

THAKUR DEGREE COLLEGE OF SCIENCE & COMMERCE
KANDIVALI (EAST)
MUMBAI

A
PROJECT REPORT
ON

SUPER CHATBOT

Designed and Developed
BY: PRATHAMESH SARJERAO VAIDYA

Submitted in partial fulfillment of
Bachelors of Science (Computer Science)

[UNIVERSITY OF MUMBAI]
Thakur Degree College of Science and Commerce
KANDIVALI (EAST), MUMBAI
ACADEMIC YEAR 2022 - 2023

CERTIFICATE FROM COLLEGE



Thakur Degree College of Science & Commerce
Shyamnarayan Thakur Marg, Thakur Village
Kandivali (E), Mumbai - 400 101

DATE:

COMPUTER DEPARTMENT

(2022-2023)

Certificate of Approval

This is to certify that the project work entitled "**Super Chatbot**" is prepared by **Prathamesh Sarjerao Vaidya** a student of "**Third Year Bachelor Of Science (Computer Science)**" course of University of Mumbai, which is conducted by our college.

This is the original study work and important sources used have been duly acknowledged in the report. The report is submitted in partial fulfillment of B.Sc. (Computer Science) course as per rules of University of Mumbai

Project Guide

Head of Department

External Examiner

INDEX

Sr. No.	Index Topic	Page
	Acknowledgement	7
I	Introduction	11
(i)	Preliminary Investigation	9
(ii)	Description of System	12
(iii)	Limitations of present system	12
(iv)	Proposed system and its advantages	13
(v)	System Requirements	14
(vi)	Feasibility Study	18
(vii)	Gantt Chart	19
III	UML Diagram	21
(i)	Use case diagram	22
(ii)	Activity Diagram	22
(iii)	Class Diagram	23
(iv)	Component Diagram	23
(v)	Sequence Diagram	24

IV	Data Flow Diagram (DFD)	25
V	Software testing	27
VI	System Coding	29
(ii)	Screen Layouts	30
(iii)	Sample Code	34
VII	Conclusion	53
VIII	Future Enhancements	53
IX	Reference and Bibliography	55

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

Achievement is finding out what you would be doing rather than what you have to do. It is not until you undertake such a project that you realize how much effort and hard work it really is, what are your capabilities and how well you can present yourself or other things. It gives me immense pleasure to present this report towards the fulfillment of my project.

It has been rightly said that we are built on the shoulder of others. For everything I have achieved, the credit goes to all those who had helped me to complete this project successfully.

I take this opportunity to express my profound gratitude to management of Thakur Degree College of Science & Commerce for giving me this opportunity to accomplish this project work.

I am very much thankful to **Dr. Mrs. C. T. Chakraborty** – Principal of Thakur College for their kind co-operation in the completion of my project.

A special vote of thanks to my faculty, **Mr. Ashish Trivedi** who is our HOD & also our project guide **Dr. Girish Tere** for their most sincere, useful and encouraging contribution throughout the project span, without them we couldn't start and complete the project on time.

Finally, I would like to thank all my friends & entire Computer Science department who directly or indirectly helped me in completion of this project & to my family without whose support, motivation & encouragement this would not have been possible.

(PRATHAMESH SARJERAO VAIDYA)

PRELIMINARY INVESTIGATION

Organizational Overview

Name : Thakur College Of Science & Commerce

Address : Thakur Village, Kandivali (East) Mumbai – 400101

Contacts : 022-2846 2565 / 2887 0627

History :

- ✓ Thakur Junior College was established in 1992, by the founding members
- ✓ It was a natural augmentation by the Thakur Educational Trust
- ✓ Thakur College had a humble beginning with only 57 students in FYJC first batch of Commerce stream
- ✓ Our college has accomplished a spectacular growth over the last two decades
- ✓ The College has consistently attained outstanding results in academics at both Degree & Juniorlevels.

INTRODUCTION

DESCRIPTION OF SYSTEM

A chatbot is a software or computer program that simulates human conversation or "chatter" through text or voice interactions.

Users in both business-to-consumer (B2C) and business-to-business (B2B) environments increasingly use chatbot virtual assistants to handle simple tasks. Adding chatbot assistants reduces overhead costs, uses support staff time better and enables organizations to provide customer service during hours when live agents aren't available

LIMITATIONS OF EXISTING SYSTEM

1. Chatbots Don't Understand Human Context.

These chatbots are programmed in a way that they only know what they are taught. The AI-powered smart-bots can understand the general context, but 40 out of 100 cases are not related to the broad context.

2. They Don't Do Customer Retention.

A chatbot is significantly less capable of retaining the customers as it only tries up to a level for which it is programmed.

3. They Can't Make Decisions.

They don't have the right know-how to differentiate between the good and the bad.

On March 23, 2016, the tech biggie Microsoft attracted many controversies due to its chatbot Tay.

The chatbot posted offensive Tweets and landed Microsoft in huge troubles. So they have to shut down the chatbot temporarily.

4. Exorbitant Installation

Yes, chatbots save you a lot of money in the long run, but their installation cost can break the bank. You need to hire professionals who have rightly programmed chatbots to match the integrity of your business.

5. Chatbots Have the Same Answer For a Query

Chatbots are easily identifiable because they have the same answer for multiple queries. Suppose you are asking something to a bot that is not available in the data server so that you will get an apology.

6. They Have Zero Research Skills

The harsh reality of chatbots is that they have zero research skills. These bots only have the answers to the available queries; they cannot research new topics on the web.

7. Chatbots Have No Emotions

Chatbots have no emotions, and they cannot relate to any low situation. Having no emotions means a chatbot can never establish a connection with the customer, which is crucial for any business's growth.

PROPOSED SYSTEM

1. A bot project is built using artificial algorithms that analyzes user's queries and understand user's message.
2. User just have to query through the bot which is used for chatting.
3. User can chat using any format there is no specific format the user has to follow.
4. The User can query any related activities through the system.
5. The system replies using an effective Graphical user interface which implies that as if a real person is talking to the user.
6. The user can query about related activities through online with the help of this web application.
7. This system helps the user to get answers about general question.

ADVANTAGES OF PROPOSED SYSTEM

1. Chatbots Can Enhance Sales

It is seen that the customer experience gets even better when they have an assisting agent with them. chatbot can assist the clients in shopping while resolving their queries on the go. Also, any purchase chances are higher when the customer is getting better services throughout the shopping journey

2. Chatbots Can Manage Accounts

Managing accounts requires one to be precise about the data and the information. And you can use a chatbot to manage accounts and retrieve details whenever needed. These artificially intelligent bots are more precise than humans, and there is a small scope of mistakes when you have the right chatbots deployed.

3. Chatbots are Customer Executives

Most of the websites render online chat options due to their ease and convenience. Chatbots are perfect for deploying as customer executives because they are available around the clock and seamlessly resolve customer issues. Chatbots have more efficiency in dealing with clients when compared to human agents.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

- ✓ Pentium 3 Processor
- ✓ 4 GB RAM
- ✓ 32 GB Hard Disk

SOFTWARE REQUIREMENTS

- ✓ Android Studio 2021.2.1.15 and above

FRONT END

- ✓ XML

BACK END

- ✓ Java

Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as primary IDE for native Android application development. 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.

The current stable version is 3.2, which was released in September 2018. Features: -

The following features are provided in the current stable version:

- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- Pro-Guard 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
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.
- Gradle-based build support
- Android Studio supports all the same programming languages of IntelliJ, and PyCharm e.g. Python, and Kotlin and Android Studio 3.0 supports "Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features

WHAT IS JAVA?

Java is a programming language and computing platform first released by Sun Microsystems in 1995. It has evolved from humble beginnings to power a large share of today's digital world, by providing the reliable platform upon which many services and applications are built. New, innovative products and digital services designed for the future continue to rely on Java, as well.

While most modern Java applications combine the Java runtime and application together, there are still many applications and even some websites that will not function unless you have a desktop Java installed. Java.com, this website, is intended for consumers who may still require Java for their desktop applications – specifically applications targeting Java 8. Developers as well as users that would like to learn Java programming should visit the dev.java website

ADVANTAGES OF JAVA

1. Simple

Its syntax is based on C++, and it uses automatic garbage collection; therefore, we don't need to remove the unreferenced objects from memory. Java has also removed the features like explicit pointers, operator overloading, etc., making it easy to read and write.

2. Object-Oriented

Everything in Java is an object which takes care of both data and behavior. Java uses object-oriented concepts like object, class, inheritance, encapsulation, polymorphism and abstraction.

3. Secured

Java is a secured programming language because it doesn't use Explicit pointers. Also, Java programs run inside the virtual machine sandbox.

4. Robust

Java is a robust programming language since it uses strong memory management. We can also handle exceptions through the Java code. Also, we can use type checking to make our code more secure.

