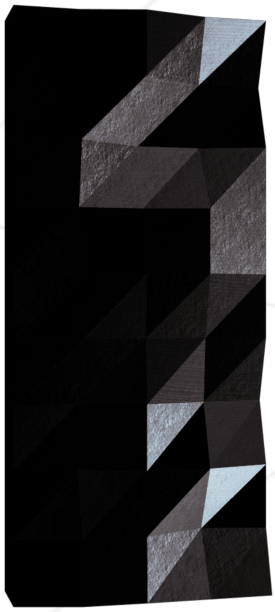

Web

Hypertube

Résumé: A web app for the 21th century

Version: 6.3

Table des matières

Chapitre I

Introduction

This project aims to create a web application that enables user's to search for and watch videos.

The player will integrated directly into the site, and the videos will be downloaded using the BitTorrent protocol.

To enhance search capabilities, the research engine will query at least two external sources of `your choice`.
Once a selection is made, the video will be downloaded from the server and streamed on the web player simultaneously.
This means that the player will not only display the video after the download is complete but also stream the video feed directly.

Chapitre II

General Instructions

- For this project you are free to use any programming language of your choice.
- All frameworks, micro-frameworks, libraries, etc. are allowed, except for those that are used to create a video stream from a torrent. This restriction is to ensure that the educational purpose of the project is not compromised. For example, libraries such as `webtorrent`, `pulsar` and `peerflix` are not permitted.
- You are free to use any web server of your choice, such as `Apache`, `Nginx` or even a `built-in web server`
- Your entire application must be compatible with the latest versions of `Firefox` and `Chrome`.
- Your website must have a decent layout including at least a header, a main section and a footer.
- Your website must be usable on a mobile phone and maintain an acceptable layout on small resolutions.
- All your forms must have correct validations and the entire website must be secure. This part is mandatory and will be extensively checked during the defense. To give you an idea, here are a few elements that are not considered secure :
  - Storing a « plain text » password in your database.
  - Allowing injection of HTML or « user » Javascript code in unprotected variables.
  - Allowing the upload of unwanted content.
  - Allowing alteration of an SQL request.

`For obvious security reasons, any credentials, API keys, environment variables, etc. must be saved locally in a .env file and excluded from git. Storing credentials publicly will result in automatic failure of the project.`

Chapitre III

Mandatory Part

You will need to create a web application with the following features :

III.1 User Interface

- The app must allow a user to register asking for at least their email address, username, last name, first name and a password that is somehow protected.
- The user must be able to register and log in via Omniauth. You must implement at least 2 strategies : the `42` strategy and another one of your choice.
- The user must be able to log in with their username and password. They must be able to receive an email allowing them to reset their password should they forget it.
- The user must be able to log out with one click from any pages on the site.
- The user must be able to select a preferred language that will default to English.

A user must also be able to :

- Modify their email address, profile picture and information.
- View the profile of any other user, including their profile picture and information. However, the email address will remain private.

III.2 Library part

`This section can only be accessed by authenticated users.`

This section must have at a minimum :

- A search field.
- A list of video thumbnails.

III.2.1 Search

The search engine will query at least two external sources (of `your choice`) that exclusively provide video content, and display the results in the form of thumbnails.

III.2.2 Thumbnails

- If search has been done, the results will be displayed as thumbnails, sorted by names.
- If no research was done, the app will display the most popular video from the external sources, sorted by the criteria of your choice (downloads, peers, seeders, etc.)
- Each thumbnail must display the name of the video, its production year (if available), its IMDb (OMDb or TMDb for free API) rating (if available), and a cover image.
- Watched and unwatched videos should be differentiate in the thumbnails.
- The list will be paginated, with the next page being loaded asynchronously as the user scrolls down. There should be no link to load the next page.
- The page will be sortable and filterable according to criteria such as name, genre, IMDb (OMDb or TMDb for free API) grade, production year, etc.

III.3 Video Part

`This section can only be accessed by authenticated users.`

- This section will present the details of a video, including a video player, if available summary, casting (at least producer, director, main cast etc.), the production year, length, IMDb (OMDb or TMDb for free API) grade, a cover image and anything else relevant.
- Users will have the option of leaving a comment on the video, and the list of prior comments will be shown.
- To launch the video on the server we must, if the file was not downloaded prior, the associated torrent on the server will be launched, and the video stream will be initiated as soon as enough data has been downloaded to ensure a seamless watching experience. Any treatment must be done in the background in a non-blocking manner.
- Once the movie is entirely downloaded, it will be saved on the server, to avoid the need to re-downloading in the future. However, if a movie is unwatched for a month, it will be erased.
- If English subtitles are available for the video, they will be downloaded and made available for the video player. Additionally, if the language of the video does not match the preferred language of the user and subtitles are available,the subtitles will be downloaded and selectable.
- If the video is not natively readable for the browser [1], it will be converted on the fly into an acceptable format. At minimum, `mkv` support is required.

III.4 API

Develop a RESTful API with an OAuth2 authentication that can be used to obtain basic information about this project.

- Authenticated users are allowed to retrieve or update any profiles.
- Any user can access the website's « front page », which displays basic inforation about the top movies.
- A GET request on a movie should return all the relevant information that has been previously collected.
- Authenticated users can access user comments via /comments/ :id and movie/ :id/comments. They can also post a comment using an appropriate payload.

1. i.e., not in `mp4`, or `webm` format

- Any other API call should not be usable. Return the appropriate 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 (OMDb or TMDb for free API) 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.

During the evaluation, you will be asked to provide evidence that your API is truly RESTful.

Chapitre IV

Bonus part

If the mandatory part is completed perfectly,you can now add any bonus features you wish. They will be evaluated at the discretion of your evaluators but you must still adhere to the basic constraints. For instance, downloading a torrent must occur on the server 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. By perfect, we mean that all mandatory requirement have been fully implemented and are functioning without any malfunctioning. If any mandatory requirement have not been met, your bonus part will not be evaluated.

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