VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belagavi – 590 018

A

Mini Project Report

On

"AI BOT"

Submitted in partial fulfillment of Bachelor of Engineering Degree

In

COMPUTER SCIENCE AND ENGINEERING

18CSMP68- Mobile Application Development Laboratory

Submitted by:

MOHAMMED FAIZ	**1HK20CS084**
MOHAMMED MUSTAFA C	**1HK20CS090**

Under the guidance of

Prof. Najmusher H

Assistant Professor, Department of Computer Science & Engineering

JULY 2023

Department of Computer Science and Engineering

HKBK COLLEGE *of* ENGINEERING

(Approved by AICTE & Affiliated to VTU)

No.22/1,Opp., Manyata Tech Park Rd, Nagavara, Bengaluru, Karnataka 560045

Email: info@hkbk.edu.in URL: www.hkbk.edu.in

HKBK COLLEGE *of* ENGINEERING

Nagawara, Bangalore–560 045

Approved by AICTE & Affiliated to VTU

Department of Computer Science and Engineering

Certificate

Certified that the Mini Project Work entitled **"AI BOT",** carried out by **Mohammed Mustafa C (1HK20CS090)** **and** **Mohammed Faiz (1HK20CS084)** are bonafide students of **HKBK COLLEGE *of* ENGINEERING**, in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University**, Belgaum, during the year **2022–23**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of **18CSMP68- Mobile Application Development Laboratory** prescribed for the said Degree.

Prof. Najmusher H	**Dr. Smitha Kurian**	**Dr. Tabassum Ara**
Guide	HOD - CSE	Principal – HKBKCE

External Viva

Name of the Examiners	Signature with Date
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

We would like to express our regards and acknowledgement to all who helped us in completing this mini project successfully.

First of all we would take this opportunity to express our heartfelt gratitude to the personalities, **Mr. C M Ibrahim**, Chairman, HKBKGI and **Mr. C M Faiz Mohammed**, Director, HKBKGI for providing facilities throughout the course.

We express our sincere gratitude to **Dr. Tabassum Ara**, Principal, HKBKCE for her support towards the attainment of knowledge.

We consider it as a great privilege to convey our sincere regards to **Dr. Smitha Kurian.**, Associate Professor and HOD, Department of CSE, HKBKCE, for her constant encouragement throughout the course of the project.

We would specially like to thank our guide **Prof. Najmusher H,** Assistant Professor, Department of CSE, HKBKCE for her vigilant supervision and her constant encouragement. She spent her precious time in reviewing the project work and provided many insightful comments and constructive criticism.

Finally, we thank Almighty, all the staff members of CSE Department, our family members and friends for their constant support and encouragement in carrying out the project work.

Mohammed Mustafa C (1HK20CS185)
Mohammed Faiz (1HK20CS084)

ABSTRACT

The AI chatbot developed using the BrainShop API as a reference. The goal of this project is to create a conversational agent capable of engaging in natural and meaningful conversations with users. The AI chatbot leverages the BrainShop API, which provides a robust platform for implementing intelligent conversational systems.The chatbot's architecture and design are based on the BrainShop API, which offers a wide range of conversational abilities and pre-existing knowledge. By integrating this API, the chatbot gains access to a wealth of information and functionalities to enhance its conversational capabilities.The AI chatbot employs techniques such as natural language processing, machine learning, and deep learning to understand and respond to user queries and statements. It utilizes advanced algorithms to generate coherent and contextually relevant responses.The evaluation of the AI chatbot involves both objective and subjective measures. Objective metrics include response accuracy, coherence, and relevancy, while subjective evaluations involve user feedback and comparisons with existing conversational agents.Preliminary results demonstrate the effectiveness of the AI chatbot built using the BrainShop API. It showcases the ability to engage in meaningful conversations, provide accurate information, and offer valuable assistance to users.Future work will focus on refining and expanding the chatbot's capabilities, incorporating additional features, and optimizing its performance. By harnessing the power of the BrainShop API, we aim to create an AI chatbot that delivers enhanced conversational experiences and caters to the diverse needs of users.

TABLE OF CONTENTS

LIST OF FIGURES:

CHAPTER 1:

INTRODUCTION

1.1 Overview:

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014.The first stable build was released in December 2014, starting from version 1.0.

On May 7, 2019, Kotlin replaced Java as Google's preferred language for Android app development.[13] Java is still supported, as is C++.

A specific feature of the Android Studio is an absence of the possibility to switch autosave feature off. The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version. “External projects backport some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

1.2 Statement of problem

The problem addressed in this research is the need for an advanced AI chatbot that can effectively engage in natural and human-like conversations with users. Existing chatbot systems often struggle to understand and respond appropriately to user queries and statements, leading to inaccurate or irrelevant responses. Additionally, many chatbots lack the ability to adapt to specific domains or refine their conversational abilities over time. Therefore, there is a demand for an AI chatbot that leverages cutting-edge technologies, such as the BrainShop API, to overcome these limitations and provide users with a highly interactive and personalized conversational experience. The objective is to develop an AI chatbot that can understand user input, generate coherent and contextually relevant responses, and continuously improve its conversational abilities through techniques like fine-tuning, transfer learning, and reinforcement learning.

1.3 Objective of the problem

The objective of this research is to develop an AI chatbot using the BrainShop API that excels in engaging users through natural and human-like conversations. The primary objective is to enhance the chatbot's conversational abilities, ensuring it accurately understands user queries and statements and generates coherent and contextually relevant responses. Additionally, the chatbot aims to personalize the conversation and adapt to specific domains or user preferences, continuously refining its responses based on user interactions. By leveraging the BrainShop API as a reference, the chatbot aims to utilize its extensive conversational abilities and pre-existing knowledge to provide valuable information to users. Furthermore, the chatbot strives for continuous improvement through techniques such as fine-tuning, transfer learning, and reinforcement learning, enabling it to learn from user feedback and enhance its conversational abilities over time. Ultimately, the objective is to create a highly interactive and personalized conversational experience that satisfies user expectations, offers accurate information, and ensures a user-friendly and engaging interaction

CHAPTER 2:

SYSTEM REQUIREMENTS SPECIFICATIONS

SYSTEM REQUIREMENTS SPECIFICATION

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system.

Functional Requirements

Functional Requirements define the internal working of the software. The following conditions must be taken care of: The ability to perform correct operations when corresponding keys are pressed.

