AI ASSISTANCE CHATBOT FOR GOVERNMENT SCHEMES

A PROJECT REPORT

Submitted by

A AAKASH (210701002)
R AAKASH (210701003)

in partial fulfillment for the course

CS19643 – Foundations of Machine Learning

for the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR

THANDALAM

CHENNAI – 602 105

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Thesis titled **"AI ASSISTANCE CHATBOT FOR GOVERNMENT SCHEMES"** is the bonafide work of **"A AAKASH (210701002), R AAKASH (210701003)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.S.Vinod Kumar

PROJECT COORDINATOR

Professor

Department of Computer Science and Engineering

Rajalakshmi Engineering College

Chennai – 602 105

Submitted to project Viva-Voce Examination held on_____

Internal Examiner **External Examiner**

ACKNOWLEDGMENT

First, we thank the almighty god for the successful completion of the project. Our sincere thanks to our chairman **Mr. S. Meganathan B.E., F.I.E.,** for his sincere endeavor in educating us in his premier institution. We would like to express our deep gratitude to our beloved Chairperson **Dr. Thangam Meganathan Ph.d.,** for her enthusiastic motivation which inspired us a lot in completing this project and Vice Chairman **Mr. Abhay Shankar Meganathan B.E., M.S.,** for providing us with the requisite infrastructure.

We also express our sincere gratitude to our college Principal, **Dr. S. N. Murugesan M.E., PhD.,** and **Dr. P. Kumar M.E., PhD,** Director computing and information science **, and** Head Of Department of Computer Science and Engineering and our project **Dr.S.Vinod Kumar** for his encouragement and guiding us throughout the project towards successful completion of this project and to our parents, friends, all faculty members and supporting staffs for their direct and indirect involvement in successful completion of the project for their encouragement and support.

A AAKASH (210701002)

R AAKASH (210701003)

ABSTRACT

In our rapidly evolving technological landscape, accessing government schemes can be daunting for many citizens due to bureaucratic complexities. To address this challenge, we propose an AI chatbot solution developed using MERN Stack technology and Python scripts for web scraping. By leveraging Open Source Transformer models for Natural Language Processing, our chatbot aims to provide personalized assistance in understanding, assessing eligibility for, and applying to government schemes. Unlike existing systems reliant on manual browsing, our solution automates information retrieval and streamlines the user experience. With features like user authentication, eligibility assessment, and proactive notifications, our chatbot represents a significant advancement in citizen engagement with public services, fostering transparency and inclusivity.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 RESEARCH PROBLEM

The complexity and inaccessibility of information regarding government schemes often hinder citizens from fully understanding and utilizing available benefits. This research aims to address this issue by developing an intelligent chatbot that leverages natural language processing (NLP) and web scraping techniques to provide accurate, timely, and comprehensible information about government schemes. The primary research problem is to evaluate the effectiveness of an AI-driven chatbot in improving user engagement and comprehension of government schemes. Key objectives include developing a web scraping mechanism to collect data from official websites, preprocessing and structuring this data for use by an AI model, and integrating a natural language processing model (GPT-4) to deliver human-like responses to user queries. Additionally, a user-friendly frontend interface will be created using React.js to facilitate interactions with the chatbot. The study will measure the chatbot's performance in terms of accuracy, user engagement, comprehension, and efficiency compared to traditional methods of information retrieval. This research will provide insights into the chatbot's potential to enhance user interaction with government schemes and improve the overall accessibility of this critical information.

1.2 PROBLEM STATEMENT

Traditional methods of disseminating this information often fall short, leading to low user engagement and poor comprehension. This research aims to tackle this issue by developing an intelligent chatbot that uses natural language processing (NLP) and web scraping techniques to provide accurate, timely, and easily understandable information about government schemes. The primary goal is to evaluate the effectiveness of this AI-driven chatbot in enhancing user engagement and understanding compared to conventional information retrieval methods. By integrating a cutting-edge AI model (GPT-4) with a user-friendly React.js interface, the project seeks to offer a seamless and efficient user experience. The chatbot's performance will be measured based on its accuracy, response time, adaptability to new information, and overall impact on user satisfaction and comprehension. Additionally, the system's scalability and potential for real-time updates will be assessed to ensure its long-term viability. This study aims to demonstrate the transformative potential of AI-driven tools in improving how citizens interact with and benefit from government schemes, ultimately contributing to a more informed and empowered populace.

