Luxor University

Murmuro

Team members:
Abdel-Hafiz Ibrahim
Abdel-Satar Ahmed
Ahmed Youeesf
Mohammed Najeh
Mohmmed Asharaf
Omar Atef
Oraby Mahmoud

2 juillet 2020

Luxor University

Faculty of Computers and Information

Computer Science Department

كلية الحاسبات والمعلومات بالأقصر
FACULTY OF COMPUTERS AND INFORMATION

Project Advisors:

Dr-Mohammed Atta Khafagy

Dr-Safynaz Abdelfattah

Sommaire

Titre 1

Sommaire 2

1 Abstract 3

2 Introduction 4

3 Analysis and Design 4
- 3.1 Updated Functional Requirements 4
- 3.2 Updated Non-Functional Requirements 5
- 3.3 Updated Use Case Requirements 6
- 3.4 Design Classes 10
- 3.5 Sequence Diagram 10
- 3.6 Software Architecture 11

4 Prototype description 11
- 4.1 Implementation Platform 12
- 4.2 Mapping between requirements and implemented functions 12
- 4.3 Implementation details 12
  - 4.3.1 Neural Network Model 12
  - 4.3.2 Authorization 13
  - 4.3.3 Chatting 16
  - 4.3.4 Live Translation 22
  - 4.3.5 Application main 24
- 4.4 Actual database schema 25

5 Testing 25
- 5.1 Expected test scenarios 25
- 5.2 Unit test 25
- 5.3 Functional test 25
- 5.4 Usability test 25

6 Deployment of the system 25

7 Limitation of the system 25

8 Conclusion and further work 25

9 Appendix 25
- 9.1 *getMessagesAdapter* method 25
- 9.2 *onActivityCreated* method 41

Bibliographie 46

1 Abstract

Hearing loss, also known as hearing impairment, is a partial or total inability to hear. A deaf person has little to no hearing. Hearing loss may occur in one or both ears. As of 2013 hearing loss affects about 1.1 billion people to some degree. It causes disability in 5% (360 to 538 million) of the world and moderate to severe disability in 124 million people. Most of the deaf cannot read or write and also they cannot communicate with other normal people. Hearing impaired people use sign language to communicate with each other. Sign language employs signs made with the hands and other movements, including facial expressions and postures of the body. There are many different sign languages as, for example, British and American sign languages. In this project we are going to employ machine learning approaches to translate American Sign Language to normal English language to be understood by non-deaf people and also translate normal English to sign language to be understood by deaf people. Also designing is a chat application for deaf/normal people with a live translation feature.

2 Introduction

Signing has always been part of human communications. The use of gestures or signs is not tied to ethnicity, age, or gender. Infants use gestures as a primary means of communication until their speech muscles are mature enough to articulate meaningful speech. For millennia, deaf people have created and used signs among themselves. These signs were the only form of communication available for many deaf people. Within the variety of cultures of deaf people all over the world, signing evolved to form complete and sophisticated languages. These languages have been learned and elaborated by succeeding generations of deaf children. Normally, there is no problem when two deaf persons communicate using their common sign language. The real difficulties arise when a deaf person wants to communicate with a non deaf person. Usually both will get frustrated in a very short time [3] Additionally, these individuals may have difficulty listening in classrooms or conferences, ordering in restaurants, watching TV or movies, listening to music, speaking on the telephone, etc. Current Solutions include communicating with pen and paper ; however, this method is quite slow and inconvenient. Furthermore, Some hard of hearing individuals may have difficulty communicating with written language as there is no commonly used written form of sign language[2] To date, most work on sign language recognition has employed expensive wired "datagloves" which the user must wear [4] In addition, these systems have mostly concentrated on finger signing, in which the user spells each word with finger signs corresponding to the letters of the alphabet [1] However, most signing does not involve finger spelling but instead, gestures which represent whole words, allowing signed conversations to proceed at about the pace of spoken conversation. In this project, we use deep learning methods to build a deep neural network for recognizing American sign language. Our model is trained on a data set of American Sign Language Linguistic Research Project (ASLLRP) of Boston university which is available at **http://www.bu.edu/asllrp/ and ***** reference data set of dr.Atta ******** In addition a 3D avatar is designed to translate normal english language into American Sign Language. Both the deep neural network model with the 3D avatar constitutes the translation engine. Hence, The translation engine will translate signs into normal English text and translate normal English text into a sign. This translation engine can be deployed in different platforms whether as a desktop software, a web application, or as a mobile application. For a comprehensive demo of how the engine works, we have applied the translation engine in a chat application for mobile devices with a live translation feature.

3 Analysis and Design

In the following sub-sections, we provide the updated design requirements for the translation engine.

3.1 Updated Functional Requirements

Hand and face detection.
This process is required for recognizing a sign from a deaf as it is required to isolate hand and face from the other objects captured from the scene where the deaf exists.

Translating captured sign.
After detecting hand and face from a sequence of frames, the hand and face are passed to the model to be classified and output the label. The outputted label is the translation result and corresponds to the English meaning of the sign.

Providing an animated 3D model (avatar) that will interact with deaf people.
Not all deaf people can read or write and most of the deaf feel more comfortable communicating with their language. Providing an animated avatar would achieve that goal to make the application more interactive, easy and comfortable for use.

Converting text to speech and speech to text.
For a normal user, it is convenient to record a voice to be translated to sign, this process requires that the voice is converted to a text in order to be inputted to the engine so the avatar is animated corresponding to the input text. Further, when the sign is translated into text, the text is converted to speech so a normal person can hear what is being said.

Provide live translation from one language to another.
Live Translation is a feature embedded in the application that allows users to directly translate sign language to speech or speech to sign. This feature is useful in situations where people are talking to each other face to face such as a meeting, or friends conversing with each other.

Chatting.
Chatting is the common way for communicating with friends. The incoming messages are represented by the avatar so deaf can understand it, the avatar will be part of the chatting screen. Also, a deaf can send messages by capturing the hand motion via camera. As most deaf cannot read or write, we provide them with a keyboard that has the representation of a letter in sign language as in Fig 1.

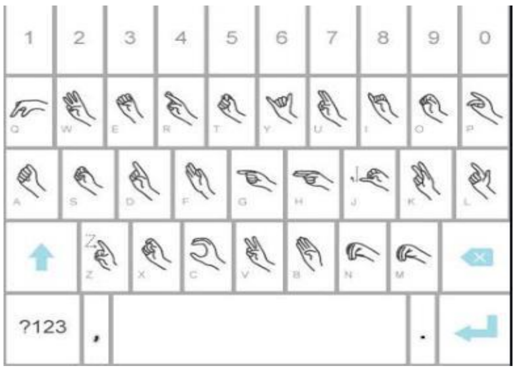

Figure 1. keyboard.

3.2 Updated Non-Functional Requirements

Software Quality Attributes.

- Scalability : Application should be able to provide instant messaging services to many users at any given time. The translation model will retain a large set of common words to achieve elastic translation.
- Robustness : In case a user's device crashes, a backup of their chat history must be stored on remote database servers to enable recoverability. The translation model will be built carefully to a large set of vocabulary to produce unambiguous, clear translation.

- Reliability : Application should translate messages correctly and sends messages instantly.The translation model will be updated regularly to achieve better translation accuracy.

Performance Requirements.
The application is light on memory, and will not require any intensive computations. At least 1-MB internet speed is required for chatting. The size of the avatar depends on the it's file format. In table 1 we provide the average size of the avatar, with a dictionary of 60 vocabulary each is 20 frames, for different file formats.

File format	Size in MB
Raw (.blend)	40
FBX (.fbx)	74
Wavefront (.obj)	8.5
Collada (.dae)	17.6
Alembic (.abc)	9
Universal Scene Description (.usd, .usdc, .usda)	14.6
Motion capture (.bvh)	78.5
STL (.stl)	4.5
glTF 2 (.gitf)	14
Rendered video	0.075/video
Rendered GIF	0.090/file

Table 1. Avatar size

***** screenshot for app profiler in android studio ******
***** provide application and model size *****
***** translation time ******

Safety and Security Requirements.
The application will use end-to-end data encryption to attain confidentiality for data when sent over the internet. Users' data will be encrypted in the database.
*****screenshot of encrypted data in firebase ******

3.3 Updated Use Case Requirements

The full use case diagram of the application system is depicted in the following figure 2.

Use Case1 (Chat).
Purpose : Enabling the user to chat with the others. This includes exchanging text messages, signs, PDFs, pictures, etc.
Requirements Traceability : Enabling users to communicate with their contacts.
Priority : High.
Preconditions : Registered user, Internet connection, open chat.
Post conditions : Users can communicate with each other.
Actors : Deaf/normal user.
Extends : None.
Flow of Events :
1. *Basic Flow* – Once the user logged in application, he/she can see available contacts and

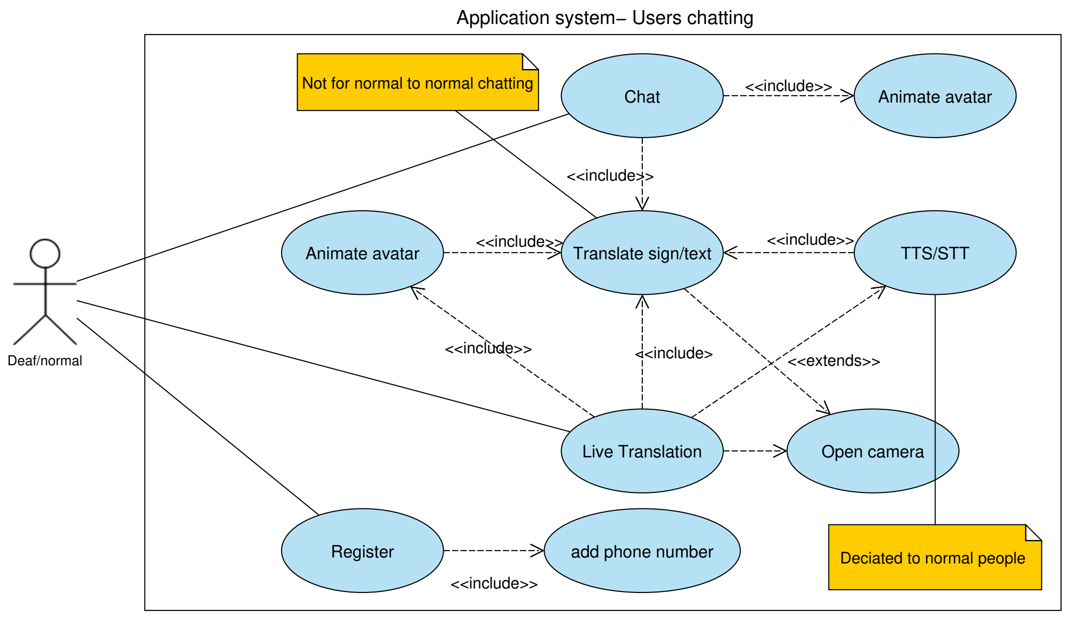

Figure 2. Use case diagram.

chooses one to chat with. If a normal person is chatting with a deaf person, the normal person sends a message then the message is represented by avatar animation at deaf side. If the deaf wants to send a message, he/she enters his/her sign via camera then the sign is translated to text using the translation model.
2. *Alternative Flow* – users can communicate only with text messages without translation from one form to another, or both users can chat with sign.
3. *Exceptions* – Users cannot contact unregistered contacts.
Includes : Translate sign/text, and animate avatar.
Notes/Issues : None.

