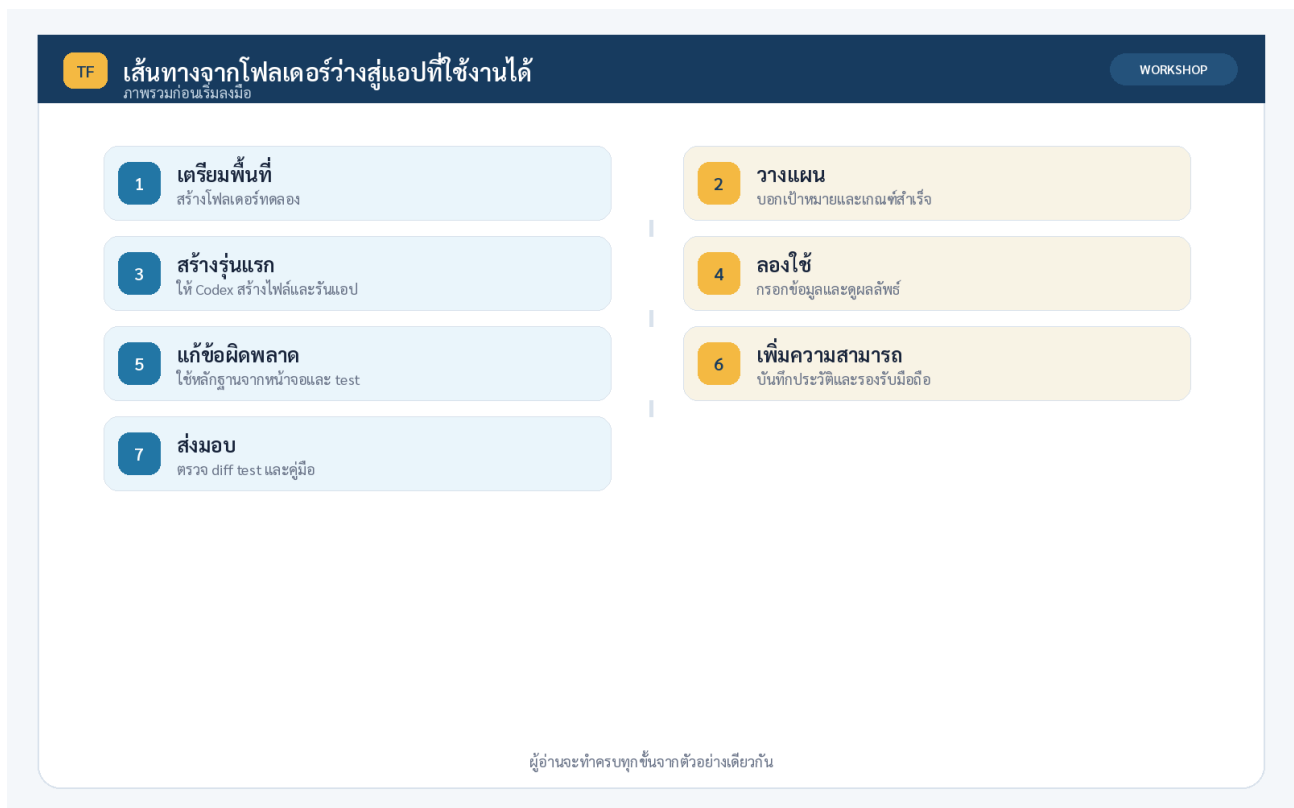


สร้างเว็บแอปแรกด้วย Codex สำหรับผู้ไม่มีพื้นฐานโปรแกรม

คู่มือปฏิบัติจากโฟลเดอร์ว่างจนได้แอปที่ทดสอบผ่าน



เส้นทางที่ผู้อ่านจะลงมือทำในหนังสือเล่มนี้

เรียบเรียงเพื่อการเรียนรู้โดย ThinkToFinish

ฉบับปรับปรุงเดือนกันยายน 2569

หนังสือเล่มนี้ต่างจากคู่มือสรุปอย่างไร

หนังสือฉบับนี้ออกแบบให้อ่านที่ไม่เคยเขียนโปรแกรมสามารถนั่งทำตามได้จริง ตัวอย่างตลอดเล่มเป็นโครงการเดียวกัน คือ Training Needs Planner เว็บแอปสำหรับรวบรวมบริบทขององค์กรและแนะนำหลักสูตรอบรมเบื้องต้น เราสร้างแอปนี้จากโฟลเตอร์ว่าง ทดสอบกติกา ทดลองกรอกข้อมูล แก้อัปเดตพลาด เพิ่มการบันทึกประวัติ และตรวจบนมือถือ

