Ultimatrix
- An Android App for Visually impaired

by

Alwin Vishal P J (2147211)

Sandeep G(2147232)

Vignesh Saminathan (2147235)

Under the guidance of

Dr Suresh T

Specialization project report submitted in partial fulfillment of the requirements of Vth trimester MCA,

CHRIST (Deemed to be University)

December 2022

CERTIFICATE

This is to certify that the report titled ***Ultimatrix*** *is a bonafide record of work done by Alwin Vishal P J (2147211), Sandeep G(2147232), and Vignesh Saminathan (2147235) of CHRIST(Deemed to be University), Bangalore, in partial fulfillment of the requirements of Vth Trimester MCA during the year 2022.*

Head of the Department **Project Guide**

Valued-by:

	Name	:Alwin Vishal P J Sandeep G Vignesh Saminathan
1.	Register Number	:2147211 2147232 2147235
	Examination Centre	: CHRIST (Deemed to be University)
2.	Date of Exam	:

TABLE OF CONTENTS

ACKNOWLEDGEMENT

During the course of the project we received encouragement and valuable support from many people. It is a pleasant aspect that we now have the opportunity to express our gratitude to acknowledge their contributions towards our project. First and foremost, we offer this endeavor to Almighty God for the wisdom he bestowed upon us, the strength, peace of mind and good health during the whole process. No words can articulate our deepest gratitude to the Lord for his inspiration and guidance.

We express our deep sense of gratitude to **Rev. Fr. Dr. Abraham. V. M**, Vice Chancellor of CHRIST (Deemed to be University) for providing the necessary facilities that helped us in accomplishing this work. We feel a deep sense of gratitude to **Rev. Fr. Dr. Joseph C.C**, Pro Vice Chancellor of CHRIST (Deemed to be University) for his moral support.

We are deeply indebted to our Head of the department **Dr.Ashok Immanuel** , Department of Computer Science, CHRIST (Deemed to be University), for rendering us invaluable support during the course of this project work.We acknowledge our sincere thanks to **Dr. Shoney Sebastian**, Coordinator of MCA Department of Computer Science, CHRIST (Deemed to be University), for his motivation and valuable guidance throughout the course of the project. We are highly indebted to our project guide, **Dr. Suresh T**, Associate Professor, Department of Computer Science, CHRIST (Deemed to be University), for guiding us with the true spirit throughout the project with profound sincerity.

We would like to thank the faculty members of the Department of Computer Science, CHRIST (Deemed to be University) for providing us advice and motivation that helped us stay on track. We consider it a pleasant duty to acknowledge the abundant help received from all our friends and classmates who were with us to share our happiness and agony. The valuable suggestions that were contributed helped us to a great extent of completing the project successfully.

ABSTRACT

It's a known fact that the estimated number of visually impaired people in the world is about 285 million, approximately equal to 20% of the Indian Population. They suffer regular and constant challenges in Navigation especially when they are on their own. They are mostly dependent on someone for even accessing their basic day-to-day needs. So, it's quite a challenging task and the technological solution for them is of utmost importance and much needed. One such try from our side is that we came up with an Integrated Machine Learning System which allows the Blind Victims to identify and classify Real Time Based Common day-to-day Objects and generate voice feedback and calculates distance which produces warnings whether he/she is very close or far away from the object. The same system can be used for Obstacle Detection Mechanism.

1. INTRODUCTION

1.1.Project Overview

Ultimatrix is an app which is used for those specially abled people that helps for those who really want to overcome those challenges. It provides a platform where people are able to know what others are talking about with the help of their mobile. For the people who can't hear it will convert other's speech into text so that they are able to understand. Likewise for the people who can't speak they can type it and the app will speak for them. Similarly for blind it will detect the objects around them and it will tell it to the people.

1.2.Existing System

- Voice4u AAC:
  - This app provides a picture-based communication system for people with speech challenges.

- Proloquo2Go:
  - This app was created to be a daily communication tool for people with speech challenges.

- Be My Eyes:
  - This app connects blind and low-vision people with sighted volunteers through live video calls.

Disadvantages of Existing system :

- There are so many apps which can help specially abled people, but they are focusing on a particular problem.
- Some of the apps are highly paid apps where in such cases some people can't afford it.
- Also, many apps are compatible with a particular operating system. For example, Proloquo2Go is only available for the iOS ecosystem.

1.3. Proposed System

Technology has evolved very rapidly over the past 2 decades. Various researches are taking place and there is a constant growth in technology in order to make our lives easier. But there are a lot of people(specially abled) in this world who face lots of difficulties in understanding and experiencing our world the way we do. These specially abled people find it very difficult to understand and communicate with other people. So thus we came up with the idea of Ultimatrix. It is an app which is used for those specially abled people that helps for those who really want to overcome those challenges. It provides a platform where people are able to know what others are talking about with the help of their mobile. For the people who can't hear it will convert other's speech into text so that they are able to understand. Likewise for the people who can't speak they can type it and the app will speak for them. Similarly for blind it will detect the objects around them and it will tell it to the people.

Advantages:

- The proposed system i.e Ultimatrix is able to solve some of the limitations in the existing system.
- It can run on any mobile OS platform.
- It will be freely available to the specially abled people.
- Moreover, since this integrates the features like speech-to-text, text-to-speech, etc., this system will be an all in one platform which can serve the specially abled people.

1.4. Literature Survey

A new separation and word spotting algorithms were acclimated to recognize handwritten Arabic words. [2] in "Segmentation-Predicated And Segmentation-Free Methods for Spotting Handwritten Arabic Words" marginally integrates ambiguity into a complete Arabic word; that is, the amount of words in observe that are distinct from one another solely by the presence of characters with the same kindred base shape, however having or not having associated dots isn't immensely colossal.
"Multilingual OCR for Indic Scripts" [3 -5] optically canvassed that, even with homogeneous architecture, performance in languages of Indian is arduous compared to English. In the latter case, the data required for RNN training would be less and the output space would be more minute. In this approach, the script is identified at a word level, for the recognition of the word demonstrated for English language and twelve Indian languages.

"Matching Ottoman Words: An image retrieval approach to historical document indexing" [6-8] was about that the characteristics of Ottoman documents needed an automatic transcription, but the character apperception predicated system was not given any efficient and copacetic results. This study faces the quandary in image retrieval and proposes a solution that was predicated on image matching techniques because of containing consequential images and signatures in Ottoman documents. Additionally, it has to face a drawback of causing most of these Ottoman documents to be inaccessible because many documents are in manuscript and manual transcription format. Additionally, the time taken for the indexing of these Ottoman documents and text was long. So, it leads to an inaccessible state.

"Handwritten Word Spotting with Redressed Attributes" [8] suggested that by utilizing an attributes-based approach that leads to a low dimensional, it has a fine-tuned-length representation and cumulated representation of word images and strings, so it is expeditious to compare and compute. It allows us to perform queries indistinctly like a query-by string. The query is an image and string. A calibration scheme was proposed to ameliorate challenging dataset results by Canonical Correlation Analysis. It was tested on two public datasets exhibiting state-of-the-art results. With all it had a disadvantage of words that were labeled were not utilized in training.

Among all the senses, the ability to see through the eyes is one of the most important senses in human beings. And the loss of this ability severely affects all the possible movements an individual is likely to do in his/her life. Since such people are not expected to grow in their profession as much as an abled person, they often experience a violation of their rights and also discrimination on social platforms & at the workplace. Government and social groups are actively participating in making the lives of the visually impaired more convenient and safer by organizing campaigns and providing education about the new tools and technology.

In [7], a method is proposed where the image is captured using the camera and the captured image is scanned from left to right for detection of the obstacle and then sound is generated. Sound is generated by analyzing the image where the top image is altered into high frequency and the bottom portion into low frequency sound. And the loudness depends on the brightness of the image as well. The image is differentiated into foreground and background using image processing techniques. The foreground is assigned with high intensity value and background are assigned with low intensity values. Then this image is converted into stereo sound where the amplitude of the sound is directly proportional to intensity of image pixels, and the frequency of sound is inversely proportional to vertical orientation of pixels.

VisualPal was a mobile application for Object Recognition to aid the visually impaired [5]. It detects the direction of maximum brightness and major colors in the image. It made use of Artificial Neural Network Technology along with Euclidean Distance measures together. It captured a video and categorized it into various frames. All frames are compared with Previous frames and response will be given based on stored objects information.An Intelligent Assistant for Blind named Blind Reader is an Android Application [6]. It used Speech Synthesis and Text Recognition to recognize the text from a pdf file and synthesize it to the user. A text document or a .ppt file is converted into a .pdf by recognizing a collection of words. As the application is built on android, it uses predefined APIs for text-to-speech conversion which makes the process even more efficient. However, it doesn't recognize Text through image Google's Vision API is used.

2. SYSTEM ANALYSIS AND REQUIREMENTS

2.1.Problem Definition

Most of the apps are not highly available for free. People can't access the full features of the designated app. And also every app is mainly focusing on a particular cause and sometimes it may run on a particular OS and also there is always a worry on the user requirements that changes drastically combination of multiple problems with different use cases in a single application is hard to find.

2.2. Requirement Specification

Requirement specification details the various requirements of the proposed system. It details the functional and non-functional requirements of the system. It also includes analysis modeling, which contains UML diagrams describing the functionalities of the system, and the processes behind certain functionalities.

2.2.1 Functional Requirements

A functional requirement is a description of the service that the software must offer. It describes a software system or its components. A function is nothing but inputs to the software system, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. Following are the functional requirements of Semantic Search Engine:

Object Detection

In this phase, it tries to detect the object which is coming in front of the screen. It tries to segregate the objects according to the shape based on the dataset that has been trained. It is mainly for the visually challenged people.

Text detection OCR

In this phase, it tries to detect the texts that have been shown. The alphabets have already been trained with different styles. So whenever a text appears it will try to detect with maximum accuracy.

Currency detection

In this phase, it tries to detect the currencies that have been displayed. Whether it is 100, 500 or even 2000 it tries to detect and the result will be told with a voice.

