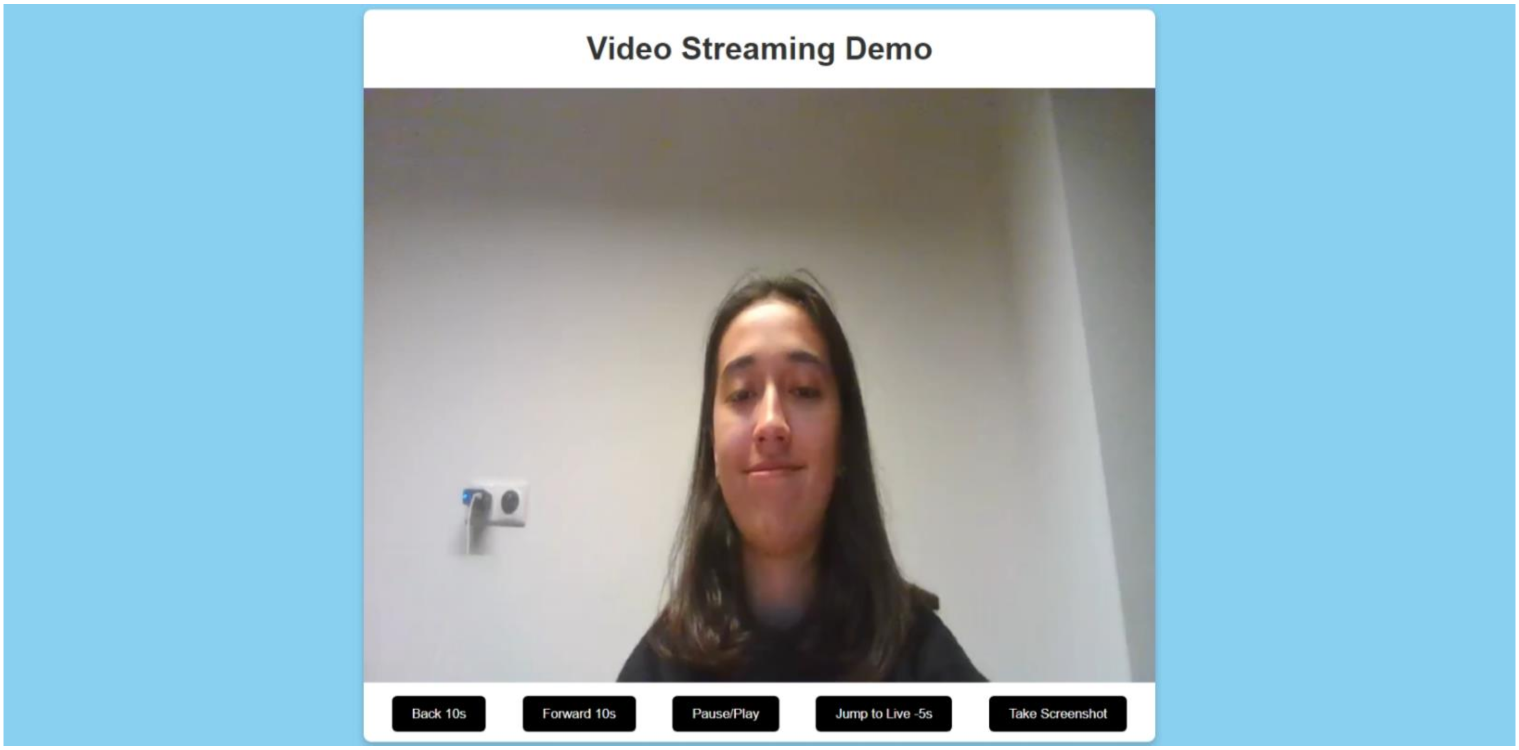
PROJECT 2- VIDEO SURVEILLANCE OVER IP

For the project I created a video streaming service that captures real-time video from webcam, encodes it and streams it to a Web client (in the browser) where users can rewind video 10 seconds, forward video 10 seconds, take a screenshot, and jump to live. I use Flask, a Python web framework used to set up the server for routing and serving web content. I used DASH, Dynamic Adaptive Streaming over http, to enable adaptive streaming of video content in HTML5. I use CSS to style webpage and also Javascript to handle user interactions.

