Programming Boot Camp #3

3: User registration, user authentication and database integration (Firebase)

Tokyo Institute of Technology 2020/11/14

Ryo Imahashi

Table of Contents

Review previous sessions

- In the first session, we learned the basics of HTML, CSS, and Javascript.
  - Reference: 1st handout
- In the second session, we learned the basics of Vue.js (Nuxt.js).
  - Reference: 2nd edition

Get your development environment ready

- If this is your first time participating in this event, set up an environment to develop on your own PC. The GuildWorks members will support you, so please ask them.
  - Reference: Environment building
- If you're ready, download the latest code from [remote repository on GitHub] (https://github.com/GuildWorks/titech-2020).
  - Open GitBash(Terminal) in the `titech-2020` directory and run the following command.

```
git pull
```

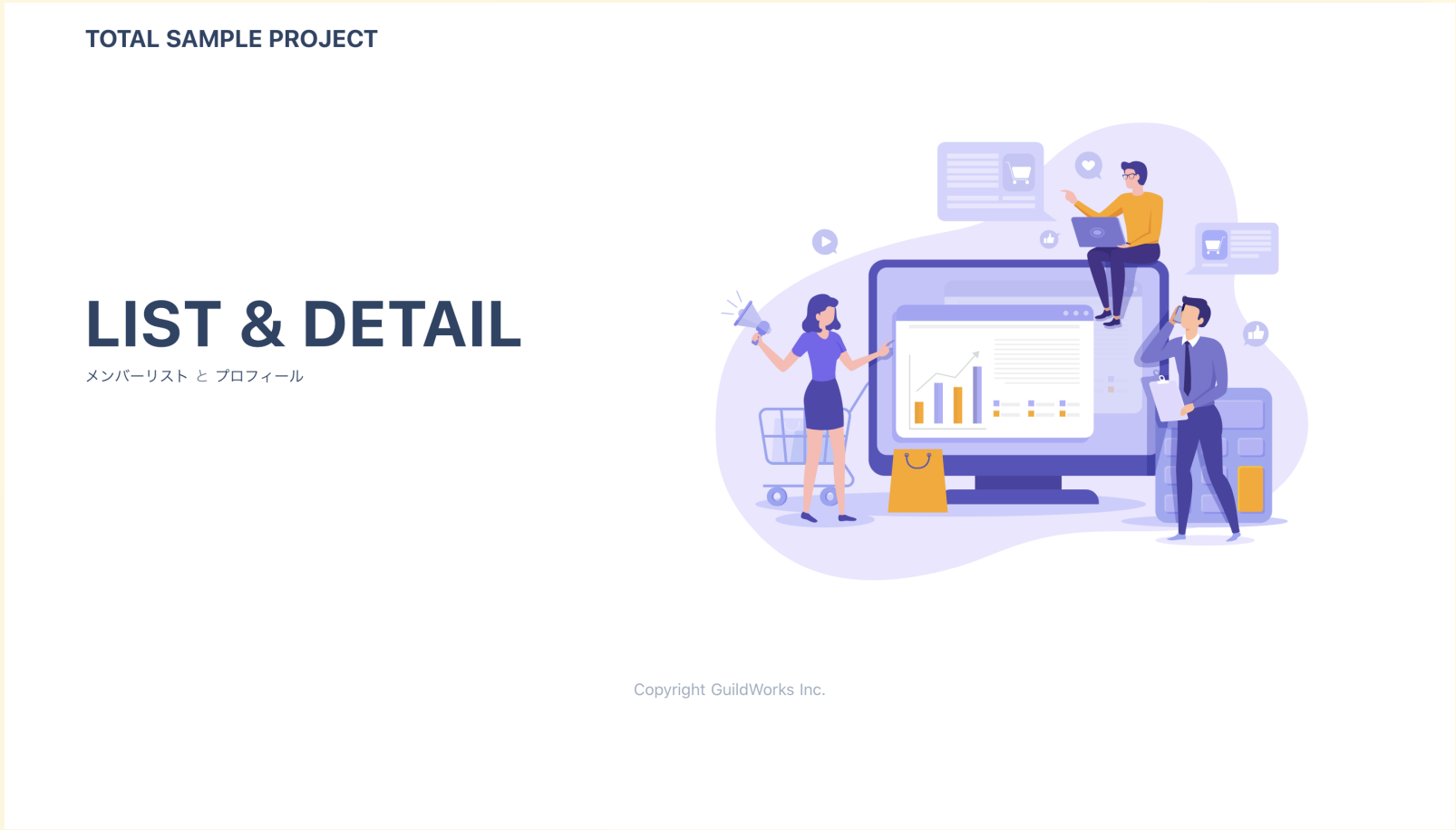
Confirm the app we created

- To recap, let's run the app we created last time to see how it works.
  - Let's run the following command in GitBash (Terminal)

```
# Lines beginning with # are comments and will not be executed

# If your current location is in the titech-2020 directory, move
cd titech-nuxt-day2-answer

# Install the required modules
npm install

# Launch the app.
npm run dev
```

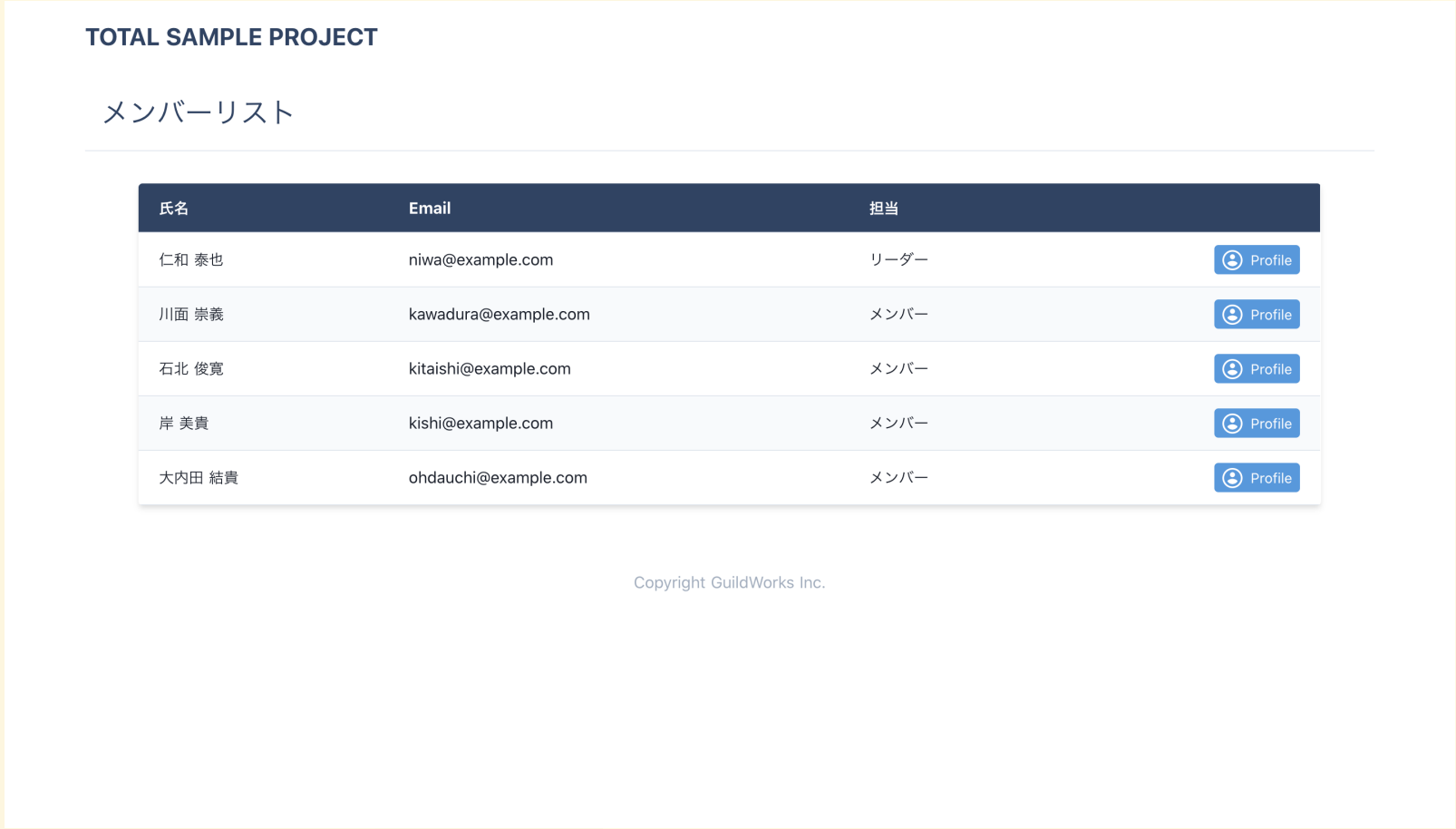
  - Then go to http://localhost:3000/.

Top Screen



http://localhost:3000/

members list



http://localhost:3000/list ...

member details

http://localhost:3000/user/0001

where I'm concerned.

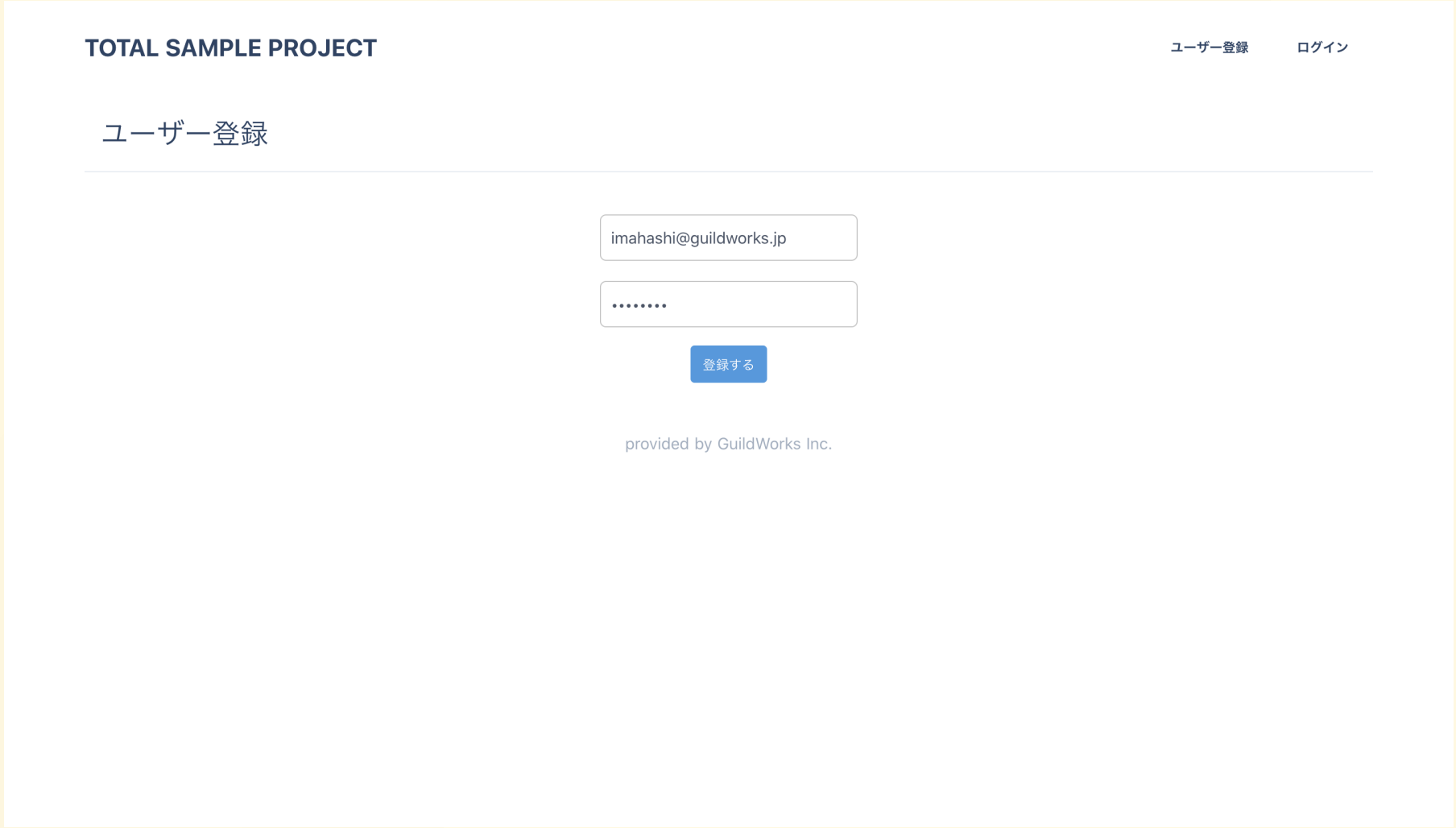
- The content of the display does not change.
  - Displayed data is sample.
    - We can't add a user.
    - We can't update our profile.
- Anyone can see the data without logging in.

