**Laravel 9 Send Web Push Notification with Firebase Tutorial**

Last updated on: September 11, 2022 by Digamber

14.5M

438

Tencent, Sony Buy Stake in ‘Elden Ring’ Developer FromSoftware



Laravel Firebase push notification tutorial; Throughout this comprehensive Laravel Firebase web push notification example, we will explain how to send web push notifications in the Laravel app using Firebase Cloud Messaging mechanism.

A push notification is a message that appears on a mobile device and a web app. Push notifications provide a great way to communicate with customers or users regarding any context.

In this tutorial, we will learn how to easily send web push notifications from the Laravel app using Firebase and gradually elevate the customer experience and add value to the product’s user experience.

Firebase is an awesome BaaS that refers to the backend as a service; it comes with powerful tools to build modern web and mobile applications. It makes the development process easy through its vast array of user-centric features.

To send Firebase web push notifications, we need to get the Firebase SDK or configuration key through which we can manage the app’s data on a centralized cloud platform.

Firebase cloud messaging offers a free service to send a notification across any device, and Firebase Cloud Messaging is also known as FCM. It gives a reliable and robust battery-efficient connection among your server and devices that empowers you to deliver and receive messages and notifications on iOS, Android, and the web.

To create a push notification, we need to acquire the server key or token from the firebase dashboard and add this token into the laravel app to send web push notifications.

So, let us go through one by one every instruction mentioned below.

**How to Send Web Push Notifications in Laravel 9 using Firebase**

* **Step 1:** Create Laravel Project
* **Step 2:** Database Connection
* **Step 3:** Install Laravel UI Scaffolding
* **Step 4:** Update Server Key Prop in User Table
* **Step 5:** Get Firebase Cloud Messaging Server Key
* **Step 6:** Create Controller
* **Step 7:** Create Routes
* **Step 8:** Set Up Blade View
* **Step 9:** Create Firebase (FCM) File
* **Step 10:** Run Laravel App

**Create Laravel Project**

Let us start creating a fresh new laravel app using the composer command, so make sure to configure composer on your system.

composer create-project --prefer-dist laravel/laravel laravel\_web\_push\_notification

Bash

Copy

Get into the app’s root:

cd laravel\_web\_push\_notification

Bash

Copy

**Database Connection**

Secondly, add database name, username and password in the **.env** file to make the database connection:

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=database\_name

DB\_USERNAME=database\_user\_name

DB\_PASSWORD=database\_password

Bash

Copy

If you are using MAMP local server in macOs; make sure to append UNIX\_SOCKET and DB\_SOCKET below database credentials in **.env**file.

UNIX\_SOCKET=/Applications/MAMP/tmp/mysql/mysql.sock

DB\_SOCKET=/Applications/MAMP/tmp/mysql/mysql.sock

Bash

Copy

**Install Laravel UI Scaffolding**

Next, we have to install the laravel ui package and create auth scaffolding based on Bootstrap CSS framework, hence execute the given below commands.

composer require laravel/ui

Bash

Copy

After that, you need to execute the following command to manifest bootstrap auth ui.

php artisan ui bootstrap --auth

Bash

Copy

Afterward, you have to run suggested commands to compile your fresh scaffolding.

npm install && npm run dev

Bash

Copy

**Update Server Key Prop in User Table**

Subsequently, we need to add the new column property in the current User table inside the database, execute the below command.

php artisan make:migration add\_column\_device\_key

Bash

Copy

After that, you have to move to **database/migrations/xxxx\_add\_column\_device\_key.php**  
file.

**<?php**

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class AddColumnDeviceKey extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

Schema::table('users', function (Blueprint $table) {

$table->string('device\_key')->nullable();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

//

}

}

PHP

Copy

After that, move to the **app/Models/User.php** file and add the device key property.

**<?php**

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

class User extends Authenticatable

{

use HasFactory, Notifiable;

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'name',

'email',

'password',

'device\_key',

];

/\*\*

\* The attributes that should be hidden for arrays.

\*

\* @var array

\*/

protected $hidden = [

'password',

'remember\_token',

];

/\*\*

\* The attributes that should be cast to native types.

