Firebase Storage

If you go to the `setup-file-upload` branch on the repo you will see the example code below.

What is it?

Firebase is a Backend as a Service (BaaS).

This is a cloud computing service that provides developers with pre-built backend infrastructure, allowing them to focus on building the frontend and business logic of their applications without managing servers or databases.

Firebase Storage is a flexible cloud storage service that safely holds user-uploaded files, works smoothly with other Firebase tools, keeps data secure, performs well, and works across different platforms.

How do you use it?

Follow the docs to integrate it into the project.

- Storage docs

Notes

The app and storage configuration is in the `firebase.ts`, you need this to get access to your project and its respected resources.

```
// firebase.ts

const firebaseConfig = {
  // ...
};

// Initialize Firebase
export const app = initializeApp(firebaseConfig);
// Initialize Firebase Storage
export const storage = getStorage(app);
```

The Firebase sdk provides utility functions to interact with the different services they provide.

```
// App.tsx
import { ref, uploadBytes, getDownloadURL, listAll }
from "firebase/storage";

import { storage } from "./firebase";
```

These functions from the Firebase Storage SDK allow you to:

- `ref()` : Create a reference to a file or folder
- `uploadBytes()` : Upload files/bytes (data) to a specified location or ref in Firebase Storage
- `getDownloadURL()` : Retrieve the download URL for a file
- `listAll()` : List all files and sub-folders within a given folder.

CREATE / UPLOAD

Using the methods above in the `<App/>` you can create/upload and read recently created files.

```
// App.tsx

```

```
const [recentUploadImg, setRecentUploadImg] =
useState("");

const handleUploadSubmit = async (event:
FormEvent<HTMLFormElement>) => {
  event.preventDefault();
  const form = event.currentTarget;
  const location = form.location;
  const input = form["image-upload"] as
HTMLInputElement;

  if (!input.files) {
    alert("No files found :S");
    return;
  }

  // EXTRACT THE FIRST FILE FROM THE FILE INPUT
  const file = input.files[0];
  // CREATE A REFERENCE TO THE FILE IN FIREBASE STORAGE
USING ITS NAME AND THE LOCATION SPECIFIED
  const fileRef = ref(storage,
`${location.value}/${file.name}`);
  // UPLOAD THE FILE TO FIREBASE STORAGE
  const fileUpload = await uploadBytes(fileRef, file);
  // GET THE DOWNLOAD URL OF THE UPLOADED FILE BY USING
ITS REF
  const fileDownloadURL = await
getDownloadURL(fileUpload.ref);
  // SET THE DOWNLOAD URL OF THE RECENTLY UPLOADED
IMAGE IN THE COMPONENT STATE
  setRecentUploadImg(fileDownloadURL);
};
```

READ / Download

Using the methods above in the `<App/>` you can read files / folders.

```
const [folderContents, setFolderContents] =
useState<string[]>([]);

const handleLoadImages = async (event:
FormEvent<HTMLFormElement>) => {
  event.preventDefault();
  const form = event.currentTarget;
  const location = form.location;

  // CREATE A REFERENCE TO THE FOLDER IN FIREBASE
STORAGE
  const folderRef = ref(storage, `${location.value}`);

  try {
    // LIST ALL ITEMS (FILES AND SUBFOLDERS) IN THE
FOLDER
    const listResult = await listAll(folderRef);

    if (listResult.items.length == 0) {
      alert("No folder found");
      return;
    }

    const files = [];

    // ITERATE OVER EACH ITEM IN THE LIST RESULT
    for (const item of listResult.items) {
      // GET THE DOWNLOAD URL FOR EACH FILE
      const downloadURL = await getDownloadURL(item);
      files.push(downloadURL);
    }

    // SET THE ARRAY OF DOWNLOAD URLs AS THE FOLDER
CONTENTS IN THE COMPONENT STATE
    setFolderContents(files);
  } catch (error) {
    console.error("Error listing files:", error);
  }
};
```