Text to speech

In this phase, it will convert the text into speech. Whatever the input maybe it will convert the text using NLP techniques and Py audio.

Color detection

In this phase, it tries to detect the color of the object which has been shown in front of the camera. It mainly detects the main colors which have been in common.

2.2.2.Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements define how a system is supposed to be

● Scalability – the system should be able to handle all the process that has been thrown into this system.

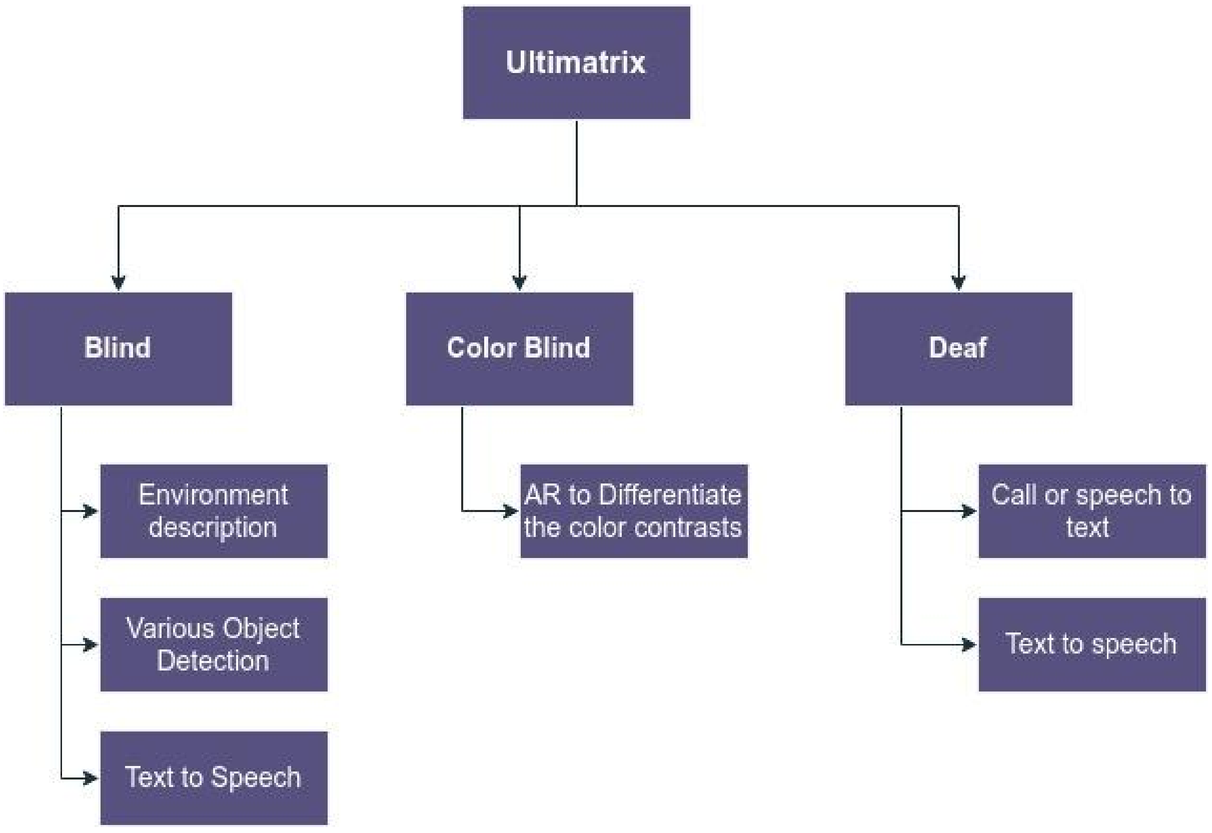
● Reliability – the system should be reliable for the people who are all using it and should not be a buggy software.

● Security – user details should be kept private in terms of the images or texts that has been imputed.

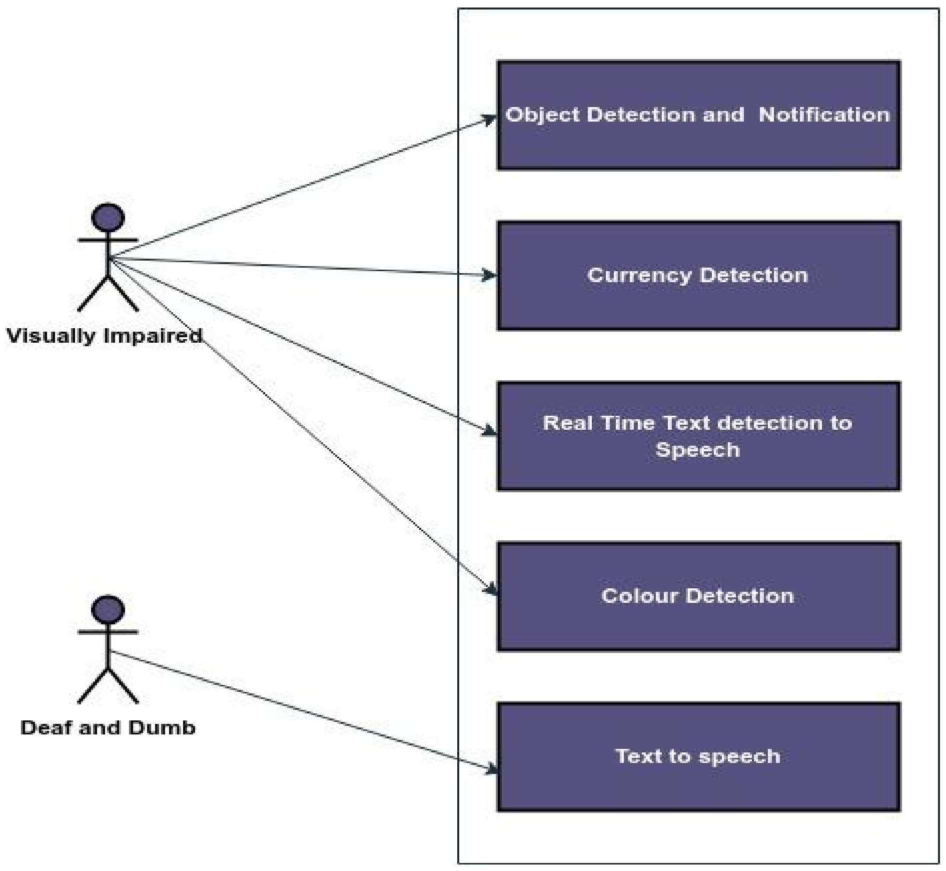
● Accuracy – the system should be able to produce the result with the maximum accuracy.

2.3. BLOCK DIAGRAM

A block diagram is a visual representation of a system that uses simple, labeled blocks that represent single or multiple items, entities or concepts, connected by lines to show relationships between them.

