A SMART CHATBOT APPLICATION TO STIMULATE HUMAN TEXT BASED TO SOLVE CUSTOMER PROBLEM

A Project Report
Submitted by

Ritik Kumar Saraogi	20201CAI0091
Vicky Kumar	20201CAI0092
Mann Kumar	20201CAI0113
Aryateja N S	20201CAI0102

Under the guidance of

Dr. Mohammadi Akheela Khanum Professor in partial fulfillment for the award of the degree of BACHELOR OF TECHNOLOGY

IN

B.Tech Computer Science & Engineering (Artificial Intelligence & Machine Learning)

At

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
PRESIDENCY UNIVERSITY BENGALURU

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Project report "**A SMART CHATBOT APPLICATION TO SIMULATE HUMAN TEXT-BASED INTERACTION TO SOLVE CUSTOMER PROBLEMS PRIZE USING AI**"

Being submitted by "Ritik Kumar Saraogi, Vicky Kumar, Mann Kumar, Aryateja N S"

bearing roll number(s) "20201CAI0091, 20201CAI0092, 20201CAI0113, 20201CAI0102" in partial fulfilment of requirement for the award of degree of Bachelor of Technology in Computer and

Artificial Intelligence is a bonafide work carried out under my supervision.

Dr. Mohammadi Akheela Khanum

Professor
School of CSE&IS
Presidency University

Dr. Zafar Ali Khan

Prof. &HOD-ECE&CCE
School of CSE&IS
Presidency University

Dr. C. KALAIARASAN

Associate Dean School of CSE&IS Presidency University

Dr. L. SHAKKEERA

Associate Dean School of CSE&IS Presidency University

Dr. SAMEERUDDIN KHAN

Dean
School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled "A SMART CHATBOT APPLICATION TO SIMULATE HUMAN TEXT-BASED INTERACTION TO SOLVE CUSTOMER PROBLEMS USING AI" in partial fulfilment for the award of Degree of Bachelor of Technology in Computer Science Engineering (AIML) is a record of our own investigations carried under the guidance Dr.Mohammadi Akheela Khanum Professor, School of Computer Science & Engineering, Presidency University, Bengaluru

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Ritik Kumar Saraogi	20201CAI0091
Vicky Kumar	20201CAI0092
Mann Kumar	20201CAI0113
Aryateja N S	20201CAI0102

ABSTRACT

In today's dynamic digital landscape, the development of a Smart Chatbot Application holds immense significance for my final year engineering project in artificial intelligence and machine learning. This project is a fusion of advanced AI and ML techniques to create a chatbot capable of simulating human-like text-based interactions, a crucial component in modern customer support services.

The primary objective of this project is to design a chatbot that can decipher and resolve customer complaints and queries. To achieve this, we'll leverage natural language processing (NLP) to understand and process customer inputs. The chatbot will then efficiently search through a database to find appropriate solutions. An innovative twist lies in its ability to recognize new, previously undocumented solutions, in which case it deftly hands over the conversation to support staff for resolution.

Furthermore, the chatbot's learning capability is a vital aspect of this project. By analyzing interactions between customers and support staff, it continually updates its knowledge base, making it better equipped to tackle similar issues in the future.

This project presents a prime example of AI and ML applications in realworld problem-solving, contributing to efficient customer support and elevating user satisfaction. It's a journey into the practical realm of AI and machine learning, aimed at enhancing human-computer interactions and redefining customer service in the digital age.

ACKNOWLEDGEMENT

First of all, we indebted to the GOD ALMIGHTY for giving me an opportunity to excel in our efforts to complete this project on time.
We express our sincere thanks to our respected dean Dr. Md. Sameeruddin Khan, Dean, School of Computer Science and Engineering and School of Information Science, Presidency University for getting us permission to undergo the project.
We record our heartfelt gratitude to our beloved Associate Deans Dr. Kalaiarasan C and Dr. Shakkeera L, School of Computer Science and Engineering and School of Information Science, Presidency University and Dr. Zafar Ali Khan, Head of the Department, School of Computer Science and Engineering, Presidency University for rendering timely help for the successful completion of this project.
We are greatly indebted to our guide Dr Mohammadi Akheela Khanum, Assistant Professor, School of Computer Science and Engineering, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.
We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS and the department Project Coordinator Dr Murali Parameswaran.
We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Ritik Kumar Saraogi – 20201CAI0091
Vicky Kumar – 20201CAI0092
Mann Kumar – 20201CAI0113
Aryateja N S– 20201CAI0102

TABLE OF CONTENTS

CHAPTER 1- INTRODUCTION

Welcome to the comprehensive engineering report detailing the development and implementation of our pioneering Smart Chatbot Application. This groundbreaking project represents the convergence of advanced artificial intelligence, natural language processing, and innovative customer service solutions. Our mission has been to engineer a state-of-the-art system capable of simulating human-like text-based interactions, providing effective solutions to customer problems, and dynamically adapting to user feedback for continuous improvement.

In the following pages, we delve into the intricate design, development, and functionality of the Smart Chatbot Application. From the initial conceptualization to the practical implementation of cutting-edge technologies, this report offers an in-depth exploration of the engineering journey undertaken to create a solution that redefines the landscape of customer support. Throughout this report, we will elucidate the core engineering principles, algorithms, and methodologies employed in crafting the Smart Chatbot. Our focus extends beyond mere functionality to encompass the robustness, scalability, and adaptability of the system, ensuring its seamless integration into diverse customer service environments.
Furthermore, we will delve into the intricacies of the scoring mechanism—a pivotal element of our innovative approach. This system, dynamically influenced by real-time customer feedback, serves as the cornerstone for optimizing solutions and prioritizing the most effective responses to common queries.

As we traverse through the technical aspects of the Smart Chatbot Application, we invite you to explore the culmination of interdisciplinary efforts, where engineering prowess intersects with artificial intelligence to create a transformative solution. The insights shared here not only represent the project's current status but also set the stage for further improvements and developments in the ever-evolving field of intelligent customer service apps.

CHAPTER 2- LITERATURE REVIEW

PAPER TITLE	AUTHOR NAME
[1] AI-based Chatbots in Customer Service and Their Effects on User Compliance	Martin Adam, Michael Wessel and Alexander Benlian
[2] My Chatbot Companion - a Study of Human-Chatbot Relationships	Marita Skjuve, Asbjørn Følstad , Knut Inge Fostervold and Petter Bae Brandtzaega
[3]Designing Interactive Chatbot Development Systems	Jasper Feine, Alexander Maedche, Stefan Morana.
[4]A Longitudinal Study of Human–Chatbot Relationships	Marita Skjuve, Asbjørn Følstad, Knut Inge Fostervold and Petter Bae Brandtzaeg.
[5]Increasing Customer Service Efficiency Through AI Chatbot	Ivan Martins De Andrade and Cleonir Tumelero.
[6]A Mental Health Chatbot with Cognitive Skills for Personalized Behavioral Activation and Remote Health Monitoring	Prabod Rathnayaka, Richard Gray, Nishan Mills , Donna Burnett , Daswin De Silva * and Damminda Alahakoon
[7]Feedback-Based Self-Learning in Large-Scale	Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, Ruhi Sarikaya.

[8]Conversational AI Agents	Kyoko Sugisaki.
[9]Chat-Bot Kit	Sophia Keyner, Vadim Savenkov, and Svitlana Vakulenko.
[10]Unsupervised Context Rewriting for Open Domain Conversation	Kun Zhou∗ , Kai Zhang, Yu Wu, Shujie Liu, and Jingsong Yu.

CHAPTER 2.1- LITERATURE REVIEW SUMMARY

AI-based Chatbots in Customer Service and Their Effects on User Compliance

- This paper explores how AI-driven chatbots are used in customer service. It examines how these chatbots influence users to follow guidelines and rules when interacting with them. The findings shed light on the impact of these chatbots on user behavior and compliance.

My Chatbot Companion - a Study of Human-Chatbot Relationships

- This study investigates the relationships between people and chatbots. It explores how users perceive and interact with chatbots, examining the emotional and practical aspects of these relationships. The research aims to understand the dynamics of human-chatbot connections.

Designing Interactive Chatbot Development Systems

- This research focuses on creating systems for developing interactive chatbots. It delves into the tools and methods used to design chatbots that can engage users effectively. The study seeks to improve the development of chatbot technology.

A Longitudinal Study of Human–Chatbot Relationships

- This study is a long-term investigation into how humans build relationships with chatbots over time. It explores the evolving dynamics and changing attitudes in these interactions. The goal is to understand how user-chatbot relationships develop and mature.

Increasing Customer Service Efficiency Through AI Chatbots

- This paper examines how AI-powered chatbots can enhance the efficiency of customer service. It discusses how chatbots can help resolve customer inquiries and issues more quickly and effectively, leading to improved customer service.

A Mental Health Chatbot with Cognitive Skills for Personalized Behavioral Activation and Remote Health Monitoring

- This research introduces a chatbot designed to support individuals dealing with mental health issues. The chatbot employs cognitive techniques to offer personalized emotional assistance and monitor users' mental health remotely. The goal is to enhance mental health support.

Feedback-Based Self-Learning in Large-Scale Conversational AI Agents

- This study explores how large-scale conversational AI agents can learn and adapt based on user feedback. It discusses a system that uses feedback from users to enhance the performance of these AI agents, leading to improved user experiences.

