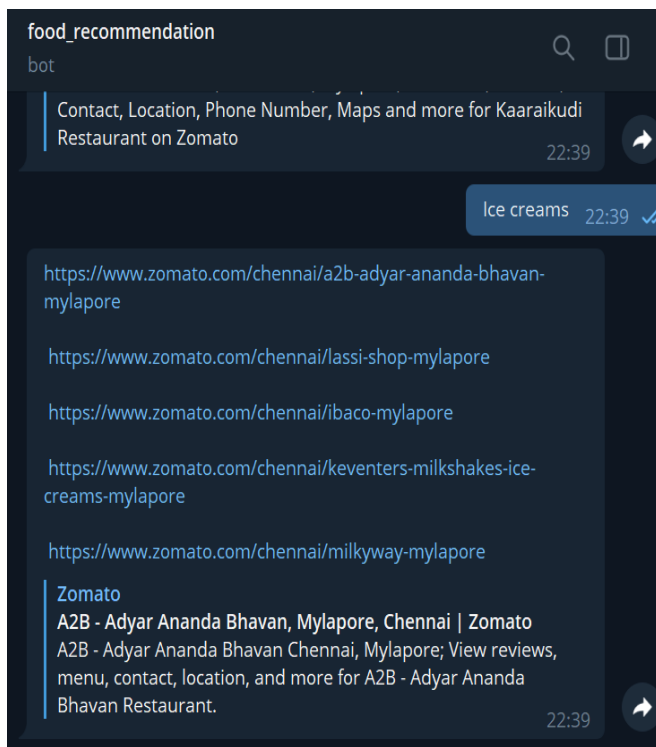
When the user selects South Indian food, they are sent list of links to choose from. The link is accessed from the database and based on the request of the user the bot selects close list of links of restaurants to that provide the respective food.

Furthermore, the bot replies with list of restaurants for other foods as shown in the report below.

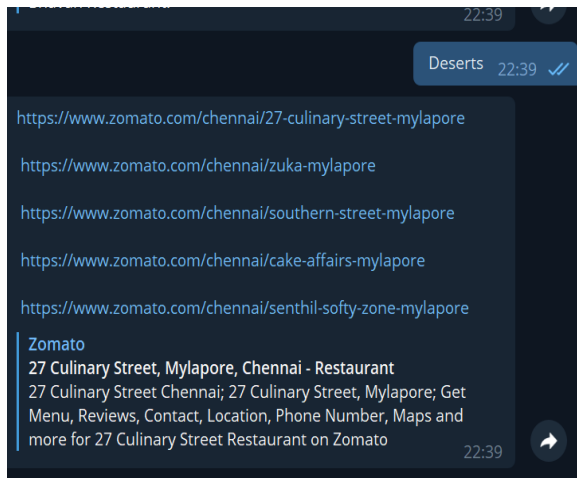




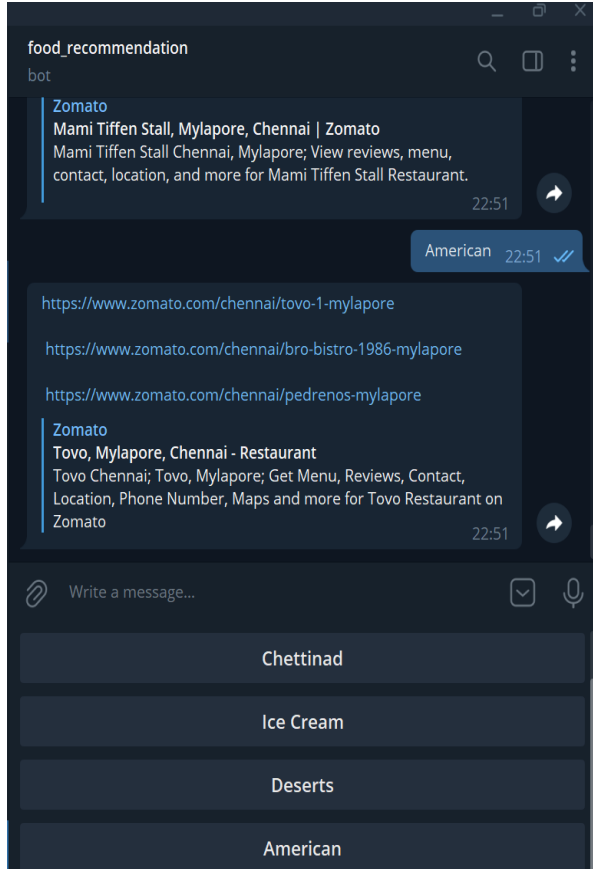
## Recommendations for Ice Creams:



