

---

# Firebase WordPress Integration

*Release 0.11.1*

Apr 08, 2020



---

## Contents:

---

<b>1</b>	<b>Introductions</b>	<b>3</b>
<b>2</b>	<b>How to integrate Firebase to WordPress</b>	<b>5</b>
2.1	Download the Integrate Firebase PRO plugin . . . . .	5
2.2	Prepare Firebase Credentials . . . . .	5
2.3	Installation Process . . . . .	5
2.4	Firebase Cloud Functions Deployment (PRO version only) . . . . .	9
<b>3</b>	<b>List of Current Shortcodes</b>	<b>11</b>
3.1	Authentication . . . . .	11
3.2	Content . . . . .	12
3.3	Realtime Database & Firestore . . . . .	12
3.4	Custom Claims (User's roles) . . . . .	12
<b>4</b>	<b>How To Enable FirebaseUI Web (User Authentication)</b>	<b>13</b>
<b>5</b>	<b>Manage Firebase Users in WordPress Dashboard</b>	<b>17</b>
5.1	Prerequisite . . . . .	17
5.2	Firebase Users Management . . . . .	18
<b>6</b>	<b>WordPress User Integration</b>	<b>21</b>
6.1	Login to WP Dashboard with Firebase Users . . . . .	21
6.2	Create a new WordPress User through API . . . . .	23
<b>7</b>	<b>Sync Data from WordPress to Firebase</b>	<b>25</b>
7.1	1. Setting Up . . . . .	25
7.2	2. Create a Sample Post . . . . .	25
<b>8</b>	<b>Firebase Cloud Message Integration</b>	<b>29</b>
<b>9</b>	<b>How to Save Data from WordPress to Firebase (Realtime + Firestore)</b>	<b>33</b>
9.1	Prerequisite . . . . .	33
9.2	Example of creating new form and writing data to Firestore . . . . .	34
9.3	Example of creating new form and writing data to Realtime . . . . .	35
9.4	Addition Settings . . . . .	35
9.5	Reference . . . . .	35
<b>10</b>	<b>How to Retrieve Data from Firestore and Display on WordPress</b>	<b>39</b>

10.1	Step 1: Create a custom shortcode . . . . .	39
10.2	Step 2: Add custom javascript file . . . . .	41
10.3	Step 3: Retrieve and display data from Firestore . . . . .	41
<b>11</b>	<b>How to Work With Firebase Custom Claims in WordPress</b>	<b>45</b>
11.1	Using shortcode to display custom data . . . . .	45
11.2	Using custom code to display custom data . . . . .	45
<b>12</b>	<b>Use Cases for Integrate Firebase PRO</b>	<b>49</b>
<b>13</b>	<b>Troubleshooting</b>	<b>51</b>
<b>14</b>	<b>Roadmap</b>	<b>53</b>
<b>15</b>	<b>CHANGELOG</b>	<b>55</b>
15.1	## [ 0.11.1 ] - 07-04-2020 . . . . .	55
15.2	## [ 0.11.0 ] - 02-04-2020 . . . . .	55
15.3	## [ 0.10.0 ] - 01-04-2020 . . . . .	55
15.4	## [ 0.9.1 ] - 29-03-2020 . . . . .	56
15.5	## [ 0.9.0 ] - 28-03-2020 . . . . .	56
15.6	## [ 0.8.0 ] - 24-03-2020 . . . . .	56
15.7	## [ 0.7.0 ] - 13-03-2020 . . . . .	56
15.8	## [ 0.6.0 ] - 01-03-2020 . . . . .	57
15.9	## [ 0.5.8 ] - 20-02-2020 . . . . .	57
15.10	## [ 0.5.7 ] - 16-02-2020 . . . . .	57
15.11	## [ 0.5.6 ] - 21-12-2019 . . . . .	57
15.12	## [ 0.5.5 ] - 21-12-2019 . . . . .	57
15.13	## [ 0.5.4 ] - 01-12-2019 . . . . .	57
15.14	## [ 0.5.3 ] - 22-09-2019 . . . . .	57
15.15	## [ 0.5.2 ] - 30-03-2019 . . . . .	58
15.16	## [ 0.5.1 ] - 11-08-2018 . . . . .	58
15.17	## [ 0.5.0 ] - 04-08-2018 . . . . .	58
15.18	## [ 0.4.0 ] - 17-07-2018 . . . . .	58
15.19	## [ 0.3.2 ] - 17-07-2018 . . . . .	58
15.20	## [ 0.3.1 ] - 17-07-2018 . . . . .	58
15.21	## [ 0.3.0 ] - 02-07-2018 . . . . .	58
15.22	## [ 0.2.0 ] - 25-5-2018 . . . . .	58
15.23	## [ 0.1.0 ] - 20-4-2018 . . . . .	59

This is a guide to [Integrate Firebase PRO](#), a plugin that helps to integrate Firebase to WordPress.

Demo site: <https://wordpress.dalenguyen.me>

If you are interested in making this plugin better, please take part in this survey: <https://forms.gle/5TBSDHUtSeVzzKno8>



# CHAPTER 1

---

## Introductions

---

After downloading the [Integrate Firebase PRO](#) plugin, please follow the video in order to have better review of how to install and make use of this plugin in your WordPress site.

The current version of the plugin works best in the situation that you want to take advantage of Firebase features without connecting with the default WordPress users database.



# CHAPTER 2

---

## How to integrate Firebase to WordPress

---

I'm not sure why you want to integrate Firebase and WordPress together because they are two separate system. The authentication is different too. But somehow, you end up here and want to combine Firebase and WordPress together. And yes, we can do it.

### 2.1 Download the Integrate Firebase PRO plugin

There is a FREE version of this [Firebase WordPress integration plugin](#) that you can find in WordPress plugin site. You can give it a try, however, the application is very limited. If you only want to interact with Firebase through WordPress frontend only, then the free version would be sufficient enough.

However, if you want a more secured and advanced feature of the plugin. Please support the development of [Integrate Firebase PRO](#) version.

### 2.2 Prepare Firebase Credentials

Before using the plugin, we need have the credentials from Firebase Console. You can get it by logging into <https://console.firebaseio.google.com>.

After signing in, you can navigate to **Project Overview > Project Settings**.

In General tab, you can get the config at the bottom. If you don't have any app ready, you can create a new one.

After creating a web app for your project, you can save your credentials for later usage.

### 2.3 Installation Process

After you have downloaded the plugin, you can use default installation process in order to install the plugin. Otherwise, you can always manually upload the plugin to your plugin folder through FTP client.

After you activate the plugin, you need to enter the Firebase credentials in the **Dashboard > Firebase**.

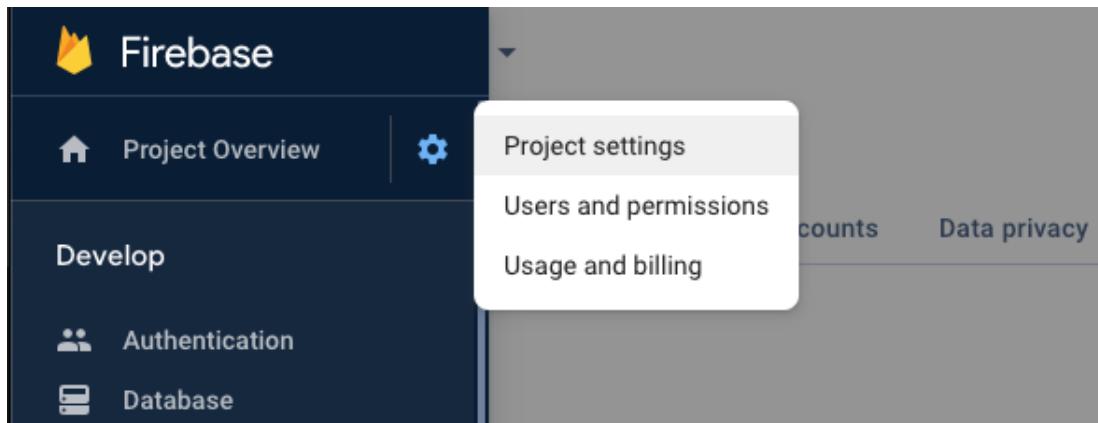


Fig. 1: General configuration

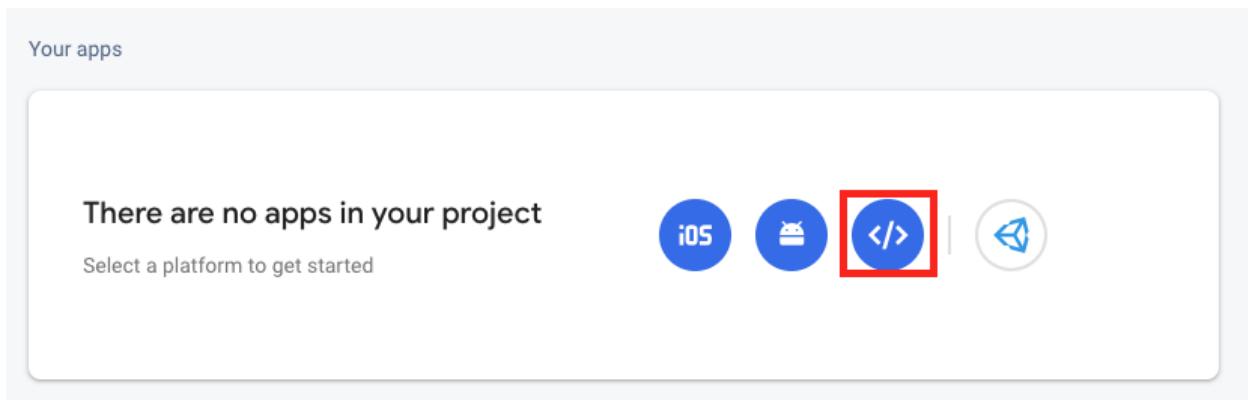


Fig. 2: Create a Web app

The screenshot shows the second step of a two-step process titled 'Add a web app'. Step 1, 'Register app', is completed with a checkmark. Step 2, 'Add Firebase SDK', is the current step, indicated by a progress bar and the number '2'. Below the steps, instructions say to copy and paste scripts into the bottom of a <body> tag. A code block provides the script, with a red box obscuring the sensitive configuration details (API key, authDomain, etc.). At the bottom, there's a note about upgrading to full analytics and a 'Continue to the console' button.

