แนวทางในการจัดกิจกรรมภาคปฏิบัติ

หัวข้อการเรียนรู้ :
การออกแบบส่วนติดต่อกับผู้ใช้ด้วย SwiftUI (Design an Interface with SwiftUI)

ระยะเวลาในการทำกิจกรรม: 180 นาที

วัตถุประสงค์ในการทำกิจกรรม :

1. การอธิบายแนวคิดการออกแบบส่วนติดต่อผู้ใช้ในฐานะโครงสร้างของการสื่อสารระหว่างระบบกับผู้ใช้ โดยเชื่อมโยงหลักการออกแบบเชิงผู้ใช้ตามกรอบ Human Interface Guidelines (HIG) กับบริบทของการพัฒนาแอปพลิเคชันด้วย SwiftUI ได้อย่างถูกต้อง
2. การเลือกใช้ VStack, HStack และ ZStack ในฐานะโครงสร้างเชิงความหมาย (semantic layout structures) ที่ใช้กำหนดลำดับข้อมูล การจัดกลุ่ม และบริบทของเนื้อหาบนหน้าจอได้อย่างเหมาะสม
3. การออกแบบโครงสร้างส่วนติดต่อผู้ใช้ของแอปพลิเคชันอย่างเป็นระบบ โดยสามารถกำหนดลำดับความสำคัญของข้อมูล (visual hierarchy) และจัดกลุ่มองค์ประกอบของ UI ให้สอดคล้องกับพฤติกรรมการรับรู้ของผู้ใช้บนแพลตฟอร์ม iOS
4. การพัฒนาส่วนติดต่อกับผู้ใช้ของแอปพลิเคชันอย่างง่ายด้วย SwiftUI โดยสามารถแยกโครงสร้าง View ออกเป็นส่วนย่อยที่มีบทบาทชัดเจนได้อย่างเหมาะสม

ความคิดรวบยอด:

กิจกรรมการเรียนรู้ชุดนี้มุ่งพัฒนาความรู้และความเข้าใจของผู้เรียนเกี่ยวกับการออกแบบส่วนติดต่อผู้ใช้ในฐานะ “โครงสร้างของการสื่อสาร” ระหว่างระบบกับผู้ใช้ การทำความเข้าใจหลักการออกแบบเชิงผู้ใช้ตามกรอบ Human Interface Guidelines (HIG) และการนำหลักการดังกล่าวมาประยุกต์ใช้จริงผ่านการพัฒนา UI ด้วย SwiftUI ตลอดกระบวนการเรียนรู้ ผู้เรียนจะได้พัฒนาความรู้เชิงแนวคิดเกี่ยวกับลำดับความสำคัญของข้อมูล (visual hierarchy) การจัดกลุ่มข้อมูลอย่างมีความหมาย และการลดภาระทางความคิดของผู้ใช้ โดยใช้ VStack, HStack และ ZStack ในฐานะโครงสร้างเชิงความหมาย (semantic layout structures) ที่เชื่อมโยงโครงสร้างของโค้ดเข้ากับโครงสร้างของการรับรู้ นอกจากนี้ ผู้เรียนยังได้ฝึกทักษะการคิดเชิงออกแบบ (design thinking) ผ่านการวิเคราะห์โจทย์ การวางโครงสร้างหน้าจอก่อนเขียนโค้ด และการอธิบายเหตุผลของการตัดสินใจเชิงออกแบบในแต่ละส่วนของ UI อย่างเป็นระบบ

บทความสำหรับอ่านประกอบการทำกิจกรรม
https://ajthiti.gitbook.io/develop-in-swift/getting-started/design-an-interface

แนวทางในการจัดกิจกรรมการเรียนรู้

ขั้นนำเข้าสู่บทเรียน (Warm-up)

การนำเข้าสู่บทเรียนในกิจกรรมการเรียนรู้เรื่อง Design an Interface with SwiftUI มีเป้าหมายสำคัญเพื่อปรับกรอบความคิดของผู้เรียนจากการมอง UI ในเชิง “ความสวยงาม” หรือ “การจัดวางองค์ประกอบ” ไปสู่การมอง UI ในฐานะโครงสร้างของการสื่อสารและการรับรู้ของผู้ใช้ ผู้สอนควรใช้ช่วงเวลานี้เพื่อสร้างบริบททางความคิด (conceptual context) และทำให้ผู้เรียนเห็นความสำคัญของการออกแบบเชิงโครงสร้างก่อนการเขียนโค้ด

ผู้สอนอาจเริ่มต้นด้วยการตั้งคำถามเชิงประสบการณ์ โดยไม่กล่าวถึง SwiftUI หรือโค้ดในทันที เช่น “เมื่อเปิดแอปหนึ่งขึ้นมา สิ่งใดทำให้เรารู้ทันทีว่าแอปนี้ใช้ทำอะไร” หรือ “เหตุใดบางแอปจึงใช้งานได้ง่าย แม้ไม่เคยใช้มาก่อน” จากนั้นให้ผู้เรียนแลกเปลี่ยนความคิดเห็นสั้น ๆ เพื่อชี้ให้เห็นว่า ความเข้าใจของผู้ใช้ไม่ได้เกิดจากความสวยงามเพียงอย่างเดียว แต่เกิดจากการจัดลำดับข้อมูล การจัดกลุ่มองค์ประกอบ และการสื่อสารที่ชัดเจน และในตอนท้ายควรสรุปประเด็นให้ชัดเจนว่า **UI ที่ดีคือ UI ที่ช่วยให้ผู้ใช้ ‘ไม่ต้องคิดมาก’** และเชื่อมโยงประเด็นนี้เข้ากับแนวคิดเรื่อง Cognitive load และ User perception โดยยังไม่ลงรายละเอียดเชิงทฤษฎีมากเกินไป

เมื่อผู้เรียนเริ่มตระหนักถึงบทบาทของ UI ในการสื่อสารแล้ว ผู้สอนจึงแนะนำว่า Apple ได้สรุปแนวคิดเหล่านี้ออกมาเป็นกรอบที่เรียกว่า Human Interface Guidelines (HIG) และเน้นย้ำว่า **HIG ไม่ใช่ชุดกฎทางเทคนิค แต่เป็นกรอบคิดเพื่อช่วยผู้ออกแบบตัดสินใจได้อย่างมีเหตุผล**
ผู้สอนควรยกตัวอย่างสั้น ๆ เช่น

- หากผู้ใช้ต้องใช้เวลานานในการค้นหาข้อมูลหลัก แสดงว่า UI ยังไม่ชัดเจน
- หากแต่ละหน้าจอมีรูปแบบต่างกัน ผู้ใช้จะต้องเรียนรู้ใหม่ซ้ำ ๆ