Use Case2 (Animate avatar).
Purpose : Triggering avatar to perform the motions and gestures so deaf can understand exchanged messages.
Requirements Traceability : Making deaf understand other people.
Priority : High.
Preconditions : Requires an input be formatted English text.
Post conditions : Deaf can understand other people.
Actors : None.
Extends : None.
Flow of Events :
1. *Basic Flow* – The avatar gets as input a valid English text, the avatar then represents each word separately.
2. *Alternative Flow* – If the avatar received an unknown or misspelled English word, the avatar represents them character by character.
3. *Exceptions* – Only text of English characters and numbers is a valid input to the avatar.
Includes : Translate sign/text .
Notes/Issues : None.

Use Case3 (TTS/STT).
Purpose : Convert text to speech and speech to text.
Requirements Traceability : A normal user can get the intuiation that a deaf is talking to him.
Priority : High.
Preconditions : The input must be a text or speech.
Post conditions : Normal people can understand the deaf.
Actors : None.
Extends : None.
Flow of Events :
1. *Basic Flow* – The normal user can choose to convert incoming messages to speech so he can hear it instead of read.
2. *Alternative Flow* – The normal user can choose to input speech to communicate with others, so the speech must be converted to text.
3. *Exceptions* – The input must be clear and correct otherwise the output will be ambiguous.
Includes : Translate sign/text.
Notes/Issues : This feature is dedicated only for normal people.

Use Case4 (Live Translation).
Purpose : This use case for translating a text or sign to another format. The input and output is in the same device, no transmission of the data.
Requirements Traceability : Deaf and normal can communicate face to face without an intermediate human translator.
Priority : High.
Preconditions : The input must be a text, speech or sign.
Post conditions : Normal and deaf can understand each other easily.
Actors : Deaf/normal user.
Extends : TTS/STT.
Flow of Events :
1. *Basic Flow* – The user can input a speech to application, the speech is translated to text, text is passed to the translation model, the model passes its output to the avatar, and finally the avatar represents the corresponding sign language.
2. *Alternative Flow* – Deaf users can use the camera to capture the sign, sign input to translator model, translator outputs a text which may be passed to be converted to speech.
3. *Exceptions* – The input must be clear and correct otherwise the output will be ambiguous.
Includes : Animate avatar, translate sign/text, and open camera.
Notes/Issues : None.

Use Case5 (Translate sign/text).
Purpose : Translate from normal English language to American Sign Language and vice versa.
Requirements Traceability : Making deaf and normal people understand each other.
Priority : High.
Preconditions : An input as English text or sequence of image frames represents a sign.
Post conditions : Translated English from sign or translated sign from text.
Actors : None.
Extends : None.
Flow of Events :
1. *Basic Flow* – This use case is to input an English text, pass the text to the translator model,

the model then outputs a text with a specific format representing movements that the avatar will do.
2. *Alternative Flow* – It may input a sequence of image frames that represent a sign, pass the frames to the translator model, then output a sequence of text that represents the corresponding meaning in English.
3. *Exceptions* – Only text and images are the valid format for translation, other formats such as voice is not allowed. Voice data must be converted to text first to be input to the translator model.
Includes : open camera.
Notes/Issues : None.

Use Case6 (Register).
Purpose : Register the user to the system so he/she can use the application.
Requirements Traceability : This is the first step for connecting users to his/her friends.
Priority : High.
Preconditions : Internet connection.
Post conditions : The user is registered at the system.
Actors : Deaf/normal user.
Extends : None.
Flow of Events :
1. *Basic Flow* – The user downloads the application from the market, run application. If it is the first time to run the app, the first screen appears is the registration screen. Otherwise the application will show its default screen.
2. *Alternative Flow* – If no Internet connection, the application will prompt the user to have an internet connection.
3. *Exceptions* – None.
Includes : Add phone number.
Notes/Issues : None.

Use Case7 (Add phone number).
Purpose : Having a unique identifier for registering the user to the system, also used by other users to connect with him.
Requirements Traceability : A complementary step for registering users.
Priority : High.
Preconditions : User should have a valid phone number, Internet connection.
Post conditions : The user is registered at the system.
Actors : Deaf/normal user.
Extends : None.
Flow of Events :
1. *Basic Flow* – The user enters his/her phone number to the required field, then the system verifies that phone number is valid and is owned by the current user by sending a unique number via SMS.
2. *Alternative Flow* – If the user entered a wrong number, he/she is prompted to enter it again and again until verification is done.
3. *Exceptions* – None.
Includes : None.
Notes/Issues : None.

3.4 Design Classes

NOT YET
WAIT ORABY

3.5 Sequence Diagram

The sequence diagram of the application is depicted in figure 3.

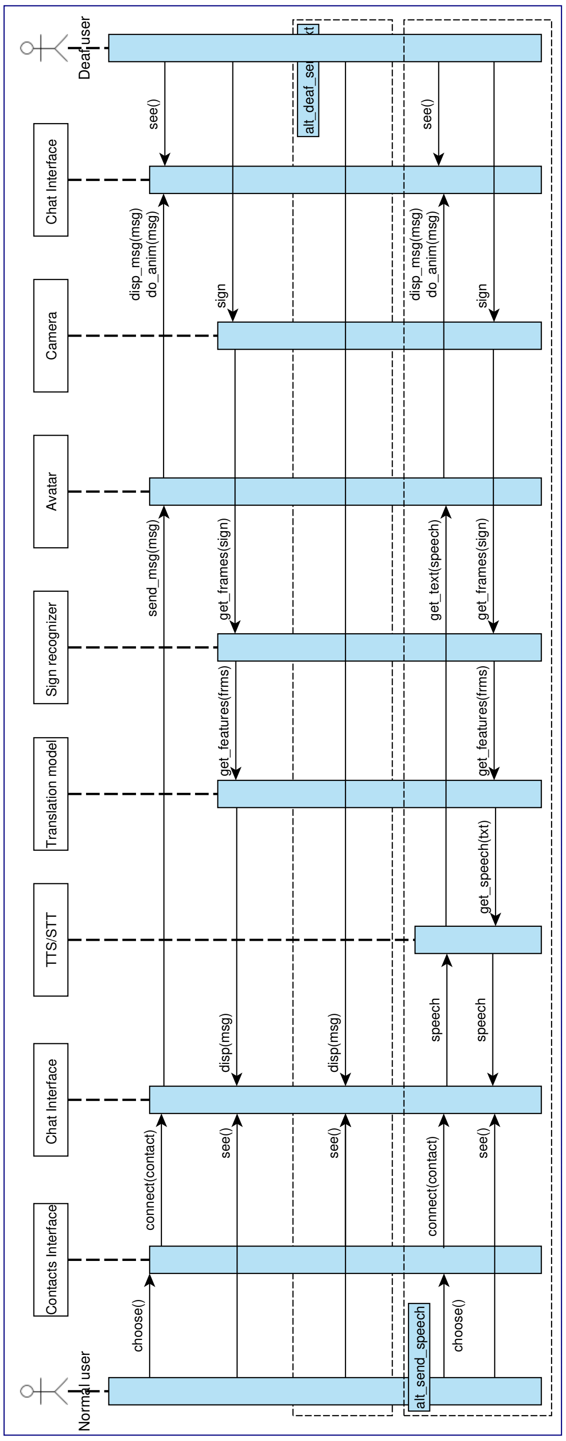

Figure 3. Sequence Diagram.

WAITING model sequence diagram from ashraf

3.6 Software Architecture

The context diagram of the application is depicted in figure 4. When the deaf user sends a message to an ordinary person, his message is passed to ML model, which is the neural network model that translates sign into text, then the translated message is sent to other side. On the other hand when an ordinary user sends a message to a deaf, the message is passed to the avatar, which is a rigged character used for expressing the sign language. Message exchanging is done between the end users through chat interface, on the other hand the chat interface, as part of the whole application, uses the engine to get proper translation. All sent messages and media are saved in the Firebase. The Firebase stores the chat between end user in a text format, meaning that no signs or avatar's animation are stored but normal-encrypted text. Other media formats like images, sound voices and documents are saved in it's original format. WRITE THAT YOU USED C4 MODEL

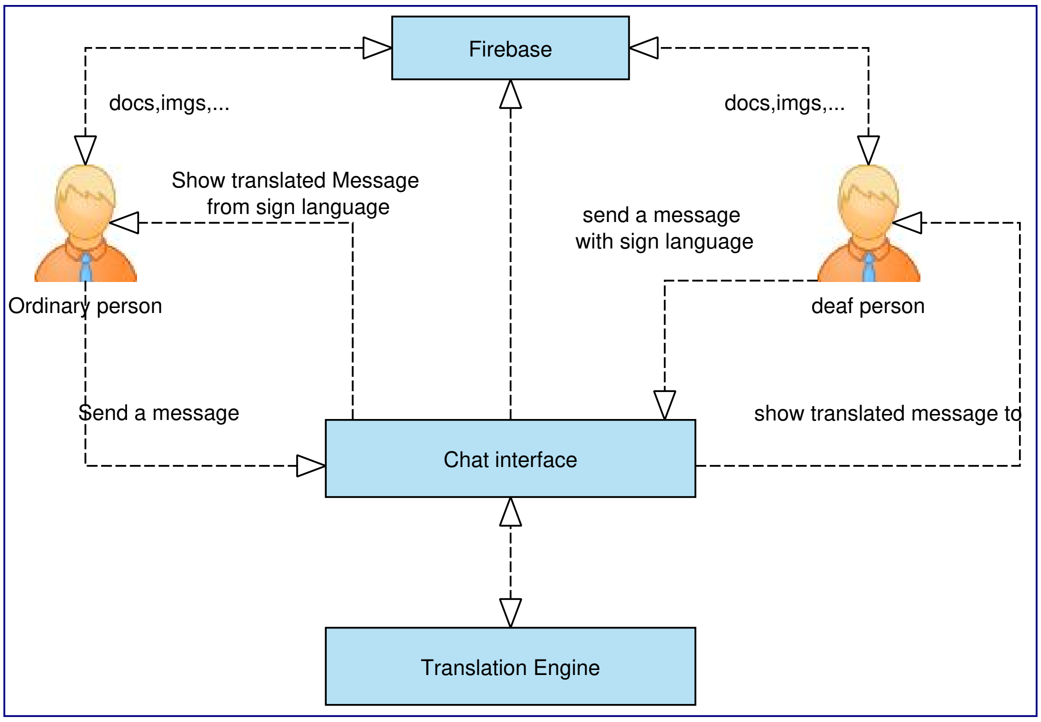

Figure 4. Use case diagram.

