วิชา Mobile Web Application Development

แบบฝึกหัด 7 การพัฒนาแอปด้วย Ionic ร่วมกับ Firebase Authentication

รหัสนักศึกษา : **ชื่อ-สกุล :**

Link Github Project [https://github.com/NattananKhamsom/lab07-authen]

Link Github Page [https://nattanankhamsom.github.io/mobileweb2/]

วัตถุประสงค์ของใบงาน

เมื่อทำใบงานนี้เสร็จแล้ว นักศึกษาจะสามารถ

1. สร้างโปรเจกต์ Ionic (React/Vue) ด้วย TypeScript และ pnpm
2. ตั้งค่า Firebase Authentication และ Provider ต่าง ๆ
3. ออกแบบโครงสร้างโค้ดแบบ maintainable: modules / interfaces / services / state
4. ทำหน้า Login (React หรือ Vue) ที่เรียกใช้ service เดียวกัน
5. Build และรันบน Android ด้วย Capacitor + config ที่จำเป็น

เครื่องมือที่ใช้

- Ionic Framework (Vue)
- Firebase Project (สร้างใน Firebase Console)
- Node.js และ pnpm
    - **ตั้งค่าให้ ionic ใช้ pnpm แทน npm**

      ```
      ionic config set -g npmClient pnpm
      ```
- Web Browser
- Android Studio + Android SDK (สำหรับ deploy Android)
- Vscode + GitHub Copilot
    - นักศึกษาสามารถใช้ prompt เพื่อช่วยพัฒนาโปรแกรมได้

ขั้นตอนการดำเนินงาน

ขั้นตอนที่ 1 สร้างโครงการ

1. สร้างโปรเจกต์ Ionic Vue หรือ React ใหม่ ตั้งชื่อเป็น lab07-authen
   ให้สร้างโดยใช้ template tabs ไว้ใน folder apps
2. ทดลองรันแอปให้แสดงผลใน Browser ให้สำเร็จ

คำถาม

เลือกคำสั่งที่ใช้สร้างโปรเจกต์ 1 อัน

[ ] ionic start lab07-authen tabs --type react

[ ] ionic start lab07-authen tabs --type vue

ขั้นตอนที่ 2 การเชื่อมต่อ Firebase

1. เปิด https://firebase.google.com/ ใช้โครงการเดิมจาก Lab06-expense
   ดูคลิปแนะนำ https://www.youtube.com/watch?v=OnYSjEehxH0
2. ติดตั้ง Firebase SDK
   cd **lab07-authen**
   pnpm install firebase @capacitor-firebase/authentication

   หมายเหตุ: firebase เป็น sdk มาตรฐาน, capacitor-firebase/authentication เป็น plugins ทำงานได้ทั้ง Android/iOS และมี API ครบสำหรับ Email/Password, Google, Phone
   Plugin นี้มีเมธอดสำคัญ เช่น `signInWithEmailAndPassword`, `signInWithGoogle`, `signInWithPhoneNumber` และ flow OTP ผ่าน listener อ่านเพิ่มเติม
   https://capawesome.io/plugins/firebase/authentication/

3. เพิ่ม android
   ionic cap add android

4. ตั้งค่า android package id

แก้ไขไฟล์ capacitor.config.ts

โดยให้ตั้งชื่อ package id ที่ไม่ซ้ำกับใคร (ชื่อหน่วยงาน+ชื่อย่อตัวเอง+ชื่อ lab)

```
import type { CapacitorConfig } from '@capacitor/cli';

const config: CapacitorConfig = {
  appId: 'cpkku.twachi.lab07',
  appName: 'lab07-authen',
  webDir: 'dist'
};

export default config;
```

เรียกคำสั่ง

ionic cap sync

คำสั่งนี้จะอัปเดต package id ในหลายไฟล์ให้อัตโนมัติ

ถ้ามีการแก้ไข appId จะต้องเรียกคำสั่ง **ionic cap sync** เสมอ

ขั้นตอนที่ 3 สร้าง Authentication Service

โดยจะสร้างไว้ใน folder lab07-authen/src/auth

1. สร้าง interface เพื่อให้รองรับทั้งเว็บ และ app (android/ ios)

auth/auth-interface.ts

TypeScript

```
export interface AuthUser {
  uid: string;
  email?: string | null;
  phoneNumber?: string | null;
  displayName?: string | null;
  photoUrl?: string | null;
}

export interface EmailPasswordCredentials {
  email: string;
  password: string;
```

```
}

export interface PhoneCredentials {
  phoneNumberE164: string; // เช่น +66812345678
}

export type AuthProvider = "email" | "phone" | "google";

export interface IAuthService {
  getCurrentUser(): Promise<AuthUser | null>;

  loginWithEmailPassword(creds: EmailPasswordCredentials):
Promise<AuthUser>;
  loginWithGoogle(): Promise<AuthUser>;

  // phone: แยกเป็น 2 ขั้น: ส่ง OTP และยืนยัน OTP
  startPhoneLogin(creds: PhoneCredentials): Promise<{ verificationId: string
}>;
  confirmPhoneCode(payload: { verificationId: string; verificationCode:
string }): Promise<AuthUser>;

  logout(): Promise<void>;
}
```