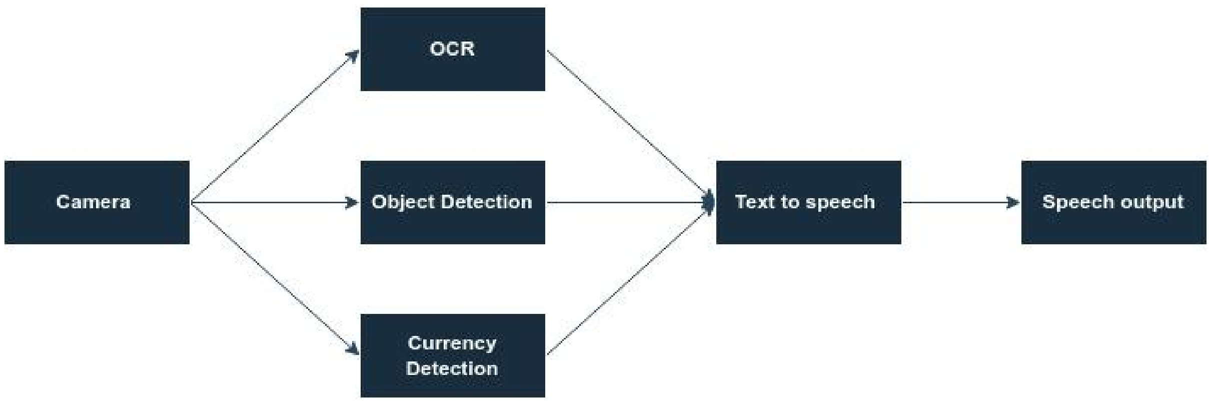


2.4.Use Case Diagram

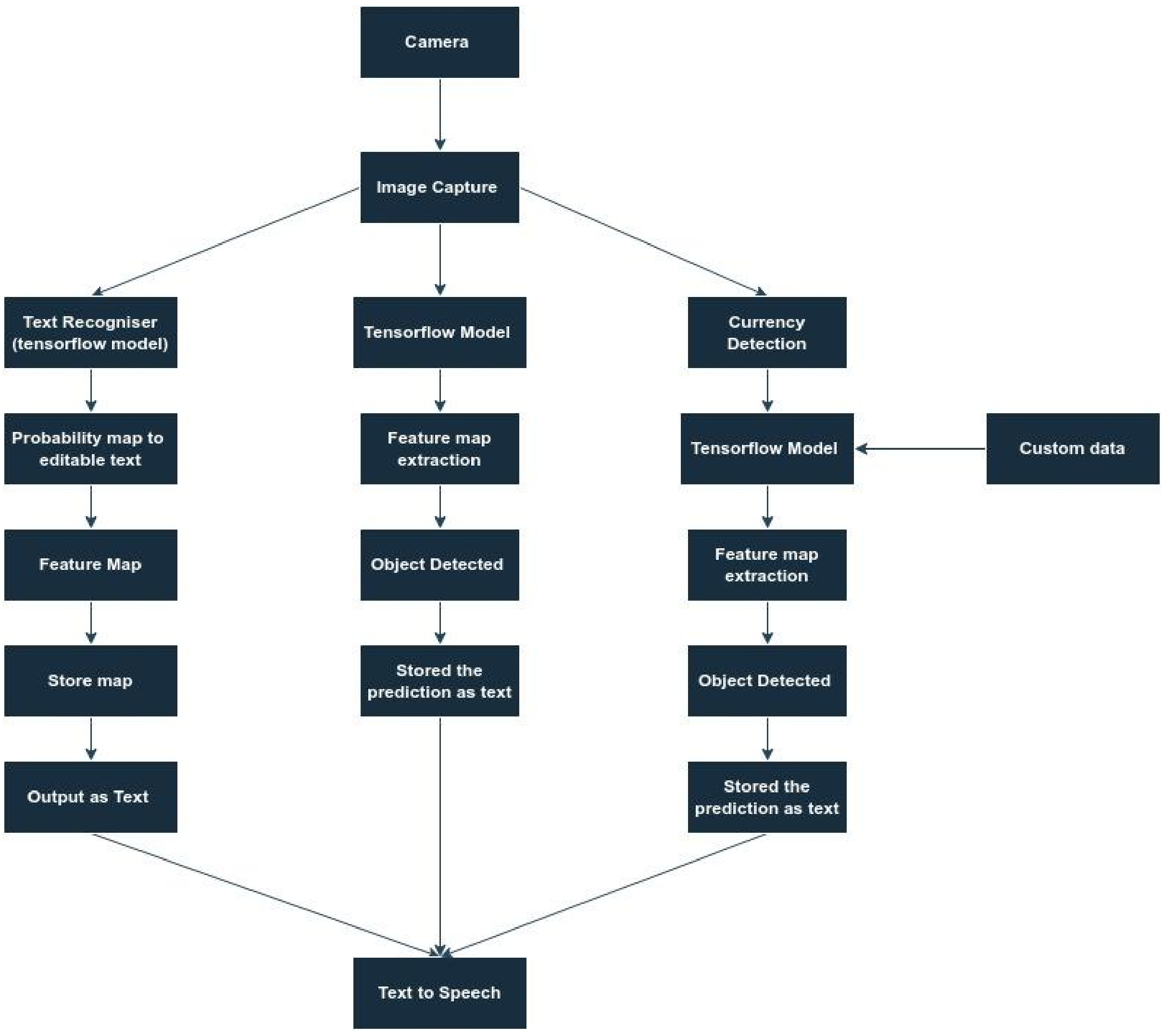


2.5.Data Flow Diagram

DFD level 0



DFD level 1



2.6. System Requirements

2.6.1. User Characteristics

Users should be able to open a webpage and use a web browser. Little technical knowledge is needed. Users should be able to type queries in English Text without making any grammar mistakes. The query will provide accurate answers that the user can see and comprehend.

2.6.2. Software and Hardware Requirements

Hardware Requirements

- Processor: Minimum 1 GHz; Recommended 2GHz or more.
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 32 GB; Recommended 64 GB or more.
- Memory (RAM): Minimum 1 GB; Recommended 4 GB or above.

Software Requirements

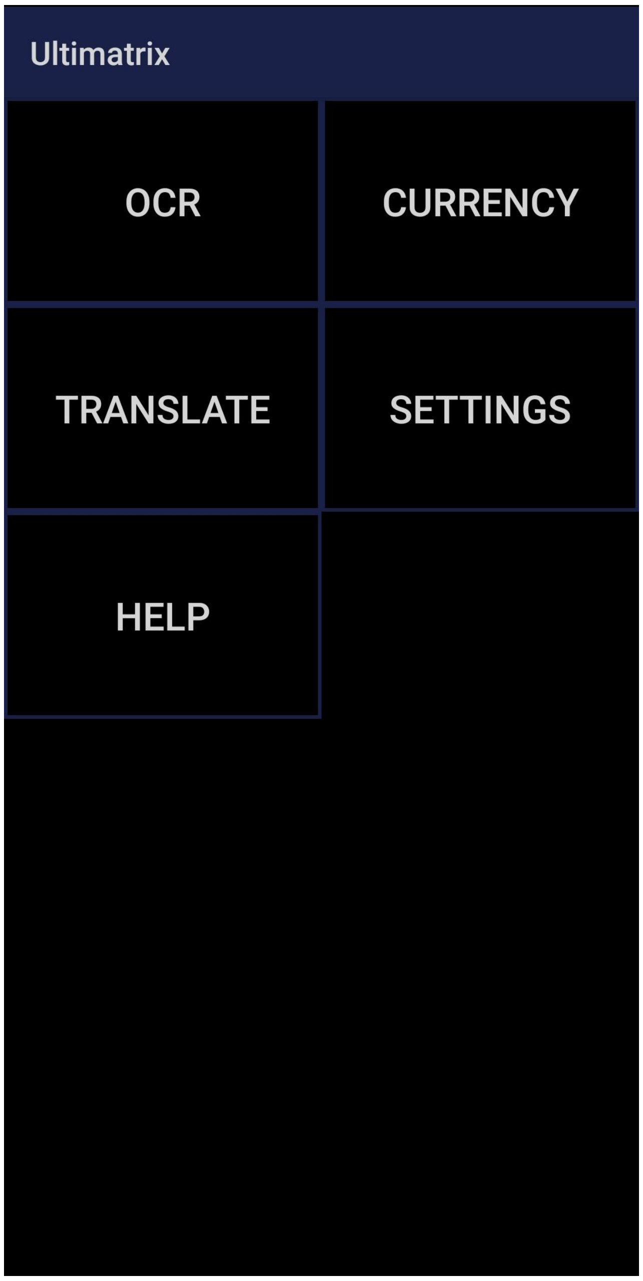
- Tensorflow / Tensorflow 2
- OpenCV
- Android Studio/Flutter
- Common Object in Context (COCO) dataset
- Cloud-hosting ML services
- Speech recognition
- Django, Python ML Packages

3. SYSTEM DESIGN

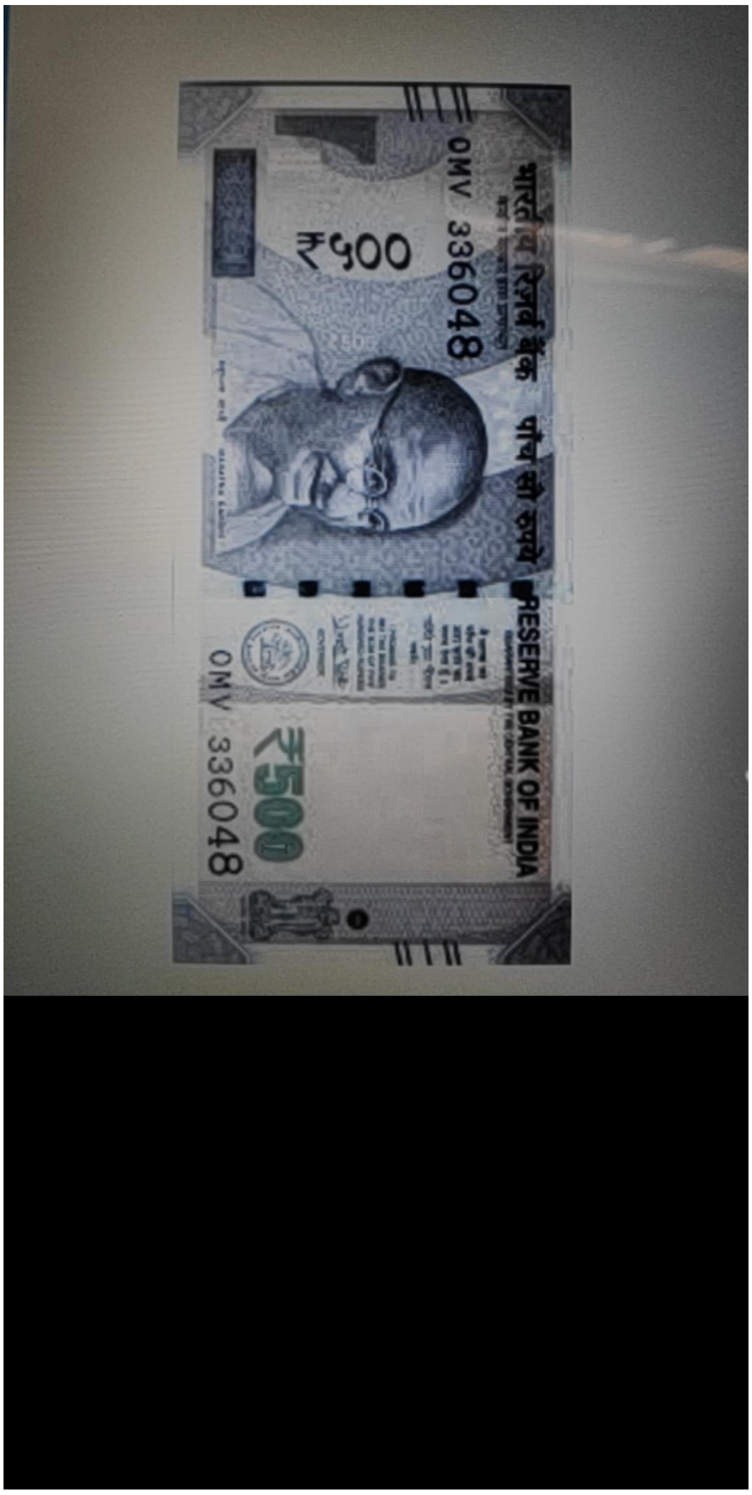
3.1. INTERFACE DESIGN AND PROCEDURAL DESIGN