4 Prototype description

Our major prototype is the chat application that enables the deaf to chat with other deaf or normal people. The chat app also contains a live translation feature to enable two users to translate a sign into text or vice versa when the users are communicating face to face. A secondary prototype is a desktop application that runs translation engine only. This prototype is used for evaluating the performance and the accuracy of the engine whether translating sign into English text or English text into signs.

4.1 Implementation Platform

For the chat app, it is implemented in Android studio and Java language. The app is provided for the any mobile platform that runs Android, IOS or Windows phone. The released version of the application is for Andorid phones with Android version 4.4 and higher. Meanwhile the desktop application is deployed for Windows 7, 8, 8.1 and 10 and is implemented in visual studio and c# language with WPF GUI design scheme. The main requirements used to running the engine properly is that the platform have at least medium-quality camera and is capable of running 3D graphics models.

4.2 Mapping between requirements and implemented functions

4.3 Implementation details

4.3.1 Neural Network Model

The embedding of the Neural Network model, which it is original format is H5, requires transforming this model format into another format suitable for mobile application; which is so called tensorflow lite. Running tensorflow lite models in mobile application requires implementing The *Classifier* interface. *TensorFlowImageClassifier* is a concrete class of *Classifier* interface and we summarize the main methods below : **create() :** This method is used to load the model from application assets and returns a classifier object.

```java
public static Classifier create(AssetManager assetManager,
                                    String modelPath,
                                    String labelPath,
                                    int inputSize,
                                    boolean quant) throws IOException {

        TensorFlowImageClassifier classifier = new TensorFlowImageClassifier();
        classifier.interpreter = new Interpreter(classifier.loadModelFile(assetManager,
    modelPath), new Interpreter.Options());
        classifier.labelList = classifier.loadLabelList(assetManager, labelPath);
        classifier.inputSize = inputSize;
        classifier.quant = quant;

        return classifier;
    }
```

recognizeImage() : The role of this method is handling the input image coming from the camera, the images are stored in a *ByteBuffer* and then returned in form of *List* object of type *Recognition*. The returned object from this method comprises the sign translation.

```java
    public List<Recognition> recognizeImage(Bitmap bitmap) {
        ByteBuffer byteBuffer = convertBitmapToByteBuffer(bitmap);
        if(quant){
            float[][] result = new float[1][labelList.size()];
            interpreter.run(byteBuffer, result);
            return getSortedResultFloat(result);
        } else {
            float [][] result = new float[1][labelList.size()];
            interpreter.run(byteBuffer, result);
            return getSortedResultFloat(result);
        }

    }
```

4.3.2 Authorization

Considering the user registration process, the user is required to input his name, email, password, and username. Each of these fields is implemented in a separate class and each class extends *ernestoyaquello.com.verticalstepperform.Step* class. The same methods and the same implementation pattern is applied in *EmailStep* class for email verification, *NameStep* class for getting full name of the use, *PasswordStep* for ensuring a strong and valid password is entered by user, and finally *UserNameStep* class for creating a unique username for each user. The code for each class is listed below.

EmailStep

```java
@Override
    protected IsDataValid isStepDataValid(String stepData) {
        // The step's data (i.e., the user name) will be considered valid only if it is longer
    than
        // three characters. In case it is not, we will display an error message for feedback.
        // In an optional step, you should implement this method to always return a valid value.
        boolean isEmailValid = !TextUtils.isEmpty(stepData) && android.util.Patterns.
    EMAIL_ADDRESS.matcher(stepData).matches();
        String errorMessage = !isEmailValid ? "Email not valid" : "";

        return new IsDataValid(isEmailValid, errorMessage);
    }

    @Override
    public String getStepData() {
        // We get the step's data from the value that the user has typed in the EditText view.
        Editable email = EmailView.getText();
        return email != null ? email.toString() : "";
    }
```

NameStep

```java
@Override
    protected IsDataValid isStepDataValid(String stepData) {
        // The step's data (i.e., the user name) will be considered valid only if it is longer
    than
        // three characters. In case it is not, we will display an error message for feedback.
        // In an optional step, you should implement this method to always return a valid value.
        boolean isNameValid = stepData.length() >= 12;
        String errorMessage = !isNameValid ? "12 characters minimum" : "";

        return new IsDataValid(isNameValid, errorMessage);
    }

    @Override
    public String getStepData() {
        // We get the step's data from the value that the user has typed in the EditText view.
        Editable name = NameView.getText();
        return name != null ? name.toString() : "";
    }
```

PasswordStep

```java
    @Override
    protected IsDataValid isStepDataValid(String stepData) {
        // The step's data (i.e., the user name) will be considered valid only if it is longer
    than
        // three characters. In case it is not, we will display an error message for feedback.
        // In an optional step, you should implement this method to always return a valid value.
        String pattern = "(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\\S+$).{8,}";
        boolean isPasswordValid = stepData.matches(pattern);
        String errorMessage = !isPasswordValid ? "Passord EX: aaZZa44@" : "";

        return new IsPasswordValid(isPasswordValid, errorMessage);
    }

    @Override
    public String getStepData() {
        // We get the step's data from the value that the user has typed in the EditText view.
        Editable password = PasswordView.getText();
        return password != null ? password.toString() : "";
    }
```

UserNameStep

```
@Override
    protected IsDataValid isStepDataValid(String stepData) {
        // The step's data (i.e., the user name) will be considered valid only if it is longer
    than
        // three characters. In case it is not, we will display an error message for feedback.
        // In an optional step, you should implement this method to always return a valid value.
        boolean isNameValid = !TextUtils.isEmpty(stepData) && android.util.Patterns.
    EMAIL_ADDRESS.matcher(stepData).matches();
        String errorMessage = !isNameValid ? "name@exm.com" : "";

        return new IsDataValid(isNameValid, errorMessage);
    }

    @Override
    public String getStepData() {
        // We get the step's data from the value that the user has typed in the EditText view.
        Editable userName = userNameView.getText();
        return userName != null ? userName.toString() : "";
    }
```

Verifying mobile phone number is more different and complex than verifying the name, username, password and email of the user. As a result phone number verification and confirmation is handled in a separate package. The class *MobileNumber* comprises the mobile number object's attributes it the application, meanwhile the *Confirmation* class is responsible for confirming that the user uses a valid phone number by sending a verification code to the user. The sending of verification below is implemented as below :

```
 private void sendVerificationCode(String number){
        Toast.makeText(getContext(), getString(R.string.sending_code)+"",Toast.LENGTH_SHORT).
    show();
        PhoneAuthProvider.getInstance().verifyPhoneNumber(
                number,
                60,
                TimeUnit.SECONDS,
                TaskExecutors.MAIN_THREAD,
                mCallBack
        );
    }

    private PhoneAuthProvider.OnVerificationStateChangedCallbacks
            mCallBack = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

        @Override
        public void onCodeSent(String s, PhoneAuthProvider.ForceResendingToken
    forceResendingToken) {
            super.onCodeSent(s, forceResendingToken);
            verificationid = s;
```

```java
        }

        @Override
        public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential) {
            String code = phoneAuthCredential.getSmsCode();
            if (code != null){
                binding.progress.setVisibility(View.VISIBLE);
                verifyCode(code);
            }
        }

        @Override
        public void onVerificationFailed(FirebaseException e) {

            Toast.makeText(getContext(),"there is "  + e.getMessage(),Toast.LENGTH_LONG).show();
            Log.e(TAG, "onVerificationFailed: " + e.getMessage() );

        }
    };
```

The confirmation requires that the user enter his own number, then the Firebase creates a temporal record for the user and creates a verification code that user must input to authorize him. The authorization is implemented by creating a credential with the verification code as listed below :

```java
private void verifyCode(String code){
        PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationid, code);
        signInWithCredential(credential);
    }
```

The authorization process is limited for 2 minutes only, if time is elapsed then the generated credential and verification code is deleted and the user must choose to resend the code again. The timer function is implemented as follows

```java
    private void timer() {
        binding.resendTx.setVisibility(View.GONE);
        countDownTimer = new CountDownTimer(120000, 1000) {
            public void onTick(long millisUntilFinished) {

                long timer = (millisUntilFinished / 1000);
                long min = timer / 60;

                long sec = timer - (min * 60);

                binding.timerTx.setText(getString(R.string.wait_we_will_resend_code_agian_after)
     + "\n " + String.format(min + "", "00") + ":" + String.format(sec + "", "00"));
            }

            public void onFinish() {
                binding.resendTx.setVisibility(View.VISIBLE);
                binding.progress.setVisibility(View.GONE);

            }
        };

        countDownTimer.start();
    }
```

After a successful registration step, the user can log in the application in any time. The log in process involves ensuring that the user is already registered to the Firebase and ensuring a valid username and password are entered. The following method of class *LogIn* do the main work as follows :

```java
public void ObserveLogIn() {
        String username = binding.usernameEt.getText().toString();
        String password = binding.passwordEt.getText().toString();
        String pattern = "(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\\S+$).{8,}";

        if (TextUtils.isEmpty(username) && android.util.Patterns.EMAIL_ADDRESS.matcher(username).matches()) {
            binding.usernameEt.setError("Invalid User name");
            return;
        }

        if (!password.matches(pattern)) {
            binding.passwordEt.setError("Invalid Password");
            return;
        }
        binding.progressBar.setVisibility(View.VISIBLE);

        mViewModel.LogIn(username, password).observe(this, new Observer<AuthResource<User>>() {
            @Override
            public void onChanged(AuthResource<User> userAuthResource) {
                if (userAuthResource != null) {
                    switch (userAuthResource.status) {
                        case NOT_AUTHENTICATED: {
                            binding.progressBar.setVisibility(View.GONE);
                        }

                        case AUTHENTICATED: {
                            binding.progressBar.setVisibility(View.GONE);
                            Intent intent = new Intent(getContext(), MainActivity.class);
                            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
                            startActivity(intent);
                            getActivity().finish();
                        }

                        case LOADING: {
                            binding.progressBar.setVisibility(View.VISIBLE);
                        }

                        case ERROR: {
                            binding.progressBar.setVisibility(View.GONE);
                            Toast.makeText(getContext(), userAuthResource.message, Toast.LENGTH_SHORT).show();
                        }
                    }
                }
            }
        });
    }
```