Add a web app

1 2

Register app Add Firebase SDK

Copy and paste these scripts into the bottom of your <body> tag, but before you use any Firebase services:

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/7.8.2.firebaseio-app.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyC3em",
    authDomain: "server",
    databaseURL: "https",
    projectId: "serverl",
    storageBucket: "ser",
    messagingSenderId:
    appId: "1:875634686
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

Learn more about Firebase for web: [Get started](#) [Web SDK API reference](#) [Samples](#)

Note: Upgrade to full analytics to measure web analytics [Upgrade now](#)

Continue to the console

Fig. 3: Firebase Credentials

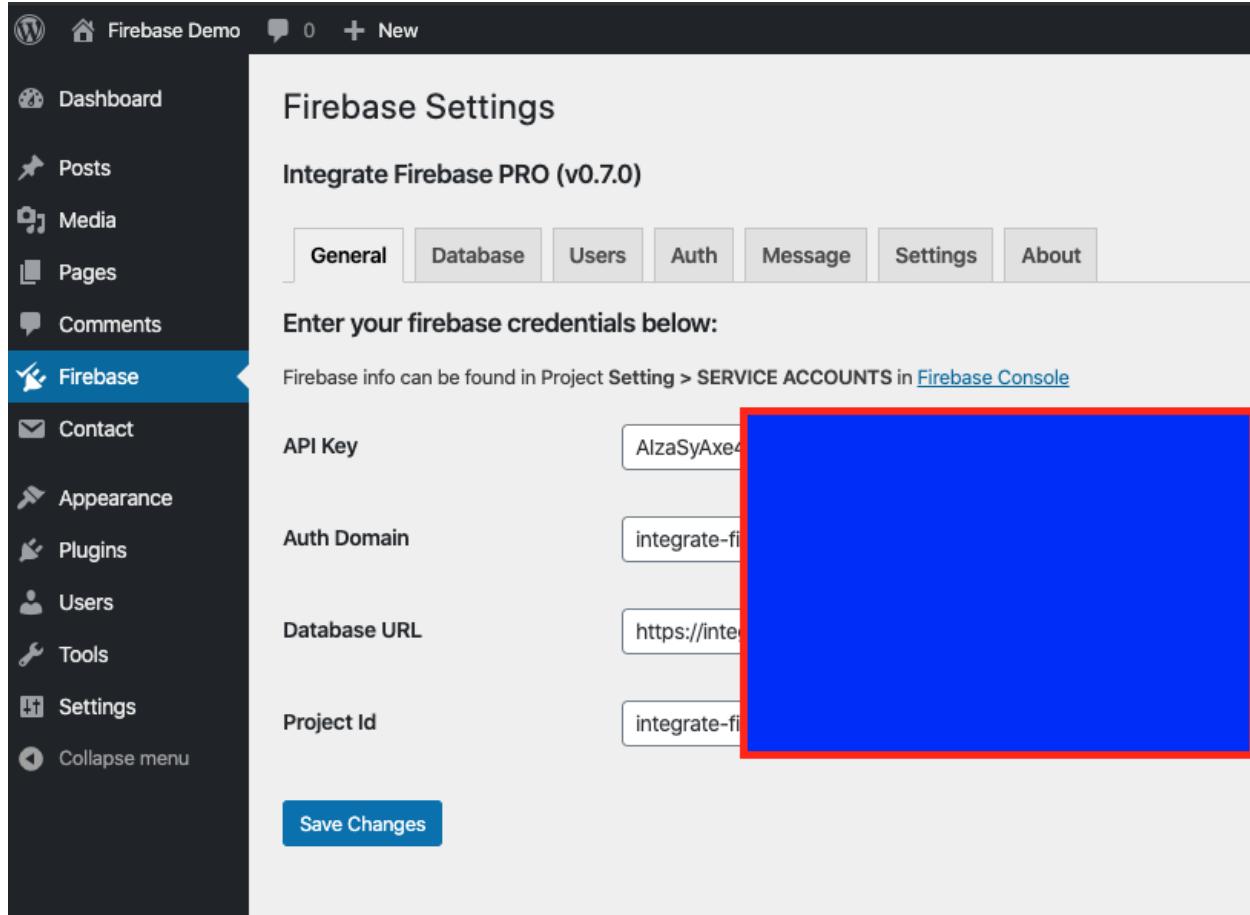


Fig. 4: General configuration

After that, you can create login form, show data, show logout button... on WordPress frontend.

## 2.4 Firebase Cloud Functions Deployment (PRO version only)

If you are using PRO version, there is another folder named **firebase-wordpress-functions**. If you want to manage database, Firebase users and custom functions, you should deploy the functions together with the plugin. Make sure that you have Nodejs installed on your machine.

In order to deploy cloud functions, you need to have [Nodejs](#) installed on your machine.

Then install [firebase-tools](#) packaged

```
npm install -g firebase-tools
```

SignIn and test firebase cli

```
firebase login
```

Since version 0.6.0, before deploying any functions, you should create two tokens for security purpose. One for Wordpress dashboard usage, the other is for Wordress frontend.

```
// Generate random token
node -e "console.log(require('crypto').randomBytes(20).toString('hex'))"

// Set your token to firebase configuration (dashboard token)
firebase functions:config:set api.dashboard_token=your-secret-key --project project-id

// Set your token to firebase configuration (frontend token)
firebase functions:config:set api.frontend_token=your-secret-key --project project-id

// Check your api token
firebase functions:config:get api --project project-id
```

Install packages and build functions. I'm using Yarn, you can use npm if you want.

```
cd functions/
yarn OR npm install
```

The code will go to *functions* folder, then installs packages with yarn / npm.

Start deploying firebase functions

```
cd functions
yarn deploy --project project-id
// OR
firebase deploy --only functions --project project-id
```

The deployment result should look like this

```
✓  functions: Finished running predeploy script.
i  functions: ensuring necessary APIs are enabled...
✓  functions: all necessary APIs are enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (103.29 KB) for uploading
✓  functions: functions folder uploaded successfully
i  functions: updating Node.js 10 (Beta) function api-user(us-central1)...
```

(continues on next page)

(continued from previous page)

```
i  functions: updating Node.js 10 (Beta) function api-database(us-central1)...
✓  functions[api-user(us-central1)]: Successful update operation.
✓  functions[api-database(us-central1)]: Successful update operation.

✓ Deploy complete!
```

Project Console: <https://console.firebaseio.google.com/project/project-id/overview>  
Done in 77.56s.

After that, you should update your Firebase setting with the dashboard token and frontend token and firebase functions url (e.g. <https://us-central1-project-id.cloudfunctions.net>)

The screenshot shows the 'Firebase Settings' interface. At the top, there's a header with the title 'Integrate Firebase PRO to WordPress'. Below the header is a navigation bar with tabs: General, Database, Users, Auth, **Settings**, and About. The 'Settings' tab is currently active. The main content area has a heading 'Please enter your credentials information.' followed by a note: '\* API secret token is mandatory for securely managing Firebase. Never reveal your dashboard api token!'. There are three input fields: 'Base Domain' containing 'https://us-central1-your-project-namelcloudfunctions.net', 'Dashboard API Secret Token' containing '.....', and 'Frontend API Secret Token' also containing '.....'. At the bottom left is a blue 'Save Changes' button.

Fig. 5: Firebase setting

# CHAPTER 3

---

## List of Current Shortcodes

---

Shortcodes are mostly used on WordPress frontend, when you create a page or a post in WordPress dashboard, you can embed in the content.

```
// Page or Post  
[firebaseui_web] [/firebaseui_web]
```

If you are a developer who knows PHP, you can use it anywhere you want.

```
// php files  
echo do_shortcode("[firebaseui_web] [/firebaseui_web]");
```

### 3.1 Authentication

```
// Login, Register through FirebaseUI Web  
[firebaseui_web] [/firebaseui_web]
```

```
// Greetings Logged in User  
[firebase_greetings] [/firebase_greetings]
```

```
// Show Firebase Error  
[firebase_error class='your-class-name'] [/firebase_error]
```

```
// Logout Button  
[firebase_logout] [/firebase_logout]
```

## 3.2 Content

```
// Show custom message for NOT Logged in Users  
[firebase_show_not_login class='your-class-name']YOUR HTML CODE[/firebase_show_not_login]
```

```
// Show custom message for Logged in Users  
[firebase_show class='your-class-name'] YOUR HTML CODE[/firebase_show]
```

## 3.3 Realtime Database & Firestore

```
// show realtime data basing on collection name & document id  
[realtime class='your-class-name' collection_name='string' document_name='string']
```

```
// show firestore data basing on collection name & document id  
[firestore class='your-class-name' collection_name='string' document_name='string']
```

## 3.4 Custom Claims (User's roles)

```
// only user with admin claim will see the content  
// otherwise they will see a Custom message  
[firebase_show_with_claims class='your-class-name' claims='admin' message='Custom_message'] HTML Data With Tags [/firebase_show_with_claims]
```

# CHAPTER 4

## How To Enable FirebaseUI Web (User Authentication)

This feature will help to you register, login and logout on WordPress with your Firebase Users. However, this won't let your users login to WordPress Dashboard. If you want to sync the Users between two systems, please read the WordPress User Integration section in this guide.

Before showing the [FirebaseUI Web](#) on your WordPress, you need to enable sign-in options in *Firebase > Authentication*

The screenshot shows the 'Sign-in method' tab selected in the Firebase Authentication interface. It lists three providers:

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Enabled

Fig. 1: Enable SignIn Options

Since version 0.5.5, all the authentication methods will be handled through FirebaseUI-Web.

```
// Deprecated shortcodes  
[firebase_user_register] and [firebase_login]
```

You need to update the Auth settings in Firebase, in order to enable this feature.

The screenshot shows the 'Integrate Firebase PRO to WordPress' plugin settings page. The 'Auth' tab is active. The 'Sign In Options' section has Google and Email checked. The 'Sign In Success Url' field contains '/test'. There are empty fields for 'Terms of Service Url' and 'Privacy Policy Url'. A 'Save Changes' button is at the bottom.

Fig. 2: FirebaseUI Web Settings

Then you well the login feature on the frontend.