Non-functional Requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of the system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical non-functional requirements are reliability and scalability. Non-functional requirements are “constraints”, “quality attributes” and “quality of service requirements”.

Types of non-functional requirements

- **Volume:** Volume of a system (the total space occupied) varies depending on how the component assemblies are arranged and connected.
- **Reliability:** System reliability depends on component reliability but unexpected interactions can cause new types of failure and therefore affect the reliability of the system.
- **Security:** The security of the system (its ability to resist attacks) is a complex property that cannot be easily measured. Attacks maybe devised that were not anticipated by the system designers and may use default built- in safeguards.
- **Repairability:** This property reflects how easy it is to fix a problem with the system once it has been discovered. It depends on being able to diagnose the problem, access the components that are faulty and modify or replace these components.
- **Usability:** This property reflects how easy it is to use the system. It depends on the technical system components, its operators and its operating environment.

Software Requirements:

Operating System : Windows

Front End : Android Studio

Coding Language : Java

Hardware Requirements

System : Intel® Core™ i3 – 6006U CPU @ 2.00GHz

Hard Disk : 30 GB or above

Monitor : 15 VGA color

RAM : 4GB or above

Introduction to Environment

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go;[20] and Android Studio 3.0 or later supports Kotlin[21] and "all Java 7 language features and a subset of Java 8 language features that vary by platform version."[22] External projects backport some Java 9 features.[23] While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.A specific feature of the Android Studio is an absence of the possibility to switch autosave feature off.

The following features are provided in the current stable version

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations[18]
- Support for building Android Wear app

- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine[19]
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

The Android Emulator has additional requirements beyond the basic system requirements for Android Studio, which are described below:[30]

- SDK Tools 26.1.1 or higher;
- 64-bit processor;
- Windows: CPU with UG (unrestricted guest) support;
- Intel Hardware Accelerated Execution Manager (HAXM) 6.2.1 or later (HAXM 7.2.0 or later recommended).

The use of hardware acceleration has additional requirements on Windows and Linux:

- Intel processor on Windows or Linux: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality;
- AMD processor on Linux: AMD processor with support for AMD Virtualization (AMD-V) and Supplemental Streaming SIMD Extensions 3 (SSSE3);
- AMD processor on Windows: Android Studio 3.2 or higher and Windows 10 April 2018 release or higher for Windows Hypervisor Platform (WHPX) functionality.

CHAPTER 3:
DESIGN

DESIGN

High level design

Requirement analysis encompasses all the tasks that go into the instigation, scoping and definition of a new or altered system. Requirement analysis is an important part of the design process. Here we identify the needs or requirements. Once the requirements have been identified the solution for the requirements can be designed.

We design a project with specific goals, tasks and outcomes. The more specific and the more closely it is aligned. UI (User Interface) and UX (User Experience) are the two vital components of your mobile app design. The former is responsible for the look and appeal of your application, whereas the latter facilitates the interaction between the design elements. While preparing a design for your mobile app, two primary considerations should be your project's scope and budget.

The time required for designing cannot be specified as it may take only a couple of hours to a few days. Another factor that impacts your app designing time is the experience of the developers from your mobile app development services provider. It is a multistep process that needs to be done carefully to ensure that the outcome provides a clear vision of your business idea.

CHAPTER 4:

IMPLEMENTATION

IMPLEMENTATION:

Implementation is an act or instance of implementing something that is, the process of making something active or effective. Implementation must follow any preliminary thinking in order for something to actually happen. In the context of information technology, implementation encompasses the process involved in getting new software or hardware to operate properly in its environment. This includes installation, configuration, running, testing and making necessary changes.

MainActivity.java

```
package com.example.ash;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
    private ArrayList<Message> messages;
    private RecyclerView recyclerView;
    private recyclerAdapter adapter;
    private ImageButton sendButton;
    private EditText msgInput;
    private getRequest request;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        request = new getRequest(this);

        recyclerView = findViewById(R.id.recyclerView);
        // Set RecyclerView layout manager.
        RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
        // Set an animation
        recyclerView.setItemAnimator(new DefaultItemAnimator());

        messages = new ArrayList<>();
        adapter = new recyclerAdapter(messages);
        recyclerView.setAdapter(adapter);

        sendButton = (ImageButton) findViewById(R.id.msgButton);
        msgInput = (EditText) findViewById(R.id.msgInput);

        sendButton.setOnClickListener(new View.OnClickListener() {

            @Override
```

```
        public void onClick(View v) {
            String message = msgInput.getText().toString();
            if(message.length() != 0){
                messages.add(new Message(true, message));
                int newPosition = messages.size() - 1;
                adapter.notifyItemInserted(newPosition);
                recyclerView.scrollToPosition(newPosition);
                msgInput.setText("");
                getReply(message);
            }
        }
    });


}

private void getReply(String message) {
    request.getResponse(message, new getRequest.VolleyResponseListener() {
        @Override
        public void onError(String message) {
            Log.d("REQUEST ERROR", message);
        }

        @Override
        public void onResponse(String reply) {
            messages.add(new Message(false, reply));
            int newPosition = messages.size() - 1;
            adapter.notifyItemInserted(newPosition);
            recyclerView.scrollToPosition(newPosition);
        }
    });

}
}
```

getRequesst.java

```
package com.example.ash;

import android.content.Context;
import android.util.Log;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

public class getRequest {
    private RequestQueue queue;
    private String APIkey = "\n" + "RjvALPtuDKl4xRv3";
    private String brainID = "176314";
    private String reply;
```

```
private char[] illegalChars = {'#', '<', '>', '$', '+', '%', '!', '`', '&',
        '*', '\'', '\"', '|', '{', '}', '/', '\\', ':', '@'};

 public getRequest(Context context){
    queue = Volley.newRequestQueue(context);
 }

 public interface VolleyResponseListener {
    void onError(String message);

    void onResponse(String reply);
 }

 private String formatMessage(String message){

    message = message.replace(' ', '-');
    for(char illegalChar : illegalChars){
       message = message.replace(illegalChar, '-');
    }
    return message;
 }

 public void getResponse(String message, final VolleyResponseListener volleyResponseListener){
    message = formatMessage(message);
    String url = "http://api.brainshop.ai/get?bid=" + brainID+ "&key=" + APIkey + "&uid=1&msg=" + message;
    Log.d("URL", url);
    JsonObjectRequest request = new JsonObjectRequest(Request.Method.GET, url, null,
          new Response.Listener<JSONObject>(){
             @Override
             public void onResponse(JSONObject response){
                try {
                   reply = response.getString("cnt");
                   Log.d("RESPONSE", reply);
                   volleyResponseListener.onResponse(reply);

                } catch (JSONException e) {
                   e.printStackTrace();
                   volleyResponseListener.onError("JSON Exception");
                }

             }
          },
          new Response.ErrorListener(){
             @Override
             public void onErrorResponse(VolleyError error){
                error.printStackTrace();
                volleyResponseListener.onError("Volley Error");
             }
          });
    queue.add(request);
```

```
    }
}
```

