Employee Management System

Yu Zhao, Xiaojing Wang, Sujeeth Nilakantan

GitHub: https://github.com/zhaoyufrank/CloudComputing-Project

Project Introduction

This is an employee management project. The home page displays all employees, the next web page shows the detailed information of an employee. The user can create a new employee, update an employee's information and delete the employee from the system. The employee information includes name, gender, email, birthday, work department, start date, position, and so on.

We used React.js to build the front end, and Node.js and Cloud Function to build the backend. The project is built using Google Cloud components, such as Compute Engine, VM Instance, Firebase, Firestore and Cloud Functions.

Service/Application Overview

- Describes how the user(s) will interact with the service

    1. If the user (admin) clicks the create button on the home page; the user is redirected to the ‘create employee’ page. The user inputs the employee related information and clicks on the save button; the input details are saved and stored in the Firestore database.
    2. If the user clicks on the edit button on the home page; the user is re-directed to the ‘edit employee’ page. The user can modify the existing employee details and click on the save button; the input details are saved and updated in the Firestore database.
    3. If the user clicks on an employee, the user will be re-directed to a new page which will display all the information of that employee.
    4. If the user clicks on the delete button, the application deletes the employee from the front end and the Firestore database

- What does the service enable?

    The project can create, update, delete and display the details of employees

Components Used

- Which components will you be using?

  JavaScript, Node.js, React.js, Cloud Functions, Compute Engine, VM Instance, Cloud Firestore database

- What is the purpose of each one?

  Programming language: JavaScript

  Frontend: React.js

  Backend: Node.js, Express, Cloud Functions

  Database: Firestore

  Compute service: Compute Engine

Architecture

- How is the data flowing through the service?

  The user enters the data (employee details) on the web page.

  The front-end code (React.js) deployed in the VM instance on Google Compute Engine passes the data from the web page to the cloud function.

  The cloud function in firebase contains the back-end code (Node.js), it writes the data to the employee database in Firestore.

- How will components connect to each other?

  The front-end code (React.js) deployed in the VM instance on Google Compute Engine captures the data from the web page and passes it to the cloud function.

  The cloud function in firebase contains the back-end code (Node.js), it writes the data to the employee database in Firestore.

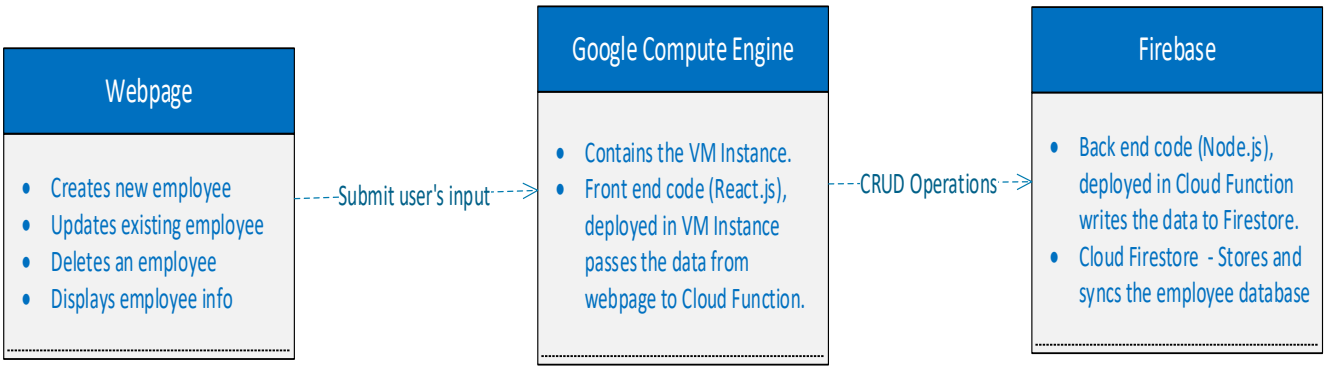
Design

- What language, game framework, etc. are you using?

  React.js, Node.js, JavaScript.

- What are the major code processes/workflows that will control functionality? And a description of major objects in the design

  For the major code processes/workflows and description of major objects, I drew the following UML to clarify them.



Implementation Plan

- What parts of the application and/or design do you plan to implement in what order?

  The order in which we plan to implement the components of the application are

  1. Compute service – Google Compute Engine
  2. VM Instance
  3. A web page – Front end
  4. A serverless execution environment - Cloud Functions
  5. A cloud database - Firestore

- Who is responsible for which parts of implementation?

  Yu - Front-end and Back-end code

  Xiaojing - Google Compute Engine, Code Deployment

  Sujeeth - Firestore, Cloud Functions

- How will you know you are on schedule for finishing and/or a planned milestones timeline?

No.	Task	Due Date
1	Set up the cloud environment	10/16/2021
2	Home page	10/20/2021
3	Create employee page	10/28/2021
4	Mid-point demo	10/28/2021
5	Update employee page	12/02/2021
6	Delete employee function	12/02/2021
7	Test the web service	12/02/2021
8	Final presentation	12/02/2021

Test Plan

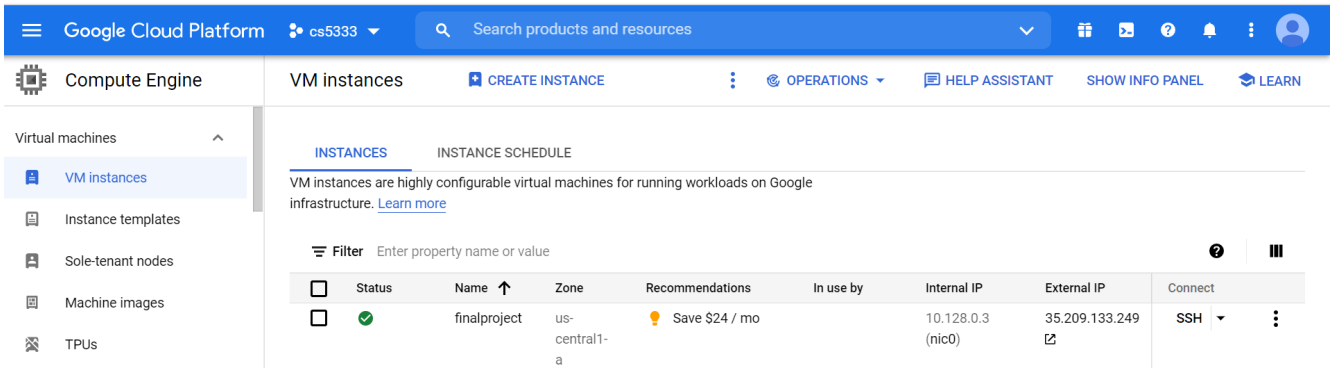
- How will you verify that the service game is working correctly?

