2018 – 2019
Ankara University
Department of Engineering
Computer Engineering

COM377 – Software Engineering
Project Report

CouchPotato Media Player

16290125 – Karanfil Eylül ŞENGÜN
16290085 – Yeter Tuğba ÇETİN

TABLE OF CONTENTS

1.Introduction

CouchPotato is a desktop application that can be controlled with simple hand and face gestures using computer vision techniques and can be runned with different file extensions.The main purpose of this player is to control videos remotely.

CouchPotato is a desktop application and will be controlled by mouse, keyboard and camera.

For more detailed information about computer vision technology:

https://techcrunch.com/2016/11/13/wtf-is-computer-vision/

2.Overview

CouchPotato allows the user control the videos without a keyboard or a mouse. To be able to do that there are some predescribed gestures. At the same time user can upload external subtitle files with .srt extension to the video that she/he is watching.

2.1.Use of Media Player

There are no extra interfaces except help and settings windows. All functions about the control of the media player are located on a transparent bar under the video.

From left to right the functionalities of the buttons are:

- Play/Pause
- Volume up/down – Mute
- Progress bar and timers
- Activate/Deactivate gestures
- Add/Choose subtitle
- Full screen

On the right corner of the screen there are settings and help buttons consecutively. User can display time and date with pressing 'T' on the keyboard.(This feature can also be activated using a specified gesture described in 3.1.)

Title and extension of the video file is displayed on the left corner of the screen.

2.2.Gesture Functions

- Play/Pause
- Volume up
- Volume down
- Mute
- Full screen
- Move forwards
- Move backwards
- Stop
- Deactivation
- Date & Time display
- Eye Tracking

3.Requirements Specification

3.1.Functional Requirements

3.1.1. Play/Pause

This pose of the right or left hand allows user to play and pause the video when used sequentally.

3.1.2.Full Screen

This pose of the right or left hand allows user minimize and maximize the screen when used sequentally.

3.1.3.Volume Up/Down – Mute

Volume up

This pose of the right or left hand allows user increase the volume. In the first 5 seconds volume will be incremented 5 units per second. If user keeps the position of the hand more than 5 seconds, volume will be started to increase 20 units per second.

Volume down

This pose of the right or left hand allows user decrease the volume. In the first 5 seconds volume will be decremented 5 units per second. If user keeps the position of the hand more than 5 seconds, volume will be started to decrease 20 units per second.

Mute

This pose of the right or left hand allows user mute and unmute the video when used sequentally.

3.1.4.Move Forwards/Backwards

Move forwards

This pose of the right or left hand allows user to jump forwards in the progress bar for the time period specified by the user every two seconds.

Move backwards

This pose of the right or left hand allows user to jump backwards in the progress bar for the time period specified by the user every two seconds.

3.1.5.Stop the Video

This pose of the right or left hand allows user to stop the video and close the media player.

3.1.6.Deactivation of Gesture Controlling

This pose of the right or left hand allows user to deactivate gesture controlling. You can control and check this feature from the gestures button from the transparent bar on the screen.

3.1.7.Time & Date Display

This pose of the right or left hand allows user to be able to display the time and date on the upper right corner of the player's screen.

3.1.8. Adding Subtitles

From the CC icon located on the transparent bar user can add or select subtitles (.srt extension) . Subtitle files must be in the same folder to be uploaded automatically to the video. Otherwise user should upload the file manually to the current displayed video.

3.1.9.Sleep Tracking

This feature only works - even the feature is active - iff eyes of the user are detected by the camera . If the eyes of the user are detected as closed more than 60 seconds, the video will be paused.

3.2. Non-Functional Requirements

3.2.1.Lightening Problem

Enviroment shall be illuminated adequately for the camera to be capable of detecting the movements clearly. If the lightnening isn't sufficient, gestures might be detected erroneously even not detected at all.

3.2.2. Camera Resolution

In order for the media player to work with a computer vision technique, there must be a camera connected to the computer. In addition, this camera should be sufficient in terms of resolution in order to effectively understand images and movements. Otherwise, just like the problem of lighting, misunderstandings or even no detection of gestures can be observed.

3.2.3. Manuel Start

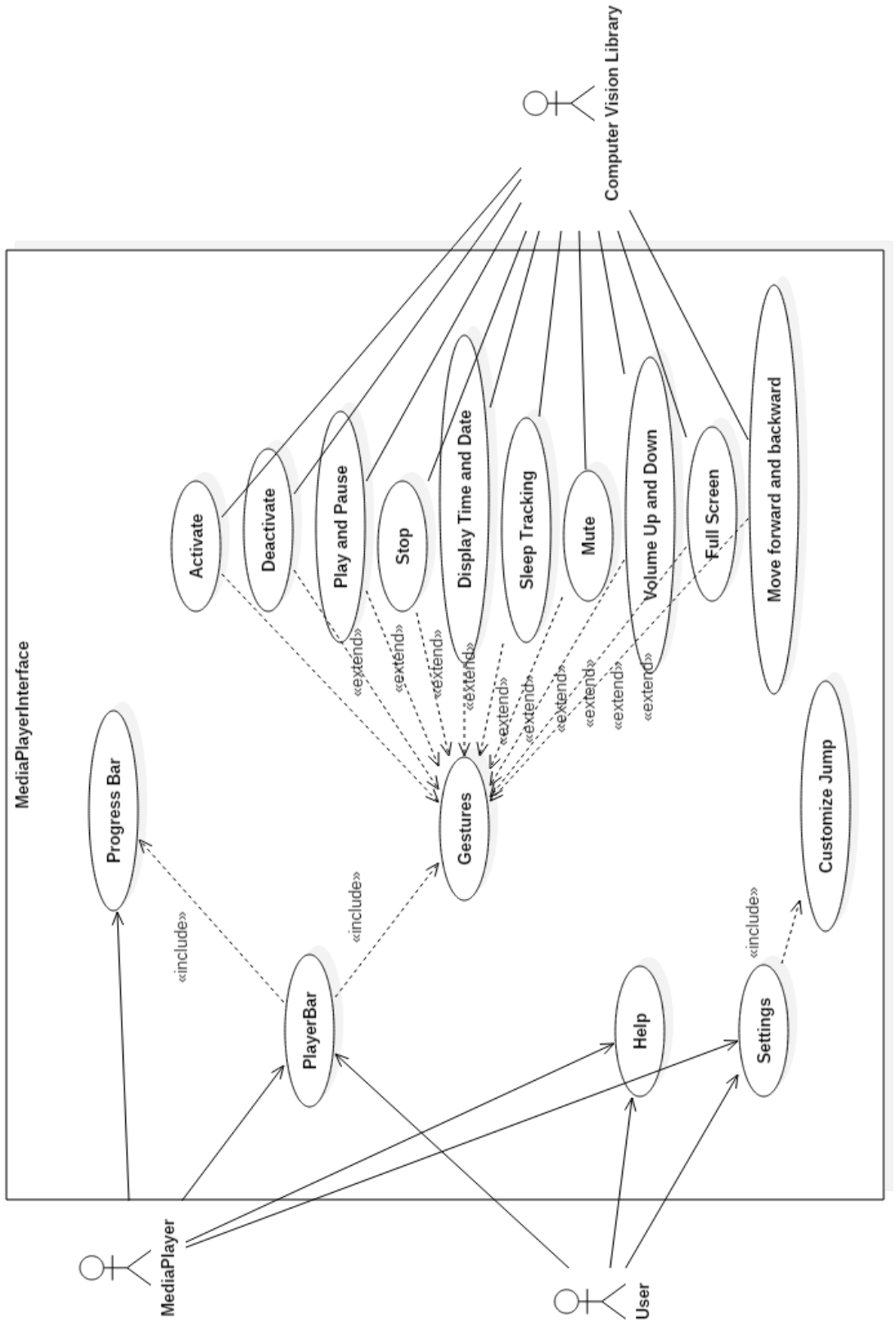
The buttons and movement activations required to disable the gestures are defined in the previous chapter. There are no specified gestures for activation. To do this, the user must manually open the hand button on the transparent bar or press the G key on the keyboard.

3.2.4. Avaliable File Containers

CouchPotato only allows .mp4, .avi, .mov (Quicktime) and .flv (Flash) video container types.

4. System Models

4.1. Use Case Model and Scenarios



4.1.1. Scenario: Activate

Use-case name : Activate

Primary Actor : User

Stakeholders and Interests :

- User wants to activate gestures
- Computer Vision Library activates gestures

Pre-Condition : When media player is opened gestures are active automatically. Gestures should be inactive.

