VIRTUAL CINEMA

IO3040 Software

Coach: Jacky Bourgeois

Date: 19-4-2018

Group 10

Ferhat Dursun 4238958

Yu Zhang

Janwillem Loonen 4162286

1. Introduction

Due to the development of technology it has become increasingly more easy to stay in touch with people at a distance. This causes people to have less hesitations about moving away to another part of the world or maintaining contact with people they met far away from home. Because of this people spend a large amount of time online with others spread all over the world, causing a need for something to aid in this interaction.

When communicating online the options for shared activities are far more limited than meeting offline and even though talking is important, it is far from the only way people usually bond. There is even a surprisingly large barrier to entry when it comes to seeking out contact just to talk with somebody, because there might not be enough to talk about. By providing an activity that people can easily default to when they run out of things to talk about, that by itself creates new topics of discussion, the barrier to entry gets lowered by a large amount.

The consumption of media and video in particular is the perfect activity in this case. Even though most people won't admit it a large part of the way we identify ourselves in this era is by the type of media we consume and we want to experience this with other people. Currently some of the most successful youtube videos consist of youtubers reacting to other videos, simply so we can identify with these "youtube personalities."

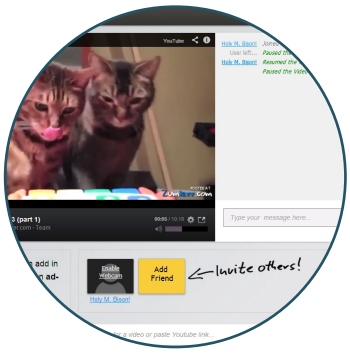
Therefor the goal of this project should be to aid people in the process of communicating online while watching the same video simultaneously. This goal can be broken down in certain tasks that need to be accomplished for this to work.

-There needs to be a means of communication between the two users, preferably video chat.
-The app needs to be able to play a video.
-The video should be synchronized on both devices.
-There has to be a way to select different videos.

2. Analysis

2.1 Functional exploration and analysis

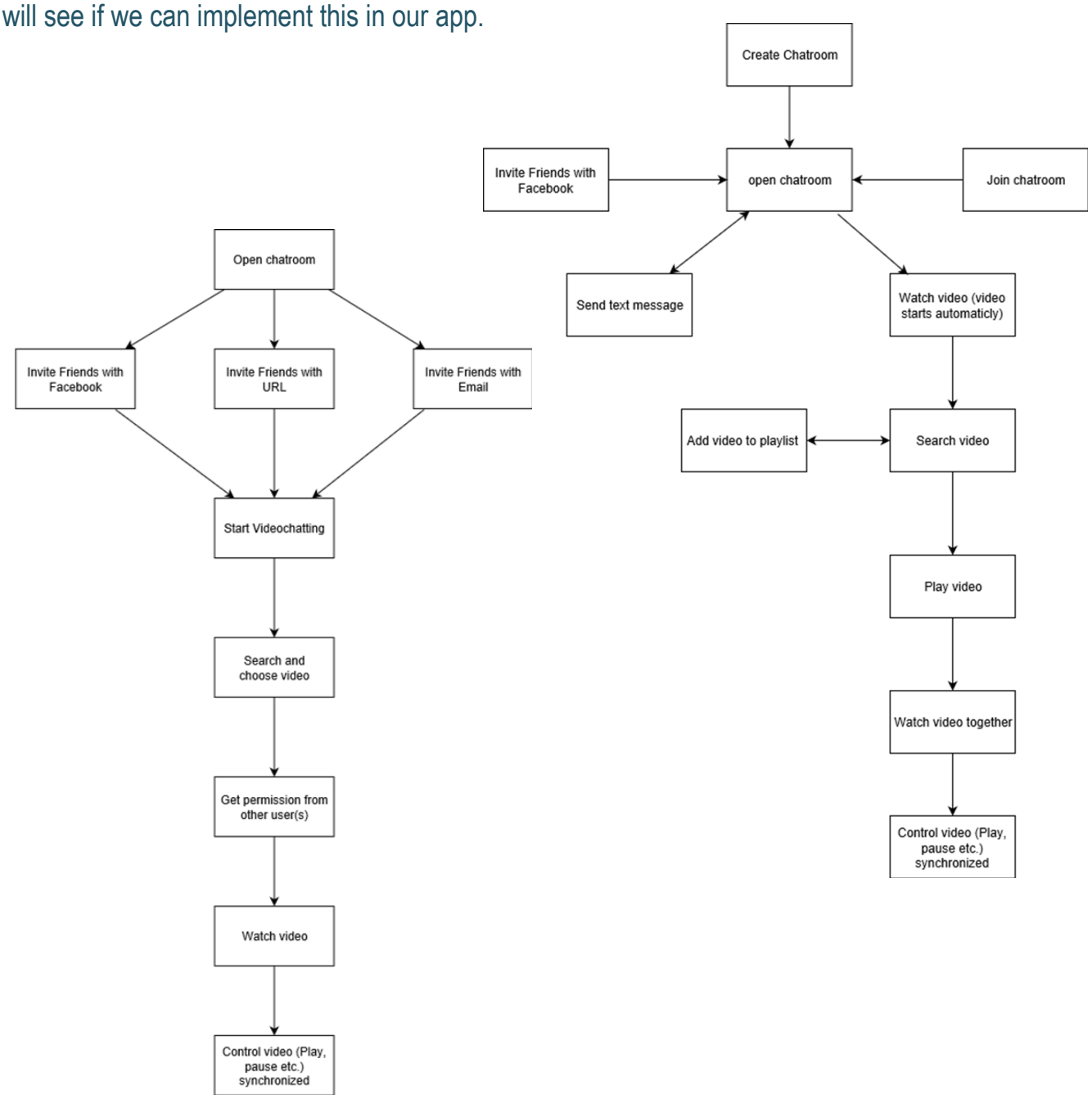
Before building our app, we have first determined which functions we need to implement in our app. These were the video watching, video chatting and video synchronizing functions. After determining these functions, firstly we have searched for apps that have all of these functions together. After a long search we actually could not find apps for phones that enables watching movies together however, there were many sites and computer programs to do this. One of these is Synaptop. This one is actually the most extensive one. Synaptop is a program for your pc that enables people to connect with each other via video and message chat. In addition to this, users can also watch movies together. Different than our app in mind, with Synaptop users can also play games, make homework, write letter, prepare presentations, read books and listen music together. All of these functions are possible to add in the form of an app to the main program.



Another option was watch2gether.com, this is a website that enables users to watch videos together online. It is not possible to video chat with this option, chatting is only possible by text messages. There are some functions in this which could be useful for our app. One of those is the video search function. This function enables to search videos on video watching websites like YouTube, Vimeo, GfyCat and Dailymotion. This is really useful and handy tool for users to choose videos and watch, which we could implement in our app. Another function is the playlist option, users can put videos on this list which will be played when the previous video will be finished. This might be useful for us in a later stadium, but in the first instance we will be focussing on much more important functions. The chatting function is done with text messaging with using a chatroom. You can invite other users to this chatroom by sending a URL to them.

The last one I want to mention is letsgaze.com. This is also a website which enables users to watch videos together online like watch2gether.com. There are only some little differences between these two websites. Like watch2gether.com you have to create a room to begin and share your URL with other users to begin, but unlike watch2gether.com the chatting functions is done by video chatting instead of text messaging. This is more useful in our case because in this way you can show your real emotions this way rather than sending emojis. Another difference is how you choose the video to watch. With watch2gether the users have 2 options; Either they have to copy and paste a YouTube link or they need to have the same video file on both of the computers downloaded which they can choose.

If we look at the last 2 existing programs we can see some of useful solutions that we might use in our app and are relevant for us. One of those functions is the external video player option. It is actually too hard and uncommon to have videos on your smartphone these days. Everyone is actually watching all the videos online. In this way you have to wait before both users have downloaded the video before watching the video, which might take a little time therefore, external video players are very useful. It will be hard for us to add more than one external video player therefore we will be using only 1 external video player. Related to this the function of searching videos in these external video players is also relevant for us, because the users need to get access to a video which they will choose by themselves. This can be done with typing the URL or implementing a search panel. This function is really useful and we will see if we can implement this in our app.



2.2 Functional exploration and analysis

Firstly the use of an online database is essential in this project, because the phones need to connect with each other from a large distance. Another option for the demo would have been to use bluetooth, but the first choice was to try to implement a real time online database, because this was ideally how the final app would work as well. The database is used to synchronize the video that is played on both phones and is part of the initial design challenge. There are countless options to complete this task, such as openfire, quickblox, phpmyadmin and firebase. The choice for this was based on how easy different libraries were to work with, this will be explained in more detail in chapter 4.4. Firebase ended up being the first one of these choices that functioned in the desired way.

A server is also required for video chatting, which is not essential in this assignment but will greatly improve the application. This could be handled by firebase as well, but it is far easier to use existing code and libraries to make this work compared to building it from scratch. There are several options available that can easily implement video chat in android, such as Qmunicate, Tokbox and Vidyo. The choice here was once again for the one that was easiest to implement, this being vidyo, which only needs a few lines of code to function. Obviously to enable the video chat to work, a camera is also required on both phones.

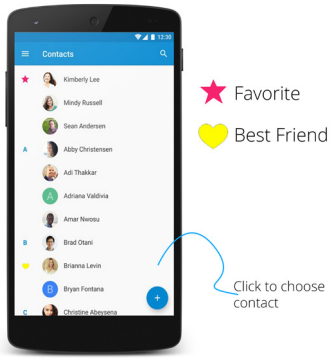
Furthermore a Graphical user interface is prefered, because currently it is the norm when it comes to fully developed apps and without it the application will look unfinished, however it is not part of the original task in the challenge and because of this reason there are parts of the interface that could still be improved in the final product.