You can enable the login feature by adding this shorcode a post or a page.

```
[firebaseui_web] [/firebaseui_web]
```

## Firebase Shortcode

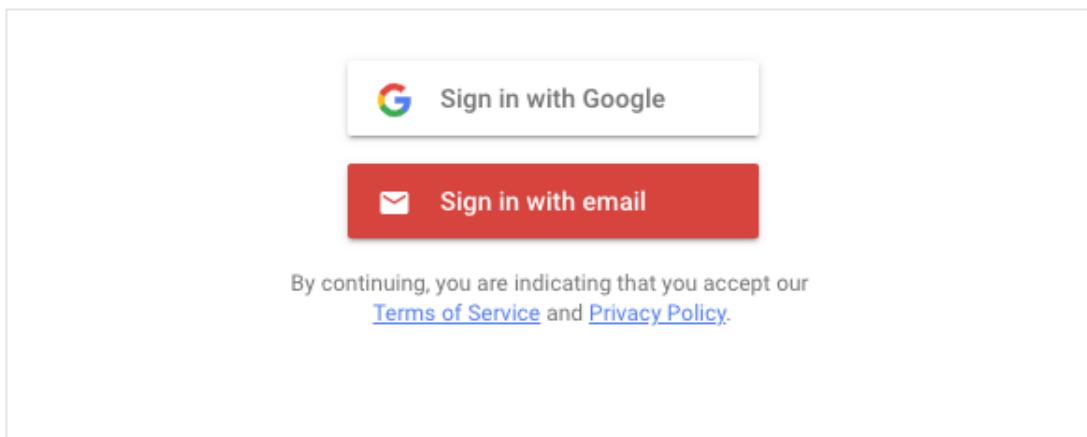


Fig. 3: FirebaseUI Web Frontend



# CHAPTER 5

## Manage Firebase Users in WordPress Dashboard

Google provided a great tool for authentication. However, it is not convenient in terms of user management. As you can see, the options are limited.

claims@dalenguyen.me	✉	15 Mar 2020	29 Mar 2020	rJ2qRtxzOCYeTuLjtffRF0vpZhD3
test9@dalenguyen.me	✉	13 Mar 2020	13 Mar 2020	CJpEaW0ch9WemkU4OT
test7@dalenguyen.me	✉	13 Mar 2020	13 Mar 2020	9j3aziDqSleRAth5eWYeul...
test5@dalenguyen.me	✉	13 Mar 2020	13 Mar 2020	BNCLLgHOPwhFvvjUddPlbEpNm2...
test4@dalenguyen.me	✉	13 Mar 2020	13 Mar 2020	ee3CKRsYiRMPh8pB7rgPNWDb08...
test3@dalenguyen.me	✉	13 Mar 2020	13 Mar 2020	cTQkaXV3pjPGgb3cjfKPmTHJrFG2

Fig. 1: Firebase default options

That is why I want to make it easier for the webmaster if they want to create, read, update and delete (CRUD) a Firebase user directly through WordPress dashboard.

### 5.1 Prerequisite

Before you can achieve this, you have to configure **Firebase > Settings** with your Base Domain and Dashboard API Secret Token. You can take a look at how to set it up the [Integrate Firebase to WordPress](#) section.

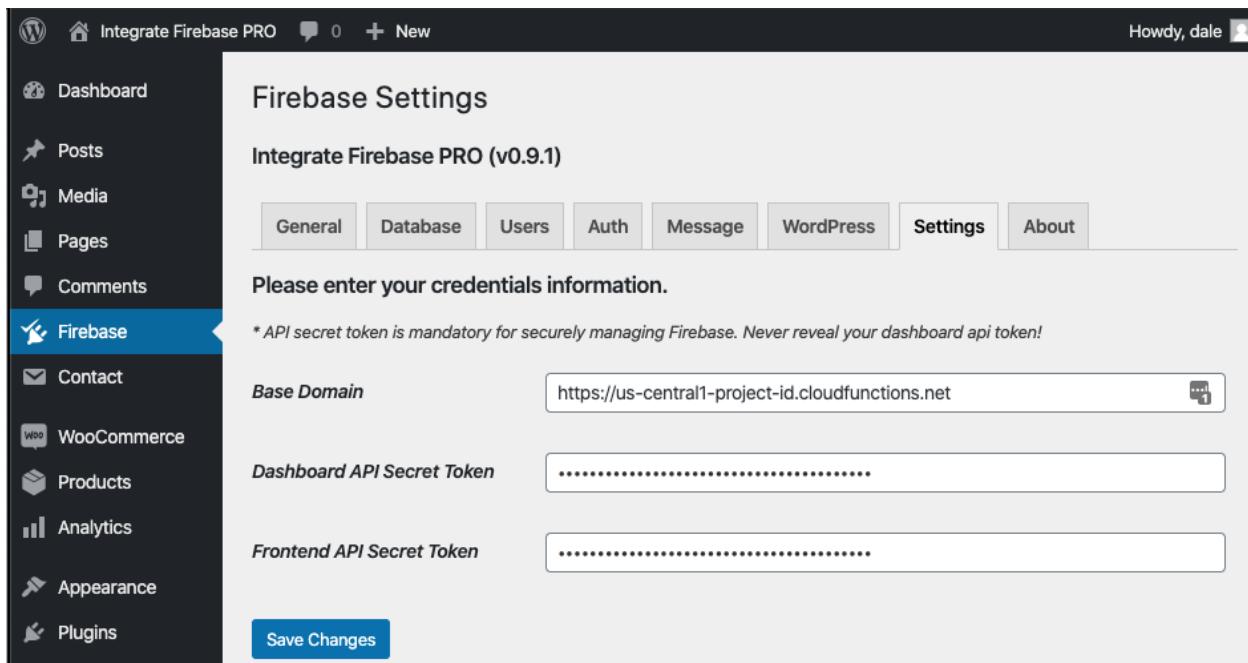


Fig. 2: Firebase settings

## 5.2 Firebase Users Management

User management is under the **Firebase > Users** tab. You will see a form where you can add a new User to Firebase. Remember that the user will be added to Firebase, not WordPress.

If the process is successfull, you will be notified with a message that contains the user id.

If you want to read, modify or delete a user, you can press on “Click to load users”. You can find the information about Firebase users where you can update the display name, email, phone number, and even user custom claims.

When you delete a user, there will a prompt to confirm if you want to delete it to make sure that you do not accidentally press the delete button.