Post-Condition : Gestures are activated.

Entry Condition : User should click the ‘activate’ icon.

Exit Condition: User should either click the activate icon again (deactivate icon) or make the ‘deactivate pose’.

Success scenario :

1. User clicks the ‘activate’ icon.
2. Gestures are activated.

Alternate scenario :

1. User either clicks the activate icon again (deactivate icon) or makes the ‘deactivate pose’.

2. Gestures are deactivated again.

4.1.2.Scenario: Deactivate

Use-case name : Deactivate

Primary Actor : User

Stakeholders and Interests :

- User wants to deactivate gestures
- Computer Vision Library deactivates gestures

Pre-Condition : Gestures should be active.

Post-Condition : Gestures are deactivated.

Entry Condition : User should either click the ‘deactivate’ icon or make the ‘deactivate’ pose

Exit Condition : User should click the ‘deactivate’ icon again (activate icon).

Success scenario :

1. User clicks the ‘deactivate’ icon or makes the deactivate pose.
2. Gestures are deactivated.

Alternate scenario :

1. User clicks the ‘deactivate’ icon again (activate icon).
2. Gestures are activated again.

4.1.3.Scenario: Play

Use-case name : Play
Primary Actor : User
Stakeholders and Interests :

- User wants to play the video.
- Media player plays the video.

Pre-Condition : File format (container) should be available and video should be paused.
Post-Condition : Video is played.
Entry Condition : User should either click the ‘play’ icon from the player bar or make the ‘play’ pose.
Exit Condition: User can click the ‘pause’ icon or make the ‘pause’ pose.
Success scenario :

1. User clicks the ‘play’ icon or makes the ‘play’ pose.
2. Video is played.

Alternate scenario :

1. User clicks the pause icon or makes the pause pose.
2. Video is paused again.

4.1.4. Scenario: Pause

Use-case name : Pause

Primary Actor : User

Stakeholders and Interests :

- User wants to pause the video.
- Media player pauses the video.

Pre-Condition : Video should be on play.

Post-Condition : Video is paused.

Entry Condition : User should either click the ‘pause’ icon from the player bar or make the ‘pause’ gesture.

Exit Condition: User should either click the ‘pause’ icon again (play icon) from the player bar or make the ‘pause’ pose again (play pose).

Success scenario :

1. User clicks the ‘pause’ icon or makes the ‘pause’ pose.
2. Video is paused.

Alternate Scenario :

1. User clicks the pause icon again (play icon) or makes the pause pose again (play pose).
2. Video is played again.

4.1.5. Scenario: Stop

Use-case name : Stop

Primary Actor : User

Stakeholders and Interests :

- User wants to stop the video and close the media player.
- Computer Vision Library stops the video and closes the media player.

Pre-Condition : -

Post-Condition : Media player is closed.

Entry Condition : User should either click the ‘close’ button that located in the upper right corner or make the ‘stop’ pose.

Exit Condition: **-**

Success scenario :

1. User clicks the ‘close’ button.
2. Media player is closed.

Alternate success scenario:

1. User makes the ‘stop’ pose.
2. Video is stopped.
3. Media player is closed.

4.1.6. Scenario: Display Time and Date

Use-case name : Display time and date

Primary Actor : User

Stakeholders and Interests :

- User wants to display time and date.
- Media player displays time and date.

Pre-Condition : Video should be in full screen and gestures should be active.

Post-Condition : Time and date is displayed.

Entry Condition : User should make the ‘display time and date’ pose.

Exit Condition: **-**

Success scenario :

1. User makes the ‘display time and date’ pose.
2. Time and date is displayed.

4.1.7. Scenario: Sleep Tracking

Use-case name : Sleep tracking

Primary Actor : Computer Vision Library

Stakeholders and Interests :

- If user's eyes are closed more than 60 seconds computer vision library wants to pause the video.
- Computer Vision Library pauses the video.

Pre-Condition : User's eyes must be detected by the camera and must be detected as closed for 60 seconds.

Post-Condition : Video is paused.

Entry Condition : User 's eyes must be closed for 60 seconds.

Exit Condition: **-**

Success scenario :

1. User's eyes are detected as closed for 60 seconds.
2. Video is paused.

4.1.8. Scenario: Mute

Use-case name : Mute

Primary Actor : User

Stakeholders and Interests :

- User wants to mute the video.
- Media player mutes the video.

Pre-Condition : The voice of the video should be bigger than zero.

Post-Condition : Video is muted

Entry Condition : User should either click the 'mute' icon or make the 'mute' pose.

Exit Condition: User should either click the 'mute' icon again (unmute) or make the 'mute' pose again.

Success scenario :

1. User either clicks the 'mute' icon or makes the 'mute' pose.
2. Video is muted.

Alternate Scenario:

1. User either clicks the 'mute' icon again (unmute) or makes the 'mute' pose again.
2. Video is unmuted.

4.1.9. Scenario: Volume Up

Use-case name : Volume up
Primary Actor : User
Stakeholders and Interests :

- User wants to increase the volume.
- Media player increases the volume.

Pre-Condition : The voice degree of the video should be smaller than 100.
Post-Condition : Volume is increased.
Entry Condition : User should either increase the volume from the player bar or make the ‘volume up’ pose.
Exit Condition: User should decrease the volume from the player bar or make the ‘volume down’ pose.
Success scenario :

1. User either increases the volume from the player bar or makes the ‘volume up’ pose.
2. Volume is increased.

Alternate Scenario:

1. User either decrease the volume from the player bar or make the ‘volume down’ pose.
2. Voice is decreased again.

4.1.10. Scenario: Volume Down

Use-case name : Volume down
Primary Actor : User
Stakeholders and Interests :

- User wants to decrease the volume.
- Media player decreases the volume.

Pre-Condition : The voice degree of the video should be bigger than 0.
Post-Condition : Volume is decreased.
Entry Condition : User should either decrease the volume from the player bar or make the ‘volume down’ pose.
Exit Condition: User should increase the volume from the player bar or make the ‘volume up’ pose.
Success scenario :

1. User either decreases the volume from the player bar or makes the ‘volume down’ pose.
2. Volume is decreased.

Alternate Scenario:

1. User either increase the volume from the player bar or make the ‘volume up’ pose.
2. Voice is increased again.

4.1.11. Scenario: Full Screen

Use-case name : Full screen

Primary Actor : User

Stakeholders and Interests :

- User wants to make the video full screen.
- Media player makes the video full screen.

Pre-Condition : Video should not be in full screen.

Post-Condition : Video becomes full screen.

Entry Condition : User should either click the ‘full screen’ icon from the player bar or make the ‘full screen’ pose.

Exit Condition: **-**

Success scenario :

1. User clicks the full screen icon or makes the full screen pose.
2. Video becomes full screen.

4.1.12. Scenario: Move Forwards

Use-case name : Move forward

Primary Actor : User

Stakeholders and Interests :

- User wants to move the video forwards.
- Media player makes the video move forwards.

Pre-Condition : -

Post-Condition : Video is moved forwards.

Entry Condition : User should drag the progress bar forwards or make the ‘move forwards’ pose.

Exit Condition: User should drag the progress bar backwards or make the ‘move backwards’ pose.

Success scenario :

1. User drags the progress bar forwards or make the ‘move forwards’ pose.
2. Video is moved forwards.

Alternate scenario:

1. User drags the progress bar backwards or make the ‘move backwards’ pose.
2. Video is moved backwards again.

4.1.13. Scenario: Move Backwards

Use-case name : Move backward

Primary Actor : User

Stakeholders and Interests :

- User wants to move the video backwards.
- Media player makes the video move backwards.

Pre-Condition : -

Post-Condition : Video is moved backwards.

Entry Condition : User should drag the progress bar backwards or make the ‘move backwards’ pose.

Exit Condition: User should drag the progress bar forwards or make the ‘move forwards’ pose.

Success scenario :

1. User drags the progress bar backwards or make the ‘move backwards’ pose.
2. Video is moved backwards.

Alternate scenario:

1. User drags the progress bar forwards or make the ‘move forwards’ pose.
2. Video is moved forwards again.

4.1.14.Scenario: Help

Use-case name : Help

Primary Actor : User

Stakeholders and Interests :

- User wants to have information about gestures, what do they do and how to use them
- Media player shows what the gestures are and what do they do.

Pre-Condition : -

Post-Condition : -

Entry Condition : User should click the ‘help’ button.

Exit Condition: User should click the ‘close’ button.

