FOURTH SEMESTER
MINI PROJECT REPORT

Famalo Telegram bot using telebot python framework,
Your All in one buddy

Submitted By

Name: ADHITHYAN V P

Reg.No:THAWBOA030

Department of Data Science
St.Thomas' College (Autonomous), Thrissur

Under the Guidance of

Sreelekha K
Assistant Professor
Department of Data Science, St.Thomas College,Thrissur

Department of Data Science

St.Thomas'College (Autonomous) ,Thrissur

Affiliated To University of Calicut & Reaccredited By NAAC with "A++" Grade

&College with Potential Excellence

CERTIFICATE

This is to certify that the Mini Project Report titled **"Famalo Telegram bot using telebot python framework,Your All in one buddy"** is a bonafide record of the work carried out by **ADHITHYAN V P (THAWBOA030)** of St.Thomas' College (Autonomous)Thrissur-680005 in partial fulfillment of the requirements for the award of Degree of B.Voc Data Science of University of Calicut, during the academic year 2022-2025. The fourth semester Mini Project report has been approved as it satisfies the academic requirements in the respect of mini project work prescribed for the said degree.

HOD **Principal** **Faculty In-Charge**

Valued On: **…….**

Examiners:

1……..…….. **2.……**

Internal Examiner External Examiner

DECLARATION

I hereby declare that the project report entitled "**Famalo Telegram bot using telebot Python framework,Your All in one buddy"** which is being submitted in partial fulfillment of the requirement of the award of the Degree in Bachelor of Vocational Studies in Data Science is the result of the project carried out by me under the guidance and supervision of **Sreelekha K** , Assistant Professor, Department of Data Science.

I further declared that I or any other person has not previously submitted this project report to any other institution / university for any other degree / diploma or any other person.

ADHITHYAN VP

Place : Thrissur

Date : (Signature)

Famalo Telegram bot using telebot python Framework, Your All In One Buddy

CONTENTS

1.INTRODUCTION

Introducing Famalo - Your All-in-One Buddy

Famalo is an innovative Telegram bot designed to offer a wide range of functionalities and services, making it your ultimate all-in-one buddy. With its diverse features across different categories, Famalo strives to meet your various needs and provide you with a convenient and efficient experience.

The main menu of Famalo presents a selection of categories, each catering to specific areas of interest. From education and calculators to generators and real-life information, Famalo has it all. Whether you're seeking educational resources, performing calculations, generating passwords or QR codes, accessing real-time weather details, or even enjoying some fun activities like jokes and meme generation, Famalo has you covered.

What sets Famalo apart is its ability to automatically detect features based on your input text. By simply interacting with the bot, Famalo will try to identify the most relevant features for your specific requests, ensuring a seamless and intuitive user experience.

Each category in the main menu further expands into submenus, offering more specific features and functionalities. Whether you need course recommendations, cheat sheets, or learning resources in the education category, or advanced calculations and playback speed calculations in the calculators category, Famalo has thoughtfully organized its features to provide you with a comprehensive range of options.

For those associated with ST Thomas College, Famalo offers specific resources tailored to your needs, enabling easy access to relevant information. Furthermore, system-related functionalities and recommendations are available for exploration, making it a valuable tool for technology enthusiasts.

Feel free to explore the diverse capabilities of Famalo and make the most of its features. Whether you're a student, professional, or simply seeking entertainment, Famalo has something for everyone.

With Famalo, you have a reliable and versatile companion at your fingertips. Whether you need assistance with educational resources, calculations, real-time information, or just want to have some fun, Famalo is here to assist you every step of the way.

2.OBJECTIVES

The objectives of the Famalo Project are as follows:

1. Efficiency and Convenience:
    - Streamline tasks and processes to save user time and effort.
    - Provide quick and easy access to a wide range of functionalities and services.
2. Comprehensive Functionality:
    - Offer diverse features across different categories to meet various user needs.
    - Cover areas such as education, calculators, generators, real-life information, and engaging activities.
3. User-Friendly Experience:
    - Design a clear and intuitive interface for easy navigation.
    - Ensure a seamless and user-friendly interaction with Famalo's features.
4. Automatic Feature Detection:
    - Implement intelligent feature detection based on user input text.
    - Automatically identify relevant features to enhance user experience.
5. Continuous Updates and Improvements:
    - Regularly update Famalo with new features and enhancements.
    - Incorporate user feedback to improve functionality, usability, and overall user experience.
6. Harnessing the Power of Python:
    - Leverage the robustness and versatility of the Python programming language to develop efficient and reliable functionalities within Famalo.
    - Utilize Python's extensive libraries and frameworks to expand the capabilities of Famalo and provide a rich set of features.
7. Integration with Telebot API:
    - Utilize the Telebot API, a Python library for interacting with the Telegram Bot API, to enable seamless integration and communication with users on the Telegram platform.
    - Leverage the Telebot API's capabilities for message handling, user interactions, and other Telegram-specific features to enhance the functionality and user experience of Famalo.
8. Ensuring Code Quality and Maintainability:
    - Adhere to best practices in software development, following clean and modular coding principles to ensure readability, maintainability, and scalability of the Famalo project.
    - Utilize Python's well-documented syntax and standardized coding conventions to create a codebase that is easy to understand and maintain for future updates and improvements.

3.CURRENT SYSTEM STUDY

Introduction

The current system for accessing desired features on the web involves a series of steps that can be time-consuming, complex, and potentially risky for users. This study aims to analyze the limitations and drawbacks of the existing system and highlight the need for a more efficient and user-friendly solution. The following pages will explore each step in detail, providing insights into the challenges users face and the opportunities for improvement.

Processes of Finding a Feature

Step 1: Open a Web Browser

In the current system, users begin by opening a web browser to search for their desired feature. While web browsers provide access to a vast amount of information, this step can be overwhelming for users. They are often bombarded with multiple tabs, distracting advertisements, and various distractions that may divert their attention from their original purpose.

Step 2: Search for Your Desired Feature

Once the web browser is open, users must conduct a search to find the desired feature. This step involves entering relevant keywords, navigating through search results, and assessing the credibility and relevance of different websites. The sheer volume of search results can make it challenging to identify the most suitable website or platform to meet the user's needs. Users may have to sift through numerous pages of search results, wasting time and effort in the process.

Step 3: Find a Website from the Ocean of Links

After conducting a search, users are presented with numerous links and websites related to the desired feature. Selecting the right website from this "ocean of links" can be overwhelming and time-consuming. Users must evaluate factors such as website design, reputation, user reviews, and content quality to ensure they are accessing a trustworthy and reliable platform. Making the wrong choice can result in a poor user experience or even expose users to potential security risks.

Step 4: Open a Website

Once users have chosen a website, they proceed to open it to access the desired feature. This step may seem straightforward, but it can introduce additional

challenges. Users may encounter slow website loading times, compatibility issues with their device or browser, or unexpected errors that hinder their ability to access the website and its functionalities.

Step 5: Sign Up with That Strange Website

To access certain features or services, users often need to create an account or sign up with the website. This process typically involves providing personal information such as an email address, username, and password. However, signing up with a new website, especially one that is unfamiliar or lacks a reputation, can raise concerns about privacy and data security. Users may hesitate to share their personal information due to the potential risks associated with unknown or untrustworthy platforms.

Step 6: Share Your Private Information (Like Email)

After signing up, users may be required to share additional personal information to gain access to the full range of features. This could include sharing sensitive data such as email addresses, phone numbers, or even financial information. Users may have reservations about sharing such information, particularly with websites they have little knowledge or trust in.

Step 7: Wait for OTP Verification

Some websites employ additional security measures such as One-Time Password (OTP) verification. This step involves waiting for a unique code to be sent to the user's provided contact information (usually email or phone) and entering the code to complete the verification process. The waiting time for receiving the OTP can be variable and may result in delays, particularly if there are issues with email deliverability or network connectivity.

Step 8: Log In

Once the sign-up and verification processes are completed, users must log in to their newly created account to access the desired feature. Logging in typically requires entering the registered username or email and the corresponding password. However, users may face challenges such as forgetting their login credentials or encountering technical issues that prevent them from successfully logging in.

Step 9: Pay a Plan If Needed (More Information Like Credit Card)

In some cases, accessing certain features or content may require users to pay for a subscription or a specific plan. This often involves providing additional information, such as credit card details, for payment processing. Users may have concerns about the security and privacy of their financial information when sharing it with online platforms.

Step 10: Complex Use of Website If User Is New

Once logged in, new users may find themselves navigating a complex user interface and struggling to understand the layout and functionalities of the website. Features may be scattered across different menus, and understanding how to access specific functionalities can be challenging. Users may need to invest additional time in exploring and learning the website's structure and operations.

Step 11: Time Waste in Understanding the Layout of That Website

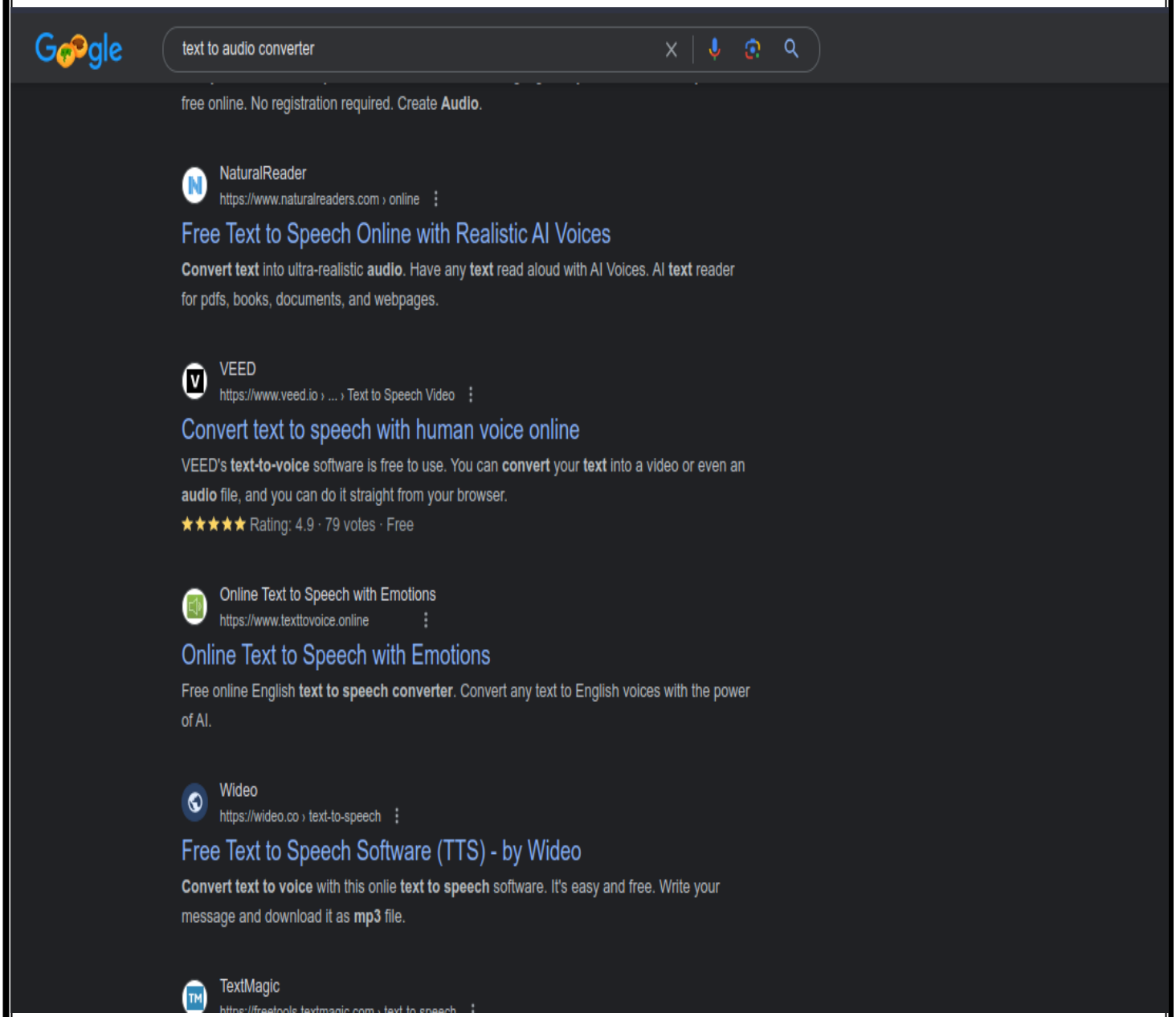
The unfamiliarity with a website's layout can lead to significant time wastage. Users may struggle to locate desired features, navigate through menus, and understand the website's overall structure. This inefficiency can frustrate users and impede their ability to fully utilize the website's offerings.

Step 12: If That Website Doesn't Do What You Wanted, Do All These Again with Another Website

In unfortunate cases, users may find that the chosen website does not meet their expectations or fails to deliver the desired feature. This requires users to repeat the entire process, starting from opening a web browser and conducting a new search for alternative websites. Repeating these steps can be tiresome, time-consuming, and may deter users from exploring alternative options.

The repetition of the entire process adds to the frustration and inefficiency of the current system. Users have to invest additional time and effort in conducting multiple searches, evaluating different websites, and signing up for new accounts. This repetitive cycle not only consumes valuable user time but also hinders the seamless and efficient access to desired features.

Moreover, repeating the process with different websites introduces a level of uncertainty and risk. Users may encounter unreliable websites, untrustworthy platforms, or even potential security threats when sharing personal information repeatedly. These risks further contribute to the user's frustration and dissatisfaction with the current system.



Challenges and Drawbacks of the Current System

The current system for accessing desired features on the web poses several challenges and drawbacks that hinder the user experience. Some of the key challenges include:

- Lack of efficiency: The current system requires users to navigate through various websites, sign up for multiple accounts, and repeat the process if the desired feature is not found. This process consumes time and effort, leading to a lack of efficiency in accessing the desired functionalities.

- Privacy and security risks: Users are often required to share personal information, such as email addresses and credit card details, with unfamiliar websites. This poses privacy and security risks, as users may be exposing sensitive data to potentially untrustworthy platforms.
- Complex user interfaces: Websites may have complex layouts, making it challenging for users, especially newcomers, to navigate and understand the functionalities. This

complexity contributes to a steep learning curve, requiring users to invest additional time in familiarizing themselves with the website's .

Conclusion:

The current system of accessing desired features on the web poses several challenges and inefficiencies for users. From the time wasted in navigating search results and understanding website layouts to the potential risks associated with sharing personal information, there is a clear need for a more streamlined and user-friendly solution. The subsequent pages of this study will explore how the Famalo project addresses these issues and aims to provide a more efficient and convenient experience for users seeking various functionalities and services in one unified platform.

4.GAP BETWEEN CURRENT SYSTEM AND PROPOSED SYSTEM

The existing system for accessing desired features on the web suffers from several limitations and challenges that hinder the user experience. This page will focus on exploring the gap between the current system and the proposed Famalo system, highlighting the significant improvements and advantages offered by the proposed solution.

1. Efficiency and Convenience:

The existing system often requires users to go through a series of time-consuming and repetitive steps to access desired features. From conducting searches, evaluating websites, signing up for accounts, to navigating complex user interfaces, users are faced with multiple barriers that hinder efficiency and convenience. On the other hand, the proposed Famalo system aims to streamline the process and provide quick and easy access to a wide range of functionalities in one unified platform. By automating feature detection and eliminating the need for users to manually search for websites and sign up for accounts, Famalo significantly reduces the time and effort required to access desired features. This enhances efficiency and convenience, allowing users to focus more on utilizing the functionalities rather than navigating through complex processes.

2. Comprehensive Functionality:

While the existing system may provide access to certain features, it often lacks comprehensive functionality across various categories. Users may have to visit multiple websites or platforms to access different types of features, resulting in a fragmented and disjointed experience. In contrast, the proposed Famalo system offers a wide range of functionalities across categories such as education, calculators, generators, real-life information, and engaging activities, all in one platform. This comprehensive approach ensures that users can access diverse features without the need to switch between multiple websites or applications. Famalo aims to be a one-stop solution, providing users with a seamless and integrated experience that caters to their various needs.

3. User-Friendly Experience:

The existing system often presents challenges in terms of user-friendliness. Complex website layouts, unfamiliar user interfaces, and the need to navigate through multiple menus can confuse and frustrate users, especially newcomers. Famalo aims to minimize the learning curve and ensure that users can efficiently utilize the functionalities from the moment they start using the platform.

4. Automatic Feature Detection:

One of the key gaps between the existing system and the proposed Famalo system lies in the approach to feature detection. In the current system, users are required to manually search for websites and identify the appropriate platforms to access desired features. This

process can be time-consuming, inefficient, and prone to errors. However, the proposed Famalo system implements automatic feature detection based on user input text. Famalo can intelligently identify and suggest relevant features to users, eliminating the need for extensive search and evaluation. This automation significantly enhances the user experience, saving time and effort while ensuring accurate and personalized feature recommendations.

In conclusion the gap between the existing system and the proposed Famalo system is substantial. The proposed system offers significant improvements in terms of efficiency, convenience, comprehensive functionality, user-friendliness, and automatic feature detection. By addressing the limitations and challenges of the current system, Famalo aims to revolutionize the way users access desired features on the web, providing a seamless and integrated platform that enhances productivity and satisfaction. The subsequent pages of this report will delve deeper into the implementation and benefits of the Famalo system, highlighting its potential to bridge the gap and transform the user experience in accessing desired functionalities.

5.ADVANTAGES AND DISADVANTAGES

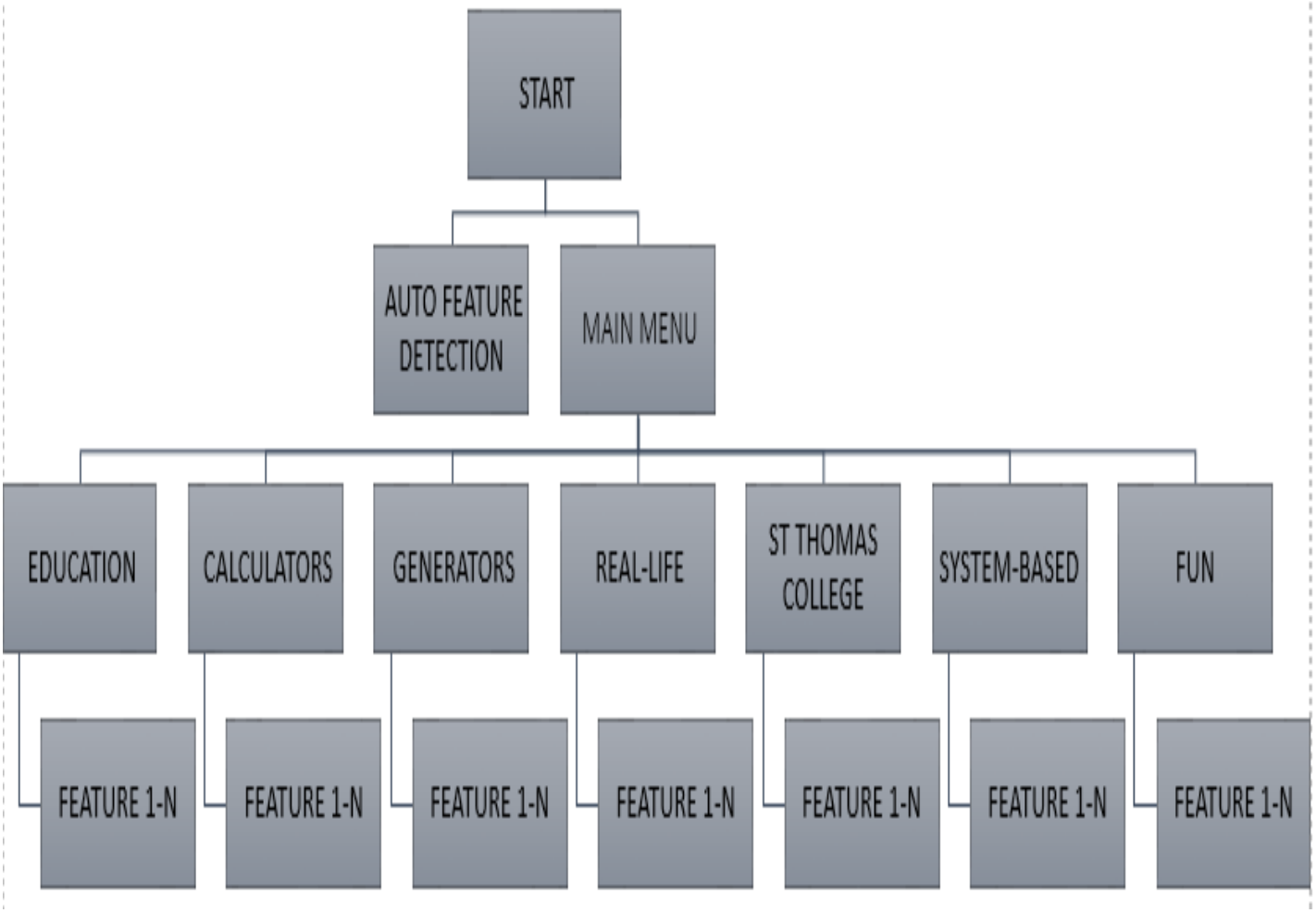
Advantages

- Friend-like interactions : Famalo provides a conversational and interactive experience, making users feel like they are interacting with a friend rather than a machine. This human touch enhances engagement and user satisfaction.
- Quick access : Famalo offers swift and convenient access to a wide range of functionalities, eliminating the need to navigate through multiple websites or applications. Users can effortlessly find and utilize desired features within the chat platform.
- Easy-to-use and intuitive chat platform: Famalo is designed with a user-friendly interface, ensuring that even those with limited technical expertise can easily navigate and utilize its features. The intuitive nature of the platform enhances usability and accessibility.
- Reduction of boredom : Famalo includes engaging activities such as random jokes, dice simulators, and random meme generation. These features provide entertainment and alleviate boredom, making interactions with the bot enjoyable and captivating.
- Simplification of complex features : Famalo simplifies intricate functionalities by automating processes and providing a streamlined user experience. Complex tasks such as password generation, text-to-audio conversion, and QR code generation are made effortless and accessible.

