Compsoft Technologies
Bangalore ,Karnataka

AN INTERNSHIP REPORT ON

"VIRTUAL ASSISTANT FOR VISUALLY IMPAIRED"

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by: ROHAN RAJENDRAN

USN: 1BO19CS084

Team Name: TECHNOCRATS

Under the Supervision of: PRAJWAL

BRINDAVAN COLLEGE OF ENGINEERING

Bangalore

(Affiliated to Visveshwarya Technological University and Approved by AICTE, New Delhi)

Accredited By NAAC with 'A' Grade

ISO 9001-2015 and 14001-2015 certified Institute

Bengaluru, Karnataka 560063

ABOUT THE COMPANY

CST is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At CST , we believe that service and quality is the key to success
We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

- ☐ Development - We develop responsive, functional and super-fast websites.
- ☐ We keep User Experience in mind while creating websites. A website should load quickly and should be accessible even on a small view-port and slow internet connection.
- ☐ Mobile Application - We offer a wide range of professional Android, iOS& Hybrid app development services for our global clients, from a start up to a large enterprise.
- ☐ Design - We offer professional Graphic design, Brochure design & Logo design. We are experts in crafting visual content to convey the right message to the customers.
- ☐ Consultancy - We are here to provide you with expert advice on your design and development requirement.
- ☐ Videos - We create a polished professional video that impresses your audience.

Table of Contents

OVERVIEW OF THE PROJECT

Project Name: VIRTUAL ASSISTANT FOR VISUALLY IMPAIRED

Team Members: JAYASHREE YADAV, ROHAN RAJENDRAN, HARSHITHA. M, PREETHAM SHARMA

OVERVIEW:

The field of artificial intelligence has led to various virtual assistants such as Siri in iPhone, Google Assistant, Microsoft Cortana, and so on. Even after such progression, very little has been done to implement these technologies to assist the visually impaired community. Recognizing a person or distinguishing an object, these tasks are straightforward for common people but can be very difficult for people that are partly or completely blind. Their lives can be made smoother by assisting them to detect what is present in front of them at that instant. We aim to develop a system/assistant that will serve to guide a visually impaired person and will indicate the person by speaking through the earpiece. The system will help the person recognize people, add new faces and detect objects that are in their vicinity. We will have a mobile application that will consist of numerous deep learning models that will help applications increase their administration. The primary working of the system will consist of the camera continuously feeding images for inputs, the core system processing this input information and the earpiece acting as the output device to provide this output to the user.

INTRODUCTION:

Today there are nearly 285 million people in the world that are visually impaired [12]. Although technology has grown leaps and bounds, the accessibility, especially that of the internet for differently-abled people is still far-fetched. In this modern world, more and more things can be performed online. From shopping, ordering food, to booking train tickets everything can be done online. Assistive technologies such as a screen reader or magnifiers can enable visually impaired individuals to access the internet. Unfortunately, these screen readers need to keep the functionality of the website in mind otherwise it becomes difficult to read data from the website. The major challenge in developing a stable software is to include as few keystrokes as possible and to provide an end to end experience with the help of voice alone. Some of the screen readers work only with a particular kind of browser and some require the user to remember complex commands thus screen readers are not an effective solution to the problem at hand and cannot be used to access the internet. The primary objective is to bridge the accessibility gap between the average user and the visually impaired individuals with regards to the internet. The internet is blind to the visually impaired, but to not make the converse the truth, in this paper we present an end-to-end voice-based software for the visually impaired to enable them to access the internet with minimal to no keystrokes required.

TOOLS USED

Software Requirements:

- Visual Studio Code 2019.
- Android Studio 2020
- Google Chrome or Microsoft Edge of latest version.
- Front End: HTML, CSS, JS
- Backend : Php, MySQL, Xampp
- Linux 7.1 or Windows XP/7/8/10 OS or Mac OS

Hardware Requirements:

- Pentium 200-MHz computer with a minimum of 64 MB of

RAM (128 MB of RAMrecommended).

- Monitor with a refresh rate of at least 40Hz for a smooth GUI experience (optional).

SYSTEM OVERVIEW AND DESIGN:

The system comprises a modular client server distributed architecture. The system consists of the main menu which first runs on the startup of the software and the website modules. The client communicates with the server and back with the use of REST APIs, thus the website modules are not local to the client. Throughout the system, the user communicates with the software via speech-to-text interface. The Google library of speech-to-text (Speech Recognition) for Python is used for this purpose. For communicating the system's output to the user as well as for confirming the user input, the recognized input is played back to the user using the Python text-to-speech library (pytssx3). The modules are written in Python and make use of Selenium for automation of the respective module and Beautiful Soup for scraping the contents of the web page. The "Script" component of each module consists of the customized code that entails the features of the website contained in the module. For instance, the Wikipedia module consists of a Question and Answer and Summary feature along with the traditional feature of reading out the entire article. The former is implemented by training a BERT model on the Stanford Question Answering Dataset (SQuaD). The APIs that hold the system together are written in Flask. The software is operating system independent to support hassle free application and usage of the system.