ทุกช่วงที่ต้องคุยกับ Codex มีพรอมต์ให้คัดลอก พร้อมคำอธิบายว่าทำไมต้องส่งเช่นนั้นและควรเห็นผลลัพธ์อะไร ภาพประกอบจะแสดงหน้าจอหรือสถานะที่คาดหวัง เพื่อให้ผู้อ่านรู้ว่าตนเองมาถูกทางหรือควรหยุดแก้ปัญหาตรงไหน สิ่งที่คุณจะได้เมื่อทำจบ เว็บแอปที่ทำงานบนเครื่อง มีแบบฟอร์มภาษาไทย ระบบจับคู่หลักสูตร การตรวจข้อมูล ประวัติการบันทึก การแสดงผลบนมือถือ และชุดทดสอบอัตโนมัติ 4 กรณี

วิธีใช้หนังสือ

อ่านบทที่ 1 ถึง 3 ก่อนเริ่ม เพื่อเข้าใจ Project สิทธิ์ และ Plan mode จากนั้นทำ Workshop บทที่ 4 ถึง 11 ตามลำดับ อย่าข้ามจุดตรวจผล หากหน้าจอของคุณต่างจากตัวอย่าง ให้ใช้บทที่ 12 เพื่อวินิจฉัยก่อนเดินหน้าต่อ

ชื่อปุ่มหรือหน้าต่างของ Codex อาจเปลี่ยนตามรุ่นของแอป แต่หลักการไม่เปลี่ยน ได้แก่ เปิดโฟลเดอร์ที่ถูกต้อง จำกัด สิทธิ์ บอกเป้าหมายให้ชัด ขอแผนก่อนแก้ไฟล์ ตรวจผลลัพธ์ และให้การทดสอบยืนยันพฤติกรรมสำคัญ

สิ่งที่ไม่ต้องรู้ล่วงหน้า

คุณไม่ต้องจำภาษา HTML CSS หรือ JavaScript และไม่ต้องเข้าใจทุกบรรทัดก่อนเริ่ม เป้าหมายคือเรียนรู้วิธีกำหนด โจทย์ สื่อสารกับ Codex ตรวจสิ่งที่เกิดขึ้น และค่อย ๆ สร้างความมั่นใจจากหลักฐาน ไม่ใช่จากการเดาว่าโค้ดน่าจะถูก

สารบัญ

หมวด	เนื้อหา
ส่วนที่ 1 เตรียมตัว	1 เข้าใจว่า Codex ทำงานอย่างไร 2 เตรียมเครื่องและพื้นที่ทดลอง 3 เปิด Project และกำหนดสิทธิ์
ส่วนที่ 2 สร้างแอปจริง	4 กำหนดโจทย์และวางแผน 5 สร้างไฟล์รุ่นแรก 6 รันแอปและอ่านผลลัพธ์ 7 ทดลองใช้แบบผู้ใช้จริง 8 แก้บั๊กด้วยหลักฐาน
ส่วนที่ 3 ทำให้แอปพร้อมใช้	9 เพิ่มกติกาและประวัติ 10 ตรวจสอบมือถือและการเข้าถึง 11 ตรวจสอบ Diff และส่งมอบ 12 แก้ปัญหาที่พบบ่อย
ภาคผนวก	ก พรอมต์ฉบับเต็ม ข คำสั่งที่ใช้ใน Workshop ค พจนานุกรมสำหรับมือใหม่ ง แหล่งอ้างอิง

แอปที่เราจะสร้าง

Training Needs Planner รับข้อมูล 6 อย่าง ได้แก่ ชื่อองค์กร ประเภทธุรกิจ จำนวนผู้เรียน งบประมาณ เวลาที่มี และ ปัญหาหลัก จากนั้นจับคู่กับหลักสูตรตัวอย่างในเครื่องและแสดงคะแนน เหตุผล รูปแบบ ระยะเวลา และขั้นตอนถัดไป ผู้ใช้สามารถบันทึกคำแนะนำไว้ในเว็บเบราว์เซอร์ได้โดยไม่ส่งข้อมูลไปยังฐานข้อมูลภายนอก

เข้าใจว่า Codex ทำงานอย่างไร

บทนี้อธิบายภาพรวมที่จำเป็นสำหรับมือใหม่ เพื่อให้รู้ว่า Codex ทำอะไรแทนเราได้ และส่วนใดที่เรายังต้องตัดสินใจเอง

