

# CIS 371 Web Application Programming

## Cloud DataBase

### Firestore User Authentication



GRAND VALLEY  
STATE UNIVERSITY®

Lecturer: **Dr. Yong Zhuang**

# Firebase: A collection of many products

- Authentication
- Cloud Firestore
- Cloud Storage
- Cloud Functions
- ...

# Setup: Install and Initialize Firebase

```
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*

# Setup: Use Auth Functions

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

OR

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

```
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);
```

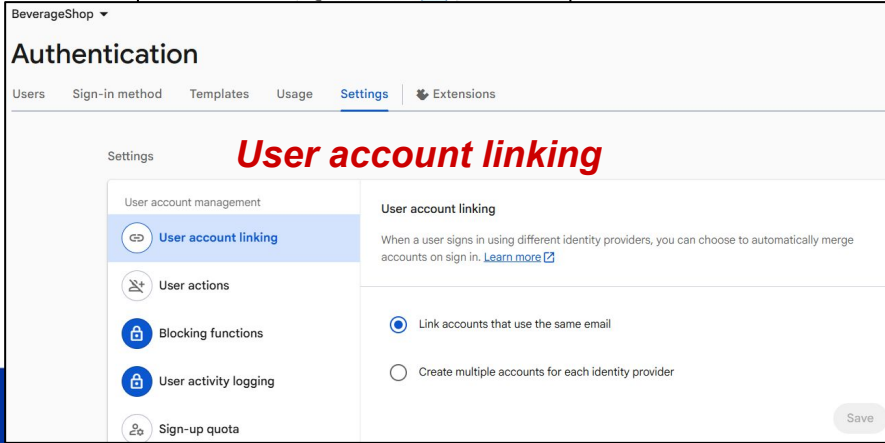
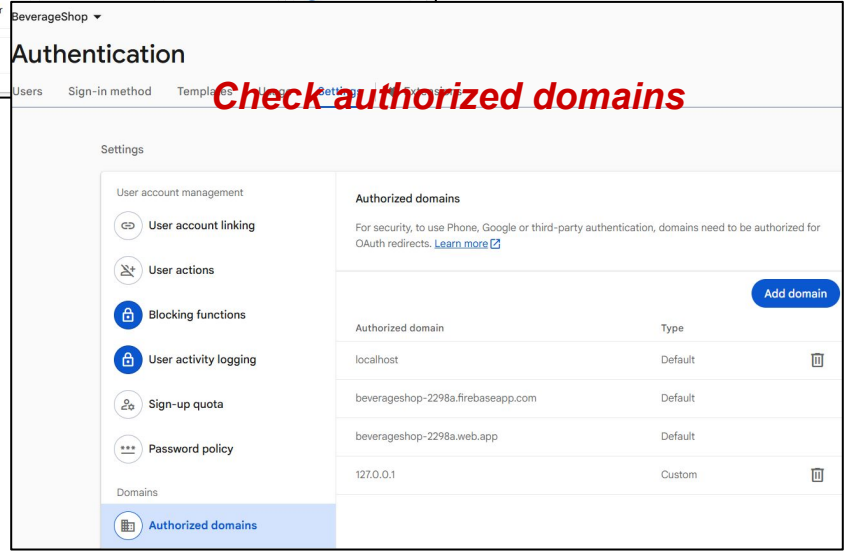
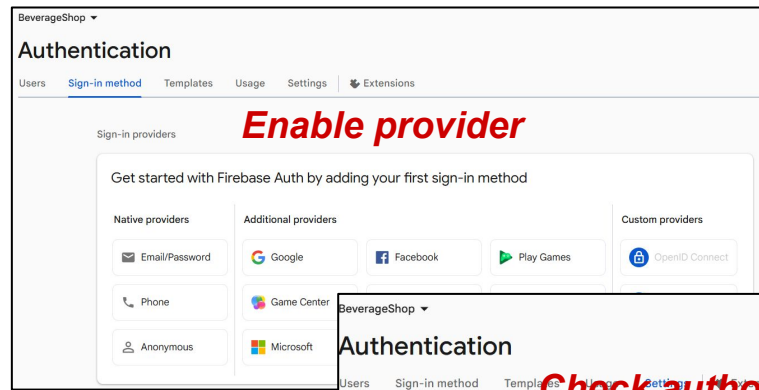
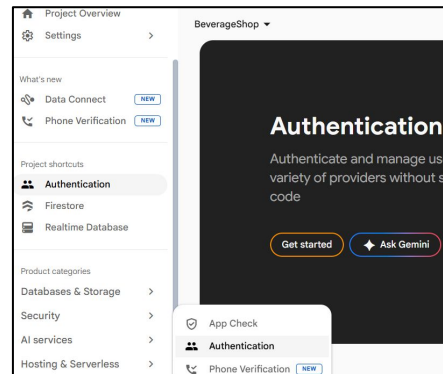
*login.vue*

# Authentication Options

- Email/Password
- GitHub accounts
- Google accounts
- ...

- **Firebase Authentication Web docs.**

# What to configure in Firebase Console



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

# Firestore 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);
  });
```



What will happen after calling the  
`createUserWithEmailAndPassword`  
function?

# Firestore Auth: Signin With Email

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

signInWithEmailAndPassword(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);
  });
```

# Firestore 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

BeverageShop ▾

## Authentication

Users Sign-in method Templates Usage Settings Extensions

Sign-in providers

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](#)

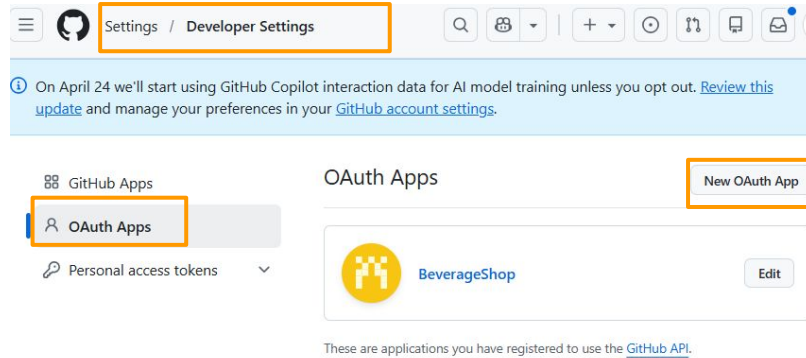
Copy callback URL

Cancel Save

Do this step from Firebase Authentication Dashboard

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

*The following 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)*

Settings / Developer settings / BeverageShop

General BeverageShop

Optional features

Advanced

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](#)

2 users [Revoke all user tokens](#)

Client ID

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

[Client secret](#)  Last used within the last 5 months [Delete](#)

You cannot delete the only client secret. Generate a new client secret first.

# Step D: Copy Client ID & Client Secret

GitHub  Enable

Client ID

Client secret

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

[Learn more](#)

`https://beverageshop-2298a.firebaseio.com/_/auth/handler`

Cancel

Save

Do this step from Firebase Authentication Dashboard

# Exercises: Try User Authentication

## 1. Open Your Firebase Project

- Go to Firebase Console and open your **BeverageShop** project
- In the left sidebar, go to **Build** → **Authentication**
- Click **Get started**

## 2. Enable a Sign-In Method

- Open the **Sign-in method** tab
- Enable **Google Sign-In**
- Save the changes

## 3. Check Authorized Domains

- Open the **Settings** tab in Authentication
- Find **Authorized domains**
- Make sure your local development domain is listed, such as `localhost`, `127.0.0.1`