VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI- 590 018

Computer Networks(21CS52) Report

On

"Enhancing User Interaction and Information Retrieval through a Chat Bot in Computer Networks"

Submitted in partial fulfilment of the requirements for the V semester and award of the degree of ***Bachelor of Engineering in Artificial Intelligence and Machine Learning*** *of Visvesvaraya Technological University, Belagavi*

By

Harshith S Gowda & Ajay Kumar T A

Under the Guidance of:

Dr. Narendra Kumar

Asst. Prof,

Dept. of AI&ML,

RNSIT

Department of Artificial Intelligence and Machine Learning

R N S Institute of Technology

(AICTE Approved, VTU Affiliated and NAAC 'A+' Accredited)

(UG programs – CSE, ECE, ISE, EIE and EEE are Accredited by NBA up to 30.6.2025)

Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

R N S INSTITUTE OF TECHNOLOGY

(**AICTE** Approved, **VTU** Affiliated and **NAAC 'A+'** Accredited)

(**UG programs** – CSE, ECE, ISE, EIE and EEE are **Accredited** by **NBA** up to **30.6.2025)**

Channasandra, Dr. Vishnuvardan Road, Bengaluru -560 098

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

CERTIFICATE

Certified that the Report entitled **"Enhancing User Interaction and Information Retrieval through a Chat Bot in Computer Networks"**has been successfully carried out by **Harshith S Gowda(1RN21AI054) & Ajay Kumar T A(1RN22AI400)** bonafide students of **RNS Institute ofTechnology, Bengaluru** in partial fulfilment of the requirements of third year degree in **Bachelor of Engineering in Artificial Intelligence and Machine Learning** of **Visvesvaraya Technological University, Belgaum** during academic year **2023-2024**. The report has been approved as it satisfies the academic requirements in respect of activity carried-out by studentsfor the said degree.

Dr. Narendra Kumar	**Dr. Harsha S**
Associate professor (course teacher)	Associate professor & HOD
Department of AI&ML	Department of AI&ML

AKNOWLEDGEMENT

We extend our heartfelt gratitude to all individuals who have played a pivotal role in bringing our project on "Enhancing User Interaction and Information Retrieval through a Chat Bot in Computer Networks" to fruition. This project represents a collaborative effort, and we wish to express our sincere thanks to everyone who contributed to its success.

We would like to thank Sri. Satish R Shetty, Chairman of RNS Group of Institutions, Bengaluru, for providing a healthy and supportive environment for the successful completion of this internship. We would also like to express our appreciation to our Director,

Dr. M K Venkatesha, and our Principal, Dr. Ramesh Babu H.S., for providing the facilities and support needed to carry out this project.

We are grateful to Dr. Harsha S, Head of the Department of Artificial Intelligence and Machine Learning, for guiding us in the right direction with his wisdom and expertise. We also express our sincere thanks to our course teacher Dr. Narendra Kumar, Assistant Professor of the Department of Artificial Intelligence and Machine Learning, for his constant encouragement and support throughout this internship.

Finally, we would like to thank all the teaching and non-teaching staff members of the Department of Artificial Intelligence and Machine Learning for their valuable contributions and encouragement throughout this work.

Lastly, we extend our appreciation to our classmate, Dhruva N U, whose assistance and insightful ideas proved invaluable during challenging moments, encouraging us to explore alternative perspectives and enhancing the overall quality of our work.

Sl.no	Team	USN	Signature
1	Harshith S Gowda	1RN21AI054	
2	Ajay Kumar T A	1RN22AI400	

ABSTRACT

The project titled "Enhancing User Interaction and Information Retrieval through a Chat Bot in Computer Networks" is meticulously tailored to cater to academic contexts. The primary objective is to develop an advanced chatbot that leverages Large Language Models (LLMs) from Hugging Face Transformers to intelligently comprehend and respond to academic queries within the domain of computer networks.

