NAME:K.SAMATHA

PROJECT NAME:BOT ASSISTANT

ABSTARCT:

Bot assistant, or conversational interfaces as they are also known, present a new way for individuals to interact with computer systems. Traditionally, to get a question answered by a software program involved using a search engine, or filling out a form. A Bot assistant allows a user to simply ask questions in the same manner that they would address a human. The most well known bots currently are voice chatbots: Alexa and Siri. However, chatbots are currently being adopted at a high rate on computer chat platforms.

The technology at the core of the rise of the bot assistant is natural language processing (“NLP”). Recent advances in machine learning have greatly improved the accuracy and effectiveness of natural language processing, making chatbots a viable option for many organizations. This improvement in NLP is firing a great deal of additional research which should lead to continued improvement in the effectiveness of chatbots in the years to come.

A simple bot can be created by loading an FAQ (frequently asked questions) into chatbot software. The functionality of the chatbot can be improved by integrating it into the organization’s enterprise software, allowing more personal questions to be answered, like“What is my balance?”, or “What is the status of my order?”.

Most commercial bots are dependent on platforms created by the technology giants for their natural language processing. These include Amazon Lex, Microsoft Cognitive Services, Google Cloud Natural Language API, Facebook DeepText, and IBM Watson. Platforms where chatbots are deployed include Facebook Messenger, Skype, and Slack, among many others.

Applications

A bot can be used anywhere a human is interacting with a computer system. These are the areas where the fastest adoption is occurring:

Customer Service — A bot can be used as an "assistant" to a live agent, increasing the agent's efficiency. When trained, they can also provide service when the call centre is closed, or eventually even act as an independent agent, if desired.

Sales/Marketing/Branding — bots can be used for sales qualification, ecommerce, promotional campaigns, or as a branding vehicle.

Human Resources — An HR bot can help with frequently asked questions ("how many vacation days do I have left?") and can act as an onboarding assistant.