Disadvantages

- A lot of features, which may be overwhelming : The wide array of features offered by Famalo might be overwhelming for some users, especially those who are new to the platform. Navigating through numerous functionalities may require some time and familiarization.
- Commands in Telegram may be confusing for new users : Telegram commands might pose a learning curve for users who are not familiar with the platform. Understanding and utilizing these commands effectively might require some initial guidance and exploration.
- Need internet for deploying and using the bot : Famalo relies on an internet connection for its deployment and usage. This dependency on connectivity may limit accessibility in areas with poor or unreliable internet access. Users must ensure a stable internet connection to fully utilize the bot's functionalities.

6.WORKING DIAGRAM



7.MODULES AND FEATURES

1. Auto Feature Detection

- Automatically identifies and suggests relevant features based on user input, enhancing the user experience by eliminating the need for explicit feature specification.

2. Education

2.1. Online Course Recommendator: Providesrecommendations for online courses based on user preferences, helping users discover relevant and high-quality educational resources.

2.2. YouTube Course Recommendator: Offers curated recommendations for educational courses available on YouTube, allowing users to explore diverse learning opportunities on the platform.

2.3. Cheatsheets: Provides access to concise and handy cheat sheets for various topics, assisting users in quickly referencing important information and concepts.

2.4. GitHub Learning Resources: Enables users to explore and discover learning resources available on GitHub.

3. Generators

3.1. Password Generator: Generates secure and random passwords, ensuring enhanced online security and privacy for users across various platforms and accounts.

3.2. Text to Audio Converter: Converts text into audio format, enabling users to listen to written content, such as articles or documents, offering accessibility options and convenience.

3.3. QR Code Generator: Generates QR codes, facilitating quick information sharing and access to websites, contact details, or other data, enhancing convenience in various contexts such as marketing, event management, and digital interactions.

3.4. Morse Code Generator: Generates Morse code translations from text input, enabling users to learn and communicate using Morse code, offering an interactive and educational experience.

4. Real Life

4.1. Weather Details of Your Current Location: Retrieves accurate and up-to-date weather information based on the user's current location, allowing users to plan their activities and stay informed about weather conditions.

4.2. Random Activity Generator: Generates random activity suggestions for users, offering a source of inspiration for leisure, hobbies, or spontaneous outings, reducing boredom and promoting exploration.

5. System-Based

5.1. Linux Distro Recommendator: Provides recommendations for Linux distributions based on user preferences and requirements, assisting users in selecting the most suitable distribution for their specific needs.

5.2. Linux Commands PDF: Offers access to a PDF document containing a comprehensive collection of Linux commands, serving as a valuable resource for users learning or working with Linux-based systems

.

5.3. External mark needed for specific grade: Calculates the required external marks needed to achieve a specific grade based on user inputs, aiding students in assessing their academic progress and setting goals.

5.4. Playback Speed Calculator: Helps users determine the ideal playback speed for audio and video content, enabling efficient consumption of media based on individual preferences and learning requirements

.

6. ST Thomas College Specific

6.1. Syllabus Sender: Allows students to access and retrieve the syllabus resources specific to ST Thomas College, providing a convenient and centralized platform for syllabus information.