IMPLEMENTATION

CAMERA ACTIVITY:

```
package com.example.vision
import android.content.pm.PackageManager
import android.graphics.*
import android.os.Build
import android.os.Bundle
import android.os.Handler
import android.os.HandlerThread
import android.speech.tts.TextToSpeech
import android.util.DisplayMetrics
import android.util.Log
import android.util.Rational
import android.util.Size
import android.view.Surface
import android.view.ViewGroup
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompatActivity
import androidx.camera.core.*
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import kotlinx.android.synthetic.main.camera_activity.*
import java.io.ByteArrayOutputStream
import java.util.*

class CameraActivity : AppCompatActivity(), TextToSpeech.OnInitListener {

private var lensFacing = CameraX.LensFacing.BACK
private val TAG = "CameraActivity"

private val REQUEST_CODE_PERMISSIONS = 101
private val REQUIRED_PERMISSIONS = arrayOf("android.permission.CAMERA")

private var tfLiteClassifier: TFLiteClassifier = TFLiteClassifier(this@CameraActivity)
private var tts: TextToSpeech? = null

@RequiresApi(Build.VERSION_CODES.LOLLIPOP)
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.camera_activity)
tts = TextToSpeech(this, this)

if (allPermissionsGranted()) {
textureView.post { startCamera() }
textureView.addOnLayoutChangeListener { _, _, _, _, _, _, _, _, _ ->
updateTransform()
```

```
}
} else {
ActivityCompat.requestPermissions(this, REQUIRED_PERMISSIONS,
REQUEST_CODE_PERMISSIONS);
}

textureView.setOnClickListener {
//tts?.speak("Hello ",TextToSpeech.QUEUE_FLUSH,null,null)
tts?.speak(predictedTextView.text, TextToSpeech.QUEUE_FLUSH, null, null)
}

tfLiteClassifier
.initialize()
.addOnSuccessListener { }
.addOnFailureListener { e -> Log.e(TAG, "Error in setting up the classifier.", e) }
}

private fun startCamera() {
//val metrics = DisplayMetrics().also { textureView.display.getRealMetrics(it) }
//val screenSize = Size(metrics.widthPixels, metrics.heightPixels)
val screenAspectRatio = Rational(1, 1)

val previewConfig = PreviewConfig.Builder().apply {
setLensFacing(lensFacing)
setTargetResolution(Size(256, 256))
setTargetAspectRatio(screenAspectRatio)
setTargetRotation(windowManager.defaultDisplay.rotation)
setTargetRotation(textureView.display.rotation)
}.build()

val preview = Preview(previewConfig)
preview.setOnPreviewOutputUpdateListener {
val parent = textureView.parent as ViewGroup
parent.removeView(textureView)
textureView.setSurfaceTexture(it.surfaceTexture)
parent.addView(textureView, 0)
updateTransform()
}


val analyzerConfig = ImageAnalysisConfig.Builder().apply {
// Use a worker thread for image analysis to prevent glitches
val analyzerThread = HandlerThread("AnalysisThread").apply {
start()
}
setCallbackHandler(Handler(analyzerThread.looper))
setImageReaderMode(ImageAnalysis.ImageReaderMode.ACQUIRE_LATEST_IMAGE)
}.build()


val analyzerUseCase = ImageAnalysis(analyzerConfig)
```

```
analyzerUseCase.analyzer =
ImageAnalysis.Analyzer { image: ImageProxy, rotationDegrees: Int ->

val bitmap = image.toBitmap()

tfLiteClassifier
.classifyAsync(bitmap)
.addOnSuccessListener { resultText ->
predictedTextView?.text = resultText
}
.addOnFailureListener { error -> }

}
CameraX.bindToLifecycle(this, preview, analyzerUseCase)
}

fun ImageProxy.toBitmap(): Bitmap {
val yBuffer = planes[0].buffer // Y
val uBuffer = planes[1].buffer // U
val vBuffer = planes[2].buffer // V

val ySize = yBuffer.remaining()
val uSize = uBuffer.remaining()
val vSize = vBuffer.remaining()

val nv21 = ByteArray(ySize + uSize + vSize)

yBuffer.get(nv21, 0, ySize)
vBuffer.get(nv21, ySize, vSize)
uBuffer.get(nv21, ySize + vSize, uSize)

val yuvImage = YuvImage(nv21, ImageFormat.NV21, this.width, this.height, null)
val out = ByteArrayOutputStream()
yuvImage.compressToJpeg(Rect(0, 0, yuvImage.width, yuvImage.height), 100, out)
val imageBytes = out.toByteArray()
return BitmapFactory.decodeByteArray(imageBytes, 0, imageBytes.size)
}

private fun updateTransform() {
val matrix = Matrix()
val centerX = textureView.width / 2f
val centerY = textureView.height / 2f

val rotationDegrees = when (textureView.display.rotation) {
Surface.ROTATION_0 -> 0
Surface.ROTATION_90 -> 90
Surface.ROTATION_180 -> 180
Surface.ROTATION_270 -> 270
else -> return
}
matrix.postRotate(-rotationDegrees.toFloat(), centerX, centerY)
```

```
textureView.setTransform(matrix)
}

override fun onRequestPermissionsResult(
requestCode: Int,
permissions: Array<String>,
grantResults: IntArray
) {

if (requestCode == REQUEST_CODE_PERMISSIONS) {
if (allPermissionsGranted()) {
startCamera()
} else {
Toast.makeText(this, "Permissions not granted by the user.", Toast.LENGTH_SHORT)
.show()
finish()
}
}
}

private fun allPermissionsGranted(): Boolean {

for (permission in REQUIRED_PERMISSIONS) {
if (ContextCompat.checkSelfPermission(
this,
permission
) != PackageManager.PERMISSION_GRANTED
) {
return false
}
}
return true
}


public override fun onDestroy() {
tfLiteClassifier.close()
// Shutdown TTS
if (tts != null) {
tts!!.stop()
tts!!.shutdown()
}
super.onDestroy()
}

override fun onInit(p0: Int) {
Log.d(com.example.vision.TAG, "Initializing TTS")
if (p0 == TextToSpeech.SUCCESS) {
Log.d(com.example.vision.TAG, "SUCCESS")
tts!!.language = Locale.US
tts?.speak(
```

```
"Currency Recognizer opened.",
TextToSpeech.QUEUE_FLUSH, null, null
)
}
}
}
```

Message Activity:

```
package com.example.vision

import android.Manifest
import android.app.Activity
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.content.IntentFilter
import android.content.pm.PackageManager
import android.os.Build
import android.os.Bundle
import android.provider.Telephony
import android.speech.RecognizerIntent
import android.speech.tts.TextToSpeech
import android.telephony.SmsManager
import android.telephony.SmsMessage
import android.util.Log
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import kotlinx.android.synthetic.main.messaging.*
import java.util.*

class MessageActivity : AppCompatActivity(), TextToSpeech.OnInitListener {

    private var tts: TextToSpeech? = null

    @RequiresApi(Build.VERSION_CODES.LOLLIPOP)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.messaging)
        tts = TextToSpeech(this, this)

        if (ActivityCompat.checkSelfPermission(
                this,
                Manifest.permission.RECEIVE_SMS
            ) != PackageManager.PERMISSION_GRANTED
        ) {
            ActivityCompat.requestPermissions(
                this,
                arrayOf(Manifest.permission.RECEIVE_SMS, Manifest.permission.SEND_SMS), 111
```

```
            )
        } else
            receiveMsg()

        editTextTextMultiLine.setOnClickListener {
            tts?.speak("Please speak your message", TextToSpeech.QUEUE_FLUSH, null, null)
            Thread.sleep(2000)
            speakMsg()
        }

        editTextPhone.setOnClickListener {
            tts?.speak(
                "Please speak recipient's phone number",
                TextToSpeech.QUEUE_FLUSH,
                null,
                null
            )
            Thread.sleep(2000)
            speakPhone()
        }

        sendMsg.setOnClickListener {
            val sms: SmsManager = SmsManager.getDefault()
            sms.sendTextMessage(
                editTextPhone.text.toString(),
                "ME",
                editTextTextMultiLine.text.toString(),
                null,
                null
            )
            tts?.speak("Message sent successfully", TextToSpeech.QUEUE_FLUSH, null, null)
        }
    }

    private fun speakMsg() {
        val msgIntent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
        msgIntent.putExtra(
            RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM
        )
        msgIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault())
        msgIntent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Hi speak something")
        try {
            startActivityForResult(msgIntent, 101)
        } catch (e: Exception) {
            e.message?.let { Log.e("MSG", it) }
        }
    }

    private fun speakPhone() {
        val phnIntent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
```

```
    phnIntent.putExtra(
        RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM
    )
    phnIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault())
    phnIntent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Hi speak something")
    try {
        startActivityForResult(phnIntent, 102)
    } catch (e: Exception) {
        e.message?.let { Log.e("Phone", it) }
    }
}

private fun receiveMsg() {
    val br = object : BroadcastReceiver() {
        @RequiresApi(Build.VERSION_CODES.LOLLIPOP)
        override fun onReceive(p0: Context?, p1: Intent?) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
                for (sms: SmsMessage in Telephony.Sms.Intents.getMessagesFromIntent(p1)) {
                    editTextPhone.setText(sms.originatingAddress)
                    editTextTextMultiLine.setText(sms.displayMessageBody)
                    Toast.makeText(applicationContext, "Msg received", Toast.LENGTH_SHORT)
                        .show()
                    val text = """ Message received form${editTextPhone.text}"""
                    tts?.speak(text, TextToSpeech.QUEUE_FLUSH, null, null)
                    Thread.sleep(5000)
                    tts?.speak(editTextTextMultiLine.text, TextToSpeech.QUEUE_FLUSH, null, null)
                }
            }
        }

    }
    registerReceiver(br, IntentFilter("android.provider.Telephony.SMS_RECEIVED"))
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == 101) {
        if (resultCode == Activity.RESULT_OK && null != data) {
            val result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS)
            editTextTextMultiLine.setText(result?.get(0))
        }
    } else if (requestCode == 102) {
        if (resultCode == Activity.RESULT_OK && null != data) {
            val result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS)
            editTextPhone.setText(result?.get(0))
        }
    }
}

public override fun onDestroy() {
```

```
        // Shutdown TTS
        if (tts != null) {
            tts!!.stop()
            tts!!.shutdown()
        }
        super.onDestroy()
    }

    override fun onInit(p0: Int) {
        Log.d(TAG, "Initializing TTS")
        if (p0 == TextToSpeech.SUCCESS) {
            Log.d(TAG, "SUCCESS")
            tts!!.language = Locale.US
            tts?.speak(
                "Messaging box opened.",
                TextToSpeech.QUEUE_FLUSH, null, null
            )
        }
    }
}
```

