INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

Implementation report

Department of Computing

5COSC021C.2 Software Development Group Project

Team Beta Tech – SE-58

Modu

raliya

Mentor : Mr. Torin Wirasingha

“Object identifier to hearing and verbally impaired students”

GROUP MEMBERS:

IIT NUMBER	UoW NUMBER	NAME
20200765	18335118/1	N.V.V De Silva
20200612	18360488/1	P. S. B. Karunarathna
20200672	18325607/1	H.K.G.J. Jayasekara
2019404	18097687	S.R.S Fernando
2018348	17618881/1	K.G.S. Tharanga

Declaration page

We hereby certify that this project report and all the artifacts associated with it is our own work and it has not been submitted before nor is currently being submitted for any degree program.

Full Name of Student: Nammuni Vathila Vilhan De Silva

Registration Number: 20200765

UoW Number: w1833511

Full Name of Student: Pramuditha Shelum Bandara Karunarathna

Registration Number: 20200612

UoW Number: w18360488

Full Name of Student: Halloluwa Kankanamge Gayana Jayanada Jayasekara

Registration Number: 20200672

UoW Number: w18325607

Full Name of Student: Siyambalapitiya Ravindu Sathyajith Fernando

Registration Number: 2019404

UoW Number: w18097687

Full Name of Student: Kevitiyagalage Sumudu Tharanga

Registration Number: 2018348

UoW Number: w17618881

Date: 30th April 2022

Abstract

The writers of this paper devised a strategy to alleviate the lack of challenges people have when talking with the verbally impaired after doing significant research. Furthermore, this solution was created as a mobile-based application, and this report will cover the application's implementation, testing, deployment, and evaluation, with a focus on the development and technologies utilized throughout the project.

Keywords – TensorFlow, AWS, GitHub

Acknowledgement

First and foremost, we want to thank God for providing us with the strength and courage to face all the challenges and finish this report.

Second, we want to thank our parents and family for always trusting us and aiding us in attaining high standards by sharing their expertise and motivating us to achieve better.

We'd also want to thank our mentor, Mr. Torin Wirasingha, for his consistent encouragement, guidance, and monitoring for this report. We are grateful to him for leading us in the correct route and encouraging us when we were going through difficult times.

A special thanks to our mentor who was assigned to us for our Haxmas finals, Mr. Suresh, for his guidance in helping us finish our application.

Finally, I'd want to thank all my colleagues and friends who have always been eager to provide a helping hand whenever we needed it.

Table of Contents

List of figures

List of tables

Abbreviations table

Abbreviation	Explanation
CNN	Convolutional Neural Networks
R-CNN	Region-Convolutional Neural Network
NLP	Natural Language Processing
HCI	Human Computer Interaction
UX	User Experience
UI	User Interface
AWS	Amazon Web Services

Chapter 1 Implementation

1.1 Chapter Overview

This chapter will focus on the prototype's implementation, including a full discussion of the programming languages, technologies, frameworks, and libraries that were utilized. The features of the ObSign prototype are discussed in detail, including how each feature was implemented, the intended application of each feature, essential code snippets and screenshots, and the obstacles and solutions were discovered while implementing each feature.

1.2 Overview of the prototype

- The Frontend of the application has been developed using Flutter dart, where it will show the sign language of the detected object.
- The backend will process once an object has been scanned where the algorithm will process the scanned object and produce its sign language.

1.3 Technology selections

The technology choices made during the development of ObSign are given here, along with the language, libraries, and frameworks utilized.

1.3.1 Frontend Technology Selection

The project's UI/UX was constructed fully in Flutter dart, as planned. However, there were several reasons why Flutter claimed to be better to its next best choice, React Native. These reasons are as follows.

1. Many of the native components in the Flutter environment are contained within the framework, therefore communication with them does not necessarily necessitate the use of a bridge. React Native, on the other hand, employs JavaScript as a bridge to connect with its native modules, resulting in poor performance.

2. Flutter's code platform is built on Dart, a programming language that can be used to create custom UI designs for the app. Allows for the creation of many layouts and widgets that may be recycled.

3. React Native is overly reliant on third-party libraries, whereas Flutter offers a large collection of UI components.

4. Simple phone access points, such as accessing the smartphone camera, are not incompatible with Flutter, which is necessary for the key activities of the developed app.

1.3.2 Backend Technology Selection

Python was described as the primary programming language for the development of this project after an evaluation of the available programming languages. Python was chosen primarily for the following reasons,

1. Free and open source
2. Supported by many libraries
3. Interpreted languages
4. Object-oriented language
5. Ideal for prototypes since it allows you to add additional features with minimal code.

AWS Lambda was used to use to run the model in the ec2 instance where a lambda function was implemented in such a form that it executes the code in the ec2 instance remotely.

1.3.2.1 TensorFlow

It is an open-source platform that is accessible to all users and capable of supporting the creation of any system. With its graphical approach, TensorFlow gives a better method of seeing data. With the aid of Tensor Board, it is also possible to debug nodes quickly. This saves time by not having to visit every line of code and effectively resolves the neural network. It works with a variety of languages, including C++, JavaScript, Python, C#, Ruby, and Swift. This allows the user to work in a setting that is familiar to them. TensorFlow also features a TPU architecture that is quicker than the GPU and CPU at doing computations. TPU-based models may be simply placed in the cloud for a lower cost and quicker execution(TensorFlow, 2022).

1.3.2.2 Matplotlib

Making use of the matplotlib library's pyplot module. Pyplot is an application programming interface(API) that contains functions and methods for processing and visualizing data. When it comes to creating visual graphs, the plot is lightning fast. Furthermore, its similarities to MATLAB make it simple to work with(Pyplot tutorial — Matplotlib 3.5.0 documentation, 2022).

1.3.2.3 NumPy

Numerical Python extension is referred to as "NumPy". This library includes a number of mathematical routines for working with multi-dimensional arrays and matrices. It can also store data of any type and interface with a wide range of databases.

1.4 Implementation of the data science component

As an overview, this project uses object detection and predict the relevant sign of the detected object. Object detection is a key turning point in computer science. Accordingly, the main data science component will be the object detection model in this project. To fulfil the task of object detection, the project team used a TensorFlow model with an architecture of Faster R-CNN. The model is pre-trained using the OpenImags V4 dataset. OpenImages V4 dataset is a 9 million-image collection that includes image-level metadata and object bounding boxes. In this dataset, there are 14.6M bounding boxes for 600 object classes.