5. Platform independent

Java code can run on multiple platforms directly, i.e., we need not compile it every time. It is right once, runs anywhere language (WORA) which can be converted into byte code at the compile time. The byte code is a platform-independent code that can run on multiple platforms.

6. Multi-Threaded

Java uses a multi-threaded environment in which a bigger task can be converted into various threads and run separately. The main advantage of multi-threading is that we need not provide memory to every running thread.

HTML

(Hyper Text Markup Language)

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `` and `<input />` directly introduce content into the page. Other tags such as `<p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

WHAT IS XML?

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. It is derived from Standard Generalized Markup Language (SGML). Basically, the XML tags are not predefined in XML. We need to implement and define the tags in XML. XML tags define the data and used to store and organize data. It's easily scalable and simple to develop. In Android, the XML is used to implement UI-related data, and it's a lightweight markup language that doesn't make layout heavy. XML only contains tags, while implementing they need to be just invoked.

Feasibility Analysis

Chatbots were made to be the alternate for support centre or inquiries specific jobs. In the hope to utilize the logical and thinking prowess of humans in more regions where necessary. But in reality, it turned out even after 70 years of development an AI chatbot feels rather robotic. The feasibility of a chatbot is not as per expected. Chatbot fails at creating an emotional connection with the user which completely destroys the conversation. The conversations don't last as long as fifteen minutes. Most of the chatbots still do this day are using poor old rule-based techniques. The complexity increases at an exponential level in regards to creating it more humanly. Artificial intelligence is increasing at the highest rate in the chatbot sector when compared to any other

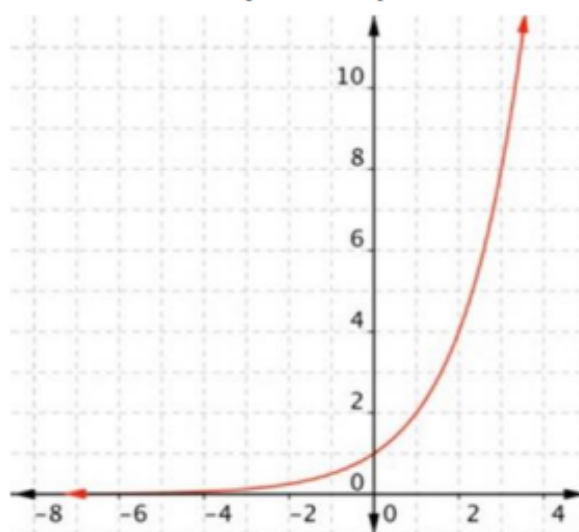


Fig - 1: Complexity of neural networks vs technological improvement curve

The Vertical line here represents the complexity of the Neural Network as in comparison to the improvement as a more human-understandable chatbot

GANTT CHART

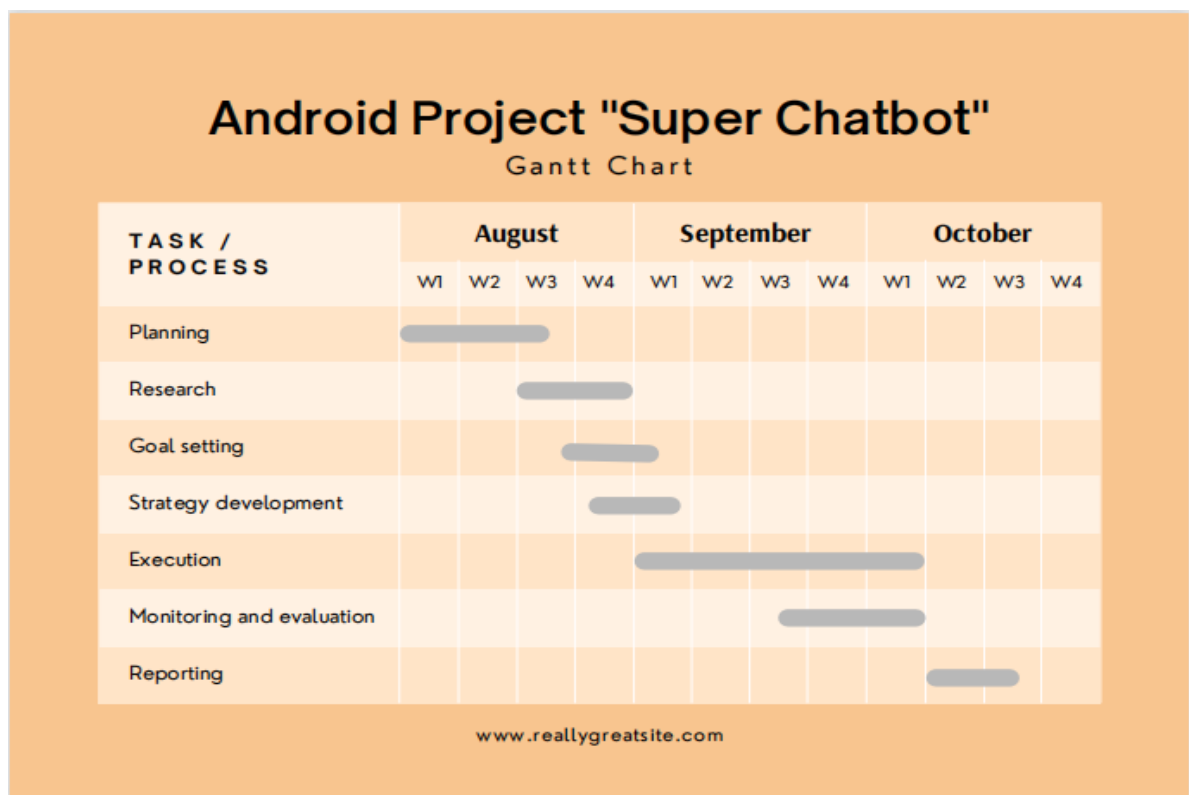


Fig - 2: Gantt Chart

UML DIAGRAMS

Use Case Diagram

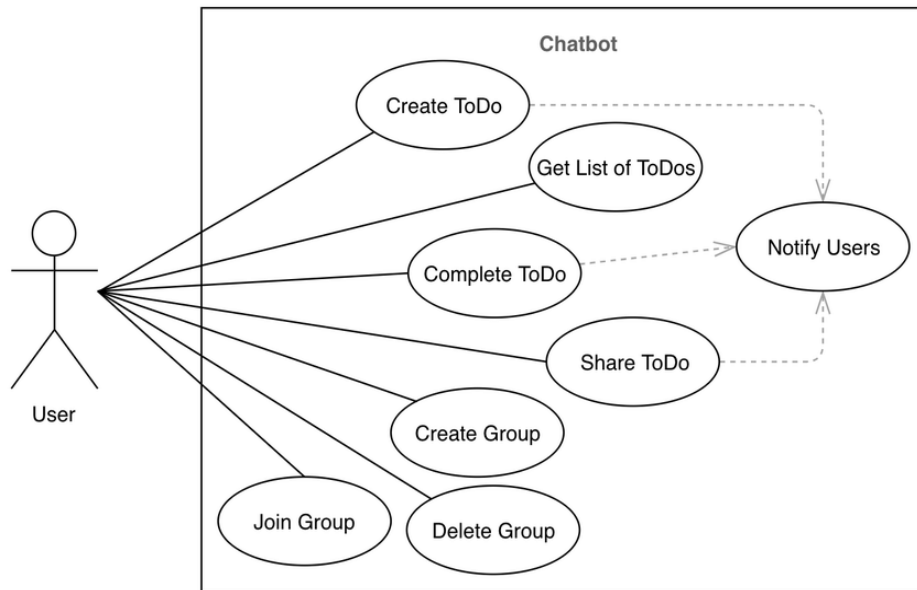


Fig - 3: Use Case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program under developed. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (such as UML).

Activity Diagram

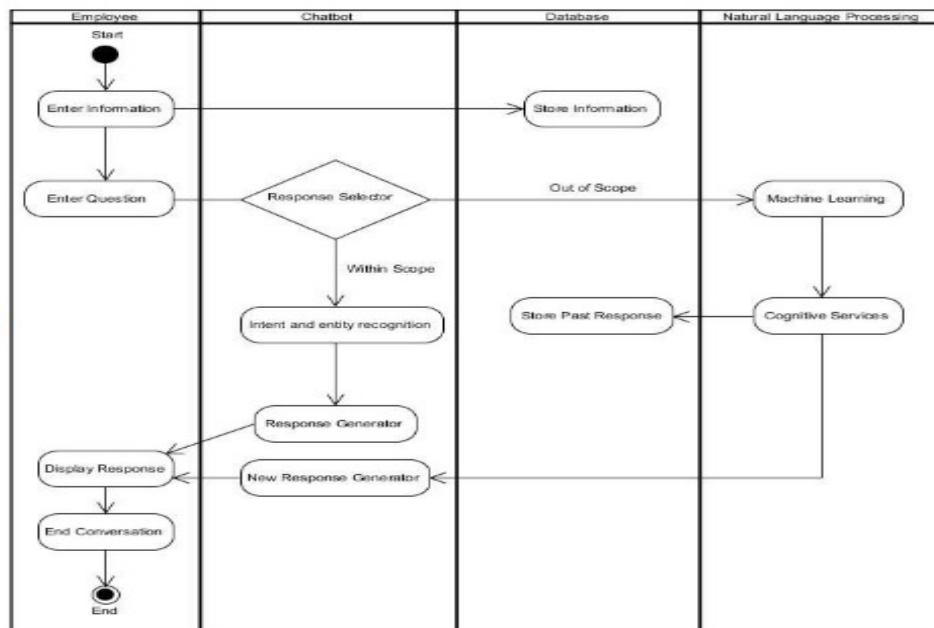


Fig - 4: Activity Diagram

Activity Diagrams describe how activities are coordinated to provide a service which can be different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require

coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination.