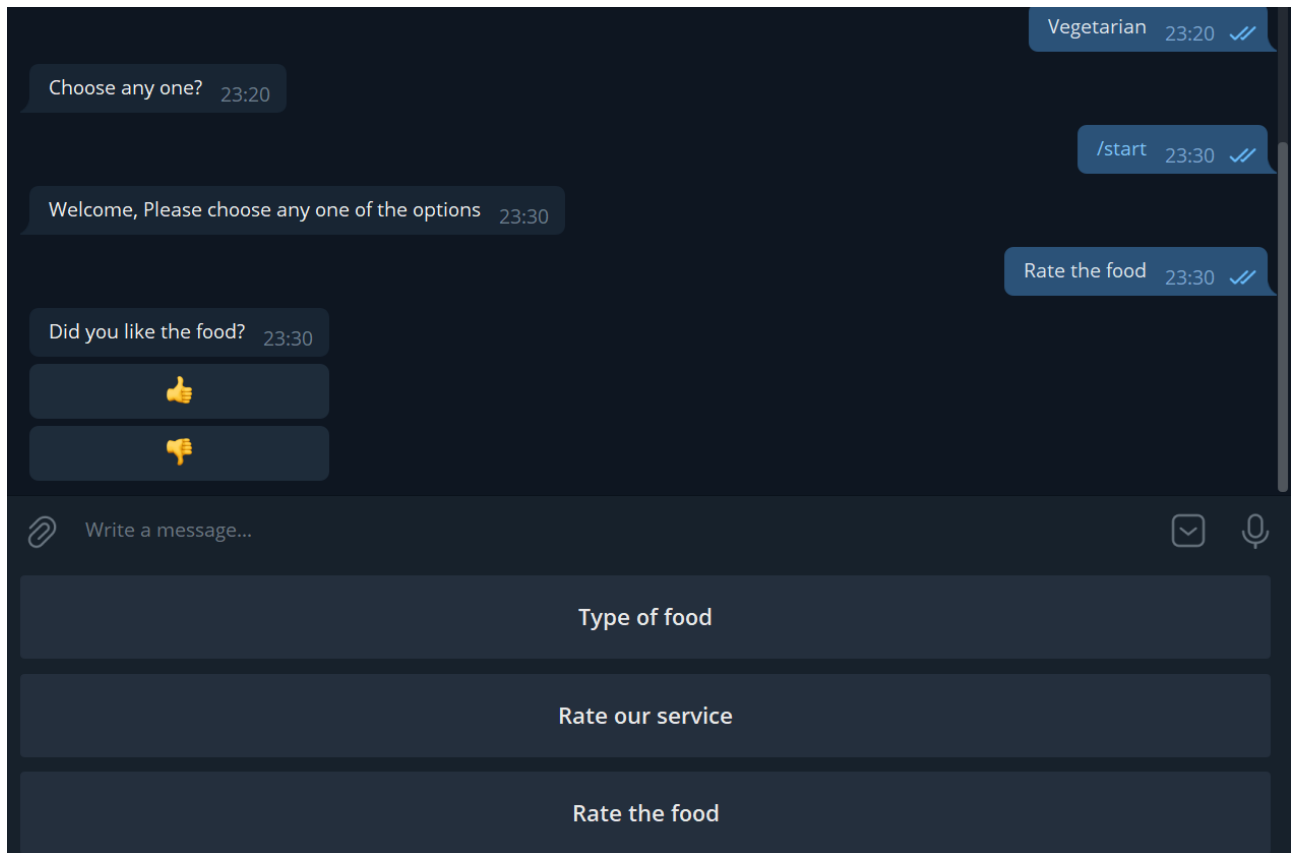
## Recommendations for Desserts:



## Restaurant Recommendation for American Food:



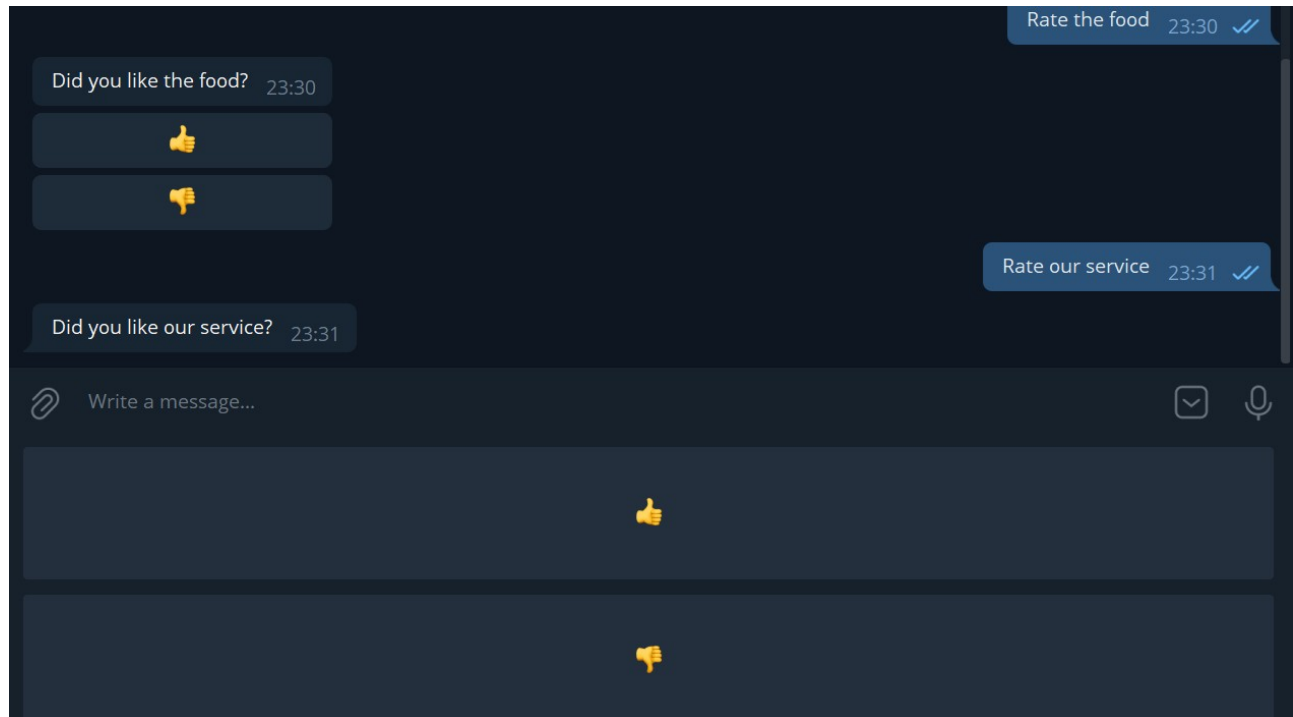
## 2.2. Option “Rate our Service”



The user is allowed to Rate our service based on how the bot performed, how it served its purpose, whether it lagged or not, whether or not it suggested correct restaurant for users liking.

The rating is stored into the database and later used to modify or change the bot based on what the users want the bot to do or perform.

## 2.3 Option “Rate the food”



Once the user is done with the bot and the food ordered, they are requested to rate the food. The rating is then stored into a database that is further used to change the recommendations of the restaurants.

The rating is later suggested to change into 5-star rating, meaning 1 star as lowest rating and 5 being the highest.

## Conclusion and Future Enhancements:

There has been a lot of research done on end-to-end trainable chatbot recommender systems. The increased use of mobile applications for ordering food has created a plethora of new opportunities to develop on and chatbot recommender systems are one of the foremost technologies that will enhance user experience. The chatbot can be modified to take audio inputs and can detect the emotions of people based on what has been recommended. A video input of the person can also be used and the face can be mapped to detect the emotion of the user and food can be recommended using that as one of the parameters as well.

## Team Members Contribution

Register Number	Name	Contribution / Role in this Project
20BCE2426	Saharsh Satish	Coding for Storing the user details in the spreadsheet  Documentation
20BDS0350	R.Shamini	Coding for Storing the user details in the spreadsheet  Telegram bot coding  Documentation
20BCE2894	Ashutosh Bashyal	Documentation
19BCE0145	Rishabh Saboo	Documentation

## References

[https://www.researchgate.net/publication/351228837\\_Design\\_and\\_Development\\_of\\_CHATBOT\\_A\\_Review/link/60a0e52fa6fdcccacb5a2a34/download](https://www.researchgate.net/publication/351228837_Design_and_Development_of_CHATBOT_A_Review/link/60a0e52fa6fdcccacb5a2a34/download)

<https://www.udemy.com/course/full-stack-telegram-bot-from-scratch-python-psql-heroku/learn/lecture/26861444?start=45>

<https://medium.com/the-hooray-media/telegram-bot-the-new-user-interface-e96047615109>

<https://www.youtube.com/watch?v=tclPVfTtD1k>

<https://www.toptal.com/python/telegram-bot-tutorial-python>

<https://pytba.readthedocs.io/en/latest/index.html>

<https://pypi.org/project/pyTelegramBotAPI/>

<https://python-telegram-bot.readthedocs.io/en/stable/>

<https://www.koreascience.or.kr/article/JAKO202008351737595.pdf>

<https://techcrunch.com/2016/09/11/pickme-a-bot-for-restaurant-suggestions/>