\*

\* @var array

\*/

protected $casts = [

'email\_verified\_at' => 'datetime',

];

}

PHP

Copy

Head over to the terminal, type the following command, and execute it to run the migration.

php artisan migrate

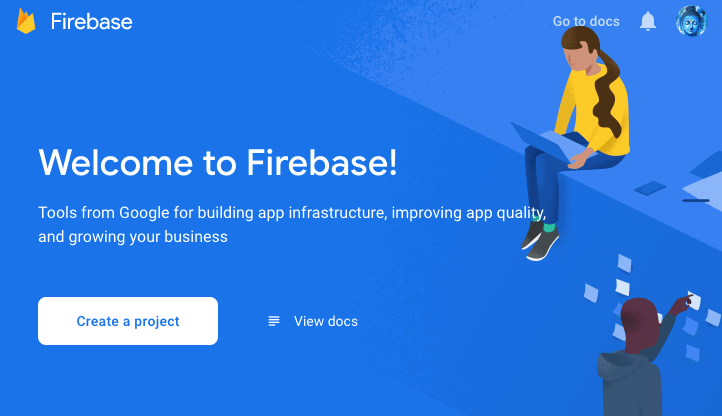
Bash

Copy

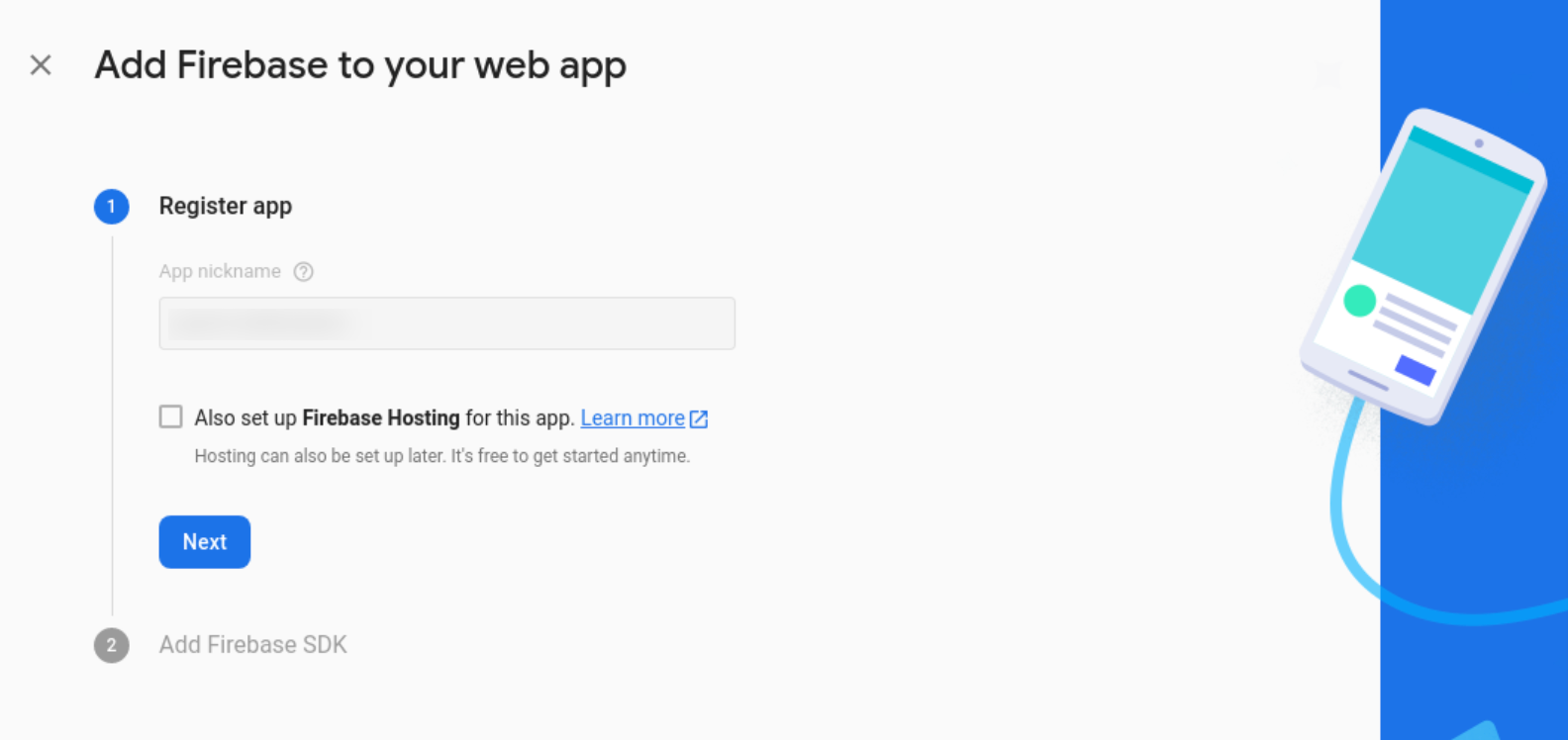
**Get Firebase Cloud Messaging (FCM) Server Key**

This step will describe how to get the Firebase cloud messaging server key and Firebase web app’s configuration credentials.