Overview of what we do today

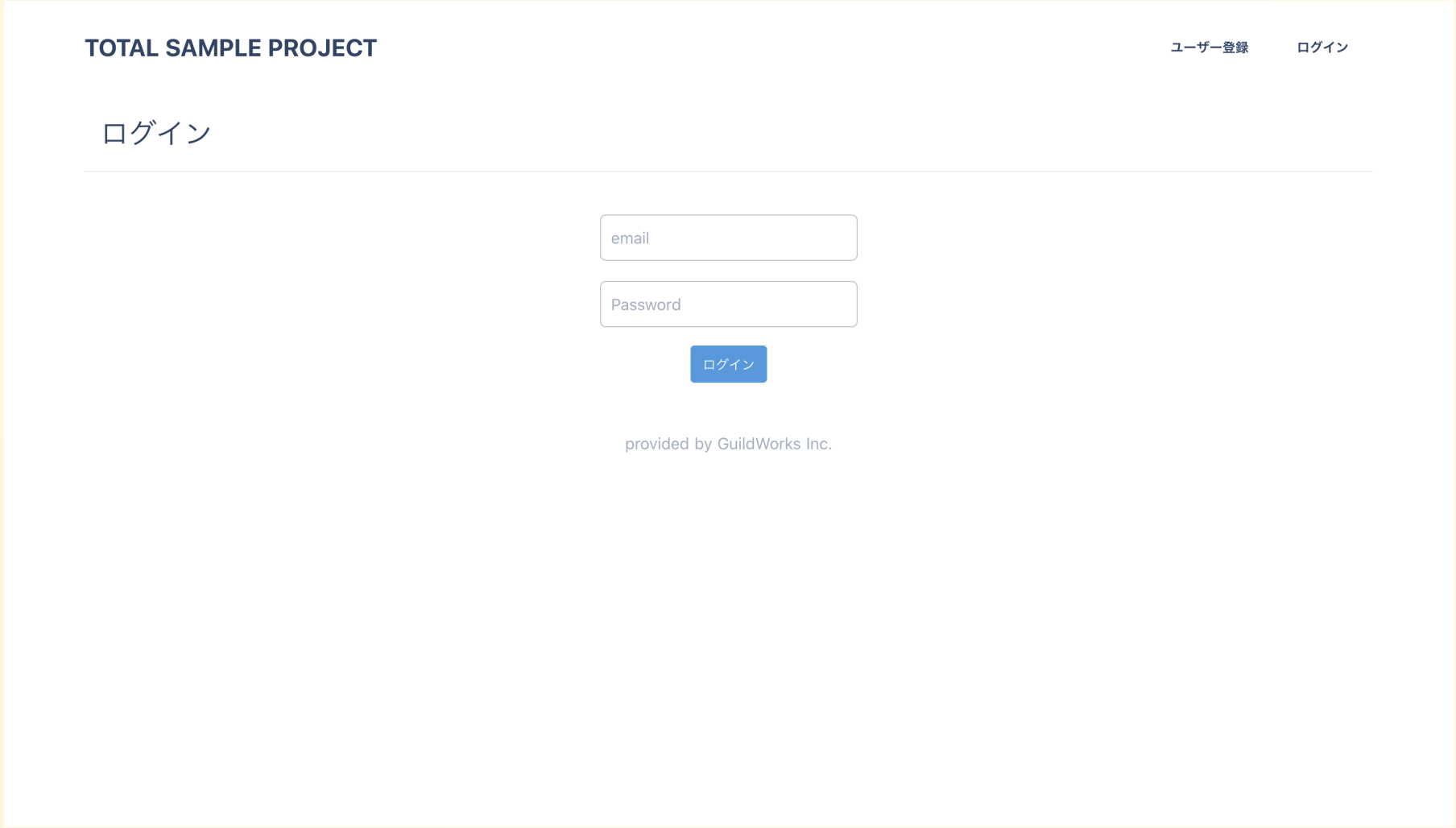
The functions and screens to be developed are as follows

- User registration
- Signin
- Access control by sign-in status
- Signout
- Edit Profile
- Your Profile
- Member list
- Member Profile

User registration

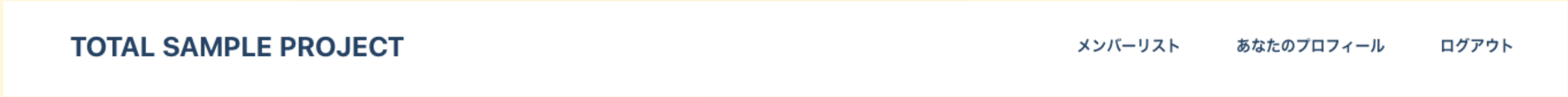


Signin

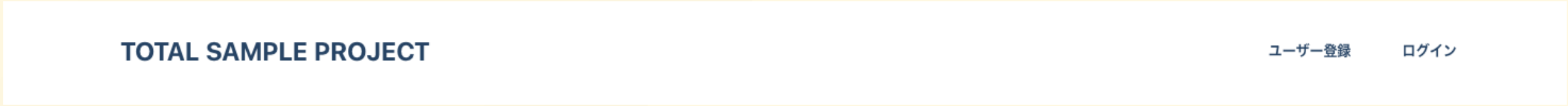


Access control by sign-in status, Signout

When you are logged in.



When you are not logged in



Profile edit

Your Profile

TOTAL SAMPLE PROJECT

メンバーリスト　あなたのプロフィール　ユーザー登録　ログアウト

あなたのプロフィール

編集

NAME

今橋 陵

imahashi+3@guildworks.jp

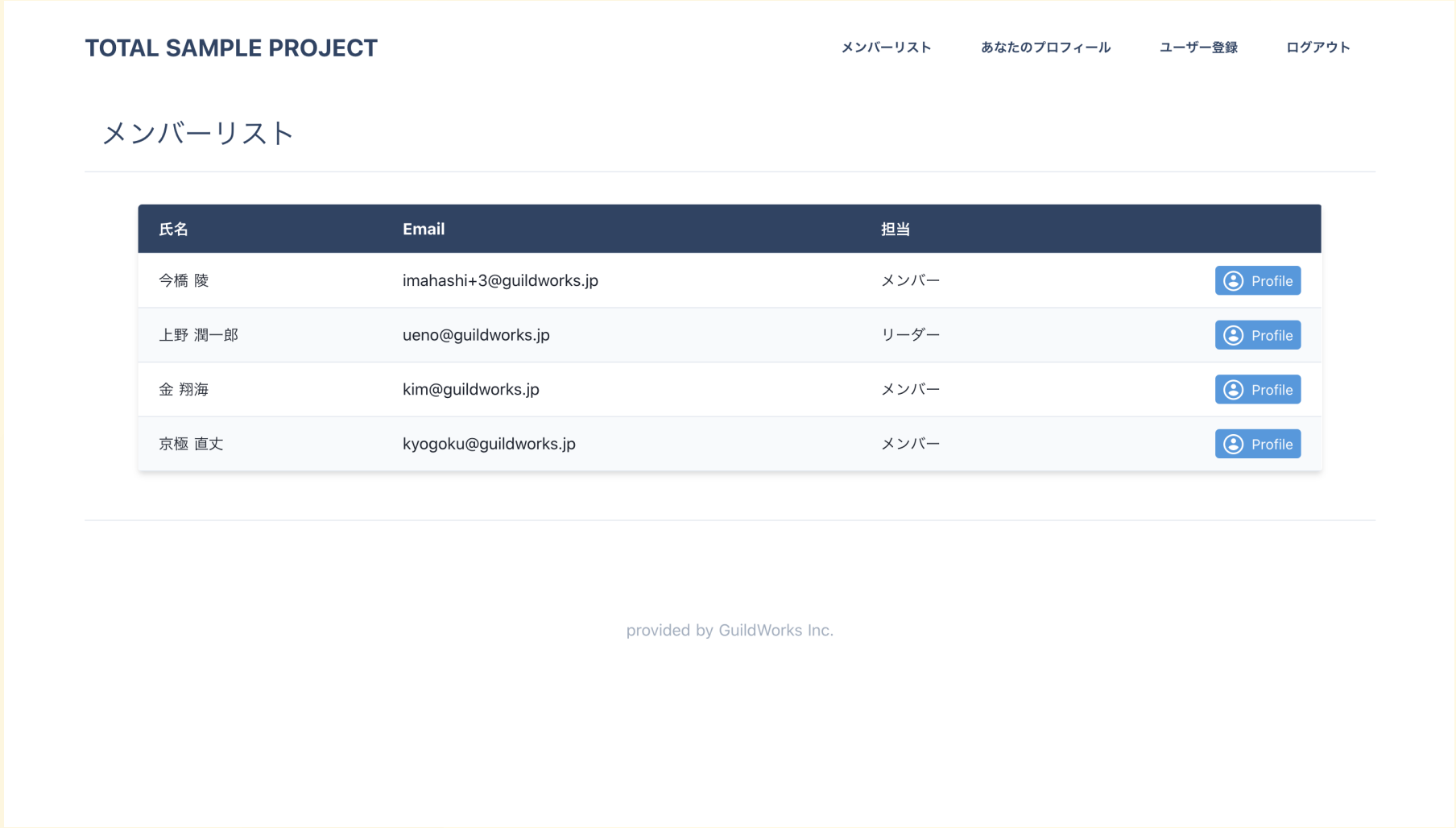
こんにちは。ギルドワークスの今橋です。

今日はみんなで楽しくプログラミングをしましょう。

所属・部署	ギルドワークス
ニックネーム	いまはし
出身地	山口
生年月日	1992年7月24日
血液型	O型
星座	獅子座
趣味	猫と遊ぶ・料理

provided by GuildWorks Inc.

Member list



Member profile

TOTAL SAMPLE PROJECT

メンバーリスト　あなたのプロフィール　ユーザー登録　ログアウト

メンバープロフィール

NAME

上野 潤一郎

ueno@guildworks.jp

上野です。よろしくお願いします。

所属・部署	ギルドワークス
ニックネーム	うえじゅん
出身地	横浜
生年月日	1977年8月15日
血液型	内緒
星座	獅子座
趣味	Appleの新製品をチェックすること

provided by GuildWorks Inc.

- Check the current state of the app that we will be modifying.
  - In GitBash(Terminal) that we used earlier, press [Ctrl]+[C] to stop the app.
  - Let's run the following command.

```
pwd # Check current position
cd ..   # One level up
pwd # Again, check the current position.
cd titech-nuxt-firebase-tutorial # Move to the directory of the application we will modify this time
npm install # Install the required modules
npm run dev # Start the app
```

  - Then go to http://localhost:3000/.
    - You'll see each pages we modify today.