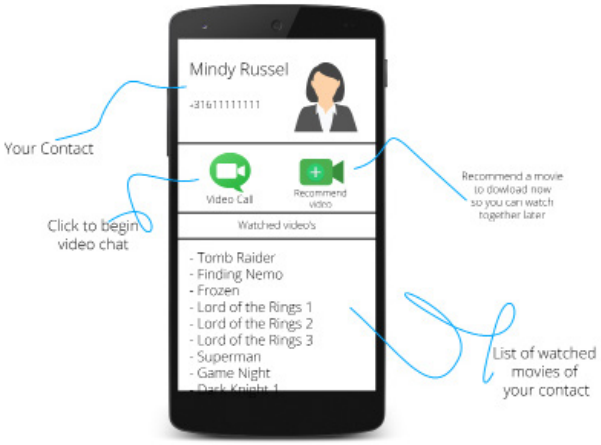
Lastly video's need to be played, this can come from a large variety of sources, such as streaming from one of the phones, having the video installed on both phones or playing the video from a source online. Since the main goal of this course is to create a demo it is not necessary to implement all of these options, but in the eventual app there is no reason why playing videos installed on a phone and from an online source couldn't coexist.

3. Ideation

There are many people that have friends, family or just colleagues abroad which they might want to watch a movie together. We will build an app that enables users to watch movies together but watching movies together will not be the only option. It will be possible to also see each other's emotions and reactions on the video with an implemented video chat function. To see how this app will be used and see possible problems we have made a storyboard. In this storyboard we have implemented our app in the ideal situation, so not the version which we will be able to build.

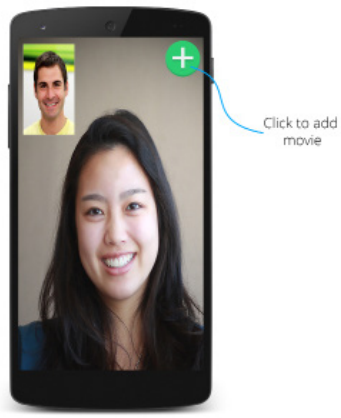
When the app is opened the user will be entered in a contact list after the start screen. In this contact list, the user will be able to see the contacts which are using this app. It is possible to add labels on contacts like favourite or best friend. On this screen you can either add a contact or choose one to make a connection.





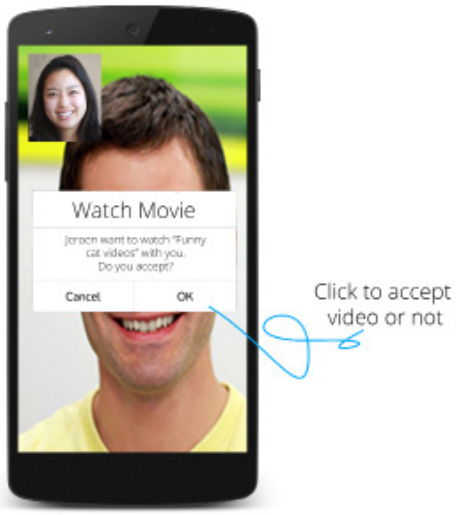
After choosing and clicking on a contact you will enter a screen within information about your contact like name and phone number. In this screen you can also invite your contact for a video call or recommend a video which your contact download to watch together later. There is also a list on this screen within the watched movies and videos so you can see what kind of movies your contact likes.

When you have clicked on video call in your contact screen, your contact will be called.



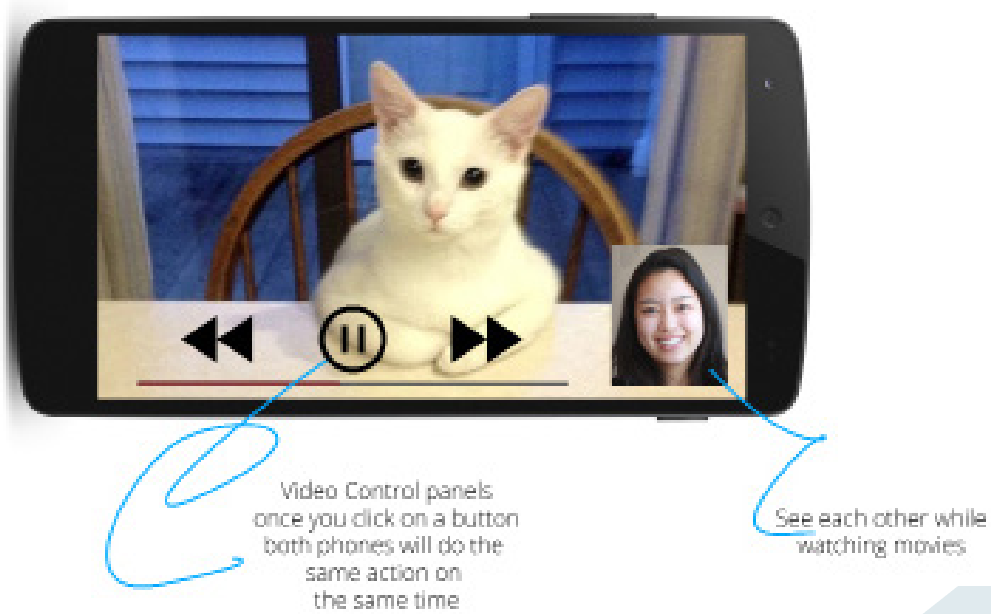
Once your contact has accepted the call, the video chatting will be started on this screen you will see your contact on the big screen and yourself on the small one. On this screen there will also be an add button. This button is to add a video or movie to the chat so you can start watching a video or movie together.

When you have clicked on the add button there will pop a list up from the bottom. On this list you will see the movies which are downloaded on both devices and are ready to watch. Next to some of the videos are check marks, this means that this video has been already watched by your contact. By clicking on a video on the listed you will choose that video to watch together.

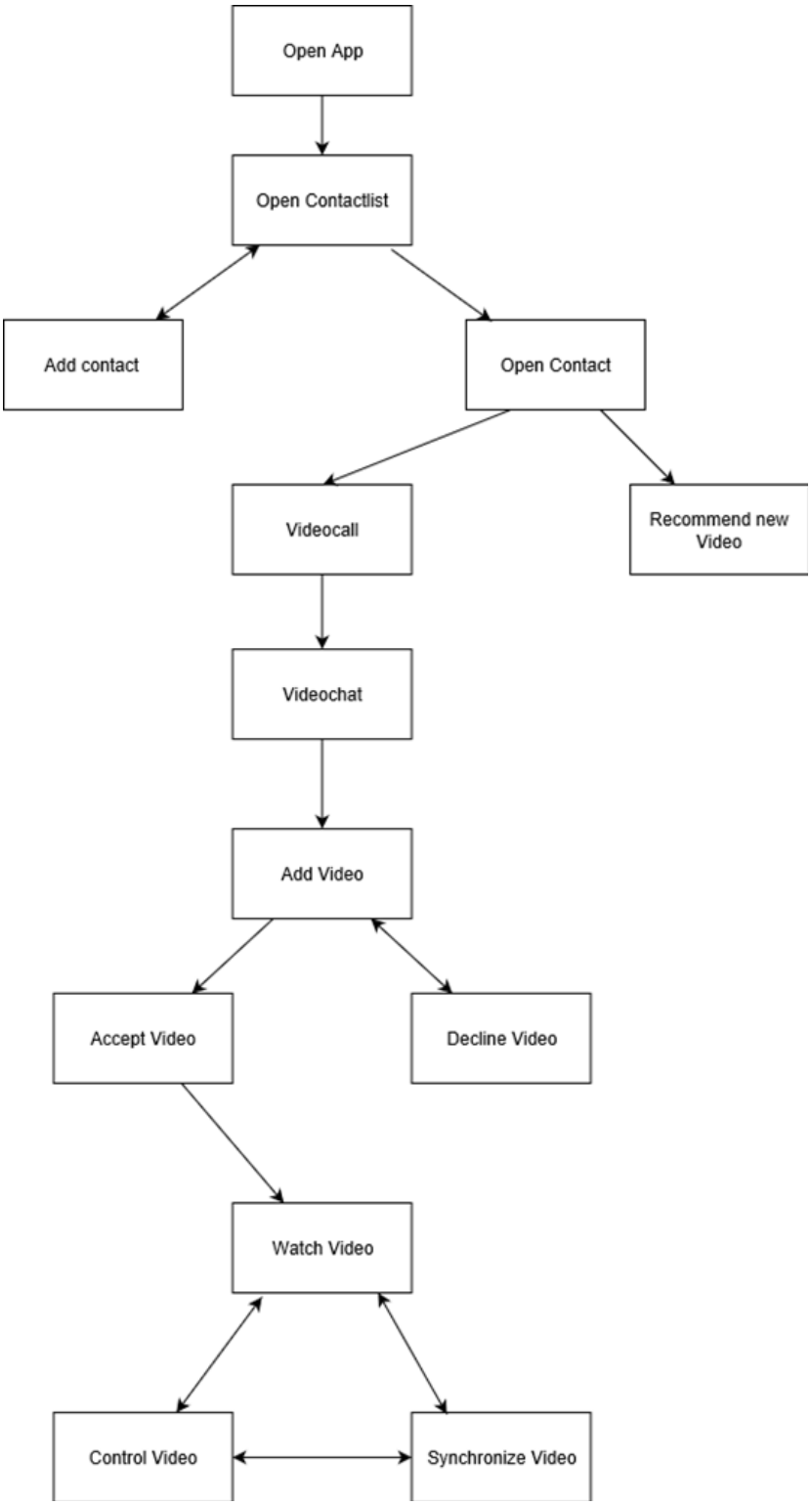


After clicking and choosing a video, there will appear a popup screen on your contacts phone. This popup screen will allow your contact to accept or decline your invite.