1.3 SCOPE OF THE WORK

The scope of this work encompasses the development and evaluation of an intelligent chatbot designed to enhance user interaction with information about government schemes. The project will involve several key phases, starting with the collection of data through web scraping from official government websites. This data will be processed and structured for use by a natural language processing (NLP) model, specifically GPT-4. The chatbot will be integrated into a user-friendly frontend interface built with React.js, facilitating seamless interactions with users. The chatbot's performance will be rigorously assessed in terms of its accuracy, response time, and ability to adapt to new and updated information. Additionally, the project will measure the impact of the chatbot on user engagement and comprehension of government schemes compared to traditional methods. The ultimate goal is to demonstrate the chatbot's potential to significantly improve accessibility and understanding of government schemes for the average citizen. This work will lay the groundwork for future enhancements and applications of AI-driven tools in public information dissemination.

1.4 AIM AND OBJECTIVE

The primary objective of this project is to develop an AI chatbot capable of assisting users in understanding and accessing government schemes efficiently. Specifically, the objectives include:

- Providing users with relevant information about available government schemes based on their queries.
- Determining user eligibility for specific schemes through intelligent conversation and data analysis.
- Guiding users through the application process, including necessary documents and procedures.

1.5 MOTIVATION

The motivation for this project arises from the barriers citizens face in accessing and understanding government scheme information. These schemes are often complex, dispersed across various sources, and written in bureaucratic language that is difficult to comprehend. This project aims to bridge that gap using natural language processing and AI, specifically GPT-4, to develop an intelligent chatbot. This chatbot will provide users with an intuitive way to access relevant information quickly and accurately. Additionally, the project aims to show AI's potential in enhancing public access to information, fostering inclusivity and engagement with government resources.

CHAPTER 2
LITERATURE REVIEW

Upon accessing the AI chatbot, users are greeted with a user-friendly interface that invites them to interact with the system. The first step involves initiating a conversation by typing or speaking their query into the chat interface. The chatbot employs Natural Language Processing (NLP) algorithms to analyze the user's input and generate a relevant response. Through intelligent conversation, the chatbot guides users through a series of inquiries to understand their specific needs and circumstances regarding government schemes.

Next, the chatbot leverages its access to up-to-date information gathered through web scraping from government websites to provide tailored recommendations and guidance. This includes informing users about the availability of relevant schemes, assessing their eligibility based on personal details provided, and explaining the application process step by step. Additionally, the chatbot may prompt users for additional information or documentation required for the application.

Throughout the interaction, users have the option to ask questions, seek clarification, or request further assistance as needed. The chatbot strives to provide clear and concise responses, utilizing

natural language to ensure a seamless and intuitive user experience. Upon completing the interaction, users may receive a summary of the information discussed or relevant links to government resources for further reference. This user-centric approach ensures that individuals can easily access and understand information about government schemes, empowering them to make informed decisions about their entitlements.

2.1 EXISTING SYSTEM

Currently, accessing information about government schemes often involves manual browsing of government websites, which can be time-consuming and confusing for users. Moreover, determining eligibility criteria and application procedures typically requires extensive research and understanding of complex bureaucratic language. Existing chatbots may lack the sophistication to handle nuanced queries or provide accurate guidance.

2.2 PROPOSED SYSTEM

The proposed system aims to overcome the limitations of the existing system by leveraging AI technology to provide personalized assistance and streamline the process of accessing government schemes. Key features of the proposed system include:

Web Scraping: Python scripts will be used to periodically scrape data from government websites, ensuring that the chatbot has access to up-to-date information on available schemes.

Natural Language Processing: An Open Source Transformer model will be employed to enable the chatbot to understand and respond to user queries in natural language, providing a more intuitive and user-friendly experience.

User Authentication: Users may be required to authenticate themselves to access personalized information or submit applications securely.

Eligibility Assessment: The chatbot will analyze user inputs to determine eligibility for specific schemes, providing tailored recommendations based on individual circumstances.

