Faculty of computers and information

Graduation project documentation

SIGN LANGUAGE TRANSLATOR MOBILE APP.

(SLT App)

Supervised by:

Prof. Hatem Mohamed Abd-Elkader

prepared by:

Bahaa Ibrahim Hassan

Ahmed Mohsen Elgarhy

Eslam Abd Elhaleem Gohar

Eslam Basiouny Abd Elmaqsoud

Aya Ayman Mahmoud Salim

Sara Said Mohamed Maghawry

Ola Galal Abada Eisa

Acknowledgement

First and foremost, we would like to thank Allah and express our gratitude to all those who share their knowledge and offer their help to let us understand and think of this project.
We will appreciate remember who provide us with ideas to improve this work. We are so grateful to our supervisor DR. Hatem Mohamed and those who help us to improve our idea and support us with information and helpful suggestion.
We are deeply indebted to the people whose innovative suggestions and encouragement helped us all the time.

This documentation describes our graduation project in five chapters:

- In Chapter 1 you will find a brief introduction about the project.
- Chapter 2 discusses the application analysis, planning.
- In Chapter 3 you will find details about project design.
- In Chapter 4 you will find project implementation work.
- In Chapter 5 include the project testing details.

Contents list:

List of figures

Chapter 1

Introduction:

1.1 Abstract

We all have at least one person with special needs (disable people). This project concerns deaf and dumb as we find difficulty in dealing with them and understand their needs or translate their sign language.
So, we decided to make a simple app project to convert our speech to sign language and vice versa to ease our communication with them. The user will use this app to type a text and service will translate it to sign language and on the other side someone else will capture a picture of sign language and the system will convert it to text to understand their need and ease our communication between us and we haven't to learn sign language. So, if we can do this app, we will help them and help us to understand each other and avoid them from embarrassment.

Sign language is used by 70 million people around the world and there's a lack in communication between deaf and dump community and our community. The deaf and dump community moves back When it comes to the interactive part with people with special needs, this communication gap has exacerbated. Because our community have no idea about sign language. Text-to-sign & Sign-to-text is an application that translate text to sign language by using stored sign videos that will act the movement of the translated word & translate Sign language to text, using mobile camera to capture the picture of sign language and there is as model that understand the sign and turns it to text.

1.2 The Triple Constraint of our project:

- Project Scope
- Time
- Cost

Project scope:

1- The main goal of our project is when we launch the application, it achieves the target of our project (text will be translated to sign language and vice versa).

2- App will help normal people in communication with deaf and dump.

3- Helps them understand our speech by converts it to sign language.

4- We aren’t need to learn sign language to communicate with them.

5- Disable people can easily communicate with outside world.

6- Decrease the risk of failure as possible.

1.3 Work elements and techniques:

1.3.1 Mobile application

The main work is the android mobile application which translate speech to sign language video and translate a picture of sign language to text type.

1.3.2 Machine learning model:

In our tracking system we need a machine algorithm meet our needs and be compatible with our data so we use deep learning by using Tensorflow library to detect the error.

1.3.3 Techniques

Object detection API using Tensorflow (SIGN TO TEXT):

it takes the object like the captured picture and make detection on it to know what this picture means in our sign language database and return the result and translate it to text.

(TEXT TO SIGN):

We have a static words or phrases in database each of them has a video of sign language acting the word so when we enter text, the system will search in database and present the video of specific text.

1.4 USED TOOLS

Before identifying the tools, we found we need:

1. Database engine to store words, phrases, sign language photos and their describing sign language videos.
2. UI to design user interfaces that will be used by user through app.
3. Database server.
4. Deep learning (Tensorflow).
5. Programming languages.

1.4.1 Database engine

It is the basic software component that a database management system uses to read and search data from a database and it is the most frequently used interchangeably with database server and in our project, we will depend on NOSQL database engine.
We search about the video equals the sent text (static data).

1.4.2 Design user interfaces (UI)

The goal of user interface design is to produce a user interface which makes it easy, efficient, and enjoyable to operate a machine in the way which produces the desired result.
In our project we will use android, HTML, CSS and JavaScript to design the layout.

1.4.3 Database server

A database server is a computer program that provides database services to other computer programs or computers as defined by the client–server model and most of the Database servers work with the base of query language.

1.3.4 Deep learning

Deep leaning is a machine learning approach depends on neural networks that produce a very good accuracy and need huge dataset to learn and find a pattern between data.
We use TensorFlow library to make detection on the sign language to know what is it mean.

1.4.5 Programming languages

A programming language is designed to communicate the user instructions to a machine
and can be used to create programs to control the behavior of a machine or to express algorithms.

We use more than one language in our project as:

1. The algorithms of machine learning will execute using android programming language.
2. The interaction between user interface and the system will also be controlled by android programming language.
3. Using NOSQL language to interact with database.
4. Using **Python** in model.
5- Using **JAVA** in android.

1.5 Stakeholders

1- disable people

2- anyone can use it

3- developer

4- system analyst

5- tester

6- sponsor

7- all team members who participated in the work of the project.

1.6 Project steps

1. Study and analysis the project idea from all aspects.
2. Search and choose the best algorithm that applies our project idea.
3. Modeling and simulating our system.
4. Solve problems and errors in the system.
5. Deploy and implement the system in real world.
6. Test the accuracy of the app and machine learning model
7. Evolution the system in the future.

Chapter 2

System Analysis

2.1 Introduction

Systems analysis is a problem-solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose.
It is a main step in any business as firstly we determine the requirements, available resources, system parts and how each part interacts with another part.
System analysis in information technology is widely used as we identify the system requirements, model the system and testing it to repair errors before final deployment the whole system.

2.2 System development life cycle

1. Planning

In this step we identify goals, establish strategies to achieve goals and develop plans to integrate activities.

2. Analysis

At first, we should analysis the whole system requirements, available resources,
and interaction between the different parts in the system and we will use in our project the UML analysis model.

3. Implementation

Put the system in the action.

4. Review

Testing the system to ensure there is no errors and will work correctly after deployment.

2.3 System requirements

It represents what system need to be implemented and this term is widely used in software engineering and in analysis any systems. There is more than type in the system requirements as software requirements, functional requirements and non-functional requirements.

2.3.1 Software requirements

1. Machine learning to detect sign language to know what is it mean to translate it to text.
2. Software to allow the system actors to interact with the system.
3. Android operating system.

2.3.2 Functional requirements

According to our project goal we have to do following things …

1. User type a text.
2. System convert it to sign video.
3. User capture a picture of sign language.
4. System converted it to PNG.
5. The model will translate it to text.

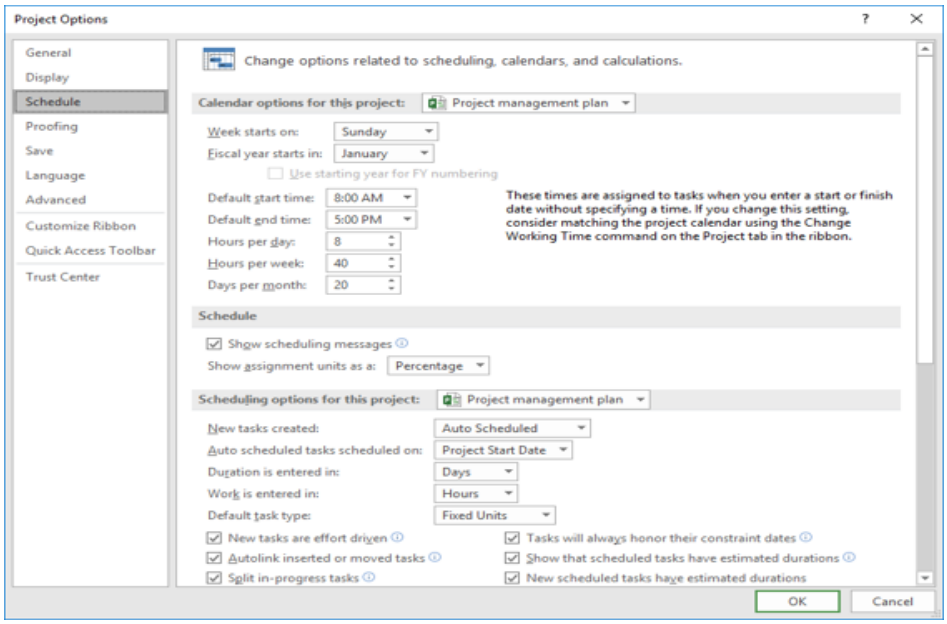
2.3.3 Non-Functional requirements

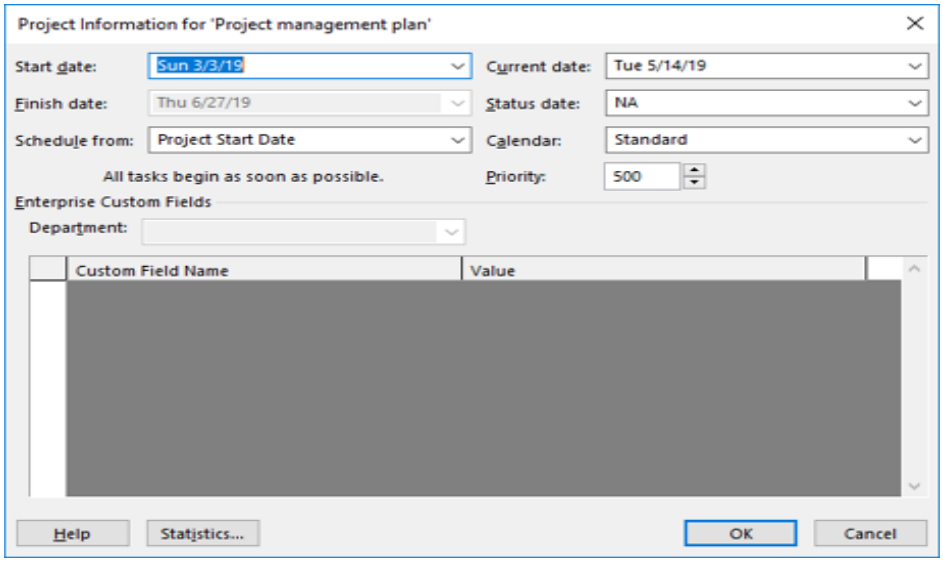
1. Security
2. Safety
3. Accuracy
4. quality
5. Speed
6. Availability
7. Usability

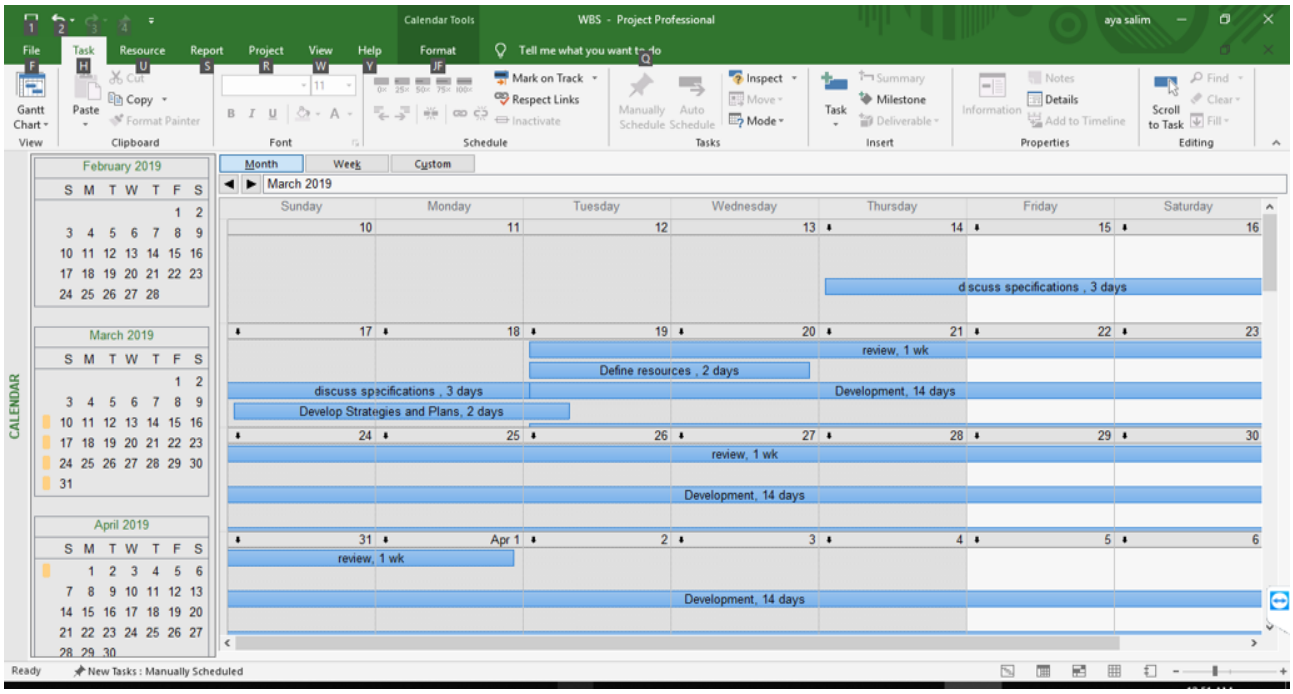
2.4 Project calendar

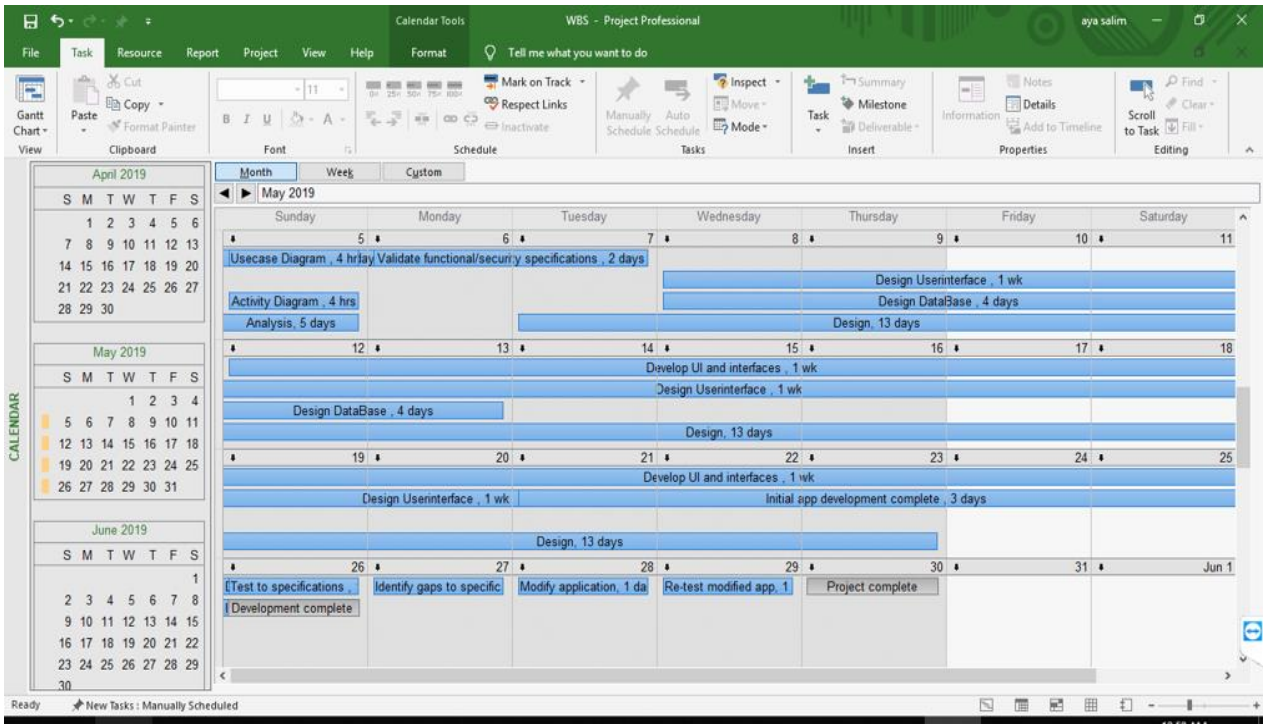
Description

The project starts in 3-3-2019 and ends 27-6-2019
Work time per day is 8 hours (8 am :12 pm and 1 pm: 5 pm)
Work time per week is 40 hours.
Week in work starts in Sunday and ends in Thursday.
Weekend days are Friday and Saturday.

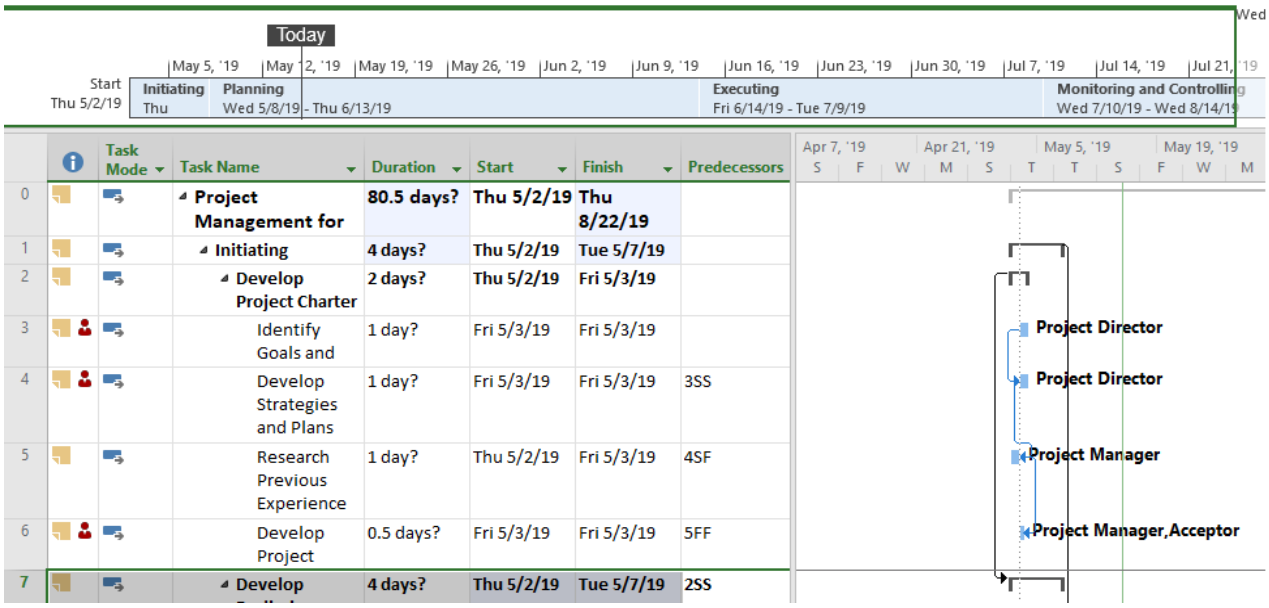




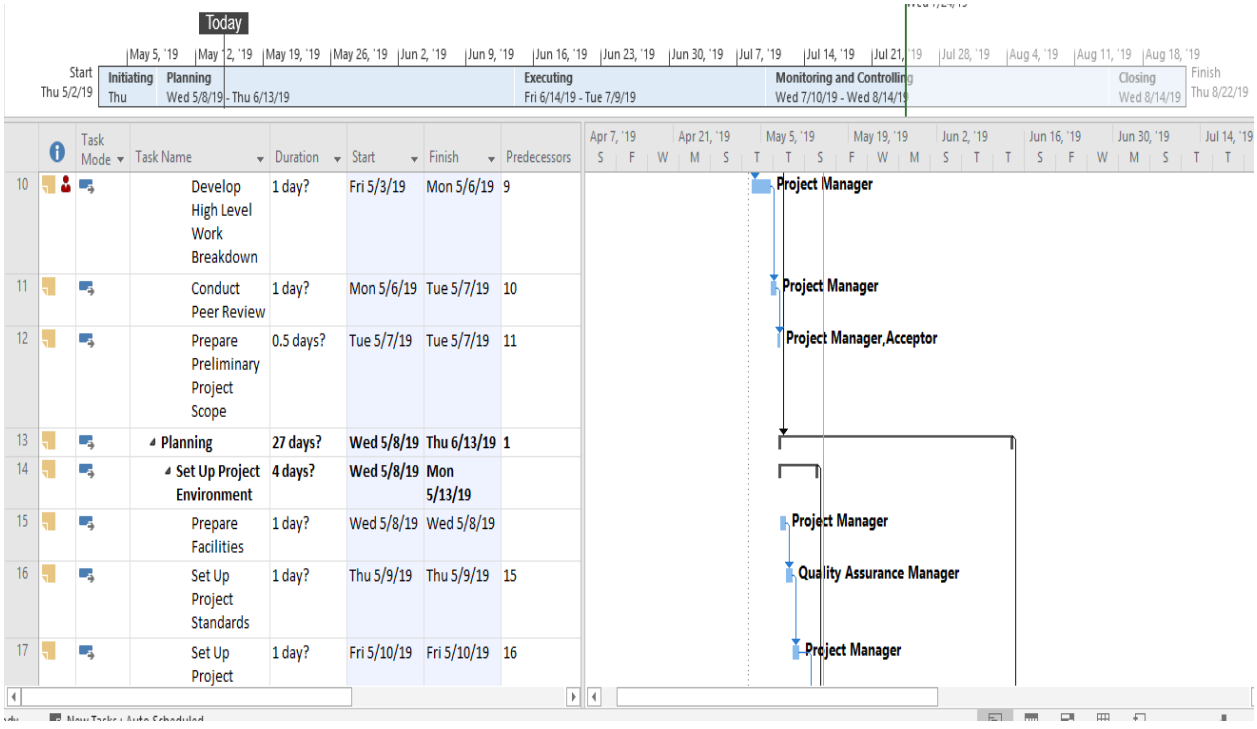




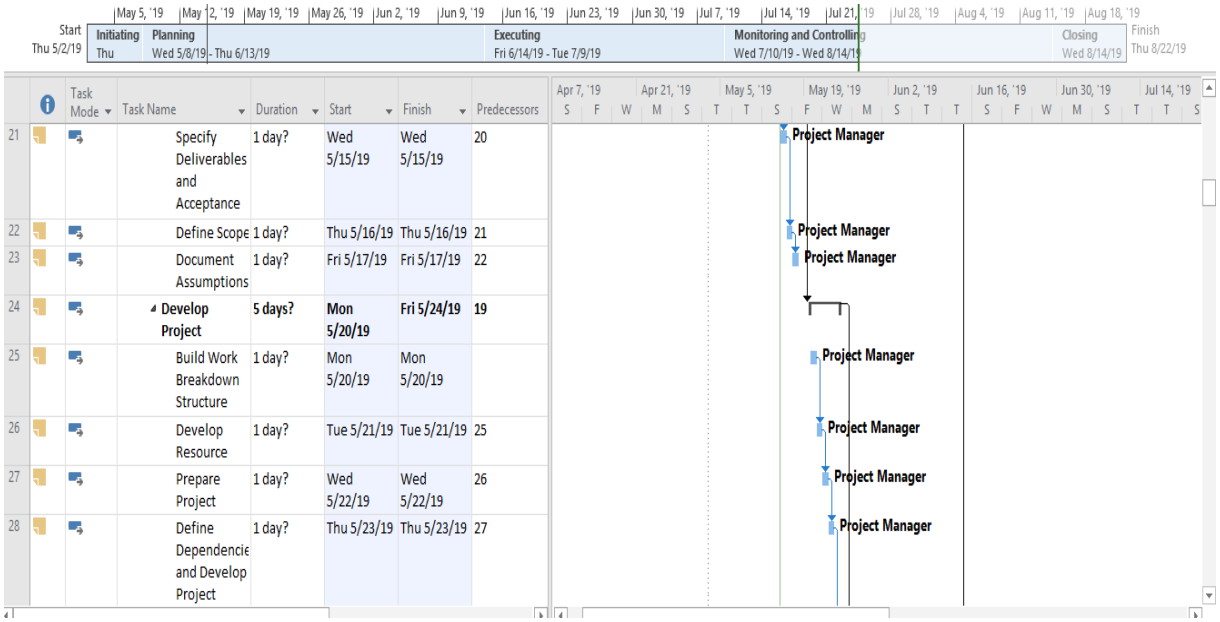
2.5 WBS & Gantt chart



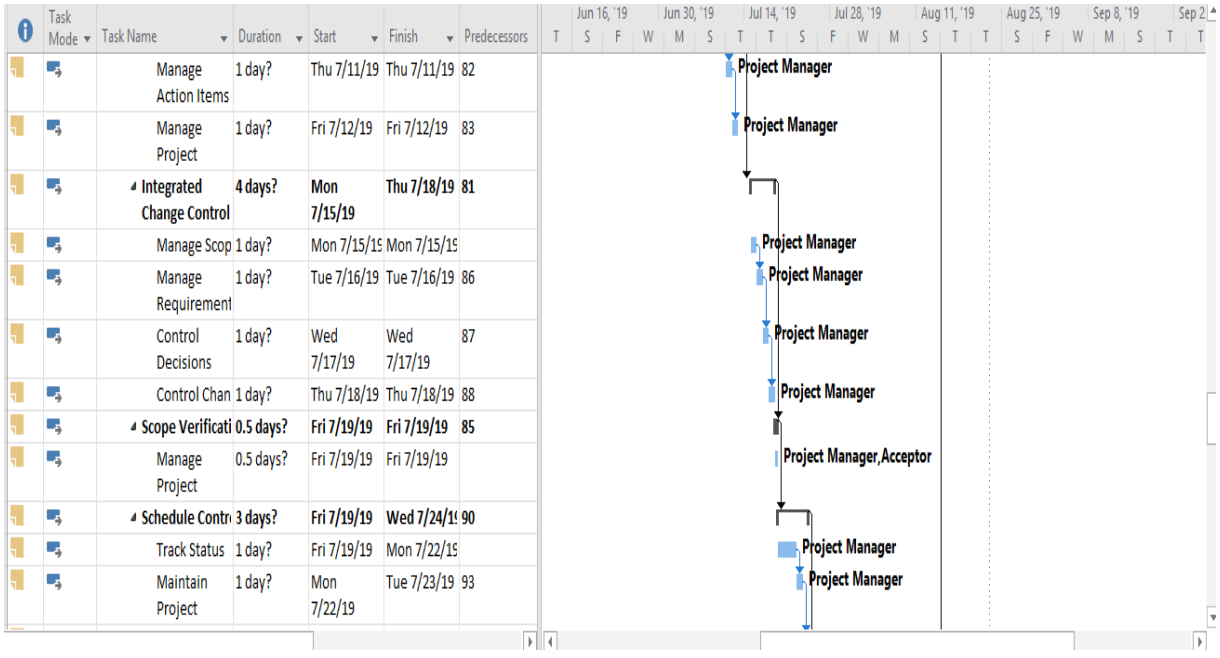
	Task Mode	Task Name	Duration	Start	Finish	Predecessors
0		**Project Management for**	**80.5 days?**	**Thu 5/2/19**	**Thu 8/22/19**	
1		**Initiating**	**4 days?**	**Thu 5/2/19**	**Tue 5/7/19**	
2		**Develop Project Charter**	**2 days?**	**Thu 5/2/19**	**Fri 5/3/19**	
3		Identify Goals and	1 day?	Fri 5/3/19	Fri 5/3/19	
4		Develop Strategies and Plans	1 day?	Fri 5/3/19	Fri 5/3/19	3SS
5		Research Previous Experience	1 day?	Thu 5/2/19	Fri 5/3/19	4SF
6		Develop Project	0.5 days?	Fri 5/3/19	Fri 5/3/19	5FF
7		**Develop**	**4 days?**	**Thu 5/2/19**	**Tue 5/7/19**	2SS