Class Diagram

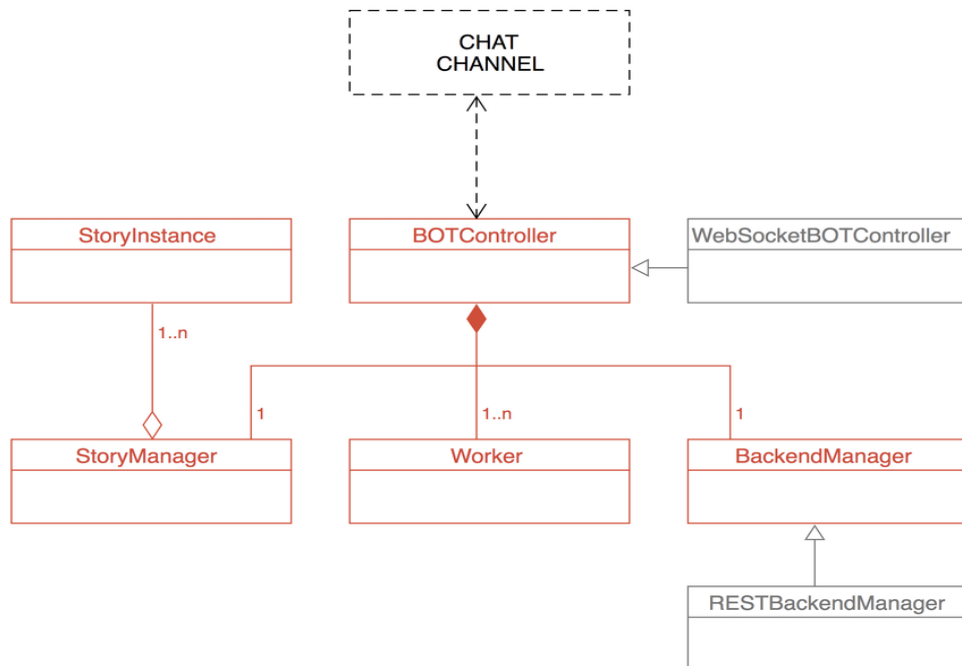


Fig - 5: Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Component Diagram

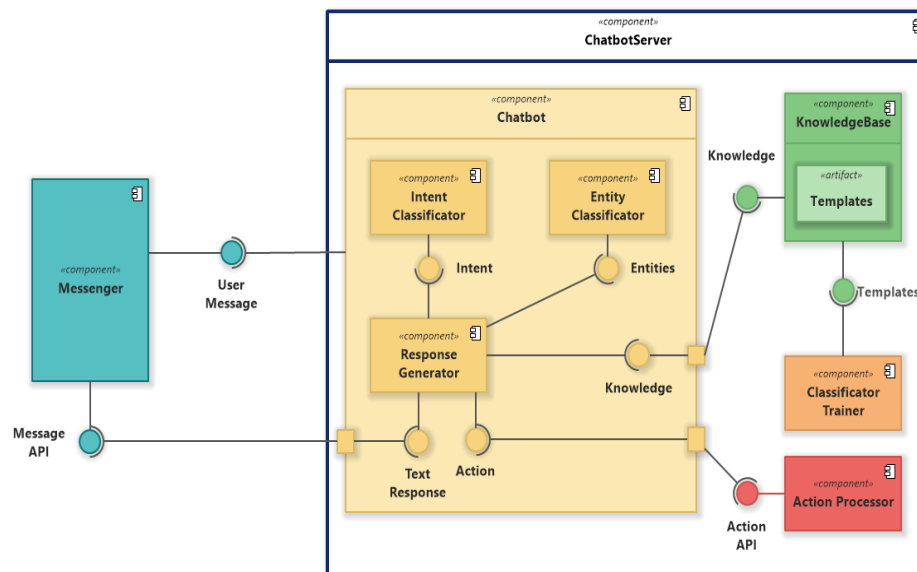


Fig - 6: Component Diagram

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

Sequence Diagram

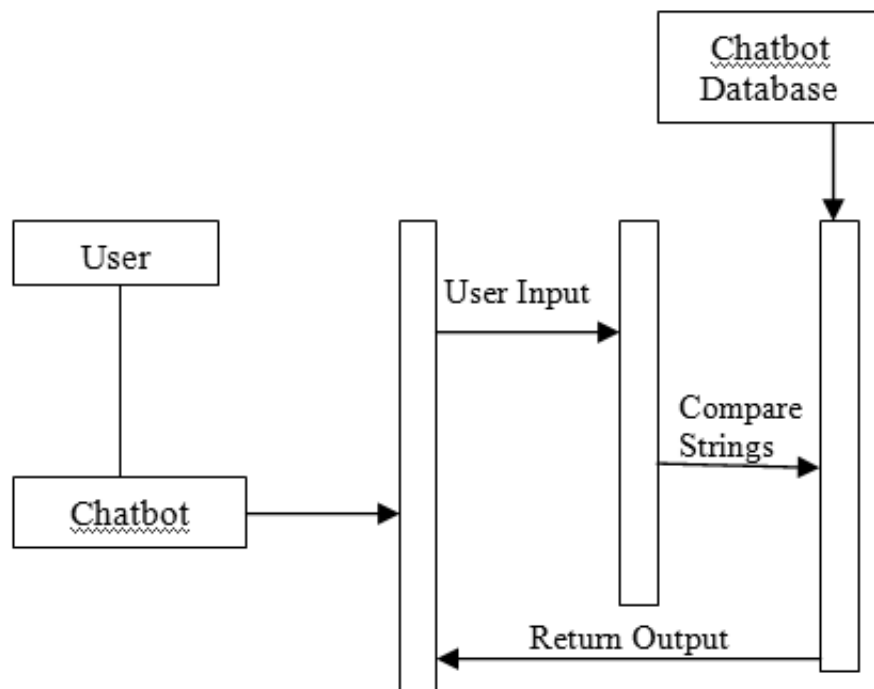


Fig - 7: Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out+.

They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.^[15] Figure 5 describes that how a user can communicate through chatbot as it take help of database and server to get response for user queries

DATA FLOW DIAGRAM

Data Flow Diagram

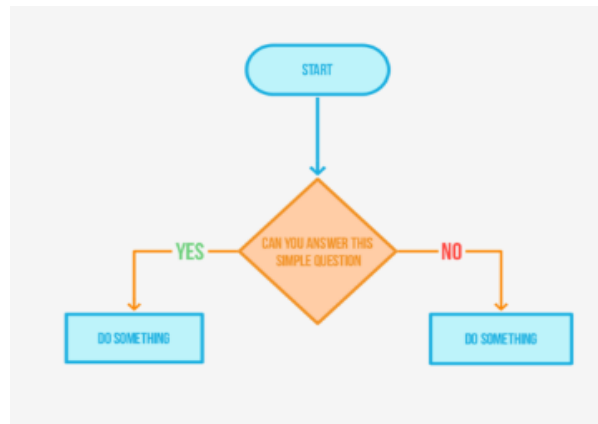
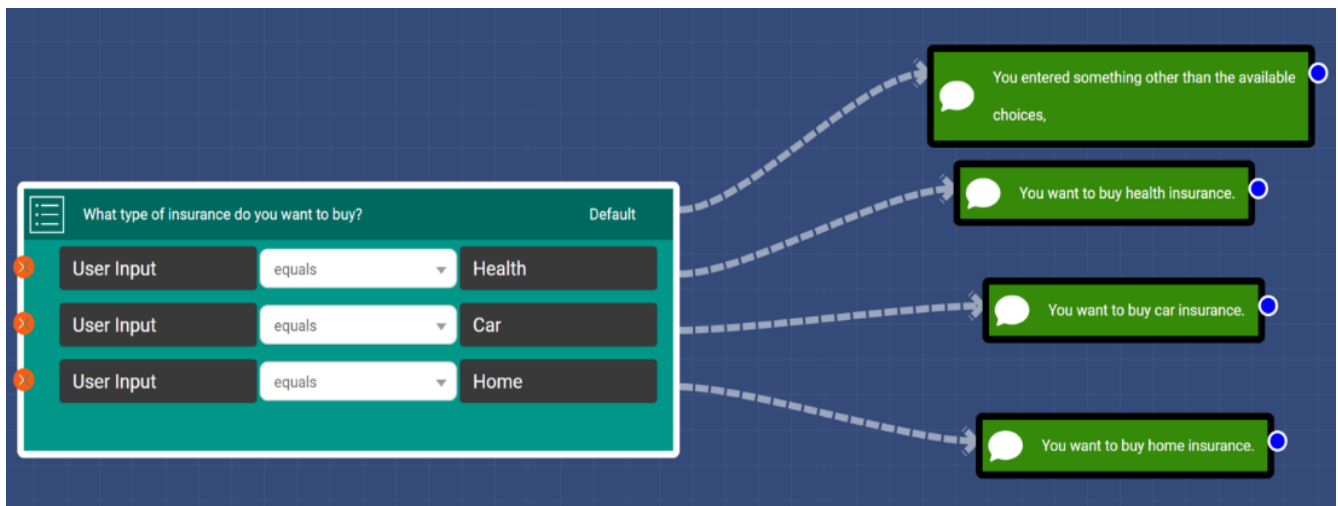


Fig - 8: Data Flow Diagram

Also known as DFD, Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.