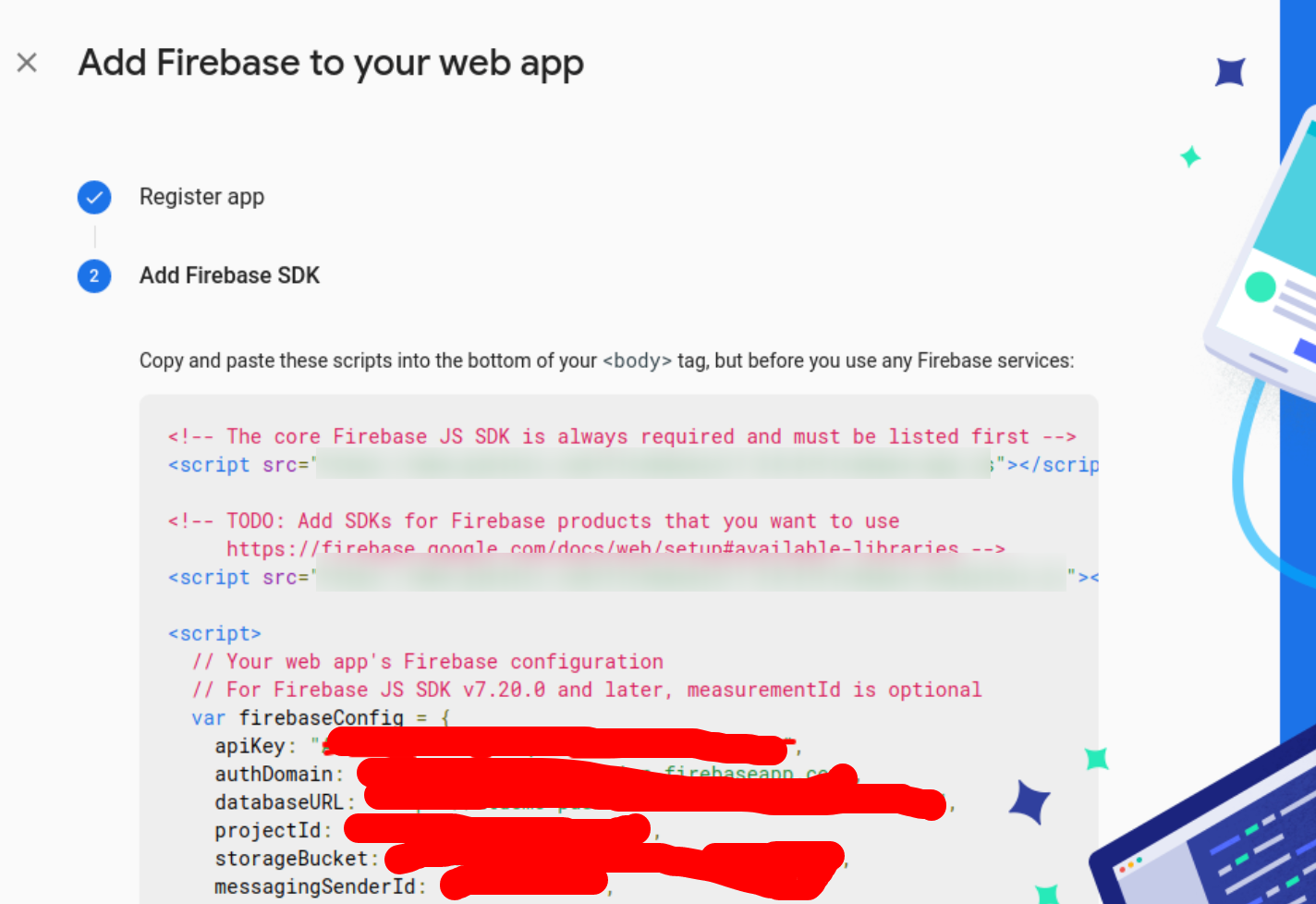
So, first go to [Firebase site](https://console.firebase.google.com/) and here you need to create a Firebase project.