The user will create/update/delete new employee details on the webpage. Upon submission, if the input details are created/updated/deleted in the Firestore database accurately and the new employee information is displayed correctly on the webpage, then we can conclude that the web service is working correctly.

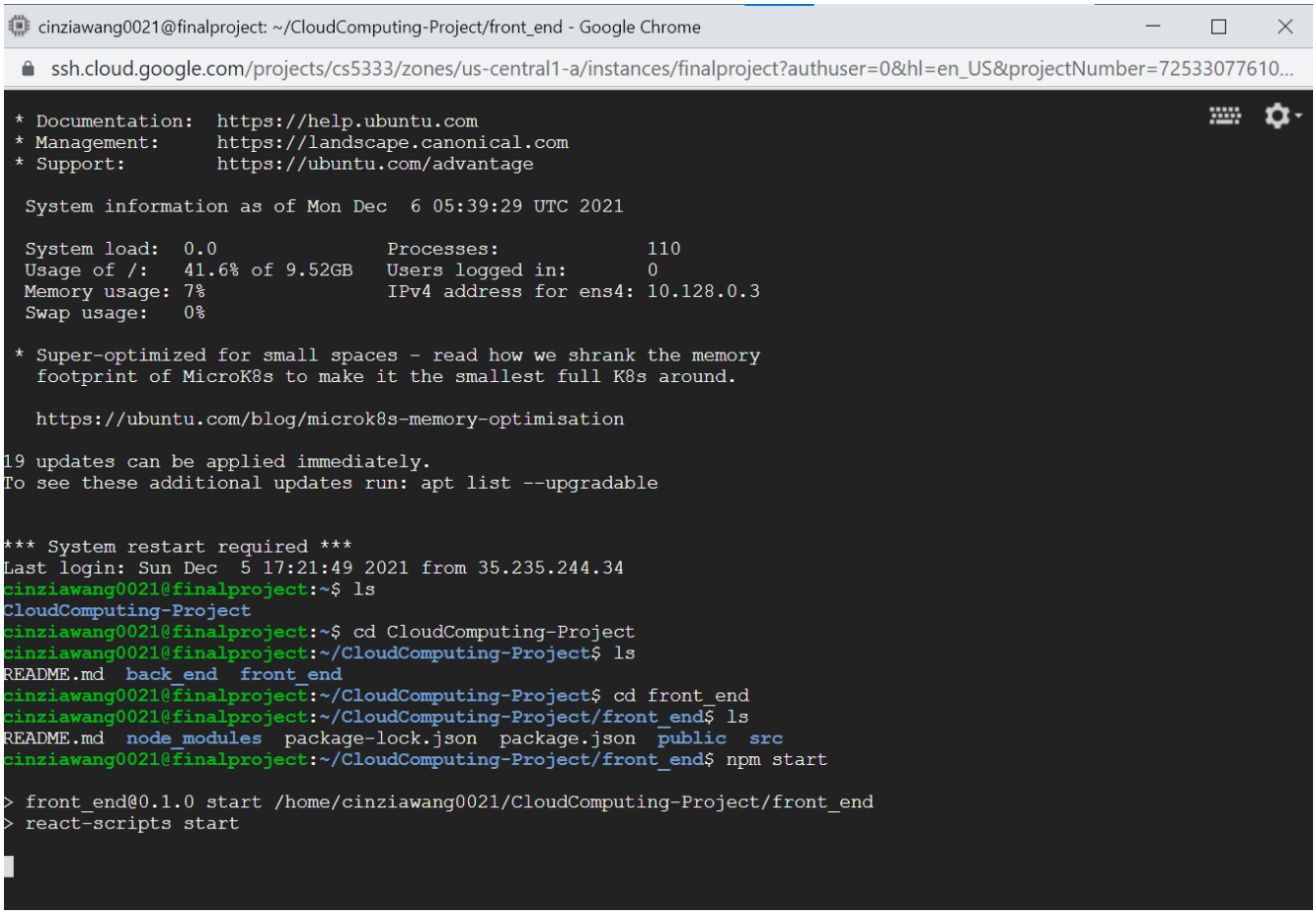
Demo:

Our source code is in GitHub: https://github.com/zhaoyufrank/CloudComputing-Project

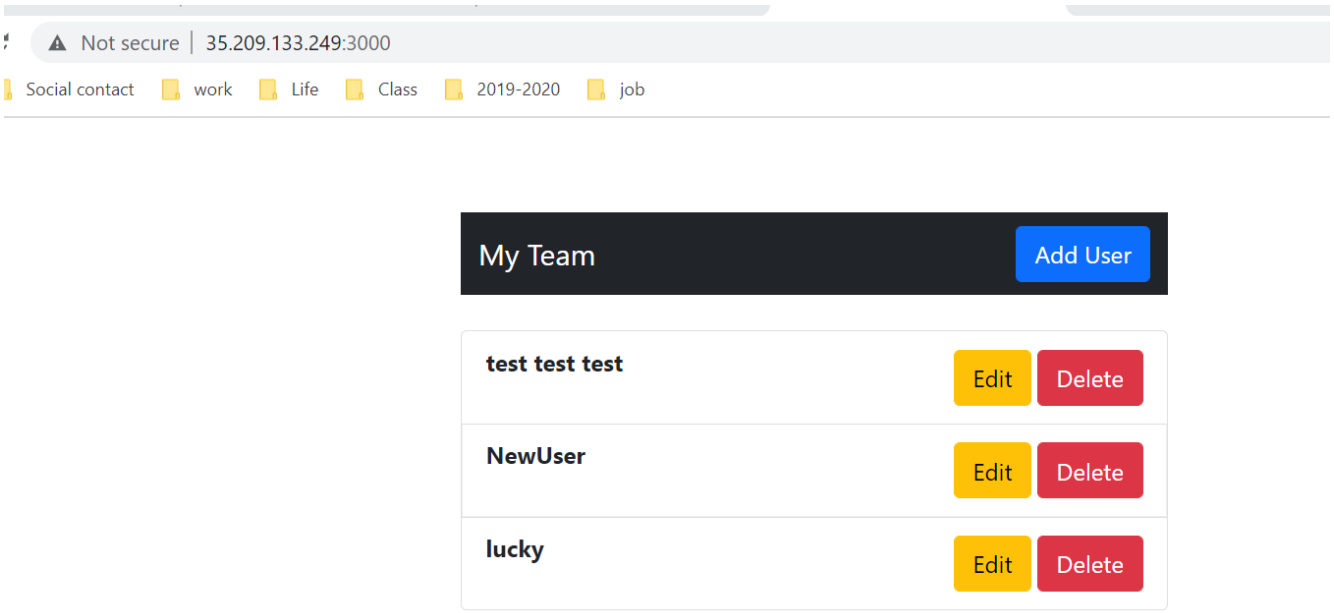
1. Create a project instance, the operating system is Ubuntu 20.0



