

EXPERIMENT NO. 06

Shubham Golwal | 2020300015 | COMPS-D | ADBMS

Aim:

To develop a multimedia database

Scenario:

Create a database to perform CRUD operations and store any type of multimedia file and retrieve it back on demand. The database will act as organized storage of your data.

Procedure:

A) Using PostgreSQL

1. First create a python file which will be used as an application to interact with the database.
2. Make connections to the database.
3. Create a table with all required data related to the file such as its name, extension and one column of type bytea to store binary data.
4. If the user wants to store some data, ask for the path of the file, find out the file's name and extension, read the file, and insert it into the database using the insert command.
5. If the user wants to retrieve the data back, ask the user for filename and the extension of the file. Now retrieve the bytea data corresponding to provided data and write it to a binary file ending with provided extension.

B) Firebase Storage and Cloud Firestore

Step 1 : Create a Project on Firebase and install firebase cli on your system

Step 2 : Create a react app and then cd into the directory and run the firebase init to initialize the firebase in your project

Step 3 : Select firestore and storage by using keyboard arrow buttons and space bar to select

Step 4 : Connect to existing firebase app created on the console

Step 5 : Complete the rules setup

Step 6 : Go to firebase console and create a database in firestore with test settings and create a bucket in storage as well

Now once the firebase is configured open the storage.rules file and change the access as follows

```
rules_version = '2'; service
firebase.storage { match
/b/{bucket}/o { match
/{allPaths=**} {
    allow read, write;
  }
}
```

```
import React, { useState, useEffect } from 'react';
```

```
import {app} from './base';
```

Change the firestore.rules file as follows

```
rules_version      =      '2';      service
cloud.firestore    {      match
/databases/{database}/documents {
    match /{document=**} {
        allow read, write;
    }
}
}
```

Now simply add your firebase configuration returned on the terminal into base.js file and import it in App.js

Now create a react component that takes image as input and username as input and it will store the image in firebase storage and the downloadable file for the image will be stored in the firestore database document along with the username provided by the user after submitting the data

INSTALLATION:

✕ Create a project (Step 1 of 3)

Let's start with a name for your project[?]

Project name

ADBMS LAB 6

 adbms-lab-6

 Select parent resource


☒ I accept the [Firebase terms](#)


✕ Create a project (Step 2 of 3)


Google Analytics for your Firebase project


Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:

 A/B testing[?]

 Crash-free users[?]

 User segmentation & targeting across[?]

 Event-based Cloud Functions triggers[?]

✕ Create a project (Step 3 of 3)


Configure Google Analytics

Analytics location ⓘ

United States ▼

Google Analytics is a business tool. Use it exclusively for purposes related to your trade, business, craft, or profession.

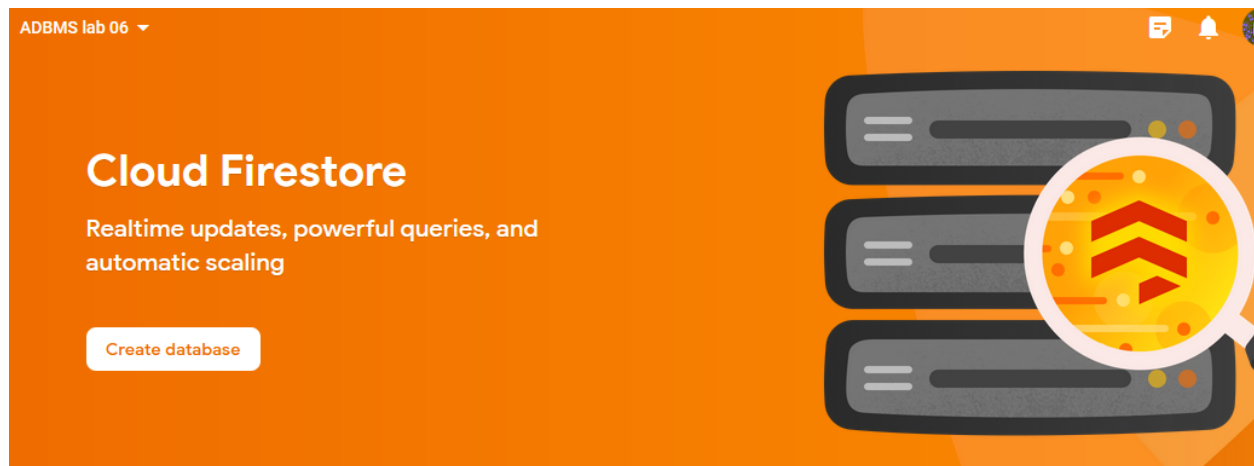
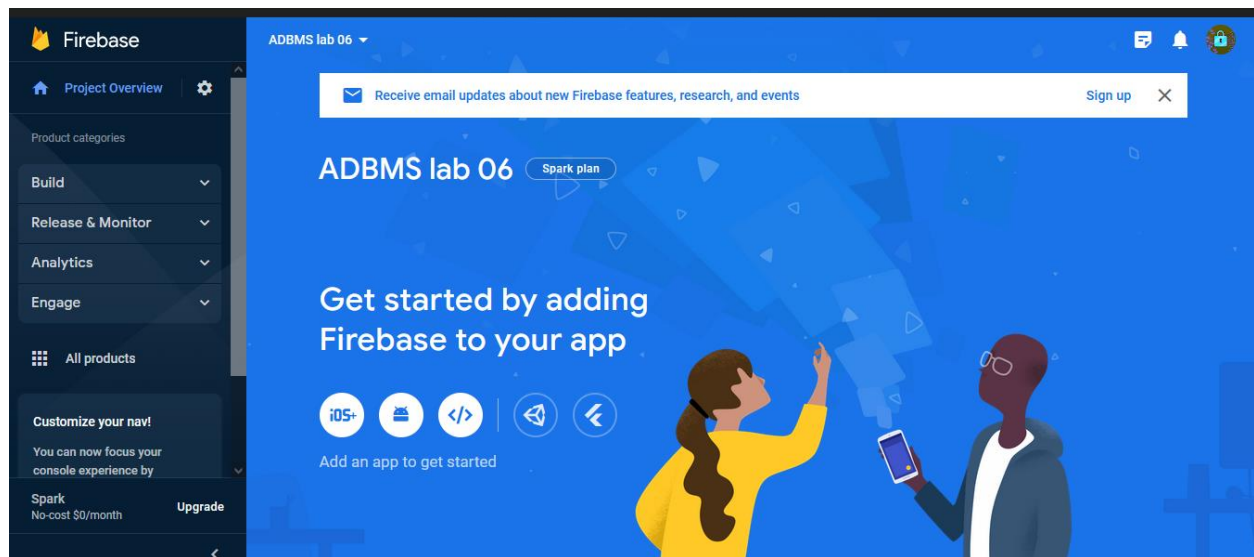
Data sharing settings and Google Analytics terms