Codex ต่างจากแชตถามตอบทั่วไป

แชตทั่วไปช่วยอธิบายหรือเสนอข้อความ แต่ Codex สามารถทำงานกับโฟลเดอร์โครงการ อ่านไฟล์ สร้างไฟล์ แก้ไขหลายจุด รันคำสั่ง และนำผลทดสอบกลับมาใช้แก้ปัญหา ความสามารถนี้ช่วยให้คนที่เขียนโปรแกรมเริ่มต้นได้เร็วขึ้น แต่ก็ทำให้การเลือกโฟลเดอร์และสิทธิ์มีความสำคัญมากขึ้น

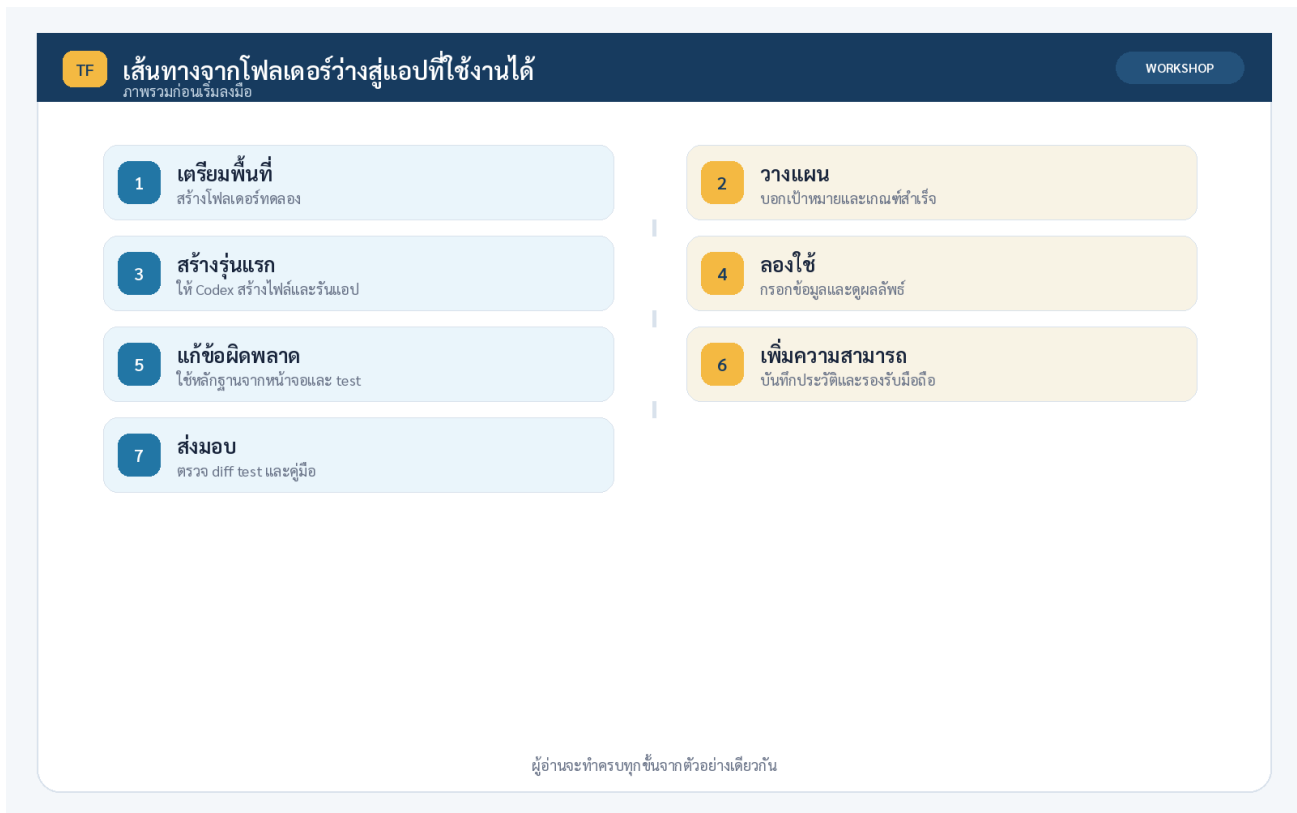
ให้นึกถึง Codex เป็นผู้ช่วยที่ทำงานเร็วและทำตามคำสั่งได้ดีเมื่อมีบริบทเพียงพอ หากคำสั่งกว้าง เช่น “สร้างแอปให้หน่อย” ผู้ช่วยต้องเดาผู้ใช้ ฟังก์ชัน เทคโนโลยี และเกณฑ์สำเร็จ การเดาเหล่านี้อาจทำให้แอปดูสวยแต่ใช้งานไม่ตรงปัญหา หนังสือเล่มนี้จึงแยกงานเป็นรอบสั้น ๆ ที่ตรวจได้

