

# CIS 371 Web Application Programming

## Cloud DataBase

### Firestore User Authentication



GRAND VALLEY  
STATE UNIVERSITY®

Lecturer: **Dr. Yong Zhuang**

# Firebase

- A collection of many products
- Cloud Firestore (beta since 2017, GA since 2019)
- Authentication
- Cloud Storage
- Realtime DB (beta since 2012, GA since 2014?)
- Cloud Messaging
- ML Kit
- Cloud Functions

# Setup

```
yarn init -y  
yarn add firebase
```

OR

```
npm init -y  
npm install firebase
```

```
import { createApp } from "vue";  
import App from "./App.vue";  
import router from "./router";  
import { initializeApp } from "firebase/app";  
  
const firebaseConfig = {  
  apiKey: "your-api-key-goes-here",  
  authDomain: "your-project-name.firebaseio.com",  
  projectId: "your-project-id",  
  storageBucket: "your-project-name.appspot.com",  
  messagingSenderId: "xxxxxxxxxxxx",  
};  
  
initializeApp(firebaseConfig);  
createApp(App).use(router).mount("#app")
```

*main.ts*

```
import {  
  getAuth,  
  signInWithPopup,  
  createUserWithEmailAndPassword,  
  signInWithEmailAndPassword,  
  sendEmailVerification,  
  signOut,  
} from "firebase/auth";  
  
const auth = getAuth();  
createUserWithEmailAndPassword(auth, "user_email", "user_password");  
sendEmailVerification(auth.currentUser);  
signInWithEmailAndPassword(auth, "user_email", "user_password");  
signInWithPopup(auth, _____);  
signOut(auth);
```

*login.vue*

# Authentication Options

- Email/Password
- Facebook accounts
- GitHub accounts
- Google accounts
- Twitter accounts
- Microsoft accounts
- Yahoo accounts
- Phone numbers
- [Online documentation: firebase.auth](#)

# Authentication Dashboard: Sign-in Providers

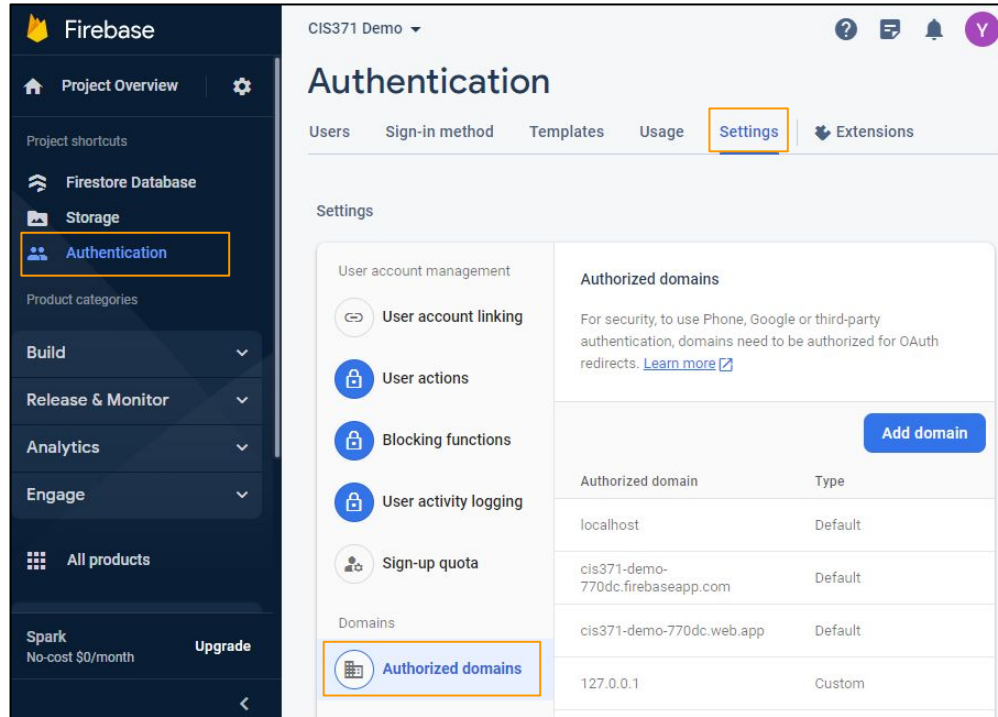
The screenshot displays the Firebase Authentication dashboard for a project named 'CIS371 Demo'. The left sidebar shows the 'Authentication' link highlighted. The main content area is titled 'Authentication' and has tabs for 'Users', 'Sign-in method', 'Templates', 'Usage', 'Settings', and 'Extensions'. The 'Sign-in method' tab is active, showing a table of 'Sign-in providers'.