☒ Use the default settings for sharing Google Analytics data. [Learn more](#) 

Creating your project... Please wait...

ADBMS lab 06





```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2023, 4, 27);
    }
  }
}
```

adbms-project

Cloud Firestore

Data Rules Indexes Usage Extensions **NEW**

Edit rules Monitor rules **Develop and Test**

Right now unpublished changes

Today • 7:46 pm

```
1 rules_version = '2';
2 service cloud.firestore {
3   match /databases/{database}/documents {
4     match /{document=**} {
5       allow read, write: if true;
6     }
7   }
8 }
```

For testing purpose giving rights to everyone

Admin SDK configuration snippet

☒ Node.js ☐ Java ☐ Python ☐ Go

```
var admin = require("firebase-admin");

var serviceAccount = require("path/to/serviceAccountKey.json");

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount)
});
```

Generate new private key

For interaction with node

```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL
powershell - AnimeFirebase + v [] [X] ...

? Allow Firebase to collect CLI and Emulator Suite usage and error reporting information? Yes
i To change your data collection preference at any time, run `firebase logout` and log in again.

Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=563584335869-fgrhgm47bqnekij5i8b5pr03ho849e6.apps.googleusercontent.com&scope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response_type=code&state=857962400&redirect_uri=http%3A%2F%2Flocalhost%3A9005

Waiting for authentication...

+ Success! Logged in as shubham.golwal@spit.ac.in
o PS E:\SEM_06\ADBMS\Expt6\AnimeFirebase>
```

Server.js

```
const express = require("express");
const app = express();

const admin = require("firebase-admin");
const credentials = require("./key.json");

admin.initializeApp({
  credential: admin.credential.cert(credentials),
});

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

const db = admin.firestore();

//POST request
app.post("/create", async(req, res) => {
  try {
    console.log(req.body);
    const id = req.body.email;
    const userJson = {
      email: req.body.email,
      firstName: req.body.firstName,
      lastName: req.body.lastName,
    };
    // const response = db.collection("users").doc(id).set(userJson);
    const response = db.collection("users").add(userJson);
    res.send(response);
  } catch (error) {
    res.send(error);
  }
});
```



```

//READ all request
app.get("/read/all", async(req, res) => {
  try {
    const usersRef = db.collection("users");
    const response = await usersRef.get();
    let responseArr = [];
    response.forEach((doc) => {
      responseArr.push(doc.data()); // data->each doc
    });
    res.send(responseArr);
  } catch (error) {
    res.send(error);
  }
});

//READ one request

app.get("/read/:id", async(req, res) => {
  try {
    const usersRef = db.collection("users").doc(req.params.id);
    const response = await usersRef.get();

    res.send(response.data());
  } catch (error) {
    res.send(error);
  }
});

//http://localhost:8080/create

//UPDATE request
app.post("/update", async(req, res) => {
  try {
    const id = req.body.id;
    const newFirstName = "updated first name";
    const userRef = db.collection("users").doc(id).update({
      firstName: newFirstName,
    });
    // const response = await usersRef.get();
    res.send(response);
  } catch (error) {
    res.send(error);
  }
});