For bonus not only video, but also audio captured, encoded and viewed. I tried to make the UI more appealing.

Here is the final version:

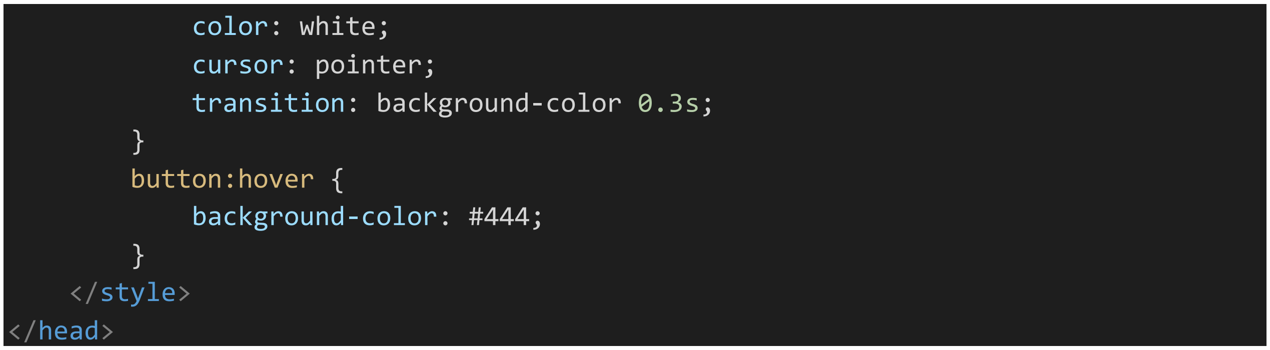


Ffmpeg Command:

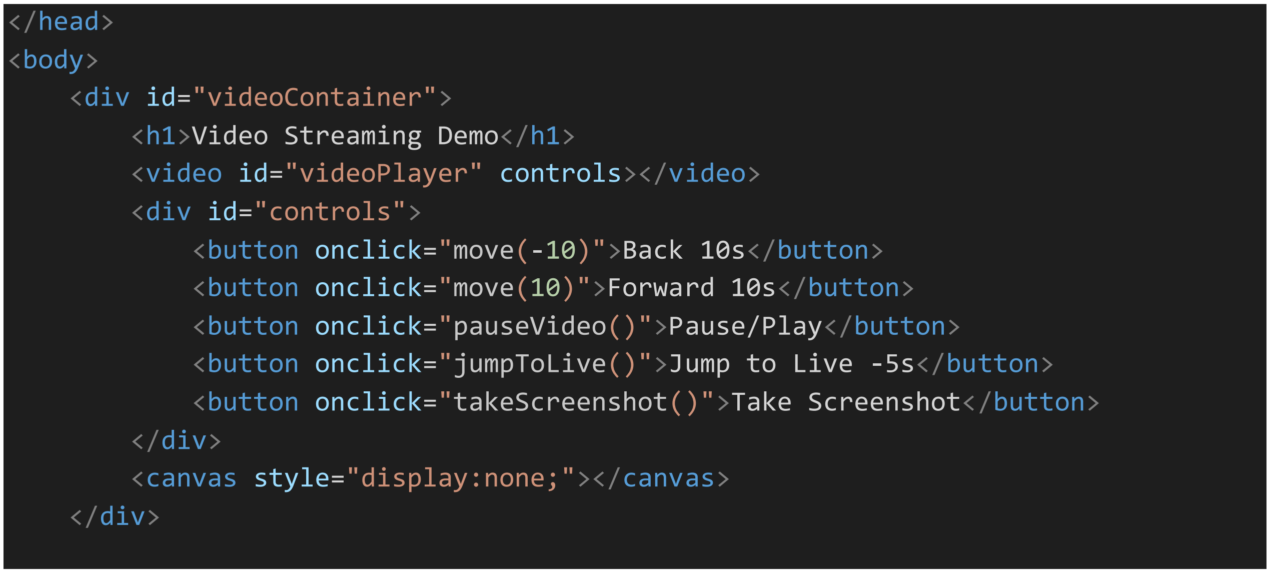
```
ffmpeg
  -f dshow \ specifies that input format is DirectShow
  -i video="Integrated Webcam":audio="Mikrofon (Realtek(R) Audio)" \ defines video and audio input sources
  -c:v libx264 \ H.264
  -preset veryfast \ encoding speed
  -s 640x480 \ output video resolution
  -pix_fmt yuv420p \ pixel format to yuv420p
  -c:a aac \ audio codec
  -b:a 128k \ audio bitrate to 128 kbps
  -ar 44100 \ audio sampling rate
  -g 30 \ group of pictures
  -f dash \
  -remove_at_exit 1 \ when process terminated, necessary files cleaned up
  -seg_duration 4 \ set duration of each segment
  -use_template 1 \ segment template
  -use_timeline 1 \ segment timeline
  -init_seg_name init-$RepresentationID$.m4s \
  -media_seg_name chunk-$RepresentationID$-$Number%05d$.m4s \
  -adaptation_sets "id=0,streams=v id=1,streams=a" \ adaptation settings
  output.mpd \ output file name and format
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Video Stream</title>
    <script src="https://cdn.dashjs.org/latest/dash.all.debug.js"></script>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #89CFF0;
        }
        h1 {
            text-align: center;
            color: #333;
        }
        #videoContainer {
            width: 90%;
            max-width: 800px;
            background: #fff;
            border-radius: 8px;
            box-shadow: 0 4px 8px rgba(0,0,0,0.2);
            overflow: hidden;
        }
        video {
            width: 100%;
        }
        #controls {
            display: flex;
            justify-content: space-around;
            padding: 10px;
            background: rgba(255,255,255,0.9);
        }
        button {
            padding: 10px 20px;
            border: none;
            border-radius: 5px;
            background-color: #000;
```



```
            color: white;
            cursor: pointer;
            transition: background-color 0.3s;
        }
        button:hover {
            background-color: #444;
        }
    </style>
</head>
```



```
</head>
<body>
    <div id="videoContainer">
        <h1>Video Streaming Demo</h1>
        <video id="videoPlayer" controls></video>
        <div id="controls">
            <button onclick="move(-10)">Back 10s</button>
            <button onclick="move(10)">Forward 10s</button>
            <button onclick="pauseVideo()">Pause/Play</button>
            <button onclick="jumpToLive()">Jump to Live -5s</button>
            <button onclick="takeScreenshot()">Take Screenshot</button>
        </div>
        <canvas style="display:none;"></canvas>
    </div>
```

Each button is linked to Javascript function that allows user to interact with the video.

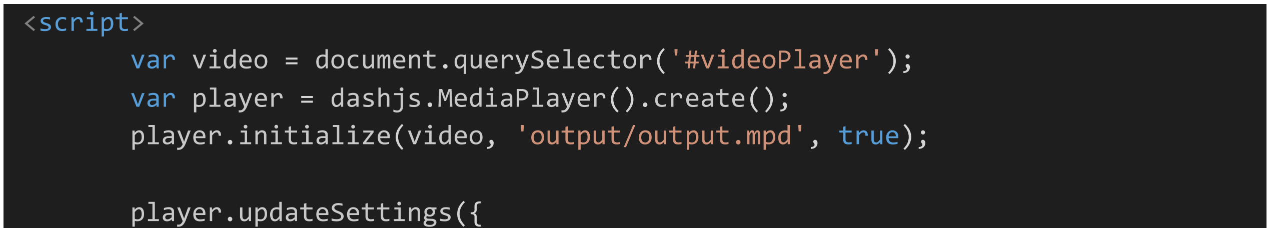
<button onclick="move(-10)">Back 10s</button>: rewind the video 10 seconds.

<button onclick="move(10)">Forward 10s</button>: forward the video 10 seconds.

<button onclick="pauseVideo()">Pause/Play</button>: pause and play video when you click.

<button onclick="jumpToLive()">Jump to Live -5s</button>: move the video to the 5 seconds back from the live stream.