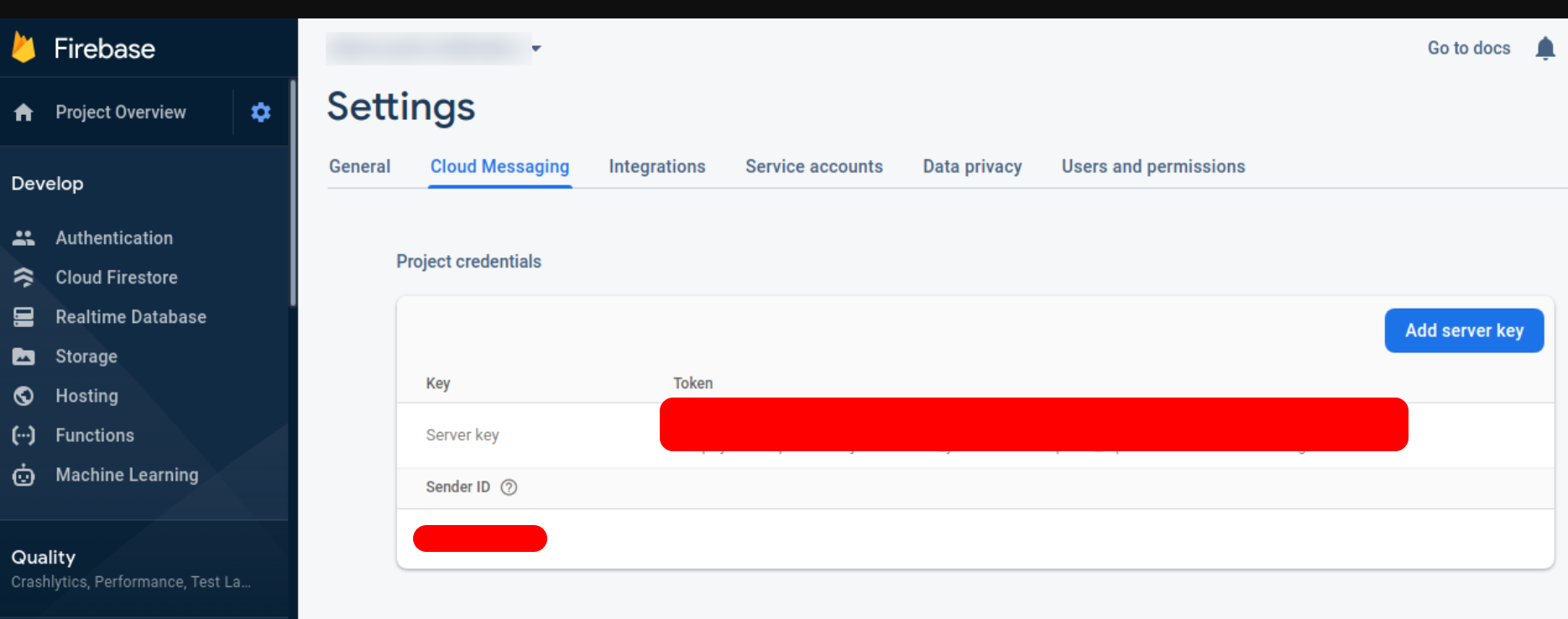
Then, add the notification app name for adding Firebase to your web app.



Copy the Firebase configuration keys, and this will help to connect laravel to Firebase.



Get into the Firebase project dashboard in the **“Settings”** page, click on the **“Cloud Messaging”** tab. You need to look for the **“Server key”**, copy this cloud messaging server key, and you have to paste this key in the $serverKey variable. That is defined in the web notification controller file’s sendWebNotification() function.



