

คู่มือการสร้างเว็บ Book Market

ติดตั้ง Node.js ผ่านเว็บไซต์

เข้า command prompt ตรวจสอบ version ของ node.js กับ โปรแกรม npm

คำสั่งใช้ตรวจสอบ node -version

npm -version

เลือก folder หรือ สร้าง folder ที่ต้องการเก็บ project

คำสั่งใช้สร้าง folder: mkdir ชื่อโฟลเดอร์

```
D:\>mkdir Project_nui
```

```
D:\>dir
Volume in drive D is New Drive
Volume Serial Number is 52B0-5F39

Directory of D:\

04/05/2024  02:22 PM    <DIR>          bookmarket
11/09/2023  10:10 PM    <DIR>          Game
03/04/2024  09:07 PM    <DIR>          New folder
02/19/2024  02:03 AM    <DIR>          pro-coffee
01/09/2024  01:24 PM    <DIR>          Program Files
02/14/2024  04:40 PM    <DIR>          project-coffe
04/28/2024  07:08 PM    <DIR>          Project_nui
11/29/2023  08:56 AM    <DIR>          Riot Games
04/28/2024  06:25 PM    <DIR>          SteamLibrary
               0 File(s)                0 bytes
               9 Dir(s)  38,755,254,272 bytes free
```

สร้าง project ใน folder ที่เลือกหรือสร้าง

คำสั่ง vue create ชื่อproject (ชื่อต้องเป็นตัวเล็กทั้งหมด และต้องไม่มีช่องว่าง สามารถใช้ - ได้)

```
D:\Project_nui>vue create Nui_shop
Invalid project name: "Nui_shop"
Warning: name can no longer contain capital letters
```

เมื่อสร้าง Project ผ่านแล้ว จะมีให้เลือก Vue CLI

```
Vue CLI v5.0.8
? Please pick a preset: (Use arrow keys)
> Default ([Vue 3] babel, eslint)
  Default ([Vue 2] babel, eslint)
  Manually select features
```

เลือก Vue 2 (สามารถติดตั้งได้ทั้งสองตัว)

สามารถติดตั้งได้ทั้งสองอัน

```
Vue CLI v5.0.8
? Please pick a preset:
  Default ([Vue 3] babel, eslint)
> Default ([Vue 2] babel, eslint)
  Manually select features
```

จากนั้นกด Enter จะขึ้นหน้าจอ:

```
Vue CLI v5.0.8
🌟 Creating project in D:\Project_nui\nuishop.
📁 Initializing git repository...
⚙️ Installing CLI plugins. This might take a while...

added 878 packages in 3m

100 packages are looking for funding
  run `npm fund` for details
🚀 Invoking generators...
📦 Installing additional dependencies...

[████████████████████] / idealTree:nuishop: sill idealTree buildDeps
```

เมื่อติดตั้งเสร็จแล้วจะแสดงดังนี้:

```
Initializing git repository...
Installing CLI plugins. This might take a while...

added 878 packages in 3m

100 packages are looking for funding
  run `npm fund` for details
Invoking generators...
Installing additional dependencies...

added 94 packages in 23s

112 packages are looking for funding
  run `npm fund` for details
Running completion hooks...

Generating README.md...

Successfully created project nuishop.
Get started with the following commands:

$ cd nuishop
$ npm run serve

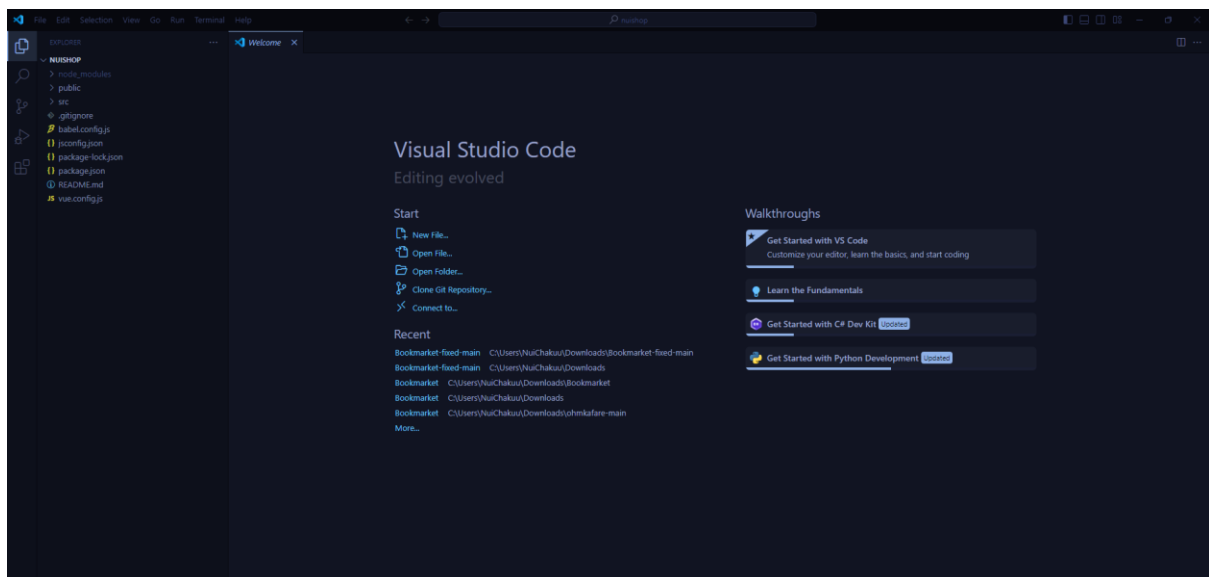
D:\Project_nui>
D:\Project_nui>
D:\Project_nui>
```

จากนั้นเข้า Project ที่สร้าง

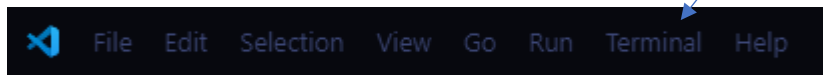
คำสั่ง: `cd` ชื่อ project ที่สร้าง

เปิด VS Code (Visual Studio Code) ผ่าน command prompt ในไฟล์ project

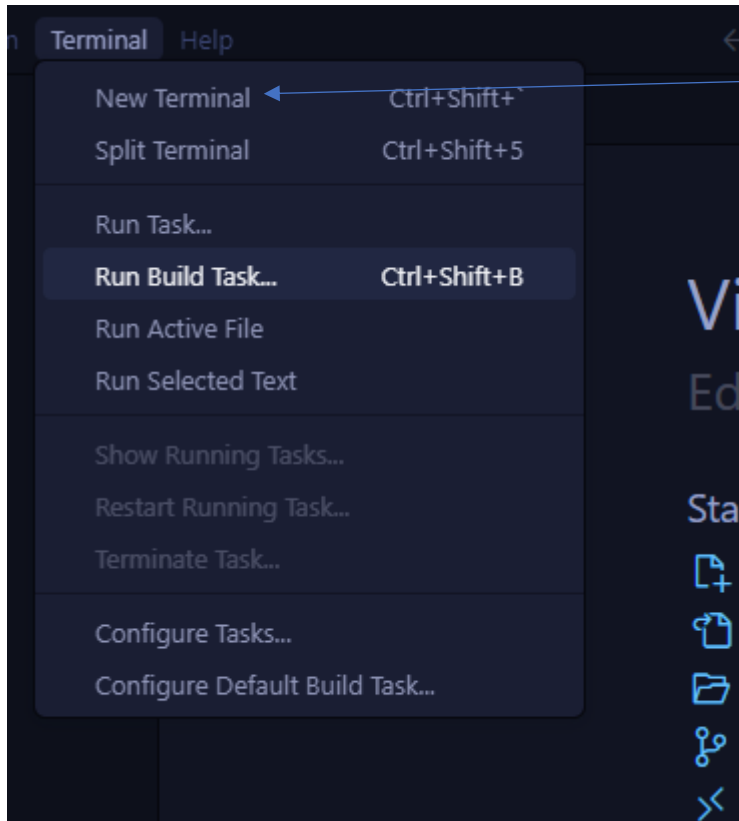
คำสั่ง: `code .`



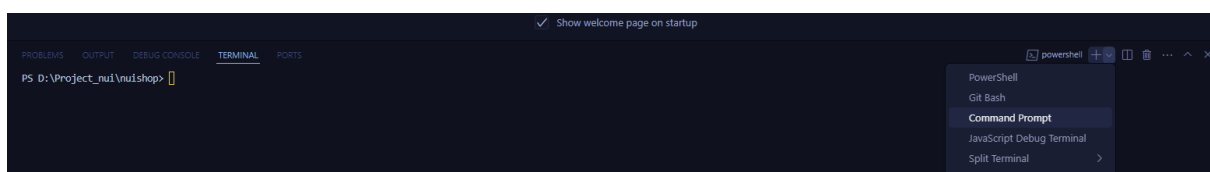
จากนั้นเปิด terminal ใน VS Code



เลือก new terminal



เปลี่ยน terminal จาก power shell เป็น cmd



เรียกใช้งานเว็บไซต์

คำสั่ง: `npm run serve`

ถ้าเว็บไซต์ถูกเรียกใช้งาน จะแสดง “App running at” ให้เอาค่าใน “App running at” ไปเปิดใน Browser

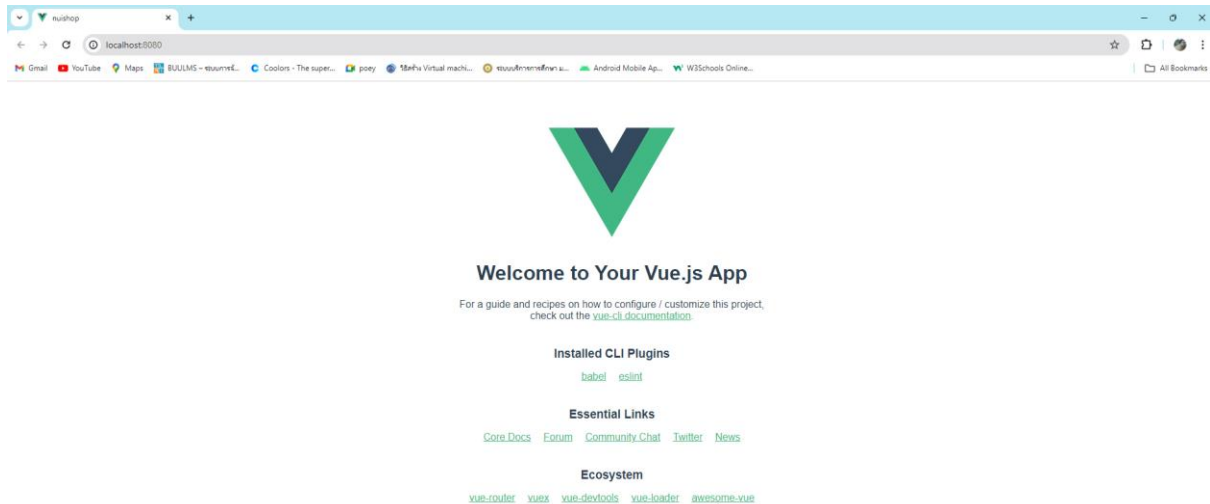
ตัวอย่างค่าใน App running at: Local: `http://localhost:8080/`

Network: `http://192.168.1.106:8080/`

```
App running at:
- Local:   http://localhost:8080/
- Network: http://192.168.1.106:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

หลังจากไปเปิดใน Browser แล้วจะแสดงดังนี้:



ทำการเพิ่มส่วนประกอบที่จำเป็นต้องใช้มี vuetify, vuex, vue-router

คำสั่งเพิ่ม vuetify :`vue add vuetify`

```
D:\Project_nui\nuishop>vue add vuetify
WARN There are uncommitted changes in the current repository, it's recommended to commit
or stash them first.
? Still proceed? (y/N) 
```

กด Y เพื่อทำการดาวน์โหลดต่อ

เมื่อดาวน์โหลดจะมี version ของ vuetify มาให้เลือก ให้โหลดให้ตรงกับ version ที่ดาวน์โหลด vue CLI

```
? Choose a preset: (Use arrow keys)
  Vuetify 2 - Configure Vue CLI (advanced)
> Vuetify 2 - Vue CLI (recommended)
  Vuetify 2 - Prototype (rapid development)
  Vuetify 3 - Vite (preview)
  Vuetify 3 - Vue CLI (preview)
```

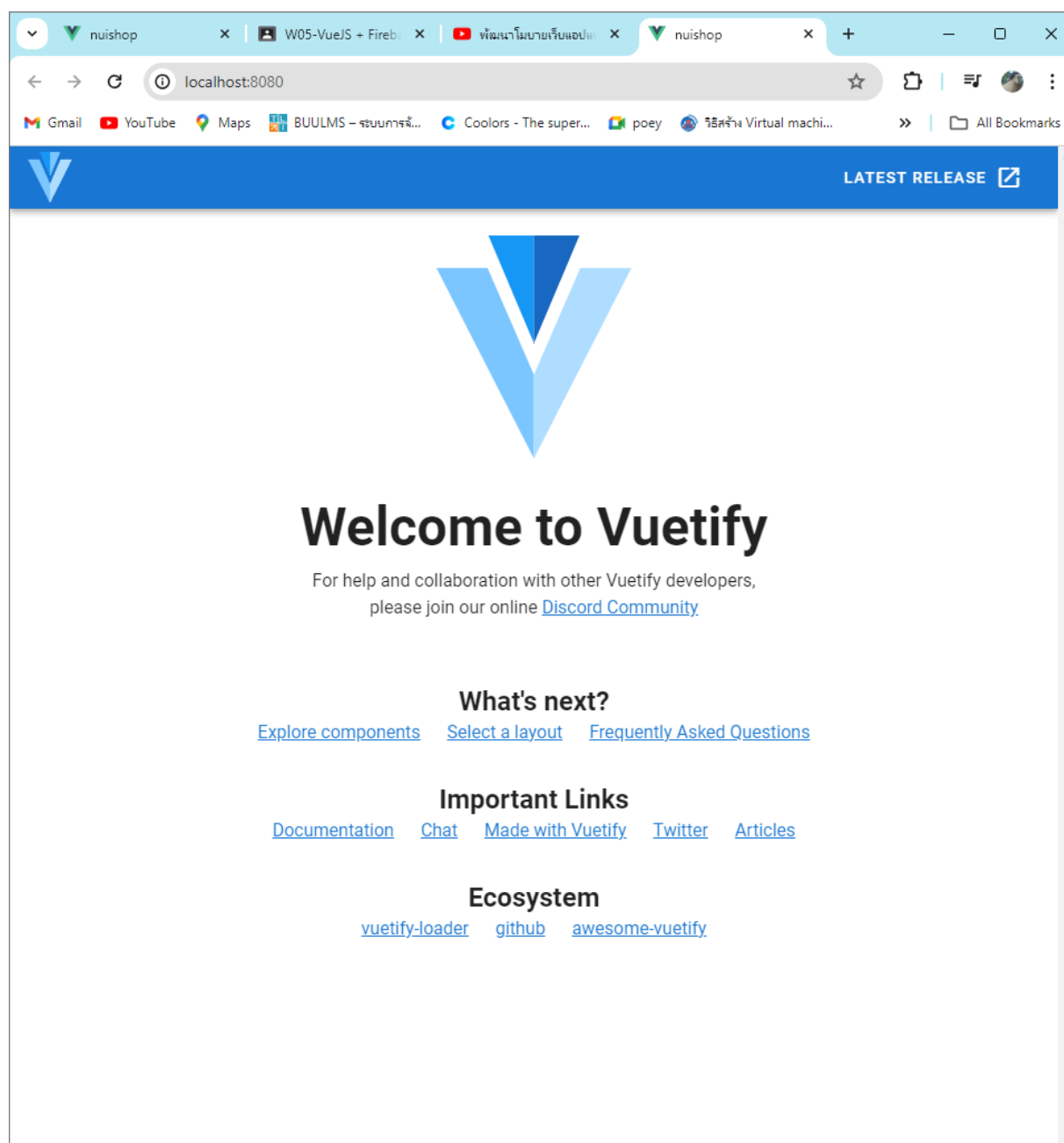
เมื่อติดตั้งเรียบร้อยแล้วจะแสดงดังนี้:

```
116 packages are looking for funding
  run `npm fund` for details
🔗 Running completion hooks...