3.5.1. USER INTERFACE DESIGN

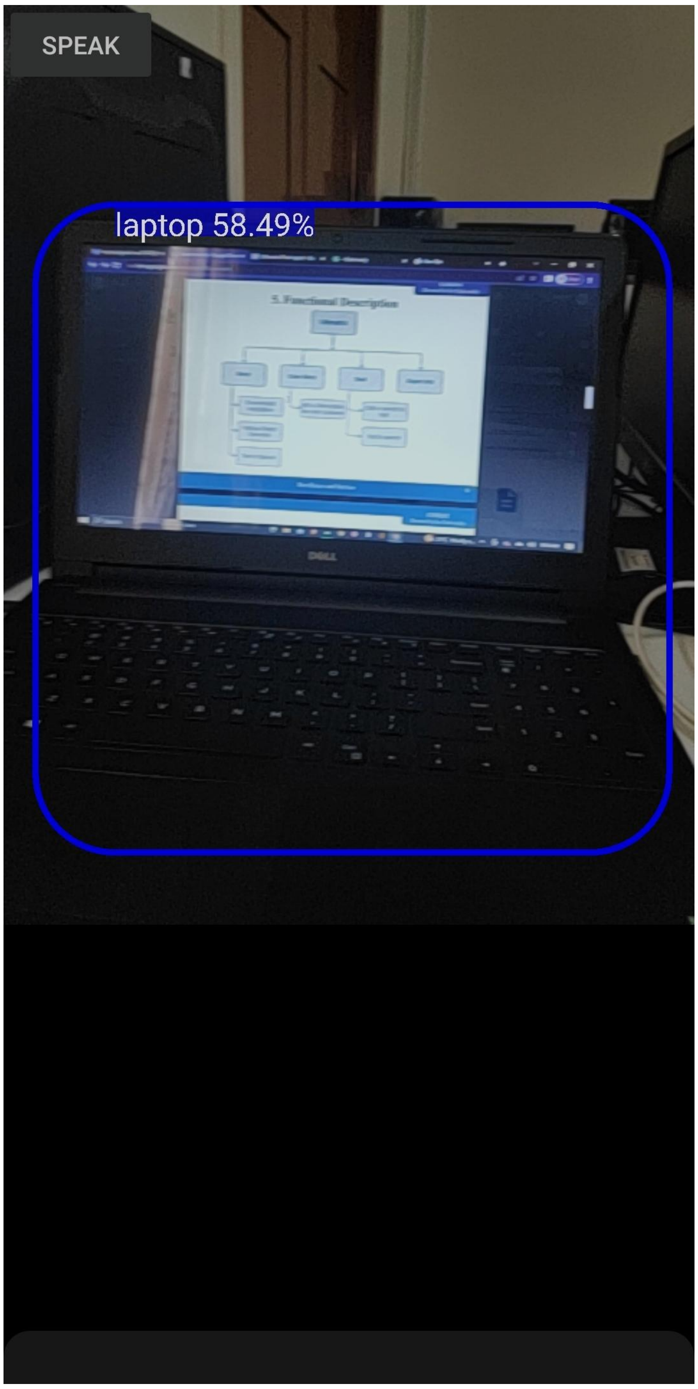
Main Menu



Currency Detection



Object Detection



4. IMPLEMENTATION

4.1. IMPLEMENTATION APPROACHES

Agile Methodology: The Agile Project Management Process is a value-centered method of project management that allows projects to get processed in small phases or cycles. The methodology is one that is extremely flexible and projects that exhibit dynamic traits would benefit from this process as you would find that project managers working in this environment treat milestones as “sprints”; the goal being to continuously adapt to abrupt changes from client feedback. It is best suited for small software projects made up of a highly collaborative team or a project that requires frequent iteration.

4.2. CODING STANDARD

- For better understanding and maintenance of the code, the header of different modules follows some standard format and information.
- Synopsis of the module about what the module does is provided.
- Different functions supported in the module along with their input output parameters
- Naming conventions for local variables, global variables, constants and functions
- Meaningful and understandable variable names help anyone to understand the reason for using it.
- It is better to avoid the use of digits in variable names.
- The name of the function must describe the reason for using the function clearly and briefly.
- Proper indentation is very important to increase the readability of the code. For making the code readable, the White spaces are used properly.
- There must be a space after giving a comma between two function arguments.
- Each nested block should be properly indented and spaced.
- Proper Indentation should be there at the beginning and at the end of each block in the program.
- All braces should start from a new line and the code following the end of braces also starts from a new line.

- All functions that encounter an error condition should either return a 0 or 1 for simplifying the debugging.On the other hand, Coding guidelines give some general suggestions regarding the coding style that to be followed for the betterment of understandability and readability of the code.
- Code should be easily understandable. Code should be well documented.The code should be properly commented for understanding easily. Comments regarding the statements increase the understandability of the code.

4.3. CODE

Main Menu

```
import android.content.Intent;
import android.os.Bundle;
//import android.os.StrictMode;
import android.speech.tts.TextToSpeech;
import android.support.annotation.IdRes;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;

import com.example.controller.app.currency.ClassifierActivity;
import com.example.controller.app.ocr.OcrCaptureActivity;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    private static final String TAG = "MainActivity";
    public static final boolean DEV_MODE = true;
    private static final int CHECK_TTS_CODE = 10;

    //    private GestureDetectorCompat mDetector;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Log.e(TAG, LogUtil.prependCallLocation("onCreate: "));
        Intent checkTTSIntent = new Intent();
        checkTTSIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
        startActivityForResult(checkTTSIntent, CHECK_TTS_CODE);
        super.onCreate(savedInstanceState);
        /*if (DEV_MODE) {
            StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
                    .detectAll() // or .detectAll() for all detectable
problems
                    .penaltyLog()
                    .build());
            StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
                    .detectAll()
                    .penaltyLog()
                    .penaltyDeath()
                    .build());
        }*/
        setContentView(R.layout.activity_main);
```

```
//        mDetector = new GestureDetectorCompat(this, new
MyGestureListener());
        findViewAndSetThisAsOnClickListener(R.id.btn_ocr,
R.id.btn_curr,R.id.btn_translate,R.id.btn_ocr_setting, R.id.btn_hlp);
    }

    private void findViewAndSetThisAsOnClickListener(@IdRes int ... viewIds)
{
        for(int viewId : viewIds) {
            findViewById(viewId).setOnClickListener(this);
        }
    }
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        if (requestCode == CHECK_TTS_CODE) {
            if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
                CTextToSpeech tts = new CTextToSpeech(this);
                tts.stop();
            }
            else {
                Intent installTTSIntent = new Intent();

installTTSIntent.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
                startActivity(installTTSIntent);
            }
        }
    }
    @Override
    public void onClick(View view) {
        Log.e(TAG, LogUtil.prependCallLocation("onClick: "+view.getId()+"
"+view.getContentDescription()));
        switch (view.getId()) {
            case R.id.btn_ocr:
                Log.e(TAG, LogUtil.prependCallLocation("onClick: OCR
button"));
                // launch Ocr capture activity.
                Intent intent = new Intent(this, OcrCaptureActivity.class);
                startActivity(intent);
                break;
            case R.id.btn_curr:
                Log.e(TAG, LogUtil.prependCallLocation("onClick: Currency
Button"));

                intent = new Intent(this, ClassifierActivity.class);
                startActivity(intent);
                break;
            case R.id.btn_translate:
                Log.e(TAG, LogUtil.prependCallLocation("onClick: "));
                intent = new Intent(this, TranslateActivity.class);
                startActivity(intent);
                break;
            case R.id.btn_hlp:
                Log.e(TAG, LogUtil.prependCallLocation("onClick: Help"));
                intent = new Intent(this, HelpActivity.class);
                startActivity(intent);
                break;
            case R.id.btn_ocr_setting:
                Log.e(TAG, LogUtil.prependCallLocation("onClick: OCR
Settings"));
                intent = new Intent(this, SettingsActivity.class);
                startActivity(intent);
        }
    }
```