```

```
//DELETE request
app.delete("/delete/:id", async(req, res) => {
  try {
    const id = req.params.id;
    const response = db.collection("users").doc(id).delete();
    res.send(response);
  } catch (error) {
    res.send(error);
  }
});

const PORT = process.env.PORT || 8080;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```

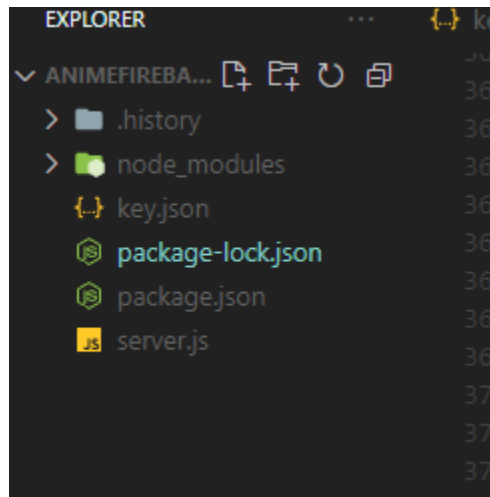
Key.json

```
{
  "type": "service_account",
  "project_id": "adbms-project-47fd8",
  "private_key_id": "a89762fa1160e1112cf34806d17a9d14f17d4dbd",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nMIIIEuwIBADANBgqhkiG9w0BAQEFAASCBAUwggShAgEAAoIBAQCp0BNY6DR0X2Tn\nsSpyM/a91DXhh
ykLud8R6zvq3nr0PczH/c/5VZEZdWvZRw+eVOVn4xBCNN1J9wrm\nnn3Sznis1o3DU/ITDq78kJQZFFJN2
5YoDIYgyfpzi1Lziv0Q0M1GmStAs7CkfWc+d\nke7GzR7HQirT9Su0Wx11E494YtQpMe579Lx9Iww4erg
d2uWsVXwhVgcyUVRlZGJ5\nkaxgvyLeoLQHogtZQ1GyfowLtrxyrykz2xsn+iKI2IMJ5SpXFMXV9XmJsQ
PB5w60\nVuKka/zCoCgnIcQ7gt3fkRE8KNDnVdKiI5wmw3A00khGYGFt94dCAQ/Zi07ZISoG\nn+VzRODQ
lAgMBAAECggEAQLgaX5SkRUQEL5+gp0TxwbiKzokWVi9BetFWFTY0xLzd\nn+szbw1Hfyr2rXHSOWdBOXH
E4RZtjSwjboTjdm4Tih/9iJfKV4QNohmyL/rEQYMyr\nn9HSHtYL25SaiVKMD6gcZEd7Irw5Uv+uZSpsPT
3pmxBe0PhkfQBKtWacaU44LASW0\nn9zNlHqOmMtSPMsGJyJV+rqAYPsWVI8onpZIENZ9X7k4sj01KXnAn
1JS9ZNe7pTqu\n/RSNEb2zWNjWCqqc2NS5XNXDsrgowB807/RNhplDE74iXwiOT9rSfEGdINADNk35\nnn
ScP4fp/cY450ZZoUnnkbaIo3vtznw00xy0uZeIDowKBgQD0SNgd16PjmI7rjeZ\nnkYyauBTGwgvuPFBN
mImE7vLGRYX0sk1Uz51HnhnIOrc6uCTRRBtmKHiAwpJOV+mI\nnf5UtUALjnfSh4NXCYNKvBGHuG3xha6J
R/cKtmY6i4iwnw0U5SvnUKbdqPBSkEoox\nn4KOAlZ0X1LcSCJBKzxcBAoB4wKBgQDGT09qy5NUBsbYeI
wSd4R0K2iCWAY/rjGH\nng6jlGZE6b5WZ3LhMpUKPedw2+db2e/+M7rvYv080xLWYdg8sFXrZMem0Pk0aK
W3W\nnebPbCsgxpMkPKsY/3gBIopRYZ087u3bnBz46YTAwQc/FAx1aJ3bQGzRHtYq1phw6\nnLm1BN10wVw
KBgGw/nZlfoBg3PYhssWwy0jSX2yJKio0b4qc+QAGVH+oJoFX9MTqg\nnu/Zi0ib13CiHDx8K02FKPgkui
qIGxPkyzo2uaY7Ni2N3qvWta7siNd0mU1ej12ZX\nnII8q0QVzGwyTXpY40H4UD05011hiEz0nyHSQ7B4d
v/w2oFFgTTuTVQGzAoGBAJ0f\nnaLTecSaTGAt9uL8JJDDQsA0sGhC0BAQ23LKQa1g/PSmJXmVBhwZ0dove
Mjt908PXi\nnU8UwLv1Iw/rB0NPFUUm3Ws40b11Dalx+ohv1NxreztdNu7fvV5o1+AT0FrV1lhkI\nnjC4F
xAD4pIAeL6u3iVTruUj4d5s50zR+gbfCM6SRAn9ggUy13r+1NA633qE4ESxj\nn1/2sE5PtVJe0pygzCld
rod2gOVDhmVfRuCPeRgpfH7fWn0ZiHUM0tk7uOyFn/bFk\nnrsXj9m9YD8sFi40rpZdGWGRKcFNCfY856W
ALZeOuJml+ofT/iI1a5UjcxEG7Kgn2\nn+59ZRhx/vlsmQ89giTDs\nn-----END PRIVATE KEY-----
\n",
```

