# 19 - Show Videos in Web App

12:22

## Show Videos in the Web Client

We now have a list of videos in our database that we can fetch, but we are not yet displaying them in our UI. Let's change that.

# 1. Add getVideos Function to Next.js App

Add the following code to `functions.ts`:

```
const getVideosFunction =
httpsCallable(functions, 'getVideos');

export interface Video {
  id?: string,
  uid?: string,
  filename?: string,
  status?: 'processing' | 'processed',
  title?: string,
  description?: string
}

export async function getVideos() {
  const response: any = await
getVideosFunction();
  return response.data as Video[];
}
```

If we want to call `getVideos` from a React server component, we have to make sure the `initializeApp(firebaseConfig)` is run before.

> If you try to do it otherwise you will most likely see the same error I did =)

To fix this we can move this code from `functions.ts` to `firebase.ts`:

```
import { getFunctions } from
"firebase/functions";

export const functions = getFunctions(app);
```

Make sure to export `functions` from `firebase.ts` and import it within `functions.ts`;

```
import { functions } from './firebase';
```

## 2. Render Video Links

Add a **thumbnail image** to your `public` directory.

Fetch the videos within the root `page.tsx` and render them as links:

```
import Image from 'next/image';
import Link from 'next/link';
import { getVideos } from
'./firebase/functions';
import styles from './page.module.css'


export default async function Home() {
  const videos = await getVideos();

  return (
    <main>
      {
        videos.map((video) => (
          <Link href={`/watch?
v=${video.filename}`}>
            <Image src={'/thumbnail.png'}
alt='video' width={120} height={80}
              className={styles.thumbnail}/>
          </Link>
        ))
      }
    </main>
  )
}
```

Add some margin for the thumbnails in `page.module.css`:

```
.thumbnail {
  margin: 10px;
}
```

> Note: If you try to execute a `console.log()` within a server-side component you will not see anything in your browser's console. You will have to open the terminal where you started the Next.js app to see the logs.
> This is of course, because the code is executed `server-side` =)

Upon clicking a thumbnail, you should be redirected to the watch page. We are using the entire filename as the video id so that on the watch page we can render the video.

> A better approach would be to use a shorter videoId, rather than the raw filename, but this would require more work.

# 3. Add Video to Watch Page

Now in the watch page we can grab the filename from the URL and render the video. We will also convert this page to a client component, which is required in order to use `useSearchParams` in Next.js 13.

```
'use client';
```

Copy

Mark Lesson Complete

```
  const videoPrefix =
'https://storage.googleapis.com/nc-yt-
processed-videos/';
  const videoSrc =
useSearchParams().get('v');

  return (
    <div>
      <h1>Watch Page</h1>
      { <video controls src={videoPrefix +
videoSrc}/> }
    </div>
  );
}
```

## 4. Test it Out

You can try uploading a few more videos and see if they show up in the UI. And it should work regardless of whether you are signed in or not.

Full Stack Development