Phone Activity:

```
package com.example.vision

import android.Manifest
import android.app.Activity
import android.content.Intent
import android.content.pm.PackageManager
import android.net.Uri
import android.os.Build
import android.os.Bundle
import android.speech.RecognizerIntent
import android.speech.tts.TextToSpeech
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.EditText
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import kotlinx.android.synthetic.main.messaging.*
import kotlinx.android.synthetic.main.phonemanager.*
import java.util.*


class PhoneActivity : AppCompatActivity(), View.OnClickListener,
    View.OnLongClickListener,
    TextToSpeech.OnInitListener {

    private var tts: TextToSpeech? = null
```

```
private lateinit var phoneNum: EditText

@RequiresApi(Build.VERSION_CODES.LOLLIPOP)
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.phonemanager)
    phoneNum = findViewById(R.id.editTextPhone2)

    tts = TextToSpeech(this, this)

    imageView2.setOnClickListener {
        tts?.speak(
            "Please speak recipient's phone number",
            TextToSpeech.QUEUE_FLUSH,
            null,
            null
        )
        Thread.sleep(2000)
        speakPhone()
    }

    val btn1: Button = findViewById(R.id.btn1)
    val btn2: Button = findViewById(R.id.btn2)
    val btn3: Button = findViewById(R.id.btn3)
    val btn4: Button = findViewById(R.id.btn4)
    val btn5: Button = findViewById(R.id.btn5)
    val btn6: Button = findViewById(R.id.btn6)
    val btn7: Button = findViewById(R.id.btn7)
    val btn8: Button = findViewById(R.id.btn8)
    val btn9: Button = findViewById(R.id.btn9)
    val btn0: Button = findViewById(R.id.btn0)
    val btnStar: Button = findViewById(R.id.btnStar)
    val btnHash: Button = findViewById(R.id.btnHash)
    val btnBack: Button = findViewById(R.id.btnBack)
    val btnCall: Button = findViewById(R.id.btnCall)

    btn1.setOnClickListener(this)
    btn2.setOnClickListener(this)
    btn3.setOnClickListener(this)
    btn4.setOnClickListener(this)
    btn5.setOnClickListener(this)
    btn6.setOnClickListener(this)
    btn7.setOnClickListener(this)
    btn8.setOnClickListener(this)
    btn9.setOnClickListener(this)
    btn0.setOnClickListener(this)
    btnStar.setOnClickListener(this)
    btnHash.setOnClickListener(this)
    btnCall.setOnClickListener(this)
    btnBack.setOnClickListener(this)
```

```
    btnBack.setOnLongClickListener {
        val text: String = phoneNum.text.toString()
        phoneNum.setText(text.substring(0, text.length - 1))
        true
    }

    btnCall.setOnLongClickListener {
        if (ActivityCompat.checkSelfPermission(
                this,
                Manifest.permission.CALL_PHONE
            ) != PackageManager.PERMISSION_GRANTED
        ) {
            ActivityCompat.requestPermissions(
                this,
                arrayOf(Manifest.permission.CALL_PHONE),
                111
            )
        } else {
            speak("calling " + editTextPhone2.text.toString())
            Thread.sleep(5000)
            startCall()
        }
        true
    }

    btn1.setOnLongClickListener(this)
    btn2.setOnLongClickListener(this)
    btn3.setOnLongClickListener(this)
    btn4.setOnLongClickListener(this)
    btn5.setOnLongClickListener(this)
    btn6.setOnLongClickListener(this)
    btn7.setOnLongClickListener(this)
    btn8.setOnLongClickListener(this)
    btn9.setOnLongClickListener(this)
    btn0.setOnLongClickListener(this)
    btnStar.setOnLongClickListener(this)
    btnHash.setOnLongClickListener(this)


}

@RequiresApi(Build.VERSION_CODES.LOLLIPOP)
override fun onClick(view: View) {
    val text = when (view.id) {
        R.id.btn1 -> "You clicked one"
        R.id.btn2 -> "You clicked two"
        R.id.btn3 -> "You clicked three"
        R.id.btn4 -> "You clicked four"
        R.id.btn5 -> "You clicked five"
        R.id.btn6 -> "You clicked six"
```

```
            R.id.btn7 -> "You clicked seven"
            R.id.btn8 -> "You clicked eight"
            R.id.btn9 -> "You clicked nine"
            R.id.btn0 -> "You clicked zero"
            R.id.btnStar -> "You clicked star"
            R.id.btnHash -> "You clicked hash"
            R.id.btnBack -> "You clicked delete"
            R.id.btnCall -> "You clicked call"
            else -> throw IllegalArgumentException("Undefined Clicked")

        }
        //Toast.makeText(this, text, Toast.LENGTH_SHORT).show()
        speak(text)
    }

    override fun onLongClick(view: View): Boolean {
        val text = when (view.id) {
            R.id.btn1 -> "1"
            R.id.btn2 -> "2"
            R.id.btn3 -> "3"
            R.id.btn4 -> "4"
            R.id.btn5 -> "5"
            R.id.btn6 -> "6"
            R.id.btn7 -> "7"
            R.id.btn8 -> "8"
            R.id.btn9 -> "9"
            R.id.btn0 -> "0"
            R.id.btnStar -> "*"
            R.id.btnHash -> "#"
            else -> throw IllegalArgumentException("Undefined Clicked")
        }
        phoneNum.append(text)
        return true
    }

    override fun onInit(status: Int) {
        if (status == TextToSpeech.SUCCESS) {
            tts!!.language = Locale.US
            tts?.speak(
                "Phone manager opened.",
                TextToSpeech.QUEUE_FLUSH, null, null
            )
        }
    }

    @RequiresApi(Build.VERSION_CODES.LOLLIPOP)
    private fun speak(text: String) {

        tts?.speak(text, TextToSpeech.QUEUE_FLUSH, null, null)

    }
```

```
public override fun onDestroy() {
    // Shutdown TTS
    if (tts != null) {
        tts!!.stop()
        tts!!.shutdown()
    }
    super.onDestroy()
}

@RequiresApi(Build.VERSION_CODES.LOLLIPOP)
private fun startCall() {
    val callIntent = Intent(Intent.ACTION_CALL)
    callIntent.data = Uri.parse("tel:" + phoneNum.text)
    startActivity(callIntent)
}

@RequiresApi(Build.VERSION_CODES.LOLLIPOP)
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    if (requestCode == 111)
        startCall()
}

private fun speakPhone() {
    val phnIntent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
    phnIntent.putExtra(
        RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM
    )
    phnIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault())
    phnIntent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Hi speak something")
    try {
        startActivityForResult(phnIntent, 102)
    } catch (e: Exception) {
        e.message?.let { Log.e("Phone", it) }
    }
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == 102) {
        if (resultCode == Activity.RESULT_OK && null != data) {
            val result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS)
            editTextPhone2.setText(result?.get(0))
        }
    }
}
```