เพื่อปูทางไปสู่แนวคิด Clarity, Consistency และ Deference to Content ซึ่งจะถูกขยายความในขั้นการเรียนรู้ต่อไป

ก่อนเข้าสู่เนื้อหา ผู้สอนควรตั้งคำถามชวนคิด อย่างเช่น **“ถ้าเรามอง UI เป็นโครงสร้างของความหมาย โครงสร้างของโค้ดควรสะท้อนสิ่งนั้นอย่างไร”** จากนั้นจึงชี้ให้ผู้เรียนเห็นว่า SwiftUI ถูกออกแบบมาเพื่อสนับสนุนแนวคิดนี้โดยตรงผ่านโครงสร้างแบบ Declarative และการใช้ VStack, HStack และ ZStack ซึ่งในบทเรียนนี้จะไม่ได้สอนในฐานะเพียงเครื่องมือจัดตำแหน่ง แต่จะสอนในฐานะ **ภาษาเชิงโครงสร้างของการออกแบบ UI**

ขั้นการเรียนรู้ (Learn)

บทเรียนนี้ มิได้มุ่งเน้นให้ผู้เรียนเขียน SwiftUI ให้ได้ผลลัพธ์รวดเร็วที่สุด หากแต่ให้ความสำคัญกับการพัฒนาความเข้าใจเชิงเหตุผลในการออกแบบส่วนติดต่อผู้ใช้ โดยผู้เรียนจะต้องสามารถ เข้าใจเหตุผลเชิงออกแบบ → เชื่อมโยงแนวคิดเข้ากับโครงสร้างของโค้ด → อธิบายการตัดสินใจของตนเองได้อย่างมีหลักการกิจกรรมทั้งหมดในขั้นนี้จึงถูกออกแบบภายใต้แนวคิด **คิดก่อนเขียน และอธิบายก่อนปรับแก้** เพื่อให้ผู้เรียนพัฒนาแนวคิดเชิงโครงสร้างควบคู่ไปกับทักษะการเขียนโปรแกรมอย่างยั่งยืน

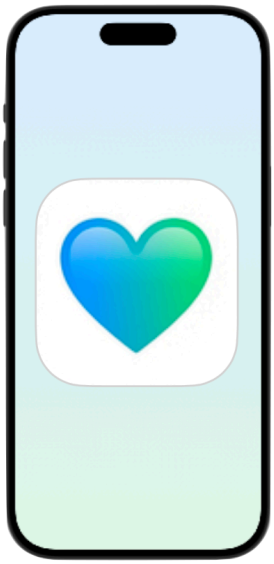
กิจกรรมที่ 1 : Understanding & Designing Semantic UI

การจัดกิจกรรมที่ 1 มีบทบาทสำคัญในฐานะ “การวางรากฐานทางความคิด” ให้กับผู้เรียนก่อนเข้าสู่การพัฒนา UI ด้วย SwiftUI ในเชิงปฏิบัติ ผู้สอนควรมองกิจกรรมนี้ว่า **ไม่ใช่ช่วงเวลาสำหรับการเร่งเขียนโค้ดหรือแสดงความสามารถทางเทคนิค** แต่เป็นช่วงเวลาสำหรับปรับกรอบความคิดของผู้เรียนจากการมอง UI เป็นเพียงงานด้านความสวยงามหรือการจัดวางองค์ประกอบ ไปสู่ การมอง UI ในฐานะโครงสร้างของการสื่อสารและการรับรู้ของผู้ใช้

แอปนี้ใช้ทำอะไร ?

ในช่วงเริ่มต้นของกิจกรรม **ผู้สอนควรใช้คำถามเชิงประสบการณ์เพื่อเชื่อมโยงบทเรียนเข้ากับการใช้งานแอปในชีวิตประจำวันของผู้เรียน** โดยหลีกเลี่ยงการกล่าวถึง SwiftUI หรือโค้ดในทันที โดยคำถามที่ใช้ควรมุ่งให้ผู้เรียนสะท้อนว่า **ทำไมบางแอปจึงใช้งานง่ายตั้งแต่ครั้งแรก ในขณะที่บางแอปกลับทำให้สับสน** คำตอบที่ได้จากผู้เรียนจะช่วยเปิดประเด็นไปสู่แนวคิดเรื่อง ความชัดเจนของข้อมูล การจัดลำดับความสำคัญ และการลดภาระทางความคิดของผู้ใช้ ผู้สอนควรสรุปให้เห็นภาพร่วมว่า **UI ที่ดีคือ UI ที่ช่วยให้ผู้ใช้ “ไม่ต้องคิดมาก” และทำให้การรับรู้เป็นไปอย่างเป็นธรรมชาติ**

เมื่อผู้เรียนเริ่มตระหนักถึงบทบาทของ UI ในการสื่อสารแล้ว ผู้สอนควรนำเข้าสู่การวิเคราะห์โครงสร้างของ UI ผ่านตัวอย่างหน้าจอ หรือ mockup โดยไม่เปิดโค้ดในขั้นแรก การวิเคราะห์ควรมุ่งไปที่การตั้งคำถาม เช่น ส่วนใดคือข้อมูลหลัก สายตาของผู้ใช้ควรถูกนำไปที่จุดใดก่อน และหากสลับตำแหน่งองค์ประกอบบางส่วนจะส่งผลต่อความเข้าใจอย่างไร วิธีการนี้จะช่วยให้ผู้เรียนเห็นว่า UI มีโครงสร้างและลำดับในตัวเอง ไม่ได้เป็นเพียงการวางองค์ประกอบแบบสุ่ม จากนั้นผู้สอนจึงเชื่อมโยงการวิเคราะห์ดังกล่าวเข้ากับแนวคิดของ VStack, HStack และ ZStack โดยอธิบายว่า Stack แต่ละประเภททำหน้าที่เป็นโครงสร้างเชิงความหมายที่สะท้อนลำดับการอ่าน การจัดกลุ่มข้อมูล และการสร้างบริบทของเนื้อหา มากกว่าการเป็นเพียงเครื่องมือจัดตำแหน่งบนหน้าจอ



Health Tips App

หน่วยงานด้านสาธารณสุขในมหาวิทยาลัยต้องการพัฒนาแอปขนาดเล็กสำหรับเผยแพร่คำแนะนำด้านสุขภาพพื้นฐานแก่บุคลากรและนักศึกษา

- แอปดังกล่าวจะถูกใช้งานบน iPhone เป็นหลัก
- มีวัตถุประสงค์เพื่อให้ผู้ใช้สามารถเปิดอ่านคำแนะนำได้อย่างรวดเร็วในช่วงเวลาสั้นๆ เช่น ระหว่างพักเรียนหรือพักทำงาน
- แอปนี้ไม่ได้มุ่งเน้นการบันทึกข้อมูลหรือการโต้ตอบซับซ้อน ไม่มีระบบสมาชิก ไม่มีการเก็บประวัติการใช้งาน และไม่มีระบบแจ้งเตือน
- จุดประสงค์หลักคือ “การนำเสนอข้อมูลที่อ่านง่าย เข้าใจเร็ว และนำไปใช้ได้ทันที”

