Answer the following questions in 1–2 sentences each:  
1. What is the purpose of the Firebase config object, and where is it used in your code?  
การจัดเก็บข้อมูลการกำหนดค่าเฉพาะสำหรับโปรเจกต์ Firebase ซึ่งทำให้แอปพลิเคชันสามารถเชื่อมต่อและสื่อสารกับบริการต่างๆ ของ Firebase เช่น Authentication, Firestore, Storage และ Realtime Database ได้อย่างถูกต้อง

ในการใช้งานโค้ด config object จะถูกใช้เมื่อทำการ initialize หรือเริ่มต้นใช้งาน Firebase ในแอปพลิเคชัน โดยจะใช้ในไฟล์ JavaScript หลักของแอปพลิเคชัน (เช่น index.js, main.js หรือ app.js) ซึ่งเป็นขั้นตอนแรกก่อนที่คุณจะเรียกใช้บริการ Firebase อื่นๆ

2. Why must you enable sign-in providers before implementing login functionality in Firebase?  
เนื่องจาก Sign-in providers ทำหน้าที่เป็นกลไกหลักในการยืนยันตัวตนของผู้ใช้

Firebase Authentication ไม่ได้มีฐานข้อมูล username และ password ของผู้ใช้โดยตรง แต่จะใช้ providers เหล่านี้ (เช่น Email/Password, Google, Facebook, Twitter, GitHub) เป็นตัวกลางในการตรวจสอบข้อมูลประจำตัว เมื่อเปิดใช้งาน Provider ใด Provider หนึ่ง นั่นคือการบอก Firebase ว่า "ให้ยอมรับการล็อกอินผ่านช่องทางนี้"

ถ้าไม่มีการเปิดใช้งาน provider ที่ต้องการ Firebase จะไม่ทราบว่าควรยอมรับการล็อกอินจากแหล่งใด ทำให้ไม่สามารถสร้างและจัดการบัญชีผู้ใช้ได้

3. What is the difference between signInWithEmailAndPassword() and signInWithPopup()? Provide one scenario where each is appropriate.  
**signInWithEmailAndPassword()**

* วิธีการทำงาน: ฟังก์ชันนี้ใช้สำหรับการล็อกอินด้วยอีเมลและรหัสผ่านโดยตรง ซึ่งข้อมูลจะถูกส่งจากฟอร์มในหน้าเว็บไซต์ของคุณไปยัง Firebase โดยตรง
* ประสบการณ์ผู้ใช้: ผู้ใช้จะอยู่ในหน้าเว็บไซต์ของคุณตลอดเวลา ไม่มีการเปลี่ยนหน้าหรือเปิดหน้าต่างใหม่
* ข้อดี: คุณสามารถควบคุม UI และการไหลของข้อมูลได้อย่างสมบูรณ์แบบ
* ข้อเสีย: คุณต้องสร้างฟอร์มสำหรับอีเมลและรหัสผ่านด้วยตัวเอง และจัดการเรื่องการ validation ข้อมูล
* สถานการณ์ที่เหมาะสม: เหมาะสำหรับแอปพลิเคชันที่เน้นการใช้งานแบบดั้งเดิม หรือต้องการ ควบคุมขั้นตอนการล็อกอินอย่างละเอียด เช่น แอปพลิเคชันที่ผู้ใช้ลงทะเบียนและล็อกอินด้วยอีเมลเป็นหลัก

**signInWithPopup()**

* วิธีการทำงาน: ฟังก์ชันนี้ใช้สำหรับการล็อกอินผ่านผู้ให้บริการภายนอก (Third-party Providers) เช่น Google, Facebook, Twitter, หรือ GitHub โดย Firebase จะเปิดหน้าต่าง Pop-up ใหม่ขึ้นมาเพื่อเรียกหน้าล็อกอินของผู้ให้บริการนั้นๆ
* ประสบการณ์ผู้ใช้: ผู้ใช้จะถูกนำไปยังหน้าต่าง Pop-up ของผู้ให้บริการเพื่อยืนยันตัวตน หลังจากยืนยันแล้ว หน้าต่างจะปิดลงและผู้ใช้จะกลับมาที่แอปพลิเคชันของคุณ
* ข้อดี: สะดวกสำหรับผู้ใช้ที่ไม่ต้องสร้างบัญชีใหม่, มีความปลอดภัยสูงเนื่องจากข้อมูล Credentials ของผู้ใช้จะถูกจัดการโดยผู้ให้บริการภายนอก และคุณไม่ต้องจัดการการ validate ข้อมูล
* ข้อเสีย: คุณไม่สามารถควบคุมหน้า Pop-up ที่เปิดขึ้นมาได้
* สถานการณ์ที่เหมาะสม: เหมาะสำหรับแอปพลิเคชันที่ต้องการให้ผู้ใช้เข้าสู่ระบบได้ง่ายและรวดเร็ว โดยเฉพาะเมื่อต้องการรองรับการล็อกอินผ่าน Social Media ต่างๆ

4. Explain what happens behind the scenes when a user signs in using Google with Firebase. Which part of the SDK handles this process?

