16 - Add Upload Video Feature to Web App

16:43

Add Upload Video Feature

1. Enable CORS for our Cloud Storage Bucket

We need to enable CORS for our Cloud Storage bucket, so that our Next.js app can upload videos to it.

Create a `utils` directory within our git repo. And add a file called `gcs-cors.json`.

```
[
  {
      "origin": ["*"],
      "responseHeader": ["Content-Type"],
      "method": ["PUT"],
      "maxAgeSeconds": 3600
  }
]
```

Then run the following command to enable CORS for our bucket:

```
gcloud storage buckets update gs://<YOUR_RAW_VIDEOS_BUCKET_NAME> --cors-
file=utils/gcs-cors.json
```

You may also need to navigate to:

https://console.developers.google.com/apis/api/iamcredentials.googleapis.com/overvi

And enable the IAM Credentials API.

2. Add Upload Video Function to Next.js App

Create a file named `functions.ts` within the `/firebase` directory.

```
import { getFunctions, httpsCallable } from 'firebase/functions';

const functions = getFunctions();

const generateUploadUrlFunction = httpsCallable(functions, 'generateUploadUrl');

export async function uploadVideo(file: File) {
  const response: any = await generateUploadUrlFunction({
    fileExtension: file.name.split('.').pop()
  });

  // Upload the file to the signed URL
  const uploadResult = await fetch(response?.data?.url, {
    method: 'PUT',
    body: file,
    headers: {
      'Content-Type': file.type,
    },
  });

  return uploadResult;
}
```

3. Add Upload Component to Next.js App

Create a `upload.tsx` file in the `/navbar` directory.

```
'use client';

import { Fragment } from "react";
import { uploadVideo } from "../firebase/functions";

import styles from "./upload.module.css";

export default function Upload() {
  const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    const file = event.target.files?.item(0);
    if (file) {
      handleUpload(file);
    }
  };

  const handleUpload = async (file: File) => {
    try {
      const response = await uploadVideo(file);
      alert(`File uploaded successfully. Server responded with:
${JSON.stringify(response)}`);
    } catch (error) {
      alert(`Failed to upload file: ${error}`);
    }
  };

  return (
    <Fragment>
      <input id="upload" className={styles.uploadInput} type="file"
accept="video/*" onChange={handleFileChange} />
      <label htmlFor="upload" className={styles.uploadButton}>
        <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24"
strokeWidth={1.2} stroke="currentColor" className="w-6 h-6">
          <path strokeLinecap="round" d="M15.75 10.5l4.72-4.72a.75.75 0
011.28.53v11.38a.75.75 0 01-1.28.53l-4.72-4.72M4.5 18.75h9a2.25 2.25 0 002.25-
2.25v-9a2.25 2.25 0 00-2.25-2.25h-9A2.25 2.25 0 002.25 7.5v9a2.25 2.25 0 002.25
2.25z" />
        </svg>
      </label>
    </Fragment>
  );
}
```

> The svg is from Heroicons: **https://heroicons.com/**. I searched for "video" and copied the jsx code.

Create a `upload.module.css` file in the `/navbar` directory.

```
.uploadInput {
  display: none; /* Hide the default input */
}

.uploadButton {
  display: flex; /* Use Flexbox to center the text */
  justify-content: center; /* Center text horizontally */
  align-items: center; /* Center text vertically */
  width: 25px;
  height: 25px;
  border-radius: 50%; /* Make it circular */
  color: black; /* Color of the text */
  border: none;
  cursor: pointer;
  font-size: 10px;
  padding: 0.4em;
}

.uploadButton:hover {
  background-color: rgb(230, 230, 230);
}
```

Add the Upload component to the Navbar component:

```
return (
  <nav className={styles.nav}>
    <Link href="/">
```

```
        <Image width={90} height={20}
          src="/youtube-logo.svg" alt="YouTube Logo"/>
      </Link>
      {
        user && <Upload />
      }
      <SignIn user={user} />
    </nav>
  );
```

4. Assign Invoker Role to All Users

I didn't need to do this in my project, but recently users have been required to run the followinç command to assign the invoker role to all users to prevent a CORS error for the `generateUploadUrl` function.

```
gcloud run services add-iam-policy-binding generateUploadUrl \
  --region="us-central1" \
  --member="allUsers" \
  --role="roles/run.invoker"
```

Note: You may need to update the region and make sure the function name matches your function name.

5. Test Upload Video Feature

Run the Next.js App and upload a video. You should see the video appear in the raw videos bucket. And a short while after it should appear in the processed videos bucket.

> If you have a CORS error in the console, try to delete the `node_modules` folder and run `npm install` in your `functions` directory and redeploy your `generateUploadUrl` function.

> If you continue to have issues, I would recommend using v1 functions. Feel free to ask for help in the Discord server.

References

1. GCS CORS: **https://cloud.google.com/storage/docs/using-cors#command-line**
2. Icons: **https://heroicons.com/**