Chat-Bot-Kit: A web-based tool to simulate text-based interactions between humans and with computers

- This paper presents a web-based tool for simulating text-based conversations between humans and computers. It is designed for research in computermediated communication, human-computer interaction, and natural language processing. The tool helps researchers study text-based chats, including user interactions and behaviors.

Open Data Chatbot

- This research combines chatbots and open data to help users discover government open datasets through conversations. It discusses how chatbots can provide a conversational search interface to access open data. The chatbot aims to promote government transparency and citizen participation.

Unsupervised Context Rewriting for Open Domain Conversation

- This paper explores a method for improving open-domain conversations. It focuses on context rewriting, a technique to enhance the flow and relevance of conversations. The goal is to make open-domain chatbot conversations more coherent and engaging.

CHAPTER 3- PROPOSED METHODS

1. Data Collection and Preprocessing
2. Natural Language Processing (NLP)
3. Knowledge Base Development
4. Search and Matching Algorithm
5. Solution Recognition
6. Handover Mechanism
7. Learning and Adaptation
8. Continuous Improvement
9. Security and Privacy
10. Multichannel Support and Scalability

CHAPTER 3.1- PROPOSED METHODS SUMMARY

Data Collection and Preprocessing:

-Involves gathering relevant information and preparing it for analysis. -This process ensures that the data is clean, organized, and suitable for further processing.

Natural Language Processing (NLP):

-A branch of artificial intelligence that gives machines the ability to comprehend, translate, and produce human language.
-Includes activities such as sentiment analysis, language production, and language comprehension.

Knowledge Base Development:

-Creating a structured repository of information that a system can use to enhance decision-making or problem-solving.
-This knowledge base may include facts, rules, and relationships relevant to a particular domain.

Search and Matching Algorithm:

-Algorithms designed to efficiently find and retrieve relevant information from a dataset.
-Used in various applications, such as search engines or matching user queries to a knowledge base.

Solution Recognition:

-Identifying and understanding potential solutions to a problem based on available data and knowledge.
-Often involves pattern recognition and analysis of historical or existing solutions.

Handover Mechanism:

-The process of transferring control or information from one system or component to another.

-Ensures smooth transitions and continuity of operations, especially in multicomponent systems.

Learning and Adaptation:

-The ability of a system to acquire new knowledge or skills and adjust its behavior accordingly.
-Machine learning techniques are often employed for this purpose.

Continuous Improvement:

-Ongoing efforts to enhance system performance, efficiency, and effectiveness over time.
-Involves feedback loops, analysis of performance metrics, and iterative refinement.

Security and Privacy:

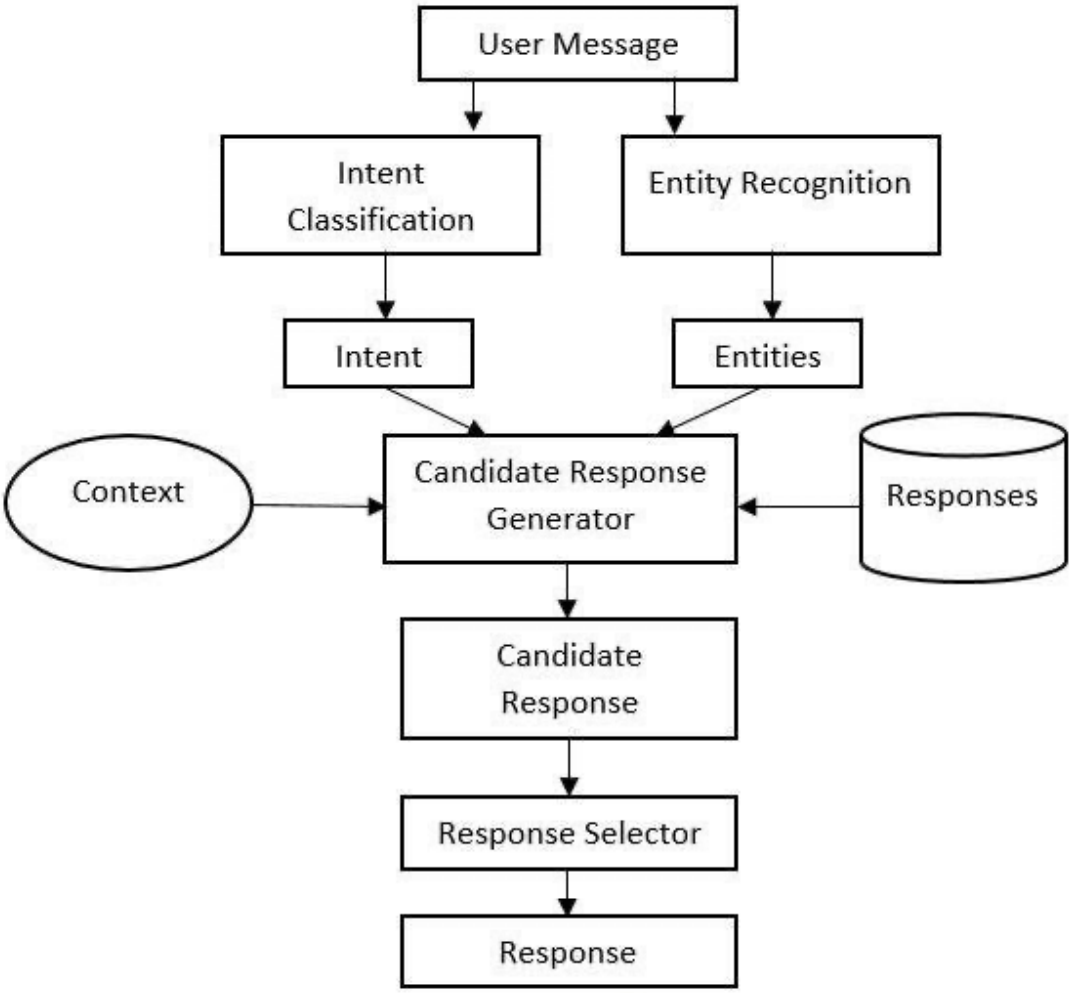
-Measures put in place to secure information, systems, and user privacy; these include policies, access controls, and encryption to prevent illegal access and data breaches.

Multichannel Support and Scalability:

-Ensuring a system can handle interactions from multiple channels (e.g., web, mobile, chat).

-Designing the system to scale efficiently as the volume of data or user interactions increases.

CHAPTER 4- ARCHITECTURE DESIGN



CHAPTER 5- PROBLEM STATEMENT

In the dynamic digital landscape, conventional customer support services struggle to meet the escalating demands for timely and personalized assistance. The surge in the complexity of user queries calls for a transformative solution. With the use of cutting-edge artificial intelligence (AI) and machine learning (ML) techniques, this project develops a smart chatbot application to meet the urgent need for a creative customer care system. The core challenge lies in creating a chatbot capable of emulating human-like text-based interactions through effective Natural Language Processing (NLP) algorithms. Integrating AI and ML techniques is pivotal to enhancing the chatbot's learning capabilities. By analyzing real-time interactions between customers and support staff, the chatbot continuously updates its knowledge base, ensuring it evolves and becomes proficient in addressing diverse queries.

The project's objectives include the seamless integration of AI and ML, the implementation of continuous learning mechanisms, and the assessment of the chatbot's efficiency in real-world problem-solving. The ultimate goal is to redefine customer service in the digital age, contributing to elevated user satisfaction through efficient, context-aware interactions. This project serves as a practical exploration of AI and ML applications, demonstrating their tangible impact on reshaping and optimizing customer support services.

CHAPTER 6- OBJECTIVES

- Reduce Support Staff Workload
- Feedback Integration
- Ranking Queries In The Chat Bot
- Update Database

OBJECTIVES SUMMARY

Reduce Support Staff Workload

This refers to leveraging a chatbot to handle routine and frequently asked questions, thereby reducing the burden on human support staff. Chatbots can be programmed to provide instant responses to common
queries, guide users through troubleshooting steps, or offer information on common topics. Support employees can concentrate on more complicated problems that call for human interaction by automating these chores, which will increase overall productivity and service quality..

Feedback Integration

Feedback integration involves incorporating mechanisms within the chatbot to gather user feedback on its performance. This could include prompts for users to rate the helpfulness of responses, provide comments on their experience, or report any issues they encountered. Analyzing this feedback helps in continuously improving the chatbot's accuracy, effectiveness, and user satisfaction. It allows developers to identify areas for enhancement and refine the chatbot's capabilities based on user input.

Ranking Queries In The Chat Bot

Putting in place a system that ranks user queries according to relevance and significance entails ranking questions. algorithms for natural language processing (NLP) that assess the context, intent, and urgency of user inquiries can do this. By ranking queries, the chatbot can address high-priority issues first, ensuring a more efficient and satisfactory user experience. It also helps in optimizing the flow of conversations and providing quicker resolutions to critical issues.

Update Database:

Updating the database involves regularly refreshing the chatbot's knowledge base or information repository. This ensures that the chatbot has access to the latest data, trends, and information relevant to its domain. Regular updates can include adding new FAQs, incorporating recent product or service details, and staying current with industry developments. This process is crucial for maintaining the accuracy and relevance of the chatbot's responses, ensuring that users receive up-to-date and reliable information.