Application Assistance: Step-by-step guidance will be provided to users on how to apply for relevant schemes, including required documentation and procedures.

Notifications: Users may receive notifications about new or updated schemes, deadlines, or changes in eligibility criteria to stay informed.

Feedback Mechanism: A feedback mechanism will be implemented to collect user input and improve the chatbot's performance over time.

CHAPTER 3
SYSTEM DESIGN

3.1 GENERAL

In this section, we would like to show how the general outline of how all the components end up working when organized and arranged together. It is further represented in the form of a flow chart below**.**

3.2 SYSTEM ARCHITECTURE DIAGRAM

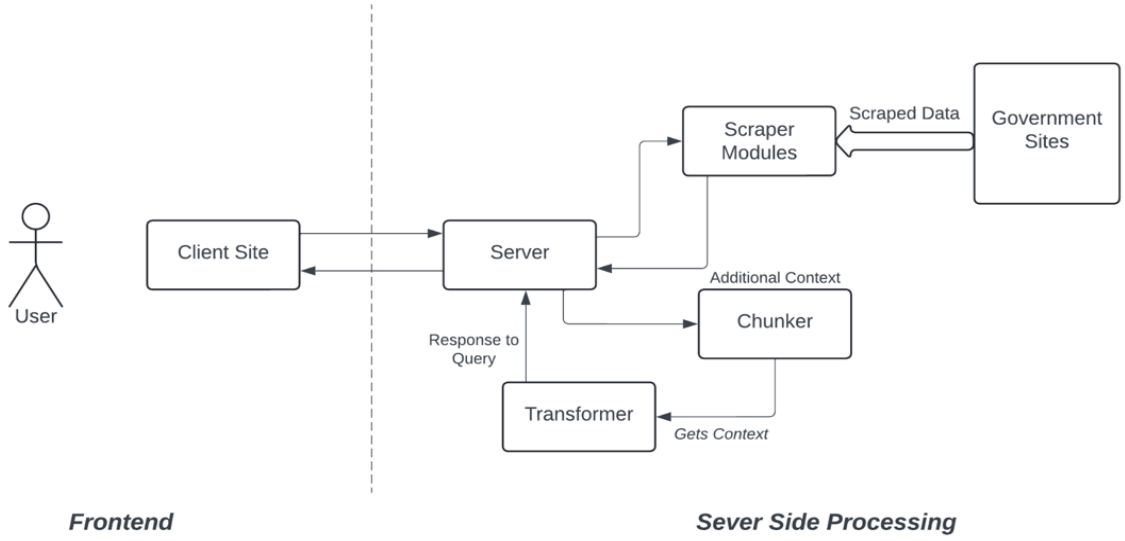

Fig 3.2.1: Architecture Diagram

3.3 DEVELOPMENT ENVIRONMENT

3.3.1 HARDWARE REQUIREMENT

The hardware requirements may serve as the basis for a contract for the system's implementation. It should therefore be a complete and consistent specification of the entire system. It is generally used by software engineers as the starting point for the system design.

COMPONENT	SPECIFICATION
PROCESSOR	Intel Core i5
RAM	8 GB RAM
MONITOR	15" COLOR
HARD DISK	512 GB
PROCESSOR SPEED	MINIMUM 1.1 GHz

Table 3.3.1.1: Hardware Requitement

3.3.2 SOFTWARE REQUREMENT

The software requirements document is the specifications of the system. It should include both a definition and a specification of requirements. It is a set of what the system should rather be doing than focus on how it should be done. As this project is a web based project, all it needs is a working browser with a modern V8 engine.