Provider	Status
Email/Password	Enabled
Google	Enabled

An orange arrow points from the 'Add new provider' button to a modal window titled 'Select a sign-in provider (Step 1 of 2)'. This modal window displays three columns of providers:

- Native providers:** Email/Password (checked), Phone, Anonymous.
- Additional providers:** Google (checked), Game Center, Microsoft, Facebook, Apple, Twitter, Play Games, GitHub, Yahoo.
- Custom providers:** OpenID Connect, SAML.

# Authentication Dashboard: Authorized Domains



The screenshot shows the Firebase Authentication dashboard for a project named 'CIS371 Demo'. The left sidebar contains navigation options: Project Overview, Project shortcuts, Firestore Database, Storage, Authentication (highlighted), Product categories, Build, Release & Monitor, Analytics, Engage, All products, and Spark. The main content area is titled 'Authentication' and has tabs for Users, Sign-in method, Templates, Usage, Settings (highlighted), and Extensions. Under the Settings tab, there is a 'Settings' section with a list of options: User account management, User account linking, User actions, Blocking functions, User activity logging, Sign-up quota, Domains, and Authorized domains (highlighted). The 'Authorized domains' section shows a table of authorized domains with columns for 'Authorized domain' and 'Type'. The table lists four domains: localhost (Default), cis371-demo-770dc.firebaseio.com (Default), cis371-demo-770dc.web.app (Default), and 127.0.0.1 (Custom). An 'Add domain' button is located above the table.