```

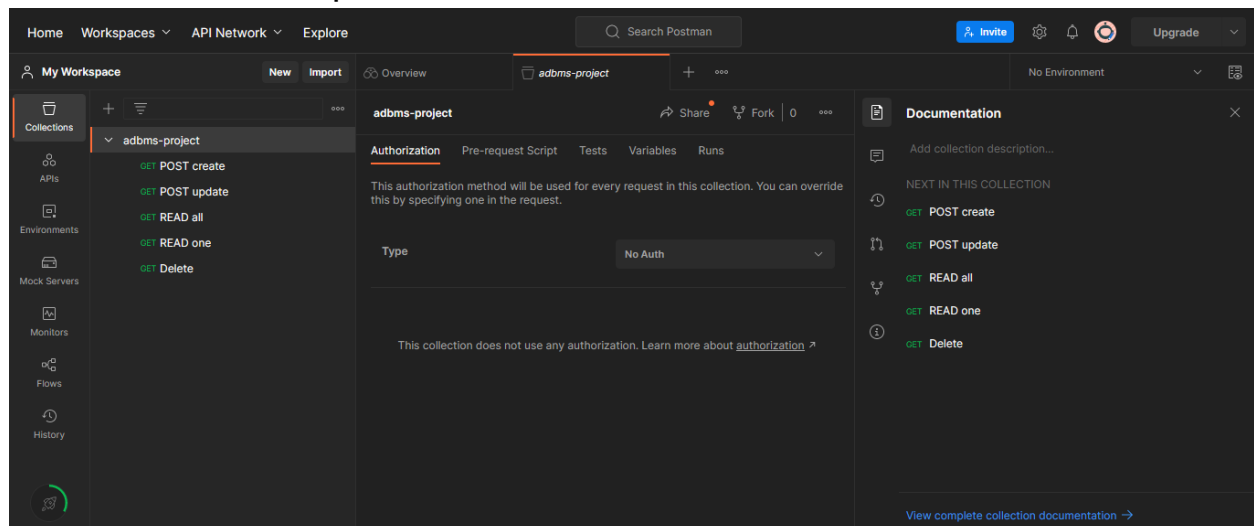
"client_email": "firebase-adminsdk-lk283@adbms-project-47fd8.iam.gserviceaccount.com",
"client_id": "107264716051288400733",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",
"token_uri": "https://oauth2.googleapis.com/token",
"auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
"client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-lk283%40adbms-project-47fd8.iam.gserviceaccount.com"
}

```



OUTPUT :

Performed CRUD operations:



Server is started:

```

14
15   const db = admin.firestore();
16
17   //POST request
18   app.post("/create", async(req, res) => {
19     try {
20       console.log(req.body);

```

OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL

```
> nodemon server.js
```

```

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server is running on port 8080.

```

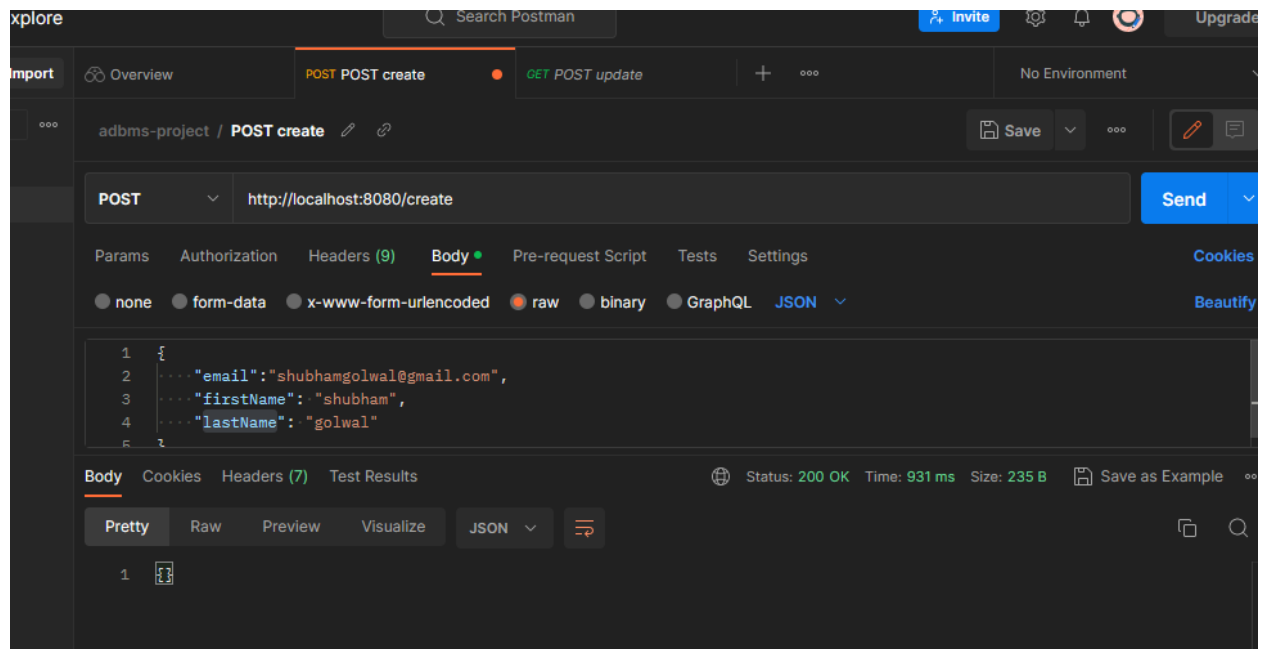
POST:

```

//POST request
app.post("/create", async(req, res) => {
  try {
    console.log(req.body);
    const id = req.body.email;
    const userJson = {
      email: req.body.email,
      firstName: req.body.firstName,
      lastName: req.body.lastName,
    };
    // const response = db.collection("users").doc(id).set(userJson);
    const response = db.collection("users").add(userJson);
    res.send(response);
  } catch (error) {
    res.send(error);
  }
});

```

Request sent from postman



Server received the request

```
OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server is running on port 8080.
{
  email: 'shubhamgolwal@gmail.com',
  firstName: 'shubham',
  lastName: 'golwal'
}
```

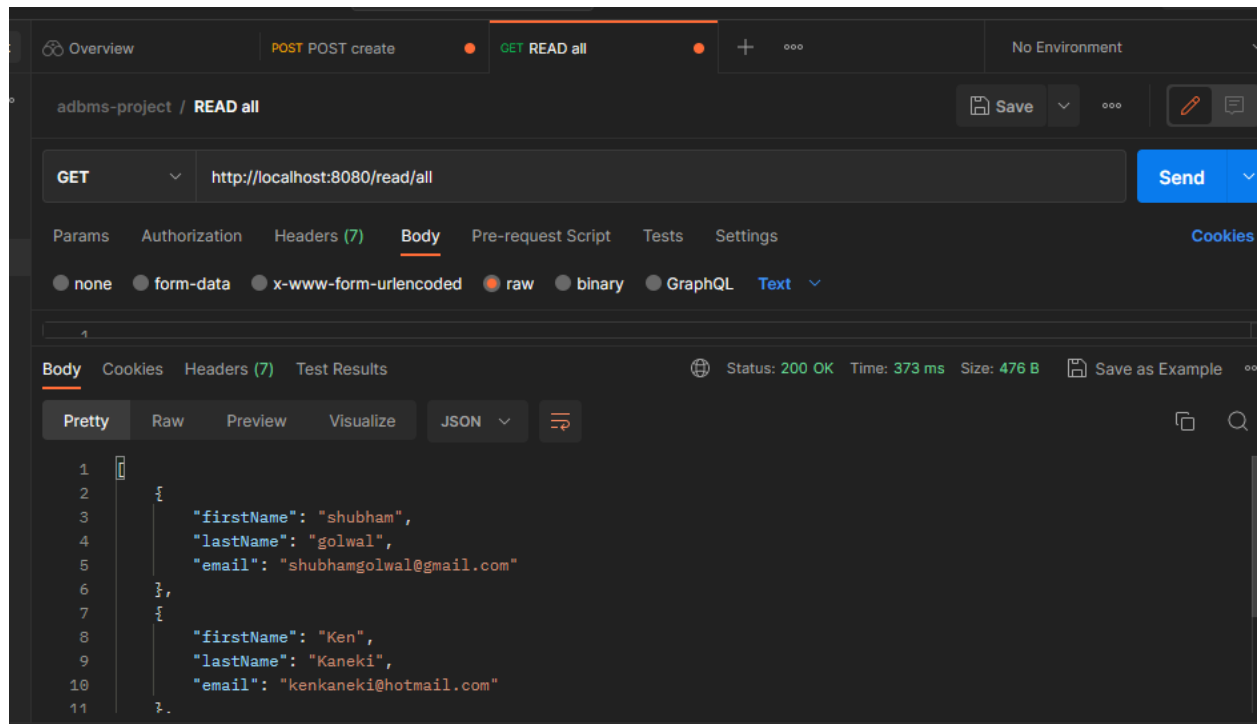
Added on firebase

Home > users > 5nVJ29PTlkr3... More in Google Cloud		
adbms-project-47fd8	users	5nVJ29PTlkr30DocCSA
+ Start collection	+ Add document	+ Start collection
users >	5nVJ29PTlkr30DocCSA >	+ Add field
	TAF9qWhV4MAidRqmWBGg shubhamgolwal@gmail.com	email: "shubhamgolwal@gmail.com" firstName: "shubham" lastName: "golwal"

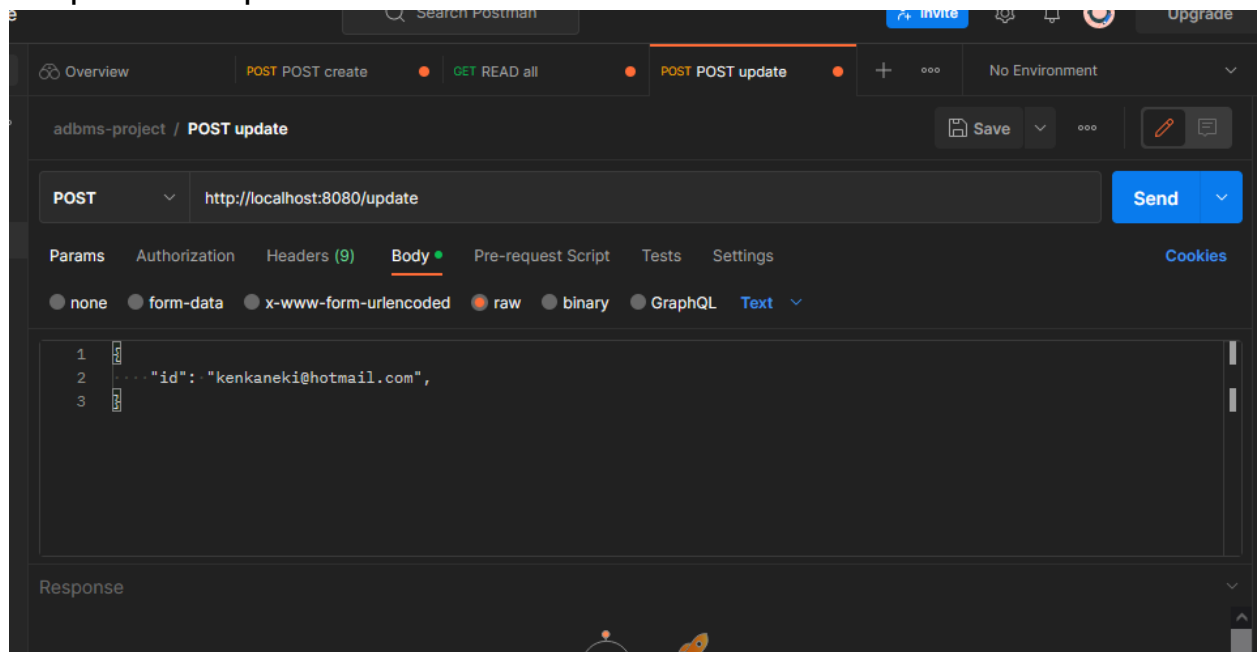
Read:

```
//READ all request
app.get("/read/all", async(req, res) => {
  try {
    const usersRef = db.collection("users");
    const response = await usersRef.get();
    let responseArr = [];
    response.forEach((doc) => {
      responseArr.push(doc.data()); // data->each doc
    });
    res.send(responseArr);
  } catch (error) {
    res.send(error);
  }
});
```

Request from postman



UPDATE:
Request from postman



Updated on database

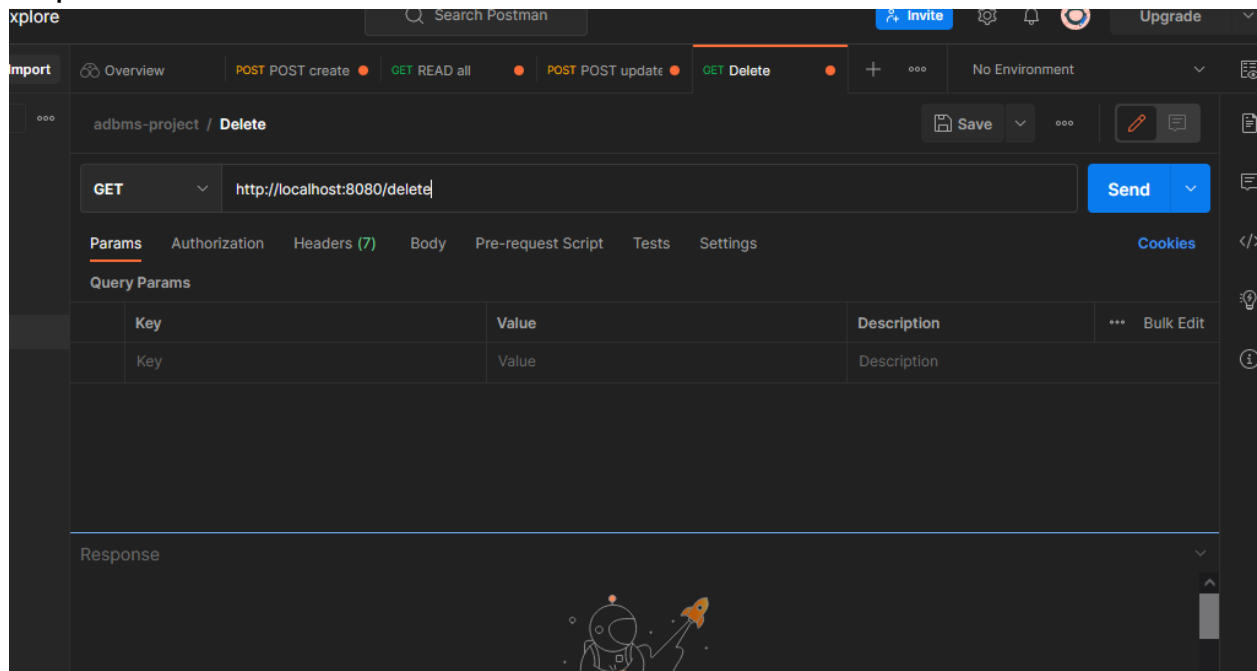
+ Add field

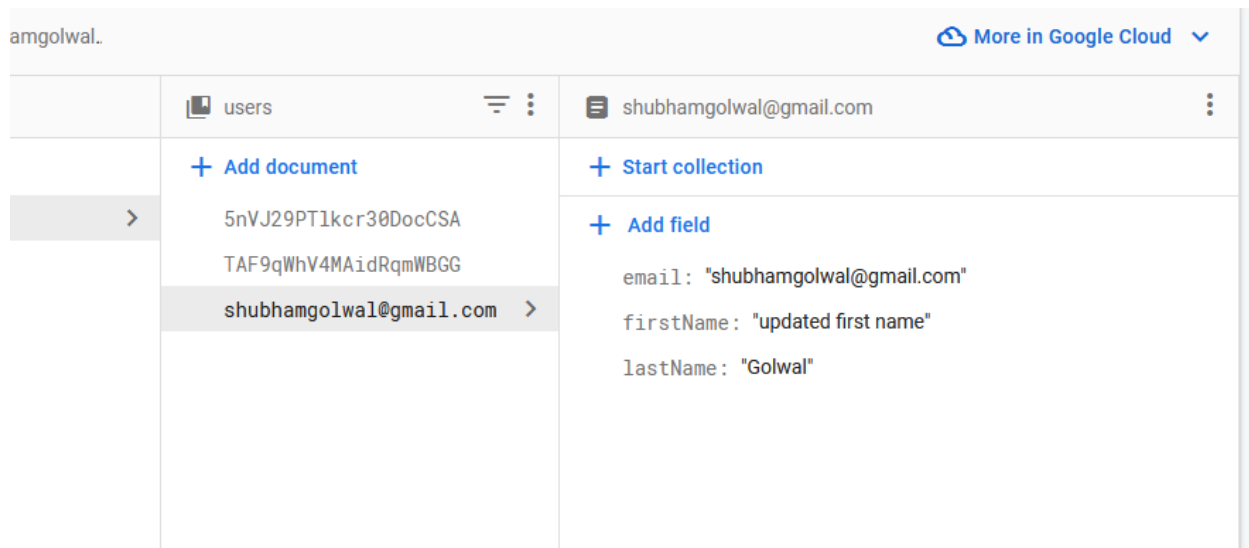
email: "shubhamgolwal@gmail.com"

firstName: "updated first name"

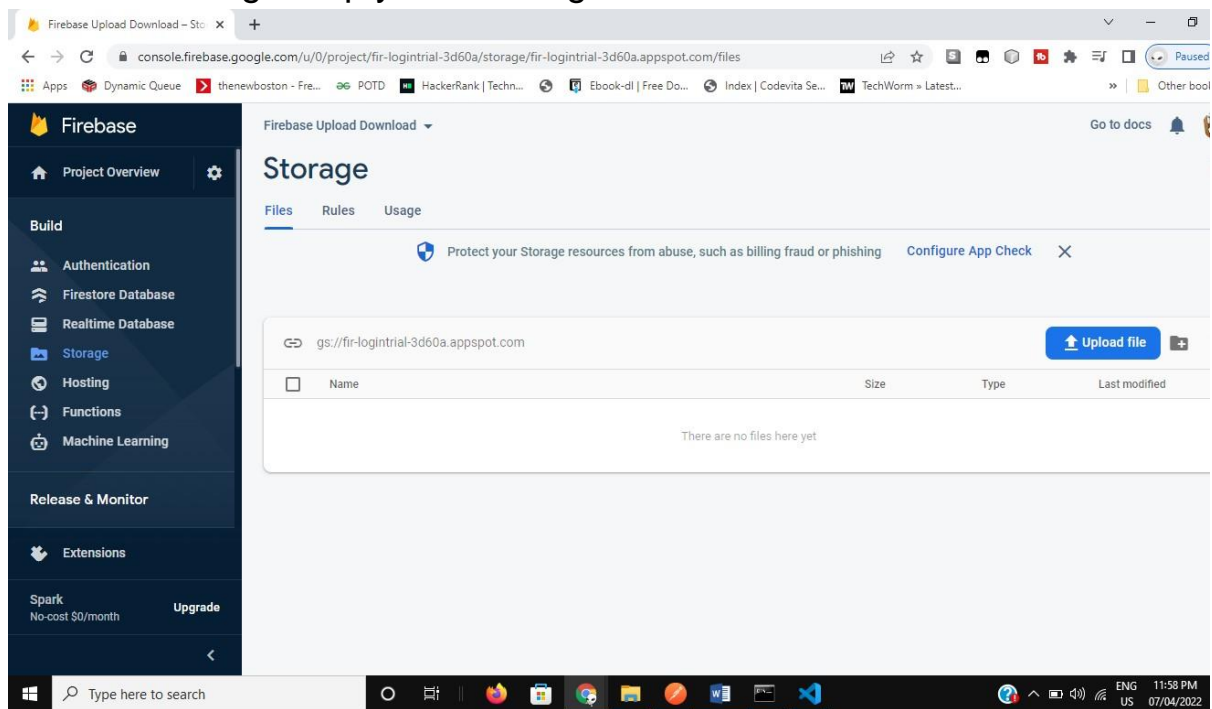
lastName: "Golwal"

DELETE:
request





Firestore collection empty i.e. nothing in the bucket



Firestore collection empty