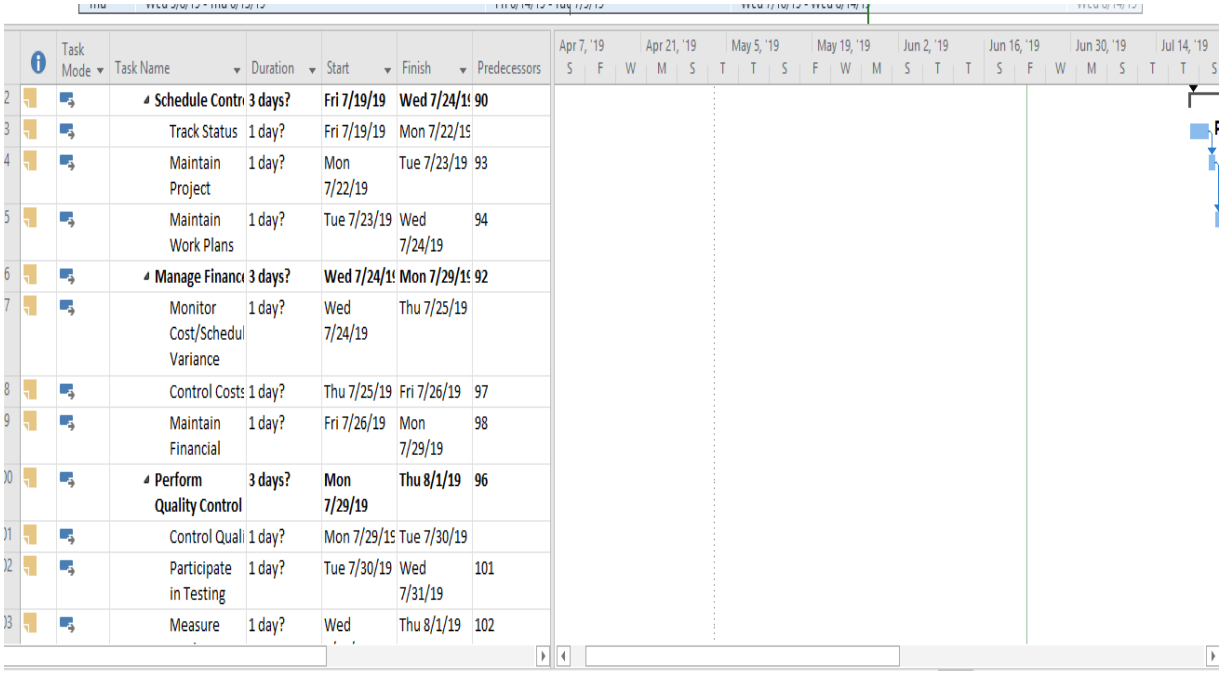
	Task Mode	Task Name	Duration	Start	Finish	Predecessors
10		Develop High Level Work Breakdown	1 day?	Fri 5/3/19	Mon 5/6/19	9
11		Conduct Peer Review	1 day?	Mon 5/6/19	Tue 5/7/19	10
12		Prepare Preliminary Project Scope	0.5 days?	Tue 5/7/19	Tue 5/7/19	11
13		**Planning**	**27 days?**	**Wed 5/8/19**	**Thu 6/13/19**	1
14		**Set Up Project Environment**	**4 days?**	**Wed 5/8/19**	**Mon 5/13/19**	
15		Prepare Facilities	1 day?	Wed 5/8/19	Wed 5/8/19	
16		Set Up Project Standards	1 day?	Thu 5/9/19	Thu 5/9/19	15
17		Set Up Project	1 day?	Fri 5/10/19	Fri 5/10/19	16



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
21		Specify Deliverables and Acceptance	1 day?	Wed 5/15/19	Wed 5/15/19	20
22		Define Scope	1 day?	Thu 5/16/19	Thu 5/16/19	21
23		Document Assumptions	1 day?	Fri 5/17/19	Fri 5/17/19	22
24		**Develop Project**	**5 days?**	**Mon 5/20/19**	**Fri 5/24/19**	**19**
25		Build Work Breakdown Structure	1 day?	Mon 5/20/19	Mon 5/20/19	
26		Develop Resource	1 day?	Tue 5/21/19	Tue 5/21/19	25
27		Prepare Project	1 day?	Wed 5/22/19	Wed 5/22/19	26
28		Define Dependencie and Develop Project	1 day?	Thu 5/23/19	Thu 5/23/19	27



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
		Manage Action Items	1 day?	Thu 7/11/19	Thu 7/11/19	82
		Manage Project	1 day?	Fri 7/12/19	Fri 7/12/19	83
		Integrated Change Control	**4 days?**	**Mon 7/15/19**	**Thu 7/18/19**	**81**
		Manage Scop	1 day?	Mon 7/15/19	Mon 7/15/19	
		Manage Requirement	1 day?	Tue 7/16/19	Tue 7/16/19	86
		Control Decisions	1 day?	Wed 7/17/19	Wed 7/17/19	87
		Control Chan	1 day?	Thu 7/18/19	Thu 7/18/19	88
		Scope Verificati	**0.5 days?**	**Fri 7/19/19**	**Fri 7/19/19**	**85**
		Manage Project	0.5 days?	Fri 7/19/19	Fri 7/19/19	
		Schedule Contr	**3 days?**	**Fri 7/19/19**	**Wed 7/24/19**	**90**
		Track Status	1 day?	Fri 7/19/19	Mon 7/22/19	
		Maintain Project	1 day?	Mon 7/22/19	Tue 7/23/19	93



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
2		**Schedule Contr**	**3 days?**	**Fri 7/19/19**	**Wed 7/24/19**	**90**
3		Track Status	1 day?	Fri 7/19/19	Mon 7/22/19	
4		Maintain Project	1 day?	Mon 7/22/19	Tue 7/23/19	93
5		Maintain Work Plans	1 day?	Tue 7/23/19	Wed 7/24/19	94
6		**Manage Financ**	**3 days?**	**Wed 7/24/19**	**Mon 7/29/19**	**92**
7		Monitor Cost/Schedul Variance	1 day?	Wed 7/24/19	Thu 7/25/19	
8		Control Costs	1 day?	Thu 7/25/19	Fri 7/26/19	97
9		Maintain Financial	1 day?	Fri 7/26/19	Mon 7/29/19	98
10		**Perform Quality Control**	**3 days?**	**Mon 7/29/19**	**Thu 8/1/19**	**96**
11		Control Quali	1 day?	Mon 7/29/19	Tue 7/30/19	
12		Participate in Testing	1 day?	Tue 7/30/19	Wed 7/31/19	101
13		Measure	1 day?	Wed	Thu 8/1/19	102

Apr 7, '19 | Apr 21, '19 | May 5, '19 | May 19, '19 | Jun 2, '19 | Jun 16, '19 | Jun 30, '19 | Jul 14, '19

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Add New Column
		Closing	**6 days?**	**Wed 8/14/19**	**Thu 8/22/19**	**80**		
		Close Project	**5 days?**	**Wed 8/14/19**	**Wed 8/21/19**			
		Assess Satisfaction	1 day?	Wed 8/14/19	Thu 8/15/19		Project Manager	
		Summarize Project Results and Lessons	1 day?	Thu 8/15/19	Fri 8/16/19	121	Project Manager	
		Review and Recognize Team	1 day?	Fri 8/16/19	Mon 8/19/19	122	Project Manager	
		Close Out the Project	1 day?	Mon 8/19/19	Tue 8/20/19	123	Project Manager	
		Review and Reconcile Financial Performance	1 day?	Tue 8/20/19	Wed 8/21/19	124	Project Manager	
		Contract Closur	**1 day?**	**Wed 8/21/19**	**Thu 8/22/19**	**120**		
		Close Contra	1 day?	Wed 8/21/19	Thu 8/22/19		Contracts Manag	

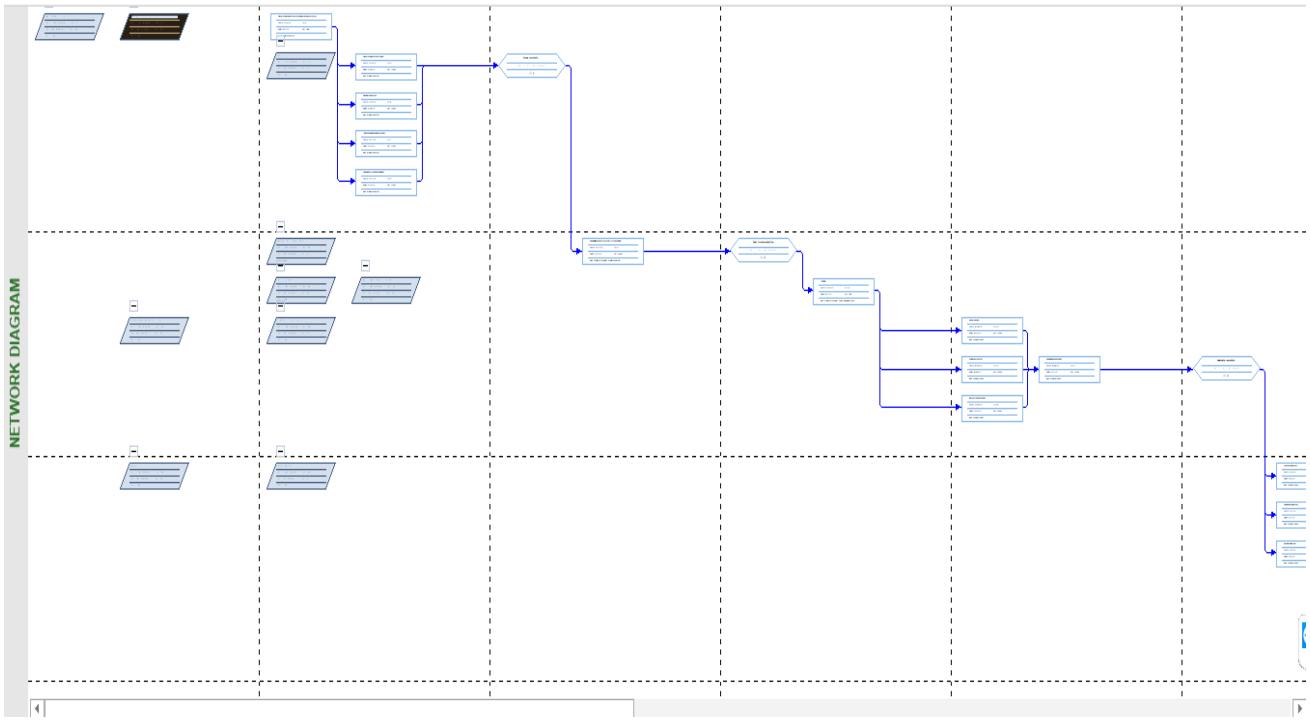
May '19 | Jun '19
21 | 28 | 5 | 12 | 19 | 26 | 2

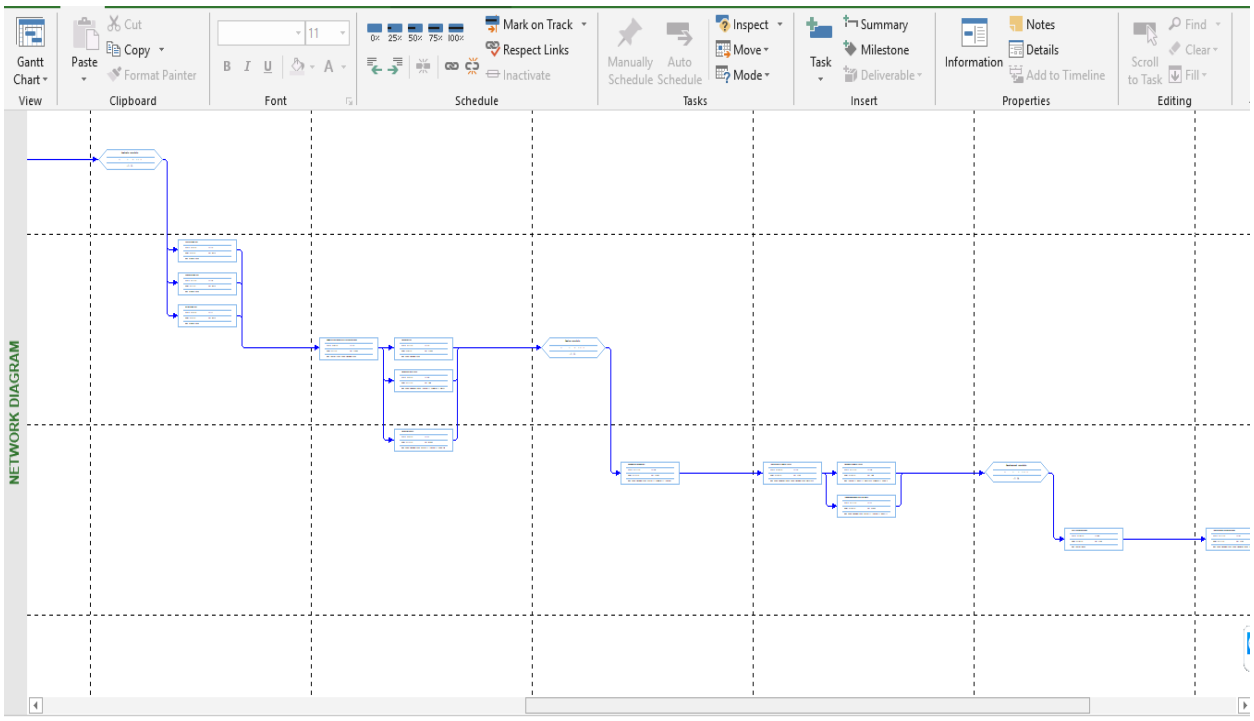
Task Mode	Task Name	Duration	Start	Finish	Predecessors
	Manage Communicati	1 day?	Thu 8/1/19	Fri 8/2/19	
	Meet With Te	1 day?	Fri 8/2/19	Mon 8/5/19	105
	Monitor Team	1 day?	Mon 8/5/19	Tue 8/6/19	106
	Conduct Team Performance	1 day?	Tue 8/6/19	Wed 8/7/19	107
	Produce Performance Reports	**1 day?**	**Wed 8/7/19**	**Thu 8/8/19**	**104**
	Report Weekly	0.5 days?	Wed 8/7/19	Wed 8/7/19	
	Report Monthly	0.5 days?	Thu 8/8/19	Thu 8/8/19	110
	Manage Stakeholders	**2 days?**	**Thu 8/8/19**	**Mon 8/12/19**	**109**
	Monitor Satisfaction	1 day?	Thu 8/8/19	Fri 8/9/19	
	Resolve Issue	1 day?	Fri 8/9/19	Mon 8/12/19	113

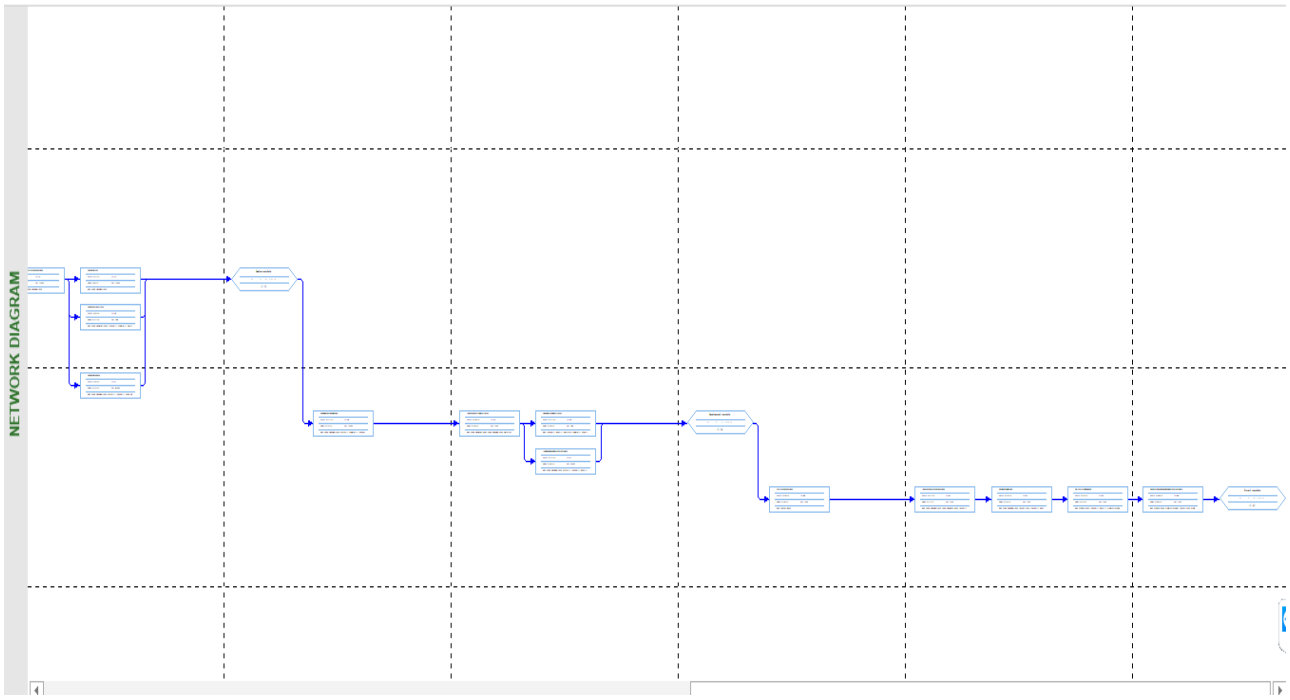
Apr 7, '19 Apr 21, '19 May 5, '19 May 19, '19 Jun 2, '19 Jun 16, '19

S F W M S T T S F W M S T T S F

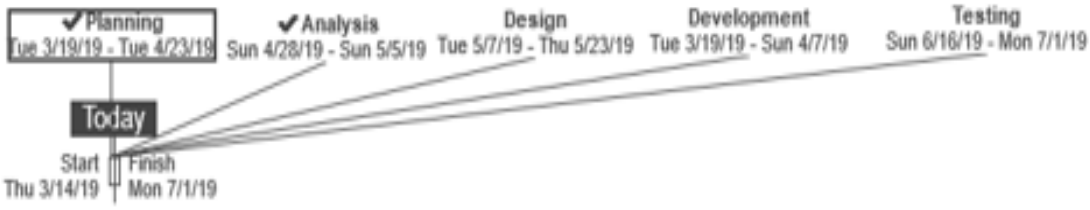
2.6 Network diagram




Gantt Chart
Paste
Cut
Copy
Format Painter
Mark on Track
Respect Links
Inactivate
Manually Schedule
Auto Schedule
Inspect
Move
Mode
Task
Summary
Milestone
Deliverable
Information
Notes
Details
Add to Timeline
Scroll to Task
Find
Clear
Fill
View
Clipboard
Font
Schedule
Tasks
Insert
Properties
Editing
NETWORK DIAGRAM


NETWORK DIAGRAM

2.7 Timeline



2.8 Activity Table

Activity id	WBS id	Activity list	Optimistic time	Most likely	Pessimistic	Duration	Predecessor
1	1	Determine the project manager &team project	4	7	10	7	-
2	4	Determine project scope	1	2	3	2	1
3	4	Define resources	1	2	4	2	2
4	4	Identify Goals and Objectives	1	2	3	2	3
5	4	Develop Strategies and Plans	2	2	4	2	4
6	10	Review with team members/stakeholder	2	3	5	3	9
7	16	observation	2	2	3	2	15
8	13	Manage Project Communication	9	10	12	10	12
9	24	Design User interface	6	7	9	7	28

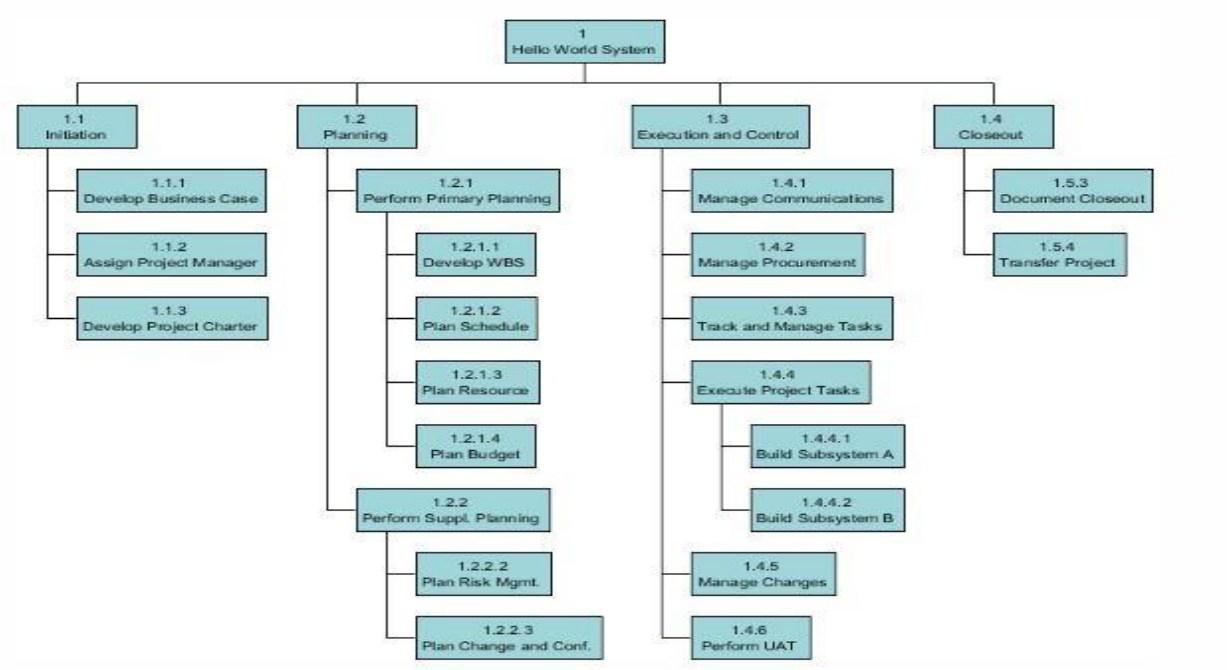
2.8.1 Sequence activity

Activity id	WBS id	Activity list	Optimistic time	Most likely	Pessimistic	Duration	Predecessor
10	33	Review functional design	2	2	3	2	32
11	33	Identify external interfaces	1	2	3	2	33
12	33	Develop UI and interfaces	1	2	4	2	34
13	33	Initial app development complete	1	2	3	2	35
14	39	Test to specifications	2	2	4	2	38
15	39	Identify gaps to specifications	1	1	2	1	40
16	39	Modify application	1	1	3	1	41
17	39	Re-test modified app	1	1	2	1	42
18	39	App Testing and Final Development complete	1	1	3	1	42,43

2.8.2 Milestones

Milestone No	Milestone	Mandatory/ Optional	Completion Date	Verification
1	Project Start	Mandatory	19/3/2019	Manager Approval
2	Scope complete	Mandatory	4/4/2019	Manager Approval
3	Post implementation	Mandatory	10/4/2019	Manager Approval
4	Analysis complete	Mandatory	6/5/2019	Manager Approval
5	Design complete	Mandatory	26/5/2019	Manager Approval
6	Development complete	Mandatory	16/6/2019	Manager Approval
7	Project complete	Mandatory	23/6/2019	Manager Approval

2.8.3 Breakdown structure



2.8.4 Activity resource requirement

Activity ID	WBS ID	Resource name	Resource Type	Quantity	Cost/unit
1	1	Node.js developer	Work	3	1000/developer
2	1.1	React native developer	Work	4	1100/developer
3	1.2	Database developer	Work	1	1300/developer
4	2, 2.1, 2.2, 2.3	Internet	Material	1	6.25$/Mbs
5	2 , 2.1, 2.2, 2.3, 4, 4.1, 4.2, 4.3	Laptops	Material	7	1500/laptop
6	3.1	Architecture creating tool	Material	1	200/unit
7	3.2	Designing tool	Material	1	200/unit
8	3.3	Schema design tool	Material	1	210/unit
9	5.1	Android mobile	Material	7	2000/unit
10	5.2	Ios mobile	Material	1	1000/unit
11	6.1	Book material	Material	50	80/paper

2.9 UML analysis model

It is referring to “Unified Modeling Language” that is a modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system.
UML has many types of diagrams which are divided into two categories structured UML
diagrams and behavioral UML diagrams.