4.3.3 Chatting

The core of the application is chatting between a deaf person and the an ordinary person. The chat functionality is modeled in three classes. The first class is *Chat* which describes the essence of the chat functionality and how it is bined to the user interface. Second class is *ChatAdapter* which is a concrete class of *RecyclerView.Adapter<ChatAdapter.MyViewHolder>*; this class describes the inflating of the view objects on the current chat activity. The third and most critical class of the chat functionality is *ChatViewModel* this class is the backbone that the chat process relies on; it contains the methods which describes how message and other media is transferred from one user to another, how the translation is handled at both sides. In addition, this class also controls the synchronicity between Firebase data and the data displayed at each user. The following method retrieves the current user's data from Firebase :

```java
    public MutableLiveData<DataResource<User>> getCurrentUserDataResourceMutableLiveData() {
        currentUserDataResourceMutableLiveData.setValue(DataResource.loading((User) null));
        murmuroRepositoryImp.getUserById(firebaseAuth.getCurrentUser().getUid())
                .subscribeOn(Schedulers.io())
                .observeOn(AndroidSchedulers.mainThread())
                .onErrorReturn(new Function<Throwable, User>() {
                    @Override
                    public User apply(Throwable throwable) throws Exception {
                        User user = new User();
                        user.setId("-1");
                        return user;
                    }
                }).map(new Function<User, Object>() {
            @Override
            public Object apply(User user) throws Exception {
                if (user.getId().equals("-1")) {
                    currentUserDataResourceMutableLiveData.setValue(DataResource.error("can not
    load user", (User) null));
                    return null;
                }
                currentUserDataResourceMutableLiveData.setValue(DataResource.success(user));
                return user;
            }
        }).subscribe();
        return currentUserDataResourceMutableLiveData;
    }
```

When the user select a contact to view or begin a chat, the conversation is retrieved from the Firebase and displayed in the current chat activity, the method below do the job. A database reference is created to point to the proper conversation, then the conversation is downloaded. Note that Internet connection is required to accomplish this process.

```java
public MutableLiveData<DataResource<Conversation>> getConversationDataResourceMutableLiveData(
    String conversationId) {
        conversationDataResourceMutableLiveData.setValue(DataResource.loading((Conversation)
    null));
        if (isInternetAvailable()) {
            DatabaseReference databaseReference = firebaseDatabase.getReference().child("
    conversations").child(conversationId);
            databaseReference.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                    Log.e(TAG, "onDataChange: " + dataSnapshot.getValue().toString());
                    Conversation conversation = dataSnapshot.getValue(Conversation.class);
                    murmuroRepositoryImp.updateConversation(conversation);
                    conversationDataResourceMutableLiveData.setValue(DataResource.success(
    conversation));
                }
                @Override
                public void onCancelled(@NonNull DatabaseError databaseError) {
                    conversationDataResourceMutableLiveData.setValue(DataResource.error(
    databaseError.getMessage(), (Conversation) null));
                }
            });

        } else if (!isInternetAvailable()) {
            murmuroRepositoryImp.getConversationById(conversationId)
                    .subscribeOn(Schedulers.io())
                    .observeOn(AndroidSchedulers.mainThread())
                    .onErrorReturn(new Function<Throwable, Conversation>() {
                        @Override
                        public Conversation apply(Throwable throwable) throws Exception {
                            Conversation conversation = new Conversation();
                            conversation.setId("-1");
                            return null;
                        }
                    })
                    .map(new Function<Conversation, Object>() {
                        @Override
                        public Object apply(Conversation conversation) throws Exception {
                            if (conversation.getId().equals("-1")) {
                                conversationDataResourceMutableLiveData.setValue(DataResource.error("Can not load conversation", (Conversation) null));
                                return null;
                            }
                            conversationDataResourceMutableLiveData.setValue(DataResource.success(conversation));
                            return conversation;
                        }
                    }).subscribe();
        }
        return conversationDataResourceMutableLiveData;
    }
```

Different media ,like voice, images, videos, and other files, are handled differently than normal text messages. This involves locating/recording the media ; then a set of database references are created to synchronize the media states(seen, not seen, sent, failed,...) between users. This is achieved by the following method :

```java
public void sendStorageMessage(final Message message, final Person friendUser, final String conversatId, final long messagesSize) {
        StorageReference ref = null;

        if (message.getMessageType().equals("File")) {
            ref = firebaseStorage.getReference().child("files/" + conversatId + "/" + message.getDateTime() + message.getText());

        } else if (message.getMessageType().equals("Audio")) {
            ref = firebaseStorage.getReference().child("audios/" + conversatId + "/" + message.getDateTime() + message.getText());

        } else if (message.getMessageType().equals("Video")) {
            ref = firebaseStorage.getReference().child("videos/" + conversatId + "/" + message.getDateTime() + message.getText());

        } else if (message.getMessageType().equals("Gif")) {
            ref = firebaseStorage.getReference().child("gifs/" + conversatId + "/" + message.getDateTime() + message.getText());

        } else if (message.getMessageType().equals("Photo")) {
            ref = firebaseStorage.getReference().child("images/" + conversatId + "/" + message.getDateTime() + message.getText());
        }

        ref.putFile(Uri.parse(message.getPhoto()))
                .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                        Log.e(TAG, "onSuccess: Uploaded");
                        Toast.makeText(context, "Start Uploading", Toast.LENGTH_SHORT).show();

                        DatabaseReference databaseReference1 = firebaseDatabase.getReference()
```

```java
                            .child("conversations")
                            .child(conversatId)
                            .child("lastMessageId");

                    databaseReference1.setValue(messagesSize + "");

                    DatabaseReference databaseReference2 = firebaseDatabase.getReference()
                            .child("conversations")
                            .child(conversatId)
                            .child("messages")
                            .child(((messagesSize) + ""));

                    databaseReference2.setValue(message);

                    final DatabaseReference databaseReference3 = firebaseDatabase.getReference()
                            .child("conversations")
                            .child(conversatId)
                            .child("undreadMessages")
                            .child(friendUser.getId());


                    databaseReference3.addValueEventListener(new ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                            unreadMessages = dataSnapshot.getValue(Integer.class);
                            unreadMessages++;
                        }

                        @Override
                        public void onCancelled(@NonNull DatabaseError databaseError) {
                            Log.e(TAG, "onCancelled: " + "can not load unread messages");
                        }
                    });

                    databaseReference3.setValue(unreadMessages);

                    DatabaseReference databaseReference4 = firebaseDatabase.getReference()
                            .child("users")
                            .child(friendUser.getId())
                            .child("conversations")
                            .child(firebaseAuth.getCurrentUser().getUid())
                            .child("undreadMessages")
                            .child(friendUser.getId());

                    databaseReference4.setValue(unreadMessages);

                    DatabaseReference databaseReference5 = firebaseDatabase.getReference()
                            .child("users")
                            .child(friendUser.getId())
                            .child("conversations")
                            .child(firebaseAuth.getCurrentUser().getUid())
                            .child("displayMessage");

                    databaseReference5.setValue(message);

                    DatabaseReference databaseReference6 = firebaseDatabase.getReference()
                            .child("users")
                            .child(firebaseAuth.getCurrentUser().getUid())
                            .child("conversations")
                            .child(friendUser.getId())
                            .child("displayMessage");

                    databaseReference6.setValue(message);
                    Toast.makeText(context, "Uploaded Success", Toast.LENGTH_SHORT).show();
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Log.e(TAG, "onFailure: Failed");
                    Toast.makeText(context, "Uploaded : Failed", Toast.LENGTH_SHORT).show();
                }
            })
            .addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {
                    double progress = (100.0 * taskSnapshot.getBytesTransferred() / taskSnapshot
                            .getTotalByteCount());
                    Log.e(TAG, "onProgress: Uploaded" + (int) progress + "%");
                }
            });
}
```

Sending a text message is same as sending media in establishing the proper database references. However, it's simpler as it gets the input directly from the user from the keyboard and does not require locating external files or recording any data. The method below summarizes this process.

```java
public void sendTextMessage(final Message message, final Person friendUser, final String conversatId, long messagesSize) {
        DatabaseReference databaseReference1 = firebaseDatabase.getReference()
                .child("conversations")
                .child(conversatId)
                .child("lastMessageId");

        databaseReference1.setValue(messagesSize + "");

        DatabaseReference databaseReference2 = firebaseDatabase.getReference()
                .child("conversations")
                .child(conversatId)
                .child("messages")
                .child(((messagesSize) + ""));


        databaseReference2.setValue(message);

        final DatabaseReference databaseReference3 = firebaseDatabase.getReference()
                .child("conversations")
                .child(conversatId)
                .child("undreadMessages")
                .child(friendUser.getId());


        databaseReference3.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                unreadMessages = dataSnapshot.getValue(Integer.class);
                unreadMessages++;
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
                Log.e(TAG, "onCancelled: " + "can not load unread messages");
            }
        });

        databaseReference3.setValue(unreadMessages);

        DatabaseReference databaseReference4 = firebaseDatabase.getReference()
                .child("users")
                .child(friendUser.getId())
                .child("conversations")
                .child(firebaseAuth.getCurrentUser().getUid())
                .child("undreadMessages")
                .child(friendUser.getId());

        databaseReference4.setValue(unreadMessages);

        DatabaseReference databaseReference5 = firebaseDatabase.getReference()
                .child("users")
                .child(friendUser.getId())
                .child("conversations")
                .child(firebaseAuth.getCurrentUser().getUid())
                .child("displayMessage");

        databaseReference5.setValue(message);

        DatabaseReference databaseReference6 = firebaseDatabase.getReference()
                .child("users")
                .child(firebaseAuth.getCurrentUser().getUid())
                .child("conversations")
                .child(friendUser.getId())
                .child("displayMessage");

        databaseReference6.setValue(message);

    }
```

Interpolating the various functions of text and media exchanging, translating a message into a sign using avatar, capturing user's sign are done using a very long adapter method. This method is listed below as individual code snippets :
In this part, the database references are created to point to a target messages, and a page of the conversation is prepared to be displayed on the screen.

```java
    public MutableLiveData<DataResource<FirebaseRecyclerPagingAdapter<Message, ChatAdapter.
    MyViewHolder>>> getMessagesAdapter(final String conversationId, final Person friendUser,
    final LifecycleOwner lifecycleOwner) {

        DatabaseReference messagesReference = firebaseDatabase.getReference()
                .child("conversations")
                .child(conversationId)
                .child("messages");


        PagedList.Config config = new PagedList.Config.Builder()
                .setEnablePlaceholders(false)
                .setPrefetchDistance(5)
                .setPageSize(10)
                .build();

        DatabasePagingOptions<Message> options = new DatabasePagingOptions.Builder<Message>()
                .setLifecycleOwner(lifecycleOwner)
                .setQuery(messagesReference, config, Message.class)
                .build();

        FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder>
    firebaseRecyclerPagingAdapter = new FirebaseRecyclerPagingAdapter<Message, ChatAdapter.
    MyViewHolder>(options) {....}
        messagesAdapterDataResourceMutableLiveData.setValue(DataResource.success(
    firebaseRecyclerPagingAdapter));

        return messagesAdapterDataResourceMutableLiveData;
    }
```