Message.java

```
package com.example.ash;

public class Message {
    // Type 0 for sent, 1 for received
    private boolean type;
    // Message content
    private String message;

    public Message(boolean type, String message) {
        this.type = type;
        this.message = message;
    }

    public String getMessage() {
        return message;
    }

    public boolean getType() {
        return this.type;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public void setType(boolean type) {
        this.type = type;
    }
}
```

recyclerAdapter.java

```
package com.example.ash;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.ArrayList;

public class recyclerAdapter extends RecyclerView.Adapter<recyclerAdapter.MessageViewHolder> {
    private ArrayList<Message> messages;

    public recyclerAdapter(ArrayList<Message> messages){
```

```
    this.messages = messages;
  }

  public class MessageViewHolder extends RecyclerView.ViewHolder {
    private LinearLayout sentLayout;
    private LinearLayout receivedLayout;
    private TextView sentText;
    private TextView receivedText;

    public MessageViewHolder(final View itemView) {
      super(itemView);
      sentLayout = itemView.findViewById(R.id.sentLayout);
      receivedLayout = itemView.findViewById(R.id.receivedLayout);
      sentText = itemView.findViewById(R.id.sentTextView);
      receivedText= itemView.findViewById(R.id.receivedTextView);
    }
  }

  @NonNull
  @Override
  public recyclerAdapter.MessageViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View itemView = LayoutInflater.from(parent.getContext()).inflate(R.layout.message, parent, false);
    return new MessageViewHolder(itemView);
  }

  @Override
  public void onBindViewHolder(@NonNull recyclerAdapter.MessageViewHolder holder, int position) {
    String message = messages.get(position).getMessage();
    boolean type = messages.get(position).getType();

    if(type){
      //If a message is sent
      holder.sentLayout.setVisibility(LinearLayout.VISIBLE);
      holder.sentText.setText(message);
      // Set visibility as GONE to remove the space taken up
      holder.receivedLayout.setVisibility(LinearLayout.GONE);
    }
    else{
      //Message is received
      holder.receivedLayout.setVisibility(LinearLayout.VISIBLE);
      holder.receivedText.setText(message);
      // Set visibility as GONE to remove the space taken up
      holder.sentLayout.setVisibility(LinearLayout.GONE);
    }
  }

  @Override
  public int getItemCount() {
    return messages.size();
  }
}
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
```

```
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@android:color/transparent" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="@color/inputBackground"
        android:padding="5dp">

        <EditText
            android:id="@+id/msgInput"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:layout_weight="1"
            android:maxLines="2"
            android:hint="@string/message"
            android:inputType="text"
            android:textSize="14sp"
            android:background="@drawable/rounded_corner"/>

        <ImageButton
            android:id="@+id/msgButton"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_marginLeft="5dp"
            android:background="@android:color/black"
            android:padding="5dp"
            android:scaleType="fitCenter"
            android:src="@drawable/outline_send_white_24dp" />

    </LinearLayout>

</LinearLayout>
```

Message.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="5dp"
    android:background="@color/textBackground">
```

```
<LinearLayout
    android:id="@+id/receivedLayout"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="left"

        android:background="@drawable/message_received">

    <TextView
        android:id="@+id/receivedTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="10dp"
        android:textColor="#fff"/>

</LinearLayout>

<LinearLayout
    android:id="@+id/sentLayout"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:background="@drawable/message_sent">

    <TextView
        android:id="@+id/sentTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="10dp"
        android:textColor="#fff"/>

