MVP

1. git commit -m “research repo”
2. Developing the program
    a. Git clone the Flipbase project
        i. Clone
        ii. NPM Install
    b. Express generate a Node server
    c. Merge the Flipbase files with the Express Generator server
    d. NPM install and configure ESLINT for code consistency
        i. Get this from the “vuesetup”
            1. package.json
            2. eslint.rc
            3. eslint.ignore
    e. Use the FFMPEG_PATH environment variable
        i. How to access this FFMPEG_PATH environment variable? According to this post, process.env will do. →
            1. FFMPEG_PATH instellen in .bash_profile → `FFMPEG_PATH=/Users/Thomas/Downloads/ffmpeg node test/ffmpeg.js;`
            2. const FFMPEG = process.env.FFMPEG_PATH;
            3. console.log(FFMPEG);
            4. Restart bash to check workings
    f. Set static file path
    g. Write a GET route in app.js
        i. NPM install --save request request-progress
        ii. Import dependencies in route file
            1. express
            2. router
            3. promise
        iii. app.get(“/download-and-encode”, (req, res) => {});
        iv. Check whether the GET request works by running the app locally and res.send() something
        v. git commit -m “server with get route”
    h. Download a video TO the server by making use of NPM request showing the progress to the user using request-progress
        i. https://www.npmjs.com/package/request + request-progress
        ii. https://s3.eu-central-1.amazonaws.com/flipbase-coding-challenge/puppies.mp4
        iii. Use res.write to render the progress
        iv. Make use of a jade file which renders the progress dynamically
            1. Create a download.jade file
            2. Res.render(“download”) → res.send(“state”)
        v. Check for obsolete routes
        vi. Check for obsolete dependency imports and delete them
        vii. Check for obsolete templates

viii. git commit -m “download video to server rendering progress”

i. Transcode the MKV file to MP4 nu format using the fluent-ffmpeg library
    i. NPM install fluent-ffmpeg
    ii. var downloadfile = “video.mp4”
    iii. var newfile = “video.webm”
    iv. res.render → res.write the progress
    v. On end progress, fs.readfile the filename and transcode it
    vi. Use the test file of Github to write out the processing
        1. Edit the paths with “./test/” to own folder “video/video.webm”
        2. on(”progress” → res.send({state: progress.timemark})
    vii. git commit -m “transcoding video on server rendering progress”
j. Render the successfully transcoded video
    i. Use fs.statSync to retrieve the size of the video
    ii. Use fs.createReadStream to create a stream of the video from the server to the client
    iii. git commit -m “rendering video”
k. Download this step by step plan and include it in the repository
l. Write a short README.md

Beyond MVP

m. Provide comments in the code
n. Refactor the entirety
    i. Create a folder called js
    ii. Create a file called download.js
    iii. Import dependencies in download.js file
    iv. In “js”, create a file named “encode.js”
    v. Import dependencies in encode.js
    vi. In “js”, create a file named “render.js”
    vii. Import dependencies in “render.js”
    viii. Import and use download, transcode and render functions in “download-and-transcode” route
o. Make use of promises and .catch(err)
p. Use socket.io or AJAX for updating progress front-end

Questions to check:

1. How does the folder structure look like?
2. Is the code style consistent?
    a. Consistent use of ES6
3. How are errors handled in the code?
4. Are git commits made consistently?
5. Are comments made in the code?