Choose an image file : as soon as file is chosen, it will be stored in the storage bucket

File Chosen emp.png

Choose File

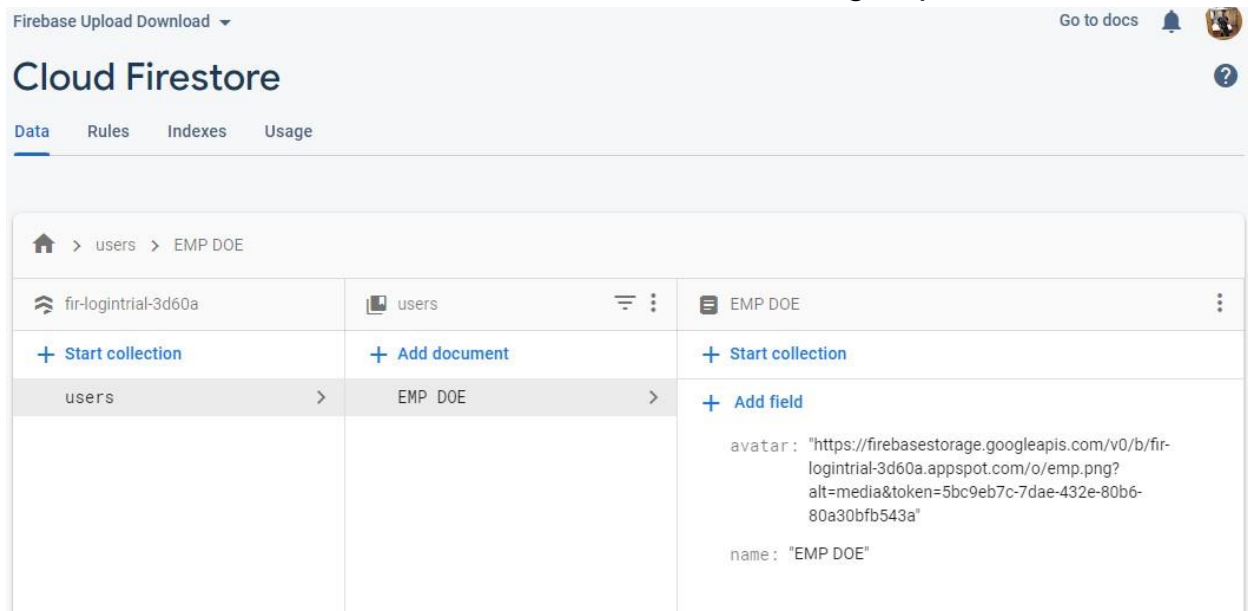
emp.png

NAME

Submit

New file named emp.png added to the storage bucket

Collection users created with EMP DOE document and the document contains name and the downloadable url of the image uploaded



Now the uploaded image and user will be listed in the homepage of the app once successfully uploaded

Submitting multiple images to see multiple images in the app

Project Overview

Firestore Database

Storage

Build

Release and monitor

Analytics

Engage

All products

Spark

adbms-project

Cloud Firestore

Data

Rules

Indexes

Usage

Extensions

Panel view

Query builder

users

TAF9qWhV4MAI

More in Google Cloud

adbms-project-47fd8	users	TAF9qWhV4MAidRqmWBGG
+ Start collection	+ Add document	+ Start collection
users	TAF9qWhV4MAidRqmWBGG	+ Add field
	shubhamgolwal@gmail.com	email: "kenkaneki@hotmail.com" firstName: "Ken" lastName: "Kaneki"

App

React App

localhost:3000

Choose File

No file chosen

NAME

Submit

Name : CJ

Name : EMP DOE

Name : Jeet Mistry

Conclusion: From this experiment, we

- Learnt that Databases can store binary files like media files.
- Used Two methods were used: PostgreSQL and Firebase.
- PostgreSQL stores entire binary data in the database, increasing its size but not requiring external storage.
- Firebase uses two components: Firebase Storage to upload images and generate downloadable URLs, and Firebase Firestore to store data in document and key-value form and retrieve images in real-time.