**ขั้นตอนเบื้องหลังการทำงาน**

* **การเรียกใช้ SDK:** เมื่อเรียกใช้เมธอด signInWithPopup() หรือ signInWithRedirect() ในโค้ดเพื่อใช้ Google Provider, Firebase SDK จะเริ่มต้นกระบวนการยืนยันตัวตน
* **การเปิดหน้าต่าง Pop-up:** Firebase จะเปิดหน้าต่าง Pop-up หรือเปลี่ยนเส้นทางผู้ใช้ไปยังหน้าล็อกอินของ Google เพื่อให้ผู้ใช้ป้อนข้อมูลบัญชี Google ของตนเอง
* **การตรวจสอบสิทธิ์โดย Google:** ผู้ใช้จะทำการล็อกอินและยินยอมให้แอปพลิเคชันของคุณเข้าถึงข้อมูลพื้นฐาน หลังจากนั้น Google จะทำการตรวจสอบสิทธิ์ของผู้ใช้ และส่ง ID Token และ Access Token ที่เข้ารหัสกลับมาให้ Firebase SDK
* **การแลกเปลี่ยน Token กับ Firebase:** Firebase SDK จะนำ ID Token ที่ได้รับจาก Google ไปแลกเปลี่ยนกับ Firebase Authentication Service เพื่อยืนยันตัวตนของผู้ใช้ Firebase จะทำการตรวจสอบความถูกต้องของ Token และหากผ่านการตรวจสอบ ก็จะสร้างบัญชีผู้ใช้ใหม่ในระบบ Firebase ให้โดยอัตโนมัติ (หากยังไม่มี)
* **การส่ง Credentials กลับมา:** Firebase จะส่ง User Credential ซึ่งประกอบด้วยข้อมูลผู้ใช้ (เช่น displayName, email, photoURL) และ ID Token เฉพาะของ Firebase กลับมายังแอปพลิเคชันของคุณ
* **การจัดการสถานะผู้ใช้:** Firebase SDK จะจัดการสถานะการล็อกอินของผู้ใช้ในแอปพลิเคชันของคุณโดยอัตโนมัติ ทำให้คุณสามารถเข้าถึงข้อมูลผู้ใช้และจัดการการแสดงผลหน้า UI ได้ทันที

5. How does onAuthStateChanged() help manage authentication state across different pages in your app?

เมธอด onAuthStateChanged() ทำหน้าที่เป็น Listener ที่คอยตรวจสอบสถานะการยืนยันตัวตนของผู้ใช้ (Authentication State) แบบ Real-time เมื่อสถานะการล็อกอินของผู้ใช้มีการเปลี่ยนแปลง เช่น ผู้ใช้ล็อกอินสำเร็จ, ผู้ใช้ล็อกเอาท์, Token ของผู้ใช้หมดอายุ onAuthStateChanged() จะทำงานทันที โดยจะให้ข้อมูลผู้ใช้ (user object) ที่ล็อกอินอยู่ หากไม่มีผู้ใช้ล็อกอินอยู่ ข้อมูลที่ส่งกลับมาจะเป็น null ทำให้สามารถใช้ข้อมูลนี้ในการอัปเดต UI ของแอปพลิเคชันได้โดยอัตโนมัติ

6. What will happen if you forget to call onAuthStateChanged() on a protected page? Can an unauthenticated user still access it?  
แอปพลิเคชันของจะไม่สามารถตรวจสอบสถานะการล็อกอินของผู้ใช้ได้ ซึ่งจะทำให้ผู้ใช้ที่ยังไม่ได้ล็อกอินสามารถเข้าถึงหน้าเพจนั้นได้ในบางกรณี

เหตุการณ์ที่เกิดขึ้น

* **ไม่สามารถตรวจสอบสถานะได้:** โดยปกติแล้ว onAuthStateChanged() จะทำหน้าที่เป็นตัวตรวจสอบหลัก เมื่อไม่มีการเรียกใช้ ฟังก์ชันการล็อกอินจะทำงานเพียงครั้งเดียวในช่วงที่ผู้ใช้ล็อกอินสำเร็จ แต่จะไม่ติดตามสถานะอย่างต่อเนื่องเมื่อมีการเปลี่ยนหน้า
* **ผู้ใช้ที่ไม่ได้ล็อกอินสามารถเข้าถึงได้:** หากผู้ใช้เข้าถึง Protected Page โดยตรง (เช่น พิมพ์ URL) แอปพลิเคชันจะไม่ทราบสถานะของเขา และจะแสดงหน้าเพจนั้นตามปกติเนื่องจากไม่มีการตรวจสอบสถานะการล็อกอิน
* **ประสบการณ์ผู้ใช้ที่ไม่ต่อเนื่อง:** ผู้ใช้ที่ล็อกอินแล้วอาจถูกนำกลับไปหน้าล็อกอินทุกครั้งที่เปลี่ยนหน้า เนื่องจากแอปพลิเคชันไม่สามารถ "จดจำ" สถานะของเขาได้

7. Imagine a user logs in successfully but closes the tab without logging out. What happens when they revisit the site? How can you handle this case in your code?