**Create Controller**

We need to create some functions in the controller file to handle web notifications, so first generate the controller.

php artisan make:controller WebNotificationController

Bash

Copy

Open and place the suggested code inside the **app\Http\Controllers\WebNotificationController.php** file:

**<?php**

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

class WebNotificationController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index()

{

return view('home');

}

public function storeToken(Request $request)

{

auth()->user()->update(['device\_key'=>$request->token]);

return response()->json(['Token successfully stored.']);

}

public function sendWebNotification(Request $request)

{

$url = 'https://fcm.googleapis.com/fcm/send';

$FcmToken = User::whereNotNull('device\_key')->pluck('device\_key')->all();

$serverKey = 'server key goes here';

$data = [

"registration\_ids" => $FcmToken,

"notification" => [

"title" => $request->title,

"body" => $request->body,

]

];

$encodedData = json\_encode($data);

$headers = [

'Authorization:key=' . $serverKey,

'Content-Type: application/json',

];

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, $url);

curl\_setopt($ch, CURLOPT\_POST, true);

curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYHOST, 0);

curl\_setopt($ch, CURLOPT\_HTTP\_VERSION, CURL\_HTTP\_VERSION\_1\_1);

// Disabling SSL Certificate support temporarly

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, false);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $encodedData);

// Execute post

$result = curl\_exec($ch);

if ($result === FALSE) {

die('Curl failed: ' . curl\_error($ch));

}

// Close connection

curl\_close($ch);

// FCM response

dd($result);

}

}

PHP

Copy

**Create Routes**

Import WebNotificationController controller, then create three routes with Get and Post methods, so open and place the code in **routes/web.php** file.

**<?php**

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\WebNotificationController;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

\*/

Auth::routes();

Route::get('/push-notificaiton', [WebNotificationController::class, 'index'])->name('push-notificaiton');

Route::post('/store-token', [WebNotificationController::class, 'storeToken'])->name('store.token');

Route::post('/send-web-notification', [WebNotificationController::class, 'sendWebNotification'])->name('send.web-notification');

PHP

Copy

**Set Up Blade View**

This step comprises setting up the home blade view file, integrate push notification code for the web app hence open **resources/views/home.blade.php** file, and replace with the following code.

@extends('layouts.app')

@section('content')

<div class="container">

<div class="row justify-content-center">

<div class="col-md-8">

<button onclick="startFCM()"

class="btn btn-danger btn-flat">Allow notification

</button>

<div class="card mt-3">

<div class="card-body">

@if (session('status'))

<div class="alert alert-success" role="alert">

{{ session('status') }}

</div>

@endif

<form action="{{ route('send.web-notification') }}" method="POST">

@csrf

<div class="form-group">

<label>Message Title</label>

<input type="text" class="form-control" name="title">

</div>

<div class="form-group">

<label>Message Body</label>

<textarea class="form-control" name="body"></textarea>

</div>

<button type="submit" class="btn btn-success btn-block">Send Notification</button>

</form>

</div>

</div>

</div>

</div>

</div>

<!-- The core Firebase JS SDK is always required and must be listed first -->

<script src="https://www.gstatic.com/firebasejs/8.3.2/firebase.js"></script>

<script>

var firebaseConfig = {

apiKey: 'api-key',

authDomain: 'project-id.firebaseapp.com',

databaseURL: 'https://project-id.firebaseio.com',

projectId: 'project-id',

storageBucket: 'project-id.appspot.com',

messagingSenderId: 'sender-id',

appId: 'app-id',

measurementId: 'G-measurement-id',

};

firebase.initializeApp(firebaseConfig);

const messaging = firebase.messaging();

function startFCM() {

messaging

.requestPermission()

.then(function () {

return messaging.getToken()

})

.then(function (response) {

$.ajaxSetup({

headers: {

'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')

}

});

$.ajax({

url: '{{ route("store.token") }}',

type: 'POST',

data: {

token: response

},

dataType: 'JSON',

success: function (response) {

alert('Token stored.');

},

error: function (error) {

alert(error);

},

});

}).catch(function (error) {

alert(error);

});

}

messaging.onMessage(function (payload) {

const title = payload.notification.title;

const options = {

body: payload.notification.body,

icon: payload.notification.icon,

};

new Notification(title, options);

});

</script>

@endsection

PHP

Copy

**Create Firebase (FCM) File**

In the final step, we have to create head over to a public folder and create a new firebase-messaging-sw.js file; this file holds the web push notification configurations.

So, after creating the file, add the given below code in the **public/firebase-messaging-sw.js** file.

// Give the service worker access to Firebase Messaging.

// Note that you can only use Firebase Messaging here. Other Firebase libraries

// are not available in the service worker.importScripts('https://www.gstatic.com/firebasejs/7.23.0/firebase-app.js');

importScripts('https://www.gstatic.com/firebasejs/8.3.2/firebase-app.js');

importScripts('https://www.gstatic.com/firebasejs/8.3.2/firebase-messaging.js');

/\*

Initialize the Firebase app in the service worker by passing in the messagingSenderId.