**Note:** the limit now is 1000 users. I will try to make it better in the next release.

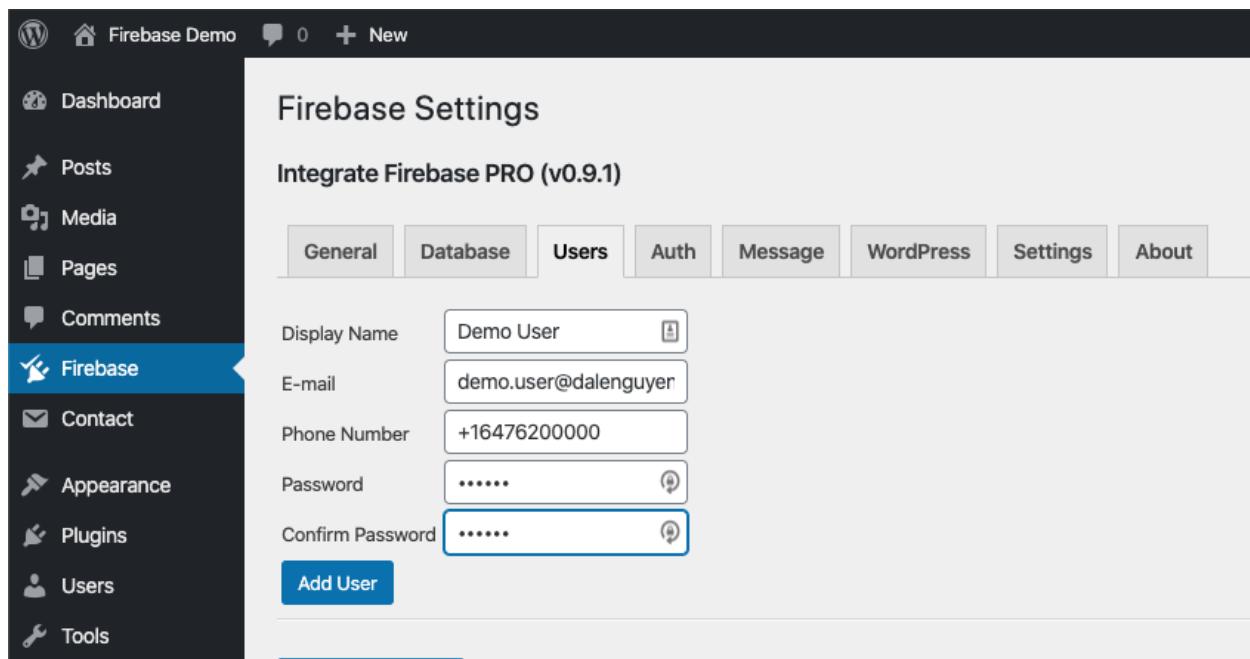


Fig. 3: Add a new Firebase user

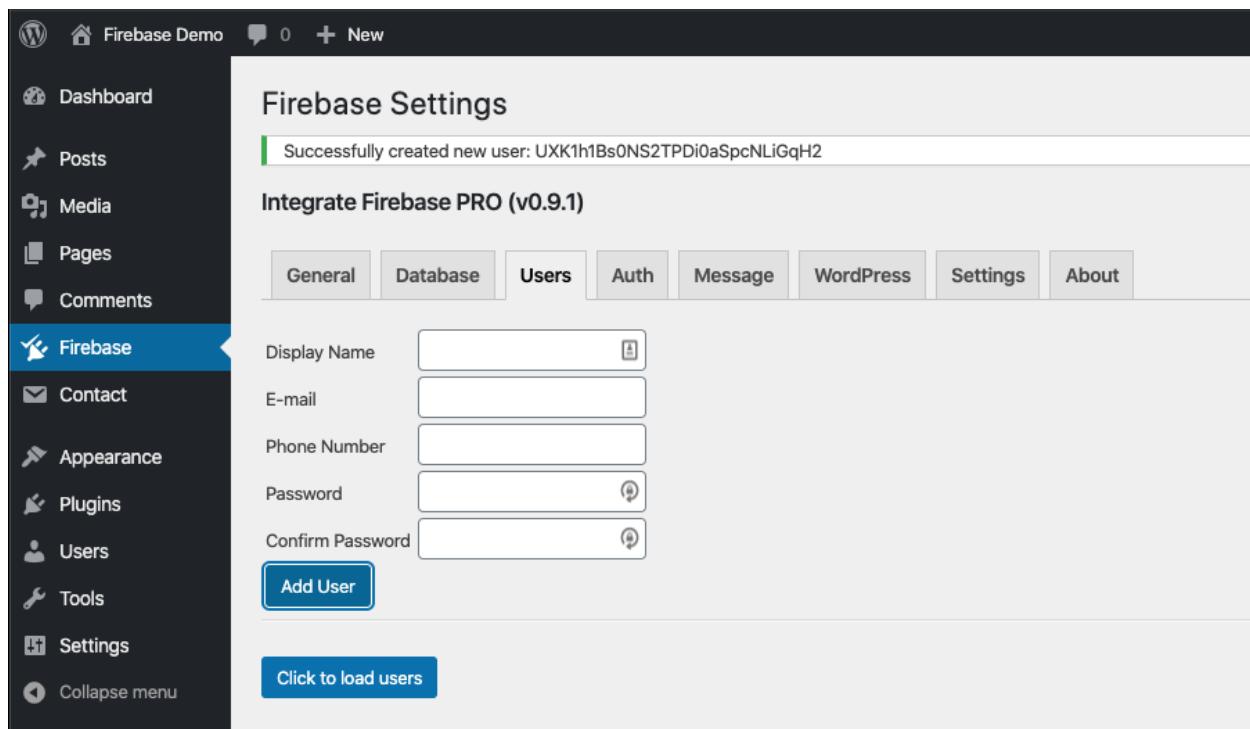


Fig. 4: Firebase user is added

## Firebase WordPress Integration, Release 0.11.1

The screenshot shows the Firebase WordPress Integration dashboard. On the left, a sidebar menu includes options like Dashboard, Posts, Media, Pages, Comments, Firebase (which is selected), Contact, Appearance, Plugins, Users, Tools, Settings, and Collapse menu. The main content area is titled "Firebase Settings" and displays a success message: "Successfully created new user: UXK1h1Bs0NS2TPDi0aSpcNLiGqH2". Below this, it says "Integrate Firebase PRO (v0.9.1)" and lists tabs for General, Database, Users (selected), Auth, Message, WordPress, Settings, and About. The "Users" tab contains fields for Display Name, E-mail, Phone Number, Password, and Confirm Password, with a blue "Add User" button. A note below the fields says "\* Custom claims format example: {"admin": true, "guest": false}\*. A table lists users with columns for UID, Display Name, Email, Phone Number, Claims, and Action. Two users are listed: "HELLO YEN1" (UID: 5WVWhZv9YgbqcJRV2QvtCaqXSh82) and "Yen Chan" (UID: 6LTWEpiYyeQLGNXwbIYY5jj30Kv1). The "Action" column for each user has "Delete" and "Edit" buttons. At the bottom of the page, there's a note about custom claims.

Fig. 5: Load Firebase Users

This screenshot is similar to Fig. 5, showing the Firebase WordPress Integration dashboard. A modal window is open in the center, prompting the user to type "DELETE" to permanently delete a user. The modal has a text input field containing a single character, a "Cancel" button, and an "OK" button. The background dashboard shows the same user list and interface as Fig. 5.

Fig. 6: Delete a Firebase User

# CHAPTER 6

---

## WordPress User Integration

---

This feature is available since **v0.7.0**. There are a lots of tasks that needs to be done in order to make the integration much better for WordPress and Firebase Users because they are in two separate systems.

This feature solves the authentication problem. This is an example from one of my potential clients. The client has a mobile app (Firebase users), and they want those users to login to a separate WordPress website without registering again.

And this plugin can help you to achieve just that.

At this stage, it helps to authenticate to WordPress dashboard with a subscriber or customer role (WooCommerce). And the password change will be dominated by Firebase Users. User cannot change password when logging into WordPress dashboard. They have to use *forgot password* feature in Firebase in order to create a new password. And if the password is different in both systems, the next time user logs in through FirebaseUI Web shortcode, it will change the password in WordPress automatically.

### 6.1 Login to WP Dashboard with Firebase Users

**Important:** Before doing this, you should keep a dashboard open for yourself, and open another private window in order to login with an email from Firebase, then assign the Admin right to that user before you log out of current window - to make sure that you can log in again with your Firebase user (with **admin** rights).

In case of logging out without assign another user with admin rights, you can rename the plugin folder, and login as usual.

This flow will utilize FirebaseUI Web workflow in order to authenticate users. In order to that, you have to log in to WordPress Dashboard, then **Dashboard > Firebase > Auth**.

Check **Allow Login to WP Dashboard** and enter you **Login Url**. It could be your homepage or a separate page just for logging in. From now, everytime users navigate to <https://your-webiste.com/wp-admin>, it will redirect to your new login page.

This page will contain the shortcode for logging in.

The screenshot shows the 'Firebase Settings' page within the WordPress admin interface. The left sidebar has a 'Firebase' section selected, indicated by a blue background. The main content area is titled 'Firebase Settings' and displays the message 'Integrate Firebase PRO (v0.7.0)'. Below this are several configuration sections:

- Login Page Configuration:** Includes a note: '\* Enter these fields if you want to login through Firebase Web UI.' A checkbox labeled 'Allow Login to WP Dashboard' is checked and highlighted with a red box.
- Login Url:** The URL 'https://wordpress.dalenguyen.me/sign-in/' is entered and highlighted with a red box.
- FirebaseUI Web Configuration:** Includes a note: '\* Configuration for FirebaseUI Web.' It contains four sign-in options: Google (unchecked), Facebook (unchecked), Twitter (unchecked), Github (unchecked), and Email (checked).
- Sign In Options:** Fields for 'Sign In Success Url' (containing '/'), 'Terms of Service Url' (containing '/terms'), and 'Privacy Policy Url' (containing '/privacy').
- Save Changes** button at the bottom.

Fig. 1: Firebase Auth Settings

```
// Login, Register through FirebaseUI Web  
[firebaseui_web] [/firebaseui_web]
```

## Firebase Demo

WordPress Firebase Integration

# Sign In

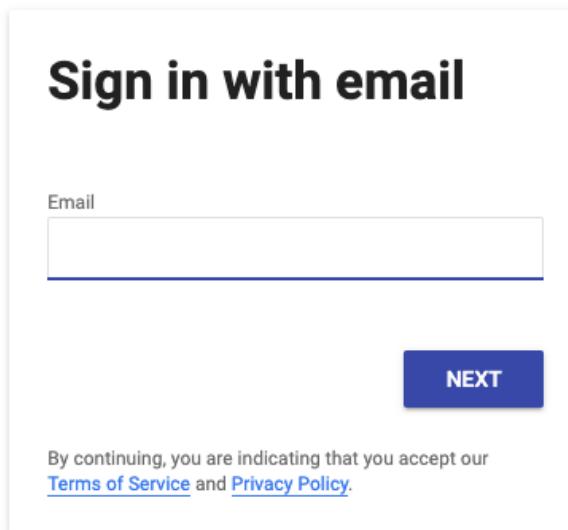


Fig. 2: New WP Login Page

After users log in, it will create a new user in WordPress if the user doesn't exist. Then authenticate it to WordPress dashboard automatically.

When users log out from dashboard, that means they will also be logged out to Firebase.

## 6.2 Create a new WordPress User through API

The Integrate Firebase PRO has its own Restful API endpoints that help to create a new WordPress user. In this scenario, when you have a mobile app, and you want to duplicate user in WordPress, you can call the API to create a new User after user register on your app.

```
Endpoint: POST https://example.com.firebaseio/v2/users/register
```

```
Example payload: {  
    username: 'dale',  
    email: 'dale@dalenguyen.me',  
    password: 'the-password'  
}
```

# CHAPTER 7

---

## Sync Data from WordPress to Firebase

---

Since version 0.9.0, I added some event triggers for syncing data from WordPress to Firebase. The plugin now supports syncing Post & Page from WordPress to Firebase. That means when you add or update a new Post or Page. The content will be updated automatically to Firebase (Realtime / Firestore).

- For posts, they locate under **wpPosts** collection name.
- For pages, they locate under **wpPages** collection name.

### 7.1 1. Setting Up

Make sure the version of the plugin is at least: v0.9.0. Then you can choose which type of data to sync to Firebase.

In this example, it will choose Firetore as the database, and every new or update posts will be synced to Firestore. You can choose both Post & Page type.

### 7.2 2. Create a Sample Post

After that, you can create a new post or update an existing one.

After you save, the post will be added to Firestore. The document id of the post is also the post id for query purpose.

When you update the current post, it will override the data in the firestore to make sure that it always updated.