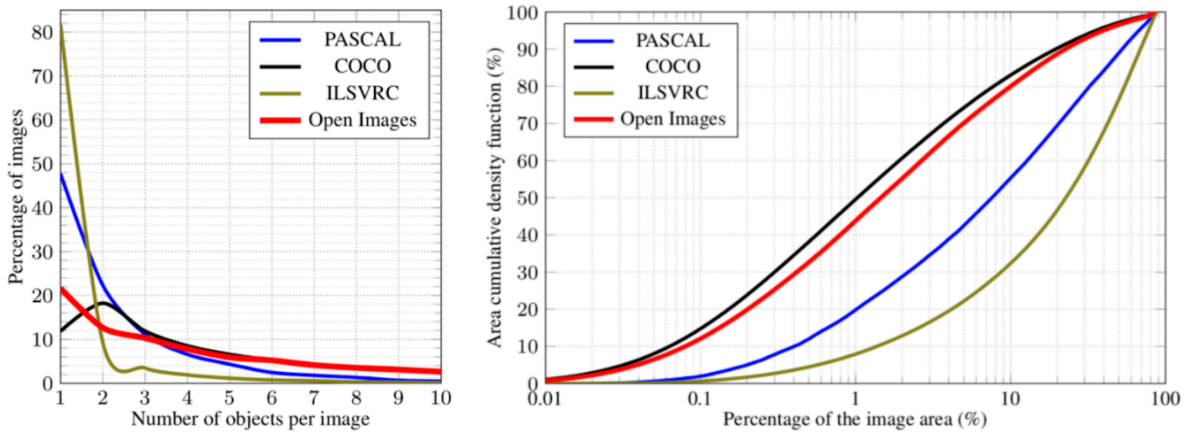
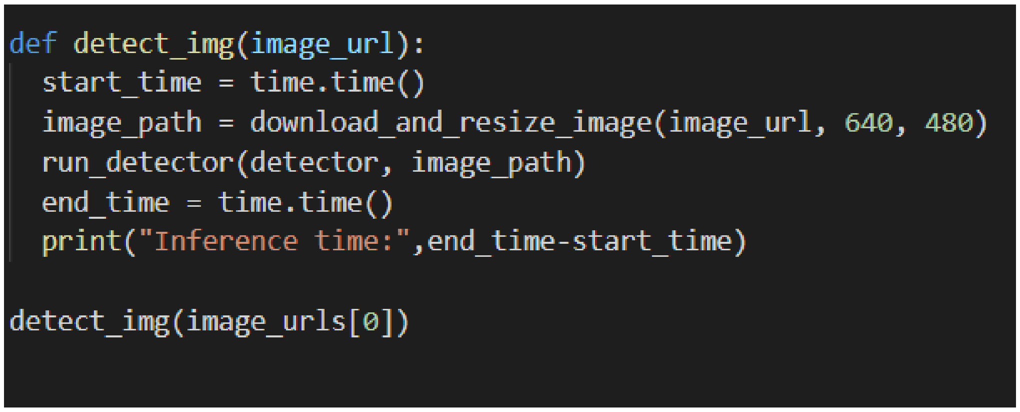


Figure 1 - Comparison between datasets

A number of objects per image (left) and object area (right) for Open Images V6/V5/V4 and other related datasets (training sets in all cases).

The path/URL of the images that should be input into the model will be stored in a list. Then the images will be input based on the index. Next, the input image will be resized. Finally, the name of the object as well as the image with bounding boxes and the processing time will be returned as the output.

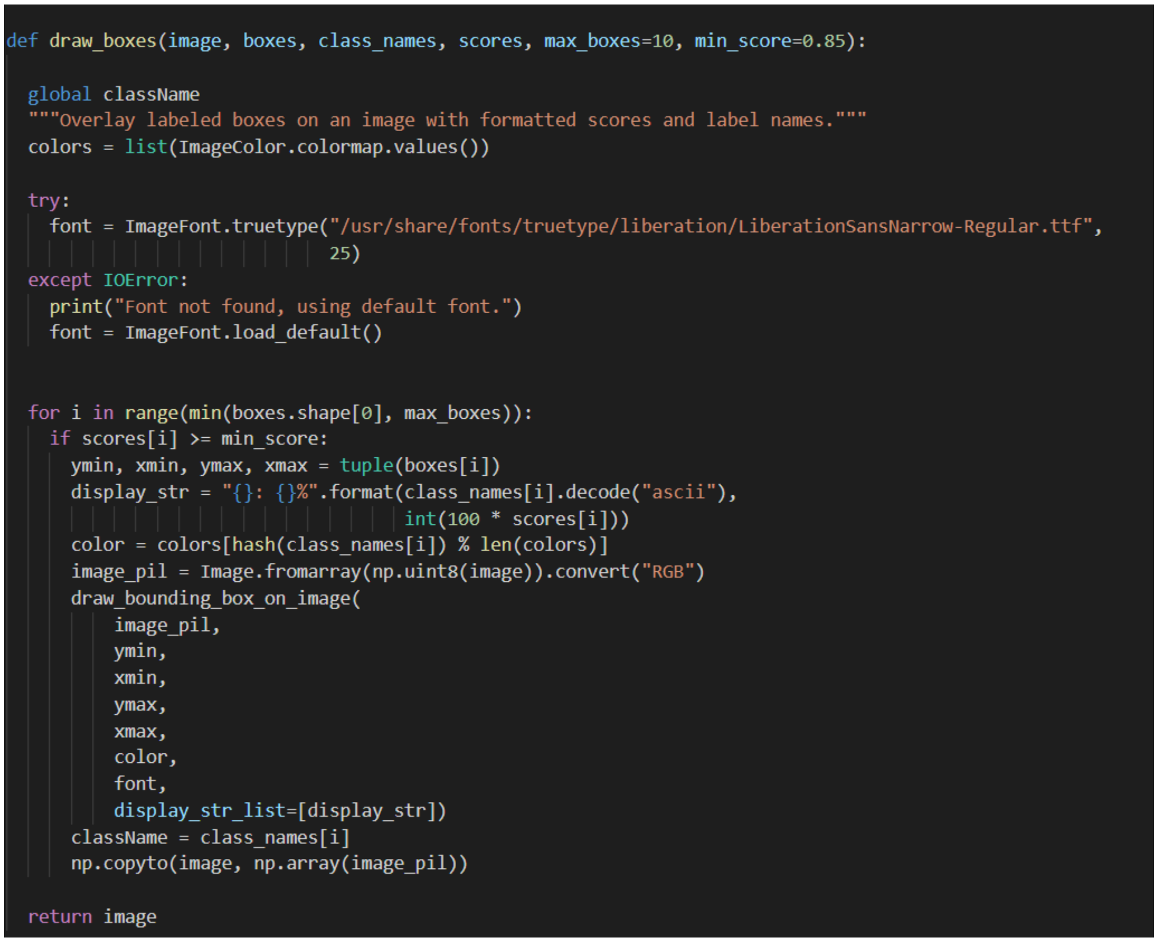


```
def detect_img(image_url):
  start_time = time.time()
  image_path = download_and_resize_image(image_url, 640, 480)
  run_detector(detector, image_path)
  end_time = time.time()
  print("Inference time:",end_time-start_time)

detect_img(image_urls[0])
```

Figure 2 - Code snippet of return class name

To obtain high accuracy and to match the correct object to the query, the min_score is increased. If the min_score is decreased the matching probability of the query and the item will be also lower as a result the model can detect more objects with less accuracy. Meanwhile when the min_score has increased the accuracy of the matching percentage is high, but the number of directed objects is low. So, min_score is inversely proportional to the number of objects detected as well as to the matching percentage of the object and the query. According to the project, accuracy is more important than the number of objects that can be detected. So, the min_score is increased to 0.85. After detecting the object, the class name will assign to a global variable.



```python
def draw_boxes(image, boxes, class_names, scores, max_boxes=10, min_score=0.85):

  global className
  """Overlay labeled boxes on an image with formatted scores and label names."""
  colors = list(ImageColor.colormap.values())

  try:
    font = ImageFont.truetype("/usr/share/fonts/truetype/liberation/LiberationSansNarrow-Regular.ttf",
                              25)
  except IOError:
    print("Font not found, using default font.")
    font = ImageFont.load_default()


  for i in range(min(boxes.shape[0], max_boxes)):
    if scores[i] >= min_score:
      ymin, xmin, ymax, xmax = tuple(boxes[i])
      display_str = "{}: {}%".format(class_names[i].decode("ascii"),
                                     int(100 * scores[i]))
      color = colors[hash(class_names[i]) % len(colors)]
      image_pil = Image.fromarray(np.uint8(image)).convert("RGB")
      draw_bounding_box_on_image(
          image_pil,
          ymin,
          xmin,
          ymax,
          xmax,
          color,
          font,
          display_str_list=[display_str])
      className = class_names[i]
      np.copyto(image, np.array(image_pil))

  return image
```

Figure 3 - Creating bounding boxes

Due to the high processing of the above TensorFlow model an alternative method was used, A TensorFlow lite model was used to for better processing and due to the limited resources used in this project. The TensorFlow model was created using Teachable Machine from google where the developers trained a model for a limited number of objects and then converted the given TensorFlow model to a TensorFlow lite model.

