

# **Firebase database demo**

**Deval Rajgor**

**Ryan Christopher**

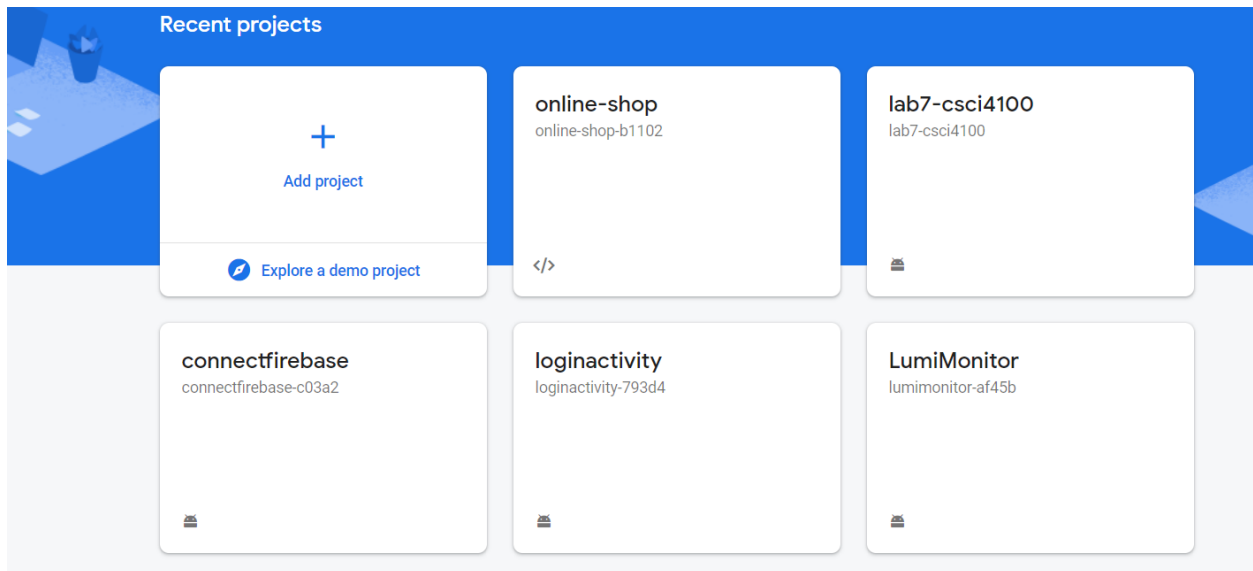
**Arshad Khan**

## Firestore SDK setup

To integrate the NodeJS with firestore admin sdk we follow the steps as following:

### Step1:

Create an empty firestore project:



Our firestore project is named online-shop and simply click on the add project link and follow the steps to create a basic setup. We finally then choose the option of the Realtime database and configure some basic read and write rules.

**Step 2:** We added the necessary import modules into our project from the command line using these commands to facilitate firestore functionality:

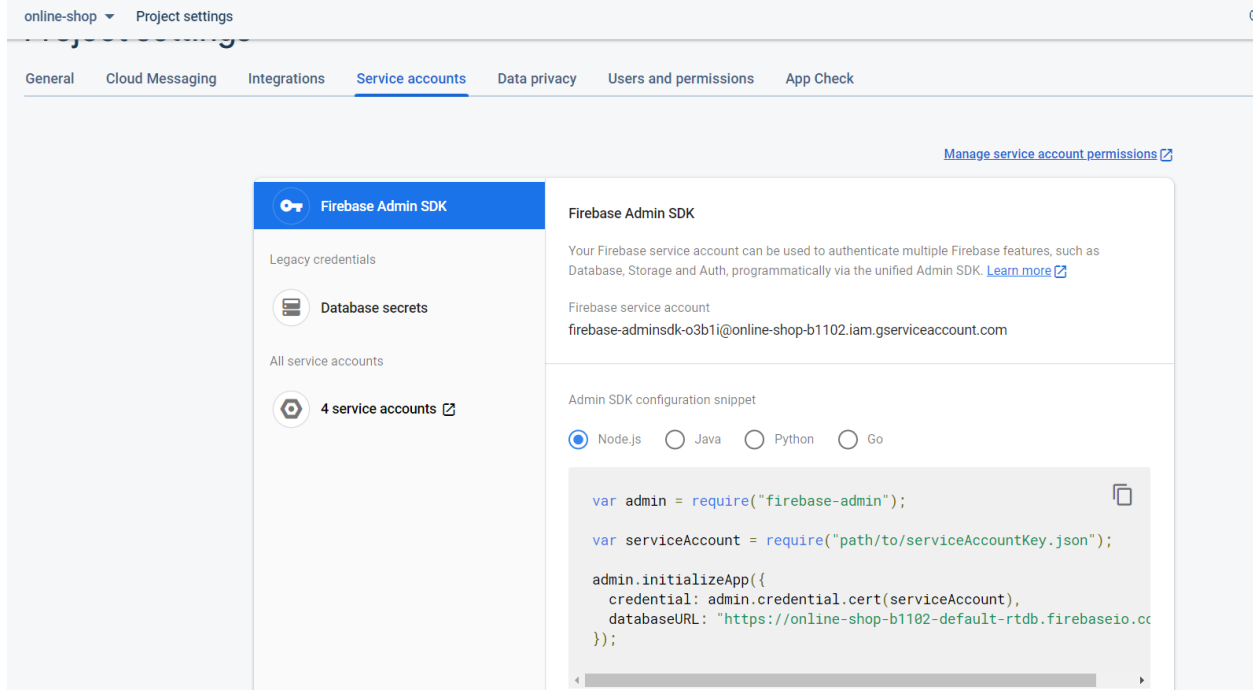
**npm install firestore**

**npm install firestore-admin --save**

We also embed the following modules in our index.js file that acts as the main point for the server:

```
const { getDatabase } = require('firestore-admin/database');  
var admin = require('firestore-admin');
```