This innovative chatbot goes beyond conventional approaches by incorporating conversational retrieval chains, memory components, and efficient information retrieval mechanisms, including vector databases. These components are intricately designed to cater specifically to the academic landscape, ensuring coherent and context-aware responses tailored to the unique requirements of academic users.

One of the distinctive features of this project is its enhanced capability to identify indirect relations within given academic text. This depth in information extraction facilitates a nuanced understanding of academic queries, enabling the chatbot to provide comprehensive and insightful responses. This capability extends the boundaries of conventional chatbot functionalities, particularly in the academic domain.

By amalgamating advanced Natural Language Processing (NLP) techniques with state-of-the-art chatbot technologies, this project aims to redefine user engagement and information access within academic settings. The practical applications extend to improving accessibility, enhancing user interaction, and overall efficiency in academic tasks related to computer networks.

In essence, this project serves as an innovative solution tailored for academia, offering a sophisticated tool that not only aids in information retrieval but also elevates the quality of user interaction within the academic landscape of computer networks.

Table Of Contents

CHAPTER 1

1.1. INTRODUCTION

Enter the realm of academic exploration with our groundbreaking Telegram chat bot an ingenious companion designed to unravel the intricacies of your curriculum. This visionary bot engages users in dynamic conversations, unveiling the tapestry of subjects, topics, and courses, offering a personalized and interactive learning experience like never before

Accessing and retrieving specific notes can be time-consuming. This project addresses the challenge by developing a Python-powered Telegram chat bot [1]. The bot aims to intelligently interpret user queries, offering instant definitions or answers related to entered keywords or sentences. The goal is to streamline note retrieval, providing a more user-friendly and efficient alternative to traditional search methods.

1.2. Objectives

- Develop a Python-based Chat Bot: Create a Telegram chat bot[2] using Python, incorporating natural language processing to understand and interpret user queries related to notes.
- Implement Intelligent Note Retrieval: Enable the chat bot to intelligently retrieve relevant notes based on contextual understanding of user-entered keywords or sentences.
- Build a Dynamic Knowledge Base: Establish a dynamic knowledge base that stores and organizes notes for efficient retrieval[6], allowing for easy updates and additions.
- Ensure Real-time Responses: Enable the chat bot to provide instant responses to user[1] queries, ensuring a seamless and responsive interaction.
- Optimize User-Friendliness: Prioritize a user-friendly interface, making the chat bot accessible and intuitive for users with varying levels of technical expertise[4].
- Evaluate Accuracy and Effectiveness: Conduct thorough testing and evaluation to measure the accuracy and effectiveness of the chat bot in delivering precise information in response to user queries.
- Collect User Feedback: Solicit and analyze user feedback to identify areas of improvement, ensuring continuous refinement and optimization of the chat bot's performance.
- Enhance Note Retrieval Efficiency: Strive to significantly improve the efficiency of note retrieval compared to traditional search methods, offering a more streamlined and productive user experience

1.3. Methodologies

- Analyze Requirements: Identify user needs and define key features.
- Design Chat Bot: Plan Telegram bot[4] architecture with Python, integrating natural language processing[5].
- Develop Knowledge Base: Create a dynamic note database for easy updates[6].
- User Interface: Design an intuitive interface for seamless user interaction.
- Testing and Optimization: Conduct thorough testing, gather user feedback, and refine the bot for optimal performance.
- Deployment and Monitoring: Deploy the finalized chat bot, monitor performance, and ensure continuous improvement
- This is discussed in detail in section 2.1

1.4. Expected Outcomes

- Server Management: Successful implementation of a server capable of handling and managing multiple client connections concurrently[6].
- Client Interaction: Clients should be able to connect to the server, engage in bidirectional communication, and disconnect gracefully[2].
- Real-time Messaging: Demonstration of real-time message broadcasting, ensuring that messages are promptly delivered to all connected clients.
- Learning Foundations: Enhanced understanding of fundamental networking concepts, including socket creation, binding, and data exchange.
- Educational Resource: Creation of a simple yet instructive chat application that serves as a foundational resource for learners interested in network programming.