2. implement สำหรับ App

auth/auth-app.ts

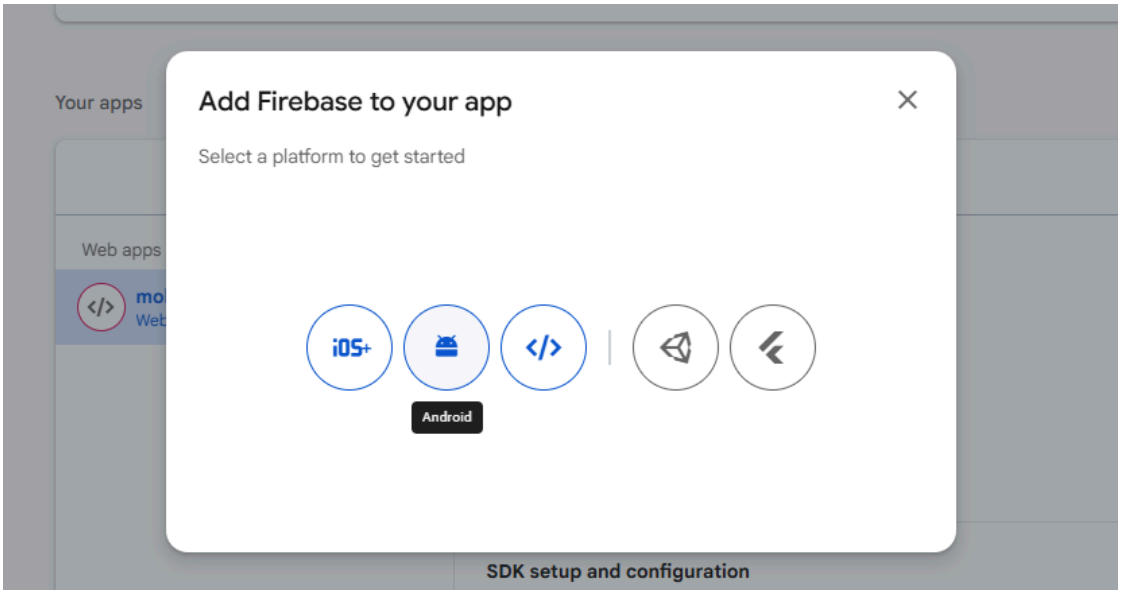
TypeScript

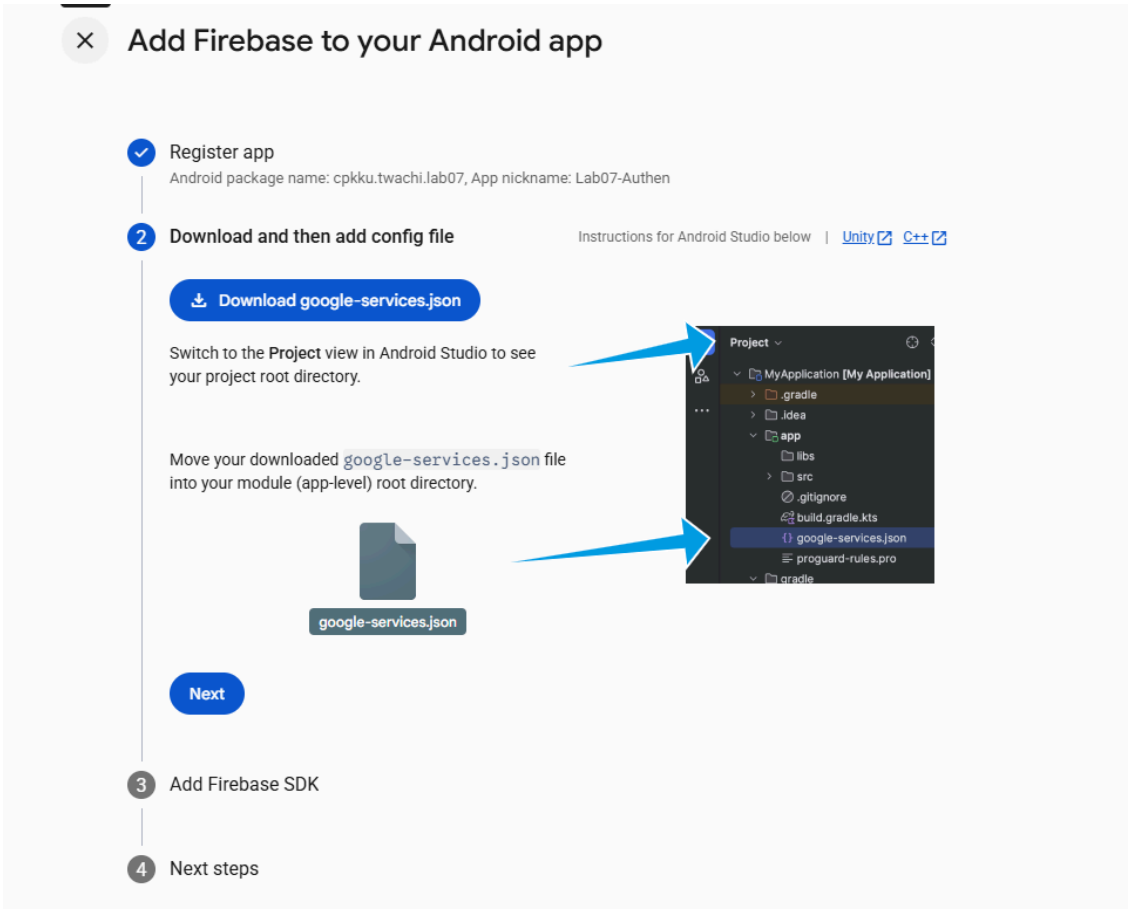
```
import { FirebaseAuthentication } from
"@capacitor-firebase/authentication";
import type { IAuthService } from "./auth-interface";
import type { AuthUser, EmailPasswordCredentials,
PhoneCredentials } from "./auth-interface";
```

```ts
function mapUser(u: any): AuthUser {
  return {
    uid: u.uid,
    email: u.email ?? null,
    phoneNumber: u.phoneNumber ?? null,
    displayName: u.displayName ?? null,
    photoUrl: u.photoUrl ?? null,
  };
}

export class FirebaseAppAuthService implements IAuthService {
  async getCurrentUser(): Promise<AuthUser | null> {
    const { user } = await
FirebaseAuthentication.getCurrentUser();
    return user ? mapUser(user) : null;
  }

  async loginWithEmailPassword(creds: EmailPasswordCredentials):
Promise<AuthUser> {
    const { user } = await
FirebaseAuthentication.signInWithEmailAndPassword(creds);
    return mapUser(user);
  }

  async loginWithGoogle(): Promise<AuthUser> {
    const { user } = await
FirebaseAuthentication.signInWithGoogle();
    return mapUser(user);
  }

  //ให้ UI คุม state เอง แต่ service คุม listener/flow ให้
  async startPhoneLogin(creds: PhoneCredentials): Promise<{
verificationId: string }> {
    // จะได้ verificationId ผ่าน event phoneCodeSent
    return new Promise(async (resolve, reject) => {
```

```
      const offFailed = await
FirebaseAuthentication.addListener("phoneVerificationFailed", (e)
=> {
        reject(new Error(e.message ?? "Phone verification
failed"));
      });

      const offSent = await
FirebaseAuthentication.addListener("phoneCodeSent", (e) => {
        // cleanup listener ฝั่ง success step 1
        offFailed.remove();
        offSent.remove();
        resolve({ verificationId: e.verificationId });
      });

      await FirebaseAuthentication.signInWithPhoneNumber({
        phoneNumber: creds.phoneNumberE164,
      });
    });
  }

  async confirmPhoneCode(payload: { verificationId: string;
verificationCode: string }): Promise<AuthUser> {
    const { user } = await
FirebaseAuthentication.confirmVerificationCode(payload);
    return mapUser(user);
  }

  async logout(): Promise<void> {
    await FirebaseAuthentication.signOut();
  }
}
```