✓ Successfully invoked generator for plugin: vue-cli-plugin-vuetify
vue-cli-plugin-vuetify: Discord community: https://community.vuetifyjs.com
vue-cli-plugin-vuetify: Github: https://github.com/vuetifyjs/vuetify
vue-cli-plugin-vuetify: Support Vuetify: https://github.com/sponsors/johnleider
```

เมื่อติดตั้งเสร็จ ใช้ `npm run serve` เพื่อตรวจสอบ

ต้องแสดงดังนี้:



ติดตั้ง vue-router

คำสั่ง: `npm install vue-router@3` ถ้าติดตั้งไม่ได้ให้ระบุ version ที่ต้องการติดตั้งด้วย

(โดยการ @ ตามด้วย version ตัวอย่าง: `npm install vue-router@3`)

(@3 สำหรับ Vue2) (@4 สำหรับ Vue3)

เมื่อติดตั้งเสร็จจะแสดงดังนี้:

```
added 1 package, and audited 999 packages in 4s

116 packages are looking for funding
  run `npm fund` for details

4 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

D:\Project_nui\nuishop>
```

ติดตั้ง vuex

คำสั่ง: `npm install vuex@3` ถ้าติดตั้งไม่ได้ให้ระบุ version ที่ต้องการติดตั้งด้วย

(โดยการ @ ตามด้วย version ตัวอย่าง: `npm install vuex@3`)

(@3 สำหรับ Vue2) (@4 สำหรับ Vue3)

เมื่อติดตั้งเสร็จจะแสดงดังนี้:

```
D:\Project_nui\nuishop>npm install vuex@3

added 1 package, and audited 1000 packages in 4s

116 packages are looking for funding
  run `npm fund` for details

4 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

D:\Project_nui\nuishop>
```

ติดตั้ง Firebase เพื่อใช้งาน Firestore ใช้เป็นฐานข้อมูล

คำสั่งใช้ติดตั้ง Firebase: `npm install firebase`

```
D:\Project_nui\nuishop>npm install firebase

up to date, audited 1064 packages in 5s

116 packages are looking for funding
  run `npm fund` for details

4 moderate severity vulnerabilities

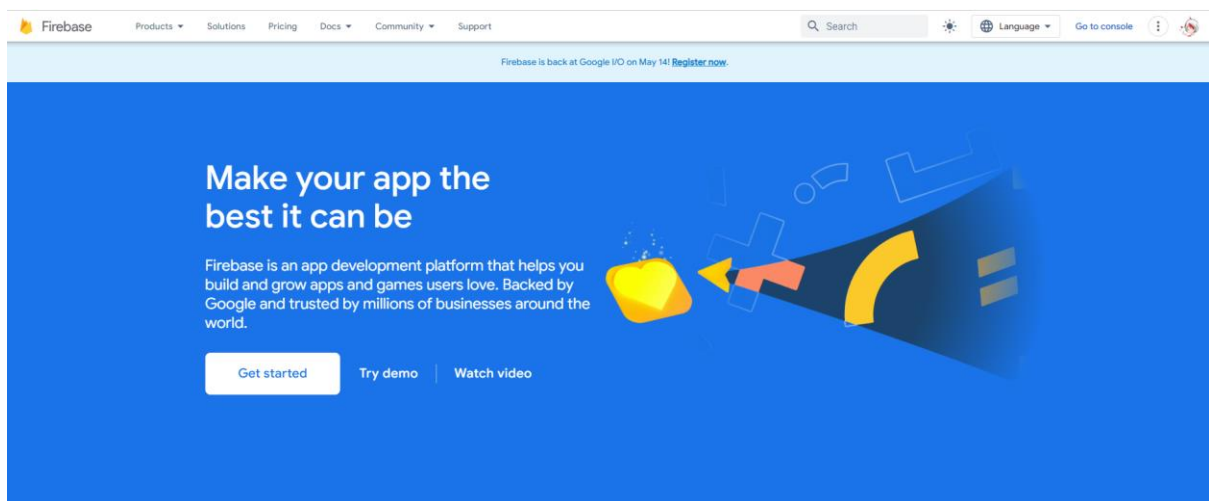
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

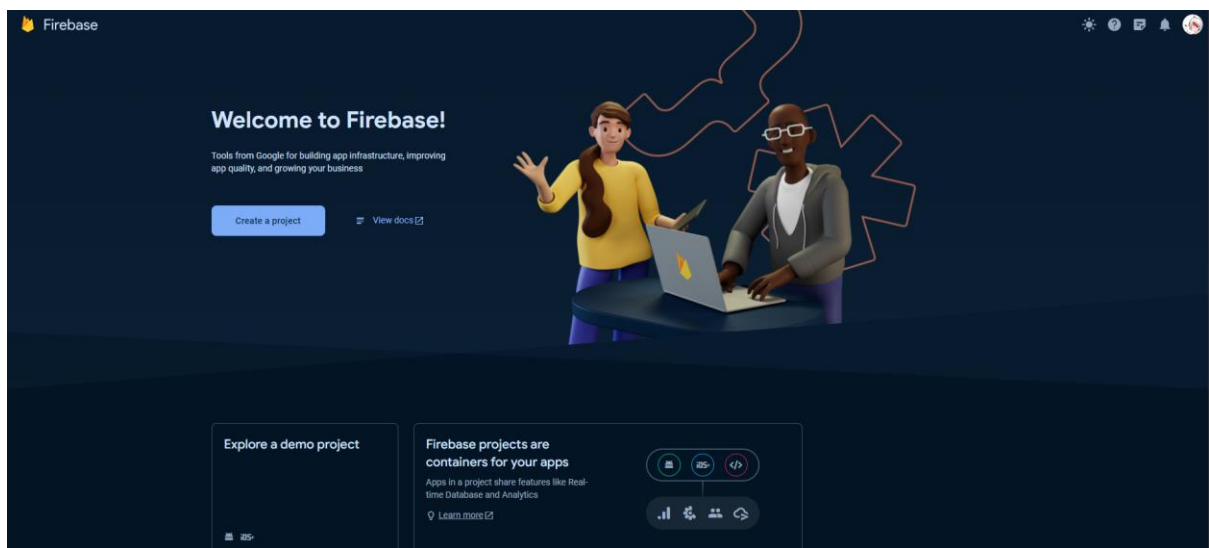
D:\Project_nui\nuishop>
```

สมัคร Firebase เพื่อใช้งาน

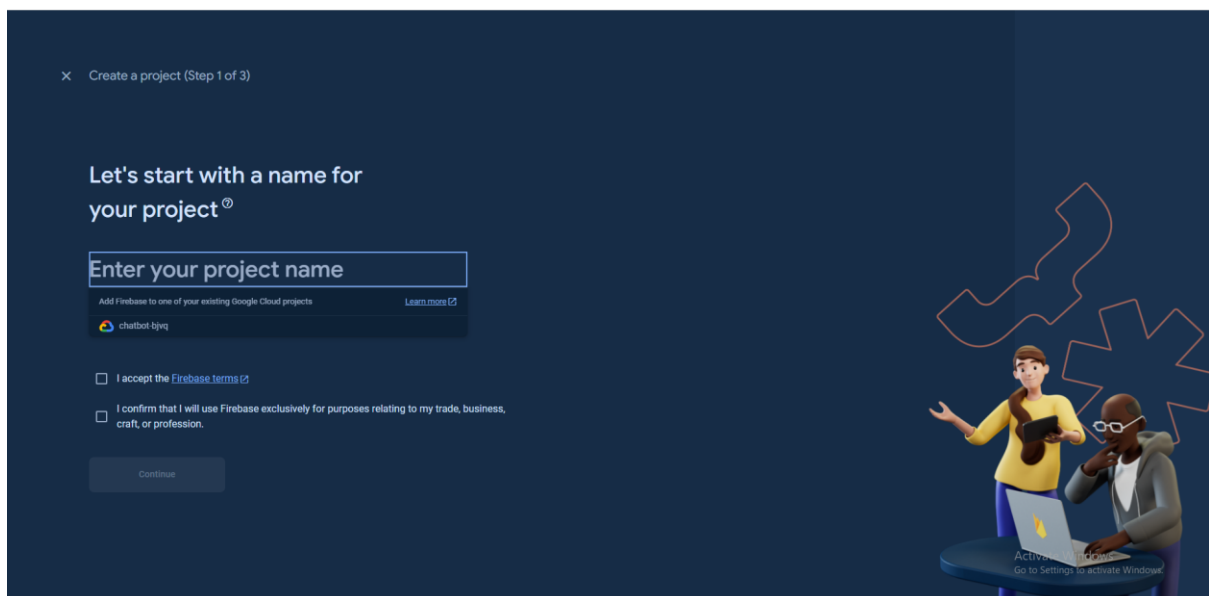
คลิก [Get started](#)



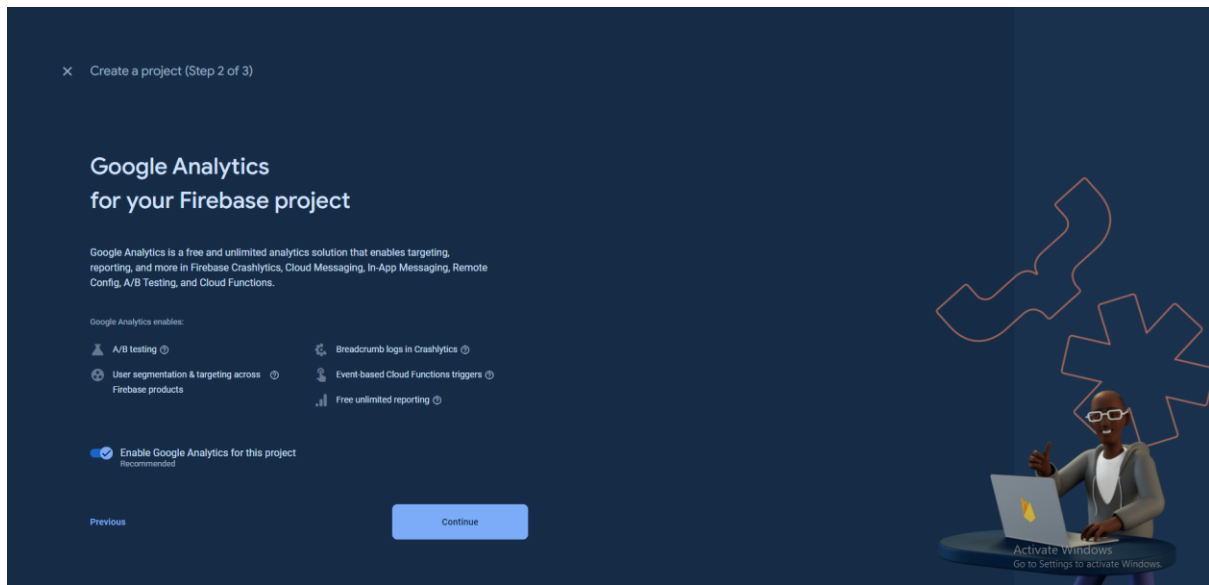
คลิก Create Project



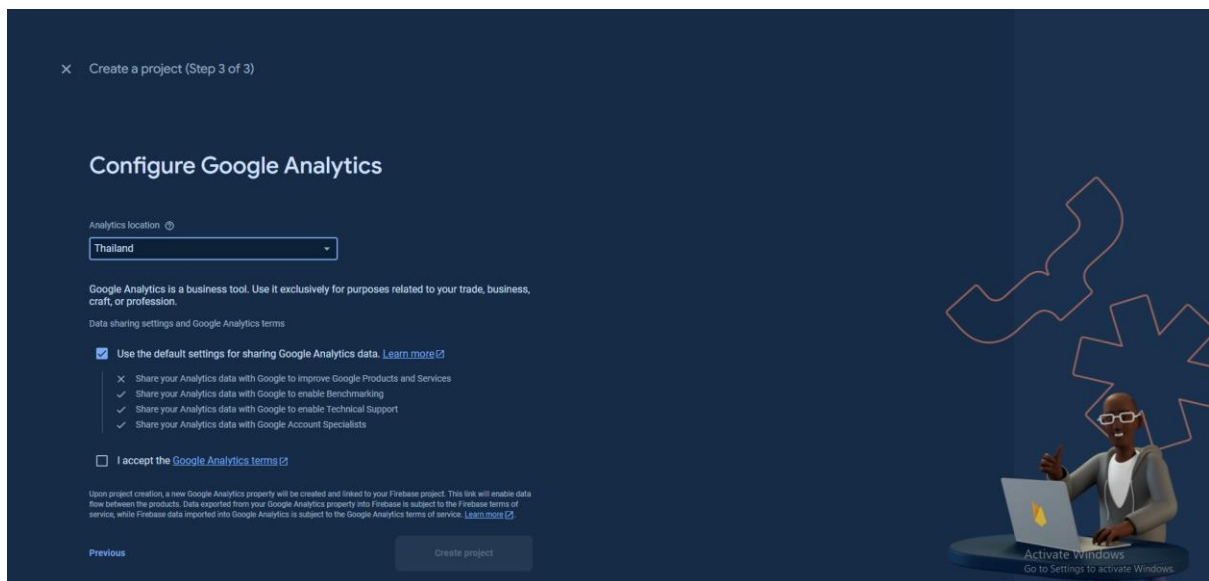
กรอกชื่อ Project



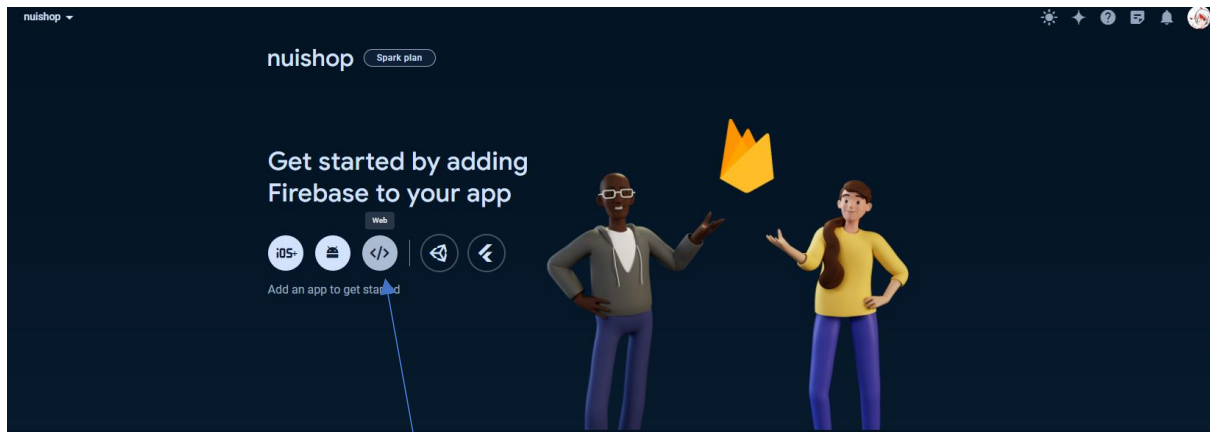
คลิก Continue



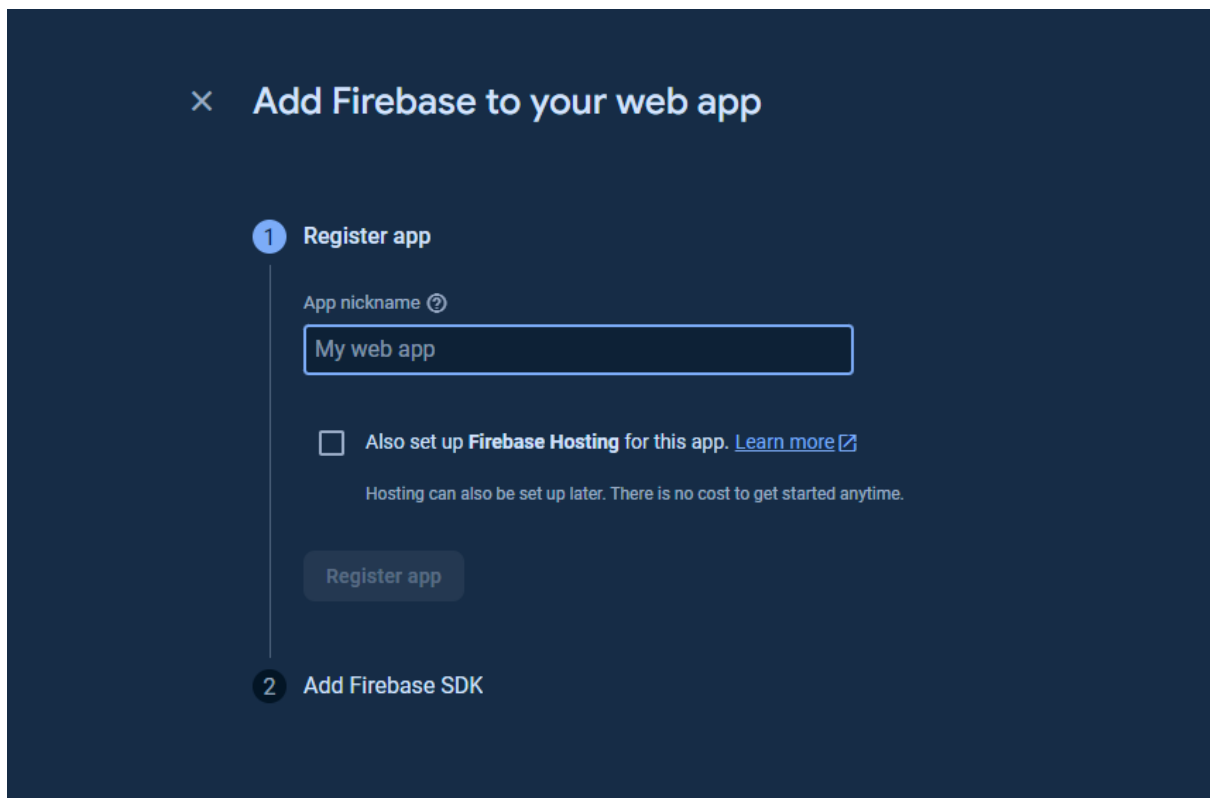
เลือก location ที่จะใช้ในการวิเคราะห์ จากนั้นกดติ๊กถูก และกด Create Project



จากนั้นคลิกที่ไอคอนตามลูกศร



กรอกชื่อ Web app ของตัวเอง



คัดลอกโค้ดในกรอบสีเหลี่ยมที่ลูกศรชี้มาใส่ไว้ในไฟล์ setting.js เพื่อสร้าง object เชื่อมต่อเว็บกับ Firebase จากคลิก Continue to Console

2 Add Firebase SDK

☒ Use npm ☐ Use a <script> tag

If you're already using [npm](#) and a module bundler such as [webpack](#) or [Rollup](#), you can run the following command to install the latest SDK ([Learn more](#)):