Once your contact has accepted your invite the video which chosen will play. From now on you will watch the video together and you will see your contact on a small screen on the right bottom corner. There are also control panels which when used, it will automatically synchronized also on the other phone.



The storyboard is based on the block diagram below



4. Implementation

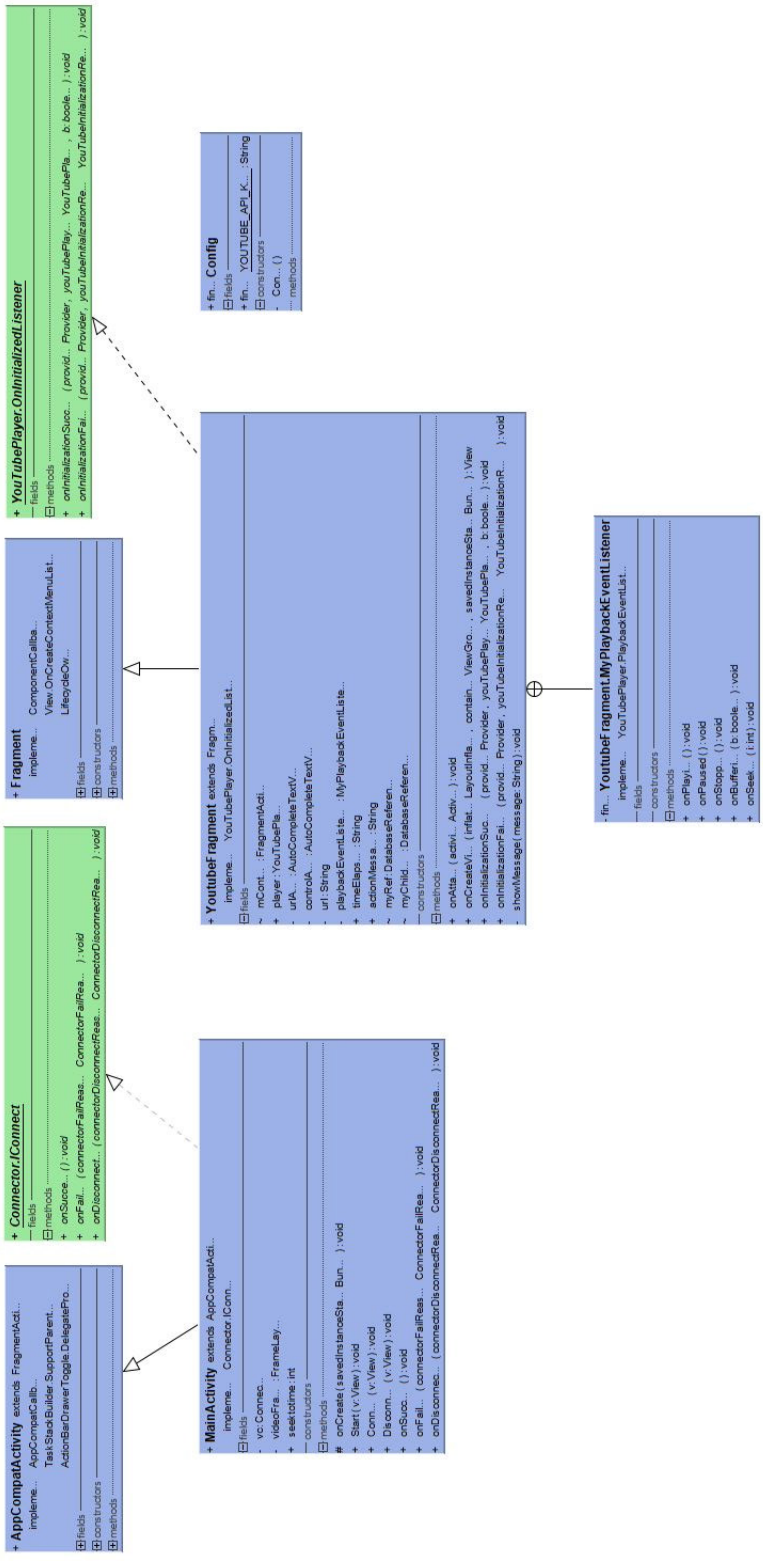
4.1 Java concepts

According to our storyboards, the video-chat window and the video window needs to be shown at the same time. To achieve this, we think there should be one class for video chat and one for video watch.

Class Main Activity - Video chat

To fulfill the video chat function in our app, we choose to use third party API. Normally a third-party API for video chat comes along with its own SDK and server, which makes it much easier for developers to implement in their own apps. During our search for API, we don't limit ourselves in video chat function, but also message chat function.

After trying TokBox, MQTT chat, Vidyo etc., Vidyo is the first API that we got to work. After implement their jar package into our android project, by simply implementing Connect. IConnect interface from Vidyo, and then generate a token for registration and connection, we can realize the video chat function with several Start(), Connect() methods in our Main Activity class. In Vidyo, chat members who share the same token can connect to each other and chat. We generate a token from Command Prompt, and then write it into our code. To video chat among devices, the devices have to share the same token in their android code to connect to each other. Up to here, we basically get the video chat function working in our main activity.



Class YoutubeFragment - Video watch & sync

Video watching

Meanwhile, we are also search for implementation of video watching function. Following coach's advice, we check Media player API and Youtube API, both can be implemented in our android project. We choose Youtube API because we get it work first, and what makes it better is that it can be extended as a fragment in android project. So it can be shown on the screen along with the Main Activity view.

We also create a sub-class called MyPlayBackEventListener implementing YoutubePlayer. PlayBackEventListener interface. With this, we create a bunch of listeners that can react something when we play, pause, drag the video to certain time. These methods give us a chance to develop further – the video sync between two devices function.

Video synchronization

One of the most challenging function we want to fulfill in this project in sync the video that the users are watching. That means if one user play/pause the video, or drag the video to certain time, or play another video (URL), the video on another user's phone should also be synced. To realize this, we decide to add a message chat function into the app. Whenever, one of the user make a move on the Youtube video (play/pause/seek to time/play URL), it will send a message (String) to another user, when another user receives it, his Youtube video will do the same move.

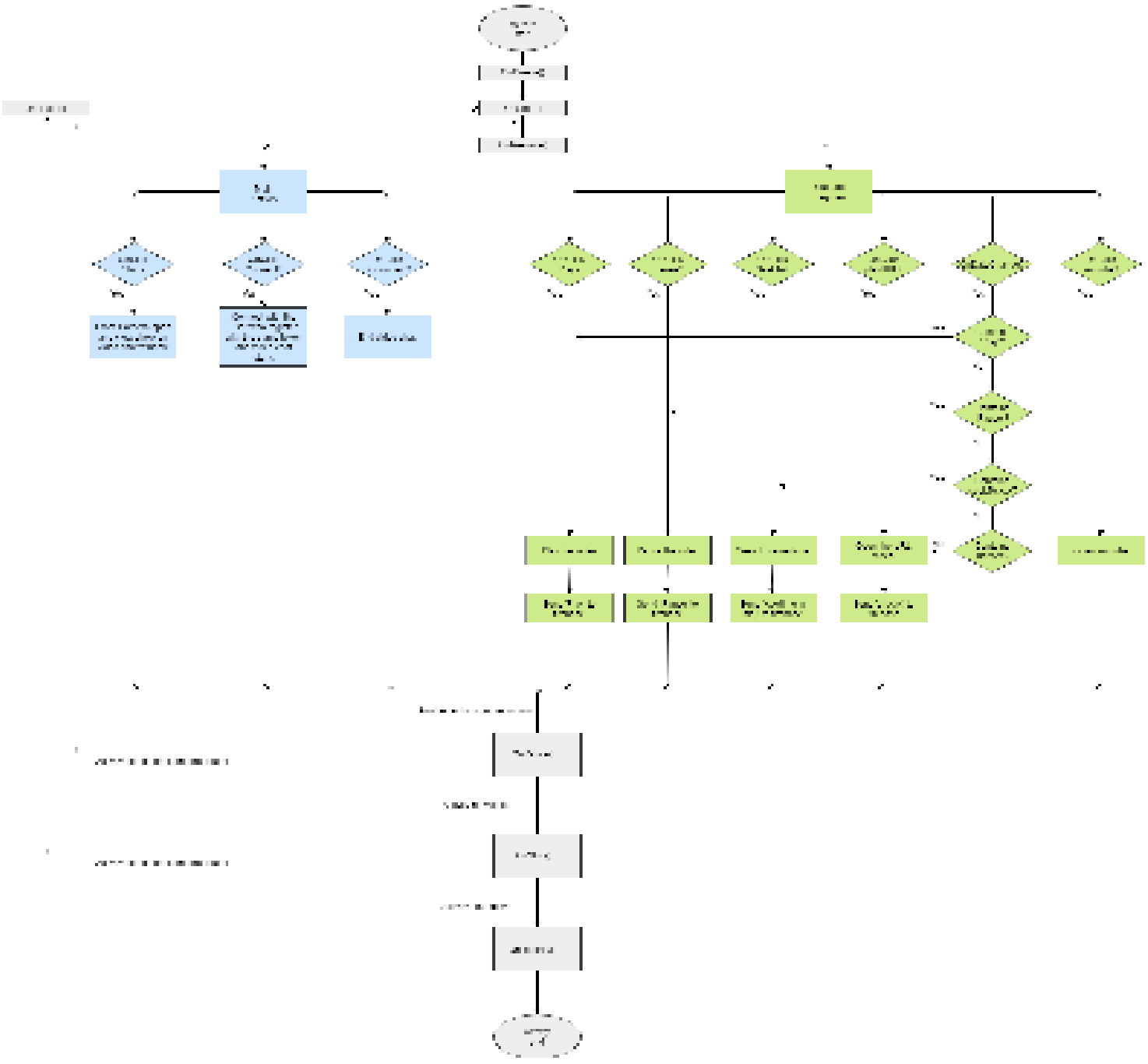
Similar to video chat, we also try to find a third-party API to realize message chat function. After trying MQTT and several others, we decided to use Firebase. We need to build a 'string sender' and a 'string listener' method in our Youtube fragment, that is why we just import Firebase library into the YoutubeFragment class. We will talk about more details about how we work with Firebase later.