</LinearLayout>

</LinearLayout>
```

CHAPTER 5:

SNAPSHOTS

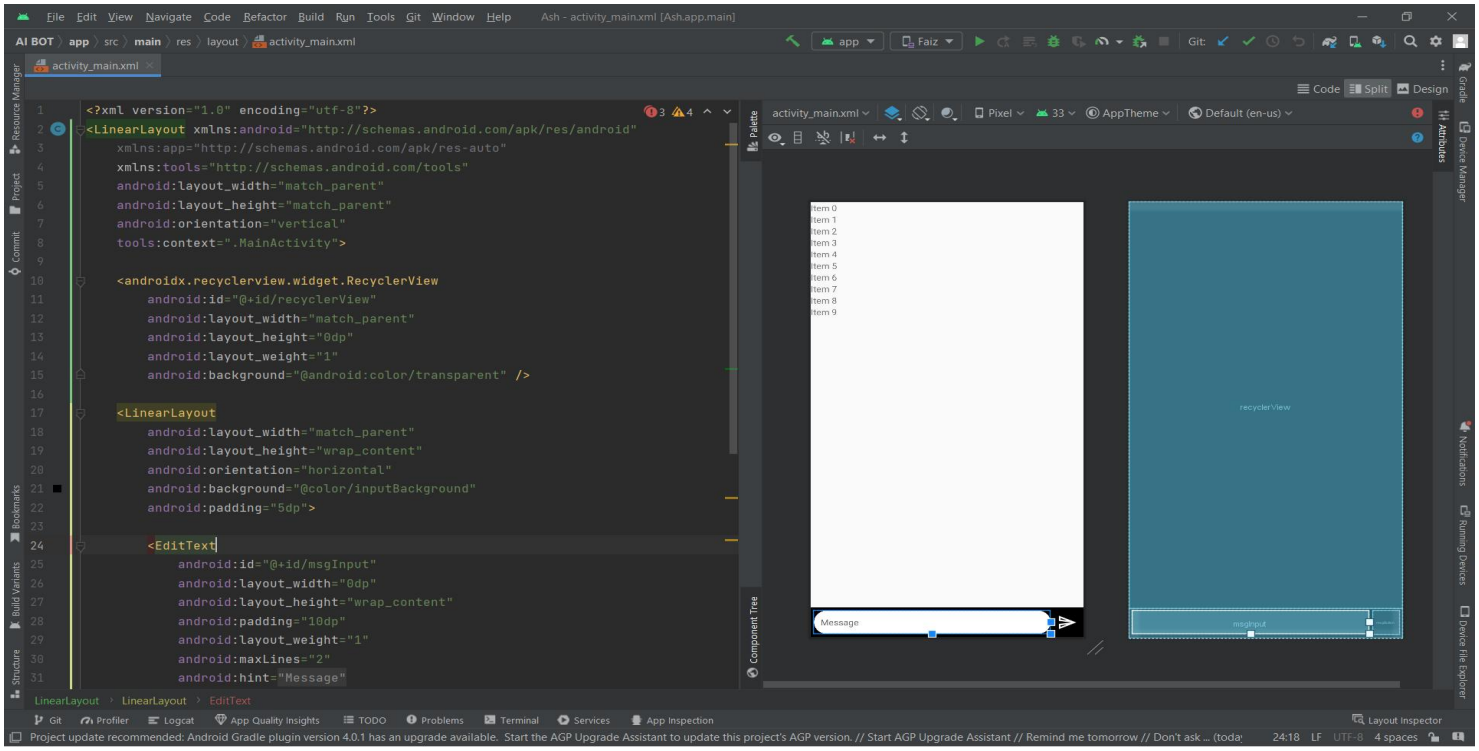

Fig 5.1 activity_main.xml

activity_main.xml file is an important component in Android development that defines the layout and appearance of the main user interface (UI) for an Android app. It is typically created as an XML (Extensible Markup Language) file and is associated with the main activity of an app.

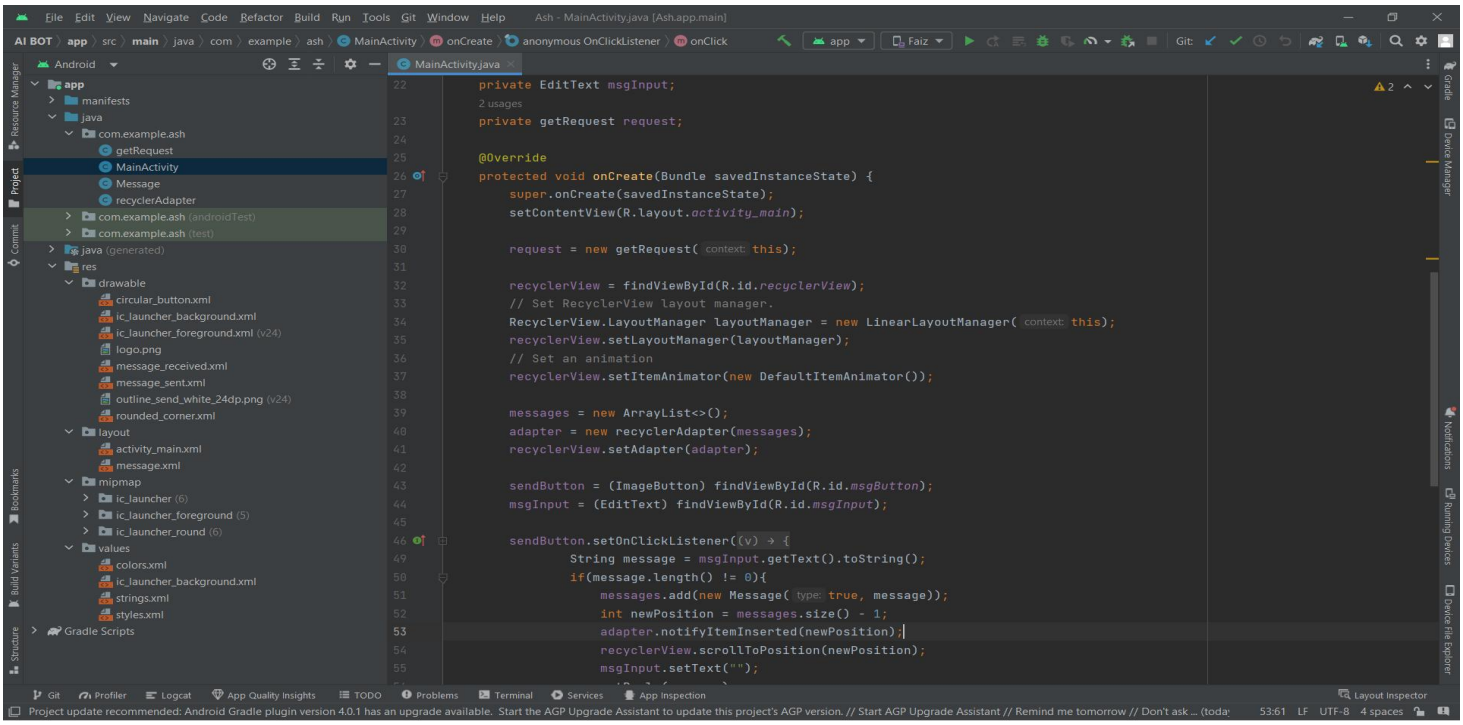

Fig 5.2 MainActivity.java

The MainActivity.java file is a Java class that serves as the main entry point for an Android app. It is responsible for controlling the behavior and logic of the main activity, which represents the primary user interface for the app.

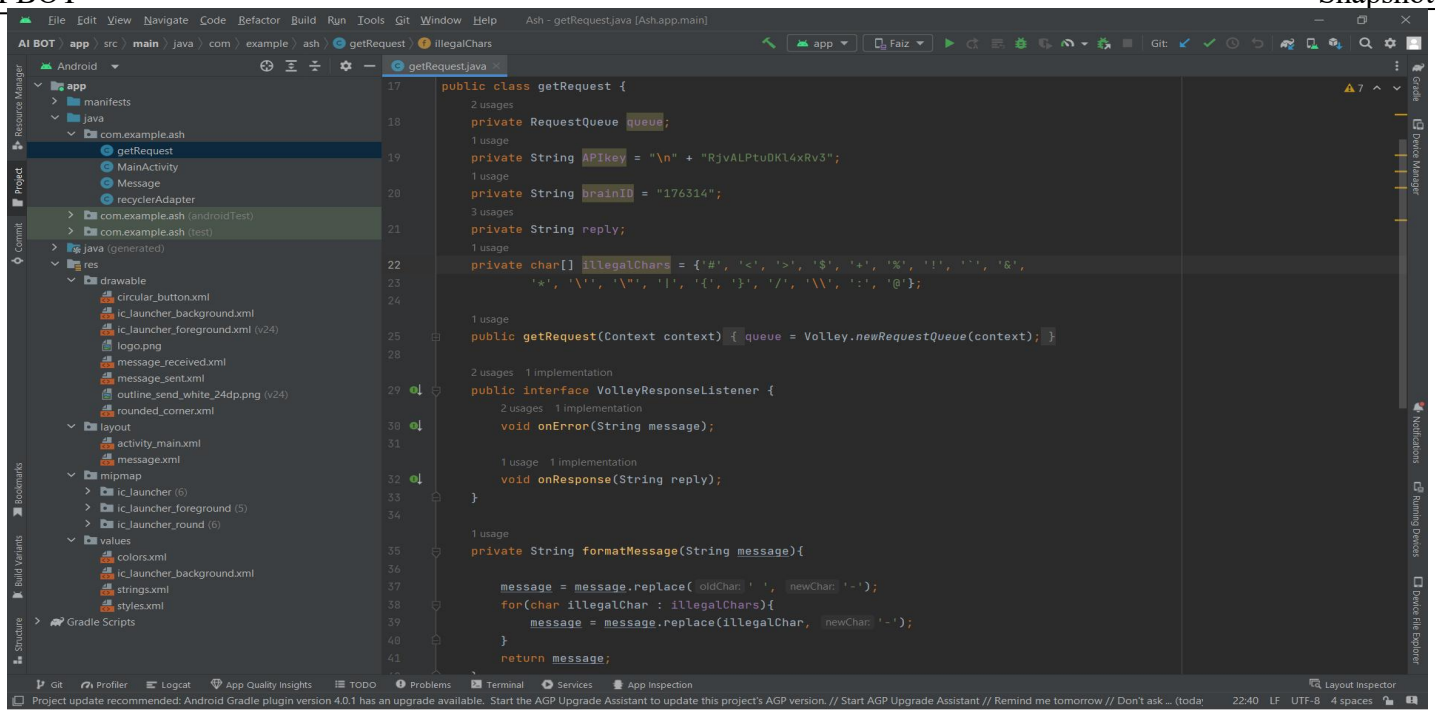

Fig 5.3 getRequest.java

getRequest.java will send a request message to the Brain shop API along with the question to get the answer. When the answer is received it is later sent to the Aativity_Main.java

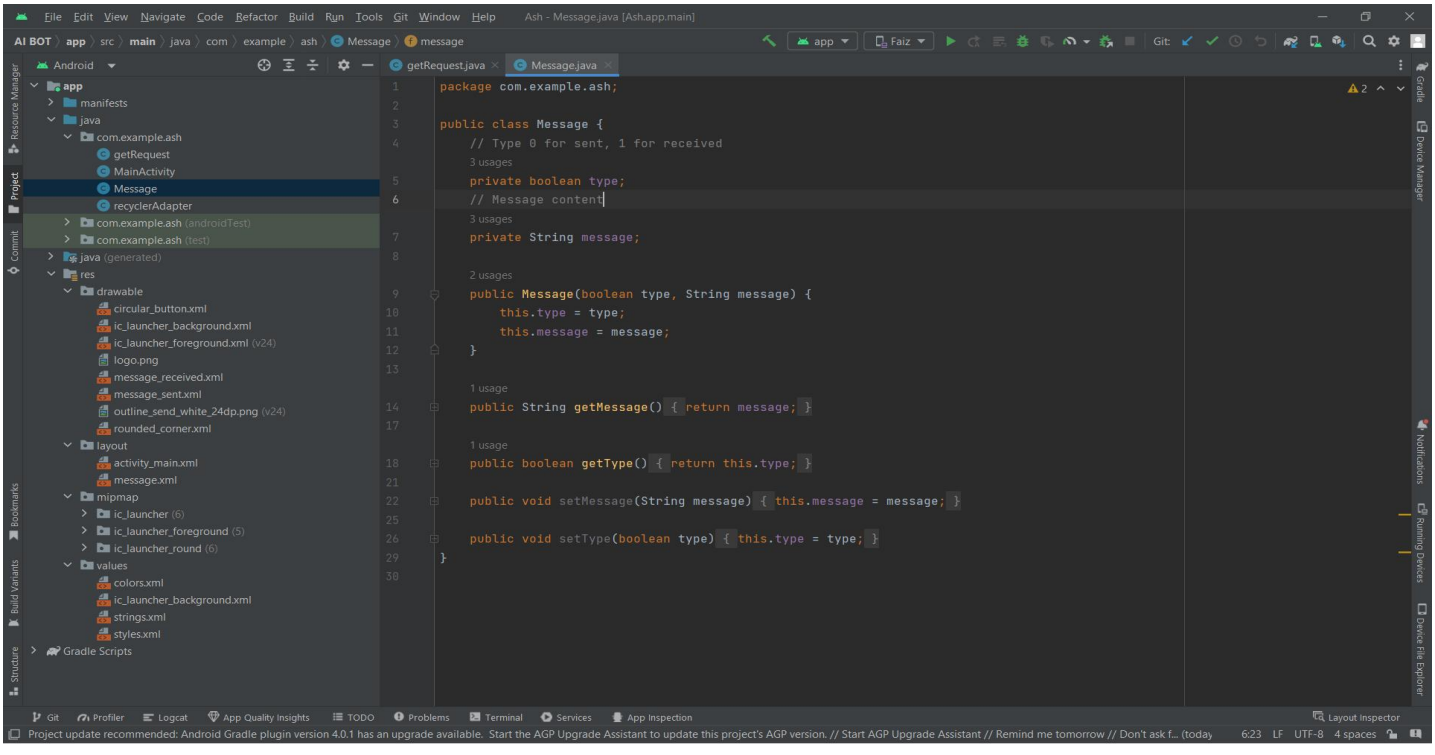

Fig 5.4 Message.java

Message.java helps in classification of messages which are sent by user and which is received by the AI BOT and accordingly places them into left or right corner

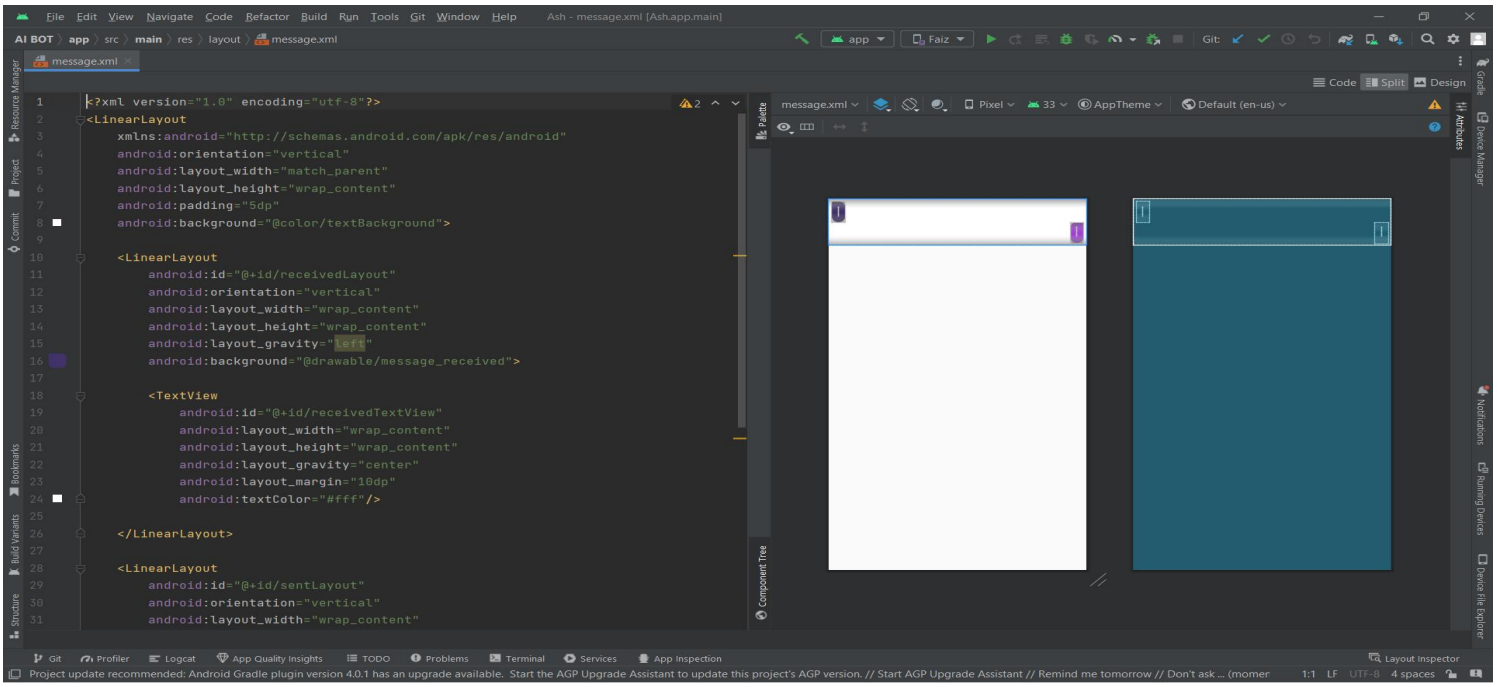

Fig 5.5: Message.xml

It is the design layout for the messages weather it should be placed left or right based on who sent the message.

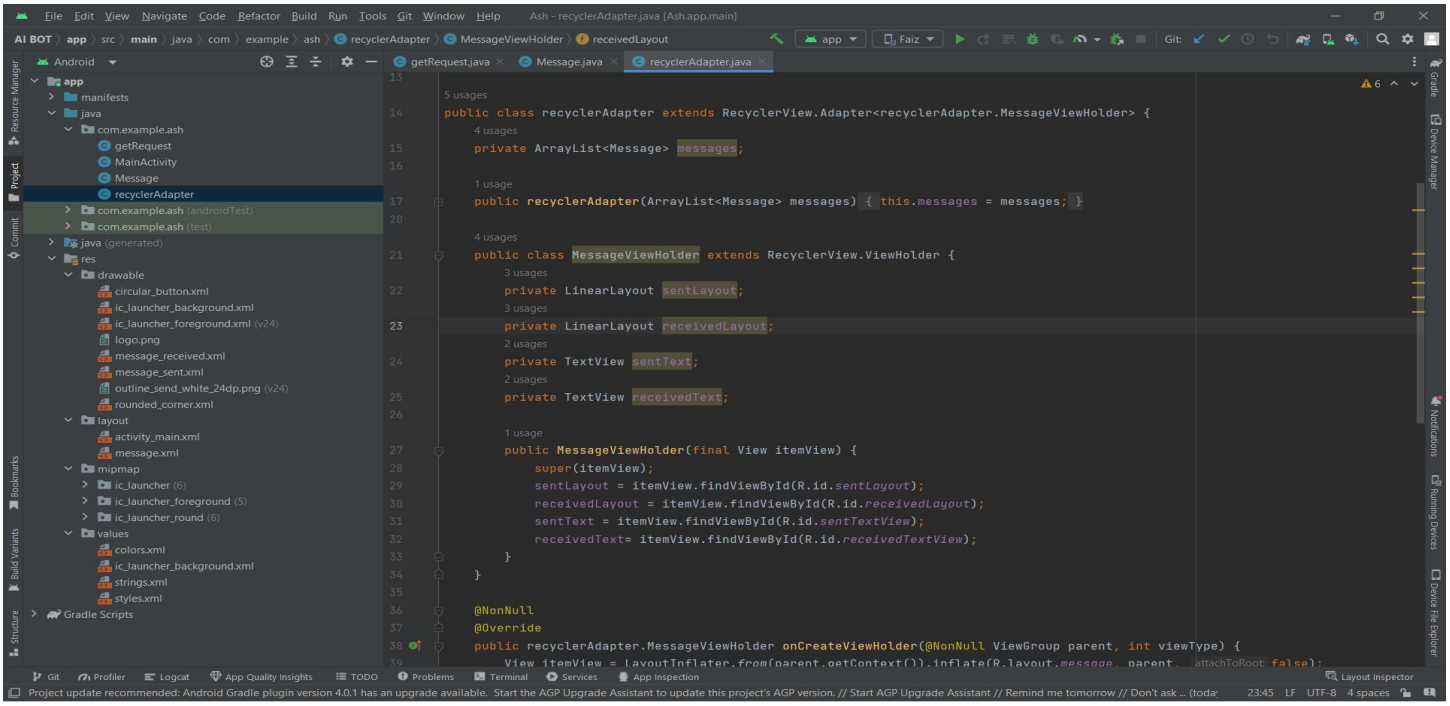

Fig 5.6: RecyclerAdapter.java

It sets Text in Recycler view of the adapter in XML it gets message from message. Java then uses the fuction Recyclerview.setText("Test_Message); to set text either on right(if sent by AI BOT) or left (IF sent by User)

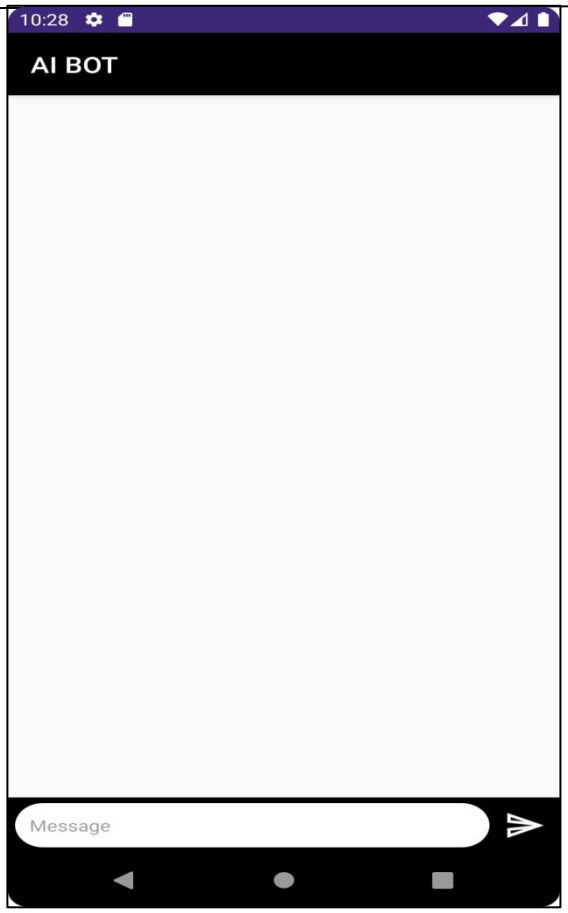

Fig 5.7 Output(1)

AI chatbots have revolutionized the way we interact with technology by providing intelligent conversational interfaces. An AI chatbot, short for Artificial Intelligence chatbot, is a computer program designed to simulate human-like conversations and engage in meaningful interactions with users. Powered by advanced machine learning algorithms and natural language processing techniques, AI chatbots can understand user queries, generate relevant responses, and even adapt to specific domains or preferences. These chatbots leverage vast amounts of data,knowledge bases, and APIs to provide accurate information, offer assistance, and create personalized experiences. With their ability to handle a wide range of tasks, AI chatbots have become valuable tools in customer support, virtual assistants, and various other applications where effective human-computer interactions are essential. As technology continues to advance, AI chatbots hold tremendous potential in transforming how we engage with digital platforms, making interactions more intuitive, efficient, and human