CHAPTER 7- GANTT CHART

CHAPTER 8- IMPLEMENTATION PLAN

1. NLP and Text Processing:

- spaCy: An industrial-strength natural language processing library for text processing.
- NLTK (Natural Language Toolkit): A library for working with human language data.

2. Web Framework (for User Interface):

- Flask: A lightweight web framework for building the user interface of your chatbot.
- Django (optional): A more feature-rich web framework that can be used for developing complex applications.

3. Database Management:

- SQL Alchemy (for relational databases): An SQL toolkit and Object-Relational Mapping (ORM) library.
- pymongo (for MongoDB): A Python driver for MongoDB, if you're using a NoSQL database.

4. Search Engine:

- Elasticsearch (optional): A powerful search engine that can be used for efficient data retrieval.

5. Support Staff Interface:

- Integration with customer support tools like Zendesk, Freshdesk, or custom support ticketing systems through their respective APIs.

6. Cloud Services (if applicable):

- If you're hosting your application in the cloud, you may need Python SDKs or libraries for the specific cloud platform you choose (e.g., boto3 for AWS, azuresdk for Azure).

7. Analytics and Reporting (optional):

● Google Analytics (for web analytics).
● Custom analytics solutions can be built using Python libraries for data analysis (e.g., pandas, matplotlib).

8. Version Control and Testing:

● Git: An application version control solution for managing your project's source code.
● Frameworks like Pytest for unit testing code.

9. Security:

● Python libraries for security and encryption, as well as standard security practices for securing your chatbot.

10. API Development:

● Libraries like Flask-RESTful for building RESTful APIs if your chatbot has API endpoints.

11. Machine Learning and Deep Learning (for advanced NLP - optional):

scikit-learn for machine learning tasks. - Deep learning libraries such as TensorFlow, PyTorch, or Hugging Face's transformers for building and using advanced NLP models.

12. Deployment and Containerization (if applicable):

● Docker: If you're containerizing your application for easier deployment and scaling. - Orchestration tools like Kubernetes for container management (if needed).

13. Logging and Monitoring (optional):

● Python libraries for logging (e.g., loguru, log4j) and monitoring tools like Prometheus and Grafana.

CHAPTER 9- EXPERIMENTAL DETAILS

Operating System:

● Choose a Linux distribution like Ubuntu or CentOS for server deployments. These operating systems are commonly used for hosting Python applications.

Web Server (if applicable):

● For serving the web-based user interface, you can use Nginx or Apache as a reverse proxy in front of your Python application server (e.g., Gunicorn or uWSGI).

Database Management System:

● Select a database system compatible with Python. Options include PostgreSQL (using the psycopg2 library), MySQL (using mysql-connector or pymysql), MongoDB (using pymongo), or other database solutions based on your project's requirements.

Python:

● Python is your primary programming language. Make sure to install Python on your server. It's recommended to use a virtual environment (e.g., venv or conda) to manage Python dependencies.

NLP Libraries:

● Depending on your choice of NLP libraries, you may need to install spaCy, NLTK, and/or the transformers library for handling natural language processing tasks in Python.

Web Framework:

● If you're building a web-based user interface, consider using Python web frameworks like Flask or Django for the backend. These can be installed using pip.

Containers (Optional):

● If you use Docker for containerization, ensure that Docker is installed on your server. You'll also need to create and manage Docker containers for your application.

Version Control:

● Set up Git for version control. Git can be installed using the package manager of your operating system.

Analytics and Monitoring Tools:

● Install and configure any analytics or monitoring tools you plan to use, such as Google Analytics, Prometheus, or Grafana. You may need Python packages for integrating with these tools.

Security Software:

● Implement security measures and secure your application with Python libraries and tools. Examples of these include firewalls, SSL/TLS certificates for secure communication, and intrusion detection systems.

Tools for Continuous Deployment and Integration (CI/CD):

● By putting up technologies like Jenkins, Travis CI, or CircleCI, you can automate the testing and deployment of your Python application if you use continuous integration and deployment (CI/CD).

AI/ML Frameworks (Optional):

● If you're implementing custom AI/ML models, you may need to install Python libraries like TensorFlow, PyTorch, or scikit-learn for machine learning and deep learning tasks.

CHAPTER 10- CONCLUSION

In conclusion, the project A Smart Chatbot Application to Simulate Human Text-Based Interaction to Solve Customer Problems has been guided by a comprehensive literature review encompassing a range of research papers. The insights drawn from these papers have collectively shaped the project's foundation and objectives.

The project is underpinned by the understanding that chatbots play a pivotal role in enhancing customer experience, as emphasized in The Role of Chatbots in Enhancing Customer Experience. Moreover, In [1] has shed light on the importance of compliance in service-oriented chatbot solutions, aligning with our project's focus on customer problem-solving.

Research exploring Human-Chatbot Relationships and the Longitudinal Study of Human–Chatbot Relationships has provided essential insights into creating chatbots that can simulate human-like interactions. In [3] has offered guidance on the design principles, and In [7] has reinforced the significance of continuous learning in chatbots.

The aspiration to increase customer service efficiency through AI chatbots, as outlined in In [5], is at the heart of our project's goals. In [6] has highlighted the potential for cognitive skills and personalization in customer problem-solving.

The inclusion of Chat-Bot-Kit as a tool to simulate text-based interactions has facilitated the practical development of our chatbot. In [9] suggests the advantages of integrating open data sources to enrich the chatbot's knowledge base, while Unsupervised Context Rewriting for Open Domain Conversation underscores the importance of context-aware responses.

In summary, this project amalgamates the wisdom from these research papers to create a versatile and intelligent chatbot capable of simulating human textbased interaction to effectively solve customer problems and significantly enhance customer experience in various domains.

Paper [9] could help us enhance the chatbot's knowledge base, while [10] could aid in making responses more coherent. [8] is a practical tool for testing and training your chatbot, and [7] offers insights on how to continuously improve the chatbot's performance through user feedback. Incorporating these concepts and tools into the project can help make our chatbot smarter and more user-friendly,

ultimately contributing to its effectiveness in solving customer problems through natural text-based interactions.

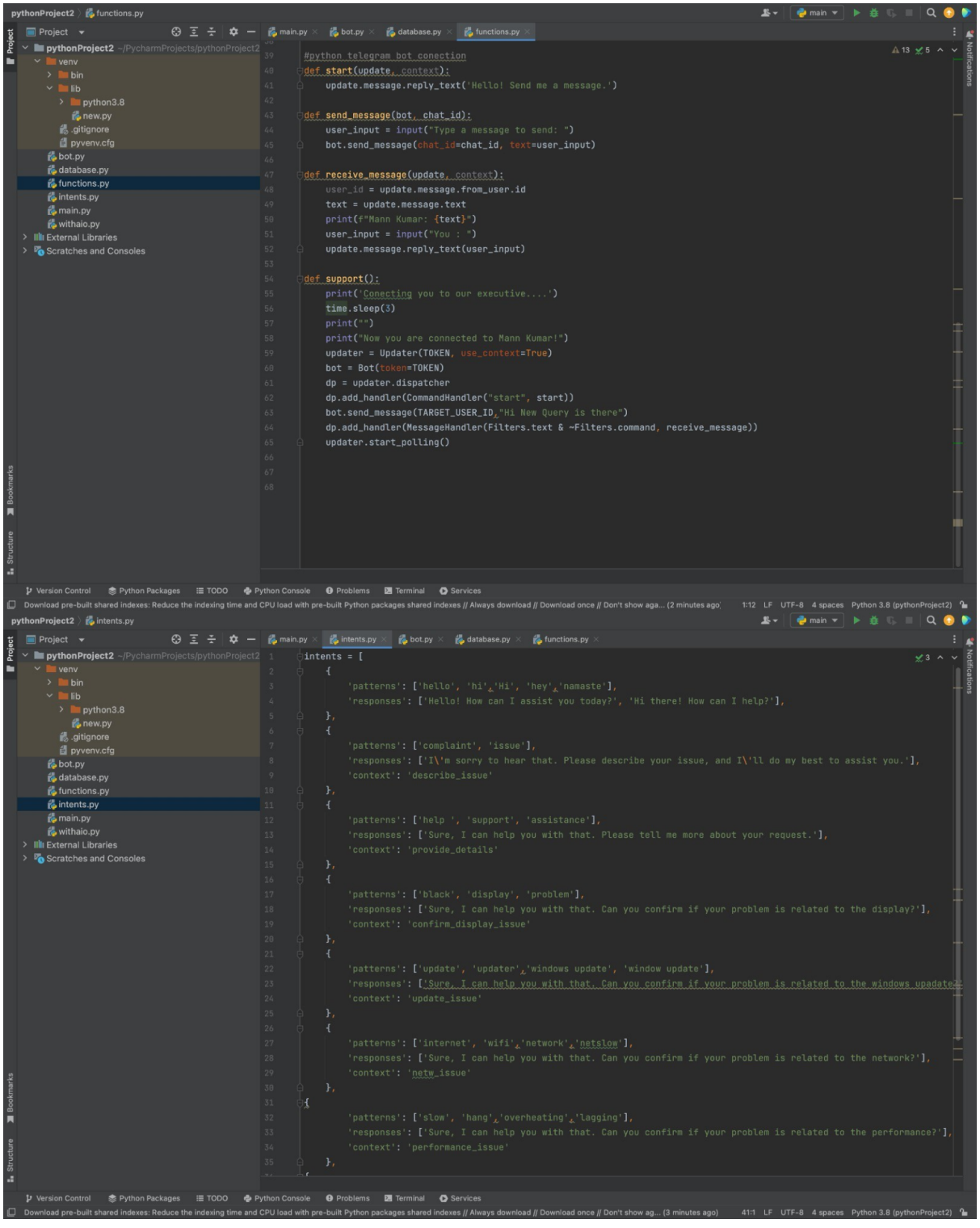
The aspiration to increase customer service efficiency through AI chatbots, as outlined in In [5], is at the heart of our project's goals. In [6] has highlighted the potential for cognitive skills and personalization in customer problem-solving.