Class Config – Youtube API

This class is just for configuration of Youtube API.

4.2 Android concepts

As shown in the diagram (a larger version can be found in the appendix), Main activity and Youtube fragment are two independent systems. In this way, both activities will be started together but run individually. Good thing about fragment is that it can run individually and also be shown on the screen at anywhere you want.



Main Activity

Main activity implements the Vidyo API to realize video chat function. By following the tutorial from Vidyo, we simply implemented three methods (Start(), Connect(), Disconnect()) to make video chat work. Although the token we mentioned before will expire after certain time, so every time we test, we need to regenerate token and put it into out code again, but for prototype phase, we can accept this. If we develop further, we can use Vidyo server to update each user's token remotely. After we get Vidyo work, we put its whole layout into a frame layout, making it works as a fragment, so that we can adjust its size and location as our will.

Youtube Fragment

As we said before, we created a sub-class for listeners to react to all events happened in Youtube player. Every time one event is listened, it will send a relative String ('play', 'pause', 'seekto: time in millis seconds') to our database created in Firebase using Firebase's server. We also create a text view for user to enter their video URL, and a button to play the URL. Once the 'playURL' button is clicked, it will also send a String ('url: url of the Youtube video') to our database.

Now we get a 'string sender', we still need a 'string receiver' to sync two user's video. Fortunately in Firebase library, there is an OnDataChange() method, we use it as an event listener. Once the last message our database received is changed, OnDataChange() will be called. Then we get this last message and transfer it into a string, we do relative action on the Youtube player according to what the string contains. In this way, we realize the remote video sync function. Both users can control the video and be synced all the time.

There is another event that won't send strings to database when it is happened, that is the Youtube icon, it will open the Youtube website in a browser or the Youtube app from user's phone and pause our app. The user jumps to the Youtube website, from there they can select their favorite video and copy the URL of that video, then by playing the URL in our app, another user will also play the same video. In this way, user can search and share videos with his chat member.

4.3 Implementation of the GUI

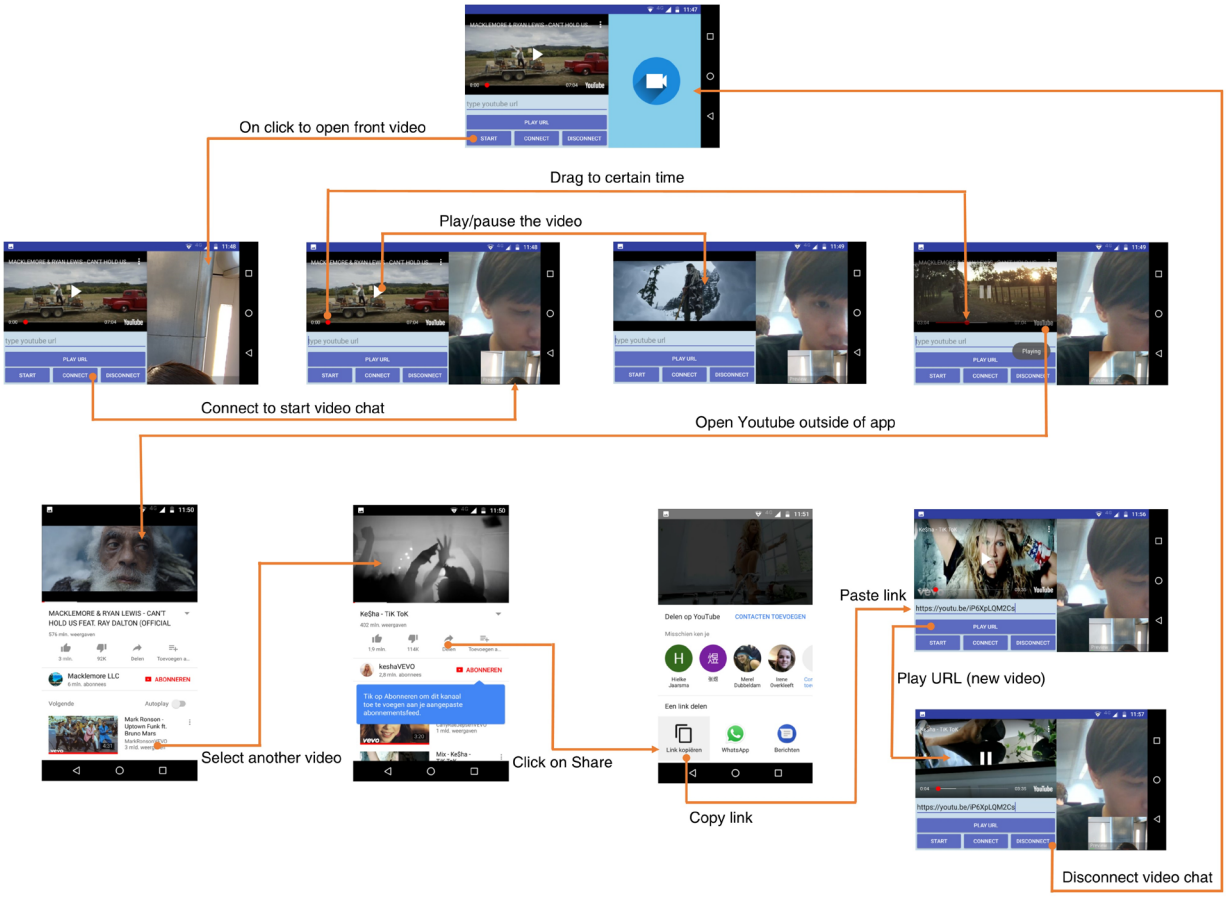
As shown in the wireframes, our implementation of GUI is really simple and straightforward. Mainly it is divided into 2 parts – video chat part and Youtube part. We choose to use a horizontal setup because in this way we can set both two windows larger that vertical setup. We also set whole layout as relative layout because for our app it is easier to arrange GUIs just around each other.

Video chat part

Originally we want to put the video chat window over the Youtube window, just like all the 'reaction videos' have been made. But Youtube fragment won't let developers to cover any part of Youtube video in their apps, so what we can do is put the video chat part next to the Youtube video window at right. For that three buttons (start, connect, disconnect) sit next to video chat window at the bottom, they are standard implementation of Vidyo API.

Youtube part

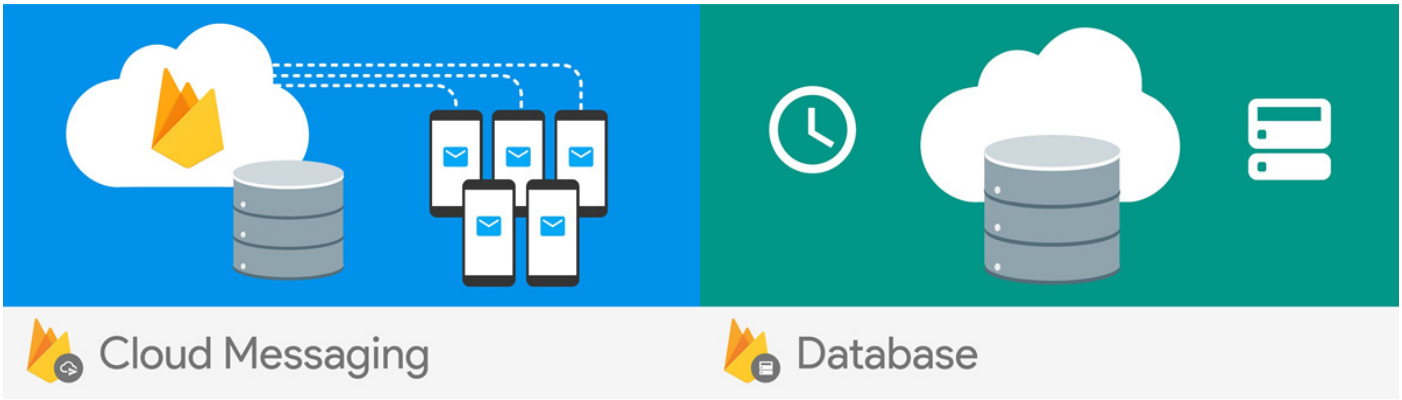
Except for the 'enter URL' text view and 'play URL' button, all other GUI elements (play/pause/drag/ Youtube) are already automatically implemented in the Youtube player. And the Youtube window is placed on the up left corner next to video chat window. The 'URL' text view and button is right under the Youtube player.



4.4 Implementation of the database and API's

Database

Due to the way this project was handled the eventual usage of the database is less efficient than it could be and it doesn't use the firebase database to its full potential. The reason for this is that originally the plan was to use a messaging chat as a means of data transfer. The idea was to take a sample chat code and reverse engineer it down to its basic core. Since the two devices would have been able to communicate with each other already there wouldn't be a reason to implement another server. Originally the plan was to use the firebase messaging service, which has a onMessageRecieved() function. This function would provide the last send string. However at this time three people were working on different parts of the app and therefore it was uncertain if this would end up being the way the communication between the phones would be handled. Yu was working on the video chat while the messaging application was being made and got it to work. Even though messaging was the minimum that needed to be accomplished, the video chat was still prefered, because watching somebody's reaction while watching a video increases the experience by a large amount. With the Videochat taking the place of the messaging application there was still an option to have the chat messaging hidden in the background and to keep using it as a means of communication between the two phones, however at this point it would be far easier and logical to just use a server.