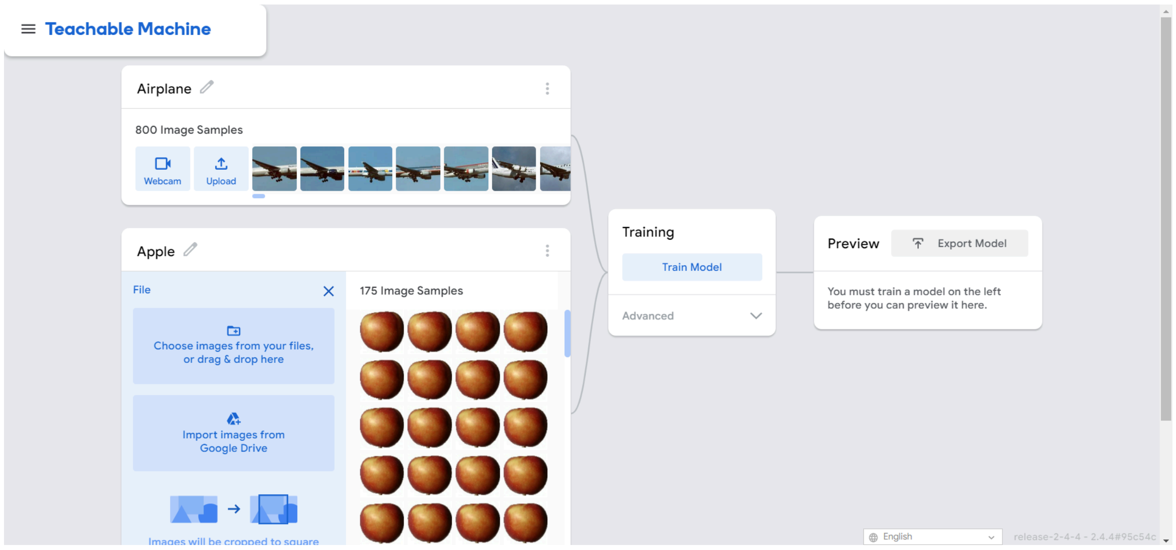
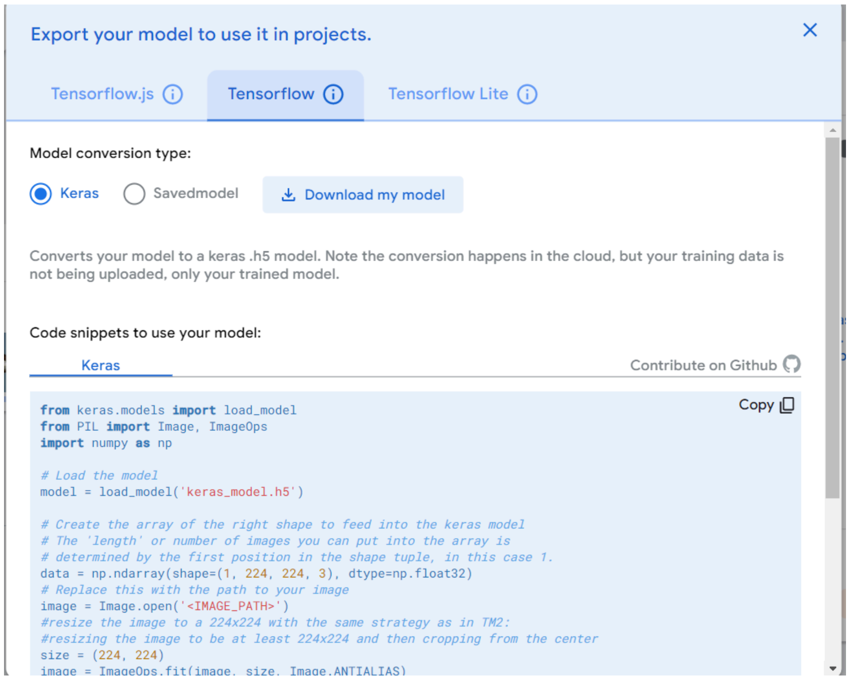

Teachable Machine
Airplane
800 Image Samples
Webcam
Upload
Apple
File
Choose images from your files, or drag & drop here
Import images from Google Drive
Images will be cropped to square
175 Image Samples
Training
Train Model
Advanced
Preview
Export Model
You must train a model on the left before you can preview it here.
English
release-2-4-4 - 2.4.4#95c54c

Figure 4 - Training the TensorFlow model

After the objects are being added to the training model and then trained it gives the necessary TensorFlow model.



Export your model to use it in projects.

Model conversion type:

[✓ Keras] Savedmodel

Converts your model to a keras .h5 model. Note the conversion happens in the cloud, but your training data is not being uploaded, only your trained model.

Code snippets to use your model:

```
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np

# Load the model
model = load_model('keras_model.h5')

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1.
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
# Replace this with the path to your image
image = Image.open('<IMAGE_PATH>')
#resize the image to a 224x224 with the same strategy as in TM2:
#resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.ANTIALIAS)
```

Figure 5 - Exporting model

1.5 Implementation of the backend component

1.5.1 Authentication

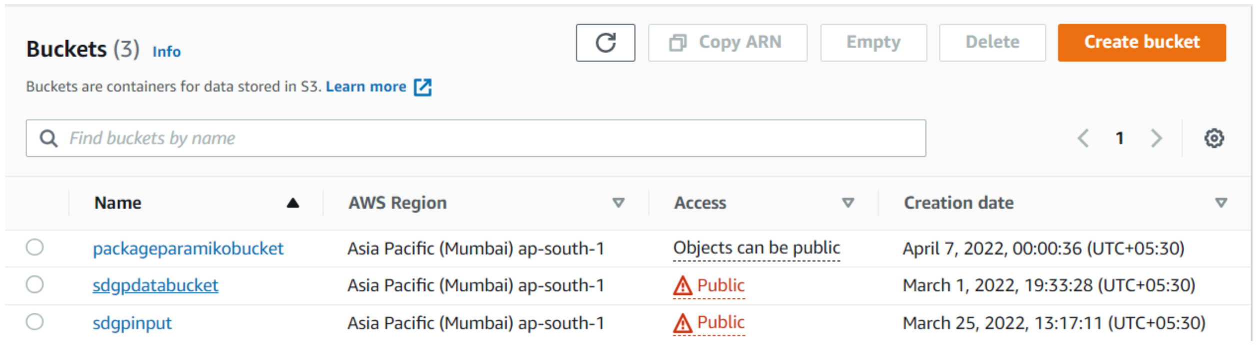
AWS Amplify is a set of tools, services, and frameworks that make it easier to create and serve online and mobile apps on Amazon Web Services. Amplify is compatible with mobile platforms such as iOS, Android, and Flutter.

The Amplify Console is an AWS service that provides a git-based workflow for continuous deployment & hosting of full-stack web apps. The Amplify Console shows cloud resources generated with the Amplify CLI.