OCR:

```
package com.example.controller.app.ocr;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Build;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.Toast;

import com.example.controller.app.CTextToSpeech;
import com.example.controller.app.LogUtil;
import com.example.controller.app.PreferenceManager;
import com.example.controller.app.R;
import com.example.controller.app.TextManagerActivity;
import com.example.controller.app.TextBlockUtil;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.vision.text.TextRecognizer;
//import com.example.controller.app.c2.CameraSource;
import java.io.IOException;
import java.util.Locale;

/**
 * Activity for the Ocr Detecting app.  This app detects text and displays
the value with the
 * rear facing camera. During detection overlay graphics are drawn to
indicate the position,
 * size, and contents of each TextBlock.
 */
public final class OcrCaptureActivity extends AppCompatActivity {
    private static final String TAG =
OcrCaptureActivity.class.getSimpleName();

    // Intent request code to handle updating play services if needed.
    private static final int RC_HANDLE_GMS = 9001;

    // Permission request codes need to be < 256
    private static final int RC_HANDLE_CAMERA_PERM = 2;
    private CameraSource mCameraSource;
    private CameraSourcePreview mPreview;
    private GraphicOverlay<OcrGraphic> mGraphicOverlay;

    // Helper objects for detecting taps and pinches.
    private ScaleGestureDetector scaleGestureDetector;
    private GestureDetector gestureDetector;

    // A TextToSpeech engine for speaking a String value.
    private CTextToSpeech tts;
    private OcrDetectorProcessor mDetectorProcessor;
    PreferenceManager manager;
    boolean autoFocus, useFlash, multiColumn;
    String langCode;
    /**
     * Initializes the UI and creates the detector pipeline.
     */
    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.ocr_capture);

        mPreview = findViewById(R.id.preview);
        mGraphicOverlay = findViewById(R.id.graphicOverlay);
        manager = new PreferenceManager(getApplicationContext());

        autoFocus = manager.getAutofocus();
        useFlash = manager.getFlash();
        multiColumn = manager.getMulticolumn();
        int ocrLangPos = manager.getOcrLangPos();
        String[] langCodes =
getResources().getStringArray(R.array.arr_ocr_lang_code);
        langCode = langCodes[ocrLangPos];
        // Check for the camera permission before accessing the camera.  If
the
        // permission is not granted yet, request permission.
        int rc = ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA);
        if (rc == PackageManager.PERMISSION_GRANTED) {
            createCameraSource(autoFocus, useFlash);
        } else {
            requestCameraPermission();
        }

        gestureDetector = new GestureDetector(this, new
CaptureGestureListener());
        scaleGestureDetector = new ScaleGestureDetector(this, new
ScaleListener());

        Snackbar.make(mGraphicOverlay, "Tap to Speak. Pinch/Stretch to zoom",
                Snackbar.LENGTH_LONG)
                .show();


        tts = new CTextToSpeech(this.getApplicationContext(), langCode);
    }

    /**
     * Handles the requesting of the camera permission.  This includes
     * showing a "Snackbar" message of why the permission is needed then
     * sending the request.
     */
    private void requestCameraPermission() {
        Log.w(TAG, "Camera permission is not granted. Requesting
permission");

        final String[] permissions = new
String[]{Manifest.permission.CAMERA};

        if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
                Manifest.permission.CAMERA)) {
            ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
            return;
        }

        final Activity thisActivity = this;

        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ActivityCompat.requestPermissions(thisActivity, permissions,
                        RC_HANDLE_CAMERA_PERM);
            }
        };

        Snackbar.make(mGraphicOverlay, R.string.permission_camera_rationale,
                Snackbar.LENGTH_INDEFINITE)
                .setAction(R.string.ok, listener)
                .show();
    }

    @Override
    public boolean onTouchEvent(MotionEvent e) {
        boolean b = scaleGestureDetector.onTouchEvent(e);

        boolean c = gestureDetector.onTouchEvent(e);

        return b || c || super.onTouchEvent(e);
    }

    /**
     * Creates and starts the camera.  Note that this uses a higher
resolution in comparison
     * to other detection examples to enable the ocr detector to detect small
text samples
     * at long distances.
     *
     * Suppressing InlinedApi since there is a check that the minimum version
is met before using
     * the constant.
     */
    @SuppressLint("InlinedApi")
    private void createCameraSource(boolean autoFocus, boolean useFlash) {
        Context context = getApplicationContext();

        // A text recognizer is created to find text.  An associated
multi-processor instance
        // is set to receive the text recognition results, track the text,
and maintain
```

```
        // graphics for each text block on screen.  The factory is used by
the multi-processor to
        // create a separate tracker instance for each text block.
        TextRecognizer textRecognizer = new
TextRecognizer.Builder(context).build();
        mDetectorProcessor = new OcrDetectorProcessor(mGraphicOverlay);
        textRecognizer.setProcessor(mDetectorProcessor);

        if (!textRecognizer.isOperational()) {
            // Note: The first time that an app using a Vision API is
installed on a
            // device, GMS will download a native libraries to the device in
order to do detection.
            // Usually this completes before the app is run for the first
time.  But if that
            // download has not yet completed, then the above call will not
detect any text,
            // barcodes, or faces.
            //
            // isOperational() can be used to check if the required native
libraries are currently
            // available.  The detectors will automatically become
operational once the library
            // downloads complete on device.
            Log.w(TAG, "Detector dependencies are not yet available.");

            // Check for low storage.  If there is low storage, the native
library will not be
            // downloaded, so detection will not become operational.
            IntentFilter lowstorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
            boolean hasLowStorage = registerReceiver(null, lowstorageFilter)
!= null;

            if (hasLowStorage) {
                Toast.makeText(this, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
                Log.w(TAG, getString(R.string.low_storage_error));
            }
        }

        // Creates and starts the camera.  Note that this uses a higher
resolution in comparison
        // to other detection examples to enable the text recognizer to
detect small pieces of text.
        mCameraSource =
                new CameraSource.Builder(getApplicationContext(),
textRecognizer)
                .setFacing(CameraSource.CAMERA_FACING_BACK)
                .setRequestedPreviewSize(1280, 1024)
                .setRequestedFps(10.0f)
                .setFlashMode(useFlash ? Camera.Parameters.FLASH_MODE_TORCH :
null)
                .setFocusMode(autoFocus ?
Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE : null)
                .build();
    }

    /**
     * Restarts the camera.
     */
    @Override
    protected void onResume() {
        super.onResume();
        startCameraSource();
    }

    /**
     * Stops the camera.
     */
    @Override
    protected void onPause() {
        if (mPreview != null) {
            mPreview.stop();
        }
        if(tts != null){
            tts.stop();
        }
        super.onPause();
    }

    /**
     * Releases the resources associated with the camera source, the
associated detectors, and the
     * rest of the processing pipeline.
     */
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mPreview != null) {
            mPreview.release();
        }
//        if(tts!=null) {
//            tts.shutdown();
//        }
    }

    /**
     * Callback for the result from requesting permissions. This method
     * is invoked for every call on {@link #requestPermissions(String[],
int)}.
     * <p>
     * <strong>Note:</strong> It is possible that the permissions request
interaction
     * with the user is interrupted. In this case you will receive empty
permissions
     * and results arrays which should be treated as a cancellation.
     * </p>
     *
     * @param requestCode  The request code passed in {@link
#requestPermissions(String[], int)}.
     * @param permissions  The requested permissions. Never null.
     * @param grantResults The grant results for the corresponding
permissions
     *                     which is either {@link
PackageManager#PERMISSION_GRANTED}
     *                     or {@link PackageManager#PERMISSION_DENIED}. Never
null.
     * @see #requestPermissions(String[], int)
     */
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                    @NonNull String[] permissions,
                                    @NonNull int[] grantResults) {
        if (requestCode != RC_HANDLE_CAMERA_PERM) {
            Log.e(TAG, LogUtil.prependCallLocation("Got unexpected permission
result: " + requestCode));
            super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
            return;
        }

        if (grantResults.length != 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            Log.e(TAG, LogUtil.prependCallLocation("Camera permission granted
- initialize the camera source"));
            // we have permission, so create the camerasource

            createCameraSource(autoFocus, useFlash);
            return;
        }

        Log.e(TAG, LogUtil.prependCallLocation("Permission not granted:
results len = " + grantResults.length +
                " Result code = " + (grantResults.length > 0 ?
grantResults[0] : "(empty)")));

        DialogInterface.OnClickListener listener = new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                finish();
            }
        };

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Realtime OCR")
                .setMessage(R.string.no_camera_permission)
                .setPositiveButton(R.string.ok, listener)
                .show();
    }

    /**
     * Starts or restarts the camera source, if it exists.  If the camera
source doesn't exist yet
     * (e.g., because onResume was called before the camera source was
created), this will be called
     * again when the camera source is created.
     */
    private void startCameraSource() throws SecurityException {
        // check that the device has play services available.
        int code =
GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
                getApplicationContext());
        if (code != ConnectionResult.SUCCESS) {
            Dialog dlg =
                    GoogleApiAvailability.getInstance().getErrorDialog(this,
code, RC_HANDLE_GMS);
            dlg.show();
        }

        if (mCameraSource != null) {
            try {
                mPreview.start(mCameraSource, mGraphicOverlay);
            } catch (IOException e) {
                Log.e(TAG, LogUtil.prependCallLocation("Unable to start
camera source."), e);
                mCameraSource.release();
                mCameraSource = null;
            }
        }
    }

    private class CaptureGestureListener extends
GestureDetector.SimpleOnGestureListener {

        @Override
        public boolean onSingleTapConfirmed(MotionEvent e) {
            if(mDetectorProcessor!=null) {
                String text =
TextBlockUtil.getString(mDetectorProcessor.textBlockSparseArray,
multiColumn);
                Log.e(TAG, LogUtil.prependCallLocation("onSingleTapConfirmed:
"+text));
                tts.speak(text);
            }
            return true;
        }

        @Override
        public void onLongPress(MotionEvent e) {
            if(mDetectorProcessor!=null) {
                String printText = null;
                if(multiColumn) {
                    printText =
TextBlockUtil.getString(mDetectorProcessor.textBlockSparseArray, false);
                }
                String textToSpeak =
TextBlockUtil.getString(mDetectorProcessor.textBlockSparseArray,
multiColumn);
                Log.e(TAG, LogUtil.prependCallLocation("onLongPress:
"+textToSpeak ));
                Intent intent = new Intent(getApplicationContext(),
TextManagerActivity.class);
                intent.putExtra(TextManagerActivity.tagPrintText, printText);
                intent.putExtra(TextManagerActivity.tagSpeakText,
textToSpeak);
                intent.putExtra(TextManagerActivity.tagLangCode, langCode);
                finish();
                startActivity(intent);

            }
        }
    }

    private class ScaleListener implements
ScaleGestureDetector.OnScaleGestureListener {

        /**
         * Responds to scaling events for a gesture in progress.
         * Reported by pointer motion.
         *
         * @param detector The detector reporting the event - use this to
         *                 retrieve extended info about event state.
         * @return Whether or not the detector should consider this event
         * as handled. If an event was not handled, the detector
         * will continue to accumulate movement until an event is
         * handled. This can be useful if an application, for example,
         * only wants to update scaling factors if the change is
         * greater than 0.01.
         */
        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            return false;
        }

        /**
         * Responds to the beginning of a scaling gesture. Reported by
         * new pointers going down.
         *
         * @param detector The detector reporting the event - use this to
         *                 retrieve extended info about event state.
         * @return Whether or not the detector should continue recognizing
         * this gesture. For example, if a gesture is beginning
         * with a focal point outside of a region where it makes
         * sense, onScaleBegin() may return false to ignore the
         * rest of the gesture.
         */
        @Override
        public boolean onScaleBegin(ScaleGestureDetector detector) {
            return true;
        }

        /**
         * Responds to the end of a scale gesture. Reported by existing
         * pointers going up.
         * <p/>
         * Once a scale has ended, {@link ScaleGestureDetector#getFocusX()}
         * and {@link ScaleGestureDetector#getFocusY()} will return focal
point
         * of the pointers remaining on the screen.
         *
         * @param detector The detector reporting the event - use this to
         *                 retrieve extended info about event state.
         */
        @Override
        public void onScaleEnd(ScaleGestureDetector detector) {
            if (mCameraSource != null) {
                mCameraSource.doZoom(detector.getScaleFactor());
            }
        }
    }
}
```

