PROTON- A VIRTUAL ASSISTANT

A PROJECT REPORT

Submitted by

JOSEPH DANIEL.C
ARUNPRASADH. C

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

CHENNAI INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

JULY 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"……PROTON- A VIRTUAL ASSISTANT…..... "** is the bonafide work of "**……JOSEPH DANIEL C & ARUNPRASADH C.. .... "** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
HEAD OF THE DEPARTMENT	**SUPERVISOR**

TABLE OF CONTENTS

ABSTRACT:

People have wanted to talk to computers almost from the moment the first computer was invented. Science fiction is full of computers that can hold a conversation, from HAL 9000 and the Starship Enterprise's computer to Marvin the Paranoid Android and KITT the car. Just a few decades ago, the idea of holding meaningful conversation with a computer seemed futuristic, but the technology to make voice interfaces useful and widely available is already here. Several consumer-level products developed in the last few years have brought inexpensive voice assistants into everyday use, and more features and platforms are being added all the time. Users can do everything from asking simple informational questions to playing music and dialing their phone or turning lights on and off via voice control. Voice assistants are software agents that can interpret human speech and respond via synthesized voices. Apple's Siri, Amazon's Alexa, Microsoft's Cortana, and Google's Assistant are the most popular voice assistants and are embedded in smart phones or dedicated home speakers. Users can ask their assistants questions, control home automation devices and media playback via voice, and manage other basic tasks such as email, to-do lists, and calendars with verbal commands. Usually these voice assistants tend to work with an internet connection but **Proton** does the same work without an internet connection.

LIST OF FIGURES:

LIST OF SAMPLE CODES

CHAPTER 1
INTRODUCTION

1.1 AIM

Nowadays the Mobile Technology is being very famous for the User Experience, because it is very easy to access the applications and services from anywhere of your Geo-location. Android, Apple, Windows, Blackberry, etc. are various famous and commonly used Mobile Operating Systems. All the Operating Systems provides plenty of applications and services for users.

For an instance, the Contacts Applications is used to store the contact details of the user's contact and also helps user to connect a call or send an SMS to other person using the contents stored in this application. We can get similar types of application all around the world via Apple Store, Play Store, etc. All this features gives birth to various kinds of sensors or functionalities to be implemented in the mobile devices.

The Most famous application of iPhone is “SIRI” which helps the end user to communicate end user to mobile with voice and it also responds to the voice commands of the user. Same kind of application is also developed by the Google that is “Google Voice Search” which is used for in Android Phones. But this Application mostly works with Internet Connections. But our Proposed System has capability to work with and without Internet Connectivity.

It’s named as Proton with Voice Recognition Intelligence, which takes the user input in form of voice or text and process it and returns the output in various forms like action to be performed or the search result is dictated to the end user.

1.2 SPEECH RECOGNITION

Speech recognition is an interdisciplinary subfield of computer science and computational linguistics that develops methodologies and technologies that enable the recognition and translation of spoken language into text by computers. It is also known as **automatic speech recognition** (**ASR**), **computer speech recognition** or **speech to text** (**STT**). It incorporates knowledge and research in the computer science, linguistics and computer engineering fields. Some speech recognition systems require "training" (also called "enrollment") where an individual speaker reads text or isolated vocabulary into the system. The system analyzes the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased accuracy. Systems that do not use training are called "speaker-independent" systems. Systems that use training are called "speaker dependent".

Speech recognition applications include voice user interfaces such as voice dialing (e.g. "call home"), call routing (e.g. "I would like to make a collect call"), demotic appliance control, search key words (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), determining speaker characteristics, speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed direct voice input).

The term *voice recognition* or *speaker identification* refers to identifying the speaker, rather than what they are saying. Recognizing the speaker can simplify the task of translating speech in systems that have been trained on a specific person's voice or it can be used to authenticate or verify the identity of a speaker as part of a security process.

From the technology perspective, speech recognition has a long history with several waves of major innovations. Most recently, the field has benefited from advances in deep learning and big data. The advances are evidenced not only by the surge of academic papers published in the field, but more importantly by the worldwide industry adoption of a variety of deep learning methods in designing and deploying speech recognition systems.

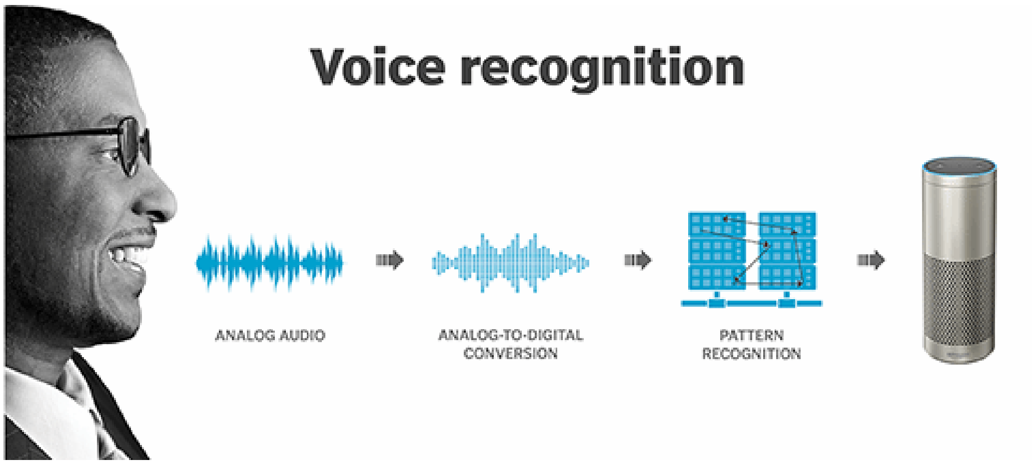

Fig 1.1 Process of Voice Recognition

1.3 MODELS AND ALGORITHMS USED:

(i) Hidden Markov Models(HMM):

Modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models that output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time scale (e.g., 10 milliseconds), speech can be approximated as a stationary process. Speech can be thought of as a Markov model for many stochastic purposes. Another reason why HMMs are popular is that they can be trained automatically and are simple and computationally feasible to use. In speech recognition, the hidden Markov model would output a sequence of *n*-dimensional real-valued vectors (with *n* being a small integer, such as 10), outputting one of these every 10 milliseconds.

(ii) Dynamic time warping (DTW)-based speech recognition:

Dynamic time warping is an algorithm for measuring similarity between two sequences that may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another he or she were walking more quickly, or even if there were accelerations and deceleration during the course of one observation. DTW has been applied to video, audio, and graphics – indeed, any data that can be turned into a linear representation can be analyzed with DTW.