สิ่งที่สำคัญจึงไม่ใช่การจัดวางองค์ประกอบอย่างไรให้สวย แต่คือ
“ผู้ใช้ควรเข้าใจอะไรภายในไม่กี่วินาทีแรก”

ในขั้นต่อมา ผู้สอนควรให้ผู้เรียนลงมือออกแบบโครงสร้างของหน้าจอจาก Design Brief โดยเน้นการคิดเชิงโครงสร้างก่อนการเขียนโค้ด ผู้เรียนควรถูกกระตุ้นให้วาดหรืออธิบายโครงสร้างหน้าจอในเชิงแนวคิด ระบุบทบาทของแต่ละส่วนอย่างชัดเจน เช่น ส่วนใดทำหน้าที่กำหนดบริบท ส่วนใดเป็นเนื้อหาหลัก และส่วนใดเป็นข้อมูลรอง การออกแบบในขั้นนี้ไม่ควรถูกตัดสินว่าถูกหรือผิด แต่ควรถูกใช้เป็นฐานสำหรับการอภิปรายถึงผลกระทบเชิงการรับรู้ของผู้ใช้ การแลกเปลี่ยนความคิดเห็นระหว่างผู้เรียนจะช่วยให้เห็นว่าการตัดสินใจเชิงโครงสร้างที่แตกต่างกันสามารถนำไปสู่ประสบการณ์ผู้ใช้ที่แตกต่างกันได้

ในช่วงท้ายของกิจกรรม ผู้สอนควรจัดให้มีการสะท้อนผลการเรียนรู้เพื่อช่วยให้ผู้เรียนตกผลึกแนวคิดที่ได้เรียนรู้ คำถามสะท้อนควรมุ่งให้ผู้เรียนอธิบายว่า **มุมมองต่อการออกแบบ UI ของตนเองเปลี่ยนไปอย่างไร** และ **โครงสร้างเชิงความหมายมีบทบาทอย่างไรต่อความเข้าใจของผู้ใช้**

ตัวอย่าง Prompt สำหรับให้ผู้เรียนใช้กับ LM Studio ในกิจกรรมนี้

- “จากโจทย์เพื่อการพัฒนาแอป ข้อมูลใดควรถูกมองว่าเป็นข้อมูลหลัก และข้อมูลใดเป็นข้อมูลรอง เพราะเหตุใด”
- “ถ้าผู้ใช้มีเวลาเพียงไม่กี่วินาทีในการดูหน้าจอนี้ เขาควรเข้าใจอะไรเป็นสิ่งแรก”

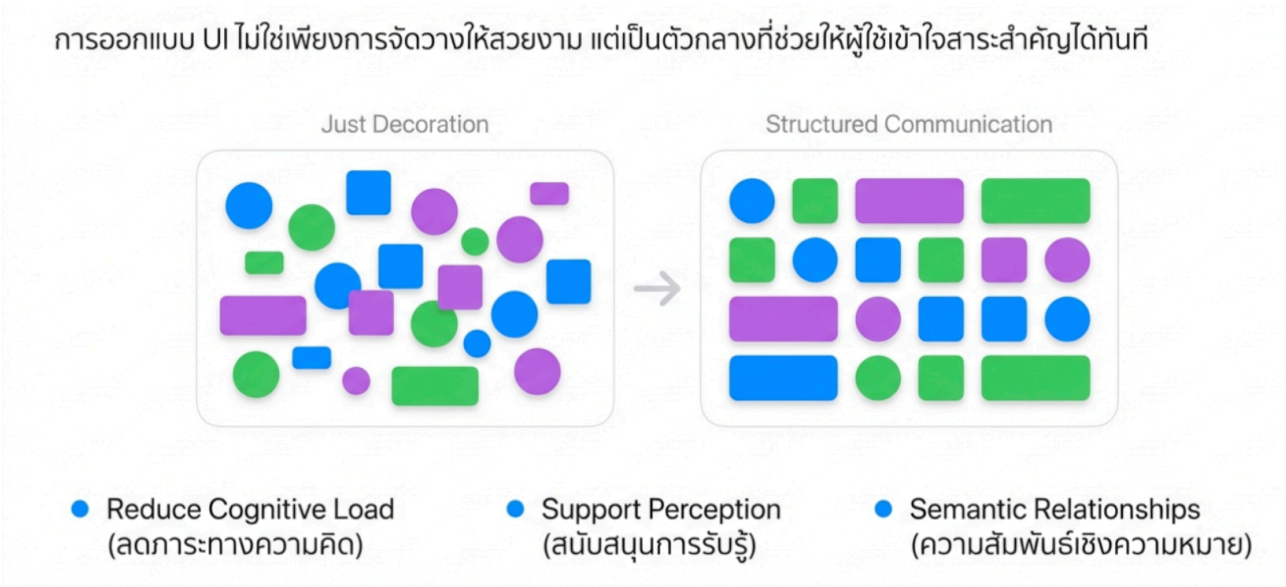
- “องค์ประกอบใดในหน้าจอที่สามารถตัดออกได้โดยไม่กระทบความเข้าใจหลักของผู้ใช้”
- “สายตาของผู้ใช้ควรถูกนำไปที่จุดใดก่อนในหน้าจอนี้ และเหตุใด”
- “องค์ประกอบใดในหน้าจอที่มีบทบาทนำสายตาโดยไม่ต้องอ่าน”
- “หากสลับตำแหน่งขององค์ประกอบหลักกับองค์ประกอบรอง จะส่งผลต่อความเข้าใจของผู้ใช้อย่างไร”
- “ถ้าต้องอธิบายหน้าจอนี้ในเชิงโครงสร้าง ควรแบ่งออกเป็นกี่ส่วน และแต่ละส่วนทำหน้าที่อะไร”