Currency Detector

```
import com.example.controller.app.CTextToSpeech;
import com.example.controller.app.LogUtil;
import com.example.controller.app.R;
import com.example.controller.app.currency.env.BorderedText;
import com.example.controller.app.currency.env.ImageUtils;
import com.example.controller.app.currency.env.Logger;

import com.example.controller.app.currency.OverlayView.DrawCallback;
import com.example.controller.app.R;
import com.example.controller.app.currency.env.BorderedText;
import com.example.controller.app.currency.env.ImageUtils;
import com.example.controller.app.currency.env.Logger;

import java.util.Arrays;
import java.util.List;
import java.util.Vector;

public class ClassifierActivity extends CameraActivity implements
OnImageAvailableListener {
    private static final Logger LOGGER = new Logger();

    // These are the settings for the original v1 Inception model. If you
want to
    // use a model that's been produced from the TensorFlow for Poets
codelab,
    // you'll need to set IMAGE_SIZE = 299, IMAGE_MEAN = 128, IMAGE_STD =
128,
    // INPUT_NAME = "Mul", and OUTPUT_NAME = "final_result".
    // You'll also need to update the MODEL_FILE and LABEL_FILE paths to
point to
    // the ones you produced.
    //
    // To use v3 Inception model, strip the DecodeJpeg Op from your retrained
    // model first:
    //
    // python strip_unused.py \
    // --input_graph=<retrained-pb-file> \
    // --output_graph=<your-stripped-pb-file> \
    // --input_node_names="Mul" \
    // --output_node_names="final_result" \
    // --input_binary=true

  /* Inception V3
  private static final int INPUT_SIZE = 299;
  private static final int IMAGE_MEAN = 128;
  private static final float IMAGE_STD = 128.0f;
  private static final String INPUT_NAME = "Mul:0";
  private static final String OUTPUT_NAME = "final_result";
  */

    private static final int INPUT_SIZE = 224;
    private static final int IMAGE_MEAN = 128;
    private static final float IMAGE_STD = 128.0f;
    private static final String INPUT_NAME = "input";
    private static final String OUTPUT_NAME = "final_result";

    private static final String MODEL_FILE =
"file:///android_asset/graph.pb";
    private static final String LABEL_FILE =
"file:///android_asset/labels.txt";

    private static final boolean SAVE_PREVIEW_BITMAP = false;

    private static final boolean MAINTAIN_ASPECT = true;

    private static final Size DESIRED_PREVIEW_SIZE = new Size(640, 480);

    private Classifier classifier;

    private Integer sensorOrientation;
    private int previewWidth = 0;
    private int previewHeight = 0;
    private byte[][] yuvBytes;
    private int[] rgbBytes = null;
    private Bitmap rgbFrameBitmap = null;
    private Bitmap croppedBitmap = null;

//    private Bitmap cropCopyBitmap;

    private boolean computing = false;

    private Matrix frameToCropTransform;
    private Matrix cropToFrameTransform;

//  private ResultsView resultsView;

    private BorderedText borderedText;

//    private long lastProcessingTimeMs;
    private String currResult;

    @Override
    protected int getLayoutId() {
        return R.layout.camera_connection_fragment;
    }

    @Override
    protected Size getDesiredPreviewFrameSize() {
        return DESIRED_PREVIEW_SIZE;
    }

    private static final float TEXT_SIZE_DIP = 10;

    @Override
    public void onPreviewSizeChosen(final Size size, final int rotation) {
        final float textSizePx =
                TypedValue.applyDimension(
                        TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP,
getResources().getDisplayMetrics());
        borderedText = new BorderedText(textSizePx);
        borderedText.setTypeface(Typeface.MONOSPACE);

        classifier =
                TensorFlowImageClassifier.create(
                        getAssets(),
                        MODEL_FILE,
                        LABEL_FILE,
                        INPUT_SIZE,
                        IMAGE_MEAN,
                        IMAGE_STD,
                        INPUT_NAME,
                        OUTPUT_NAME);

//    resultsView = (ResultsView) findViewById(R.id.results);
        previewWidth = size.getWidth();
        previewHeight = size.getHeight();

        final Display display = getWindowManager().getDefaultDisplay();
        final int screenOrientation = display.getRotation();
        LOGGER.i("Sensor orientation: %d, Screen orientation: %d", rotation, 
screenOrientation);

        sensorOrientation = rotation + screenOrientation;

        LOGGER.i("Initializing at size %dx%d", previewWidth, previewHeight);
        rgbBytes = new int[previewWidth * previewHeight];
        rgbFrameBitmap = Bitmap.createBitmap(previewWidth, previewHeight, 
Config.ARGB_8888);
        croppedBitmap = Bitmap.createBitmap(INPUT_SIZE, INPUT_SIZE, 
Config.ARGB_8888);

        frameToCropTransform =
                ImageUtils.getTransformationMatrix(
                        previewWidth, previewHeight,
                        INPUT_SIZE, INPUT_SIZE,
                        sensorOrientation, MAINTAIN_ASPECT);

        cropToFrameTransform = new Matrix();
        frameToCropTransform.invert(cropToFrameTransform);

        yuvBytes = new byte[3][];

        /*addCallback(
                new OverlayView.DrawCallback() {
                    @Override
                    public void drawCallback(final Canvas canvas) {
                        renderDebug(canvas);
                    }
                });*/
    }

    @Override
    public void onImageAvailable(final ImageReader reader) {
        Image image = null;

        try {
            image = reader.acquireLatestImage();

            if (image == null) {
                return;
            }

            if (computing) {
                image.close();
                return;
            }
            computing = true;

            Trace.beginSection("imageAvailable");

            final Plane[] planes = image.getPlanes();
            fillBytes(planes, yuvBytes);

            final int yRowStride = planes[0].getRowStride();
            final int uvRowStride = planes[1].getRowStride();
            final int uvPixelStride = planes[1].getPixelStride();
            ImageUtils.convertYUV420ToARGB8888(
                    yuvBytes[0],
                    yuvBytes[1],
                    yuvBytes[2],
                    previewWidth,
                    previewHeight,
                    yRowStride,
                    uvRowStride,
                    uvPixelStride,
                    rgbBytes);

            image.close();
        } catch (final Exception e) {
            if (image != null) {
                image.close();
            }
            LOGGER.e(e, "Exception!");
            Trace.endSection();
            return;
        }

        rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0,
previewWidth, previewHeight);
        final Canvas canvas = new Canvas(croppedBitmap);
        canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);

        // For examining the actual TF input.
        if (SAVE_PREVIEW_BITMAP) {
            ImageUtils.saveBitmap(croppedBitmap);
        }

        runInBackground(
                new Runnable() {
                    @Override
                    public void run() {
//                          final long startTime = SystemClock.uptimeMillis();
                        final List<Classifier.Recognition> results =
classifier.recognizeImage(croppedBitmap);
//                          lastProcessingTimeMs = SystemClock.uptimeMillis() -
startTime;
                        Log.e("TAG", LogUtil.prependCallLocation("run: " +
results));
                        if(results.size()!=0) {
                            currResult = results.get(0).getTitle()+" Rupees";

                        } else {
                            currResult = "No Note Found";
                        }
                        /*cropCopyBitmap =
Bitmap.createBitmap(croppedBitmap);
                        resultsView.setResults(results);
                        requestRender();*/
                        computing = false;
                    }
                });
        Trace.endSection();
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Log.e("TAG", LogUtil.prependCallLocation("onTouchEvent: " +
event.toString()));
        if (event.getAction() == MotionEvent.ACTION_UP) {
            Log.e("TAG", LogUtil.prependCallLocation("onTouchEvent:
"+currResult ));
```

```
            tts.speak(currResult);
        }
        return super.onTouchEvent(event);
    }
/*@Override
  public void onSetDebug(boolean debug) {
    classifier.enableStatLogging(debug);
  }

  private void renderDebug(final Canvas canvas) {
    if (!isDebug()) {
      return;
    }
    final Bitmap copy = cropCopyBitmap;
    if (copy != null) {
      final Matrix matrix = new Matrix();
      final float scaleFactor = 2;
      matrix.postScale(scaleFactor, scaleFactor);
      matrix.postTranslate(
          canvas.getWidth() - copy.getWidth() * scaleFactor,
          canvas.getHeight() - copy.getHeight() * scaleFactor);
      canvas.drawBitmap(copy, matrix, new Paint());

      final Vector<String> lines = new Vector<String>();
      if (classifier != null) {
        String statString = classifier.getStatString();
        String[] statLines = statString.split("\n");
        lines.addAll(Arrays.asList(statLines));
      }

      lines.add("Frame: " + previewWidth + "x" + previewHeight);
      lines.add("Crop: " + copy.getWidth() + "x" + copy.getHeight());
      lines.add("View: " + canvas.getWidth() + "x" + canvas.getHeight());
      lines.add("Rotation: " + sensorOrientation);
      lines.add("Inference time: " + lastProcessingTimeMs + "ms");

      borderedText.drawLines(canvas, 10, canvas.getHeight() - 10, lines);
    }
  }*/
}
```

4.4 MODEL TRAINING

Tensor flow Model

Object detection works in two ways:

The first division permits networks to isolate the tasks of locating objects and classifying them (Faster R-CNN); The second division allows networks to predict class scores and bounding boxes (YOLO and SSD networks). Nevertheless, most object detection tutorials are done by drawing bounding boxes around the input image to define the objects and their locations.

However, object detection is different from image identification, although both usually go hand-in-hand. The difference lies in how models detect the objects. In image identification, detectors see and label an entire image, while only objects within an image are detected in object detection. More like zooming into the pexels of the image.

The closest example of object TF detection is the Google Lens, an image and object recognition program. Google Lens works by identifying objects for curious users. All users need to begin is to take a picture of the thing. Google Lens then identifies the real-world object and fetches the required information from the internet.