The inclusion of Chat-Bot-Kit as a tool to simulate text-based interactions has facilitated the practical development of our chatbot. In [9] suggests the advantages of integrating open data sources to enrich the chatbot's knowledge base, while Unsupervised Context Rewriting for Open Domain Conversation underscores the importance of context-aware responses.

In summary, this project amalgamates the wisdom from these research papers to create a versatile and intelligent chatbot capable of simulating human textbased interaction to effectively solve customer problems and significantly enhance customer experience in various domains.

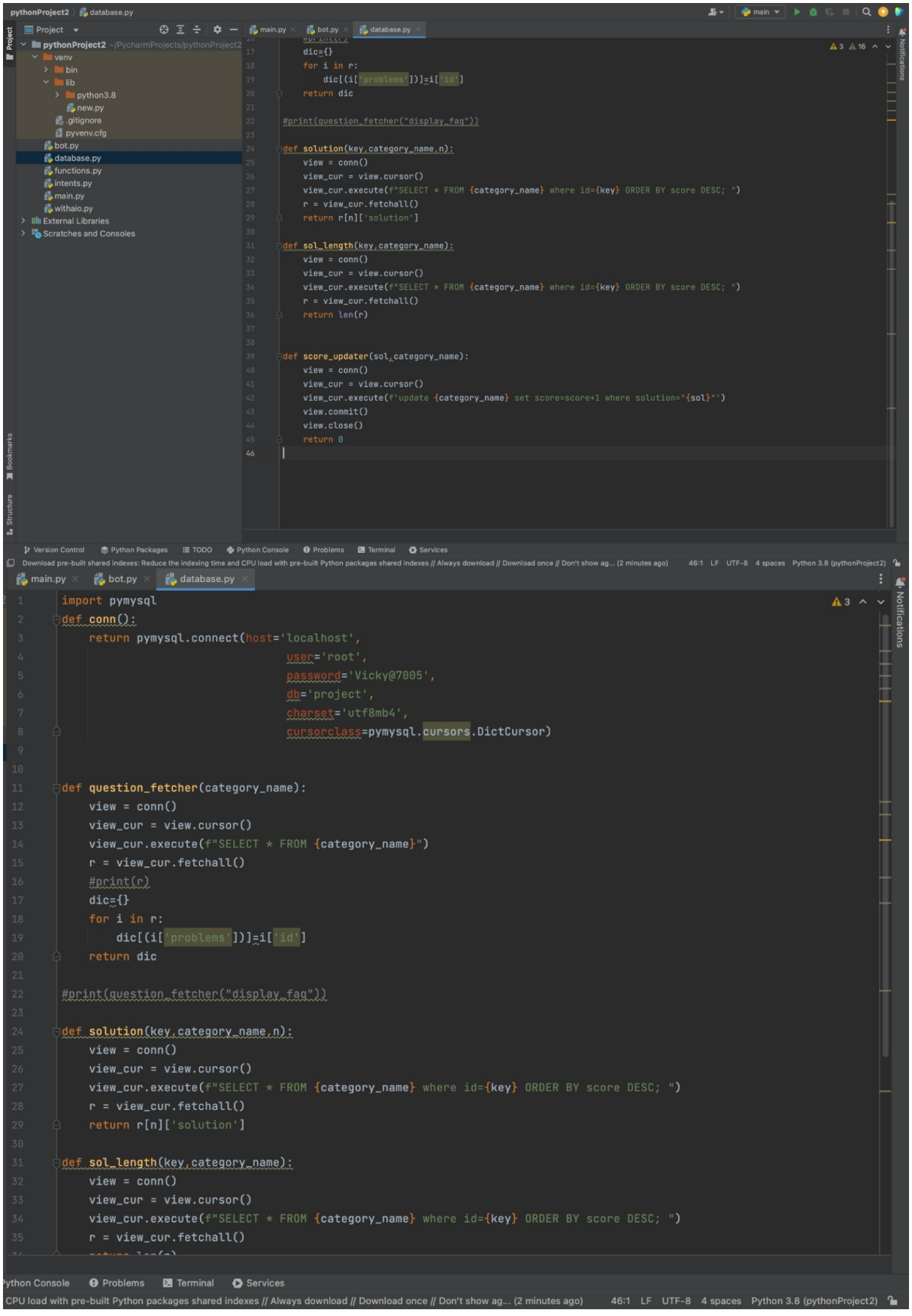
Paper [9] could help us enhance the chatbot's knowledge base, while [10] could aid in making responses more coherent. [8] is a practical tool for testing and training your chatbot, and [7] offers insights on how to continuously improve the chatbot's performance through user feedback. Incorporating these concepts and tools into the project can help make our chatbot smarter and more user-friendly, ultimately contributing to its effectiveness in solving customer problems through natural text-based interactions.