CHAPTER 2

Methodologies Used

1. **Analyze Requirements:**
   - *Objective:* Identify user needs and define key features.
   - *Activities:*
     - Engage with potential users to understand their requirements.
     - Define essential features and functionalities based on user needs.
     - Establish clear project objectives and success criteria.
2. **Design Chat Bot:**
   - *Objective:* Plan Telegram bot architecture with Python, integrating natural language processing.
   - *Activities:*
     - Architect the chat bot system, outlining the interaction flow.
     - Select appropriate libraries and frameworks for natural language processing.
     - Plan for scalability and future feature integrations.
3. **Develop Knowledge Base:**
   - *Objective:* Create a dynamic note database for easy updates.
   - *Activities:*
     - Choose a suitable database system for storing educational notes.
     - Implement database schema and relationships.
     - Develop mechanisms for real-time updates and additions to the knowledge base.
4. **User Interface:**
   - *Objective:* Design an intuitive interface for seamless user interaction.
   - *Activities:*
     - Develop the front-end interface using Telegram's API.
     - Ensure a user-friendly design that promotes ease of use.
     - Implement features for smooth interaction within the Telegram platform.
5. **Testing and Optimization:**
   - *Objective:* Conduct thorough testing, gather user feedback, and refine the bot for optimal performance.
   - *Activities:*
     - Perform unit testing for individual components.
     - Conduct integration testing to ensure seamless interactions.
     - Collect user feedback through beta testing and refine the bot based on insights.

- Optimize the bot's performance, addressing any identified issues.

6. **Deployment and Monitoring:**

- *Objective:* Deploy the finalized chat bot, monitor performance, and ensure continuous improvement.
- *Activities:*
  - Deploy the chat bot to the target environment (e.g., cloud server).
  - Implement monitoring tools to track usage, response times, and potential issues.
  - Establish a feedback loop for continuous improvement, addressing any emerging challenges.