Benefits

· Economically offer 24/7 Service

· Improve Customer Satisfaction

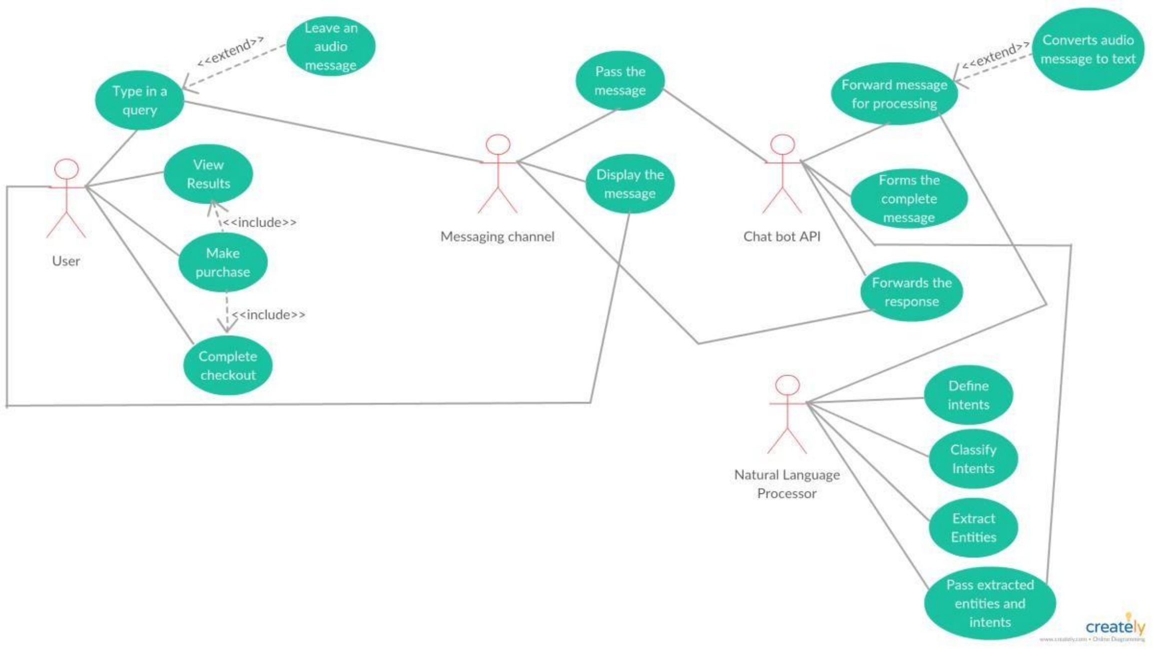
· Reach a Younger Demographic