- 2.9.1 Use case Diagram

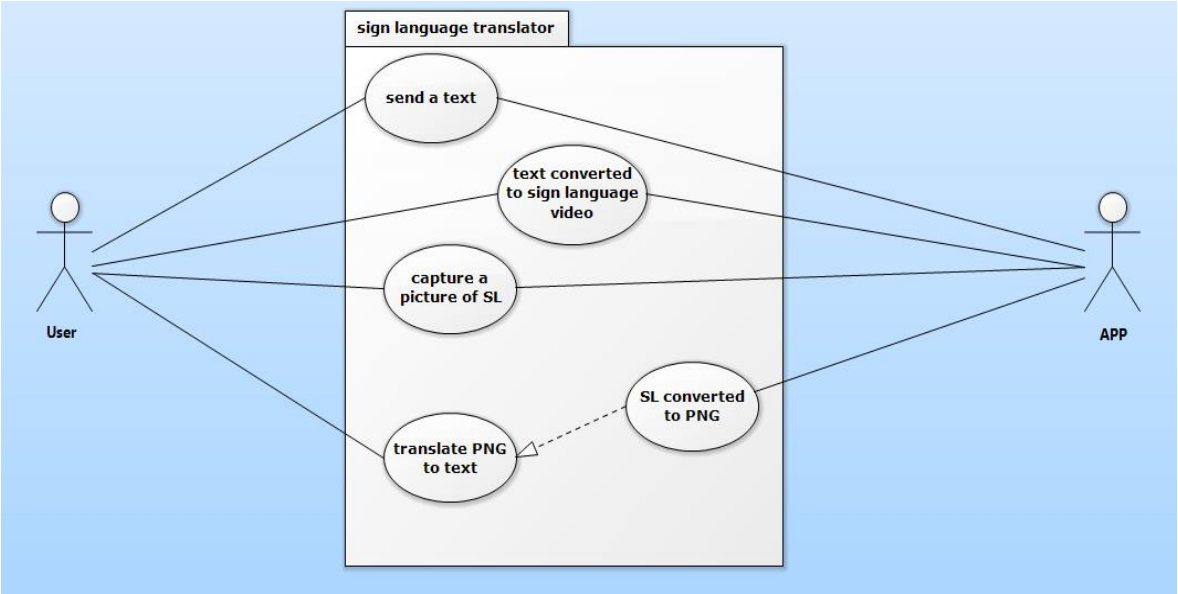


Figure 2.1 Use Case Diagram

1- User will type a text in the App.
2- The app will convert it to sign video and return the result to user.
3- User take a picture of sign language and send it to the app.
4- The app will convert SL to PNG.
5- The app will convert PNG to text.
6- The user will receive the result of translation (text).

- 2.9.2 Sequence Diagram

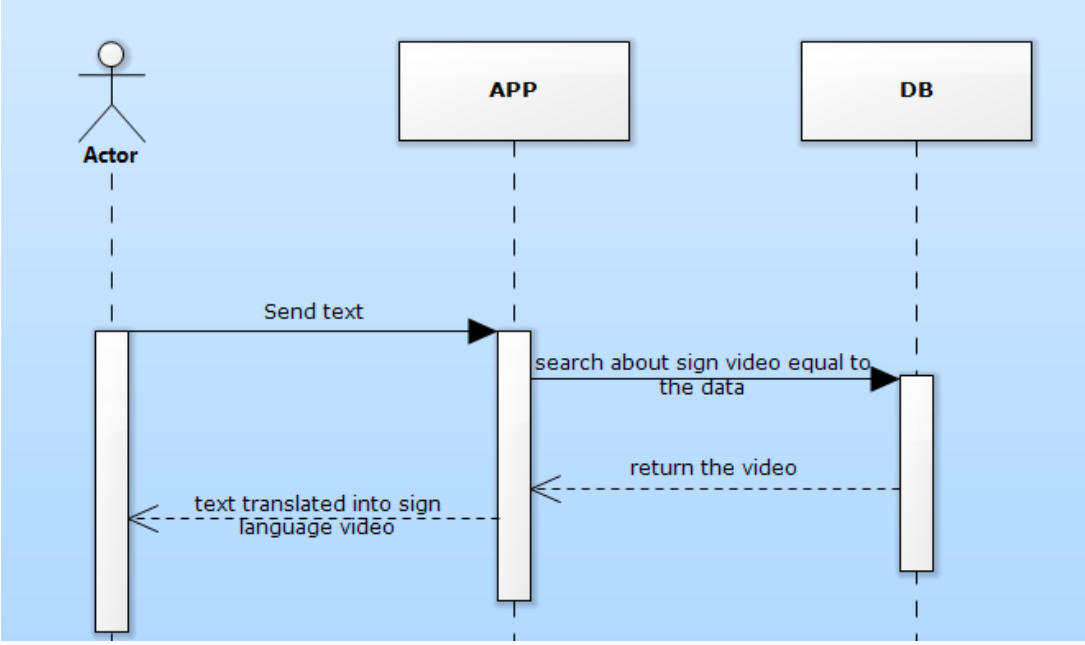


Figure 2.2 User1 Sequence Diagram

When we type a text, the app searches in DB (saved data) about a sign language video act the mean of the text and return with the result (specific video) to send it to the user.

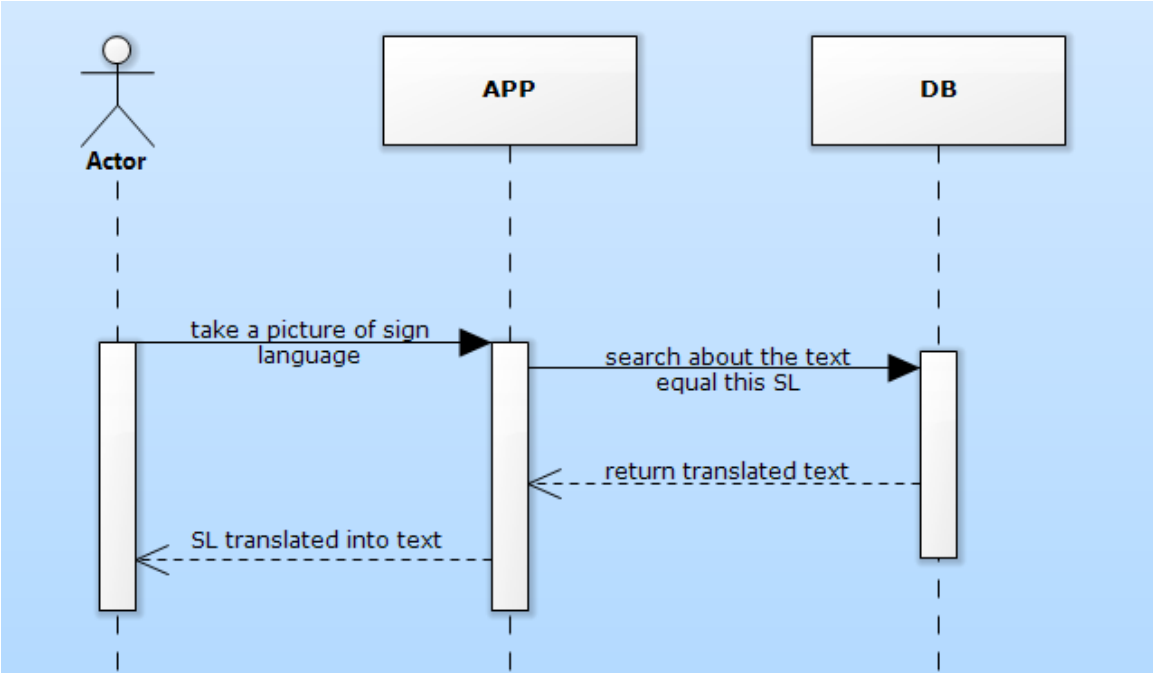


Figure 2.3 User2 Sequence Diagram

The user will capture a picture with sign language and send it to the app, the app will search in DB what this sign means in text type and send the reply to the user.

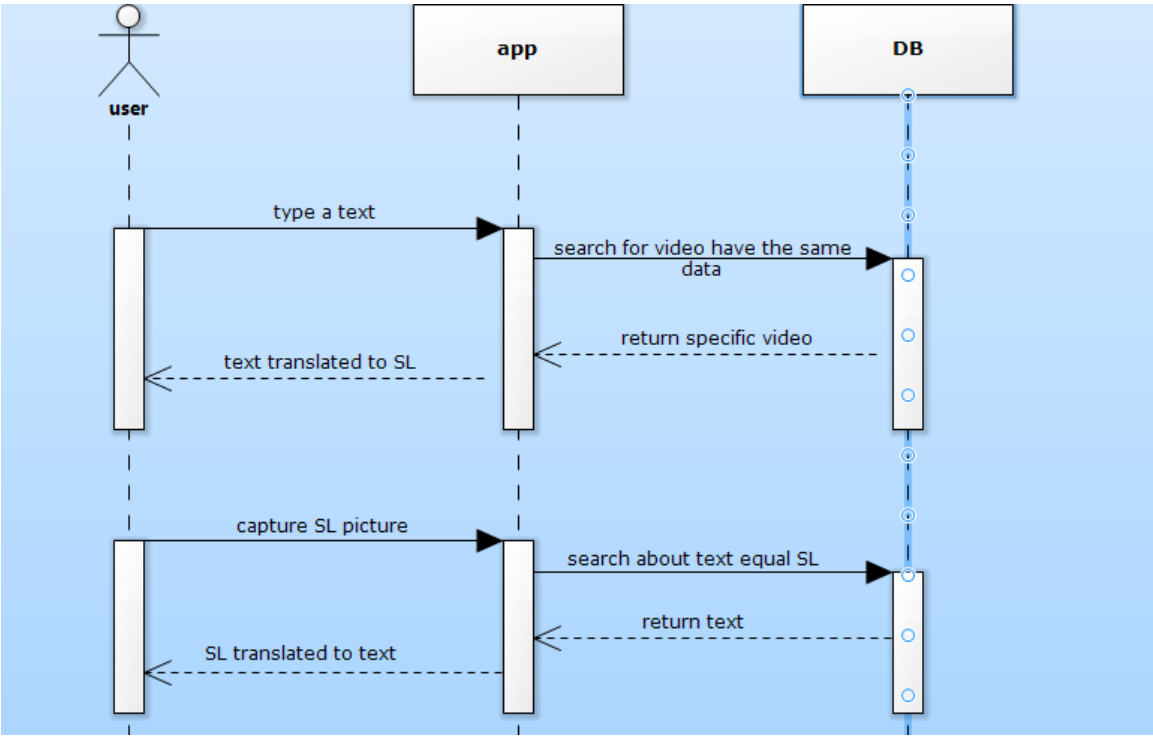


Figure 2.4 Users Sequence Diagram

It shows the interaction between users with the system in details in two scenarios if we want to translate sign to text and vice versa. Shows all of the user and system actions in the sequence diagram.

- 2.9.3 Activity Diagram

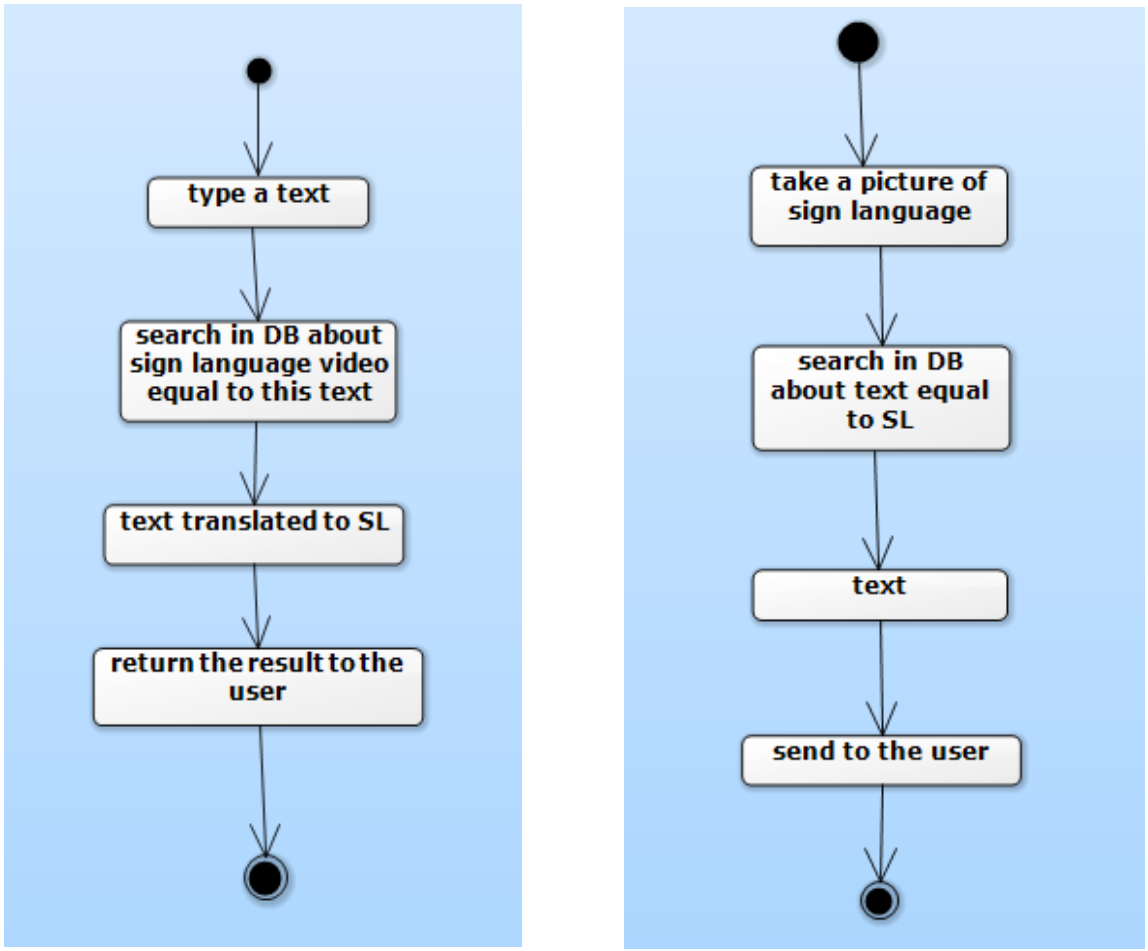


Figure 2.5 Activity Diagram

This diagram describes the interaction between set of objects. It also shows how processes operate with one another and what their order is. Sequence diagram is a behavioral UML diagram.

- 2.9.4 Class Diagram

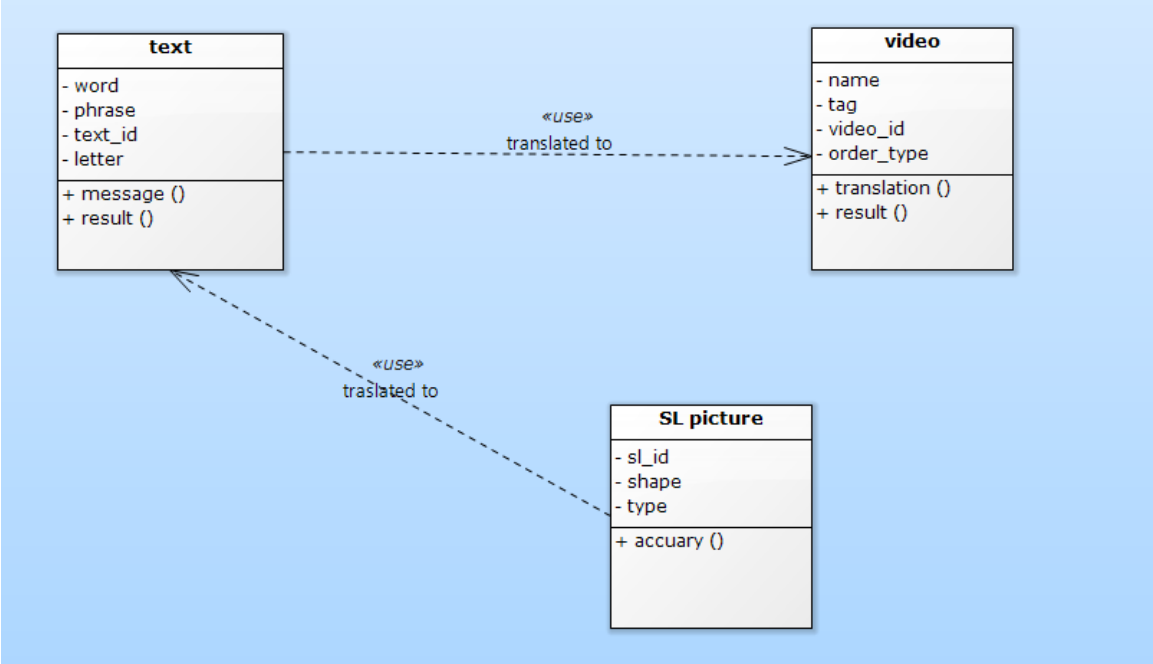


Figure 2.6 Class Diagram

Class diagram is a diagram which describes the project classes and its attributes and operations and the relations between classes in our project.

2.10 index diagrams

We have an unstructured data so we use index program to act this data. The data is pictures and videos. So, when we search for a text to translate it to sign language, we will choose from list of videos in 2 order: (2.10.1 Ascending, 2.10.2 Descending).

When we choose the column of tag (name) and choose the ascending order in (figure 2.7) and the result is in (figure 2.8).

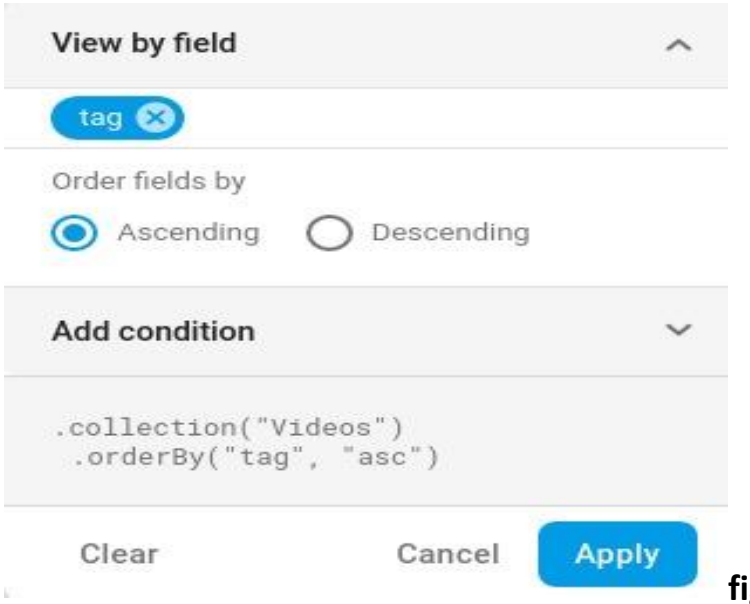


figure 2.7: Ascending order

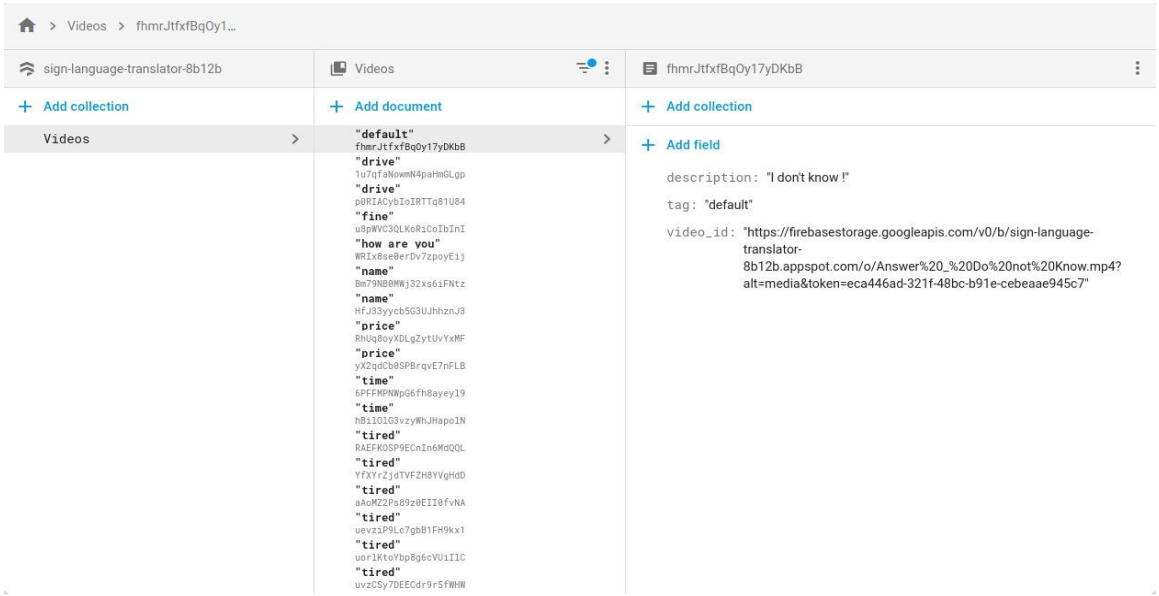


Figure 2.8: Result of Ascending order

When we choose the column of tag (name) and choose the descending order in (figure 2.9) and the result is in (figure 2.10).

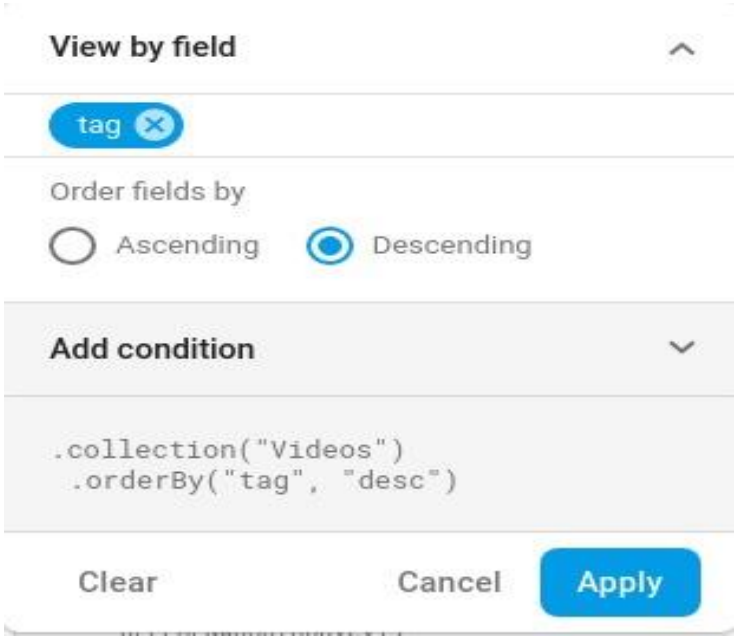


figure 2.9: Descending order

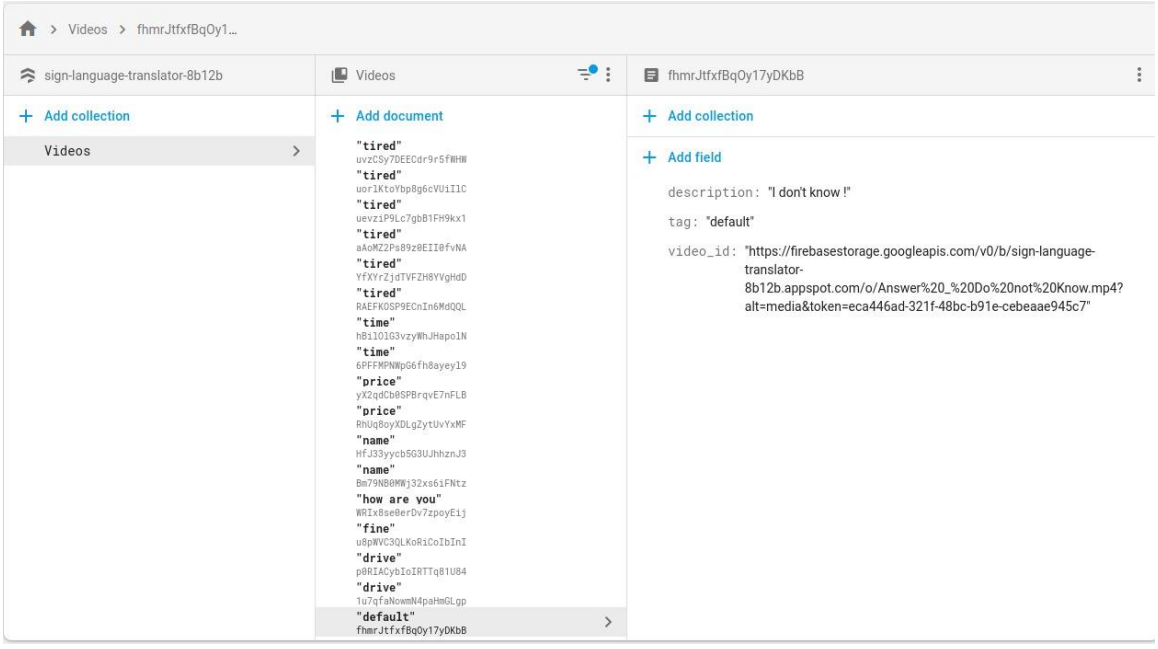


Figure 2.10: Result of Descending order

2.11 cost management plan

1- labtops

2- software

3- team members

4- developer

2.12 Kind of Risks in our project:

1- Cost is too high.
2- Laptops has damaged.
3- Internet connection has lost.

Chapter 3

Project Design

3.1 Database Design

3.1 Introduction

In the Sign Language Translator app, the data is one of the most important components to build the application, it plays vital role in the structure of it. So, it had to be gathered and arranged in an orderly manner, management and effectiveness of running the data is essential to transfer and apply processes on it. So, it is important to build accurate database in order to deal easily and accurately to obtain results. In the Sign Language Translator, we deal with static data when we translate a text to SL as this translation return to the user in video type, we store SL actions in videos. And dynamic data in model when we translate the photo of sign language to text and voice. It is a small translator engine like google translator. After the previous steps of building our project, gathering information and planning, here we are started the design phase and we are going to talk in details about designing our Database and used Database Management System.