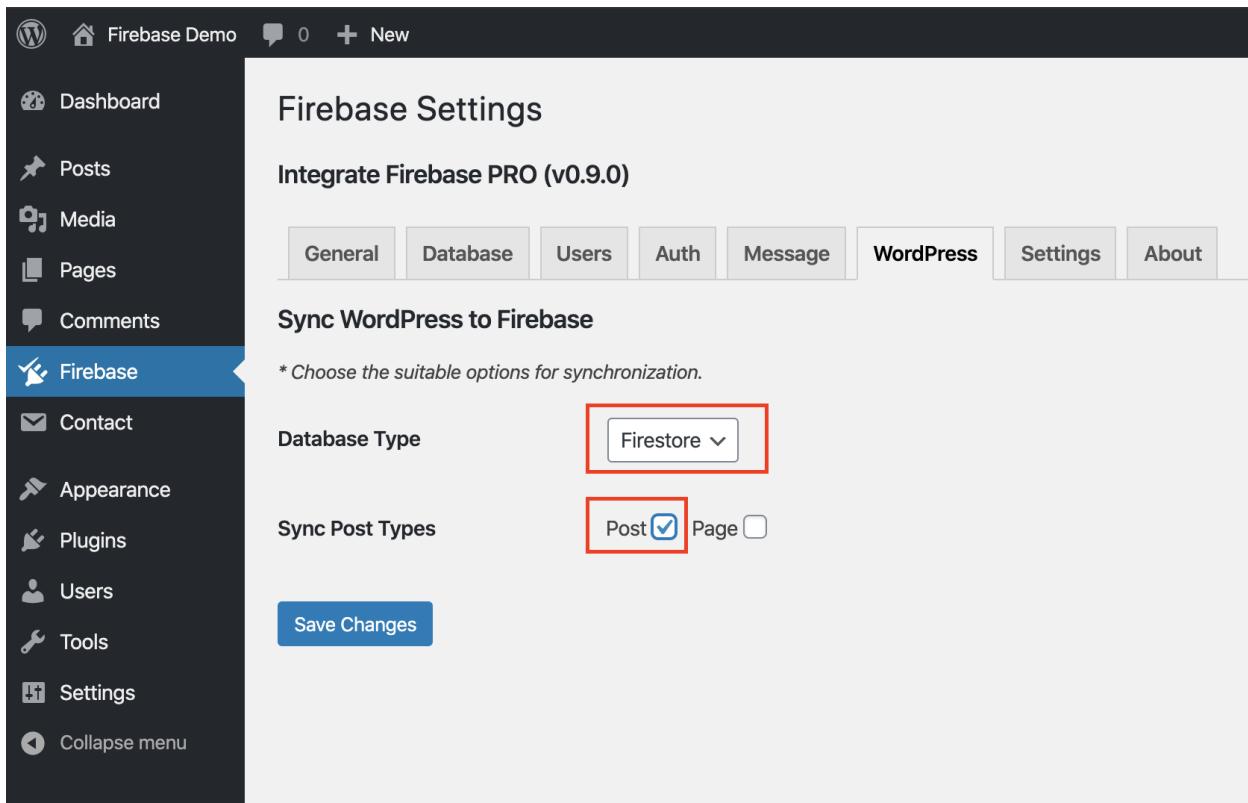


Fig. 1: Synchronization configuration

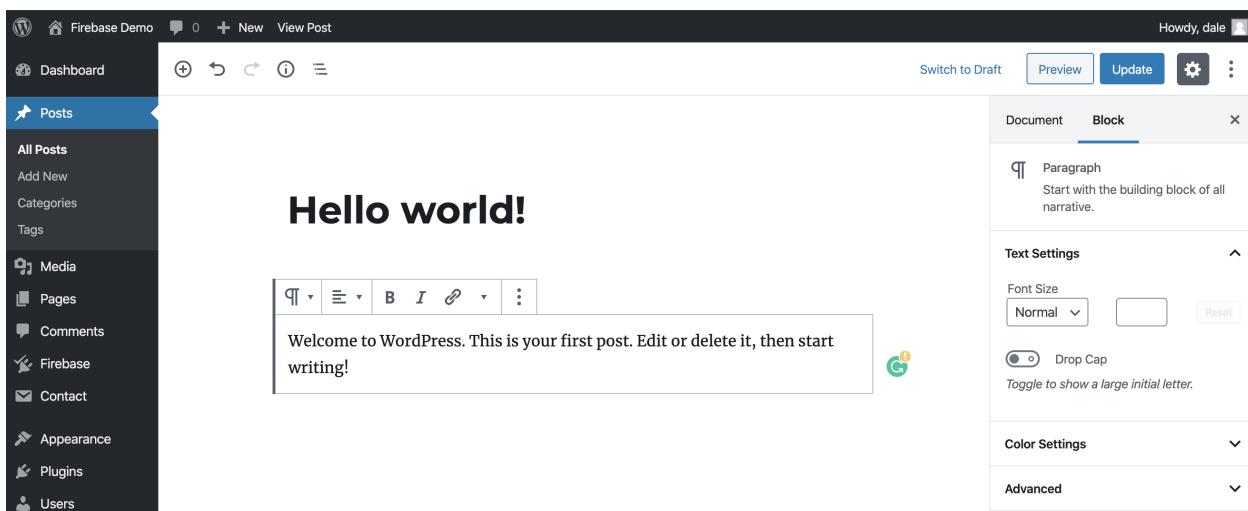


Fig. 2: Create a new post

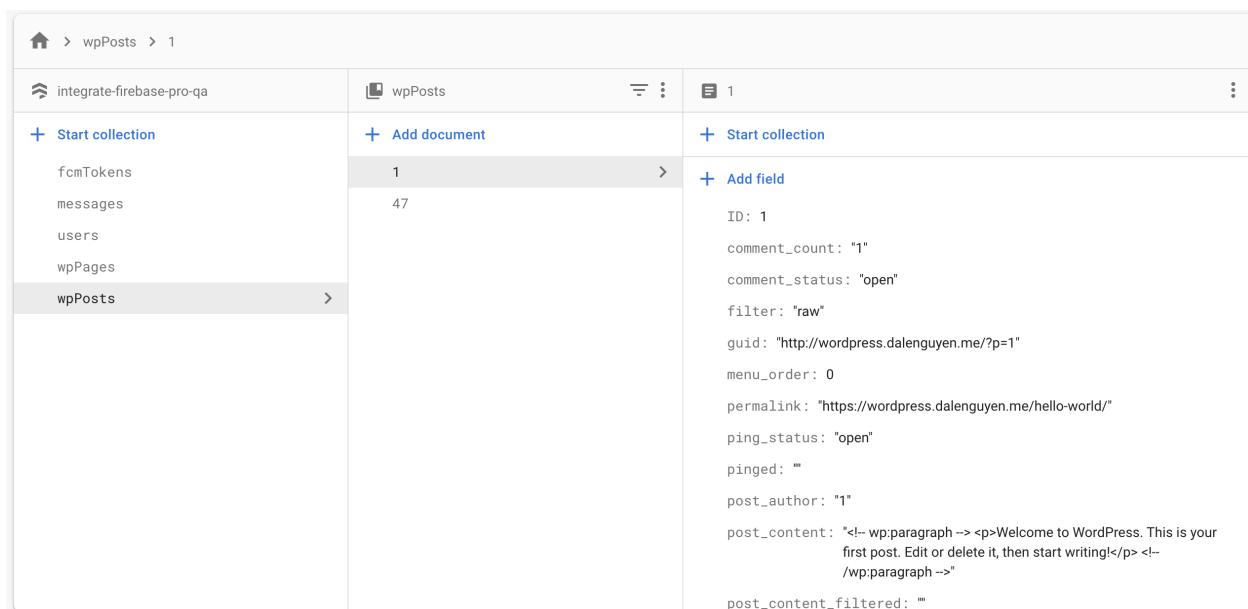


Fig. 3: New post in firestore



# CHAPTER 8

## Firebase Cloud Message Integration

This feature only helps you to send notification or data to a topic. In order to improve this feature, you can participate in [this 3 minutes survey](#).

In order to use this feature, you need to update the plugin's cloud functions to at least v0.6.0 which comes together with the Integrate Firebase PRO plugin.

After that, you can go to WordPress dashboard > **Settings > Firebase > Message**, prepare your message and send it.

There is a sample notification in the Message tab, that will send a notification to "topic-name" topic. You will have to modify it basing on your needs.

This is the structure of the message. You can send a notification or data to a specific topic. Below are samples of the message. If you want to learn more on the message structure, please check the [firebase send message documentation](#).

You also can send notification together with data.

```
// Send notification
{
    "notification": {
        "title": "Background Message Title",
        "body": "Background message body"
    },
    "topic": "topic-name"
}
```

```
// Send data
{
    "data": {
        "score": "850",
        "time": "2:45"
    },
    "topic": "topic-name"
}
```

If the message is sent successfully, you will see a message.

## Firebase Settings

### Integrate Firebase PRO to WordPress (v0.6.0)

General Database Users Auth Message Settings About

Message only support send to topic or condition at the moment. If you want to improve this feature Please read the [document](#) for more example of topic and condition

Please modify the data basing on your need.

```
{  
  "notification": {  
    "title": "Background Message Title",  
    "body": "Background message body"  
  },  
  "topic": "topic-name"  
}
```

**Send Message**

Fig. 1: Sample notification message

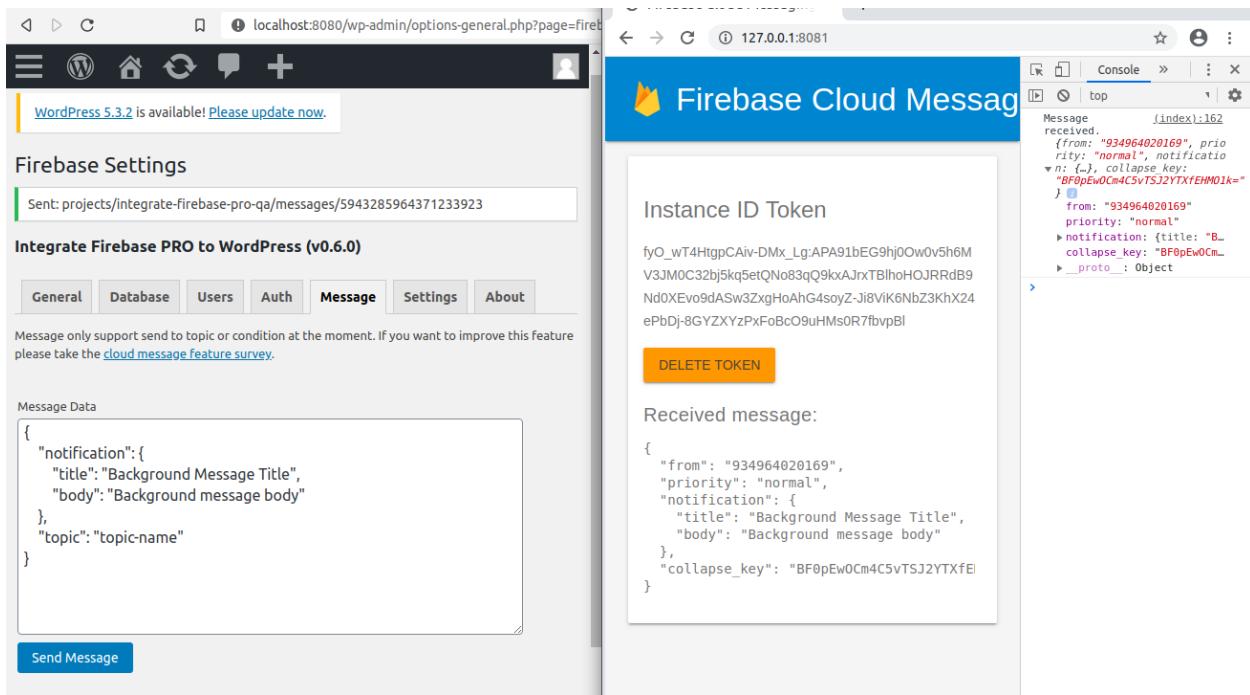


Fig. 2: Successfully sent a message

If you send a notification to a topic, and website is in the background, this is what user will see.

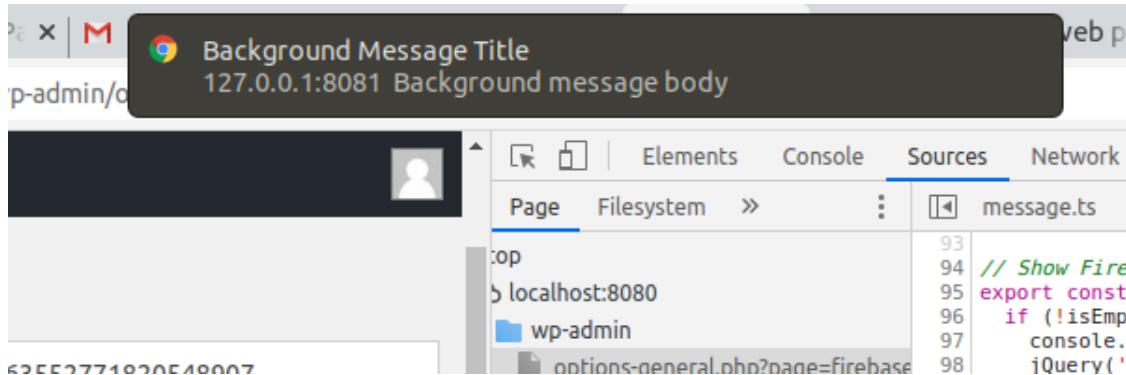


Fig. 3: Background notification



# CHAPTER 9

## How to Save Data from WordPress to Firebase (Realtime + Firestore)

In this guide, I will show you how to add new data to Realtime database Firestore. In this example, I use [Contact Form 7](#), but you feel free to design your own form or use other plugin.