Success scenario :

1. User clicks the help button
2. Gestures and what do they do are shown.

4.1.15. Scenario: Settings

Use-case name : Settings

Primary Actor : User

Stakeholders and Interests :

- User wants to change the settings
- Media player views and updates the settings.

Pre-Condition : At first settings are in default.

Post-Condition : Settings are viewed and updated.

Entry Condition : User should click the ‘settings’ button.

Exit Condition: User should click the ‘close’ button.

Success scenario :

1. User clicks the ‘settings’ button.
2. Settings are viewed and updated.

Alternate scenario:

1. User clicks the ‘settings’ button.
2. Settings are viewed.
3. User clicks the close button.
4. Settings are closed.

4.1.16.Scenario: Customize Jump

Use-case name : Customize jump

Primary Actor : User

Stakeholders and Interests :

- User wants to change the amount of time video is moved forwards or backwards.
- Media player changes the amount of time.

Pre-Condition : User should be in settings.

Post-Condition : Customize jump setting is updated.

Entry Condition : User should click the ‘customize jump’ button.

Exit Condition: User should click the ‘accept’ button.

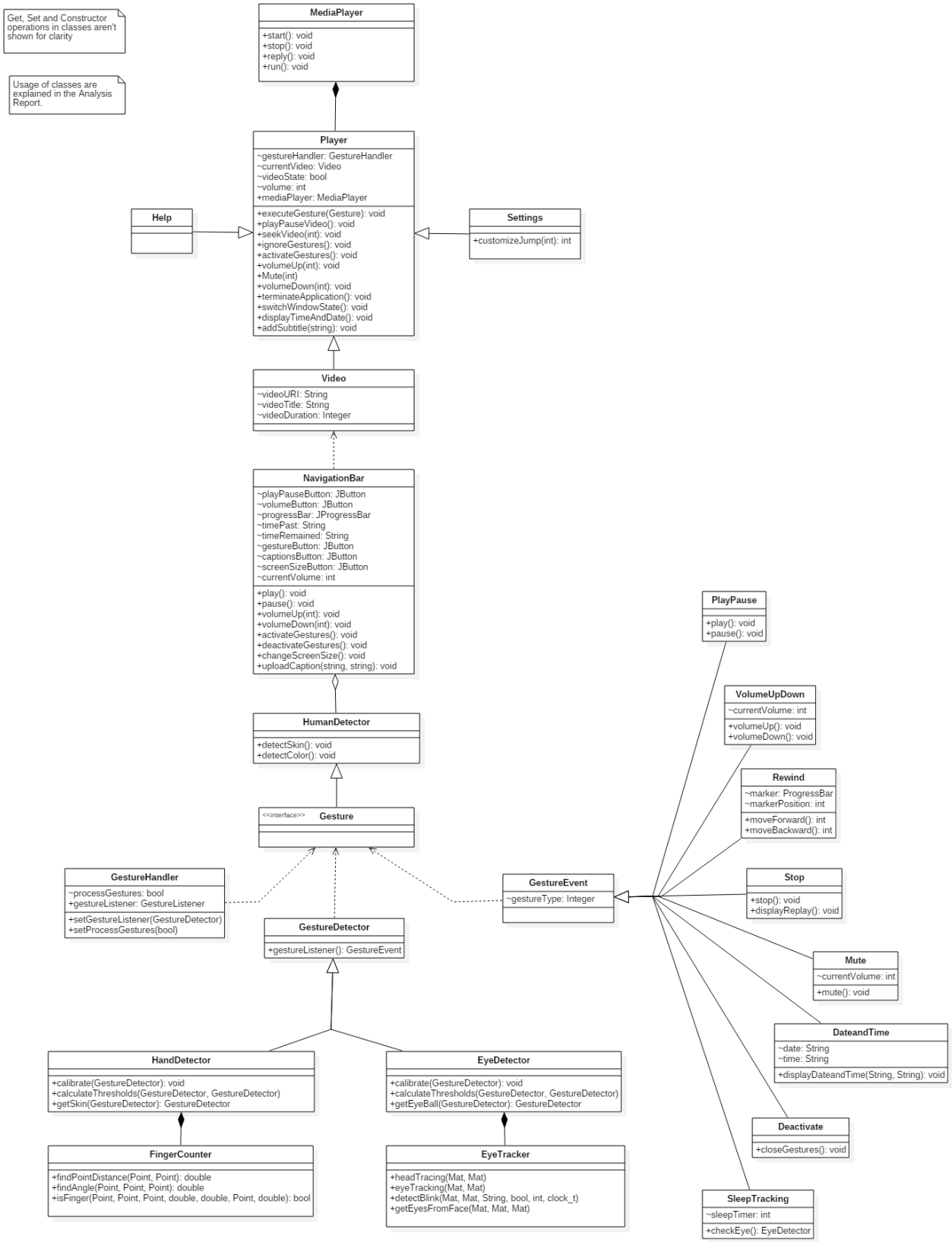
Success scenario :

1. User clicks the ‘customize jump’ button.
2. User enters the amount of time.
3. User clicks the ‘accept’ button.
4. Amount of time is updated.

Alternate scenario:

1. User clicks the ‘customize jump’ button.
2. User clicks the close button.
3. Settings are closed.

4.2. Class Diagram – Object and Operations



For starters we have a class called Player which is the Façade class of our whole media player system. Player class consists of all necessary interactions (at this context functions) with a mouse or a keyboard. We have three main classes inherited from the Player class.

1. Help
2. Settings and
3. Video

Video class is related to the NavigationBar class that allows us to see the transparent bar and control our media player's functionality (such as play, pause, increase volume etc.) We also have a HumanDetector class to discover if there is a person in front of the camera.

Our media player also has a Gesture interface because we have to detect the gesture (GestureDetector class), understand what gesture it is (GestureEvent class) and finally handle the gesture (GestureHandler class).

We have eight different gesture events that are inherited from the GestureEvent class.

1. PlayPause
2. VolumeUpDown
3. Rewind
4. Stop
5. Mute
6. DateandTime
7. Deactivate
8. SleepTracking

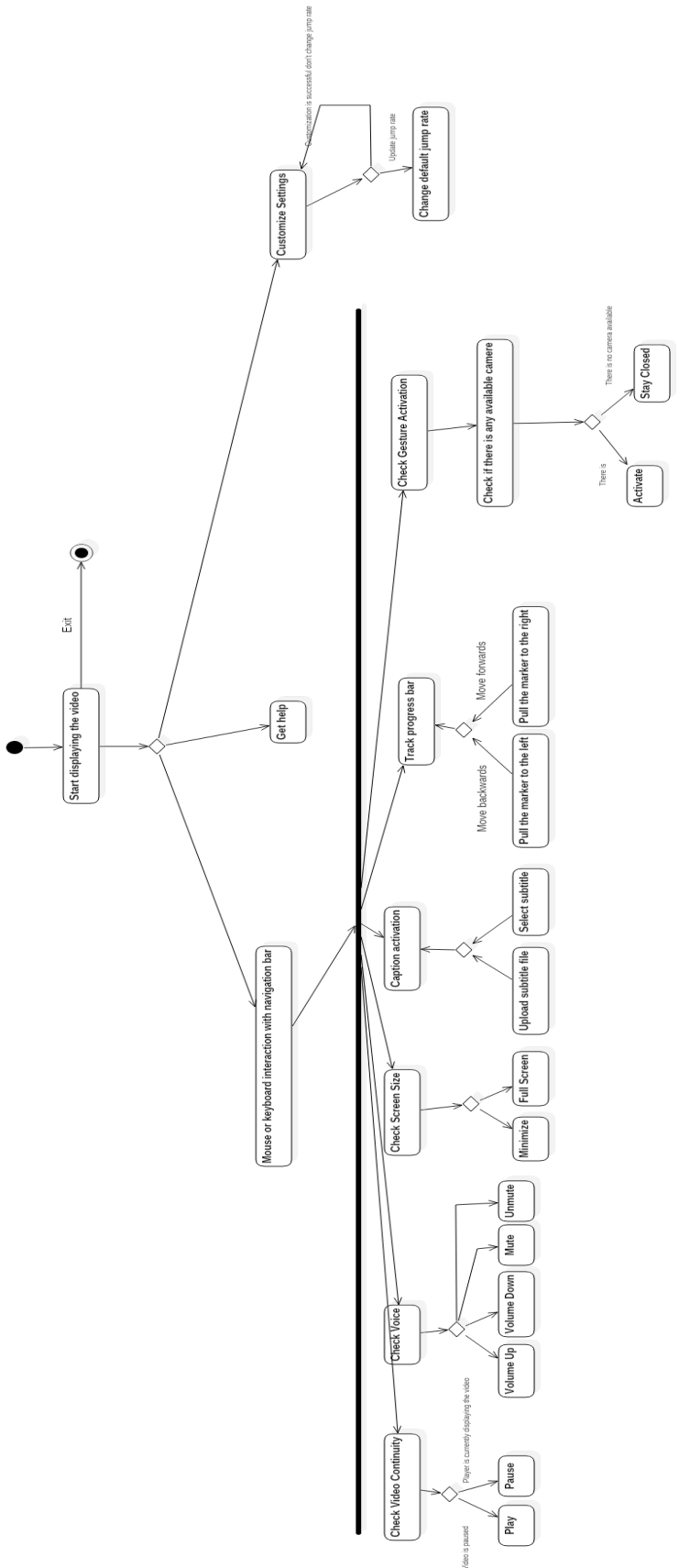
Ultimately, we have two different classes inherited from the GestureDetector class.