SOFTWARE TESTING

WHAT IS SOFTWARE TESTING

Software Testing is a process of executing a program with the intent of finding error. Testing demonstrates that the software functions appear to be working according to specification, that behavior and performance requirements appear to have been met. The system has been thoroughly tested to ensure robustness and reliability. Various testing strategies and techniques have been used as follows:

BLACK-BOX TESTING

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

WHITE-BOX TESTING

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

GREY-BOX TESTING

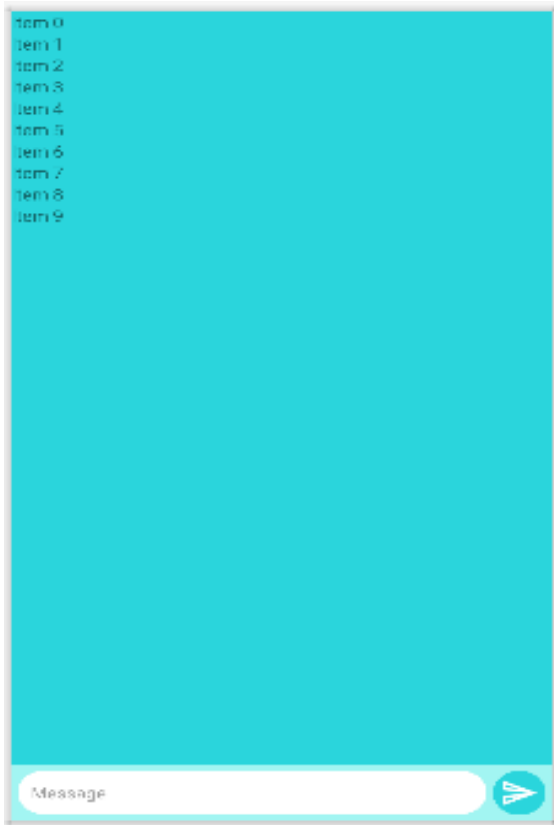
Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

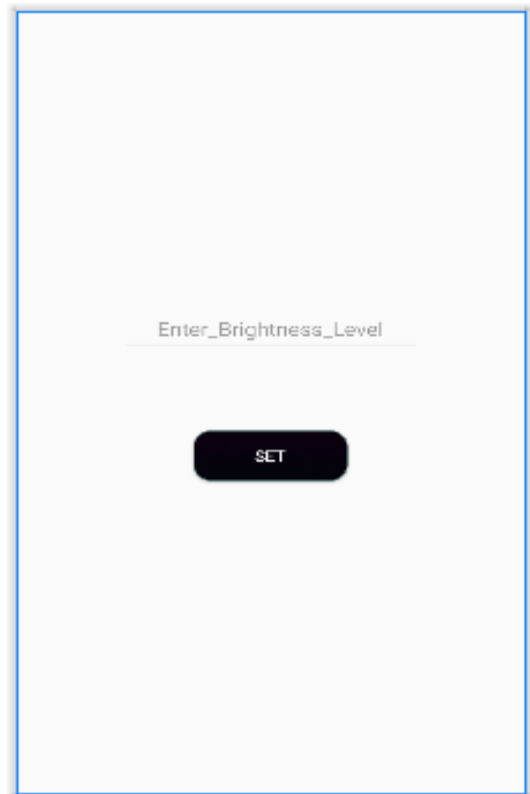
SYSTEM CODING

SCREEN LAYOUTS

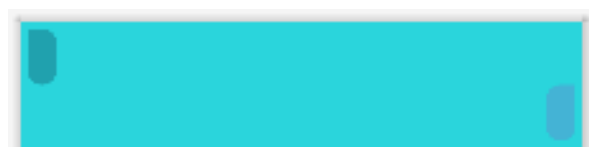
Activity_main.xml



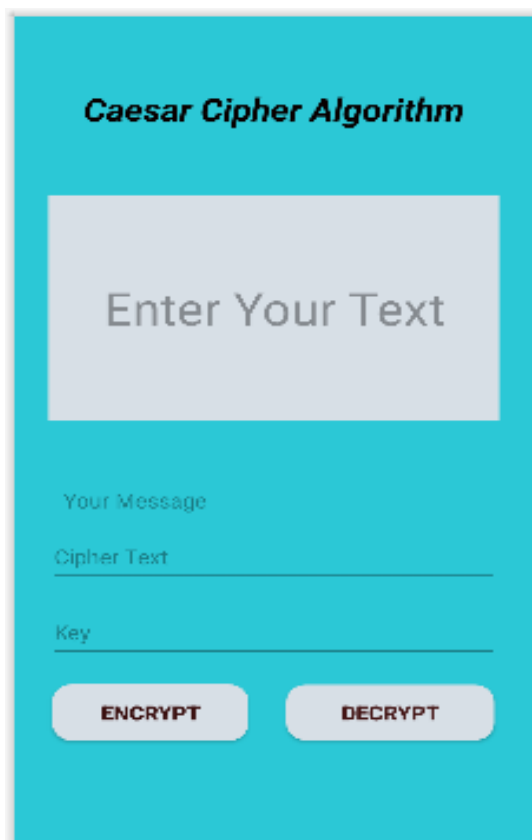
bright_activity.xml



message.xml



caeser_lay_activity.xml



Caesar Cipher Algorithm

Enter Your Text

Your Message

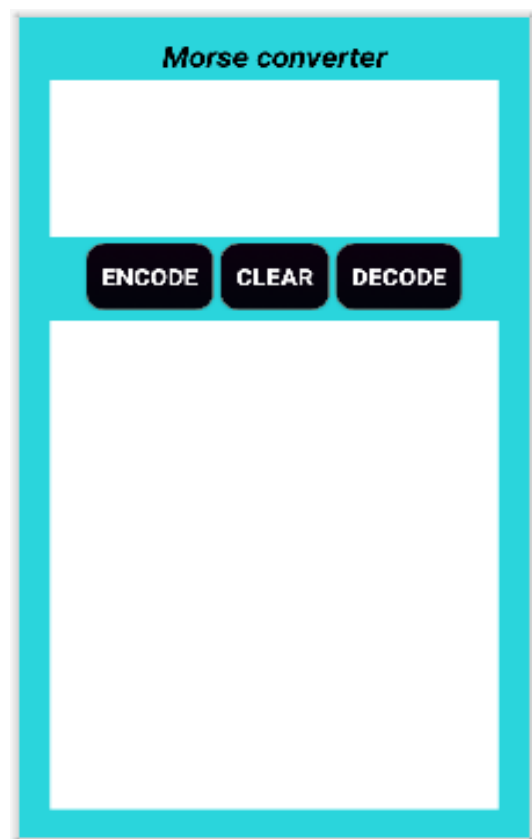
Cipher Text

Key

ENCRYPT DECRYPT

The image shows a mobile application interface for a Caesar Cipher algorithm. It has a teal background. At the top, the title "Caesar Cipher Algorithm" is in bold. Below it is a large light gray box with the text "Enter Your Text". Underneath this box are three input fields: "Your Message", "Cipher Text", and "Key". At the bottom, there are two buttons: "ENCRYPT" and "DECRYPT".

