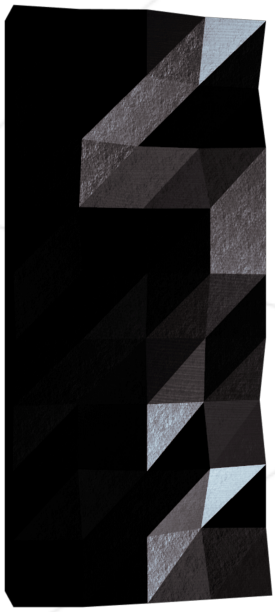

Web

Hypertube

Résumé: A web app for the 21th century

Version: 6.1

Table des matières

Chapitre I

Introduction

This project proposes to create a web application that allows the user to research and watch videos.

The player will be directly integrated to the site, and the videos will be downloaded through the BitTorrent protocol.

The research engine will interrogate least two external sources of `your choice`.
Once the element selected, it will be downloaded from the server and streamed on the web player at the same time. Which means that the player won't only show the video once the download is completed, but will be able to stream directly the video feed.

Chapitre II

General Instructions

- For this project you are free to use any language you choose.
- All the framework, micro-framework, libraries etc. . . are authorized within the limits where they are not used to create a video stream from a torrent, thus limiting the educational purpose of this project. For example, libraries such as `webtorrent`, `pulsar` and `peerflix` are forbidden.
- You are free to use the web server of your choice, may it be `Apache`, `Nginx` or even a `built-in web server`
- Your whole application will have to be at minimum compatible with the latest versions of `Firefox` and `Chrome`.
- Your website must have a decent layout : at least a header, a main section and a footer.
- Your website must be usable on a mobile phone and keep an acceptable layout on small resolutions.
- All your forms must have correct validations and the whole website must be secure. This part is mandatory and will be checked extensively in defense. To give you an idea, here are a few elements that are not considered secure :
  - To have a “plain text” password stored in your database.
  - To be able to inject HTML of “user” Javascript code in unprotected variables.
  - To be able to upload unwanted content.
  - To be able to alter a SQL request.

`For obvious security reasons, any credentials, API keys, env variables etc... must be saved locally in a .env file and ignored by git. Publicly stored credentials will lead you directly to a failure of the project.`

Chapitre III

Mandatory Part

You will need to create a Web App with the following features :

III.1 User Interface

- The app must allow a user to register asking at least an email address, a username, a last name, a first name and a password that is somehow protected.
- The user must be able to register and connect via Omniauth. You must then implement at least 2 strategies : the `42` strategy and another one of your choice.
- The user must then be able to connect with his/her username and password. He/She must be able to receive an email allowing him/her to re-initialize his/her password should the first one be forgotten.
- The user must be able to disconnect with 1 click from any pages on the site.
- The user must be able to select a preferred language that will be English by default.

A user will also be able to :

- Modify the email address, profile picture and information.
- Consult the profile of any other user, ie see the profile picture and information. The email address however will remain private.

III.2 Library part

`This part can only be accessible to connected users.`

This part will have at minimum :

- A research field.
- A thumbnails list.

III.2.1 Recherche

The search engine will interrogate at least two external sources of `your choice`, and return the ensemble of results in thumbnails forms.

You will limit the research to videos only.

III.2.2 Thumbnails

- If a research has been done, the results will show as thumbnails sorted by names.
- If no research was done, you will show the most popular medias from your external sources, sorted as per the criteria of your choice (downloads, peers, seeders, etc...)
- In addition to the name of the video, a thumbnail must be composed, if available, of its production year, its IMDb note and a cover image.
- You will differentiate the videos watched from unwatched, as you prefer.
- The list will be paginated, at the end of each page. The following one must be automatically charged asynchronically. That means there cannot be a link from each page.
- The page will be sortable and filtered according to criteria such as name, genre, the IMDb grade, the gap of production year etc...

III.3 Video Part

`This part can only be accessible to connected users.`

- This section will present the details of a video, ie show the player of the video as well as – if available - the summary, casting (at least producer, director, main cast etc...) the production year, length, IMDb grade, a cover story and anything you think relevant.
- You will also give the users the option of leaving a comment on the video, and show the list of prior comments.
- To launch the video on the server we must - if the file wasn't downloaded prior – launch the download from the associated torrent on the server, and stream the video flux from that one as soon as enough data has been downloaded to ensure a seamless watching of the video. Of course, any treatment must be done in the background in a non-blocking manner.
- Once the movie is entirely downloaded, it must be saved on the server, so that we don't need to re-download the movie again. If a movie is unwatched for a month, it will have to be erased.
- If English subtitles are available for this video, they will need to be downloaded and available for the video player. In addition, if the language of the video does not match the preferred language of the user, and some subtitles are available for this video, they will need to be downloaded and selectable as well.
- If the video is not natively readable for the browser [1], you will convert it on the fly in an acceptable format. The `mkv` support is a minimum.

III.4 API

Develop a RESTful API with an OAuth2 authentification which can be used to get basic infos on everything about this project.

- Authentified users are allowed to get any profiles or update their profiles.
- Any user can get the "frontpage" of the website, meaning all the top movies (only basic infos).
- A GET on a movie must return all the useful information gathered previously
- Authentified users can get users comments via /comments/ :id and movie/ :id/comments. They can post a comment using a proper payload.

1. That means it isn't either`mp4`, nor`webm`

- Any other API call mustn't be usable. Return the proper HTTP code.

Here's a basic documentation : `POST oauth/token`
Expects client + secret, returns an auth token
`GET /users`
returns a list of users with their id and their username
`GET /users/:id`
returns username, email address, profile picture URL
`PATCH /users/:id`
Expected data : username, email, password, profile picture URL
`GET /movies`
returns the list of movies available on the frontpage, with their id and their name
`GET /movies/:id`
return a movie's name, id, imdB mark, production year, length, available subtitles, number of comments
`GET /comments`
returns a list of latest comments which includes comment's author username, date, content, and id.
`GET /comments/:id`
returns comment, author's username, comment id, date posted
`PATCH /comments/:id`
Expected data : comment, username
`DELETE /comments/:id`
`POST /comments OR POST /movies/:movie_id/comments`
Expected data : comment, movie_id. Rest is filled by the server.

Chapitre IV

Bonus part

If the mandatory part is entirely done and is perfect, you can now add the bonuses you wish ; they will be evaluated at the discretion of your correctors. You will however need to respect the basic constraints. For example, to download a torrent will have to happen from the server's side in the background.

If you are needing some inspiration, here are a few ideas :

- Some additional Omniauth strategies.
- Manage various video resolutions.
- Stream the video via the MediaStream API.
- More API routes to add, delete movies, etc...

The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

Chapitre V

Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.

The following instructions will be part of your defense. Be cautious when you apply them as they will be graded with a non-negotiable 0.

V.1 Eliminatory rules

- Your code cannot produce any errors, warnings or notices either from the server or the client side in the web console.
- Anything not specifically authorized is forbidden.
- The slightest security breach will give you 0. You must at least manage what is indicated in the general instructions, ie NOT have plain text passwords stored inyour database, be protected against SQL injections, and have a validation of all the forms and upload