1. HandDetector
2. EyeDetector

HandDetector class has a 'FingerCounter' which decides are they really fingers, if so, what pose do those fingers (and hand of course) make.

Lastly our EyeDetector class has a 'EyeTracker' which traces the head and eyes of the user, detects blinking and if the eyes are open.

4.3. Activity Diagram



When user opened a video file, media player starts displaying the video, then waits for a user interaction. If user selects help option, help page will be displayed. If user selects customize settings, he/she can change default jump rate. Then changes will be updated. User can use default jump rate if he/she wants. If user performed an interaction with navigation bar using mouse or keyboard media player checks what kind of an interaction was it. (What did or should this interaction do).

If user clicked the play or pause icons, player checks the video continuity. If the video is paused, video is played; if it is on play (player is currently displaying the video), video is paused.

If user clicked the volume icon or somewhere on its slider, media player checks the voice. If user wanted to increase the volume, media player increases it. If user wanted to decrease the volume, media player decreases it. If user clicked on the icon media player mutes the video if it is unmuted; if the video is on mute, player unmutes the video.

If user clicked the full screen icon, media player checks the screen size. If the screen is on full, player minimizes it; if the screen is minimized, it becomes full screen.

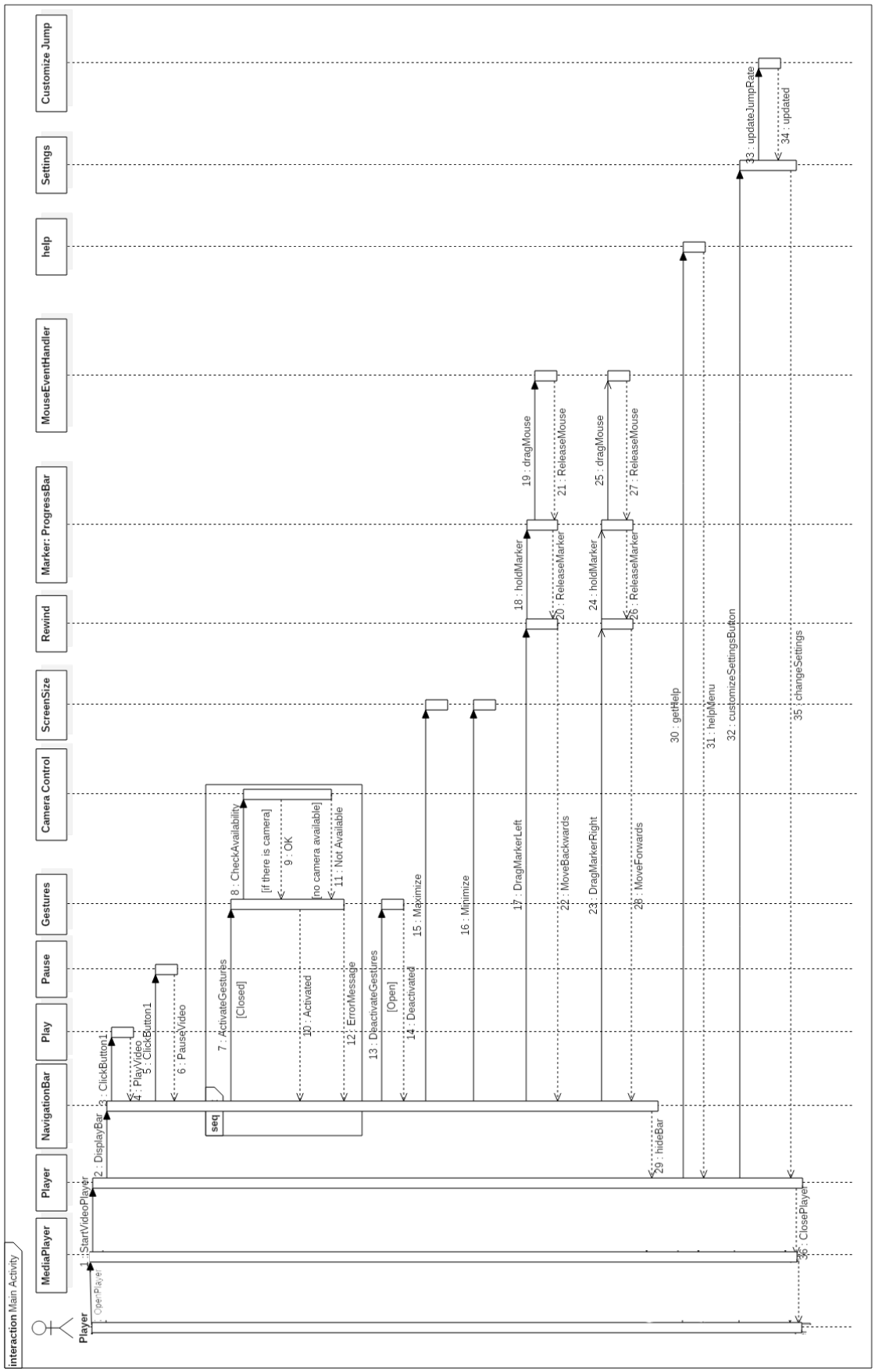
If user clicked the subtitles icon, media player performs caption activation. User can use the subtitle (in English) that we already have or he/she can select a subtitle file and upload it to the video.

If user dragged progress bar, media player tracks the progress bar. If user pulled the marker to the left, video is moved backwards, if user pulled it to the right, video is moved forwards.

Lastly; if user clicked the ‘activate gestures’ icon, media player checks gesture activation. Before allow the activation instantly it checks if there is any camera available. If there is player activates the gestures. If there are no cameras available, gestures stay deactivated.

4.4. Sequence Diagrams

Image 4.1: Main Activity of Appplication



A user opens the media player with the convenient file format. Then media player, starts the video player and the player displays the bar. If user clicks the play button, video is played; if user clicks the pause button, video is paused.

Then user requests to activate the gestures. If gestures are deactivated; player checks if there is an available camera. If there is, gestures are activated. If there are no cameras to detect gestures, it displays an error message.

Similarly if user requests to deactivate the gestures, player checks if the gestures are active. If they are, gestures are deactivated.

Our user –after enabling the gestures- lays down in his/her bed and starts to watch the video. But he/she should make the screen full. So user demands full screen. If the screen is minimized, it becomes full.

Some time later user gets bored and wants to move forward in the video. At this point player waits for an input. If user drags the marker to the left as long as he/she wants (while holding the marker with the cursor-mouse), video will move backwards after user releases the marker. Similarly; if user drags the marker to the right as long as he/she wants, video will move forwards (afterwards user releases the marker).

