# React notes

* **FIREBASE:-**
* install uuid library of firebase to get the characters with unique id: npm intall firebase uuid
* import {ref,uploadBytes,listAll} from "firebase/storage":
  + - ref for getting reference
    - uploadBytes to upload any files
    - listAll to list all the files in specific path
* import {v4} from "uuid" : to create unique string

1. snapshot:-
   1. The snapshot object in the code you provided is not user-defined; it is a standard part of the Firebase Storage API. Firebase SDKs, including the one for JavaScript, return this snapshot object as everything includes in firebase.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Storage\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **lisAll**:-
   1. In Firebase, listAll is a method provided by the Firebase Storage API. It allows you to list all the items (files and directories) within a specified storage location.
   2. Here's a breakdown of the method and its usage:
   3. Method: listAll(reference)
   4. reference: This is a reference to a specific location in your Firebase Storage. It could be a root directory or any other directory within your storage bucket.
   5. Returns: A promise that resolves to an object with information about the items in the specified location.
   6. items: An array of Reference objects representing the items (files or directories) in the specified location.
   7. prefixes: An array of Reference objects representing subdirectories within the specified location.

Example Usage:

import { listAll, ref } from 'firebase/storage';

import { storage } from '../firebase'; // Importing your Firebase Storage instance

const imageListRef = ref(storage, 'images/');

listAll(imageListRef)

.then((response) => {

// response.items is an array of Reference objects representing files

// response.prefixes is an array of Reference objects representing subdirectories

console.log('Items:', response.items);

console.log('Subdirectories:', response.prefixes);

})

.catch((error) => {

console.error('Error listing items:', error);

});

NOTE:All the data of listAll is stored in “response” variable of .then((response))

1. To download and store urls:-

// const [imageUpload, setImageUpload] = useState(null)

// const uploadImage = () => {

// if (imageUpload == null) return;

// //uploading images and then downloading it

// const imageRef = ref(storage, `images/ + ${imageUpload.name + v4()}`)

// uploadBytes(imageRef, imageUpload )

// .then((snapshot) => {

// getDownloadURL(snapshot.ref)

// .then((url) => {

// setimageList((prev) => [...prev, url])

// })

// })

// }

To display all the images of firebase storage

* 1. Must use “useeffect”

useEffect(() => {

listAll(imageListRef)

.then((snapshot) => {

const promises = snapshot.items.map((item) => getDownloadURL(item));

return Promise.all(promises);//waits until all the url fetches,once fetching is complited ,it returns

})

.then((urls) => {

setimageList(urls);

})

.catch((error) => {

console.error('Error fetching image list:', error);

});

}, [])

* **OPERATORS:-**
  1. Spread Syntax ( …[arrayname] ):-
     + It spread the array/object element and store in the new one
     + The spread syntax is used for several purposes, including creating shallow copies of arrays and objects, merging arrays, and spreading elements within arrays or function arguments.
     + Eg:-
       - Copying Arrays:
         * const originalArray = [1, 2, 3];
         * const copiedArray = [...originalArray]; // Creates a copy of the original array
       - Merging Arrays:
         * const array1 = [1, 2];
         * const array2 = [3, 4];
         * const mergedArray = [...array1, ...array2]; // Merges two arrays
       - Adding Elements to an Array:
         * const originalArray = [1, 2];
         * const newArray = [...originalArray, 3]; // Adds a new element to the array
       - Copying Objects:
         * const originalObject = { a: 1, b: 2 };
         * const copiedObject = { ...originalObject }; // Creates a copy of the original object
  2. Promise:-
     + Its make to wait until all the data is fetched ,once the data is fetched is returns all the fetched data
     + in this case, fetching download URLs from Firebase Storage. These operations typically return promises. When you need to perform several of these operations in parallel and wait for all of them to complete before continuing with your code, you can use Promise.all().
     + Here's how it works:
     + You have an array of promises, which are essentially tasks that will be executed asynchronously.
     + Promise.all(promises) takes an array of promises as its argument.
     + It returns a new promise that resolves when all the promises in the input array have resolved successfully or rejects if any of the promises in the array reject.
     + If all promises resolve successfully, the resolved values are collected into an array in the same order as the input promises.
  3. Map:-
     + map Function: The map function is used to transform and iterate over each element of an array, applying a provided function to each element and returning a new array with the results. It is mainly used for creating a new array based on the original array's elements.
       - Eg:
         * const numbers = [1, 2, 3];
         * const doubledNumbers = numbers.map((number) => number \* 2);
         * // doubledNumbers is now [2, 4, 6]

•Eg: Object.values(postdata).map((data, i) => (

<div key={i}/> /\*create div for every data\*/