- Let's make it possible to use the backend process of the application, which is necessary for the function we will develop in this article.
  - What we created in the first and second part of this article is the front end (application). It works on the browser side.
  - The backend (the application) runs on the end of the Internet and returns the data you need. It is also called the server side (application).
  - It is often talked about in terms of the contrast between front-end and back-end, or the contrast between client-side and server-side.

- In this bootcamp, we won't be developing a backend from scratch, but instead will use a service called Firebase.

What is Firebase?

- Firebase is a mobile and web application backend service from Google that allows developers to focus on developing their applications, without having to create and manage a service to run on the backend.
- No servers or deployments are required, making it easy to use the backend process.
- There are free plans available that do not require registration of payment information, so let's use them.

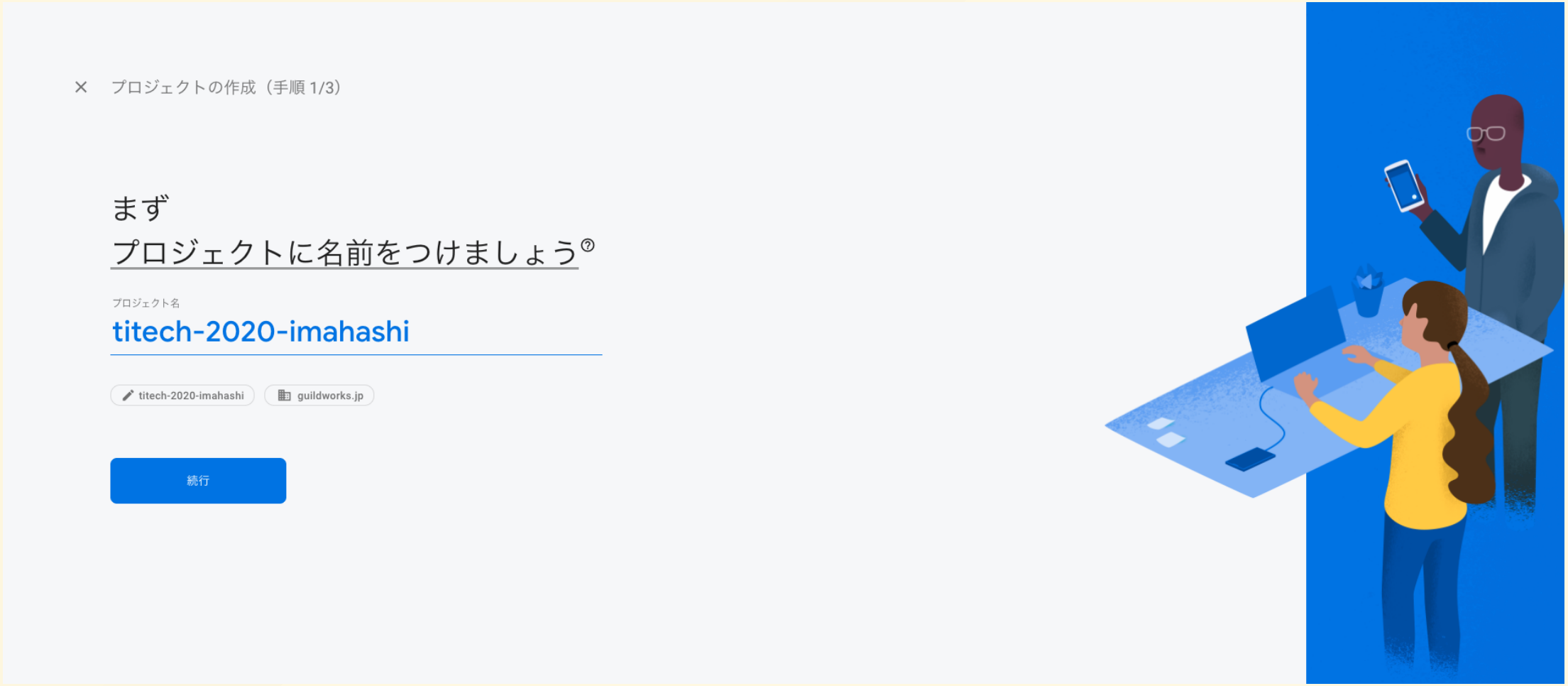
Get ready to use Firebase!

- Go to the following link.
  - https://console.firebase.google.com/?hl=ja&pli=1
- If you are asked to sign in to your Google account
  - If you already have a Google account, log in.
  - If you don't have a Google account, click the "Create Account" link at the bottom left to create an account.

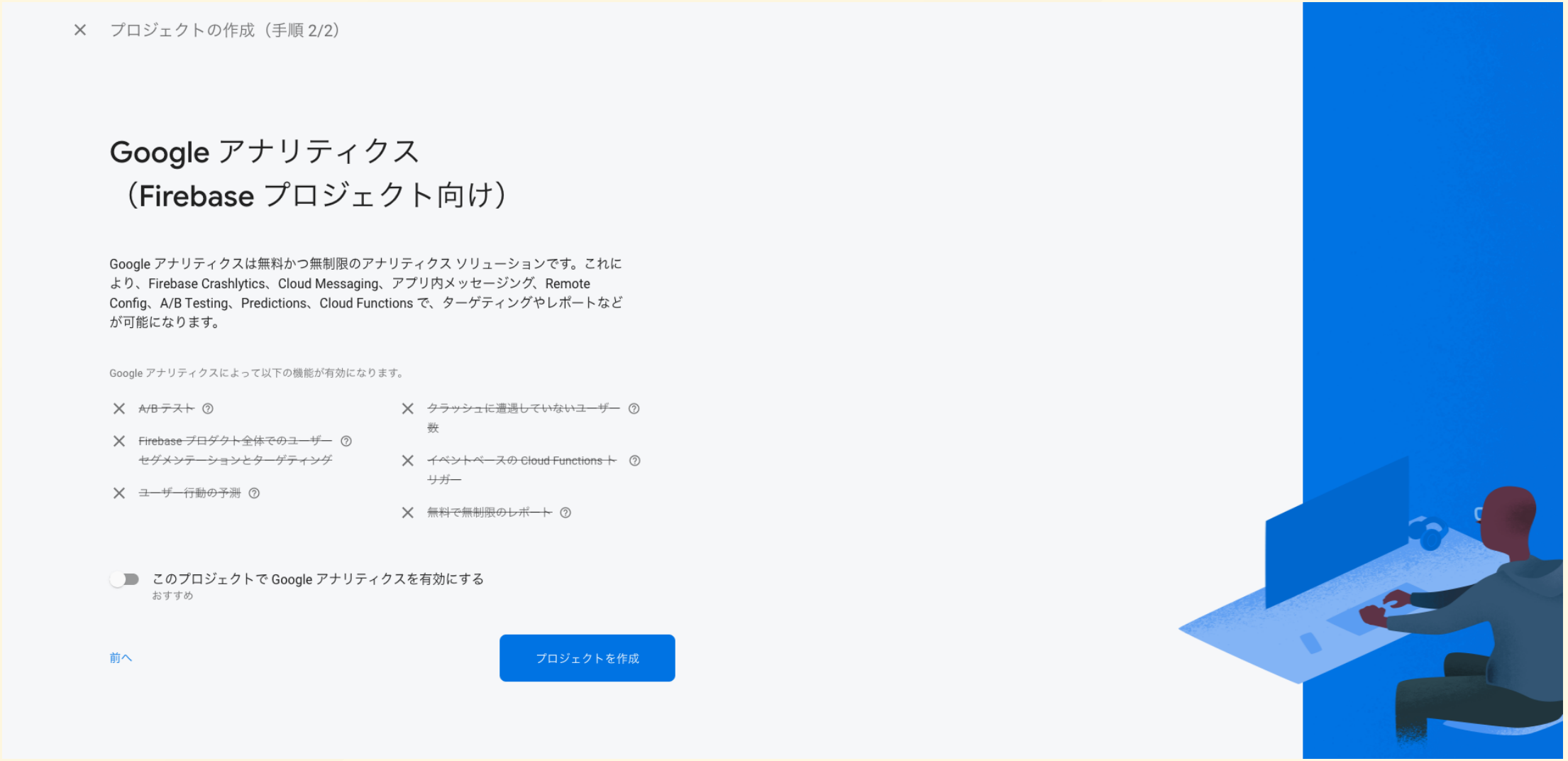
- If you see a screen like this, you're good to go.

- Click on "Create Project

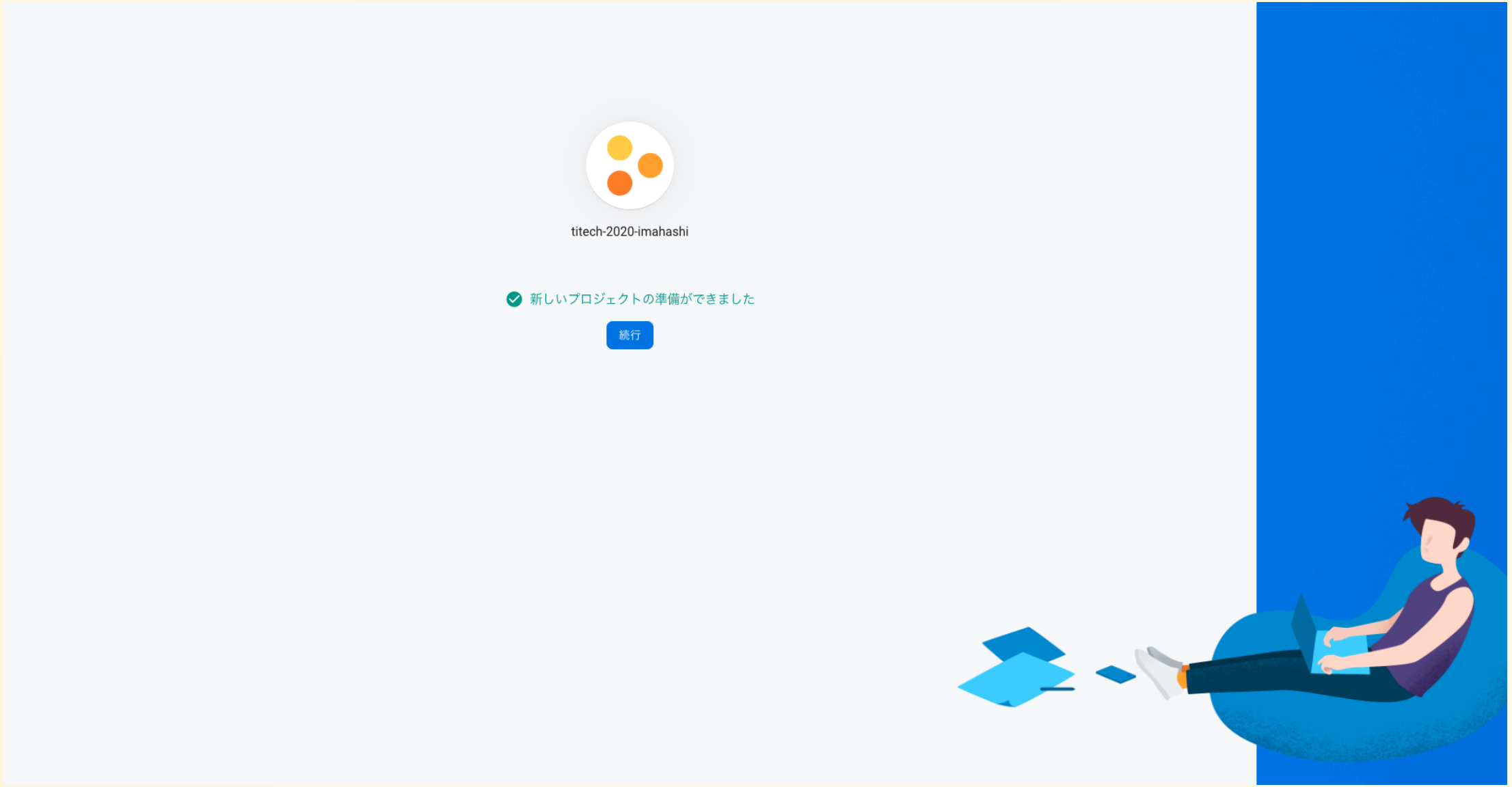
- Give your project a name and click "Continue".
  - You can use any name.