The code of *FirebaseRecyclerPagingAdapter* is collapsed in the previous code snippet and is detailed in the next code fragments. *FirebaseRecyclerPagingation* Library binds Firebase Realtime Database Query to a *RecyclerView* by loading Data in pages. This is enable real-time data exchange and reliable message translation. Instantiating an object of *FirebaseRecyclerPagingAdapter* requires overriding the following-collapsed methods.

```
FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder> firebaseRecyclerPagingAdapter
    = new FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder>(options) {
            @Override
            protected void onBindViewHolder(@NonNull final ChatAdapter.MyViewHolder myViewHolder
    ,
                                                int i, @NonNull final Message message) {...}
            @Override
            protected void onLoadingStateChanged(@NonNull LoadingState state){...}

            @NonNull
            @Override
            public ChatAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
    viewType){...}

             @Override
            protected void onError(@NonNull DatabaseError databaseError){...}
    }
```

Note You can see the full implementation of *getMessagesAdapter* method in the appendix

4.3.4 Live Translation

A second important feature of the application is Live Translation. This feature is used when two users meet up and a translation from one language to the other is required. Live Translation is modeled in the application in the *LiveTranslation* class, which is responsible for inflating layout on the screen and allow the user to switch between sign language translation and English translation. As a normal user can talk directly instead of writing, a method is implemented to handle user's speech :

```
private void startVoiceInput() {
        Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.
    LANGUAGE_MODEL_FREE_FORM);
        intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, "en-US");
        intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Hello, Say your message?");
        try {
            startActivityForResult(intent, REQ_CODE_SPEECH_INPUT);
        } catch (ActivityNotFoundException a) {

        }
    }
```

Also, as the deaf user will use sign to communicate with his partner, a method is required for loading Neural Network model to be executable :

```
private void initTensorFlowAndLoadModel() {
        executor.execute(new Runnable() {
            @Override
            public void run() {
                try {
                    classifier = TensorFlowImageClassifier.create(
                            getActivity().getAssets(),
                            MODEL_PATH,
                            LABEL_PATH,
                            INPUT_SIZE,
                            QUANT);
                } catch (final Exception e) {
                    throw new RuntimeException("Error initializing TensorFlow!", e);
                }
            }
        });
    }
```

Putting engine components to work together and enabling user to switch between Avatar and model is accomplished by the overridden method below. The essence of this method is to bind application's activity with appropriate *Listeners* for each task, for example : running avatar, opening camera to capture sign, record audio from mic.

```
@Override
    public void onActivityCreated(@Nullable Bundle savedInstanceState){
        super.onActivityCreated(savedInstanceState);
        mViewModel = ViewModelProviders.of(this,providerFactory).get(LiveTranslationViewModel.
    class);
        if(savedInstanceState != null){
            Navigation.findNavController(getActivity(), R.id.host_fragment).restoreState(
    savedInstanceState.getBundle("nav_state"));
            Log.e(TAG, "onRestoreInstanceState: " +  savedInstanceState.getBundle("nav_state").
    describeContents());
        }

        binding.camera.setLifecycleOwner(this);

          handler = new Handler();
         runnable = new Runnable() {...};
        if(handler!= null && runnable != null){...}

        binding.camera.close();


        binding.camera.addCameraListener(new CameraListener(){...});

        initTensorFlowAndLoadModel();

        binding.toolbar.setNavigationOnClickListener(new View.OnClickListener(){...});

        binding.messageEditText.addTextChangedListener(new TextWatcher(){...});

        binding.sendImage.setOnClickListener(new View.OnClickListener(){...});

        binding.signTranslationButton.setOnClickListener(new View.OnClickListener(){...});
    }
```

Note You can see the full implementation of *onActivityCreated* method in the appendix

4.3.5 Application main

The entry point of the application is defined in *MainActivity* class which is a child class of *BaseActivity* class. Application initialization and layout inflating is defined in the following method :

```java
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = DataBindingUtil.setContentView(this , R.layout.activity_main);

        bottomNavigationView = binding.bottomNavigation;
        floatingActionButton_LiveTranslation = binding.liveTranslationFAB;

        bottomNavigationView.setOnNavigationItemSelectedListener(new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {

                switch (item.getItemId()) {
                    case R.id.conversations_item:
                        if(Navigation.findNavController(MainActivity.this, R.id.host_fragment).getCurrentDestination() != null){
                            navigate(0);
                        }
                        break;
                    case R.id.people_item:
                        if(Navigation.findNavController(MainActivity.this, R.id.host_fragment).getCurrentDestination() != null){
                            navigate(1);
                        }
                        break;
                return true;
            }
        });

        MainActivity.floatingActionButton_LiveTranslation.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if( Navigation.findNavController(MainActivity.this, R.id.host_fragment).getCurrentDestination().getId() == R.id.conversations){
                    Navigation.findNavController(MainActivity.this,R.id.host_fragment).navigate(ConversationsDirections.actionConversationsToLiveTranslation());
                }else if(Navigation.findNavController(MainActivity.this, R.id.host_fragment).getCurrentDestination().getId() == R.id.poeple)
                { Navigation.findNavController(MainActivity.this, R.id.host_fragment).navigate(PeopleDirections.actionPoepleToLiveTranslation());
                }

                MainActivity.bottomNavigationView.setVisibility(View.GONE);
                MainActivity.floatingActionButton_LiveTranslation.setVisibility(View.GONE);
            }
        });

    }
```

