ASSIGNMENT 4

Instructions:

- Code will be executed using NodeJS
- Grading is based on implementing the required functionality and coding style, specifically: clearly organized code, appropriate variable naming, code readability, coding conventions demo'd in class, etc.
- Javascript syntax rules:
    - Variables must be declared using let/const, not var
    - Functions must be declared using arrow function syntax, not function() syntax
    - When checking equality, use strict equality (triple equals ===)**, not** double equals (==)
    - Do **NOT** use higher order array functions: forEach, map, reduce, filter, closest, find, etc.

Submission Checklist:

For your submission to be graded, you must submit a zip file containing your project code and a screen recording demonstrating your app's functionalities.

1. Create NodeJS project

- ❑ Create a folder called **A3FirstName**. Replace **FirstName** with your name, example: **A3David**
- ❑ Inside the folder, create a new NodeJS project.
- ❑ Within the project, create a Javascript file called **server.js.** Put solution code in the **server.js** file.
- ❑ When you are ready to submit:
    - ❑ Create a zip file containing your project folder.
    - ❑ Rename your zip file **A3FirstName**.zip. Replace FirstName with your name, example: **A3David.**zip. Ensure you use a zip file. Rar and 7zip files are not accepted.

2. Create A Screen Recording

- ❑ Create a screen recording demonstrating your app's functionality. When performing operations on the database, ensure you show the results of the operations in the MongoAtlas web interface.
- ❑ When done, upload the video to your college Microsoft OneDrive account. Set the link sharing so that it is viewable by anyone in the college who has the link.

2. In the submission dropbox

- ❑ Submit your zip file
- ❑ Paste a link to your screen recording

Academic Integrity

- You are responsible for familiarizing yourself with the college's Academic Integrity Policy.
- This is an individual assessment
- Situations which often cause academic integrity issues:
    - Reposting any part of the assessment to online forums or homework help websites
    - Contract plagiarism: Purchasing a solution, or completing a solution for financial compensation
    - Sharing or receiving source code, references, or assistance from others

Problem Description

Create a web application that displays and manipulates a list of Youtube videos.

The application consists of 3 pages:

- Home Page
- Video Details Page
- Admin Page

The application must be created using Node, Express, and Handlebars. The application must include a persistent MongoDB database hosted on MongoAtlas

Requirements:

- Must use Mongoose to programmatically access and manipulate the collections. Use functions described in class (save, deleteOne, find, findOne, etc)
- Must use .lean() when outputting retrieved documents to a Handlebars document
- Make appropriate usage of a Handlebars layouts, templates, partials

User Interface

Every page of the application must contain a navigation menu that enables the user to easily navigate between the **Home** and **Admin** page.

Database Collections

The database must be hosted on MongoAtlas.

The database consists of these collections:

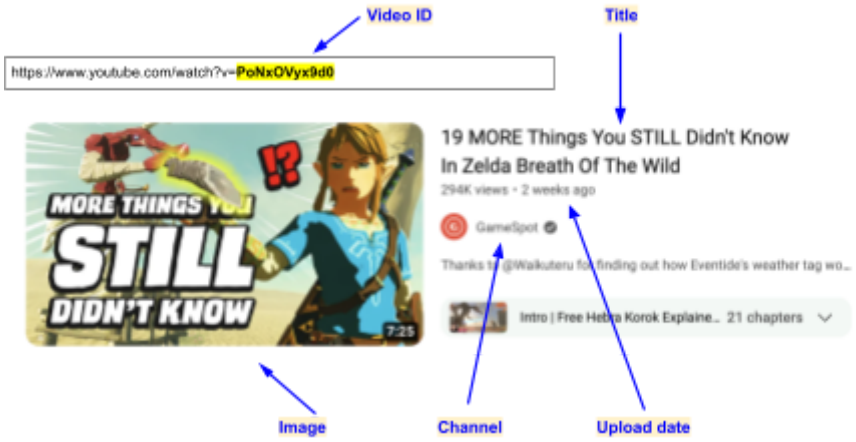
- Videos collection
- Comments collection

Video Collection

Each document in the video collection must contain attributes for:

- Video Id → this is the unique ID that Youtube assigns to each video
- Title → title of the video
- Channel → name of channel where the video is posted
- Likes → set this to 0
- Image → the video's preview image
- Upload Date → string containing when the video was uploaded

Example Video: https://www.youtube.com/watch?v=PoNxOVyx9d0



Comments Collection:

Each document in this collection consists of attributes for:

- Username → name of user that leaves a comment
- Text → text of the comment
- VideoID → Youtube Video Id of the video that the comment is for

NOTE: Do not use Mongoose's populate() function!

Home Page

On the HomePage, show:

- A search bar where user can enter a search keyword
- A list of the videos in the videos collection. For each video, display image, title, channel name, upload date.
- Each video must have a link called “View details”. Clicking on this link should navigate the user to the Video Details Page

Search Functionality:

If the user enters a search keyword into the search bar, the application should:

- Search for a video that contains the specified keyword in the video title.
- If matching videos can be found, display them on the home page
- If videos cannot be found, display an error message.
- If no keyword is entered, then display all videos in the video collection.

Video Details Page

This page displays details about a specified video. You must display:

- A video player that plays the specified video. Use the HTML for an embedded Youtube video.
- Video title, channel, upload date, and number of likes
- A form that contains a button to add a like to the video.A form for the user to add a comment to the current video video
- A list of any comments associated with the current video

HTML for Embedded Youtube Video

The HTML for an embedded Youtube video player can be found by:

- Selecting any video on Youtube.com
- Pressing the SHARE button
- Tapping the EMBED option

The code should look like this:

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/JMWFS26v9fM"
title="YouTube video player" frameborder="0" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"
allowfullscreen></iframe>
```

Your application should programmatically update this HTML with the video ID of the selected video.

Form for Adding Comments

This form enables the user to associate a comment to the currently displayed video. After adding a comment, you should show the same details page with the most updated list of comments.

Form for Adding a Like

When the user presses the LIKE button, the application must:

- Increase the number of likes on the video by 1
- Display the current video details page, but with the most up to date number of likes

Admin Page

The admin page must be available on the /admin endpoint.
This page should display a list of videos in the collection. For each video, show:

- the *name* of the video,
- the number of likes associated with the video,
- the total number of comments associated with the video
- a DELETE button

When the user presses the DELETE button, remove the video and its associated comments from the database. To remove the comments, you should use the deleteMany() function.

After a video is deleted, show the user the admin page, but with the most updated list of videos.

END OF ASSESSMENT