(iii) Neural networks

Neural networks emerged as an attractive acoustic modeling approach in ASR in the late 1980s. Since then, neural networks have been used in many aspects of speech recognition such as phoneme classification, phoneme classification through multi-objective evolutionary algorithms, isolated word recognition, audiovisual speech recognition and audiovisual speaker recognition and speaker adaptation.

(iv) Deep feed forward and recurrent neural networks

Deep Neural Networks and De-noising Auto encoders are also under investigation. A deep feed forward neural network (DNN) is an artificial neural network with multiple hidden layers of units between the input and output layers. Similar to shallow neural networks, DNNs can model complex non-linear relationships. DNN architectures generate compositional models, where extra layers enable composition of features from lower layers, giving a huge learning capacity and thus the potential of modeling complex patterns of speech data.

1.4 KEY FEATURES OF SPEECH RECOGNITION

(i) **Language weighting:** Improve precision by weighting specific words that are spoken frequently (such as product names or industry jargon), beyond terms already in the base vocabulary.

(ii) **Speaker labeling:** Output a transcription that cites or tags each speaker's contributions to a multi-participant conversation.

(iii) **Acoustics training:** Attend to the acoustical side of the business. Train the system to adapt to an acoustic environment (like the ambient noise in a call center) and speaker styles (like voice pitch, volume and pace).

(iv) **Profanity filtering:** Use filters to identify certain words or phrases and sanitize speech output.

1.5 ADVANTAGES OF SPEECH RECOGNITION

(i) **Talking is faster than typing**

Voice commands are a far more efficient tool than typing a message. Advancements are being made in technology to make life easier and voice recognition is being built-in to more devices to help boost convenience and efficiency. Voice recognition software has improved and according to a study at the University of Stanford, it has become significantly faster and more accurate at producing text (through speech-based dictation on a mobile device) than we are at typing on its keyboard.

(ii) Voice recognition boosts productivity levels

This technology is making it possible to access big data instantly, allowing professionals to retrieve important information upon a voice command. As the technology develops, it will become commonplace to ask a question or request data for any specific case or project – taking less time than it would for us to manually search for information.

1.6 DRAWBACKS OF VOICE RECOGNITION

(i) Privacy of voice recorded data

More devices are using VUI technology, which may present more challenges related to data privacy. If a device has this capability, the additional data can get tracked by the manufacturer. There have been concerns in the past that manufacturers would be capable of listening in on private conversations. This area of concern and questioning incentivized action from companies to work on offering better privacy controls for users.

(ii) Error and misinterpretation of words

Not all words are accurately interpreted with voice recognition. It is far easier for a human to decode words and turn it into meaning, than it is for voice recognition software to do so. The software’s limitation of understanding the contextual relation of words may cause disruption to any given task assigned to the software along the way. It may encounter problems with slang words, acronyms or technical words/jargon.

CHAPTER 2

LITERATURE SURVEY

Authors:

- Xin Lei
- Andrew Senior
- Alexander Gruenstein
- Jeffrey Sorensen

Description:

Automatic speech recognition (ASR) is a natural, and increasingly popular, alternative to typing on mobile sevices. Google offers the ability to search by voice [1] on Android, iOS, and Chrome; Apple's iOS devices come with Siri, a conversational assistant. On both Android and iOS devices, users can also speak to fill in any text field where they can type , a capability heavily used to dictate SMS messages and e-mail. A major limitation of these products is that speech recognition is performed on a server. Mobile network connections are often slow or intermittent, and sometimes non-existant. Therefore, in this study, we investigate techniques to build an accurate, small-footprint speech recognition system that can run in real-time on modern mobile devices. Previously, speech recognition on handheld computers and smartphones has been studied in the DARPA sponsored Transtac Program, where speech-to-speech translation systems were developed on the phone [3, 4, 5]. In the Transtac systems, Gaussian mixture models (GMMs) were used to as acoustic models. While the task was a small domain, with limited training data, the memory usage in the resulting systems was moderately high. In this paper, we focus on large vocabulary on-device dictation.
We show that deep neural networks (DNNs) can provide large accuracy improvements over GMM acoustic models, with a significantly smaller footprint. We also demonstrate how memory usage can be significantly reduced by performing on the fly rescoring with a compressed language model during decoding.

Authors:

- Ekenta Elizabeth Odokuma
- Orluchukwu Great Ndidi

Description:

The application will simplify the process of using a computer, by reducing the need for using the mouse or typing in certain cases and since it's a desktop application it makes the process a little more acceptable to most people who are of age, primarily due to the larger display. Furthermore, the whole process of launching programs, playing music, opening websites etc. will go faster thereby saving the user's time and in turn improve their productivity. The application will also enable disabled users (e.g. users who can't use the keyboard/mouse or users without sight) use a computer. Lastly, since the application will contain certain speech synthesis capabilities, it'll be possible for users (including those with bad sight) to have a level of interaction, no matter how little, with it. Despite the various benefits provided by speech recognition, the system is also plagued with limitations. By implication the development of speech recognition applications also inherits these limitations, some them are:

1. Lack of accuracy, and misinterpretations.
2. Time, costs and productivity
3. User accents
4. Background noise interference is also another daunting problem with speech recognition software.We used the Object-oriented analysis and design (OOAD) methodology since the programs were be written entirely in the Java programming language, a very popular Object Oriented Programming Language.

CHAPTER 3
SYSTEM REQUIREMENT

3.1 INTRODUCTION

The system requirement is a technical specification of requirements for the software . It is the first step in the requirements analysis process; that lists the requirements of a particular software system including functional, performance and security requirements.The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

3.2 HARWARE AND SOFTWARE REQUIREMENT

3.2.1 HARDWARE REQUIREMENT

- **Smart phone with Flashlight and Camera**
- **Good quality Microphone(mostly inbuilt)**
- **SIM card for calling purpose**

3.2.2 SOFTWARE REQUIREMENT

- **Android version 7.0 or up(Nougat)**
- **Target version: Android 12**
- **SDK version: API level 24**
- **SDK target version: API level 31**
- **Required permissions**

3.3 TECHNOLOGIES USED

3.3.1 JAVA:

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers *write once, run anywhere* (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub particularly for client-server web applications, with a reported 9 million developers.

3.3.2 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 or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the 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.

3.3.3 XML

Extensible Markup Language (**XML**) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The World Wide Web Consortium's XML 1.0 Specification of 1998 and several other related specifications all of them free open standard define XML.

The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while programmers have developed many application programming interfaces (APIs) to aid the processing of XML data.