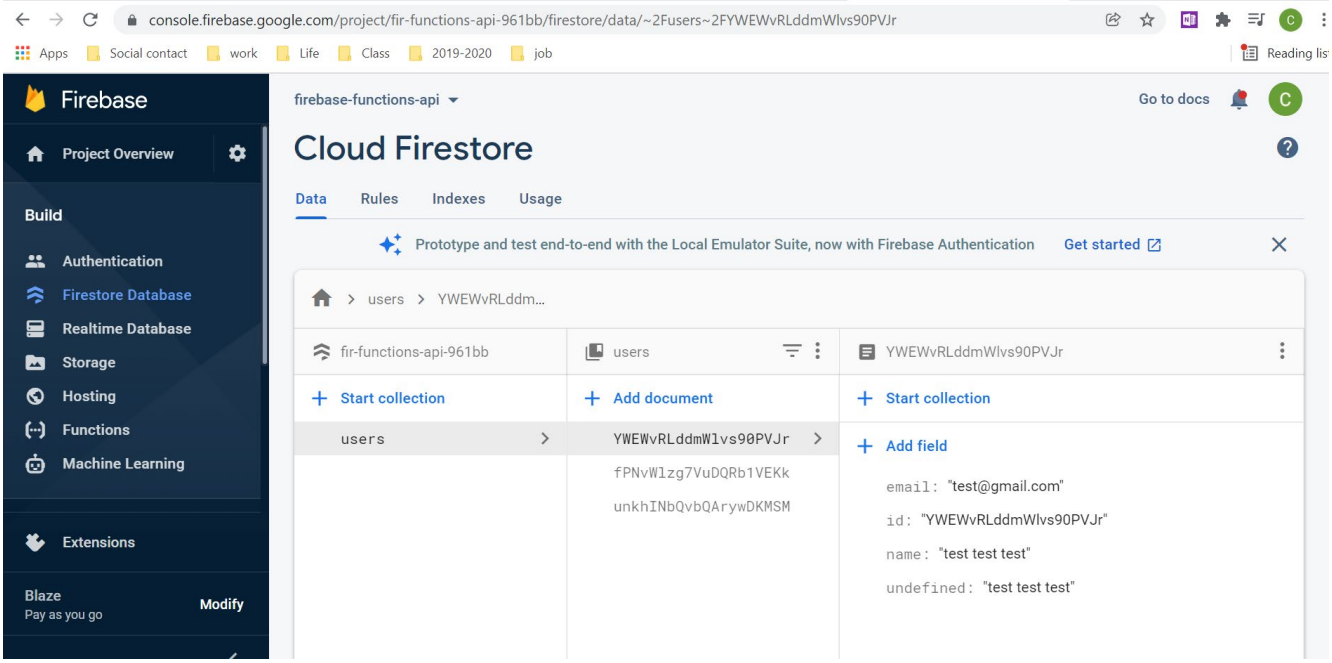
2. Login in and run project.



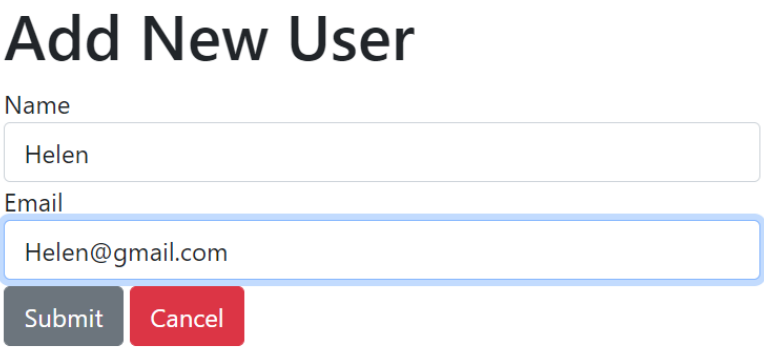
3. Use external IP address and port 3000 to access the application.