· Reduce Costs

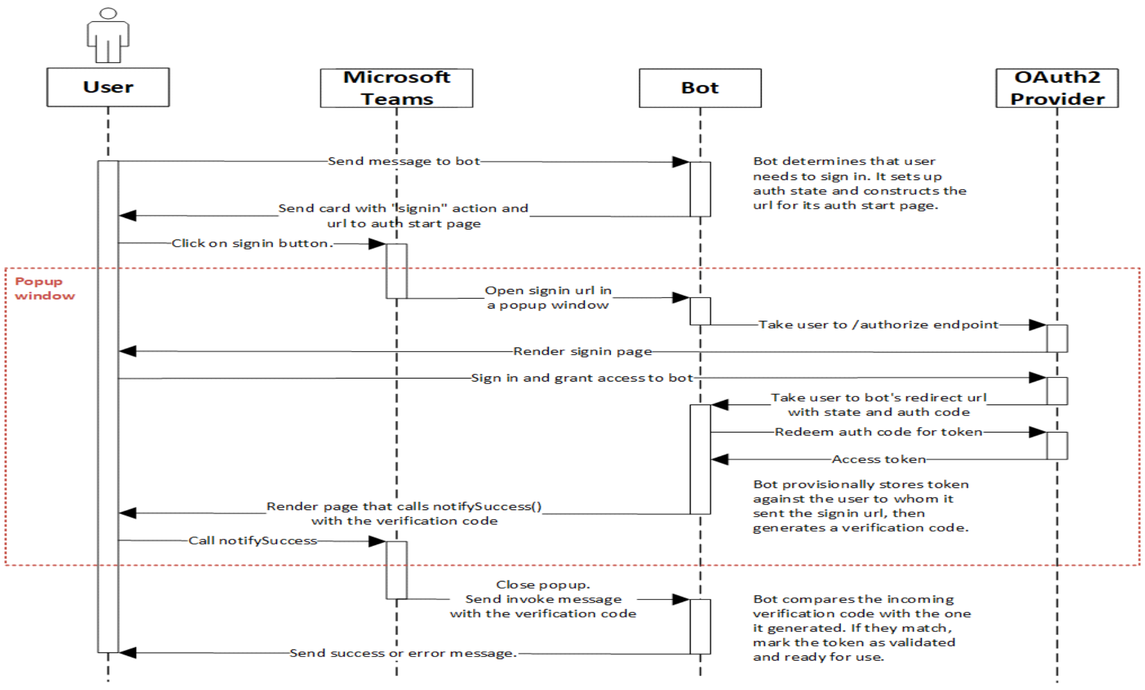
· Increase Revenue

DESIGN:

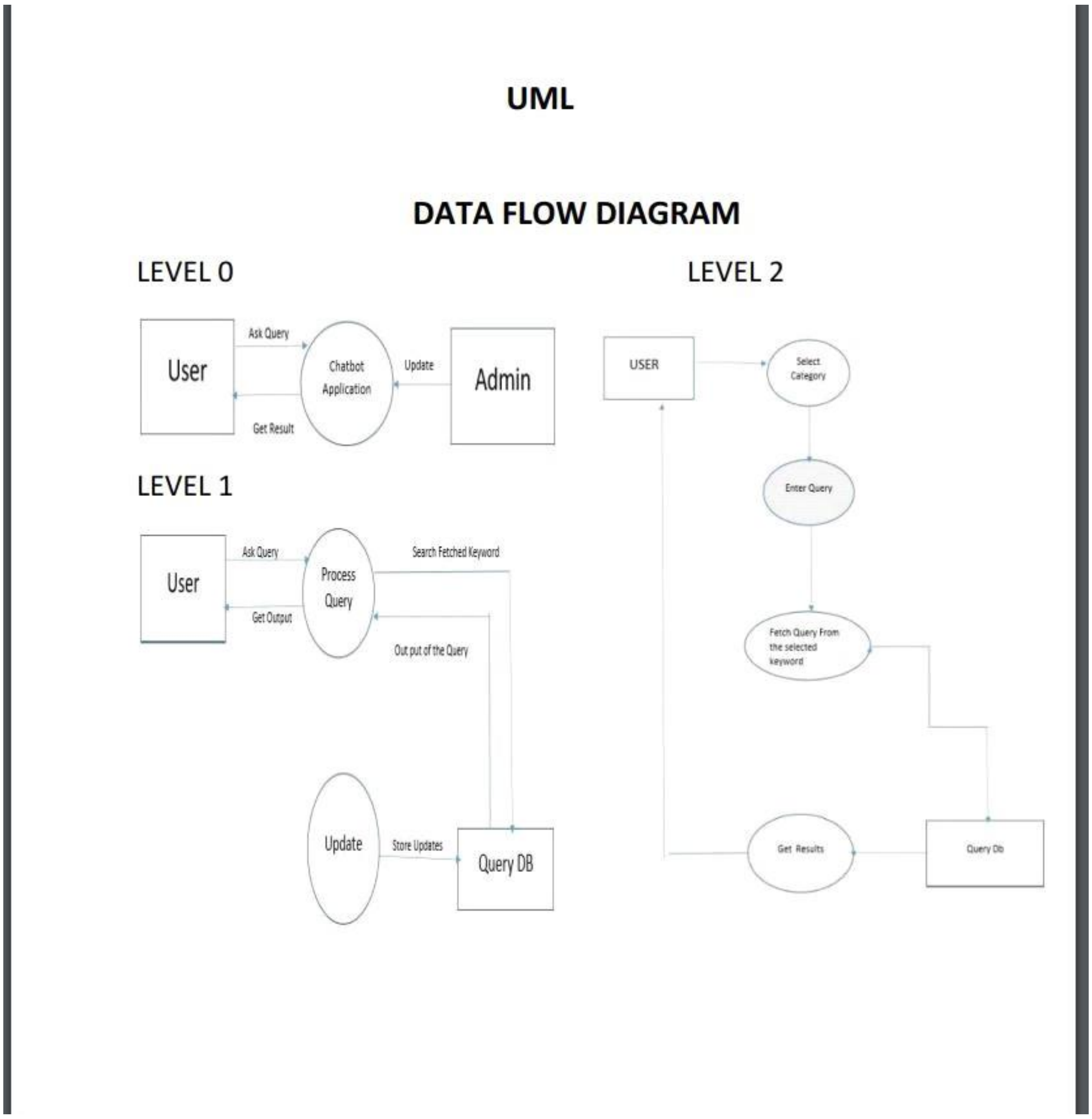
USE CASE DIAGRAM:



SEQUENCE DIAGRAM:


User
Microsoft Teams
Bot
OAuth2 Provider
Send message to bot
Bot determines that user needs to sign in. It sets up auth state and constructs the url for its auth start page.
Send card with "signin" action and url to auth start page
Click on signin button.
Popup window
Open signin url in a popup window
Take user to /authorize endpoint
Render signin page
Sign in and grant access to bot
Take user to bot's redirect url with state and auth code
Redeem auth code for token
Access token
Bot provisionally stores token against the user to whom it sent the signin url, then generates a verification code.
Render page that calls notifySuccess() with the verification code
Call notifySuccess
Close popup. Send invoke message with the verification code
Bot compares the incoming verification code with the one it generated. If they match, mark the token as validated and ready for use.
Send success or error message.

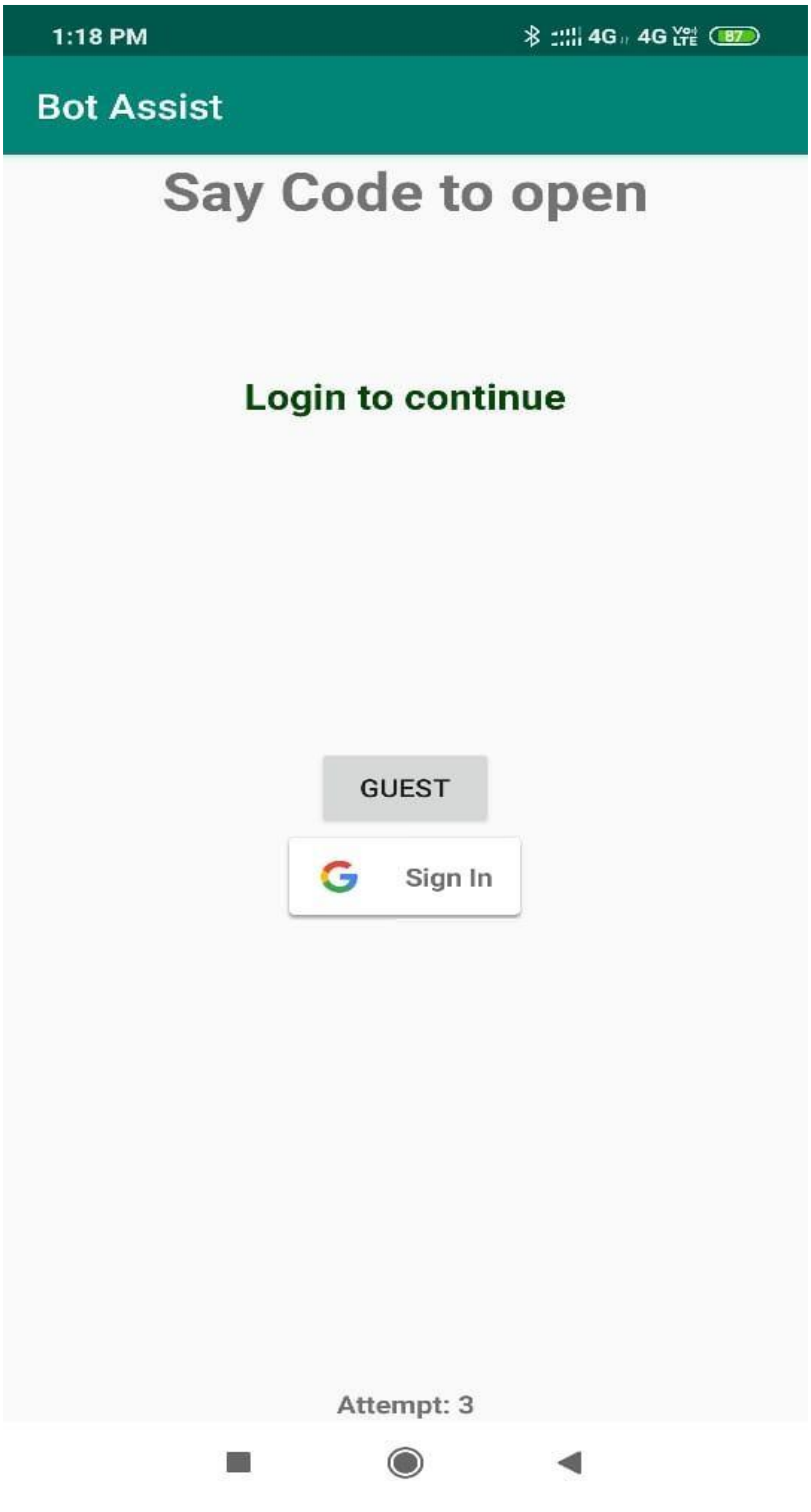
DATA FLOW DIAGRAM:



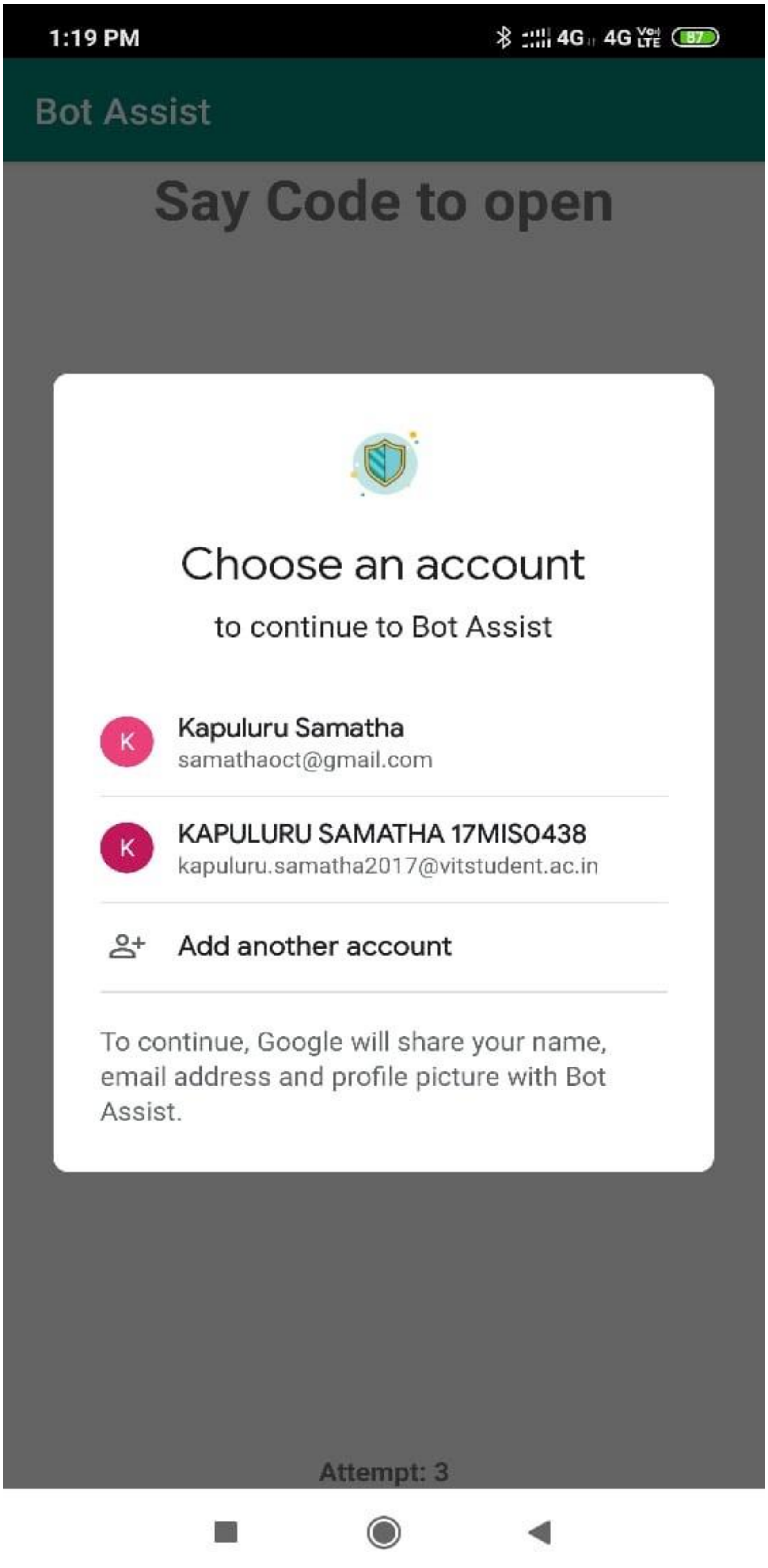
IMPLEMENTATION:/CODING

```
package com.gvr.botassist

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.speech.RecognizerIntent
import android.speech.tts.TextToSpeech
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.ImageButton
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import  androidx.recyclerview.widget.RecyclerView
import kotlinx.android.synthetic.main.activity_chat.*
import kotlinx.android.synthetic.main.chat_list.view.*
import java.util.*
import com.google.android.gms.auth.api.signin.GoogleSignInAccount
import androidx.core.content.ContextCompat.getSystemService
import android.icu.lang.UCharacter.GraphemeClusterBreak.T
import com.google.android.gms.auth.api.signin.GoogleSignIn
import com.google.android.gms.tasks.Task
import androidx.annotation.NonNull
import com.google.android.gms.tasks.OnCompleteListener
import androidx.core.content.ContextCompat.getSystemService
import android.icu.lang.UCharacter.GraphemeClusterBreak.T
import androidx.core.text.isDigitsOnly
import com.google.android.gms.auth.api.signin.GoogleSignInOptions
import java.lang.Exception


class ChatActivity : Activity(), View.OnClickListener,TextToSpeech.OnInitListener {
    override fun onInit(status: Int) {
        if (status != TextToSpeech.ERROR) {
            //t!!.language = Locale.UK
        }
    }

    val messages: ArrayList<String> = ArrayList()
    var t: TextToSpeech? = null
    var str: String = ""
    var gv:Boolean = false
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_chat)
        chatList.layoutManager = LinearLayoutManager(this)
        val c = findViewById<ImageButton>(R.id.mic)
        c.setOnClickListener(this)
        val s = findViewById<Button>(R.id.sent)
        s.setOnClickListener(this)
        t = TextToSpeech(this, this)
    }

    class MessageAdapter(val items: ArrayList<String>, val context: Context) :
        RecyclerView.Adapter<ViewHolder>() {

        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ViewHolder {
            return ViewHolder(
                LayoutInflater.from(context).inflate(
                    R.layout.chat_list,
                    parent,
                    false
                )
            )
        }

        override fun onBindViewHolder(holder: ViewHolder, position: Int) {
            holder?.tvAnimalType?.text = items.get(position)
        }

        override fun getItemCount(): Int {
            return items.size
        }
    }

    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val tvAnimalType = view.chatItems
    }

    override fun onClick(view: View) {
        when (view.id) {
            R.id.mic -> {
                gVoice()
                gv = false
            }
            R.id.sent -> {
                gSent()
                gv = true
            }
        }
    }

    private fun gVoice() {
        val intent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
        intent.putExtra(
            RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM
        )
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, Locale.getDefault())
        if (intent.resolveActivity(getPackageManager()) != null) {
            startActivityForResult(intent, 10)
        } else {
            Toast.makeText(this, "Device not Supported!!!",
Toast.LENGTH_SHORT).show()
        }
    }
    private fun gSent() {
            var mess = findViewById<EditText>(R.id.voice1).text.toString()
            if (gv) {
                messages.add(mess)
            } else {
                messages.add(str)
            }
            if (mess == "") {
                mess = str
            } else {
                str = mess
            }
            if (mess == "open alarm") {
                str = "Here we go"
                set(true)
                val managerclock = packageManager
                var i = managerclock.getLaunchIntentForPackage("com.gvr.alarm")
                i!!.addCategory(Intent.CATEGORY_LAUNCHER)
                startActivity(i)
            } else if (mess == "logout") {
                signOut()
            } else if (mess == "start calculation") {
                calculation()
            } else if (mess == "hello" || mess == "hi") {
                str = "hello!"
                set(true)
            } else if (mess == "who are you" || mess == "what is your name" || mess
== "tell me about yourself") {
                str = "I am Bot Assist"
                set(true)
            } else if (mess == "how are you" || mess == "how do you do") {
                str = "I am good"
