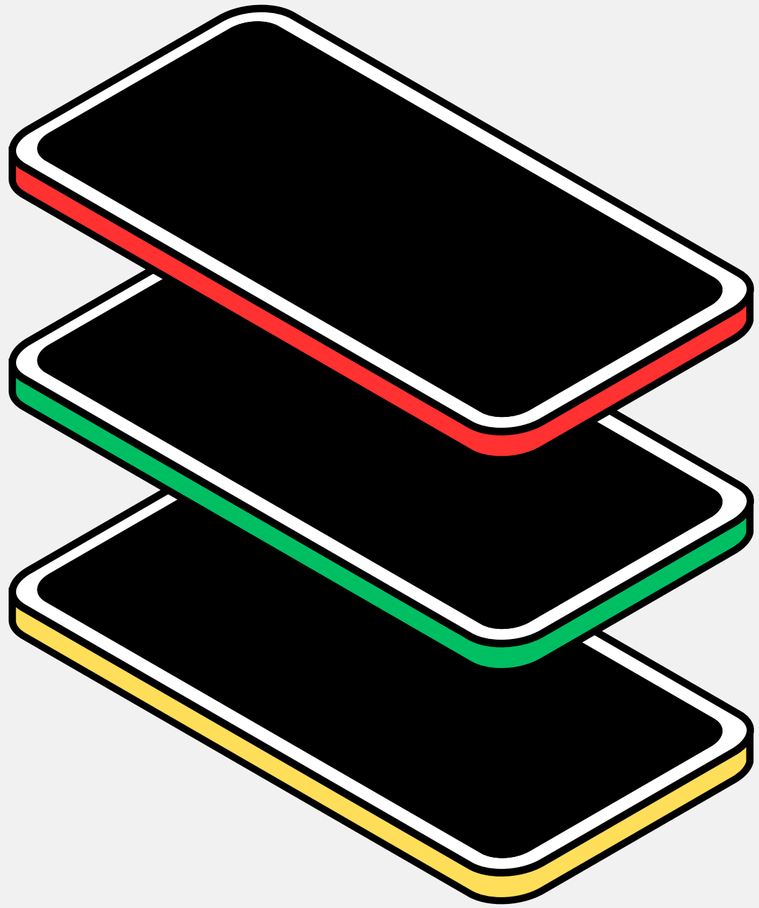
บทที่ 4

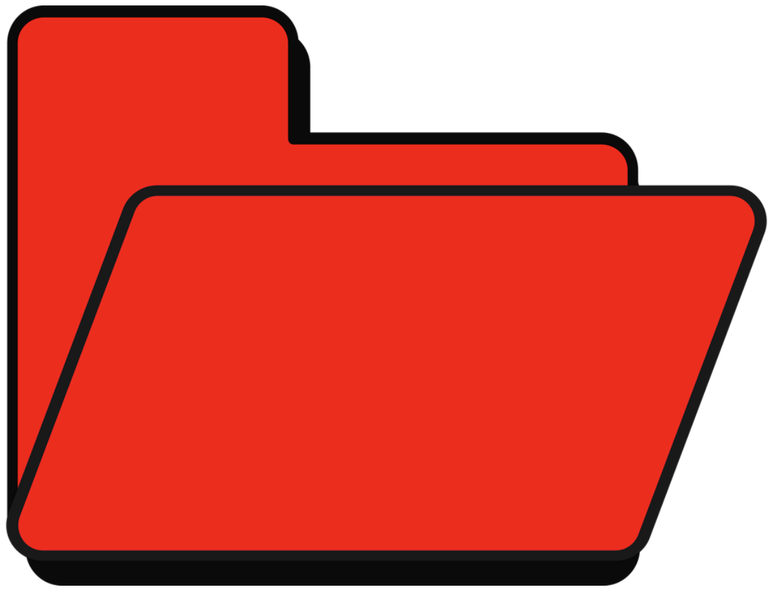
แนวทางการพัฒนาแอปพลิเคชันบนอุปกรณ์เคลื่อนที่



ดร.พงษ์ดนัย จิตตวิสุทธิกุล

Topic

4 หัวข้อหลัก:



- การพัฒนาแบบเนทีฟ (Native Development)
- การพัฒนาแบบครอสแพลตฟอร์ม (Cross-Platform Development)
- วิเคราะห์กรณีศึกษา: เลือกเทคโนโลยีที่เหมาะสม
- แนะนำโครงงานประจำภาคการศึกษา

Objective behavior

2 วัตถุประสงค์เชิงพฤติกรรม:

อธิบายความแตกต่าง ข้อดี และข้อเสียของการพัฒนาแอปพลิเคชันแบบเนทีฟ (Native) และแบบครอสแพลตฟอร์ม (Cross-Platform) ได้

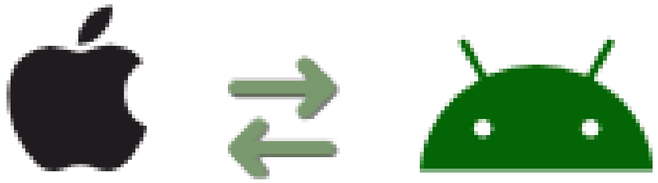
วิเคราะห์สถานการณ์ของโครงงานและให้เหตุผลในการเลือกใช้แนวทางการพัฒนาแบบเนทีฟ (Native) หรือครอสแพลตฟอร์ม (Cross-Platform) ได้อย่างเหมาะสม

เลือก "เครื่องมือ" ให้ถูกงาน

(Choosing the Right Blueprint)

"การตัดสินใจเลือกระหว่าง Native และ Cross-Platform

คือ จุดเริ่มต้นที่กำหนดชะตากรรมของแอปพลิเคชันของคุณ"



iOS Android

การพัฒนาแบบเนทีฟ (Native)

นิยาม: การสร้างแอปพลิเคชันสำหรับแพลตฟอร์มใดแพลตฟอร์มหนึ่งโดยเฉพาะ โดยใช้ภาษาและเครื่องมือที่เป็นทางการ (Official Tools)

ภาษา: Swift

IDE: Xcode

ภาษา: Kotlin

IDE: Android Studio

จุดเด่นและจุดด้อย (Native)

จุดเด่น

- ประสิทธิภาพสูงสุด
- UX ดีเยี่ยม ลื่นไหล เป็นธรรมชาติ
- ความปลอดภัยสูง และใช้ฟีเจอร์ใหม่ได้ทันที

จุดด้อย

- ต้นทุนสูง (ต้องทำ 2 แอป)
- ใช้เวลานาน (Time-to-market ช้า)
- ต้องใช้ทีมพัฒนา 2 ทีม
  (iOS Team + Android Team)

การพัฒนาแบบครอสแพลตฟอร์ม (Cross-Platform)

นิยาม: เขียนโค้ดเพียงชุดเดียว (Single Codebase) แต่นำไปรันได้บนหลาย แพลตฟอร์ม เปรียบเสมือนการใช้ "ภาษากลาง" ในการสื่อสาร

ใช้ UI Components ของเนทีฟ: React Native (Meta) ใช้ UI ของ Native ผ่าน Bridge

วาด UI ของตัวเอง: Flutter (Google) วาด UI เองทุกพิกเซล (เหมือน Game Engine)

คอมไพล์เป็นเนทีฟ: Xamarin/.NET MAUI (Microsoft) ใช้ API&UI ของ Native ผ่าน Bindings

จุดเด่นและจุดด้อย (Cross-Platform)

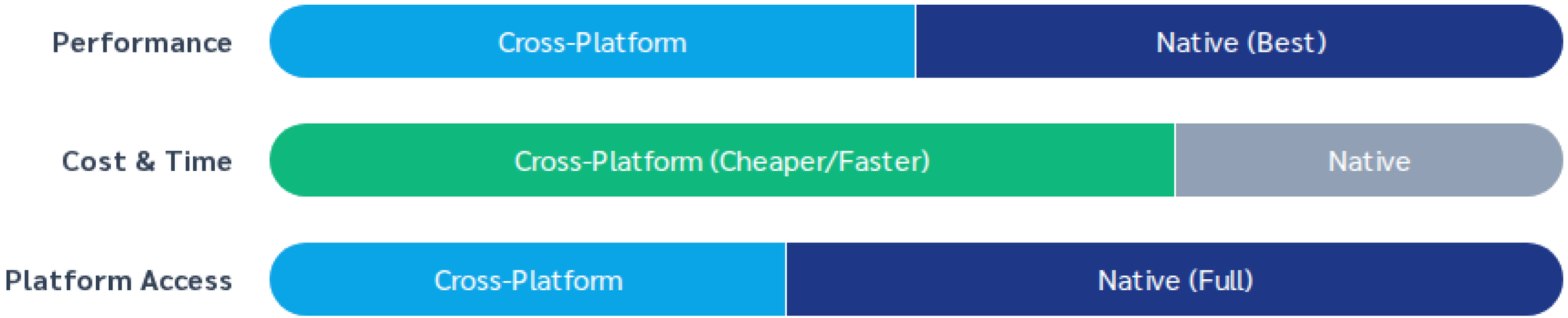
จุดเด่น

- ประหยัดต้นทุนและเวลา (Code Once)
- ดูแลรักษาง่าย (Single Codebase)
- เปิดตัวพร้อมกันได้ทุกตลาด (Faster Time-to-Market)