This project used Amplify CLI to configure all the services needed to power the backend through a simple command-line interface for creating a proper backend and also used amplify Libraries to integrate the app code with a backend using declarative interfaces.

1.5.2 Database

In the ObSign database a s3 bucket called "'sdgpdatabucket', It include the sign language images that are used are stored in a s3 bucket database with its text of the relevant sign language. Another s3 bucket was created called "sdgpinput " where it holds the images taken by the user and then the image uploaded is used in the ec2 instance which runs the model using the uploaded image. A third bucket "packageparamikobucket" was used to store the paramiko package which is a library used to run the ec2 instance remotely.



Buckets (3) Info

Buckets are containers for data stored in S3. Learn more

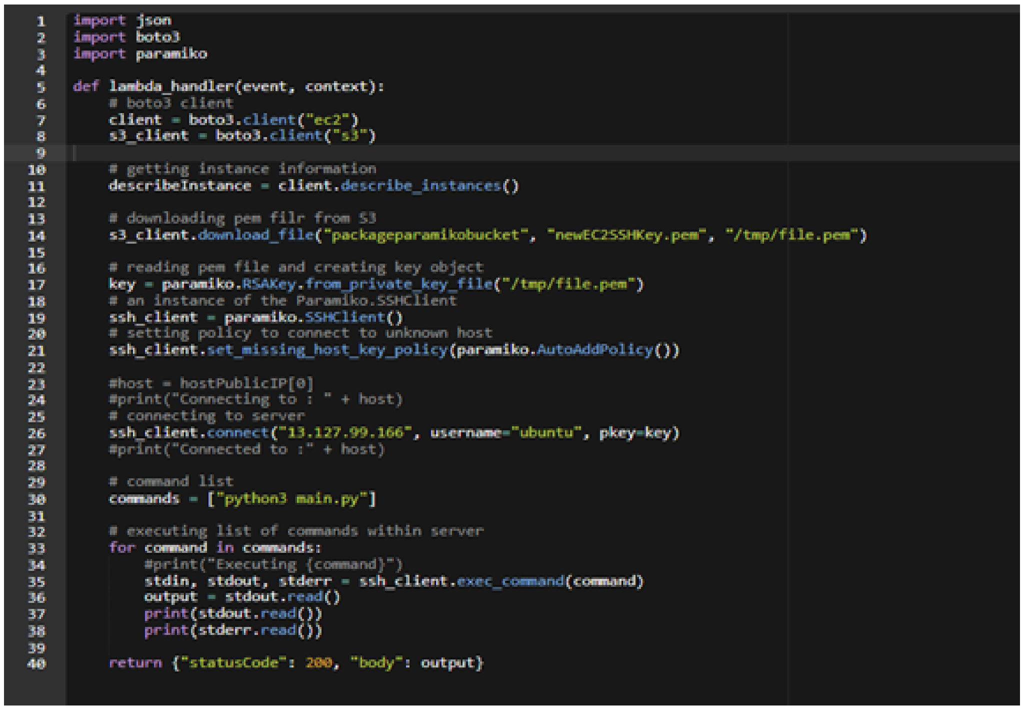
Name	AWS Region	Access	Creation date
packageparamikobucket	Asia Pacific (Mumbai) ap-south-1	Objects can be public	April 7, 2022, 00:00:36 (UTC+05:30)
sdgpdatabucket	Asia Pacific (Mumbai) ap-south-1	⚠ Public	March 1, 2022, 19:33:28 (UTC+05:30)
sdgpinput	Asia Pacific (Mumbai) ap-south-1	⚠ Public	March 25, 2022, 13:17:11 (UTC+05:30)

Figure 6 - Amazon S3 Bucket folders

The uploading of files to the s3 bucket when the user captures the image was done using dart itself, using an inbuilt dart function.

1.5.3 EC2 Instance

For operating the TensorFlow model used in the development of ObSign on the Amazon Web Services (AWS) infrastructure, an Amazon EC2 instance was established, which is a virtual server in Amazon's Elastic Compute Cloud (EC2). An Ubuntu virtual system was used to run the model with the help of WinSCP and PUTTY to connect run it locally furthermore a lambda function was implemented to SSH to the EC2 instance remotely using the paramiko library.



```python
import json
import boto3
import paramiko

def lambda_handler(event, context):
    # boto3 client
    client = boto3.client("ec2")
    s3_client = boto3.client("s3")

    # getting instance information
    describeInstance = client.describe_instances()

    # downloading pem filr from S3
    s3_client.download_file("packageparamikobucket", "newEC2SSHKey.pem", "/tmp/file.pem")

    # reading pem file and creating key object
    key = paramiko.RSAKey.from_private_key_file("/tmp/file.pem")
    # an instance of the Paramiko.SSHClient
    ssh_client = paramiko.SSHClient()
    # setting policy to connect to unknown host
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    #host = hostPublicIP[0]
    #print("Connecting to : " + host)
    # connecting to server
    ssh_client.connect("13.127.99.166", username="ubuntu", pkey=key)
    #print("Connected to :" + host)

    # command list
    commands = ["python3 main.py"]

    # executing list of commands within server
    for command in commands:
        #print("Executing {command}")
        stdin, stdout, stderr = ssh_client.exec_command(command)
        output = stdout.read()
        print(stdout.read())
        print(stderr.read())

    return {"statusCode": 200, "body": output}
```

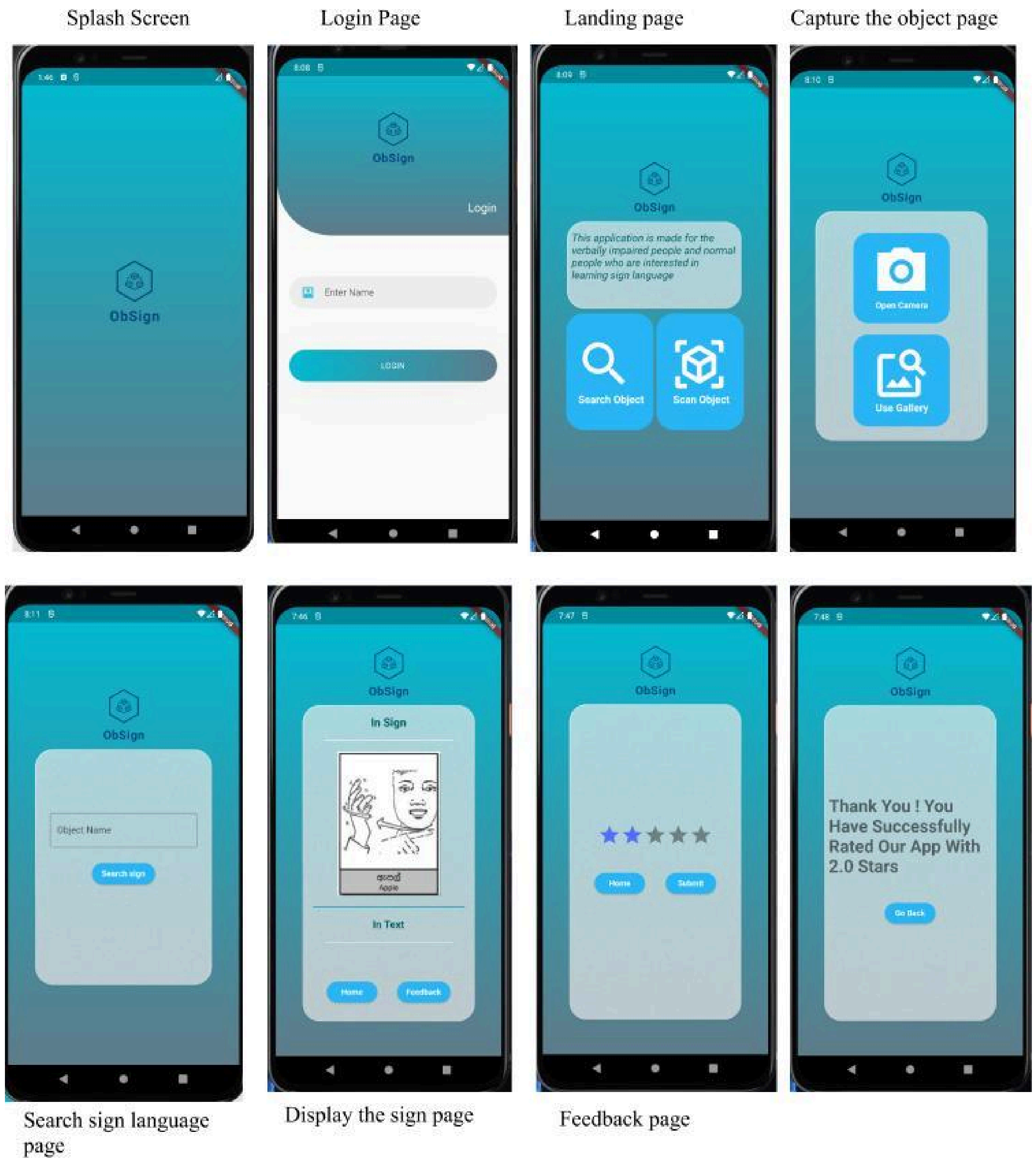
Figure 7 - Remotely run EC2 Instance code snippet