3.3.4 ANDROID TEXT-TO-SPEECH API

A ***TextToSpeech*** instance can only be used to synthesize text once it has completed its initialization. Implement the ***TextToSpeech.OnInitListener*** to be notified of the completion of the initialization. When you are done using the ***TextToSpeech*** instance, call the ***shutdown()*** method to release the native resources used by the ***TextToSpeech*** engine. Apps targeting Android 11 that use text-to-speech should declare ***TextToSpeech.Engine.INTENT_ACTION_TTS_SERVICE*** in the queries elements of their manifest:

3.3.5 GSON (Google JSON Library)

Google Gson is a simple Java-based library to serialize Java objects to JSON and vice versa. It is an open-source library developed by Google.

The following points highlight why you should be using this library −

- **Standardized** − Gson is a standardized library that is managed by Google.
- **Efficient** − It is a reliable, fast, and efficient extension to the Java standard library.
- **Optimized** − The library is highly optimized.
- **Support Generics** − It provides extensive support for generics.
- **Supports complex inner classes** − It supports complex objects with deep inheritance hierarchies.

3.3.6 ANDROID SPEECH RECOGNITION API

This class provides access to the speech recognition service. This service allows access to the speech recognizer. Do not instantiate this class directly, instead, call ***SpeechRecognizer-createSpeechRecognizer(Context),*** or ***SpeechRecognizer-createOnDeviceSpeechRecognizer(Context).*** This class's methods must be invoked only from the main application thread. The implementation of this API is likely to stream audio to remote servers to perform speech recognition. As such this API is not intended to be used for continuous recognition, which would consume a significant amount of battery and bandwidth.

3.3.7 VERSION CONTROL GIT

Version control systems are a category of software tools that help a software team manage changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

CHAPTER 4
RESULTS AND CONCLUSION

RESULT:

The voice assistant was completed successfully and was tested successfully by taking the required “test cases”.

CONCLUSION AND FUTURE WORKS:

Proton was created for people who have difficulty typing and navigating through phone and has the capability to interpret voice commands without internet connection. It has various functionalities like network connection and managing different applications in a mobile on just voice commands. The complexity and accuracy of voice recognition has grown exponentially along the years and has a huge potential for the future. There are many possible uses for this technology from Home automation to controlling heavy machinery. However there are some problems with the existing voice assistant products. Privacy and security controls should be addressed before there is an increase in use of these products in the field of confidentiality. There can be several changes made in the future in proton such as Multiple Language Support, More accurate interpretation, Support for more commands and Home Automation and also in the field of Internet Of Things.