3.2 Used Data

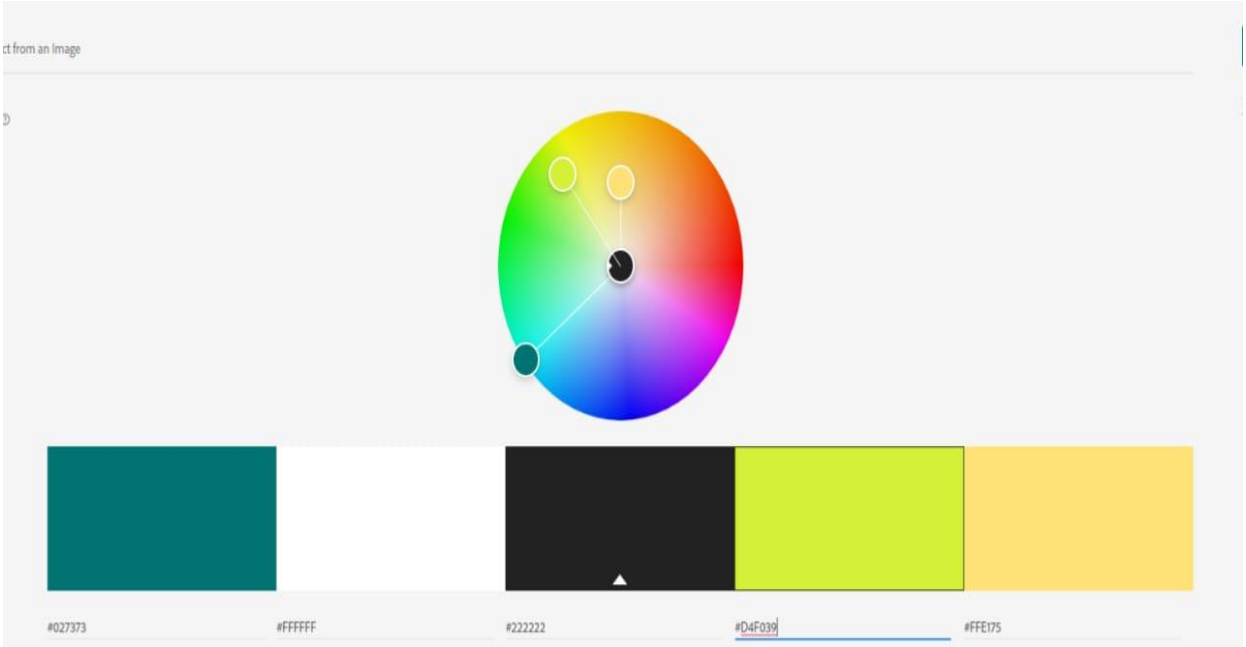
The application mainly based on the data that are stored in our database system:

- Sign language videos which would be displayed to the user from the system when he enters to translate a specific text.

- Text and voice which would be displayed to the user from the system when he captures a photo of sign language to translate it.

3.3 UI/UX Design

Our pallet



We chose this pallet as it’s one of the most suitable pallets for deaf and dump.

3.3.1 Mobile App screens for (T to SL)

All designs are xml files.

- This the first screen which will face the user when he opens the app

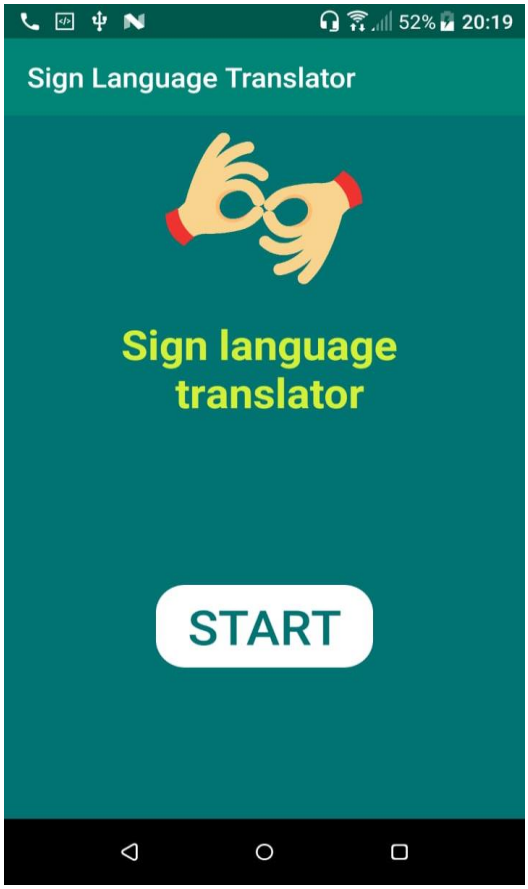


Figure 3.1 start SLT app

When the user clicks on start, he moves to the next screen to start translation.

-Second screen of our app to choose if you want to translate (T to SL) or (SL to T).

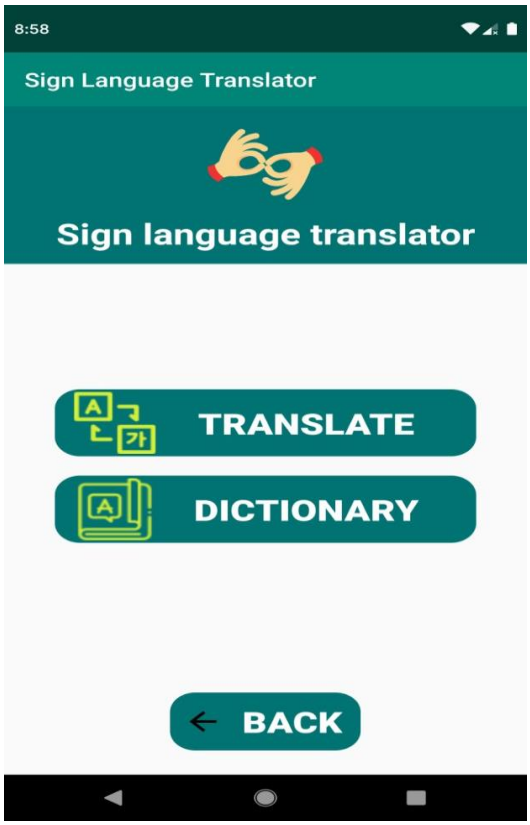


Figure 3.2 choose type of translation

If user chooses dictionary, it means that he chooses to translate text to sign language so user will type the text to translate it to SL video.

But if user chooses translate, it means that he chooses to translate sign language to text so user will capture a picture of sign language to translate it into text.

And if he clicks on back, he will return back to the first screen of start page.

-when user choose dictionary, this screen will appear and give you some examples of texts you may want to convert it to SL like (figure 3.3).

The user starts to type the text like how in the search space and some suggestions will appear like (figure 3.4).

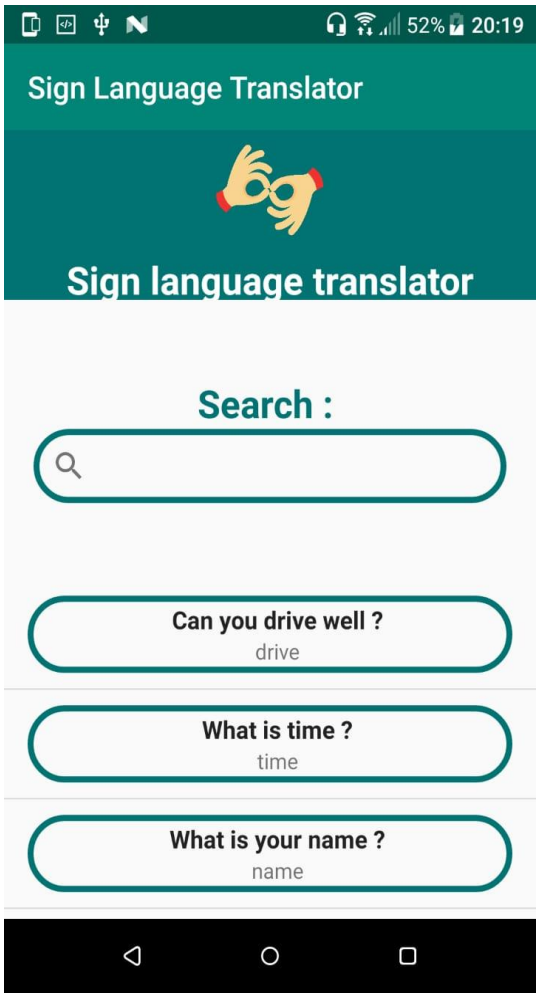


Figure 3.3 Dictionary chosen

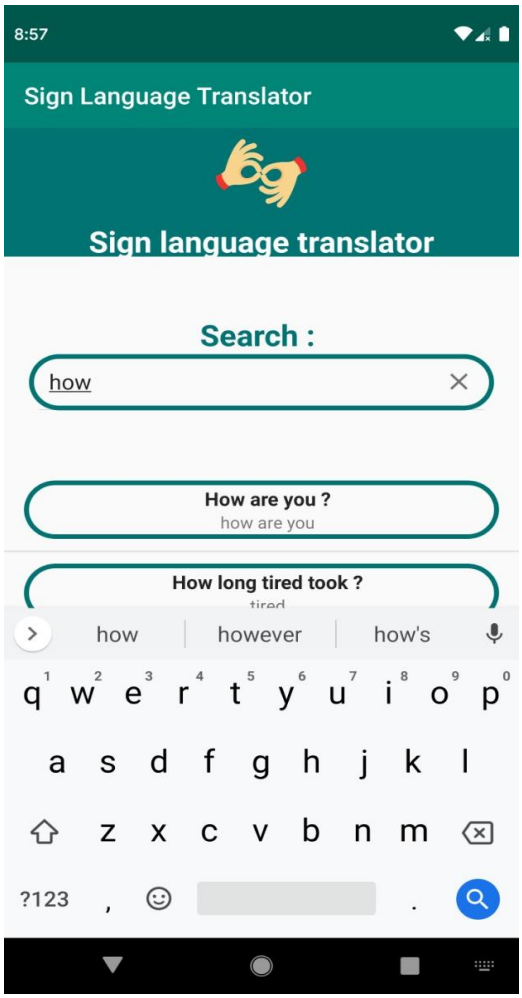


Figure 3.4 example of search

for example, if we choose how are you? , the result will show as a SL video that act the sign of how are you like (figure 3.7).

-screen from the translation video of (how are you?) text and the video will show in the presentation.

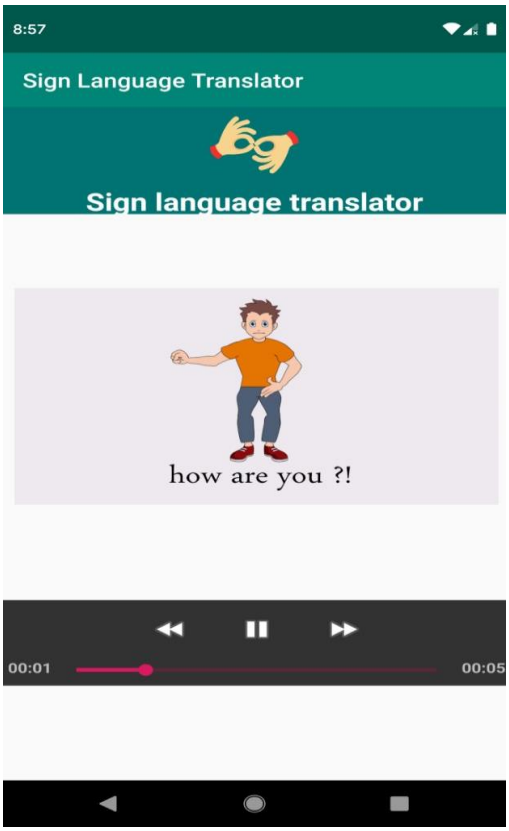


Figure 3.5 sign language video (result)

This is the first type of translation and the result of it.

So, we succeeded in translating Text to Sign Language.

The second type will be vice versa (sign language to text) by using mobile camera in taking a photo of sign language and return the result in text and voice type.

3.3.2 Mobile App screens for (SL to T)

When we choose translate in (figure 3.2), the next screen will be mobile camera to shoot a sign to translate it to text.

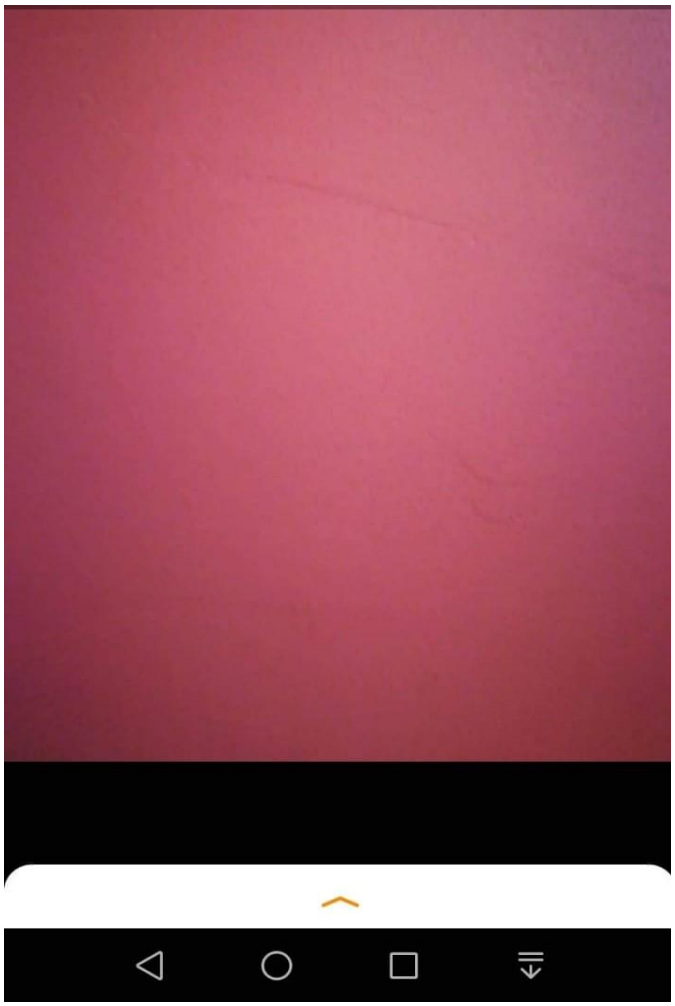


Figure 3.6: Mobile camera opened

This screen is mobile camera to capture a sign photo to translate it to text. The app has an object detection so the cam will detect the sign and translate it with the accuracy without any action from user like the next figures.

For example ein letter(ع) , user will take a photo with sign and the app will translate it to text.

Figure 3.7: Sign translated to letter

The app detected the sign language picture and think that the sign is ع and the accuracy of sign is 94.62% so if the user moves his hands a bit little, the accuracy will differ.

Another example of translation sign of sheen letter (ش) with 96.79% accuracy.

Figure 3.8: High accuracy of translation

The sign is very accurate because it acted right but if we change the angle of picture, the accuracy will be different like the next figure.

Low accuracy of the same example sheen letter (ش) with different angle.

Figure 3.9: Low accuracy of translation

As shown figure, the accuracy is too low because the sign is not accurate like the right sign and if the user moves more, the accuracy will be lower.

All the Arabic letters in sign language shown in this figure and the accuracy will be high if the user acts the sign right and the percentage will lower if he acts wrong or with different angle.

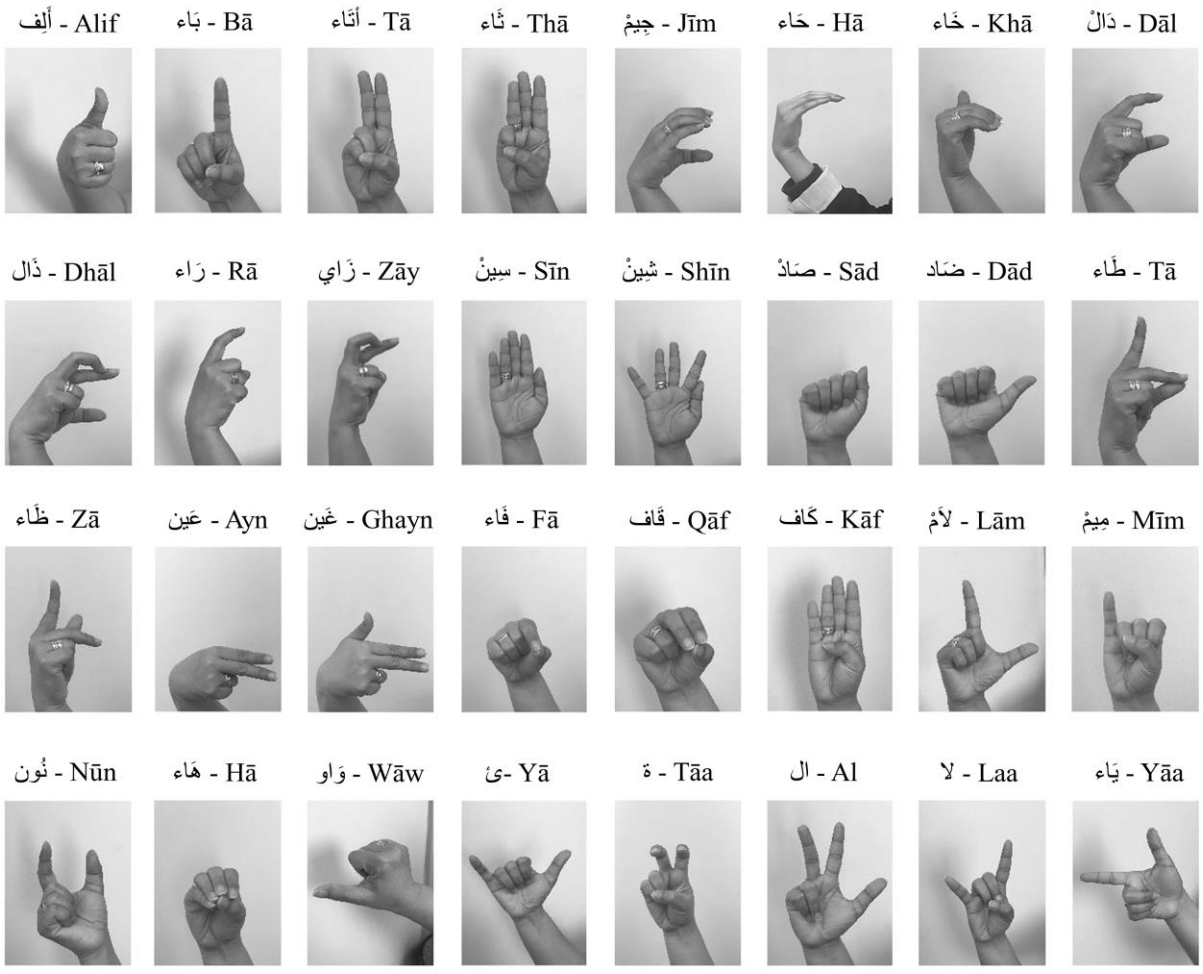


Figure 3.10: Arabic letters in Sign language

Chapter 4

Implementation:

4.1.1 Introduction

In this chapter we discuss the implementation and programming techniques that we
use, we discuss how we convert the system from analysis state into real system that can be used by android users.

4.1.2 Sign Language Translator implementation

SLT system implementation divided into 2 parts, each part has its own steps:

4.1.2.1 Implementation of model:

1. Create dataset.
2. Choosing best algorithm to build the model.
3. Implementation of model.
    a. Label images.
    b. Preprocessing for input to model.
4. Testing the model.
5. Applying model.

4.1.2.2 Implementation the rest of the system:

1. Implement android.
2. Firebase integration

4.1.2.1.1 Create dataset:

We search for existing dataset but we not find so we create our own dataset.

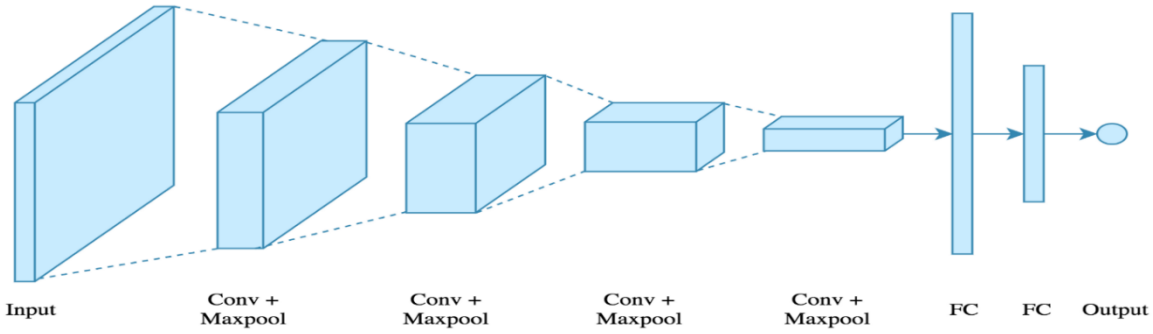
Dataset contains 28 class (Characters from ي - أ) each class has 200 images.

Figure 4.1: sample classes we use

4.1.2.1.2 Choosing best algorithm to build the model:

We trying more than one choice to build our model all of them gives us a good accuracy but our problem is we need one that can work fast and give good results on mobile phone, for this reason we decide to use (TensorFlow Object Detection API).

1. Convolution neural network:



Input is an images of pixel size—32×32.Let's understand the architecture of the model. The model contained 7 layers excluding the input layer, let's go layer by layer:

1. Layer 1: A convolutional layer with kernel size of 5×5, stride of 1×1 and 6 kernels in total. So, the input image of size 32x32x1 gives an output of 28x28x6. Total params in layer = 5 * 5 * 6 + 6 (bias terms)
2. Layer 2: A pooling layer with 2×2 kernel size, stride of 2×2 and 6 kernels in total. This pooling layer acted a little differently than what we discussed in previous post. The input values in the receptive were summed up and then were multiplied to a trainable parameter (1 per filter), the result was finally added to a trainable bias (1 per filter). Finally, sigmoid activation was applied to the output. So, the input from previous layer of size 28x28x6 gets sub-sampled to 14x14x6. Total params in layer = [1 (trainable parameter) + 1 (trainable bias)] * 6 = 12
3. Layer 3: Similar to Layer 1, this layer is a convolutional layer with same configuration except it has 16 filters instead of 6. So, the input from previous layer of size 14x14x6 gives an output of 10x10x16. Total params in layer = 5 * 5 * 16 + 16 = 416.

4. Layer 4: Again, similar to Layer 2, this layer is a pooling layer with 16 filters this time around. Remember, the outputs are passed through sigmoid activation function. The input of size 10x10x16 from previous layer gets sub-sampled to 5x5x16. Total params in layer = (1 + 1) * 16 = 32
5. Layer 5: This time around we have a convolutional layer with 5×5 kernel size and 120 filters. There is no need to even consider strides as the input size is 5x5x16 so we will get an output of 1x1x120. Total params in layer = 5 * 5 * 120 = 3000
6. Layer 6: This is a dense layer with 84 parameters. So, the input of 120 units is converted to 84 units. Total params = 84 * 120 + 84 = 10164. The activation function used here was rather a unique one. I'll say you can just try out any of your choice here as the task is pretty simple one by today's standards.
7. Output Layer: Finally, a dense layer with 10 units is used. Total params = 84 * 10 + 10 = 924.

This technique give accuracy .97 this model overfitting on training data so we cannot use it for this reason and reason of mobile.

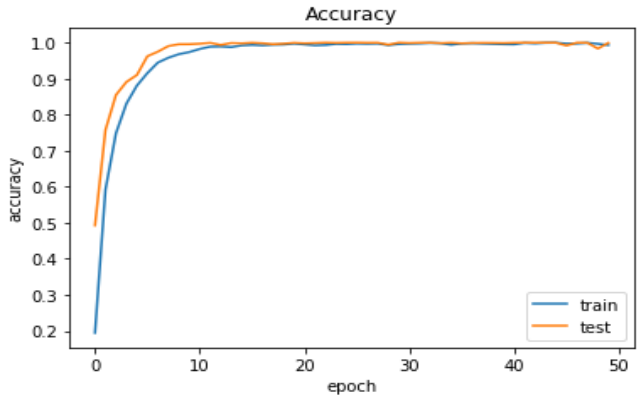


figure 4.2: Accuracy of CNN Model

Tensor Flow Object Detection API:

This is the best choice for a lot of reasons:

1- We can use output graph and convert it to tensor flow lite which it can used in phones.

2- Solve the problem of overfitting.

3- Dealing will if any noise in input data.

4.1.2.1.3 Implement of the model:

A) Labeling input images:

The operation of Label Image is to assign each image to class and this operation done for all 28 classes and all images in training and testing sets.