                set(true)
            }else if(mess == "clear") {
                messages.clear()
               // str = "cleared"
                set(true)

            } else if (mess == "close") {
                str = "Bye Bye"
                set(false)
                this.finish()
            } else if (mess == "logout and close") {
                str = "Done"
                set(false)
                var gso: GoogleSignInOptions =

GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN).requestEmail()
                        .build()
                var mGoogleSignInClient = GoogleSignIn.getClient(this, gso)
                mGoogleSignInClient.signOut()
                this.finish()
            } else {
                try{
                var delimiter = " "
                val parts = str.split(delimiter)
                if (parts[0].isDigitsOnly() && parts[2].isDigitsOnly()) {
                    if (parts[1] == "+") {
                        var ans = parts[0].toInt() + parts[2].toInt()
                        str = "Answer is: " + ans.toString()
                        set(true)
                    }
                    if (parts[1] == "-") {
                        var ans = parts[0].toInt() - parts[2].toInt()
                        str = "Answer is: " + ans.toString()
                        set(true)
                    }
                    if (parts[1] == "x" || parts[1] == "into" || parts[1] ==
"multiply") {
                        var ans = parts[0].toInt() * parts[2].toInt()
                        str = "Answer is: " + ans.toString()
                        set(true)
                    }
                    if (parts[1] == "/" || parts[1] == "by") {
                        var ans = parts[0].toInt() / parts[2].toInt()
                        str = "Answer is: " + ans.toString()
                        set(true)
                    }
                } else if (parts[4] == "Armstrong") {
                    var dig = parts[1].length
                    var ans = armstrong(parts[1].toInt(), dig)
                    if (ans)
                        str = parts[1] + " is an Armstrong number"
                    else
                        str = parts[1] + " is not an Armstrong number"
                    set(true)
                } else {
                    str = "Sorry I don't know that"
                    set(true)
                }

            }catch (e:Exception){
                    Log.d("answer",e.toString())
                    str = "Sorry say again"
                    set(false)
                }
        }
    }

    fun armstrong(number:Int,n1:Int):Boolean {
        var originalNumber: Int
        var remainder: Int
        var result =  0
        var n = n1

        originalNumber = number

        while (originalNumber != 0) {
            originalNumber /= 10
            ++n
        }

        originalNumber = number

        while (originalNumber != 0) {
            remainder = originalNumber % 10
            result += Math.pow(remainder.toDouble(), n.toDouble()).toInt()
            originalNumber /= 10
        }
        if (result == number)
            return true
        else
            return false
    }
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?)
{
        super.onActivityResult(requestCode, resultCode, data)

        when (requestCode) {
            10 -> {
                if (resultCode == RESULT_OK && data != null) {
                    var list =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS)
                    str = list[0]
                    gSent()
                }
            }
        }
    }

    fun calculation() {

    }

    fun set(ver:Boolean)
    {
        if(ver) {
            messages.add(str)
            chatList.adapter = MessageAdapter(messages, this)
        }
        t!!.speak(str,TextToSpeech.QUEUE_FLUSH,null)
        voice1.setText("")
    }
    private fun signOut() {
        var gso:GoogleSignInOptions =
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN).requestEmail().bui
ld();
        var mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
        mGoogleSignInClient.signOut().addOnCompleteListener(this,
OnCompleteListener<Void> {
            val intent = Intent(this, MainActivity::class.java)
            startActivity(intent)
            })
    }
    public override fun onPause() {
        if (t != null) {
            //t!!.stop()
            //t!!.shutdown()
        }
        super.onPause()
    }

    override fun onBackPressed() {
        super.onBackPressed()
    }
}
```