เพิ่ม android app ใน firebase config โดยใช้ package id ที่สร้างในข้อ 2.4





ให้ download google-services.json ใน project firebase

แล้ววางไว้ที่ android/app/google-services.json

3. implement สำหรับ Web

auth/auth-web.ts

```
// นำมาจากหน้าโครงการ firebase
const firebaseConfig = {
  apiKey: "xxxx",
  authDomain: "xxx",
  projectId: "xxx",
  storageBucket: "web2025-wachi.firebasestorage.app",
  messagingSenderId: "71907196915",
  appId: "1:71907196915:web:f61190eb685f25e3431c86",
  measurementId: "G-3HMD9F71XR"
};
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { AuthUser, IAuthService, EmailPasswordCredentials,PhoneCredentials }
from "./auth-interface";
import {
  signInWithEmailAndPassword,
  GoogleAuthProvider,
  signInWithPopup,
  signInWithPhoneNumber,
  ConfirmationResult,
} from "firebase/auth";

export const firebaseApp = initializeApp(firebaseConfig);
export const firebaseAuth = getAuth(firebaseApp);

function mapUser(u: any): AuthUser {
  return {
    uid: u.uid,
    email: u.email,
    displayName: u.displayName,
    photoUrl: u.photoURL,
  };
}

import { RecaptchaVerifier } from "firebase/auth";
import { code } from "ionicons/icons";
```

```ts
let verifier: RecaptchaVerifier | null = null;
let confirmationResult: ConfirmationResult | null = null;

// ควรมี div สำหรับ reCAPTCHA ในหน้า login สำหรับโทรศัพท์ ด้วย
id="recaptcha-container"
const recaptchaContainerId: string = "recaptcha-container";

export function getRecaptchaVerifier(
  containerId: string
): RecaptchaVerifier {
  if (!verifier) {
    verifier = new RecaptchaVerifier(
      firebaseAuth,
      containerId,
      {
        size: "invisible", // หรือ "normal"
      }
    );
  }
  return verifier;
}

export class FirebaseWebAuthService implements IAuthService {
  async getCurrentUser() {
    return firebaseAuth.currentUser
      ? mapUser(firebaseAuth.currentUser)
      : null;
  }

  async loginWithEmailPassword(creds: EmailPasswordCredentials) {
    const r = await signInWithEmailAndPassword(
      firebaseAuth,
      creds.email,
      creds.password
    );
    return mapUser(r.user);
  }

  async loginWithGoogle() {
    const provider = new GoogleAuthProvider();
    const r = await signInWithPopup(firebaseAuth, provider);
```

```
    return mapUser(r.user);
  }

  async logout() {
    await firebaseAuth.signOut();
  }

  async startPhoneLogin(
    creds: PhoneCredentials
  ): Promise<{ verificationId: string }> {
    const verifier = getRecaptchaVerifier(recaptchaContainerId);
    confirmationResult = await signInWithPhoneNumber(
      firebaseAuth,
      creds.phoneNumberE164,
      verifier
    );
    return { verificationId: confirmationResult.verificationId };
  }

  async confirmPhoneCode(payload: { verificationId: string;
verificationCode: string }): Promise<AuthUser> {
    if (!confirmationResult) {
      throw new Error("No confirmation result");
    }
    const r = await confirmationResult.confirm(payload.verificationCode);
    return mapUser(r.user);
  }

}
```

4. สร้าง auth service ที่เลือก app/web โดยอัตโนมัติ

auth/auth-service.ts

```
import { Capacitor } from "@capacitor/core";
import type { IAuthService } from "./auth-interface";
import { FirebaseWebAuthService } from "./auth-web";
import { FirebaseAppAuthService } from "./auth-app";
```

```
export const authService: IAuthService =
  Capacitor.isNativePlatform()
    ? new FirebaseAppAuthService()      // Android / iOS
    : new FirebaseWebAuthService();     // Web
```

ขั้นตอน 4 สร้างหน้าจอ Login

1. ทำ login flow เพื่อทำ Auth Guard

   ถ้า ไม่มี user ⟶ redirect ไป /login

   ตัวอย่างสำหรับ vue

```
src/router/index.ts
เพิ่ม
{
   path: "/login",
   component: LoginPage,
 },

// ส่วนท้ายไฟล์ เพิ่ม
import { authService } from '@/auth/auth-service';
router.beforeEach(async (to) => {
  const user = await authService.getCurrentUser();
  // login แล้ว ห้ามเข้า /login
  if (to.path === "/login" && user) {
    return "/tabs/tab1";
  }
  if (to.meta.requiresAuth && !user) {
    return "/login";
  }
  return true;
});
```

```
export default router
```

2. สร้างหน้า login

2.1 โดยให้มี ปุ่มให้เลือก 3 วิธี

Login Email/Password

Login Google

Login by Phone

2.2 เมื่อ Login สำเร็จให้ไปที่หน้า Tab1

3. **แก้ไขหน้า Tab1**

แสดงข้อมูล User จากคำสั่ง

```
const user = await authService.getCurrentUser();
```

```
AuthUser {

  uid: string;
  email?: string | null;
  phoneNumber?: string | null;
  displayName?: string | null;
  photoUrl?: string | null;

}
```

ขั้นตอน 5 Build Android App

pnpm run build

ionic cap copy android

ionic cap sync android

ionic cap open android

เชื่อมต่อ มือถือ android ผ่าน USB / เปิดโหมด Developer

ถ้าไม่มีมือถือ Android ให้ใช้ Android Emulator

ถาม/ตอบ

คำถาม 1. อธิบายความแตกต่างระหว่าง Web vs Android Auth Flow

ตอบ: Web Authen ใช้ผ่านบราวเซอร์ปกติ Andorid Authen จะใช้บริการผ่าน google play service สำหรับ login