We use a tool called LabelImg for this operation:

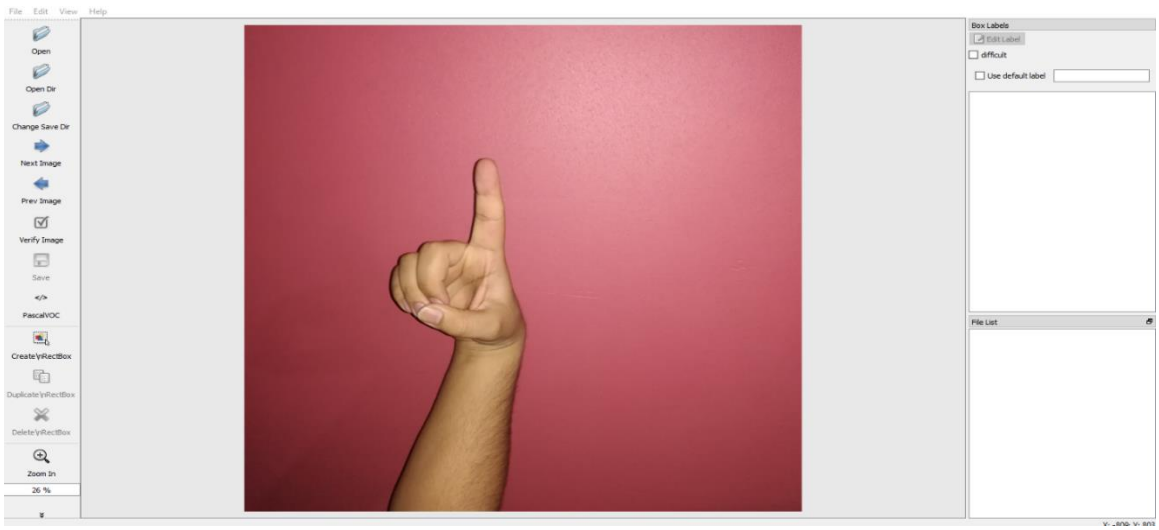


Figure 4.3: LabelImg Screen

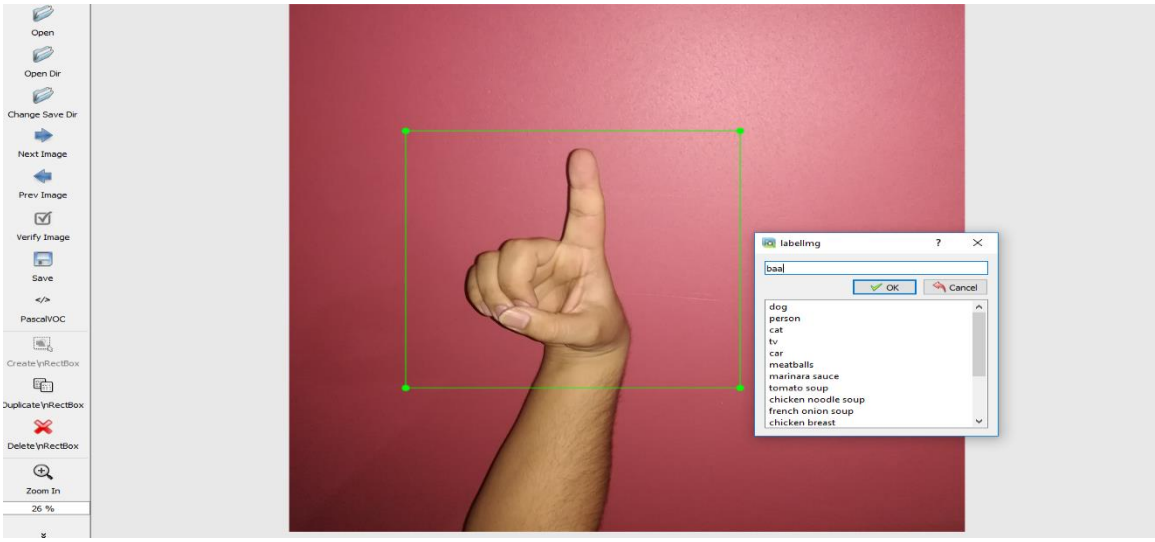


Figure 4.4: Operation of label character baa

B) preprocessing for inputs to model:

In this step a lot of steps done:

1- Installing TensorFlow-GPU

```
# For CPU
pip install tensorflow
# For GPU
pip install tensorflow-gpu
```

The remaining libraries can be installed on Ubuntu 16.04 using via apt-get:

```
sudo apt-get install protobuf-compiler python-pil python-lxml python-tk
pip install --user Cython
pip install --user contextlib2
pip install --user jupyter
pip install --user matplotlib
```

Alternatively, users can install dependencies using pip:

```
pip install --user Cython
pip install --user contextlib2
pip install --user pillow
pip install --user lxml
pip install --user jupyter
pip install --user matplotlib
```

Figure 4.5: TensorFlow install

2 - Setting up the Object Detection directory structure and Anaconda Virtual Environment:

A – configure python variable:

```
flow1) C:\> set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim
```

B - Compile Protobufs and run setup.py

```
(base) C:\Users\Dev>protoc --python_out=. .\object_detection\protos\anchor_generator.proto .\object_detection\protos\argmax_matcher.proto .\object_detection\protos\bipartite_matcher.proto .\object_
detection\protos\box_coder.proto .\object_detection\protos\box_predictor.proto .\object_detection\protos\eval.proto .\object_detection\protos\faster_rcnn.proto .\object_detection\protos\faster_rcnn
_box_coder.proto .\object_detection\protos\grid_anchor_generator.proto .\object_detection\protos\hyperparams.proto .\object_detection\protos\image_resizer.proto .\object_detection\protos\input_read
er.proto .\object_detection\protos\losses.proto .\object_detection\protos\matcher.proto .\object_detection\protos\mean_stddev_box_coder.proto .\object_detection\protos\model.proto .\object_detectio
n\protos\optimizer.proto .\object_detection\protos\pipeline.proto .\object_detection\protos\post_processing.proto .\object_detection\protos\preprocessor.proto .\object_detection\protos\region_simil
arity_calculator.proto .\object_detection\protos\square_box_coder.proto .\object_detection\protos\ssd.proto .\object_detection\protos\ssd_anchor_generator.proto .\object_detection\protos\string_int
_label_map.proto .\object_detection\protos\train.proto .\object_detection\protos\keypoint_box_coder.proto .\object_detection\protos\multiscale_anchor_generator.proto .\object_detection\protos\graph
_rewriter.proto
```

3- Generate Training Data:

With the images labeled, it's time to generate the TFRecords that serve as input data to the TensorFlow training model. This tutorial uses the xml_to_csv.py and generate_tfrecord.py scripts from [Dat Tran's Raccoon Detector dataset], with some slight modifications to work with our directory structure.

First, the image .xml data will be used to create .csv files containing all the data for the train and test images. From the \object_detection folder, issue the following command in the Anaconda command prompt:

```
(tensorflow1) C:\tensorflow1\models\research\object_detection> python xml_to_csv.py
```

This creates a train_labels.csv and test_labels.csv file in the \object_detection\images folder. Next, open the generate_tfrecord.py file in a text editor. Replace the label map starting at line 31 with your own label map, where each object is assigned an ID number. This same number assignment will be used when configuring the labelmap.pbtxt file.

```
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'aleff':
        return 1
    elif row_label == 'ba':
        return 2
    elif row_label == 'ta':
        return 3
    elif row_label == 'tha':
        return 4
    elif row_label == 'geem':
        return 5
```

For example, say you are training a classifier to detect ASL Characters. You will do the following code in generate_tfrecord.py:

```
python generate_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\train --output_path=train.record
python generate_tfrecord.py --csv_input=images\test_labels.csv --image_dir=images\test --output_path=test.record
```

4- Create Label Map:

The label map tells the trainer what each object is by defining a mapping of class names to class ID numbers. Use a text editor to create a new file and save it as labelmap.pbtxt in the C:\tensorflow1\models\research\object_detection\training folder.

```
item {
  name: "aleff"
  id: 1
  display_name: "aleff"
}
item {
  name: "bb"
  id: 2
  display_name: "bb"
}
item {
  name: "taa"
  id: 3
  display_name: "taa"
}
item {
  name: "thaa"
  id: 4
  display_name: "thaa"
}
```

C) Training the Model:

Finally, the object detection training pipeline must be configured. It defines which model and what parameters will be used for training. This is the last step before running training!

Navigate to C:\tensorflow\models\research\object_detection\samples\configs and copy the ssdlite_mobilenet_v2.config file into the \object_detection\training directory. Then, open the file with a text editor. There are several changes to make to the .config file, mainly changing the number of classes and examples, and adding the file paths to the training data.

Make the following changes to the ssdlite_mobilenet_v2.config file. Note: The paths must be entered with single forward slashes (NOT backslashes), or TensorFlow will give a file path error when trying to train the model! Also, the paths must be in double quotation marks ("), not single quotation marks (').

- Line 9. Change num_classes to the number of different objects you want the classifier to detect.
- Line 110. Change fine_tune_checkpoint to: "C:/tensorflow/models/research/object_detection/ssdlite_mobilenet_v2_coco_2018_05_09/model.ckpt"
- Lines 126 and 128. In the train_input_reader section, change input_path and label_map_path to:
    - input_path: "C:/tensorflow/models/research/object_detection/train.record"
    - label_map_path: "C:/tensorflow/models/research/object_detection/training/labelmap.pbtxt"
- Line 132. Change num_examples to the number of images you have in the \images\test directory.
- Lines 140 and 142. In the eval_input_reader section, change input_path and label_map_path to:
    - input_path: "C:/tensorflow/models/research/object_detection/test.record"

- label_map_path: "C:/tensorflow/models/research/object_detection/training/labelmap.pbtxt"

```
(base) C:\tensorflow\models\research\object_detection>python train.py --logtostderr --train_dir=training/
                        --pipeline_config_path=training/ssdlite_mobilenet_v2_coco_2018_05_09.config
```

4.1.2.1.4 Model Accuracy:

We can see our result in tensor board Graphs:

Figure 4.6: Actual loss for Model

4.1.2.1.5 Applying model:

To Apply model on android we need first to get frozen_graph.py and convert to tensor flow lite and file.tflite is the file we will use on android application.

We can do that as number of steps:

1- Export Inference Graph

Now that training is complete, the last step is to generate the frozen inference graph (.pb file). From the \object_detection folder, issue the following command, where "XXXX" in "model.ckpt-XXXX" should be replaced with the highest-numbered .ckpt file in the training folder:

```
base) C:\tensorflow\models\research\object_detection>python export_inference_graph.py --input_type image_tensor
                --pipeline_config_path training/ssdlite_mobilenet_v2_coco.config
              --trained_checkpoint_prefix training/model.ckpt-XXXX --output_directory inference_graph
```

2- Convert Frozen_graph.py to file.tflite:

```
import tensorflow as tf

graph_def_file = "C:/tensorflow/models/research/object_detection/inference_graph/frozen_inference_graph.pb"
input_arrays = ["image_tensor"]
output_arrays = ["ToFloat"]
input_shapes=[1,300,300,3]
converter = tf.lite.TFLiteConverter.from_frozen_graph(
  graph_def_file, input_arrays, output_arrays)
tflite_model = converter.convert()
open("converted_model.tflite", "wb").write(tflite_model)
```

4.1.2.2 Implementation the rest of the system

Here we will talk about the 2nd half of the project or the user interface of the project that make the result of the model displayed for the user, also methods that connect Fire base and DL Model to android.

4.1.2.2.1 List all methods in the system

i. **Design.**
ii. **Translate.**
iii. **Dictionary.**

4.1.2.2.2 Building system design

In order to make simple and interactive design with the users we used android because the popularity of this OS and now a days most of people have a smart phone and the higher percentage of them use phones run on android system, we used XML to design this application.

What Is XML means?

- XML stands for extensible Markup Language.
- XML is a markup language much like HTML.
- XML was designed to store and transport data.
- XML was designed to be self-descriptive.

Design File Using XML:

1- activity_main.xml
2- activity_home.xml
3- activity_dictionary.xml
    3.1- dictionary_item.xml
4- activity_video_display.xml
5- activity_camera.xml
    5.1- camera_connection_fragment.xml
    5.2- camera_connection_fragment_tracking.xml

I – activity_main

is the Start Screen (Welcome Screen) of our application
that contains the app logo and start button to start processes.

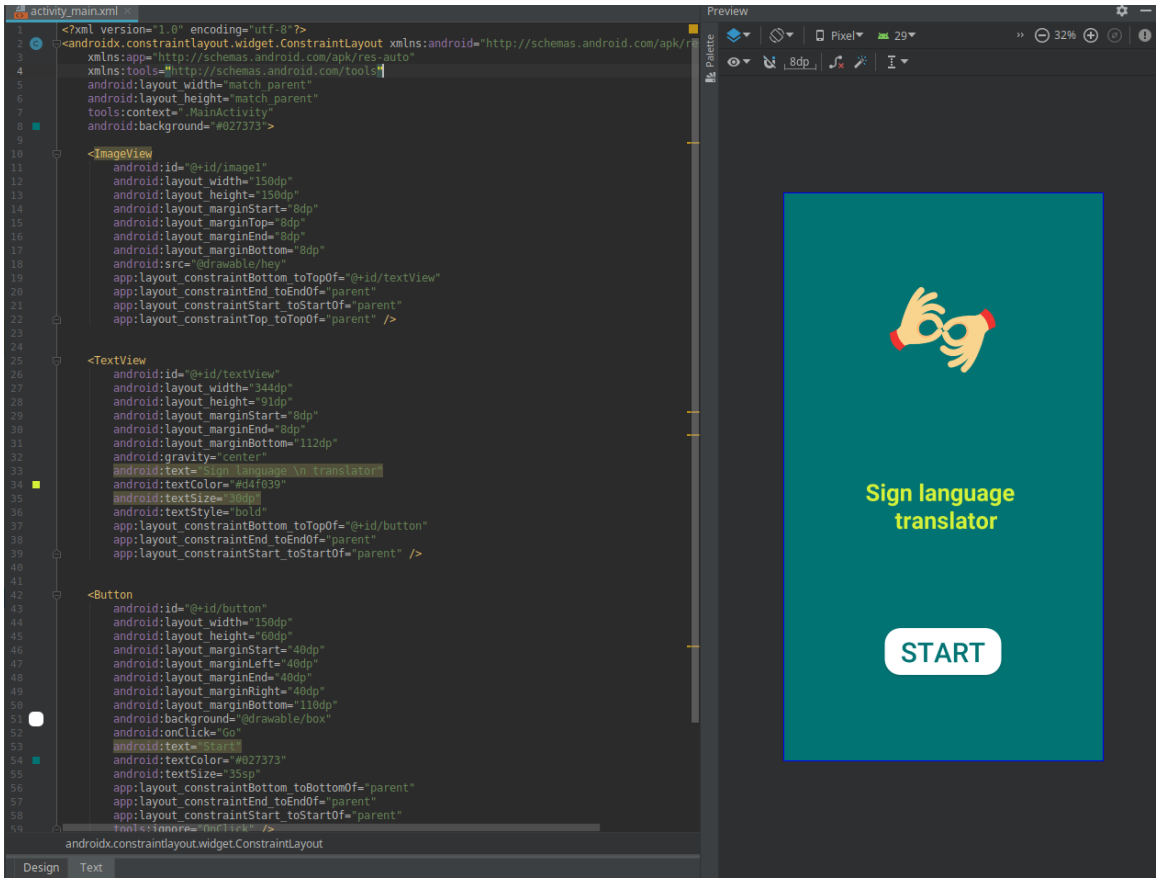


Figure 4.7: Activity_main

II – activity_home

the 2nd Screen (The main Screen) that contains buttons for processes we have:

1- Translate button that open the custom camera to detect the object (detect the sign and determine the character).

2- Dictionary button that open the list of videos and for the statements.

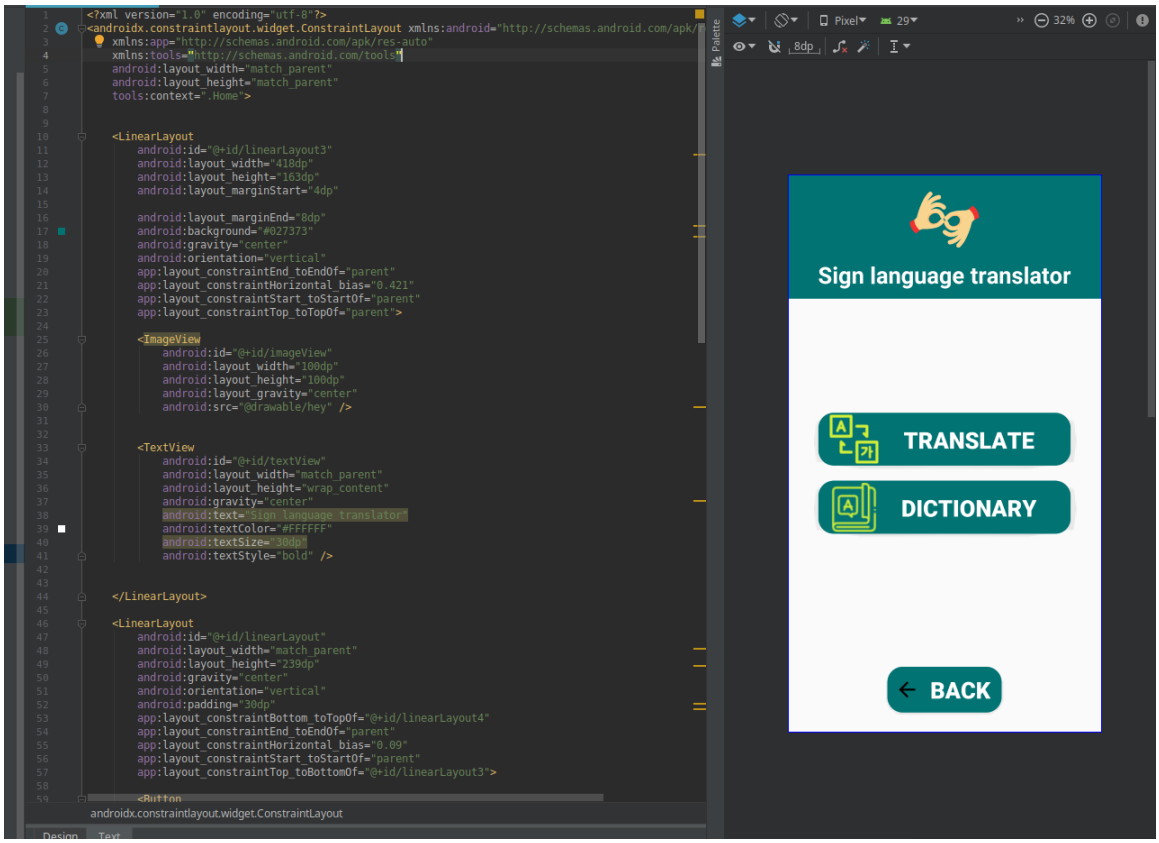


Figure 4.8: Activity_home

III – activity_dictionary

Screen for Dictionary button that contains Recycler View that is a place holder for videos and Search bar with filter to filter data while searching.

III.I – Dictionary_item

Describe the element of Recycler View.

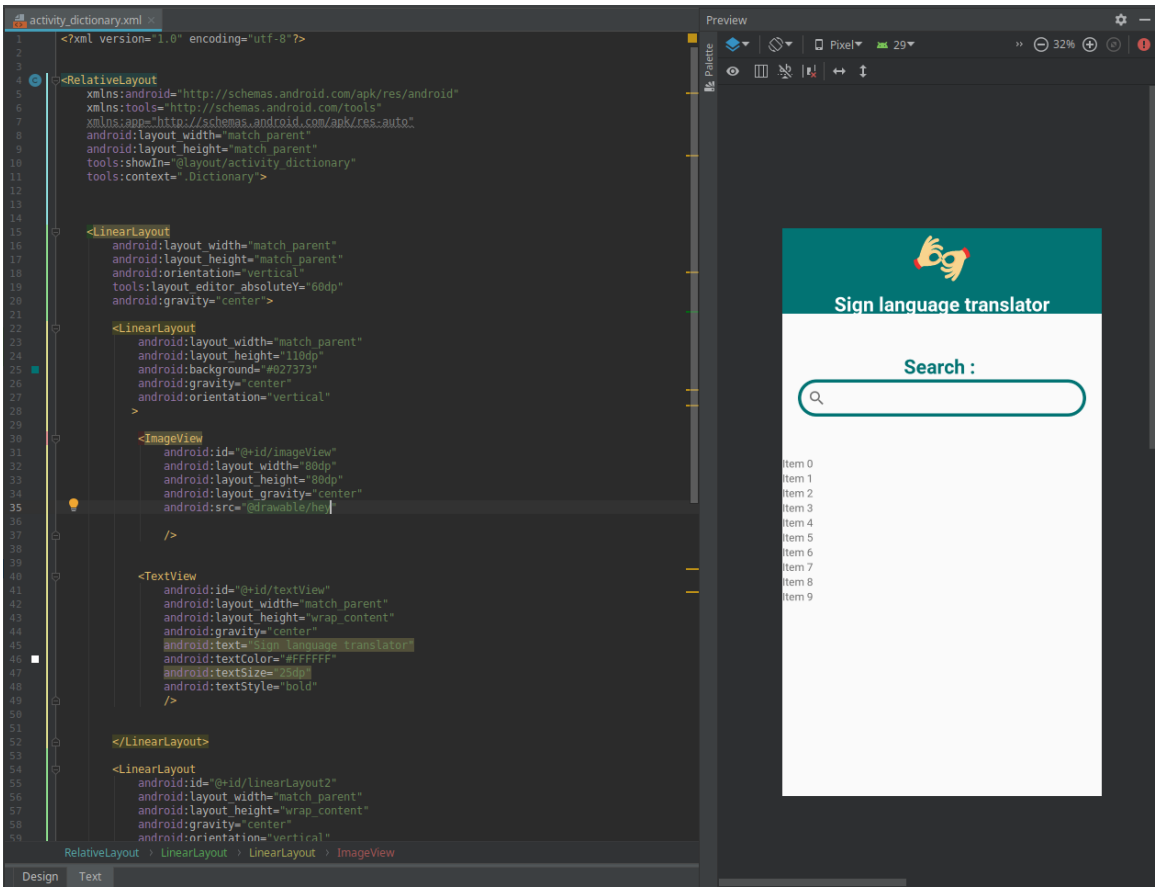
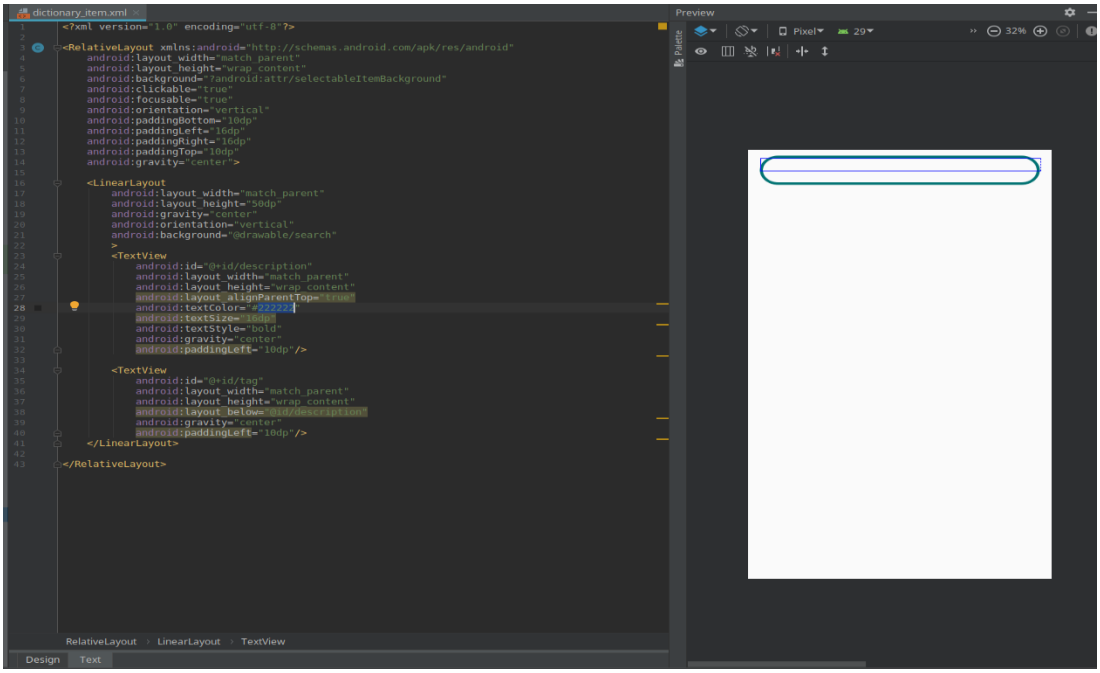


Figure 4.9: Dictionary_item

IV – activity_video_display

Screen that the video play in it after the user choose of the list it contains Video View that is a place holder for video, Progress bar that inform the user that the video is loading also Media controller to control the video.



controller to control the video.

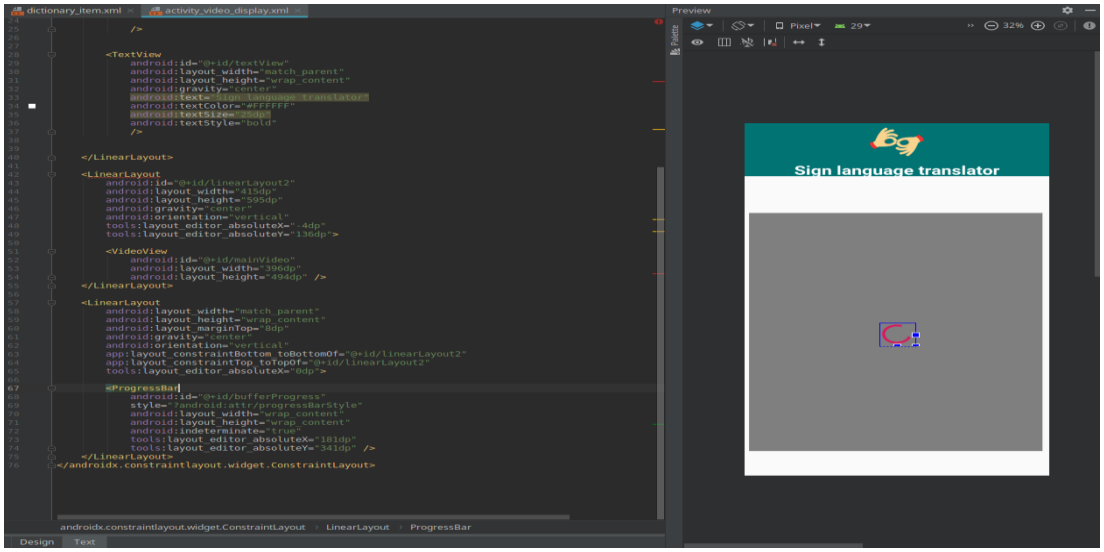


Figure 4.10: Activity video display

V- activity_camera

a Custom camera for object detection.