หลังใช้ LM Studio ผู้เรียนต้องสามารถตอบคำถามต่อไปนี้ได้ด้วยตนเอง

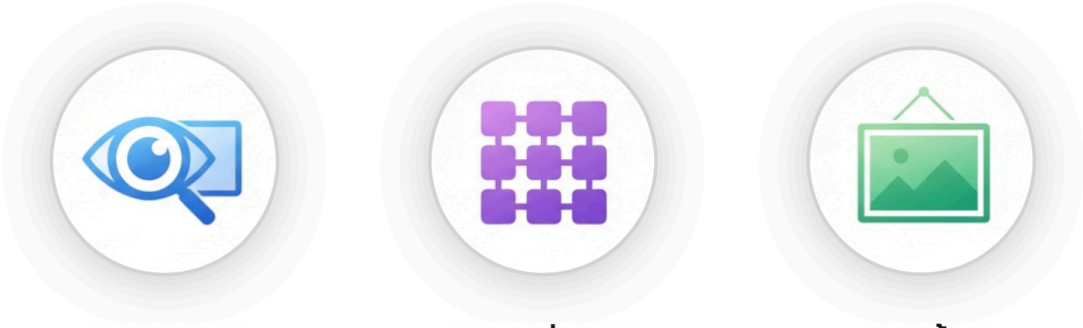
- “หลังจากวิเคราะห์หน้าจอนี้ ฉันมองการออกแบบ UI เปลี่ยนไปอย่างไร”
- “สิ่งใดเกี่ยวกับ UI ที่ฉันเคยมองข้าม แต่เริ่มเห็นความสำคัญมากขึ้น”
- “แนวคิดใดจากการวิเคราะห์นี้ที่ฉันคิดว่าสำคัญต่อการเขียน SwiftUI มากที่สุด”
- “AI ช่วยให้ฉันคิดอะไรเพิ่มขึ้น และฉันเห็นด้วยหรือไม่ เพราะเหตุใด”

กิจกรรมที่ 2 : Implementing & Reflecting with SwiftUI

การจัดกิจกรรมที่ 2 มีบทบาทเป็นช่วงเวลาสำคัญในการเชื่อมโยงแนวคิดเชิงนามธรรมจากกิจกรรมที่ 1 เข้าสู่การปฏิบัติจริงด้วย SwiftUI ผู้สอนควรมอง Session นี้ว่าเป็น “พื้นที่ของการแปลงความคิดเป็นโครงสร้างของโค้ด” มากกว่าการสอนเทคนิคการเขียนโปรแกรมเป็นรายบรรทัด เป้าหมายหลักไม่ใช่การทำให้แอปทำงานได้ครบถ้วนหรือสวยงามที่สุด แต่คือการทำให้ผู้เรียน **เห็นความสัมพันธ์ระหว่างเหตุผลเชิงออกแบบกับโครงสร้างของ View hierarchy อย่างชัดเจน**



ในช่วงเริ่มต้น ผู้สอนควรเริ่มต้นด้วยการทบทวนกรอบคิดหลักว่า **UI คือโครงสร้างของการสื่อสาร ไม่ใช่เพียงการจัดวางองค์ประกอบให้สวยงาม** ผู้สอนอาจชี้ให้ผู้เรียนเห็นภาพเปรียบเทียบระหว่างการจัดวางแบบไร้โครงสร้างกับการจัดวางแบบมีโครงสร้าง เพื่อย้ำว่า UI ที่ดีทำหน้าที่ช่วยลดภาระทางความคิด (cognitive load) สนับสนุนการรับรู้ (perception) และสร้างความสัมพันธ์เชิงความหมายระหว่างองค์ประกอบต่าง ๆ บนหน้าจอ



ความชัดเจน (Clarity)

ผู้ใช้ต้องเข้าใจวัตถุประสงค์ของหน้าจอได้ทันที สื่อสารสาระสำคัญได้อย่างรวดเร็ว

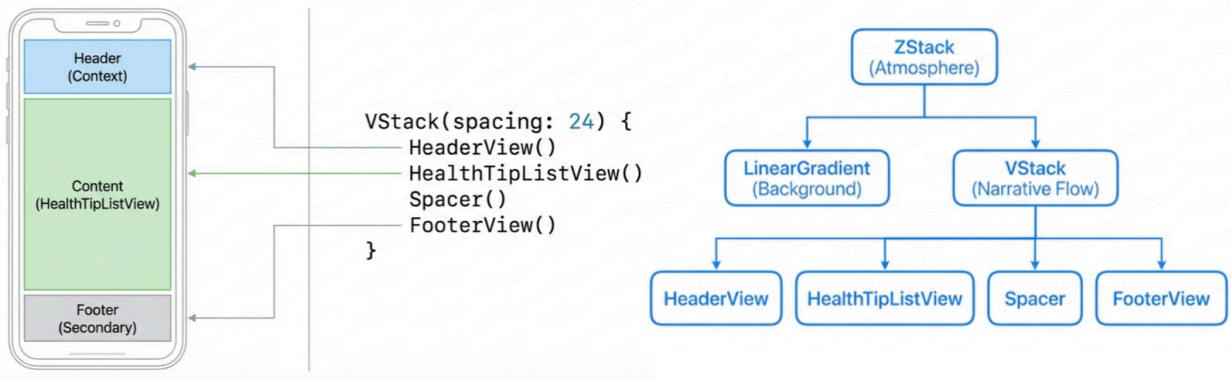
ความสม่ำเสมอ (Consistency)

ใช้รูปแบบที่สอดคล้องกับพฤติกรรมการใช้งาน เพื่อให้สามารถคาดเดาการใช้งานได้

เน้นเนื้อหา (Deference to Content)

UI ต้องทำหน้าที่สนับสนุนเนื้อหา ไม่แย่งความสนใจจากเนื้อหาหลัก

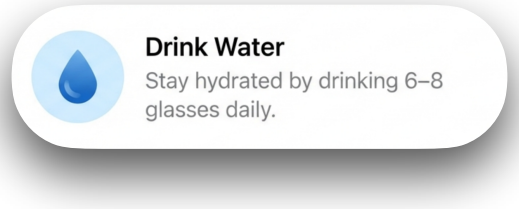
จากนั้น ผู้สอนควรเชื่อมโยงเข้าสู่กรอบ **Human Interface Guidelines (HIG)** โดยทบทวนหลักการสำคัญสามประการ ได้แก่ ความชัดเจน (Clarity) ความสม่ำเสมอ (Consistency) และการเน้นเนื้อหา (Deference to Content) ผู้สอนควรย้ำให้ผู้เรียนเข้าใจว่า หลักการเหล่านี้ไม่ใช่ checklist ทางเทคนิค แต่เป็นกรอบคิดที่ใช้กำกับการตัดสินใจเชิงออกแบบในทุกระดับของหน้าจอ การตั้งคำถามเชิงสะท้อน เช่น **“ผู้ใช้ควรเข้าใจวัตถุประสงค์ของหน้าจอนี้ภายในกี่วินาที”** หรือ **“องค์ประกอบใดในหน้าจอที่ควรถอยออกมาเพื่อสนับสนุนเนื้อหา”** จะช่วยให้ผู้เรียนเห็นว่าหลัก HIG จะถูกนำมาใช้จริงในกิจกรรมนี้อย่างไร