จุดด้อย

- ประสิทธิภาพอาจต้อยกว่า (Overhead)
- อาจรออัปเดตฟีเจอร์ใหม่ช้ากว่า Native
- พึ่งพา 3rd Party Plugin

บทสรุปการเปรียบเทียบ (The Trade-off)



"ไม่มีคำตอบที่ดีที่สุด มีแต่คำตอบที่เหมาะสมที่สุดกับโจทย์"

กรณีศึกษา 1: ConnectSphere

โจทย์: Startup งบจำกัด ต้องการ MVP เร็วที่สุด

คำตัดสิน: Cross-Platform

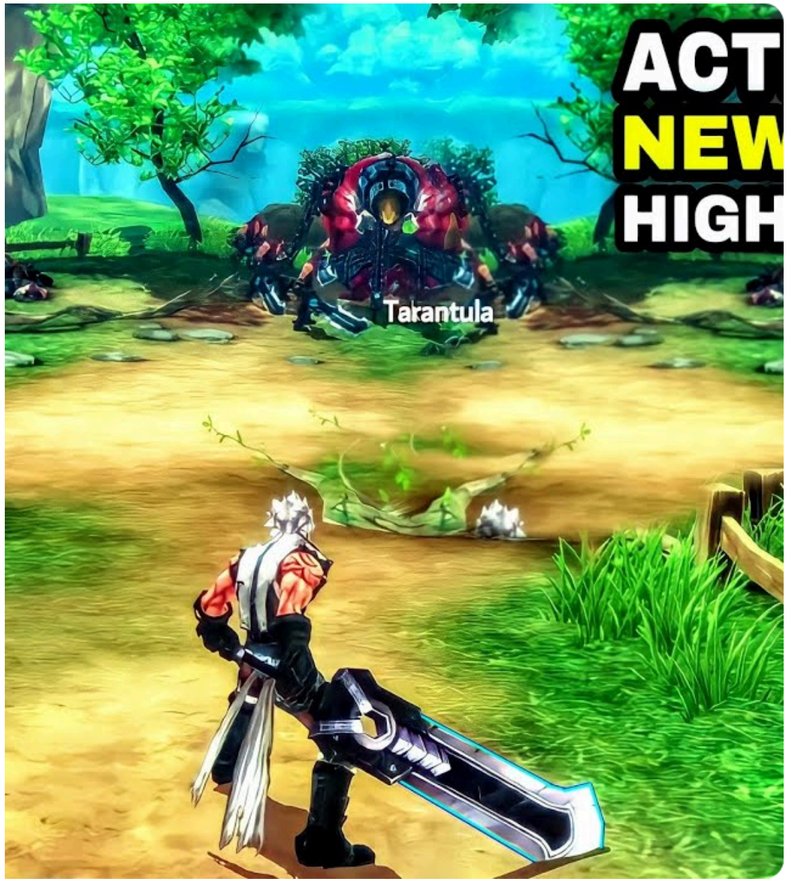
เหตุผล: ต้องการความเร็วในการเข้าตลาด (Time-to-market) และประหยัดงบจ้างทีม React Native ตอบโจทย์เพราะทีมมีพื้นฐาน Web Tech

กรณีศึกษา 2: Quantum Realms

โจทย์: เกม 3D กราฟิกสูง FPS นิ่ง

คำตัดสิน: Native (หรือ Game Engine)

เหตุผล: ประสิทธิภาพคือกุญแจสำคัญ
ต้องการเข้าถึง GPU (Metal/Vulkan)
โดยตรง ไม่มี Overhead ของ Bridge