- Disable Google Analytics and click on "Create Project".



- Once you have created your project, click "Continue".



- This is the screenshot of the project that was created. We will use this project to develop it.

Let's set up the initial setup of Firebase in this app we're developing.

- Click on the gear icon in the top left corner of the screen and select "Set up your project".

- Click on the "</>" icon in the My Apps section at the bottom of the screen to create a web app for your project.

- Enter a nickname for your app and click "Register App".
  - You can use any nickname.

- Copy the selection (the line from apiKey to appId).

2 Firebase SDK の追加

これらのスクリプトをコピーして <body> タグの下部に貼り付けます。この作業は Firebase サービスを使用する前に行ってください。

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.0.1/firebase-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: "AIzaSyAZk4BIcu84kIK72SEkHSnafuZTP-7TCS8",
    authDomain: "titech-2020-imahashi.firebaseapp.com",
    databaseURL: "https://titech-2020-imahashi.firebaseio.com",
    projectId: "titech-2020-imahashi",
    storageBucket: "titech-2020-imahashi.appspot.com",
    messagingSenderId: "454271650",
    appId: "1:454271650:web:2252a5355e5f24d6052dfb"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```

ウェブ向け Firebase の詳細については、こちらをご覧ください: 使ってみる、

- Open Visual Studio Code and paste the copied settings into `/titech-nuxt-firebase-tutorial/plugins/firebase.ts`.

```
import firebase from 'firebase/app'
import 'firebase/auth'
import 'firebase/firestore'
import 'firebase/storage'

if (!firebase.apps.length) {
  firebase.initializeApp({
    apiKey: "AIzaSyAZk4BIcu84kIK72SEkHSnafuZTP-7TCS8",
    authDomain: "titech-2020-imahashi.firebaseapp.com",
    databaseURL: "https://titech-2020-imahashi.firebaseio.com",
    projectId: "titech-2020-imahashi",
    storageBucket: "titech-2020-imahashi.appspot.com",
    messagingSenderId: "454271650",
    appId: "1:454271650:web:2252a5355e5f24d6052dfb"
  })
}

export default firebase
```

- Now you're done with the initial setup of Firebase.

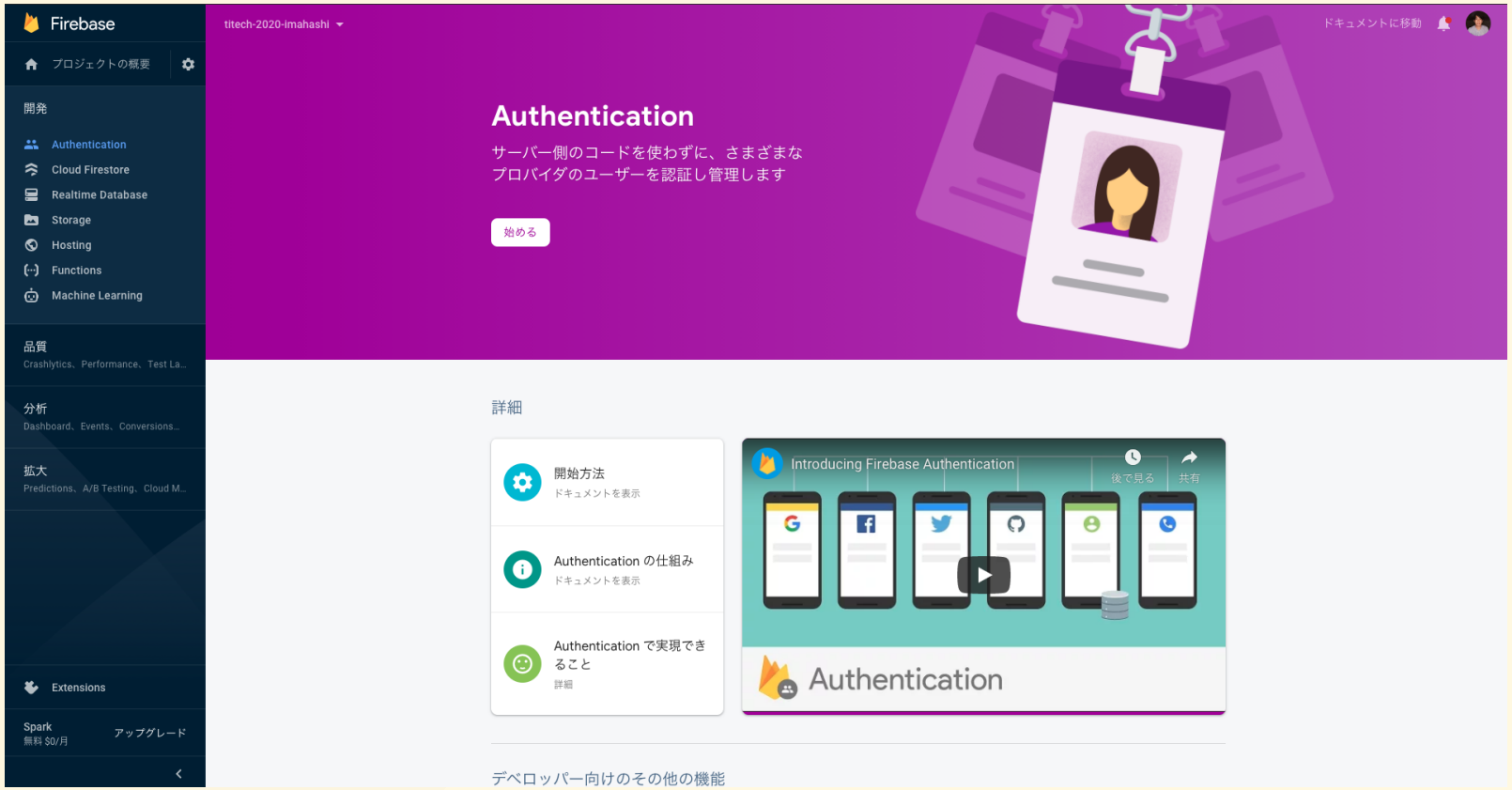
User Registration

- We use Firebase Authentication.
  - It is a service that provides authentication capabilities.
  - It supports various methods of authentication.
    - Email address and password
    - Google Account
    - Facebook account
    - Twitter account

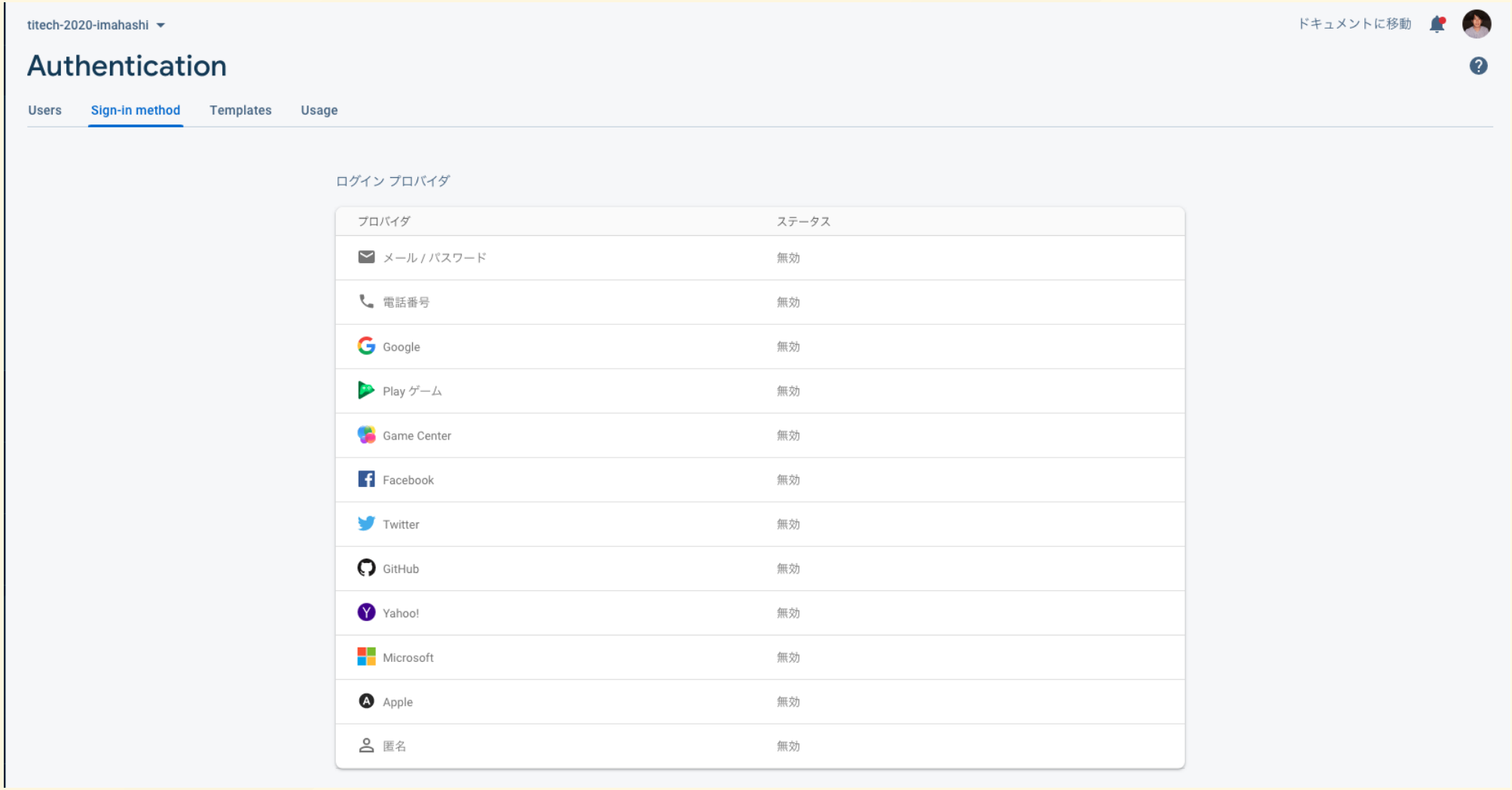
      etc.