```
$ npm install firebase
```

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDugIqX_1J17Q0XZ1R9GuFgnbWeHxrnDzM",
  authDomain: "nuishop-bcb7d.firebaseio.com",
  projectId: "nuishop-bcb7d",
  storageBucket: "nuishop-bcb7d.appspot.com",
  messagingSenderId: "336703890708",
  appId: "1:336703890708:web:19b9064cb3de96a5a3ce01",
  measurementId: "G-2FNBY8LRZZ"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Note: This option uses the [modular JavaScript SDK](#), which provides reduced SDK size.

Learn more about Firebase for web: [Get Started](#), [Web SDK API Reference](#), [Samples](#)

Continue to console

จากนั้นสร้างไฟล์ setting.js เพื่อสร้าง object กำหนดค่าของ firebase

ตัวอย่าง:

```
1 export default{
2   firebaseConfig : {
3     apiKey: "AIzaSyDqsp13hqPtKSt9pZo800dvgBXihHMzDsw",
4     authDomain: "bookmarket-706cb.firebaseio.com",
5     projectId: "bookmarket-706cb",
6     storageBucket: "bookmarket-706cb.appspot.com",
7     messagingSenderId: "337816208269",
8     appId: "1:337816208269:web:9b4ae1783b85ef179573c4",
9   }
10 }
11 }
```

สร้าง route.js กำหนดเส้นทางในการไปยังหน้าต่างๆของตัวเว็บไซต์

```
src > JS routes.js > routes
1 import Vue from "vue";
2 import VueRouter from "vue-router";
3
4 import Home from "../components/Home.vue";
5 import Order from "../components/Order.vue";
6 import Checkout from "../components/Checkout.vue";
7 import CheckoutProcess from "../components/CheckoutProcess.vue";
8 import Signin from "../components/Signin.vue";
9 import ShowD from "../components/ShowD.vue";
10
11 Vue.use(VueRouter);
12
13 const routes = [
14   {
15     path: "/",
16     component: Home
17   },
18   {
19     path: "/showdata",
20     component: ShowD
21   },
22   {
23     path: "/order/:id",
24     component: Order
25   },
26   {
27     path: "/checkout",
28     component: Checkout
29   },
30   {
31     path: "/checkout-process",
32     component: CheckoutProcess
33   },
34   {
35     path: "/signin",
36     component: Signin
37   }
38 ];
39
40 const router = new VueRouter({
41   routes
42 });
43
44 export default router;
45
```

หน้า Home ("/): เมื่อผู้ใช้เข้าสู่ URL หลัก จะแสดง Home.vue

หน้าแสดงข้อมูล: เมื่อผู้ใช้เข้าสู่ URL `"/showdata"` จะแสดง `ShowD.vue`

หน้า Order: เมื่อผู้ใช้เข้าสู่ URL `"/order/:id"` จะแสดง `Order.vue` เพื่อแสดงรายละเอียดของคำสั่งซื้อ

หน้าชำระสินค้า : เมื่อผู้ใช้เข้าสู่ URL `"/checkout"` จะแสดง `Checkout.vue` เพื่อแสดงสรุปของคำสั่งซื้อ

หน้าชำระสินค้าเสร็จสิ้น : เมื่อผู้ใช้เข้าสู่ URL `"/checkout-process"` จะแสดง `CheckoutProcess.vue` เพื่อแสดงข้อความยืนยันหลังการชำระเงินเสร็จสิ้น

หน้าล็อกอิน : เมื่อผู้ใช้เข้าสู่ URL `"/signin"` จะแสดง `Signin.vue` เพื่อให้ผู้ใช้เข้าสู่ระบบหรือลงทะเบียน

ส่งออก object ของ router ในไฟล์นี้เพื่อให้สามารถนำไปใช้งานได้ไฟล์อื่น ๆ

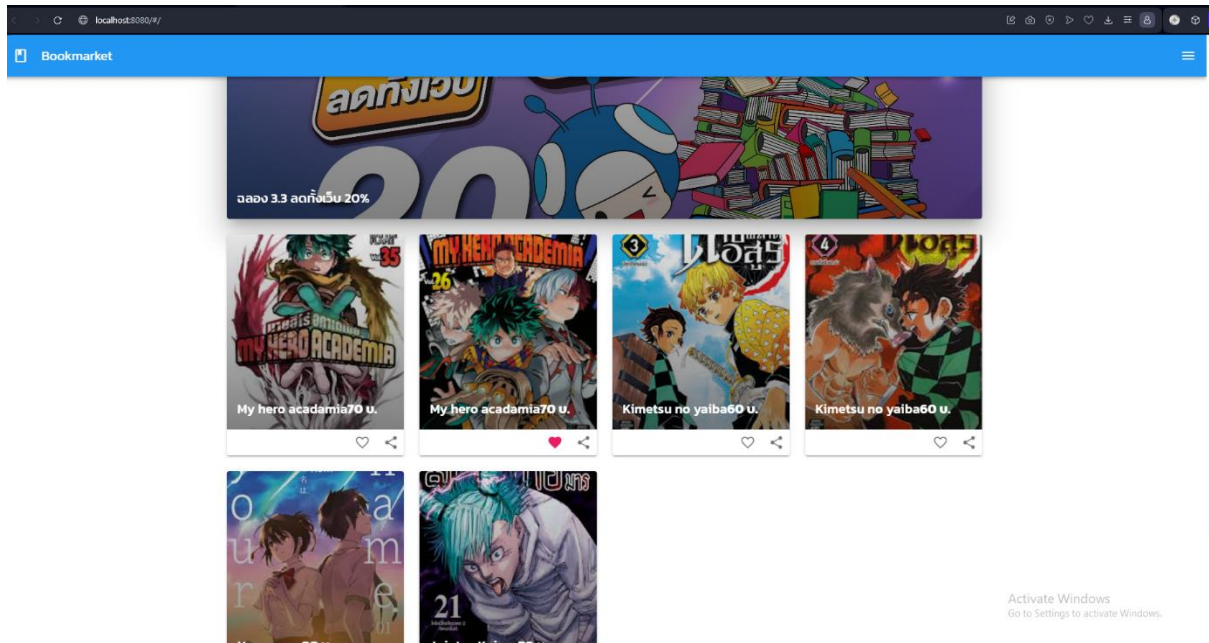
สร้าง `main.js` กำหนดเส้นทางในการไปยังหน้าต่างๆของตัวเว็บไซต์

```
src > JS main.js > ...
1  import Vue from 'vue'
2  import App from './App.vue'
3  import vuetify from './plugins/vuetify';
4  import router from './routes.js';
5  import store from './store.js';
6  import firebase from "firebase/app";
7  import "firebase/firestore";
8  import settings from './settings.js';
9
10  firebase.initializeApp(settings.firebaseConfig);
11
12  Vue.config.productionTip = false
13
14  new Vue({
15    vuetify,
16    router,
17    store,
18    render: h => h(App)
19  }).$mount('#app')
```

โค้ดด้านบนเป็นการนำเข้าและใช้งานโมดูลและคอมโพเนนต์ต่างๆ ใน Project ซึ่งรวมถึงการเชื่อมต่อกับ Firebase เพื่อใช้งานบริการต่างๆ

ส่วนของหน้าการทำงานต่างๆของเว็บไซต์

App.vue จะเป็นหน้าหลักในการเรียกตัว component โดยที่โค้ดทั้งหมดของไฟล์ App.vue มาแสดงเป็นหน้าหลักของตัวเว็บไซต์



ตัวอย่างโค้ด:

```
1 <template>
2   <v-app>
3     <v-app-bar app color="blue" dark>
4       <v-icon class="mr-5">mdi-book-outline</v-icon>
5       <v-toolbar-title>Bookmarket</v-toolbar-title>
6
7       <v-spacer></v-spacer>
8
9       <v-menu>
10        <template v-slot:activator="{ on }">
11          <v-btn icon v-on="on">
12            <v-icon>mdi-menu</v-icon>
13          </v-btn>
14        </template>
15
16        <v-list>
17          <v-list-item to="/showdata">
18            <v-list-item-title>ข้อมูลการสั่งซื้อ</v-list-item-title>
19          </v-list-item>
20          <v-divider></v-divider>
21          <v-list-item @click="signout">
22            <v-list-item-title class="red--text">ออกจากระบบ</v-list-item-title>
23          </v-list-item>
24        </v-list>
25      </v-menu>
26    </v-app-bar>
27
28    <v-content>
29      <router-view></router-view>
30    </v-content>
31  </v-app>
32</template>
```

โค้ดด้านบนเป็นโครงสร้างหน้าเว็บไซต์ใช้ Vuetify ในการสร้าง:

<template>: ส่วนนี้ใช้ในการเขียนโค้ด

<v-app>: ใช้ในการรองรับแอปพลิเคชันทั้งหมด โดยใช้เป็นคอนเทนเนอร์สำหรับส่วนอื่น ๆ ของแอปพลิเคชัน

<v-app-bar>: เป็นแถบเมนูบนสุดของเว็บไซต์ ซึ่งมีความสำคัญในการแสดงชื่อแอปพลิเคชันและเมนูการนำทาง

app: แสดงว่าแถบเมนูนี้เป็นส่วนหนึ่งของแอปพลิเคชัน

color="blue" dark: ตั้งค่าสีพื้นหลังเป็นสีน้ำเงินและเนื้อที่ของแถบเป็นสีเข้ม

<v-icon class="mr-5">mdi-book-outline</v-icon>: ไอคอนหนังสือแสดงไอคอนในแถบเมนู

<v-toolbar-title>: แสดงชื่อแอปพลิเคชัน

<v-spacer></v-spacer>: สร้างพื้นที่ว่างทางขวาของแถบเมนูเพื่อจัดหน้าเมนู

<v-menu>: เมนูที่มีไอเท็มในเมนูดรอปดาวน์

<v-btn icon>: ปุ่มเปิดเมนูดรอปดาวน์

<v-list>: รายการเมนูที่มีไอเท็มในเมนู

<v-list-item to="/showdata">: เมนูที่นำไปยังเส้นทาง /showdata ในการนำทาง Vue Router เพื่อดูข้อมูลการสั่งซื้อ

<v-list-item @click="signout">: เมนูที่ใช้ในการออกจากระบบ โดยเรียกใช้เมทอด signout

<v-content>: แสดงเนื้อหาหลักของแอปพลิเคชัน โดยใช้คอมโพเนนต์ <router-view> เพื่อแสดงหน้าเว็บที่เกี่ยวข้องกับเส้นทางที่เลือกผ่าน Vue Router ภายในโครงสร้างของเว็บไซต์

ส่วน script ใน App.vue