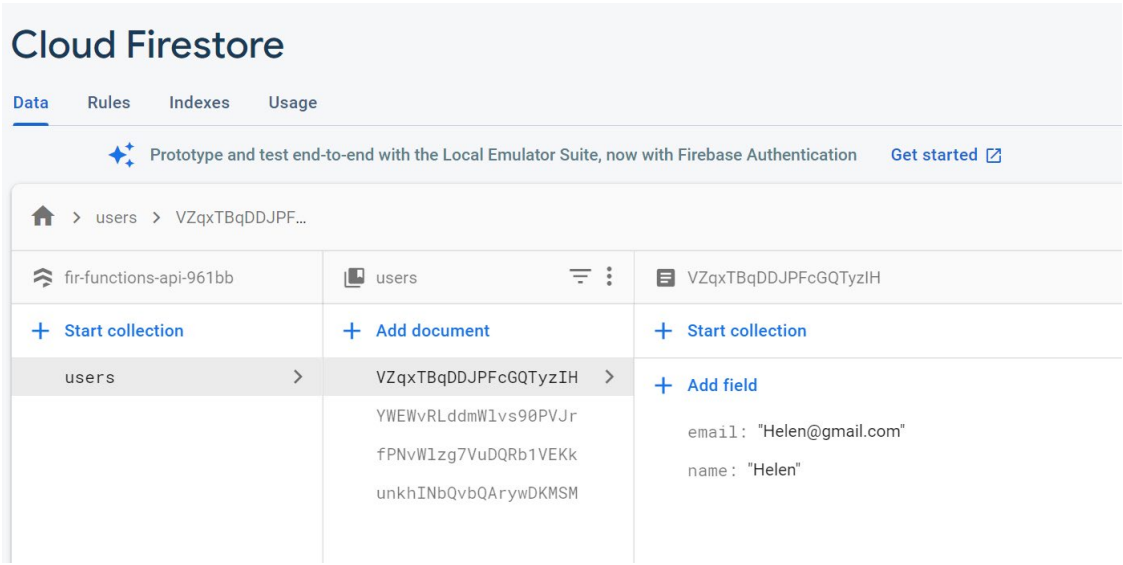


4. Login to Firestore.

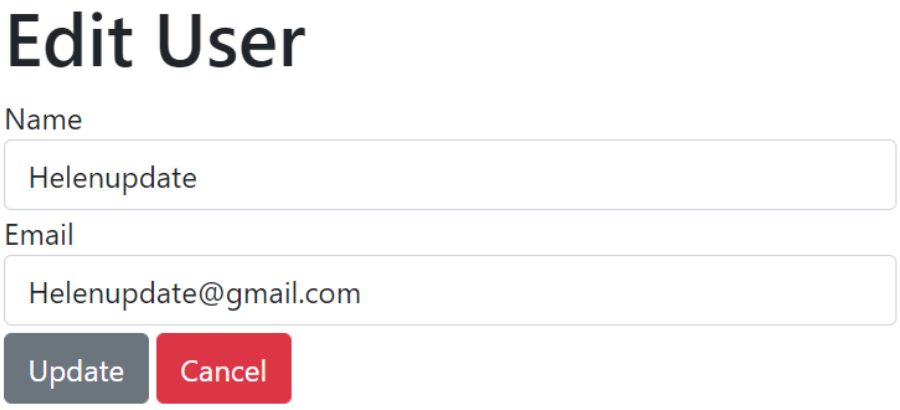


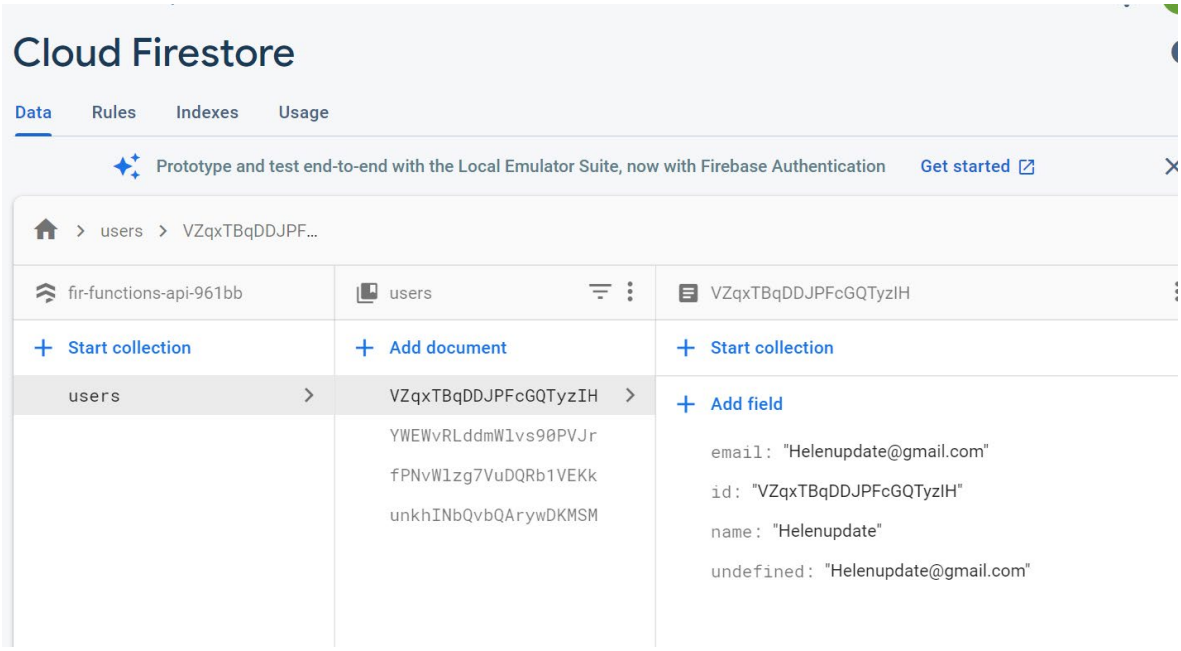
5. Next, we will display three functions of this application
    - Add a user