หน้าที่ของคุณในฐานะเจ้าของโจทย์

คุณเป็นผู้บอกว่าปัญหาใดควรถูกแก้ ใครเป็นผู้ใช้ ข้อมูลใดห้ามนำออกนอกเครื่อง และงานเสร็จเมื่อพฤติกรรมใดเกิดขึ้น Codex ช่วยเปลี่ยนข้อกำหนดเหล่านี้เป็นไฟล์และการทดสอบ เมื่อผลลัพธ์ไม่ตรง คุณไม่จำเป็นต้องบอกวิธีแก้โค้ด เพียงแสดงอาการที่ทำซ้ำได้และผลที่ต้องการ

วงจรที่ใช้ตลอดเล่ม

1. กำหนดผลลัพธ์และขอบเขตของงานเป็นภาษาธรรมดา
2. ขออนุญาตแก้ไขไฟล์ เพื่อเปิดเผยสมมติฐานและสิ่งที่ Codex จะทำ
3. ให้สร้างหรือแก้ครั้งละหนึ่งช่วงที่ตรวจได้
4. เปิดแอป ทดลองใช้ และบันทึกอาการที่เกิดขึ้นจริง
5. รันการทดสอบและอ่าน Diff ก่อนยอมรับงาน



ภาพที่ 1 วงจรการสร้างแอปที่ใช้ตลอดหนังสือ

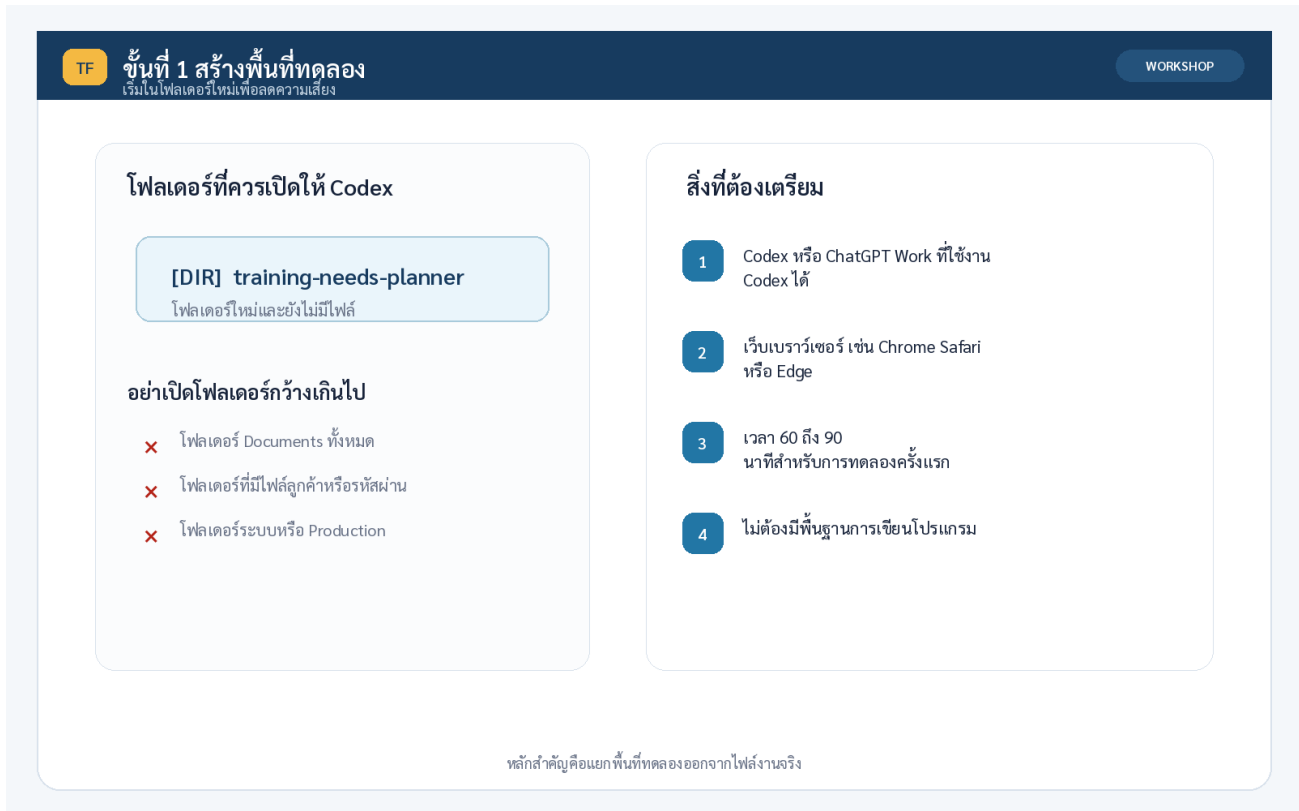
สิ่งที่หนังสือจะไม่ให้ Codex ทำโดยอัตโนมัติ