Configuring the Firebase side

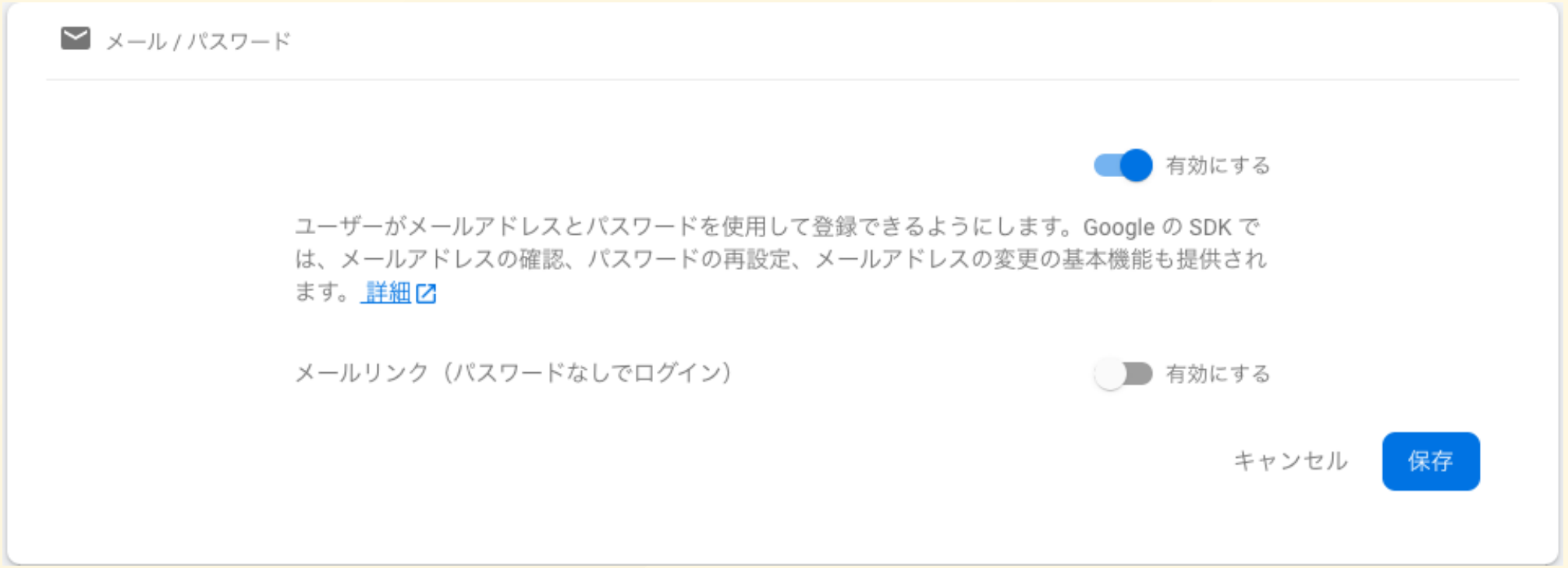
- Select "Authentication" in the menu on the left side and click "Get Started".



- In this case, click on "Email/Password" to use email and password authentication.



- Click on the first slider to activate it and click "Save".



- This will allow you to use the email and password authentication feature.

Editing the program

- Open `/pages/signup.vue`. This is the file of the user registration page.
- It shows the empty contents of the `submit()` function, which is executed when you click the "register" button.

```
function submit() {
  // TODO
}
```

  - In this function, we call the user registration process provided by Firebase Authentication by email address and password to allow users to register.

- Let's import firebase so that we can handle it in `/pages/signup.vue`.

```
<script lang="ts">
import { defineComponent, reactive } from 'nuxt-composition-api'
import PageHeading from '@/components/page-heading.vue'
import firebase from '@/plugins/firebase.ts' // Add this line
```

- Please open the following link. https://firebase.google.com/docs/auth/web/password-auth#create_a_password-based_account
  - It contains the code to invoke the user registration process by email address and password.
- Click on the "Copy code sample" icon in the upper right corner of the code block to copy it.

- Paste the copied code into the `submit()` function.

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().createUserWithEmailAndPassword(email, password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

  - Note: If you have a broken indent after pasting a code, you can add indentation to all selected lines by selecting multiple lines and pressing [Tab]. You can also press [Shift] + [Tab] to delete indentation.

- Change the argument `email` to `state.email` and `password` to `state.password` of `createUserWithEmailAndPassword()` since it doesn't work as is.

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().createUserWithEmailAndPassword(state.email, state.password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

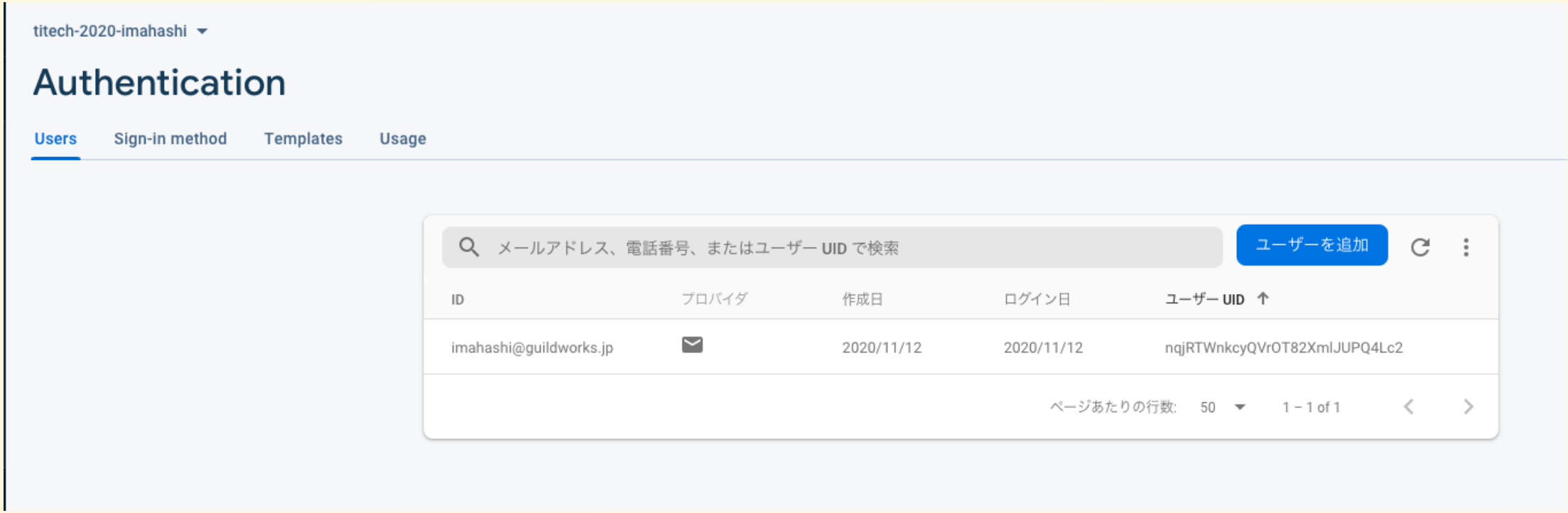
- The `email` and `password` entered into the screen is passed to `createUserWithEmailAndPassword()`, and the user registration request including them is executed.

- After completing the user registration, you should be taken to the edit profile page.

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().createUserWithEmailAndPassword(state.email, state.password)
  .then(() => (location.href = '/profile/edit')) // Add this line with a new line
  .catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

- Use the screen to actually register a user.
  - Enter your email address and password and press the register button.

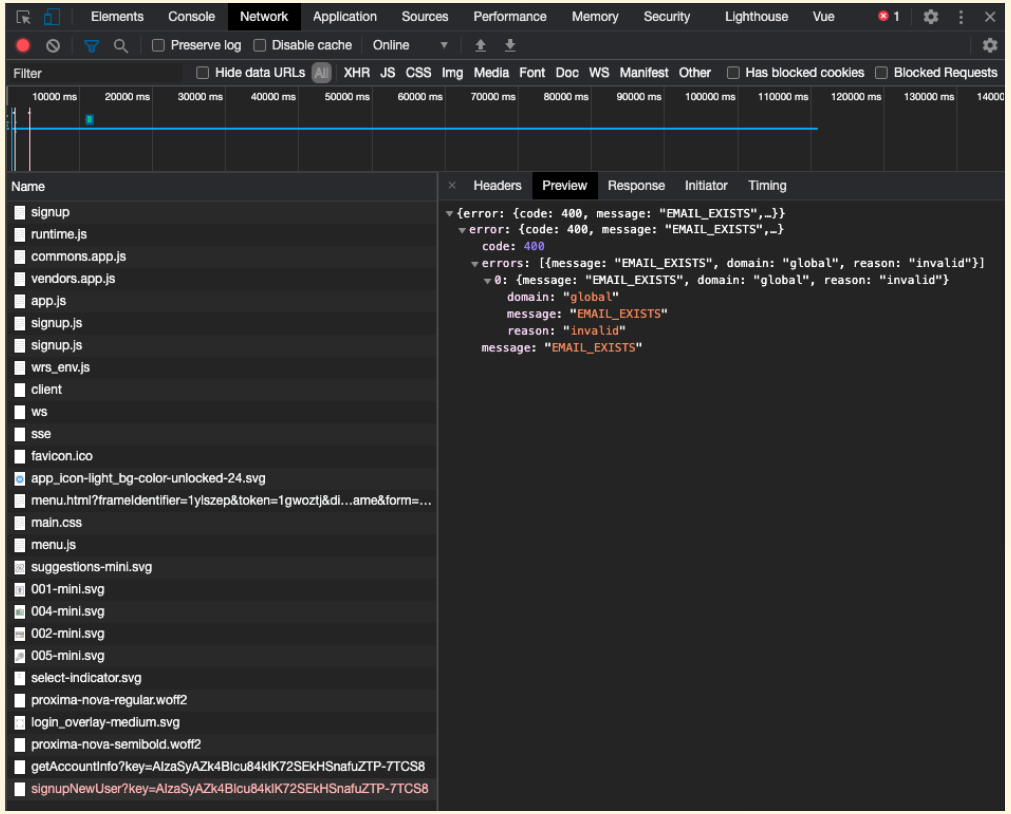
- Firebase Authentication verifies that the user has been registered.



- Open the developer tools in your browser (in Chrome, right click and select "Verify").
- Select the Network tab from the tab at the top.

- Now, let's register again with the same email address as before.

- There is an error with the message `EMAIL_EXISTS`.

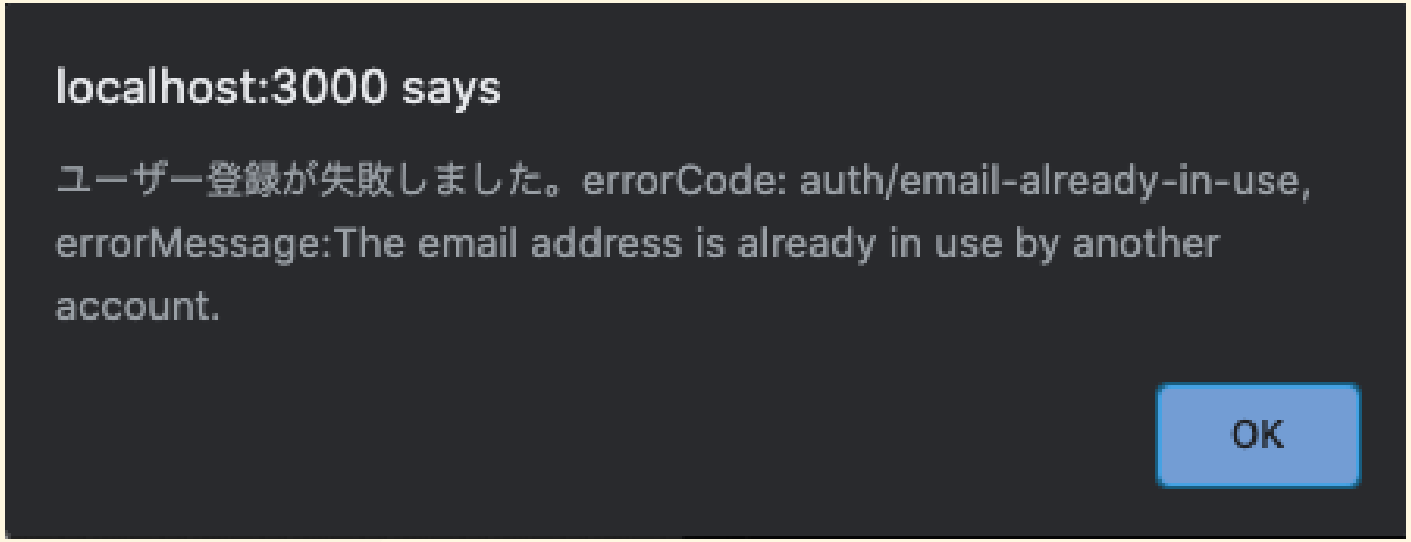


  - This is because the default setting of Firebase Authentication restricts the ability of multiple users to register with the same email address.

