Video Streaming Platform

Project overview

Before we begin coding our project, let’s review how our app will work at a high-level. In the image above, the browser is on the left and the server is on the right. On your site, you’ll have an HTML5 video element with a source that points to the /video endpoint.
First, the video element makes a request to the server, then the header provides the desired range of bytes from the video. For example, at the beginning of the video, the requested range would be from the 0th byte onwards, hence the 0-. The server will respond with a 206 HTTP status, indicating it is returning partial content with the proper header response, which includes the range and content length.

The response headers indicate to the video element that the video is incomplete. As a result, the video element will play what it has downloaded so far. When this happens, the video element will continue making requests, and the cycle will continue until there are no bytes left

Logic and Algorithm

Route: Every chunk of video streamed will be sent as a response to the request-response initiated by the user or the client computing device to our server. Routes are the endpoints to which a request is sent from the frontend.

The response to such requests will be sent as chunks, video, metadata, audio, and captions (if any). Express will be used to create these routes.

Getting the file size: fs in Node carries a method called statSync that will synchronously return the stats of any file. The returned response will be in a JSON key, value format. Among those stats, we will need to know the file size while loading every chunk. You can also use stat to get said file size.

Creating a stream from a file: fs contains another method called createReadStream which will generate a stream when given a file and the start and end chunks.

The creator can define the start and end of the data chunk to be streamed in continuation. The video chunk data is referenced by the return of the fs.createReadStream.

Size of the chunks: When a video is streamed, its content will only contain the starting chunk of the video in the request. To identify the total size of the chunk,

HTTP 206: This is used for partial content. In our video streaming server, we are working with video chunks, i.e., small packets of videos buffered and streamed one after another over a user's computing device.

Over HTTP, these video packets are called partial content because once the initial packet of information loads, the next point is sent in the request-response header. This fetches the subsequent attached packet's information, buffers, and streams it.

The HTTP 206 code signifies that the partial video chunks have been sent as per the request-response header while streaming and communicating over the HTTP protocol. Here are the HTTP headers to keep in mind when working with the HTTP 206 code: -

Content-Range: This defines the format of the stored data as well as how the range of the data chunk is calculated. -

Accept-Ranges: This defines the format of the information stored, routed, transferred, and streamed. -

Content-Length: This defines the key value that will be referenced in the request and response header. -

Content-Type: This defines the container format in our request-response headers (video streaming-specific ones) involving the server and client computing devices. Hence, HTTP 206, at a minimum, contains:

'Content-Range': 'bytes chunkStart**chunkEnd/chunkSize'**

'Accept-Ranges': 'bytes'

'Content-Length': chunkSize

'Content-Type': 'video/mp4'

Dependencies

1)Express
2)Multer
3)Nodemon

Express

Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License.

It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

Multer

Multer is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency.

Nodemon

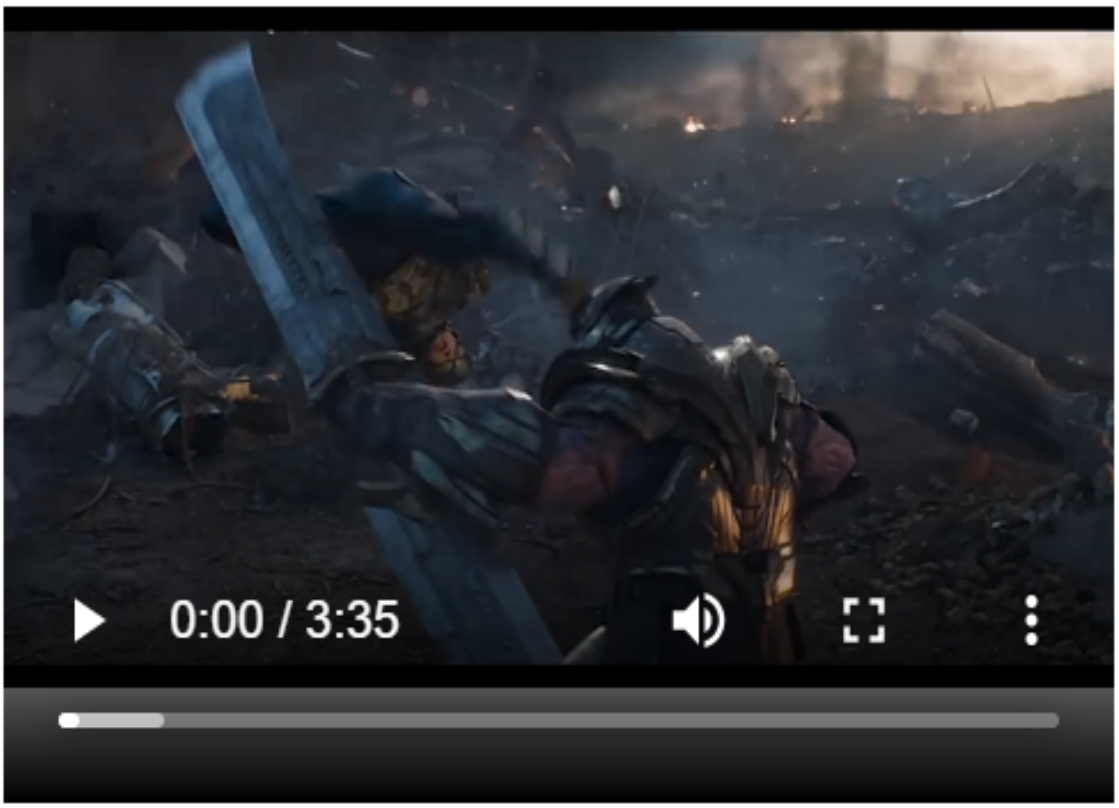
Nodemon is a popular tool that is used for **the development of applications based on node.js**. It simply restarts the node application whenever it observes the changes in the file present in the working directory of your project.

Select file to upload: Choose File No file chosen

Video Details:

Upload File

Video Uploaded successfully!!. Please click here to view the video.



Video Details: NA

Video Name: video.mp4