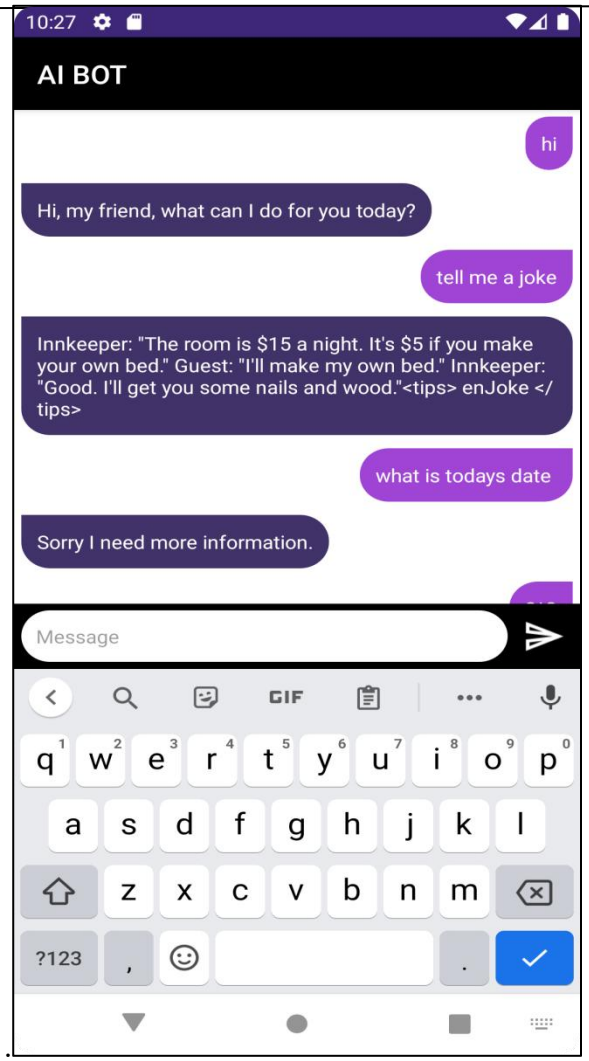

Fig 5.8 Output(2)

- Natural Language Processing: AI chatbots utilize advanced natural language processing techniques to understand and interpret user queries and statements in a conversational manner.

- Personalization and Contextual Understanding: AI chatbots can adapt to specific domains or user preferences, tailoring their responses and recommendations to provide a personalized experience. They strive to grasp the context of conversations, considering previous interactions to provide accurate and relevant responses.

- Knowledge Base Integration: AI chatbots leverage extensive knowledge bases and databases to access a wide range of information, allowing them to provide accurate answers and insights to user queries.

- Learning and Improvement: AI chatbots continuously learn from user interactions, feedback, and data to improve their conversational abilities, refining their responses and becoming more effective over time. They can adapt and enhance their capabilities through machine learning techniques.

CHAPTER 6:

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion:

In conclusion, the development of an AI chatbot utilizing the BrainShop API has shown promising results in achieving natural and human-like conversations with users. Through its advanced conversational abilities and integration of pre-existing knowledge, the chatbot has been able to understand user queries and statements accurately, generating coherent and contextually relevant responses. The personalization and adaptability of the chatbot have allowed it to cater to specific domains and user preferences, continuously refining its conversational abilities based on user interactions. Leveraging the power of the BrainShop API has further enhanced the chatbot's capabilities, providing valuable information to users and enriching the conversational experience. Techniques such as fine-tuning, transfer learning, and reinforcement learning have enabled the chatbot to continuously improve its performance and adapt to user feedback. The result is a highly interactive and personalized conversational experience that meets user expectations, offering accurate information and engaging interactions. Looking ahead, further advancements in AI technologies and ongoing research will continue to enhance the capabilities of AI chatbots, enabling them to become even more sophisticated and valuable in providing effective and satisfying conversational interactions