```
35 <script>
36 import firebase from "firebase/app";
37 import "firebase/auth";
38 import "firebase/firestore";
39
40 export default {
41   name: "App",
42   data() {
43     return {
44       orders: []
45     };
46   },
47   mounted() {
48     firebase.auth().onAuthStateChanged(user => {
49       if (user == null) this.$router.replace("/signin");
50       else this.$router.replace("/");
51     });
52
53     // Retrieve orders from Firestore
54     this.fetchOrders();
55   },
56   methods: {
57     signout() {
58       firebase.auth().signOut();
59     },
60     fetchOrders() {
61       firebase
62         .firestore()
63         .collection("order")
64         .get()
65         .then(querySnapshot => {
66           querySnapshot.forEach(doc => {
67             this.orders.push(doc);
68           });
69         })
70         .catch(error => {
71           console.error("Error fetching orders: ", error);
72         });
73     }
74   }
75 };
76 </script>
```

Import ส่วนเพิ่มเติมที่เกี่ยวกับ Firebase ที่จำเป็นต้องใช้

การตรวจสอบสถานะการเข้าสู่ระบบ: ใน `mounted()` มีการใช้

`firebase.auth().onAuthStateChanged()` เพื่อตรวจสอบสถานะการเข้าสู่ระบบของผู้ใช้ โดยหากผู้ใช้ไม่ได้เข้าสู่ระบบ (`user == null`) จะถูกนำไปยังหน้าเข้าสู่ระบบ (`/signin`) และหากผู้ใช้เข้าสู่ระบบแล้วจะถูกนำไปยังหน้าหลัก (`/`)

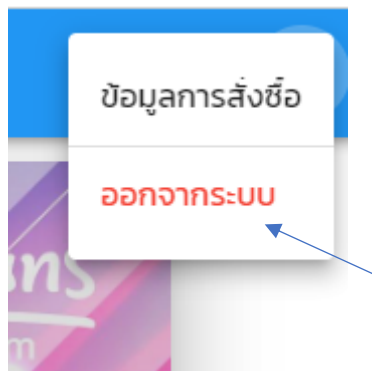
การดึงข้อมูลการสั่งซื้อ: ใน `fetchOrders()` มีการใช้ `firebase.firestore().collection("order").get()` เพื่อดึงข้อมูลการสั่งซื้อจาก Firestore โดยที่ข้อมูลนี้จะถูกนำมาแสดงผลในตารางในหน้าเว็บแอปพลิเคชัน

การออกจากระบบ: ใน `signout()` มีการใช้ `firebase.auth().signOut()` เพื่อให้ผู้ใช้ออกจากระบบ

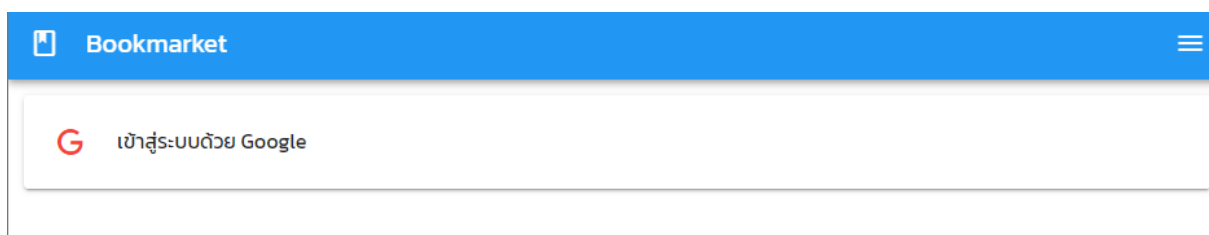
สามารถออกจากระบบเพื่อทำการ **signin** เข้ามาใช้งานได้



คลิก “ออกจากระบบ” ตามที่ลูกศรชี้

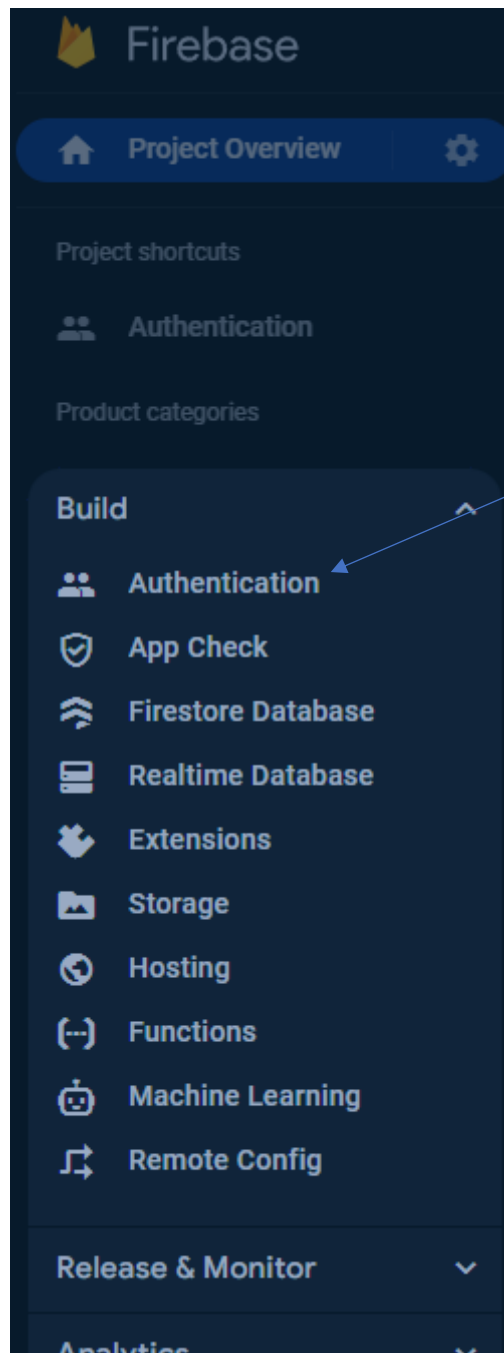


หน้า Signin.vue



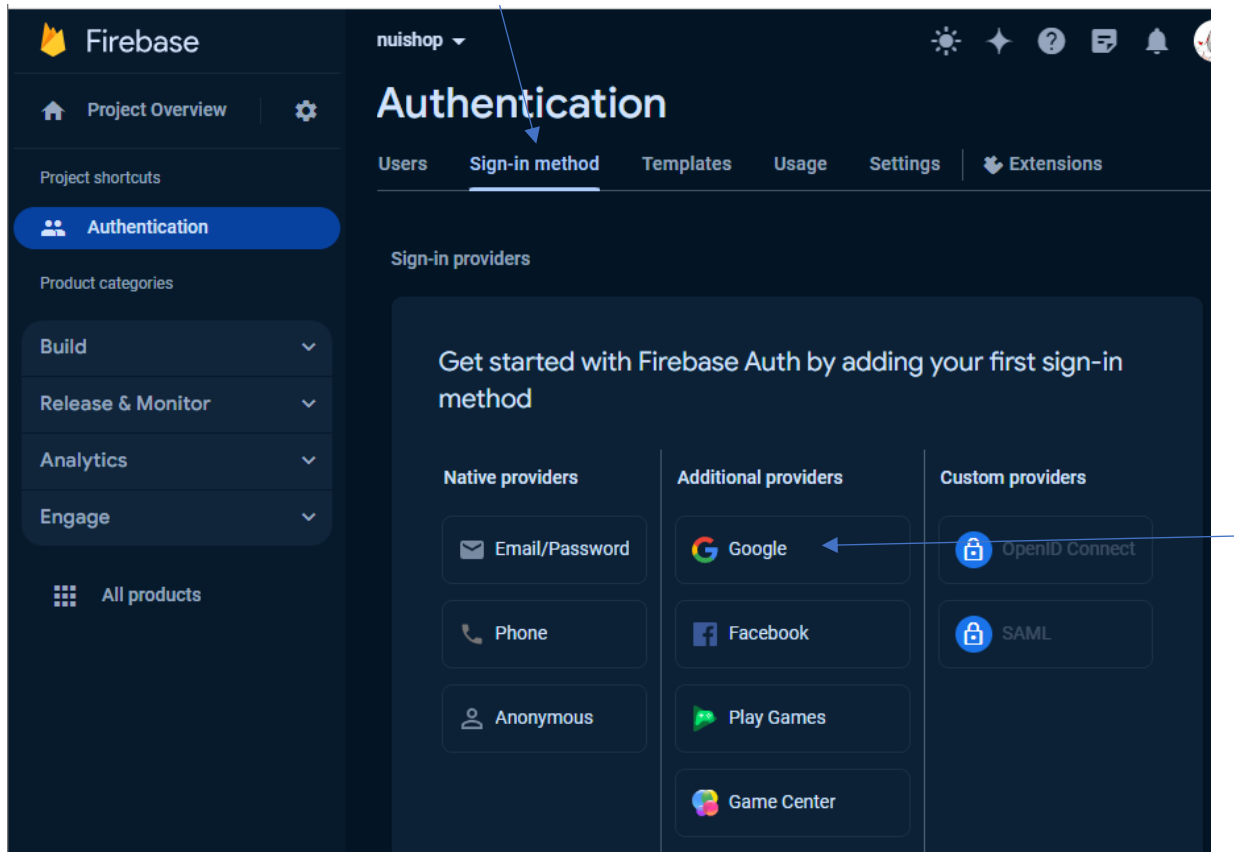
ต้องไปตั้งค่าการ authentication ใน firebase ก่อน

คลิกเลือก “Authentication” ตามลูกศร

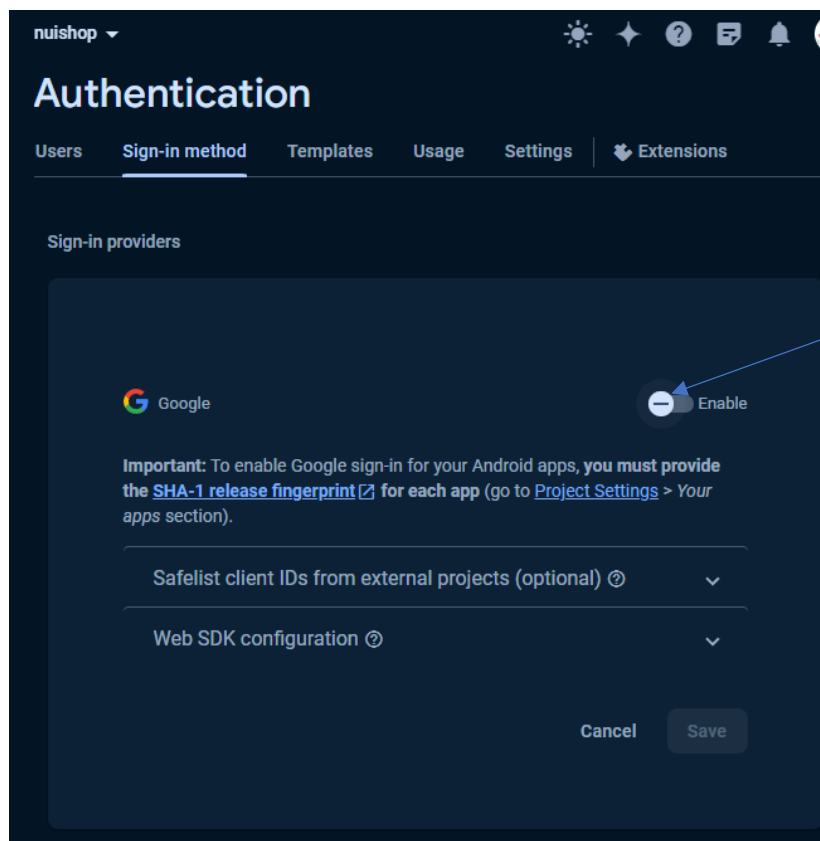


จากนั้นเลือก “sign-in method” ตามที่ลูกศรชี้

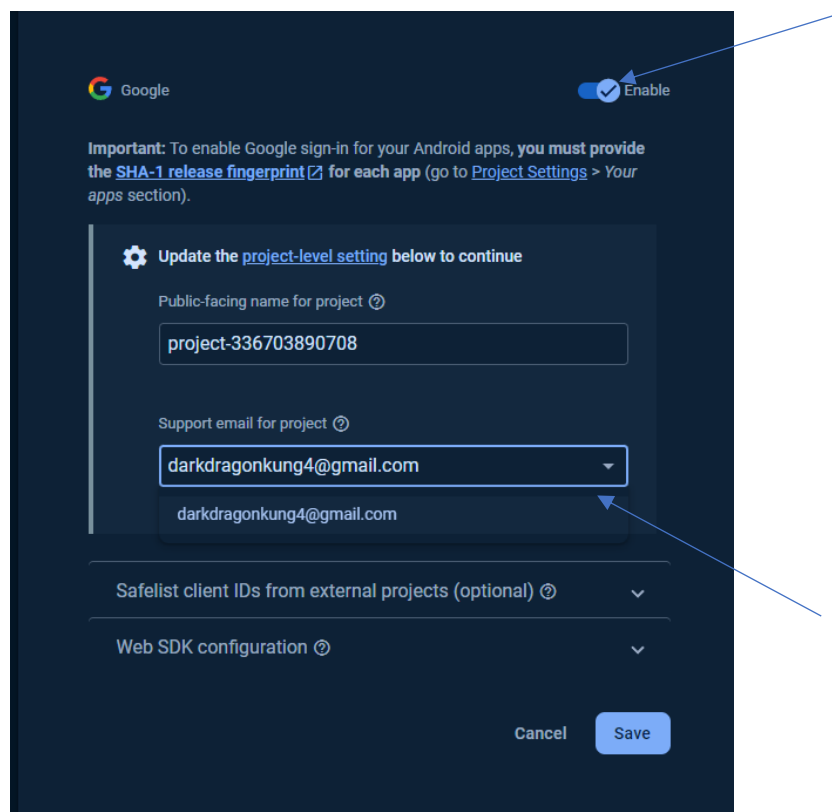
จากนั้นเลือก Google ตามที่ลูกศรชี้



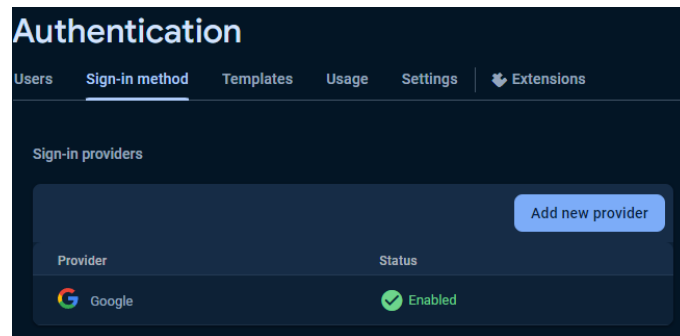
จากนั้นคลิกที่ “Enable” ตามลูกศรชี้



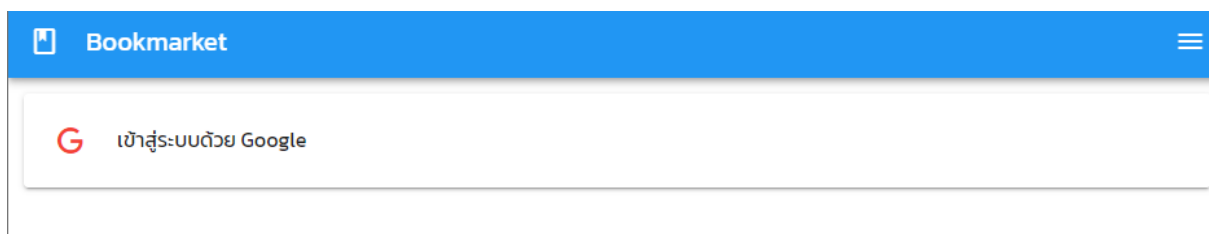
เลือกอีเมลที่จะใช้ใน Project ให้เรียบร้อย หลังจากนั้นก็กด save



ถ้าขึ้นภาพดังนี้แสดงว่าถูกต้อง:



เท่านี้ก็จะสามารถล็อกอินผ่าน Google ได้



ส่วนโค้ดที่ใช้แสดงตัวหน้าล็อกอิน