คำถาม 2. reCAPTCHA verifie คืออะไร จะใช้ตอนไหน

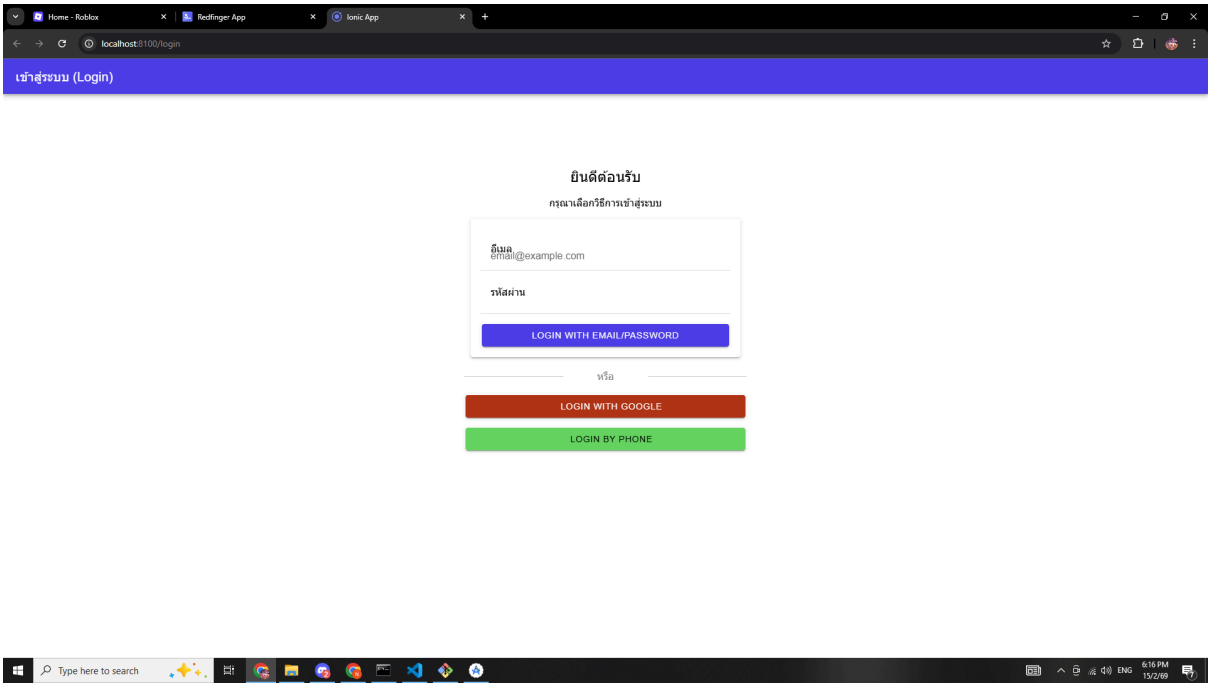
ตอบ:reCAPTCHA Verifier คือระบบที่คนสร้างขึ้นมาเพื่อตรวจจับคนกับbot เป็นการยืนยันความปลอดภัยว่าจะไม่มีหรือน้อยลงที่เป็นบอทมาสร้าง account