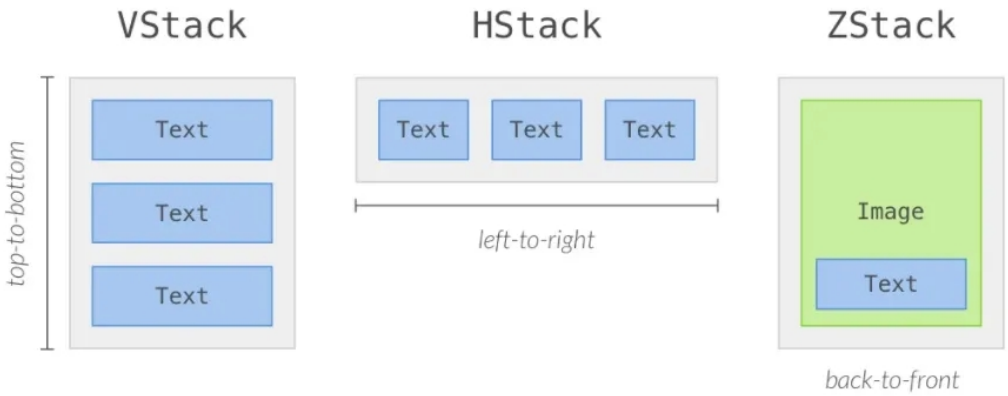
เมื่อกรอบคิดเชิงปรัชญาถูกวางไว้อย่างชัดเจนแล้ว ผู้สอนควรนำผู้เรียนกลับมาที่แนวคิดเรื่อง **การกำหนดลำดับความสำคัญของข้อมูล (Visual Hierarchy)** ตามสไลด์ที่แสดงให้เห็นโครงสร้างหน้าจอในเชิงแนวคิดก่อนการเขียนโค้ด ผู้สอนควรเน้นว่า ใน Session 2 ผู้เรียนจะไม่เริ่มจากการพิมพ์โค้ดทันที แต่จะเริ่มจากการ “อ่านหน้าจอ” และ “อธิบายโครงสร้าง” เสมอ การทบทวนภาพที่แสดง Header, Content และ Footer จะช่วยให้ผู้เรียนเห็นความสัมพันธ์ระหว่างลำดับการอ่านของผู้ใช้กับโครงสร้างของ View hierarchy ใน SwiftUI อย่างเป็นรูปธรรม

ต่อเนื่องจากนั้น ผู้สอนควรทบทวนแนวคิดเรื่อง **Typographic Hierarchy** โดยชี้ให้เห็นว่า การใช้ขนาดตัวอักษร น้ำหนัก และสี ไม่ใช่เรื่องของความสวยงาม แต่เป็นเครื่องมือสำคัญในการนำสายตาและลดภาระทางความคิด ผู้สอนอาจตั้ง

คำถามนำว่า **หากหัวข้อและข้อความอธิบายมีขนาดและน้ำหนักเท่ากัน ผู้ใช้จะต้องใช้ความพยายามเพิ่มขึ้นอย่างไรในการแยกแยะความสำคัญ** การเชื่อมโยงแนวคิดนี้กับ SwiftUI จะช่วยให้ผู้เรียนเข้าใจว่าการเลือกใช้ .font(.headline) หรือ .font(.caption) เป็นการตัดสินใจเชิงความหมาย ไม่ใช่เพียงการตกแต่ง



ในลำดับถัดมา ผู้สอนควรเน้นแนวคิดเรื่อง **การจัดกลุ่มข้อมูลอย่างมีความหมาย (Meaningful Group)** โดยอธิบายว่า หน่วยข้อมูลหนึ่งหน่วย เช่น Card คำแนะนำด้านสุขภาพ ควรถูกมองเป็นข้อมูลชุดเดียวที่ประกอบด้วย ไอคอน หัวข้อ และคำอธิบาย การทบทวนสไลด์ส่วนนี้จะช่วยให้ผู้เรียนเข้าใจว่า ใน Session 2 เมื่อเริ่มสร้าง HealthTipCardView พวกเขาไม่ได้กำลังสร้าง “กล่องสวย ๆ” แต่กำลังสร้างหน่วยข้อมูลที่ต้องสื่อสารสาระสำคัญได้ด้วยตนเอง



สุดท้าย ผู้สอนควรทบทวนแนวคิดแกนกลางของบทเรียนจากสไลด์ที่อธิบายว่า **การจัดวางองค์ประกอบคือการกำหนดความสัมพันธ์เชิงความหมาย (semantic relationships)** ไม่ใช่เพียงการควบคุมตำแหน่งเชิงพิกัด การอธิบายบทบาทของ VStack, HStack และ ZStack ในฐานะโครงสร้างที่สะท้อนลำดับการอ่าน การจัดกลุ่มข้อมูลในระดับเดียวกัน และความสัมพันธ์ระหว่างองค์ประกอบหลักกับองค์ประกอบรอง จะช่วยเตรียมผู้เรียนให้พร้อมสำหรับการแปลงแนวคิดเหล่านี้ไปสู่โค้ด SwiftUI อย่างมีเหตุผลในกิจกรรมถัดไป

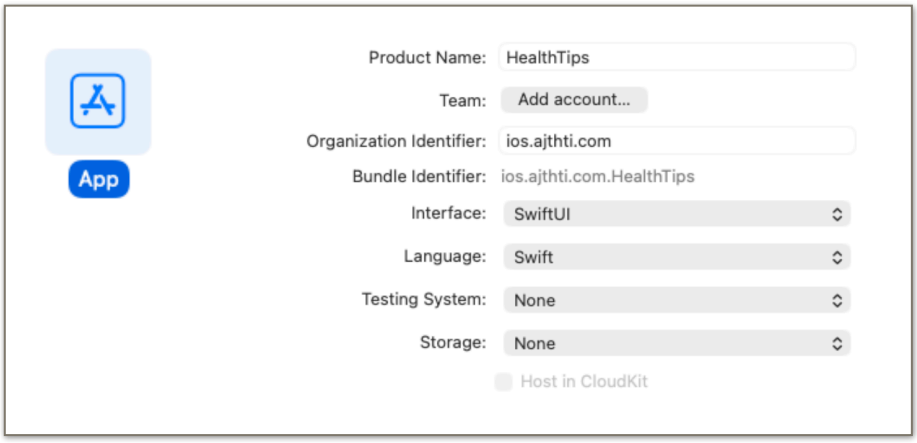
กิจกรรมที่ 3 : การสร้างแอป HealthTips

การจัดกิจกรรมในช่วงการสร้างแอปถือเป็นหัวใจสำคัญของบทเรียนนี้ เพราะ เป็นช่วงเวลาที่ผู้เรียนจะได้**แปลงกรอบคิดเชิงนามธรรมจาก กิจกรรมที่ 1 ให้กลายเป็นโครงสร้างของ SwiftUI อย่างเป็นรูปธรรม** ผู้สอนควรทำความเข้าใจก่อนว่า จุดมุ่งหมายของกิจกรรมนี้ไม่ใช่การทำให้ผู้เรียน “เขียนโค้ดให้เสร็จตามสไลด์” แต่คือการทำให้ผู้เรียน **มองเห็นความสัมพันธ์ระหว่างเหตุผลเชิงออกแบบกับโครงสร้างของ View hierarchy ในแต่ละขั้นตอน**