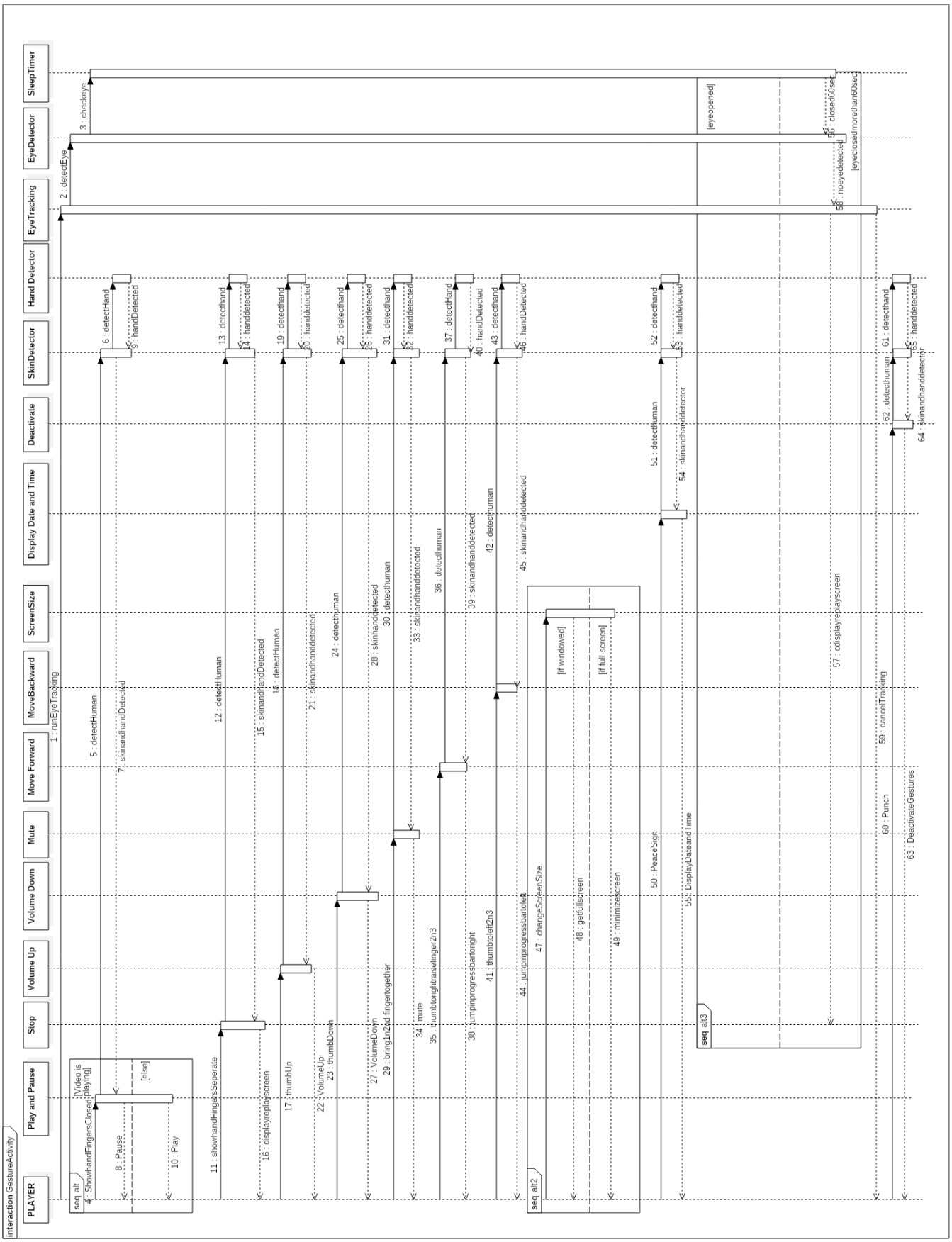
Suddenly our user forgets how a gesture works. He/she requests help. Player displays the help page.

User may want to change the time interval for moving the video forwards or backwards. He/she desires to change it (customize it). So player displays the settings page.

After finish watching the video , our user closes the media player.

As we can see the vast majority of the operations are performed in the NavigationBar. MouseEventHandler helps us users interact with the media player.

Image 4.2.:Gesture Activities After enabling Gesture Button on NavigationBarImage



While starting, media player essentially runs the eye tracking. It detects the eye and checks it continuously to catch if user closed his/her eyes.

User wants to control the video he/she is watching with the gestures. The first thing our user requests is to play or pause the video. He/she shows her hand his/her fingers closed to the camera. Media player detects a human, detects human's hand and play the video if it is on pause, pause the video otherwise.

Then our user show his/her hand with his/her fingers separated to the camera. Media player searches for a human, detects a human hand and stops the video. Replay screen is displayed.

If user shows thumbs up pose to the camera; media player detects a human and a human hand and volume is increased. Similarly; if user shows thumbs down pose to the camera; media player detects a person and person's hand and volume is decreased.

If our user makes a pose by connecting his/her thumb and index finger; media player detects user's hand and mutes the video.

If user wants to move forwards in the video he/she shows the 'move forwards' pose to the camera (index and middle fingers up, thumb is pointed to the right). Media player detects human and a human hand and the video is moved forwards.

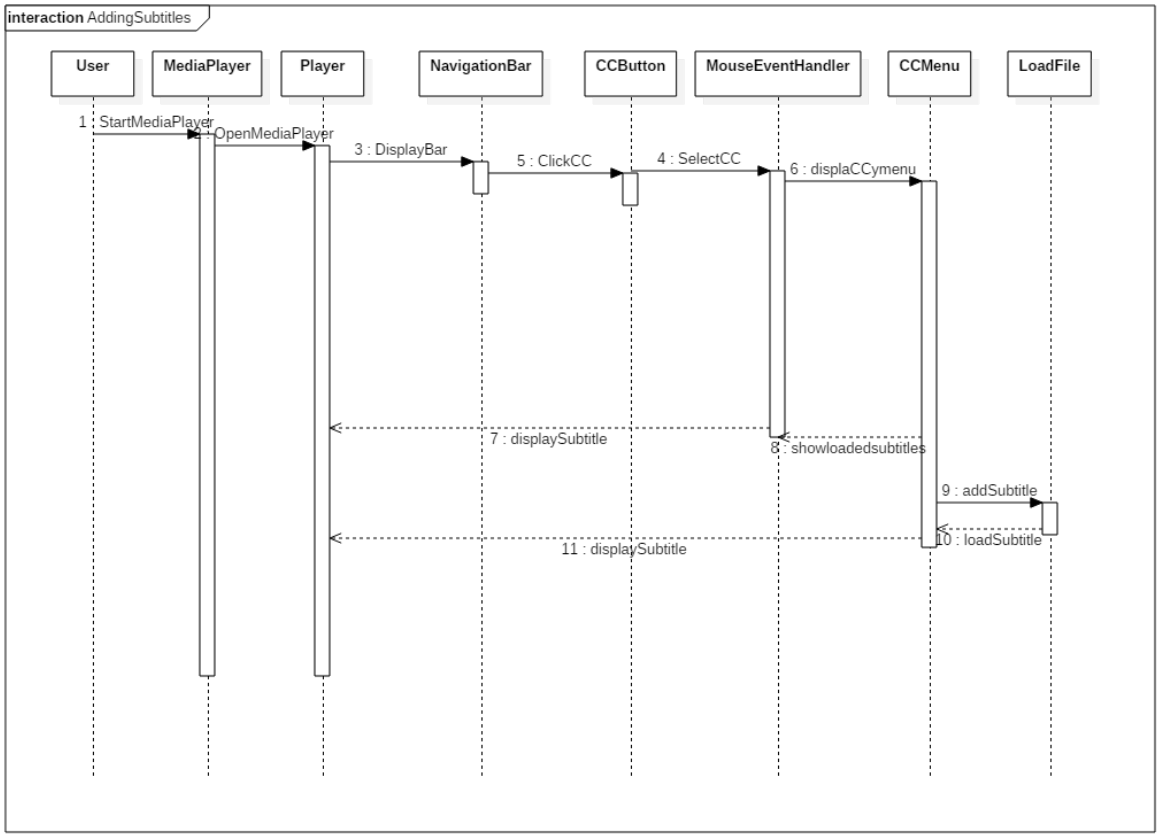
Similarly; if user wants to move backwards in the video he/she shows the 'move backwards' pose to the camera (index and middle fingers up, thumb is pointed to the left). Media player detects human and a human hand and the video is moved backwards.

When user requests to change screen size to full screen, he/she makes the 'full screen' pose (index finger is pointed upwards). If screen is minimized then media player makes the screen full; else the screen becomes minimized.

Our user may become curious about time or date while watching a video in full screen. If so, it is enough for he/she to make a peace sign. Media player detects the person and the person's hand and shows the time and the date.