7. FUN

7.1 Random Jokes: Generate and display jokes randomly from a collection of jokes.

7.2 Dice simulator: Simulate rolling dice to generate random numbers based on a dice.

7.3 Corporate BS: Generate random corporate jargon or buzzwords to create realistic-sounding but meaningless phrases.

7.4 Superhero Details: Provide random details about superheroes, such as their name, powers, and backstory.

7.5 Random Meme Generator: Generate and display random memes sourced from internet.

8.RUNNING CODE

Famalo.py

```
import telebot
import time
import random
import pyjokes
from gtts import gTTS
from password_generator import PasswordGenerator
import pyqrcode
import io
from datetime import timedelta,datetime
import requests
from telebot import types
import os
import png
from tabulate import tabulate
import Data_File
from API_of_adhi import API_key_Famalo,coporate_bs_api_key,API_weather_key #API
keys

#bot api
token = API_key_Famalo()
bot = telebot.TeleBot(token)

@bot.message_handler(commands = ['help','start'])
def send_start_help_option(message):
    bot.send_message(message.chat.id,"You can choose my auto feature detection -/action")
    bot.send_message(message.chat.id,"OR")
    bot.send_message(message.chat.id,"You can choose traditional menu -/mainmenu")

@bot.message_handler(commands=['list'])
def send_list_features(message):
    bot.send_message(message.chat.id,Data_File.list_features)
################################################## Action/command detection
#################################
@bot.message_handler(commands=['action']) #feature detection
def input_action(message):
    bot.reply_to(message,"I will try to automatically detect feature from your text")
    bot.send_message(message.chat.id,"Enter your text")
    bot.register_next_step_handler(message,action_input_processing)

def action_input_processing(message):
    text = message.text
    text = text.lower()
    text_list = text.split(" ")
    index = 0
    for keyword in text_list:
        if index + 1<len(text_list): # preventing index error
```

```
            # for detecting words like "online course" where 2 words are seperated with space
            _2_keywords = f"{text_list[index]} {text_list[index + 1]}"
            if index + 2 <len(text_list): # preventing index error
                #for detecting words like 'git hub learn' where 3 words are there
                _3_keywords = f"{text_list[index]} {text_list[index+1]} {text_list[index+2]}"
        if len(text_list) < 4 : # to handle if input is too small
            _3_keywords ="#####" # random text which is not available in possible_keywords
list
            if len(text_list) < 3 : # to handle if input is too small
                _2_keywords = "#####" # random text which is not available in
possible_keywords list
        if keyword in Data_File.youtube_course_possible_keywords :
            bot.send_message(message.chat.id,"Did you mean youtube course
reccomendator(Y/N)")
            possible_command = 'youtube'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.online_course_possible_keywords or _2_keywords in
Data_File.online_course_possible_keywords :
            bot.send_message(message.chat.id,"Did you mean online course
reccomendator(Y/N)")
            possible_command = 'onlinecourse'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.cheat_sheet_possible_keywords or _2_keywords in
Data_File.cheat_sheet_possible_keywords :
            bot.send_message(message.chat.id,"Did you mean Cheat sheet sender (Y/N)")
            possible_command = 'cheatsheet'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.github_learn_possible_keywords or _2_keywords in
Data_File.github_learn_possible_keywords or _3_keywords in
Data_File.github_learn_possible_keywords :
            bot.send_message(message.chat.id,"Did you mean top github learning resources
(Y/N)")
            possible_command = 'githublearn'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.external_mark_calculator_possbile_keywords or
_2_keywords in Data_File.external_mark_calculator_possbile_keywords:
            bot.send_message(message.chat.id,"Did you mean external mark calculator(Y/N)")
            possible_command = 'gradecalculator'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.playback_speed_possible_keywords or _2_keywords in
Data_File.playback_speed_possible_keywords or _3_keywords in
Data_File.playback_speed_possible_keywords :
            bot.send_message(message.chat.id,"Did you mean playbackspeed calculator(Y/N)")
            possible_command = 'playbackspeedcalculator'
```

```
bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.password_generator_possible_keywords or _2_keywords in
Data_File.password_generator_possible_keywords :
            bot.send_message(message.chat.id,"Did you mean random password generator
(Y/N)")
            possible_command = 'password'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.text_to_audio_possible_keywords or _2_keywords in
Data_File.text_to_audio_possible_keywords or _3_keywords in
Data_File.text_to_audio_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean text to audio converter (Y/N)")
            possible_command = 'textaudio'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.qr_code_generator_possible_keywords or _2_keywords in
Data_File.qr_code_generator_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean QR code Generator (Y/N)")
            possible_command = 'qrcodegenerator'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.morse_code_generator_possible_keywords or _2_keywords
in Data_File.morse_code_generator_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean Morse Code generator (Y/N)")
            possible_command = 'morsecode'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.weather_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean Weather deatils of your current
location (Y/N)")
            possible_command = 'weather'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.random_activity_possible_keywords or _2_keywords in
Data_File.random_activity_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean random activity generator
(Y/N)")
            possible_command = 'activity'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.st_thomas_syllabus_possible_keywords :
            bot.send_message(message.chat.id,"Did you mean syllabus sender of ST thomas
college (Y/N)")
            possible_command = 'syllabus'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
```

```
            break
        elif keyword in Data_File.linux_distro_reccomendator_possible_keywords or
_2_keywords in  Data_File.linux_distro_reccomendator_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean linux distro reccomendator
(Y/N)")
            possible_command = 'linuxdistro'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.linux_commands_pdf_sender_possible_keywords or
_2_keywords in  Data_File.linux_commands_pdf_sender_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean linux commands pdf (Y/N)")
            possible_command = 'linuxcommandpdf'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.random_joke_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean random joke generator (Y/N)")
            possible_command = 'joke'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.Dice_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean Dice simulator (Y/N)")
            possible_command = 'dice'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.Coporate_bs_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean coporate bs (Y/N)")
            possible_command = 'coporatebs'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.super_hero_possible_keywords or _2_keywords in
Data_File.super_hero_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean super hero deatils (Y/N)")
            possible_command = 'super'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        elif keyword in Data_File.Random_meme_possible_keywords:
            bot.send_message(message.chat.id,"Did you mean random meme generator (Y/N)")
            possible_command = 'meme'

bot.register_next_step_handler(message,Y_N_possible_command,possible_command)
            break
        else:
            if index == len(text_list) - 1:
                bot.send_message(message.chat.id,"Feature couldn't be detected")
                bot.send_message(message.chat.id,"Please try again! -/action")
                break
        index+=1
```

```
def Y_N_possible_command(message,possible_command):
    Y_N = message.text
    Y_N = Y_N.upper()
    if Y_N =='Y' and possible_command == 'youtube':
        bot.send_message(message.chat.id,"Here you go youtube course reccomendator")
        youtubecourse_menu(message)
    elif Y_N == 'Y' and possible_command == 'onlinecourse':
        bot.send_message(message.chat.id,"Here you go online course reccomendator")
        onlinecourse_menu(message)
    elif Y_N == 'Y' and possible_command == 'cheatsheet':
        bot.send_message(message.chat.id,"Here you go cheatsheet sender")
        cheatsheet_menu(message)
    elif Y_N == 'Y' and possible_command == 'githublearn':
        bot.send_message(message.chat.id,"Here you go Top github learning resources")
        github_learn_send_links(message)
    elif Y_N == 'Y' and possible_command == 'gradecalculator':
        bot.send_message(message.chat.id,"Here you go External mark Calculator")
        gradecalculator(message)
    elif Y_N == 'Y' and possible_command == 'playbackspeedcalculator':
        bot.send_message(message.chat.id,"Here you go Playbackspeed Calculator")
        playbackspeedcalculator(message)
    elif Y_N == 'Y' and possible_command == 'password':
        bot.send_message(message.chat.id,"Here you go random password generator")
        password_generator1(message)
    elif Y_N == 'Y' and possible_command == 'textaudio':
        bot.send_message(message.chat.id,"Here you go text to audio converter")
        textaudio(message)
    elif Y_N == 'Y' and possible_command == 'qrcodegenerator':
        bot.send_message(message.chat.id,"Here you go QR code generator")
        qrcodegenerator(message)
    elif Y_N == 'Y' and possible_command == 'morsecode':
        bot.send_message(message.chat.id,"Here you go Morse code generator")
        text_morse_code(message)
    elif Y_N == 'Y' and possible_command == 'weather':
        bot.send_message(message.chat.id,"Here you go Weather")
        weather(message)
    elif Y_N == 'Y' and possible_command == 'activity':
        bot.send_message(message.chat.id,"Here you go random activity generator")
        activity(message)
    elif Y_N == 'Y' and possible_command == 'syllabus':
        bot.send_message(message.chat.id,"Here you go syllabus sender of ST thomas
college")
        syllabus_Department_menu(message)
    elif Y_N == 'Y' and possible_command == 'linuxdistro':
        bot.send_message(message.chat.id,"Here you go linux distro reccomendator")
        linux_distro_reccomendator_menu(message)
    elif Y_N == 'Y' and possible_command == 'linuxcommandpdf':
        bot.send_message(message.chat.id,"Here you go linux commands pdf")
        send_linux_commands_pdf(message)
    elif Y_N == 'Y' and possible_command == 'joke':
        bot.send_message(message.chat.id,"Here you go random joke generator")
        joke(message)
    elif Y_N == 'Y' and possible_command == 'dice':
        bot.send_message(message.chat.id,"Here you go Dice simulator")
```

```
        dice(message)
    elif Y_N == 'Y' and possible_command == 'coporatebs':
        bot.send_message(message.chat.id,"Here you go coporate bs")
        coporatebs(message)
    elif Y_N == 'Y' and possible_command == 'super':
        bot.send_message(message.chat.id,"Here you go superhero deatils")
        super(message)
    elif Y_N == 'Y' and possible_command == 'meme':
        bot.send_message(message.chat.id,"Here you go random meme generator")
        meme(message)
    elif Y_N == 'N':
        bot.send_message(message.chat.id,"Sorry , Feature not available")
        bot.send_message(message.chat.id,"You can feel free to contact the developer to
request for this feature - /contact")
    else:
        bot.send_message(message.chat.id,"Invalid command try again!")
        input_action(message)
##############################################################################
#################################### 
################################################## Main
Menu##################################################

@bot.message_handler(commands=['mainmenu']) #main menu
def send_main_menu(message):#message default parameter
    bot.reply_to(message,"Hi!\nI'm Famalo \nYour all in one buddy \nYou can choose one of
the categories from below ")
    bot.send_message(message.chat.id,"Education                -/education")
    bot.send_message(message.chat.id,"Calculators        -/calculators")
    bot.send_message(message.chat.id,"Generators   -/generators")
    bot.send_message(message.chat.id,"Real-life  -/reallife")
    bot.send_message(message.chat.id,"ST Thomas college specific  -/stthomascollege")
    bot.send_message(message.chat.id,"System-based           -/system")
    bot.send_message(message.chat.id,"FUN                  -/fun")
    bot.send_message(message.chat.id,"Contact     -/contact")

##############################################################################
#################################### 

################################################## Sub Menus
################################################## 

@bot.message_handler(commands=['education'])
def send_education_menu(message):
    bot.send_message(message.chat.id,"Education MENU")
    bot.send_message(message.chat.id,"Online Course Recommendator -/onlinecourse")
    bot.send_message(message.chat.id,"Youtube Course Recommendator -/youtubecourse")
    bot.send_message(message.chat.id,"Cheat sheets -/cheatsheet")
    bot.send_message(message.chat.id,"Github learning Resources -/githublearn")

@bot.message_handler(commands=['calculators'])
def send_calculator_menu(message):
    bot.send_message(message.chat.id,"Calculator MENU")
    bot.send_message(message.chat.id,"External mark needed for specific grade
-/gradecalculator") #feature implemented from scratch(tabulate and html for output)
```

```
    bot.send_message(message.chat.id,"Playbackspeed Calculator       -
/playbackspeedcalculator") #feature implemented from scratch


@bot.message_handler(commands=['generators'])
def send_Generator_menu(message):
    bot.send_message(message.chat.id,"Generator MENU")
    bot.send_message(message.chat.id,"PasswordGenerator       - /password") #uses
password_generator libary
    bot.send_message(message.chat.id,"Text to audio converter      - /textaudio") # use GTTS
libary
    bot.send_message(message.chat.id,"QR Code Generator - /qrcodegenerator") #uses pyqr
code libary
    bot.send_message(message.chat.id,"Morse Code Generator - /morsecode")

@bot.message_handler(commands=['reallife'])
def send_real_life_menu(message):
    bot.send_message(message.chat.id,"Real-Life MENU")
    bot.send_message(message.chat.id,"Weather deatils of your current location
-/weather") # imeplemented using buttons from scratch and openweather api for data
    bot.send_message(message.chat.id,"Random Activity Generator                -/activity")
#sends http request to activity generator api with buttons implemented from scratch


@bot.message_handler(commands=['stthomascollege'])
def send_st_thomas_college_menu(message):
    bot.send_message(message.chat.id,"ST Thomas College specific MENU")
    bot.send_message(message.chat.id,"Syllabus Sender -/syllabus")

@bot.message_handler(commands=['system'])
def send_system_menu(message):
    bot.send_message(message.chat.id,"System-based MENU")
    bot.send_message(message.chat.id,"Linux distro Recommendator -/linuxdistro")
    bot.send_message(message.chat.id,"Linux command pdf - /linuxcommandpdf")

@bot.message_handler(commands=['fun'])
def send_fun_menu(message):
    bot.send_message(message.chat.id,"FUN MENU")
    bot.send_message(message.chat.id,"Ask me a joke     - /joke") #uses pyijokes libary
    bot.send_message(message.chat.id,"Roll a dice      - /dice") # uses random module
    bot.send_message(message.chat.id,"Coporate BS            - /coporatebs") #sends http
request to Coporate bs api
    bot.send_message(message.chat.id,"Superhero Deatils            -/super") #sends http
request to super hero deatils api
    bot.send_message(message.chat.id,"Random Meme Generator       -/meme")#sends http
request to meme generator api

##############################################################################
#####################################

################################################## Education Menu
Features##################################################

### Online Course ###
```

```
@bot.message_handler(commands=['onlinecourse'])
def onlinecourse_menu(message):
    keyboard = types.InlineKeyboardMarkup() #defining buttons
    option1 = types.InlineKeyboardButton("Python", callback_data='Python_online')
    option2 = types.InlineKeyboardButton("Java", callback_data='Java_online')
    option3 = types.InlineKeyboardButton("Javascript",callback_data='Javascript_online')
    option4 = types.InlineKeyboardButton("C/C++",callback_data = 'C_online')
    keyboard.add(option1, option2,option3,option4)
    bot.send_message(message.chat.id,"Select your desired Topic:",reply_markup=keyboard)

@bot.callback_query_handler(func=lambda call: call.data.endswith('_online'))
def handle_onlinecourse_buttons(call):
    if call.data == 'Python_online':
        keyboard_2 = types.InlineKeyboardMarkup()
        option2_1 = types.InlineKeyboardButton("Udemy",
callback_data='Python_Udemy_online')
        option2_2 = types.InlineKeyboardButton("Coursera",
callback_data='Python_Coursera_online')
        keyboard_2.add(option2_1, option2_2)
        bot.send_message(call.message.chat.id, "Select your desired Platform:",
reply_markup=keyboard_2)

    elif call.data == 'Python_Udemy_online':
        random_index = random.randint(0, len(Data_File.Online_course_Python_Udemy) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Online_course_Python_Udemy[random_index])

    elif call.data == 'Python_Coursera_online':
        random_index = random.randint(0, len(Data_File.Online_course_Python_Coursera) - 1
)
        bot.send_message(call.message.chat.id,
Data_File.Online_course_Python_Coursera[random_index])

    elif call.data == 'Java_online':
        keyboard_2 = types.InlineKeyboardMarkup()
        option2_1 = types.InlineKeyboardButton("Udemy",
callback_data='Java_Udemy_online')
        option2_2 = types.InlineKeyboardButton("Coursera",
callback_data='Java_Coursera_online')
        keyboard_2.add(option2_1, option2_2)
        bot.send_message(call.message.chat.id, "Select your desired Platform:",
reply_markup=keyboard_2)

    elif call.data == 'Java_Udemy_online':
        random_index = random.randint(0, len(Data_File.Online_course_Java_Udemy) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Online_course_Java_Udemy[random_index])

    elif call.data == 'Java_Coursera_online':
        random_index = random.randint(0, len(Data_File.Online_course_Java_Coursera) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Online_course_Java_Coursera[random_index])

    elif call.data == 'Javascript_online':
```

```
        keyboard_2 = types.InlineKeyboardMarkup()
        option2_1 = types.InlineKeyboardButton("Udemy",
callback_data='Javascript_Udemy_online')
        option2_2 = types.InlineKeyboardButton("Coursera",
callback_data='Javascript_Coursera_online')
        keyboard_2.add(option2_1, option2_2)
        bot.send_message(call.message.chat.id, "Select your desired Platform:",
reply_markup=keyboard_2)

    elif call.data == 'Javascript_Udemy_online':
        random_index = random.randint(0, len(Data_File.Online_course_Javascript_Udemy) -
1 )
        bot.send_message(call.message.chat.id,
Data_File.Online_course_Javascript_Udemy[random_index])

    elif call.data == 'Javascript_Coursera_online':
        random_index = random.randint(0, len(Data_File.Online_course_Javascript_Coursera)
- 1 )
        bot.send_message(call.message.chat.id,
Data_File.Online_course_Javascript_Coursera[random_index])

    elif call.data == 'C_online':
        keyboard_2 = types.InlineKeyboardMarkup()
        option2_1 = types.InlineKeyboardButton("Udemy", callback_data='C_Udemy_online')
        option2_2 = types.InlineKeyboardButton("Coursera",
callback_data='C_Coursera_online')
        keyboard_2.add(option2_1, option2_2)
        bot.send_message(call.message.chat.id, "Select your desired Platform:",
reply_markup=keyboard_2)

    elif call.data == 'C_Udemy_online':
        random_index = random.randint(0, len(Data_File.Online_course_C_Udemy) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Online_course_C_Udemy[random_index])

    elif call.data == 'C_Coursera_online':
        random_index = random.randint(0, len(Data_File.Online_course_C_Coursera) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Online_course_C_Coursera[random_index])
##############################
### Youtube_recommendator ###
@bot.message_handler(commands='youtubecourse')
def youtubecourse_menu(message):
    keyboard = types.InlineKeyboardMarkup() #defining buttons
    option1 = types.InlineKeyboardButton("Python", callback_data='Python_youtube')
    option2 = types.InlineKeyboardButton("Java", callback_data='Java_youtube')
    option3 = types.InlineKeyboardButton("Javascript",callback_data='Javascript_youtube')
    option4 = types.InlineKeyboardButton("C/C++",callback_data = 'C_youtube')
    keyboard.add(option1,option2,option3,option4)
    bot.send_message(message.chat.id,"Select your desired Topic:",reply_markup=keyboard)

# Define the callback query handlers for different topics
@bot.callback_query_handler(func=lambda call: call.data.endswith('_youtube'))
def handle_youtubecourse_buttons(call):
```

```
    if call.data == 'Python_youtube':
        random_index = random.randint(0, len(Data_File.Youtube_video_python) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Youtube_video_python[random_index])

    elif call.data == 'Java_youtube':
        random_index = random.randint(0, len(Data_File.Youtube_video_Java) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Youtube_video_Java[random_index])

    elif call.data == 'Javascript_youtube':
        random_index = random.randint(0, len(Data_File.Youtube_video_Javascript) - 1 )
        bot.send_message(call.message.chat.id,
Data_File.Youtube_video_Javascript[random_index])

    elif call.data == 'C_youtube':
        random_index = random.randint(0, len(Data_File.Youtube_video_C) - 1 )
        bot.send_message(call.message.chat.id, Data_File.Youtube_video_C[random_index])
####################
### Cheatsheets ####
@bot.message_handler(commands=['cheatsheet'])
def cheatsheet_menu(message):
    keyboard = types.InlineKeyboardMarkup() #defining buttons
    option1 = types.InlineKeyboardButton("Python", callback_data='Python_cheatsheet')
    option2 = types.InlineKeyboardButton("Java", callback_data='Java_cheatsheet')
    option3 =
types.InlineKeyboardButton("Javascript",callback_data='Javascript_cheatsheet')
    option4 = types.InlineKeyboardButton("C/C++",callback_data = 'C_cheatsheet')
    option5 = types.InlineKeyboardButton("Git",callback_data = 'Git_cheatsheet')
    keyboard.add(option1, option2,option3,option4,option5)
    bot.send_message(message.chat.id,"Select your desired Topic:",reply_markup=keyboard)
@bot.callback_query_handler(func=lambda call: call.data.endswith('_cheatsheet'))
def handle_cheatsheet_buttons(call):
    if call.data == 'Python_cheatsheet':
        bot.send_document(call.message.chat.id,Data_File.Cheat_sheet_Python)

    elif call.data == 'Java_cheatsheet':
        bot.send_document(call.message.chat.id,Data_File.Cheat_sheet_Java)

    elif call.data == 'Javascript_cheatsheet':
        bot.send_document(call.message.chat.id,Data_File.Cheat_sheet_Javascript)

    elif call.data == 'C_cheatsheet':
        bot.send_document(call.message.chat.id,Data_File.Cheat_sheet_Cpp)

    elif call.data == 'Git_cheatsheet':
        bot.send_document(call.message.chat.id,Data_File.Cheat_sheet_Git)
##############
## Github learning Resources ###
@bot.message_handler(commands=['githublearn'])
def github_learn_send_links(message):
    bot.send_message(message.chat.id,"Top Github learning Resources")
    bot.send_message(message.chat.id,"Learn ML in 100days\nhttps://github.com/Avik-
Jain/100-Days-Of-ML-Code")
```

```
    bot.send_message(message.chat.id,"Developer
Roadmaps\nhttps://github.com/kamranahmedse/developer-roadmap")
    bot.send_message(message.chat.id,"The
Algorithms\nhttps://github.com/TheAlgorithms")
    bot.send_message(message.chat.id,"Ebooks
Foundation\nhttps://github.com/EbookFoundation")
    bot.send_message(message.chat.id,"Public APIs\nhttps://github.com/public-apis/public-
apis")
############################

##############################################################################
#################################################

################################################## Calculator Menu Features
##################################################
### Grade Calculator ###
@bot.message_handler(commands=['gradecalculator'])
def gradecalculator(message):
    bot.send_message(message.chat.id,"This calculator will tell the external marks required
for each grade according to your internal mark")
    bot.send_photo(message.chat.id,"https://imgtr.ee/images/2023/06/05/bAsnI.png")
    bot.send_message(message.chat.id,"This will be used as reference for calculation")
    bot.send_message(message.chat.id,"Enter your total internal mark(15 or 20)")
    bot.register_next_step_handler(message,total_internal_mark_validation)

def total_internal_mark_validation(message):
    total_internal_mark = message.text
    total_internal_mark = int(total_internal_mark)
    if total_internal_mark in (20,15):
        bot.send_message(message.chat.id,"OK")
        bot.send_message(message.chat.id,f"Enter your internal mark (out of
{total_internal_mark})")
        bot.register_next_step_handler(message,internal_mark_validation,total_internal_mark)
    else:
        bot.send_message(message.chat.id,"Wrong input! Try again")
        gradecalculator(message)

def internal_mark_validation(message,total_internal_mark):
    internal_mark = message.text
    internal_mark = int(internal_mark)
    if internal_mark <= total_internal_mark and internal_mark >= 0 :
        bot.send_message(message.chat.id,"OK")
        list_with_internal_total_internal = [internal_mark,total_internal_mark]
        send_output(message,list_with_internal_total_internal)
    else:
        bot.send_message(message.chat.id,"Wrong input! Try again")
        gradecalculator(message)


def send_output(message,list_with_internal_total_internal):
    internal_mark = list_with_internal_total_internal[0]
    total_internal_mark = list_with_internal_total_internal[1]
    if total_internal_mark == 20:
        total_mark = 100
```

```
        total_external_mark = 80
    elif total_internal_mark == 15:
        total_mark = 75
        total_external_mark = 60
    O_grade = (total_mark  * 95/100 ) - internal_mark
    A1_grade = (total_mark  * 85/100 ) - internal_mark
    A2_grade = (total_mark  * 75/100 ) - internal_mark
    B1_grade = (total_mark  * 65/100 ) - internal_mark
    B2_grade = (total_mark  * 55/100 ) - internal_mark