APPENDIX

SAMPLE CODING

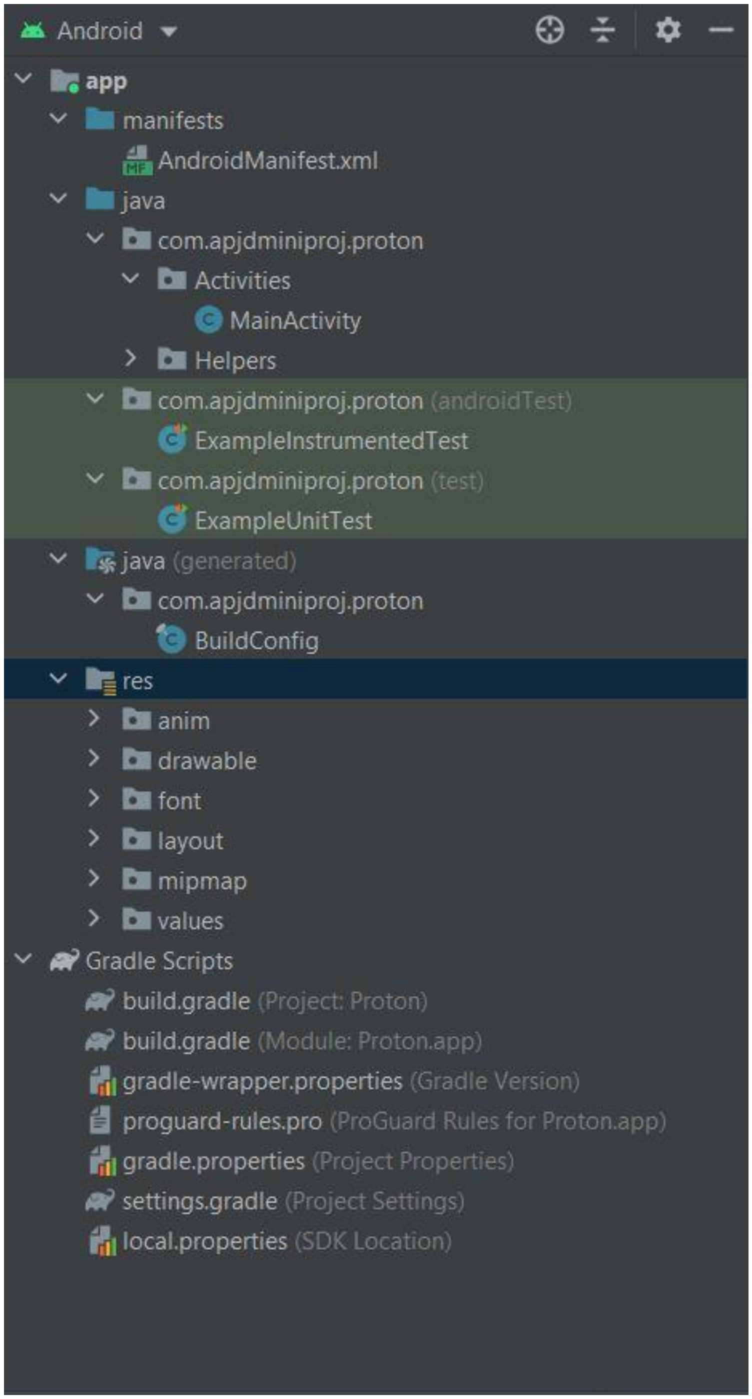

Fig A.1

```
package com.apjdminiproj.proton.Activities;

import android.Manifest;
import android.app.Activity;
import android.app.SearchManager;
import android.app.TimePickerDialog;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.drawable.ColorDrawable;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.provider.AlarmClock;
import android.provider.MediaStore;
import android.provider.Settings;
import android.speech.RecognitionListener;
import android.speech.RecognizerIntent;
import android.speech.SpeechRecognizer;
import android.speech.tts.TextToSpeech;
import android.speech.tts.UtteranceProgressListener;
import android.telephony.SmsManager;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.apjdminiproj.proton.Helpers.Chat;
import com.apjdminiproj.proton.Helpers.ChatAdapter;
import com.apjdminiproj.proton.Helpers.PermissionHelper;
import com.apjdminiproj.proton.Helpers.PreferenceUtils;
import com.apjdminiproj.proton.R;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Locale;
```

```
import java.util.Map;
import java.util.Objects;
import java.util.Random;
import java.util.regex.Pattern;

public class MainActivity extends AppCompatActivity
{
    private String numbersRegex;
    private ImageView sendBtn;
    private EditText cmdInput;
    private String command,smsRec;
    private boolean hasCameraFlash;
    private SpeechRecognizer speechRecognizer;
    private TextToSpeech textToSpeech;
    private Intent speechRecognizerIntent;
    private ChatAdapter chatAdapter;
    private ArrayList<Chat> mChat;
    private RecyclerView recyclerView;
    private LinearLayoutManager linearLayoutManager;
    private boolean waitingForInput;
    private PreferenceUtils preferenceUtils;
    private AlertDialog SpeechRecognitionDialog;
    private ActivityResultLauncher<Intent> launcherForPicture,launcherForSelfie;
    private Intent originalIntent;
    private String[] conversationResponses;
    private Random randomEngine;
    private boolean isTorchOn;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView=findViewById(R.id.recyclerView);
        sendBtn=findViewById(R.id.send_button);
        cmdInput=findViewById(R.id.cmdInput);
        sendBtn.setTag(R.drawable.ic_speech);
        recyclerView.setHasFixedSize(true);
        linearLayoutManager=new LinearLayoutManager(MainActivity.this);
        linearLayoutManager.setStackFromEnd(true);
        recyclerView.setLayoutManager(linearLayoutManager);
        originalIntent=getIntent();
        hasCameraFlash =
getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA_FLASH);
        isTorchOn=false;
        randomEngine=new Random();
        sendBtn.setOnClickListener(v -> {
            if((Integer)sendBtn.getTag()==R.drawable.ic_send) {
                command = cmdInput.getText().toString();
                if (command.isEmpty() || command == null) {
                    receiveMessage("No command was entered to be executed
!",false);
                } else {
                    if (!waitingForInput) {
                        sendMessage(command, false);
                        executeCommand(preprocessCommand(command));
                        cmdInput.setText(null);
                    } else {
                        sendMessage(command, true);
                        cmdInput.setText(null);
                    }
```

```
linearLayoutManager.scrollToPositionWithOffset(chatAdapter.getItemCount() - 1, 0);
                    recyclerView.post(() -> {
                        View target =
linearLayoutManager.findViewByPosition(chatAdapter.getItemCount() - 1);
                        if (target != null) {
                            int offset = recyclerView.getMeasuredHeight() -
target.getMeasuredHeight();

linearLayoutManager.scrollToPositionWithOffset(chatAdapter.getItemCount() - 1,
offset);
                        }
                    });
                }
            }
            else
            {
                textToSpeech.speak("Listening to
you",TextToSpeech.QUEUE_FLUSH,null,"ListeningToYou");
            }
        });
        if(!Settings.System.canWrite(this))
        {
            final AlertDialog.Builder builder2=new
AlertDialog.Builder(MainActivity.this,R.style.AlertDialogTheme);
            final ActivityResultLauncher<Intent> launcher =
registerForActivityResult(
                    new ActivityResultContracts.StartActivityForResult(),
                    result -> {
                        if(result.getResultCode() == Activity.RESULT_OK)
                        {
                            if(Settings.System.canWrite(MainActivity.this))
                                Log.d("ProtectedPermissions","Write Settings
Permission granted");
                            else
                            {
                                StringBuilder message = new StringBuilder("The app
has not been granted the following permission(s):\n\n");

message.append(Manifest.permission.WRITE_SETTINGS);
                                message.append("\n");
                                message.append("\nHence, it cannot function
properly." +
                                        "\nPlease consider granting it these
permissions in the Phone Settings.");
                                final AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this,R.style.AlertDialogTheme);
                                builder.setTitle("Permission Required")
                                        .setMessage(message)
                                        .setPositiveButton("OK", (dialog, id) -> {
                                            dialog.cancel();
                                            MainActivity.this.finish();
                                        });
                                final AlertDialog alert = builder.create();
                                alert.show();
                            }
                            checkPermission();
                        }
                    }
            );
```

```
            builder2.setTitle("Needs Protected Permission").setCancelable(false)
                    .setMessage("This app needs the following permission to run
properly:\n"
                            +Manifest.permission.WRITE_SETTINGS+"\n You will now
be redirected to the Settings to grant the Permission")
                    .setPositiveButton("OK", (dialog,which) -> {
                        Intent intent=new
Intent(Settings.ACTION_MANAGE_WRITE_SETTINGS);
                        intent.setData(Uri.parse("package:"+getPackageName()));
                        launcher.launch(intent);
                    }).setNegativeButton("CANCEL",(dialog,which)->{
                dialog.cancel();
                MainActivity.this.finish();
            });
            final AlertDialog alertd=builder2.create();
            alertd.show();
        }
        else
            checkPermission();
        waitingForInput=false;
        preferenceUtils=PreferenceUtils.getInstance(getApplicationContext());
        if(preferenceUtils.getChatList()==null)
            mChat=new ArrayList<>();
        else
            mChat=preferenceUtils.getChatList();
        chatAdapter=new ChatAdapter(MainActivity.this,mChat);
        recyclerView.setAdapter(chatAdapter);
        cmdInput.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count,
int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int
count)
            {
                if (s == null || s.length() == 0) {

sendBtn.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.ic_speech));
                    sendBtn.setTag(R.drawable.ic_speech);
                }
                else
                    {

sendBtn.setImageDrawable(ContextCompat.getDrawable(getApplicationContext(),
R.drawable.ic_send));
                        sendBtn.setTag(R.drawable.ic_send);
                    }
            }
            @Override
            public void afterTextChanged(Editable s) {
            }
        });
        launcherForPicture=registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), result -> {
            if(result.getResultCode()==RESULT_OK)
            {
```

```
                if(result.getData().getExtras()!=null)
                {
                    Bitmap picture = (Bitmap)
result.getData().getExtras().get("data");
                    startActivity(originalIntent);
                    try {
                        String path =
saveImage(picture,"ProtonCapture"+Calendar.getInstance().getTimeInMillis());
                        receiveMessage("Stored the pic at
"+path.replaceFirst("/","")+" successfully !",false);
                    }
                    catch (IOException e)
                    {
                        e.printStackTrace();
                        receiveMessage("Failed to save the pic ! Try again
!",false);
                    }
                }
                else
                {
                    startActivity(originalIntent);
                    receiveMessage("Failed to save the pic ! Try again !",false);
                }
            }
            else
            {
                startActivity(originalIntent);
                receiveMessage("Failed to capture a pic ! Try again !",false);
            }
        });
        launcherForSelfie=registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), result -> {