4.4 Actual database schema

5 Testing

5.1 Expected test scenarios

5.2 Unit test

5.3 Functional test

5.4 Usability test

6 Deployment of the system

7 Limitation of the system

8 Conclusion and further work

9 Appendix

9.1 *getMessagesAdapter* method

```java
public MutableLiveData<DataResource<FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder>>> getMessagesAdapter(final String conversationId, final Person friendUser, final LifecycleOwner lifecycleOwner) {

    DatabaseReference messagesReference = firebaseDatabase.getReference()
            .child("conversations")
            .child(conversationId)
            .child("messages");


    PagedList.Config config = new PagedList.Config.Builder()
            .setEnablePlaceholders(false)
            .setPrefetchDistance(5)
            .setPageSize(10)
            .build();

    DatabasePagingOptions<Message> options = new DatabasePagingOptions.Builder<Message>()
            .setLifecycleOwner(lifecycleOwner)
            .setQuery(messagesReference, config, Message.class)
            .build();


    FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder> firebaseRecyclerPagingAdapter =
            new FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder>(options) {
        @Override
        protected void onBindViewHolder(@NonNull final ChatAdapter.MyViewHolder myViewHolder,
                                        int i, @NonNull final Message message) {
            myViewHolder.bind(message);

            myViewHolder.itemView.findViewById(R.id.message_layout).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    if (myViewHolder.itemView.findViewById(R.id.message_options).getVisibility() != GONE) {
```

```java
            myViewHolder.itemView.findViewById(R.id.message_options).setVisibility(GONE);

          }
        }
      });

      myViewHolder.itemView.findViewById(R.id.message_layout).setOnLongClickListener(new View.OnLongClickListener() {
        @Override
        public boolean onLongClick(View v) {
          if (myViewHolder.itemView.findViewById(R.id.message_options).getVisibility() == GONE) {
            myViewHolder.itemView.findViewById(R.id.message_options).setVisibility(View.VISIBLE);

            if (message.getMessageType().equals("Text") || message.getMessageType().equals("Welcome")) {
              myViewHolder.itemView.findViewById(R.id.download_file_option).setVisibility(GONE);
            } else if (message.getMessageType().equals("Photo")) {
              myViewHolder.itemView.findViewById(R.id.translate_to_sign_option).setVisibility(GONE);
              myViewHolder.itemView.findViewById(R.id.convert_to_voice_message_option).setVisibility(GONE);
            } else if (message.getMessageType().equals("File")) {
              myViewHolder.itemView.findViewById(R.id.translate_to_sign_option).setVisibility(GONE);
              myViewHolder.itemView.findViewById(R.id.convert_to_voice_message_option).setVisibility(GONE);
            } else if (message.getMessageType().equals("Gif")) {
              myViewHolder.itemView.findViewById(R.id.translate_to_sign_option).setVisibility(GONE);
              myViewHolder.itemView.findViewById(R.id.convert_to_voice_message_option).setVisibility(GONE);
            } else if (message.getMessageType().equals("Audio")) {
              myViewHolder.itemView.findViewById(R.id.translate_to_sign_option).setVisibility(GONE);
              myViewHolder.itemView.findViewById(R.id.convert_to_voice_message_option).setVisibility(GONE);
            } else if (message.getMessageType().equals("Video")) {
              myViewHolder.itemView.findViewById(R.id.translate_to_sign_option).setVisibility(GONE);
              myViewHolder.itemView.findViewById(R.id.convert_to_voice_message_option).setVisibility(GONE);
            }


            if (!message.getSentBy().getId().equals(firebaseAuth.getCurrentUser().getUid())) {
              myViewHolder.itemView.findViewById(R.id.delete_message_option).setVisibility(GONE);
            } else {
              myViewHolder.itemView.findViewById(R.id.delete_message_option).setVisibility(VISIBLE);
            }

          } else {
            myViewHolder.itemView.findViewById(R.id.message_options).setVisibility(GONE);
          }
          return true;
        }
```

```java
            });
            if (!message.getStatus().equals("Seen") && !message.getSentBy().getId().equals(firebaseAuth.getCurrentUser().getUid())) {
                DatabaseReference messageSatatusDatabaseReference = firebaseDatabase
                        .getReference()
                        .child("conversations")
                        .child(conversationId)
                        .child("messages")
                        .child(message.getId())
                        .child("status");
                messageSatatusDatabaseReference.setValue("Seen");
            }


            DatabaseReference usersUnreadMessagesDatabaseReference = firebaseDatabase
                    .getReference()
                    .child("users")
                    .child(firebaseAuth.getCurrentUser().getUid())
                    .child("conversations")
                    .child(friendUser.getId())
                    .child("undreadMessages")
                    .child(firebaseAuth.getCurrentUser().getUid());

            usersUnreadMessagesDatabaseReference.setValue(0);

            LinearLayout sended_messageLinearLayout = myViewHolder.itemView.findViewById(R.id.sended_message);
            LinearLayout recieved_messageLinearLayout = myViewHolder.itemView.findViewById(R.id.recieved_message);
            TextView converstaion_time = myViewHolder.itemView.findViewById(R.id.converstaion_time);
            TextView converstaion_date = myViewHolder.itemView.findViewById(R.id.converstaion_date);


            if (!message.getId().equals("-1")) {
                if (!c_date.equals(message.getDateTime().toString().substring(6, 8))) {
                    converstaion_date.setText(message.getDateTime().toString().substring(6, 8) + " - " + message.getDateTime().toString().substring(4, 6));
                    c_date = message.getDateTime().toString().substring(6, 8);
                } else {
                    converstaion_date.setVisibility(View.GONE);
                }

                if (!c_time.equals(message.getDateTime().toString().substring(8, 10))) {

                    c_time = message.getDateTime().toString().substring(8, 10);

                    converstaion_time.setText(message.getDateTime().toString().substring(8, 10) + " : " + message.getDateTime().toString().substring(10, 12));
                } else {
                    converstaion_time.setVisibility(View.GONE);
                }
            }


            if (message.getSentBy().getId().equals(firebaseAuth.getCurrentUser().getUid()))
{
                recieved_messageLinearLayout.setVisibility(View.GONE);
                sended_messageLinearLayout.setVisibility(View.VISIBLE);


                ImageView message_statusImageView = myViewHolder.itemView.findViewById(R.id.
message_status);
                final MaterialTextView message_textMaterialTextView = myViewHolder.itemView.findViewById(R.id.s_message_text);
                final ImageView message_photoImageView = myViewHolder.itemView.findViewById(R.id.s_message_photo);
                final VideoView message_videoVideoView = myViewHolder.itemView.findViewById(R.id.s_message_video);
                LinearLayout message_audioLinearLayout = myViewHolder.itemView.findViewById(R.id.s_message_audio);
                final ImageView message_audio_playImageView = myViewHolder.itemView.findViewById(R.id.s_message_audio_play);
                final SeekBar message_audio_seekBarSeekBar = myViewHolder.itemView.findViewById(R.id.s_message_audio_seekBar);
                final MaterialTextView message_audio_timeMaterialTextView = myViewHolder.itemView.findViewById(R.id.s_message_audio_time);
                MaterialTextView message_timeMaterialTextView = myViewHolder.itemView.findViewById(R.id.s_message_time);
                LinearLayout file_layout = myViewHolder.itemView.findViewById(R.id.s_file_layout);
                final ImageView downloadFile = myViewHolder.itemView.findViewById(R.id.s_downloadFile);
                TextView fileName = myViewHolder.itemView.findViewById(R.id.s_fileName);

                if (message.getMessageType().equals("Text") || message.getMessageType().equals("Welcome")) {
                    message_textMaterialTextView.setText(message.getText());
                } else if (message.getMessageType().equals("Photo")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    message_photoImageView.setVisibility(VISIBLE);
                    firebaseStorage.getReference().child("images/" + conversationId + "/" + message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    requestManager.load(uri.toString()).into(message_photoImageView);
                                }
                            });
                } else if (message.getMessageType().equals("File")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    file_layout.setVisibility(VISIBLE);
                    fileName.setText(message.getText());

                } else if (message.getMessageType().equals("Video")) {
                    message_videoVideoView.setVisibility(VISIBLE);
                    message_textMaterialTextView.setVisibility(GONE);

                    firebaseStorage.getReference().child("videos/" + conversationId + "/" + message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {

                                    message_videoVideoView.setVideoURI(uri);
                                    message_textMaterialTextView.setVisibility(VISIBLE);
                                    message_textMaterialTextView.setText("Loading.");
                                    message_videoVideoView.requestFocus();
                                    message_videoVideoView.start();

                                    message_videoVideoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
                                        @Override
                                        public void onPrepared(MediaPlayer mp) {
                                            mp.setLooping(true);
                                        }
                                    });

                                    message_videoVideoView.setOnInfoListener(new MediaPlayer.OnInfoListener() {
                                        @Override
                                        public boolean onInfo(MediaPlayer mp, int what, int extra) {
                                            MediaController mediaController = null;
                                            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                                                mediaController = new MediaController(context);
                                            }
                                            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                                                message_videoVideoView.setMediaController(mediaController);
                                            }
                                            mediaController.setAnchorView(message_videoVideoView);

                                            if (what == MediaPlayer.MEDIA_INFO_BUFFERING_END) {
                                                message_textMaterialTextView.setVisibility(GONE);
                                                return true;
                                            } else if (what == MediaPlayer.MEDIA_INFO_BUFFERING_START) {
                                                message_textMaterialTextView.setVisibility(VISIBLE);
                                                message_textMaterialTextView.setText("Loading..");
                                                return true;
                                            }
                                            return false;
                                        }
                                    });

                                }
                            });

                } else if (message.getMessageType().equals("Audio")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    message_audioLinearLayout.setVisibility(VISIBLE);

                    firebaseStorage.getReference().child("audios/" + conversationId + "/" + message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    final MediaPlayer mp = new MediaPlayer();
                                    try {
                                        mp.setDataSource(context, uri);
                                        mp.prepare();
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }
                                    message_audio_seekBarSeekBar.setMax(mp.getDuration());

                                    final Handler mSeekbarUpdateHandler = new Handler();
                                    final Runnable mUpdateSeekbar = new Runnable() {
                                        @Override
                                        public void run() {
                                            message_audio_seekBarSeekBar.setProgress(mp.getCurrentPosition());
                                            mSeekbarUpdateHandler.postDelayed(this, 50);
                                        }
                                    };

                                    message_audio_timeMaterialTextView.setText(getTimeString(mp.getDuration()));
                                    message_audio_playImageView.setOnClickListener(new View.OnClickListener() {
                                        @Override
                                        public void onClick(View v) {
                                            if (mp.isPlaying()) {
                                                mp.pause();
                                                message_audio_seekBarSeekBar.removeCallbacks(mUpdateSeekbar);
                                                message_audio_playImageView.setImageResource(R.drawable.ic_play);
                                            } else {
                                                message_audio_playImageView.setImageResource(R.drawable.ic_pause);
                                                mp.start();
                                                mSeekbarUpdateHandler.postDelayed(mUpdateSeekbar, 0);
                                                Log.e(TAG, "onClick: " + mp.getCurrentPosition());
                                                Log.e(TAG, "onClick: " + mp.getDuration());
                                            }
                                        }
                                    });


                                    message_audio_seekBarSeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
                                        @Override
                                        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
                                            if (b)
                                                mp.seekTo(i);
                                            Log.e(TAG, "onProgressChanged: " + i);
                                        }

                                        @Override
                                        public void onStartTrackingTouch(SeekBar seekBar) {

                                        }

                                        @Override
                                        public void onStopTrackingTouch(SeekBar seekBar) {

                                        }
                                    });

                                }
                            });


                } else if (message.getMessageType().equals("Gif")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    message_photoImageView.setVisibility(VISIBLE);
                    firebaseStorage.getReference().child("gifs/" + conversationId + "/" + message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    requestManager.asGif().load(uri.toString()).into(message_photoImageView);
                                }
                            });
                }

                if (message.getId().equals("-1")) {
                    converstaion_time.setText(message.getDateTime().toString().substring(8, 10) + " : " + message.getDateTime().toString().substring(10, 12));
                    converstaion_date.setText(message.getDateTime().toString().substring(6, 8) + " - " + message.getDateTime().toString().substring(4, 6));
                    c_date = message.getDateTime().toString().substring(8, 10);
                    c_time = message.getDateTime().toString().substring(4, 6);
                }

                // Message options
                myViewHolder.itemView.findViewById(R.id.translate_to_sign_option).setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        if (message.getMessageType().equals("Text") || message.getMessageType().equals("Welcome")) {
                            String text = message.getText();
                            message_photoImageView.setVisibility(VISIBLE);

                            final List<String> words = new ArrayList<>();

                            String word = "";

                            for (int i = 0; i < text.length(); i++) {
                                if (text.charAt(i) == ' ') {
                                    Log.e(TAG, "onClick: add " + word);
                                    words.add(word);
                                    word = "";
                                } else {
                                    word += text.charAt(i);
                                }
                            }

                            if (!word.equals("")) {
                                words.add(word);
                                Log.e(TAG, "onClick: add " + word);
                            }


                            wordsHandler = new Handler();
                            wordsRunnable = new Runnable() {
                                @Override
                                public void run() {
                                    if (wordsIndex < words.size()) {
                                        firebaseStorage.getReference().child("Signs/" + words.get(wordsIndex) + ".gif").getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                                            @Override
                                            public void onSuccess(Uri uri) {
                                                requestManager.asGif().load(uri.toString()).into(message_photoImageView);
                                                if (wordsIndex < words.size()) {
                                                    Log.e(TAG, "onSuccess: loaded a " + words.get(wordsIndex));
                                                }
                                            }
                                        });
                                        wordsIndex++;
                                    } else {
                                        wordsIndex = 0;
                                        wordsHandler = null;
                                        wordsRunnable = null;
                                        message_photoImageView.setVisibility(GONE);
                                        myViewHolder.itemView.findViewById(R.id.message_options).setVisibility(GONE);
                                    }

                                    if (wordsHandler != null || wordsRunnable != null) {
                                        wordsHandler.postDelayed(this, 1000);
                                    }
                                }
                            };
                            if (wordsHandler != null || wordsRunnable != null) {
                                wordsHandler.postDelayed(wordsRunnable, 1000);
                            }

                        }
                    }
                });


                message_timeMaterialTextView.setText(
                        message.getDateTime().toString().substring(8, 10) + " : " + message.getDateTime().toString().substring(10, 12)
                );

                t1 = new TextToSpeech(context, new TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status != TextToSpeech.ERROR) {
                            t1.setLanguage(Locale.UK);
                        }
                    }
                });

                myViewHolder.itemView.findViewById(R.id.convert_to_voice_message_option).setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        t1.speak(message.getText(), TextToSpeech.QUEUE_FLUSH, null);
                    }
                });

                myViewHolder.itemView.findViewById(R.id.download_file_option).setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        StorageReference ref = null;

                        String filePath = "";

                        if (message.getMessageType().equals("File")) {
```

```java
                            ref = firebaseStorage.getReference().child("files/" + conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Files";
                        } else if (message.getMessageType().equals("Audio")) {
                            ref = firebaseStorage.getReference().child("audios/" + conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Audios";
                        } else if (message.getMessageType().equals("Video")) {