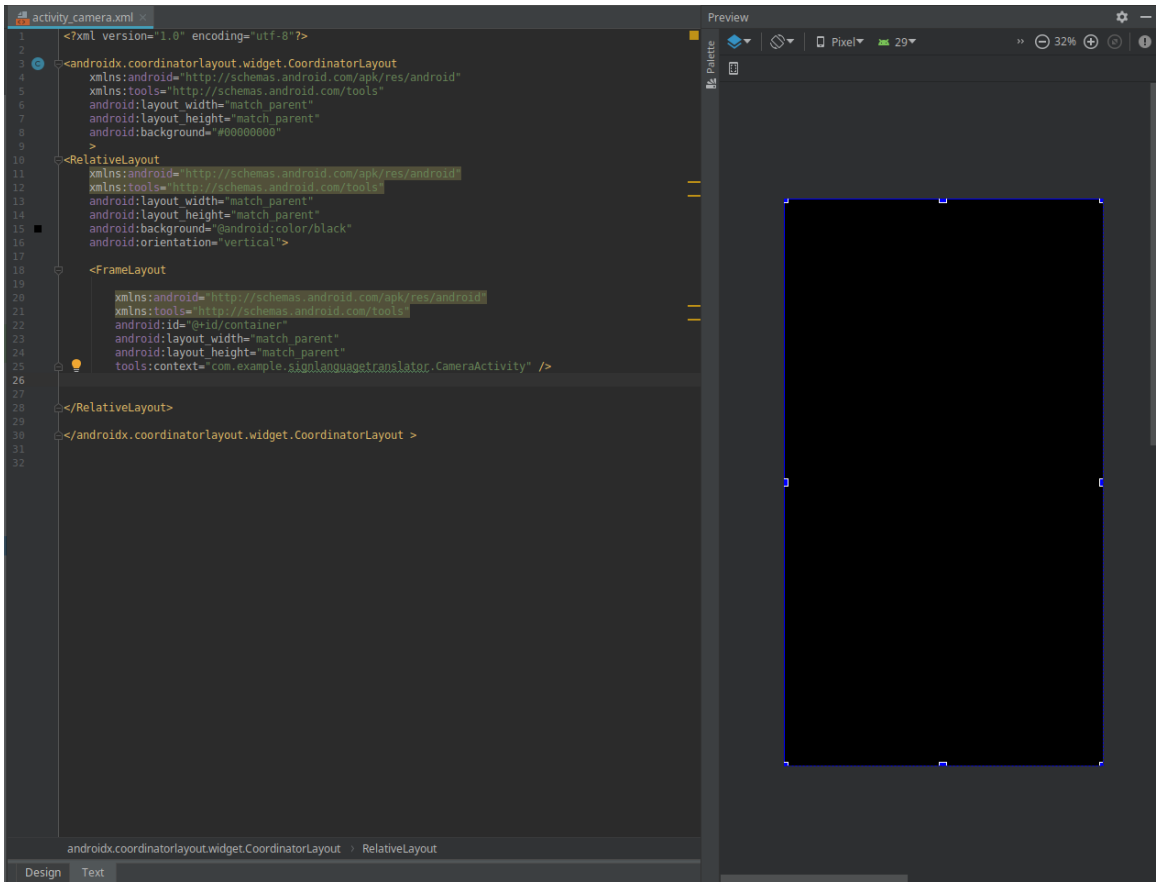
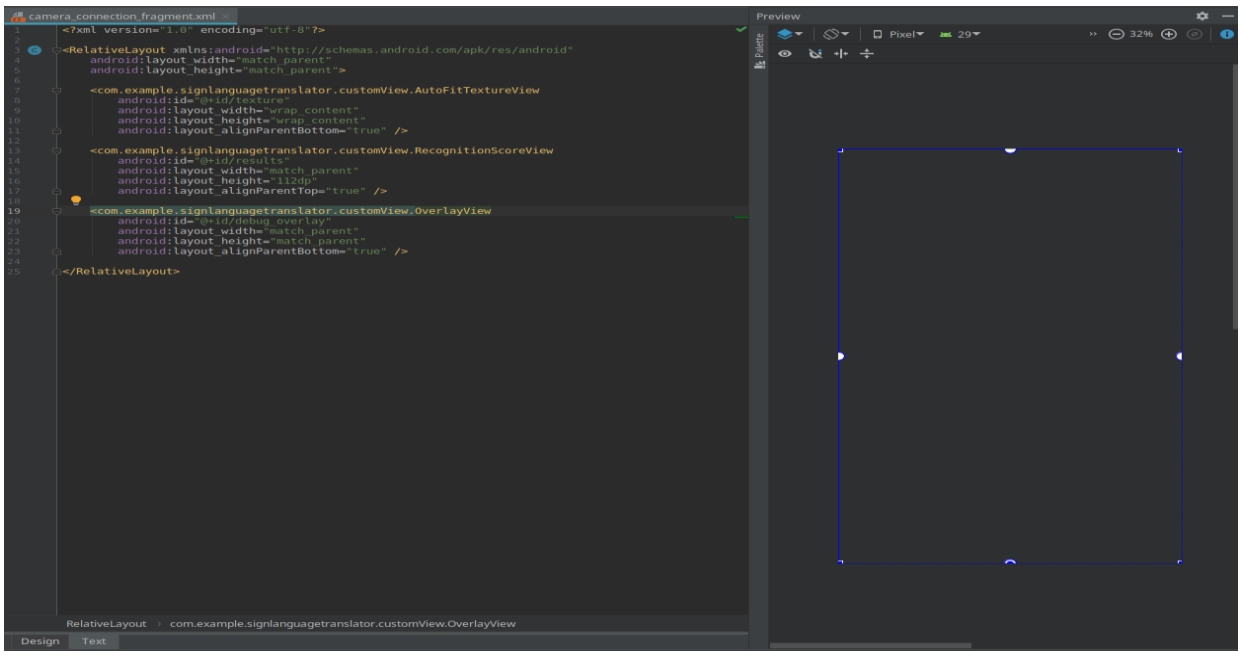


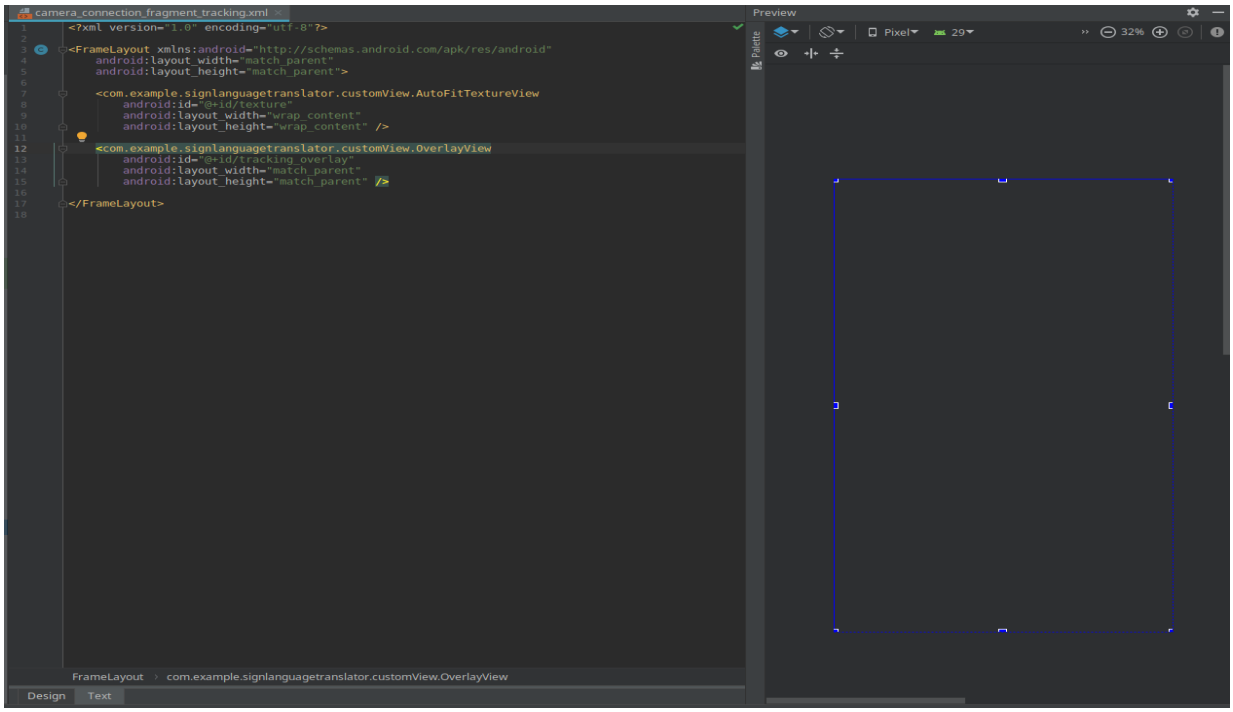
figure 4.11: Activity_camera

V.I - camera_connection_fragment

The Fragment of screen of the Custom Camera.



V.II – camera_connection_fragment_tracking
The frame for detection.



4.1.2.2.3 Connect Model to Android (Translate Process)

In order to make a simple process the function of this sector is translate the sign to the word or character and show the similarity percentage.

We used TensorFlow Lite API to connect The Model to android so we can detect the object
but first converted the model to predect.tflite so we could work with the API, and used a text file to save labels in it. And created an assets Directory in android to save these 2 files in it

To connect it we implement: 4 Packages, 4classes

I - Packages:

1- customView: that contains Classes / AutoFitTextureView, overlayView, RecognitionScoreView, (InterFace) ResultView.

2- env: that contains Classes / BorderedText, ImageUtils,

(Final) Loger, Size.

3- tfLite: that contains Classes / (Interface)Classifier, TFLiteObjectDetectionAPIModel.

4- tracking: that contains Classes / MultiBoxingTracker.

II - Classes: CameraActivity, DetectorActivity,
LegacyCameraConnectionFragment, CameraConnectionFragment

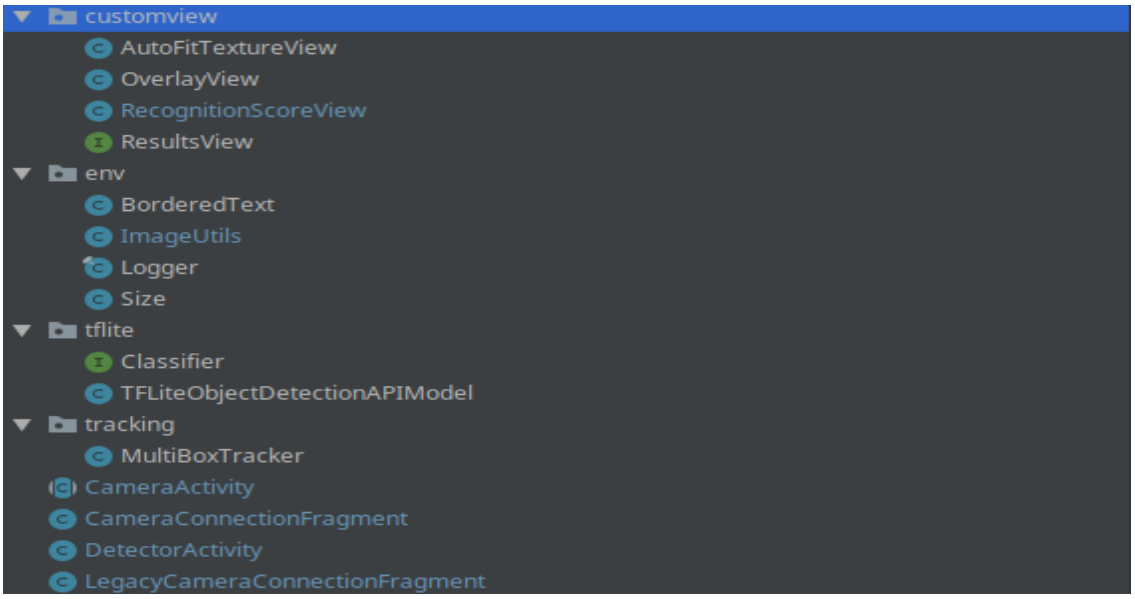


Figure 4.12: Packages

I - customView Package:

I.I - AutoFitTextureView

Sets the aspect ratio for this view. The size of the view will be measured based on the ratio calculated from the parameters.

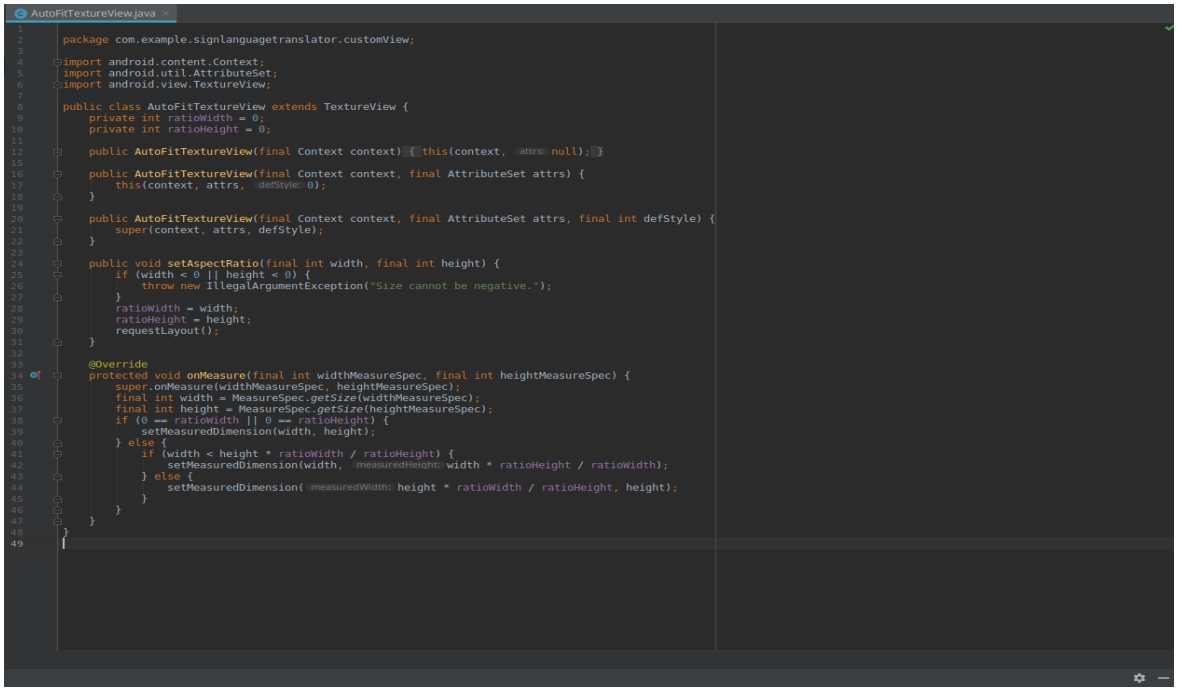


Figure 4.13: Texture view

I.II - OverlyView

A simple View providing a render callback to other classes.

OverlayView.java

```java
package com.example.signlanguagetranslator.customView;

import android.content.Context;
import android.graphics.Canvas;
import android.util.AttributeSet;
import android.view.View;
import java.util.LinkedList;
import java.util.List;

/** A simple View providing a render callback to other classes. */
public class OverlayView extends View {
    private final List<DrawCallback> callbacks = new LinkedList<>();

    public OverlayView(final Context context, final AttributeSet attrs) {
        super(context, attrs);
    }

    public void addCallback(final DrawCallback callback) { callbacks.add(callback); }

    @Override
    public synchronized void draw(final Canvas canvas) {
        for (final DrawCallback callback : callbacks) {
            callback.drawCallback(canvas);
        }
    }

    /** Interface defining the callback for client classes. */
    public interface DrawCallback {
        public void drawCallback(final Canvas canvas);
    }
}
```

Figure 4.14: Overly view

I.III - RecognitionScoreView

Calculate the Score of Recognition Object.

OverlayView.java | RecognitionScoreView.java

```java
package com.example.signlanguagetranslator.customView;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.util.TypedValue;
import android.view.View;

import com.example.signlanguagetranslator.tfLite.Classifier.Recognition;

import java.util.List;

public class RecognitionScoreView extends View implements ResultsView {
    private static final float TEXT_SIZE_DIP = 14;
    private final float textSizePx;
    private final Paint fgPaint;
    private final Paint bgPaint;
    private List<Recognition> results;

    public RecognitionScoreView(final Context context, final AttributeSet set) {
        super(context, set);

        textSizePx =
                TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP, getResources().getDisplayMetrics());
        fgPaint = new Paint();
        fgPaint.setTextSize(textSizePx);

        bgPaint = new Paint();
        bgPaint.setColor(0xcc4285f4);
    }

    @Override
    public void setResults(final List<Recognition> results) {
        this.results = results;
        postInvalidate();
    }

    @Override
    public void onDraw(final Canvas canvas) {
        final int x = 10;
        int y = (int) (fgPaint.getTextSize() * 1.5f);

        canvas.drawPaint(bgPaint);

        if (results != null) {
            for (final Recognition recog : results) {
                canvas.drawText( text: recog.getTitle() + ": " + recog.getConfidence(), x, y, fgPaint);
                y += (int) (fgPaint.getTextSize() * 1.5f);
            }
        }
    }
}
```

Figure 4.15: Recognition Score View

II - ENV package:

I - BorderedText

A class that encapsulates the tedious bits of rendering legible, bordered text onto a canvas.

BorderedText.java

```
public class BorderedText {
    private final Paint interiorPaint;
    private final Paint exteriorPaint;

    private final float textSize;

    public BorderedText(final float textSize) { this(Color.WHITE, Color.BLACK, textSize); }


    public BorderedText(final int interiorColor, final int exteriorColor, final float textSize) {
        interiorPaint = new Paint();
        interiorPaint.setTextSize(textSize);
        interiorPaint.setColor(interiorColor);
        interiorPaint.setStyle(Style.FILL);
        interiorPaint.setAntiAlias(false);
        interiorPaint.setAlpha(255);

        exteriorPaint = new Paint();
        exteriorPaint.setTextSize(textSize);
        exteriorPaint.setColor(exteriorColor);
        exteriorPaint.setStyle(Style.FILL_AND_STROKE);
        exteriorPaint.setStrokeWidth(textSize / 8);
        exteriorPaint.setAntiAlias(false);
        exteriorPaint.setAlpha(255);

        this.textSize = textSize;
    }

    public void setTypeface(Typeface typeface) {
        interiorPaint.setTypeface(typeface);
        exteriorPaint.setTypeface(typeface);
    }

    public void drawText(final Canvas canvas, final float posX, final float posY, final String text) {
        canvas.drawText(text, posX, posY, exteriorPaint);
        canvas.drawText(text, posX, posY, interiorPaint);
    }

    public void drawText(
            final Canvas canvas, final float posX, final float posY, final String text, Paint bgPaint) {

        float width = exteriorPaint.measureText(text);
        float textSize = exteriorPaint.getTextSize();
        Paint paint = new Paint(bgPaint);
        paint.setStyle(Paint.Style.FILL);
        paint.setAlpha(160);
        canvas.drawRect(posX, (posY + (int) (textSize)), (posX + (int) (width)), posY, paint);

        canvas.drawText(text, posX, (posY + textSize), interiorPaint);
    }

    public void drawLines(Canvas canvas, final float posX, final float posY, Vector<String> lines) {
        int lineNum = 0;
        for (final String line : lines) {
            drawText(canvas, posX,  posY: posY - getTextSize() * (lines.size() - lineNum - 1), line);
            ++lineNum;
        }
    }


    public float getTextSize() {
```

BorderedText

Figure 4.16: Bordered text

II -ImageUtils

A class for manipulating images. And convert YUV to RGB, returns a transformation matrix from one reference frame into another. Handles cropping (if maintaining aspect ratio is desired) and rotation.

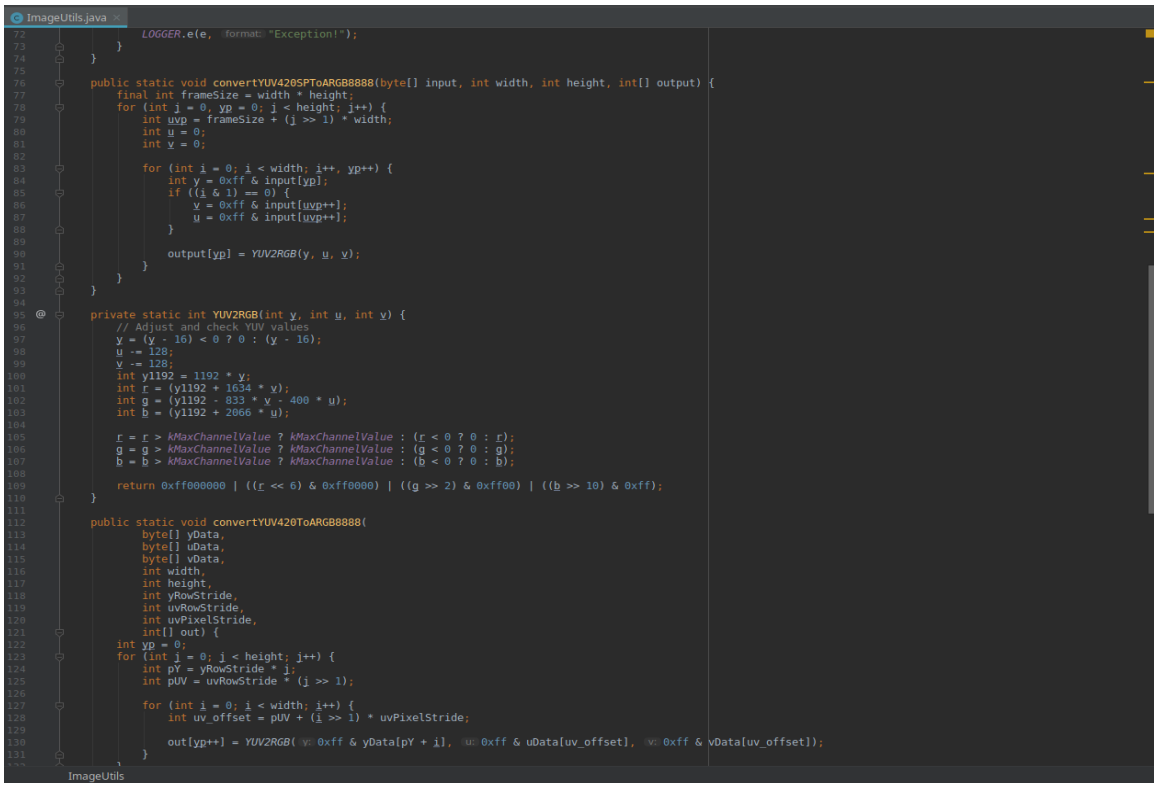


Figure 4.17: Image utils

III - Size

A class independent for a camera object.

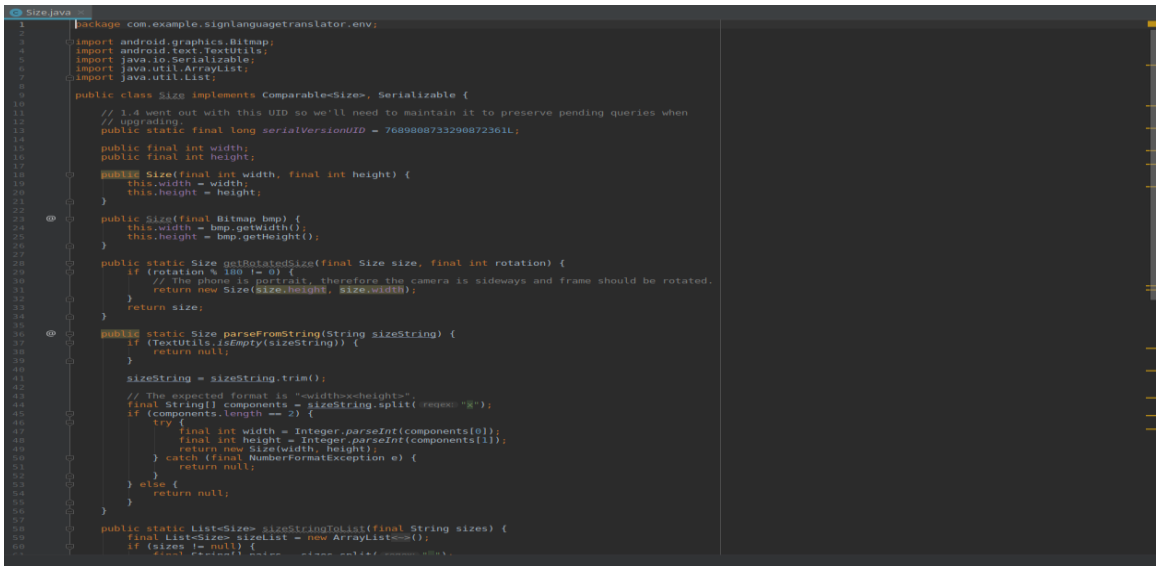
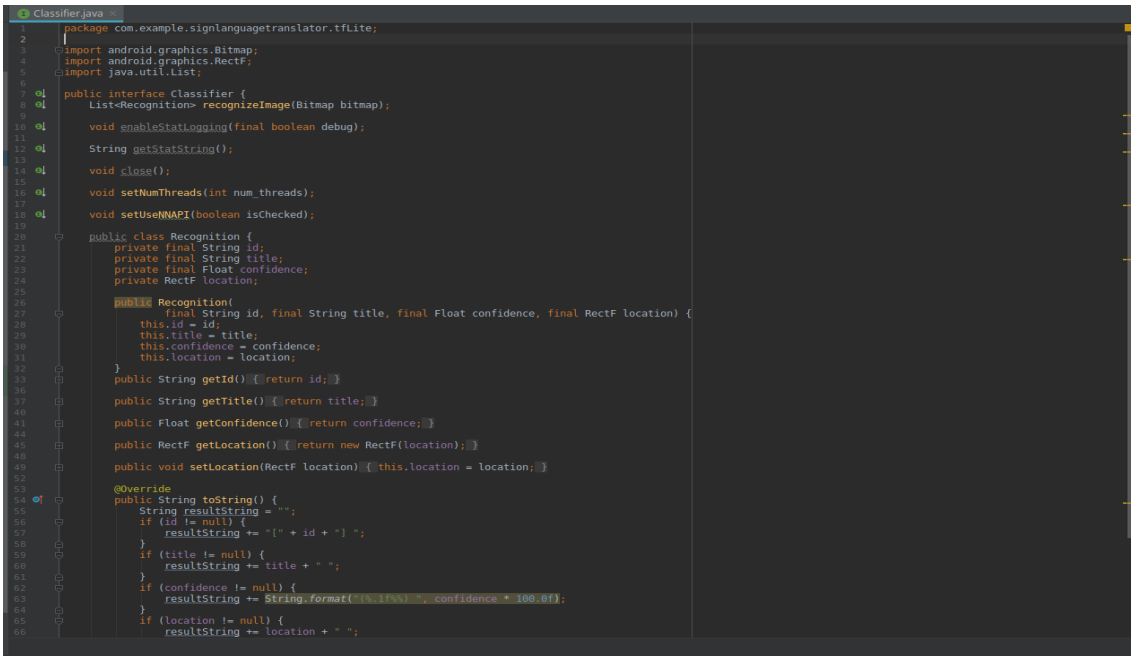


Figure 4.18: Size

III. tfLite

III.I Classifier (Interface)

Generic interface for interacting with different recognition engines.