    C_grade = (total_mark  * 45/100 ) - internal_mark
    P_grade = (total_mark  * 35/100) - internal_mark
    F_grade = f"Below {P_grade}"
    list_with_External_Marks =
[O_grade,A1_grade,A2_grade,B1_grade,B2_grade,C_grade,P_grade,F_grade]
    for i in range(len(list_with_External_Marks)-1):
        if list_with_External_Marks[i] > total_external_mark:
            list_with_External_Marks[i] = "Not possible"
        else:
            list_with_External_Marks[i] = "Above " + str(list_with_External_Marks[i])
    data = [["O",list_with_External_Marks[0]],
            ["A+",list_with_External_Marks[1]],
            ["A",list_with_External_Marks[2]],
            ["B+",list_with_External_Marks[3]],
            ["B",list_with_External_Marks[4]],
            ["C",list_with_External_Marks[5]],
            ["P",list_with_External_Marks[6]],
            ["F",list_with_External_Marks[7]]]
    #converts list to a table with borders
    table_string = tabulate(data, headers=["Grade","External marks Needed"],
tablefmt="grid",colalign=("left", "left"))
    table_html = f"<pre>{table_string}</pre>" #used html for making borders uniform
(according to data)
    bot.send_message(message.chat.id, text=table_html, parse_mode="HTML")
################################
### Playbackspeed calculator ##
@bot.message_handler(commands=['playbackspeedcalculator'])
def playbackspeedcalculator(message):
    bot.send_message(message.chat.id,"So you wanna know is it worth to watch your lecture
in fast mode ")
    message = bot.send_message(message.chat.id,"Type the length of your video in this
format (00:00:00)\nlimit (23:59:59)")
    bot.register_next_step_handler(message,Validating_input)

def Validating_input(message): # error management
    global time1 # for using it in all functions
    time1 = message.text
    if len(time1) == 8:
        conditions = (time1[0] in '012',time1[1] in '0123456789', # conditions for true input
                      time1[2] == ':',time1[3] in '012345',
                      time1[4] in '0123456789',time1[5] == ':',
                      time1[6] in '012345',time1[7] in '0123456789')
        if all(conditions):
            bot.send_message(message.chat.id,"OK")
            input_playback_speed(message)
```

```
        else:
            bot.send_message(message.chat.id,"Wrong input format, Try again")
            playbackspeedcalculator(message)
    else:
        bot.send_message(message.chat.id,"Wrong input format, Try again")
        playbackspeedcalculator(message)

def input_playback_speed(message):
    message = bot.send_message(message.chat.id,"Type Your playbackspeed in this
format(1.50)")
    bot.register_next_step_handler(message,validating_input2)

def validating_input2(message): #error management
    speed = message.text
    if len(speed) == 4:
        conditions = (len(speed) == 4, speed[1] == '.', speed[0] in '0123456789', # conditions
for true
                      speed[2] in '0123456789', speed[3] in '0123456789')
        if all(conditions):
            bot.send_message(message.chat.id,"OK")
            calculating_sending_output(speed,message)

        else:
            bot.send_message(message.chat.id,"Wrong input format, Try again")
            input_playback_speed(message)
    else:
        bot.send_message(message.chat.id,"Wrong input format, Try again")
        input_playback_speed(message)


def calculating_sending_output(speed,message): # final calculation and output
    # converting to seconds
    original_total_seconds = (int(time1[0]+time1[1]) * 3600)+ (int(time1[3]+time1[4]) * 60)
+ (int(time1[6]+time1[7]))
    calculated_result_seconds = float(original_total_seconds)/float(speed) # result
    rounded_calculated_result_seconds = round(calculated_result_seconds, 0) # removing
decimal part
    result_time = timedelta(seconds=rounded_calculated_result_seconds) # changing format
    bot.send_message(message.chat.id,f"Calculated Time = {result_time}")  # send
    datetime1 = datetime.strptime(time1,"%H:%M:%S")
    datetime2 = datetime.strptime(str(result_time),"%H:%M:%S")
    saved_time = datetime1 - datetime2 # subracting to find saved time
    bot.send_message(message.chat.id,f"Time you can save at {speed} = {saved_time}")
#send saved time

################################
#################################################################################
#################################################

################################################## Generator Menu Features
##################################################
### Password generator ###
@bot.message_handler(commands=['password'])
def password_generator1(message):
```

```
    bot.reply_to(message,"So you decided to bo more secure than a normal user who puts
their pet's name as password")
    bot.send_sticker(message.chat.id,"https://media.tenor.com/9Ez46wr-voMAAAAC/
lock.gif")
    bot.send_message(message.chat.id,"You can copy the below password and use")
    pwo = PasswordGenerator() #calling libary
    pwo1=pwo.generate() # randomly generating password
    bot.send_message(message.chat.id,pwo1)
####################
### Text to audio ##
@bot.message_handler(commands=['textaudio'])
def textaudio(message):
    bot.reply_to(message,"This will convert your text to audio")
    msg = bot.send_message(message.chat.id,"Enter the text")
    bot.register_next_step_handler(msg,text_to_audio_processing)

def text_to_audio_processing(message): #processing input of user
    text=message.text
    audio=gTTS(text=text,lang="en",slow=False) # converting it to audio
    audio.save("audio.mp3") # saving it locally
    with open("audio.mp3", "rb") as audio_file:
    # Send the audio file
        bot.send_audio(message.chat.id, audio_file)
    os.remove("audio.mp3") #delete after sending

#########################
### QR code Generator ###
@bot.message_handler(commands=['qrcodegenerator'])
def qrcodegenerator(message):
    bot.reply_to(message,"Say goodbye to typing long links or text. Say hello to the easy
way. Just scan and let the magic unfold")
    message = bot.send_message(message.chat.id,"Type your link or text")
    bot.register_next_step_handler(message,Processing_qrcodegenerator)


def Processing_qrcodegenerator(message): # handling user input
    text_link = message.text
    QRCode = pyqrcode.create(text_link) #creating qr code
    qr_code_image = io.BytesIO() # to avoid saving of file locally before sending it
    QRCode.png(qr_code_image, scale=10) # scale is size
    qr_code_image.seek(0)
    bot.send_photo(message.chat.id, qr_code_image)

############################
### Morse Code Generator###
@bot.message_handler(commands=['morsecode'])
def text_morse_code(message):
    bot.reply_to(message,"This will convert your text to morsecode")
    msg = bot.send_message(message.chat.id,"Enter the text")
    bot.register_next_step_handler(msg,text_morse_code_converter)
def text_morse_code_converter(message):
    text = message.text
    morse_code = ''
    for i in text.upper():
```

```
        if i in Data_File.morse_dicitonary:
            morse_code += Data_File.morse_dicitonary[i] + ' '
    bot.send_message(message.chat.id,morse_code)
############################

##############################################################################
#################################################


################################################## Real life Menu Features
##################################################


@bot.message_handler(commands=['weather'])
### Weather ###
def weather(message):
    #defining Send Location button
    keyboard_markup = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    button_location = types.KeyboardButton(text="Send Location", request_location=True)
    keyboard_markup.add(button_location)

    # Send the message with the request for location
    bot.send_message(message.chat.id, "Please send your location.",
reply_markup=keyboard_markup)
    bot.register_next_step_handler(message,handle_location)

def handle_location(message):
    #taking input from user and storing data in specific variables
    location = message.location
    latitude = location.latitude
    longitude = location.longitude
    remove_keyboard_command = types.ReplyKeyboardRemove()
    bot.send_message(message.chat.id, "Location identified",
reply_markup=remove_keyboard_command)
    send_output_weather(message,latitude,longitude)

def send_output_weather(message,latitude,longitude):
    API_weather = API_weather_key()
    response = requests.get(f"https://api.openweathermap.org/data/2.5/weather?
lat={latitude}&lon={longitude}&appid={API_weather}&units=metric")
    data = response.json()
    #assigning each variable to a data from response
    name = data['name']
    description = data['weather'][0]['description']
    temp = data['main']['temp']
    feels_like = data['main']['feels_like']
    temp_min = data['main']['temp_min']
    temp_max = data['main']['temp_max']
    pressure = data['main']['pressure']
    humidity = data['main']['humidity']
    wind_speed = data['wind']['speed']

    # Sending the extracted data to user
    bot.send_message(message.chat.id, f"Name: {name}")
```

```
    bot.send_message(message.chat.id, f"Description: {description}")
    bot.send_message(message.chat.id, f"Temperature: {temp}°C")
    bot.send_message(message.chat.id, f"Feels like: {feels_like}°C")
    bot.send_message(message.chat.id, f"Minimum Temperature: {temp_min}°C")
    bot.send_message(message.chat.id, f"Maximum Temperature: {temp_max}°C")
    bot.send_message(message.chat.id, f"Pressure: {pressure} hPa")
    bot.send_message(message.chat.id, f"Humidity: {humidity}%")
    bot.send_message(message.chat.id, f"Wind Speed: {wind_speed} m/s")
    #forecast data can be added in the future

##################################
### Random Activity generator ###
@bot.message_handler(commands=['activity'])
def activity(message):
    #inline keyboard buttons
    keyboard = types.InlineKeyboardMarkup()
    option1 = types.InlineKeyboardButton("Education", callback_data='education_activity')
    option2 = types.InlineKeyboardButton("Recreational",
callback_data='recreational_activity')
    option3 = types.InlineKeyboardButton("Social",callback_data='social_activity')
    option4 = types.InlineKeyboardButton("DIY",callback_data = 'diy_activity')
    option5 = types.InlineKeyboardButton("Charity",callback_data='charity_activity')
    option6 = types.InlineKeyboardButton("Cooking",callback_data='cooking_activity')
    option7 = types.InlineKeyboardButton("Relaxation",callback_data='relaxation_activity')
    option8 = types.InlineKeyboardButton("Music",callback_data='music_activity')
    option9 = types.InlineKeyboardButton("Busywork",callback_data='busywork_activity')
    option10 = types.InlineKeyboardButton("Any",callback_data='any_activity')
    #defining keyboard
    keyboard.add(option1,
option2,option3,option4,option5,option6,option7,option8,option9,option10)
    bot.send_message(message.chat.id,"So your bored and don't know what to do\nI will help
you")
    bot.send_message(message.chat.id, 'Choose a Catergory of Activity:',
reply_markup=keyboard) #calling buttons below this message

# Define the callback query handlers for different activity types
@bot.callback_query_handler(func=lambda call: call.data.endswith('_activity'))
def handle_activity_buttons(call):
    if call.data == "any_activity":
        response = requests.get('http://www.boredapi.com/api/activity/')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "education_activity":
        type = 'education'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "recreational_activity":
        type = 'recreational'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])
```

```
    elif call.data == "social_activity":
        type = 'social'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "diy_activity":
        type = 'diy'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "charity_activity":
        type = 'charity'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "cooking_activity":
        type = 'cooking'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "relaxation_activity":
        type = 'relaxation'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "music_activity":
        type = 'music'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])

    elif call.data == "busywork_activity":
        type = 'busywork'
        response = requests.get(f'http://www.boredapi.com/api/activity?type={type}')
        data = response.json()
        bot.send_message(call.message.chat.id, data['activity'])
#################

##############################################################################
#################################################

################################################ St thomas college Specific
Menu Features#################################################

@bot.message_handler(commands=['syllabus'])
def syllabus_Department_menu(message):
    keyboard = types.InlineKeyboardMarkup()
    # Add the department buttons
```

```
    button1 = types.InlineKeyboardButton(text='Data Science',
callback_data='data_science_syllabus')
    button2 = types.InlineKeyboardButton(text='Botany', callback_data='botany_syllabus')
    button3 = types.InlineKeyboardButton(text='Chemistry',
callback_data='chemistry_syllabus')
    button4 = types.InlineKeyboardButton(text='Commerce',
callback_data='commerce_syllabus')
    button5 = types.InlineKeyboardButton(text='Computer Application',
callback_data='computer_application_syllabus')
    button6 = types.InlineKeyboardButton(text='Computer Science',
callback_data='computer_science_syllabus')
    button7 = types.InlineKeyboardButton(text='Criminology',
callback_data='criminology_syllabus')
    button8 = types.InlineKeyboardButton(text='Economics',
callback_data='economics_syllabus')
    button9 = types.InlineKeyboardButton(text='Electronics',
callback_data='electronics_syllabus')
    button10 = types.InlineKeyboardButton(text='English', callback_data='english_syllabus')
    button11 = types.InlineKeyboardButton(text='Forensic Science',
callback_data='forensic_science_syllabus')
    button12 = types.InlineKeyboardButton(text='BBA', callback_data='bba_syllabus')
    button13 = types.InlineKeyboardButton(text='Mathematics',
callback_data='mathematics_syllabus')
    button14 = types.InlineKeyboardButton(text='Media', callback_data='media_syllabus')
    button15 = types.InlineKeyboardButton(text='Physics', callback_data='physics_syllabus')
    button16 = types.InlineKeyboardButton(text='Psychology',
callback_data='psychology_syllabus')
    button17 = types.InlineKeyboardButton(text='Social Work',
callback_data='social_work_syllabus')
    button18 = types.InlineKeyboardButton(text='Statistics',
callback_data='statistics_syllabus')
    button19 = types.InlineKeyboardButton(text='Zoology',
callback_data='zoology_syllabus')
    button20 = types.InlineKeyboardButton(text='Commerce SF',
callback_data='commercesf_syllabus')

    # Add the buttons to the keyboard
    keyboard.add(button1, button2, button3, button4, button5, button6, button7, button8,
button9,
                 button10, button11, button12, button13, button14, button15, button16, button17,
                 button18, button19,button20)
    bot.send_message(message.chat.id, 'Please select a department:',
reply_markup=keyboard)

@bot.callback_query_handler(func=lambda call: call.data.endswith('_syllabus'))
def syllabus_button_callback(call):
    if call.data == 'data_science_syllabus':
        bot.send_document(call.message.chat.id, Data_File.Data_science_bvoc_data_science)
    elif call.data == 'botany_syllabus':
        bot.send_document(call.message.chat.id, Data_File.botany_bsc_botany)
        bot.send_document(call.message.chat.id, Data_File.botany_msc_botany)
    elif call.data == 'chemistry_syllabus':
        bot.send_document(call.message.chat.id, Data_File.chemistry_bsc_chem)
        bot.send_document(call.message.chat.id, Data_File.chemistry_msc_chem)
```

```
    elif call.data == 'commerce_syllabus':
        bot.send_document(call.message.chat.id, Data_File.commerce_bcom_banking)
        bot.send_document(call.message.chat.id, Data_File.commerce_bcom_finance)
        bot.send_document(call.message.chat.id, Data_File.commerce_mcom)
    elif call.data == 'commercesf_syllabus':
        bot.send_document(call.message.chat.id, Data_File.commerce_sf_bcom)
    elif call.data == 'computer_application_syllabus':
        bot.send_document(call.message.chat.id, Data_File.computer_application_bca)
        bot.send_document(call.message.chat.id, Data_File.computer_application_msc_cs)
    elif call.data == 'computer_science_syllabus':
        bot.send_document(call.message.chat.id, Data_File.computer_science_bsc_cs)
        bot.send_document(call.message.chat.id, Data_File.computer_science_msc_cs)
    elif call.data == 'criminology_syllabus':
        bot.send_document(call.message.chat.id, Data_File.criminology_ba_criminology)
    elif call.data == 'economics_syllabus':
        bot.send_document(call.message.chat.id, Data_File.economics_ba_economics)
        bot.send_document(call.message.chat.id, Data_File.economics_ma_economics)
    elif call.data == 'electronics_syllabus':
        bot.send_document(call.message.chat.id, Data_File.electronics_bsc_electronics)
        bot.send_document(call.message.chat.id, Data_File.electronics_msc_electronics)
    elif call.data == 'english_syllabus':
        bot.send_document(call.message.chat.id, Data_File.english_ba_english)
        bot.send_document(call.message.chat.id, Data_File.english_ba_double_main)
        bot.send_document(call.message.chat.id, Data_File.english_ma_english)
    elif call.data == 'forensic_science_syllabus':
        bot.send_document(call.message.chat.id, Data_File.FS_bvoc_forensic_science)
    elif call.data == 'bba_syllabus':
        bot.send_document(call.message.chat.id, Data_File.MS_bba)
    elif call.data == 'mathematics_syllabus':
        bot.send_document(call.message.chat.id, Data_File.Maths_bsc_mathematics)
        bot.send_document(call.message.chat.id, Data_File.Maths_msc_mathematics)
    elif call.data == 'media_syllabus':
        bot.send_document(call.message.chat.id, Data_File.Media_ba_multimedia)
        bot.send_document(call.message.chat.id, Data_File.Media_ba_visual_communication)
        bot.send_document(call.message.chat.id, Data_File.Media_ma_visual_communication)
    elif call.data == 'physics_syllabus':
        bot.send_document(call.message.chat.id, Data_File.Physics_bsc_physics)
        bot.send_document(call.message.chat.id, Data_File.Physics_msc_physics)
    elif call.data == 'psychology_syllabus':
        bot.send_document(call.message.chat.id,
Data_File.Psychology_integrated_msc_psychology)
    elif call.data == 'social_work_syllabus':
        bot.send_document(call.message.chat.id, Data_File.SW_msw)
    elif call.data == 'statistics_syllabus':
        bot.send_document(call.message.chat.id, Data_File.Stati_bsc_statistics)
        bot.send_document(call.message.chat.id, Data_File.Stati_msc_statistics)
    elif call.data == 'zoology_syllabus':
        bot.send_document(call.message.chat.id, Data_File.Zoology_bsc_zoology)
        bot.send_document(call.message.chat.id, Data_File.Zoology_msc_zoology)
##############################
##########################################################################
#######################################################
```

```
################################################## Computer-based Menu
Features##################################################
### Linux Distro  Reccomendator ####
@bot.message_handler(commands=['linuxdistro'])
def linux_distro_reccomendator_menu(message):
    keyboard = types.InlineKeyboardMarkup()

    # Defining inline buttons
    button1 = types.InlineKeyboardButton(text='Beginners',
callback_data='beginners_linuxdistro')
    button2 = types.InlineKeyboardButton(text='Gamers',
callback_data='gamers_linuxdistro')
    button3 = types.InlineKeyboardButton(text='Professional Use',
callback_data='professional_use_linuxdistro')
    button4 = types.InlineKeyboardButton(text='Low-spec System',
callback_data='low_spec_system_linuxdistro')
    button5 = types.InlineKeyboardButton(text='Hackers',
callback_data='hackers_linuxdistro')
    button6 = types.InlineKeyboardButton(text='Developers',
callback_data='developers_linuxdistro')
    keyboard.add(button1, button2, button3, button4, button5, button6)
    bot.send_message(message.chat.id, 'Please select an option:', reply_markup=keyboard)

@bot.callback_query_handler(func=lambda call : call.data.endswith('linuxdistro'))
def handle_callback_linuxdistro_buttons(call):
    if call.data == 'beginners_linuxdistro':
        bot.send_message(call.message.chat.id, 'Top linux Distro for beginners')
        bot.send_message(call.message.chat.id,
Data_File.linux_distro_recomendator_beginners)

    elif call.data == 'gamers_linuxdistro':
        bot.send_message(call.message.chat.id, 'Top linux Distro for gamers')
        bot.send_message(call.message.chat.id, Data_File.linux_distro_recomendator_gamers)

    elif call.data == 'professional_use_linuxdistro':
        bot.send_message(call.message.chat.id, 'Top linux Distro for Professional_use')
        bot.send_message(call.message.chat.id,
Data_File.linux_distro_recomendator_proffesional_use)

    elif call.data == 'low_spec_system_linuxdistro':
        bot.send_message(call.message.chat.id, 'Top linux Distro for low_spec_system')
        bot.send_message(call.message.chat.id,
Data_File.linux_distro_recomendator_low_spec_system)

    elif call.data == 'hackers_linuxdistro':
        bot.send_message(call.message.chat.id, 'To. linux Distro for hackers')
        bot.send_message(call.message.chat.id, Data_File.linux_distro_recomendator_hackers)

    elif call.data == 'developers_linuxdistro':
        bot.send_message(call.message.chat.id, 'Top linux Distro for Developers')
        bot.send_message(call.message.chat.id,
Data_File.linux_distro_recomendator_developers)
################################
### Linux commands pdf #######
```

```
@bot.message_handler(commands=['linuxcommandpdf'])
def send_linux_commands_pdf(message):
    bot.send_document(message.chat.id,Data_File.linux_commands_pdf)
##############################
##################################################################################
##########################################


################################################## FUN Menu Features
##################################################
### Joke ###
@bot.message_handler(commands=['joke'])
def joke(message):
    bot.reply_to(message,"So you want a joke")
    #send GIF
    bot.send_sticker(message.chat.id,"https://media.tenor.com/7R4_EnS5IPIAAAAM/
younge-sheldon-i-dont-have-time-for-jokes.gif")
    time.sleep(2) #for joke purpose
    bot.send_sticker(message.chat.id,"https://media.tenor.com/07qImC4D1ToAAAAC/i-was-
just-kidding-kidding.gif")
    bot.send_message(message.chat.id,"Here is your joke")
    #Takes a joke randomly from pyjokes libary
    jokes_text=pyjokes.get_joke("en","all")
    bot.send_message(message.chat.id,jokes_text) #send joke
################
#### Dice ######
@bot.message_handler(commands=['dice'])
def dice(message):
    bot.reply_to(message,"So you don't have a dice with you")
    bot.send_sticker(message.chat.id,"https://media.tenor.com/ND2XiIDIX3IAAAAC/trust-
me-i-got-you.gif")
    bot.send_message(message.chat.id,"In .....1....2..")
    send_sticker =
bot.send_sticker(message.chat.id,"https://media.tenor.com/i_L5KauoCcoAAAAi/dice.gif")
    time.sleep(3) # for building up tension while dice is rolling
    bot.delete_message(message.chat.id,send_sticker.message_id) # stopping GIF for
unwanted confusion
    number=random.randint(1,6) # random number from 1-6
    Dice_number=(f"The number is {number}")
    bot.send_message(message.chat.id,Dice_number)
#################
### Coporatebs ##
@bot.message_handler(commands=['coporatebs'])
def coporatebs(message):
    url = "https://sameer-kumar-corporate-bs-generator-v1.p.rapidapi.com/" #url of api

    #get response from api
    response = requests.get(url, headers= coporate_bs_api_key()) #api key and host
    #converting it to dictionary from json
    phrase = response.json()
    bot.send_message(message.chat.id,"So you wanna here some Coporate          \nWhich
makes no sense\nBut sounds complex          ")
    bot.send_message(message.chat.id,phrase['phrase']) # sending phrase
```

```
    bot.send_sticker(message.chat.id,"https://media.tenor.com/bkbcsCcDMJIAAAAC/try-to-
figure-it-out-emma.gif")
#######################
### Superhero ########
@bot.message_handler(commands=['super'])
def super(message):
    bot.send_message(message.chat.id,"So you wanna know the deatils of your favourite
Superhero/Villian")
    bot.send_message(message.chat.id,"Go to this website and send me the ID of your
Superhero/Villian")
    #asking user to send id
    bot.send_message(message.chat.id,"https://www.superheroapi.com/ids.html")
    message = bot.send_message(message.chat.id,"You can use 'Find in page' feature of your
browser")
    bot.send_message(message.chat.id,"For Random send a number between (1-731)")
    bot.register_next_step_handler(message,processing_sending_superhero_deatils) # taking
input

def processing_sending_superhero_deatils(message):
    id = message.text
    url = f'https://superheroapi.com/api/248489081095040/{id}'
    response = requests.get(url)
    deatils = response.json()
    #if id not found try again
    try:
        bot.send_message(message.chat.id,deatils['name'])
    except KeyError:
        bot.send_message(message.chat.id,'ID not found, Try again')
        super(message)
    #send photo of super hero
    image_url = str(deatils['image']['url'])
    bot.send_photo(message.chat.id,image_url)
    data = deatils
    #avoid repeated data
    del data['name']
    del data['response']
    del data['image']
    #organising nested dictionary data in a string for readability
    def get_nested_dictionary_string(data, indent=0):
        result = ""
        for key, value in data.items():
            if isinstance(value, dict):
                result += '\t' * indent + f'{key}:\n'
                result += get_nested_dictionary_string(value, indent + 1)
                result += '\n'
            else:
                result += '\t' * indent + f'{key}: {value}\n'
                result += '\n'
        return result
    #sending string
    bot.send_message(message.chat.id,get_nested_dictionary_string(data))
 ##########################
 #### Meme generator ######
@bot.message_handler(commands=['meme'])
```

```
def meme(message):
    bot.reply_to(message,"Here is a random meme for you")
    #meme api response
    response = requests.get('https://meme-api.com/gimme')
    meme = response.json()
    meme_url = meme['url']
    if meme_url[-1:-4:-1] =='fig': #gif written backwards
        # if it is a gif
        bot.send_sticker(message.chat.id,meme_url)
    else:
        # else photo
        bot.send_photo(message.chat.id,meme_url)
############################
##############################################################################
#############################################

@bot.message_handler(commands=['contact'])
def contact(message):
    bot.reply_to(message,"So you wanna contact the developer")
    bot.send_message(message.chat.id,"Here you go")
    bot.send_message(message.chat.id,"@ADHIVP") #telegram user id

bot.infinity_polling()
```