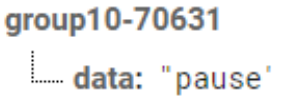


Luckily during the creation of the code this was already taken into account, because the code that actuated the youtube player was made to be functional receiving only single strings. This way no matter what method of data transfer was used, it would instantly work with the code if it could fulfill the simple task of sending a string. The way this string would be used to actuate the youtube player is by checking the string for certain commands such as play pause or seekto using the contains() method. Then to actuate playing a video from an url or skip to a certain timestamp all parts of the the string that did not contain the necessary information would be replaced with nothing.

```
if(actionMessage.contains("seekto")){
    int skipToSecs = Integer.valueOf(actionMessage.replace("seekto",""));
    player.seekToMillis(skipToSecs);
    actionMessage = "";
```

Because at this time firebase was already functioning and it also has a realtime database, the decision was made to try to implement this, instead of searching for other types of servers, even though plenty of functioning ones exist. The selection process at this point was not to critical anymore since the code could work with just a singular string.

The firebase database is incredibly simple to understand, it works in a branched structure where data can be stored underneath a common tag. In the app this functionality is not used since it only required one string, however if it were used the code would have been far more efficient. Instead of checking if the string has a certain command several tags for every command can simply be made. There could even be a higher tag containing the user that gives the command. However for the prototype the efficient type of coding was deemed sufficient and therefore was kept.



The onMessageRecieved() method was now no longer usable, but another event listener method called onDataChange() would perform the same function. Writing to the database is as simple as using the method setValue() from firebase, with which you only have to target the tag in the branch that the string needs to be stored in.

```
DatabaseReference myRef = FirebaseDatabase.getInstance().getReference();
DatabaseReference myChildRef = myRef.child("data");
        myChildRef.setValue("play");
myChildRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        String actionMessage = dataSnapshot.getValue(String.class);
        if(actionMessage.contains("play")){
            player.play();
            actionMessage = "";
```

Video player

Three options were explored when it comes to using a video player of some sort. The first two were by using the android videoview to either play a video that is downloaded on both phones or to play a video from youtube. Playing a video that was already downloaded on the phone seemed to be fairly simple, but using the player itself to play a youtube video seemed to require a webview instead. Before this option could be further explored, the third option, the youtube API, was already functional. For this reason the other options ended up being discarded during the rest of the project, however if the app would ever further be developed the option of playing videos stored on the phone should be examined once more, because it does have added value, since the user shouldn't be forced to upload to youtube just to share a personal video.

The youtube player has various methods that allow it to easily fulfill the synchronization tasks. There are listeners for when the video has been paused, has stopped, is playing, is buffering or is set to a certain time stamp. Furthermore it can easily be actuated in such an event because it also has methods to perform these tasks. This makes it possible to use the regular youtube player controls and not implement new ones. However a problem arises with this relatively lazy usage (or time efficient) usage of the player. Now if the play button is pressed a signal is send to the server that still has to be received by the other user. However the video will instantly start playing. Because of this the other user will be slightly behind in when watching his video. A solution to this could be to have custom buttons that only send a command to the server to actuate both phones. This way the delay will only be based on the difference in time it takes for both phones to receive the signal from the server, but it would also make the app seem less responsive for the first user. For this reason the current way the buttons are used seems to be close to optimal in function.

```
@Override
public void onSeekTo(int i) {
    timeElapsed = Integer.toString(player.getCurrentTimeMillis());
    showMessage(timeElapsed);
    myChildRef.setValue("seekto"+timeElapsed);
}
```

A large limitation when using the Youtube player only became clear later on in the project when a large amount of code had already been written to function with it. While trying to change the layout to have the video chat overlap the youtube player the youtube player kept stopping itself. Firstly this seemed to be a known bug when using the youtube api, because it doesn't function with a relativelayout, which was necessary to achieve the overlap. However this could easily be solved by placing the youtube player inside of a framelayout that itself is inside of the relativelayout. However nothing changed, the youtube player would still stop itself. The reason for this was because it was programmed to not function when the player is in the background, since this is against the terms of service, which were obviously not read.

For the youtube player to function inside of an activity it needed to extend youtubeBaseActivities. Since the youtubeplayer was the first element to function in the project the it was unsure how easily it could be combined with the other element. If the video chat api needed to extend another class it could create issues, since an activity can only extend a single class. For this reason the choice was made to rewrite the code for the youtube player to a fragment, since this way the activity that it is in only needs to reference the fragment in the layout file. This could have created other problems because communication between a fragment and the activity that it is in is fairly complex, however this problem has been dealt with by handling communication with firebase in the fragment, instead of the activity and only have the video chat inside of the main activity, because it doesn't need to communicate with the elements in the fragment.

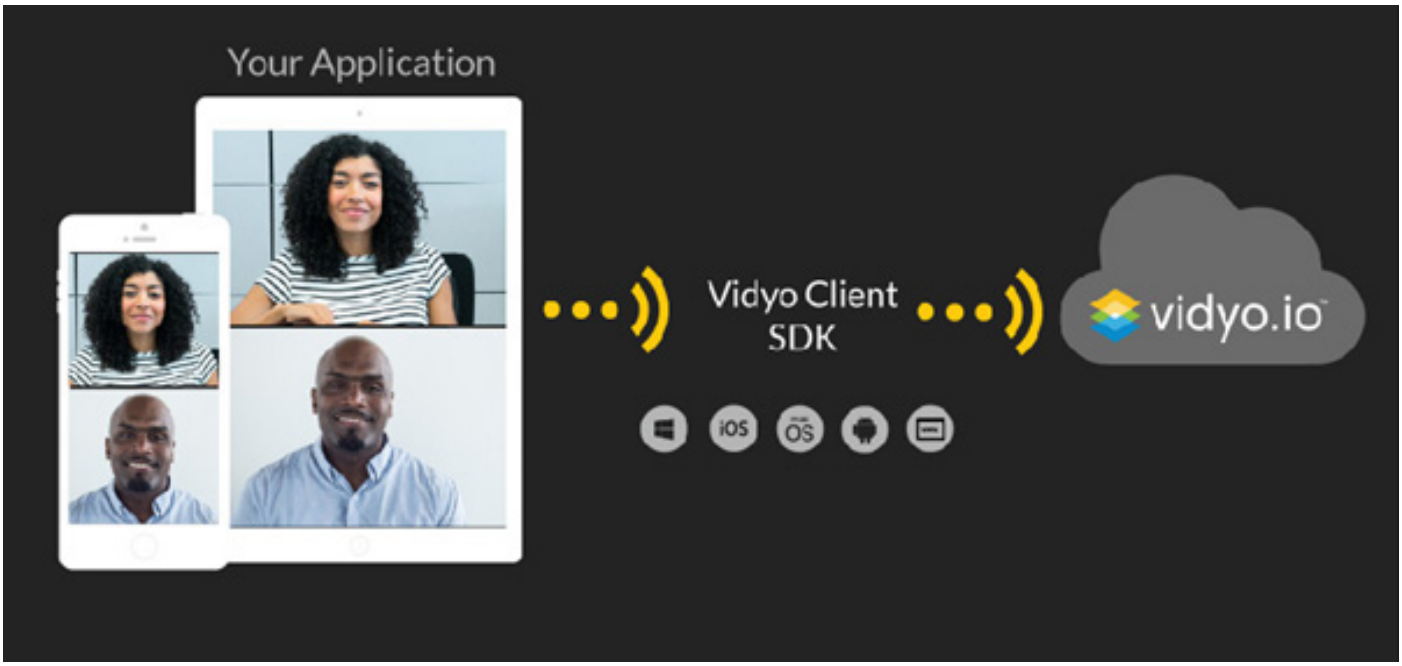
Video chat

The videochat is both a blessing and a curse. Creating one from scratch is far to complicated for beginners and therefore it was required that an api is used. Since the video chat was a stand alone element the only criteria in its selection was that it functioned. This eventually led us to using a vidyo sample code. This sample code is incredibly basic, to ensure that it can be easily applied in an application. However for this same reason there is little to understand about the code and the way it behaves. A large number of problems were only found during testing, which will be discussed in the next chapter. But because the explanation of the code is lackluster it takes experimentation to finds it limitations. For instance there are buttons to start the camera and one to connect to the server, but the application won't function if the code gets moved into the oncreate() method. To this day it still remains a mystery why this is.

```
public void Start (View v){
    vc=new Connector(videoFrame,Connector.ConnectorViewStyle.VIDYO_CONNECTORVIEWSTYLE_Default,3,"","",0);
    vc.showViewAt(videoFrame,0,0,videoFrame.getWidth(),videoFrame.getHeight());
}

public void Connect(View v){
    String token="cHJvdmlzaW9uAHZpc3VhbGNpbmVtYUBkYzQzYTQudmlkeW8uaW8ANjM2OTA1MDkwNzMAADg4YWQxODVhNGMzNGFiZ
    vc.connect("prod.vidyo.io",token,"visualcinema","group10", this);
```

Furthermore the servers for vidyo only stay active for a maximum of 10000 seconds, which is around three hours. Every time after this the code will have to be updated to contain a newly generated token. It is possible to recreate the token generation code inside of the application but it uses methods from an API level that was too high compared to that of most phones that were available for testing. The whole code could have been rewritten, but this simply would have cost to much time and would not accomplish much extra for the demo.



5. Testing and evaluation

5.1 Test setup

Three aspects were of importance when testing the application, the first being the clarity of the interface, the second being the user experience when using the app and lastly the presence of bugs. In order to test this our user test consisted of three stages.

In the first stage the app was opened and the test subjects were asked to describe how they thought all elements of the interface would function, after this they were given the task to press the buttons to accomplish the following tasks:

-Start up the video chat.
-Move to a certain time in the video.
-Play another video.