```
1 <template>
2   <v-container>
3     <div v-if="loading" class="text-center">
4       <v-progress-circular size="50" color="red" indeterminate></v-progress-circular>
5       <p>กำลังดำเนินการ...</p>
6     </div>
7
8     <v-card v-else>
9       <v-list>
10        <v-list-item @click="signinByGoogle">
11          <v-list-item-avatar>
12            <v-icon color="red">mdi-google</v-icon>
13          </v-list-item-avatar>
14          <v-list-item-title>เข้าสู่ระบบด้วย Google</v-list-item-title>
15        </v-list-item>
16      </v-list>
17    </v-card>
18  </v-container>
19 </template>
```

แสดงผลดังนี้:

<v-container>: ใช้เพื่อกำหนดความกว้างของเนื้อหาภายใน

`v-if="loading"`: เงื่อนไขที่ใช้เพื่อตรวจสอบว่ากำลังโหลดหรือไม่ ถ้ากำลังโหลดอยู่จะแสดงข้อความว่า "กำลังดำเนินการ..."

`<v-progress-circular>`: ใช้แสดงการโหลดข้อมูลในรูปแบบวงกลม ตอนกำลังโหลด

`v-else`: ถ้าไม่ใช้การโหลดข้อมูล จะแสดงเนื้อหาภายใน `<v-card>`

`<v-list>`: แสดงรายการข้อความ

`<v-list-item @click="signinByGoogle">`: เมื่อคลิกที่รายการนี้ จะเรียกใช้เมธอด `signinByGoogle` เพื่อเข้าสู่ระบบด้วยบัญชี Google

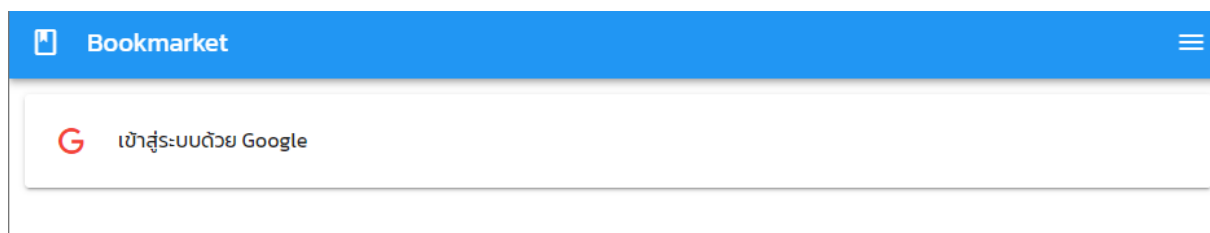
```
methods: {  
  signinByGoogle() {  
    const provider = new firebase.auth.GoogleAuthProvider();  
    firebase.auth().signInWithRedirect(provider);  
  }  
};  
</script>
```

`signinByGoogle()` มีวัตถุประสงค์เพื่อเริ่มกระบวนการเข้าสู่ระบบด้วยบัญชี Google ในเว็บไซต์โดยใช้ Firebase Authentication

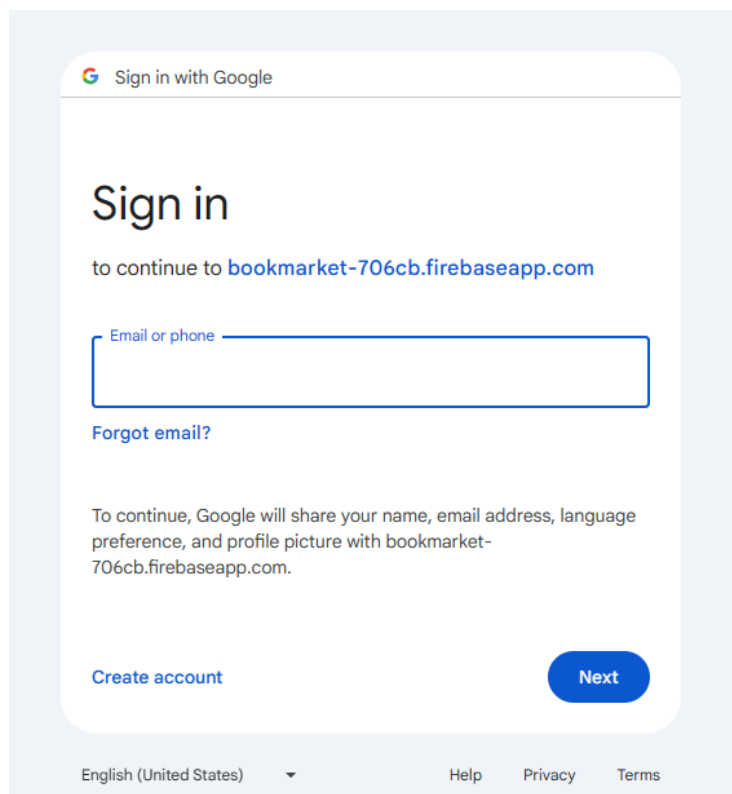
`<v-list-item-avatar>`: ใช้สำหรับแสดงไอคอนของ Google

`<v-list-item-title>`: แสดงข้อความ "เข้าสู่ระบบด้วย Google"

เมื่อคลิก "เข้าสู่ระบบด้วย Google"



จากนั้นจะขึ้นหน้า Sign in ของ google มาดังภาพ:



Sign in with Google

Sign in

to continue to bookmarket-706cb.firebaseio.com

Email or phone

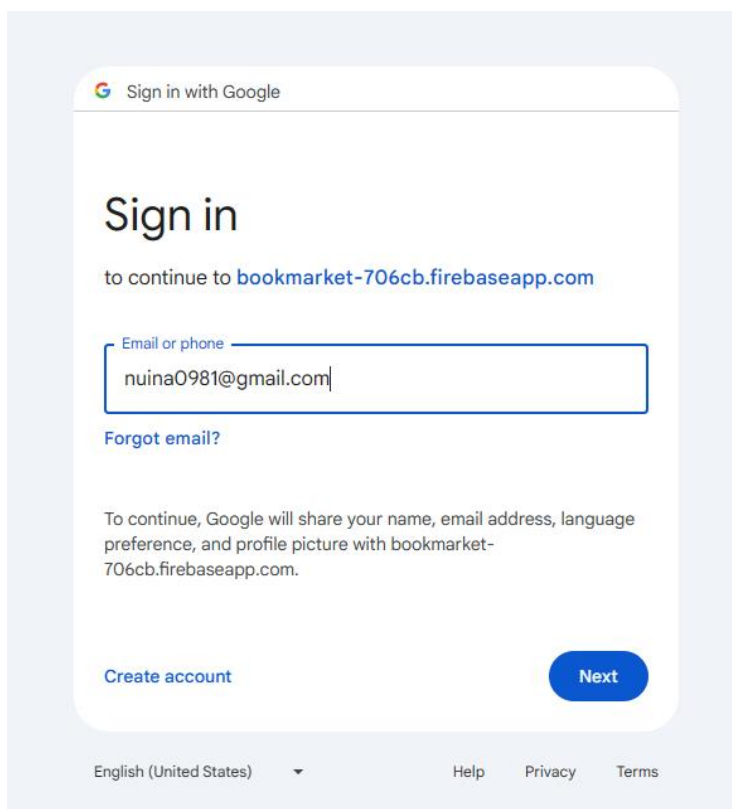
[Forgot email?](#)

To continue, Google will share your name, email address, language preference, and profile picture with bookmarket-706cb.firebaseio.com.

[Create account](#) [Next](#)

English (United States) [Help](#) [Privacy](#) [Terms](#)

กรอกอีเมลให้เรียบร้อย จากนั้นคลิก “Next”



Sign in with Google

Sign in

to continue to bookmarket-706cb.firebaseio.com

Email or phone

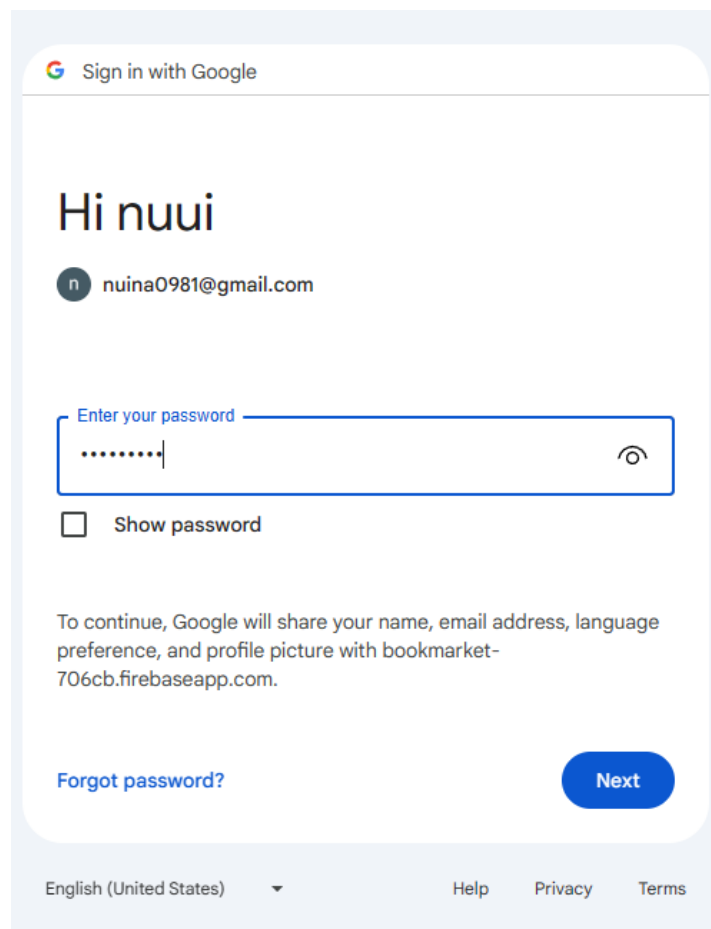
[Forgot email?](#)

To continue, Google will share your name, email address, language preference, and profile picture with bookmarket-706cb.firebaseio.com.

[Create account](#) [Next](#)

English (United States) [Help](#) [Privacy](#) [Terms](#)

กรอกพาสเวิร์ดให้เรียบร้อย จากนั้นคลิก “Next”



Sign in with Google

Hi nuui

nuina0981@gmail.com

Enter your password

.....

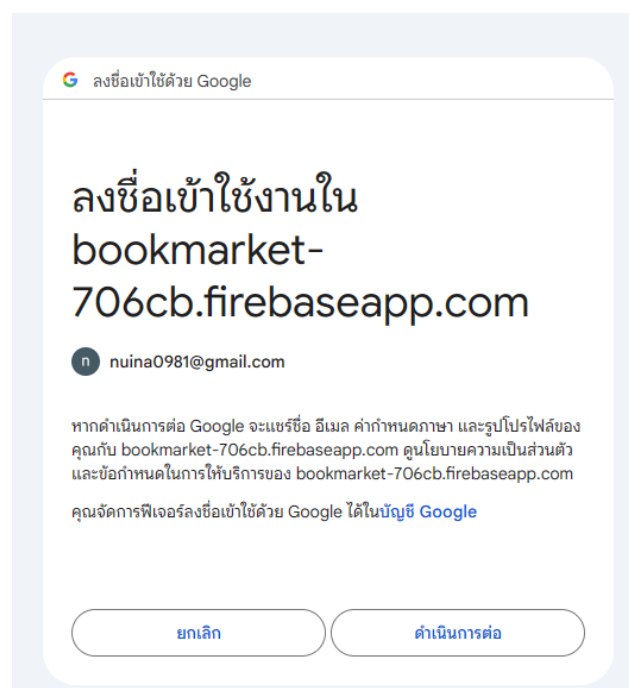
☐ Show password

To continue, Google will share your name, email address, language preference, and profile picture with bookmark-706cb.firebaseio.com.

[Forgot password?](#) [Next](#)

English (United States) Help Privacy Terms

คลิก “ดำเนินการต่อ”



ลงชื่อเข้าใช้ด้วย Google

ลงชื่อเข้าใช้งานใน
bookmark-
706cb.firebaseio.com

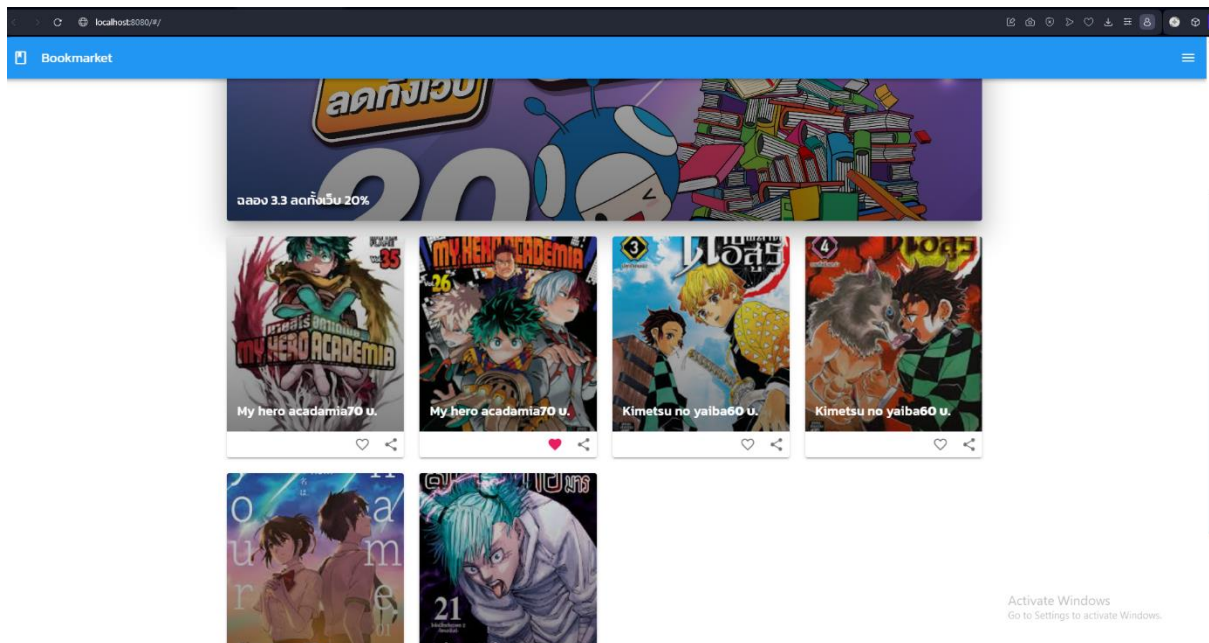
nuina0981@gmail.com

หากดำเนินการต่อ Google จะแชร์ชื่อ อีเมล คำกำหนดภาษา และรูปโปรไฟล์ของคุณกับ bookmark-706cb.firebaseio.com คุณนโยบายความเป็นส่วนตัว และข้อกำหนดในการให้บริการของ bookmark-706cb.firebaseio.com

คุณจัดการฟิเจอร์ลงชื่อเข้าใช้ด้วย Google ได้ใน [บัญชี Google](#)

[ยกเลิก](#) [ดำเนินการต่อ](#)

หลังจากนั้นก็เข้ามาสู่หน้าเว็บไซต์



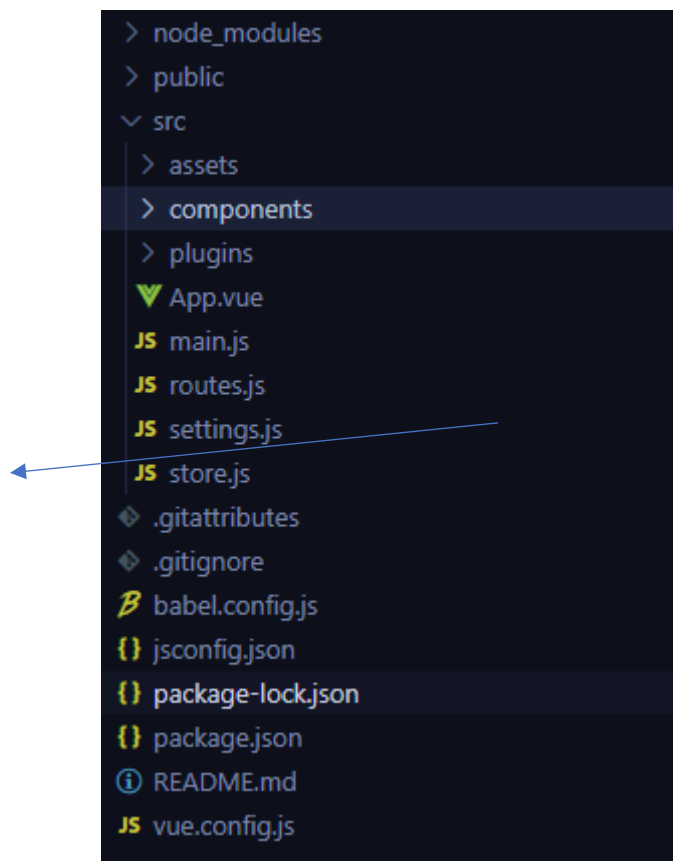
Script เพิ่มเติม หน้า Signin.vue

```
<script>
import firebase from "firebase/app";
import "firebase/auth";
import 'firebase/firestore';
export default {
  data() {
    return {
      loading: true
    };
  },
  mounted() {
    firebase
      .auth()
      .getRedirectResult()
      .then(result => {
        if (result.user == null) {
          this.loading = false;
        } else {
          this.$router.replace("/");
        }
      });
  }
};
```

Import Firebase เพื่อใช้งาน Firebase Authentication และ Firebase Firestore

หลังจากล็อกอินด้วย Firebase Authentication. โค้ดจะทำการตรวจสอบว่ามีการเข้าสู่ระบบหรือไม่ ถ้าไม่มีการเข้าสู่ระบบ โค้ดจะกำหนดค่า loading เป็น false แสดงว่าไม่มีการโหลด แต่ถ้ามีผู้ใช้เข้าสู่ระบบแล้ว โค้ดจะใช้ `$router.replace("/")` เพื่อเปลี่ยนเส้นทางของ router ไปยังหน้าหลักเว็บไซต์หรือหน้า Home.vue ("/").

สร้างไฟล์ Store.js ไว้เก็บข้อมูลตัวสินค้าและโปรโมชั่นของเว็บไซต์ เพื่อเอาไว้เรียกใช้ในสถานการณ์ต่างๆ



สร้าง object ที่มีชื่อ state ซึ่งประกอบด้วย specials เพื่อไว้เก็บข้อมูลโปรโมชั่น มีโครงสร้างดังนี้:

Image: ตำแหน่งของภาพเพื่อเอามาอ้างอิงในการใช้เปิดไฟล์ภาพ

Text: คำอธิบายของภาพ

```
import Vue from "vue";
import Vuex from "vuex";

const state = {
  specials : [
    {
      image: "/img/promo3.jpg",
      text: "เก็บตกงานหนังสือลดล้างคลัง 20% สูงสุด 1,200 บาท"
    },
    {
      image: "/img/promo2.jpg",
      text: "ฉลอง 3.3 ลดทั้งเว็บ 20%"
    },
    {
      image: "/img/promo4.jpg",
      text: "ลดทั้งเว็บ 15% เมื่อซื้อครบ 4 เล่นลดเพิ่มสูงสุด 20%"
    }
  ]
}
```

สร้าง object ที่มีชื่อ state ซึ่งประกอบด้วย menus เพื่อไว้เก็บข้อมูลของสินค้า มีโครงสร้างดังนี้:

Image: ตำแหน่งของภาพเพื่อเอามาอ้างอิงในการใช้เปิดไฟล์ภาพ

Text: ชื่อของตัวสินค้า

Price: ราคาสินค้า

Love: เป็นการกำหนดหัวใจ

```
menus: [  
  {  
    image: "/img/1.jpg",  
    text: "My hero acadamia",  
    price: 70,  
    love: false  
  },  
  {  
    image: "/img/2.jpg",  
    text: "My hero acadamia",  
    price: 70,  
    love: true  
  },  
  {  
    image: "/img/4.jpg",  
    text: "Kimetsu no yaiba",  
    price: 60,  
    love: false  
  },  
]
```

สร้าง getters method ไว้สำหรับใช้ในการดึงข้อมูลจาก state ดังนี้:

```

    orders : [],
    orderInfo: null
  };

  const getters = {
    specials(state) {
      return state.specials;
    },
    menus(state) {
      return state.menus;
    },
    haveOrders(state) {
      return state.orders.length > 0;
    },
    numberOfOrders(state) {
      return state.orders.length;
    },
    orders(state) {
      return state.orders;
    }
  };

```

specials: สำหรับดึงข้อมูลโปรโมชั่นพิเศษ

menus: สำหรับดึงข้อมูลรายการเมนู

haveOrders: สำหรับตรวจสอบว่ามีรายการสั่งซื้อหรือไม่

numberOfOrders: สำหรับคืนค่าจำนวนรายการสั่งซื้อ

orders: สำหรับดึงข้อมูลรายการสั่งซื้อ

สร้าง mutations ใช้สำหรับในการเปลี่ยนแปลงของข้อมูลใน state

```
const mutations = {  
  ADD_ORDER(state, order) {  
    state.orders.push(order);  
  },  
  DELETE_ORDER(state, index) {  
    state.orders.splice(index, 1);  
  },  
  SUBMIT_ORDER(state, orderInfo) {  
    state.orderInfo = orderInfo;  
  },  
  CLEAR_ORDER(state) {  
    state.orders = [];  
  }  
};
```

ADD_ORDER: ใช้สำหรับการเพิ่มสินค้าเมื่อสั่งซื้อใหม่มันจะเพิ่มรายการสั่งซื้อลงใน state.orders

DELETE_ORDER: ใช้สำหรับลบรายการสินค้าออกจาก state.orders

SUBMIT_ORDER: ใช้สำหรับเมื่อยืนยันการสั่งซื้อจะมีการเก็บข้อมูลลงใน state.orderInfo

CLEAR_ORDER: ใช้สำหรับลบรายการสั่งซื้อออกจาก state.orders