The reason that I use Contact Form 7 is because it's the most popular form in WordPress plugin. It is well supported and highly customisation. You don't have to worry about form validation or email handling after submitting a form.

If you decide to create your own custom form, remember to add "if-create-data-form" as an ID to your form, together with the hidden fields below.

```
<form action="/?p=12#wpcf7-f11-p12-o1" method="post" class="wpcf7-form demo sent" novalidate="novalidate" id="if-create-data-form" _lpchecked="1">
  <div style="display: none;">...</div>
  <input type="hidden" name="collectionName" value="users" class="wpcf7-form-control wpcf7-hidden">
  <input type="hidden" name="databaseType" value="realtime" class="wpcf7-form-control wpcf7-hidden">
  <input type="hidden" name="arrayType" value="hobbies,food" class="wpcf7-form-control wpcf7-hidden">
```

Fig. 1: Custom Create Form

### 9.1 Prerequisite

Before saving data to the Firebase, we need to add the public collections to the Firebase environment. Without this step, your data won't be saved.

In order to add the public collections, you can open the terminal and run this

```
firebase functions:config:set public.collections="users" --project project-id
```

**NOTE:** This will allow WordPress public frontend to write data to **users** collection. Remember that it's case sensitive. It should only be used for public collection such as **Contact**. Please don't use this approach for private or secured collection. If you don't want public to write data to this collection. You can set the post as private and add new data after you log in to WordPress.

## 9.2 Example of creating new form and writing data to Firestore

Sample of data that I will use to update to Firestore

```
export interface Contact {  
    firstName: string  
    lastName: string  
    email: string  
    phone: string  
    age: string  
    dateOfBirth: string  
    hobbies: array<string>  
    food: array<string>  
    gender: string  
}
```

From that you can create a sample form in Contact tab. The hidden fields are important.

- [hidden collectionName “users”] -> collection name is users.
- [hidden documentId “1”] -> it will override the document id 1. This field is optional.
- [hidden databaseType “firebase”] -> data will be saved in firestore.
- [hidden arrayType “hobbies,food”] -> array data should be added to arrayType field. This field is optional.
- [hidden dateType “dateOfBirth”] -> will save as ISO string type. This field is optional.

```
[hidden collectionName "users"]  
[hidden documentId "1"]  
[hidden databaseType "firebase"]  
[hidden arrayType "hobbies, food"]  
[hidden dateType "dateOfBirth"]  
  
[text* firstName placeholder "First Name"]  
[text* lastName placeholder "Last Name"]  
[text* email placeholder "Email"]  
[tel phone placeholder "+1 647 620 0000"]  
  
<label for="age">Age</label>  
[number age id:age min:1 max:100]  
  
<label for="dateOfBirth">Date of Birth</label>  
[date* dateOfBirth id:dateOfBirth]  
  
<label for="hobbies">Hobbies</label>  
[select* hobbies id:hobbies multiple "Archery" "Slap Dance" "Rock Climbing"]  
  
<label for="food">Food</label>  
[checkbox food id:food "Pho" "Ramen" "Dimsum"]  
  
<label for="gender">Gender</label>  
[radio gender id:gender default:1 "Male" "Female" "Other"]  
  
[submit id:if-data-submit "Submit"]
```

The shortcode will be added to the WordPress page or post. With the id: “if-create-data-form”. The id is important in order for the plugin to save the data to firebase.

```
[contact-form-7 id="11" html_id="if-create-data-form" title="Contact form 1"]
```

After submitting, data will be saved to Firestore

## 9.3 Example of creating new form and writing data to Realtime

If you want to save data to Realtime database, the only thing that you need to change is the databaseType hidden field.

```
[hidden databaseType "realtime"]
```

If there form is valid, the data will be saved to realtime database

## 9.4 Addition Settings

Contact form 7 comes with extra configurations. If you don't want to send a confirmation email to customers, you can this line to the addition settings.

```
skip_mail: on
```

## 9.5 Reference

<https://contactform7.com/additional-settings/> <https://contactform7.com/hidden-field/>

Dale



Nguyen

dale@dalenguyen.me

+1 234 567 8910

Age

21

Date of Birth

14/05/1980

Hobbies

- Archery
- Slap Dance
- Rock Climbing

Food

- Pho    Ramen    Dimsum

Gender

- Male    Female    Other

The screenshot shows the Firebase Firestore interface. On the left, there's a sidebar with a '+ Start collection' button and a 'users' collection listed. The main area shows a '+ Add document' button and a list of document IDs: Xtc0qvdztcDwHoWvJ2Cs, hsVgqbQgfdlnEpaB1uwu, kDbfDIYKnUzzSUQfy0zN2Hf6WF11, and o9qHiFliNmW1sxbAcmWK. The document 'o9qHiFliNmW1sxbAcmWK' is selected. Its fields are displayed on the right: dateOfBirth: "1980-10-10", email: "dale@dalenguyen.me", firstName: "Dale", food: ["Pho", "Ramen"], gender: "Male", hobbies: ["Archery", "Rock Climbing"], lastName: "Nguyen", and phone: "+1 234 567 8910".

Fig. 3: Data saved to firestore

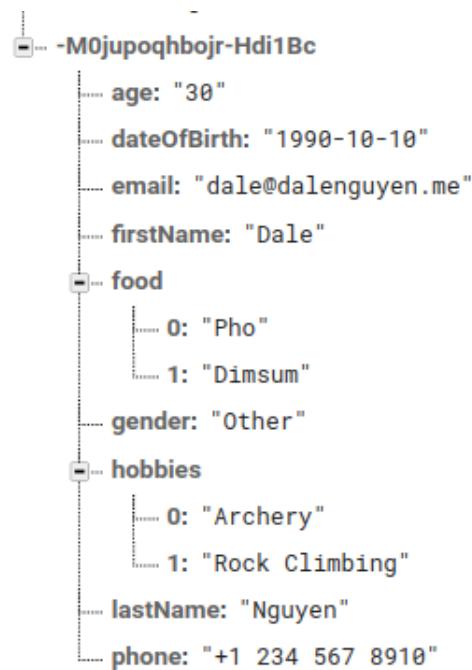


Fig. 4: Data saved to realtime



# CHAPTER 10

---

## How to Retrieve Data from Firestore and Display on WordPress

---

In this tutorial, I will show you how to retrieve the database from your Firestore and display it on the WordPress website. Before that, make sure you have:

- Install and active Integrate Firebase plugin
- Update the firebase configuration on the Firebase dashboard
- Make sure that Firebase security rules allow you to access the document with or without logging in

The plugin support getting one document and display on the frontend as a table with the shortcode.

```
// Get document id '1' from 'users' collection from realtime database  
[realtime class='your-class-name' collection_name='users' document_name='1']  
  
// Get document id '1' from 'users' collection from firestore  
[firestore class='your-class-name' collection_name='users' document_name='1']
```

If the default shortcode doesn't suite your needs, you can create a custom one. Please follow the guide below.

### 10.1 Step 1: Create a custom shortcode

You can do this by edit functions.php in your current theme

```
// functions.php  
  
// custom firebase short code  
function custom_firebase_func($atts)  
{  
    return "<div id='custom-firebase'>Test</div>";  
}  
add_shortcode('custom_firebase', 'custom_firebase_func');
```

Add the shortcode on a page or post

displayName	Dale Nguyen
email	dungnq@itbox4vn.com
photoURL	<a href="https://lh5.googleusercontent.com/-z_469Er-kgo/AAAAAAAAAAI/AAAAAAAACc/RUPjZINaMpQ/photo.jpg">https://lh5.googleusercontent.com/-z_469Er-kgo/AAAAAAAAAAI/AAAAAAAACc/RUPjZINaMpQ/photo.jpg</a>
role	guest,admin
uid	kDbfDIYKnUZzSUQfy0zN2Hf6WFl1