1.6 Implementation of the front-end component

1.6.1 Mobile Application

The project is a mobile application. Flutter, a cross-platform open-source user interface software development kit built by Google, was used to construct the mobile app (compatible with multiple operating systems).

The mobile application consists of 6 main pages such as Splash screen, Login Page, landing page, Capture the object page, Display the sign page and the Feedback page which are described below one after the other :



This application is made for the verbally impaired people and normal people who are interested in learning sign language

Thank You ! You Have Successfully Rated Our App With 2.0 Stars

Figure 8 - Application prototype

1.6.1.1 Splash Screen

The splash screen of the application consists of the logo and the application name which will appear when the application is launched from the user's mobile phone.

1.6.1.2 Login Page

If the user is already registered the application, then the user taps the login button. In this component, a form group has been created to validate the Login component. The validator will check whether the entered username and the password are valid. Even though the user changes the username and the password values in the validators it will still validate the form belonging to the entered value. The error message will be displayed only when an error occurs. If the user isn't registered to the application the user will have to register before moving to the main page. After the user enters the username & password the page will be allowed to move to the main page.

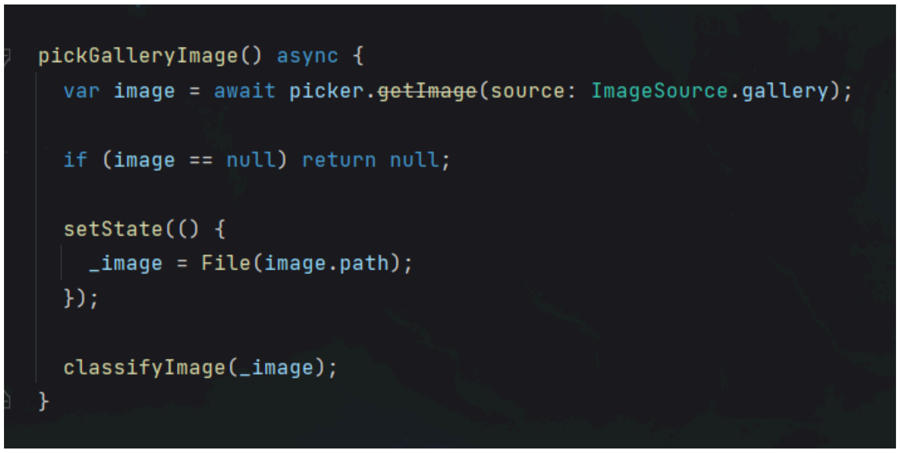
1.6.1.3 Landing Page

In the center of the page there is a one button called "Scan the object" that will open a new window in the user's mobile phone with the camera opened and "Search object" will open a new window with a text field for the user to enter the name of the object.

1.6.1.4 Capture the object page

In this page the user will capture the object and then the application will submit the image which is been taken to the s3 bucket where it is done using a dart function or the user can upload an image from the device.

The application was developed with the help of flutter plugins such as : Camera plugin which is a Flutter plugin for Android allowing access to the device cameras used to take images for the objects. Getting the available cameras and setting it, in this case the first camera which is the main camera(back):



```
pickGalleryImage() async {
  var image = await picker.~~getImage~~(source: ImageSource.gallery);

  if (image == null) return null;

  setState(() {
    _image = File(image.path);
  });

  classifyImage(_image);
}
```

Figure 9 - Code Snippet for opening camera

1.6.1.5 Display the sign language page

This page shows the relevant sign and the text of the object which was captured previously. In the bottom of the bar box users can see this page has two buttons called “Exit” & “Feedback”. When the user taps the “Exit” button, the user will be sent back to the landing page of the application. And when the user taps the “Feedback” button it will go to the feedback page.

1.6.1.6 Feedback page

This is the feedback page where users leave a feedback of the application get. There is a star rating system in the center of the bar box which users can select stars from 1 to 5 according to the performance and accuracy of the application. Below code snippet is used to import the star rating system to the feedback page.

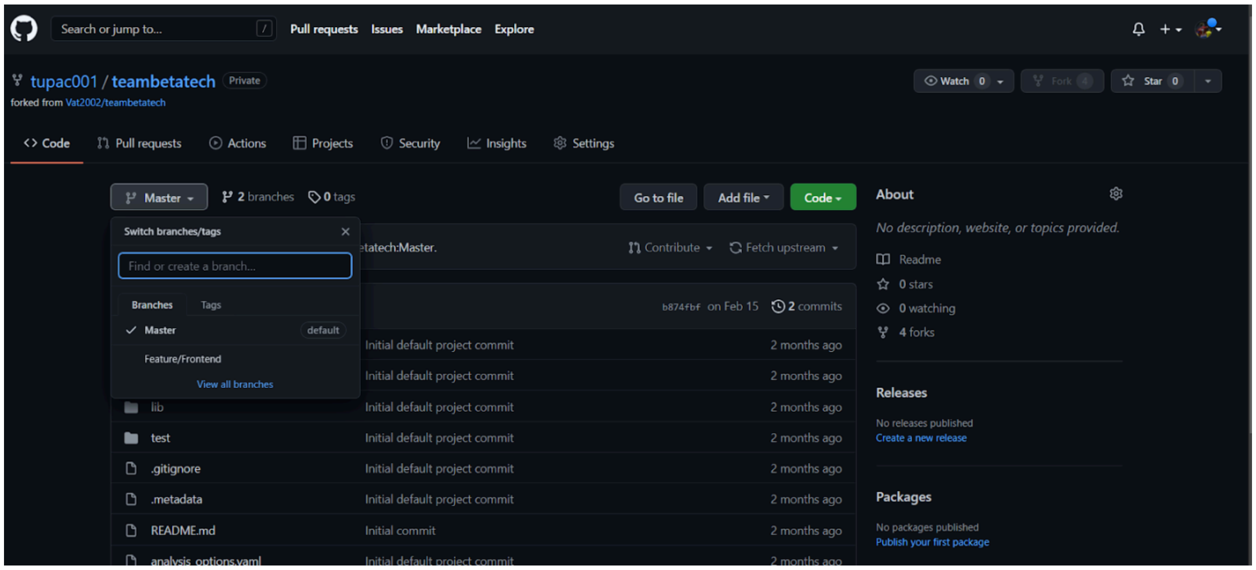
1.7 GIT Repository

While working on this project, the authors used GitHub for version control and collaboration.