<button onclick="takeScreenshot()">Take Screenshot</button>: captures the current frame and takes screenshot and saves as an image.



```
 <script>
        var video = document.querySelector('#videoPlayer');
        var player = dashjs.MediaPlayer().create();
        player.initialize(video, 'output/output.mpd', true);

        player.updateSettings({
            streaming: {
                delay: {
                    liveDelay: 30,
                }
            }
        });

        function move(seconds) {
            video.currentTime += seconds;
        }

        function pauseVideo() {
            if (video.paused) {
                video.play();
            } else {
                video.pause();
            }
        }

        function jumpToLive() {
            var seekable = video.seekable;
            if (seekable.length > 0) {
                var livePoint = seekable.end(seekable.length - 1) - 10;
                video.currentTime = livePoint > seekable.start(seekable.length -
1) ? livePoint : seekable.start(seekable.length - 1);
            }
        }

        function takeScreenshot() {
            var canvas = document.querySelector('canvas');
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            var ctx = canvas.getContext('2d');
            ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
            var dataURI = canvas.toDataURL('image/png');
            var a = document.createElement('a');
            a.href = dataURI;
            a.download = 'screenshot.png';
            document.body.appendChild(a);
            a.click();
            document.body.removeChild(a);
        }
    </script>
</body>
</html>
```

server.py

```
import subprocess
from flask import Flask, send_from_directory
from flask_cors import CORS
import configparser
import threading
import os
```

subprocess: to use Ffmpeg command.

flask: main framework for creating web server.

CORS: flask extension that handles Cross-Origin Resource Sharing, allowing your resources to be accessed by web pages from different domains.

```
config = configparser.ConfigParser()
config.read('config.ini')
```

Loads settings from my configuration file.

```
app = Flask(__name__, static_folder='output')
CORS(app)
```

Applied CORS to the Flask app to allow cross-domain requests.

```
@app.route('/output/<path:filename>')
def dash_content(filename):
    return send_from_directory(app.static_folder, filename)

@app.route('/')
def index():
    return send_from_directory(app.root_path, 'index.html')
```

dash_content: Serves dash content files from output folder.

index: Serves main page.

```
def run_ffmpeg():
    os.chdir('./output')

    cmd = (
        f"ffmpeg -f dshow -i video=\"{config['FFMPEG']['camera_name']}\":"
        f"audio=\"{config['FFMPEG']['microphone_name']}\" -c:v libx264 -preset veryfast "
        "-s 640x480 -pix_fmt yuv420p -c:a aac -b:a 128k -ar 44100 -g 30 -f dash "
        "-remove_at_exit 1 -seg_duration 6 -use_template 1 -use_timeline 1 "
        "-init_seg_name init-$RepresentationID$.m4s "
        "-media_seg_name chunk-$RepresentationID$-$Number%05d$.m4s "
        "-adaptation_sets \"id=0,streams=v id=1,streams=a\" output.mpd"
    )
    subprocess.run(cmd, shell=True)
```

Constructs and executes Ffmpeg command that captures video and audio.

```
def run_flask():
    app.run(debug=True, host=config['SERVER']['ip'],
port=int(config['SERVER']['port']), use_reloader=False)
```

Runs the Flask server with given settings from configuration file.

```
if __name__ == '__main__':
    flask_thread = threading.Thread(target=run_flask)
    ffmpeg_thread = threading.Thread(target=run_ffmpeg)
    flask_thread.start()
    ffmpeg_thread.start()
```

When script is executed flask_thread and ffmpeg_thread run as separated threads ensuring that both operations run concurrently.

<u>How to run:</u>

1-Ffmpeg and python should be installed.