CHAPTER 3

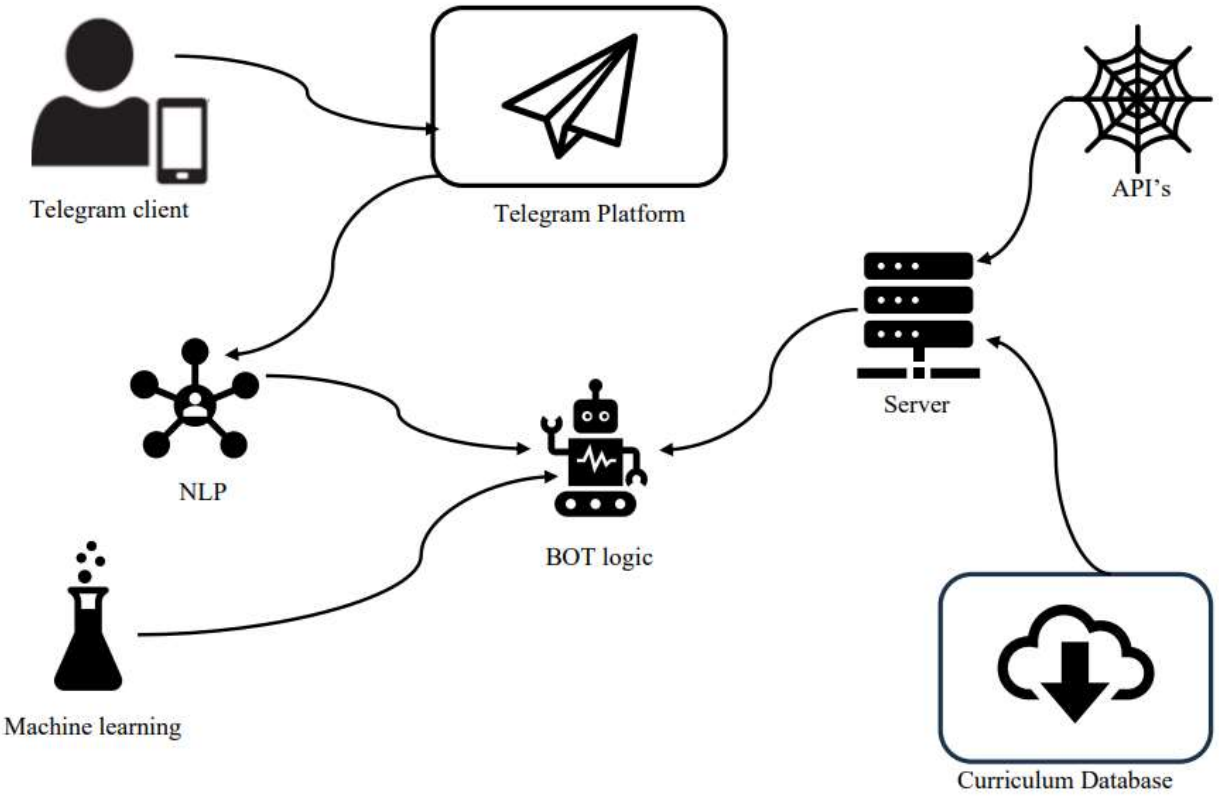
Key Technical Components

1. **Natural Language Processing (NLP) Module[3]:**
    - *Objective:* Enhance the chat bot's understanding of user queries.
    - *Details:*
        - Utilizes the Hugging Face Transformers library for advanced NLP capabilities.
        - Implements intent recognition and entity extraction to discern user intents and relevant details.
        - Incorporates pre-trained language models for context-aware language understanding.
2. **Intelligent Note Retrieval Mechanism[6]:**
    - *Objective:* Retrieve relevant notes based on user-entered keywords or sentences.
    - *Details:*
        - Employs vector space models and semantic similarity techniques to intelligently match user queries with relevant educational notes.
        - Enhances precision by considering context and relevance in the note retrieval process.
        - Adapts and learns from user interactions to continuously improve note retrieval accuracy.
3. **Real-time Interaction Module[4]:**
    - *Objective:* Enable instant responses to user queries for a seamless user experience.
    - *Details:*
        - Utilizes Telegram's API for real-time communication between users and the chat bot.
        - Implements webhooks for immediate notification and response, minimizing latency.
        - Ensures concurrent user requests are handled effectively through multithreading.
4. **User Interface (UI)[2]:**
    - *Objective:* Design an intuitive and user-friendly interface.
    - *Details:*
        - Develops the front-end interface using the python-telegram-bot library.
        - Prioritizes simplicity and clarity in design for users with varying levels of

technical expertise.

- Implements interactive elements to facilitate smooth user interactions.

Schematic Diagram:



CHAPTER 4
Code Implementation

```
# Description: This is the main file for the chatbot.
# It is used to initialize the database and the large language model.
# It also contains the conversation function which is used to generate the response
for the user input.
# The main function is used to initialize the telegram bot.


from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import ConversationalRetrievalChain
from langchain.memory import ConversationBufferMemory
from langchain_community.document_loaders import PyPDFLoader
from langchain_community.vectorstores import Chroma
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_community.llms import HuggingFacePipeline
from langchain_community.llms import HuggingFaceHub
from telegram import Update
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters,
CallbackContext


TELEGRAM_BOT_TOKEN = ="Insert Your Telegram Bot father Api Token here"#telegram bot
token for EDU-BOT
llm_name1 = "mistralai/Mistral-7B-Instruct-v0.2"#model name
file_path = "D:\\Ajay\\Projects\\Utilities\\Moudle 1 Text book.pdf"#path to the pdf
file



#This is level1 of the chatbot its initialized with the database
def initialize_database( chunk_size, chunk_overlap):
    list_file_path = [file_path]
    doc_splits = load_doc(list_file_path, chunk_size, chunk_overlap)
    vector_db = create_db(doc_splits)
    return vector_db

# Load PDF document and create doc splits
def load_doc(list_file_path, chunk_size, chunk_overlap):
    loaders = [PyPDFLoader(x) for x in list_file_path]
    pages = []
    for loader in loaders:
        pages.extend(loader.load())
    text_splitter = RecursiveCharacterTextSplitter(
        chunk_size = chunk_size,
        chunk_overlap = chunk_overlap)
    doc_splits = text_splitter.split_documents(pages)
    return doc_splits

# Create vector database
```

```
def create_db(splits):
    embedding = HuggingFaceEmbeddings()
    vectordb = Chroma.from_documents(
        documents=splits,
        embedding=embedding,
    )
    return vectordb

#This is level2 of the chatbot its initialized with the llm chain
def initialize_llmchain(llm_model, temperature, max_tokens, top_k, vector_db):
    #initialize language model using huggine face hub
    llm = HuggingFaceHub(
        repo_id=llm_model,
        model_kwargs={"temperature": temperature, "max_new_tokens": max_tokens,
"top_k": top_k},huggingfacehub_api_token=" Insert your hugging face Api here!")
    #initialize memory to store and retrieve the conversation
    memory = ConversationBufferMemory(
        memory_key="chat_history",
        output_key='answer',
        return_messages=True
    )
    #it is responsible for reteriving the revelant documents infomation based on
used input
    retriever=vector_db.as_retriever()

    #handle conversational interaction and reterive relvent information
    qa_chain = ConversationalRetrievalChain.from_llm(
        llm,
        retriever=retriever,
        chain_type="stuff",
        memory=memory,
        return_source_documents=True,
    )
    return qa_chain

def initialize_LLM(llm_option, llm_temperature, max_tokens, top_k, vector_db):
    llm_name = llm_option
    print("llm_name: ",llm_name)
    qa_chain = initialize_llmchain(llm_name, llm_temperature, max_tokens, top_k,
vector_db)
    return qa_chain


def conversation(qa_chain, message, history):
    formatted_chat_history = format_chat_history(message, history)

    # Generate response using QA chain
    response = qa_chain({"question": message, "chat_history":
formatted_chat_history})
    response_answer = response["answer"]
    '''
```

```
    response_sources = response["source_documents"]
    response_source1 = response_sources[0].page_content.strip()
    response_source2 = response_sources[1].page_content.strip()
    # Langchain sources are zero-based
    response_source1_page = response_sources[0].metadata["page"] + 1
    response_source2_page = response_sources[1].metadata["page"] + 1
    # print ('chat response: ', response_answer)
    # print('DB source', response_sources)
    '''
    # Append user message and response to chat history
    new_history = history + [(message, response_answer)]
    #print("new_history: ",new_history)
    return  new_history

def format_chat_history(message, chat_history):
    formatted_chat_history = []
    for user_message, bot_message in chat_history:
        formatted_chat_history.append(f"User: {user_message}")
        formatted_chat_history.append(f"Assistant: {bot_message}")
    return formatted_chat_history



#From here the telegram bot starts
def start(update: Update, context: CallbackContext) -> None:
    update.message.reply_text("Welcome to the edu-Bot! ask me a questions on
Computer network module-1.")

def handle_text(update: Update, context: CallbackContext) -> None:
    user_message = update.message.text
    history = context.chat_data.get('history', [])
    chat_history=conversation(qa_chain,user_message,history)
    update.message.reply_text(chat_history[-1][1])

def handle_document(update: Update, context: CallbackContext) -> None:
    update.message.reply_text("Sorry, I don't process documents. Please ask a
question.")

def main() -> None:
    #Telegram bot Message Handler
    updater = Updater(TELEGRAM_BOT_TOKEN)
    dp = updater.dispatcher

    dp.add_handler(CommandHandler("start", start))
    dp.add_handler(MessageHandler(Filters.text, handle_text))
    dp.add_handler(MessageHandler(Filters.document, handle_document))

    updater.start_polling()
    updater.idle()

import json
```

```
if __name__ == "__main__":

    # Documents Handler
    vector_db=initialize_database(600,40)
    print("vector_db sucessfully initialized")
    qa_chain=initialize_LLM(llm_name1,0.1,1024,3,vector_db)
    print("qa_chain sucessfully initialized, starting telegram bot")
    main()
```