การเริ่มต้นสร้างโปรเจกต์

ในขั้นแรก ผู้สอนควรชี้ให้ผู้เรียนเห็นว่า การตั้งค่าโปรเจกต์ SwiftUI ไม่ใช่เพียงขั้นตอนเชิงเทคนิค แต่เป็นการกำหนด ขอบเขตและบริบทของแอป ตั้งแต่ ชื่อแอป (HealthTips) ไปจนถึง ภาษาที่เลือกใช้ ผู้สอนควรย้ำว่า **การเลือก SwiftUI เป็น Interface หมายถึงการยอมรับแนวคิด Declarative UI ซึ่งโครงสร้างของโค้ดจะสะท้อนโครงสร้าง**

ของหน้าจอโดยตรง การตั้งกรอบคิดนี้ตั้งแต่ต้นจะช่วยให้ผู้เรียนไม่หลงไปกับรายละเอียดเชิงเครื่องมือ และเข้าใจว่าทุกบรรทัดโค้ดที่กำลังจะเขียนต่อจากนี้มีบทบาทเชิงความหมาย



การสร้าง ContentView ในฐานะสถาปัตยกรรมของหน้าจอ

```
struct ContentView: View {
    var body: some View {
        ZStack {
            LinearGradient(
                colors: [
                    Color.blue.opacity(0.15),
                    Color.green.opacity(0.15)
                ],
                startPoint: .top,
                endPoint: .bottom
            )
            .ignoresSafeArea()

            VStack(spacing: 24) {
                HeaderView()
                HealthTipListView()
                Spacer()
                FooterView()
            }
            .padding()
        }
    }
}
```

เมื่อเข้าสู่การเขียนโค้ดในส่วนของ ContentView ผู้สอนควรหยุดอธิบายที่คำว่า **ZStack** และ **VStack** อย่างมีนัยสำคัญ โดยเน้นว่า หน้านี้ไม่ใช่เพียงตัวอย่างโค้ด แต่เป็น **การประกาศโครงสร้างระดับบนของหน้าจอทั้งหมด** การใช้ ZStack เพื่อวางพื้นหลังแบบ LinearGradient ควรถูกอธิบายในฐานะการกำหนด “บริบทของประสบการณ์ผู้ใช้” ขณะที่ VStack ภายในทำหน้าที่กำหนดลำดับการรับรู้ข้อมูลจากบนลงล่าง

ผู้สอนควรตั้งคำถามนำ เช่น “หากตัด ZStack ออก หน้าจอนี้จะสูญเสียบริบทอะไรไป” หรือ “การจัดลำดับ Header → Content → Footer สอดคล้องกับพฤติกรรมการอ่านของผู้ใช้อย่างไร” การตั้งคำถามเช่นนี้จะช่วยให้ผู้เรียนมองโค้ดเป็นการตัดสินใจเชิงออกแบบ ไม่ใช่เพียงการเรียงคำสั่งให้แสดงผลได้

การสร้าง HeaderView: การกำหนดบริบทและลำดับความสำคัญ

```
struct HeaderView: View {
    var body: some View {
        VStack(alignment: .leading, spacing: 8) {

            Image(systemName: "heart.fill")
                .font(.largeTitle)
                .foregroundStyle(.blue.opacity(0.6))

            Text("Health Tips")
                .font(.system(size: 34, weight: .bold))

            Text("Daily wellness guidance.")
                .font(.subheadline)
                .foregroundStyle(.secondary)
        }
        .frame(maxWidth: .infinity, alignment: .leading)
    }
}
```

ในส่วนของ HeaderView ผู้สอนควรเน้นบทบาทของส่วนนี้ในฐานะ **จุดกำหนดบริบท (context-setting element)** ของทั้งหน้าจอ การใช้ VStack(alignment: .leading) ไม่ควรถูกอธิบายเพียงว่าเป็นการจัดชิดซ้าย แต่ควรเชื่อมโยงกับพฤติกรรมการอ่านของผู้ใช้และหลัก Consistency ของ HIG รวมทั้ง ควรชี้ให้เห็นว่า การเลือกขนาดตัวอักษร น้ำหนัก และสีของข้อความ เป็นการสร้าง typographic hierarchy เพื่อบอกผู้ใช้ว่า อะไรคือข้อมูลหลัก และข้อมูลรอง และตั้งคำถามกับผู้เรียนว่า **หากข้อความทุกบรรทัดมีขนาดเท่ากัน ผู้ใช้จะต้องใช้ความพยายามมากขึ้นอย่างไรในการทำความเข้าใจหน้าจอ**

การสร้าง FooterView: การจัดการข้อมูลรองอย่างมีความหมาย (สไลด์หน้า 4)

```
struct FooterView: View {
    var body: some View {
        Text("Stay healthy, feel great!")
            .font(.footnote)
            .foregroundStyle(.secondary)
            .multilineTextAlignment(.center)
            .frame(maxWidth: .infinity)
    }
}
```

เมื่อเข้าสู่ FooterView ผู้สอนควรใช้โอกาสนี้อธิบายแนวคิด **Deference to Content** อย่างเป็นรูปธรรม โดยชี้ให้เห็นว่า Footer ถูกออกแบบให้มีน้ำหนักเชิงสายตาน้อยกว่าส่วนอื่น ผ่านการใช้ .footnote และสีรอง การมีอยู่ของ Footer ไม่ได้มีเป้าหมายเพื่อดึงความสนใจ แต่เพื่อสนับสนุนประสบการณ์ผู้ใช้โดยไม่รบกวนเนื้อหาหลัก ผู้สอนควรตั้งคำถามว่า **หาก Footer ถูกออกแบบให้เด่นเท่า Header จะส่งผลต่อการรับรู้ของผู้ใช้อย่างไร**

การสร้าง HealthTipCardView: หน่วยข้อมูลที่สมบูรณ์

```
struct HealthTipCardView: View {

    let title: String
    let detail: String
    let systemImage: String
    let iconColor: Color

    var body: some View {
        HStack(alignment: .top, spacing: 16) {

            ZStack {
                Circle()
                    .fill(iconColor.opacity(0.2))
                    .frame(width: 40, height: 40)

                Image(systemName: systemImage)
                    .foregroundStyle(iconColor)
            }

            VStack(alignment: .leading, spacing: 4) {
                Text(title)
                    .font(.headline)

                Text(detail)
                    .font(.caption)
                    .foregroundStyle(.secondary)
            }

            Spacer()
        }
        .padding()
        .background(
            RoundedRectangle(cornerRadius: 16)
                .fill(Color.white)
        )
        .shadow(color: .black.opacity(0.05), radius: 5, x: 0, y: 3)
    }
}
```