3.4 SEQUENCE DIAGRAM

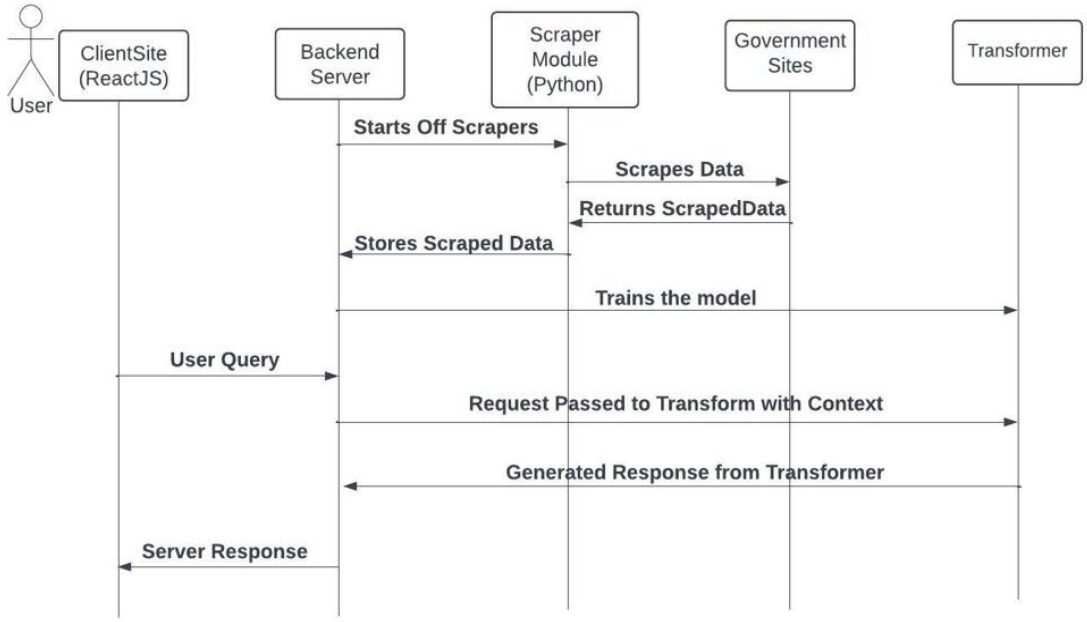

Fig 3.4.1: Sequence Diagram

CHAPTER 4
PROJECT DESCRIPTION

4.1 MODULES

4.1.1 DATA SCRAPING AND PREPROCESSING

The Scraper module is designed to automate the process of data collection from various government websites. Given the absence of a unified API endpoint for accessing information about government schemes, the Scraper module provides a solution by systematically navigating through the HTML structures of target websites using Python and the Beautiful Soup 4 library. It extracts relevant data such as eligibility criteria, benefits, and application procedures, which are then stored in a structured format such as JSON for further processing

4.1.2 SPLITTING RAW DATA INTO CHUNKS

The Chunker module is designed to process the raw JSON data generated by the Scraper module and split it into several manageable data blocks. As the data from government websites can be extensive and complex, the Chunker module ensures that it is organized into smaller, more manageable. Implemented using Python with the JSON module, this module parses the raw JSON data and divides it into discrete sections or subsets based on predefined criteria such as the number of records or the size of the data blocks. By breaking down the data into smaller chunks, the Chunker module optimizes resource utilization and enhances system performance

4.1.3 FEEDING CONTEXT TO TRANSFORMER:

The Transformer module is a pivotal component of our project, leveraging the state-of-the-art capabilities of the GPT-3.5 Turbo model imported through the G4F library. This module plays a crucial role in processing the chunked data generated by the Chunker module and training the transformer model to comprehend and respond to queries about government schemes effectively. Implemented using Python, this module orchestrates the training process, feeding the chunked data as context to the pre-trained transformer model. Through iterative training iterations, the transformer model learns to analyze and interpret the nuances of government scheme-related information, thereby enhancing its proficiency in answering user queries accurately and comprehensively. Furthermore, to optimize the model's performance and specialization in government scheme-related queries, we implement fine-tuning techniques. By constraining the model's training data to exclusively focus on government schemes, we tailor its learning process to prioritize and excel in this specific domain. This fine-tuning process further enhances the model's accuracy, relevance, and responsiveness to user queries, ensuring that it becomes a reliable and valuable resource for users seeking information about government schemes. Overall, the Transformer module constitutes a pivotal component in our project, harnessing the power of advanced AI technology to facilitate seamless interaction and engagement between users and government schemes.