Time and Date:

```
package com.example.vision

import android.content.Intent
import android.content.IntentFilter
import android.os.BatteryManager
import android.os.Build
import android.os.Bundle
import android.speech.tts.TextToSpeech
import android.util.Log
import android.view.View
import android.widget.TextView
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.datetime.*
import java.text.SimpleDateFormat
import java.util.*

val TAG = "SPEECH"

class TimeDateActivity : AppCompatActivity(), View.OnClickListener,
TextToSpeech.OnInitListener {

    private var tts: TextToSpeech? = null

    @RequiresApi(Build.VERSION_CODES.LOLLIPOP)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.datetime)
        tts = TextToSpeech(this, this)
        batteryCard.setOnClickListener(this)

    }

    @RequiresApi(Build.VERSION_CODES.LOLLIPOP)
    override fun onClick(view: View) {

        if (view.id == R.id.batteryCard) {

            val currDate = findViewById<TextView>(R.id.currDateTime)
            val date = getCurrentDateTime()
            val dateInString = date.toString("E,dd MMMM yyyy HH:mm:ss")
            currDate.text = dateInString
            tts?.speak(dateInString, TextToSpeech.QUEUE_FLUSH, null, null)
            Thread.sleep(6000)
            val batteryStatus: Intent? =
                IntentFilter(Intent.ACTION_BATTERY_CHANGED).let { ifilter ->
                    this.registerReceiver(null, ifilter)
```

```
        }
        val status: Int = batteryStatus?.getIntExtra(BatteryManager.EXTRA_STATUS, -1) ?: -1
        var isCharging = when (status) {
            BatteryManager.BATTERY_STATUS_CHARGING -> "Phone is charging"
            else -> "Phone is not charging"
        }

        val txtView = findViewById<TextView>(R.id.batteryStatus)

        txtView.text = isCharging

        val batstate = "Your $isCharging"


        val batteryPct: Float? = batteryStatus?.let { intent ->
            val level: Int = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1)
            val scale: Int = intent.getIntExtra(BatteryManager.EXTRA_SCALE, -1)
            level * 100 / scale.toFloat()
        }

        val txtview = findViewById<TextView>(R.id.batteryPer)

        txtview.text = batteryPct.toString()

        val bst = "Your battery level is " + batteryPct.toString() + "percent and $batstate"
        tts?.speak(bst, TextToSpeech.QUEUE_FLUSH, null, null)
    }


}


public override fun onDestroy() {
    // Shutdown TTS
    if (tts != null) {
        tts!!.stop()
        tts!!.shutdown()
    }
    super.onDestroy()
}

override fun onInit(p0: Int) {
    Log.d(TAG, "Initializing TTS")
    if (p0 == TextToSpeech.SUCCESS) {
        Log.d(TAG, "SUCCESS")
        tts!!.language = Locale.US
        tts?.speak(
            "Time,date and battery status opened.",
            TextToSpeech.QUEUE_FLUSH, null, null
        )
```

```
        }
    }

    private fun Date.toString(format: String): String {
        val formatter = SimpleDateFormat(format)
        return formatter.format(this)
    }

    private fun getCurrentDateTime(): Date {
        return Calendar.getInstance().time
    }

}
```

Currency Detection:

```
package org.tensorflow.lite.examples.classification;

import android.Manifest;
import android.app.Fragment;
import android.content.Context;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraCharacteristics;
import android.hardware.camera2.CameraManager;
import android.hardware.camera2.params.StreamConfigurationMap;
import android.media.Image;
import android.media.Image.Plane;
import android.media.ImageReader;
import android.media.ImageReader.OnImageAvailableListener;
import android.media.MediaPlayer;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.os.Trace;
import androidx.annotation.NonNull;
import androidx.annotation.UiThread;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.util.Size;
import android.view.Surface;
import android.view.View;
import android.view.ViewTreeObserver;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