```
Vue.use(Vuex);  
  
const store = new Vuex.Store({  
  state,  
  getters,  
  mutations  
});  
  
export default store;
```

Vue.use(Vuex): ใช้สำหรับเรียกใช้งาน Vuex ใน Project

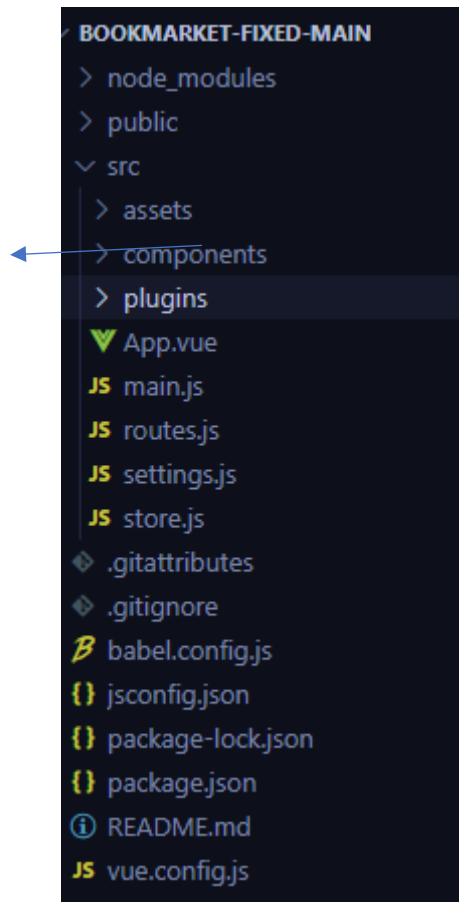
สร้าง store ใหม่ โดยรับ object ที่ประกอบด้วย state, getters และ mutations

จากนั้น export default store ที่สร้างขึ้น เพื่อเอาไปใช้ต่อใน Project

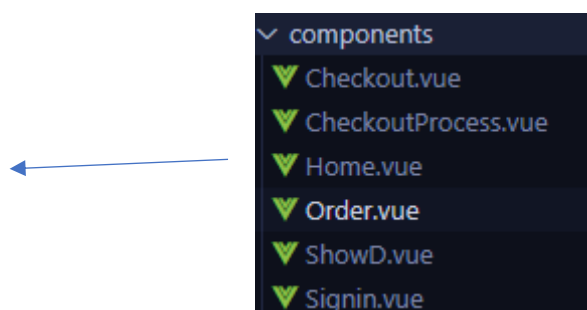
main.js

route.js

โดยที่จะสร้างหน้าต่างๆที่เป็นส่วนประกอบไว้มัน Folder ที่ชื่อใน component



สร้างหน้าจอหลัก Home.vue เพื่อแสดงสินค้าหน้าเว็บไซต์



โดยที่จะมี ส่วนของตัวแบนเนอร์โปรโมชั่น และส่วนแสดงสินค้า

ส่วนแรกเป็นส่วนแบนเนอร์แสดงโปรโมชั่น



โค้ดที่ไปดึงเอาข้อมูลโปรโมชั่นมาทำเป็นแบนเนอร์

```
<v-row>
  <v-col cols="12">
    <v-card elevation="24">
      <v-carousel cycle show-arrows hide-delimiters height="auto">
        <v-carousel-item v-for="(item, index) in specials" :key="index">
          <v-img
            height="auto"
            width="auto"
            :src="item.image"
            class="align-end"
            gradient="to bottom, rgba(0,0,0,.1), rgba(0,0,0,.5)"
          >
          <v-card-title class="white--text">{{ item.text }}</v-card-title>
        </v-carousel-item>
      </v-carousel>
    </v-card>
  </v-col>
```

`<v-carousel-item v-for="(item, index) in specials" :key="index">`: เป็นการใช้งาน `v-for` วนลูปข้อมูลใน `specials` ที่อยู่ใน `Stoe.js` และสร้าง `<v-carousel-item>` สำหรับแต่ละโปรโมชั่น

ส่วนของตัวสินค้าหน้าเว็บไซต์



My hero acadamia70 u.



My hero acadamia70 u.



โค้ดที่ไปถึงเอาข้อมูลตัวสินค้ามาแสดง

```

</v-row>
<v-col cols="6" md="3" v-for="(item, index) in menu" :key="index">

  <v-card :to="'/order/'+index">
    <v-img
      height="auto"
      width="auto"
      :src="item.image"
      class="align-end"
      gradient="to bottom, rgba(0,0,0,.1), rgba(0,0,0,.5)"
    >
    <v-card-title class="white--text">
      {{ item.text }} <strong>{{ item.price }} ฿.</strong>
    </v-card-title>
  </v-img>

  <v-card-actions>
    <v-spacer></v-spacer>

    <v-icon v-if="item.love" class="mr-5" color="pink">mdi-heart</v-icon>
    <v-icon v-else class="mr-5">mdi-heart-outline</v-icon>

    <v-icon>mdi-share-variant</v-icon>
  </v-card-actions>
</v-card>

</v-col>
</v-row>

```

```
<v-col cols="6" md="3" v-for="(item, index) in menu" :key="index">
```

ใช้สำหรับกำหนดคอลัมน์ในการแสดงโดยที่จะมีทั้งหมด 6 ในหน้าจอขนาดเล็ก และ 3 คอลัมน์ในหน้าจอขนาดใหญ่ เพื่อ
 วนลูป menu ตัวของสินค้า

```

<v-card :to="'/order/'+index">
  <v-img
    height="auto"
    width="auto"
    :src="item.image"
    class="align-end"
    gradient="to bottom, rgba(0,0,0,.1),
  rgba(0,0,0,.5)"
  >
  <v-card-title class="white--text">
    {{ item.text }} <strong>{{ item.price }} ฿.</strong>
  </v-card-title>
</v-img>

```

ใช้สำหรับแสดงข้อมูลในรูปแบบการ์ด โดยส่วนด้านบนของการ์ดจะเป็นรูปภาพ และถ้ากดที่รูปภาพจะไปยังหน้า
 Order.vue(/order) ตามตัวสินค้าที่กดเลือก

height: กำหนดความสูงของรูปภาพ

width: กำหนดความกว้างของรูปภาพ

:src: เรียกใช้งานรูปภาพจากตำแหน่งที่ระบุไว้

class = "align-end": จัดตำแหน่งของรูปภาพ

```

<v-card-actions>
  <v-spacer></v-spacer>

```

```

        <v-icon v-if="item.love" class="mr-5" color="pink">mdi-heart</v-
icon>

        <v-icon v-else class="mr-5">mdi-heart-outline</v-icon>

        <v-icon>mdi-share-variant</v-icon>
    </v-card-actions>

```

เป็นส่วนของการตกแต่งไอคอนรูปภาพหัวใจกับไอคอนแชร์

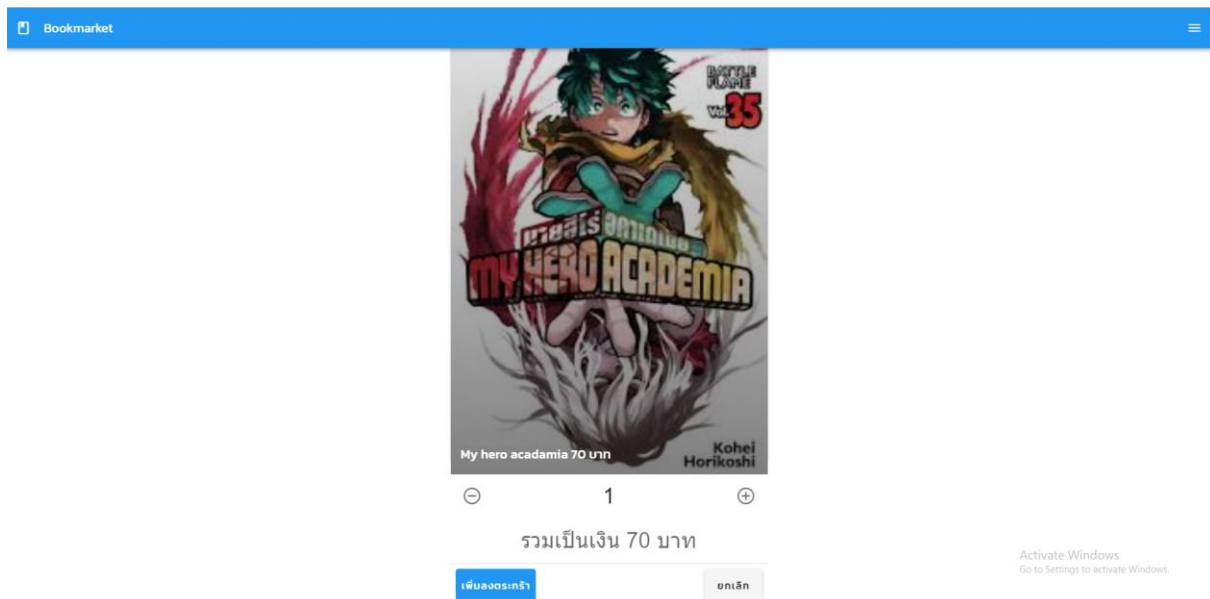
```

<script>
export default {
  computed: {
    specials() {
      return this.$store.getters.specials;
    },
    menus() {
      return this.$store.getters.menus;
    },
    haveOrders() {
      return this.$store.getters.haveOrders;
    },
    numberOfOrders() {
      return this.$store.getters.numberOfOrders;
    }
  }
};
</script>

```

เป็นส่วนของการคืนค่าของข้อมูล

หน้า Order.vue สำหรับการสั่งซื้อสินค้า



โค้ดนี้ใช้สำหรับแสดงหน้าสิ่งซื้อ

```
<template>
  <v-container>
    <v-card height="700"
      width="500"
      class="mx-auto">
      <v-img
        height="auto"
        width="auto"
        :src="menu.image"
        class="white--text align-end"
        gradient="to bottom, rgba(0,0,0,.1), rgba(0,0,0,.5)"
      >
      <v-card-title>{{ menu.text }} {{ menu.price }} บาท</v-card-title>
    </v-img>
```

แสดงหน้าสิ่งซื้อในรูปแบบการ์ด โดยใช้ v-card และใช้ v-img เพื่อแสดงรูปภาพ

height: กำหนดความสูงของรูปภาพ

width: กำหนดความกว้างของรูปภาพ

:src: เรียกใช้งานรูปภาพจากตำแหน่งที่ระบุไว้ โดยที่จะเรียกผ่านตัวแปรที่เคยสร้างไว้แล้ว menu.image

class = "white—text align-end": ทำให้ตัวหนังสือเป็นสีขาว และจัดตำแหน่งของรูปภาพ

ใช้ v-card-title เพื่อแสดงชื่อกับราคาของตัวสินค้า

```
<v-card-actions>
  <v-btn icon x-large @click="decQuantity">
    <v-icon>mdi-minus-circle-outline</v-icon>
  </v-btn>

  <v-spacer></v-spacer>

  <strong class="display-1">{{ quantity }}</strong>

  <v-spacer></v-spacer>

  <v-btn icon x-large @click="incQuantity">
    <v-icon>mdi-plus-circle-outline</v-icon>
  </v-btn>
</v-card-actions>

<v-card-text class="text-center">
  <strong class="display-1">รวมเป็นเงิน {{ quantity * menu.price }} บาท</strong>
</v-card-text>

<v-divider></v-divider>
```

การสั่งซื้อจะใช้การกดปุ่มเพื่อเพิ่มและลดจำนวนของสินค้า และมีการแสดงจำนวนของสินค้าที่สั่งซื้อ

ราคาทั้งหมดของสินค้าที่สั่งซื้อ และใส่เส้นต่อท้าย

```
<v-card-actions>
  <v-btn x-large color="blue" dark @click="addOrder">เพิ่มลงตระกร้า</v-btn>
  <v-spacer></v-spacer>
  <v-btn x-large @click="$router.go(-1)">ยกเลิก</v-btn>
</v-card-actions>
</v-card>
</v-container>
</template>
```

สร้างปุ่ม 2 ปุ่ม

```

addOrder() {
  this.$store.commit("ADD_ORDER", {
    id: this.$route.params.id,
    price: this.menu.price,
    text: this.menu.text,
    quantity: this.quantity
  });

  this.$router.go(-1);
}

```

โดยที่ปุ่มแรกเพื่อกดยืนยันการสั่งซื้อ

`<v-btn x-large color="blue" dark @click="addOrder">เพิ่มลงตะกร้า</v-btn>`

เมื่อกดปุ่มจะเรียกใช้เมธอด `addOrder` เพื่อเพิ่มข้อมูลลงในรายการสั่งซื้อสินค้า และจะย้อนกลับไปหน้าก่อนหน้า

โดยปุ่มกดที่สองเพื่อยกเลิกการสั่งซื้อและกลับไปหน้าก่อนหน้า

`<v-btn x-large @click="$router.go(-1)">ยกเลิก</v-btn></v-card-actions>`

`@click="$router.go(-1)"` เป็นส่วนย้อนกลับไปหน้าก่อนหน้า

เมื่อกดยืนยันการสั่งซื้อสินค้า จะแสดงดังนี้ที่หน้าหลัก(Home.vue):



ตัวอย่างโค้ด:

```

<template>
  <v-container>
    <v-btn v-if="haveOrders" large block color="#1565C0" dark to="/checkout">
      มีสินค้า {{ numberOfOrders }} รายการรอการชำระเงิน
    </v-btn>
  </v-container>
</template>

```

ไว้สำหรับเพื่อรอชำระสินค้า และยังสามารถไปซื้อสินค้าอื่นเพิ่มได้

จากนั้นกดปุ่ม “มีสินค้ารอการชำระ” เพื่อไปขั้นตอนการชำระ



ขั้นตอนการชำระเงิน Checkout.vue

ตัวอย่างหน้าชำระสินค้า:

1 →

2 →

3 →

4 →

Bookmark

รายการสินค้าในตระกร้า

My hero acadamia
ราคา 70 x 2 = 140

รวมเป็นเงิน 140 บาท

ข้อมูลส่งถึง
ชื่อ-นามสกุล

วิธีการส่งสินค้า
☒ มารับสินค้าที่ร้าน
☐ ให้ส่งมาอยู่ที่บ้าน

เบอร์โทรศัพท์

วิธีการชำระเงิน
☒ ชำระเงินปลายทาง
☐ ชำระเงินผ่าน e-Banking
☐ ชำระเงินผ่านบัตรเครดิต

ชำระเงิน

ส่วนที่หนึ่ง

เป็นส่วนแสดงรายการสินค้าทั้งหมดที่รอการชำระ และจำนวนเงินทั้งหมดที่ต้องจ่าย

ตัวอย่างโค้ด:

```
<template>
  <v-container>
    <v-card>
      <v-card-title>รายการสินค้าในตระกร้า</v-card-title>

      <v-list>
        <v-list-item v-for="(item, index) in orders" :key="index">
          <v-list-item-content>
            <v-list-item-title>{{ item.text }}</v-list-item-title>
            <v-list-item-subtitle>
              ราคา {{ item.price }} x {{ item.quantity }} =
              {{ item.price * item.quantity }}
            </v-list-item-subtitle>
          </v-list-item-content>