            if(result.getResultCode()==RESULT_OK)
            {
                if(result.getData().getExtras()!=null)
                {
                    Bitmap picture = (Bitmap)
result.getData().getExtras().get("data");
                    startActivity(originalIntent);
                    try {
                        String path =
saveImage(picture,"ProtonSelfie"+Calendar.getInstance().getTimeInMillis());
                        receiveMessage("Stored the Selfie pic at
"+path.replaceFirst("/","")+" successfully !",false);
                    }
                    catch (IOException e)
                    {
                        e.printStackTrace();
                        receiveMessage("Failed to save the Selfie pic ! Try again
!",false);
                    }
                }
                else
                {
                    startActivity(originalIntent);
                    receiveMessage("Failed to save the Selfie pic ! Try again
!",false);
                }
            }
            else
```

```
            {
                startActivity(originalIntent);
                receiveMessage("Failed to capture a Selfie pic ! Try again
!",false);
            }
        });

conversationResponses=getResources().getStringArray(R.array.conversations);
    }
    private String preprocessCommand(String cmd)
    {
        cmd=cmd.toLowerCase();
        cmd=cmd.replaceAll("\\p{Punct}","");
        cmd=cmd.replaceAll("\\s","");
        if(cmd.contains("heyproton"))
            cmd=cmd.replace("heyproton","");
        return cmd;
    }
    private boolean executeCommand(String cmd)
    {
        if (cmd.isEmpty() || cmd == null)
            return false;
        if (cmd.contains("callthisnumber"))
        {
            cmd=cmd.replace("callthisnumber","");
            Pattern numberPat = Pattern.compile(numbersRegex);
            if(numberPat.matcher(cmd).matches())
            {
                receiveMessage("Calling " + cmd,false);
                cmd = "tel:+91" + cmd.trim();
                Uri uri = Uri.parse(cmd);
                Intent intent = new Intent(Intent.ACTION_CALL);
                intent.setData(uri);
                startActivity(intent);
                return true;
            }
            else
                {
                    receiveMessage("Invalid Phone Number",false);
                    return false;
                }
        }
        else if (cmd.contains("googlethis"))
        {
            command=command.substring(command.toLowerCase().indexOf("google
this"));
            String
searchQuery=command.substring(command.toLowerCase().indexOf("google
this")+11).trim();
            if(searchQuery.length()==0||searchQuery==null)
            {
                receiveMessage("Invalid search term",false);
                return false;
            }
            else
                { Intent intent=new Intent(Intent.ACTION_WEB_SEARCH);
                intent.putExtra(SearchManager.QUERY, searchQuery);
                receiveMessage("Opening Google App", false);
                startActivity(intent);
                return true;
```

```
            }
        }
        else if(cmd.contains("playthis"))
        {
            command=command.substring(command.toLowerCase().indexOf("play this"));
            String
searchQuery=command.substring(command.toLowerCase().indexOf("play
this")+9).trim();
            if(searchQuery.length()==0||searchQuery==null)
            {
                receiveMessage("Incomplete command",false);
                return false;
            }
            else
            {
                Intent intent=new
Intent(Intent.ACTION_VIEW,Uri.parse("https://www.youtube.com/results?search_query=
"+searchQuery));
                receiveMessage("Opening YouTube App", false);
                startActivity(intent);
                return true;
            }
        }
        else if(cmd.contains("ping"))
        {
            receiveMessage("Enter the number of the recipient",true);
        }
        else if(cmd.contains("takeaselfie"))
        {
            Intent intent=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            intent.putExtra("android.intent.extras.CAMERA_FACING", 1);
            receiveMessage("Opening Selfie Cam ! Say Cheese !",false);
            launcherForSelfie.launch(intent);
            return true;
        }
        else if(cmd.contains("takeapicture"))
        {
            Intent intent=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            intent.putExtra("android.intent.extras.CAMERA_FACING", 0);
            receiveMessage("Opening Back Cam !",false);
            launcherForPicture.launch(intent);
            return true;
        }
        else if(cmd.contains("turnonflashlight"))
        {
            if(!hasCameraFlash)
            {
                receiveMessage("This phone does not support Flashlight",false);
                return false;
            }
            CameraManager
cameraManager=(CameraManager)getSystemService(Context.CAMERA_SERVICE);
            try
            {
                if(isTorchOn)
                    receiveMessage("The FlashLight has already been turned ON
!",false);
                else
                    {
                    String cameraId = cameraManager.getCameraIdList()[0];
```

```
                    cameraManager.setTorchMode(cameraId, true);
                    receiveMessage("Turned ON FlashLight successfully !", false);
                    isTorchOn = true;
                }
                return true;
            }
            catch (CameraAccessException e)
            {
                receiveMessage("Unable to Access Camera to turn ON
Flashlight",false);
                return false;
            }
        }
        else if(cmd.contains("turnoffflashlight"))
            {
                CameraManager
cameraManager=(CameraManager)getSystemService(Context.CAMERA_SERVICE);
                try
                {
                    if(!isTorchOn)
                        receiveMessage("The FlashLight has already been turned OFF
!",false);
                    else
                        {
                        String cameraId = cameraManager.getCameraIdList()[0];
                        cameraManager.setTorchMode(cameraId, false);
                        receiveMessage("Turned OFF FlashLight successfully !",
false);
                        isTorchOn=false;
                        return true;
                    }
                }
                catch (CameraAccessException e)
                {
                    receiveMessage("Unable to Access Camera to turn OFF
Flashlight",false);
                    return false;
                }
            }
        else if(cmd.contains("increasebrightness"))
        {
            int
currentBrightness=Settings.System.getInt(getContentResolver(),Settings.System.SCRE
EN_BRIGHTNESS,0);
            if(currentBrightness+30<=255) {
                Settings.System.putInt(getContentResolver(),
Settings.System.SCREEN_BRIGHTNESS, currentBrightness + 30);
                receiveMessage("Increased Brightness successfully !",false);
            }else if(currentBrightness==255)
                receiveMessage("The Phone is already set with Max
Brightness",false);
            else
                {
                Settings.System.putInt(getContentResolver(),
Settings.System.SCREEN_BRIGHTNESS,
                        currentBrightness + (255 - currentBrightness));
                receiveMessage("Increased Brightness successfully !",false);
            }
            return true;
        }
```

```
        else if(cmd.contains("decreasebrightness"))
        {
            int
currentBrightness=Settings.System.getInt(getContentResolver(),Settings.System.SCRE
EN_BRIGHTNESS,255);
            if(currentBrightness-30>=0)
            {
                Settings.System.putInt(getContentResolver(),
Settings.System.SCREEN_BRIGHTNESS, currentBrightness - 30);
                receiveMessage("Decreased Brightness successfully !",false);
            }
            else if(currentBrightness==0)
                receiveMessage("The Phone is already set with Min
Brightness",false);
            else
            {

Settings.System.putInt(getContentResolver(),Settings.System.SCREEN_BRIGHTNESS,
                        currentBrightness+(currentBrightness-30));
                receiveMessage("Decreased Brightness successfully !",false);
            }
            return true;
        }
        else if(cmd.contains("cleartheconversation"))
        {
            if(mChat!=null)
            {
                mChat.clear();
                receiveMessage("The Preserved Messages were cleared successfully
!",false);
                return true;
            }
        }
        else if(cmd.contains("setanalarm"))
        {
            receiveMessage("Choose when to set an alarm",false);
            Calendar calendar=Calendar.getInstance();
            TimePickerDialog setAlarmTimePicker = new
TimePickerDialog(MainActivity.this, R.style.TimePickerDialogTheme, (view,
hourOfDay, minute) -> {
                Calendar calendar1 =Calendar.getInstance();
                calendar1.set(Calendar.HOUR_OF_DAY,hourOfDay);
                calendar1.set(Calendar.MINUTE,minute);
                calendar1.set(Calendar.SECOND,0);
                Intent intent=new Intent(AlarmClock.ACTION_SET_ALARM);