ส่วน HealthTipCardView เป็นจุดที่ผู้สอนควรให้ความสำคัญเป็นพิเศษ เพราะเป็นตัวอย่างชัดเจนของ **การออกแบบเชิงโครงสร้างในระดับ component** ผู้สอนควรอธิบายว่า Card หนึ่งใบไม่ใช่กล่องตกแต่ง แต่เป็น “หน่วยข้อมูล” ที่ต้องสามารถสื่อสารสาระสำคัญได้ด้วยตนเอง

การใช้ HStack เพื่อจัดกลุ่มไอคอนและข้อความ ควรถูกอธิบายว่าเป็นการจัดข้อมูลที่มีความสัมพันธ์ในระดับเดียวกัน ขณะที่ ZStack ภายในไอคอนทำหน้าที่สร้างบริบทเชิงภาพให้กับสัญลักษณ์ ผู้สอนควรถามผู้เรียนว่า หากแยกองค์ประกอบเหล่านี้ออกจากกัน ผู้ใช้จะยังเข้าใจสาระของคำแนะนำข้อนั้นได้หรือไม่

การรวม Card เป็นรายการด้วย HealthTipListView

```
struct HealthTipListView: View {
    var body: some View {
        VStack(spacing: 16) {
            HealthTipCardView(
                title: "Drink Water",
                detail: "Stay hydrated by drinking 6–8 glasses daily.",
                systemImage: "drop.fill",
                iconColor: .blue
            )
```

```
            HealthTipCardView(
                title: "Walk 30 Mins",
                detail: "Light exercise improves heart health.",
                systemImage: "figure.walk",
                iconColor: .green
            )

            HealthTipCardView(
                title: "Sleep 8 Hours",
                detail: "Quality sleep supports body recovery.",
                systemImage: "moon.fill",
                iconColor: .purple
            )
        }
    }
}
```

ในขั้นสุดท้ายของการสร้างแอป ผู้สอนควรชี้ให้เห็นว่า HealthTipListView เป็นตัวอย่างของ **ความสม่ำเสมอเชิงโครงสร้าง (structural consistency)** การใช้ VStack(spacing: 16) เพื่อเรียง Card ที่มีรูปแบบเหมือนกัน ช่วยลดภาระทางความคิดของผู้ใช้ เพราะผู้ใช้ไม่ต้องเรียนรู้รูปแบบใหม่ทุกครั้งที่เลื่อนหน้าจอ และควรย้ำว่าความสม่ำเสมอในที่นี้ไม่ได้มีเป้าหมายเพื่อความสวยงาม แต่เพื่อสนับสนุนการรับรู้และการสแกนข้อมูลอย่างรวดเร็ว ซึ่งเป็นพฤติกรรมการใช้งานหลักบนอุปกรณ์พกพา

โดยตลอดการสร้างแอป ผู้สอนควรทำหน้าที่เป็นผู้ตั้งคำถามและสะท้อนความคิด มากกว่าผู้แก้โค้ดให้ผู้เรียนโดยตรง ทุกครั้งที่ผู้เรียนเขียนโค้ด ผู้สอนควรถามว่า **“โครงสร้างนี้สื่อสารอะไรกับผู้ใช้”** และ **“หากปรับโครงสร้างนี้ ผู้ใช้จะรับรู้ต่างไปอย่างไร”** แนวทางนี้จะช่วยให้ผู้เรียนพัฒนา SwiftUI ในฐานะเครื่องมือสำหรับการออกแบบประสบการณ์ผู้ใช้ ไม่ใช่เพียงทักษะการเขียนโค้ด

ตัวอย่าง Prompt สำหรับให้ผู้เรียนใช้กับ LM Studio ในกิจกรรมนี้

- “โครงสร้าง UI ของแอปที่ฉันสร้าง สื่อสารวัตถุประสงค์ของหน้าจอได้ชัดเจนเพียงใด”ข
- “โครงสร้าง View hierarchy ใน SwiftUI ของฉันสะท้อนลำดับการรับรู้ของผู้ใช้อย่างไร”
- “การใช้ VStack, HStack และ ZStack ในแอปนี้ ช่วยอธิบายความสัมพันธ์ของข้อมูลได้ชัดเจนหรือไม่”
- “UI ของฉันสอดคล้องกับหลัก Clarity ในจุดใด และยังขาดในจุดใด”
- “การออกแบบส่วนใดในแอปนี้สะท้อนแนวคิด Deference to Content ได้ชัดเจนที่สุด”

หลังใช้ LM Studio ผู้เรียนต้องสามารถตอบคำถามต่อไปนี้ได้ด้วยตนเอง

- “หลังจากทำกิจกรรมนี้ ฉันเข้าใจการออกแบบ UI แตกต่างจากเดิมอย่างไร”
- “แนวคิดใดจากกิจกรรมนี้ที่ฉันคิดว่าจะส่งผลต่อการเขียน SwiftUI ของฉันในอนาคต”
- “ฉันสามารถอธิบายเหตุผลในการใช้โครงสร้างของโค้ดที่เขียนได้มากขึ้นหรือไม่ อย่างไร”

ขั้นสรุปบทเรียน (Conclusion)

ช่วงสรุปบทเรียนควรถูกใช้เป็นช่วงเวลาในการ **ตกผลึกความคิดร่วมกับผู้เรียน** มากกว่าการทบทวนเนื้อหาเชิงเทคนิค ผู้สอนควรเริ่มจากการชวนผู้เรียนย้อนมองตนเองว่า มุมมองต่อการออกแบบ

UI ก่อนและหลังเรียนแตกต่างกันอย่างไร เพื่อให้ผู้เรียนตระหนักว่าการเรียนรู้ในบทเรียนนี้คือการเปลี่ยนกรอบคิด ไม่ใช่เพียงการเพิ่มทักษะการเขียนโค้ด

ผู้สอนควรย้ำแนวคิดหลักของบทเรียนว่า **UI คือโครงสร้างของการสื่อสารและการรับรู้ของผู้ใช้** ไม่ใช่เพียงการจัดวางองค์ประกอบให้สวยงาม การวิเคราะห์หน้าจอใน Session 1 และการพัฒนาแอปใน Session 2 แสดงให้เห็นว่า ลำดับข้อมูล การจัดกลุ่ม และการสร้างบริบท สามารถถูกถ่ายทอดออกมาเป็นโครงสร้างของ SwiftUI ได้อย่างเป็นระบบ จากนั้น ผู้สอนควรเชื่อมโยงกิจกรรมทั้งหมดเข้ากับหลัก **Human Interface Guidelines** โดยเน้นว่า Clarity, Consistency และ Deference to Content เป็นกรอบคิดที่ช่วยกำกับการตัดสินใจเชิงออกแบบ ไม่ใช่กฎที่ต้องท่องจำ พร้อมชวนผู้เรียนยกตัวอย่างจากผลงานของตนเองว่า โครงสร้างใดสะท้อนหลักการเหล่านี้ได้ชัดเจนที่สุด