After this they were told to just use and enjoy the app for the next 10 minutes with another test subject in a different location. From this the goal was to see what the behaviour was like when using the app and the way the user experienced it. This gave the users a better idea of what could be improved, such as the ratio in size between the videochat and the videoplayer.

Lastly the users where tasked with breaking the app. They were allowed to press anything in any order, originally there was a plan of handing out a reward to the people who did, but bragging rights seemed to already be sufficient.

At the end the users were asked what they would still improve about the app.

5.2 Evaluation

The testing was done only by 4 other students and therefore doesn't show accurately how the general population would use the app. For this reason the first stage of accomplishing basic tasks is expected to show far less problems compared to when this is done by people inexperienced with technology. However the last task of breaking the application is best tested among people with an understanding of technology, since they are most likely to think of ways that bugs can occur.

In the first stage the order that buttons needed to be pressed in was understood by many due to the positioning of the buttons and familiarity with the youtubeplayer. The play url button was surprisingly clear to everybody, due to the hint in the the typebar above the button. When it came to the video chat controlling buttons the order that they needed to be pressed in was clear due to them being ordered from left to right, but the difference between start and connect wasn't completely clear to anybody. As expected this part of the testing showed no real signs of struggle only confusion with the naming of the buttons.

In the second stage it became very clear that the way the videos are selected is far from optimal. Because if users take to long inside of the youtube application on a phone with only a limited amount of working memory the video chat application gets shut down and needs to reset. This was the largest cause of frustration, with the second and thirth being only very minor complaints. The second complaint was about a lack of an volume slider, which caused people to have not be able to talk to each other as easily during a video. Lastly one user thought the youtube player could be slightly increased in size. Over time the users got more comfortable and started enjoying the core concept of the application, by sending each other strange videos to get a reaction out of one another.

The final stage is where it became clear that the video chat functionality has a lot to be improved upon. If a user disconnects accidently by leaving the application for to long the disconnect doesn't get registered. Which means that the next time he tries to connect the maximum amount of users (which is two) has already been reached. This makes the app highly unstable in its current form in the hands of a user that is unaware of this problem. Secondly pressing the vidyo controlling buttons in the wrong order will easily cause the app to crash (and we still have no idea as to why this is because the vidyo code is so unclear.) Lastly as expected there is a way for users to cause the videos to no longer be synced if only one user repeatedly presses play and pause many times.

Unfortunately there is not enough time in the project to fix these problems, but a suggestion as to how they can be fixed will be given. Most important is to cause the app to not crash, this means that whenever the app is exited it needs to also disconnect from vidyo. Furthermore buttons should only be pressable in the order they are supposed to be pressed. Another problem that needs fixing rather badly is the fact that the videos can get desynchronized by repeated pressing of the play and pause button. Whenever the play event occurs a timestamp can be send to the other user, if the difference in timestamps is to big the second user can jump to the timestamp of user one. To further optimize the app it is recommended that a youtube search bar gets added into the application itself and a volume slider also gets implemented. When these main concerns have been addressed the next step would be to make the app function more like the one we had in mind originally at the start of the project, in which you could select anybody from your contacts to start a session with. But the way the application is right now has already been deemed very enjoyable by the test subjects.

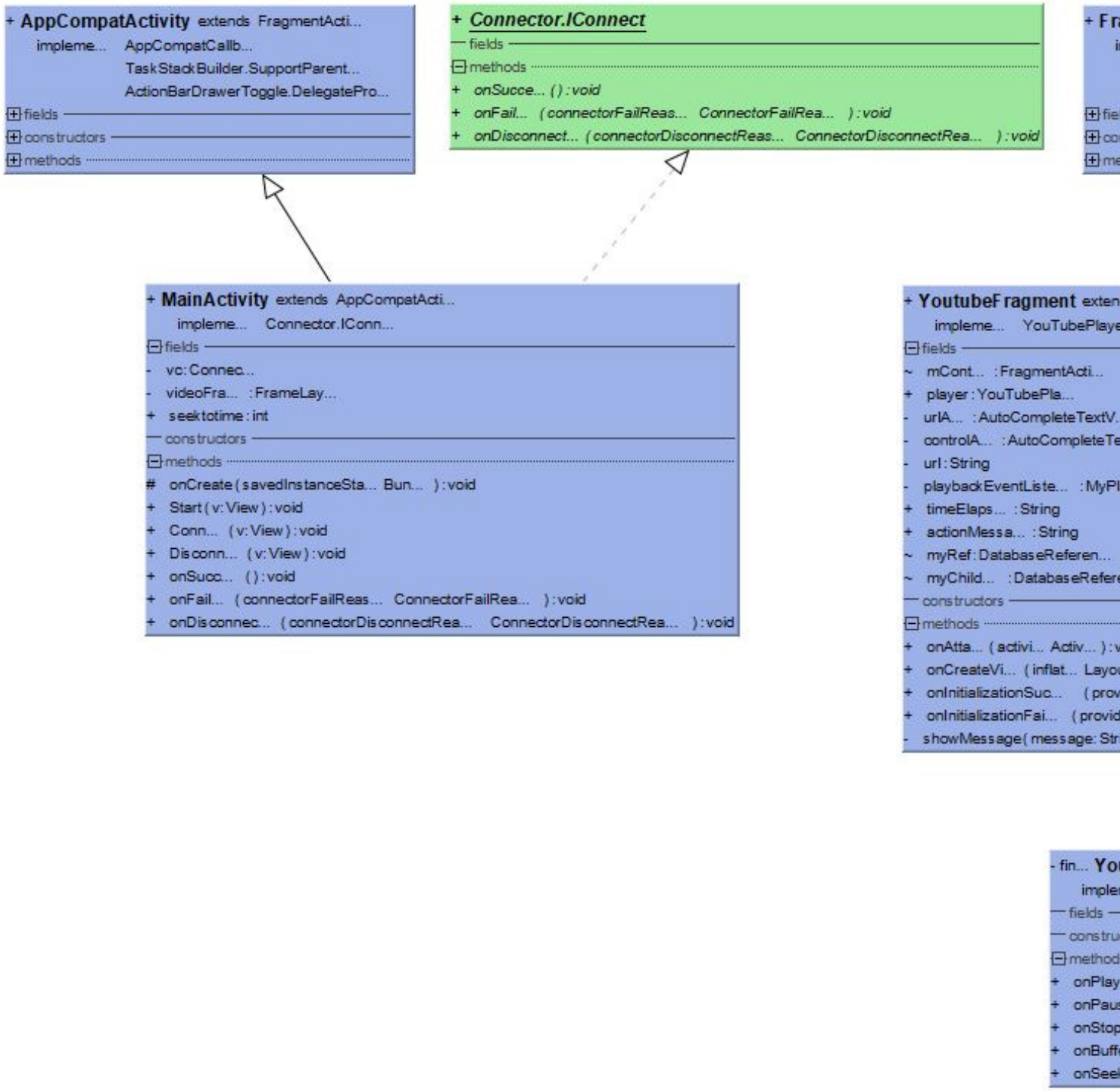
6. Conclusion

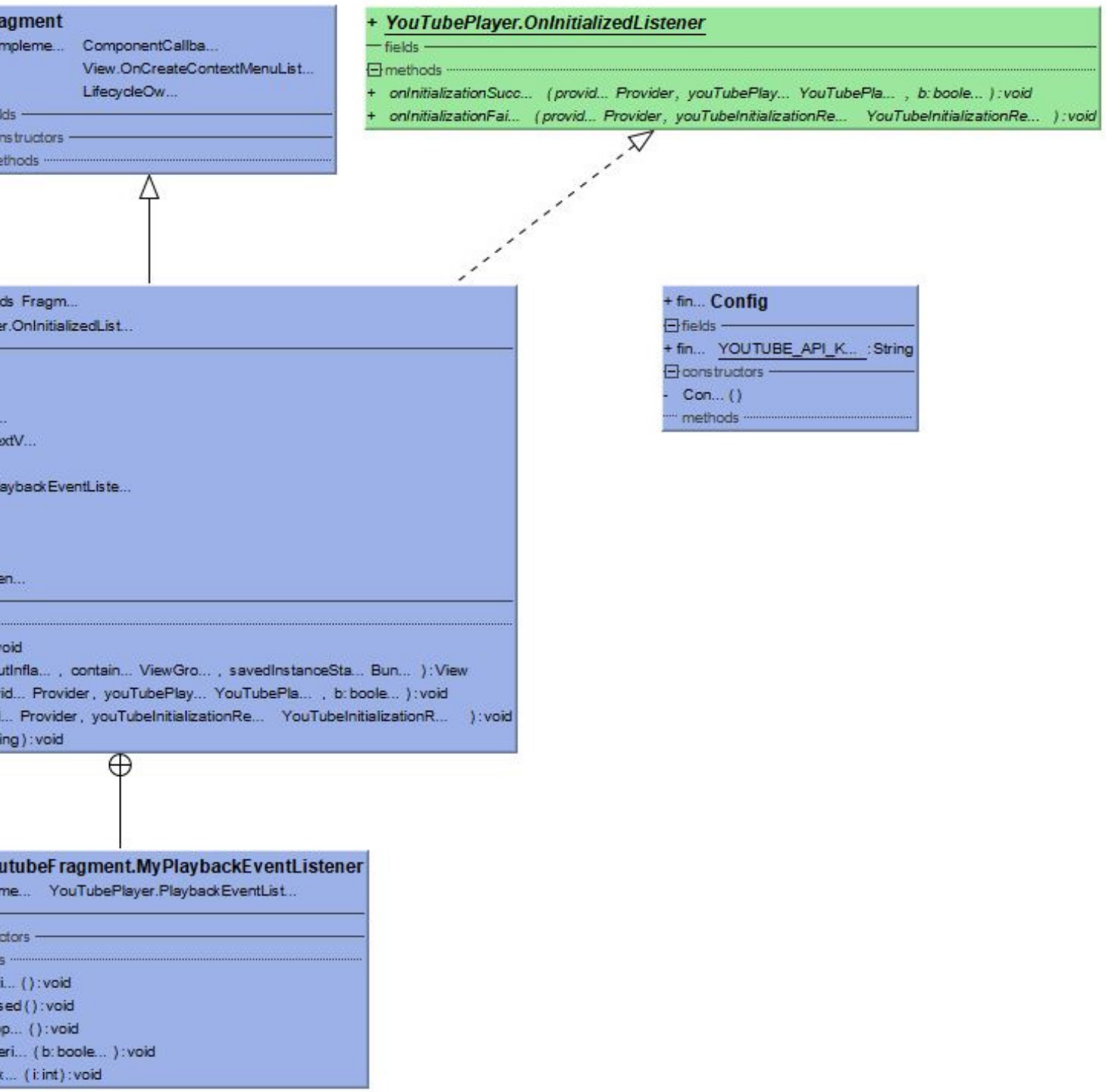
The demo that we were able to present at the end of the course may have been a lot simpler than what we originally envisioned, but it does contain the essence of the idea and is a strong proof of concept. For this reason we are very pleased with the way this application turned out, but would still recommend a large amount of improvements to make it closer to the app we envisioned in our idealisation. The demo app can only work with two users and has a clunky way of selecting videos, ideally this gets improved upon first after the main bugs in the demo have been fixed. But improvements can always be made and the way the app is right now is already a great alternative to other options that exist.