```
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.material.bottomsheet.BottomSheetBehavior;
import java.nio.ByteBuffer;
import java.util.List;
import java.util.concurrent.TimeUnit;

import org.tensorflow.lite.examples.classification.env.ImageUtils;
import org.tensorflow.lite.examples.classification.env.Logger;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Device;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Model;
import org.tensorflow.lite.examples.classification.tflite.Classifier.Recognition;

public abstract class CameraActivity extends AppCompatActivity
    implements OnImageAvailableListener,
        Camera.PreviewCallback,
        View.OnClickListener,
        AdapterView.OnItemSelectedListener {
  private static final Logger LOGGER = new Logger();

  private static final int PERMISSIONS_REQUEST = 1;

  private static final String PERMISSION_CAMERA = Manifest.permission.CAMERA;
  protected int previewWidth = 0;
  protected int previewHeight = 0;
  private Handler handler;
  private HandlerThread handlerThread;
  private boolean useCamera2API;
  private boolean isProcessingFrame = false;
  private byte[][] yuvBytes = new byte[3][];
  private int[] rgbBytes = null;
  private int yRowStride;
  private Runnable postInferenceCallback;
  private Runnable imageConverter;
  private LinearLayout bottomSheetLayout;
  private LinearLayout gestureLayout;
  private BottomSheetBehavior sheetBehavior;
  protected TextView recognitionTextView,
      recognition1TextView,
      recognition2TextView,
      recognitionValueTextView,
      recognition1ValueTextView,
      recognition2ValueTextView;
  protected TextView frameValueTextView,
      cropValueTextView,
      cameraResolutionTextView,
      rotationTextView,
      inferenceTimeTextView;
  protected ImageView bottomSheetArrowImageView;
  private ImageView plusImageView, minusImageView;
```

```
private Spinner modelSpinner;
private Spinner deviceSpinner;
private TextView threadsTextView;

// private Model model = Model.QUANTIZED;
private Model model = Model.FLOAT;
private Device device = Device.CPU;
private int numThreads = -1;
 MediaPlayer mp,mp1,mp2;
@Override
protected void onCreate(final Bundle savedInstanceState) {
  LOGGER.d("onCreate " + this);
  super.onCreate(null);
  getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

  setContentView(R.layout.activity_camera);

  Toolbar toolbar = findViewById(R.id.toolbar);
  setSupportActionBar(toolbar);
  getSupportActionBar().setDisplayShowTitleEnabled(false);

  if (hasPermission()) {
    setFragment();
  } else {
    requestPermission();
  }

  threadsTextView = findViewById(R.id.threads);
  plusImageView = findViewById(R.id.plus);
  minusImageView = findViewById(R.id.minus);
  modelSpinner = findViewById(R.id.model_spinner);
  deviceSpinner = findViewById(R.id.device_spinner);
  bottomSheetLayout = findViewById(R.id.bottom_sheet_layout);
  gestureLayout = findViewById(R.id.gesture_layout);
  sheetBehavior = BottomSheetBehavior.from(bottomSheetLayout);
  bottomSheetArrowImageView = findViewById(R.id.bottom_sheet_arrow);

  ViewTreeObserver vto = gestureLayout.getViewTreeObserver();
  vto.addOnGlobalLayoutListener(
      new ViewTreeObserver.OnGlobalLayoutListener() {
        @Override
        public void onGlobalLayout() {
          if (Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAN) {
            gestureLayout.getViewTreeObserver().removeGlobalOnLayoutListener(this);
          } else {
            gestureLayout.getViewTreeObserver().removeOnGlobalLayoutListener(this);
          }
          //              int width = bottomSheetLayout.getMeasuredWidth();
          int height = gestureLayout.getMeasuredHeight();

          sheetBehavior.setPeekHeight(height);
```

```
      }
    });
sheetBehavior.setHideable(false);

sheetBehavior.setBottomSheetCallback(
    new BottomSheetBehavior.BottomSheetCallback() {
      @Override
      public void onStateChanged(@NonNull View bottomSheet, int newState) {
        switch (newState) {
          case BottomSheetBehavior.STATE_HIDDEN:
            break;
          case BottomSheetBehavior.STATE_EXPANDED:
            {
              bottomSheetArrowImageView.setImageResource(R.drawable.icn_chevron_down);
            }
            break;
          case BottomSheetBehavior.STATE_COLLAPSED:
            {
              bottomSheetArrowImageView.setImageResource(R.drawable.icn_chevron_up);
            }
            break;
          case BottomSheetBehavior.STATE_DRAGGING:
            break;
          case BottomSheetBehavior.STATE_SETTLING:
            bottomSheetArrowImageView.setImageResource(R.drawable.icn_chevron_up);
            break;
        }
      }

      @Override
      public void onSlide(@NonNull View bottomSheet, float slideOffset) {}
    });

recognitionTextView = findViewById(R.id.detected_item);
recognitionValueTextView = findViewById(R.id.detected_item_value);
recognition1TextView = findViewById(R.id.detected_item1);
recognition1ValueTextView = findViewById(R.id.detected_item1_value);
recognition2TextView = findViewById(R.id.detected_item2);
recognition2ValueTextView = findViewById(R.id.detected_item2_value);

frameValueTextView = findViewById(R.id.frame_info);
cropValueTextView = findViewById(R.id.crop_info);
cameraResolutionTextView = findViewById(R.id.view_info);
rotationTextView = findViewById(R.id.rotation_info);
inferenceTimeTextView = findViewById(R.id.inference_info);

modelSpinner.setOnItemSelectedListener(this);
deviceSpinner.setOnItemSelectedListener(this);

plusImageView.setOnClickListener(this);
minusImageView.setOnClickListener(this);
```

```
    model = Model.valueOf(modelSpinner.getSelectedItem().toString().toUpperCase());
    device = Device.valueOf(deviceSpinner.getSelectedItem().toString());
    numThreads = Integer.parseInt(threadsTextView.getText().toString().trim());
  }

  protected int[] getRgbBytes() {
    imageConverter.run();
    return rgbBytes;
  }

  protected int getLuminanceStride() {
    return yRowStride;
  }

  protected byte[] getLuminance() {
    return yuvBytes[0];
  }

  /** Callback for android.hardware.Camera API */
  @Override
  public void onPreviewFrame(final byte[] bytes, final Camera camera) {
    if (isProcessingFrame) {
      LOGGER.w("Dropping frame!");
      return;
    }

    try {
      // Initialize the storage bitmaps once when the resolution is known.
      if (rgbBytes == null) {
        Camera.Size previewSize = camera.getParameters().getPreviewSize();
        previewHeight = previewSize.height;
        previewWidth = previewSize.width;
        rgbBytes = new int[previewWidth * previewHeight];
        onPreviewSizeChosen(new Size(previewSize.width, previewSize.height), 90);
      }
    } catch (final Exception e) {
      LOGGER.e(e, "Exception!");
      return;
    }

    isProcessingFrame = true;
    yuvBytes[0] = bytes;
    yRowStride = previewWidth;

    imageConverter =
        new Runnable() {
          @Override
          public void run() {
            ImageUtils.convertYUV420SPToARGB8888(bytes, previewWidth, previewHeight,
rgbBytes);
```

```
        }
      };

  postInferenceCallback =
      new Runnable() {
        @Override
        public void run() {
          camera.addCallbackBuffer(bytes);
          isProcessingFrame = false;
        }
      };
  processImage();
}