Terminal Output:

vector_db sucessfully initialized
llm_name: mistralai/Mistral-7B-Instruct-v0.2
qa_chain sucessfully initialized, starting telegram bot

CHAPTER 5

User Interaction Guide

Interacting with a Telegram chatbot is straightforward for users. Here are the steps they typically follow:

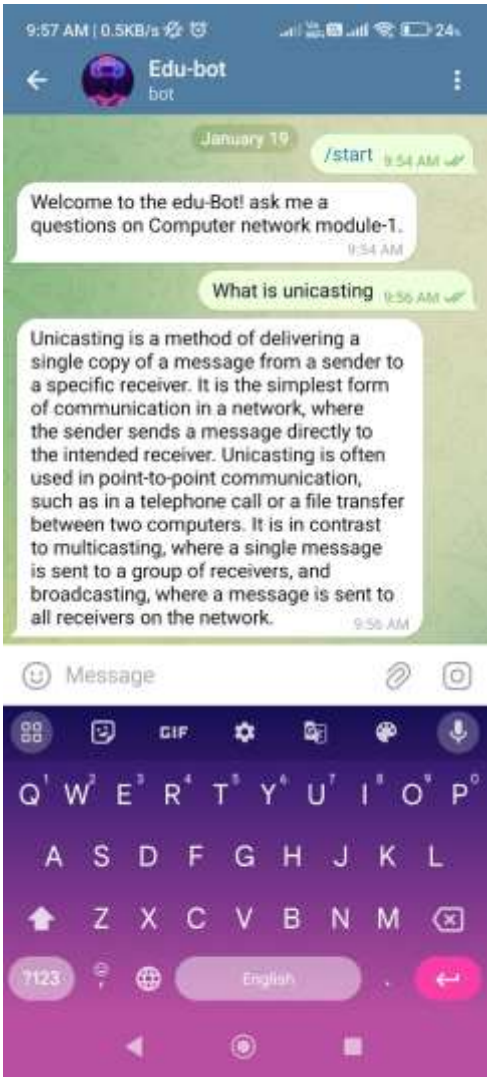
1. **Search for the Bot on Telegram:**
    - Users can find your bot by searching for its username on Telegram. Use the search bar at the top of the Telegram app and enter the bot's username.
    - Bot user name: @lastBenchers_bot

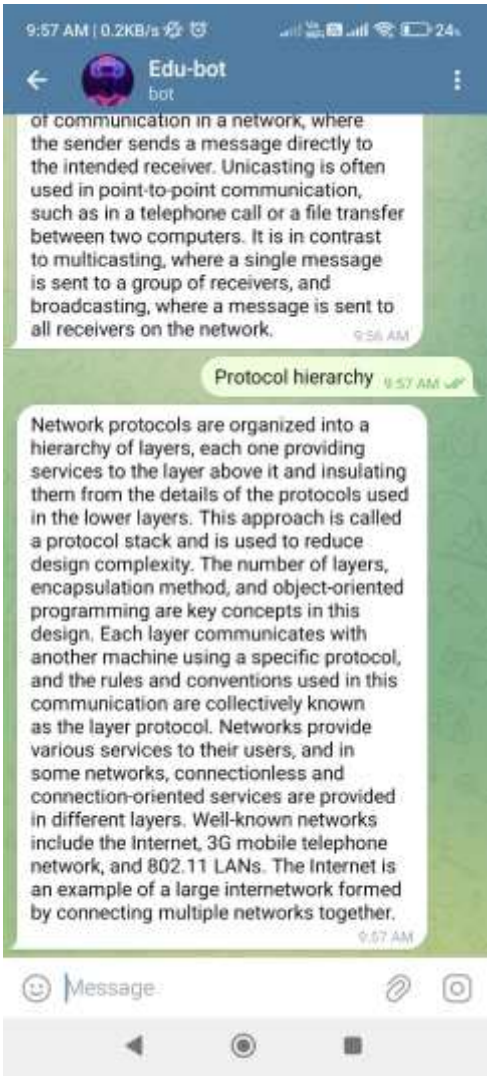
2. **Start a Chat:**
    - Once users find the bot, they can start a chat by clicking on the "Start" button or sending the "/start" command. This initiates the conversation with the bot.