morse_lay_activity.xml

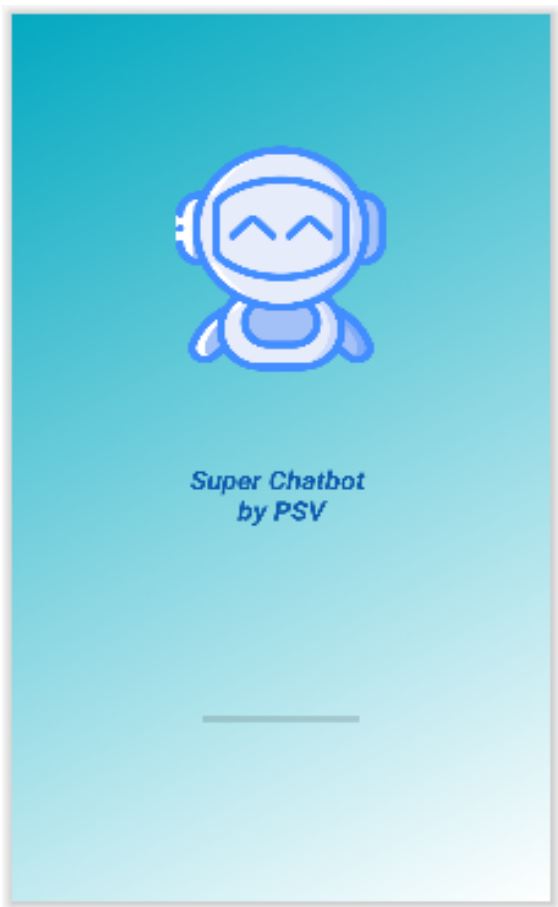


Morse converter

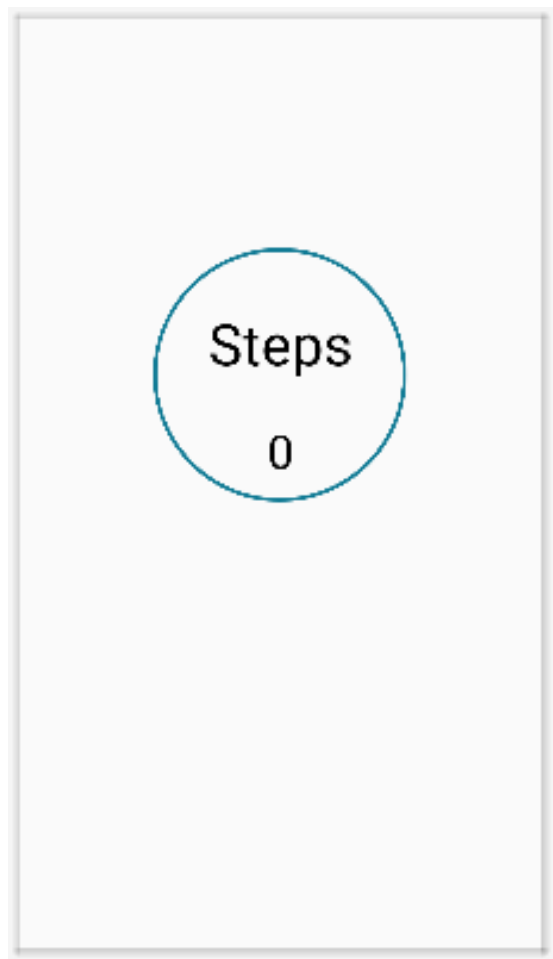
ENCODE CLEAR DECODE

The image shows a mobile application interface for a Morse converter. It has a teal background. At the top, the title "Morse converter" is in bold. Below it is a large white rectangular area for text input. Underneath this area are three buttons: "ENCODE", "CLEAR", and "DECODE". Below the buttons is another large white rectangular area for the output.

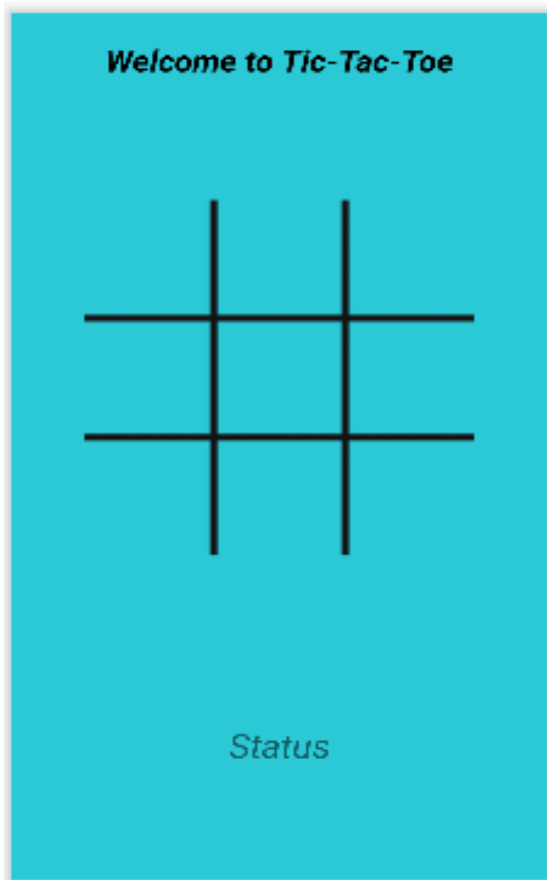
splash_activty.xml



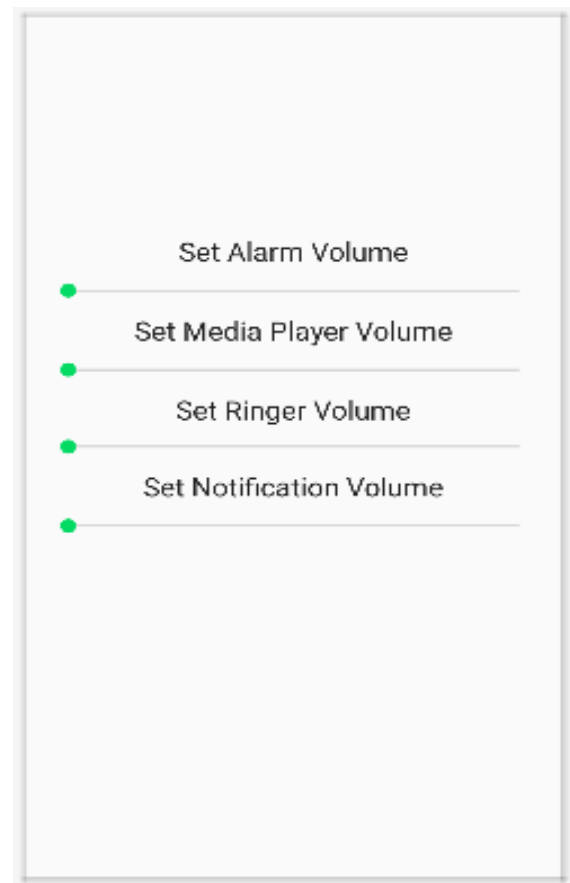
Step_activity.xml



ttt_lay_activity.xml



volume_lay_activity.xml



SAMPLECODE

brightActivity.java

```
package com.example.superchatbot;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import android.content.ContentResolver;
import android.content.Intent;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.provider.Settings;
import android.widget.EditText;
public class brightActivity extends AppCompatActivity {
    private EditText editText;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.bright_activity);
        if(!Settings.System.canWrite(this)){
            Intent intent = new Intent(Settings.ACTION_MANAGE_WRITE_SETTINGS);
            intent.setData(Uri.parse("package:" + this.getPackageName()));
            startActivity(intent);
        }
        editText = findViewById(R.id.editText);

        public void setButton(View view){
            ContentResolver contentResolver = this.getApplicationContext().getContentResolver();
            Settings.System.putInt(contentResolver, Settings.System.SCREEN_BRIGHTNESS,
            Integer.parseInt(editText.getText().toString()));
        }
    }
}
```

caeserActivity.java

```
package com.example.superchatbot;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
public class caeserActivity extends AppCompatActivity {
    private Button encrypt, decrypt;
    private EditText message, cipher, key;
    private TextView screen_output;
```

```

private static final String alphabetString = "abcdefghijklmnopqrstuvwxyz";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(com.example.superchatbot.R.layout.caeser_lay_activity);
    encrypt = findViewById(R.id.btnencrypt);
    decrypt = findViewById(R.id.btndecrypt);
    screen_output = findViewById(R.id.tv1);
    message = findViewById(R.id.inputMessage);
    cipher = findViewById(R.id.ciphEdt);
    key = findViewById(R.id.key_dt);
    encrypt.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            encrypt12(message.getText().toString(),
Integer.parseInt(key.getText().toString()));
        }
    });
    decrypt.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            decrypt12(cipher.getText().toString(),
Integer.parseInt(key.getText().toString()));
        }
    });
}
public void decrypt12(String cipher, int key) {
    screen_output.setText((caeserUtil.decrypt1(cipher, key).toLowerCase()));
}
public String encrypt12(String message, int shiftkey) {
    message = message.toLowerCase();
    String cipherText = "";
    for (int i = 0; i < message.length(); i++) {
        int charPosition = alphabetString.indexOf(message.charAt(i));
        int keyval = (shiftkey + charPosition) % 26;
        char replaceVAL = alphabetString.charAt(keyval);
        cipherText += replaceVAL;
        screen_output.setText(cipherText);
        cipher.setText(cipherText);
    }
    return cipherText;
}
}