/** Callback for Camera2 API */
@Override
public void onImageAvailable(final ImageReader reader) {
  // We need wait until we have some size from onPreviewSizeChosen
  if (previewWidth == 0 || previewHeight == 0) {
    return;
  }
  if (rgbBytes == null) {
    rgbBytes = new int[previewWidth * previewHeight];
  }
  try {
    final Image image = reader.acquireLatestImage();

    if (image == null) {
      return;
    }

    if (isProcessingFrame) {
      image.close();
      return;
    }
    isProcessingFrame = true;
    Trace.beginSection("imageAvailable");
    final Plane[] planes = image.getPlanes();
    fillBytes(planes, yuvBytes);
    yRowStride = planes[0].getRowStride();
    final int uvRowStride = planes[1].getRowStride();
    final int uvPixelStride = planes[1].getPixelStride();

    imageConverter =
        new Runnable() {
          @Override
          public void run() {
            ImageUtils.convertYUV420ToARGB8888(
                yuvBytes[0],
                yuvBytes[1],
                yuvBytes[2],
```

```
            previewWidth,
            previewHeight,
            yRowStride,
            uvRowStride,
            uvPixelStride,
            rgbBytes);
        }
      };

    postInferenceCallback =
        new Runnable() {
          @Override
          public void run() {
            image.close();
            isProcessingFrame = false;
          }
        };

    processImage();
  } catch (final Exception e) {
    LOGGER.e(e, "Exception!");
    Trace.endSection();
    return;
  }
  Trace.endSection();
}

@Override
public synchronized void onStart() {
  LOGGER.d("onStart " + this);
  super.onStart();
}

@Override
public synchronized void onResume() {
  LOGGER.d("onResume " + this);
  super.onResume();

  handlerThread = new HandlerThread("inference");
  handlerThread.start();
  handler = new Handler(handlerThread.getLooper());

  mp = MediaPlayer.create(this, R.raw.hun);
  mp1 = MediaPlayer.create(this, R.raw.ten);
  mp2 = MediaPlayer.create(this, R.raw.five);

/*   mp.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
      mp.release();
    }
```

```
    });
    mp1.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
      @Override
      public void onCompletion(MediaPlayer mp) {
        mp.release();
      }
    });
    mp2.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
      @Override
      public void onCompletion(MediaPlayer mp) {
        mp.release();
      }
    });*/
  }

  @Override
  public synchronized void onPause() {
    LOGGER.d("onPause " + this);

    handlerThread.quitSafely();
    try {
      handlerThread.join();
      handlerThread = null;
      handler = null;
    } catch (final InterruptedException e) {
      LOGGER.e(e, "Exception!");
    }

    super.onPause();
  }

  @Override
  public synchronized void onStop() {
    LOGGER.d("onStop " + this);
    super.onStop();
  }

  @Override
  public synchronized void onDestroy() {
    LOGGER.d("onDestroy " + this);
    super.onDestroy();
  }

  protected synchronized void runInBackground(final Runnable r) {
    if (handler != null) {
      handler.post(r);
    }
  }

  @Override
  public void onRequestPermissionsResult(
```

```
      final int requestCode, final String[] permissions, final int[] grantResults) {
    if (requestCode == PERMISSIONS_REQUEST) {
      if (allPermissionsGranted(grantResults)) {
        setFragment();
      } else {
        requestPermission();
      }
    }
  }

  private static boolean allPermissionsGranted(final int[] grantResults) {
    for (int result : grantResults) {
      if (result != PackageManager.PERMISSION_GRANTED) {
        return false;
      }
    }
    return true;
  }

  private boolean hasPermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
      return checkSelfPermission(PERMISSION_CAMERA) ==
PackageManager.PERMISSION_GRANTED;
    } else {
      return true;
    }
  }

  private void requestPermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
      if (shouldShowRequestPermissionRationale(PERMISSION_CAMERA)) {
        Toast.makeText(
                CameraActivity.this,
                "Camera permission is required for this demo",
                Toast.LENGTH_LONG)
            .show();
      }
      requestPermissions(new String[] {PERMISSION_CAMERA}, PERMISSIONS_REQUEST);
    }
  }

  // Returns true if the device supports the required hardware level, or better.
  private boolean isHardwareLevelSupported(
      CameraCharacteristics characteristics, int requiredLevel) {
    int deviceLevel =
characteristics.get(CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL);
    if (deviceLevel ==
CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL_LEGACY) {
      return requiredLevel == deviceLevel;
    }
    // deviceLevel is not LEGACY, can use numerical sort
```

```
    return requiredLevel <= deviceLevel;
  }

  private String chooseCamera() {
    final CameraManager manager = (CameraManager)
getSystemService(Context.CAMERA_SERVICE);
    try {
      for (final String cameraId : manager.getCameraIdList()) {
        final CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);

        // We don't use a front facing camera in this sample.
        final Integer facing = characteristics.get(CameraCharacteristics.LENS_FACING);
        if (facing != null && facing == CameraCharacteristics.LENS_FACING_FRONT) {
          continue;
        }

        final StreamConfigurationMap map =
            characteristics.get(CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);

        if (map == null) {
          continue;
        }

        // Fallback to camera1 API for internal cameras that don't have full support.
        // This should help with legacy situations where using the camera2 API causes
        // distorted or otherwise broken previews.
        useCamera2API =
            (facing == CameraCharacteristics.LENS_FACING_EXTERNAL)
                || isHardwareLevelSupported(
                    characteristics,
CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL_FULL);
        LOGGER.i("Camera API lv2?: %s", useCamera2API);
        return cameraId;
      }
    } catch (CameraAccessException e) {
      LOGGER.e(e, "Not allowed to access camera");
    }

    return null;
  }

  protected void setFragment() {
    String cameraId = chooseCamera();

    Fragment fragment;
    if (useCamera2API) {
      CameraConnectionFragment camera2Fragment =
          CameraConnectionFragment.newInstance(
              new CameraConnectionFragment.ConnectionCallback() {
                @Override
                public void onPreviewSizeChosen(final Size size, final int rotation) {
```

```
                previewHeight = size.getHeight();
                previewWidth = size.getWidth();
                CameraActivity.this.onPreviewSizeChosen(size, rotation);
              }
            },
            this,
            getLayoutId(),
            getDesiredPreviewFrameSize());

      camera2Fragment.setCamera(cameraId);
      fragment = camera2Fragment;
    } else {
      fragment =
          new LegacyCameraConnectionFragment(this, getLayoutId(),
getDesiredPreviewFrameSize());
    }

    getFragmentManager().beginTransaction().replace(R.id.container, fragment).commit();
  }

  protected void fillBytes(final Plane[] planes, final byte[][] yuvBytes) {
    // Because of the variable row stride it's not possible to know in
    // advance the actual necessary dimensions of the yuv planes.
    for (int i = 0; i < planes.length; ++i) {
      final ByteBuffer buffer = planes[i].getBuffer();
      if (yuvBytes[i] == null) {
        LOGGER.d("Initializing buffer %d at size %d", i, buffer.capacity());
        yuvBytes[i] = new byte[buffer.capacity()];
      }
      buffer.get(yuvBytes[i]);
    }
  }

  protected void readyForNextImage() {
    if (postInferenceCallback != null) {
      postInferenceCallback.run();
    }
  }

  protected int getScreenOrientation() {
    switch (getWindowManager().getDefaultDisplay().getRotation()) {
      case Surface.ROTATION_270:
        return 270;
      case Surface.ROTATION_180:
        return 180;
      case Surface.ROTATION_90:
        return 90;
      default:
        return 0;
    }
  }
```