- If an error occurs, let's make sure we know about it. Please modify the `catch()` as follows.

```
        .catch(function(error) {
          // Handle Errors here.
          alert('User registration failed. errorCode: ' + error.code + ', errorMessage:' + error.message)
        });
```

- If you get an error, you will get the following pop-up Please try to register as a user again with the same email address.



User Authentication

- Again, we'll use Firebase Authentication.
- Let's open `/pages/signin.vue`, which is the login screen file. This is the login screen file.
- The `submit()` function, which is executed when you click the "login" button, is empty.

```
function submit() {
  // TODO
}
```

  - In this function, we need to call the login process with your email address and password to enable the login.

- Let's import firebase so that we can handle it in `/pages/signin.vue`.

```
<script lang="ts">
import { defineComponent, reactive } from 'nuxt-composition-api'
import PageHeading from '@/components/page-heading.vue'
import firebase from '@/plugins/firebase.ts' // Add this line
```

- Please open the following link. https://firebase.google.com/docs/auth/web/password-auth#sign_in_a_user_with_an_email_address_and_password
  - It contains the code to invoke the login process by email address and password.
- Click on the "Copy code sample" icon in the upper right corner of the code block to copy it.

メールアドレスとパスワードを使用してユーザーのログインを行う

パスワードを使用したユーザーのログイン手順は、新しいアカウントの作成手順と似ています。アプリのログインページで、次の手順に沿って操作します。

1. ユーザーがアプリにログインしたら、そのユーザーのメールアドレスとパスワードを `signInWithEmailAndPassword` に渡します。

```
firebase.auth().signInWithEmailAndPassword(email, password).catch(function(error) {
  // Handle Errors here.
  var errorCode = error.code;
  var errorMessage = error.message;
  // ...
});
```

ログインしたユーザーの詳細情報を取得する方法については、下記の「次のステップ」をご覧ください。

- Paste the copied code into the `submit()` function.

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().signInWithEmailAndPassword(email, password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

- Let's change the code as follows

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().signInWithEmailAndPassword(state.email, state.password)
  .then(() => (location.href = '/users')) // takes us to the user list after login is complete
  .catch(function(error) {
    // Handle Errors here.
    alert('login failed. errorCode: ' + error.code + ', errorMessage:' + error.message)
  });
}
```

- Changed the argument `email` to `state.email` and `password` to `state.password` of `signInWithEmailAndPassword()`.
- Changed `email` and `password` to `state.email` and `state.password` to `state.password` when login succeeds.
- Added an alert on login failure.

- Use the screen to actually log in.
  - Enter your email address and password and press the login button.
    - If the user's email address and password are the same as the one you just created, you will be taken to the user list screen.
    - If the email address is different from the user's email address, an `auth/user-not-found` error will occur.
    - If the email address is the same as the user we just created and the password is different, the `auth/wrong-password` error will occur.

Access control by sign-in status

- Let's control whether or not to display each screen according to the login status.
- We'll use the middleware mechanism in Nuxt.js to determine the login status.
  https://ja.nuxtjs.org/docs/2.x/directory-structure/middleware/ ...
  - In the middleware section, describe what you want to do before the screen is drawn.

- Open `/middleware/Auth.js`.
  - Open `/middleware/Auth.js`, which has no processing written in it yet.

```
export default function () {

}
```

- Let's change `/middleware/Auth.js` as follows.

```
import firebase from '@/plugins/firebase.ts'
export const skipAuthPaths = ['/signin', '/signup']

export default function ({ redirect, route, store }) {
  firebase.auth().onAuthStateChanged(function (user) {
    if (user) {
      // User is signed in.
      store.commit('signedIn', true);
    } else { { // User is signed in.
      // No user is signed in. store.commit('signedIn', false;)
      store.commit('signedIn', false);
      if (!skipAuthPaths.includes(route.path)) {
        redirect('/signin')
      }
    }
  })
}
```

- The `firebase.auth().onAuthStateChanged()` fetches the currently logged in user, and then branches off the rest of the process depending on whether the user is retrieved (= logged in or not). https://firebase.google.com/docs/auth/web/manage-users#get_the_currently_signed-in_user ...
- Store.commit()` stores the login state into the container named "store" to manage the state of the application.
  - We'll implement store later.
- If a user wants to show a screen for authentication without login, the function redirects the user to the login screen with `redirect('/signin')`.

- Open `/store/index.ts` and change it to look like this.

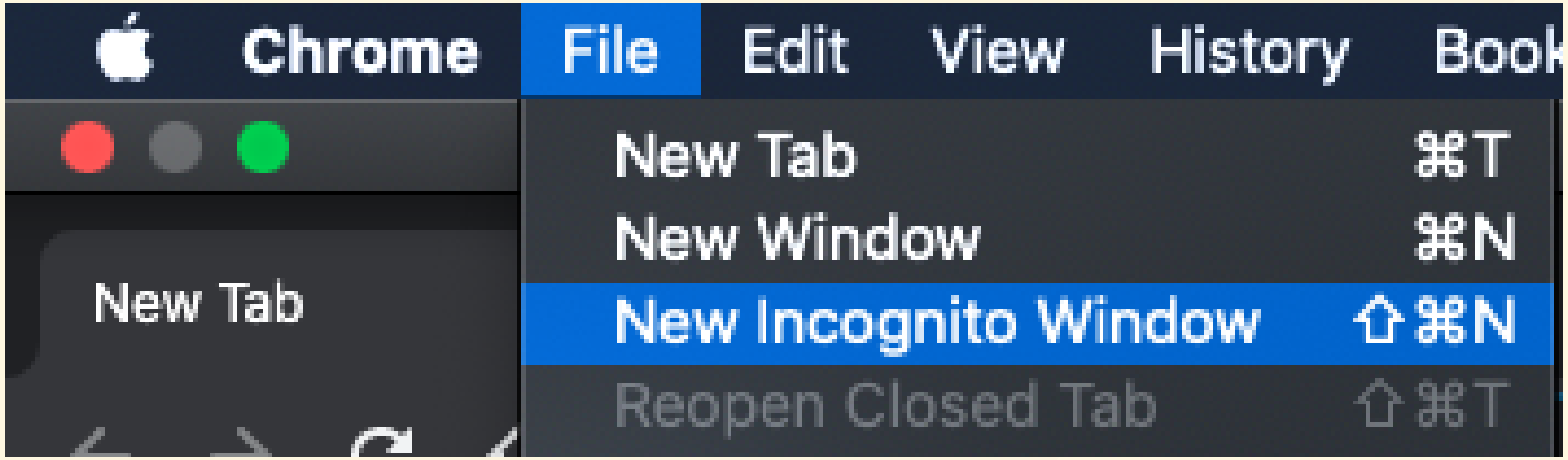
```
interface State {
  signedIn: boolean
}

export const state = () : State => ({
  signedIn: false
})

export const mutations = {
  signedIn(state: State, signedIn: boolean) {
    state.signedIn = signedIn
  }
}
```

  - The `store.commit('signedIn', true)` described in `/middleware/Auth.js` allows you to save the login status.

- Add a line to your `/layouts/default.vue` to run the `middleware/Auth.js` before the default layout is drawn.

```
<script lang='ts'>
import { defineComponent } from 'nuxt-composition-api'
export default defineComponent({
  middleware: ['Auth'], // Add this line and load Auth.js
  setup(_, { root: { $store } }) {
  }
})
</script>
```

- Open your browser in a secret window.



  - If you don't, your browser will remember that you have just logged in, so this will reopen the screen as it was before you logged in.

- Check the following in your secret window.
  - That when you go to http://localhost:3000 (top), you will be redirected to the login screen.
  - That you will be redirected to the login screen when you try to transition to the member list or your profile.
  - When you log in, you should be able to go to the top, the member list, and your profile.

Menu display control

- Open `/layouts/default.vue`.
- Nuxt.js makes sure that the content in this file is reflected in each page as the default layout.
  https://ja.nuxtjs.org/docs/2.x/concepts/views/#default-layout

- Let's change the menu to be displayed according to the login status.
- Let's modify `setup()` as follows.

```
setup(_, { root: { $store } }) {
  const isSignedIn = (): boolean => {
    return $store.state.signedIn
  }
  return {
    isSignedIn
  }
}
```

  - You can determine the login status from `isSignedIn`.
- Follow the code on the next page and add a branch to determine the login status by `v-if` in the `<template>` tag.

```
<a
  v-if="isSignedIn()"
  href="/users"
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"
>
  Member List
</a>
<a
  v-if="isSignedIn()"
  href="/profile"
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"
>
  Your Profile
</a>
<a
  href="/signup"
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"
>
  user registration
</a>
<a
  v-if="!isSignedIn()"
  href="/signin"
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"
>
  login
</a>
<a
  v-if="isSignedIn()"
  href="#"
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"
>
  logout
</a>
```

Log out

- Change the `<script>` in `/layouts/default.vue` as follows. The changes are as follows.
  - importing firebase.
  - Add `signOut` in `setup()` and return it to call it from `template`.
  - In `firebase.auth().signOut()`, the logout process is executed. https://firebase.google.com/docs/auth/web/password-auth#next_steps

```
import { defineComponent } from 'nuxt-composition-api'
import firebase from '@/plugins/firebase.ts'
export default defineComponent({
  middleware: ['Auth'],
  setup(_, { root: { $store } }) {
    const isSignedIn = (): boolean => {
      return $store.state.signedIn
    }
    const signOut = (): void => {
      firebase.auth().signOut()
    }
    return {
      isSignedIn,
      signOut
    }
  }
})
```

- Let's add an `@click="signOut"` to the `<a>` tag of the logout.