                            ref = firebaseStorage.getReference().child("videos/" + conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Videos";
                        } else if (message.getMessageType().equals("Gif")) {
                            ref = firebaseStorage.getReference().child("gifs/" + conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Gifs";
                        } else if (message.getMessageType().equals("Photo")) {
                            ref = firebaseStorage.getReference().child("images/" + conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Photos";
                        }


                        File fileNameOnDevice = null;


                        final File folder = new File(Environment.getExternalStorageDirectory() +
                                File.separator + filePath);
                        boolean success = true;
                        if (!folder.exists()) {
                            success = folder.mkdirs();
                        }
                        if (success) {
                            // Do something on success
                            fileNameOnDevice = new File(folder + "/" + message.getText());
                        } else {
                            // Do something else on failure
                        }

                        Toast.makeText(context, "Start Downloading", Toast.LENGTH_SHORT).show();

                        ref.getFile(fileNameOnDevice).addOnSuccessListener(new OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
                            @Override
                            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                                Log.e(TAG, "onSuccess: downloaded in " + folder.getName());
                                Toast.makeText(context, "downloaded in " + folder.getName() + " folder", Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                });


                if (message.getStatus().equals("Seen")) {
                    message_statusImageView.setImageResource(R.drawable.ic_online);
                } else if (message.getStatus().equals("Arrived")) {
                    message_statusImageView.setImageResource(R.drawable.ic_away);

                } else if (message.getStatus().equals("Sended")) {
                    message_statusImageView.setImageResource(R.drawable.ic_busy);

                }

            } else if (message.getSentBy().getId().equals(friendUser.getId())) {
                recieved_messageLinearLayout.setVisibility(View.VISIBLE);
                sended_messageLinearLayout.setVisibility(View.GONE);

                final MaterialTextView message_textMaterialTextView = myViewHolder.itemView.findViewById(R.id.message_text);
                final ImageView message_photoImageView = myViewHolder.itemView.findViewById(R.id.message_photo);
                final VideoView message_videoVideoView = myViewHolder.itemView.findViewById(R.id.message_video);
                LinearLayout message_audioLinearLayout = myViewHolder.itemView.findViewById(R.id.message_audio);
                final ImageView message_audio_playImageView = myViewHolder.itemView.findViewById(R.id.message_audio_play);
                final SeekBar message_audio_seekBarSeekBar = myViewHolder.itemView.findViewById(R.id.message_audio_seekBar);
                final MaterialTextView message_audio_timeMaterialTextView = myViewHolder.itemView.findViewById(R.id.message_audio_time);
                MaterialTextView message_timeMaterialTextView = myViewHolder.itemView.findViewById(R.id.message_time);
                LinearLayout file_layout = myViewHolder.itemView.findViewById(R.id.file_layout);
                ImageView dowenloadFile = myViewHolder.itemView.findViewById(R.id.downloadFile);
                TextView fileName = myViewHolder.itemView.findViewById(R.id.fileName);

                if (message.getMessageType().equals("Text") || message.getMessageType().equals("Welcome")) {
                    message_textMaterialTextView.setText(message.getText());
                } else if (message.getMessageType().equals("Photo")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    message_photoImageView.setVisibility(VISIBLE);
                    firebaseStorage.getReference().child("images/" + conversationId + "/" + message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    requestManager.load(uri.toString()).into(message_photoImageView);
                                }
                            });
                } else if (message.getMessageType().equals("File")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    file_layout.setVisibility(VISIBLE);
                    fileName.setText(message.getText());

                } else if (message.getMessageType().equals("Video")) {
                    message_videoVideoView.setVisibility(VISIBLE);
                    message_textMaterialTextView.setVisibility(GONE);

                    firebaseStorage.getReference().child("videos/" + conversationId + "/" + message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {

                                    message_videoVideoView.setVideoURI(uri);
                                    message_textMaterialTextView.setVisibility(VISIBLE);
                                    message_textMaterialTextView.setText("Loading.");
                                    message_videoVideoView.requestFocus();
                                    message_videoVideoView.start();

                                    message_videoVideoView.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {
                                        @Override
                                        public void onPrepared(MediaPlayer mp) {
                                            mp.setLooping(true);
                                        }
                                    });

                                    message_videoVideoView.setOnInfoListener(new MediaPlayer.
OnInfoListener() {
                                        @Override
                                        public boolean onInfo(MediaPlayer mp, int what, int
extra) {
                                            MediaController mediaController = null;
                                            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.
LOLLIPOP) {
                                                mediaController = new MediaController(
context);
                                            }
                                            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.
LOLLIPOP) {
                                                message_videoVideoView.setMediaController(
mediaController);
                                            }
                                            mediaController.setAnchorView(
message_videoVideoView);

                                            if (what == MediaPlayer.MEDIA_INFO_BUFFERING_END
) {
                                                message_textMaterialTextView.setVisibility(
GONE);
                                                return true;
                                            } else if (what == MediaPlayer.
MEDIA_INFO_BUFFERING_START) {
                                                message_textMaterialTextView.setVisibility(
VISIBLE);
                                                message_textMaterialTextView.setText("
Loading..");
                                                return true;
                                            }
                                            return false;
                                        }
                                    });

                                }
                            });

                } else if (message.getMessageType().equals("Audio")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    message_audioLinearLayout.setVisibility(VISIBLE);

                    firebaseStorage.getReference().child("audios/" + conversationId + "/" +
message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    final MediaPlayer mp = new MediaPlayer();
                                    try {
```

```java
                                        mp.setDataSource(context, uri);
                                        mp.prepare();
                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }
                                    message_audio_seekBarSeekBar.setMax(mp.getDuration());

                                    final Handler mSeekbarUpdateHandler = new Handler();
                                    final Runnable mUpdateSeekbar = new Runnable() {
                                        @Override
                                        public void run() {
                                            message_audio_seekBarSeekBar.setProgress(mp.getCurrentPosition());
                                            mSeekbarUpdateHandler.postDelayed(this, 50);
                                        }
                                    };

                                    message_audio_timeMaterialTextView.setText(getTimeString(mp.getDuration()));
                                    message_audio_playImageView.setOnClickListener(new View.OnClickListener() {
                                        @Override
                                        public void onClick(View v) {
                                            if (mp.isPlaying()) {
                                                mp.pause();
                                                message_audio_seekBarSeekBar.removeCallbacks(mUpdateSeekbar);
                                                message_audio_playImageView.setImageResource(R.drawable.ic_play);
                                            } else {
                                                message_audio_playImageView.setImageResource(R.drawable.ic_pause);
                                                mp.start();
                                                mSeekbarUpdateHandler.postDelayed(mUpdateSeekbar, 0);
                                                Log.e(TAG, "onClick: " + mp.getCurrentPosition());
                                                Log.e(TAG, "onClick: " + mp.getDuration());
                                            }
                                        }
                                    });


                                    message_audio_seekBarSeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
                                        @Override
                                        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
                                            if (b)
                                                mp.seekTo(i);
                                            Log.e(TAG, "onProgressChanged: " + i);
                                        }

                                        @Override
                                        public void onStartTrackingTouch(SeekBar seekBar) {

                                        }

                                        @Override
                                        public void onStopTrackingTouch(SeekBar seekBar) {

                                        }
                                    });
```

```

                                }
                            });


                } else if (message.getMessageType().equals("Gif")) {
                    message_textMaterialTextView.setVisibility(GONE);
                    message_photoImageView.setVisibility(VISIBLE);
                    firebaseStorage.getReference().child("gifs/" + conversationId + "/" + message.getDateTime() + message.getText())
                            .getDownloadUrl()
                            .addOnSuccessListener(new OnSuccessListener<Uri>() {
                                @Override
                                public void onSuccess(Uri uri) {
                                    requestManager.asGif().load(uri.toString()).into(message_photoImageView);
                                }
                            });
                }


                if (message.getId().equals("-1")) {
                    converstaion_time.setText(message.getDateTime().toString().substring(8, 10) + " : " + message.getDateTime().toString().substring(10, 12));
                    converstaion_date.setText(message.getDateTime().toString().substring(6, 8) + " - " + message.getDateTime().toString().substring(4, 6));
                    c_date = message.getDateTime().toString().substring(8, 10);
                    c_time = message.getDateTime().toString().substring(4, 6);
                }

                // Message options

                myViewHolder.itemView.findViewById(R.id.translate_to_sign_option).setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        if (message.getMessageType().equals("Text") || message.getMessageType().equals("Welcome")) {
                            String text = message.getText();
                            message_photoImageView.setVisibility(VISIBLE);

                            final List<String> words = new ArrayList<>();

                            String word = "";

                            for (int i = 0; i < text.length(); i++) {
                                if (text.charAt(i) == ' ') {
                                    Log.e(TAG, "onClick: add " + word);
                                    words.add(word);
                                    word = "";
                                } else {
                                    word += text.charAt(i);
                                }
                            }

                            if (!word.equals("")) {
                                words.add(word);
                                Log.e(TAG, "onClick: add " + word);
                            }


                            wordsHandler = new Handler();
                            wordsRunnable = new Runnable() {
```

```java
                                @Override
                                public void run() {
                                    if (wordsIndex < words.size()) {
                                        firebaseStorage.getReference().child("Signs/" +
words.get(wordsIndex) + ".gif").getDownloadUrl().addOnSuccessListener(new OnSuccessListener<
Uri>() {
                                            @Override
                                            public void onSuccess(Uri uri) {
                                                requestManager.asGif().load(uri.toString()).
into(message_photoImageView);
                                                if (wordsIndex < words.size()) {
                                                    Log.e(TAG, "onSuccess: loaded a " +
words.get(wordsIndex));
                                                }
                                            }
                                        });
                                        wordsIndex++;
                                    } else {
                                        wordsIndex = 0;
                                        wordsHandler = null;
                                        wordsRunnable = null;
                                        message_photoImageView.setVisibility(GONE);
                                        myViewHolder.itemView.findViewById(R.id.
message_options).setVisibility(GONE);
                                    }

                                    if (wordsHandler != null || wordsRunnable != null) {
                                        wordsHandler.postDelayed(this, 1000);
                                    }
                                }
                            };
                            if (wordsHandler != null || wordsRunnable != null) {
                                wordsHandler.postDelayed(wordsRunnable, 1000);
                            }

                        }
                    }
                });

                message_timeMaterialTextView.setText(
                        message.getDateTime().toString().substring(8, 10) + " : " + message.
getDateTime().toString().substring(10, 12)
                );


                t1 = new TextToSpeech(context, new TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status != TextToSpeech.ERROR) {
                            t1.setLanguage(Locale.UK);
                        }
                    }
                });

                t1 = new TextToSpeech(context, new TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status != TextToSpeech.ERROR) {
                            t1.setLanguage(Locale.UK);
                        }
                    }
                });

                myViewHolder.itemView.findViewById(R.id.convert_to_voice_message_option).
setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        t1.speak(message.getText(), TextToSpeech.QUEUE_FLUSH, null);
                    }
                });

                message_timeMaterialTextView.setText(
                        message.getDateTime().toString().substring(8, 10) + " : " + message.
getDateTime().toString().substring(10, 12)
                );


                myViewHolder.itemView.findViewById(R.id.download_file_option).
setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        StorageReference ref = null;

                        String filePath = "";

                        if (message.getMessageType().equals("File")) {
                            ref = firebaseStorage.getReference().child("files/" +
conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Files";
                        } else if (message.getMessageType().equals("Audio")) {
                            ref = firebaseStorage.getReference().child("audios/" +
conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Audios";
                        } else if (message.getMessageType().equals("Video")) {
                            ref = firebaseStorage.getReference().child("videos/" +
conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Videos";
                        } else if (message.getMessageType().equals("Gif")) {
                            ref = firebaseStorage.getReference().child("gifs/" +
conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Gifs";
                        } else if (message.getMessageType().equals("Photo")) {
                            ref = firebaseStorage.getReference().child("images/" +
conversationId + "/" + message.getDateTime() + message.getText());
                            filePath = "Murmuro/Photos";
                        }


                        File fileNameOnDevice = null;


                        final File folder = new File(Environment.getExternalStorageDirectory
() +
                                File.separator + filePath);
                        boolean success = true;
                        if (!folder.exists()) {
                            success = folder.mkdirs();
                        }
                        if (success) {
                            // Do something on success
                            fileNameOnDevice = new File(folder + "/" + message.getText());
                        } else {
                            // Do something else on failure
                        }

                        Toast.makeText(context, "Start Downloading", Toast.LENGTH_SHORT).
```

```java
show();

                        ref.getFile(fileNameOnDevice).addOnSuccessListener(new OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
                            @Override
                            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                                Log.e(TAG, "onSuccess: downloaded in " + folder.getName());
                                Toast.makeText(context, "downloaded in " + folder.getName() + " folder", Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                });

            }


            myViewHolder.itemView.findViewById(R.id.delete_message_option).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    DatabaseReference databaseReference2 = firebaseDatabase.getReference()
                            .child("conversations")
                            .child(conversationId)
                            .child("messages")
                            .child(message.getId());

                    databaseReference2.setValue(null);
                }
            });

        }

        @Override
        protected void onLoadingStateChanged(@NonNull LoadingState state) {
            switch (state) {
                case LOADING_INITIAL:
                case LOADING_MORE:
                    // Do your loading animation
                    messagesAdapterDataResourceMutableLiveData.setValue(DataResource.loading((FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder>) null));
                    break;

                case LOADED:
                    // Stop Animation
                    messagesAdapterDataResourceMutableLiveData.setValue(DataResource.error("LOADED", (FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder>) null));
                    break;

                case FINISHED:
                    //Reached end of Data set
                    messagesAdapterDataResourceMutableLiveData.setValue(DataResource.error("FINISHED", (FirebaseRecyclerPagingAdapter<Message, ChatAdapter.MyViewHolder>) null));

                    break;

                case ERROR:
                    retry();
                    break;
            }
        }