          <v-list-item-action>
            <v-btn icon @click="deleteOrder">
              <v-icon color="red">mdi-delete</v-icon>
            </v-btn>
          </v-list-item-action>
        </v-list-item>
      </v-list>
    </v-card>
  </v-container>
</template>
```

<v-list>: ใช้ในการแสดงรายการข้อมูลในรูปแบบของลิสต์

<v-list-item v-for="(item, index) in orders" :key="index">: เป็นการใช้งาน v-for วนลูปข้อมูลใน orders เพื่อสร้าง <v-list-item> สำหรับแต่ละรายการสินค้าที่อยู่ในตะกร้า

<v-list-item-content>: ใช้ในการแสดงเนื้อหาหลักของรายการ เช่น ชื่อสินค้าและราคา

<v-list-item-title>: ใช้ในการแสดงชื่อของสินค้า

<v-list-item-subtitle>: ใช้ในการแสดงรายละเอียดเสริมของสินค้า เช่น ราคา และ จำนวนสินค้า

<v-list-item-action>: ใช้ในการแสดงองค์ประกอบเสริมของรายการ เช่น ปุ่มลบสินค้า

<v-btn icon @click="deleteOrder">: สร้างปุ่มใช้ในการลบรายการสินค้าออกจากตะกร้า

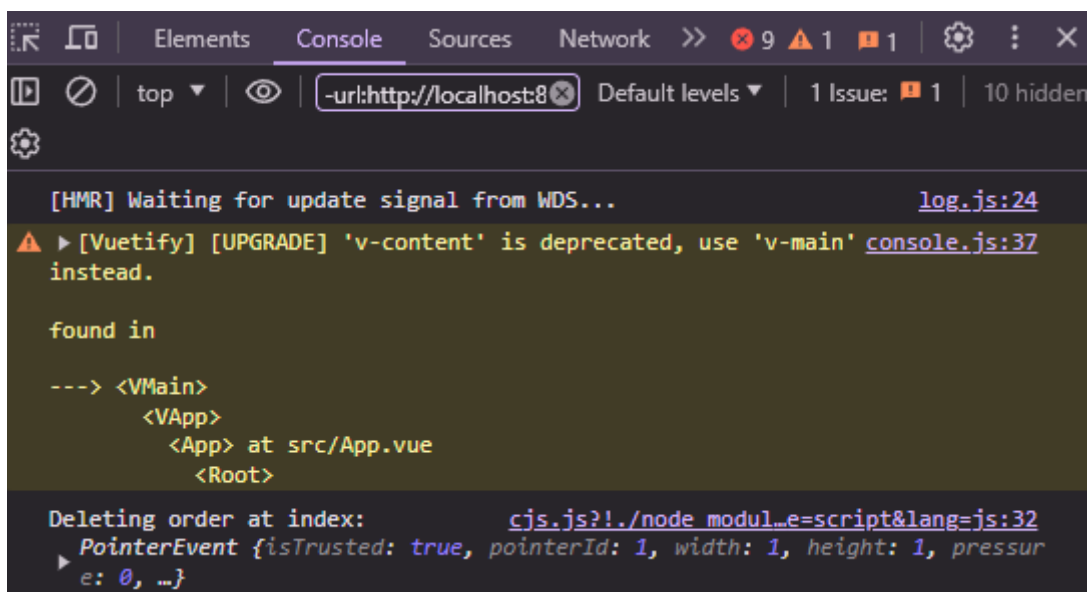
icon: กำหนดให้ปุ่มเป็นไอคอน

@click="deleteOrder": เป็นการเรียกใช้งานเมทอด deleteOrder เพื่อให้เรียกเมทอดนี้เมื่อปุ่มถูกคลิก โดยปุ่มนี้จะแสดงไอคอน ถึงขยะสีแดง

```
methods: {  
  deleteOrder(index) {  
    console.log("Deleting order at index:", index);  
    this.$store.commit("DELETE_ORDER", index);  
  },  
}
```

ใช้ลบรายการสั่งซื้อของสินค้าจากตะกร้า โดยอ้างอิง index ของรายการที่ต้องการลบ

เมื่อกดลบจะข้อความ ""Deleting order at index:", index"" ขึ้นแจ้งเตือนใน cosole ใน google



จากนั้นก็จะเรียกใช้ mutations ใน Store.js ชื่อ "DELETE_ORDER" โดยจะส่งค่า index ที่ต้องการลบเข้าไปที่ mutation รายการที่ตรงกับ index ที่ระบุจะถูกลบออกจากตะกร้า

ส่วนที่สอง

เป็นส่วนของการกรอกชื่อ-นามสกุล ของผู้สั่งซื้อ

```
<template>
  <v-container>
    <v-card>

      <v-divider></v-divider>

      <v-form v-model="valid">

        <v-card-text>
          <p>ชื่อผู้สั่งซื้อ</p>
          <v-text-field
            :rules="[v=>!!v || 'โปรดกรอกชื่อ-นามสกุล']"
            v-model="nameBuyer"
            label="ชื่อ-นามสกุล"
            filled
          ></v-text-field>
        </v-card-text>
      </v-form>
    </v-card>
  </v-container>
</template>
```

<v-card-text>: ใช้ในการแสดงเนื้อหาข้อความภายในการ์ด (card)

<p>ชื่อผู้สั่งซื้อ</p> ใช้ในการแสดงข้อความ "ชื่อผู้สั่งซื้อ"

<v-text-field>: ใช้ในการสร้างช่องกรอกข้อความ (text field)

:rules="[v=>!!v || 'โปรดกรอกชื่อ-นามสกุล']": กำหนดการตรวจสอบในช่อง v-text-field ว่าต้องใส่ชื่อไม่จะแสดงข้อความ "โปรดกรอกชื่อ-นามสกุล"

v-model="nameBuyer": ใช้ในเก็บข้อมูลที่กรอกลงในช่องกรอกกับตัวแปร nameBuyer เพื่อทำการเก็บค่าที่กรอกไว้

label="ชื่อ-นามสกุล": กำหนดป้ายชื่อ (label) ให้กับช่องกรอกข้อความ เพื่อแสดงคำอธิบายว่าควรกรอกข้อมูลอะไร

filled: มีพื้นหลังสีเต็มเมื่อได้รับการกรอกข้อมูลแล้ว

ส่วนที่สาม

```
<p>วิธีการส่งสินค้า</p>

<v-radio-group v-model="shippingType">
  <v-radio label="มารับสินค้าที่ร้าน" value="picking"></v-radio>
  <v-radio label="ให้ส่งมายังที่อยู่ด้านล่าง" value="shipping"></v-radio>
</v-radio-group>

<v-textarea
  v-if="shippingType=='shipping'"
  :rules="[v=>!!v||'โปรดกรอกที่อยู่']"
  v-model="shippingAddress"
  label="ที่อยู่จัดส่งสินค้า"
  filled
></v-textarea>

<v-text-field
  :rules="[v=>!!v||'โปรดกรอกเบอร์โทรศัพท์']"
  v-model="telephone"
  label="เบอร์โทรศัพท์"
  filled
></v-text-field>
</v-card-text>

<v-divider></v-divider>
```

<p>วิธีการส่งสินค้า</p>: ใช้ในการแสดงข้อความ "วิธีการส่งสินค้า"

<v-radio-group v-model="shippingType">: ใช้ในการให้ผู้ใช้เลือกตัวเลือกเดียวจากตัวเลือกที่มีอยู่

v-model="shippingType": ใช้ในเก็บข้อมูลที่กรอกลงในช่องกรอกกับตัวแปร shippingType

<v-radio label="มารับสินค้าที่ร้าน" value="picking"></v-radio>: ใช้ในเลือกวิธีการส่งสินค้า

label="มารับสินค้าที่ร้าน": กำหนดข้อความที่แสดงด้านข้างของปุ่มว่า "มารับสินค้าที่ร้าน"

value="picking": กำหนดค่าของปุ่มวิทยุเป็น "picking" เมื่อถูกเลือก

ตัวอย่าง:

วิธีการส่งสินค้า

- ☒ มารับสินค้าที่ร้าน
- ☐ ให้ส่งมายังที่อยู่ด้านล่าง

เบอร์โทรศัพท์

`<v-radio label="ให้ส่งมายังที่อยู่ด้านล่าง" value="shipping"></v-radio>`: ใช้ในการสร้างปุ่มวิทยุเพื่อเลือกวิธีการส่งสินค้า โดยมีคุณสมบัติดังนี้:

`label="ให้ส่งมายังที่อยู่ด้านล่าง"`: กำหนดข้อความที่แสดงด้านข้างของปุ่มว่า "ให้ส่งมายังที่อยู่ด้านล่าง"

`value="shipping"`: กำหนดค่าของปุ่มเป็น "shipping" เมื่อถูกเลือก

ตัวอย่าง:

วิธีการส่งสินค้า

☐

มารับสินค้าที่ร้าน

☒

ให้ส่งมายังที่อยู่ด้านล่าง

ที่อยู่จัดส่งสินค้า

เบอร์โทรศัพท์

`<v-text-field>`: ใช้ในการสร้างช่องกรอกข้อความ (text field)

`:rules="[v => !!v || 'โปรดกรอกเบอร์โทรศัพท์']"`: กำหนดการตรวจสอบในช่องว่าไม่ควรเป็นค่าว่าง ถ้าไม่ถูกต้องจะแสดงข้อความ "โปรดกรอกเบอร์โทรศัพท์"

`v-model="telephone"`: ใช้ในเก็บข้อมูลที่กรอกลงในช่องกรอกกับตัวแปร `telephone`

`label="เบอร์โทรศัพท์"`: กำหนดข้อความ ให้กับช่องกรอกข้อความ เพื่อแสดงคำอธิบายว่าควรกรอกข้อมูลอะไร

`filled`: กำหนดให้ช่องกรอกข้อความมีพื้นหลังสีเต็มเมื่อได้รับการกรอกข้อมูลแล้ว

เบอร์โทรศัพท์

ส่วนที่ดี

<v-card-actions>: ใช้ในการจัดวางปุ่มหรือตัวควบคุมอื่นๆ ในการ์ด (card)

<v-btn x-large color="green" class="white--text" :disabled="valid==false" @click="submitOrder">ชำระ
เงิน</v-btn>: ใช้ในการชำระเงิน:

x-large: กำหนดขนาดของปุ่มเป็นขนาดใหญ่

color="green": กำหนดสีของปุ่มเป็นสีเขียว

class="white--text": กำหนดให้ข้อความบนปุ่มเป็นสีขาว

:disabled="valid==false": กำหนดให้ปุ่มไม่สามารถคลิกได้ ต้องกรอกข้อมูลให้ครบทุกช่อง

@click="submitOrder": เป็นการเรียกใช้เมทอด submitOrder ที่ถูกกำหนดไว้ในเมทอด methods เพื่อทำการส่งคำสั่งเมื่อปุ่มถูกคลิก

```
methods: {
  deleteOrder(index) {
    console.log("Deleting order at index:", index);
    this.$store.commit("DELETE_ORDER", index);
  },
  submitOrder() {
    console.log("Submitting order...");
    const db = firebase.firestore();

    const orderData = {
      order: this.orders,
      name: this.nameBuyer,
      totalPrice: this.totalPrice,
      shippingType: this.shippingAddress,
      telephone: this.telephone,
      paymentMethod: this.paymentMethod,
      timestamp: firebase.firestore.FieldValue.serverTimestamp()
    };
  }
};
```

submitOrder ทำหน้าที่ส่งคำสั่งเพื่อบันทึกข้อมูลคำสั่งซื้อลงในฐานข้อมูล Firestore ของ Firebase โดยมีรายละเอียดดังนี้:

console.log("Submitting order..."): แสดงข้อความ "Submitting order..." ในคอนโซลเพื่อแสดงว่ากำลังทำการส่งคำสั่งซื้อ

const db = firebase.firestore(): เรียกใช้ Firestore ของ Firebase และเก็บไว้ในตัวแปร db

const orderData สร้าง object ของ orderData ที่เก็บข้อมูลคำสั่งซื้อที่จะบันทึกลงในฐานข้อมูล โดยมีรายละเอียดดังนี้:

order: รายการสินค้าที่ถูกสั่งซื้อ

name: ชื่อผู้สั่งซื้อ

totalPrice: ราคารวมของคำสั่งซื้อ

shippingType: วิธีการจัดส่ง

telephone: เบอร์โทรศัพท์ของผู้สั่งซื้อ

paymentMethod: วิธีการชำระเงิน

timestamp: เวลาของการส่งคำสั่งซื้อโดยใช้เวลาปัจจุบันของเซิร์ฟเวอร์ Firestore

```
// Submit order to Vuex store
this.$store.commit("SUBMIT_ORDER", {
  nameBuyer: this.nameBuyer,
  shippingType: this.shippingType,
  shippingAddress: this.shippingAddress,
  telephone: this.telephone,
  paymentMethod: this.paymentMethod
});

console.log("Order submitted to Vuex store:", orderData);
```

โค้ดนี้ใช้ในการส่งคำสั่งให้ Vuex store ด้วยฟังก์ชัน commit โดยส่งชื่อ mutation และข้อมูลคำสั่งซื้อเป็นพารามิเตอร์ เมื่อ mutation ถูกเรียก สถานะของ Vuex store จะถูกอัปเดตตามข้อมูลที่ส่งเข้ามา เช่น ชื่อผู้สั่งซื้อ วิธีการจัดส่ง ที่อยู่จัดส่ง เบอร์โทรศัพท์ และวิธีการชำระเงินที่ส่งมา ในที่นี้คือ nameBuyer, shippingType, shippingAddress, telephone, และ paymentMethod ตามลำดับ

หลังจากนั้น โค้ดยังทำการแสดงข้อความ "Order submitted to Vuex store" พร้อมกับข้อมูลคำสั่งซื้อที่ถูกส่งไปยัง Vuex store ในคอนโซล เพื่อให้เราตรวจสอบการส่งข้อมูลได้

```
Submitting order... cjs.js?!. /node modul_e=script&lang=js:36
Order submitted to Vuex store: cjs.js?!. /node modul_e=script&lang=js:56
  {order: Array(1), name: 'akkarapon chaikot', totalPrice: 60, shippingType: null, telephone: '0831711167', ...}
Document written with ID: cjs.js?!. /node modul_e=script&lang=js:60
Dm7ZIAOrKMkztMokWbE1
Redirecting to checkout-process... cjs.js?!. /node modul_e=script&lang=js:62
```

```

// Add orderData to firestore
db.collection("order").add(orderData)
  .then((docRef) => {
    console.log("Document written with ID:", docRef.id);
    // Redirect to checkout-process with the document ID
    console.log("Redirecting to checkout-process...");
    this.$router.replace("/checkout-process");
    // Clear orders in Vuex store
    this.$store.commit("CLEAR_ORDERS");
  })
  .catch((error) => {
    console.error("Error adding document: ", error);
  });

```

โค้ดนี้ใช้ในการเพิ่มข้อมูลคำสั่งซื้อลงในคอลเล็กชัน "order" ใน Firestore โดยใช้

`db.collection("order").add(orderData)` ซึ่งจะเพิ่มข้อมูล `orderData` ลงในคอลเล็กชัน "order" ในฐานข้อมูล Firestore โดยอัตโนมัติ

`.then((docRef))`: เมื่อการเพิ่มข้อมูลเสร็จสิ้น โค้ดในบล็อก `.then()` จะถูกเรียก ซึ่งจะรับพารามิเตอร์ `docRef` ซึ่งเป็นอ้างอิงของเอกสารใหม่ที่ถูกเพิ่มเข้าไปในคอลเล็กชัน "order"

`console.log("Document written with ID:", docRef.id)`: แสดงข้อความ "Document written with ID: docRef.id" ในคอนโซล เพื่อแสดง ID ของเอกสารที่ถูกเพิ่ม

`.catch((error))`: หากเกิดข้อผิดพลาดขณะเพิ่มข้อมูล โค้ดในบล็อก `.catch()` จะถูกเรียก

`console.error("Error adding document: ", error)`: แสดงข้อความข้อผิดพลาดในคอนโซล เพื่อแสดงข้อความข้อผิดพลาดที่เกิดขึ้น

หลังจากนั้นโค้ดทำการเปลี่ยนเส้นทางการนำทาง (routing) ไปยังหน้า `/checkout-process` ด้วย

`this.$router.replace("/checkout-process")` เพื่อไปยังหน้าที่แสดงข้อมูลการสั่งซื้อหลังจากที่ทำการสั่งซื้อเสร็จสิ้น

สุดท้ายเรียกใช้ mutation `"CLEAR_ORDERS"` ใน Vuex store (รายการสั่งซื้อสินค้า) ด้วย

`this.$store.commit("CLEAR_ORDERS")` เพื่อลบรายการสินค้าที่ถูกสั่งซื้อออกจาก Vuex store (รายการสั่งซื้อสินค้า) หลังจากที่ได้ทำการสั่งซื้อเสร็จสิ้นแล้ว

`<v-spacer></v-spacer>`: เป็นคอมโพเนนต์ใน Vuetify ที่ใช้ในการสร้างช่องว่าง (spacer) เพื่อจัดตำแหน่งปุ่มในกลุ่ม

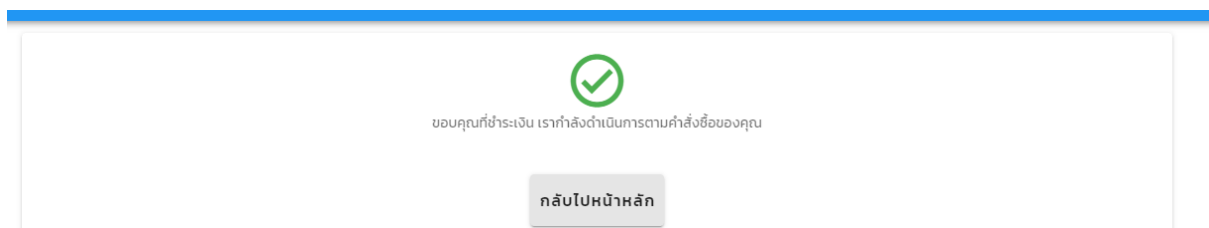
`<v-btn x-large to="/">กลับไปหน้าหลัก</v-btn>`: เป็นปุ่มใน Vuetify ที่ใช้ในการกลับไปหน้าหลัก โดยมีคุณสมบัติดังนี้:

x-large: กำหนดขนาดของปุ่มเป็นขนาดใหญ่

to="/": กำหนด URL ของปุ่มเพื่อให้กลับไปหน้าหลัก เมื่อปุ่มถูกคลิก

หน้า CheckoutProcess.vue

หลังจากดยืนยันแล้วจะแสดงหน้าขึ้น



โค้ดหน้าCheckoutProcess.vue