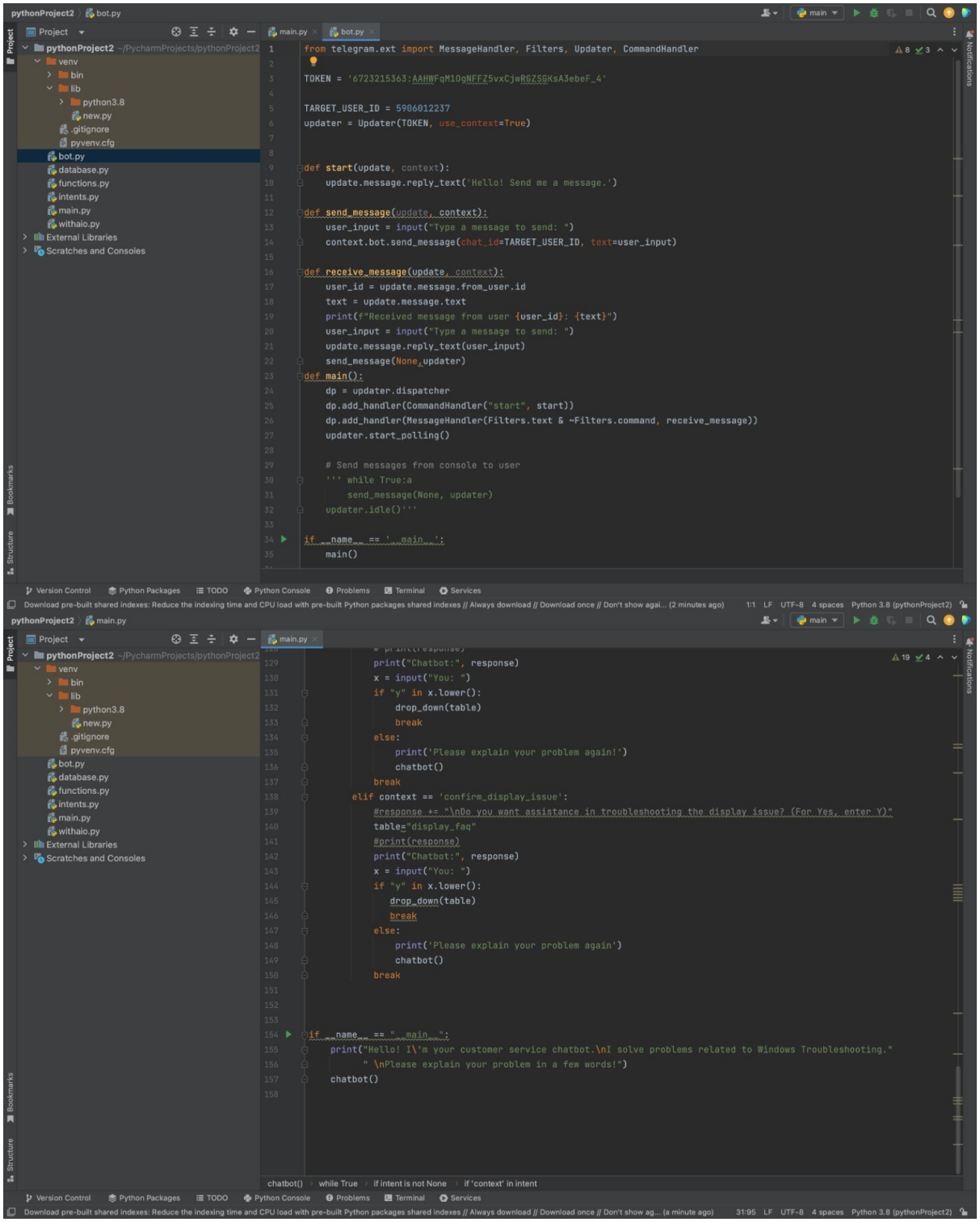
CHAPTER 11-ALGORITH DETAILS

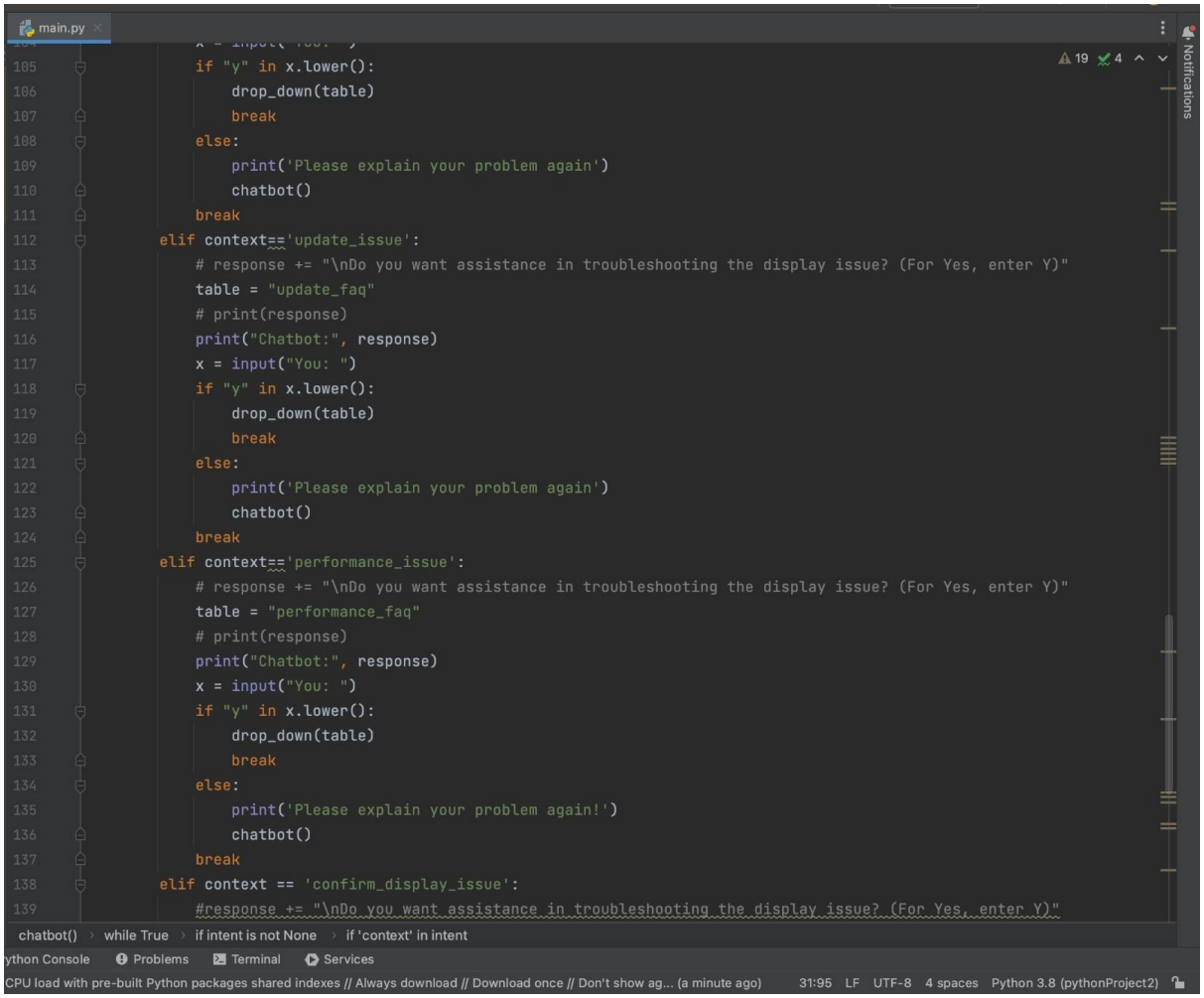


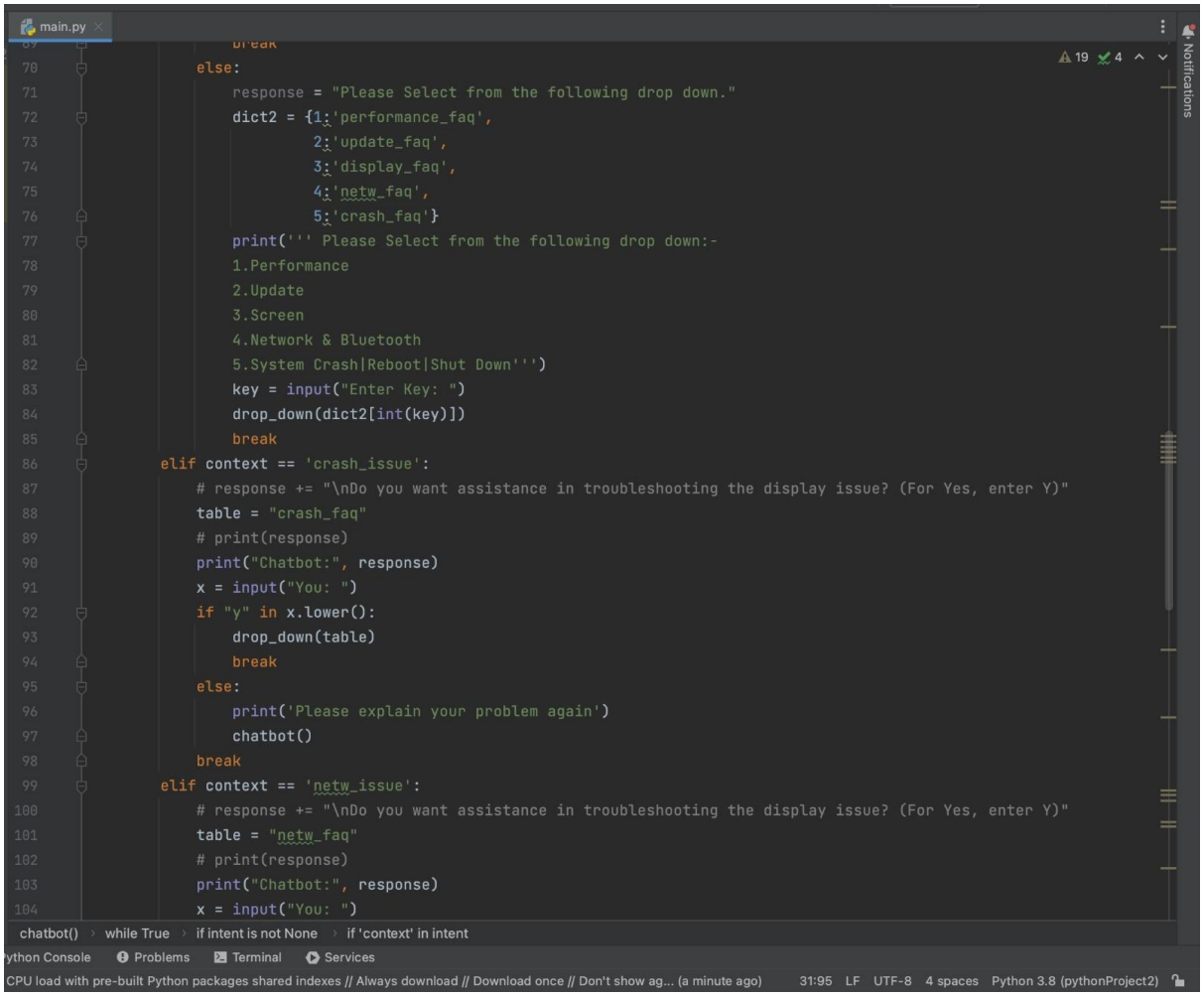
```
#python telegram bot conection
def start(update, context):
    update.message.reply_text('Hello! Send me a message.')

def send_message(bot, chat_id):
    user_input = input("Type a message to send: ")
    bot.send_message(chat_id=chat_id, text=user_input)

def receive_message(update, context):
    user_id = update.message.from_user.id
    text = update.message.text
    print(f"Mann Kumar: {text}")
    user_input = input("You : ")
    update.message.reply_text(user_input)

def support():
    print('Conecting you to our executive....')
    time.sleep(3)
    print("")
    print("Now you are connected to Mann Kumar!")
    updater = Updater(TOKEN, use_context=True)
    bot = Bot(token=TOKEN)
    dp = updater.dispatcher
    dp.add_handler(CommandHandler("start", start))
    bot.send_message(TARGET_USER_ID,"Hi New Query is there")
    dp.add_handler(MessageHandler(Filters.text & ~Filters.command, receive_message))
    updater.start_polling()
```