6.2Future enhancements+

In the future, there are several potential enhancements that can be considered to further advance the capabilities of AI chatbots built using the BrainShop API. These enhancements include:

- Improved Contextual Understanding: Enhancing the chatbot's ability to grasp and understand complex contexts within conversations will lead to more accurate and contextually relevant responses. This can involve incorporating advanced natural language processing techniques, such as contextual embeddings or contextual language models, to better capture the nuances of conversations.

- Expanded Knowledge Base: Continuously expanding and updating the chatbot's knowledge base will enable it to provide more comprehensive and up-to-date information to users. This can involve integrating additional external data sources, leveraging web scraping techniques, or implementing mechanisms for community-driven knowledge contributions.

- Emotional Intelligence: Introducing emotional intelligence into the chatbot can enhance its ability to recognize and respond to users' emotions. This can involve incorporating sentiment analysis techniques or utilizing affective computing to understand and empathize with users' emotional states, leading to more personalized and emotionally aware interactions.

- Conversational Flow and Coherence: Improving the chatbot's ability to maintain a smooth and coherent conversation over extended interactions will contribute to a more engaging user experience. Techniques such as dialogue state tracking, dialogue management, and reinforcement learning can be employed to ensure a seamless flow of conversation and reduce instances of repetitive or inconsistent responses.

- Enhanced User Interaction and Interface: Developing user-friendly interfaces and interactive features can enhance the overall user experience. This can include implementing voice assistants, chatbot widgets for websites, or integration with messaging platforms to make the chatbot more accessible and convenient for users to engage with.

By focusing on these future enhancements, AI chatbots built using the BrainShop API can continue to evolve and provide users with more sophisticated, personalized, and engaging conversational experiences, further bridging the gap between human-like conversations and AI-driven conversational systems.

6.3 Limitations:

- Contextual Understanding Challenges: AI chatbots may struggle with understanding complex or ambiguous contexts within conversations. They may have difficulty interpreting sarcasm, irony, or subtle nuances in language, leading to inaccurate or inappropriate responses.