As shown in the diagram below, the group leader created a repository called teambetatech and added all the team members as collaborators. The authors have developed two primary branches in this repository as

1.Main branch

2. Feature/Frontend



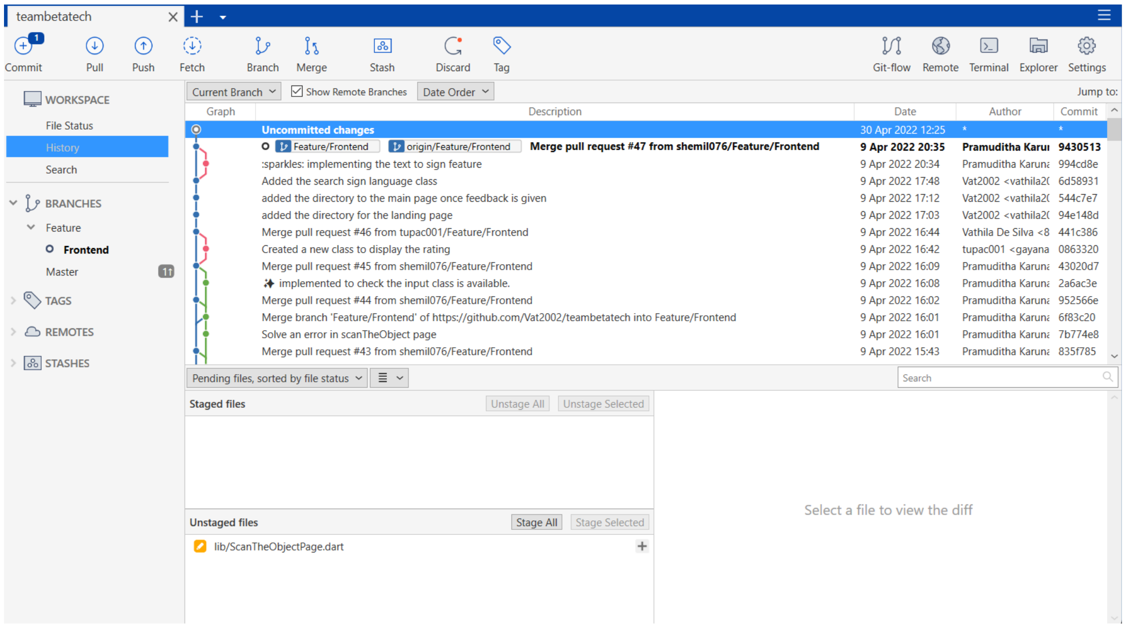
	b874fbf on Feb 15	2 commits
	Initial default project commit	2 months ago
	Initial default project commit	2 months ago
lib	Initial default project commit	2 months ago
test	Initial default project commit	2 months ago
.gitignore	Initial default project commit	2 months ago
.metadata	Initial default project commit	2 months ago
README.md	Initial commit	2 months ago
analysis_options.yaml	Initial default project commit	2 months ago

Figure 10 - GitHub repository

Feature Frontend -

Each member of the team updated their work and made changes to this branch, which contains all the frontend code. ObSign frontend contains the flutter project, which includes all front-end code. Before merging into the main branch, each contribution pushed by team members to the remote GitHub repository was reviewed.

As for the committing and pushing of the code SourceTree was used.



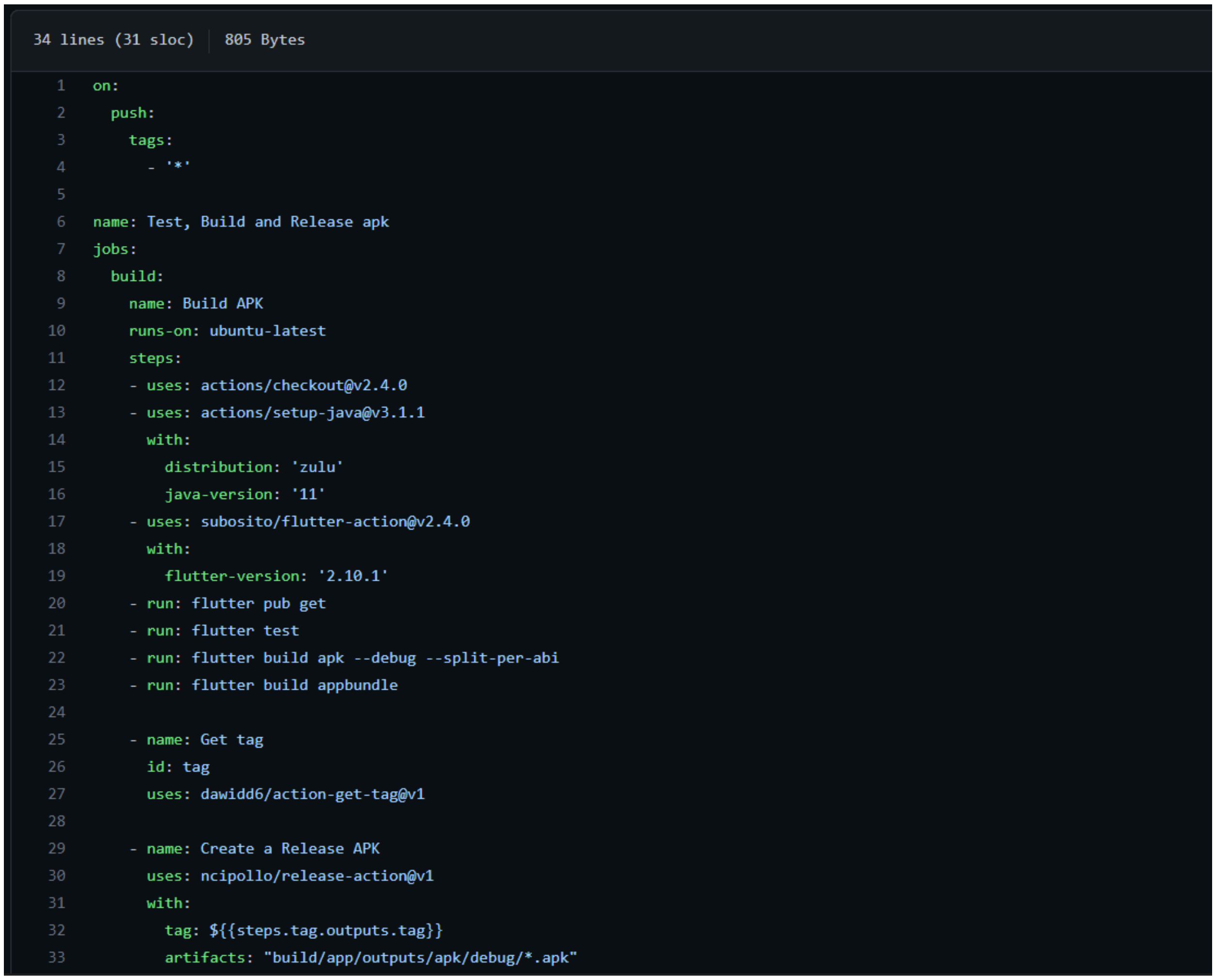
Description	Date	Author	Commit
Uncommitted changes	30 Apr 2022 12:25	*	*
Feature/Frontend origin/Feature/Frontend **Merge pull request #47 from shemil076/Feature/Frontend**	**9 Apr 2022 20:35**	**Pramuditha Karu**	**9430513**
:sparkles: implementing the text to sign feature	9 Apr 2022 20:34	Pramuditha Karun	994cd8e
Added the search sign language class	9 Apr 2022 17:48	Vat2002 <vathila2	6d58931
added the directory to the main page once feedback is given	9 Apr 2022 17:12	Vat2002 <vathila2	544c7e7
added the directory for the landing page	9 Apr 2022 17:03	Vat2002 <vathila2	94e148d
Merge pull request #46 from tupac001/Feature/Frontend	9 Apr 2022 16:44	Vathila De Silva <8	441c386
Created a new class to display the rating	9 Apr 2022 16:42	tupac001 <gayana	0863320
Merge pull request #45 from shemil076/Feature/Frontend	9 Apr 2022 16:09	Pramuditha Karun	43020d7
✨ implemented to check the input class is available.	9 Apr 2022 16:08	Pramuditha Karun	2a6ac3e
Merge pull request #44 from shemil076/Feature/Frontend	9 Apr 2022 16:02	Pramuditha Karun	952566e
Merge branch 'Feature/Frontend' of https://github.com/Vat2002/teambetatech into Feature/Frontend	9 Apr 2022 16:01	Pramuditha Karun	6f83c20
Solve an error in scanTheObject page	9 Apr 2022 16:01	Pramuditha Karun	7b774e8
Merge pull request #43 from shemil076/Feature/Frontend	9 Apr 2022 15:43	Pramuditha Karun	835f785