Data_File.py

```
list_features = '''
1. Online Course Recommendator - '/onlinecourse'
2. YouTube Course Recommendator - '/youtubecourse'
3. Cheat Sheets - '/cheatsheet'
4. GitHub Learning Resources - '/githublearn'
5. External Mark Needed for Specific Grade - '/gradecalculator'
6. Playback Speed Calculator - '/playbackspeedcalculator'
7. Password Generator - '/password'
8. Text to Audio Converter - '/textaudio'
9. QR Code Generator - '/qrcodegenerator'
10. Morse Code Generator - '/morsecode'
11. Weather Details of Your Current Location - '/weather'
12. Random Activity Generator - '/activity'
13. Syllabus Sender - '/syllabus'
14. Linux Distro Recommendator - '/linuxdistro'
15. Linux Command PDF - '/linuxcommandpdf'
16. Ask Me a Joke - '/joke'
17. Roll a Dice - '/dice'
18. Corporate BS - '/corporatebs'
19. Superhero Details - '/super'
20. Random Meme Generator - '/meme'
'''

##################Possible keywords for each command
#####################################
online_course_possible_keywords = ['online course', 'udemy', 'coursera', 'onlinecourse',
'online cors',
                                   'onlie course', 'ondline course', 'oneline course', 'udamy', 'udmey',
                                   'udemu', 'courseraa', 'corsera', 'coursesra', 'onlinecouse', 'onlinecoarse',
                                   'onlinecours', 'onlnie course', 'onilne course', 'onlne course', 'ondilne
course',
                                   'onlinne course', 'udemmy', 'udemyy', 'udemyy', 'courser', 'couser',
'corusera']

youtube_course_possible_keywords = ['youtube','yutube', 'youtub', 'uoutube', 'youtibe',
'yuotube',
                                    'youtuube', 'youtubee', 'youtbe', 'youtubr', 'yuotube',
                                    'youtub', 'youtubbe', 'youtupe', 'youtub3', 'youtibe',
                                    'youube', 'yooutube', 'youtub4', 'youtbue']

cheat_sheet_possible_keywords = ['cheat sheet', 'cheatsheet', 'cheet sheet', 'cheetshet',
'cheetseet', 'cheat seet',
                                 'cheat shet', 'cheet sheat', 'cheetsheet', 'cheetshett', 'cheetseett', 'cheet
seett',
                                 'cheet shett', 'cheet sheatt', 'cheatsheeet', 'cheatshett', 'cheat sheat',
'cheetsheatt',
                                 'cheatsheett', 'cheat sheatt']

github_learn_possible_keywords = ['github', 'github learn', 'github resources', 'git hub', 'git
hub learn', 'git hub resources',
```

```
                    'github learn', 'github resources', 'github learn', 'github resources', 'gith
ub', 'gith ub learn',
                    'gith ub resources', 'git-hub', 'git-hub learn', 'git-hub resources',
'githublearn', 'githubresources',
                    'git hublearn', 'git hubresources']

external_mark_calculator_possbile_keywords = ['grade', 'external mark', 'internal mark',
'externalmark', 'internalmark', 'grad',
                                'externl mark', 'internal mrk', 'externlmark', 'internlmark',
'external assessment',
                                'internal assessment', 'external marking', 'internal marking',
'garde', 'external marks',
                                'internal marks', 'externalmarks', 'internalmarks', 'grde', 'externel
mark', 'internel mark',
                                'externelmark', 'internlmark', 'externl assessment', 'external
marking', 'internal marking']

playback_speed_possible_keywords = ['speed', 'playbackspeed', 'play back speed', 'play
backspeed', 'playback speed', 'sped', 'playbakspeed',
                                 'play bak speed', 'play back sped', 'playbak sped', 'playbak sped', 'play
bak speed']

password_generator_possible_keywords = ['password', 'passwordgenerator', 'password
generator', 'pssword', 'passwordgeneratr',
                                  'password generatr', 'pasword', 'passwordgeneraotr', 'password
generaotr']

text_to_audio_possible_keywords = ['text to audio', 'audio', 'texttoaudio', 'txt to audio',
'audi', 'texttoaudi', 'text toaudi', 'text 2 audio',
                                'audio text', 'text2audio', 'audio', 'txt 2 audio', 'audi', 'text2audi', 'text
2audi', 'text-to-audio', 'audio',
                                'text-to-audio', 'txt-to-audio', 'audi', 'texttoaudio', 'text-toaudio', 'text to
audio', 'audio', 'texttoaudio',
                                'txt to audio', 'audi', 'texttoaudi', 'text toaudi', 'txt2audio', 'aud', 'text-
toaudio', 'text 2-audio', 'aud',
                                'text2-audio', 'audio text', 'txt2-audio', 'text 2audio', 'audio-text',
'text2audio']

qr_code_generator_possible_keywords = ['qr', 'qr code', 'qrcode', 'q-r', 'q-r code', 'qr-code']

morse_code_generator_possible_keywords = ['morse', 'morse code', 'morsecode', 'morze',
'morze code',
                                  'morzecode', 'mors', 'mors code', 'morscode']

weather_possible_keywords = ['weather', 'temperature', 'humidity', 'rain', 'location', 'wether',
'temprature', 'humidity', 'rane', 'loction',
                        'weathr', 'tempreture', 'humidity', 'rein', 'locatin', 'weathe', 'temperatur',
'humidty', 'locaton', 'wather',
                        'temparature', 'hmidity', 'raain', 'locaion', 'waether', 'temperatue', 'humidy',
'rainn', 'lcoation',
                        'wahter', 'temperate', 'hunidity', 'lcation', 'weatther', 'temperture',
'humididy', 'rean', 'locatioon', 'wethr',
                        'tmeperature', 'hmidty', 'ranin']
```

```
random_activity_possible_keywords = ['activity', 'task', 'activty', 'taks', 'actvity']

st_thomas_syllabus_possible_keywords = ['syllabus', 'curriculum', 'syllbus', 'curiculum',
'silabus', 'curruculum', 'program', 'course outline',
                                        'study plan', 'subject matter', 'lesson plan', 'outline' , 'syllab',
                                        'curriculm', 'syllabu', 'curricum', 'sylabus', 'curruculm', 'sylabus',
'curruculum', 'syllabs', 'curriculam', 'silabas',
                                        'curruculam', 'silabus', 'curriculum', 'syllbass', 'curruculm', 'sylabas',
'curruculum']
linux_distro_reccomendator_possible_keywords = ['distro','linux distro', 'linuxdistro', 'linx
distro', 'linxdistro', 'linu distro', 'linudistro', 'lnux distro',
                                                'lnuxdistro', 'linux disto', 'linuxdistr', 'linux distor',
'linuxdistro']

linux_commands_pdf_sender_possible_keywords = ['command','linuxcommand', 'linux
command', 'linux commands', 'linuxcommands', 'linxcommand', 'linx command', 'linx
commands',
                                                'linxcommands', 'linu command', 'linu commands', 'lnux
command', 'lnux commands', 'linux comand', 'linux comands',
                                                'linuxcomand', 'linuxcomands']

random_joke_possible_keywords = ['joke', 'comedy', 'jok', 'comdy', 'jooke', 'comedi', 'jok',
'comdy', 'joek', 'comdey', 'joake', 'comdei']

Dice_possible_keywords = ['dice', 'die', 'gambling cube', 'game of chance', 'random number
generator', 'dice roll', 'dice game',
                          'rolling the dice', 'craps']

Coporate_bs_possible_keywords = ['bs', 'bullshit', 'busswords', 'bussword', 'buzzwords',
'coporate bs', 'coporatebs', 'bull shit',
                                 'buswords', 'busword', 'buzwords', 'cporate bs', 'cporatebs', 'bulshit']

super_hero_possible_keywords = ['hero', 'super', 'superhero', 'super hero', 'heroe', 'supr',
'suprhero', 'supr hero',
                                'herro', 'sper', 'sperhero', 'sper hero']

Random_meme_possible_keywords = ['meme']
##############################################################################
#############
Online_course_Python_Udemy = ("https://www.udemy.com/course/100-days-of-code",
    "https://www.udemy.com/course/complete-python-bootcamp/",
    "https://www.udemy.com/course/complete-python-developer-zero-to-mastery/",
    "https://www.udemy.com/course/the-modern-python3-bootcamp/",
    "https://www.udemy.com/course/python3-fundamentals/",
    "https://www.udemy.com/course/total-python/",
    "https://www.udemy.com/course/python-core-and-advanced/",
    "https://www.udemy.com/course/python-for-absolute-beginners-u/",
    "https://www.udemy.com/course/python-the-complete-python-developer-course/",
    "https://www.udemy.com/course/complete-python-developer-zero-to-master")
Online_course_Python_Coursera = ("https://www.coursera.org/specializations/python-3-
programming",
    "https://www.coursera.org/learn/python-crash-course",
    "https://www.coursera.org/learn/python-for-applied-data-science-ai",
    "https://www.coursera.org/learn/get-started-with-python",
```

```
    "https://www.coursera.org/learn/python-programming-intro",
    "https://www.coursera.org/specializations/python-3-programming",
    "https://www.coursera.org/learn/python-crash-course",
    "https://www.coursera.org/learn/python-for-applied-data-science-ai",
    "https://www.coursera.org/learn/get-started-with-python",
    "https://www.coursera.org/learn/python-programming-intro")

Online_course_Java_Udemy = ("https://www.udemy.com/course/java-the-complete-java-
developer-course/",
    "https://www.udemy.com/course/java-se-programming/",
    "https://www.udemy.com/course/java-programming-tutorial-for-beginners/",
    "https://www.udemy.com/course/the-complete-java-development-bootcamp/",
    "https://www.udemy.com/course/full-stack-java-developer-java/",
    "https://www.udemy.com/course/java-development-for-beginners-learnit/",
    "https://www.udemy.com/course/java-programming-complete-beginner-to-advanced/",
    "https://www.udemy.com/course/master-practical-java-development/",
    "https://www.udemy.com/course/practical-java-course/",
    "https://www.udemy.com/course/java-11-complete-beginners/")
Online_course_Java_Coursera = ("https://www.coursera.org/specializations/javascript-
beginner",
    "https://www.coursera.org/learn/html-css-javascript-for-web-developers",
    "https://www.coursera.org/learn/programming-with-javascript",
    "https://www.coursera.org/learn/javascript-basics",
    "https://www.coursera.org/learn/learn-javascript",
    "https://www.coursera.org/specializations/javascript-beginner",
    "https://www.coursera.org/learn/html-css-javascript-for-web-developers",
    "https://www.coursera.org/learn/programming-with-javascript",
    "https://www.coursera.org/learn/javascript-basics",
    "https://www.coursera.org/learn/learn-javascript")

Online_course_Javascript_Udemy = ("https://www.udemy.com/course/the-complete-
javascript-course/",
    "https://www.udemy.com/course/the-complete-web-development-bootcamp/",
    "https://www.udemy.com/course/javascript-the-complete-guide-2020-beginner-
advanced/",
    "https://www.udemy.com/course/advanced-javascript-concepts/",
    "https://www.udemy.com/course/javascript-basics-for-beginners/",
    "https://www.udemy.com/course/javascript-beginners-complete-tutorial/",
    "https://www.udemy.com/course/modern-javascript-from-the-beginning/",
    "https://www.udemy.com/course/javascript-tutorial-for-beginners-w/",
    "https://www.udemy.com/course/js-algorithms-and-data-structures-masterclass/",
    "https://www.udemy.com/course/the-complete-web-developer-zero-to-mastery/"
)
Online_course_Javascript_Coursera = ("https://www.coursera.org/specializations/javascript-
beginner",
    "https://www.coursera.org/learn/html-css-javascript-for-web-developers",
    "https://www.coursera.org/learn/programming-with-javascript",
    "https://www.coursera.org/learn/javascript-basics",
    "https://www.coursera.org/learn/learn-javascript",
    "https://www.coursera.org/specializations/javascript-beginner",
    "https://www.coursera.org/learn/html-css-javascript-for-web-developers",
    "https://www.coursera.org/learn/programming-with-javascript",
    "https://www.coursera.org/learn/javascript-basics",
    "https://www.coursera.org/learn/learn-javascript")
```

```
Online_course_C_Udemy = ( "https://www.udemy.com/course/the-complete-web-
developer-zero-to-mastery/",
    "https://www.udemy.com/course/beginning-c-plus-plus-programming/",
    "https://www.udemy.com/course/cpp-deep-dive/",
    "https://www.udemy.com/course/the-modern-cpp-20-masterclass/",
    "https://www.udemy.com/course/beg-modern-cpp/",
    "https://www.udemy.com/course/learn-c-and-c-beginner-to-advance/",
    "https://www.udemy.com/course/c-coding-learn-c-programming-with-examples-in-one-
day/",
    "https://www.udemy.com/course/the-complete-cpp-developer-course/",
    "https://www.udemy.com/course/quick-start-to-modern-c-for-programmers/",
    "https://www.udemy.com/course/cplusplus-programming-step-by-step/")
Online_course_C_Coursera = ("https://www.coursera.org/specializations/coding-for-
everyone",
    "https://www.coursera.org/specializations/c-programming",
    "https://www.coursera.org/specializations/hands-on-cpp",
    "https://www.coursera.org/learn/programming-languages-part-c",
    "https://www.coursera.org/specializations/cplusplusunrealgamedevelopment",
    "https://www.coursera.org/specializations/coding-for-everyone",
    "https://www.coursera.org/specializations/c-programming",
    "https://www.coursera.org/specializations/hands-on-cpp",
    "https://www.coursera.org/learn/programming-languages-part-c",
    "https://www.coursera.org/specializations/cplusplusunrealgamedevelopment")


Youtube_video_python = ("https://www.youtube.com/watch?
v=_uQrJ0TkZlc&pp=ygUHUHl0aG9uIA%3D%3D",
    "https://www.youtube.com/watch?v=7wnove7K-
ZQ&list=PLu0W_9lII9agwh1XjRt242xIpHhPT2llg",
    "https://www.youtube.com/watch?v=VchuKL44s6E&pp=ygUHUHl0aG9uIA%3D%3D",
    "https://www.youtube.com/watch?v=rfscVS0vtbw&pp=ygUOUHl0aG9uICBjb3Vyc2U
%3D",
    "https://www.youtube.com/watch?
v=6i3EGqOBRiU&list=PLdo5W4Nhv31bZSiqiOL5ta39vSnBxpOPT",
    "https://www.youtube.com/watch?
v=XKHEtdqhLK8&pp=ygUOUHl0aG9uICBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=mRMmlo_Uqcs&list=PLIhvC56v63ILPDA2DQBv0IKzqsWTZxCkp",
    "https://www.youtube.com/watch?v=nLRL_NcnK-
4&pp=ygUOUHl0aG9uICBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?v=sxTmJE4k0ho&pp=ygUOUHl0aG9uICBjb3Vyc2U
%3D",
    "https://www.youtube.com/watch?
v=fqF9M92jzUo&t=4s&pp=ygUOUHl0aG9uICBjb3Vyc2U%3D")

Youtube_video_Java = ("https://www.youtube.com/watch?
v=yRpLlJmRo2w&list=PLfqMhTWNBTe3LtFWcvwpqTkUSlB32kJop",
    "https://www.youtube.com/watch?v=eIrMbAQSU34&pp=ygUMSmF2YSAgY291cnNl",
    "https://www.youtube.com/watch?v=BGTx91t8q50&pp=ygUMSmF2YSAgY291cnNl",
    "https://www.youtube.com/watch?
v=VHbSopMyc4M&list=PLBlnK6fEyqRjKA_NuK9mHmlk0dZzuP1P5",
    "https://www.youtube.com/watch?v=wdkP056q0Nc&pp=ygUMSmF2YSAgY291cnNl",
```

```
    "https://www.youtube.com/watch?
v=grEKMHGYyns&pp=ygUMSmF2YSAgY291cnNl",
    "https://www.youtube.com/watch?
v=RJ733wzbNoA&list=PLxgZQoSe9cg00xyG5gzb5BMkOClkch7Gr",
    "https://www.youtube.com/watch?v=j9VNCI9Xo80&pp=ygUMSmF2YSAgY291cnNl",
    "https://www.youtube.com/watch?v=drQK8ciCAjY&pp=ygUMSmF2YSAgY291cnNl",
    "https://www.youtube.com/watch?v=UmnCZ7-
9yDY&t=433s&pp=ygUMSmF2YSAgY291cnNl")

Youtube_video_Javascript =(
    "https://www.youtube.com/watch?
v=ER9SspLe4Hg&list=PLu0W_9lII9ahR1blWXxgSlL4y9iQBnLpR",
    "https://www.youtube.com/watch?
v=W6NZfCO5SIk&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=PkZNo7MFNFg&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=SBmSRK3feww&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?v=jS4aFq5-
91M&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=Qqx_wzMmFeA&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=lI1ae4REbFM&t=76s&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=acUEBly28UU&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=IC5vBKc21X8&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D",
    "https://www.youtube.com/watch?
v=8dWL3wF_OMw&pp=ygURamF2YXNjcmlwdCBjb3Vyc2U%3D"
)

Youtube_video_C = ("https://www.youtube.com/watch?
v=z9bZufPHFLU&list=PLfqMhTWNBTe0b2nM6JHVCnAkhQRGiZMSJ",
    "https://www.youtube.com/watch?
v=vLnPwxZdW4Y&pp=ygUPYysrIGZ1bGwgY291cnNl",
    "https://www.youtube.com/watch?
v=8jLOx1hD3_o&pp=ygUPYysrIGZ1bGwgY291cnNl",
    "https://www.youtube.com/watch?
v=ZzaPdXTrSb8&pp=ygUPYysrIGZ1bGwgY291cnNl",
    "https://www.youtube.com/watch?v=-
TkoO8Z07hI&pp=ygUPYysrIGZ1bGwgY291cnNl",
    "https://www.youtube.com/watch?
v=oOmbSpOzvYg&list=PLdo5W4Nhv31YU5Wx1dopka58teWP9aCee",
    "https://www.youtube.com/watch?
v=WQoB2z67hvY&list=PLDzeHZWIZsTryvtXdMr6rPh4IDexB5NIA",
    "https://www.youtube.com/watch?
v=iOfUlcUy0as&pp=ygUPYysrIGZ1bGwgY291cnNl",
    "https://www.youtube.com/watch?
v=FpfHmAkRVK4&pp=ygUPYysrIGZ1bGwgY291cnNl",
    "https://www.youtube.com/watch?v=bL-
o2xBENY0&list=PLxgZQoSe9cg0df_GxVjz3DD_Gck5tMXAd")

Cheat_sheet_Python = "https://cheatography.com/davechild/cheat-sheets/python/pdf/"
```

```
Cheat_sheet_Java =
"https://cheatography.com/sschaub/cheat-sheets/java-fundamentals/pdf/"
Cheat_sheet_Javascript = "https://cheatography.com/pyro19d/cheat-sheets/javascript/pdf/"
Cheat_sheet_Cpp = "https://cheatography.com/pmg/cheat-sheets/c/pdf/"
Cheat_sheet_Git = "https://cheatography.com/samcollett/cheat-sheets/git/pdf/"

morse_dicitonary = {
    'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.', 'G': '--.', 'H': '....', 'I': '..', 'J': '.---',
    'K': '-.-', 'L': '.-..', 'M': '--', 'N': '-.', 'O': '---', 'P': '.--.', 'Q': '--.-', 'R': '.-.', 'S': '...', 'T': '-',
    'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-', 'Y': '-.--', 'Z': '--..',
    '0': '-----', '1': '.----', '2': '..---', '3': '...--', '4': '....-', '5': '.....', '6': '-....', '7': '--...', '8': '---..', '9':
'----.',
    '.': '.-.-.-', ',': '--..--', '?': '..--..', '"": '.----.', '!': '-.-.--', '/': '-..-.', '(': '-.--.', ')': '-.--.-',
    '&': '.-...', ':': '---...', ';': '-.-.-.', '=': '-...-', '+': '.-.-.', '-': '-....-', '_': '..--.-', "": '.-..-.',
    '$': '...-..-', '@': '.--.-.', ' ': '/'
}

## Syllabus Finder###
Data_science_bvoc_data_science =
"https://stthomas.ac.in/wp-content/uploads/2021/12/B.Voc-Data-Science.pdf"
botany_bsc_botany = "https://stthomas.ac.in/wp-content/uploads/2021/12/BSc-Botany.pdf"
botany_msc_botany = "https://stthomas.ac.in/wp-content/uploads/2021/12/M-Sc-
botany.pdf"
chemistry_bsc_chem = "https://stthomas.ac.in/wp-content/uploads/2021/12/BSc-Chem.pdf"
chemistry_msc_chem = "https://stthomas.ac.in/wp-content/uploads/2021/12/MSc-
Chemistry.pdf"
commerce_mcom = "https://stthomas.ac.in/wp-content/uploads/2021/12/MCom.pdf"
commerce_bcom_finance = "https://stthomas.ac.in/wp-content/uploads/2021/12/BCom-
Finance.pdf"
commerce_bcom_banking = "https://stthomas.ac.in/wp-content/uploads/2021/12/BCom-
Banking.pdf"
commerce_sf_bcom ="https://stthomas.ac.in/wp-content/uploads/2020/08/B-Com1.pdf"
computer_application_bca = "https://stthomas.ac.in/wp-content/uploads/2021/12/BCA.pdf"
computer_application_msc_cs = "https://stthomas.ac.in/wp-content/uploads/2021/12/MSc-
Computer-Science.pdf"
computer_science_bsc_cs = "https://stthomas.ac.in/wp-content/uploads/2022/01/Computer-
Science-Final.pdf"
computer_science_msc_cs = "https://stthomas.ac.in/wp-content/uploads/2021/12/MSc-
Computer-Science.pdf"
criminology_ba_criminology =
"https://stthomas.ac.in/wp-content/uploads/2021/12/Criminology.pdf"
economics_ba_economics = "https://stthomas.ac.in/wp-content/uploads/2021/12/BA-
Economics.pdf"
economics_ma_economics = "https://stthomas.ac.in/wp-content/uploads/2021/12/MA-
Economics-1.pdf"
electronics_bsc_electronics = "https://stthomas.ac.in/wp-content/uploads/2021/12/BSc-
Electronics.pdf"
electronics_msc_electronics= "https://stthomas.ac.in/wp-content/uploads/2021/12/MSC-
Electronics.pdf"
english_ba_english = "https://stthomas.ac.in/wp-content/uploads/2021/12/BA-English-.pdf"
english_ba_double_main = "https://stthomas.ac.in/wp-content/uploads/2021/12/BA-Double-
Main.pdf"
english_ma_english = "https://stthomas.ac.in/wp-content/uploads/2021/12/MA-English.pdf"
```