App Example

```
import "./styles/main.scss";
import { ref, uploadBytes, getDownloadURL, listAll }
from "firebase/storage";
// YOU WILL NEED TO ADD THIS SEE CODE ABOVE ON HOW TO
SET IT UP
import { storage } from "./firebase";
import { FormEvent, useState } from "react";

import {
  Input,
  Button,
  Box,
  TextField,
  Card,
  CardMedia,
  CardContent,
  Typography,
} from "@mui/material";

const App = () => {
  const [recentUploadImg, setRecentUploadImg] =
useState("");
  const [folderContents, setFolderContents] =
useState<string[]>([]);

  const handleUploadSubmit = async (event:
FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    const form = event.currentTarget;
    const location = form.location;
    const input = form["image-upload"] as
HTMLInputElement;

    if (!input.files) {
      alert("No files found :S");
      return;
    }

    const file = input.files[0];
    const fileRef = ref(storage,
`${location.value}/${file.name}`);
    const fileUpload = await uploadBytes(fileRef,
file);
    const fileDownloadURL = await
getDownloadURL(fileUpload.ref);
    setRecentUploadImg(fileDownloadURL);
  };

  const handleLoadImages = async (event:
FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    const form = event.currentTarget;
    const location = form.location;
    const folderRef = ref(storage,
`${location.value}`);
    try {
      const listResult = await listAll(folderRef);

      if (listResult.items.length == 0) {
        alert("No folder found");
        return;
      }

      const files = [];

      for (const item of listResult.items) {
        const downloadURL = await getDownloadURL(item);
        files.push(downloadURL);
      }

      setFolderContents(files);
    } catch (error) {
      console.error("Error listing files:", error);
    }
  };

  return (
    <Box maxWidth={600} mx="auto" p={5}>
      {recentUploadImg && (
        <Box mb={2}>
          <Card>
            <CardContent>
              <Typography
                gutterBottom
                variant="h5"
                component="div"
                color="primary"
              >
                Recently uploaded
              </Typography>
            </CardContent>
            <CardMedia
              component="img"
              height="140"
              image={recentUploadImg}
              alt="Uploaded Image"
            />
          </Card>
        </Box>
      )}

      <Card
        component="form"
        onSubmit={handleUploadSubmit}
        sx={{
          padding: 2,
          marginBottom: 2,
          "& .MuiTextField-root": {
            marginBottom: 2,
            width: "100%",
          },
          "& .MuiInputBase-input[type='file']": {
display: "none" },
        }}
      >
        <TextField
          required={true}
          id="location"
          label="Storage Location"
          defaultValue="images"
          fullWidth
        />
        <label htmlFor="image-upload">
          <Input
            id="image-upload"
            name="image-upload"
            type="file"
            required={true}
            inputProps={{ accept: "image/png,
image/jpeg" }}
          />
          <Button variant="contained" component="span"
color="primary">
            Upload Image
          </Button>
        </label>
        <Box mt={2}>
          <Button type="submit" variant="contained"
color="primary">
            Upload
          </Button>
        </Box>
      </Card>

      <Card
        component="form"
        onSubmit={handleLoadImages}
        sx={{
          padding: 2,
          "& .MuiTextField-root": {
            marginBottom: 2,
            width: "100%",
          },
          "& .MuiInputBase-input[type='file']": {
display: "none" },
        }}
      >
        <TextField
          required={true}
          id="location"
          label="Storage Location"
          defaultValue="images"
          fullWidth
        />

        <Box mt={2}>
          <Button type="submit" variant="contained"
color="primary">
            Load images
          </Button>
          {folderContents.length > 0 &&
            folderContents.map((url) => (
              <Box m={2} key={url}>
                <Card>
                  <CardMedia
                    component="img"
                    height="140"
                    image={url}
                    alt="Uploaded Image"
                  />
                </Card>
              </Box>
            ))}
        </Box>
      </Card>
    </Box>
  );
};

export default App;
```