ผู้สอนควรสรุปบทบาทของ **VStack, HStack และ ZStack** ว่าเป็น “ภาษาเชิงโครงสร้าง” สำหรับการออกแบบ UI แบบ Declarative หากผู้อื่นสามารถอ่านโค้ดแล้วเข้าใจโครงสร้างหน้าจอได้ นั่นคือสัญญาณของการออกแบบที่ดี

ในช่วงท้าย ควรเปิดโอกาสให้ผู้เรียนสะท้อนสั้น ๆ ว่า สิ่งใดที่เข้าใจชัดขึ้น สิ่งใดที่ท้าทาย และแนวคิดใดจะนำไปใช้ต่อ พร้อมย้ำว่า ความสำเร็จของบทเรียนนี้ไม่ได้อยู่ที่ความซับซ้อนของแอป แต่คือการที่ผู้เรียนสามารถ **อธิบายเหตุผลของโครงสร้าง UI ที่ตนเองสร้างขึ้นได้อย่างมีหลักการ** หากผู้เรียน “อธิบายได้ว่าทำไมโค้ดจึงเป็นแบบนี้” แสดงว่าบทเรียนบรรลุเป้าหมายแล้ว

คำถามเพื่อการสะท้อนผลการเรียนรู้จากการทำกิจกรรม

1. อธิบายความหมายของการออกแบบส่วนติดต่อผู้ใช้ (UI) ในฐานะ “โครงสร้างของการสื่อสาร” พร้อมยกตัวอย่างมาโดยสังเขป
2. อธิบายบทบาทของ VStack, HStack และ ZStack ใน SwiftUI ในฐานะโครงสร้างเชิงความหมาย
3. หลักการใดใน Human Interface Guidelines (Clarity, Consistency, Deference to Content) ถูกนำมาใช้ชัดเจนที่สุดในงานของคุณ และสะท้อนออกมาในโครงสร้าง UI อย่างไร
4. อธิบายเหตุผลในการแยก UI ออกเป็น View ย่อย (เช่น HeaderView, Content, FooterView) และผลที่เกิดขึ้นต่อความอ่านง่ายของโค้ดและความเข้าใจของผู้ใช้
5. จงอธิบายว่าโครงสร้าง View hierarchy ใน SwiftUI ของคุณช่วยสนับสนุนการนำสายตา (visual hierarchy) และลดภาระทางความคิดของผู้ใช้อย่างไร
6. อธิบายบทบาทของ Card หรือ หน่วยข้อมูล (information unit) ในแอปที่คุณพัฒนา และเหตุใดองค์ประกอบภายใน Card จึงควรถูกจัดกลุ่มร่วมกัน
7. จากกิจกรรมทั้งหมด คุณสามารถอธิบายได้หรือไม่ว่า “โค้ด SwiftUI ที่ดี” ควรมีลักษณะอย่างไรในเชิงการออกแบบ พร้อมให้เหตุผลสนับสนุน

คำศัพท์สำคัญ

User Interface (UI)
ส่วนติดต่อระหว่างระบบกับผู้ใช้ ทำหน้าที่เป็นช่องทางในการสื่อสารข้อมูล การโต้ตอบ และการรับรู้ของผู้ใช้

Human Interface Guidelines (HIG)
กรอบแนวคิดการออกแบบของ Apple ที่มุ่งเน้นการออกแบบ UI ให้สอดคล้องกับพฤติกรรมและความคาดหวังของผู้ใช้

Clarity (ความชัดเจน)
หลักการออกแบบที่เน้นให้ผู้ใช้เข้าใจวัตถุประสงค์ของหน้าจอและข้อมูลสำคัญได้ทันที

Consistency (ความสม่ำเสมอ)
การใช้รูปแบบ โครงสร้าง และพฤติกรรมของ UI อย่างคงเส้นคงวา เพื่อลดภาระการเรียนรู้ของผู้ใช้

Deference to Content (การให้ความสำคัญกับเนื้อหา)
หลักการที่กำหนดให้ UI ทำหน้าที่สนับสนุนเนื้อหา ไม่แย่งความสนใจจากข้อมูลหลัก

Visual Hierarchy (ลำดับความสำคัญเชิงสายตา)
การจัดลำดับองค์ประกอบบนหน้าจอเพื่อกำหนดว่าสายตาของผู้ใช้ควรมองสิ่งใดก่อนหลัง

Semantic Layout (โครงสร้างเชิงความหมาย)
การจัดวางองค์ประกอบ UI โดยคำนึงถึงความหมายและความสัมพันธ์ของข้อมูล ไม่ใช่เพียงตำแหน่งเชิงพิกัด

Cognitive Load (ภาระทางความคิด)
ปริมาณความพยายามทางความคิดที่ผู้ใช้ต้องใช้ในการทำความเข้าใจ UI

Perception (การรับรู้)
กระบวนการที่ผู้ใช้ตีความข้อมูลจากสิ่งที่มองเห็นบนหน้าจอ

View Hierarchy
โครงสร้างลำดับชั้นของ View ใน SwiftUI ที่สะท้อนโครงสร้างของหน้าจอ

Typographic Hierarchy (ลำดับชั้นทางตัวอักษร)
การใช้ขนาด น้ำหนัก และสีของตัวอักษรเพื่อแยกความสำคัญของข้อมูล
VStack (Vertical Stack)
โครงสร้างสำหรับจัดวางองค์ประกอบในแนวตั้ง ใช้สื่อถึงลำดับการอ่านและลำดับความสำคัญของข้อมูล

HStack (Horizontal Stack)
โครงสร้างสำหรับจัดวางองค์ประกอบในแนวนอน ใช้สื่อถึงการจัดกลุ่มข้อมูลในระดับเดียวกัน

ZStack (Z-Axis Stack)
โครงสร้างสำหรับซ้อนองค์ประกอบ ใช้สร้างบริบทระหว่างองค์ประกอบหลักและองค์ประกอบรอง เช่น พื้นหลังกับเนื้อหา

รายละเอียดสำหรับการอ้างอิงเอกสารชุดนี้
- **ผู้เขียน** ธิติ ธีระเธียร
- **วันที่เผยแพร่** วันที่ 1 กุมภาพันธ์ 2569
- **เข้าถึงได้จาก** https://ajthiti.gitbook.io/develop-in-swift/getting-started/design-an-interface
- **เงื่อนไขในการใช้งาน** This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.