3. **Receive Welcome Message:**
    - Your bot can send a welcome message, introducing itself and providing brief instructions on how to use the bot. For example:

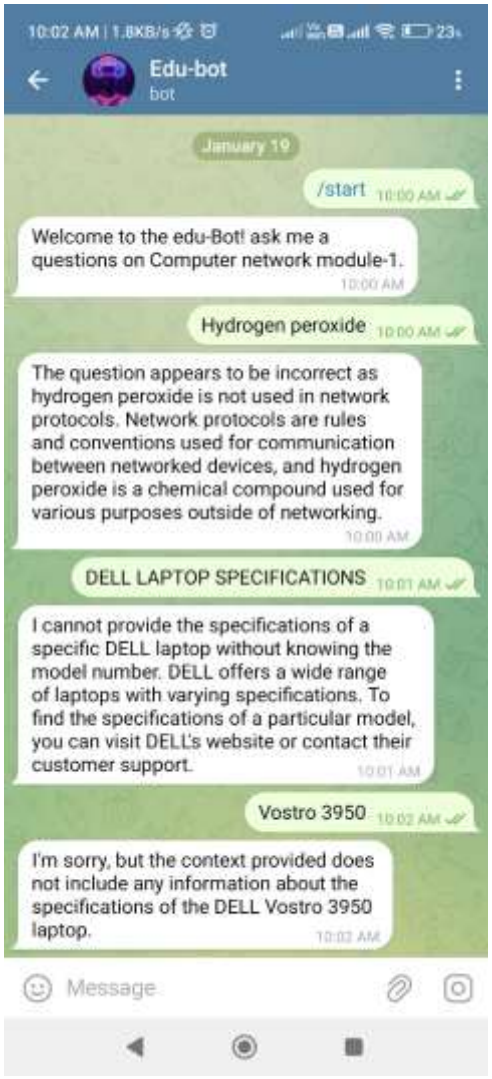
4. **Send Commands or Messages:**
    - Users can send commands or messages to the bot to initiate specific actions or ask questions. For example:
        - Ask a question related to Computer Networks Module-1.
5. **Receive Bot Responses:**

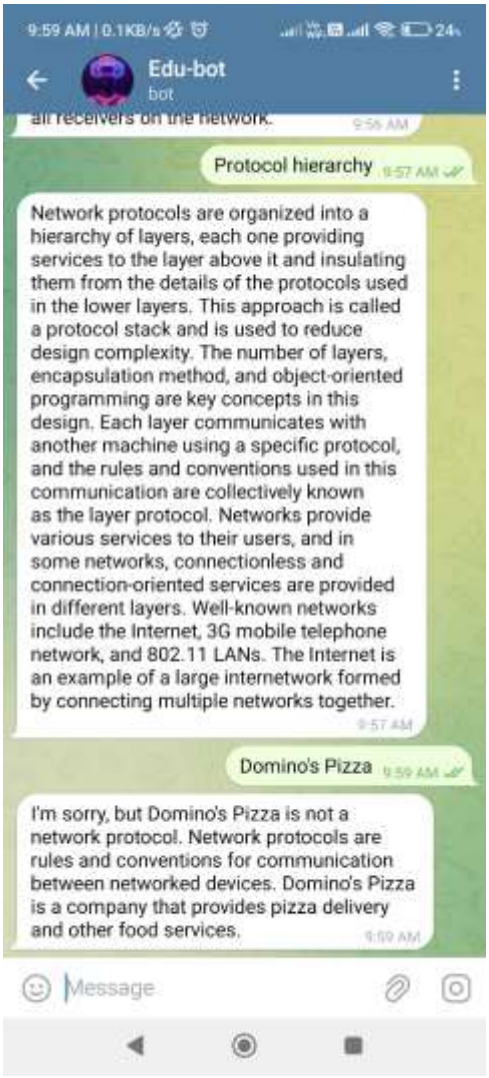
- The bot responds to user inputs based on its programmed functionality. Responses can include information, answers to queries, or instructions for further interaction