\*/

firebase.initializeApp({

apiKey: 'api-key',

authDomain: 'project-id.firebaseapp.com',

databaseURL: 'https://project-id.firebaseio.com',

projectId: 'project-id',

storageBucket: 'project-id.appspot.com',

messagingSenderId: 'sender-id',

appId: 'app-id',

measurementId: 'G-measurement-id',

});

// Retrieve an instance of Firebase Messaging so that it can handle background

// messages.

const messaging = firebase.messaging();

messaging.setBackgroundMessageHandler(function (payload) {

console.log("Message received.", payload);

const title = "Hello world is awesome";

const options = {

body: "Your notificaiton message .",

icon: "/firebase-logo.png",

};

return self.registration.showNotification(

title,

options,

);

});

JavaScript

Copy

**Send FCM Web Push Notification in Laravel**

Finally, we have to send the push notification in Laravel using FCM, so we need to execute the php artisan command with serve flag to start the laravel development server.

php artisan serve

Bash

Copy

Once the app is started, make sure to register a new user. After creating a new account then you have to login into the app using your email and password.

# Register new user

http://127.0.0.1:8000/register

# Sign-in to account

http://127.0.0.1:8000/login

Bash

Copy

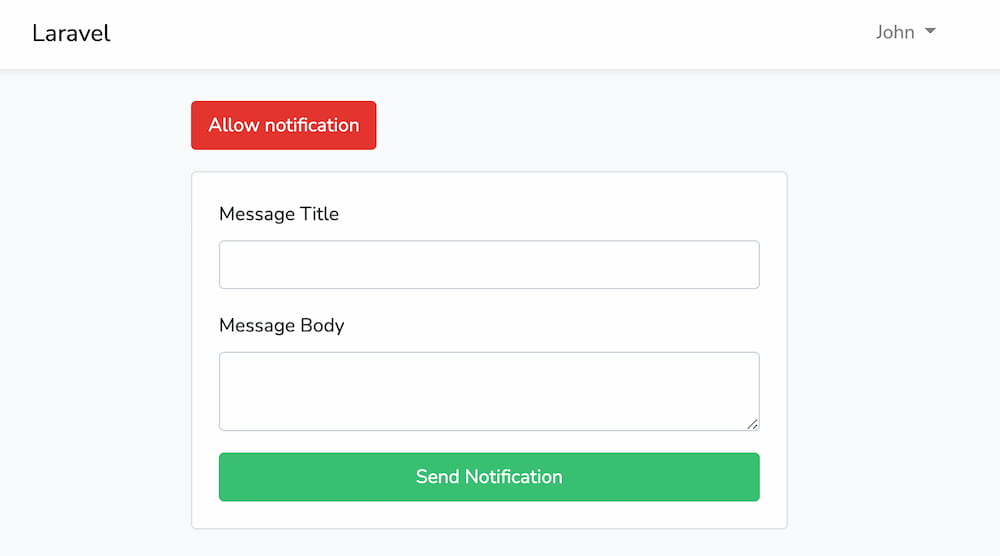
After, you signed-in, then head over to following url:

http://127.0.0.1:8000/push-notificaiton

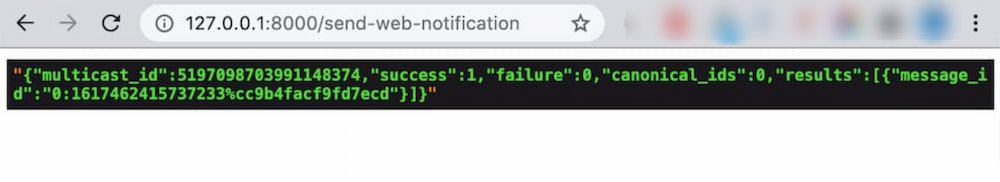
Bash

Copy

Next, you need to click on the allow notification button, it will generate the device id, also add push notification title and body in the given form.



Here is the response for web push notification sent by you on the browser:



**Conclusion**

The laravel push notification example is finished; we have described how to send web push notifications in laravel using the Firebase database. We exclusively understood how to connect Firebase cloud messaging (FCM) with Laravel to send notification.

Here is the complete code example that you can download from [GitHub](https://github.com/SinghDigamber/laravel_web_push_notification" \t "_blank).