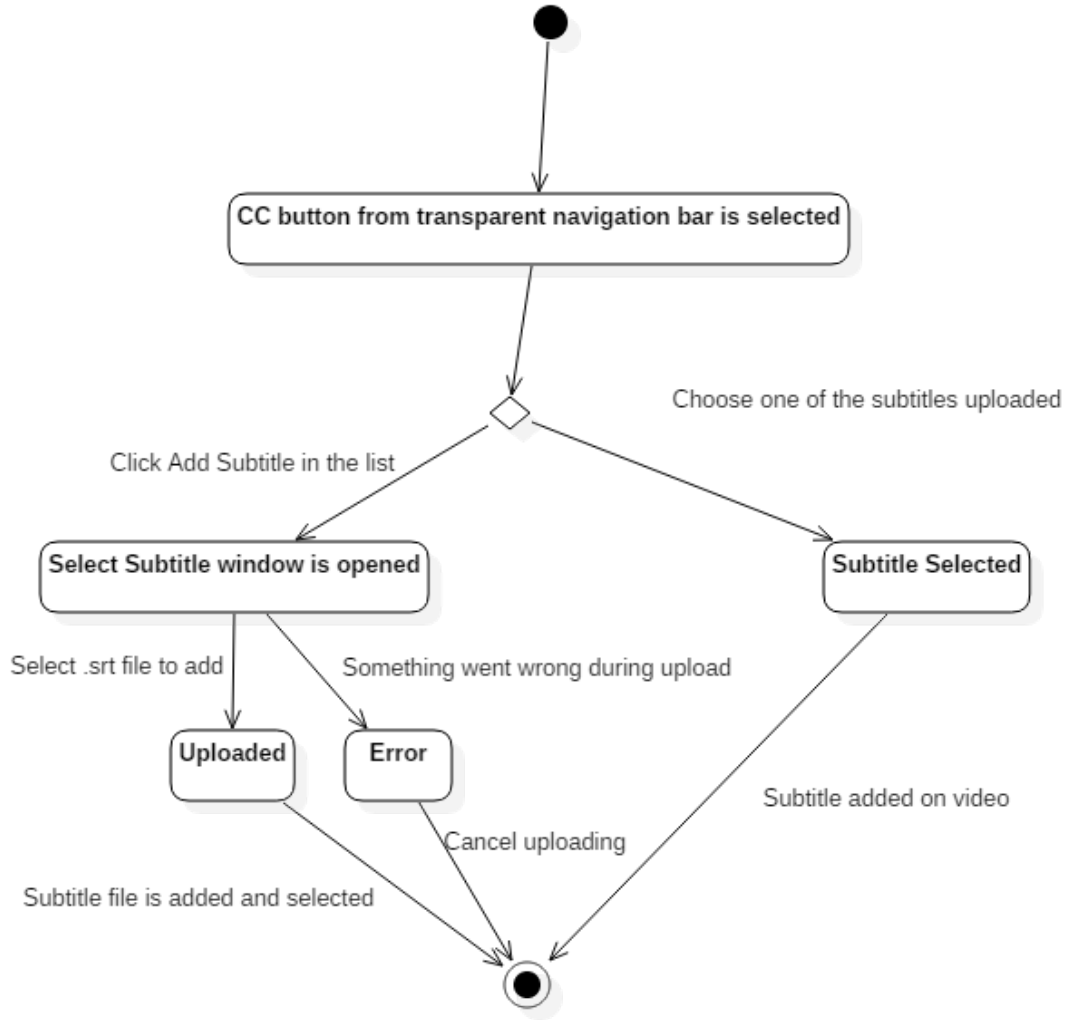
If user falls asleep while watching, eye tracking which checks his/her eyes continuously detects it and media player displays the replay screen.

Lastly, if our user does not want neither any of the gestures nor eye tracking to work, he/she can simply show his/her fist (a punch) to the camera. This will allow the media player detect a human and a human hand. So it will cancel eye tracking and deactivate all the gestures.



A user may desire to add subtitles to the video he/she will watch. User starts the media player by selecting an appropriate file with an appropriate format (container). After the player is opened navigation bar is displayed in the player. When user clicked the subtitles icon, at this point he/she has two options. He/she can either use previously added subtitles or add new subtitles from a file. If user wants to use previously loaded subtitles, he/she can choose it and the subtitles will be displayed. If our user wants to add new subtitles, he/she can choose an appropriate file with an appropriate container and load it to the subtitles menu. When user chooses it, it will be displayed on the screen.

4.5. Statechart Diagram for Adding Subtitles



Statechart diagram for adding subtitles is illustrated above. When user wishes for adding subtitles to the video, he/she will select the CC button from the transparent bar.

At this point; if add subtitle option is clicked, select subtitle window will be opened. User will select a subtitle file. If the file is an appropriate format ,which is the .srt format, subtitle will be uploaded. If the file container is not supported an error message will be displayed and uploading will be canceled.

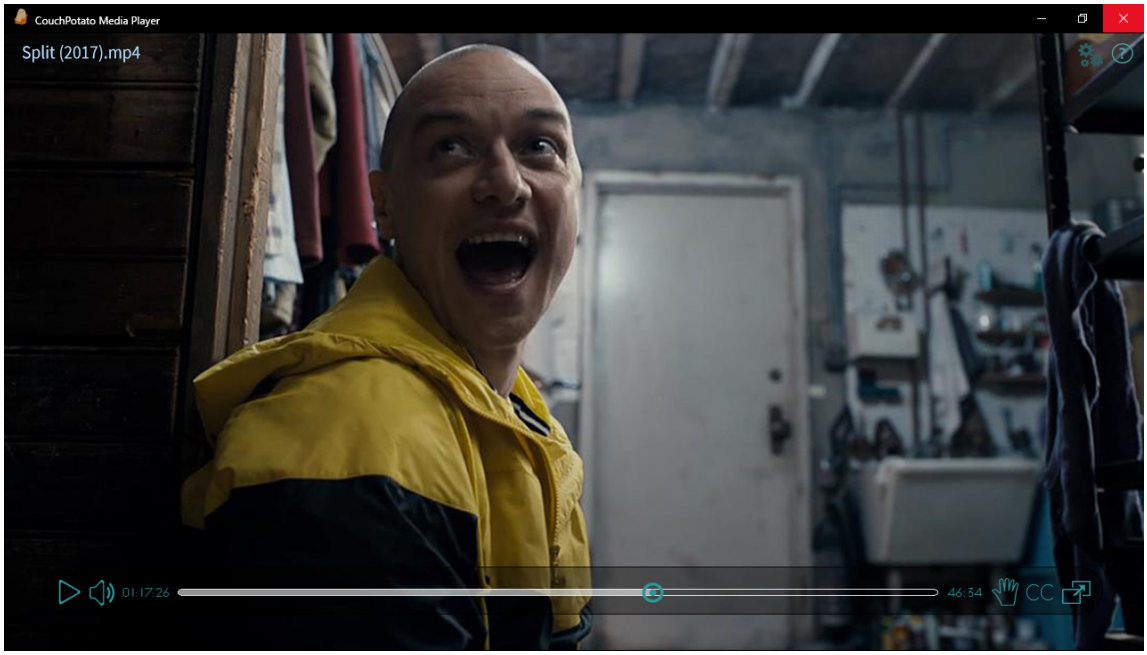
If user wishes to use one of the subtitles that is already uploaded, he/she will choose one of the subtitles (subtitle will be selected) and that subtitle will be added to the video.

5. User Interface

5.1. Screen Mockups

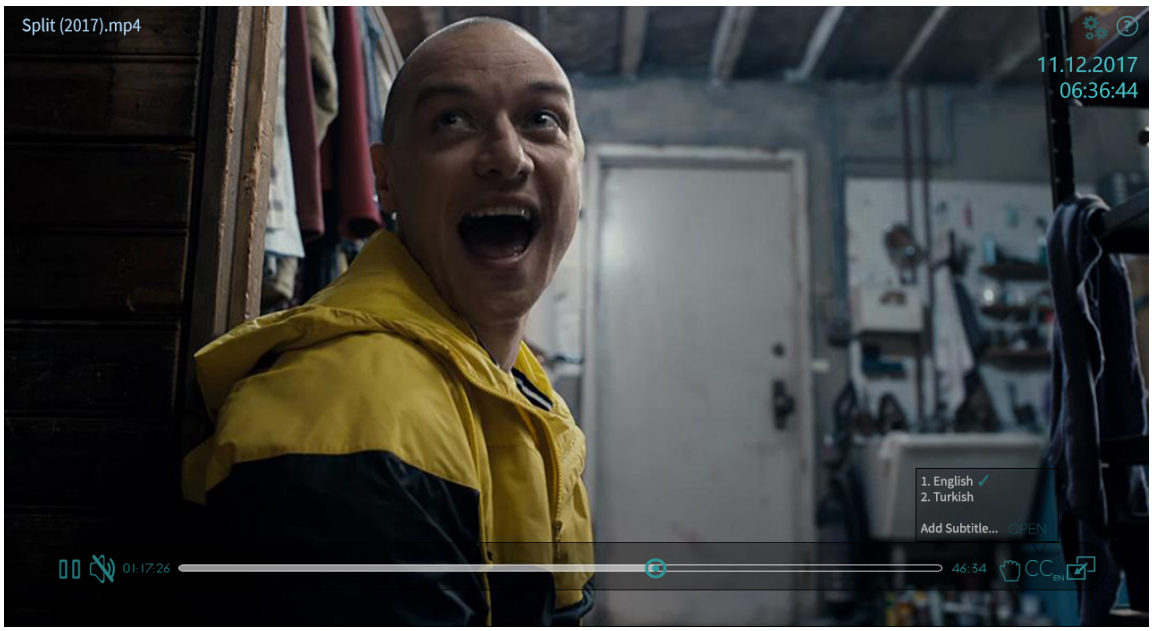
5.1.1. Media Player Interface

Image 5.1: Default Player



In the first image, we see that the application continues with its default settings when it is opened. In the upper left corner there is the file name and in the upper right corner there are settings and help buttons. The buttons in the transparent bar are respectively: Play, Unmuted, Gestures Enabled, No Subtitles, Window Mode Screen.