```
          <a
            v-if="isSignedIn()"
            href="#"
            @click="signOut"
            class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"
          >
            logout
          </a>
```

  - Click Logout to start the logout process.

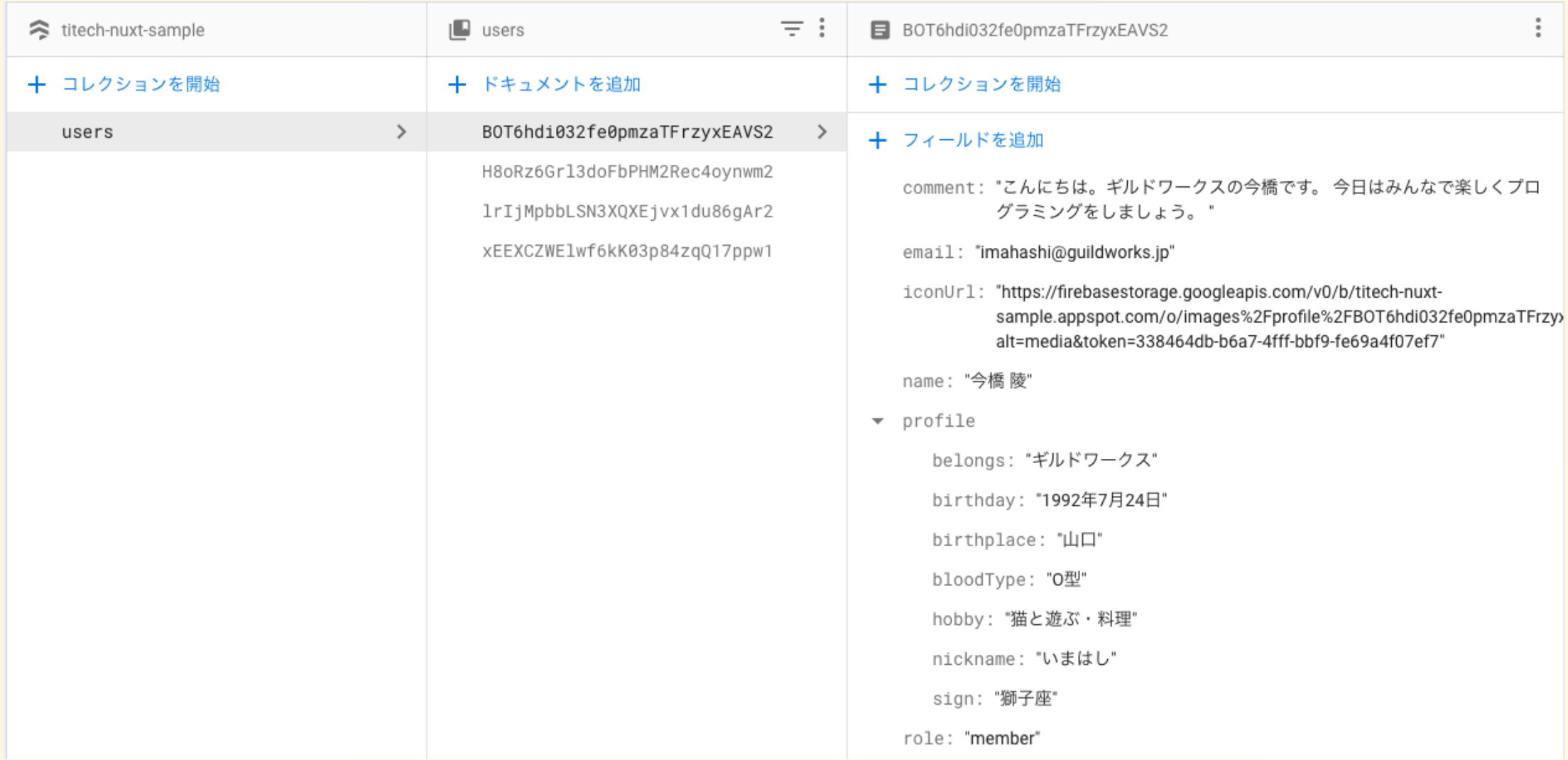
- Make sure that the menu changes as follows
  - When you are logged in.

  - When I logged out.

Database integration.

- Use Cloud Firestore.
  https://firebase.google.com/docs/firestore
  - It provides a database for storing data and the basic operations for handling data.
  - Data is stored in a hierarchical structure.

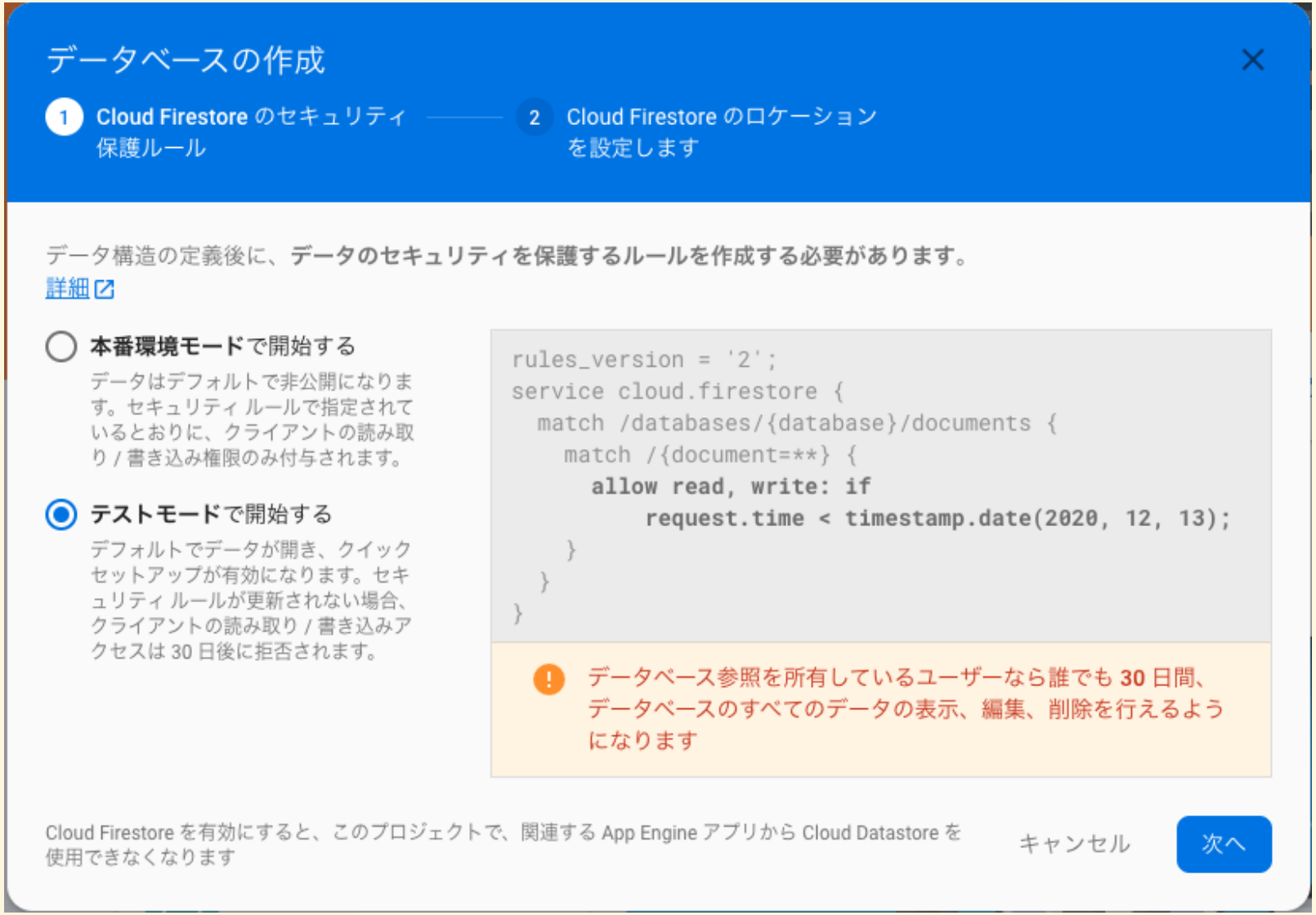
- Data Storage Image



Configuring the Firebase side

- Select "Cloud Firestore" in the menu on the left side and click on "Create Database".

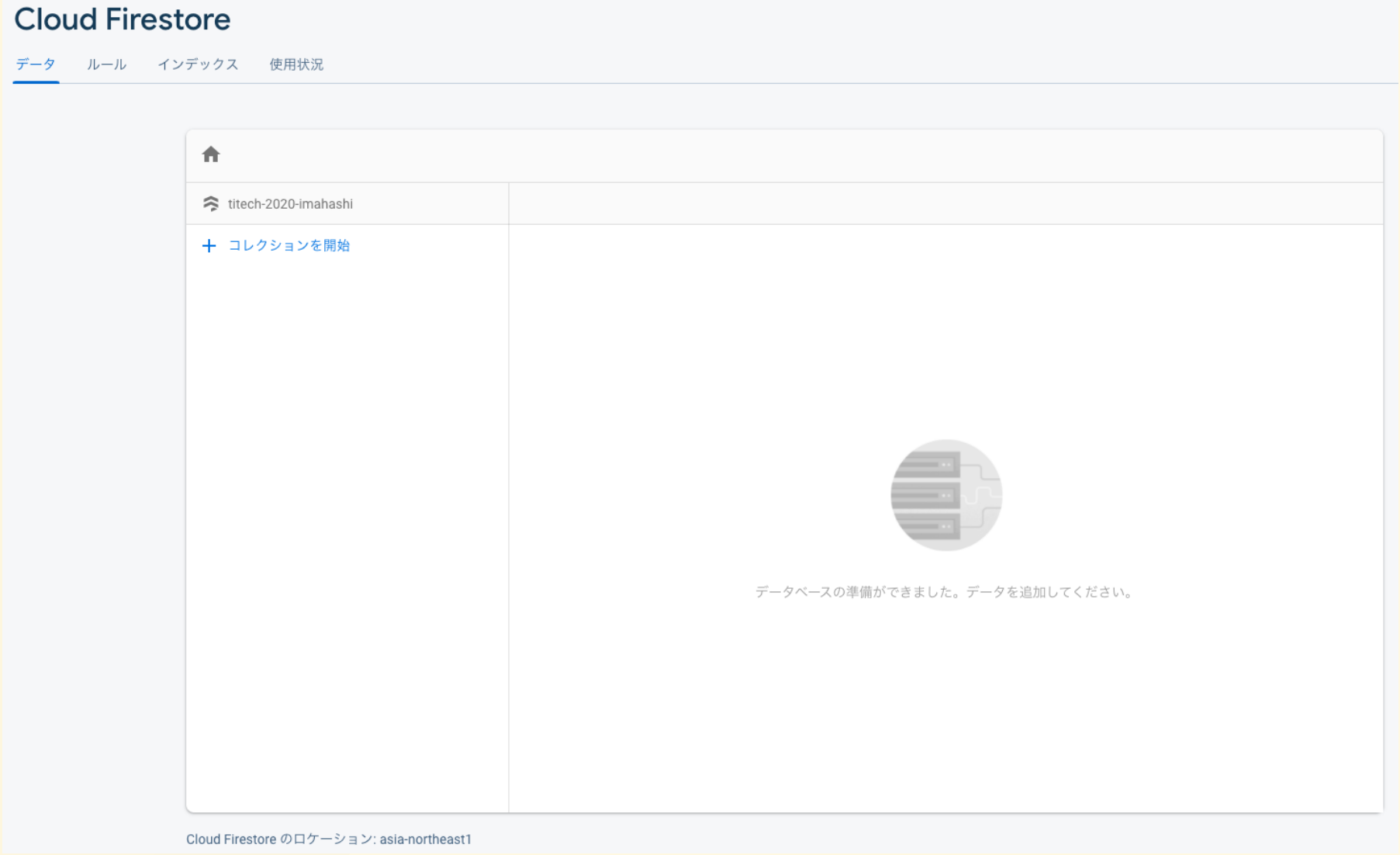
- Select "Start in Test Mode" and click "Next".



- Select "asia-northeast1" (Tokyo) and click "Enable".

https://firebase.google.com/docs/firestore/locations#location-r

- The database is ready to go.



Edit your profile (except for the icon image)

- Create a profile editor to register your own profile.
- Go to http://localhost:3000/profile/edit.
- Open `/pages/profile/edit.vue`, which is your profile's editing file. This is the file for your profile's edit screen.

- First, let's import the firebase.

```
import firebase from '@/plugins/firebase.ts'
```

- The next step is to get your user ID, which is needed to register your profile.

- Remove the comment `// TODO: get user ID, email address` and paste the following code into it.