The way this project was approached was surprisingly efficient the first day we were living in a dream-world and believed that we could make the perfect app in just a few weeks, but after just one coach meeting the goal had been set and work has been done very efficiently from that point. The difference in roles of people in the group wasn't that different from one another we trusted each other to work with similar efficiency at any aspect that we would tackle, since we all had no android programming expe-rience. Whenever we could we split up tasks in a way that would interfere with each other as little as possible. Whenever this was not an option we looked for multiple approaches to a problem and have everybody work on each approach until one of us succeeded. This caused us to never get stuck for a long time, because there was always somebody that was able to accomplish the task that was set out. This method would be incredibly inefficient in almost any other field, but when it comes to programming avoiding issues due to incorrect choices seemed like a very beneficial choice.

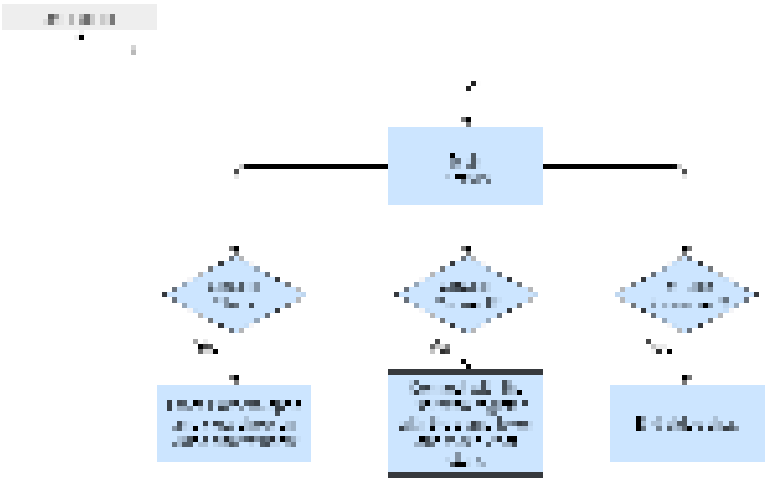
As mentioned before none of us had any real programming skills in android and we now all are fairly confident that we can find the resources and have the knowledge to create almost any kind of app for a demo. Even though we learned an incredible amount about coding itself we believe that it is moreso a mindset that is required for coding compared to a fast knowledge of all available libraries. We became very quick at finding and examining existing code and using elements of it in our own. Besides it is very likely that if we were given another programming task in a group our approach would now be very simi-lar. Oddly it requires a certain amount of experience and confidence to not be intimidated by the task of coding and once that has been reached the amount of resources available are endless. So at the end of the day we learned a lot about coding but what made the most impact is the mindset we have about it now.