lib/ScanTheObjectPage.dart

Figure 11 - Usage of SourceTree software

1.8 Deployments/CI-CD Pipeline

The CI/CD pipeline was created using GitHub Actions. All the project codes are committed to the GitHub repository. After pushing the repository through the GitHub actions, the code builds up and runs the automated application testing.



```
34 lines (31 sloc) | 805 Bytes
on:
  push:
    tags:
      - '*'

name: Test, Build and Release apk
jobs:
  build:
    name: Build APK
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2.4.0
    - uses: actions/setup-java@v3.1.1
      with:
        distribution: 'zulu'
        java-version: '11'
    - uses: subosito/flutter-action@v2.4.0
      with:
        flutter-version: '2.10.1'
    - run: flutter pub get
    - run: flutter test
    - run: flutter build apk --debug --split-per-abi
    - run: flutter build appbundle

    - name: Get tag
      id: tag
      uses: dawidd6/action-get-tag@v1

    - name: Create a Release APK
      uses: ncipollo/release-action@v1
      with:
        tag: ${{steps.tag.outputs.tag}}
        artifacts: "build/app/outputs/apk/debug/*.apk"
```

Figure 12 - Code snippet of the GitHub action

1.9 Chapter Summary

The ObSign application's feature implementation was addressed in this chapter. This chapter also covered the frameworks and libraries utilized in the development of this application, as well as how each feature was implemented, with explanations provided through code fragments and screenshots. The project's testing phase will be discussed in the next chapter.

Chapter 2 Testing

2.1 Chapter Introduction

The testing phase of the developed object detection to sign language application is the subject of this chapter. The criteria for testing will be outlined first. To validate the functionality, the results of the functional and non-functional requirements testing will be examined. To ensure that components are operating effectively, unit testing will be carried out. To give knowledge on such areas, usability and compatibility testing will be covered.

2.2 Testing Criteria

Prior beginning the testing process, a criterion for what components of the platform should be examined and how had to be established. Testing would primarily focus on the functional requirements based on the fundamental data science component; it was decided. As a result, all potential flaws must be extensively evaluated. All the cores' performance must also be evaluated. Non-functional needs, as well as usability and compatibility testing, will be completed next.

2.3 Testing functional requirements

Table 1 - Functional requirements

Number	Requirements	Description	Priority	Status
FR1	Username login	To proceed, the user will be needed to provide the registered login at the start of the program.	Low	Completed
FR2	Recognizing and identifying the object through the user's mobile camera	The first step of the primary detection method. The user will have to hold the phone camera in an aligned manner to detect the object.	High	Completed

FR3	Providing the default character for use of the application	Default camera is added in the application to click the picture to process the relevant feature for the object selected by the user. Only difference is the default camera will not use any filter to the image so it will be more accurate when identifying the object.	Medium	Completed
FR 4	Search the sign language	The user can enter the name of the object and then the application will return the sign language	Medium	Completed
FR 5	Upload object image	The user can upload an image from the device and get the relevant sign language for the uploaded image	Medium	Completed

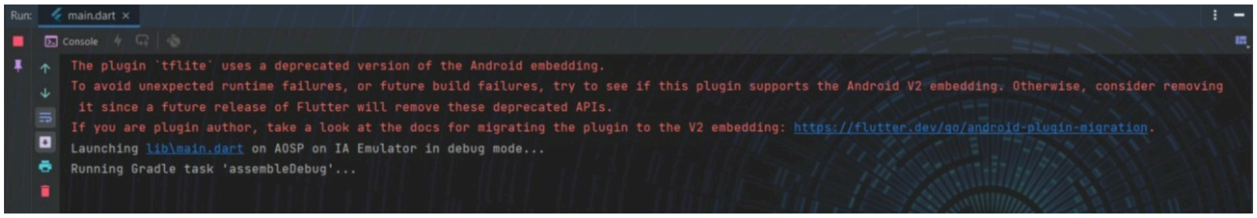
2.4 Testing non-functional requirements

Table 2 - Non-functional requirements

Number	Requirement	Description	Status
NFR1	Accuracy	The suggested project should be able to identify the object by requiring the user to hold the mobile camera in the right position, after which the system will compare the object to an existing database and display the object's sign language.	80%
NFR2	Performance	The system should be able to identify the object and process the data with the existing database in real time without any delays.	80%
NFR3	Usability	The system will be basic and simple, with interactive elements that make it simple to use and increase user experience.	70%
NFR4	Scalability	The system should be able to identify a wide range of items while being consistent.	85%
NFR5	Security	User's data should be kept private and cannot be misused.	80%
NFR6	Maintainability	Comprehensive and clear documentation should be given upon request so that designers or analysts may quickly become acquainted with the framework.	75%

2.5 Unit testing

Unit tests were the most commonly used test during the development and integration of this application. while in the process of testing the application it was found out that the camera component which uses the camera plugin was not responding properly, however by changing some dependencies in the pubspec.yaml file this issue was resolved. Also, when using the tflite model version depreciations were also shown which was also resolved later.



```
The plugin `tflite` uses a deprecated version of the Android embedding.
To avoid unexpected runtime failures, or future build failures, try to see if this plugin supports the Android V2 embedding. Otherwise, consider removing
 it since a future release of Flutter will remove these deprecated APIs.
If you are plugin author, take a look at the docs for migrating the plugin to the V2 embedding: https://flutter.dev/go/android-plugin-migration.
Launching lib\main.dart on AOSP on IA Emulator in debug mode...
Running Gradle task 'assembleDebug'...
```

Figure 13 - Tflite deprecated version error

2.6 Performance testing

The mobile application was tested on a generic emulator to confirm that it functions and performs the same in the current line of Android smartphones. The application launch speed was tested on several smartphones such as Huawei, Samsung, and Xiaomi as well as the optimal usage of the camera was tested.

2.7 Usability testing

Usability testing examines how easy the application is to use and how well the user interface and experience are facilitated. The mobile application was created utilizing user experience criteria, with the goal of making it simple and easy to use across all devices.