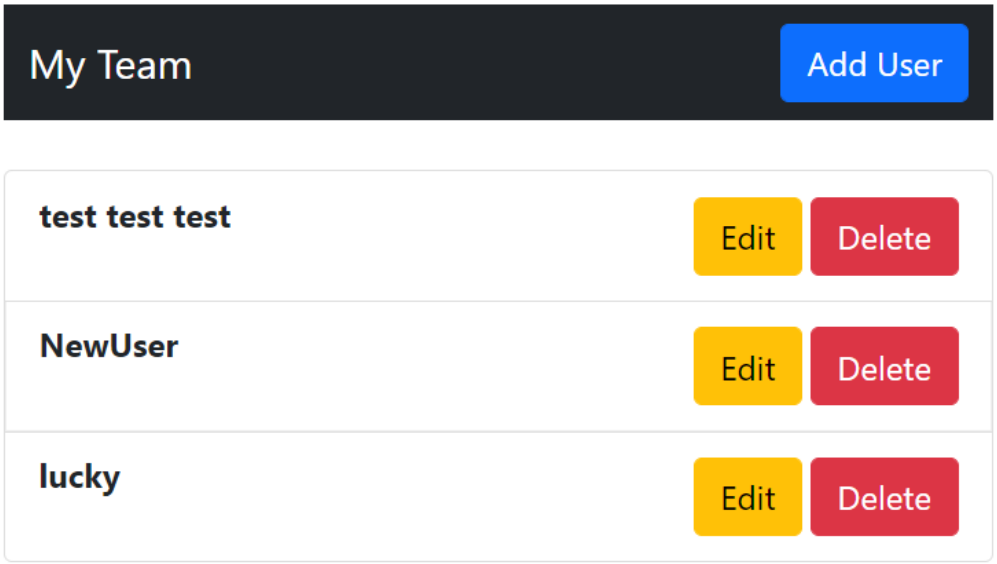


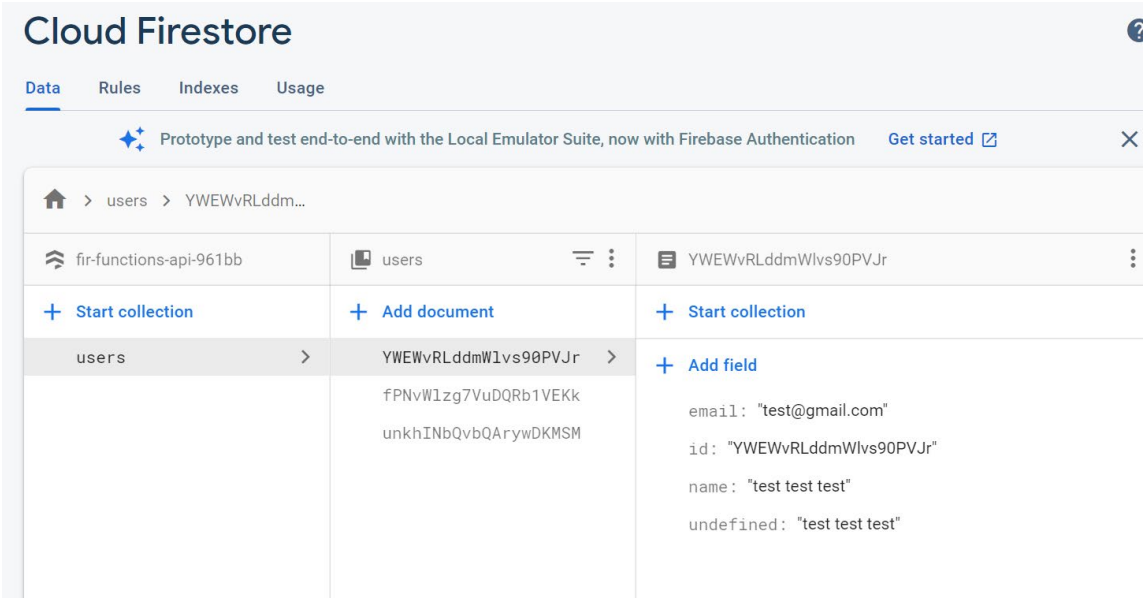
- Edit the user information





- Delete a user



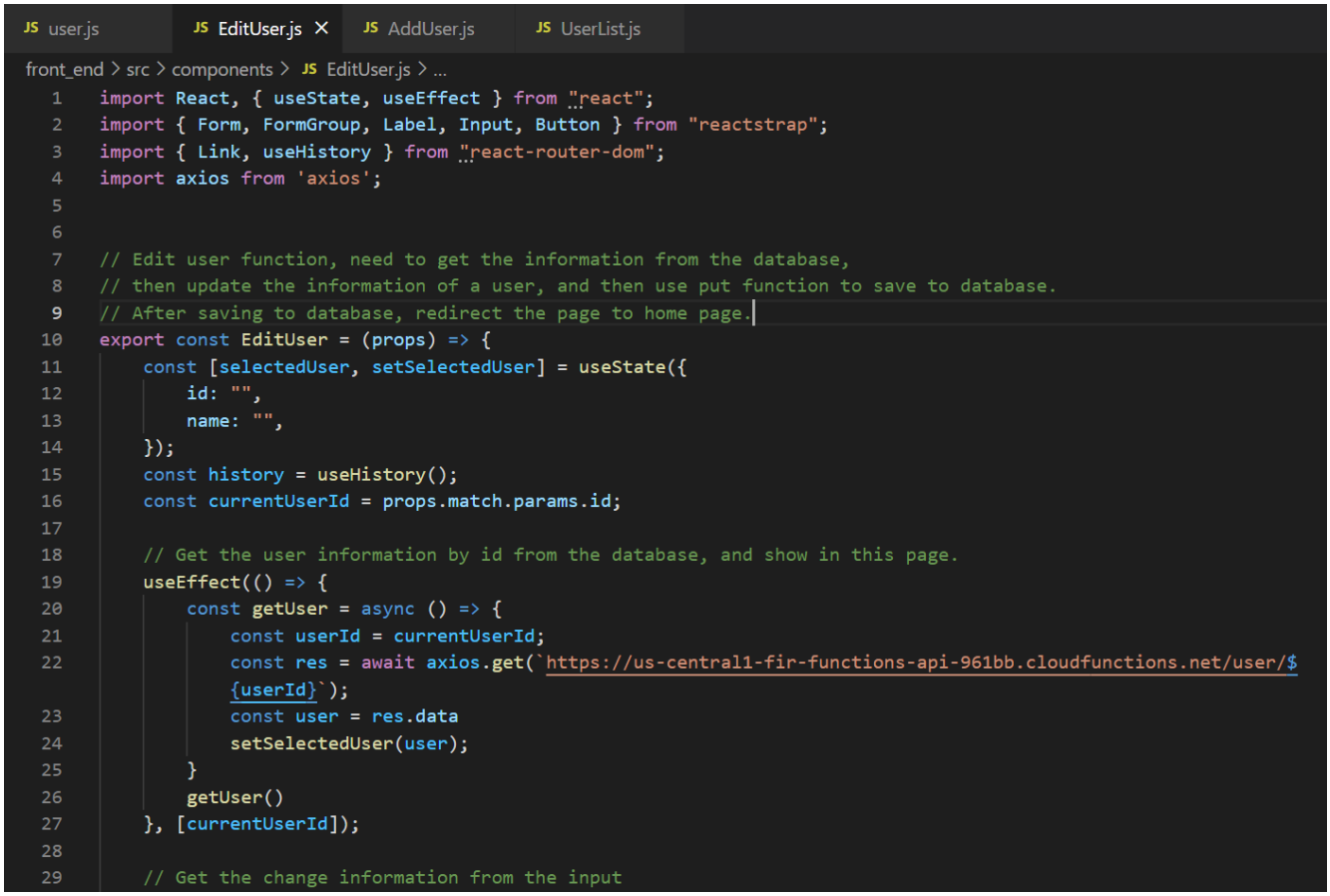

Cloud Firestore
Data
Rules
Indexes
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
users > YWEWvRLddm...
fir-functions-api-961bb
users
YWEWvRLddmWlvs90PVJr
Start collection
Add document
Start collection
users
YWEWvRLddmWlvs90PVJr
Add field
fPNvWlzg7VuDQRb1VEKk
unkhINbQvbQArywDKMSM
email: "test@gmail.com"
id: "YWEWvRLddmWlvs90PVJr"
name: "test test test"
undefined: "test test test"