2-Run the commands:

```
pip install Flask

pip install flask-cors
```

3- To execute the script open server.py with the following command:

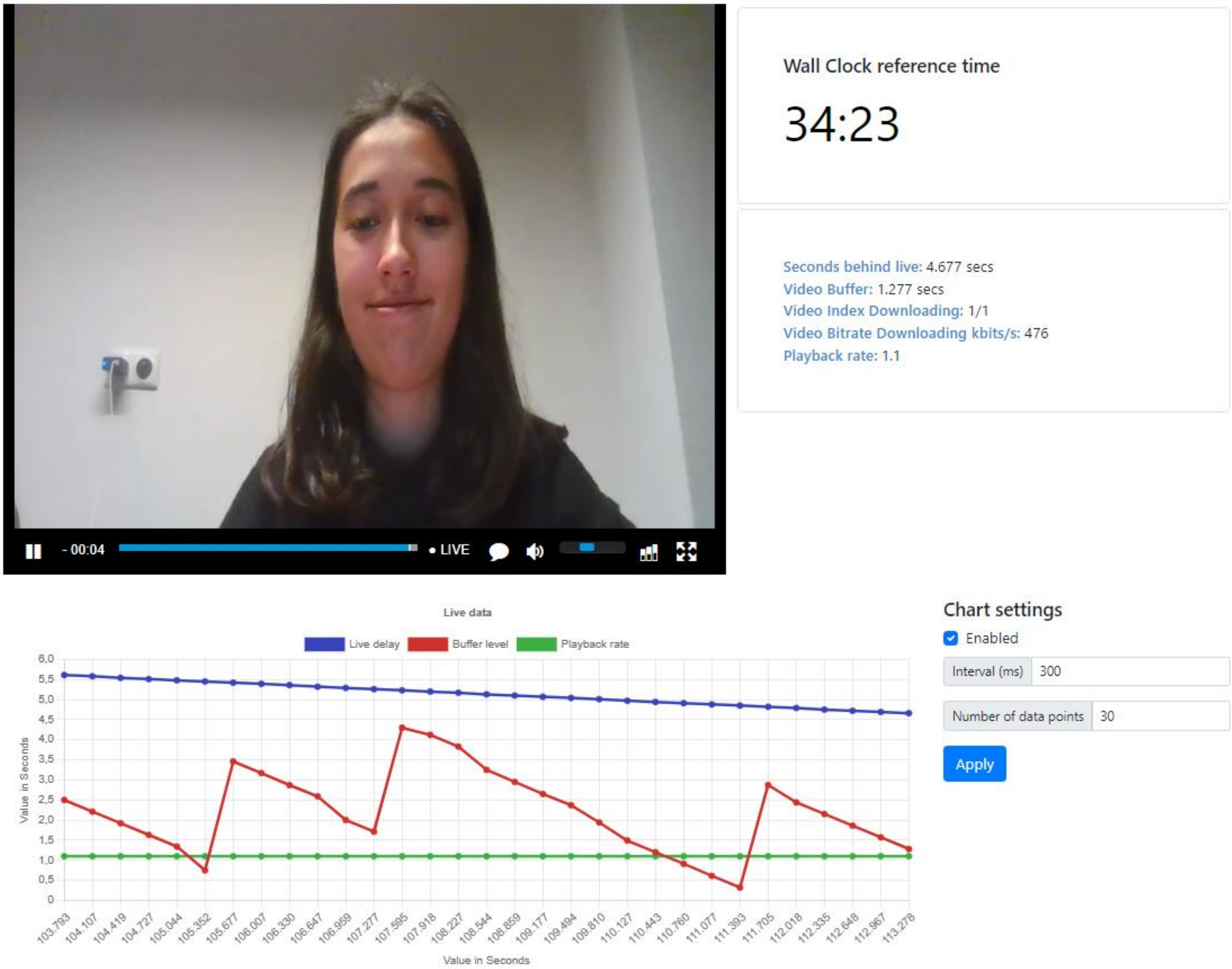
```
Python server.py
```

4- Open this link in your browser:

http://127.0.0.1:2000/

Test Results:

Segment Size: 2



Wall Clock reference time

34:23

Seconds behind live: 4.677 secs
Video Buffer: 1.277 secs
Video Index Downloading: 1/1
Video Bitrate Downloading kbits/s: 476
Playback rate: 1.1