* เมื่อผู้ใช้ล็อกอินสำเร็จและปิดแท็บไปโดยไม่ได้ล็อกเอาท์ สถานะการล็อกอินของผู้ใช้จะยังคงอยู่ ใน Cache ของเบราว์เซอร์หรือ Local Storage โดย Firebase Token จะยังไม่ถูกลบออกจากเครื่องของผู้ใช้
* เมื่อผู้ใช้กลับมาที่เว็บไซต์อีกครั้ง Firebase จะทำการตรวจสอบสถานะการล็อกอินที่ถูกเก็บไว้ทันที และหาก Firebase Token ยังไม่หมดอายุ onAuthStateChanged() Listener จะทำงานและรับรู้ถึงสถานะการล็อกอินของผู้ใช้โดยอัตโนมัติ ทำให้สามารถเข้าถึงข้อมูลของผู้ใช้ได้เลยโดยไม่ต้องให้ผู้ใช้ล็อกอินใหม่อีกครั้ง
* ซึ่งสามารถจัดการกรณีนี้ในโค้ดได้โดยใช้เมธอด onAuthStateChanged() ถูกเรียกใช้เมื่อแอปพลิเคชันเริ่มต้นทำงาน และจะตรวจสอบว่ามี user object อยู่หรือไม่ หากมีแสดงว่าผู้ใช้ยังล็อกอินอยู่ (แม้จะปิดแท็บไปแล้ว) คุณก็สามารถนำผู้ใช้ไปยังหน้าสำหรับผู้ที่ล็อกอินแล้วได้ทันที

8. Design a login flow for a game where players can only access leaderboard and multiplayer features after logging in. Describe what pages or routes would be protected and how you would enforce access control using Firebase Authentication.

**เริ่มจากแบ่งหน้าเพจออกเป็น 2 ประเภทหลัก**

1. **Public Routes (หน้าสาธารณะ):** หน้าที่ผู้ใช้สามารถเข้าถึงได้โดยไม่ต้องล็อกอิน
   * / (Home): หน้าแรกของเกมที่ผู้ใช้สามารถดูข้อมูลเบื้องต้นหรือเริ่มต้นการเล่นได้
   * /login: หน้าสำหรับให้ผู้ใช้ล็อกอิน
   * /register: หน้าสำหรับให้ผู้ใช้สมัครบัญชี
2. **Protected Routes (หน้า Protected):** หน้าที่ผู้ใช้จะเข้าถึงได้ก็ต่อเมื่อล็อกอินแล้วเท่านั้น
   * /leaderboard: หน้าแสดงอันดับผู้เล่น
   * /multiplayer: หน้าสำหรับเข้าสู่โหมดผู้เล่นหลายคน
   * /profile: หน้าข้อมูลส่วนตัวของผู้เล่น

### การบังคับใช้ Access Control ด้วย Firebase Authentication

ใช้เมธอด **onAuthStateChanged()** ของ Firebase Authentication เพื่อตรวจสอบสถานะการล็อกอินของผู้ใช้ในทุกครั้งที่มีการเปลี่ยนเส้นทาง (Routing) โดยมีขั้นตอนการทำงานดังนี้:

1. **การตั้งค่า Listener:** ตั้งค่า onAuthStateChanged() ในไฟล์หลักของแอปพลิเคชัน (เช่น App.js) เพื่อให้ Listener นี้ทำงานอยู่ตลอดเวลา
2. **การตรวจสอบสถานะผู้ใช้:** เมื่อผู้ใช้พยายามเข้าถึง Protected Route (เช่น /leaderboard) จะตรวจสอบสถานะของผู้ใช้
   * **กรณีที่ 1: ผู้ใช้ล็อกอินแล้ว (user is not null)** \* อนุญาตให้ผู้ใช้เข้าถึงหน้านั้นได้ตามปกติ
   * **กรณีที่ 2: ผู้ใช้ยังไม่ได้ล็อกอิน (user is null)**
     + เปลี่ยนเส้นทาง (redirect) ผู้ใช้ไปยังหน้า /login ทันที เพื่อบังคับให้ล็อกอินก่อนที่จะเข้าถึงฟีเจอร์ที่ต้องการ