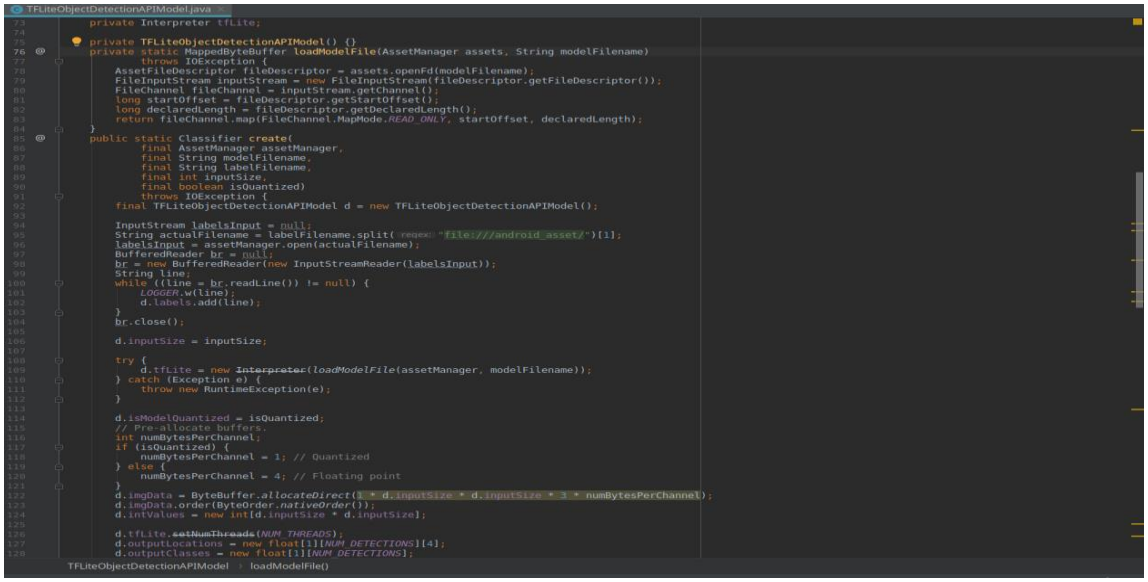


```
package com.example.signlanguagetranslator.tfLite;

import android.graphics.Bitmap;
import android.graphics.RectF;
import java.util.List;

public interface Classifier {
    List<Recognition> recognizeImage(Bitmap bitmap);

    void enableStatLogging(final boolean debug);

    String getStatString();

    void close();

    void setNumThreads(int num_threads);

    void setUseNNAPI(boolean isChecked);

    public class Recognition {
        private final String id;
        private final String title;
        private final Float confidence;
        private RectF location;

        public Recognition(
                final String id, final String title, final Float confidence, final RectF location) {
            this.id = id;
            this.title = title;
            this.confidence = confidence;
            this.location = location;
        }
        public String getId() { return id; }

        public String getTitle() { return title; }

        public Float getConfidence() { return confidence; }

        public RectF getLocation() { return new RectF(location); }

        public void setLocation(RectF location) { this.location = location; }

        @Override
        public String toString() {
            String resultString = "";
            if (id != null) {
                resultString += "[" + id + "] ";
            }
            if (title != null) {
                resultString += title + " ";
            }
            if (confidence != null) {
                resultString += String.format("(%.1f%%) ", confidence * 100.0f);
            }
            if (location != null) {
                resultString += location + " ";
```

Figure 4.19: Classifier

III.II TFLiteObjectDetectionAPIModel

Wrapper for frozen detection models trained using the TensorFlow Object Detection API.



```
    private Interpreter tfLite;

    private TFLiteObjectDetectionAPIModel() {}
    private static MappedByteBuffer loadModelFile(AssetManager assets, String modelFilename)
            throws IOException {
        AssetFileDescriptor fileDescriptor = assets.openFd(modelFilename);
        FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel = inputStream.getChannel();
        long startOffset = fileDescriptor.getStartOffset();
        long declaredLength = fileDescriptor.getDeclaredLength();
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
    }
    public static Classifier create(
            final AssetManager assetManager,
            final String modelFilename,
            final String labelFilename,
            final int inputSize,
            final boolean isQuantized)
            throws IOException {
        final TFLiteObjectDetectionAPIModel d = new TFLiteObjectDetectionAPIModel();

        InputStream labelsInput = null;
        String actualFilename = labelFilename.split( regex: "file:///android_asset/")[1];
        labelsInput = assetManager.open(actualFilename);
        BufferedReader br = null;
        br = new BufferedReader(new InputStreamReader(labelsInput));
        String line;
        while ((line = br.readLine()) != null) {
            LOGGER.w(line);
            d.labels.add(line);
        }
        br.close();

        d.inputSize = inputSize;

        try {
            d.tfLite = new Interpreter(loadModelFile(assetManager, modelFilename));
        } catch (Exception e) {
            throw new RuntimeException(e);
        }

        d.isModelQuantized = isQuantized;
        // Pre-allocate buffers.
        int numBytesPerChannel;
        if (isQuantized) {
            numBytesPerChannel = 1; // Quantized
        } else {
            numBytesPerChannel = 4; // Floating point
        }
        d.imgData = ByteBuffer.allocateDirect(1 * d.inputSize * d.inputSize * 3 * numBytesPerChannel);
        d.imgData.order(ByteOrder.nativeOrder());
        d.intValues = new int[d.inputSize * d.inputSize];

        d.tfLite.setNumThreads(NUM_THREADS);
        d.outputLocations = new float[1][NUM_DETECTIONS][4];
        d.outputClasses = new float[1][NUM_DETECTIONS];
TFLiteObjectDetectionAPIModel  ›  loadModelFile()
```

IV – tracking

IV.I – MultiBoxingTracker

A tracker that handles non-max suppression and matches existing objects to new detections.

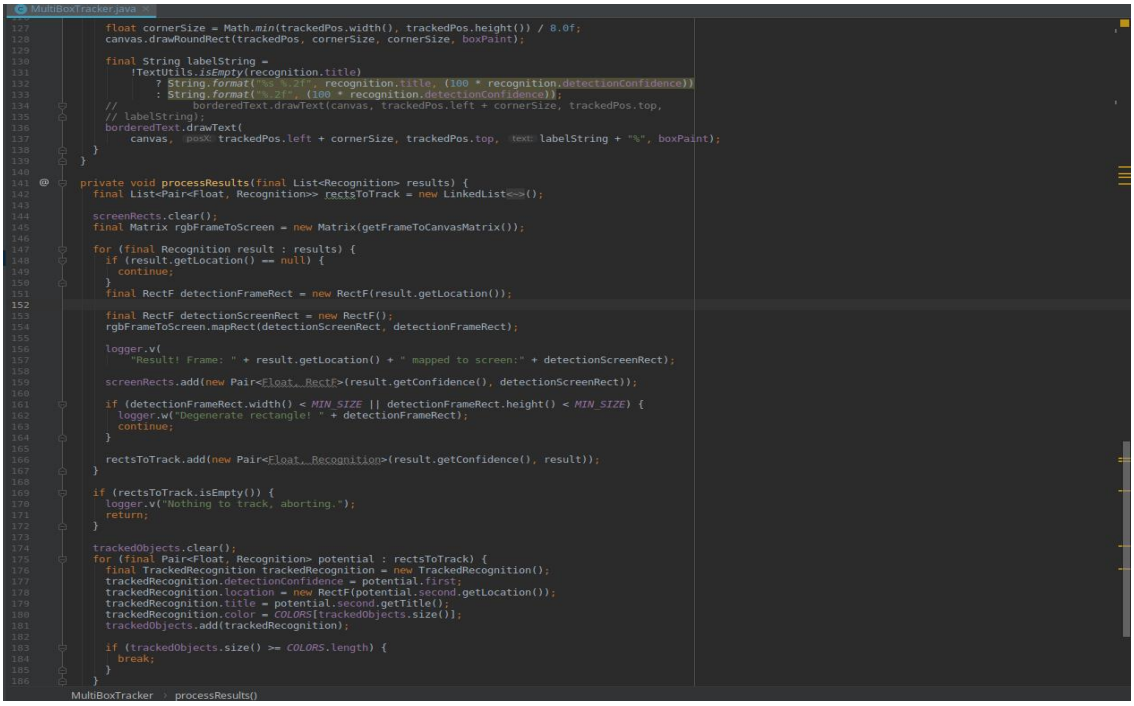


Figure 4.20: MultiBoxing Tracker

X - Camera Activity

An abstract class that controls the camera.

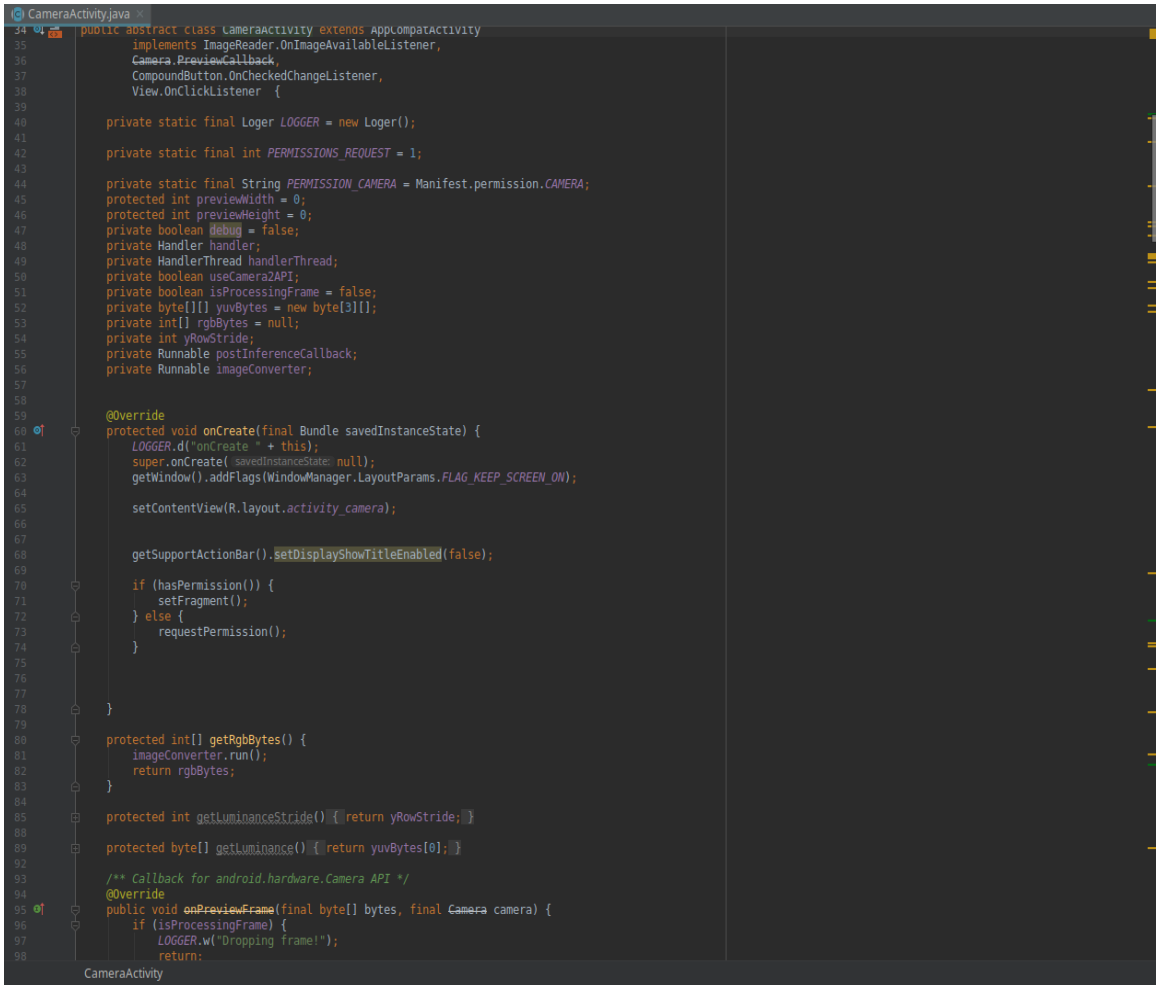


Figure 4.21: Activity camera

XI - CameraConnectionFragment

The camera preview size will be chosen to be the smallest frame by pixel size capable of

containing a DESIRED_SIZE x DESIRED_SIZE square, also convert from screen rotation to JPEG orientation.

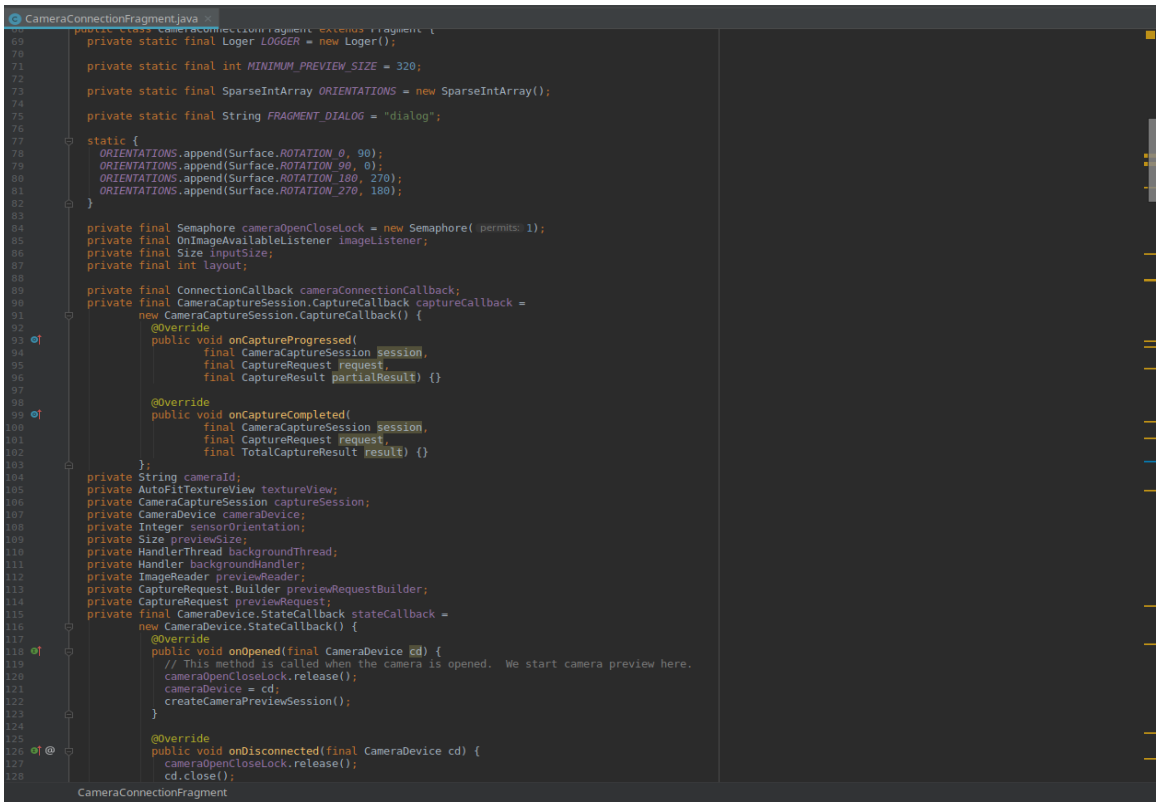


Figure 4.22: Camera connection fragment

XI - Detector Activity

The Main class of connecting the model to android that contains connection to detect.tflite file (Final Model) and label text file for labeling signs.

An activity that uses a TensorFlowMultiBoxDetector and ObjectTracker to detect and then track objects.

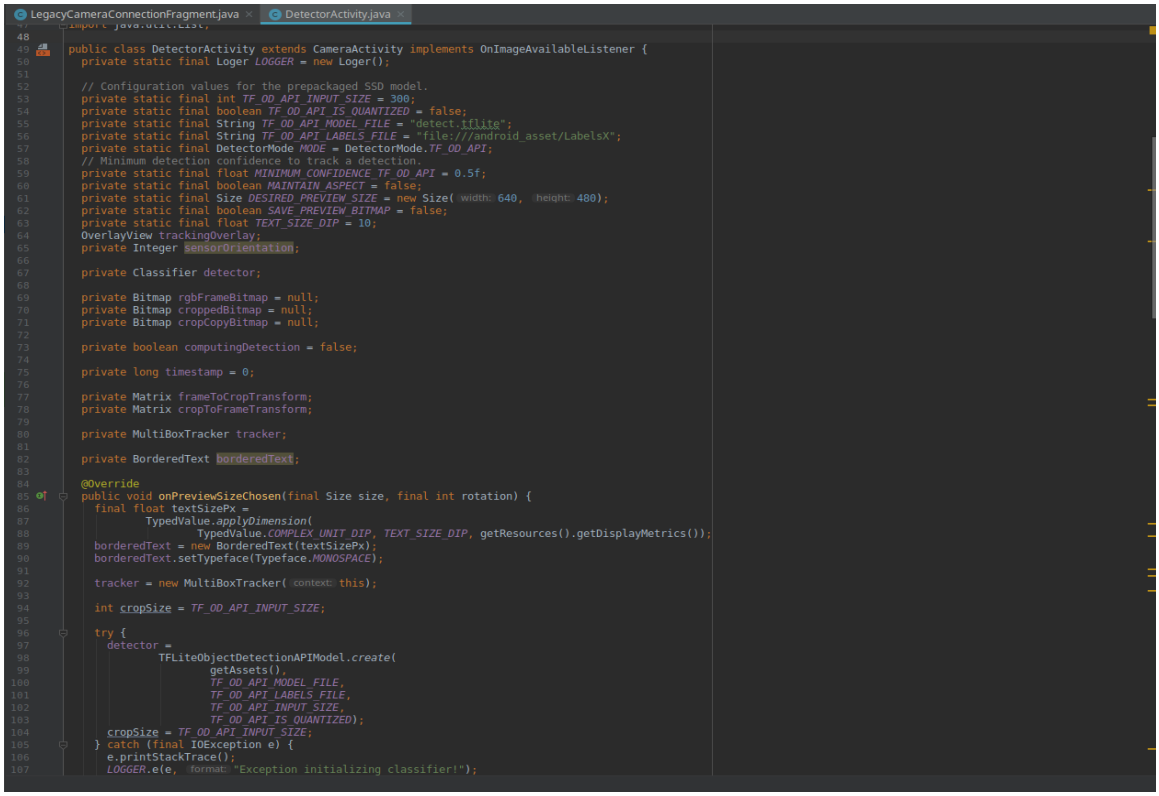


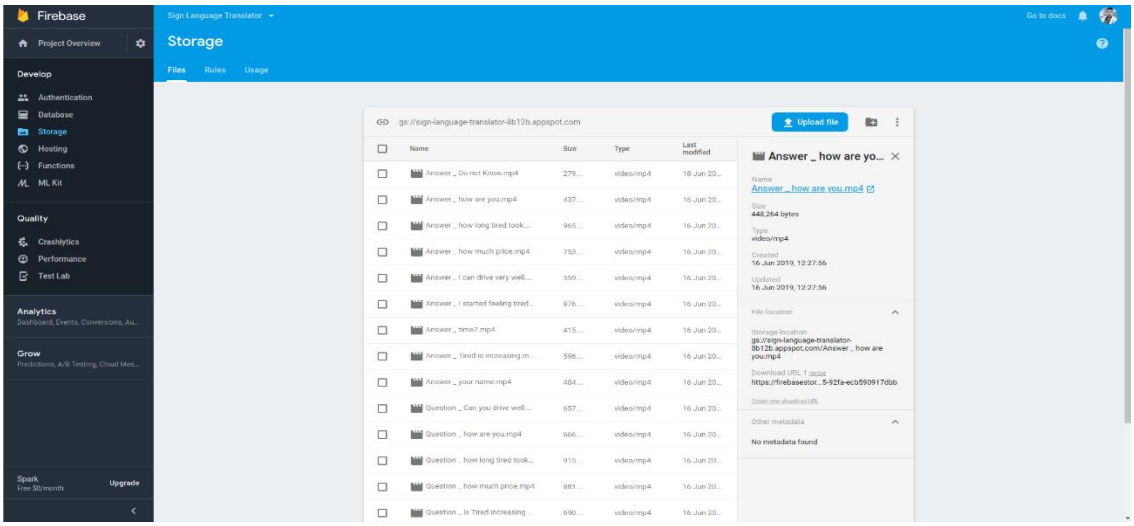
Figure 4.23: Detector activity

4.1.2.2.4 Connect Firebase to android (Dictionary Process)

In order to make a simple process the function of this sector is translate the Natural Language of specified sentences to sign language.

Here we used cloud Database (Firebase) to store the videos to reduce the space in the app.

We used cloud Firestore and from it we use Database(create table “document” for each object that contains description , tag , video_id for document) and Storage (to store videos and generate a link for each one so we can download it) , and print Description , tag for each Document in a dictionary item.



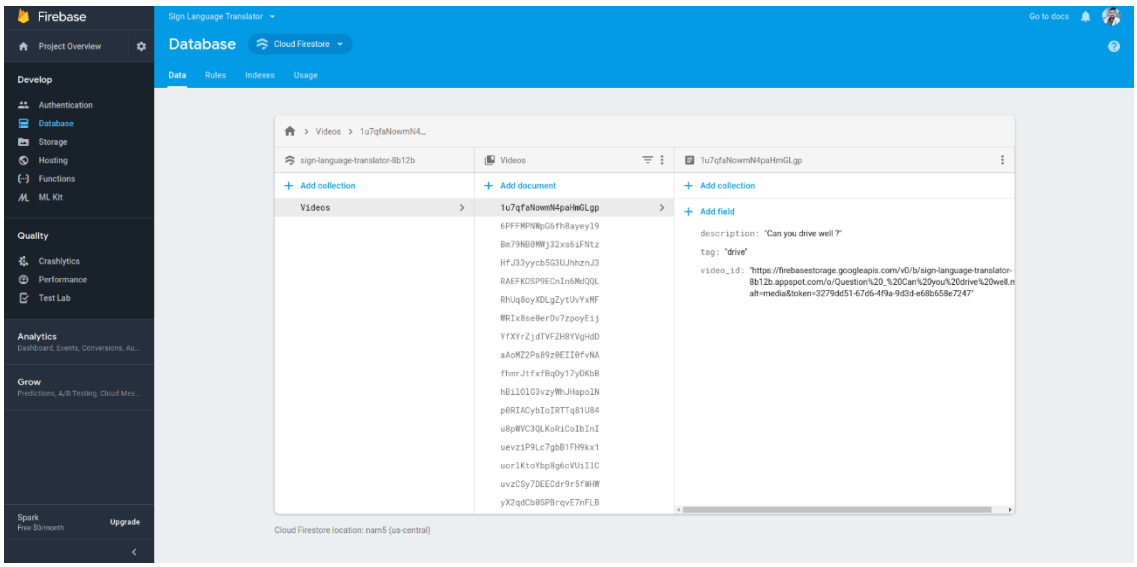


Figure 4.24: Dictionary process

4.2 Connection

we implement a class to open connection with cloud firestore and get the data in each document (Description, tag, video_id) so we have the data and we could load the video with its id.

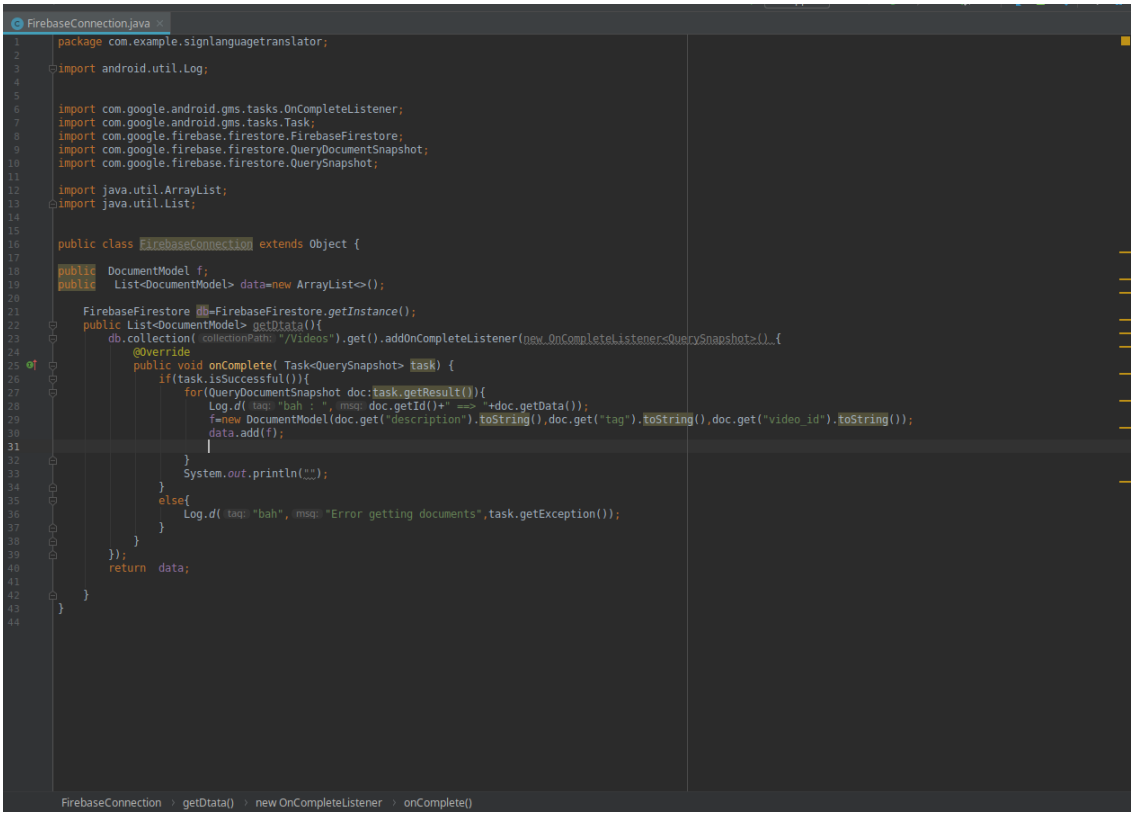


figure 4.25: Connections

4.3 Model

We created a Document model for models from cloud DB so we can interact with data.