```

caeserUtil.java

```

package com.example.superchatbot;
public class caeserUtil {
    private static String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private static int index;
    private static int updated_index;
    private static int final_index;
    private static int index_p_t;
    private static int index_s_t;
    private static String plainTxt;
    private static String cipherTxt;
}

```

```

private static String finalTxt;
public static String encrypt1(String plaintext, int encryptionKey) {
    reset();
    plaintext = plaintext.toUpperCase();
    for (index = 0; index < plaintext.length(); index++) {
        if (plaintext.charAt(index) != ' ') {
            index_p_t_l = alphabet.indexOf(plaintext.charAt(index));
            updated_index = encryptionKey + alphabet.indexOf(plaintext.charAt(index));
            if (updated_index >= alphabet.length()) {
                final_index = updated_index - alphabet.length();
            } else
                final_index = updated_index;
            cipherTxt = alphabet.substring(final_index, final_index + 1);
            finalTxt = finalTxt + cipherTxt;
        }
    }
    return finalTxt;
}

public static String decrypt1(String ciphertext, int decryptionKey) {
    reset();
    ciphertext = ciphertext.toUpperCase();
    for (index = 0; index < ciphertext.length(); index++) {
        if (ciphertext.charAt(index) != ' ') {
            index_p_t_l = alphabet.indexOf(ciphertext.charAt(index));
            index_s_t_l = index_p_t_l;
            updated_index = alphabet.indexOf(ciphertext.charAt(index)) - decryptionKey;
            if (updated_index < 0) {
                final_index = updated_index + alphabet.length();
            } else
                final_index = updated_index;
            plainTxt = alphabet.substring(final_index, final_index + 1);
            finalTxt += plainTxt;
        }
    }
    return finalTxt;
}

private static void reset() {
    finalTxt = "";
}
}

```

getRequest.java

```

package com.example.superchatbot;
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 = "VW3vScNafXh2ge73";
private String brainID = "167598";
private String reply;
private char[] illegalChars = {'#', '<', '>', '+', '%', '!', '"', '&',
    '*', '\\', '\\", '|', '{', '}', '/', '\\', ':', '@'};
private final String[] illegalWords =
{"Opencaeser_cipher", "Opencontrol_volume", "Openmorse_code",
"/Opentic_tac_toe", "OpenStepCount", "OpenBrightnessSettings1", "ShowDateandTime", "Exit",
"/RefreshContent", "OpenYoutube", "Openchrome", "OpenFacebook", "Brightness"};
public getRequest(Context context){
    queue = Volley.newRequestQueue(context);
}
public interface VolleyResponseListener {
    void onError(String message);

    void onResponse(String reply);
}
private String formatStrMessage(String message){
    message = message.replace(" ", "");
    for(String illegalWord : illegalWords){
        message = message.replace(illegalWord, "");
    }
    return message;
}
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 = formatStrMessage(message);
    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);
}
}

```

mainActivty.java

```

package com.example.superchatbot;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import java.util.Date;
import java.text.SimpleDateFormat;
import androidx.annotation.RequiresApi;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;
import android.content.ActivityNotFoundException;
import android.content.ContentResolver;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Log;
import android.view.View;
import android.widget.Toast;
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 recyclerViewAdapter adapter;
    private ImageButton sendButton;
    private EditText msgInput;
    private getRequest request;
    public SwipeRefreshLayout swipeRefreshLayout;
    private long pressedTime;
    private int BrightS;
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActionBar actionBar = getSupportActionBar();
        actionBar.setDisplayHomeAsUpEnabled(false);
        setContentView(R.layout.activity_main);
        request = new getRequest(this);
        swipeRefreshLayout = (SwipeRefreshLayout) findViewById(R.id.swipeRefreshLayout);
        recyclerView = findViewById(R.id.recyclerView);
        RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this);
    }
}

```



```

recyclerView.setLayoutManager(layoutManager);
recyclerView.setItemAnimator(new DefaultItemAnimator());
messages = new ArrayList<>();
adapter = new RecyclerViewAdapter(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);
            switch(message) {
                case "/Open caesar_cipher":
                    Intent CC = new Intent(MainActivity.this, caesarActivity.class);
                    MainActivity.this.startActivity(CC);
                    Toast forCaesar = Toast.makeText(getApplicationContext(),
"Opening Caesar..", Toast.LENGTH_SHORT);
                    forCaesar.show();
                    msgInput.setText("");
                    break;
                case "/Open control_volume":
                    //code to be executed;
                    Intent cV = new Intent(MainActivity.this, volumeActivity.class);
                    MainActivity.this.startActivity(cV);
                    Toast forVolume = Toast.makeText(getApplicationContext(),
"Opening Volume..", Toast.LENGTH_SHORT);
                    forVolume.show();

                    msgInput.setText("");

                    break;
                case "/Open morse_code":
                    Intent mC = new Intent(MainActivity.this, morseActivity.class);
                    MainActivity.this.startActivity(mC);
                    Toast forMorse = Toast.makeText(getApplicationContext(),
"Opening Morse..", Toast.LENGTH_SHORT);
                    forMorse.show();
                    msgInput.setText("");
                    break;
                case "/Open tic_tac_toe":
                    Intent ttt = new Intent(MainActivity.this, tttActivity.class);
                    MainActivity.this.startActivity(ttt);
                    Toast forTtt = Toast.makeText(getApplicationContext(), "Opening
Ttt..", Toast.LENGTH_SHORT);
                    forTtt.show();
                    msgInput.setText("");
                    break;
                case "/Open StepCount":

```

```

Intent SP = new Intent(MainActivity.this, stepActivity.class);
MainActivity.this.startActivity(SP);

Toast stepCount = Toast.makeText(getApplicationContext(),
"Opening Step Counter..", Toast.LENGTH_SHORT);
stepCount.show();
msgInput.setText("");
break;
case "/ShowDateandTime":
Toast DT = Toast.makeText(getApplicationContext(), "Displaying
D&T..", Toast.LENGTH_SHORT);
DT.show();
SimpleDateFormat sdf = new SimpleDateFormat("Date\n'dd-MM-
yyyy '\n\nand\n\nTime\n'HH:mm:ss z");
String currentDateAndTime = sdf.format(new Date());
messages.add(new Message(false, currentDateAndTime));
int newPosition8 = messages.size() - 1;
adapter.notifyItemInserted(newPosition8);
recyclerView.scrollToPosition(newPosition8);
msgInput.setText("");
break;
case "/Open BrightnessSettings1":
Intent BS = new Intent(MainActivity.this, brightActivity.class);
MainActivity.this.startActivity(BS);
Toast Brightness = Toast.makeText(getApplicationContext(),
"Opening Brightness Settings..", Toast.LENGTH_SHORT);
Brightness.show();
msgInput.setText("");
break;
case "/Brightness":
String hw = "Give a number from Range 0 to 255";
messages.add(new Message(false, hw));
int position4 = messages.size() - 1;
adapter.notifyItemInserted(position4);
recyclerView.scrollToPosition(position4);
sendButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
String msg = msgInput.getText().toString();
int BrightS = new Integer(msg).intValue();
if (BrightS >= 0 && BrightS <= 255) {
messages.add(new Message(true, "You typed: " +
BrightS));

int newPos = messages.size() - 1;
adapter.notifyItemInserted(newPos);
recyclerView.scrollToPosition(newPos);
msgInput.setText("");
messages.add(new Message(false, "Setting
brightness at " + BrightS));

int position5 = messages.size() - 1;
adapter.notifyItemInserted(position5);
recyclerView.scrollToPosition(position5);
ContentResolver contentResolver =
getContentResolver();

Settings.System.putInt(contentResolver,

```

```

Settings.System.SCREEN_BRIGHTNESS, Integer.parseInt(String.valueOf(BrightS)));
    shallRunBright();
    return;

    } else {
        String reply2 = "Please give correct Range";

        messages.add(new Message(false, reply2));
        int position5 = messages.size() - 1;
        adapter.notifyItemInserted(position5);
        recyclerView.scrollToPosition(position5);
        msgInput.setText("");
        return;
    }

    }

});
break;

case "/Refresh Content":
    refreshOn();
    msgInput.setText("");
    break;
case "/Exit":
    Toast forEx = Toast.makeText(getApplicationContext(), "Exiting
app", Toast.LENGTH_SHORT);
    forEx.show();
    MainActivity.this.finish();

    System.exit(0);
    break;
default:
    break;
    }
    }
    }
});
swipeRefreshLayout.setOnRefreshListener(
    new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            swipeRefreshLayout.setRefreshing(false);
            refreshOn();
        }
    });
}

@RequiresApi(api = Build.VERSION_CODES.M)
public void shallRunBright() {
    if(!Settings.System.canWrite(this)){
        Intent intent = new Intent(Settings.ACTION_MANAGE_WRITE_SETTINGS);
        intent.setData(Uri.parse("package:" + this.getPackageName()));
        startActivity(intent);
    }
    return;
}