                intent.putExtra(AlarmClock.EXTRA_HOUR,hourOfDay);
                intent.putExtra(AlarmClock.EXTRA_MINUTES,minute);
                intent.putExtra(AlarmClock.EXTRA_SKIP_UI,true);
                startActivity(intent);
                receiveMessage("Alarm set at "+

DateFormat.getTimeInstance(DateFormat.SHORT).format(calendar1.getTime())+"
successfully !",false);

},calendar.get(Calendar.HOUR_OF_DAY),calendar.get(Calendar.MINUTE),true);
            setAlarmTimePicker.setCancelable(false);
            setAlarmTimePicker.setCanceledOnTouchOutside(false);
            setAlarmTimePicker.setOnCancelListener(dialog -> receiveMessage("Alarm
Setting Task cancelled successfully !",false));
            setAlarmTimePicker.show();
```

```
        }
        else if(cmd.contains("deleteanalarm"))
        {
            receiveMessage("Opening Clock App where you can delete an
alarm",false);
            Intent intent=new Intent(AlarmClock.ACTION_DISMISS_ALARM);
            startActivity(intent);
        }
        else if(cmd.contains("openphonesettings"))
        {
            Intent intent=new Intent(Settings.ACTION_SETTINGS);
            receiveMessage("Opening Phone Settings",false);
            startActivity(intent);
            return true;
        }
        else if(cmd.contains("howareyou")||cmd.contains("whatsup")
                ||cmd.contains("howsitgoing")||cmd.contains("hopeyouaredoingwell")
                ||cmd.contains("howisitgoing")||cmd.contains("hopeyouredoingwell")
                ||cmd.contains("whatisup"))
        {
            int randomReplyIndex=randomEngine.nextInt(3);
            receiveMessage(conversationResponses[randomReplyIndex],false);
            return true;
        }
        else if(cmd.contains("ilikeyou")||cmd.contains("iloveyou")

||cmd.contains("ilikeyourservice")||cmd.contains("iloveyourservice"))
        {
            int randomReplyIndex=randomEngine.nextInt(3)+3;
            receiveMessage(conversationResponses[randomReplyIndex],false);
            return true;
        }
        else if(cmd.contains("goodday")||cmd.contains("haveagoodday")

||cmd.contains("haveaniceday")||cmd.contains("niceday")||cmd.contains("hello"))
        {
            int randomReplyIndex=randomEngine.nextInt(2)+6;
            receiveMessage(conversationResponses[randomReplyIndex],false);
            return true;
        }
        else if(cmd.contains("goodmorning")||cmd.contains("haveagoodmorning")
                ||cmd.contains("haveanicemorning")||cmd.contains("nicemorning"))
        {
            int randomReplyIndex=randomEngine.nextInt(2)+8;
            receiveMessage(conversationResponses[randomReplyIndex],false);
            return true;
        }
        else if(cmd.contains("goodafternoon")||cmd.contains("haveagoodafternoon")

||cmd.contains("haveaniceafternoon")||cmd.contains("niceafternoon"))
        {
            int randomReplyIndex=randomEngine.nextInt(2)+10;
            receiveMessage(conversationResponses[randomReplyIndex],false);
            return true;
        }
        else if(cmd.contains("goodevening")||cmd.contains("haveagoodevening")
                ||cmd.contains("haveaniceevening")||cmd.contains("niceevening"))
        {
            int randomReplyIndex=randomEngine.nextInt(2)+12;
            receiveMessage(conversationResponses[randomReplyIndex],false);
```

```
            return true;
        }
        else if(cmd.contains("goodnight")||cmd.contains("goodnightsweetdreams"))
        {
            int randomReplyIndex=randomEngine.nextInt(2)+14;
            receiveMessage(conversationResponses[randomReplyIndex],false);
            return true;
        }
        else if(cmd.contains("goodbye")||cmd.contains("goodbyefornow")
                ||cmd.contains("bye")||cmd.contains("byefornow"))
        {
            int randomReplyIndex=randomEngine.nextInt(2)+16;
            receiveMessage(conversationResponses[randomReplyIndex],false);
            Handler handler=new Handler(getMainLooper());
            handler.postDelayed(() -> finish(),3000);
            return true;
        }
        else if(cmd.contains("whocreatedyou"))
        {
            receiveMessage(conversationResponses[18],false);
            return true;
        }
        else if(cmd.contains("canyouhelpme")||cmd.contains("howcanyouhelpme")

||cmd.contains("ineedyourhelp")||cmd.contains("iwantyourassistance")||cmd.equals("
heyproton"))
        {
            receiveMessage(conversationResponses[19],false);
            return true;
        }
        else
        {
            receiveMessage("Invalid Command/Message",false);
            return false;
        }
        command=null;
        return true;
    }
    private String saveImage(Bitmap bitmap,String name) throws IOException
    {
        OutputStream fos;
        String path;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
            ContentResolver resolver = getContentResolver();
            ContentValues contentValues = new ContentValues();
            contentValues.put(MediaStore.MediaColumns.DISPLAY_NAME, name +
".jpg");
            contentValues.put(MediaStore.MediaColumns.MIME_TYPE, "image/jpg");
            contentValues.put(MediaStore.MediaColumns.RELATIVE_PATH,
Environment.DIRECTORY_PICTURES);
            Uri imageUri =
resolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, contentValues);
            fos = resolver.openOutputStream(Objects.requireNonNull(imageUri));
            path=imageUri.getPath();
        } else {
            String imagesDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).toSt
ring();
            File image = new File(imagesDir,name+".jpg");
            fos = new FileOutputStream(image);
```

```
            path=image.getAbsolutePath();
        }
        bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fos);
        Objects.requireNonNull(fos).close();
        return path;
    }
    private void sendSMS(String recipient)
    {
        smsRec=recipient;
        receiveMessage("Enter the Message to be sent",true);
    }
    private void sendSMS(String recipient,String message)
    {
        if(message.trim().toLowerCase().contains("stop pinging"))
        {
            receiveMessage("SMS Task cancelled successfully !",false);
        }
        else
            {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(recipient, null, message, null, null);
            receiveMessage("Message: " + message + "\nsent to " + recipient + "
successfully !", false);
        }
        waitingForInput=false;
    }
    private void checkPermission()
    {
        PermissionHelper permissionHelper=new PermissionHelper();
        if(Build.VERSION.SDK_INT<=Build.VERSION_CODES.P) {
            permissionHelper.checkAndRequestPermissions(this,
Manifest.permission.SEND_SMS,
                    Manifest.permission.CALL_PHONE, Manifest.permission.CAMERA,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.RECORD_AUDIO);
        }
        else
        {
            permissionHelper.checkAndRequestPermissions(this,
Manifest.permission.SEND_SMS,
                    Manifest.permission.CALL_PHONE, Manifest.permission.CAMERA,
                    Manifest.permission.RECORD_AUDIO);
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions,
int[] grantResults)
    {
        super.onRequestPermissionsResult(requestCode,permissions,grantResults);
        if (requestCode == 100) {
            Map<String, Integer> perms = new HashMap<>();
            for (String permission : permissions) {
                perms.put(permission, PackageManager.PERMISSION_GRANTED);
            }
            if (grantResults.length > 0) {
                for (int i = 0; i < permissions.length; i++)
                    perms.put(permissions[i], grantResults[i]);

                boolean allPermissionsGranted = true;
                for (String permission1 : permissions) {
```

```
                    allPermissionsGranted = allPermissionsGranted &&
(perms.get(permission1) == PackageManager.PERMISSION_GRANTED);
                }
                if (allPermissionsGranted) {
                    Log.d(PermissionHelper.class.getSimpleName(),
"onRequestPermissionsResult: all permissions granted");
                } else {
                    for (String permission2 : perms.keySet())
                        if (perms.get(permission2) ==
PackageManager.PERMISSION_GRANTED)
                            perms.remove(permission2);

                    StringBuilder message = new StringBuilder("The app has not
been granted the following permission(s):\n\n");
                    for (String permission : perms.keySet()) {
                        message.append(permission);
                        message.append("\n");
                    }
                    message.append("\nHence, it cannot function properly." +
                            "\nPlease consider granting it these permissions in
the Phone Settings.");

                    final AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this, R.style.AlertDialogTheme);
                    builder.setTitle("Permission Required")
                            .setMessage(message)
                            .setPositiveButton("OK", (dialog, id) -> {
                                dialog.cancel();
                                MainActivity.this.finish();
                            });
                    final AlertDialog alert = builder.create();