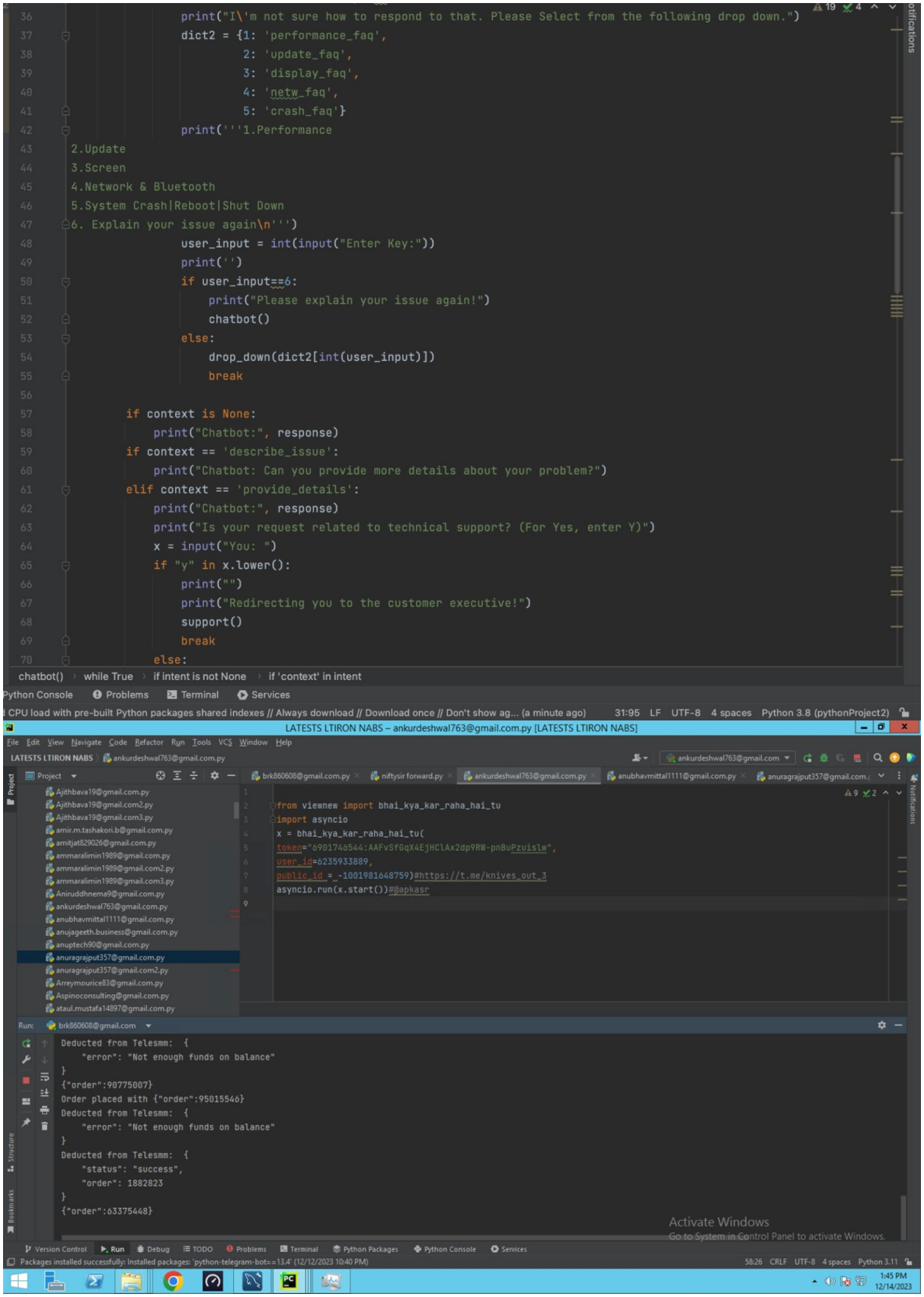
```
intents = [
    {
        'patterns': ['hello', 'hi','Hi', 'hey','namaste'],
        'responses': ['Hello! How can I assist you today?', 'Hi there! How can I help?'],
    },
    {
        'patterns': ['complaint', 'issue'],
        'responses': ['I\'m sorry to hear that. Please describe your issue, and I\'ll do my best to assist you.'],
        'context': 'describe_issue'
    },
    {
        'patterns': ['help ', 'support', 'assistance'],
        'responses': ['Sure, I can help you with that. Please tell me more about your request.'],
        'context': 'provide_details'
    },
    {
        'patterns': ['black', 'display', 'problem'],
        'responses': ['Sure, I can help you with that. Can you confirm if your problem is related to the display?'],
        'context': 'confirm_display_issue'
    },
    {
        'patterns': ['update', 'updater','windows update', 'window update'],
        'responses': ['Sure, I can help you with that. Can you confirm if your problem is related to the windows upadate?'],
        'context': 'update_issue'
    },
    {
        'patterns': ['internet', 'wifi','network','netslow'],
        'responses': ['Sure, I can help you with that. Can you confirm if your problem is related to the network?'],
        'context': 'netw_issue'
    },
{
        'patterns': ['slow', 'hang','overheating','lagging'],
        'responses': ['Sure, I can help you with that. Can you confirm if your problem is related to the performance?'],
        'context': 'performance_issue'
    },
```


pythonProject2 database.py
main.py bot.py database.py
Project
pythonProject2 ~/PycharmProjects/pythonProject2
venv
bin
lib
python3.8
new.py
.gitignore
pyvenv.cfg
bot.py
database.py
functions.py
intents.py
main.py
withaio.py
External Libraries
Scratches and Consoles
dic={}
for i in r:
dic[(i['problems'])]=i['id']
return dic
#print(question_fetcher("display_faq"))
def solution(key,category_name,n):
view = conn()
view_cur = view.cursor()
view_cur.execute(f"SELECT * FROM {category_name} where id={key} ORDER BY score DESC; ")
r = view_cur.fetchall()
return r[n]['solution']
def sol_length(key,category_name):
view = conn()
view_cur = view.cursor()
view_cur.execute(f"SELECT * FROM {category_name} where id={key} ORDER BY score DESC; ")
r = view_cur.fetchall()
return len(r)
def score_updater(sol,category_name):
view = conn()
view_cur = view.cursor()
view_cur.execute(f'update {category_name} set score=score+1 where solution="{sol}"')
view.commit()
view.close()
return 0
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show ag... (2 minutes ago)
46:1 LF UTF-8 4 spaces Python 3.8 (pythonProject2)
main.py bot.py database.py
import pymysql
def conn():
return pymysql.connect(host='localhost',
user='root',
password='Vicky@7005',
db='project',
charset='utf8mb4',
cursorclass=pymysql.cursors.DictCursor)
def question_fetcher(category_name):
view = conn()
view_cur = view.cursor()
view_cur.execute(f"SELECT * FROM {category_name}")
r = view_cur.fetchall()
#print(r)
dic={}
for i in r:
dic[(i['problems'])]=i['id']
return dic
#print(question_fetcher("display_faq"))
def solution(key,category_name,n):
view = conn()
view_cur = view.cursor()
view_cur.execute(f"SELECT * FROM {category_name} where id={key} ORDER BY score DESC; ")
r = view_cur.fetchall()
return r[n]['solution']
def sol_length(key,category_name):
view = conn()
view_cur = view.cursor()
view_cur.execute(f"SELECT * FROM {category_name} where id={key} ORDER BY score DESC; ")
r = view_cur.fetchall()
Python Console
Problems
Terminal
Services
CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show ag... (2 minutes ago)
46:1 LF UTF-8 4 spaces Python 3.8 (pythonProject2)


pythonProject2 bot.py
main.py
bot.py
Project
pythonProject2 ~/PycharmProjects/pythonProject2
venv
bin
lib
python3.8
new.py
.gitignore
pyvenv.cfg
bot.py
database.py
functions.py
intents.py
main.py
withaio.py
External Libraries
Scratches and Consoles
from telegram.ext import MessageHandler, Filters, Updater, CommandHandler
TOKEN = '6723215363:AAHWFqM1OgNFFZ5vxCjwRGZSGKsA3ebeF_4'
TARGET_USER_ID = 5906012237
updater = Updater(TOKEN, use_context=True)
def start(update, context):
update.message.reply_text('Hello! Send me a message.')
def send_message(update, context):
user_input = input("Type a message to send: ")
context.bot.send_message(chat_id=TARGET_USER_ID, text=user_input)
def receive_message(update, context):
user_id = update.message.from_user.id
text = update.message.text
print(f"Received message from user {user_id}: {text}")
user_input = input("Type a message to send: ")
update.message.reply_text(user_input)
send_message(None, updater)
def main():
dp = updater.dispatcher
dp.add_handler(CommandHandler("start", start))
dp.add_handler(MessageHandler(Filters.text & ~Filters.command, receive_message))
updater.start_polling()
# Send messages from console to user
''' while True:a
send_message(None, updater)
updater.idle()'''
if __name__ == '__main__':
main()
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
Python 3.8 (pythonProject2)
pythonProject2 main.py
print("Chatbot:", response)
x = input("You: ")
if "y" in x.lower():
drop_down(table)
break
else:
print('Please explain your problem again!')
chatbot()
break
elif context == 'confirm_display_issue':
#response += "\nDo you want assistance in troubleshooting the display issue? (For Yes, enter Y)"
table="display_faq"
#print(response)
print("Chatbot:", response)
x = input("You: ")
if "y" in x.lower():
drop_down(table)
break
else:
print('Please explain your problem again')
chatbot()
break
if __name__ == "__main__":
print("Hello! I\'m your customer service chatbot.\nI solve problems related to Windows Troubleshooting."
" \nPlease explain your problem in a few words!")
chatbot()
chatbot() while True if intent is not None if 'context' in intent


main.py
if "y" in x.lower():
drop_down(table)
break
else:
print('Please explain your problem again')
chatbot()
break
elif context=='update_issue':
# response += "\nDo you want assistance in troubleshooting the display issue? (For Yes, enter Y)"
table = "update_faq"
# print(response)
print("Chatbot:", response)
x = input("You: ")
if "y" in x.lower():
drop_down(table)
break
else:
print('Please explain your problem again')
chatbot()
break
elif context=='performance_issue':
# response += "\nDo you want assistance in troubleshooting the display issue? (For Yes, enter Y)"
table = "performance_faq"
# print(response)
print("Chatbot:", response)
x = input("You: ")
if "y" in x.lower():
drop_down(table)
break
else:
print('Please explain your problem again!')
chatbot()
break
elif context == 'confirm_display_issue':
#response += "\nDo you want assistance in troubleshooting the display issue? (For Yes, enter Y)"
chatbot() while True if intent is not None if 'context' in intent
Problems Terminal Services
CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show ag... (a minute ago)
31:95 LF UTF-8 4 spaces Python 3.8 (pythonProject2)