**Step 3:** To integrate firebase with our project we further retrieve the service account located under the project settings in firebase and include the following configuration code at the top of the index.js file



We press the option to get the service json file at the bottom and include it in the same directory as our index.js file so that it can be referenced for administrative database authorization and include the code in the picture to grant us access to the cloud database.

#### Step 4: Reference the database

```
// As an admin, the app has access to read and write all data, regardless of Security Rules
var db = admin.database();
var ref = db.ref("/users");
```

We are referencing the cloud database and creating a json structure that stores key value pairs and the /users acting as the node or reference point.


#### Step 5: Start adding to the database

We first created the login and registration files that send/receive data from the server through the /login and /register paths to the node server. We split this section up into two processes login and registration.

## Registration process

When we are first guided to the default login page, we click on the registration hyper link where we are guided to the registration page and simply enter a sample username and password to create an entry in the cloud Realtime database as shown in the below picture:

Registering for an account:

A screenshot of a registration form with a light blue background and a dark blue border. The form contains two text input fields and two buttons. The first input field is labeled 'Enter a UserName:' and contains the text 'deval'. The second input field is labeled 'Enter a Password:' and contains the text 'deval123'. Below the password field are two buttons: 'Register' and 'Cancel', both with a dark blue background and white text.

Enter a UserName:

Enter a Password:

Register

Cancel

Registration entry added into our cloud Realtime database

## Realtime Database

[Data](#) [Rules](#) [Backups](#) [Usage](#)Protect your Realtime Database resources from abuse, such as billing fraud or phishing [Configure App Check](#) ✕<https://online-shop-b1102-default-rtdb.firebaseio.com>

- ▶ -N0bbd7I3mF3Nk1kvcBw
- ▶ -N0bbuC19-xVQS1fSJNc
- ▶ -N0bc4\_TgpmharJ1CYy
- ▶ -N0bc7wD54Y62nIz9ZD0
- ▶ -N0bcALy0S9fqNjWck12
- ▶ -N0bcC7oIcdbItRD9yZW
- ▶ -N0bgYmeegtA7aI5MYBt
- ▼ -N0gvi-axfK0\_\_tMq4r1
  - Password: "deval123"
  - UserName: "deval"

Database location: United States (us-central1)

To do this we have added a NodeJS path that handles request for registration by getting the form fields and then defining a user through username and password pairs and pushing those values to the database reference through a call to push as follows:

```
app.get('/registration', function(request,response){  
  
  let flag = 0  
  var username = request.query.username;  
  var pass = request.query.pass;  
  
  let new_user = ref.push('users');  
  new_user.set({  
    UserName: username,  
    Password:pass  
  });  
  flag =1  
  if(flag = 1)  
  {  
    response.sendFile( __dirname + '/public/register.html');  
  }  
});
```

## Login process

In the login process we first get the login form fields and then query through the database through “an order by value” in the json structure and match the values given by the user to what’s on the database. If a corresponding match is found the page will let the user enter, otherwise it will simply refresh and refuse entry.

## Json database structure

online-shop ▾

### Realtime Database

[Data](#) [Rules](#) [Backups](#) [Usage](#)

Protect your Realtime Database resources from abuse, such as billing fraud or phishing

<https://online-shop-b1102-default-rtdb.firebaseio.com>

Copy reference url

```
https://online-shop-b1102-default-rtdb.firebaseio.com/
├── users
│   ├── -N0RpV7_oa9ER3sshH3L
│   │   ├── Password: "rajgor"
│   │   └── UserName: "dev"
│   ├── -N0Rpp1T0bGKXhNCfGyp
│   │   ├── Password: "dev123"
│   │   └── UserName: "dev"
│   ├── -N0Rq-yXYdFc0gjQ8R9z
│   └── -N0baHb6AAx9wLeTJ0ya
```

Database location: United States (us-central1)

## Code to authenticate user

```

5
7 app.get('/login', function(request,response){
8     var user = request.query.username;
9     var password = request.query.pass;
10    console.log(user);
11    ref.orderByValue().on('value', (item) => {
12        item.forEach((data) => {
13
14            if(data.val().Password== password && data.val().UserName == user)
15            {
16                flag = 1
17            }
18        });
19
20        if(flag == 0 )
21        {
22            response.redirect("/");
23
24            // setTimeout(redirect, 1000);
25
26
27
28
29        }
30        else{
31            response.sendFile(__dirname+'/public/homepage.html');
32            flag = 0;
33        }
34
35    });
36    });

```

## Conclusion:

This demonstration finally concludes adding/ retrieving user information through the firebase Realtime database which provides a quick and efficient implementation of authenticating users and keeping information in a concise, secure, and organized manner.