FS_bvoc_forensic_science = "https://stthomas.ac.in/wp-content/uploads/2021/12/Bvoc-Forensic-Science.pdf"
MS_bba = "https://stthomas.ac.in/wp-content/uploads/2021/12/BBA.pdf"
Maths_bsc_mathematics = "https://stthomas.ac.in/wp-content/uploads/2021/12/BSc-Mathematics.pdf"
Maths_msc_mathematics = "https://stthomas.ac.in/wp-content/uploads/2021/12/MSc-Mathematics-.pdf"
Media_ba_visual_communication = "https://stthomas.ac.in/wp-content/uploads/2021/12/BA-Visual-Communication.pdf"
Media_ba_multimedia = "https://stthomas.ac.in/wp-content/uploads/2021/12/BA-Multimedia.pdf"
Media_ma_visual_communication = "https://stthomas.ac.in/wp-content/uploads/2021/12/MA-Visual-Communication.pdf"
Physics_bsc_physics = "https://stthomas.ac.in/wp-content/uploads/2021/12/BSc-Physics.pdf"
Physics_msc_physics = "https://stthomas.ac.in/wp-content/uploads/2021/12/M.Sc_.-Physics.pdf"
Psychology_integrated_msc_psychology = "https://stthomas.ac.in/wp-content/uploads/2022/01/Integrated-MSc-Psychology.pdf"
SW_msw = "https://stthomas.ac.in/wp-content/uploads/2021/12/MSW.pdf"
Stati_bsc_statistics = "https://stthomas.ac.in/wp-content/uploads/2021/12/BSc-Statistics-Updated.pdf"
Stati_msc_statistics = "https://stthomas.ac.in/wp-content/uploads/2021/12/MSc-Statistics.pdf"
Zoology_bsc_zoology = "https://stthomas.ac.in/wp-content/uploads/2021/12/BSc-Zoology.pdf"
Zoology_msc_zoology = "https://stthomas.ac.in/wp-content/uploads/2021/12/Msc-Zoology.pdf"