Image 5.2: Functions enabled



The second picture shows that some functions are executed. The video stream is in progress, muted, and gestures are disabled. English and Turkish subtitles are automatically defined because two .srt files are in the same folder with the video file. The screen is in full-screen mode.

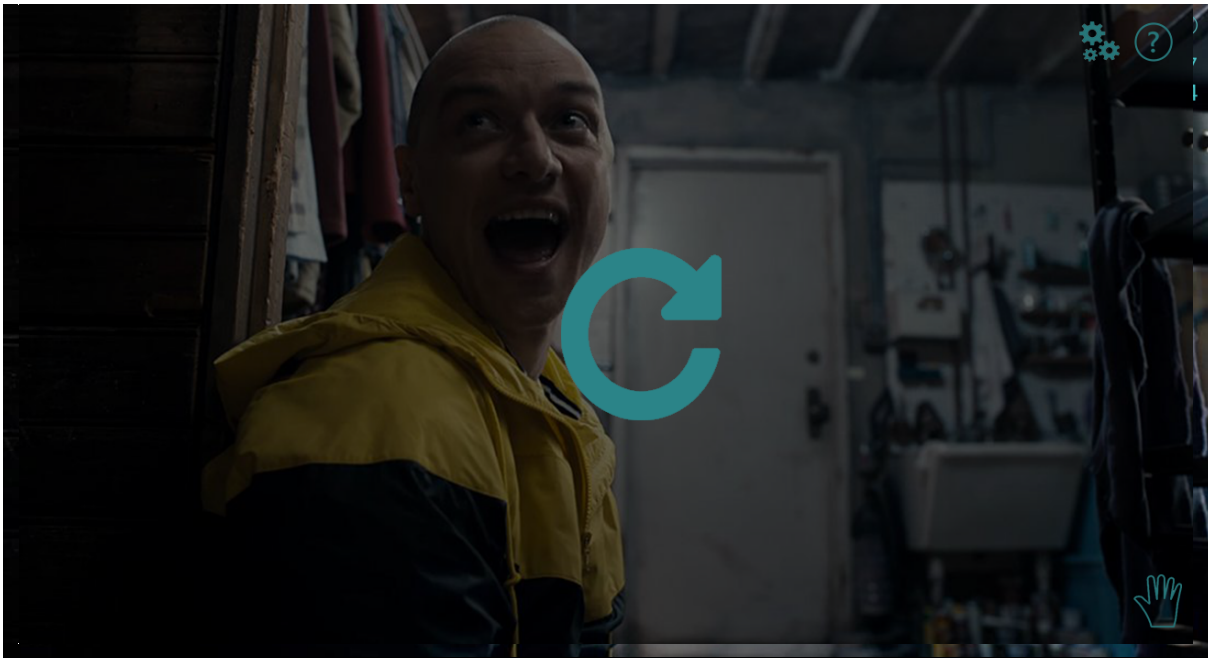


Image 5.3: Video is stopped

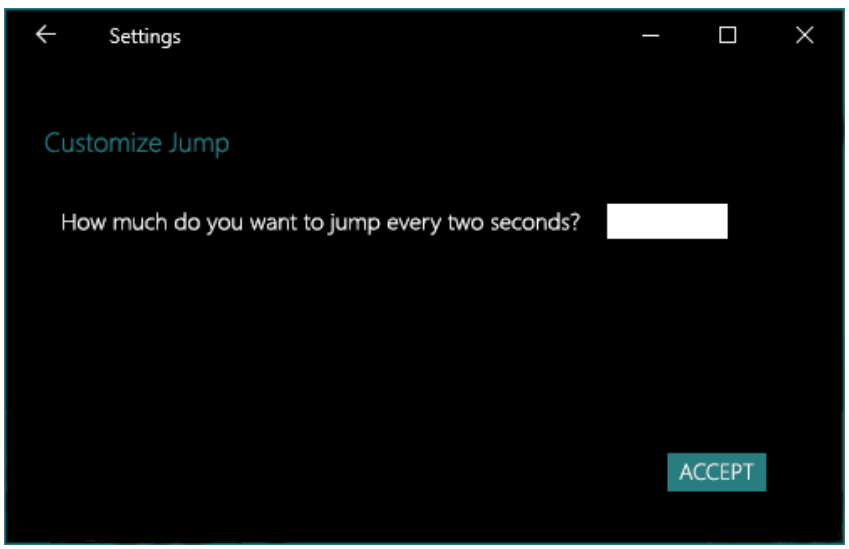
In the third picture, the sleep tracking function is activated or the user showed his / her hand as shown in the figure 3.1.5.. In short terms, it is the ‘stop screen’.

5.1.2.Customize Gestures

Image 5.4: Settings Window

It is possible for the user to edit the settings screen to determine which gestures to use. At the bottom of the window there is an extension to guide the help window to get more information. There is also an additional setting window for the user to define as much time as the user wishes to rewind the video. (Image 5.4.)

Image 5.5: Customize Jump Time



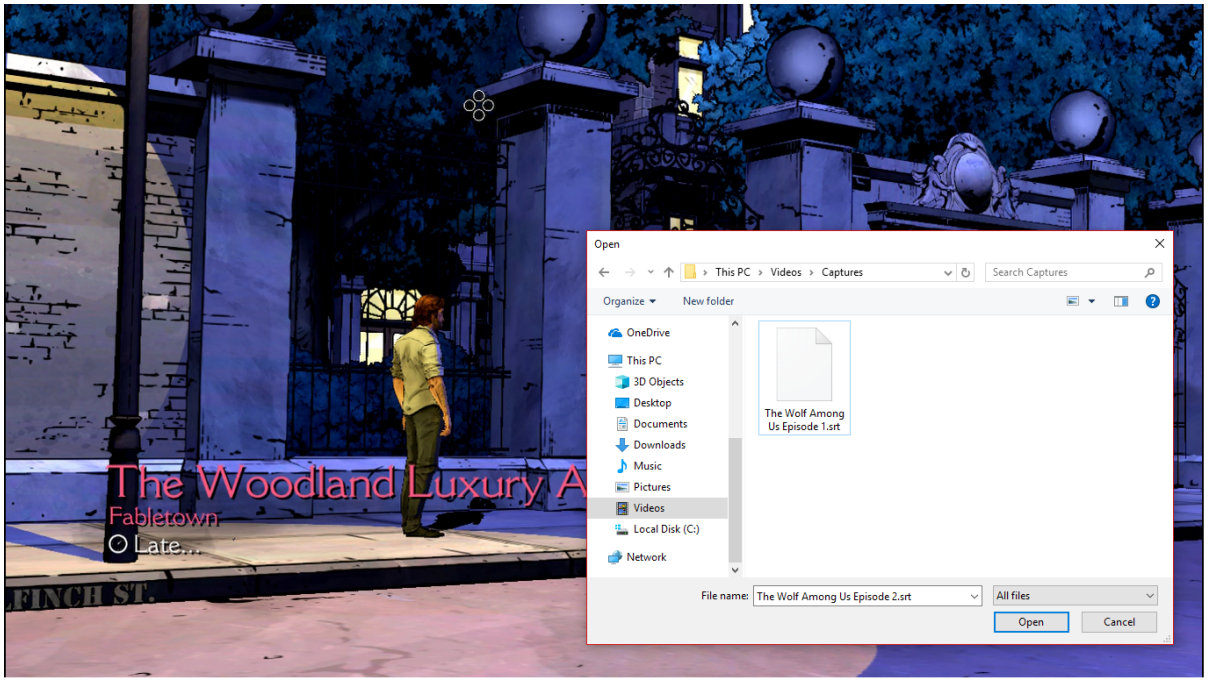
5.1.3. Help

The help screen can be accessed from the question mark in the upper right corner of the player and from the extension in the settings section. It gives information about how to use computer vision technique.