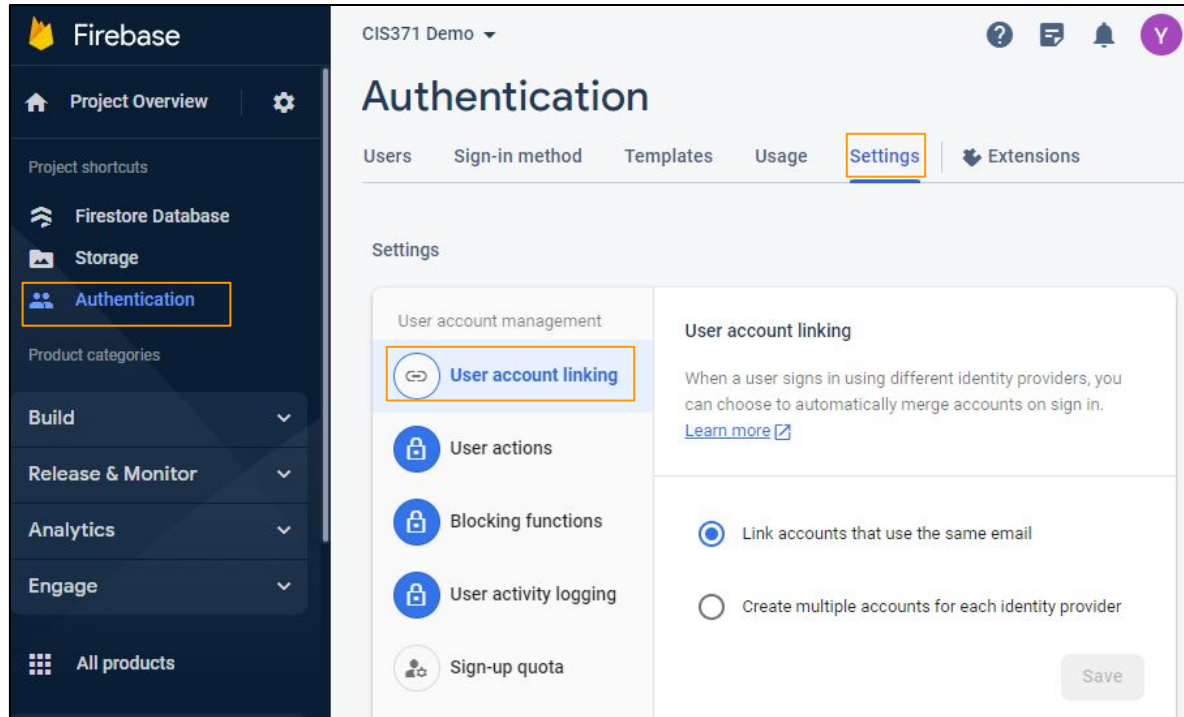
Authorized domains

For security, to use Phone, Google or third-party authentication, domains need to be authorized for OAuth redirects. [Learn more](#)

[Add domain](#)

Authorized domain	Type
localhost	Default
cis371-demo-770dc.firebaseio.com	Default
cis371-demo-770dc.web.app	Default
127.0.0.1	Custom

# Authentication Dashboard: Advanced Option



**Authentication Functions are async**  
**Use Promise.then or await to handle the result**



# Firebase Auth: Create A New Account

```
import {
  getAuth,
  createUserWithEmailAndPassword,
  UserCredential,
} from "firebase/auth";
const auth = getAuth();

createUserWithEmailAndPassword(auth, "me@sample.com", "1q2w3e4r5")
  .then((cred: UserCredential) => {
    sendEmailVerification(cred.user);
    console.log("Verification email has been sent to", cred.user?.email);
    auth.signOut();
  })
  .catch((err: any) => {
    console.error("Oops", err);
  });
```

# Firebase Auth: Signin With Email

```
import {
  getAuth,
  signInWithEmailAndPassword,
  UserCredential,
} from "firebase/auth";
const auth = getAuth();

signInWithUserWithEmailAndPassword(auth, "me@sample.com", "1q2w3e4r5")
  .then((cred: UserCredential) => {
    if (cred.user?.emailVerified) console.log("Signed in as", cred.user?.email);
    else {
      console.log("Please verify your email first");
      auth.signOut();
    }
  })
  .catch((err: any) => {
    console.error("Oops", err);
  });
```

# Firebase Auth: Sign In With Providers

```
import { getAuth, GoogleAuthProvider, signInWithPopup } from "firebase/auth";
const auth = getAuth();

const provider = new GoogleAuthProvider();

signInWithPopup(auth, provider)
  .then((result) => {
    const cred = GoogleAuthProvider.credentialFromResult(result);
    console.log("Signed in as", cred.user?.email);
  })
  .catch((err: any) => {
    console.error("Oops", err);
  });
```

*Make sure you have added the login provider in the Authentication Dashboard*

# Monitor Authentication in Background

```
import { getAuth, User, onAuthStateChanged } from "firebase/auth";
const auth = getAuth();

onAuthStateChanged(auth, (user: User | null) => {
  if ((currentUser == null) & (user != null)) {
    currentUser = user;
    /* User just logged in */
  } else if ((currentUser != null) & (user == null)) {
    /* User just logged out, do clean up work */
  }
});
```

a list of available properties

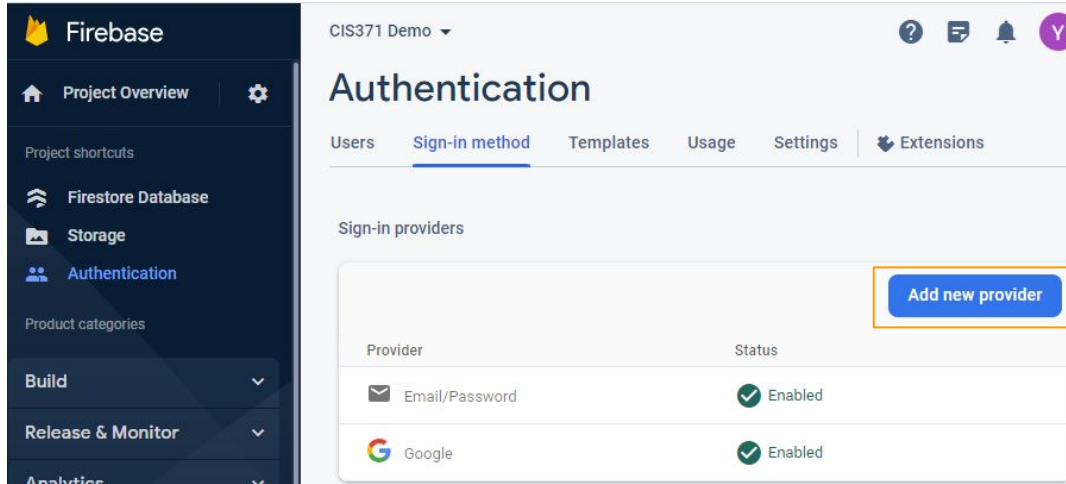
*Use onAuthStateChanged if you need to monitor login status in background*