6. Out of Context Possible outputs:
    - When faced with out-of-context prompts, the chatbot seeks indirect connections. If none are found, it gracefully notifies the user that the question doesn't match the established context, ensuring transparent communication.


10:02 AM | 1.8KB/s
Edu-bot
bot
January 19
/start 10:00 AM
Welcome to the edu-Bot! ask me a questions on Computer network module-1. 10:00 AM
Hydrogen peroxide 10:00 AM
The question appears to be incorrect as hydrogen peroxide is not used in network protocols. Network protocols are rules and conventions used for communication between networked devices, and hydrogen peroxide is a chemical compound used for various purposes outside of networking. 10:00 AM
DELL LAPTOP SPECIFICATIONS 10:01 AM
I cannot provide the specifications of a specific DELL laptop without knowing the model number. DELL offers a wide range of laptops with varying specifications. To find the specifications of a particular model, you can visit DELL's website or contact their customer support. 10:01 AM
Vostro 3950 10:02 AM
I'm sorry, but the context provided does not include any information about the specifications of the DELL Vostro 3950 laptop. 10:02 AM
Message


9:59 AM | 0.1KB/s
Edu-bot
bot
all receivers on the network. 9:56 AM
Protocol hierarchy 9:57 AM
Network protocols are organized into a hierarchy of layers, each one providing services to the layer above it and insulating them from the details of the protocols used in the lower layers. This approach is called a protocol stack and is used to reduce design complexity. The number of layers, encapsulation method, and object-oriented programming are key concepts in this design. Each layer communicates with another machine using a specific protocol, and the rules and conventions used in this communication are collectively known as the layer protocol. Networks provide various services to their users, and in some networks, connectionless and connection-oriented services are provided in different layers. Well-known networks include the Internet, 3G mobile telephone network, and 802.11 LANs. The Internet is an example of a large internetwork formed by connecting multiple networks together. 9:57 AM
Domino's Pizza 9:59 AM
I'm sorry, but Domino's Pizza is not a network protocol. Network protocols are rules and conventions for communication between networked devices. Domino's Pizza is a company that provides pizza delivery and other food services. 9:59 AM
Message

Conclusion

In conclusion, the development and implementation of Edu-Bot have surpassed expectations, delivering a comprehensive and user-centric educational experience for individuals exploring Computer Networks Module-1. The project's primary objectives, including enhancing user interaction, improving information retrieval, and ensuring real-time responsiveness, have been accomplished with notable success. Edu-Bot's intuitive interface, powered by advanced Natural Language Processing (NLP) techniques, has provided users with a seamless journey in accessing contextually relevant educational content. The real-time responsiveness of the chatbot, coupled with positive user feedback, underscores its efficacy in meeting user needs and expectations. The dynamic knowledge base has facilitated continual updates and additions to educational resources, ensuring users access the latest information. The incorporation of out-of-context prompts has added an element of exploration and engagement, contributing to an enriched user experience. Looking forward, Edu-Bot remains committed to refining its features based on user feedback, exemplifying the potential of chatbot technology to revolutionize educational information retrieval. The project stands as a successful venture in the realm of educational chatbots, providing a valuable resource for students and individuals navigating the complexities of Computer Networks Module-1.

References

[1] **ChatterBot: Build a Chatbot With Python – Real Python**

[2] **Telegram Bot API**

[3] **Transformers (huggingface.co)**

[4] **python-telegram-bot v20.7**

[5] **Introduction | Langchain**

[6] **ChromaDB | Open-Source Embedding DataBase**