The algorithms deployed for object detection work in two ways: trained before use or used without being trained (unsupervised). In essence, it is either you're using

- the Machine Learning approach to detect objects or
- the Deep Learning approach.
- But bear in mind that:

The Machine Learning approach scans the surface features of an image to detect objects. Some of these features include the histogram or edges of the picture. After that, ML runs a regression test to predict the thing and identify its location.

In the Deep Learning approach, it's a different ball game. Deep Learning uses convolutional neural networks to identify objects and their location. Since these networks are unsupervised, the system identifies objects on their own as well as their areas.

TensorFlow is an open-source framework used in several algorithmic functions such as image and object detection, voice search, and several others. TensorFlow boasts the application of Python and C++ APIs, as the framework hosts both Deep and Machine Learning algorithms.

TensorFlow is perfect for object detection because developers do not worry about the approach they should use to build an object detection model since both ML and DL algorithms are in TensorFlow. But before you plunge into training an object detection model with TensorFlow, here are details you should know.

- TensorFlow uses Python as a front-end language and runs on C++ — both languages are popular in object detection.
- TensorFlow uses consultation graphs, where nodes are mathematical operations and data is represented through the connections.
- TensorFlow object detection API prevents developers from building a model from scratch (training a model) by providing a set of existing operations.
- TensorFlow APIs are built on the "Model Zoo" which are already-trained models in the framework.
- The datasets deployed in training object detection models in TensorFlow include the KITTI dataset, COCO dataset, and Dataset of open Images.

5. TESTING

This chapter consists of the test plan adopted with respect to the project. All the testing approaches which were followed are listed in this section. The chapter also consists of the various test cases identified along with their verification and description. The corresponding test reports are specified to confirm the working of test cases.

5.1 TESTING APPROACHES

5.1.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. In unit testing we examine the local data structures and interface within individual module. We check for errors within its boundary and make sure its functionality is fulfilled. Ulrimatetric is an Android application which consists of many modules. It is important that each of these modules are tested individually to make sure all units are working as per requirement.

In the android application, individual activity pages are tested to make sure their activity is as per the specified requirements. Individual modules were tested to minimize errors. Most importantly the Models were tested through each unit for optimization and identification of object or character . The system was thoroughly tested at every phase to minimize future complexities.

5.1.2 Integrated Testing

Integration testing is the phase in software testing in which individual software units and modules are combined and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

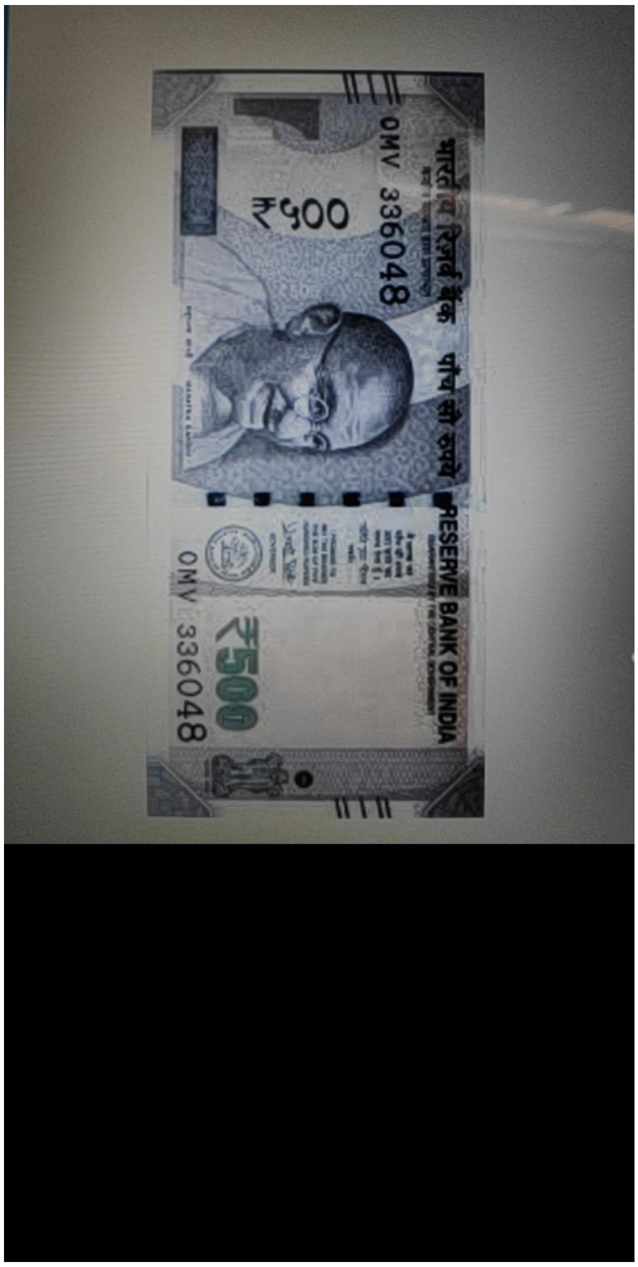
Integration testing is the phase in software testing in which individual software units and modules are combined and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

5.2 Test Cases

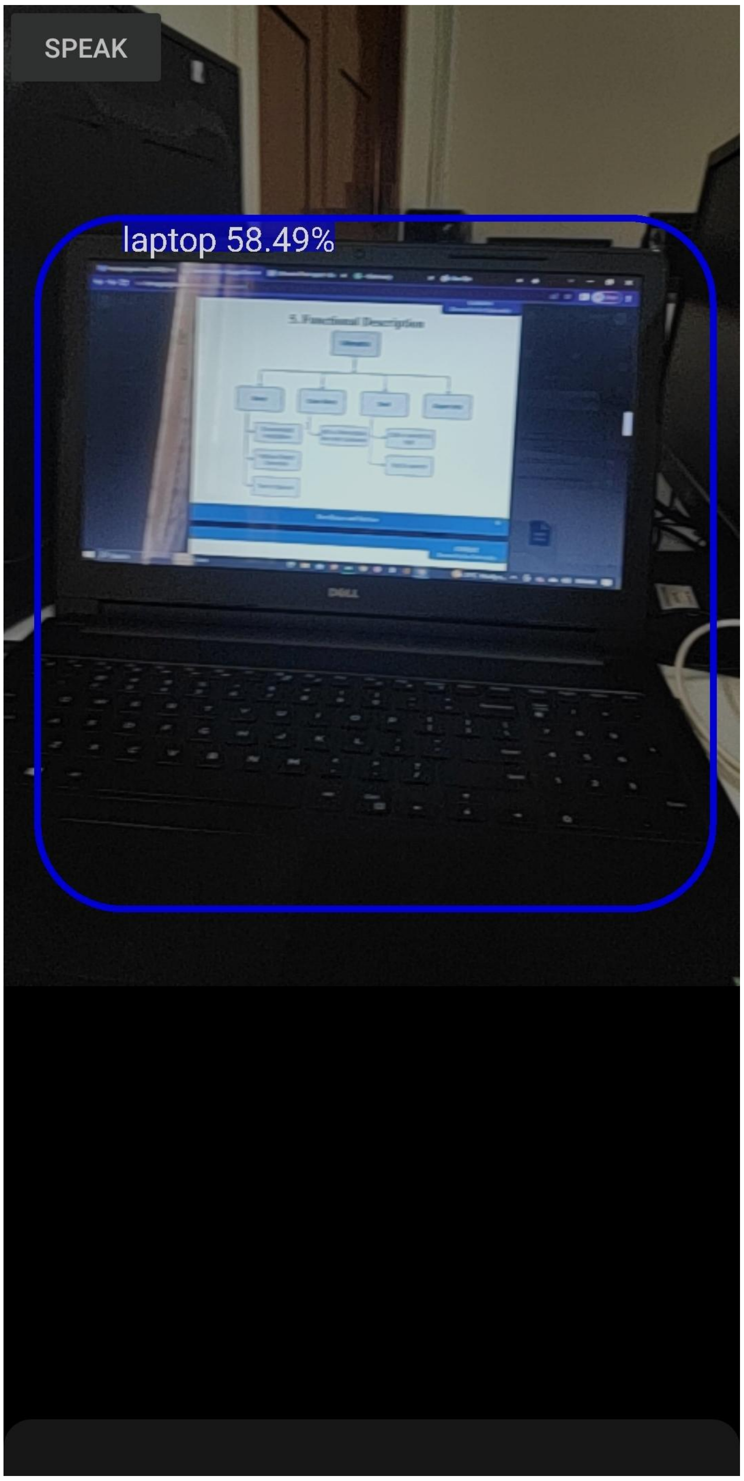
Test ID	Test Name	Description	Procedure	Expected Output	Result
1.1	Object Detection	To check if the app abel to detect the objects and notify the person	Open the App and choose Environment desc	Able to detect the object and notify the person in audio format	Pass
1.2	Text detection OCR	Able to read the text written and read out to the user	Open the app choose OCR	Detect the text from the image and read out to the user	Pass
1.3	Currency detection	detect some of the indian currency and notify the user	Open the app and choose currency	Detect the currency and tell the user in aaudio format	Pass
1.4	Text to speech	Able to convert the text to speeches	Open the app and choose text to speech	Convert the text written in the images and speak out to the user	Pass
1.5	Color detection	detect the color in the environment and tell the user the colour they are looking at.	Open the app and choose Color detect	Detect the color and tells the user	Fail