```
<template>
  <v-container>
    <v-card>
      <v-card-text class="text-center">
        <v-icon color="green" size="64">mdi-check-circle-outline</v-icon>
        <p>ขอบคุณที่ชำระเงิน เรากำลังดำเนินการตามคำสั่งซื้อของคุณ</p>
      </v-card-text>

      <v-card-actions class="d-flex justify-center">
        <v-btn x-large to="/">กลับไปหน้าหลัก</v-btn>
      </v-card-actions>
    </v-card>
  </v-container>
</template>

<script>
export default {
  mounted() {
    this.$store.commit("CLEAR_ORDER");
  }
}
</script>
```

<v-icon color="green" size="64">mdi-check-circle-outline</v-icon>: แสดงไอคอนที่แสดงสถานะการชำระเงินเสร็จสิ้น โดยมีสีเขียวและขนาดใหญ่

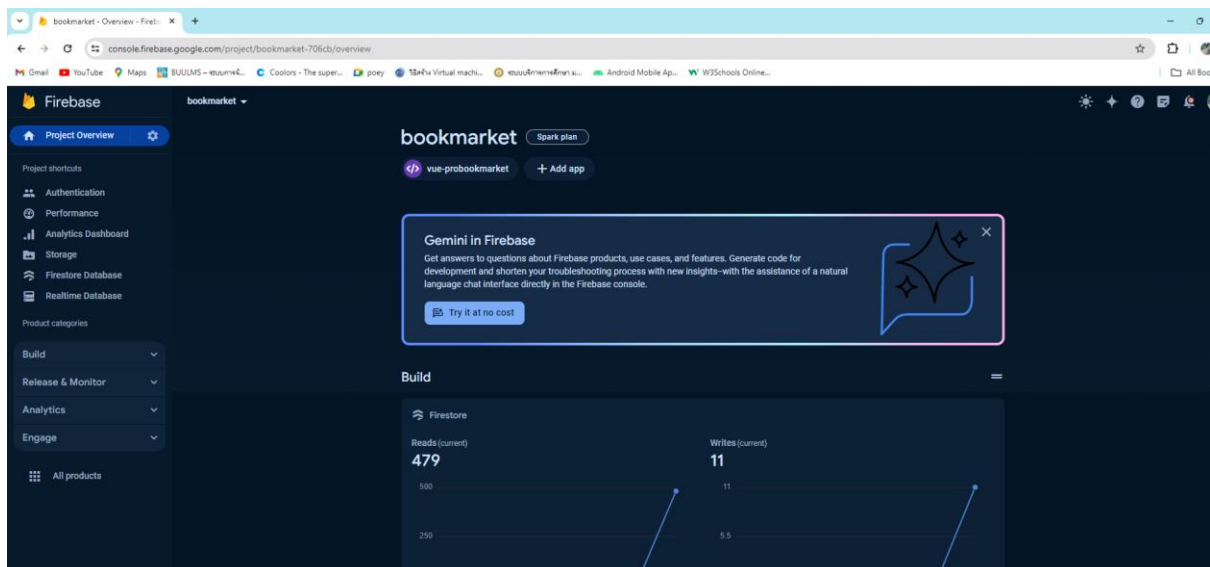
<p>ขอบคุณที่ชำระเงิน เรากำลังดำเนินการตามคำสั่งซื้อของคุณ</p>: แสดงข้อความขอบคุณ

<v-btn x-large to="/">กลับไปหน้าหลัก</v-btn>: ปุ่มที่ให้คลิกเพื่อกลับไปหน้าหลัก โดยใช้ Vue Router เพื่อนำผู้ใช้กลับไปยังหน้าหลัก

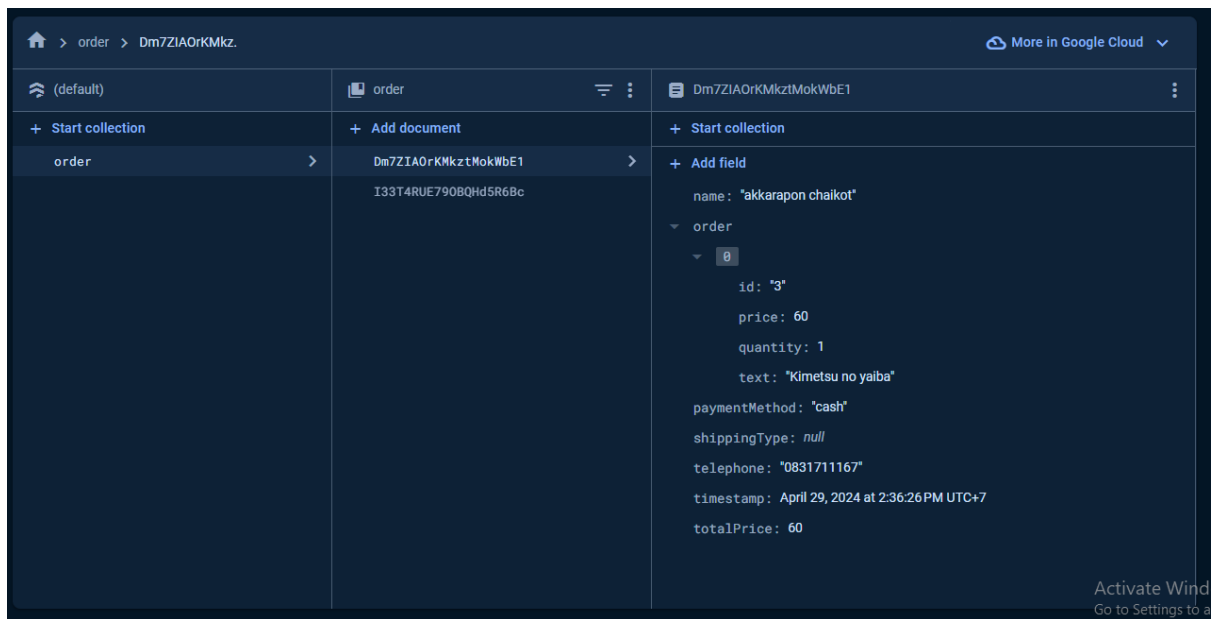
เรียกใช้ mutation "CLEAR_ORDER" ใน Vuex store เพื่อล้างข้อมูลคำสั่งซื้อออกจาก state ใน Vuex store

หลังจากสิ้นสุดขั้นตอนการชำระเงินแล้ว ข้อมูลการสั่งซื้อทั้งหมดสามารถตรวจสอบดูใน Firestore ใน Firebase

เข้า Project ที่สร้างไว้ แล้วคลิกเลือก Firestore Database



ข้อมูลการสั่งซื้อจะแสดงดังนี้:

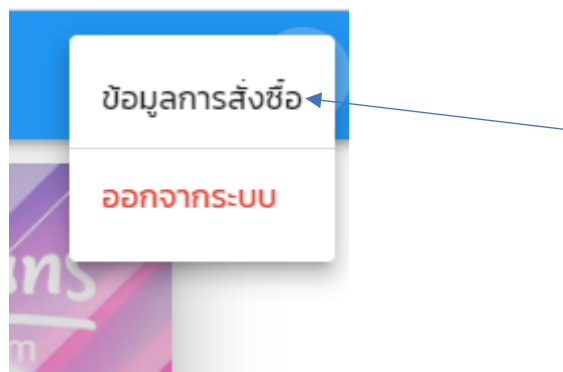


สามารถเช็ครายการสั่งซื้อสินค้าได้จากหน้าเว็บไซต์ได้ด้วย

คลิกตามที่ลูกศรชี้



จากนั้นคลิกเลือก “ข้อมูลการสั่งซื้อ”



หน้าข้อมูลการสั่งซื้อสินค้า ShowD.vue

Bookmarket

รายการสั่งซื้อ

รหัสการสั่งซื้อ	ชื่อผู้สั่งซื้อ	จำนวนเงินทั้งหมด(บาท)	เบอร์โทร	วิธีการชำระ	จัดการ
Dm7ZIAOrKMkztMokWbE1	akkarapon chaikot	60	0831711167	cash	<div>ลบ</div>
I33T4RUE79OBQHd5R6Bc	โชคดี มีชัย	140	0875398804	eBanking	<div>ลบ</div>

กลับไปหน้าหลัก

ตัวอย่างโค้ด:

```
1 <template>
2   <v-app>
3     <v-content>
4       <router-view></router-view>
5
6       <div v-if="orders.length > 0">
7         <v-card><h2>รายการสั่งซื้อ</h2></v-card>
8         <table>
9           <thead>
10            <tr>
11              <th>รหัสการสั่งซื้อ</th>
12              <th>ชื่อผู้สั่งซื้อ</th>
13              <th>จำนวนเงินทั้งหมด(บาท)</th>
14              <th>เบอร์โทร</th>
15              <th>วิธีการชำระ</th>
16              <th>จัดการ</th>
17            </tr>
18          </thead>
19          <tbody>
20            <tr v-for="(order, index) in orders" :key="index">
21              <td>{{ order.id }}</td>
22              <td>{{ order.data().name }}</td>
23              <td>{{ order.data().totalPrice }}</td>
24              <td>{{ order.data().telephone }}</td>
25              <td>{{ order.data().paymentMethod }}</td>
26              <td>
27                <v-btn color="red" class="white--text" @click="deleteOrder(order.id)">ลบ</v-btn>
28              </td>
29            </tr>
30          </tbody>
31        </table>
32      </div>
33      <div v-else>
34        <h2>ไม่พบรายการสั่งซื้อ</h2>
35      </div>
36      <v-card-actions class="d-flex justify-center">
37        <v-btn x-large to="/">กลับไปหน้าหลัก</v-btn>
38      </v-card-actions>
39    </v-content>
40  </v-app>
41 </template>
```

โค้ดนี้ใช้สำหรับการแสดงผลของหน้าแสดงรายการสั่งซื้อของเว็บไซต์

ส่วนของตาราง

`<v-app>`: ใช้ในการกำหนดโครงสร้างหลักของแอปพลิเคชัน

`<v-content>`: ใช้ในการจัดเนื้อหาในหน้าแสดงผล

`<router-view></router-view>`: เป็นตำแหน่งที่เนื้อหาของเส้นทางจะถูกแสดงผล

`<div v-if="orders.length > 0"> </div>`: ใช้เงื่อนไข `v-if` เพื่อตรวจสอบว่ามีรายการสั่งซื้อหรือไม่ ถ้ามีจะแสดงตารางรายการสั่งซื้อ และถ้าไม่มีจะแสดงข้อความ "ไม่พบรายการสั่งซื้อ"

`<v-card><h2>รายการสั่งซื้อ</h2></v-card>`: ใช้ `<v-card>` เพื่อสร้างกล่องรอบข้อมูลและแสดงหัวข้อ "รายการสั่งซื้อ"

`<table>...</table>`: ใช้สร้างตารางแสดงรายการสั่งซื้อ

`<tbody>`: เป็นส่วนของตารางที่เก็บข้อมูลรายการสั่งซื้อ

`<tr v-for="(order, index) in orders" :key="index">`: ใช้ `v-for` เพื่อวนลูปผ่านข้อมูลในอาร์เรย์ `orders` เพื่อแสดงผลรายการสั่งซื้อทั้งหมด โดยกำหนดคีย์ด้วย `index` ที่ไม่ซ้ำกัน

`<td>{{ order.id }}</td>`: แสดง ID ของรายการสั่งซื้อที่อยู่ใน Firestore ที่อยู่ใน `id`

`<td>{{ order.data().name }}</td>`: แสดงชื่อผู้สั่งซื้อที่อยู่ใน Firestore ที่อยู่ใน `name`

`<td>{{ order.data().totalPrice }}</td>`: แสดงราคารวมของรายการสั่งซื้อที่อยู่ใน Firestore ที่อยู่ใน `totalPrice`

`<td>{{ order.data().telephone }}</td>`: แสดงเบอร์โทรศัพท์ของผู้สั่งซื้อที่อยู่ใน Firestore ที่อยู่ใน `telephone`

`<td>{{ order.data().paymentMethod }}</td>`: แสดงวิธีการชำระเงินของรายการสั่งซื้อที่อยู่ในเอกสาร Firestore ที่อยู่ใน `paymentMethod`

`<td>`: ส่วนของการจัดการซึ่งมีปุ่ม "ลบ" ในแต่ละแถว เมื่อคลิกที่ปุ่มจะเรียกใช้เมธอด `deleteOrder` และส่ง ID ของรายการสั่งซื้อเป็นพารามิเตอร์

โค้ดตัวอย่าง:

```

77   deleteOrder(orderId) {
78     firebase
79       .firestore()
80       .collection("order")
81       .doc(orderId)
82       .delete()
83       .then(() => {
84         console.log("Order deleted successfully:", orderId);
85         this.orders = this.orders.filter(order => order.id !== orderId);
86       })
87       .catch(error => {
88         console.error("Error deleting order:", error);
89       });
90   }

```

โค้ดด้านบนเป็นเมธอด `deleteOrder(orderId)` ที่ใช้สำหรับลบรายการสั่งซื้อที่มี ID ที่กำหนดออกจาก Firestore โดยที่จะลบค่าพารามิเตอร์ของ `orderId` ที่รับมา ดังนี้:

Import ตัว component ที่จะต้องใช้

`import firebase from "firebase/app";`

`firebase/app`: โมดูลหลักของ Firebase ที่ใช้ในการเรียกใช้งาน Firebase services ต่างๆ

`import "firebase/auth";`

`firebase/auth`: โมดูลสำหรับ Firebase Authentication เพื่อการจัดการระบบการตรวจสอบสิทธิ์และการเข้าสู่ระบบของผู้ใช้

`import "firebase/firestore";`

`firebase/firestore`: โมดูลสำหรับ Firebase Firestore เพื่อการจัดการกับฐานข้อมูลแบบ NoSQL ในโปรเจกต์ Vue.js

`.firestore()`: เรียกใช้งาน Firestore database ของ Firebase

`.collection("order")`: ระบุชื่อของคอลเลกชันที่ต้องการลบข้อมูลออก

`.doc(orderId)`: เลือกข้อมูลในคอลเลกชันที่มี ID ตามที่กำหนดในพารามิเตอร์ `orderId`

`.delete()`: เรียกใช้เมธอดเพื่อลบข้อมูลที่ถูกละเลือก

`.then(() => { }):`

`this.orders = this.orders.filter(order => order.id !== orderId)`: อัปเดตค่าของตัวแปร `orders` โดยการเข้าถึงข้อมูล `orders` จะถูกกรองเพื่อลบรายการที่ตรงกับ `orderId` ออกจากอาร์เรย์ `orders` ที่เก็บข้อมูลรายการสั่งซื้อทั้งหมด

`.catch(error => { })`: ในกรณีที่เกิดข้อผิดพลาดในการลบ จะแสดงข้อผิดพลาดที่เกิดขึ้นใน `console.log` ด้วยคำสั่ง `console.error()`

`<div v-else> </div>`: ใช้เงื่อนไข `v-else` เพื่อแสดงข้อความ "ไม่พบรายการสั่งซื้อ" เมื่อไม่มีรายการสั่งซื้อ.

`<v-card-actions> </v-card-actions>`: ใช้ในการจัดการปุ่มบนหน้าแสดงผล ซึ่งในที่นี้ใช้ปุ่มเพื่อกลับไปยังหน้าหลัก

`<v-btn x-large to="/">กลับไปหน้าหลัก</v-btn>`: ปุ่มใช้ในการกลับไปยังหน้าหลัก โดยใช้ `to` สำหรับการกำหนดเส้นทางไปยังหน้าหลักของตัวเว็บไซต์

ส่วน script เพิ่มเติม หน้าข้อมูลการสั่งซื้อ (ShowD.vue)

`data()`: มีคุณสมบัติ `orders` เป็นอาร์เรย์ที่เก็บรายการสั่งซื้อที่ได้รับมาจาก Firestore

`mounted()`: ใช้สำหรับโหลดและประมวลผล ฟังก์ชัน `fetchOrders()` จะถูกเรียก เพื่อดึงข้อมูลรายการสั่งซื้อจาก Firestore

`fetchOrders()`: เป็นเมทอดที่ใช้เรียก Firestore เพื่อดึงข้อมูลรายการสั่งซื้อทั้งหมด โดยจะเรียก `.get()` บนคอลเล็กชัน "order" และจัดการกับผลลัพธ์ได้ด้วย `.then()` และ `.catch()` โดยเพิ่มข้อมูลรายการสั่งซื้อลงในอาร์เรย์ `orders`