Code

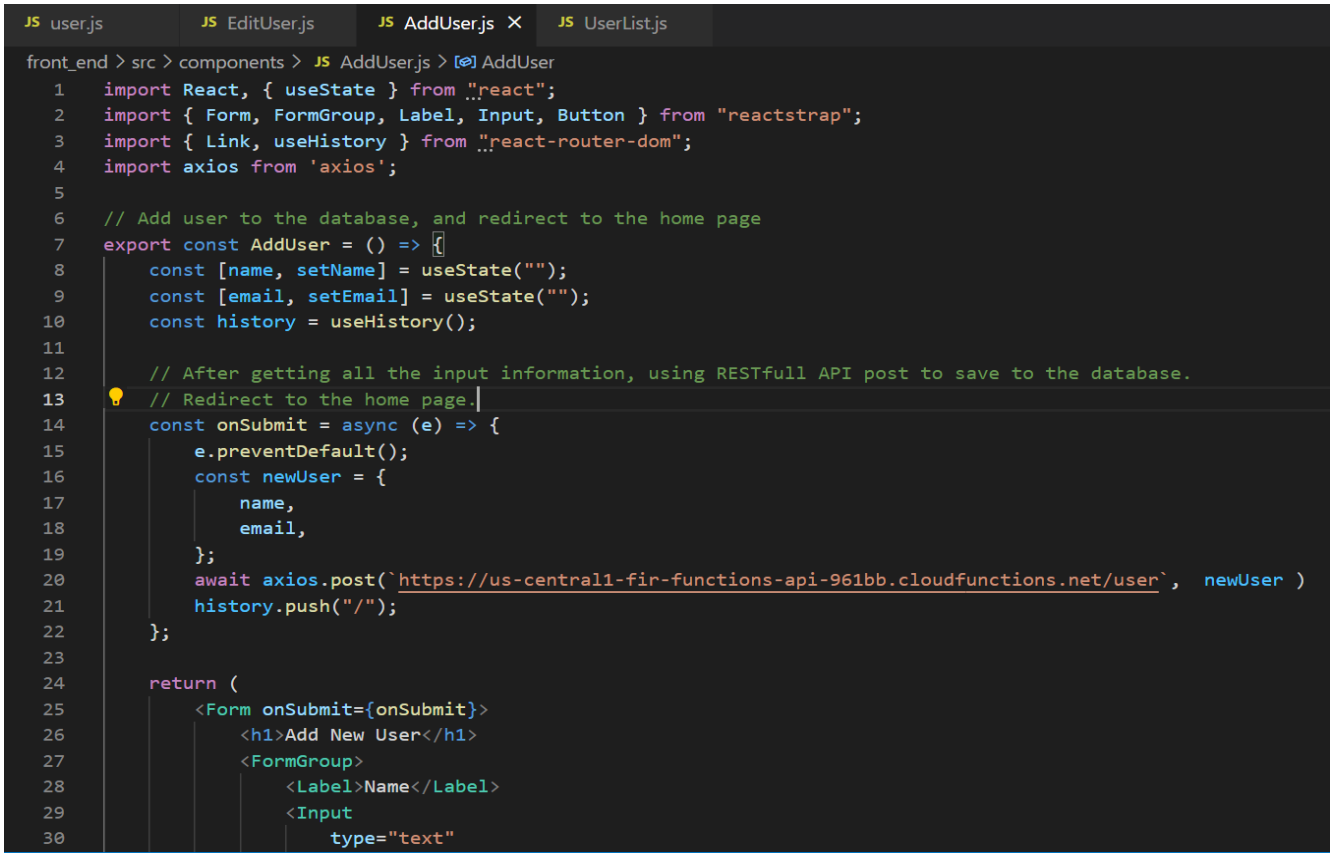
Front end:

- Built the front end by react.js framework.
- Used the “axios” to call the cloud functions URI deployed on the Google Firebase platform.
- Created the Add User page, Edit User page, List User pages and some other components.