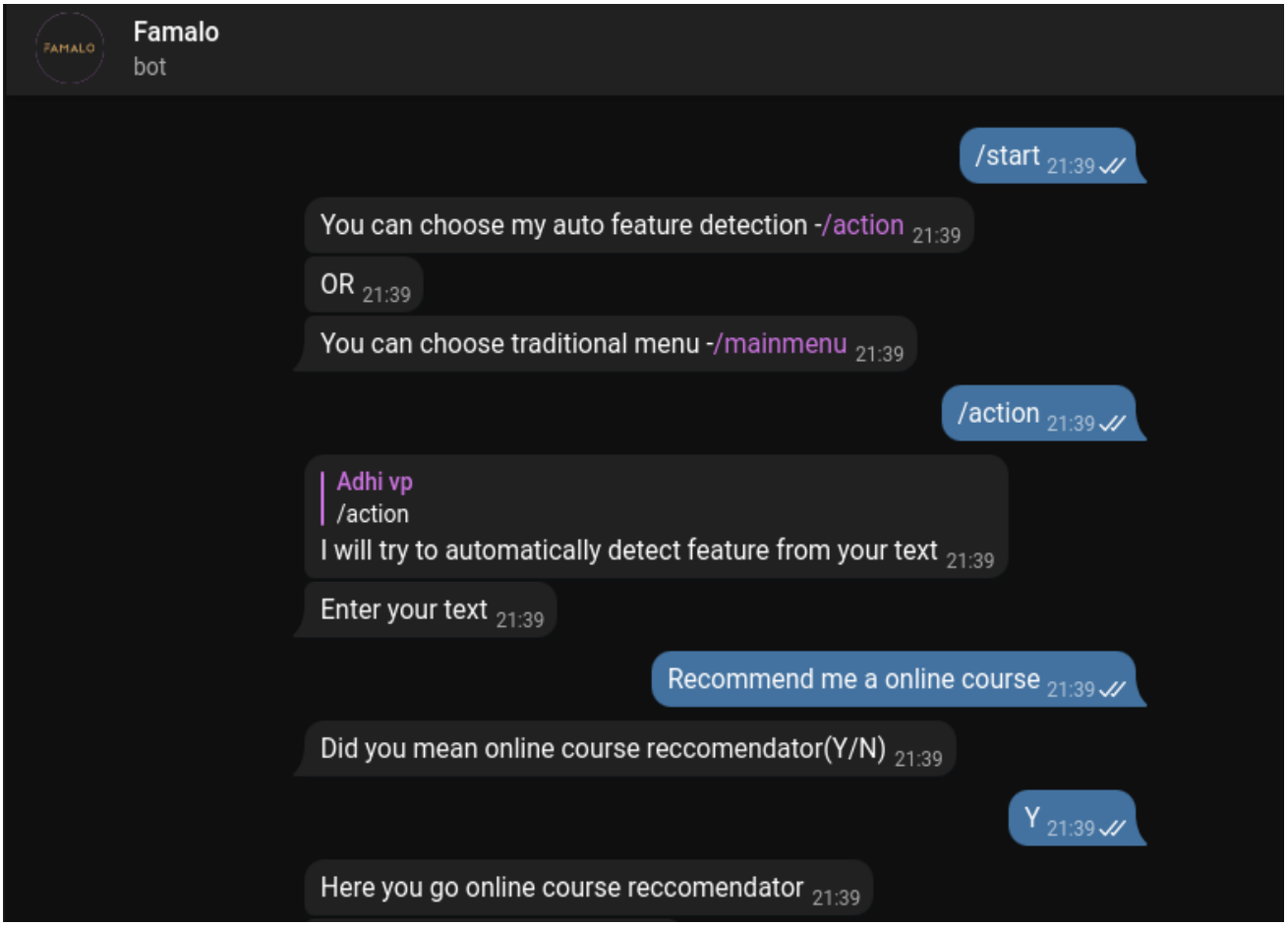
#####################

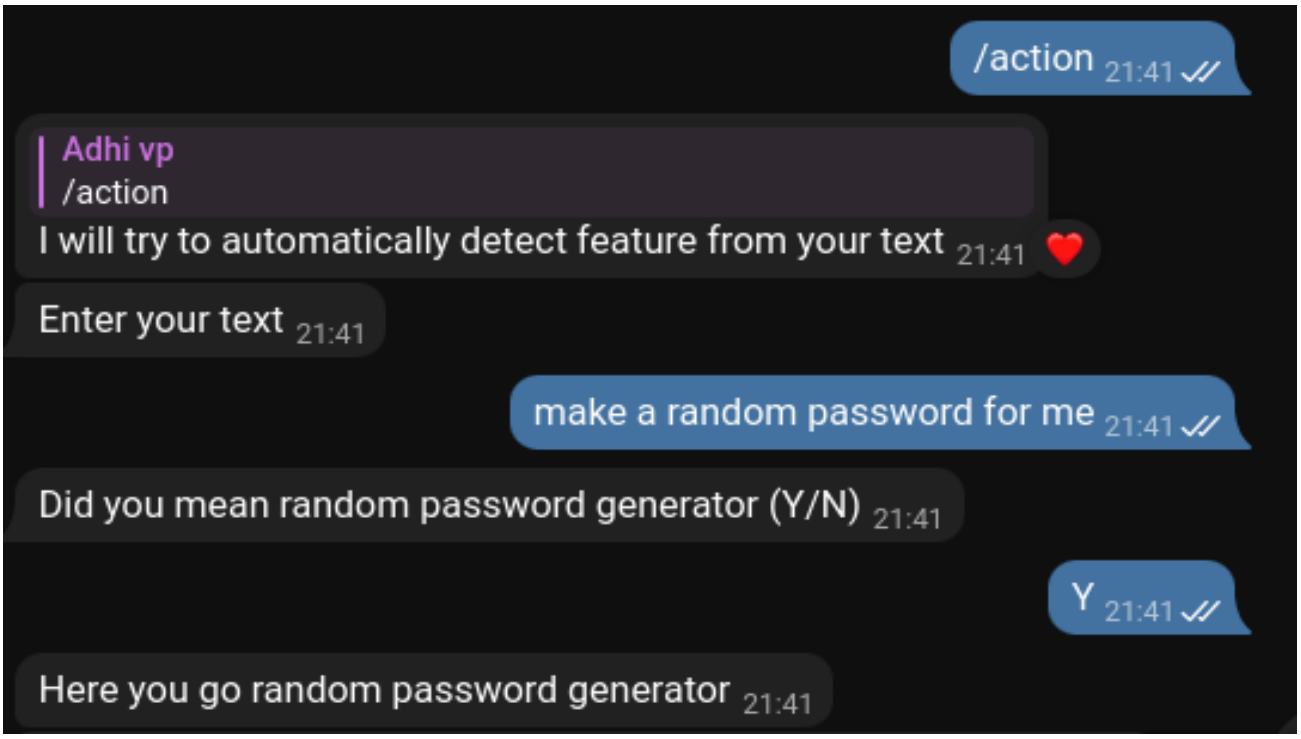
###Linux distro recomendator data ###
linux_distro_recomendator_beginners = "1. Ubuntu\n2. Linux Mint\n3. Zorin OS\n4. Elementary OS\n5. Linux Lite\n6. Manjaro Linux\n7. Pop!_OS\n8. Peppermint OS\n9. Deepin"
linux_distro_recomendator_gamers = "1. Fedora\n2. Garuda Linux\n3. Pop!_OS\n4. Ubuntu\n5. Drauger OS\n6. Regata OS\n7. SteamOS (Holo ISO)\n8. Manjaro\n9. Linux Mint\n10. Lakka Linux"
linux_distro_recomendator_proffesional_use = "1. Debian Linux\n 2. Manjaro Linux\n3. AlmaLinux/Rocky Linux\n4. Ubuntu\n5. OpenSUSE\n 6. Linux Mint"
linux_distro_recomendator_low_spec_system = "'1. Tiny Core
2. Puppy Linux
3. SparkyLinux
4. antiX Linux
5. Bodhi Linux
6. CrunchBang++
7. LXLE
8. Linux Lite
9. Lubuntu
10. Peppermint
11. Linux Mint Xfce
12. Xubuntu
13. Zorin OS Lite
14. Ubuntu MATE

15. Slax
16. Q4OS'''
linux_distro_recomendator_hackers = "1. Kali Linux\n2. BackBox\n3. ParrotOS\n4. BlackArch\n5. Samurai Web Testing Framework\n6. Pentoo Linux\n7. CAINE\n8. Network Security Toolkit\n9. Fedora Security Spin\n10. ArchStrike"
linux_distro_recomendator_developers = "1. Manjaro\n2. Ubuntu\n3. Pop!_OS\n4. Debian\n5. openSUSE\n6. Arch Linux\n7. Fedora Workstation\n8. Kali Linux\n9. Raspberry Pi OS\n10. Solus OS"
##
linux commands pdf
linux_commands_pdf = "https://cheatography.com/davechild/cheat-sheets/linux-command-line/pdf/"

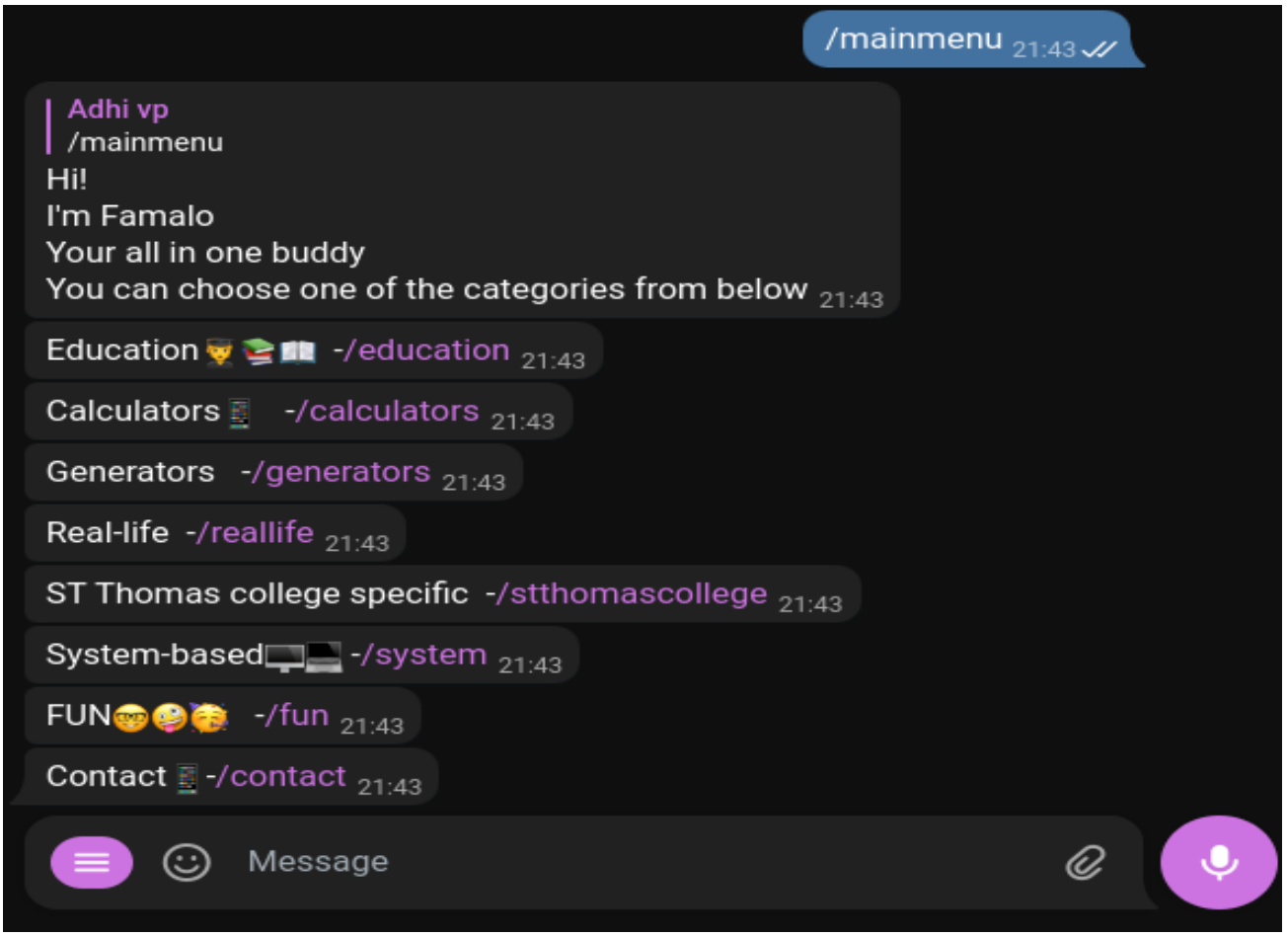
9.SCREEN SHOTS AND WORKING FLOW

Auto Feature Detection

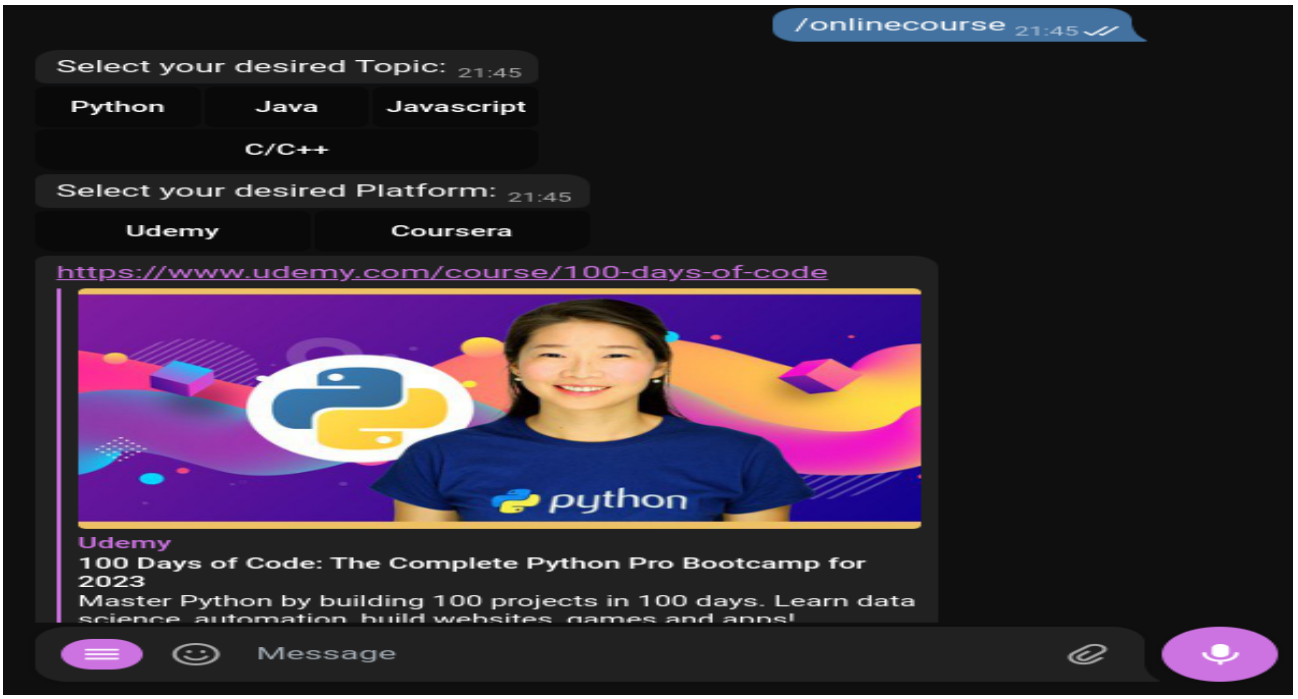




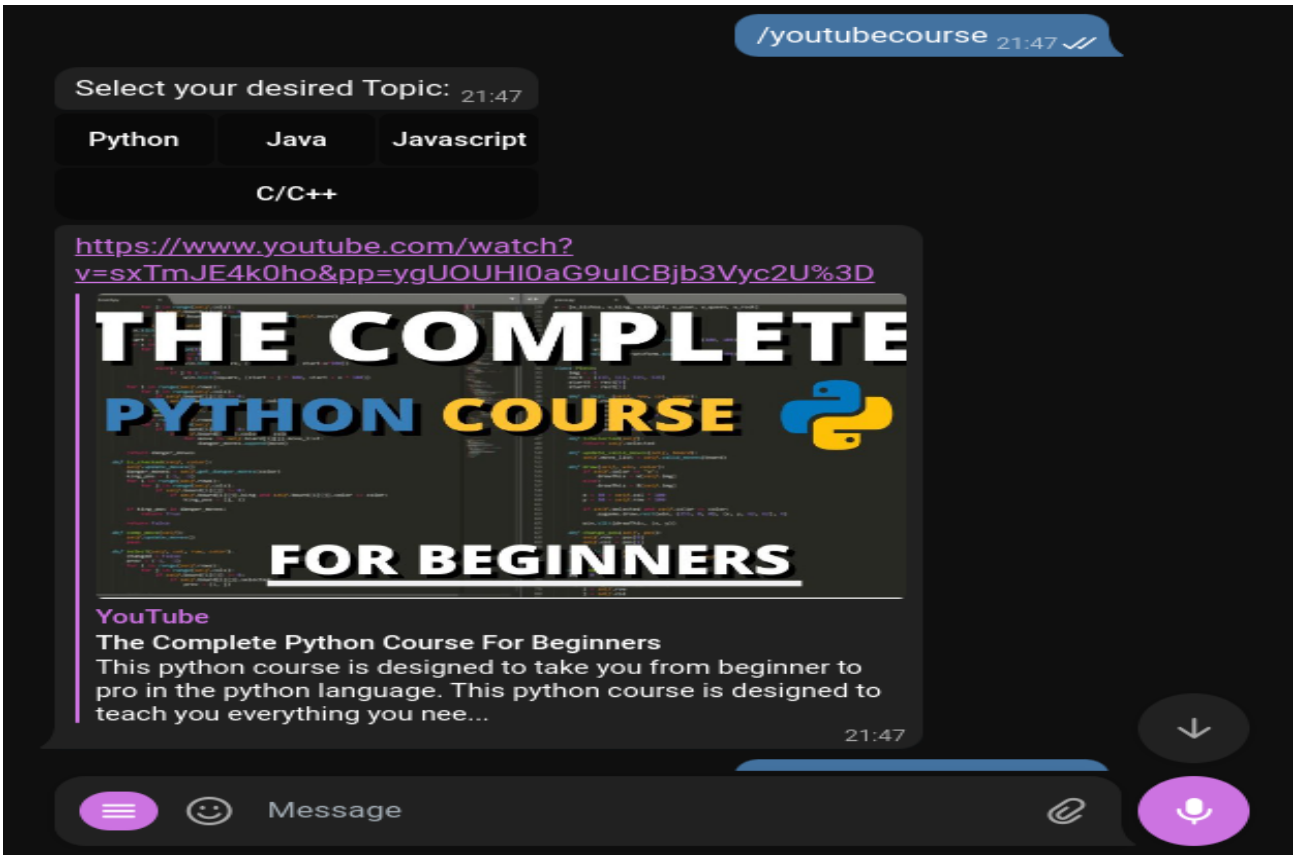
MAIN MENU



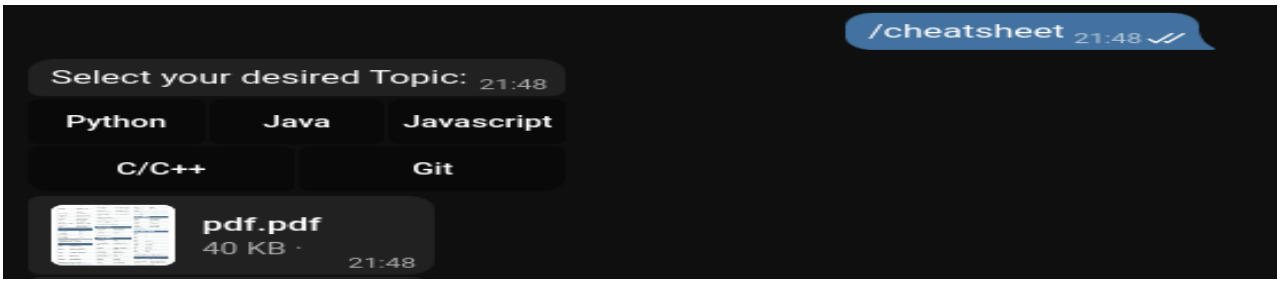
ONLINE COURSE RECOMMENDATOR



YOUTUBE COURSE RECOMMENDATOR



CHEATSHEETS



GITHUB LEARNING RESOURCES

EXTERNAL MARK CALCULATOR

Enter your internal mark (out of 20) 22:11

19 22:11

OK 22:11

Grade	External marks Needed
O	Above 76.0
A+	Above 66.0
A	Above 56.0
B+	Above 46.0
B	Above 36.0
C	Above 26.0
P	Above 16.0
F	Below 16.0