```
firebase.auth().onAuthStateChanged(function (user) {
  if (user) {
    // User is signed in.
    userData.id = user.uid
    userData.email = user.email
    getUserData(user)
  } else { }
    // No user is signed in.
  }
})
```

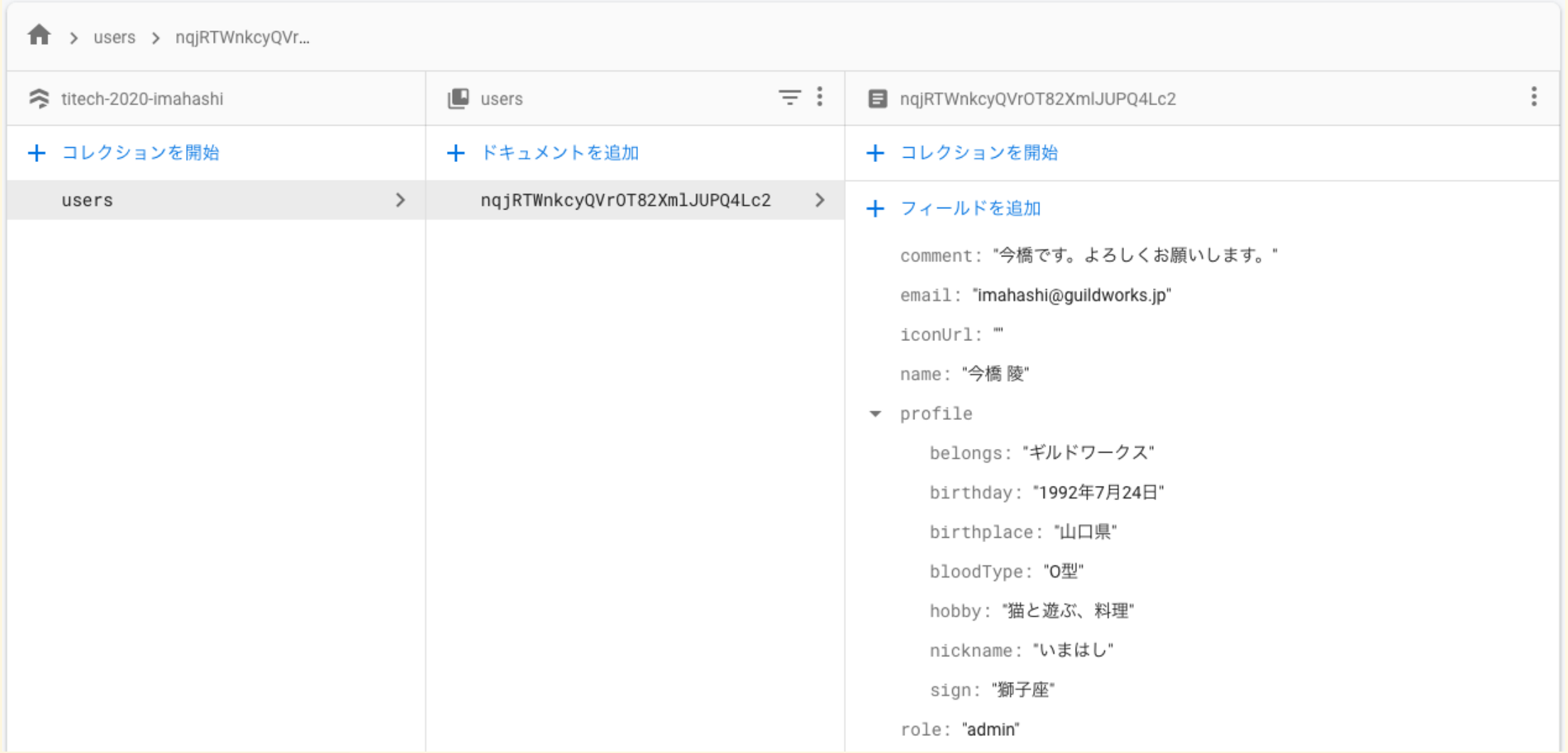
  - The same `onAuthStateChanged()` as used in `/middleware/Auth.js` is used to get the user ID and email address and set them where needed.

- Change the `setProfile` as follows.

```
const setProfile = (): void => {
  const data = {
    name: userData.name,
    email: userData.email,
    role: userData.role,
    iconUrl: userData.iconUrl,
    comment: userData.comment,
    profile: userData.profile,
  }
  firebase
    .firestore()
    .collection('users') // The users collection, in
    .doc(userData.id) // In a document called <user ID>.
    .set(data) // Set the data
    .then(()) => {
      window.location.href = '/profile' // After completion, the user is taken to the profile screen
    })
}
```

- You can actually fill out your profile and press the "Register" button.

- Make sure your data is registered in Cloud Firestore.



- Let's show the registered data on the profile edit screen.
  - Change the `getUserData` as follows.

```
const getUserData = (user) => {
  firebase
    .firestore()
    .collection('users') // From the users collection, the
    .doc(user.uid) // The document with the specified uid is
    .get() // Get
    .then((doc) => {
      if (doc.exists) {
        userData.name = doc.data().name
        userData.role = doc.data().role
        userData.iconUrl = doc.data().iconUrl
        userData.profile = doc.data().profile
        userData.comment = doc.data().comment
      }
    })
    .catch((err) => {
      console.log('Error getting user document', err);
    })
}
```

- When you go to the Edit Profile screen, you will see the data you have already registered.

Display data on your profile screen

- Likewise, your profile page (http://localhost:3000/profile) should display the data you've already registered.
- The file for your profile page is `/pages/profile/index.vue`.
- You can modify the code on your own based on what you did in the profile edit screen.

- Answer is here. https://github.com/GuildWorks/titech-2020/blob/master/titech-nuxt-firebase-day3-answer/pages/profile/index.vue

Member list

- Let's make the member list page (http://localhost:3000/users) also show the registered data.
- The page file of the member list is `/pages/users/index.vue`.

- Import firebase.

```
import firebase from '@/plugins/firebase.ts'
```

- Please modify `setup()` as follows.

```
setup(_) {
  const userList = reactive<User[]>([])
  firebase
    .firestore()
    .collection('users') // From the users collection, the
    .get() // Get all the documents
    .then(function (querySnapshot) {
      querySnapshot.forEach(function (doc) { // For each element of the fetched document array
        userList.push({ // take the value of each field and add it to the userList array as an object
          id: doc.id,
          name: doc.data().name,
          email: doc.data().email,
          role: doc.data().role,
          iconUrl: doc.data().iconUrl,
          comment: doc.data().comment,
          profile: doc.data().profile,
        })
      })
    })
  const userLink = (userId: string): void => {
    window.location.href = '/users/' + userId
  }
  return {
    userList,
    userLink,
  }
},
```

- Make sure you see the users you have registered.

- Add users to see what it looks like if you have more than one user.
  - Log out and register a user with a different email address.
    - We don't check the existence of the email address, so you can register a user with an random email address.
    - However, it is preferable to use your own email address if possible because the email will be sent when you modify the app and add the email sending function in the future.

Reference

- Gmail's alias feature makes it convenient to use multiple email addresses in one email address.
  - If you have a `imahashi@gmail.com` account, you can add `+` and any alphanumeric characters before `@`, such as `imahashi+2@gmail.com` or `imahashi+third@gmail.com`, and mail will be sent to the same address.

- We can confirm that the added users are also shown.

Member profile

- Make sure that the registered data is also displayed on the member profile screen.
  - The user ID is required in the `_id` part of the path (http://localhost:3000/users/_id), so you need to display it from the user list screen.
- The file for the member profile screen is `/pages/users/_id.vue`.

- Import firebase.

```
import firebase from '@/plugins/firebase.ts'
```

- Please modify `setup()` as follows.

```
setup(_, { root }: SetupContext) {
  const userData = reactive<User>({
    id: '',
    name: '',
    email: '',
    role: '',
    iconUrl: '',
    comment: '',
    profile: {
      belongs: '',
      nickname: '',
      birthplace: '',
      birthday: '',
      bloodType: '',
      sign: '',
      hobby: '',
    },
  })
  firebase
    .firestore()
    .collection('users')
    .doc(root.$route.params.id) // Get the ID from the URL and get the document for that ID
    .get()
    .then((doc) => {
      if (doc.exists) {
        userData.id = root.$route.params.id
        userData.name = doc.data().name
        userData.email = doc.data().email
        userData.role = doc.data().role
        userData.iconUrl = doc.data().iconUrl
        userData.profile = doc.data().profile
        userData.comment = doc.data().comment
      }
    })
    .catch((err) => {
      console.log('Error getting document', err)
    })
  return {
    userData,
  }
},
```

- You can confirm that your member profile will be displayed.

Extra.

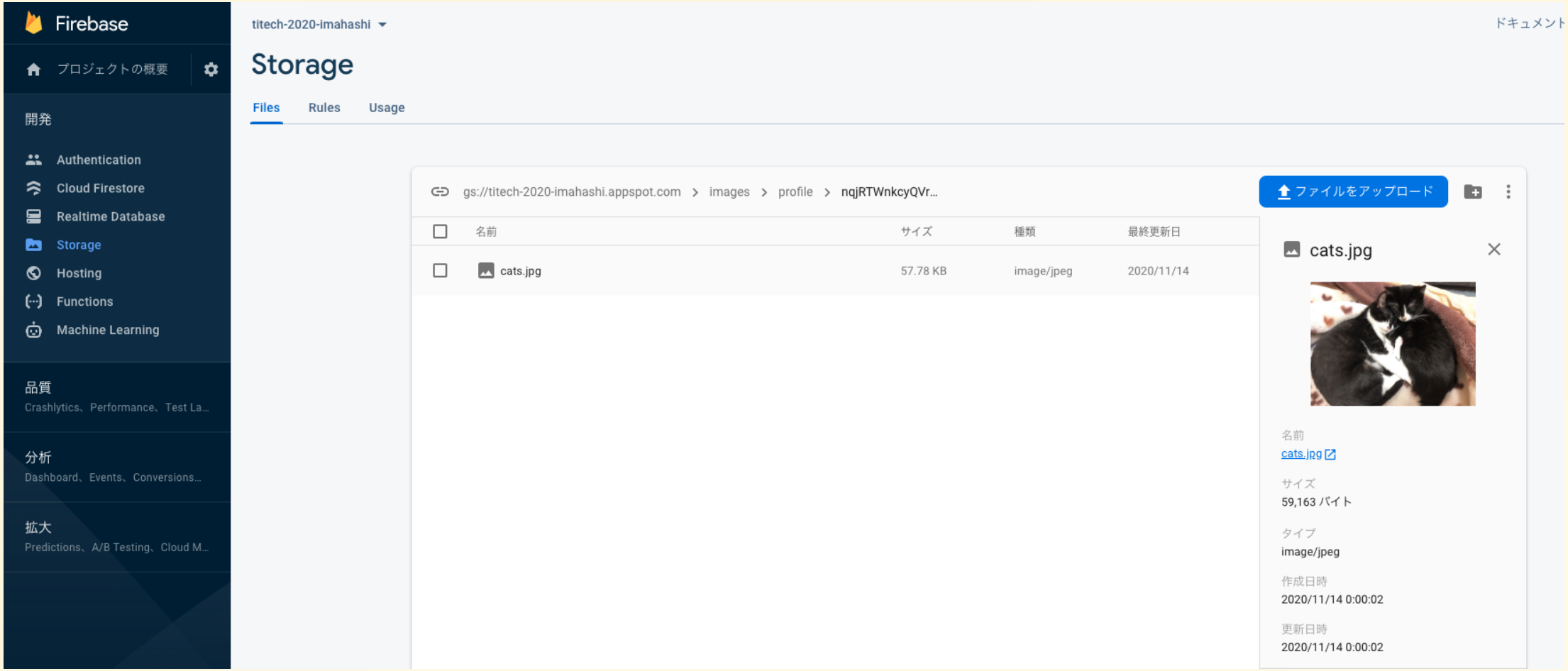
- Make sure you can upload your photos in your profile edit screen.
- Use Cloud Storage for Firebase.
  https://firebase.google.com/docs/storage
  - This service saves files such as photos and videos.
- Open `/pages/profile/edit.vue`.

- It is already possible to drag and drop an image on the icon part.
- Let's change the `setIcon` as follows.

```
const setIcon = (file: File): void => {
  const storageRef = firebase.storage().ref()
  // Get a reference to the profile image upload location
  const fileRef = storageRef.child(
    'images/profile/' + userData.id + '/' + file.name
  )
  // Upload your profile image to storage
  fileRef.put(file).then(function (snapshot) {
    // Update the URL of the user data
    snapshot.ref.getDownloadURL().then((url) => {
      userData.iconUrl = url
    })
  })
}
```

- After dragging and dropping the image, press the "Register" button to set the icon.

- When you open Cloud Storage, you can see that the images are stored.



Summary

✅ This time you can now use Firebase to register users, login, register/read/update data, etc.

✅ The complete app is in the `titech-nuxt-firebase-day3-answer` directory. If you missed something, please check here.

✅ If you understand the first three articles, you should be able to develop a simple app like this one.

If you're interested, feel free to customize your app or create your own!

Thank you for your hard work!