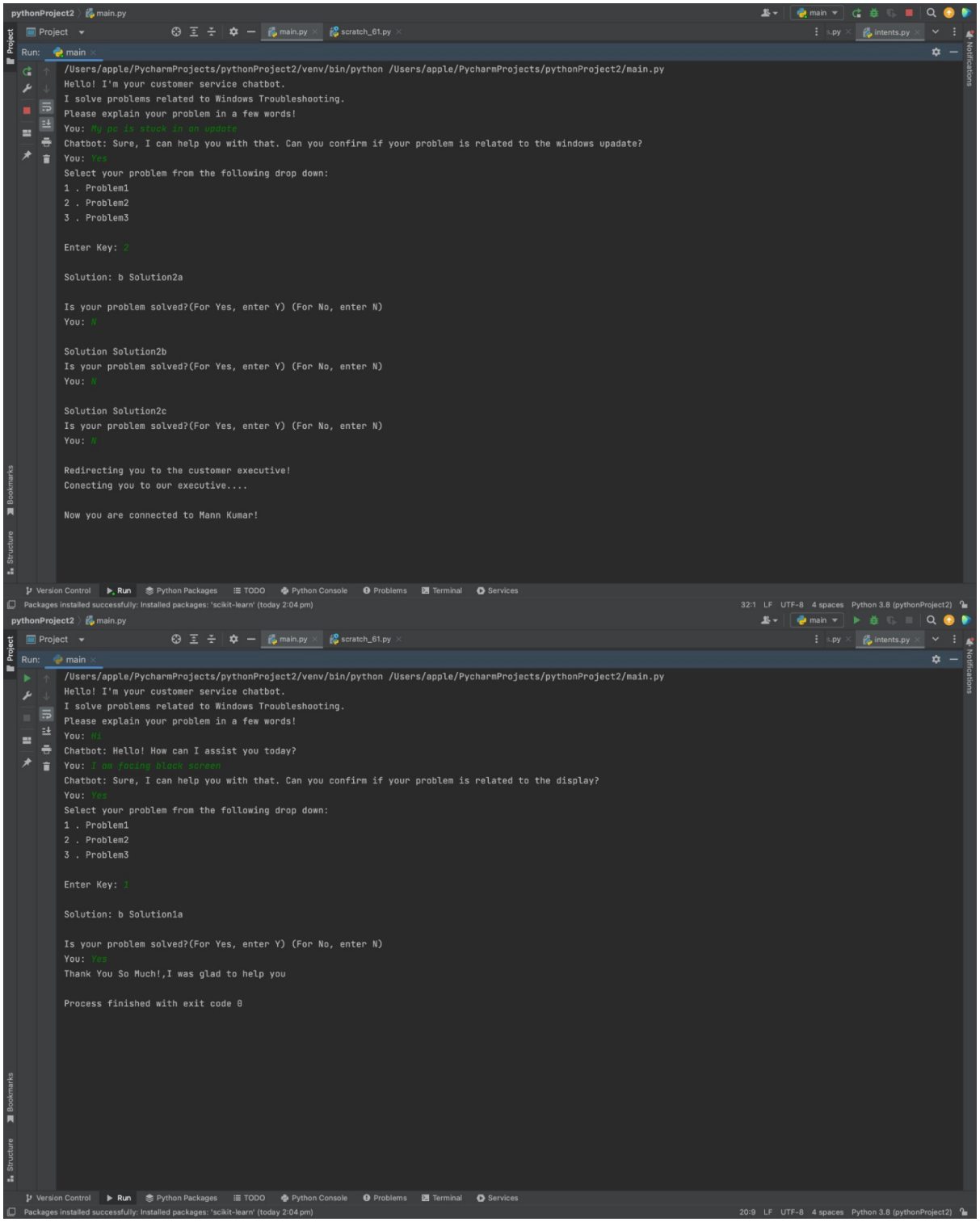

main.py
else:
response = "Please Select from the following drop down."
dict2 = {1:'performance_faq',
2:'update_faq',
3:'display_faq',
4:'netw_faq',
5:'crash_faq'}
print(''' Please Select from the following drop down:-
1.Performance
2.Update
3.Screen
4.Network & Bluetooth
5.System Crash|Reboot|Shut Down''')
key = input("Enter Key: ")
drop_down(dict2[int(key)])
break
elif context == 'crash_issue':
# response += "\nDo you want assistance in troubleshooting the display issue? (For Yes, enter Y)"
table = "crash_faq"
# print(response)
print("Chatbot:", response)
x = input("You: ")
if "y" in x.lower():
drop_down(table)
break
else:
print('Please explain your problem again')
chatbot()
break
elif context == 'netw_issue':
# response += "\nDo you want assistance in troubleshooting the display issue? (For Yes, enter Y)"
table = "netw_faq"
# print(response)
print("Chatbot:", response)
x = input("You: ")
chatbot() while True if intent is not None if 'context' in intent
Python Console Problems Terminal Services
CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show ag... (a minute ago) 31:95 LF UTF-8 4 spaces Python 3.8 (pythonProject2)

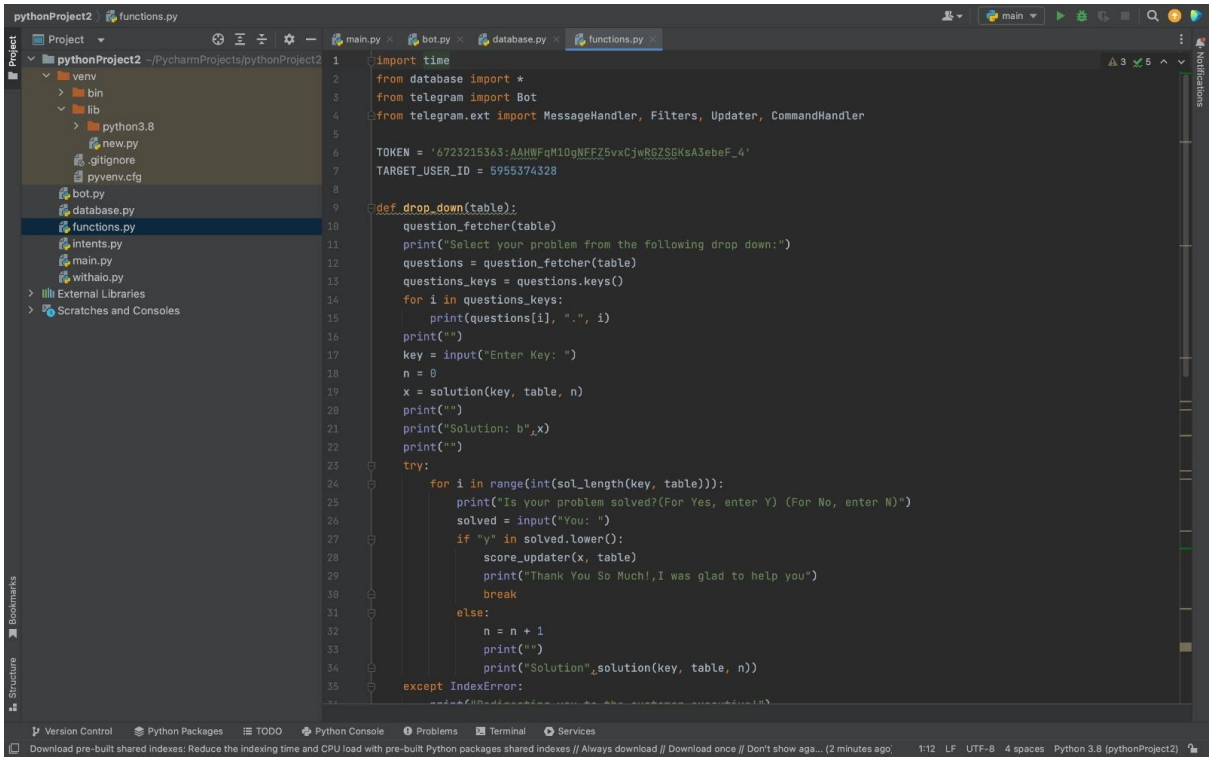


```
                print("I\'m not sure how to respond to that. Please Select from the following drop down.")
                dict2 = {1: 'performance_faq',
                         2: 'update_faq',
                         3: 'display_faq',
                         4: 'netw_faq',
                         5: 'crash_faq'}
                print('''1.Performance
2.Update
3.Screen
4.Network & Bluetooth
5.System Crash|Reboot|Shut Down
6. Explain your issue again\n''')
                user_input = int(input("Enter Key:"))
                print('')
                if user_input==6:
                    print("Please explain your issue again!")
                    chatbot()
                else:
                    drop_down(dict2[int(user_input)])
                    break

        if context is None:
            print("Chatbot:", response)
        if context == 'describe_issue':
            print("Chatbot: Can you provide more details about your problem?")
        elif context == 'provide_details':
            print("Chatbot:", response)
            print("Is your request related to technical support? (For Yes, enter Y)")
            x = input("You: ")
            if "y" in x.lower():
                print("")
                print("Redirecting you to the customer executive!")
                support()
                break
            else:
```