22:11

PLAYBACKSPEED CALCULATOR

/playbackspeedcalculator 22:12

So you wanna know is it worth to watch your lecture in fast mode 22:12

Type the length of your video in this format (00:00:00) limit (23:59:59) 22:12

12:00:00 22:13

OK 22:13

Type Your playbackspeed in this format(1.50) 22:13

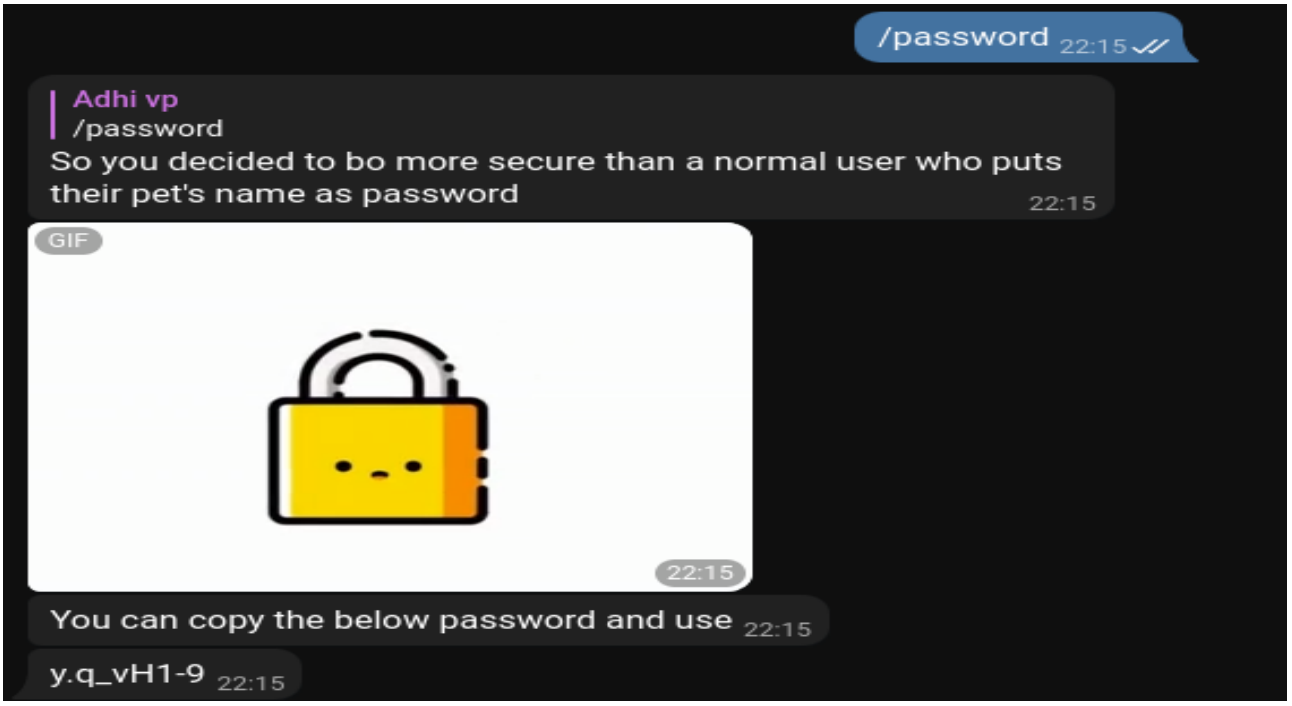
1.75 22:13

OK 22:13

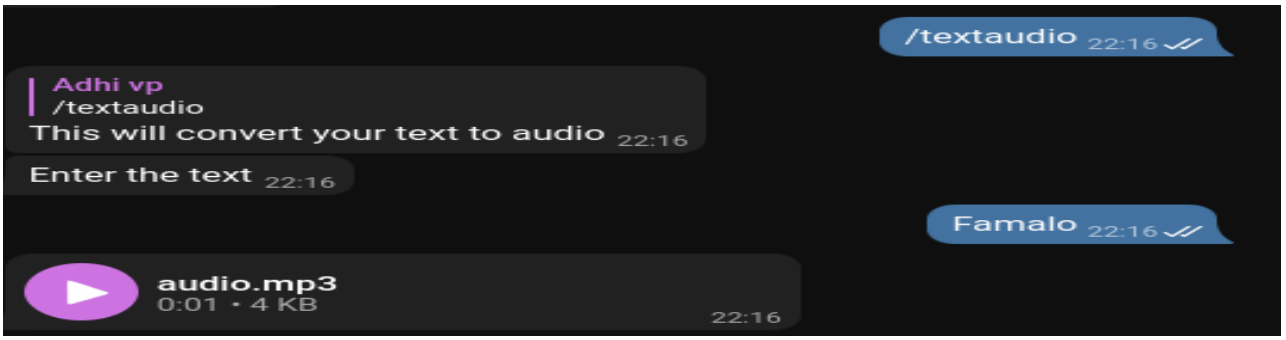
Calculated Time = 6:51:26 22:13

Time you can save at 1.75 = 5:08:34 22:13

PASSWORD GENERATOR

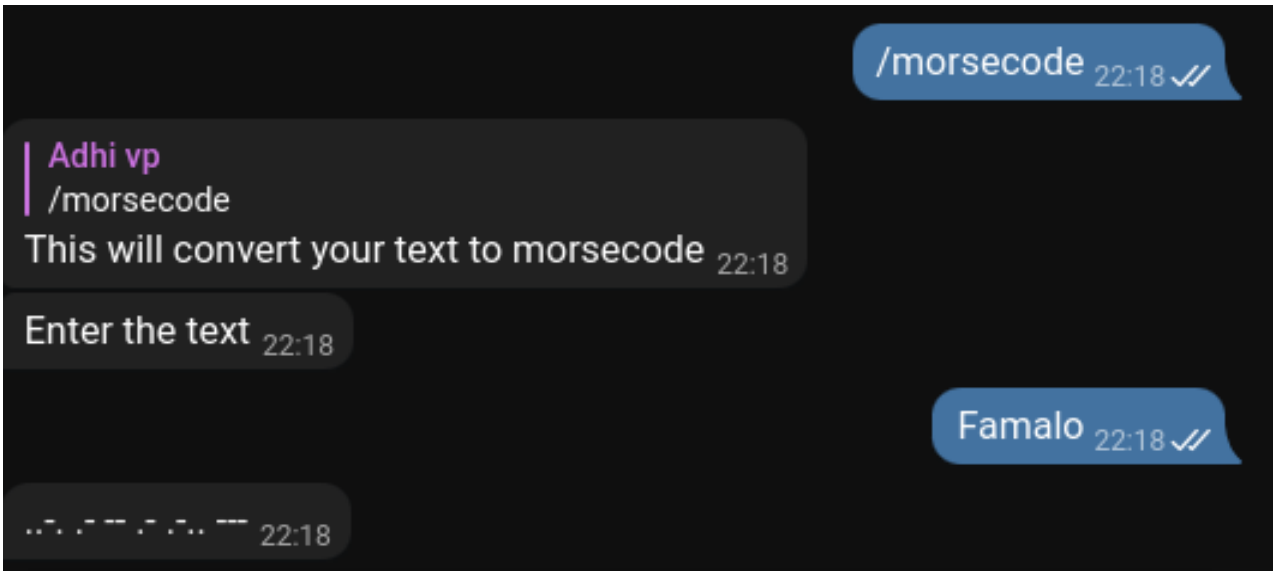


TEXT TO AUDIO CONVERTER

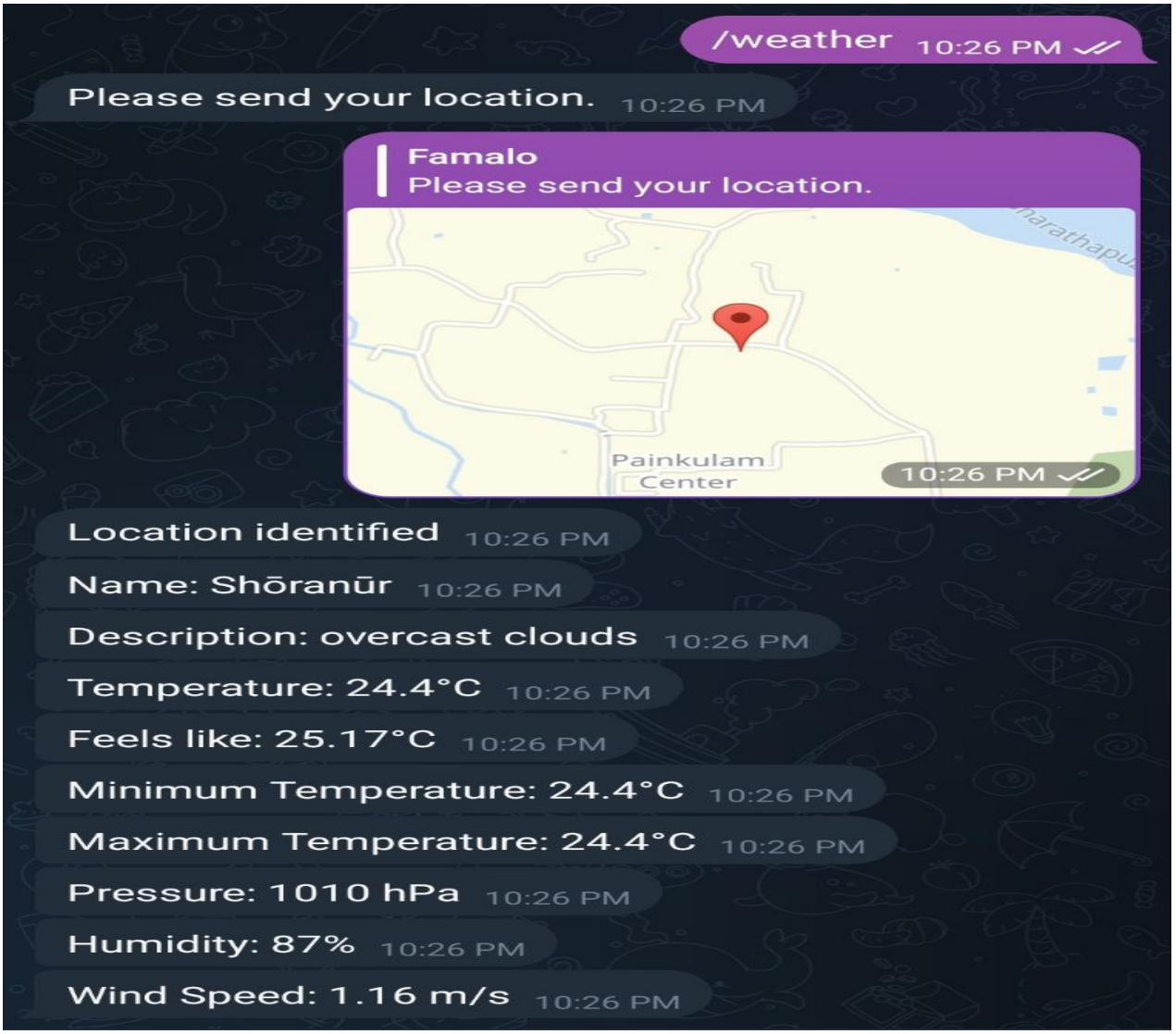


QR CODE GENERATOR

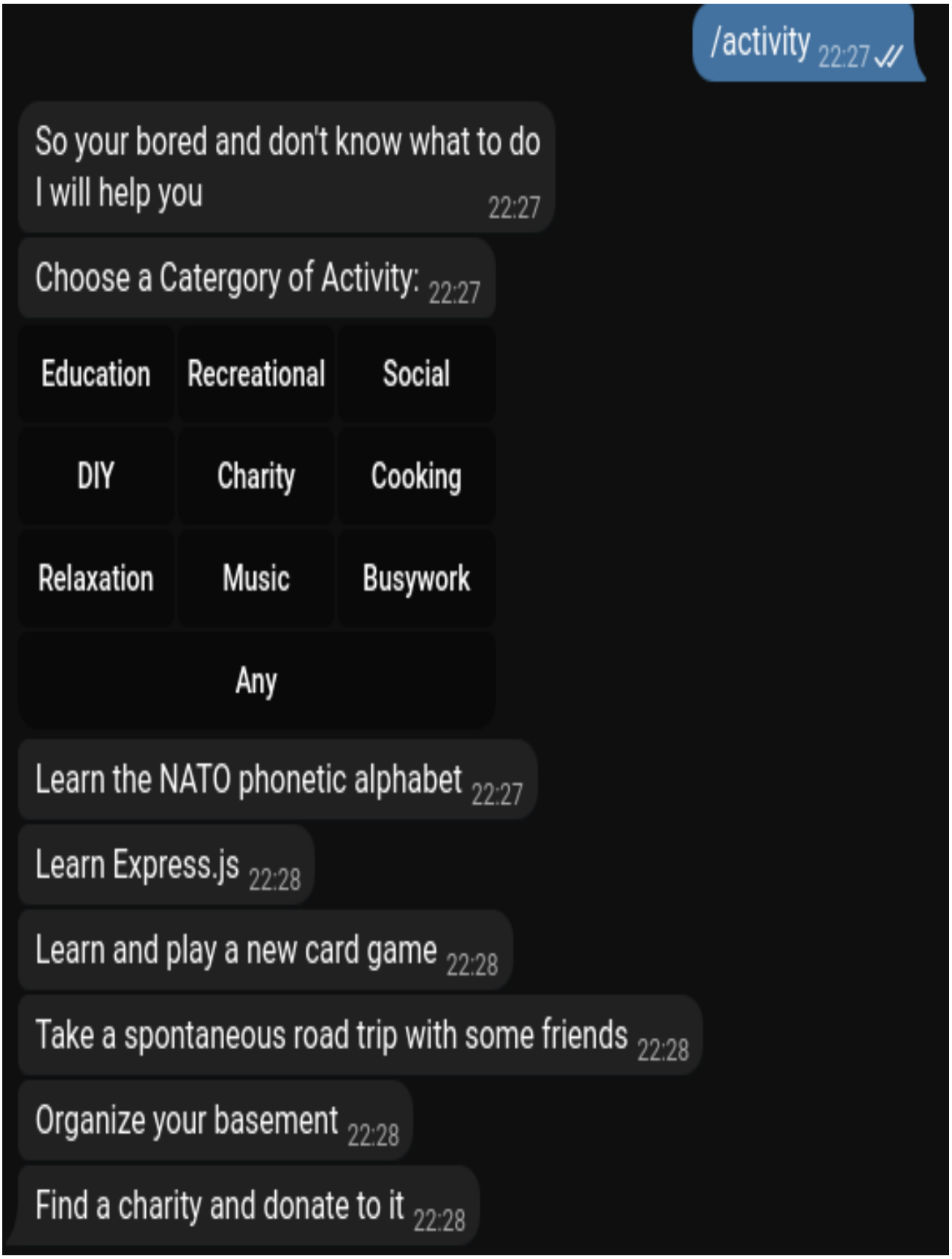
MORSE CODE GENERATOR



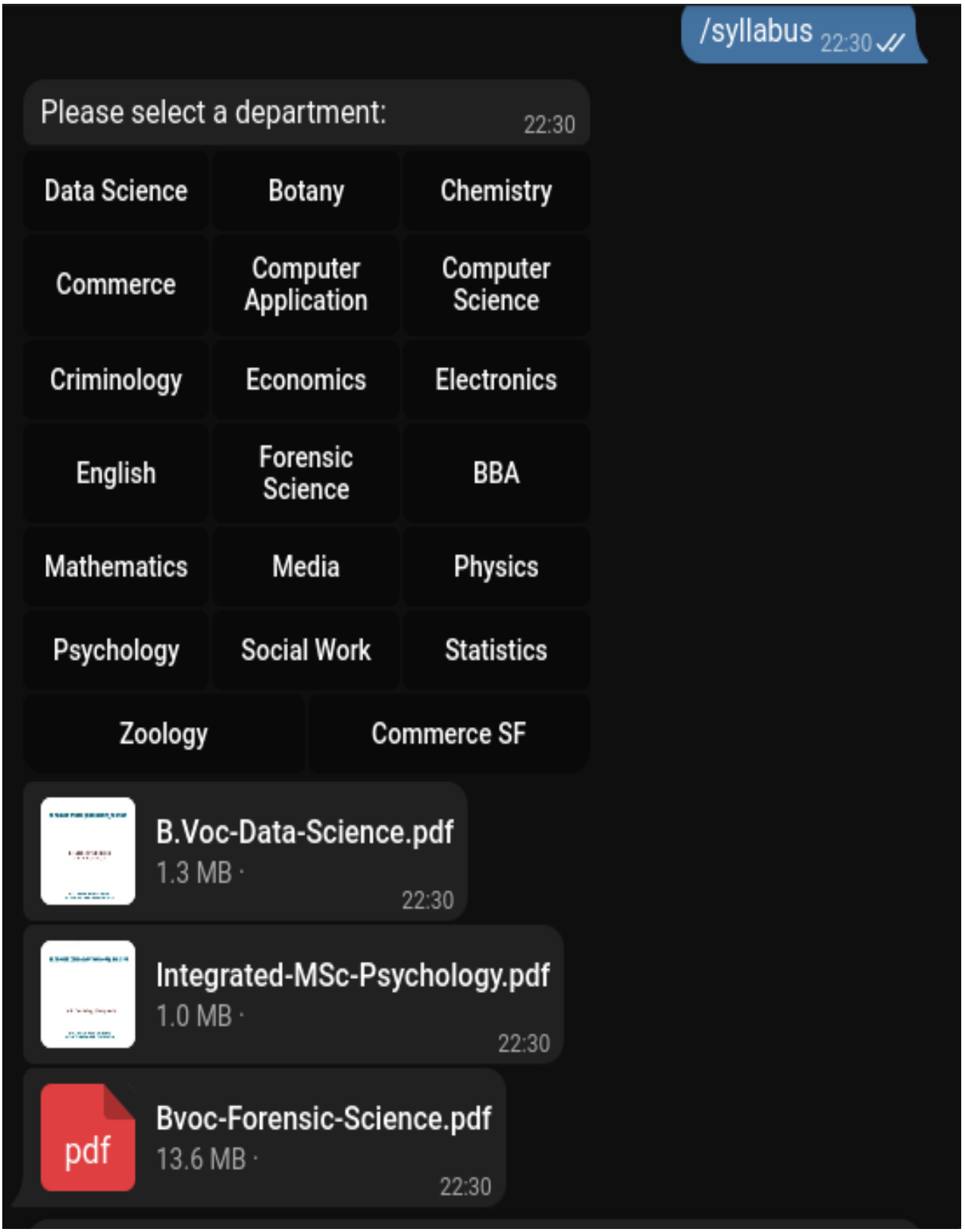
WEATHER OF YOUR CURRENT LOCATION



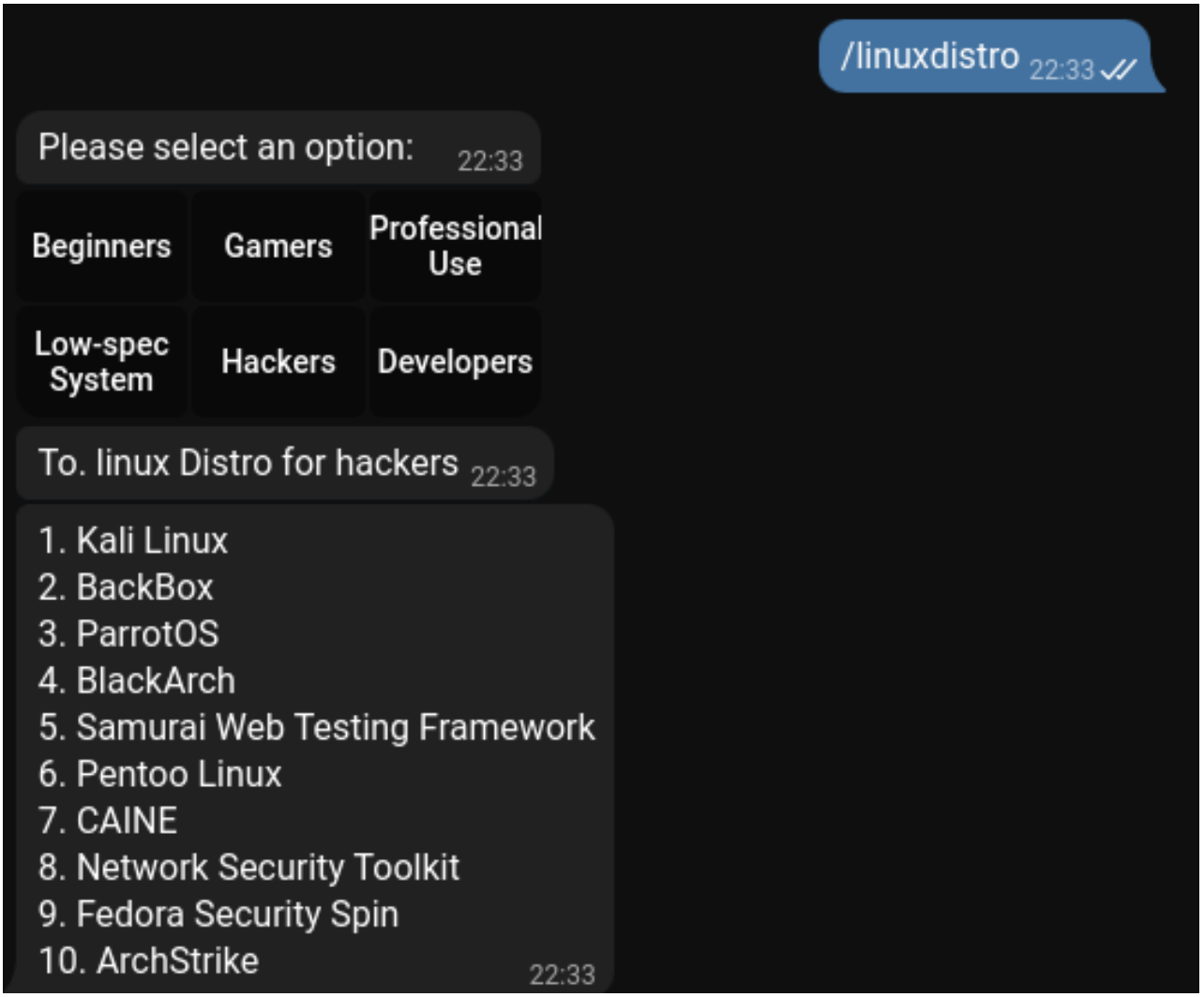
RANDOM ACTIVITY GENERATOR



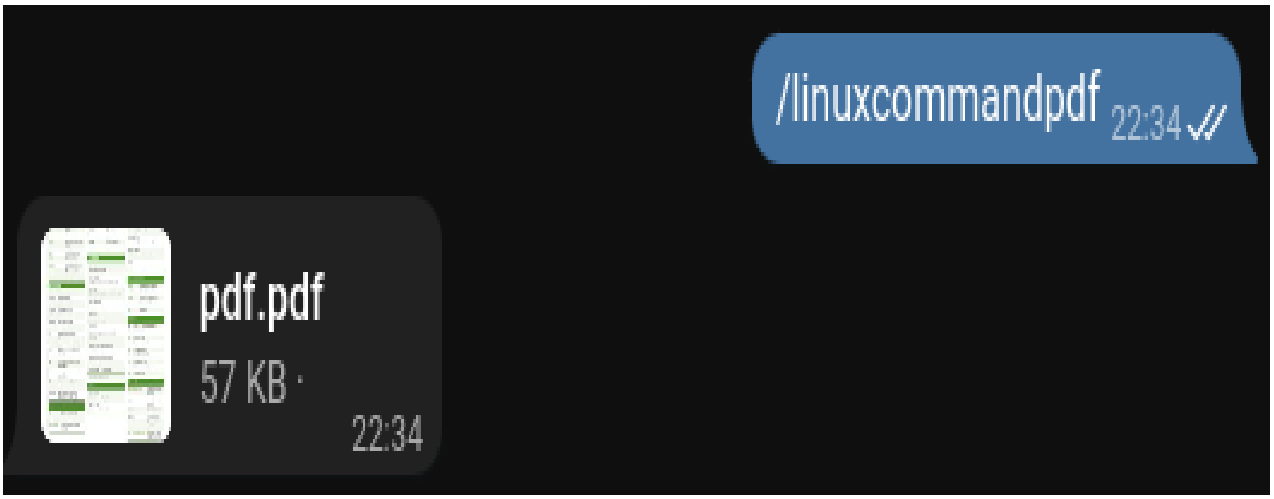
SYLLABUS SENDER



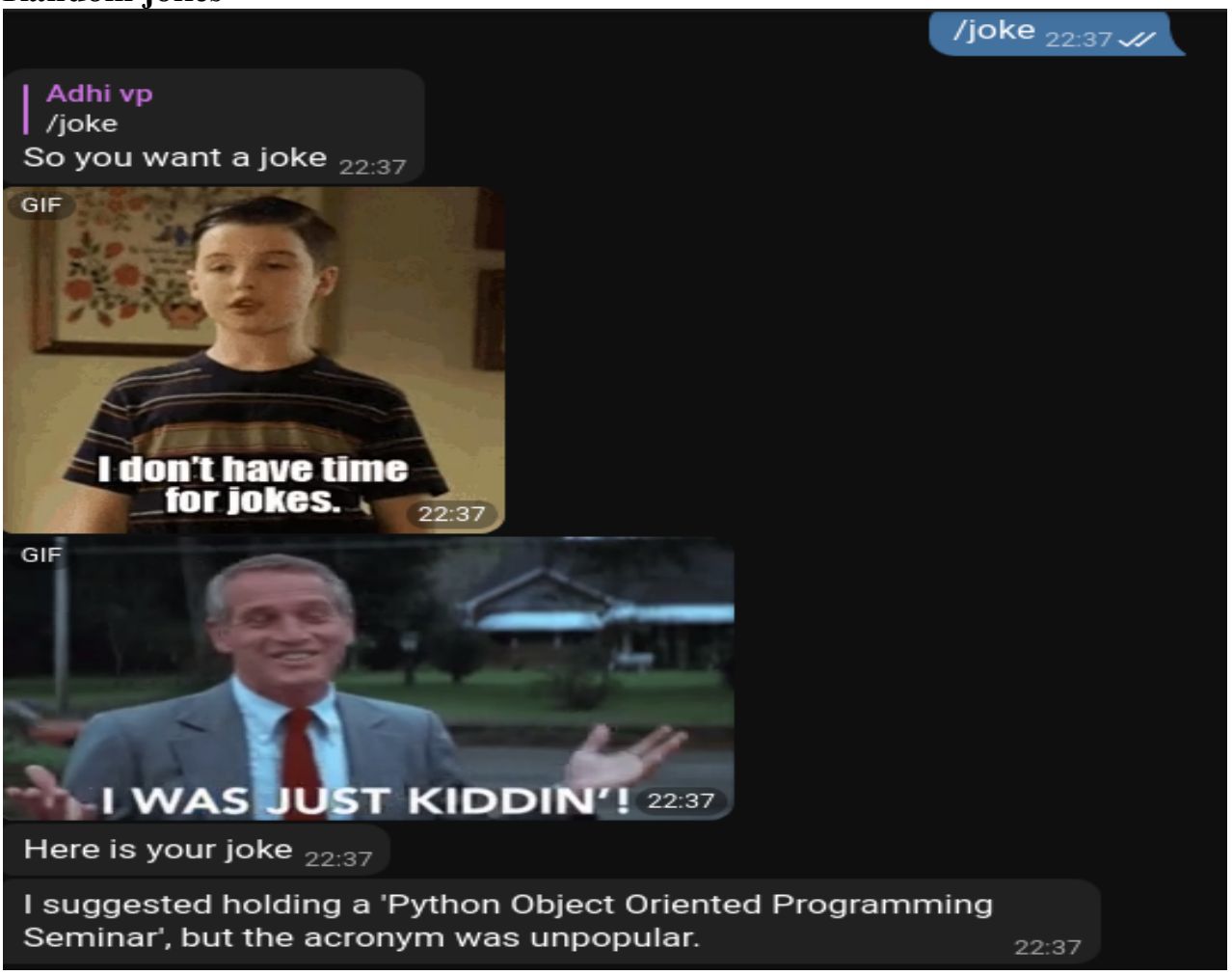
LINUX DISTRO Recommendator



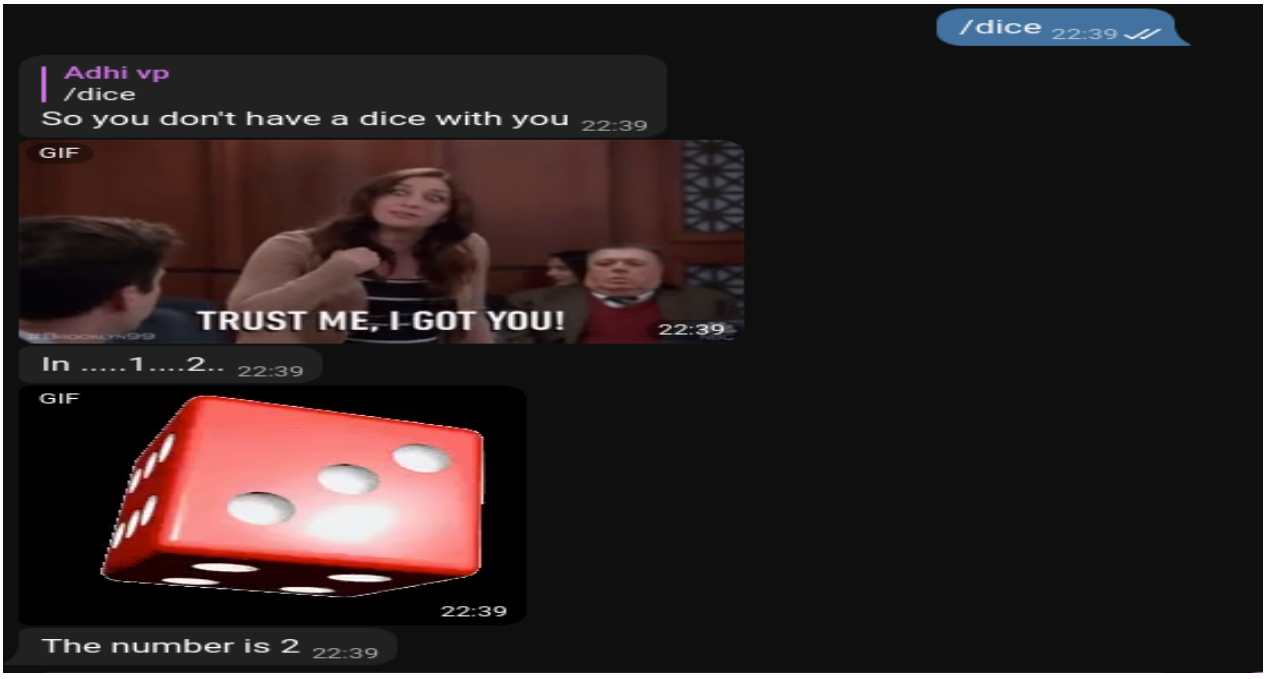
LINUX Commands PDF



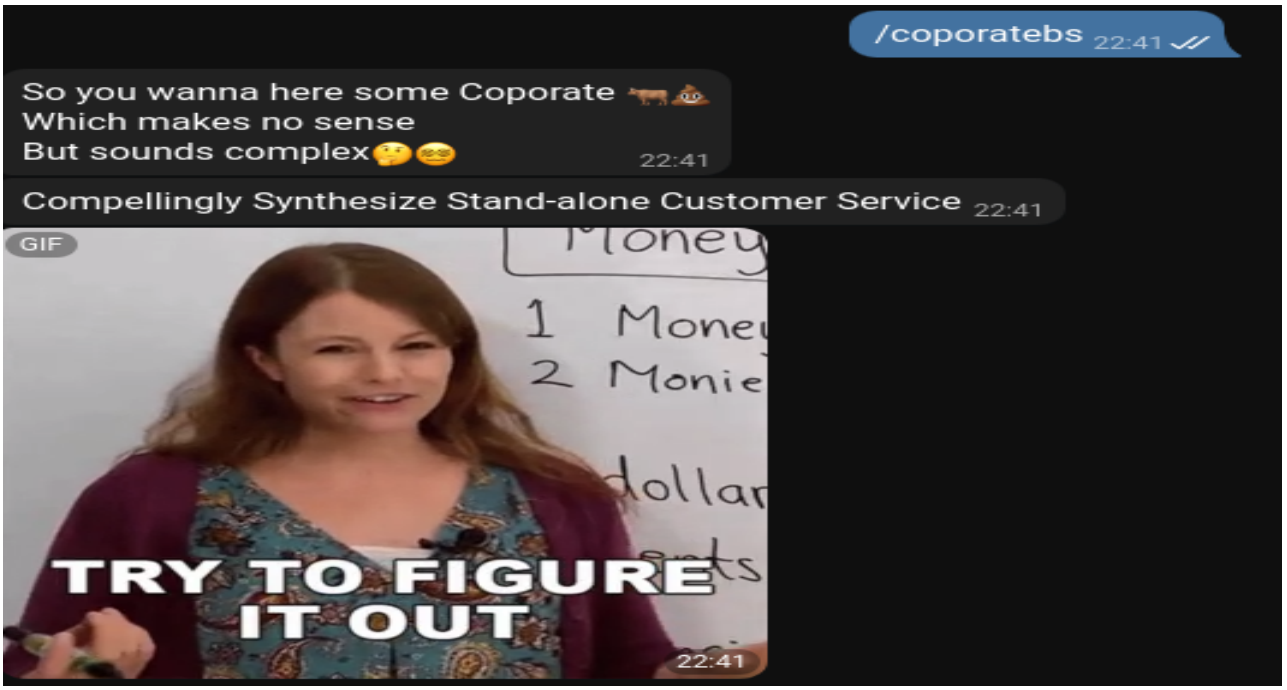
Random jokes



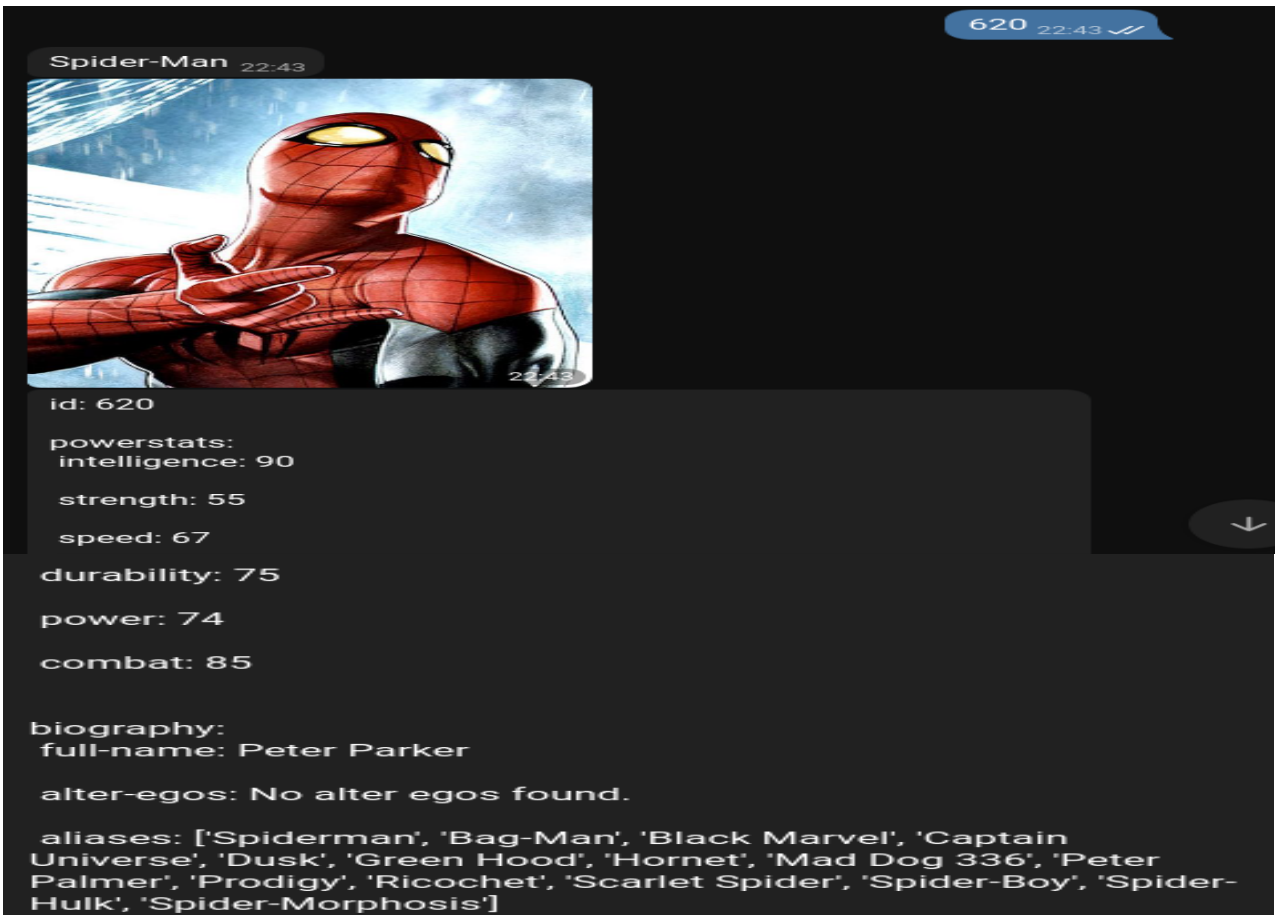
Dice Simulator



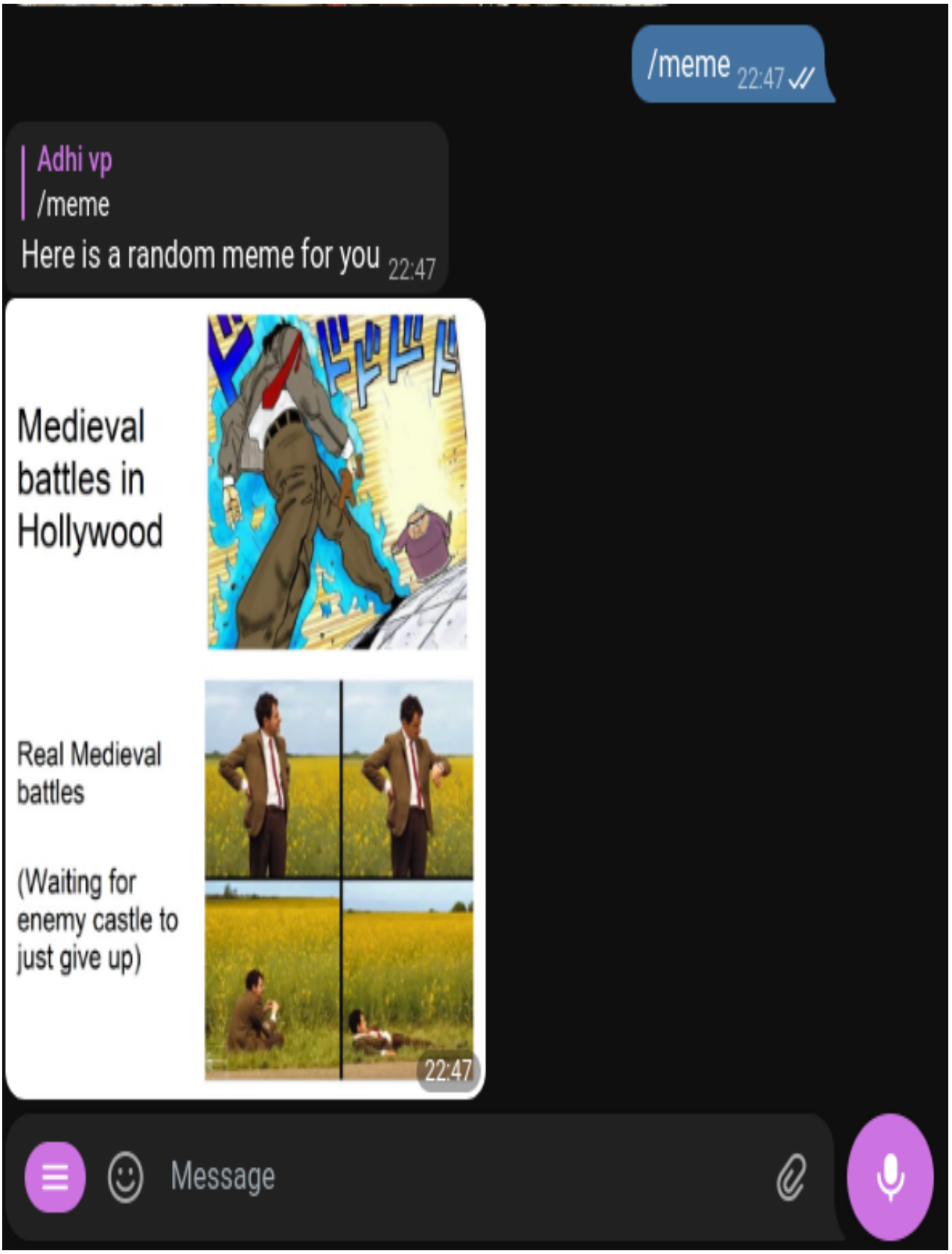
CORPORATE BS



SUPERHERO DEATILS



MEME GENERATOR



Working flow

1. User Initiation: The user starts the interaction by sending the "/start" command to Famalo.

2. Auto Feature Detection or Main Menu: Famalo detects if the user wants to use the auto feature detection by analyzing the input. If the user's input indicates a specific feature request, Famalo proceeds to execute that feature. Otherwise, Famalo presents the main menu options to the user.

3. Main Menu: If the user chooses to explore the main menu, Famalo presents a list of available categories and options to the user. These categories could include education, calculators, generators, real-life information, system-based functionalities, fun activities, and more.

4. Feature Execution: When the user selects a specific feature from the main menu, Famalo executes the corresponding command or process associated with that feature. This could involve accessing external APIs, performing calculations, generating content, or retrieving information.

5. User Interaction and Navigation: Famalo allows the user to engage further by providing additional input or navigating through submenus if applicable. The user can explore more features within the chosen category or go back to the main menu for further options.

6. /List Command: If the user enters the "/list" command, Famalo presents a comprehensive list of all available features, allowing the user to easily browse through the entire range of functionalities.

7. Error Handling and Termination: Famalo incorporates error handling mechanisms to handle unexpected or invalid inputs from users. It provides appropriate error messages or prompts for clarification when necessary.

10.FUTURE SCOPE

The future scope of the Famalo project involves the development of a completely new LLM (Language Model) specifically designed to cater to a wide range of tasks and functionalities. Currently, developers have to manually code each feature in Famalo, which limits the scalability and versatility of the bot. However, with the integration of the LLM model, the potential for expanding the features becomes virtually unlimited.

By leveraging the power of LLM, Famalo can automate the process of feature creation and adaptability. The model's ability to understand natural language and generate responses allows for dynamic and context-aware interactions with users. This opens up new possibilities for adding diverse functionalities and expanding the range of services offered by Famalo, without the need for extensive manual coding.

With the new LLM model, Famalo can quickly learn from user interactions and adapt to evolving user needs. This iterative learning process enables the bot to continuously improve its performance and provide a more personalized user experience. Additionally, the LLM model allows for more efficient development and deployment of new features, reducing the time and effort required to enhance the bot's capabilities.

In summary, the integration of an advanced LLM model in Famalo's development holds tremendous potential for unlocking unlimited features and functionalities. By automating the feature creation process, Famalo can evolve into a more versatile and dynamic platform, providing users with an enhanced and personalized experience. The LLM model streamlines development efforts, allowing for rapid adaptation to user needs and ensuring Famalo remains at the forefront of innovation in the chatbot domain.

This advancement not only improves the efficiency and convenience of Famalo but also introduces a new level of flexibility and scalability. The traditional approach of manual coding restricts the number of features and limits the bot's adaptability. However, with the integration of the LLM model, Famalo gains the ability to generate new features and adapt to evolving user requirements dynamically.

The LLM model's natural language understanding capabilities enable Famalo to provide context-aware responses and interact with users in a more intuitive and personalized manner. Instead of relying solely on predetermined commands and hardcoded responses, Famalo can leverage the power of the LLM model to understand user input and generate appropriate and relevant output.

Furthermore, the integration of the LLM model allows Famalo to rapidly expand its range of services and functionalities without significant manual coding efforts. This means that Famalo can quickly adapt to new use cases and user demands, making it a highly versatile and adaptable bot.

Overall, the integration of the advanced LLM model in Famalo's development paves the way for a more efficient, flexible, and user-centric chatbot experience. It not only enables Famalo to cater to a wide range of tasks and functionalities but also ensures its continuous improvement and ability to meet evolving user needs.

11.CONCLUSION

In conclusion, Famalo stands as a versatile and efficient tool that simplifies various tasks and enhances the user experience. By offering a wide range of functionalities in a single platform, Famalo becomes a valuable asset for users seeking convenience and efficiency. With its intuitive interface and diverse features, Famalo streamlines processes, saves time, and eliminates the need for users to navigate multiple websites or applications. Whether it's accessing educational resources, generating secure passwords, obtaining weather details, or simply engaging in fun activities like telling jokes or rolling dice, Famalo unlocks a new level of convenience and productivity.

12.REFERENCES

- Telegram - https://core.telegram.org/bots/api](https://core.telegram.org/bots/api
- MainFramework https://pypi.org/project/pyTelegramBotAPI/](https://pypi.org/project/pyTelegramBotAPI/
- Features
    https://pypi.org/project/pyjokes/
    https://pypi.org/project/gTTS/
    https://pypi.org/project/random-password-generator/
    https://github.com/D3vd/Meme_Api
    https://github.com/sameerkumar18/corporate-bs-generator-api
    https://www.superheroapi.com/
    https://stthomas.ac.in/
    https://www.boredapi.com/
    https://www.boredapi.com/
    https://pypi.org/project/PyQRCode/