# 3rd party Account Providers


Account Provider	Provider Class
Facebook	FacebookAuthProvider
GitHub	GithubAuthProvider
Google	GoogleAuthProvider
Phone	PhoneAuthProvider
Twitter	TwitterAuthProvider

# Example: GitHub SignIn

# Step A: Enable Github Login




Configure provider (Step 2 of 2)

 GitHub ☒ Enable

Client ID  
  
A client ID is required

Client secret  
  
A client secret is required

To complete set up, add this authorization callback URL to your GitHub app configuration.  
[Learn more](#)

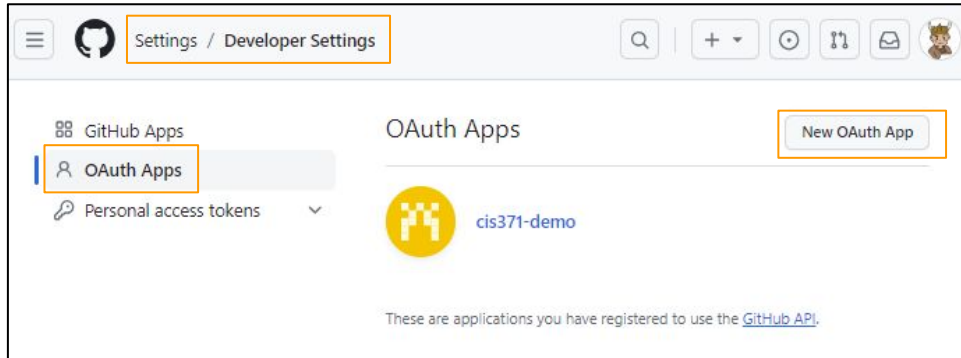


Cancel Save

Do this step from Firebase Authentication Dashboard

# Step B: Create an OAuth App (On Github Settings)

*The following GitHub page is under Settings => Developer Setting*



*Create a new  
Github OAuth App*

**Do this step from GitHub Developer Settings**



# Step D: Copy Client ID & Client Secret

Register a new OAuth application

Application name \*

Something users will recognize and trust.

Homepage URL \*

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL \*

Your application's callback URL. Read our [OAuth documentation](#) for more information.

☐ Enable Device Flow

Allow this OAuth App to authorize users via the Device Flow.  
Read the [Device Flow documentation](#) for more information.

**Register application** Cancel


*app name to show to the user at log in time*

*enter a valid URL and this can be changed later*

*copy and paste from  
Firebase Authentication  
(from Step A)*

General Optional features Advanced

cis371-demo


 Yong-Zhuang owns this application. [Transfer ownership](#)

You can list your application in the [GitHub Marketplace](#) so that other users can discover it. [List this application in the Marketplace](#)

0 users [Revoke all user tokens](#)

Client ID

Client secrets [Generate a new client secret](#)

 Added 24 minutes ago by Yong-Zhuang  
Never used  
You cannot delete the only client secret.  
Generate a new client secret first. [Delete](#)

# Step D: Copy Client ID & Client Secret

Configure provider (Step 2 of 2)



☒ Enable

Client ID

Client secret

To complete set up, add this authorization callback URL to your GitHub app configuration.

[Learn more](#)

`https://cis371-demo-770dc.firebaseio.com/_/auth/handler`



Cancel

Save

**Do this step from Firebase Authentication Dashboard**