4.1.4 GETTING REQUESTS FROM USER

The Frontend module serves as the user interface for our project, providing a seamless and intuitive platform for users to interact with the system. Built on top of React.js, this module facilitates the exchange of information between users and the backend system.

At its core, the Frontend module features a chat interface where users can input queries related to government schemes. This chat interface is designed to be user-friendly and engaging, resembling a typical messaging platform. Alongside the chat area, a text box and a submit button are provided, enabling users to input their queries conveniently and submit them for processing.

Upon receiving a query from the user, the Frontend module sends a request to the backend, requesting the relevant response from the Transformer module. This request is typically made using asynchronous HTTP requests, allowing for seamless communication between the frontend and backend components of the system.

Once the response is received from the backend, the Frontend module displays it in the chat area, allowing users to view the information provided by the system. This response is presented in a conversational format, mimicking a natural dialogue between the user and the system.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 OUTPUT

The following images contain the images of the chatbot responding user queries:

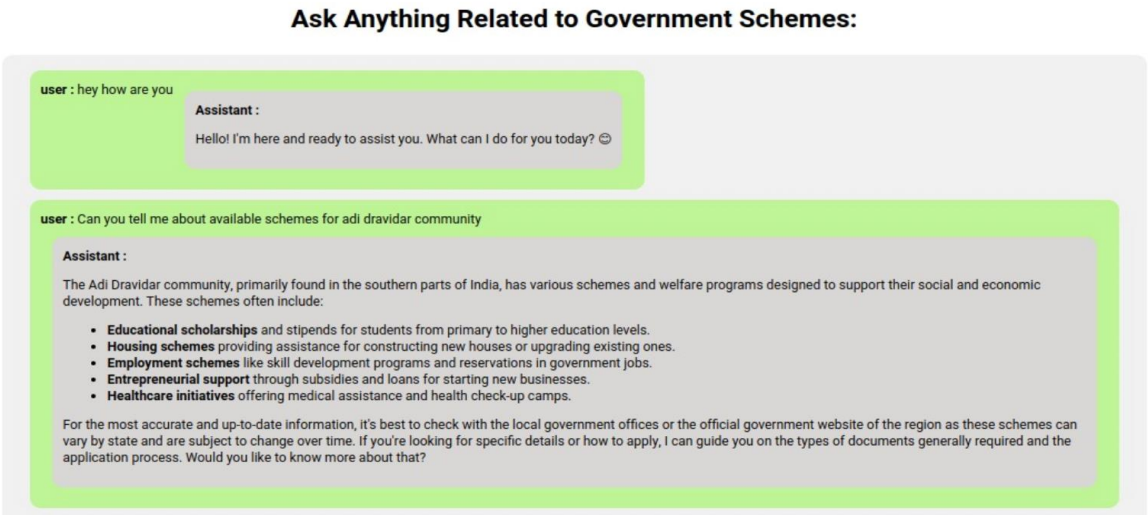

Fig 5.1.1: Basic Querying by User

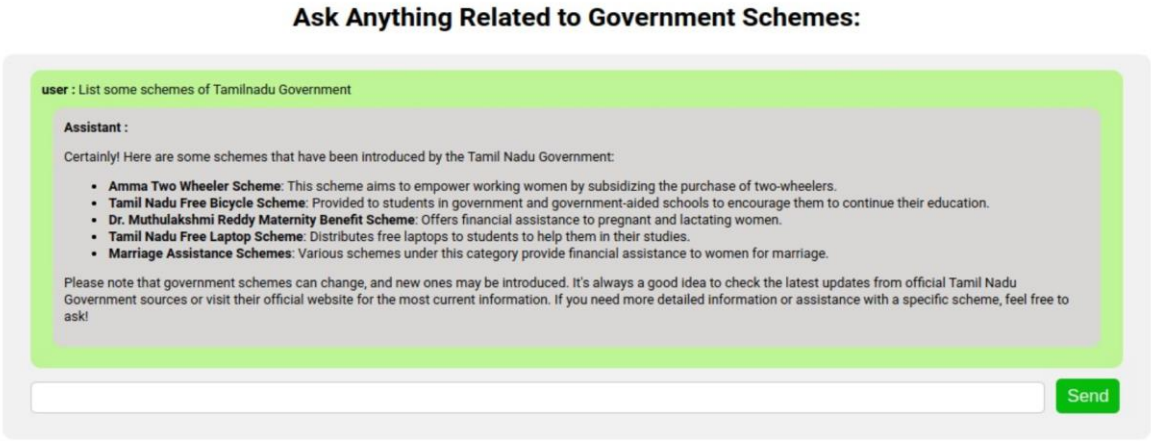

Fig 5.1.2: Listing Queries by User

5.2. RESULT

The result of our project is a transformative solution that significantly enhances citizen access to government schemes. Through the integration of cutting-edge technologies such as AI-driven chatbots, web scraping, and natural language processing, we have developed a robust and user-friendly system that streamlines the process of accessing information about government programs. By automating data collection and organization, our solution eliminates the need for manual browsing of multiple websites and documents, improving accessibility and efficiency. The personalized assistance offered by our chatbot empowers users to understand eligibility criteria, benefits, and application procedures for various government programs, fostering transparency and inclusivity in citizen-government interactions. Moreover, with modular design and fine-tuning capabilities, our system is scalable and adaptable to evolving user needs and government policies. Overall, our project delivers a comprehensive and empowering solution that empowers citizens to navigate the complex landscape of government schemes with confidence and ease, ultimately contributing to a more informed and inclusive society. Furthermore, the implementation of our project not only enhances citizen access to government schemes but also facilitates efficient resource utilization and system performance. By automating data collection and organization through web scraping and chunking techniques, we optimize the handling of extensive and complex datasets, ensuring timely and accurate information retrieval. This streamlined approach not only improves user experience but also enhances the scalability and adaptability of our system to meet the evolving needs of users and government policies.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