```
boolean hun = false;
  boolean five = false;
  boolean ten = false;
  @UiThread
  protected void showResultsInBottomSheet(List<Recognition> results) {

    if (results != null && results.size() >= 3) {
      Recognition recognition = results.get(0);
      if (recognition != null) {
        if (recognition.getTitle() != null) recognitionTextView.setText(recognition.getTitle());
        if (recognition.getConfidence() != null)
          recognitionValueTextView.setText(
                  String.format("%.2f", (100 * recognition.getConfidence())) + "%");
        float confi = 100 * recognition.getConfidence();
        try {
          if (!five && recognitionTextView.getText().toString().equalsIgnoreCase("500") &&
confi>99 ) {
            mp2.start();
            five =true;
            ten = false;
            hun = false;
          } else if (!hun&& recognitionTextView.getText().toString().equalsIgnoreCase("100")&&
confi>99) {
            mp.start();
            hun = true;
            five =false;
            ten = false;
          } else if (!ten&&recognitionTextView.getText().toString().equalsIgnoreCase("10")&&
confi>90 ) {
            mp1.start();
            ten  =true;
            five =false;
            hun = false;
          }
        }catch (Exception e){
          e.printStackTrace();
        }
      }

      Recognition recognition1 = results.get(1);
      if (recognition1 != null) {
        if (recognition1.getTitle() != null) recognition1TextView.setText(recognition1.getTitle());
        if (recognition1.getConfidence() != null)
          recognition1ValueTextView.setText(
              String.format("%.2f", (100 * recognition1.getConfidence())) + "%");
      }

      Recognition recognition2 = results.get(2);
      if (recognition2 != null) {
        if (recognition2.getTitle() != null) recognition2TextView.setText(recognition2.getTitle());
        if (recognition2.getConfidence() != null)
```

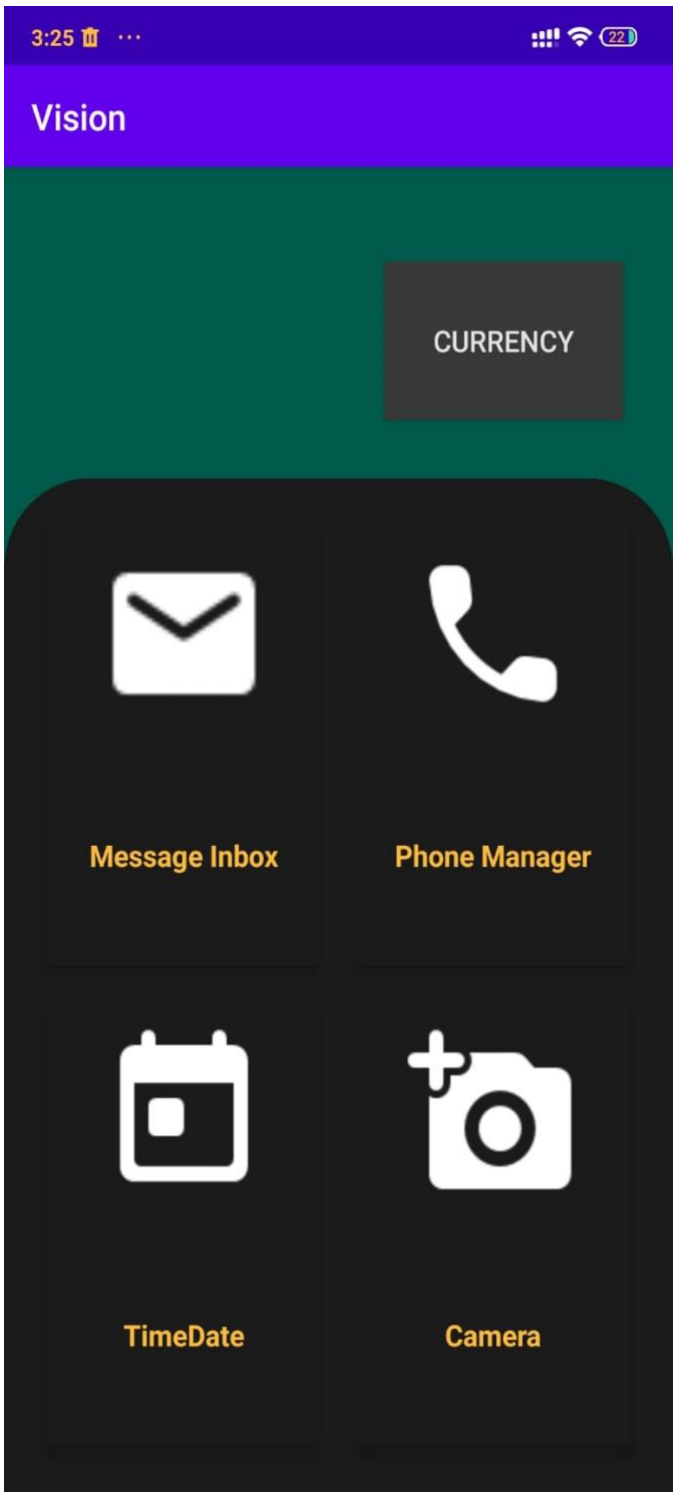
```
        recognition2ValueTextView.setText(
            String.format("%.2f", (100 * recognition2.getConfidence())) + "%");
      }
    }
  }

  protected void showFrameInfo(String frameInfo) {
    frameValueTextView.setText(frameInfo);
  }

  protected void showCropInfo(String cropInfo) {
    cropValueTextView.setText(cropInfo);
  }

  protected void showCameraResolution(String cameraInfo) {
    cameraResolutionTextView.setText(cameraInfo);
  }

  protected void showRotationInfo(String rotation) {
    rotationTextView.setText(rotation);
  }

  protected void showInference(String inferenceTime) {
    inferenceTimeTextView.setText(inferenceTime);
  }

  protected Model getModel() {
    return model;
  }

  private void setModel(Model model) {
    if (this.model != model) {
      LOGGER.d("Updating  model: " + model);
      this.model = model;
      onInferenceConfigurationChanged();
    }
  }

  protected Device getDevice() {
    return device;
  }

  private void setDevice(Device device) {
    if (this.device != device) {
      LOGGER.d("Updating  device: " + device);
      this.device = device;
      final boolean threadsEnabled = device == Device.CPU;
      plusImageView.setEnabled(threadsEnabled);
      minusImageView.setEnabled(threadsEnabled);
      threadsTextView.setText(threadsEnabled ? String.valueOf(numThreads) : "N/A");
      onInferenceConfigurationChanged();
```