5.3 TEST REPORTS

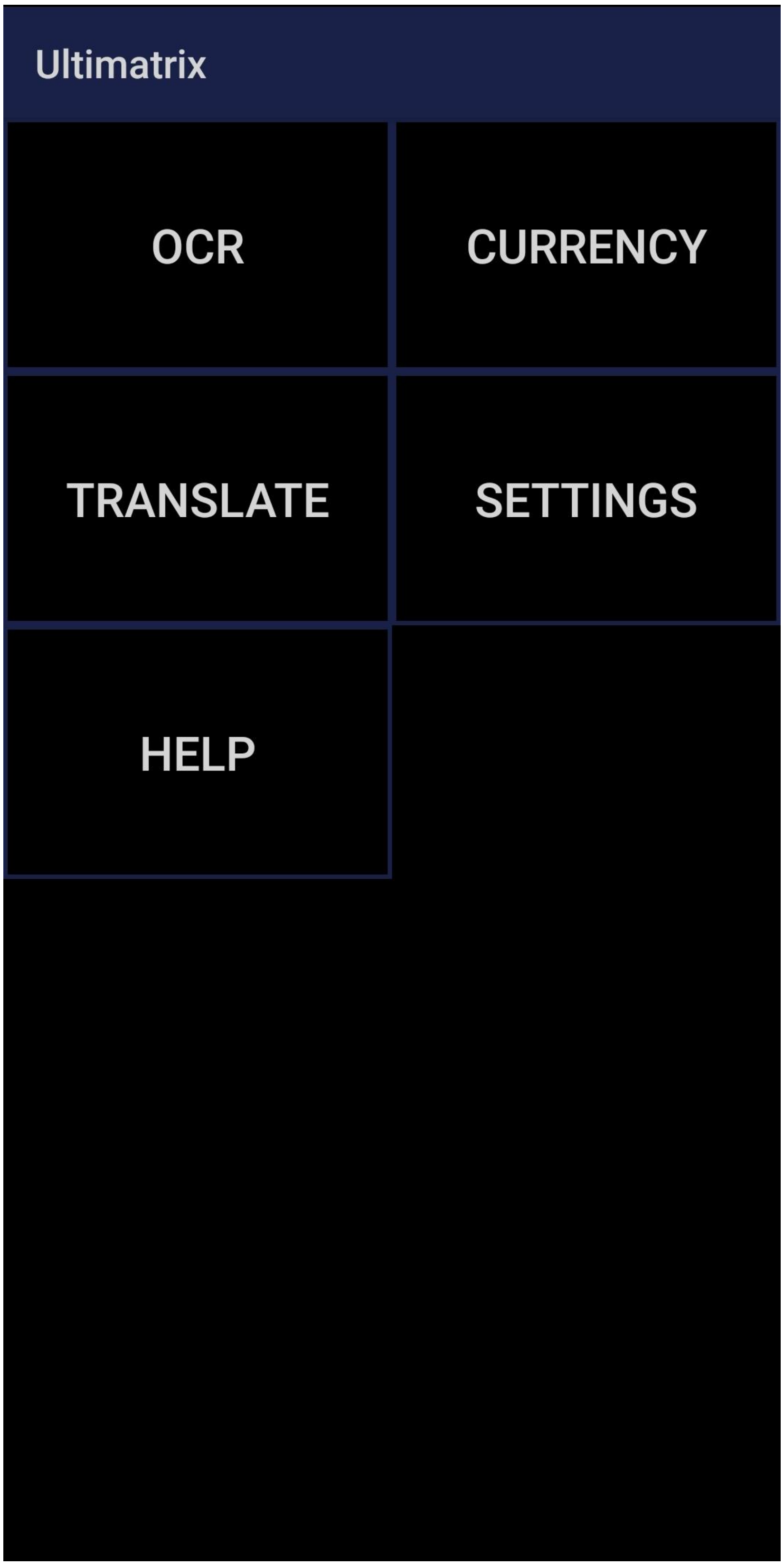
Currency Detection



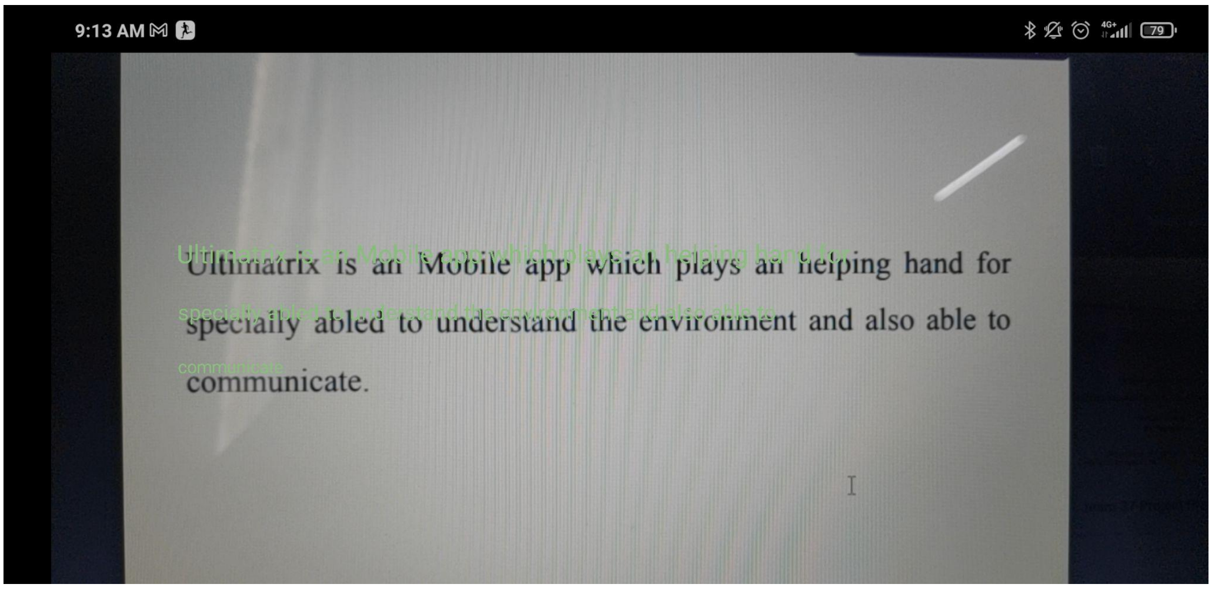
Object Detection



Main Menu



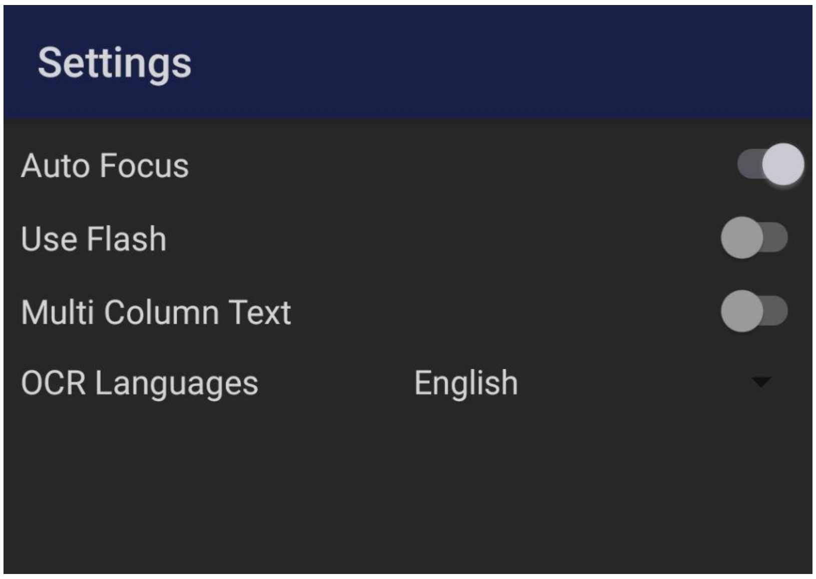
OCR



9:13 AM

Ultimatrix is an Mobile app which plays an helping hand for specially abled to understand the environment and also able to communicate.

Settings



Settings

Auto Focus

Use Flash

Multi Column Text

OCR Languages English

6 Conclusion

6.1. DESIGN AND IMPLEMENTATION ISSUES

There were issues with respect to model integration in the application while the implementation phase of the project after the model was trained. Linking the model with the api was the biggest challenge encountered in the project development process due to which many changes were made in the code. Inconsistency of data type was an area of focus for an efficient model.

6.2 Advantages:

The proposed system successfully detects 90 objects,labels them and also shows its accuracy.The model also calculates the distance from the object to the camera and gives a voice feedback as when the person with the camera is approaching the object.The dataset was tested on two different models, SSD Mobilenet V1 and SSD Inception V2.However the SSD Mobilenet V1 model showed less latency and was faster in detecting objects.

6.4 Limitations:

Even though we focused on making the app feature rich it still lacks an user friendly way to communicate with the Visually Impaired people.

6.5 Future Enhancements

- Environment description
- AR filter for Colour Blind
- Real time Call to text
- Maintain and record the Logs

7. References

- [1] World Health Organization, “Visual Impairment and Blindness,” WHO Factsheet no. FS282 , Dec. 2014.
- [2] Mingmin Zhao, FadelAdib, Dina Katabi Emotion Recognition using wireless signals.
- [3] N. Senthil kumar, A. Abinaya, E. Arthi, M. Atchaya, M. Elakkiya, “SMART EYE FOR VISUALLY IMPAIRED PEOPLE”, International Research Journal of Engineering and Technology, Volume: 07 Issue: 06, June 2020.
- [4] Liang – Bi Chen, Ming-Che Chen, “An implementation of an intelligent assistace system for visually impaired/blind people, ”IEEE, 2018.
- [5] Shagufta Md.Rafique Bagwan, Prof. L.J.Sankpal,” VisualPal: A Mobile App forOject Recognition for the Visually Impaired”, IEEE International Conference on Computer, Communication and Control (IC4-2015).
- [6] Shahed Anzarus Sabab, Md. Hamjajul Ashmafee, “Blind Reader: An IntelligentAssistant for Blind”, 19th International Conference on Computer and Information Technology, December 18-20, 2016, North South University, Dhaka, Bangladesh
- [7] Shreyash Patil, Oshin Gawande, Shivam Kumar, Pradip Shewale,“Assistant Systems for the Visually Impaired”, International Research Journal of Engineering and Technology (IRJET), Volume: 07 Issue: 01 | Jan 2020.
- [8] Gagandeep Singh, Omkar Kandale, Kevin Takhtani, Nandini Dadhwal, “A Smart Personal AI Assistant for Visually Impaired People”, International Research Journal of Engineering and Technology (IRJET), Volume: 07 Issue: 06 | June 2020.