```

```

    }
    public void refreshOn() {
        messages.clear();
        adapter.notifyDataSetChanged();
        Toast rh = Toast.makeText(getApplicationContext(), "Refreshing
Content...", Toast.LENGTH_SHORT);
        rh.show();
    }
    private void getReply(String message) {
        request.getResponse(message, new
com.example.superchatbot.getRequest.VolleyResponseListener() {
            @Override
            public void onError(String message) {
                Log.d("REQUEST ERROR", message);
            }
            @Override
            public void onResponse(String reply) {
                messages.add(new com.example.superchatbot.Message(false, reply));
                int newPosition = messages.size() - 1;
                adapter.notifyItemInserted(newPosition);
                recyclerView.scrollToPosition(newPosition);
            }
        });
    }
    @Override
    public void onBackPressed() {

        if (pressedTime + 2000 > System.currentTimeMillis()) {
            super.onBackPressed();
            finish();
        } else {
            Toast.makeText(getBaseContext(), "Press back again to exit",
Toast.LENGTH_SHORT).show();
        }
        pressedTime = System.currentTimeMillis();
    }
}

```

Message.java

```

package com.example.superchatbot;
public class Message {
    private boolean type;
    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;
    }
}

```

morseActivity.java

```

package com.example.superchatbot;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import androidx.appcompat.app.AppCompatActivity;
public class morseActivity extends AppCompatActivity {
    EditText etinput,
            etoutput;
    Button btnEncode,
            btnDecode,
            btnclear;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.morse_lay_activity);
        etinput = findViewById(R.id.etinput);
        etoutput = findViewById(R.id.etoutput);
        btnDecode = findViewById(R.id.btndecode);
        btnEncode = findViewById(R.id.btnencode);
        btnclear = findViewById(R.id.btnclear);
        final String[] AlphaNumeric = new String[37];
        final String[] AlphaNumeric1 = new String[37];
        AlphaNumeric[0] = "A";
        AlphaNumeric[1] = "B";
        AlphaNumeric[2] = "C";
        AlphaNumeric[3] = "D";
        AlphaNumeric[4] = "E";
        AlphaNumeric[5] = "F";
        AlphaNumeric[6] = "G";
        AlphaNumeric[7] = "H";
        AlphaNumeric[8] = "I";
        AlphaNumeric[9] = "J";
        AlphaNumeric[10] = "K";
        AlphaNumeric[11] = "L";
        AlphaNumeric[12] = "M";
        AlphaNumeric[13] = "N";
        AlphaNumeric[14] = "O";
        AlphaNumeric[15] = "P";
        AlphaNumeric[16] = "Q";
        AlphaNumeric[17] = "R";
        AlphaNumeric[18] = "S";
        AlphaNumeric[19] = "T";
        AlphaNumeric[20] = "U";
        AlphaNumeric[21] = "V";
        AlphaNumeric[22] = "W";
        AlphaNumeric[23] = "X";
        AlphaNumeric[24] = "Y";
    }
}

```

```

AlphaNumeric[25] = "Z";
AlphaNumeric[26] = "0";
AlphaNumeric[27] = "1";
AlphaNumeric[28] = "2";
AlphaNumeric[29] = "3";
AlphaNumeric[30] = "4";
AlphaNumeric[31] = "5";
AlphaNumeric[32] = "6";
AlphaNumeric[33] = "7";
AlphaNumeric[34] = "8";
AlphaNumeric[35] = "9";
AlphaNumeric[36] = " ";
AlphaNumeric1[0] = "-.";
AlphaNumeric1[1] = "-...";
AlphaNumeric1[2] = "-.-.";
AlphaNumeric1[3] = "-..";
AlphaNumeric1[4] = "-.";
AlphaNumeric1[5] = "-...";
AlphaNumeric1[6] = "-.-.";
AlphaNumeric1[7] = "-....";
AlphaNumeric1[8] = "-..";
AlphaNumeric1[9] = "-.-.";
AlphaNumeric1[10] = "-.-.";
AlphaNumeric1[11] = "-.-.";
AlphaNumeric1[12] = "-.-.";
AlphaNumeric1[13] = "-.-.";
AlphaNumeric1[14] = "-.-.";
AlphaNumeric1[15] = "-.-.";
AlphaNumeric1[16] = "-.-.";
AlphaNumeric1[17] = "-.-.";
AlphaNumeric1[18] = "-...";
AlphaNumeric1[19] = "-.-.";
AlphaNumeric1[20] = "-.-.";
AlphaNumeric1[21] = "-.-.";
AlphaNumeric1[22] = "-.-.";
AlphaNumeric1[23] = "-.-.";
AlphaNumeric1[24] = "-.-.";
AlphaNumeric1[25] = "-.-.";
AlphaNumeric1[26] = "-.-.";
AlphaNumeric1[27] = "-.-.";
AlphaNumeric1[28] = "-.-.";
AlphaNumeric1[29] = "-.-.";
AlphaNumeric1[30] = "-.-.";
AlphaNumeric1[31] = "-.-.";
AlphaNumeric1[32] = "-.-.";
AlphaNumeric1[33] = "-.-.";
AlphaNumeric1[34] = "-.-.";
AlphaNumeric1[35] = "-.-.";
AlphaNumeric1[36] = "/";
btnEncode.setOnClickListener(new View.OnClickListener() {@Override
public void onClick(View v) {
    String input = etinput.getText().toString();
    String output = "";
    int l = input.length();
    int i, j;

```

```

        for (i = 0; i < l; i++) {
            String ch = input.substring(i, i + 1);
            for (j = 0; j < 37; j++) {
                if (ch.equalsIgnoreCase(AlphaNumeric[j])) {
                    output = output.concat(AlphaNumeric1[j]).concat(" ");
                }
            }
        }
        etoutput.setText(output);
    }
});
btnClear.setOnClickListener(new View.OnClickListener() {@Override
public void onClick(View v) {
    etinput.setText("");
    etoutput.setText("");
}
});
btnDecode.setOnClickListener(new View.OnClickListener() {@Override
public void onClick(View v) {
    String input1 = etinput.getText().toString();
    String input = input1.concat(" ");
    int l = input.length();
    int i, j, p = 0;
    int pos = 0;
    String letter = "";
    String output = "";
    for (i = 0; i < l; i++) {
        int flag = 0;
        String ch = input.substring(i, i + 1);
        if (ch.equalsIgnoreCase(" ")) {
            p = i;
            letter = input.substring(pos, p);
            pos = p + 1;
            flag = 1;
        }
        String letter1 = letter.trim();
        if (flag == 1) {
            for (j = 0; j <= 36; j++) {
                if (letter1.equalsIgnoreCase(AlphaNumeric1[j])) {
                    output = output.concat(AlphaNumeric[j]);
                    break;
                }
            }
        }
    }
    etoutput.setText(output);
}
});
}
}
}

```

recyclerAdapter.java

```

package com.example.superchatbot;
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<com.example.superchatbot.Message> messages;
    public recyclerAdapter(ArrayList<com.example.superchatbot.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){
            holder.sentLayout.setVisibility(LinearLayout.VISIBLE);
            holder.sentText.setText(message);
            holder.receivedLayout.setVisibility(LinearLayout.GONE);
        }
        else{
            holder.receivedLayout.setVisibility(LinearLayout.VISIBLE);
            holder.receivedText.setText(message);
            holder.sentLayout.setVisibility(LinearLayout.GONE);
        }
    }
    @Override
    public int getItemCount() {
        return messages.size();
    }
}