Image 5.6: Help Screen

5.1.4.Adding Subtitles

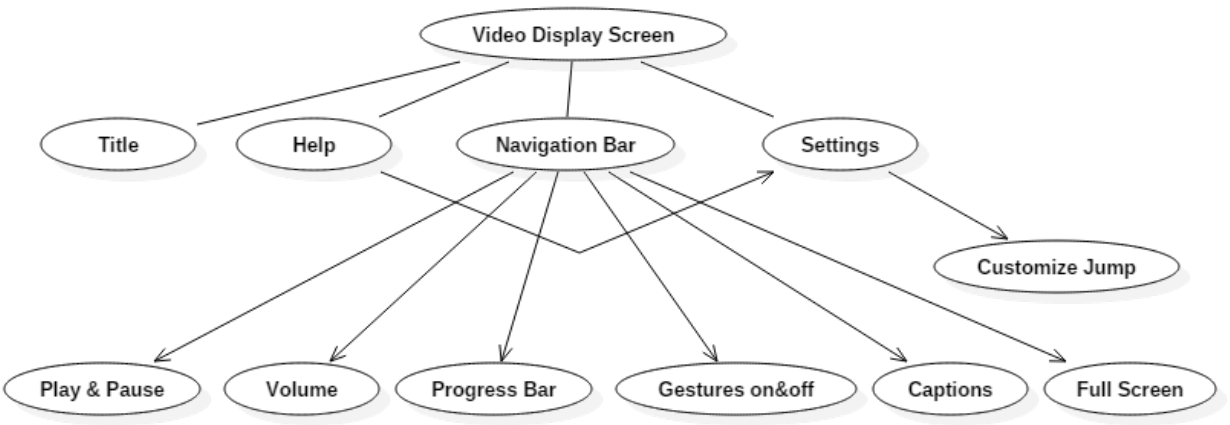
Image 5.7: Adding Subtitle to the Video



The user can define subtitles in two different ways for a video. The first is to automatically identify the video and .srt file by putting them into the same folder. The other way is to select the file and integrate it into the video. (Image 7)

5.2. Navigational Path

Image 5.8: A simple path for user interactions



6. Software Size Estimation

The Function Point Model consists of:

1. External Input (EI)

Functions that move data into the application without presenting data manipulation.

2. External Output (EO)

Functions that move data to the user and presents some data manipulation.

3. Online Inquiries (EQ)

Functions that move data to the user without presenting data manipulation.

4. Logical Files (IF)

The logic in the form of fixed data managed by the application through the use of External Input (EI)

5. External Interface Files (EIF)

The logic in the form of fixed data used by the application but did not run in it

The number of functional components of the system were first identified and followed to evaluated the complexity of quantization weight of each component. Weighting was the summed and become the number of CFP.

Software system components	Complexity level									Total CFP
	Simple			average			complex			
	Count	Weight Factor	Points	Count	Weight Factor	Points	Count	Weight Factor	Points	
	A	B	C= AxB	D	E	F= DxE	G	H	I= GxH	
User inputs	17	3	51	1	4	4	1	6	6	61
User outputs	3	4	12	0	5	0	0	7	0	12
User online queries	0	3	0	1	4	4	1	6	0	4
Logical files	3	7	21	2	10	20	2	15	30	61
External interfaces	0	5	0	0	7	0	0	10	0	0
Total CFP										138

Complexity of Inputs

Play/Pause	*low*
Volume up/down- Mute	*low*
Full screen	*low*
Rewinding	*low*
Stop	*low*
Activation/Deactivation	*low*
Add/Choose subtitle	*low*
Help	*low*
Settings	*average*
Customization of jump	*low*

Gestures	
Play/Pause	*low*
Volume up/down-Mute	*low*
Full screen	*low*
Move forwards	*low*
Move backwards	*low*
Display time & date	*low*
Stop	*low*
Deactivation	*low*
Eye Tracking	*high*

Complexity of Outputs

Display time & date	*low*
Add & Choose subtitle	*low*
Customize Jump	*low*

Complexity of Online Queries

Update date & Time	*average*
Running Camera in background	*high*

Complexity of Logical Files

Media Player	*low*
Tracker	*high*
Detect Gestures	*high*

Navigation Bar	*low*
Player	*average*
Settings	*average*
Help	*low*

RCAF is to calculate the complexity assessment of software system from several characteristics of subject. Rating scale from 0 to 5 is given to each subject that most affect the development effort required.

No	Subject	Grade
1	Requirement for reliable backup and recovery	0 (1) 2 3 4 5
2	Requirement for data communication	0 1 2 (3) 4 5
3	Extent of distributed processing	0 1 (2) 3 4 5
4	Performance requirements	0 1 2 (3) 4 5
5	Expected operational environment	0 1 2 (3) 4 5
6	Extent of online data entries	0 1 (2) 3 4 5
7	Extent of multi-screen or multi-operation online data input	0 1 2 (3) 4 5
8	Extent of online updating of master files	(0) 1 2 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 (2) 3 4 5
10	Extent of complex data processing	0 1 2 (3) 4 5
11	Extent that currently developed code can be designed for reuse	0 1 (2) 3 4 5
12	Extent of conversion and installation included in the design	0 1 2 (3) 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	(0) 1 2 3 4 5
14	Extent of change and focus on ease of use	0 1 2 (3) 4 5
	Total = RCAF	30

Calculating Function Points by the formula

FP = CFP{crude_function_points} x

(0.65 + 0.01 x RCAF{relative_complexity_adjustment_factor})

FP = 138 x (0.65 + 0.01 x 30)

=131.1

PF = 53 for Java Programming Language

LOC {lines_of_code} = FP {function_point} x PF {productivity_factor}

LOC ~= 131.1 x 53

~=6948 lines

7. Important Decisions In Overall Analysis

- We know that it's important to implement a software with object-oriented programming principles. So we intended to use inheritance as much as we can for building a hierarchical structure in our software system. After this phase we thought it is best to use the façade design pattern.

8. Conclusion

We created our analysis report to design and implement a media player that can be controlled with simple hand and face gestures using computer vision technique called CouchPotato. Our report resides three main parts.

1. Requirements Analysis
2. System Model
3. Estimated Size of Software (Lines of Code)

In the requirements analysis part we have thought about the functionalities system should have and user may wants to use. We determined functional and non-functional requirements carefully. We know it is important not to have conflicts, ambiguities, incompleteness or vagueness. We tried our best for not having them. After we were satisfied enough we proceeded to design our system models.

Our system model consists:

1. Use-case model
2. Class diagram
3. Activity diagram
4. Sequence diagram
5. Statechart diagram
6. User interface

According to the requirements of our software we created scenarios to solve our problems. We showed those solutions (usage, structure etc.) in our UML diagrams. The explanation of these diagrams are written below them.

In the user interface part of our analysis report , we created mock-ups for media player interface, gesture settings, help and adding subtitles. We tried to simulate them nicely. One of our main purpose was to see if our interface will be useful when the functionalities will be added. We also have added a navigational path with the use of our use-cases.

For the last part of our analysis report we have done an estimation for the size of our software (lines of code) using function point calculation technique.

9. References

- https://www.researchgate.net/profile/Volkan_Tunali2/publication/262373232_Software_Size_Estimation_Using_Function_Point_Analysis_-_A_Case_Study_for_a_Mobile_Application/links/02e7e5380c2f87e14a000000/Software-Size-Estimation-Using-Function-Point-Analysis-A-Case-Study-for-a-Mobile-Application.pdf
- https://galorath.com/products/software/ten-steps
- https://www.luxoft.com/blog/smukhina/how-to-estimate-application-size-in-function-points/
- http://www.qsm.com/resources/function-point-languages-table
- Implementation of Function Point Analysis in Measuring the Volume Estimation of Software System in Object Oriented and Structural Model of Academic System Dian Pratiwi Trisakti University Jl. Kyai Tapa No.1 Jakarta, 15000, Indonesia
- https://www.linkedin.com/pulse/cosmic-function-point-k%C4%B1sa-bir-giri%C5%9F-selami-bagriyanik