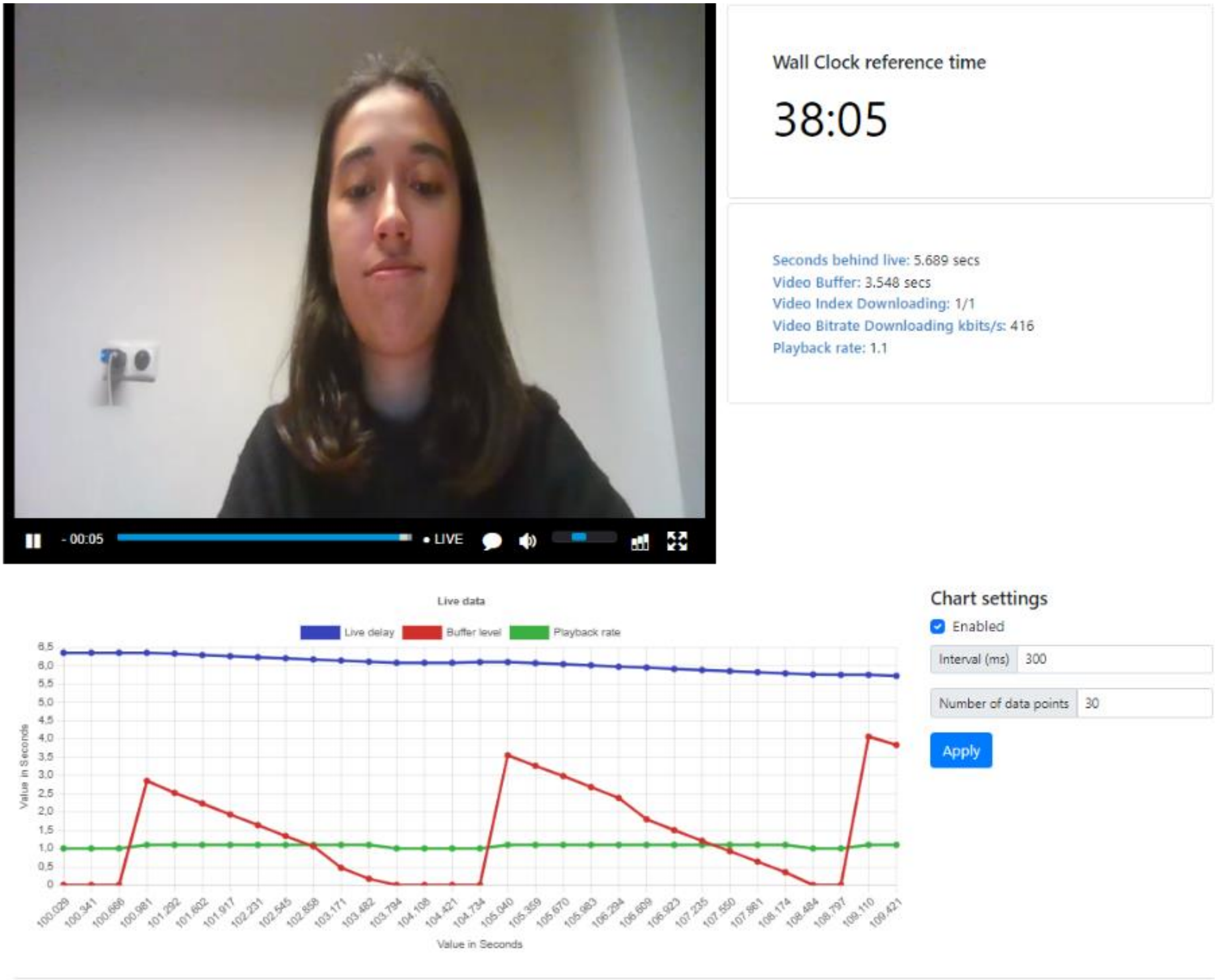
Chart settings

☑ Enabled

Interval (ms) 300

Number of data points 30

Segment Size: 4



Wall Clock reference time

38:05

Seconds behind live: 5.689 secs
Video Buffer: 3.548 secs
Video Index Downloading: 1/1
Video Bitrate Downloading kbits/s: 416
Playback rate: 1.1