OUTPUT:



1:18 PM
Bot Assist
Say Code to open
Login to continue
GUEST
Sign In
Attempt: 3


1:19 PM

Bot Assist

Say Code to open

Choose an account

to continue to Bot Assist

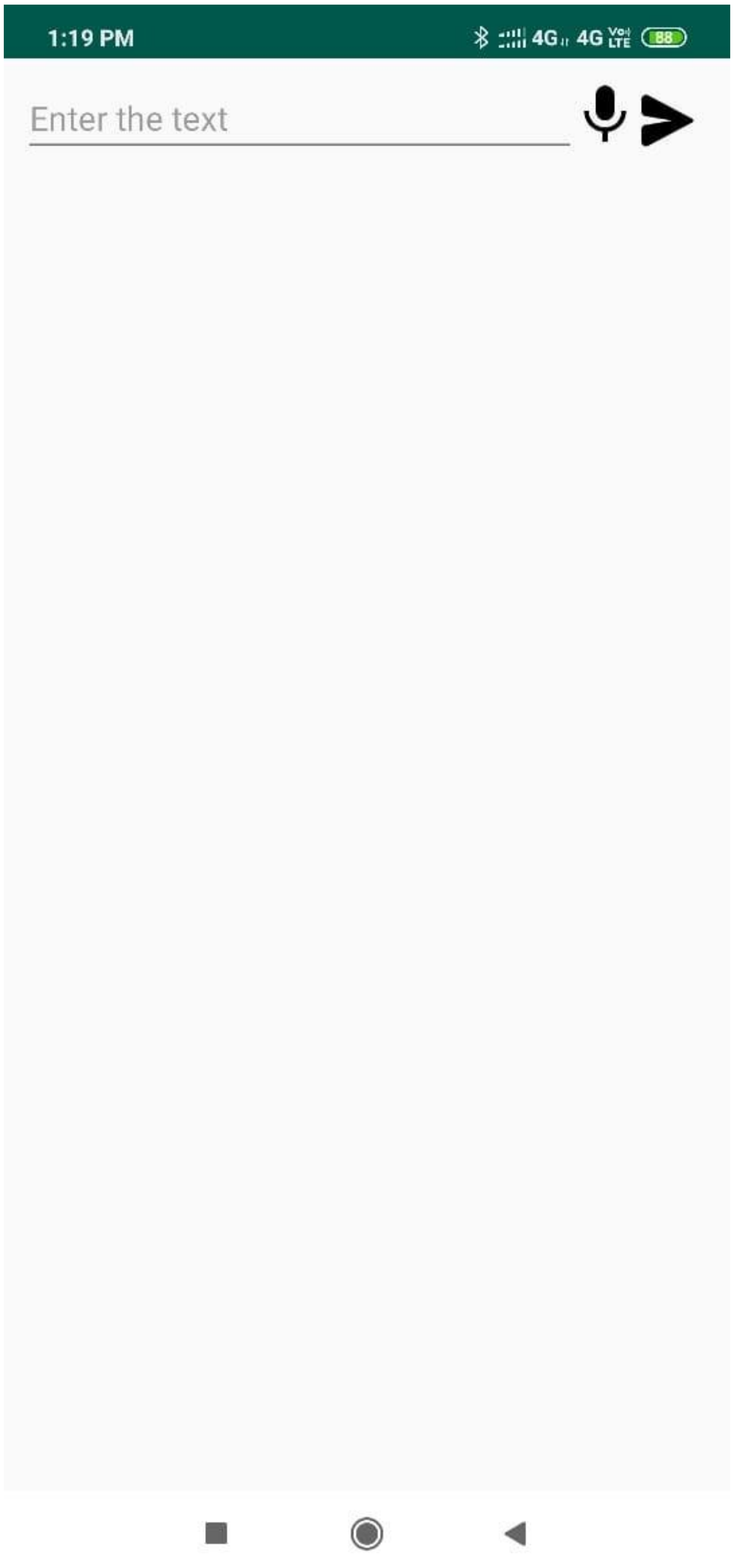
Kapuluru Samatha
samathaoct@gmail.com

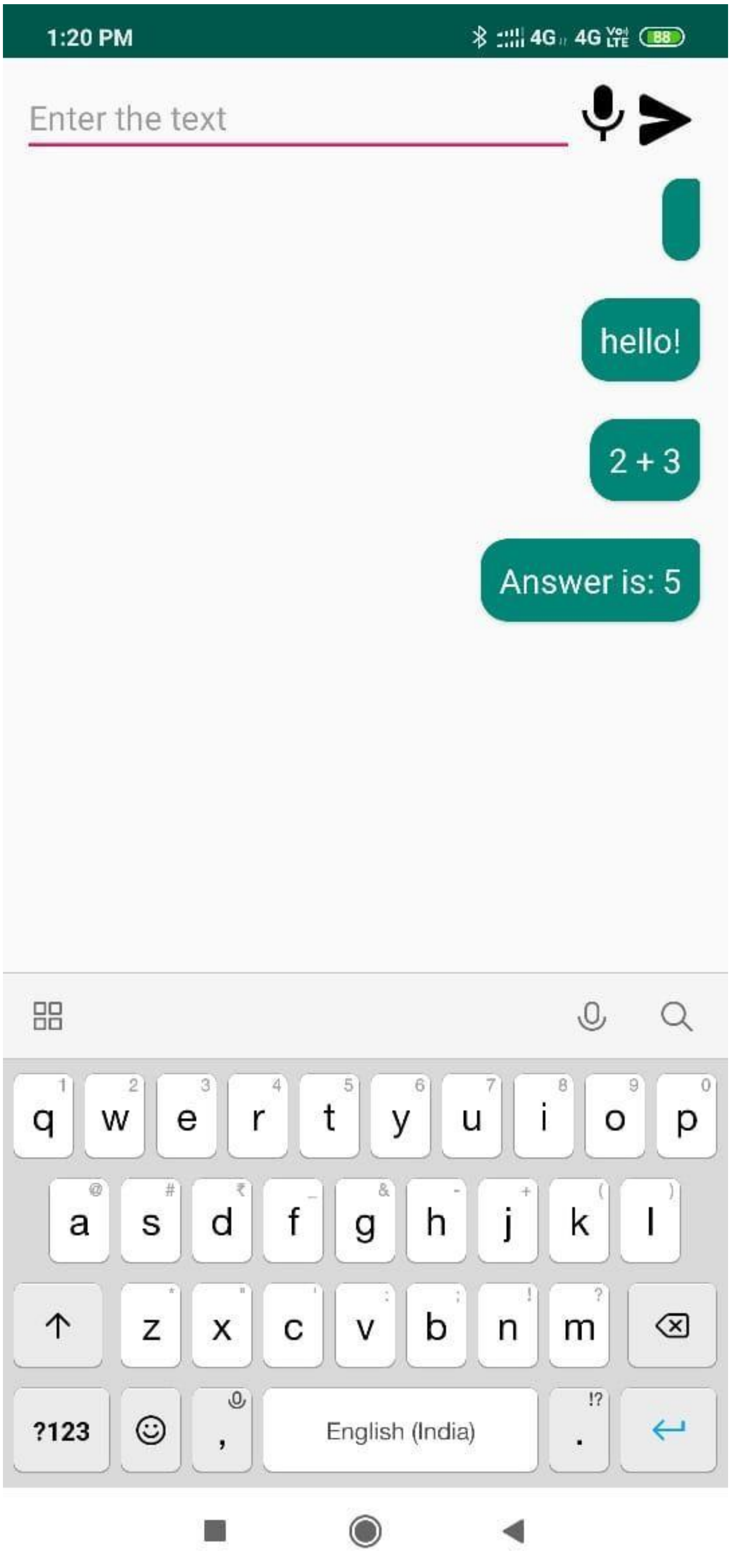
KAPULURU SAMATHA 17MIS0438
kapuluru.samatha2017@vitstudent.ac.in

Add another account

To continue, Google will share your name, email address and profile picture with Bot Assist.