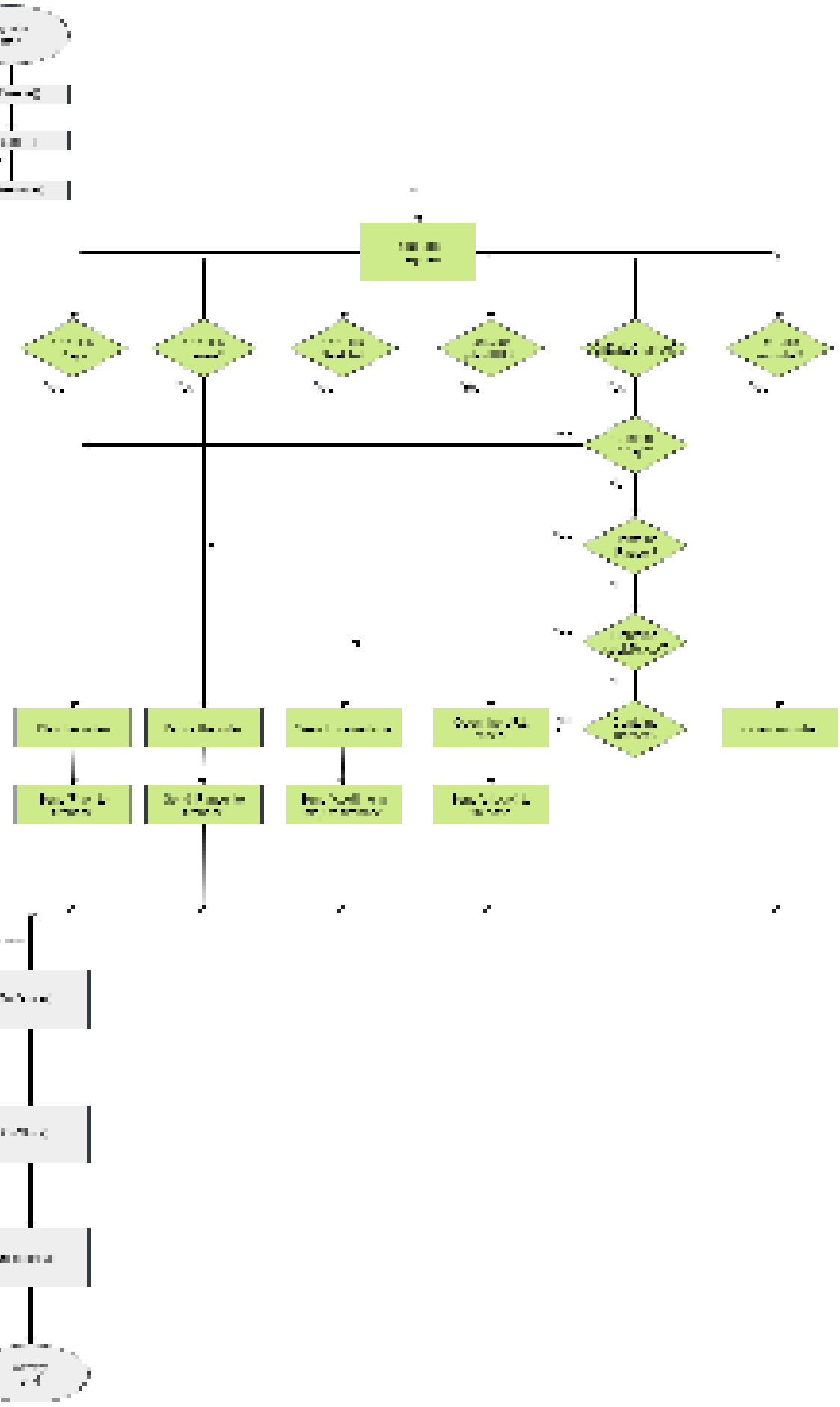
Apendix A. Class diagram



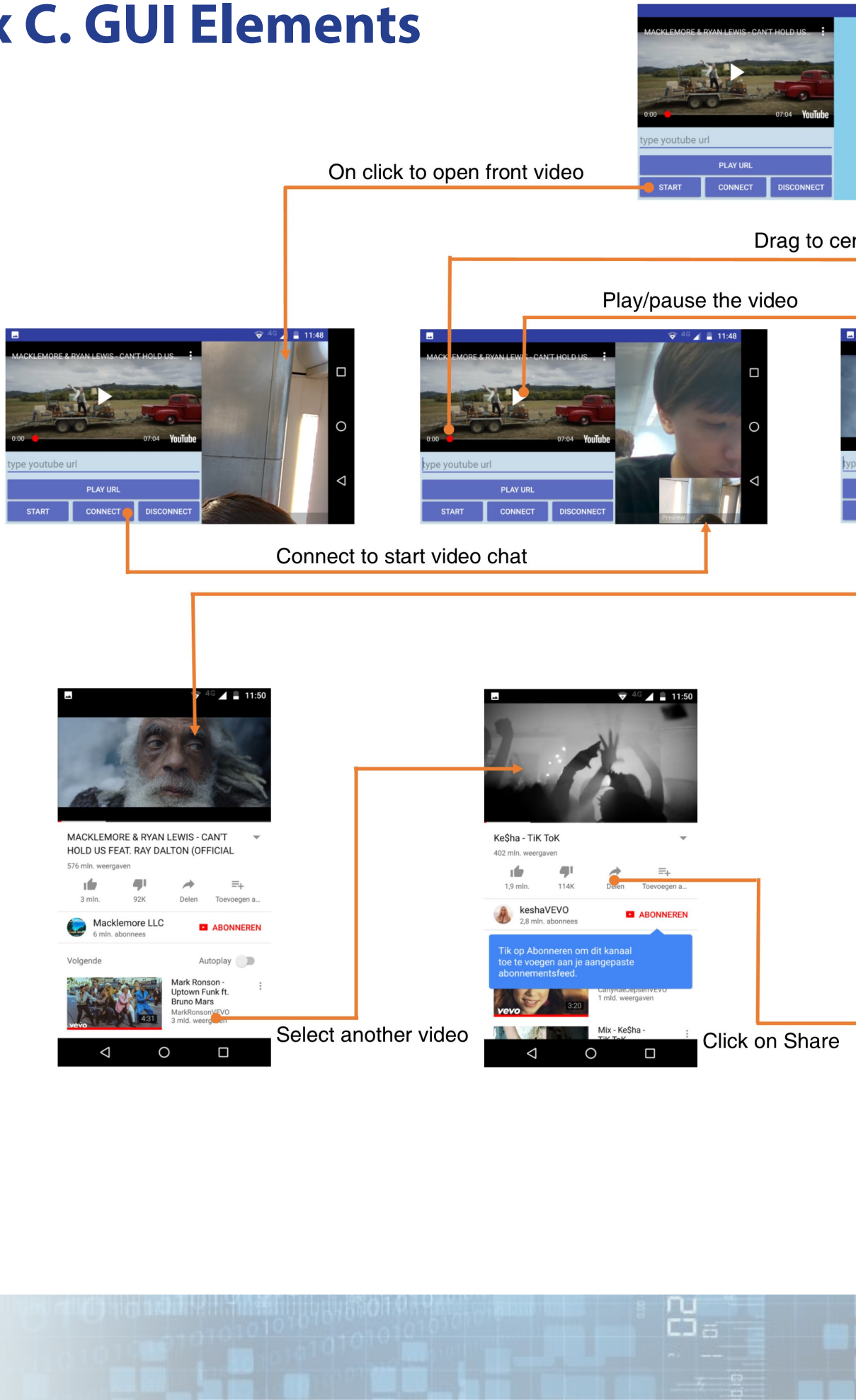

agment
mpleme... ComponentCallba...
View.OnCreateContextMenuList...
LifecycleOw...
ds
nstructors
ethods
+ YouTubePlayer.OnInitializedListener
fields
methods
+ onInitializationSucc... (provid... Provider, youTubePlay... YouTubePla... , b: boole...) : void
+ onInitializationFai... (provid... Provider, youTubeInitializationRe... YouTubeInitializationRe...) : void
ds Fragm...
r.OnInitializedList...
xtV...
aybackEventListe...
en...
oid
utInfla... , contain... ViewGro... , savedInstanceSta... Bun...) : View
id... Provider, youTubePlay... YouTubePla... , b: boole...) : void
... Provider, youTubeInitializationRe... YouTubeInitializationR...) : void
ing) : void
+ fin... Config
fields
+ fin... YOUTUBE_API_K... : String
constructors
- Con... ( )
methods
utubeFragment.MyPlaybackEventListener
me... YouTubePlayer.PlaybackEventList...
ctors
i... ( ) : void
ed ( ) : void
p... ( ) : void
eri... (b: boole...) : void
t... (i: int) : void

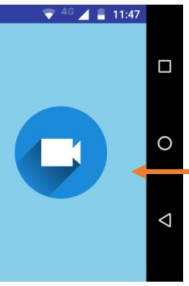
Apendix B. Flowchart



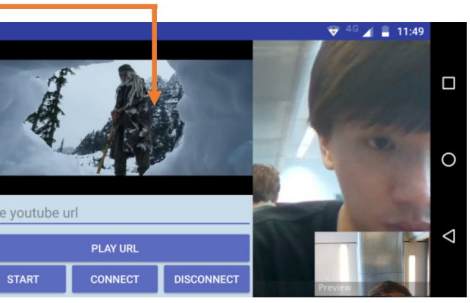


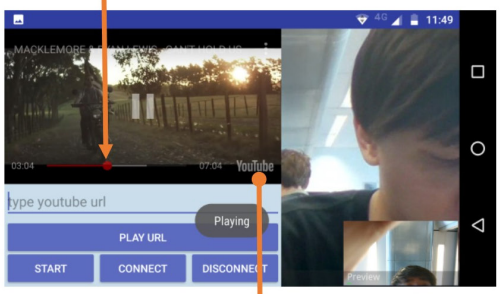
Apendix C. GUI Elements



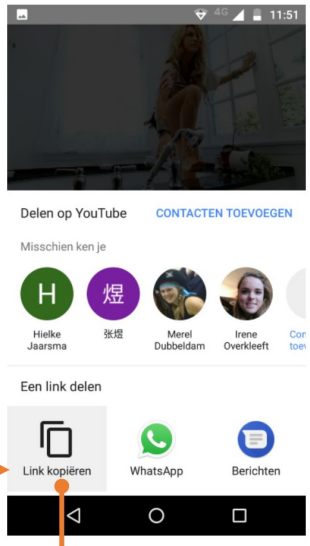


tain time

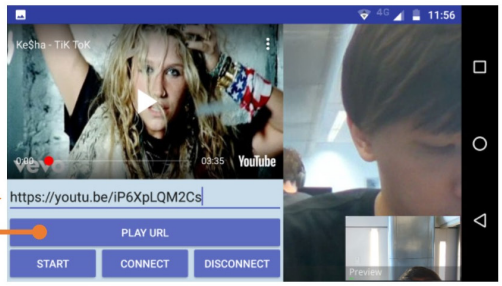




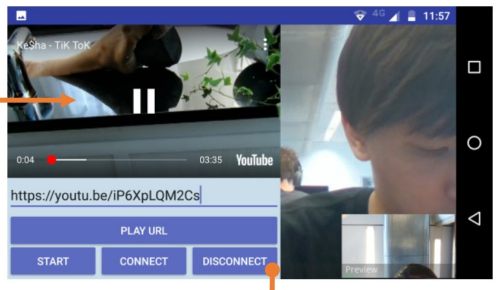
Open Youtube outside of app



Paste link



Play URL (new video)



Copy link

Disconnect video chat

Apendix D. Implementation

Youtube api

Before the libraries could be used the API first needed to be downloaded and moved to the libs folder in the application. Then a dependency needed to be added to compile the api (compile files('libs/YouTubeAndroidPlayerApi.jar. Now to make the player work you would usually need to have the activity exend YoutubeBaseActivity, but recently youtube has updated the API to work inside of a fragment, in this case it is no longer necesarry. However in this project implementing YoutubePlayer.OnInitializedListener is desired because it can be used to find out when the user presses one of the buttons from the youtube API. Next the frame inside of the fragment should be changed into a youtubeviewer and initialized with the youtub API key that can be aquired from the youtube developer console.

```
YouTubePlayerSupportFragment youTubePlayerFragment = YouTubePlayerSupportFragment.newInstance();
FragmentTransaction transaction = getChildFragmentManager().beginTransaction();
transaction.add(R.id.youtube_fragment, youTubePlayerFragment).commit();
youTubePlayerFragment.initialize(Config.YOUTUBE_API_KEY, this);
```

Now when the initialization is succesful the youtubeplayer (named player in the code) can be linked to a listener class for when the youtubeplayer is playing, paused, stopped or set to a different time stamp. And the methods play, pause, cuevideo and seekto can be used to actuate the player.

```
@Override
public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer youTubePlayer, boolean b) {
    if (!b) {
        player = youTubePlayer;

        player.setShowFullscreenButton(false);




        player.setPlaybackEventListener(playbackEventListener);


        player.cueVideo("2zNSgSzhBfM");
    }
}

private final class MyPlaybackEventListener implements YouTubePlayer.PlaybackEventListener {

    @Override
    public void onPlaying() {
        // Called when playback starts, either due to user action or call to play().
        showMessage("Playing");
        myChildRef.setValue("play");
    }
```

Firebase

Firebase makes connecting to the real time database even simpler by adding an assistant to android studio. Simply by going to the SDK Tools and installing the google repository the assistant window can be used from tools>Firebase. From there a simple step program can be followed to implement it's libraries. From here all you need is a reference to the FirebaseDatabase to work with the realitime database and properly target the branch in the database that you want to store the data under.

```
DatabaseReference myRef = FirebaseDatabase.getInstance().getReference();
DatabaseReference myChildRef = myRef.child("data");
```

From here we only used the ondatachanged listener from the targeted branch, that runs whenever anything happens to data stored in this tag and dataSnapshot.getValue that returns the value of this branch.

```
myChildRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        String actionMessage = dataSnapshot.getValue(String.class);
```

And lastly to upload data from firebase we select the targeted branch and use the command setValue to change it to the string value we want it to be.

Vidyo

Just like the youtube player Vidyo needs an API Package to be downloaded and moved to the libs folder to be used and the connector.Iconnect interface needs to be used. From there there is little changed to the code other than the token that has to be generated in the command prompt, the name of the user and userroom.

```
ConnectorPkg.setApplicationUIContext(this);
ConnectorPkg.initialize();
```

```
public void Start (View v){
    vc=new Connector(videoFrame,Connector.ConnectorViewStyle.VIDYO_CONNECTORVIEWSTYLE_Default,3,"","",0);
    vc.showViewAt(videoFrame,0,0,videoFrame.getWidth(),videoFrame.getHeight());
}

public void Connect(View v){
    String token="cHJvdmlzaW9uAHZpc3VhbGNpbmVtYUBkYzQzYTQudmlkeW8uaW8ANjM2OTA1MDkwNzMAADg4YWQxODVhNGMzNGFi
    vc.connect("prod.vidyo.io",token,"visualcinema","group10", this);
}
```

Apendix E. Sequence diagram

Unfortunatly we didn't think of the tracing the results during the test, instead we worked as described in chapter 5. The sequence diagram should give a clear idea on how the options the user has available work inside the system