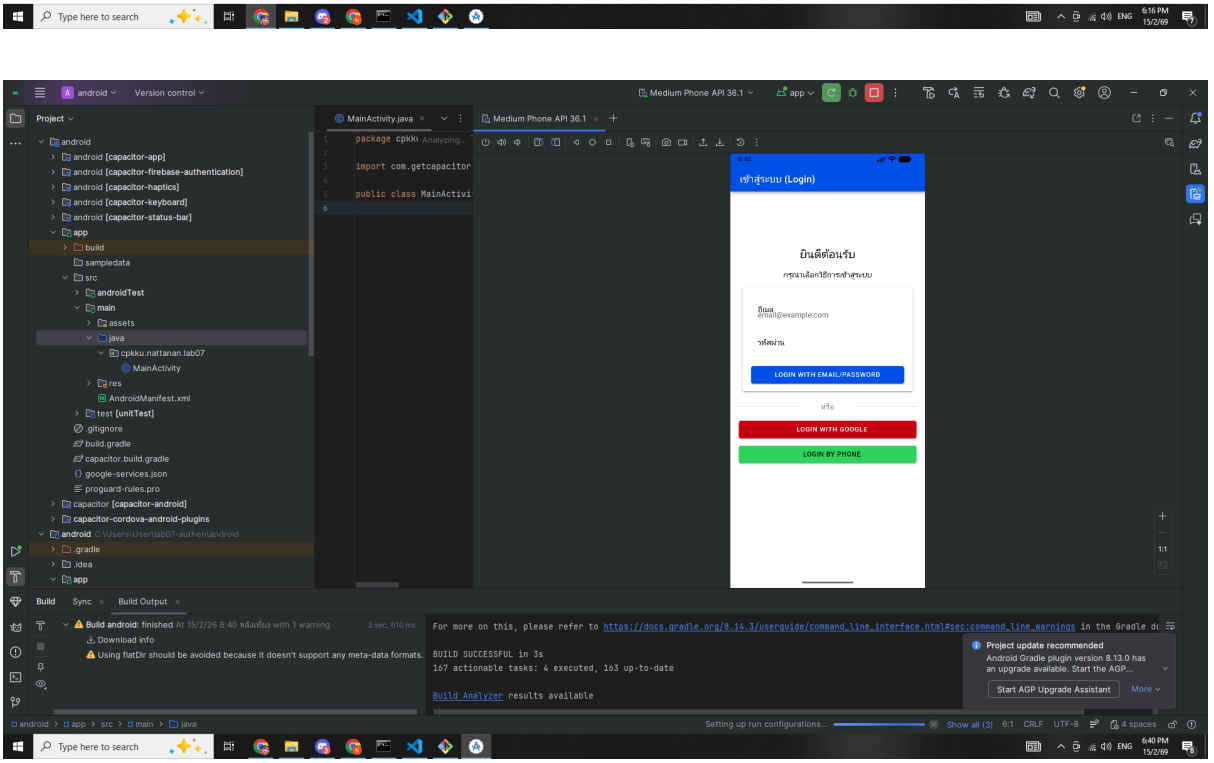
คำถาม 3. อธิบายข้อดีของการออกแบบ interface สำหรับ auth ตามตัวอย่างนี้

ตอบ: การบำรุงรักษา Firebase ไปใช้ระบบอื่น เราแค่แก้ที่ไฟล์ Service
ความเข้มงวดของข้อมูล

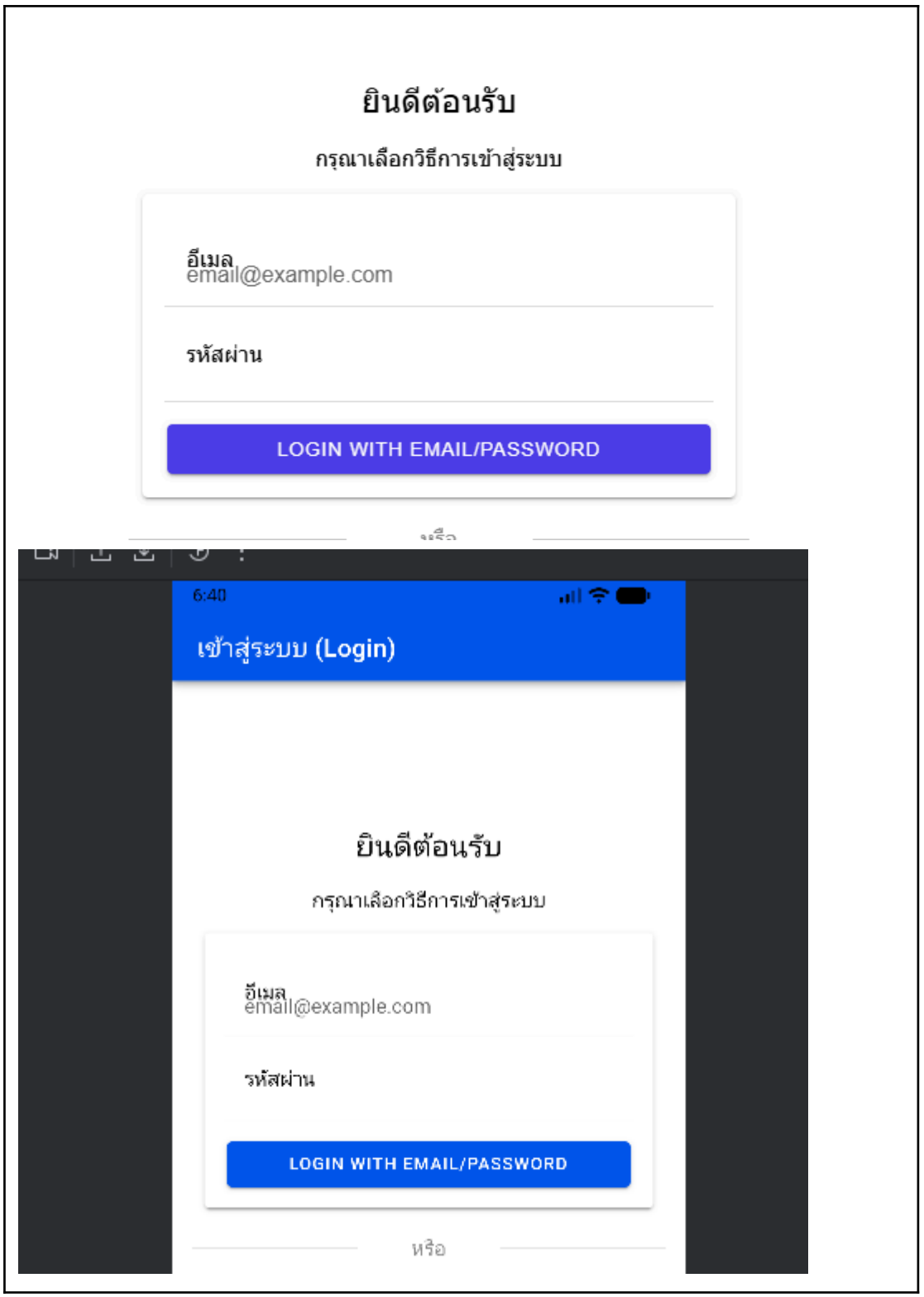
หลักฐานการทำงาน โดยให้แสดงทั้ง Web และ Android