In conclusion, our project successfully addresses the challenges faced by citizens in accessing and navigating government schemes through the development of a sophisticated AI-driven chatbot system. By leveraging advanced technologies such as the MERN Stack, Python-based web scraping, and pretrained transformer models for natural language processing, we have created a comprehensive and user-friendly solution that enhances accessibility, efficiency, and transparency. The system's ability to provide personalized assistance, coupled with its robust data management and processing capabilities, ensures that users receive accurate and relevant information in a timely manner. Ultimately, this project not only empowers citizens by simplifying their interaction with government schemes but also contributes to fostering a more informed and inclusive society.

6.2 FUTURE ENHANCEMENT

For future enhancements, our project aims to further expand its capabilities and impact. We plan to incorporate advanced machine learning algorithms to continuously improve the accuracy and relevance of responses provided by the chatbot. Additionally, we can bring multilingual support to cater to a broader audience. Enhancing the user interface with more interactive features and accessibility options will further improve user experience. We also aim to establish partnerships with government agencies to ensure real-time updates and comprehensive coverage of new schemes and policy changes. Moreover, by leveraging user feedback and analytics, we will refine the system to better meet the evolving needs of citizens.

APPENDIX

Scraper.py

```
from bs4 import BeautifulSoup;
import requests;
import json
import re;
import os;

main_text = requests.get('https://www.tn.gov.in/scheme/department_wise/',
verify=False).text;
main_soup = BeautifulSoup(main_text, 'lxml');
maine = main_soup.find_all('div', class_='scheme_list');
main_links = [];
for div in maine:
    p_tag = div.find('p')
    if p_tag:
        a_tag = p_tag.find('a')
        if a_tag and 'href' in a_tag.attrs:
            href_value = a_tag['href']
            main_links.append(href_value)
            print(f"Found href: {href_value}")

for m_link in main_links:
    html_text = requests.get(m_link, verify=False).text;
    soup = BeautifulSoup(html_text, 'lxml');

    full = soup.find_all('div', class_='scheme_lst');
    links = [];
    for div in full:
        p_tag = div.find('p')
        if p_tag:
            a_tag = p_tag.find('a')
            if a_tag and 'href' in a_tag.attrs:
                links.append(href_value)
                print(f"Found href: {href_value}")

    for slink in links:
        spage_text = requests.get(slink, verify=False).text;
        soup = BeautifulSoup(spage_text, 'lxml');
        full = soup.find_all('span', class_='right_column');

        ext_data = [];
        for f in full:
            ext_data.append(f.text);
```

```
    extracted_data = {
        'concerned_dept' :ext_data[0],
        'concerned_dist' :ext_data[1],
        'org_name'       :ext_data[2],
        'name'           :ext_data[3],
        'associated'     :ext_data[4],
        'sponsered_by'   :ext_data[5],
        'funding_pattern':ext_data[6],
        'beneficiaries'  :ext_data[7],
        'benefits'       :ext_data[8],
        'income'         :ext_data[9],
        'age'            :ext_data[10],
        'community'      :ext_data[11],
        'other'          :ext_data[12],
        'how'            :ext_data[13],
        'introduced_on'  :ext_data[14],
        'valid_upto'     :ext_data[15],
        'desc'           :ext_data[17],
    }
    try:
        with open(file_path, 'r') as file:
            existing_data = json.load(file)

        if not isinstance(existing_data, list):
            existing_data = [existing_data]

    except FileNotFoundError:
        existing_data = []

    existing_data.append(extracted_data)
    with open(file_path, 'w') as file:
        json.dump(existing_data, file, indent=2)
```