```
  }
}

protected int getNumThreads() {
  return numThreads;
}

private void setNumThreads(int numThreads) {
  if (this.numThreads != numThreads) {
    LOGGER.d("Updating  numThreads: " + numThreads);
    this.numThreads = numThreads;
    onInferenceConfigurationChanged();
  }
}

protected abstract void processImage();

protected abstract void onPreviewSizeChosen(final Size size, final int rotation);

protected abstract int getLayoutId();

protected abstract Size getDesiredPreviewFrameSize();

protected abstract void onInferenceConfigurationChanged();

@Override
public void onClick(View v) {
  if (v.getId() == R.id.plus) {
    String threads = threadsTextView.getText().toString().trim();
    int numThreads = Integer.parseInt(threads);
    if (numThreads >= 9) return;
    setNumThreads(++numThreads);
    threadsTextView.setText(String.valueOf(numThreads));
  } else if (v.getId() == R.id.minus) {
    String threads = threadsTextView.getText().toString().trim();
    int numThreads = Integer.parseInt(threads);
    if (numThreads == 1) {
      return;
    }
    setNumThreads(--numThreads);
    threadsTextView.setText(String.valueOf(numThreads));
  }
}

@Override
public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
  if (parent == modelSpinner) {
    setModel(Model.valueOf(parent.getItemAtPosition(pos).toString().toUpperCase()));
  } else if (parent == deviceSpinner) {
    setDevice(Device.valueOf(parent.getItemAtPosition(pos).toString()));
  }
```

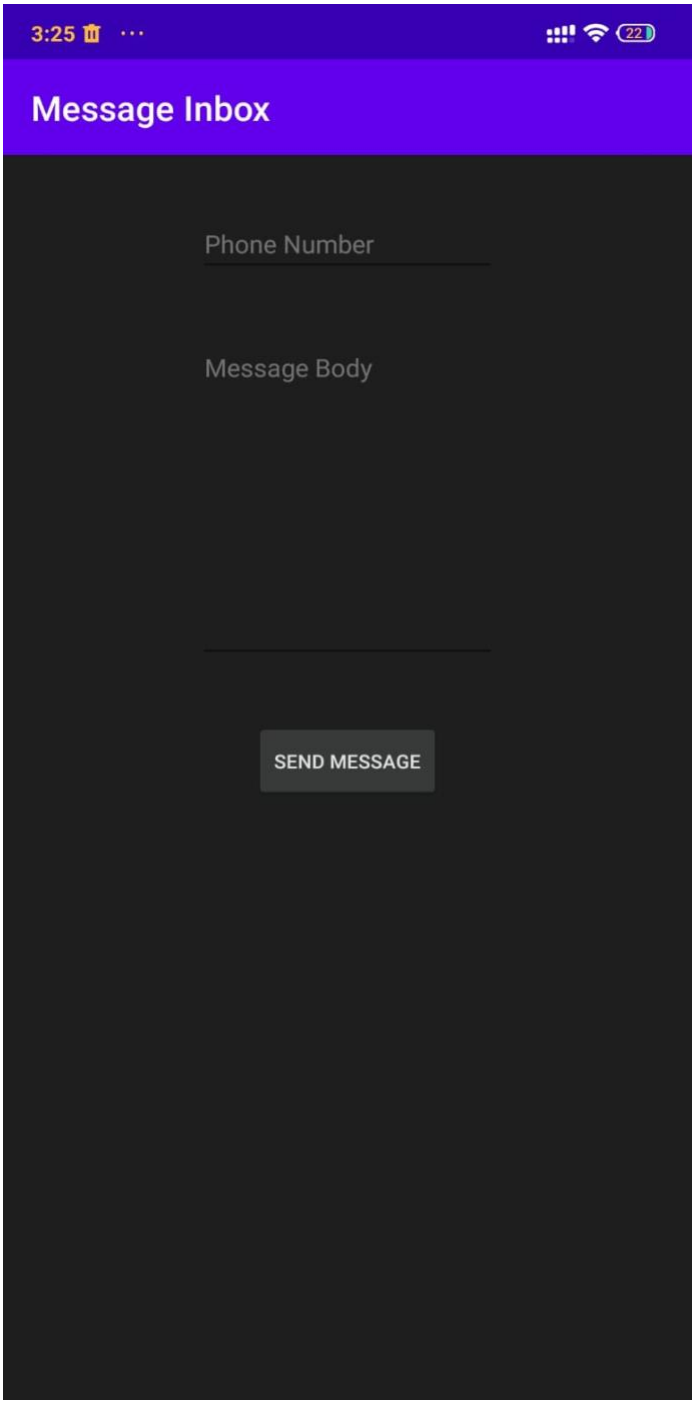
```
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
      // Do nothing.
    }
  }
```

SNAPSHOTS

Home Page:



Message:



Phone Activity:

Time and Date:

Camera and Currency Detection:

RESULT

The built-in modules of text to speech (pyttsx3) and speech to text (speech recognition library by Google) in python provide a good accuracy and also provide an easy and quick way to convert the text. The speech-to-text recognized the words with 96.25% accuracy with 4 different voice samples each containing 20 different inputs in a moderate to quiet environment. The BERT model on SQuAD dataset for the question answering feature in the Wikipedia module showed an Exact Match accuracy of 80.88% which is the percentage of predictions that match any one of the ground truths answers exactly, and the F1 score was found to be 88.49%. Results showed that we were able to run our software on the three most popular sites: Google, Gmail and Wikipedia. The software was run on each of them separately. The software could send an email effectively using the commands from the user. The software also provided an accurate answer to the question the user asked on Wikipedia. The software managed to summarize the text in Wikipedia accurately and thus we were able to test and build a software that will make the website easily, quickly and efficiently accessible for the visually impaired.

CONCLUSION

In this project, we presented a modular solution to improve application based accessibility for the visually impaired. The virtual assistant is operating system independent and does not rely on keyboard inputs from the user to maximize ease of use and aims to provide a hassle-free experience for the user. Through speech to text and text to speech interfaces, the user can communicate with and customize the system. We presented the system design and methodology of the three modules that is currently implemented. The Wikipedia module uses a BERT model on the SQuAD dataset to answer user queries quickly and accurately. The Exact Match was found to be 80.88%. The virtual assistant provides an easy way to access any website for the visually impaired. It eliminates the need to remember complex keyboard commands or the use of screen readers. The assistant is not only a great way to interact with the websites but also an effective way to do so. The software works as a steppingstone towards Web 3.0 where everything will work on voice commands.

BIBLIOGRAPHY

- https://www.w3schools.com
- https://www.geeksforgeeks_org
- https://freefrontend.com
- https://css-tricks.com/
- https://www.takeiteasyengineers.com
- https://dev.to/mychi_darko/php-tips-and