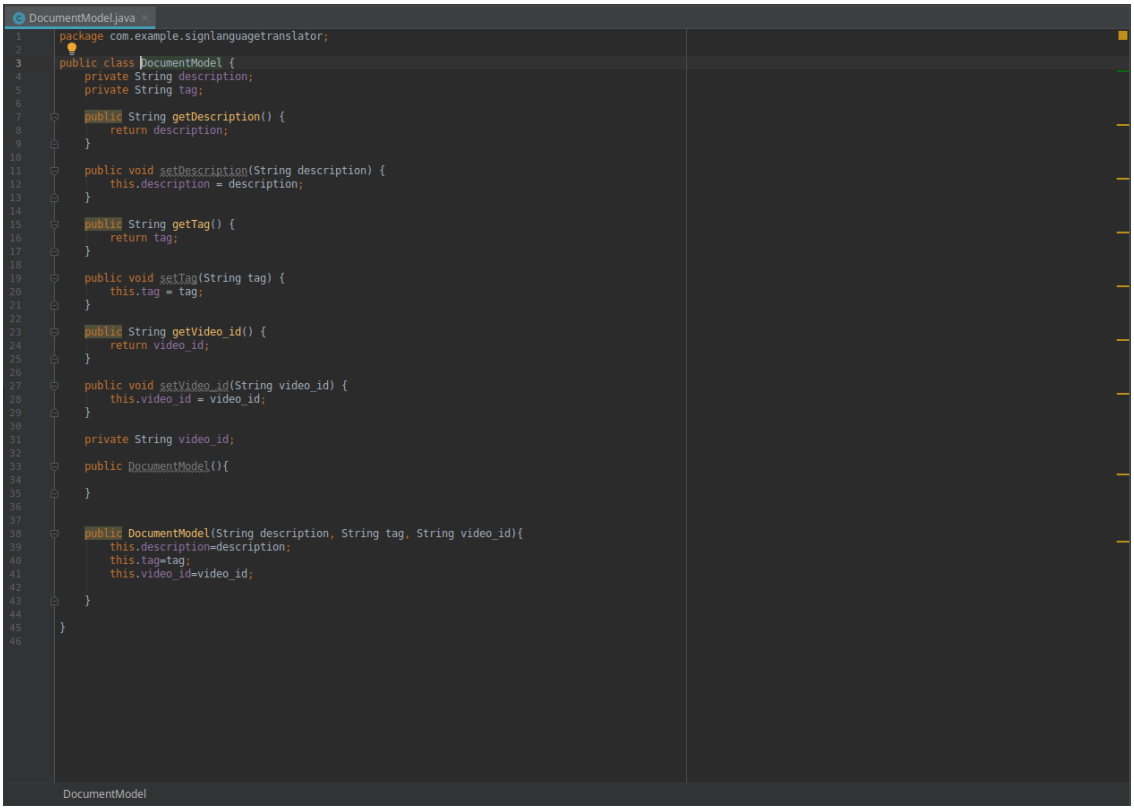
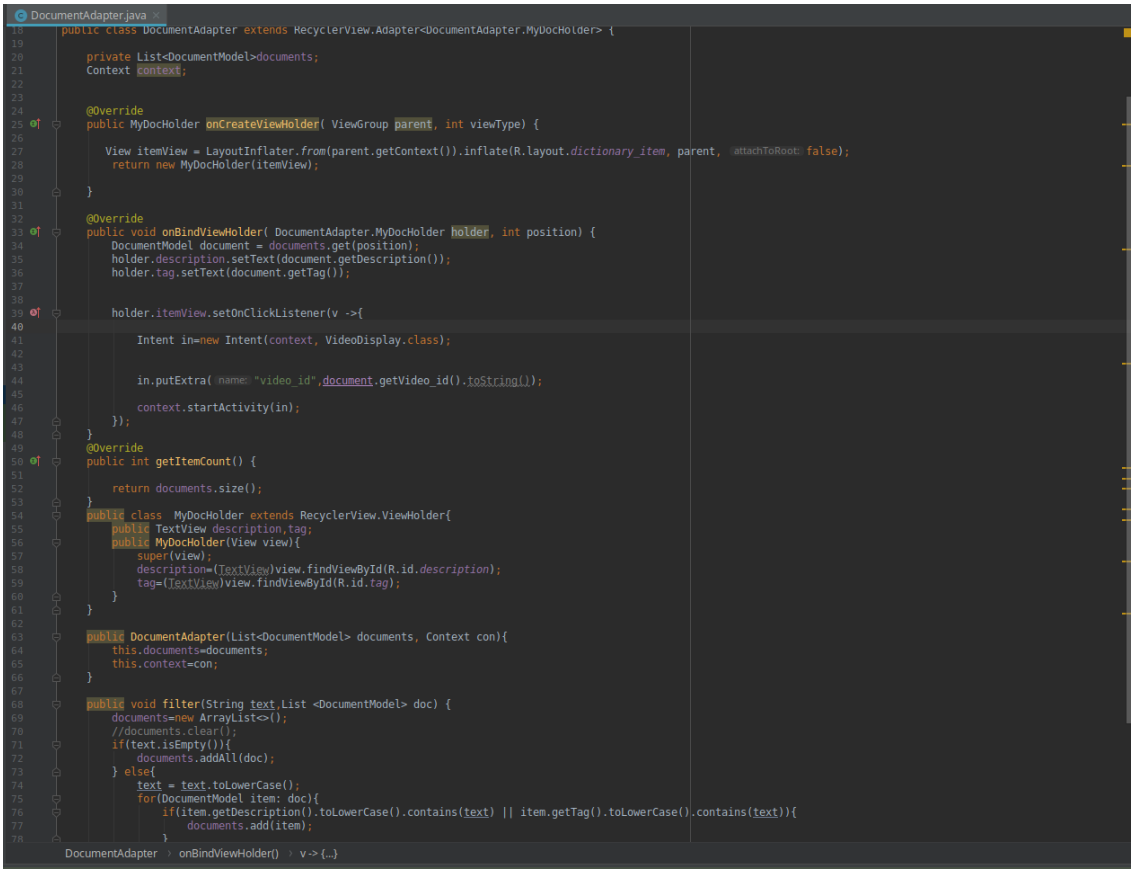


figure 4.26: Model

4.4 Document Adapter

We implement an Adapter so we can control (Display) the data in the dictionary and filter it.



```
DocumentAdapter.java
public class DocumentAdapter extends RecyclerView.Adapter<DocumentAdapter.MyDocHolder> {

    private List<DocumentModel>documents;
    Context context;


    @Override
    public MyDocHolder onCreateViewHolder( ViewGroup parent, int viewType) {

       View itemView = LayoutInflater.from(parent.getContext()).inflate(R.layout.dictionary_item, parent,  attachToRoot: false);
        return new MyDocHolder(itemView);

    }

    @Override
    public void onBindViewHolder( DocumentAdapter.MyDocHolder holder, int position) {
        DocumentModel document = documents.get(position);
        holder.description.setText(document.getDescription());
        holder.tag.setText(document.getTag());


        holder.itemView.setOnClickListener(v ->{

            Intent in=new Intent(context, VideoDisplay.class);


            in.putExtra( name: "video_id",document.getVideo_id().toString());

            context.startActivity(in);
        });
    }
    @Override
    public int getItemCount() {

        return documents.size();
    }
    public class  MyDocHolder extends RecyclerView.ViewHolder{
        public TextView description,tag;
        public MyDocHolder(View view){
            super(view);
            description=(TextView)view.findViewById(R.id.description);
            tag=(TextView)view.findViewById(R.id.tag);
        }
    }

    public DocumentAdapter(List<DocumentModel> documents, Context con){
        this.documents=documents;
        this.context=con;
    }

    public void filter(String text,List <DocumentModel> doc) {
        documents=new ArrayList<>();
        //documents.clear();
        if(text.isEmpty()){
            documents.addAll(doc);
        } else{
            text = text.toLowerCase();
            for(DocumentModel item: doc){
                if(item.getDescription().toLowerCase().contains(text) || item.getTag().toLowerCase().contains(text)){
                    documents.add(item);
                }
DocumentAdapter › onBindViewHolder() › v -> {...}
```

figure 4.27: Document Adapter

4.5 Control Video

we implement a VideoDisplay Class to load video from cloud Storage and play it.

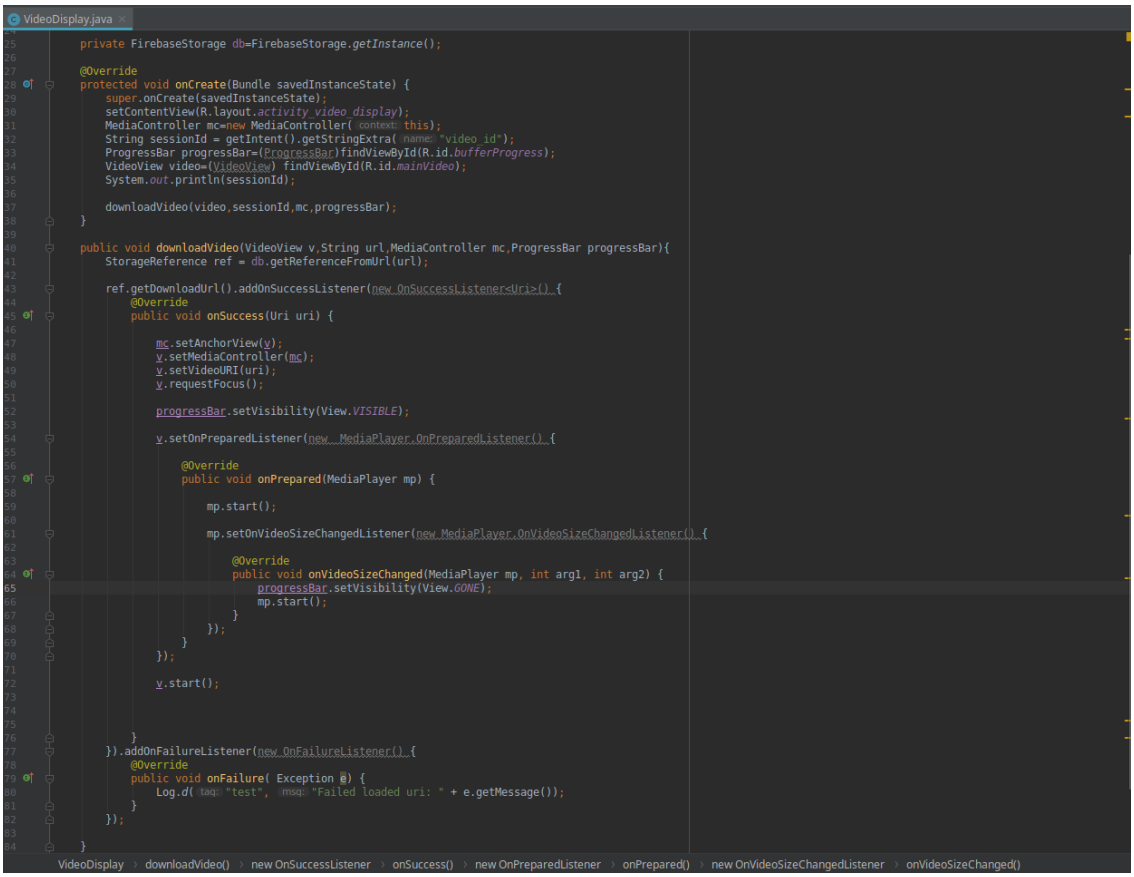


figure 4.28: Control Video

General Classes

1- MainActivity

- The back-end file (Java File) for the start screen of our app

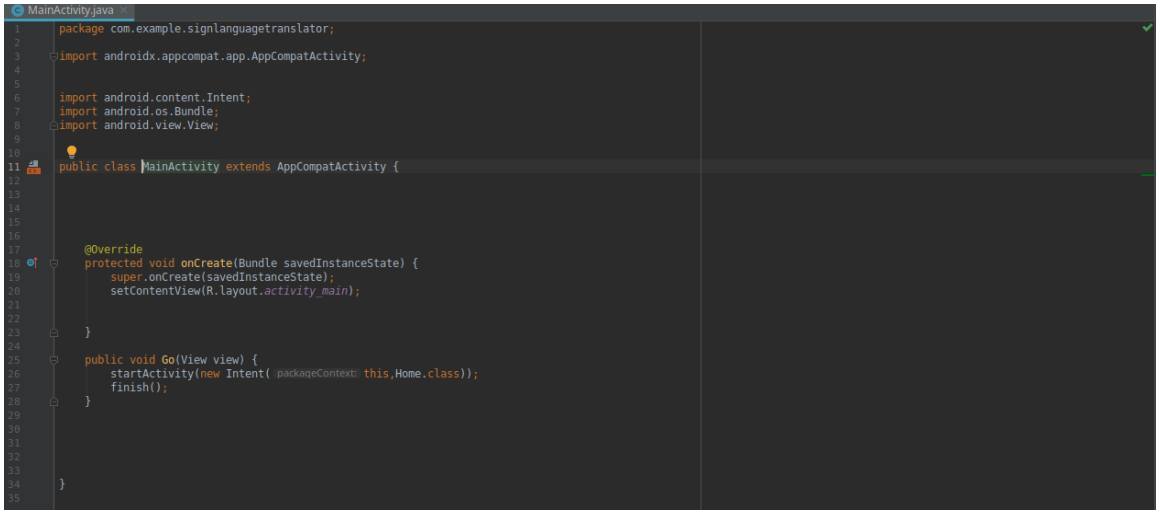


Figure 4.29: Main activity

2- Home

- The back-end file (Java File) for the main screen of the app.

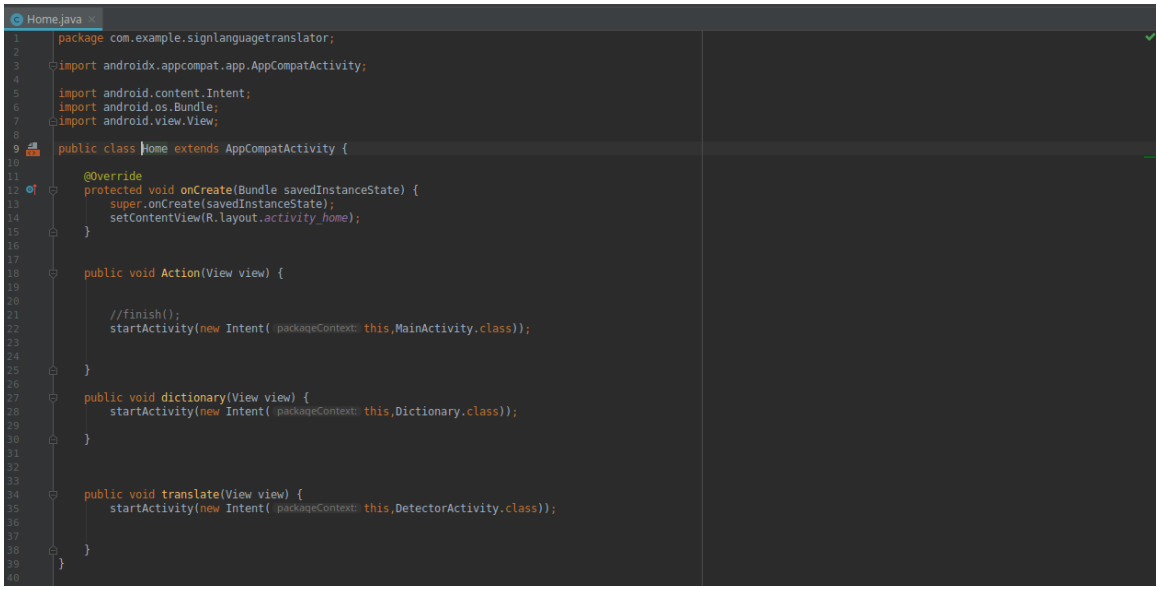


Figure 4.30: Home

4.6 Project Structure

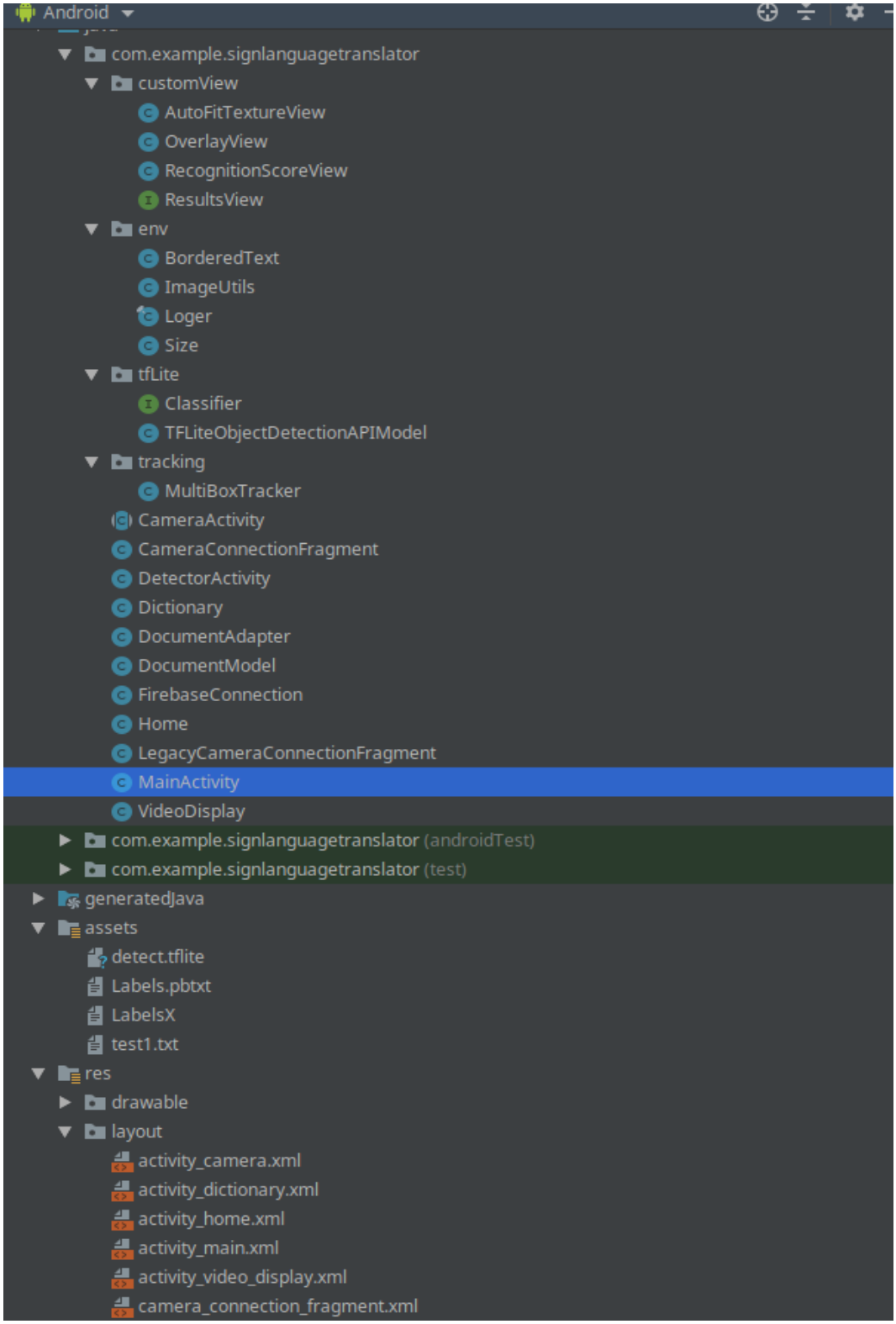


figure 4.31: Project Structure

Chapter 5

Testing:

5.1 Android App:

5.1.1 Introduction

Testing app is an integral part of the app development process. By running tests against app consistently, we can verify your app's correctness, functional behavior, and usability before you release it publicly.

Testing also provides you with the following advantages:

- **Rapid feedback** on failures.
- **Early failure detection** in the development cycle.
- **Safer code refactoring**, letting you optimize code without worrying about regressions.
- **Stable development velocity**, helping you minimize technical debt.

5.1.2 Fundamentals of testing

Users interact with app on a variety of levels, from pressing a **Submit** button to downloading information onto their device. Accordingly, we should test a variety of use cases and interactions of app.

5.1.3 Use an iterative development workflow

We might find it necessary to fetch data from a server, access local storage, or render complex user interfaces. When developing a feature, we start by either writing a new test or by adding cases and assertions to an existing unit test

5.1.4 Testing Pyramid

The Testing Pyramid illustrates how app should include the three categories of tests small, medium, and large:

1. Small tests are unit tests that you can run in isolation from production systems. It should run quickly on your machine.

2. Medium tests are integration tests that sit in between small tests and large tests. it they run on emulators or real devices.

3. Large tests are integration and UI tests that run by completing a UI workflow. They ensure that key end-user tasks work as expected on emulators or real devices.

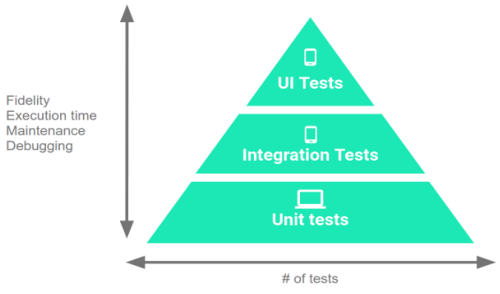


Figure 5.1 Testing pyramid

5.1.5 Test plan

1- Goal of test plan

This Test Plan for SL translator is to check the user availability of using our mobile app and the usability.

Check the accuracy of the result of the translation and reach our project goal.

2- Scope

1- Features to be tested:
Translation,
Dictionary.

2- Features Not to be tested:
There are no features not to be tested.

3- Approaches

The test manager will create test runs for each tester. The tester will execute the tests and mark each case as Pass / Fail / Skip. The tester should leave notes on actual results and any other relevant details when possible.

When tests are marked as Fail, bug reports will automatically be created.
Once complete, the test manager should review the test run reports and report to the team accordingly

a. pass / fail criteria

All core functionality of the systems should function as expected and outlined in the individual test cases. There must be no critical defects found and an end user must be able to complete a purchase cycle successfully and initiate a refund without any errors. 95% of all test cases should pass and no failed cases should be crucial to the end-user's ability to use the application.

b. Environment

1- Google pixel 3 XL → API 29
2- HTC One M9 → API 24
3- Huawei P8 → API 25

5.1.5.1 Functional Testing Plan

Test id	User Goal	Identification
1	Start	General user
2	Dictionary	General user
3	Video	General user
4	Translate	General user
5	Model	General user

5.1.5.2 Non-functional Testing Plan

Specific Goal	Subcategory	Category
Screen Size Compatibility, Camera Functionality, Proper Data Storage.	Android	Platform dependency
User Response Time per Step.	Intuitive Flow of Application	Usability
Essentially Instantaneous Screen Transitions.	Load Time	Performance
High level of translation.	Accuracy	Correction

5.2 Our project test is on accuracy of translation (Text Sign language).

App detect photos of sign language to translate it to text and make a test on the result to get the accuracy of the translation.

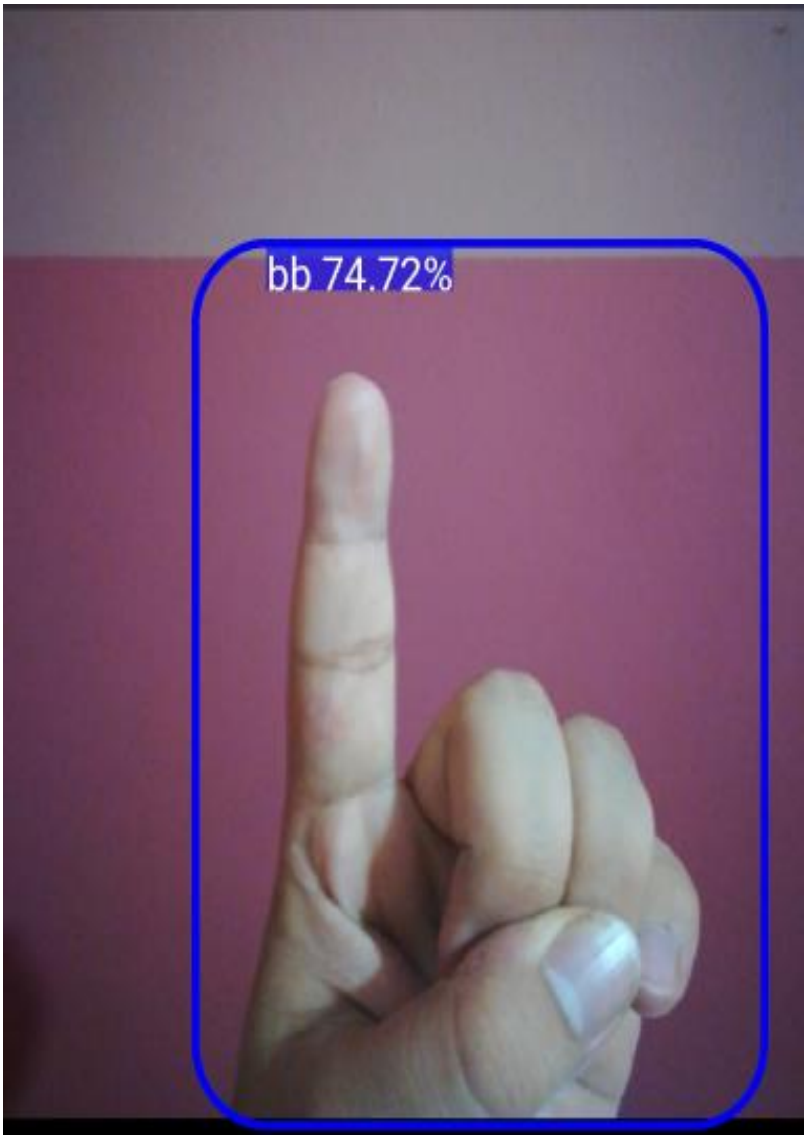


Figure 5.2: example1 of test

This sign is not accurate so the percentage is not very high.

Another example of testing.

Figure 5.3: example2 of test

This accuracy of translation is too high as the sign acted well.

Finally, the last example of testing and our project.

Figure 5.4: Accuracy test

So, we finished the test as we achieved our target to have a high accuracy of translation.

Future Work:

We have a lot of plans about app future work:

1- Create website for our app.
2- Detect the sign and convert it to text and sound.
3- Translate text and sound to sign language (dynamic with avatar).
4- Choose any language want to convert it to sign language and vice versa.
5- Update user interface.

References:

1. https://en.wikipedia.org/wiki/Sign_language
2. https://www.lifeprint.com/asl101/pages-layout/evolutionofsignlanguage.htm
3. https://www.sciencedirect.com/science/article/pii/S1877050917320720
4. http://tmu.ac.in/college-of-computing-sciences-and-it/wp-content/uploads/sites/17/2016/10/T203.pdf
5. https://www.tensorflow.org/lite/models/object_detection/overview
6. https://stackoverflow.com/questions/56187449/problem-building-tensorflow-lite-for-android
7. https://firebase.google.com/docs

Thanks

We would like to thank you all for your help and effort to achieve our goal and launch our application.

We hope you find the idea of our project is interesting and helpful to our community and you enjoyed our documentation DR/ HATEM MOHAMED and your staff.

We wish the app will launch in real world soon.

Thank ALLAH for the success, doing our work in the certain time and give us the chance and time for trying to get our project to perfection.