Edit User page

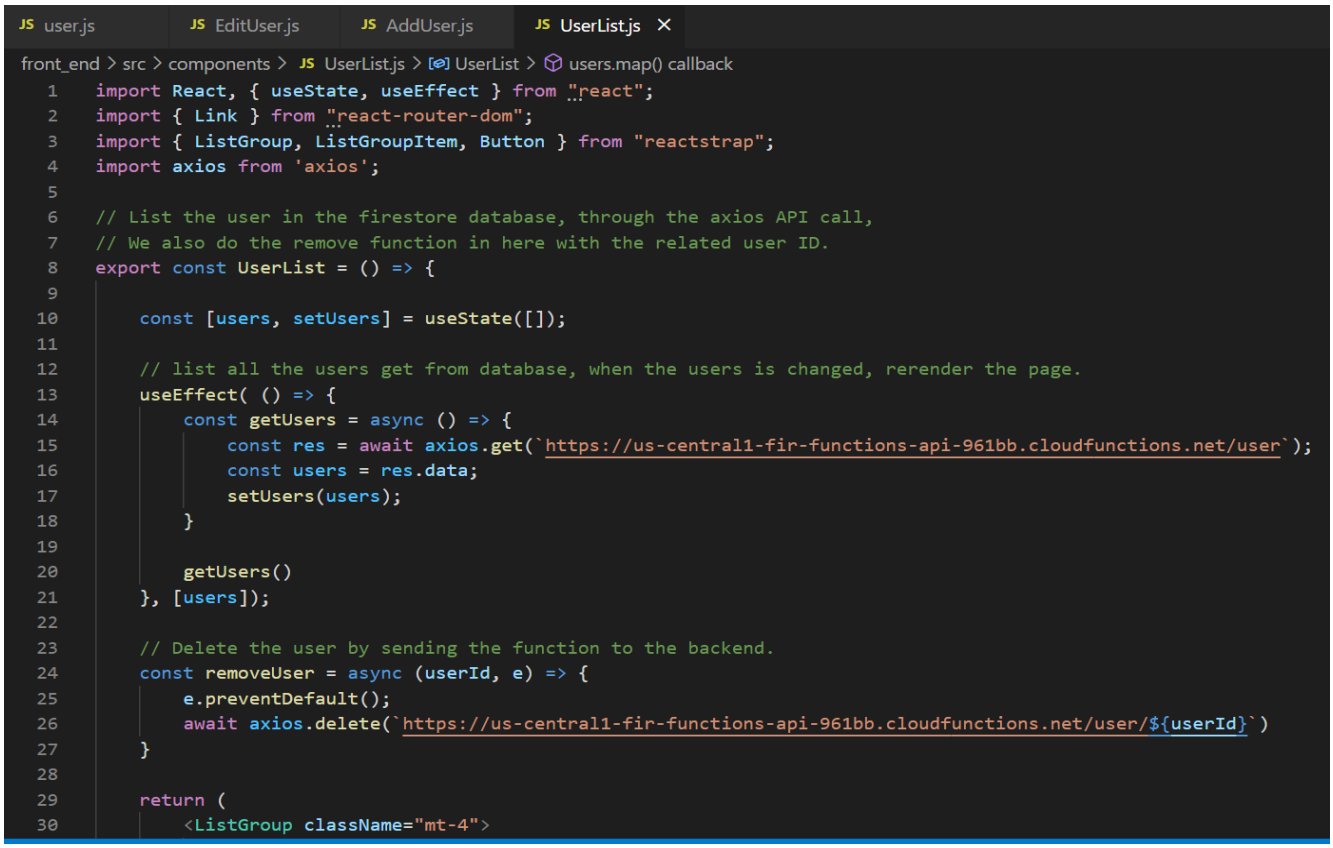


Add User page



```
front_end > src > components > JS AddUser.js > [@] AddUser
1   import React, { useState } from "react";
2   import { Form, FormGroup, Label, Input, Button } from "reactstrap";
3   import { Link, useHistory } from "react-router-dom";
4   import axios from 'axios';
5
6   // Add user to the database, and redirect to the home page
7   export const AddUser = () => {
8       const [name, setName] = useState("");
9       const [email, setEmail] = useState("");
10      const history = useHistory();
11
12      // After getting all the input information, using RESTfull API post to save to the database.
13      // Redirect to the home page.
14      const onSubmit = async (e) => {
15          e.preventDefault();
16          const newUser = {
17              name,
18              email,
19          };
20          await axios.post(`https://us-central1-fir-functions-api-961bb.cloudfunctions.net/user`,  newUser )
21          history.push("/");
22      };
23
24      return (
25          <Form onSubmit={onSubmit}>
26              <h1>Add New User</h1>
27              <FormGroup>
28                  <Label>Name</Label>
29                  <Input
30                      type="text"
```

List User page

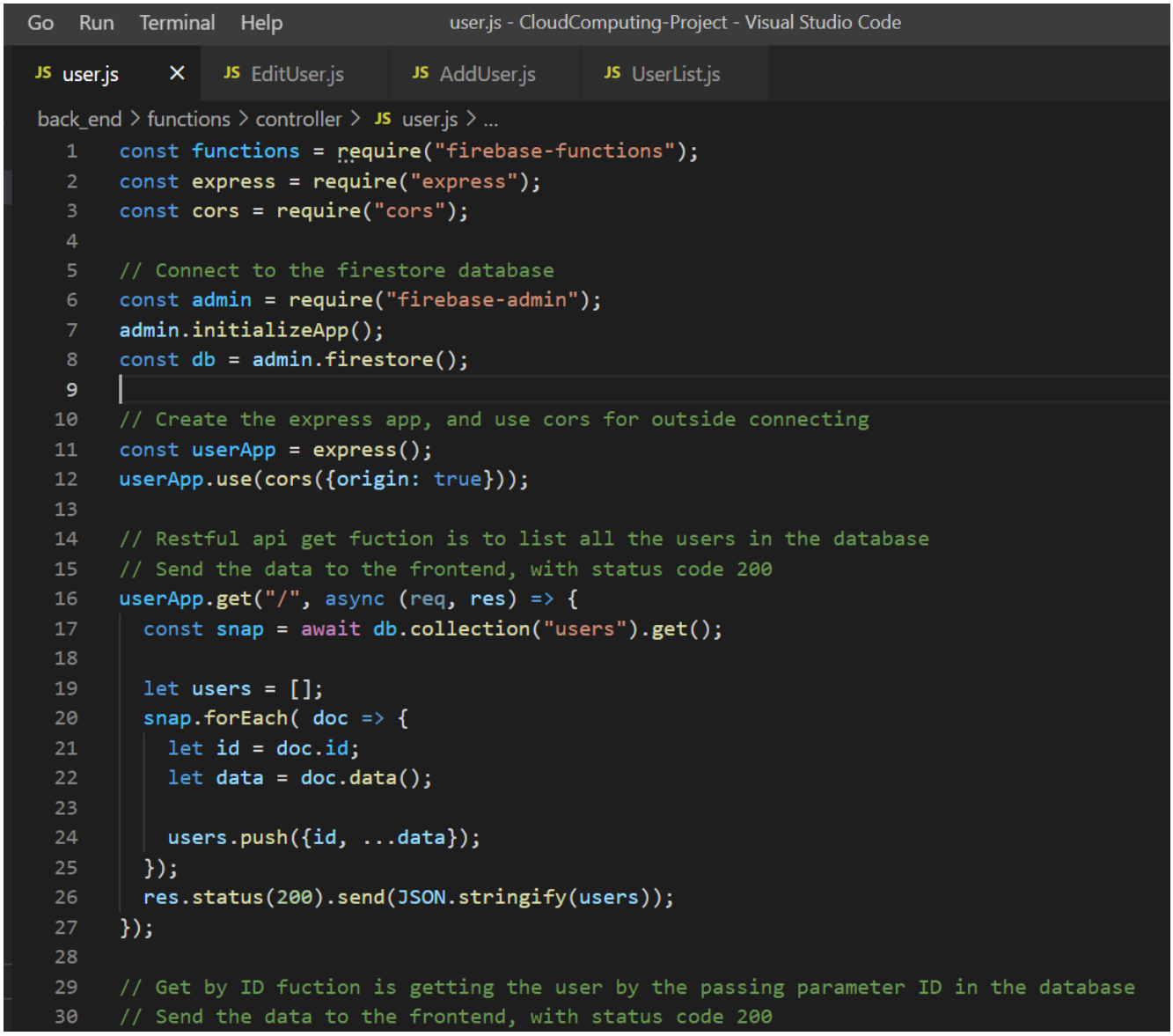


```
front_end > src > components > JS UserList.js > [@] UserList > users.map() callback
1   import React, { useState, useEffect } from "react";
2   import { Link } from "react-router-dom";
3   import { ListGroup, ListGroupItem, Button } from "reactstrap";
4   import axios from 'axios';
5
6   // List the user in the firestore database, through the axios API call,
7   // We also do the remove function in here with the related user ID.
8   export const UserList = () => {
9
10      const [users, setUsers] = useState([]);
11
12      // list all the users get from database, when the users is changed, rerender the page.
13      useEffect( () => {
14          const getUsers = async () => {
15              const res = await axios.get(`https://us-central1-fir-functions-api-961bb.cloudfunctions.net/user`);
16              const users = res.data;
17              setUsers(users);
18          }
19
20          getUsers()
21      }, [users]);
22
23      // Delete the user by sending the function to the backend.
24      const removeUser = async (userId, e) => {
25          e.preventDefault();
26          await axios.delete(`https://us-central1-fir-functions-api-961bb.cloudfunctions.net/user/${userId}`)
27      }
28
29      return (
30          <ListGroup className="mt-4">
```