2.8 Compatibility testing

Due to the major released update of Flutter 2, the mobile application is now easily available in the form of a web application, desktop application as well, with the same codebase and proper automation between every application type which would equally share the same features as well as no coding required to ensure compatibility between both applications due to its automation provided from the new flutter engine.

2.9 Chapter Summary

The major goal of this chapter was to evaluate the prototype's functionalities and non-functionalities. During this phase, there were a lot of little issues that needed to be dealt with right away. This chapter also details all the extra testing carried out at this phase, such as compatibility testing. The evaluation of the developed system will be the subject of the following chapter.

Chapter 3 Evaluation

3.1 Chapter Overview

The testing phase of the ObSign application was the topic of the preceding chapter. The project's evaluation will be described in this chapter. It will conduct a detailed evaluation of the end users' evaluations, as well as a self-evaluation to demonstrate how successfully the project was done.

3.2 Evaluation methods

It was considered that a qualitative evaluation by specialists is critical to completely examine the developed prototype. As a result, the team members called a mentor and requested comments and team encompasses all the decisions made at various stages throughout the project's timeframe and is entirely dependent on the methodologies we used during implementation.

The evaluation is completed with the goal of determining whether our system contains the essential features and that the result is as predicted. It also ensures that the aforementioned judgments are appropriate and that the project follows proper guidelines.

3.3 Quantitative evaluation

Quantitative evaluation was performed to test the accuracy and overall performance of the system with statistics. The further details or result and testing about prediction under the testing chapter 2

3.4 Qualitative evaluation

The prototype was shown to a group of domain experts and end users. They looked through it and gave feedback and future modifications and improvements. The flow of the software and the essential functionalities were well understood by the reviewers. They were also given further explanations as to why this functionality was implemented at the time, they requested it. Below is feedback the developers received from, Mr. Dulanga Weragama , who is a quality assurance engineer.

"It's a good idea to use flutter since this can be used on cross platforms. And it is nice to see machine learning has been used as it will be a trending topic in the future. Overall application look is good and functioning as expected."

Another feedback given by Mr. Suresh Peiris, who is a software engineer at Inforwaves,

“Team Beta Tech has done an excellent job in developing a sign language cross conversion application that has features such as scanning objects and converting them to sign language, as well as text to sign language conversion, among other capabilities.

The team has created a custom machine learning model that can recognize objects and map them to a dataset of images. It would have been preferable in the future if they could have used GANs to generate results for custom phrases; additionally, if they had been able to develop gesture detection, that would have been an added advantage.

Overall, the team has done an outstanding job of identifying a real-world problem and developing a solution that addresses the "Quality Education" aspect of the Sustainable Development Goals.”

3.5 Self evaluation

Navigation between pages takes around 0 to 1.3 seconds.

Object detection - Object detection works flawlessly, as it identifies the object as the user presses the scan button, and the system begins the process of detecting the object, which takes around 1.8 seconds.

Once the object has been detected then it will return the sign language for the detected object which take around 1.2 seconds.

3.6 Chapter summary

This chapter details the project's overall evaluations as well as the assessment methodology utilized. This chapter delves deeper into the information received through qualitative and quantitative assessments, as well as a self-evaluation to determine how successfully the project was done. The conclusion chapter includes the project's goals and objectives, as well as legal, social, ethical, and professional concerns, as well as some limits of the research conducted.

Chapter 4 Conclusion

4.1 Chapter Overview

This chapter addresses the project's ultimate conclusion and the challenges that developed throughout its development, as well as the project's subsequent aims and how the authors employed object detection and their prior expertise to build a solution for the identified problem.

4.2 Achievements of aims and objectives

Within the timeline allotted, this project was able to accomplish the stated goal. The objectives that were completed are listed below.

1. The first project proposal included an overview of the project, including the problem context, goals and objectives, prototype scope and functionalities, and a feature comparison chart.
2. The literature review (LR) outlined the existing research activity in linked disciplines, as well as the research gap and a full explanation of each's strengths and weaknesses.
3. The Methodology chapter provided a brief overview of the methodologies used in the application's analysis, development, design, evaluation, and management. The hazards involved were emphasized, as well as the system's mitigation plan. Structures for teamwork breakdown structure were also designed. The project's use of project management and collaboration technologies was thoroughly examined.
4. The Onion model, selection of requirement elicitation techniques/methods with functional and non-functional needs, as well as their priority levels, use case diagram, and detailed use case descriptions were all part of the system requirements definition. It will be found in Chapter 4.
5. The chapter on Social, Legal, Ethical, and Professional Concerns provided a brief description of how social, legal, ethical, and professional issues are suitable for the study endeavor, as well as how they are mitigated. It was part of Chapter 5.
6. The System Architecture & Design chapter examined the high-level and low-level designs needed to build the application, as well as illustrations that help users understand the system. These include a high-level architectural diagram, a class diagram, an activity diagram, sequence diagrams, a UI design prototype with wireframes, and a description of the system process flow, all of which were discussed in Chapter 6.
7. The ObSign application's feature implementation was addressed in this chapter. This chapter also covered the frameworks and libraries utilized in the development of this application, as well as how each feature was implemented, with explanations provided through code fragments and screenshots, all of which were discussed in Chapter 1 Implementation.

8. The major goal of this chapter was to evaluate the prototype's functionalities and non-functionalities. This chapter also details all the extra testing carried out at this phase, such as compatibility testing, all of which were discussed in Chapter 2 Testing.
9. The project's overall evaluations as well as the assessment methodology utilized was discussed. This chapter delves deeper into the information received through qualitative and quantitative assessments, as well as a self-evaluation to determine how successfully the project was done, all of which were discussed in the Chapter 3 Evaluation.

4.3 Limitations of the research

A lot of conceptual and applied principles had to be learned over the internet with sub-optimal coaching due to undergraduate program models followed by team members who had a limited connection to computer vision.

4.4 Future enhancements

- Since this application only uses Sinhala sign language it can be increased to other sign languages such as the use of ASL, BANZSL, CSL, LSF, JSL, LSE, LSM etc.
- Another feature could be added where the user can do the sign language once it's shown and then the application will give a rating on how well the user does the sign language.
- A quiz feature can be added where once the user is shown the sign language a pop quiz will be shown to check if the user has learned the showed sign language of the scanned or given object.
- Full duplex translation will be introduced where the user can show the sign language and it shows and the object or the user can scan the object and it shows the relevant sign language.

4.5 Extra work

- Finalist of CodeSprint 6.0(Vathila, Pramuditha, Ravindu, Gayana, Tharanga)
- Finalist of Haxmas 2022(Vathila, Pramuditha, Ravindu, Gayana, Tharanga)
- Participants of google hash code(Vathila, Pramuditha, Ravindu)
- Participants of Code Rally(Pramuditha, Ravindu)
- Participants of Aurora Condigo(Pramuditha, Ravindu)

4.6 Concluding remarks

To summarize, the project's essential objectives were met, and with future modifications, this product may be effectively launched into the market without jeopardizing users' privacy.

References

TensorFlow. 2022. TensorFlow. [online] Available at: <https://www.tensorflow.org/> [Accessed 26 March 2022].

Matplotlib.org. 2022. Pyplot tutorial — Matplotlib 3.5.0 documentation. [online] Available at: <https://matplotlib.org/3.5.0/tutorials/introductory/pyplot.html> [Accessed 26 March 2022].

Appendix

Appendix A

Video presentation link :

https://drive.google.com/file/d/1UU3THITCo96aOPt92GapcBsKLj-WK806/view?usp=sharing