Chart settings

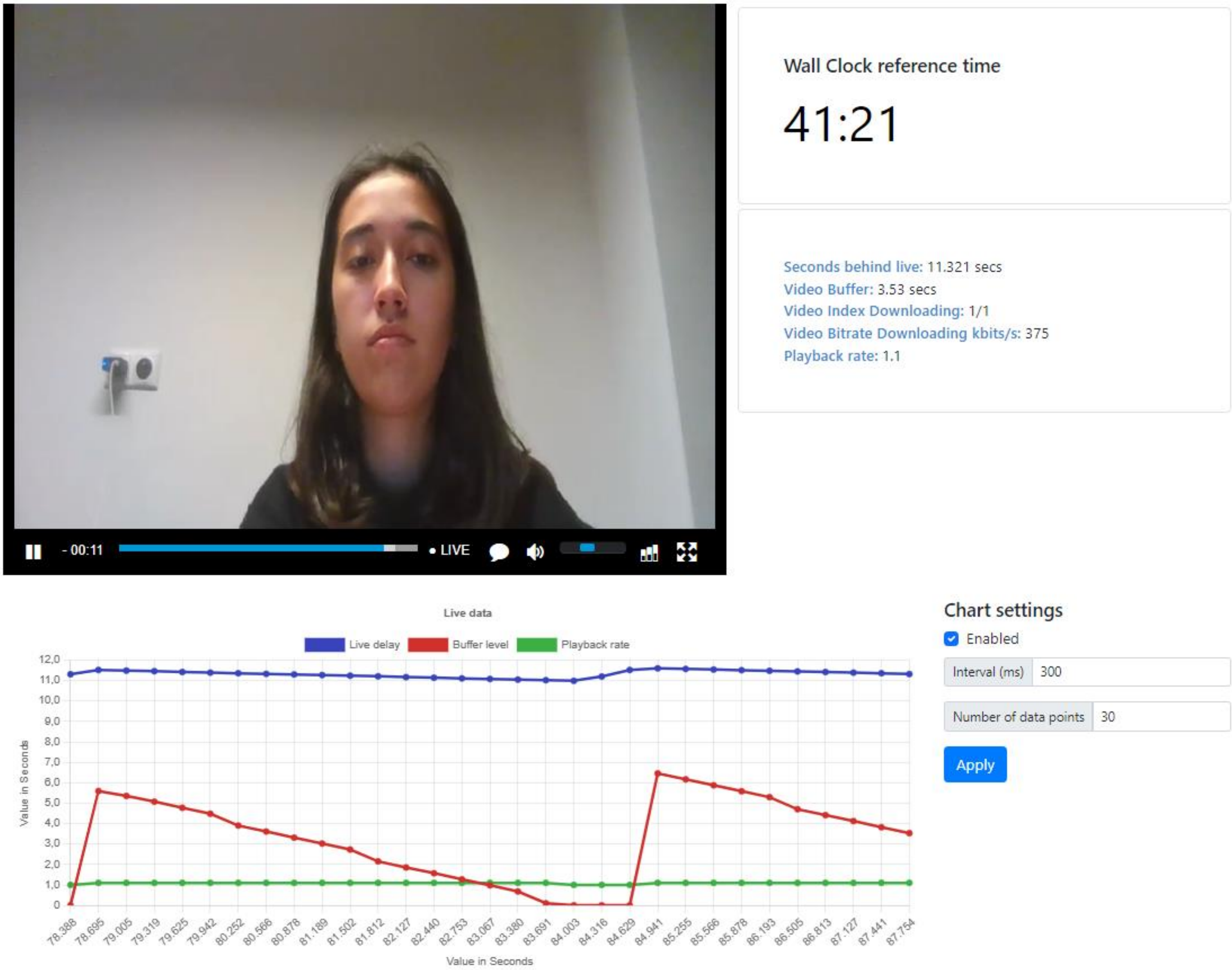
☑ Enabled

Interval (ms) 300

Number of data points 30

Apply

<u>Segment Size: 6</u>



Wall Clock reference time

41:21

Seconds behind live: 11.321 secs
Video Buffer: 3.53 secs
Video Index Downloading: 1/1
Video Bitrate Downloading kbits/s: 375
Playback rate: 1.1

Chart settings

☑ Enabled

Interval (ms) 300

Number of data points 30

Apply

As segment size increases latency increases. SS:2 result in lower latency, SS:6 result in higher latency, cause more delay.