Attempt: 3

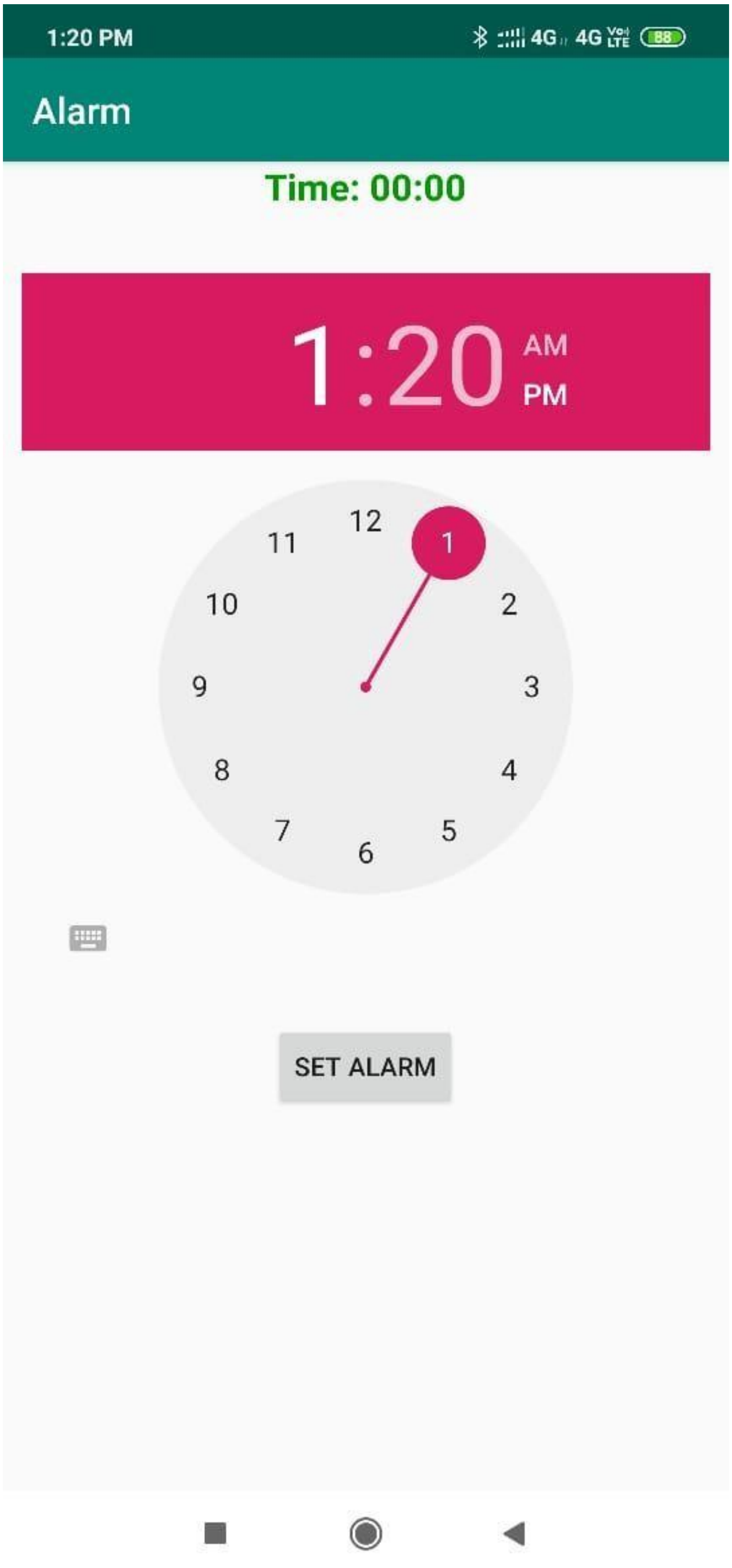

1:19 PM
4G 4G
88
Enter the text



hello!

2 + 3

Answer is: 5


1:20 PM
Alarm
Time: 00:00
1:20
AM
PM
12
1
2
3
4
5
6
7
8
9
10
11
SET ALARM

TESTING:

Using Selendroid:

- Full compatibility with the JSON Wire Protocol/Selenium 3 Ready.
- No modification of app under test required in order to automate it
- Testing the mobile web using built in Android driver webview app
- Same concept for automating native or hybrid apps
- UI elements can be found by different locator types
- Gestures are supported: Advanced User Interactions API
- Selendroid can interact with multiple Android devices (emulators or hardware devices) at the same time
- Existing Emulators are started automatically
- Selendroid supports hot plugging of hardware devices
- Full integration as a node into Selenium Grid for scaling and parallel testing
- Multiple Android target API support (10 to 19)
- Built in Inspector to simplify test case development.
- Selendroid can be extended at runtime with your own extensions !

Selendroid is a test automation framework which drives off the UI of Android native and hybrid applications (apps) and the mobile web.

CONCLUSIONS:

from my perspective, chatbots or smart assistants with artificial intelligence are dramatically changing businesses. There is a wide range of chatbot building platforms that are available for various enterprises, such as e-commerce, retail, banking, leisure, travel, healthcare, and so on.

Chatbots can reach out to a large audience on messaging apps and be more effective than humans. They may develop into a capable information-gathering tool in the near future.

REFERENCES:

dzone.com › articles › significance-of-chatbot

^ "What is a chatbot?". techtarget.com. Retrieved 30 January 2017.

^ Luka Bradeško, Dunja Mladenić. "A Survey of Chabot Systems through a Loebner Prize Competition" (PDF). Retrieved 28 June 2019.

^ Mauldin 1994

^ Orf, Darren. "Google Assistant Is a Mega AI Bot That Wants To Be Absolutely Everywhere".

^ "The 8 best chatbots of 2016". 21 December 2016.

^ "2017 Messenger Bot Landscape, a Public Spreadsheet Gathering 1000+ Messenger Bots". 3 May 2017.

^ "An Overview of Conversational AI".

^ (Turing 1950)

^ (Weizenbaum 1966, p. 36)

^ (Weizenbaum 1966, pp. 44–5)

^ GüzeldereFranchi 1995

^ Computer History Museum 2006

^ Sondheim 1997

^ Network Working Group 1973—Transcript of a session between Parry and Eliza. (This is not the dialogue from the ICCC, which took place October 24–26, 1972, whereas this session is from September 18, 1972.)

FUTURE OF BOT ASSISTANT:

It would be wrong or ignorant to say that bot is evolving and their evolution will become complete in 2019. bot evolved in 2018 and are more intelligent as well as human than ever.

There is no denying to this fact. The successful adoption of bots by end users has led to the use of more and more bots in advanced artificial intelligence technologies and their usage by a custom software development company. Even there are reports that 80–85% of businesses will be deploying advanced chatbots by 2020.

Giving human-like experience

If you are talking about personalized customer experience and offering services which are similar to the ones provided by humans, then chatbots aren't far behind. The advancement in artificial intelligence and machine learning in today's era has made chatbot services more similar to human-like and even impeccable.