1. ภาพหน้า Login

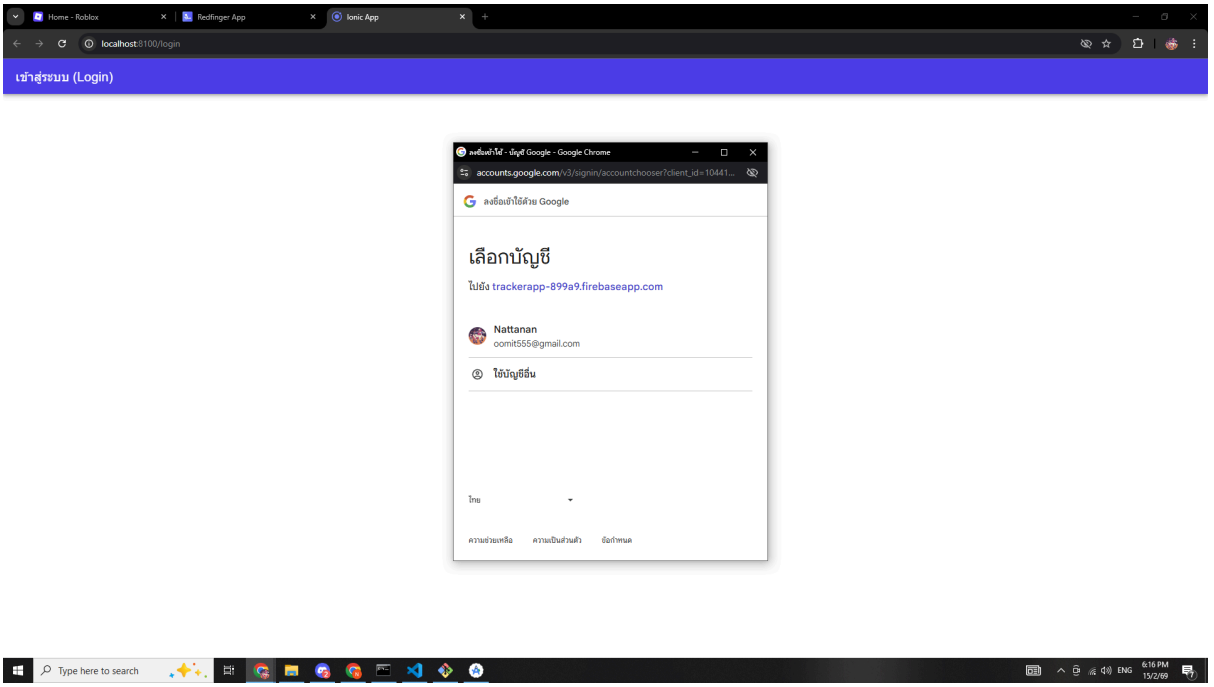




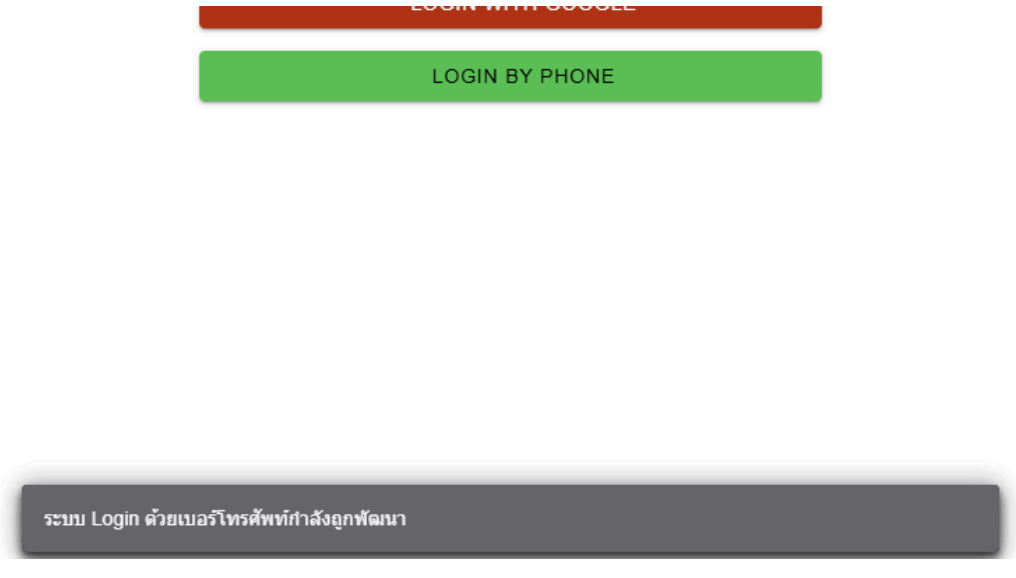
2. ภาพหน้า Login Email/Password



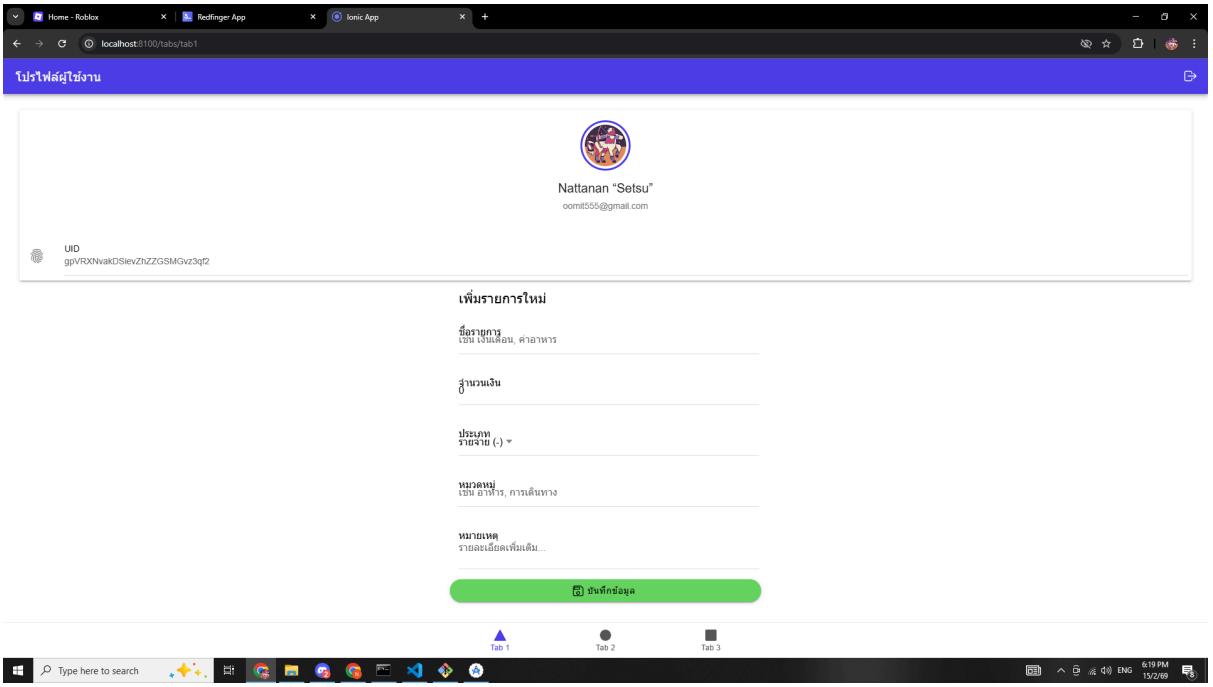
3. ภาพหน้า Login Google



4. ภาพหน้า Login by Phone



5. ภาพหน้าจอ Tab1 แสดงข้อมูล User



หมายเหตุ

- ส่งซอร์สโค้ดใน GitHub Repository
- แนบแสดงหลักฐานการทำงาน
- สามารถใช้ AI ช่วยในการพัฒนาได้ แต่ ห้ามคัดลอกผลงานผู้อื่น