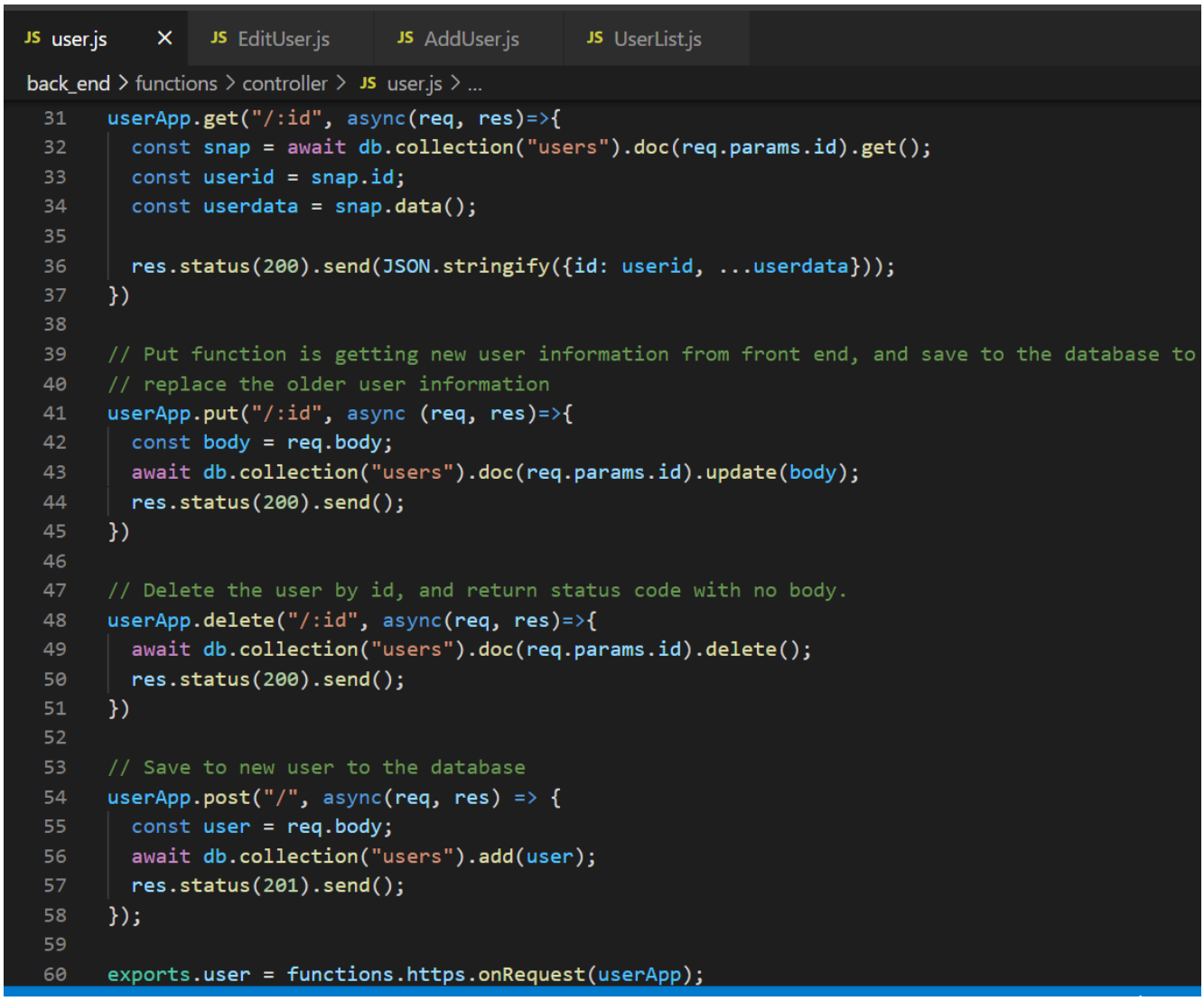
Back end:

- Used the Node.js with express framework to create the back end.
- Used the RESTful API to create GetAll, GetById, EditUser, DeleteById functions.
- Deployed the cloud functions to the Google firebase platform.
- Connected to the Firestore database and do the CRUD operations on the database.

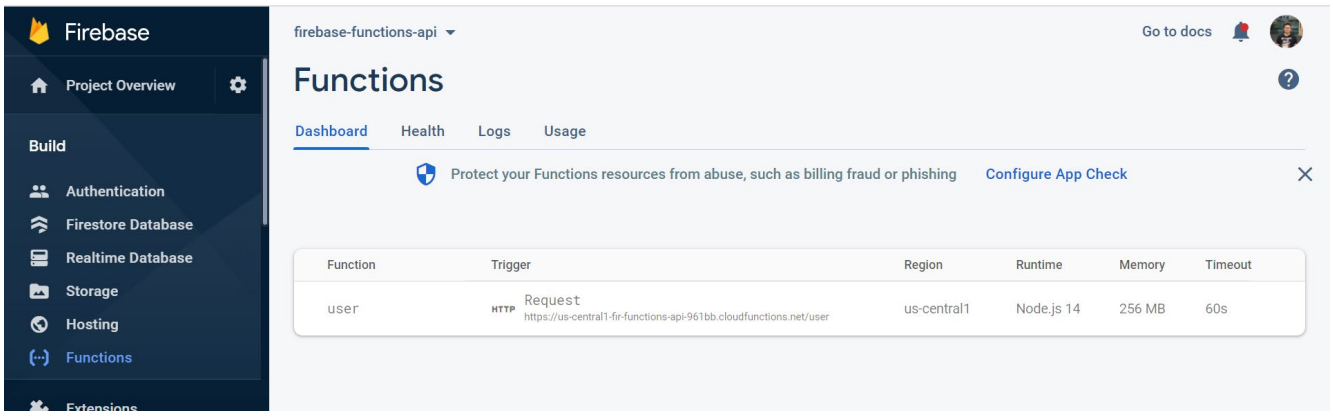
User functions 1



User functions 2



Deployed cloud functions



Teammate Evaluation

Yu Zhao: Project Introduction, Service/Application Overview, Components Used, Front-end and Back-end code

Xiaojing Wang: Architecture, Design, Google Compute Engine, Code Deployment

Sujeeth Nilakantan: Implementation Plan, Test Plan, Firestore, Cloud Functions