```


SplashActivity.java

```
package com.example.superchatbot;
import android.animation.ObjectAnimator;
import android.content.Intent;
import android.os.Handler;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ProgressBar;
public class SplashActivity extends AppCompatActivity {
    ProgressBar splashProgress;
    int SPLASH_TIME = 3000;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash_activity);
        splashProgress = findViewById(R.id.splashProgress);
        playProgress();
        new Handler(getMainLooper()).postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent mySuperIntent = new Intent(SplashActivity.this, MainActivity.class);
                startActivity(mySuperIntent);
                finish();
            }
        }, SPLASH_TIME);
    }
    private void playProgress() {
        ObjectAnimator.ofInt(splashProgress, "progress", 100)
            .setDuration(5000)
            .start();
    }
}
```

stepActivity.java

```
package com.example.superchatbot;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.View.OnLongClickListener;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
```



```

    }
}
public void onSensorChanged(@Nullable SensorEvent event) {
    TextView tv_stepsTaken = (TextView)this.findViewById(R.id.tv_stepsTaken);
    if (this.running) {
        Intrinsics.checkNotNull(event);
        this.totalSteps = event.values[0];
        int currentSteps = (int)this.totalSteps - (int)this.previousTotalSteps;
        Intrinsics.checkNotNullExpressionValue(tv_stepsTaken, "tv_stepsTaken");
        tv_stepsTaken.setText((CharSequence)String.valueOf(currentSteps));
    }
}
public final void resetSteps() {
    final ObjectRef tv_stepsTaken = new ObjectRef();
    tv_stepsTaken.element = (TextView)this.findViewById(R.id.tv_stepsTaken);
    ((TextView)tv_stepsTaken.element).setOnClickListener((OnClickListener)(new
OnClickListener() {
    public final void onClick(View it) {
        Toast.makeText((Context)stepActivity.this, (CharSequence)"Long tap to reset
steps", Toast.LENGTH_SHORT).show();
    }
})));
    ((TextView)tv_stepsTaken.element).setOnLongClickListener((OnLongClickListener)(new
OnLongClickListener() {
    public final boolean onLongClick(View it) {
        stepActivity.this.previousTotalSteps = stepActivity.this.totalSteps;
        TextView var10000 = (TextView)tv_stepsTaken.element;
        Intrinsics.checkNotNullExpressionValue(var10000, "tv_stepsTaken");
        var10000.setText((CharSequence)String.valueOf(0));
        stepActivity.this.saveData();
        return true;
    }
})));
}
private final void saveData() {
    SharedPreferences sharedPreferences = this.getSharedPreferences("myPrefs", 0);
    Editor editor = sharedPreferences.edit();
    editor.putFloat("key1", this.previousTotalSteps);
    editor.apply();
}
private final void loadData() {
    SharedPreferences sharedPreferences = this.getSharedPreferences("myPrefs", 0);
    float savedNumber = sharedPreferences.getFloat("key1", 0.0F);
    Log.d("MainActivity", String.valueOf(savedNumber));
    this.previousTotalSteps = savedNumber;
}
public void onAccuracyChanged(@Nullable Sensor sensor, int accuracy) {
}
public static final float access$getPreviousTotalSteps$p(stepActivity $this) {
    return $this.previousTotalSteps;
}
public static final void access$setTotalSteps$p(stepActivity $this, float var1) {
    $this.totalSteps = var1;
}
}

```

tttActivity.java

```
package com.example.superchatbot;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
public class tttActivity extends AppCompatActivity{

    boolean gameActive = true;
    int activePlayer = 0;
    int[] gameState = {2, 2, 2, 2, 2, 2, 2, 2, 2};
    int[][] winPositions = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8},
        {0, 3, 6}, {1, 4, 7}, {2, 5, 8},
        {0, 4, 8}, {2, 4, 6}};
    public static int counter = 0;
    public void playerTap(View view) {
        ImageView img = (ImageView) view;
        int tappedImage = Integer.parseInt(img.getTag().toString());
        if (!gameActive) {
            gameReset(view);
        }
        if (gameState[tappedImage] == 2) {
            counter++;
            if (counter == 9) {
                gameActive = false;
            }
            gameState[tappedImage] = activePlayer;
            img.setTranslationY(-1000f);
            if (activePlayer == 0) {
                img.setImageResource(R.drawable.ttt_x);
                activePlayer = 1;
                TextView status = findViewById(R.id.status);
                status.setText("O's Turn - Tap to play");
            } else {
                img.setImageResource(R.drawable.ttt_o);
                activePlayer = 0;
                TextView status = findViewById(R.id.status);
                status.setText("X's Turn - Tap to play");
            }
            img.animate().translationYBy(1000f).setDuration(300);
        }
        int flag = 0;
        for (int[] winPosition : winPositions) {
            if (gameState[winPosition[0]] == gameState[winPosition[1]] &&
                gameState[winPosition[1]] == gameState[winPosition[2]] &&
                gameState[winPosition[0]] != 2) {
                flag = 1;
                String winnerStr;
                gameActive = false;
                if (gameState[winPosition[0]] == 0) {
```

```

        winnerStr = "X has won";
    } else {
        winnerStr = "O has won";
    }
    TextView status = findViewById(R.id.status);
    status.setText(winnerStr);
}
}
if (counter == 9 && flag == 0) {
    TextView status = findViewById(R.id.status);
    status.setText("Match Draw");
}
}
public void gameReset(View view) {
    gameActive = true;
    activePlayer = 0;
    for (int i = 0; i < gameState.length; i++) {
        gameState[i] = 2;
    }
    ((ImageView) findViewById(R.id.imageView0)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView1)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView2)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView3)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView4)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView5)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView6)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView7)).setImageResource(0);
    ((ImageView) findViewById(R.id.imageView8)).setImageResource(0);
    TextView status = findViewById(R.id.status);
    status.setText("X's Turn - Tap to play");
}
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.ttt_lay_activity);
}
}

```

volumeActivity.java

```

package com.example.superchatbot;
import android.app.ActionBar;
import android.app.Activity;
import android.content.Context;
import android.media.AudioManager;
import android.os.Bundle;
import android.widget.SeekBar;
public class volumeActivity extends Activity {
    SeekBar alarm, mediaPlayer, ringer, notification ;
    AudioManager audioManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(com.example.superchatbot.R.layout.volume_lay_activity);
        alarm = (SeekBar)findViewById(R.id.seekBar1);
        mediaPlayer = (SeekBar)findViewById(R.id.seekBar2);
    }
}

```

```

        ringer = (SeekBar)findViewById(R.id.seekBar3);
        notification = (SeekBar)findViewById(R.id.seekBar4);
        audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
        alarm.setMax(audioManager.getStreamMaxVolume(AudioManager.STREAM_ALARM));
        mediaPlayer.setMax(audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC));
        ringer.setMax(audioManager.getStreamMaxVolume(AudioManager.STREAM_RING));
        notification.setMax(audioManager.getStreamMaxVolume(AudioManager.STREAM_NOTIFICATION));
    };

    alarm.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
            audioManager.setStreamVolume(AudioManager.STREAM_ALARM, i, 0); }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {}
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {} });
    mediaPlayer.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    {
        @Override
        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
            audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, i, 0); }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {}
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {} });
    ringer.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
            audioManager.setStreamVolume(AudioManager.STREAM_RING, i, 0); }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {}
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {}
    });
    notification.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
    {
        @Override
        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
            audioManager.setStreamVolume(AudioManager.STREAM_NOTIFICATION, i,
0);
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        } }); } }

```

CONCLUSION AND FUTURE WORK

Conclusion:

The “SUPER CHATBOT” project developed meets certain objectives of the system. Through this project, it can be concluded that the chatbot should be able to solve any queries of the user within seconds. So that it can reduce the hassle of user and provide a quick response with accuracy.

Future Work:

Removing barriers to entry

Currently chatbots are growing at a rate of 24% annually, and the industry is projected to be a \$1.25 billion market by 2025, according to Grand View Research Inc. Beyond the next five years, however, the future of chatbots relies on widespread adoption of the technology; enterprise use has to go far beyond common industries (technology, finance, healthcare) and become universal.

Linguistic and conversational ability must improve

Chatbots are terrible conversationalists. The anticipated benefits of chatbots often fall short due to their notoriously robotic language, inflexibility and difficulty in understanding the intent and nuance of language. Experts say that stifled customer interactions with chatbots are throttling the success of the technology.

Voice interface

If the future demands advanced chatbots that do more than use scripted, single-turn exchanges, then their method of interface will also have to advance. A voice interface can assist users with disabilities or those who are skeptical of technology, but it also requires another layer of NLP development.

REFERENCES

REFERENCES:

- Alepis, E., & Virvou, M. (2011). Automatic generation of emotions in tutoring agents for affective e learning in medical education. *Expert Systems with Applications*, 38(8): 9840–9847.
- Ashok, G., Brian, C., Mithun, K., Shanu, S., Abhinaya, S., & Bryan, W. (2015). Using Watson for Enhancing Human-Computer Co-Creativity. *AAAI Symposium*: 22–29.
- Avalverde, D. (2019). A Brief History of Chatbots. *Perception, Control, Cognition*. Retrieved March 9, 2019 from: <https://pcc.cs.byu.edu/2018/03/26/a-brief-history-of-chatbots/>
- Ayedoun, E., Hayashi, Y., & Seta, K. (2015). A Conversational Agent to Encourage Willingness to Communicate in the Context of English as a Foreign Language. *Procedia Computer Science*, 60(1): 1433–1442.
- Ben Mimoun, Mohammed Slim, & Poncin, I. (2015). A valued agent: How ECAs affect website customers' satisfaction and behaviors. *Journal of Retailing and Consumer Services*, 26: 70–82.
- Chatbot Magazine (2019). A Visual History of Chatbots. Retrieved March 9, 2019 from: <https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2>
- Colace, F., De Santo, M., Lombardi, M., Pascale, L., Pietrosanto, A. (2018). Chatbot for E-Learning: A Cases Study. *International Journal of Mechanical Engineering and Robotics Research* Vol. 7, No. 5, September.