Chunker.py

```
import json
import os

with open('../Datasets/raw_data.json', 'r') as file:
    data = json.load(file)

chunk_size = 50
chunks = [data[i:i+chunk_size] for i in range(0, len(data), chunk_size)]
```

```
for i, chunk in enumerate(chunks):
    new_filename = f'../Datasets/preprompt_ds_{i + 1}.json'
    with open(new_filename, 'w') as new_file:
        json.dump(chunk, new_file, indent=2)

print(f"split into {len(chunks)} chunks.")

os. remove(f'../Datasets/raw_data.json')
```

Api.py

```
import g4f
import sys
import json
import os

def read_file_content(file_path):
    try:
        with open(file_path, 'r') as file:
            return file.read()
    except FileNotFoundError:
        print(f"Error: File not found at {file_path}")
        sys.exit(1)
    except Exception as e:
        print(f"Error reading file: {e}")


# Hardcoded path to the folder containing external text files
external_files_folder = './preprompts'

# User-provided JSON data as a command line argument
user_json_data = sys.argv[1]

try:
    # Parse the user-provided JSON string into a Python data structure
    user_data = json.loads(user_json_data)
except json.JSONDecodeError as e:
    print(f"Error decoding user-provided JSON: {e}")
    sys.exit(1)

g4f.debug.logging = False  # Enable debug logging
g4f.debug.version_check = False  # Disable automatic version checking
# print(g4f.Provider.Bing.params)  # Print supported args for Bing
# print(type(user_data))  # Ensure user_data is of the correct type

# Iterate over each external file in the folder
```

```
for file_name in os.listdir(external_files_folder):
    file_path = os.path.join(external_files_folder, file_name)

    # Read the content from the external file
    external_content = read_file_content(file_path)

    # Create a new element to prepend to the user-provided list
    external_element = {"role": "user", "content": external_content}

    user_data.append(external_element)

user_data.append({"role": "user", "content": user_json_data})

# Combine the user-provided data with the additional external elements
data = user_data

# Using automatic a provider for the given model
## Streamed completion
response = g4f.ChatCompletion.create(
    # model="gpt-3.5-turbo",
    model = "gpt-4",
    provider = g4f.Provider.Bing,
    messages=data,
    stream=True,
```

REFERENCES

[1] “Deep Learning" by Ian Goodfellow, Yoshua Bengio(published in 2018).

[2] "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron (Published in 2019)

[3] "Python Deep Learning" by Ivan Vasilev and Daniel Slater in 2017.

[4] "Deep Learning with Python" by Francois Chollet (Published in 2017).

[5] "Practical Python and OpenCV" by Adrian Rosebrock (Published in 2019)

[6] “Machine Learning Yearning" by Andrew Ng (Published in 2018)

[7] "Python Deep Learning Projects" by Matthew Lamons, Rahul Kumar, and Abhishek Nagaraja (Published in 2019)

[8] “Blockchain and the Supply Chain Concepts, Strategies and Practical Applications" by Tiana Laurence (Published in 2019)