- Knowledge Limitations: While AI chatbots strive to provide accurate information, their effectiveness heavily relies on the quality and breadth of their knowledge base. Limited access to up-to-date or specific information may result in incomplete or outdated responses.

- Lack of Common Sense Reasoning: AI chatbots often lack real-world experiences and intuitive understanding, making it challenging for them to apply common sense reasoning. This can lead to instances where the chatbot fails to provide logical or common-sense responses, resulting in user dissatisfaction.

BIBLIOGRAPHY:

1. Erik Hellman, “Android Programming – Pushing the Limits”, 1st Edition, Wiley India Pvt Ltd, 2014. ISBN-13: 978-8126547197

2. Dawn Griffiths and David Griffiths, “Head First Android Development”, 1st Edition, O’Reilly SPD Publishers, 2015. ISBN-13: 978-9352131341

3. Bill Phillips, Chris Stewart and Kristin Marsicano, “Android Programming: The Big Nerd Ranch Guide”, 3rd Edition, Big Nerd Ranch Guides, 2017. ISBN-13: 978-0134706054

Links:

Google Developer Training, "Android Developer Fundamentals Course – Concept Reference”, Google Developer Training Team, 2017. https://www.gitbook.com/book/google-developer-training/android-developer- fundamentals- course-concepts/details

HKBK College of Engineering
No.22/1, Opp., Manyata Tech Park Rd, Nagawara, Bengaluru, Karnataka 560045.
Approved by AICTE & Affiliated by VTU

Department of Computer Science & Engineering

DEPARTMENT MISSION AND VISION

VISION

To advance the intellectual capacity of the nation and the international community by imparting knowledge to graduates who are globally recognized as innovators, entrepreneur and competent professionals.

MISSION

M-1. To provide excellent technical knowledge and computing skills to make the graduates globally competitive with professional ethics.

M-2. To involve in research activities and be committed to lifelong learning to make positive contributions to the society. Institute

INSTITUTE MISSION AND VISION

VISION

To empower students through wholesome education and enable the students to develop into highly qualified and trained professionals with ethics and emerge as responsible citizen with broad outlook to build a vibrant nation.

MISSION

M - 1. To achieve academic excellence through in-depth knowledge in science, engineering and technology through dedication to duty, innovation in teaching and faith in human values.