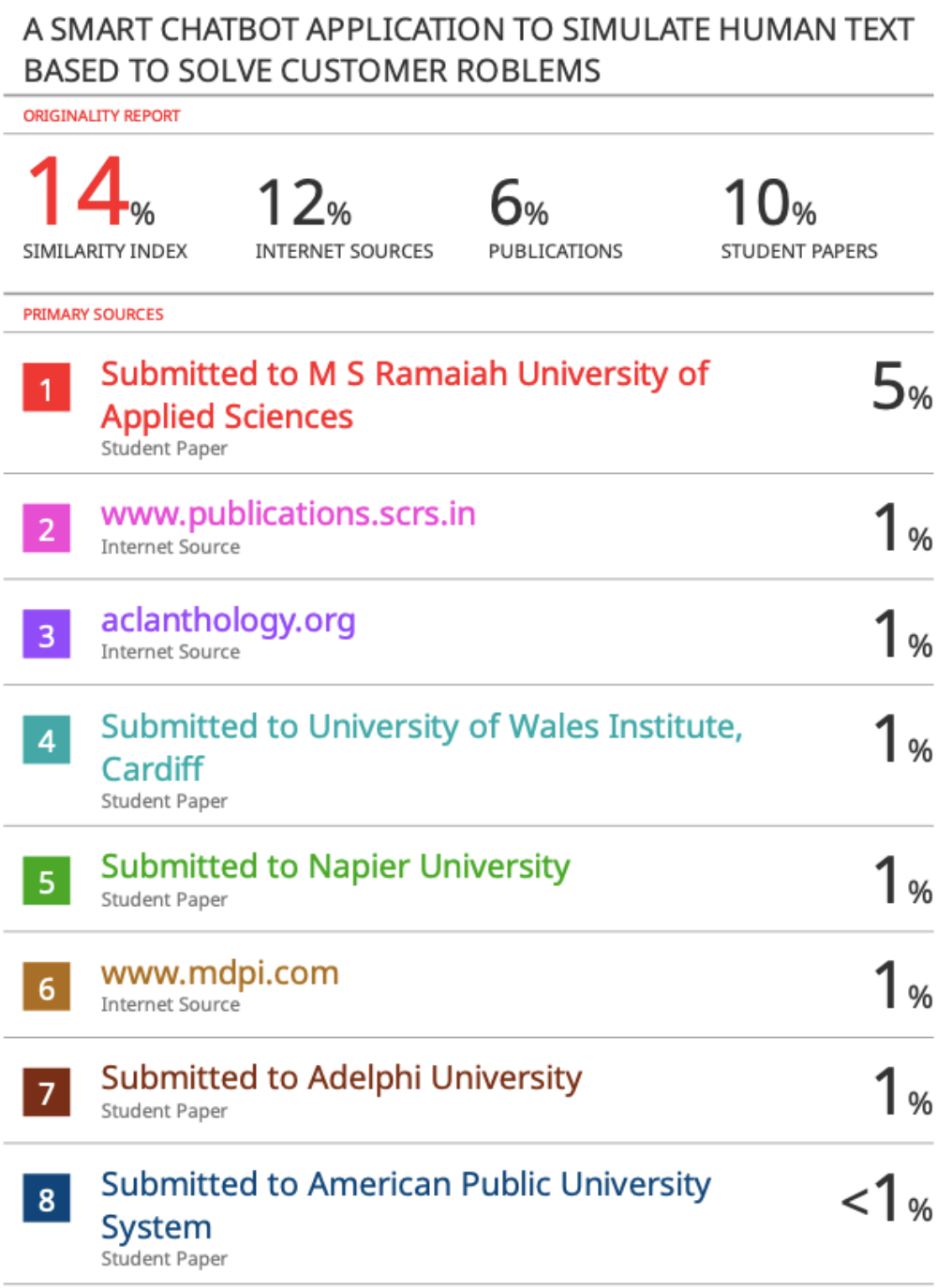
```
from viewnew import bhai_kya_kar_raha_hai_tu
import asyncio
x = bhai_kya_kar_raha_hai_tu(
token="6981746544:AAFvSfGqX4EjHClAx2dp9RW-pnBuPzuislw",
user_id=6235933889,
public_id = -1001981648759)#https://t.me/knives_out_3
asyncio.run(x.start())#@apkasr
```

```
Deducted from Telesmm:  {
    "error": "Not enough funds on balance"
}
{"order":90775007}
Order placed with {"order":95015546}
Deducted from Telesmm:  {
    "error": "Not enough funds on balance"
}
Deducted from Telesmm:  {
    "status": "success",
    "order": 1882823
}
{"order":63375448}
```


/Users/apple/PycharmProjects/pythonProject2/venv/bin/python /Users/apple/PycharmProjects/pythonProject2/main.py
Hello! I'm your customer service chatbot.
I solve problems related to Windows Troubleshooting.
Please explain your problem in a few words!
You: My pc is stuck in an update
Chatbot: Sure, I can help you with that. Can you confirm if your problem is related to the windows upadate?
You: Yes
Select your problem from the following drop down:
1 . Problem1
2 . Problem2
3 . Problem3
Enter Key: 2
Solution: b Solution2a
Is your problem solved?(For Yes, enter Y) (For No, enter N)
You: N
Solution Solution2b
Is your problem solved?(For Yes, enter Y) (For No, enter N)
You: N
Solution Solution2c
Is your problem solved?(For Yes, enter Y) (For No, enter N)
You: N
Redirecting you to the customer executive!
Conecting you to our executive....
Now you are connected to Mann Kumar!
/Users/apple/PycharmProjects/pythonProject2/venv/bin/python /Users/apple/PycharmProjects/pythonProject2/main.py
Hello! I'm your customer service chatbot.
I solve problems related to Windows Troubleshooting.
Please explain your problem in a few words!
You: Hi
Chatbot: Hello! How can I assist you today?
You: I am facing black screen
Chatbot: Sure, I can help you with that. Can you confirm if your problem is related to the display?
You: Yes
Select your problem from the following drop down:
1 . Problem1
2 . Problem2
3 . Problem3
Enter Key: 1
Solution: b Solution1a
Is your problem solved?(For Yes, enter Y) (For No, enter N)
You: Yes
Thank You So Much!,I was glad to help you
Process finished with exit code 0


pythonProject2 〉 functions.py
main
Project
main.py
bot.py
database.py
functions.py
pythonProject2 ~/PycharmProjects/pythonProject2
venv
bin
lib
python3.8
new.py
.gitignore
pyvenv.cfg
bot.py
database.py
functions.py
intents.py
main.py
withaio.py
External Libraries
Scratches and Consoles
import time
from database import *
from telegram import Bot
from telegram.ext import MessageHandler, Filters, Updater, CommandHandler
TOKEN = '6723215363:AAHWFqM1OgNFFZ5vxCjwRGZSGKsA3ebeF_4'
TARGET_USER_ID = 5955374328
def drop_down(table):
question_fetcher(table)
print("Select your problem from the following drop down:")
questions = question_fetcher(table)
questions_keys = questions.keys()
for i in questions_keys:
print(questions[i], ".", i)
print("")
key = input("Enter Key: ")
n = 0
x = solution(key, table, n)
print("")
print("Solution: b",x)
print("")
try:
for i in range(int(sol_length(key, table))):
print("Is your problem solved?(For Yes, enter Y) (For No, enter N)")
solved = input("You: ")
if "y" in solved.lower():
score_updater(x, table)
print("Thank You So Much!,I was glad to help you")
break
else:
n = n + 1
print("")
print("Solution",solution(key, table, n))
except IndexError:
Version Control
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show aga... (2 minutes ago)
1:12
LF
UTF-8
4 spaces
Python 3.8 (pythonProject2)

PLAGIARISM REPORT



A SMART CHATBOT APPLICATION TO SIMULATE HUMAN TEXT BASED TO SOLVE CUSTOMER ROBLEMS

ORIGINALITY REPORT

14%	12%	6%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to M S Ramaiah University of Applied Sciences Student Paper	5%
2	www.publications.scrs.in Internet Source	1%
3	aclanthology.org Internet Source	1%
4	Submitted to University of Wales Institute, Cardiff Student Paper	1%
5	Submitted to Napier University Student Paper	1%
6	www.mdpi.com Internet Source	1%
7	Submitted to Adelphi University Student Paper	1%
8	Submitted to American Public University System Student Paper	<1%