Fig. 1: Display data from firebase

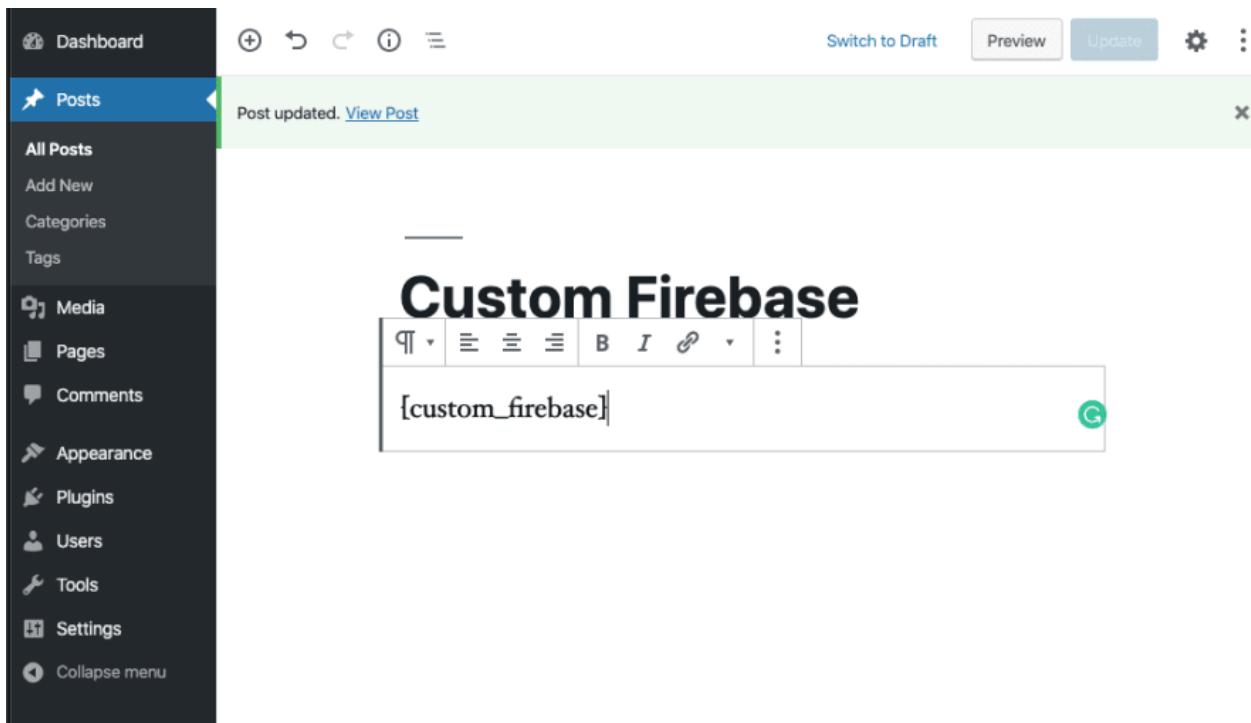


Fig. 2: Custom Firebase Shortcode

## Custom Firebase

admin May 25, 2019

1 Comment Edit

div#custom-firebase 314x39

Test

admin May 25, 2019

Uncategorized Edit

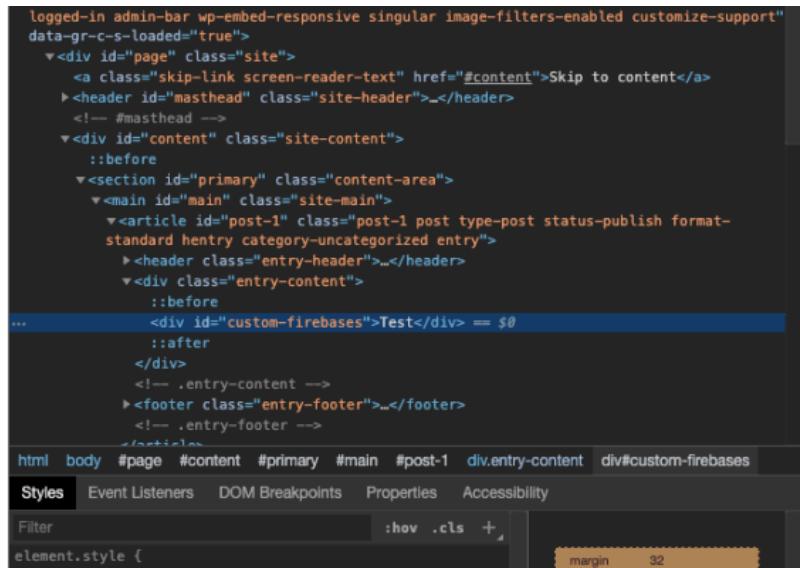


Fig. 3: Custom Firebase Shortcode Frontend

### 10.2 Step 2: Add custom javascript file

Again, you can do this by editing your functions.php

```
// functions.php

// Custom JavaScript for Firebase
function custom_firebase_scripts_function()
{
wp_enqueue_script('custom_firebase', get_template_directory_uri() . '/js/custom-
firebase.js', array('firebase_app', 'firebase_auth', 'firebase_database', 'firebase_
firestore', 'firebase'));
}
add_action('wp_enqueue_scripts', 'custom_firebase_scripts_function');
```

Your custom-firebase.js will be under js/ folder

Verify it on the front-end. You have access to firebase now.

Custom JS shows

### 10.3 Step 3: Retrieve and display data from Firestore

Now, it's all about JavaScript. You can customize, modify and do whatever you want.

```
(function ($) {
    'use strict';
    $(document).ready(function () {
        const showFirestoreDatabase = () => {
            const db = firebase.firestore();
            const firestoreEl = jQuery('#custom-firebase');
```

(continues on next page)

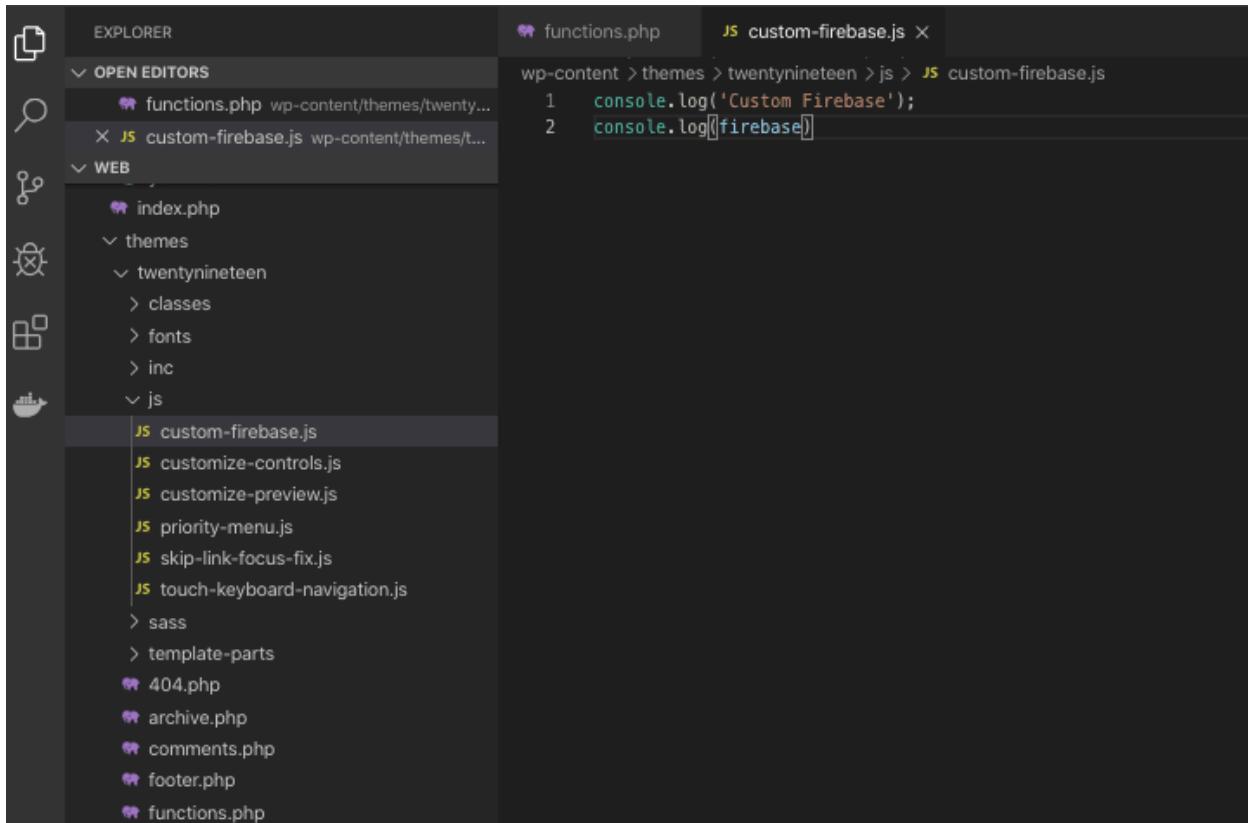


Fig. 4: Custom Firebase Location

The screenshot shows the browser's developer tools with the 'Console' tab selected. The output window displays the logs from the 'custom-firebase.js' script. It includes the following messages:

- 'Custom Firebase'
- 'custom-firebase.js?ver=5.2.4:1 custom-firebase.js?ver=5.2.4:2'
- Object with properties: INTERNAL, User, app, auth, database, default, firestore, initializeApp, \_esModule, \_SDK\_VERSION.
- JQMIGRATE: Migrate is installed, version 1.4.1 jquery-migrate.min.js?ver=1.4.1:2
- DomDistiller debug level: 0 VM1057:140
- ⚠ Please check your collection and document name in the [realtime] realtime.ts:28
- ⚠ Please check your collection and document name in the [firestore] firestore.ts:32
- ⚠ [Deprecation] Element.createShadowRoot is deprecated and will be removed in M80, around February 2020. Please use Element.attachShadow instead. See https://www.chromestatus.com/features/4507242028072960 and https://developers.google.com/web/updates/2019/07/web-components-time-to-upgrade for more details. content.min.js:2

Fig. 5: Verify custom firebase on frontend

(continued from previous page)

```

// You can get the collectionName and documentName from the shortcode ↵
attribute
    const collectionName = 'users';
    const documentName = 'document-1'

    if (collectionName && documentName) {
        const docRef = db.collection(collectionName).doc(documentName);

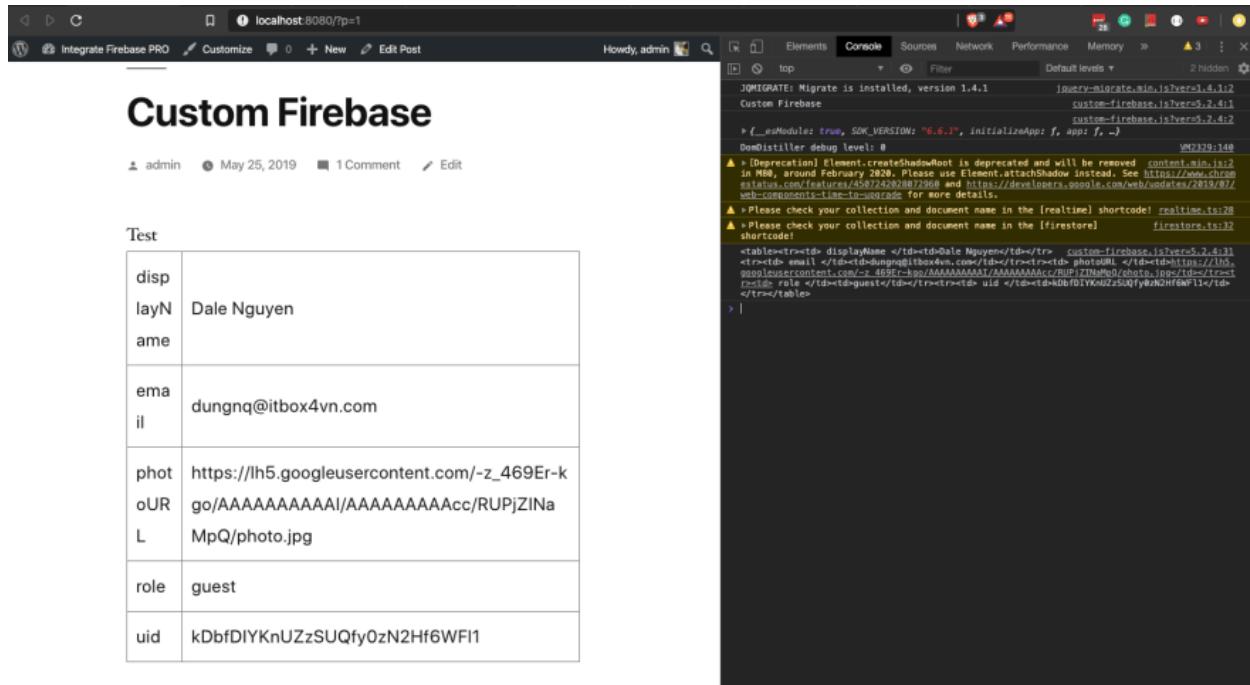
        docRef.get().then(doc => {
            if (doc.exists) {
                // console.log('Document data:', doc.data());
                let html = '<table>';
                jQuery.each(doc.data(), function (key, value) {
                    // You can put condition to filter your value
                    // and it won't show on the frontend
                    html += '<tr>';
                    html += `<td> ${String(key)} </td>`;
                    html += `<td> ${value} </td>`;
                    html += '</tr>';
                })
                html += '</table>';
                firestoreEl.append(html)
            } else {
                // doc.data() will be undefined in this case
                console.error('Please check your collection and document name ↵
in the [firestore] shortcode!');
            }
        }).catch(error => {
            console.error('Please check your collection and document name in ↵
the [firestore] shortcode!', error);
        });
    } else {
        console.warn('Please check your collection and document name in the ↵
[firestore] shortcode!');
    }
}

showFirestoreDatabase()
})
}) (jQuery)

```

Check the code on the WordPress post.