```

```
            @NonNull
            @Override
            public ChatAdapter.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
                LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());
                ViewDataBinding binding = DataBindingUtil.inflate(layoutInflater, R.layout.chat_item, parent, false);
                return new ChatAdapter.MyViewHolder(binding);
            }


            @Override
            protected void onError(@NonNull DatabaseError databaseError) {
                super.onError(databaseError);
                databaseError.toException().printStackTrace();
            }
        };

        messagesAdapterDataResourceMutableLiveData.setValue(DataResource.success(firebaseRecyclerPagingAdapter));

        return messagesAdapterDataResourceMutableLiveData;
    }
```

9.2 *onActivityCreated* method

```
     @Override
    public void onActivityCreated(@Nullable Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        mViewModel = ViewModelProviders.of(this,providerFactory).get(LiveTranslationViewModel.class);
        if(savedInstanceState != null)
        {
            Navigation.findNavController(getActivity(), R.id.host_fragment).restoreState(savedInstanceState.getBundle("nav_state"));
            Log.e(TAG, "onRestoreInstanceState: " +  savedInstanceState.getBundle("nav_state").describeContents());
        }

        binding.camera.setLifecycleOwner(this);

          handler = new Handler();
         runnable = new Runnable() {
            @Override
            public void run() {
               // binding.camera.setFilter(Filters.GRAYSCALE.newInstance());
                binding.camera.setAudio(Audio.OFF);
                binding.camera.takePicture();
                if(handler!= null && runnable != null)
                {
                    handler.postDelayed(this, 2000);
                }
            }
        };
        if(handler!= null && runnable != null)
        {
            handler.postDelayed(runnable, 2000);
        }


        binding.camera.close();


        binding.camera.addCameraListener(new CameraListener() {
```

```
        @Override
        public void onPictureTaken(final PictureResult result) {
            result.toBitmap(40, 40, new BitmapCallback() {
                @SuppressLint("WrongThread")
                @Override
                public void onBitmapReady(@Nullable Bitmap bitmap) {

                    bitmap = Bitmap.createScaledBitmap(bitmap, INPUT_SIZE, INPUT_SIZE, false
);
                    final List<Classifier.Recognition> results = classifier.recognizeImage(
bitmap);

                    Log.d(TAG, "oraby onBitmapReady: " + results.toString());
                    String message = "";

                    for(int i=0; i<results.size();i++)
                    {
                        message +=  results.get(i).getTitle() + " ";
                    }
                    binding.translatedText.setText(binding.translatedText.getText() +
message);

                }
            });


        }
    });

    initTensorFlowAndLoadModel();


    binding.toolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Navigation.findNavController(getActivity(), R.id.host_fragment).popBackStack();
        }
    });


    binding.avtarButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            binding.camera.setVisibility(View.GONE);
            binding.camera.close();
            handler = null;
            runnable = null;

            binding.avtarLayout.setVisibility(View.VISIBLE);
            binding.editTextLayout.setVisibility(View.VISIBLE);
            binding.signTranslationButton.setVisibility(View.VISIBLE);
            binding.translatedText.setVisibility(View.GONE);
            binding.avtarButton.setVisibility(View.GONE);

        }
    });

    binding.messageEditText.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

```

```java
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {


            }

            @Override
            public void afterTextChanged(Editable s) {
                if(s.toString().equals(""))
                {
                    binding.sendImage.setImageResource(R.drawable.ic_microphone);
                }else
                {
                    binding.sendImage.setImageResource(R.drawable.ic_send);
                }
            }
        });

        binding.sendImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(!binding.messageEditText.getText().toString().trim().equals(""))
                {
                    String text = binding.messageEditText.getText().toString().trim();

                    final List<String> words = new ArrayList<>();

                    String word = "";

                    for(int i=0; i < text.length();i++)
                    {
                        if(text.charAt(i) == ' ')
                        {
                            Log.e(TAG, "onClick: add " + word );
                            words.add(word);
                            word = "";
                        }else
                        {
                            word += text.charAt(i);
                        }
                    }

                    if(!word.equals(""))
                    {
                        words.add(word);
                        Log.e(TAG, "onClick: add " + word );
                    }


                    wordsHandler = new Handler();
                    wordsRunnable = new Runnable() {
                        @Override
                        public void run() {
                            if(wordsIndex < words.size())
                            {
                                firebaseStorage.getReference().child("Signs/" + words.get(wordsIndex) + ".gif").getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
                                    @Override
                                    public void onSuccess(Uri uri) {
                                        requestManager.asGif().load(uri.toString()).into(binding.avtarImageView);
                                        if(wordsIndex < words.size())
                                        {
```

```
                                        Log.e(TAG, "onSuccess: loaded a " + words.get(wordsIndex) );
                                    }
                                }
                            });
                            wordsIndex++;
                        }else
                        {
                            wordsIndex = 0;
                            wordsHandler = null;
                            wordsRunnable = null;
                        }

                        if(wordsHandler != null || wordsRunnable != null)
                        {
                            wordsHandler.postDelayed(this, 1000);
                        }
                    }
                };
                if(wordsHandler != null || wordsRunnable != null)
                {
                    wordsHandler.postDelayed(wordsRunnable, 1000);
                }

            }else
            {
                startVoiceInput();
            }
        }
    });

    binding.signTranslationButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            binding.avtarLayout.setVisibility(View.GONE);
            binding.editTextLayout.setVisibility(View.GONE);
            binding.signTranslationButton.setVisibility(View.GONE);
            binding.translatedText.setVisibility(View.VISIBLE);
            binding.avtarButton.setVisibility(View.VISIBLE);



            binding.camera.open();
            binding.camera.setVisibility(View.VISIBLE);
            binding.camera.setLifecycleOwner(LiveTranslation.this);

            handler = new Handler();
            runnable = new Runnable() {
                @Override
                public void run() {
                    // binding.camera.setFilter(Filters.GRAYSCALE.newInstance());
                    binding.camera.setAudio(Audio.OFF);
                    binding.camera.takePicture();
                    if(handler!= null && runnable != null) {
                        handler.postDelayed(this, 2000);
                    }
                }
            };
            if(handler!= null && runnable != null) {
                handler.postDelayed(runnable, 2000);
            }

```

```java

            binding.camera.addCameraListener(new CameraListener() {
                @Override
                public void onPictureTaken(final PictureResult result) {
                    result.toBitmap(40, 40, new BitmapCallback() {
                        @SuppressLint("WrongThread")
                        @Override
                        public void onBitmapReady(@Nullable Bitmap bitmap) {

                            bitmap = Bitmap.createScaledBitmap(bitmap, INPUT_SIZE, INPUT_SIZE, false);
                            if(bitmap != null)
                            {
                                final List<Classifier.Recognition> results = classifier.recognizeImage(bitmap);

                                Log.d(TAG, "oraby onBitmapReady: " + results.toString());

                                String message = "";

                                for(int i=0; i<results.size();i++)
                                {
                                    message +=  results.get(i).getTitle() + " ";
                                }

                                binding.messageEditText.setText(binding.messageEditText.getText() + message);
                            }

                        }
                    });


                }
            });

            initTensorFlowAndLoadModel();
        }
    });


}
```

Références

[1] B Dorner. « Hand shape identification and tracking for sign language interpretation ». In : *IJCAI'93 Looking at people Workshop*. 1993.

[2] Jason Andre Gilbert et Shau-yuh YU. *Sign language translation system*. US Patent App. 12/167,978. Jan. 2009.

[3] Sami M Halawani. « Arabic sign language translation system on mobile devices ». In : *IJCSNS International Journal of Computer Science and Network Security* 8.1 (2008), p. 251-256.

[4] Tomoichi Takahashi et Fumio Kishino. « Hand gesture coding based on experiments using a hand gesture interface device ». In : *Acm Sigchi Bulletin* 23.2 (1991), p. 67-74.