                    alert.show();
                }
            }
        }
    }
    private void sendMessage(String message,boolean givingInput)
    {
        Chat newMsg = new Chat(message, Chat.MSG_TYPE_RIGHT);
        mChat.add(newMsg);
        chatAdapter.notifyDataSetChanged();
        if(givingInput)
        {
            String current=mChat.get(mChat.size()-1).getMessage();
            String preprocessedCurrent=preprocessCommand(current);
            String previous=mChat.get(mChat.size()-2).getMessage();
            if(previous.equals("Enter the number of the recipient"))
            {
                Pattern num=Pattern.compile(numbersRegex);
                if(num.matcher(preprocessedCurrent).matches())
                {
                    sendSMS(preprocessedCurrent);
                }
                else
                {
                    receiveMessage("Invalid Recipient",false);
                }
            }
            else if(previous.equals("Enter the Message to be sent"))
            {
```

```
                sendSMS(smsRec,current);
            }
        }
    }
    private void receiveMessage(String message,boolean needsInput)
    {

textToSpeech.speak(message,TextToSpeech.QUEUE_FLUSH,null,"receiveMessage:"+message
);
        Chat newMsg = new Chat(message, Chat.MSG_TYPE_LEFT);
        mChat.add(newMsg);
        chatAdapter.notifyDataSetChanged();
        scrollToRecentMessage();
        if(needsInput)
        {
            waitingForInput=true;
        }
    }
    @Override
    protected void onResume()
    {
        super.onResume();
        if (textToSpeech == null)
        {
            textToSpeech = new TextToSpeech(MainActivity.this, status ->
            {
                if (status == TextToSpeech.SUCCESS)
                {
                    textToSpeech.setLanguage(Locale.UK);
                }
            });
            textToSpeech.setOnUtteranceProgressListener(new
UtteranceProgressListener() {
                @Override
                public void onStart(String utteranceId)
                {

                }

                @Override
                public void onDone(String utteranceId)
                {
                    if(utteranceId.equals("ListeningToYou"))
                    {
                        runOnUiThread(() -> showSpeechRecognitionDialog());
                    }
                }

                @Override
                public void onError(String utteranceId)
                {

                }

                @Override
                public void onError(String utteranceId,int errorCode)
                {

                }
            });
```

```
        }
        if (speechRecognizer == null)
        {
            if(SpeechRecognizer.isRecognitionAvailable(MainActivity.this))
            {
                speechRecognizer =
SpeechRecognizer.createSpeechRecognizer(MainActivity.this);
                speechRecognizerIntent = new
Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
                speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE,
Locale.getDefault());

speechRecognizerIntent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE,getPackageN
ame());
                speechRecognizer.setRecognitionListener(new RecognitionListener()
                {
                    @Override
                    public void onReadyForSpeech(Bundle params)
                    {

                    }

                    @Override
                    public void onBeginningOfSpeech()
                    {

                    }

                    @Override
                    public void onRmsChanged(float rmsdB)
                    {

                    }

                    @Override
                    public void onBufferReceived(byte[] buffer)
                    {

                    }

                    @Override
                    public void onEndOfSpeech() {

                    }

                    @Override
                    public void onError(int error)
                    {

if(error==SpeechRecognizer.ERROR_SPEECH_TIMEOUT||error==SpeechRecognizer.ERROR_NO_
MATCH) {
                            SpeechRecognitionDialog.dismiss();
                            receiveMessage( "No response ! Try again !",false);
                        }
                    }

                    @Override
```

```
                    public void onResults(Bundle results)
                    {
                        ArrayList<String> matches =
results.getStringArrayList(SpeechRecognizer.RESULTS_RECOGNITION);
                        command = matches.get(0);
                        Log.d("recognitionResults", command);
                        SpeechRecognitionDialog.dismiss();
                        if(waitingForInput)
                        {
                            sendMessage(command,true);
                        }
                        else
                            {
                                sendMessage(command, false);
                                executeCommand(preprocessCommand(command));
                            }
                            scrollToRecentMessage();
                    }
                    @Override
                    public void onPartialResults(Bundle partialResults) {

                    }

                    @Override
                    public void onEvent(int eventType, Bundle params) {

                    }
                });
            }
            else
            {
                receiveMessage("Speech Recognition is unavailable in this Device!
You can't use Voice Mode!"
                        ,false);
                SpeechRecognitionDialog.dismiss();
            }
        }
        if(preferenceUtils.getIsFirstTime())
        {
            receiveMessage(conversationResponses[19], false);
            preferenceUtils.setIsFirstTime(false);
        }
    }

    private void scrollToRecentMessage()
    {
        linearLayoutManager.scrollToPositionWithOffset(chatAdapter.getItemCount()
- 1, 0);
        recyclerView.post(() -> {
            View target =
linearLayoutManager.findViewByPosition(chatAdapter.getItemCount() - 1);
            if (target != null) {
                int offset = recyclerView.getMeasuredHeight() -
target.getMeasuredHeight();

linearLayoutManager.scrollToPositionWithOffset(chatAdapter.getItemCount() - 1,
offset);
            }
        });
    }
```

```
    private void showSpeechRecognitionDialog()
    {
        if(SpeechRecognitionDialog==null)
        {
            AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
            View view =
LayoutInflater.from(this).inflate(R.layout.layout_speechrecognition,findViewById(R
.id.speechRecognitionLayout));
            builder.setView(view);
            SpeechRecognitionDialog = builder.create();
            SpeechRecognitionDialog.setCancelable(false);
            SpeechRecognitionDialog.setOnShowListener(dialog ->
                    speechRecognizer.startListening(speechRecognizerIntent));
            SpeechRecognitionDialog.setOnDismissListener(dialog ->
speechRecognizer.stopListening());
            if (SpeechRecognitionDialog.getWindow() != null)
            {
                SpeechRecognitionDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(0));
            }
            view.findViewById(R.id.cancelSpeechBut).setOnClickListener(v->{
                SpeechRecognitionDialog.dismiss();
                textToSpeech.stop();
            });
        }
        SpeechRecognitionDialog.show();
    }
    @Override
    protected void onPause()
    {
        if(textToSpeech!=null)
        {
            textToSpeech.stop();
        }
        if(speechRecognizer!=null)
        {
            speechRecognizer.stopListening();
        }
        if(mChat!=null)
        {
            Iterator<Chat> iterator = mChat.iterator();
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss", Locale.getDefault());
            String todayS = formatter.format(new Date());
            Date today;
            try
            {
                today = formatter.parse(todayS);
            } catch (Exception e) {
                    e.printStackTrace();
                    return;
                }
                if (today == null)
                    return;
                while (iterator.hasNext()) {
                    Chat c = iterator.next();
                    Date date;
                    try {
```

```
                        date = formatter.parse(c.getDateOfSending());
                    } catch (Exception e) {
                        e.printStackTrace();
                        break;
                    }
                    if (date == null)
                        continue;
                    long diffTime = today.getTime() - date.getTime();
                    long diffDays = (diffTime / (1000 * 60 * 60 * 24)) % 365;
                    if (diffDays > 7)
                        iterator.remove();
                }
                preferenceUtils.setChatList(mChat);
            }
            super.onPause();
        }
        @Override
        protected void onDestroy()
        {
            if(textToSpeech!=null)
                textToSpeech.shutdown();
            if(speechRecognizer!=null)
                speechRecognizer.destroy();
            super.onDestroy();
        }
}
```