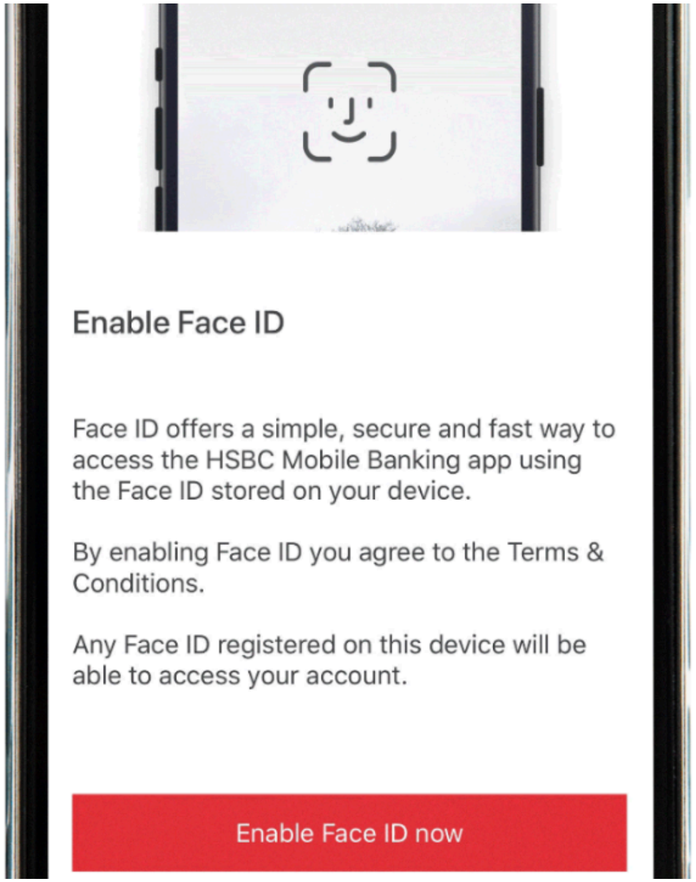


กรณีศึกษา 3: SecureBank

โจทย์: แอปธนาคาร ความปลอดภัยสูงสุด

คำตัดสิน: Native

เหตุผล: ความปลอดภัย (Security) และความน่าเชื่อถือสำคัญที่สุด ต้องใช้ Biometrics API ของระบบโดยตรง ไม่เสี่ยงกับ Plugin ภายนอก

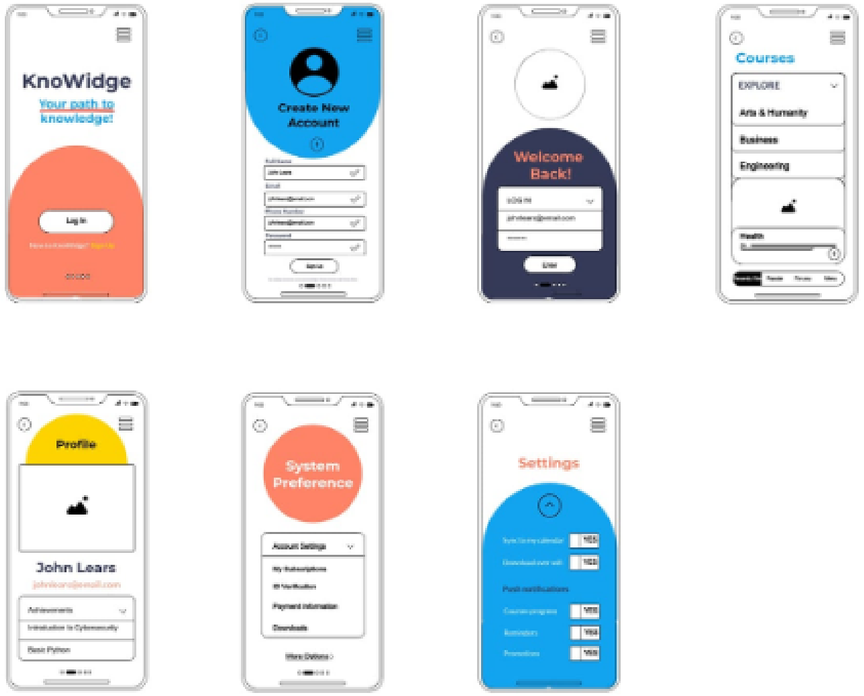


โครงงานประจำภาคการศึกษา

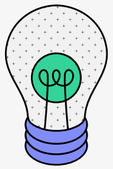
หัวข้อ: "Mobile App for Education"

ให้นักศึกษาจับกลุ่มและออกแบบแอปพลิเคชันเพื่อการศึกษา (เช่น สอนภาษา, ฝึกทำโจทย์)

- Proposal 1 หน้า: ชื่อแอป, แนวคิด, กลุ่มเป้าหมาย
- Key Decision: เลือก Native หรือ Cross-Platform?
- Reasoning: ให้เหตุผลประกอบการตัดสินใจ โดยอ้างอิงจากบทเรียน



Questions & Discussion



"... การเป็นนักพัฒนาที่ดีย่อมต้องสามารถวิเคราะห์ข้อดีและข้อเสียของแต่ละแนวทาง และเลือกใช้เครื่องมือที่เหมาะสมกับปัญหาที่ต้องการแก้ไขได้อย่างมีหลักการ ..."