Yay, firestore data is retrieved



The screenshot shows a browser window with the URL `localhost:8080/?p=1`. The page title is "Custom Firebase". The content area displays a table titled "Test" with the following data:

	Test
displayN ame	Dale Nguyen
ema il	dungnq@itbox4vn.com
phot oUR L	<a href="https://lh5.googleusercontent.com/-z_469Erkgo/AAAAAAAAl/AAAAAAAACc/RUPjZInaMpQ/photo.jpg">https://lh5.googleusercontent.com/-z_469Erkgo/AAAAAAAAl/AAAAAAAACc/RUPjZInaMpQ/photo.jpg</a>
role	guest
uid	kDbfDIYKnUZzSUQfy0zN2Hf6WFl

The browser's developer tools are open, showing the JavaScript console output. The output includes:

```

JQMIGRATE: Migrate is installed, version 1.4.1
Custom Firebase
  <!-- esModule: true, SDK_VERSION: "6.6.1", initializeApp: f, app: f, -->
DomDistiller debug level: 0
VM329:148
⚠ [Deprecation] Element.createShadowRoot is deprecated and will be removed in M68, around February 2020. Please use Element.attachShadow instead. See https://www.chromestatus.com/features/458724290072368 and https://developers.google.com/web/updates/2019/07/web-components-lace-up-shadow for more details.
⚠ Please check your collection and document name in the [realtime] shortcode! realtime.ts:28
⚠ Please check your collection and document name in the [firestore] shortcode! firestore.ts:32
> |
<table><tr><td> displayName </td><td> Dale Nguyen </td></tr>
<tr><td> email </td><td> dungnq@itbox4vn.com </td></tr><tr><td> photoURL </td><td> https://lh5.googleusercontent.com/-z_469Erkgo/AAAAAAAAl/AAAAAAAACc/RUPjZInaMpQ/photo.jpg </td></tr>
<tr><td> role </td><td> guest </td></tr><tr><td> uid </td><td> kDbfDIYKnUZzSUQfy0zN2Hf6WFl </td>
</tr></table>
  
```

Fig. 6: Firestore data retrieved

# CHAPTER 11

---

## How to Work With Firebase Custom Claims in WordPress

---

We already knew how to get and display data to WordPress. What if you want to display data to the user who is admin (through custom claims of course), and display something else to the others or just show nothing.

### 11.1 Using shortcode to display custom data

Since version 0.5.6, I added a shortcode that helps to display data if users log in and have the right custom claims.

```
[firebase_show_with_claims
    class='your-class-name'
    claims='admin'
    message='Custom message'
]
    HTML Data With Tags
[/firebase_show_with_claims]
```

- `class='your-class-name'`: custom class that you can add style to the element
- `claims='admin'`: user's claims
- `message='Custom message'`: this message will show to invalid user

### 11.2 Using custom code to display custom data

This is the original code to retrieve and display data.

```
(function ($) {
    'use strict';
    $(document).ready(function () {
        const showFirestoreDatabase = () => {
            const db = firebase.firestore();
```

(continues on next page)

(continued from previous page)

```

        const firestoreEl = jQuery('#custom-firebase');

        // You can get the collectionName and documentName from the shortcode ↵
attribute
        const collectionName = 'users';
        const documentName = 'document-1'

        if (collectionName && documentName) {
            const docRef = db.collection(collectionName).doc(documentName);

            docRef.get().then(doc => {
                if (doc.exists) {
                    // console.log('Document data:', doc.data());
                    let html = '<table>';
                    jQuery.each(doc.data(), function (key, value) {
                        // You can put condition to filter your value
                        // and it won't show on the frontend
                        html += '<tr>';
                        html += `<td> ${String(key)} </td>`;
                        html += `<td> ${value} </td>`;
                        html += '</tr>';
                    })
                    html += '</table>';
                    firestoreEl.append(html)
                } else {
                    // doc.data() will be undefined in this case
                    console.error('Please check your collection and document name ↵
in the [custom_firebase] shortcode!');
                }
            }).catch(error => {
                console.error('Please check your collection and document name in ↵
the [custom_firebase] shortcode!', error);
            });
        } else {
            console.warn('Please check your collection and document name in the ↵
[custom_firebase] shortcode!');
        }
    }

    showFirestoreDatabase()
})
}) (jQuery)

```

The **showFirestoreDatabase()** function will display data on WordPress. Now, we will wrap it under custom claims check.

```

(function ($) {
    'use strict';
    $(document).ready(function () {
        const showFirestoreDatabase = () => {
            ...
        }
        // We won't call the function directly here
        // showFirestoreDatabase()

        const getUserCustomClaims = () => {

```

(continues on next page)

(continued from previous page)

```
firebase.auth().onAuthStateChanged(function (user) {
    if (firebase.auth().currentUser) {
        firebase.auth().currentUser.getIdTokenResult()
            .then((idTokenResult) => {
                // Confirm the user is an Admin.
                if (!!idTokenResult.claims.admin) {
                    // We will call the function here
                    showFirestoreDatabase()
                } else {
                    // Show something else
                }
            })
            .catch((error) => {
                console.log(error);
            });
    }
}

getUserCustomClaims()
})
}) (jQuery)
```

After checking the user's custom claims, and make sure that it's admin. Then we will call showFirestoreDatabase() function.

This tutorial doesn't limit to retrieve data from Firestore, you can put any functions or features after checking the custom claims.



# CHAPTER 12

---

## Use Cases for Integrate Firebase PRO

---

This part contains a list of use cases that can be achieved with the plugin:

[WordPress CMS for Mobile Apps \(Ionic + Firebase\)](#)



# CHAPTER 13

---

## Troubleshooting

---

If you have any errors or questions during the setting up, you can open the inspect on the browser, please go to the console log tab, then take a snapshot and send an email to me at: [dale@dalenguyen.me](mailto:dale@dalenguyen.me).

Before doing that, you can take a look at the playlist on [How to Integrate Firebase to WordPress](#).



# CHAPTER 14

---

## Roadmap

---

Here is the of the feature that I will add for the next release.

- Realtime & Firestore CRUD on Dashboard
- Improve Firebase Cloud Message (FCM) integration
- Integrate Storage management on Dashboard
- WooCommerce integration
- Import WordPress posts, pages... to Firebase

If you have a feature request, please create a [ticket](#).



# CHAPTER 15

---

## CHANGELOG

---

All notable changes to this project will be documented in this file.

### **15.1 ## [ 0.11.1 ] - 07-04-2020**

- Showed warning if base domain is not set
- Check for undefined in order to pass error check
- Updated options for plugin deletion

### **15.2 ## [ 0.11.0 ] - 02-04-2020**

- Used wait for element rather than setTimeOut
- Added logout event to all logout links
- Added post thumbnail and author name to Firebase Sync
- Updated Firebase script from 7.9.3 to 7.13.1

### **15.3 ## [ 0.10.0 ] - 01-04-2020**

- Added date type for saving data to Firebase
- Increase time wait for error in form submit to Firebase

Dependency: cloud functions: v0.9.0

## **15.4 ## [ 0.9.1 ] - 29-03-2020**

- Fixed ArrayType when saving data to Realtime/Firestore
- Fixed WP post type is null when sync data to Firebase
- Removed notice warning for post types

Dependency: cloud functions: v0.8.0

## **15.5 ## [ 0.9.0 ] - 28-03-2020**

- Fixed save data to realtime / firestore token error
- Added document id option when saving data
- Added trigger for syncing post and page to Firebase

Dependency: cloud functions: v0.8.0

## **15.6 ## [ 0.8.0 ] - 24-03-2020**

- Logout of everything when clicking signout buttons
- Added warning before deleting a Firebase user
- Added user role (Customer) for WooCommerce sites
- Prevent user to change password when login through firebase is active
- User password will be dominated by Firebase procedure

Dependency: cloud functions: v0.7.0

## **15.7 ## [ 0.7.0 ] - 13-03-2020**

- Styled add new user button
- Created and log in Firebase Users to WordPress
- Redirect login page feature
- Added Rest API for creating new Users (Subscriber)
- Updated FirebaseUI Web to 4.5.0
- Bring Firebase Menu to the front
- Prevent normal user to see dashboard token when they log in
- Updated about page
- Show realtime & firestore data based on security rules

## 15.8 ## [ 0.6.0 ] - 01-03-2020

- Update firebase scripts from 7.8.2 to 7.9.3
- Added send cloud message to a topic feature

## 15.9 ## [ 0.5.8 ] - 20-02-2020

- Breaking change for getting database: you need to update wordpress firebase functions to 0.5.8.
- Added create data for Realtime database & firestore with Contact Form 7
- Added warning for missing [firebaseui\_web] globally
- Moved environment variables to one source

## 15.10 ## [ 0.5.7 ] - 16-02-2020

- Updated firebase scripts to v7.8.2
- Hide greetings when signing out

## 15.11 ## [ 0.5.6 ] - 21-12-2019

- Display data with claims

## 15.12 ## [ 0.5.5 ] - 21-12-2019

- Breaking changes
- Deprecated authention process and replaced with firebasui-web

## 15.13 ## [ 0.5.4 ] - 01-12-2019

- Updated packages
- Moved error and message to the top of dashboard
- Add CRUD to manage Firebase User from Dashboard

## 15.14 ## [ 0.5.3 ] - 22-09-2019

- Added user register form to frontend #4
- Show firestore database after login #10
- Added delete user from dashboard #11
- Search document from firestore or realtime

- Update firebase version

## **15.15 ## [ 0.5.2 ] - 30-03-2019**

- Show realtime database after login

## **15.16 ## [ 0.5.1 ] - 11-08-2018**

- Hide login form after logging in

## **15.17 ## [ 0.5.0 ] - 04-08-2018**

- Add shortcode to display when not login
- Add error handling shortcode

## **15.18 ## [ 0.4.0 ] - 17-07-2018**

- Added Firestore database support in Dashboard

## **15.19 ## [ 0.3.2 ] - 17-07-2018**

- Fixed firebase-show shortcode

## **15.20 ## [ 0.3.1 ] - 17-07-2018**

- Fixed getting credentials

## **15.21 ## [ 0.3.0 ] - 02-07-2018**

- Added about information
- Added Real Time database support in Dashboard

## **15.22 ## [ 0.2.0 ] - 25-5-2018**

- Added firebase scripts and styles to header
- Implement login and logout features

## 15.23 ## [ 0.1.0 ] - 20-4-2018

- Started the project and add an authentication method