MainActivity.java file

SCREENSHOTS

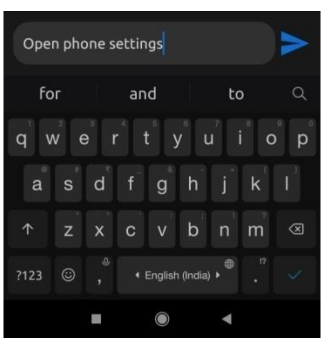

Fig A.2

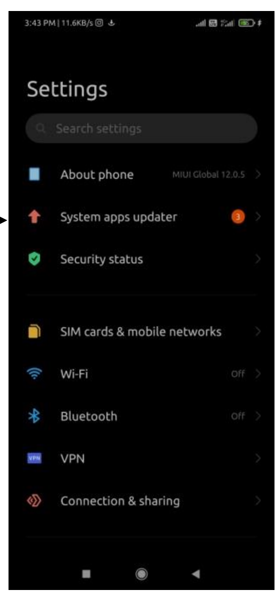

Fig A.3

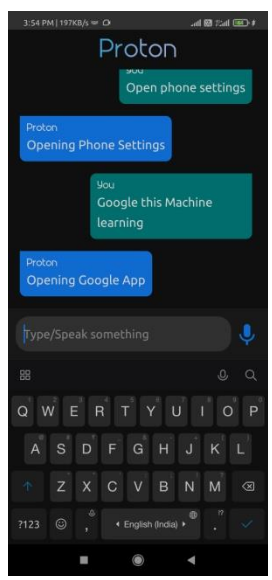

Fig A.4

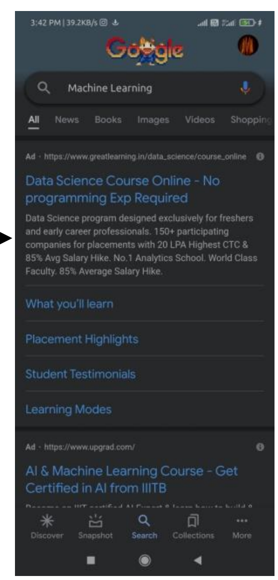

Fig A.5

REFERENCES

1) **Android Studio** provides the fastest tools for building apps on every type of Android device-Link: https://developer.android.com/studio
2) **Accurate and compact large vocabulary speech recognition on mobile devices**, in INTERSPEECH. 2013, pp. 662–665, ISCA.
3) Matthew B. Hoy (2018) **Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants**, Medical Reference Services Quarterly, 37:1, 81-88, DOI:10.1080/02763869.2018.1404391
4) Odokuma, Ekenta & Great, Orluchukwu. (2016). **Development of a voice-controlled personal assistant for the elderly and disabled**. International Research Journal of Computer Science (IRJCS) 2393-9842. 04.
5) **Wikipedia** A place for all information Link- https://en.wikipedia.org\
6) **Voice Assistants and Smart Speakers in Everyday Life and in Education** George TERZOPOULOS, Maya SATRATZEMI Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
7) **AI Based Voice Assistant Using Python**, International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.6, Issue 2, page no.506-509, February-2019