M - 2. To enable our students to develop into outstanding professionals with high ethical standards to face the challenges of the 21st century

M-3. To provide educational opportunities to the deprived and weaker section of the society, to uplift their socio-economic status.

PROGRAM OUTCOMES(PO'S)

PO-1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO-4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO-5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO-6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO-9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1. Problem-Solving Skills: An ability to investigate and solve a problem by analysis, interpretation of data, design and implementation through appropriate techniques, tools and skills.

PSO-2. Professional Skills: An ability to apply algorithmic principles, computing skills and computer science theory in the modelling and design of computer-based systems.

PSO-3. Entrepreneurial Ability: An ability to apply design, development principles and management skills in the construction of software product of varying complexity to become an entrepreneur.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

PEO-1. To provide students with a strong foundation in engineering fundamentals and in the computer science and engineering to work in the global scenario.

PEO-2. To provide sound knowledge of programming and computing techniques and good communication and interpersonal skills so that they will be capable of analyzing, designing and building innovative software systems.

PEO-3. To equip students in the chosen field of engineering and related fields to enable him to work in multidisciplinary teams.

PEO-4. To inculcate in students professional, personal and ethical attitude to relate engineering issues to broader social context and become responsible citizen.

PEO-5. To provide students with an environment for life-long learning which allow them to successfully adapt to the evolving technologies throughout their professional career and face the global challenges.

COURSE OUTCOMES(COs)

CO-1. Create, test and debug Android application by setting up Android development environment.

CO-2. Implement adaptive, responsive user interfaces that work across a wide range of devices.

CO-3. Infer long running tasks and background work in Android applications.

CO-4. Demonstrate methods in storing, sharing and retrieving data in Android applications.

CO-5. Analyse performance of android applications and understand the role of permissions and security.

CO-6. Describe the steps involved in publishing Android application to share with the world.