Workshop นี้ไม่ใช่ข้อมูลลูกค้าจริง ไม่ติดตั้งบริการออนไลน์ ไม่สร้างบัญชีภายนอก ไม่เผยแพร่สู่ Production และไม่ใช้ Full Access เป็นค่าเริ่มต้น การกระทำเหล่านี้มีผลเกินกว่าโพลเดอร์ทดลองและควรได้รับการตัดสินใจแยกต่างหาก

บทที่ 2

เตรียมเครื่องและพื้นที่ทดลอง

ก่อนให้ Codex สร้างไฟล์ เราจะเตรียมโพลเดอร์ที่แยกจากงานจริงและตรวจว่าเครื่องมี Node.js สำหรับรันเว็บแอปตัวอย่าง



ภาพที่ 2 พื้นที่ทดลองและสิ่งที่ต้องเตรียม

สร้างโฟลเดอร์ใหม่

บน Windows ให้เปิด File Explorer และสร้างโฟลเดอร์ชื่อ training-needs-planner ในพื้นที่ที่หาได้ง่าย เช่น Documents หรือ Desktop แต่ไม่ควรเลือก Documents ทั้งโฟลเดอร์เป็น Project บน macOS ให้ใช้ Finder สร้างโฟลเดอร์ชื่อเดียวกัน โฟลเดอร์นี้ควรว่างและไม่มีไฟล์ส่วนตัวยู่ภายใน

จุดตรวจผล เมื่อเปิดโฟลเดอร์ คุณควรเห็นพื้นที่ว่างและชื่อโฟลเดอร์ตรงกับ training-needs-planner

ตรวจ Node.js

เปิด Terminal ใน Codex หรือ Terminal ของเครื่อง แล้วพิมพ์คำสั่งด้านล่าง Node.js คือโปรแกรมที่ใช้เปิดเว็บตัวอย่างและรัน test คำสั่งนี้เพียงตรวจสอบเวอร์ชัน ไม่ได้แก้ไขไฟล์ในเครื่อง

คำสั่งตรวจสอบเครื่อง

```
node -v
npm -v
```

หากเห็นหมายเลขเวอร์ชันทั้งสองบรรทัด ให้เดินหน้าต่อไป หากเห็นข้อความว่าไม่พบคำสั่ง อย่าให้ Codex ติดตั้งทันทีโดยไม่ตรวจสอบ ให้ใช้พรอมต์นี้เพื่อขอคำแนะนำที่เหมาะสมกับระบบปฏิบัติการของคุณ

พรอมต์เมื่อตรวจไม่พบ Node.js

```
ฉันใช้ [Windows หรือ macOS]
คำสั่ง node -v ใช้งานไม่ได้
ช่วยอธิบายทางเลือกในการติดตั้ง Node.js รุ่นเสถียร
ยังไม่ต้องติดตั้งหรือเปลี่ยนการตั้งค่าเครื่องจนกว่าฉันจะอนุมัติ
```

สำรองก่อนทดลองกับงานเดิม

Workshop เริ่มจากโฟลเดอร์ว่างจึงไม่มีไฟล์เดิมให้เสียหาย หากคุณนำวิธีนี้ไปใช้กับโครงการจริง ควรมี Git หรือสำเนาสำรองก่อนให้ Codex แก่ไฟล์ คำว่า Git หมายถึงระบบบันทึกประวัติการเปลี่ยนแปลง ซึ่งช่วยย้อนกลับเมื่อการแก้ไขไม่เป็นไปตามที่ต้องการ

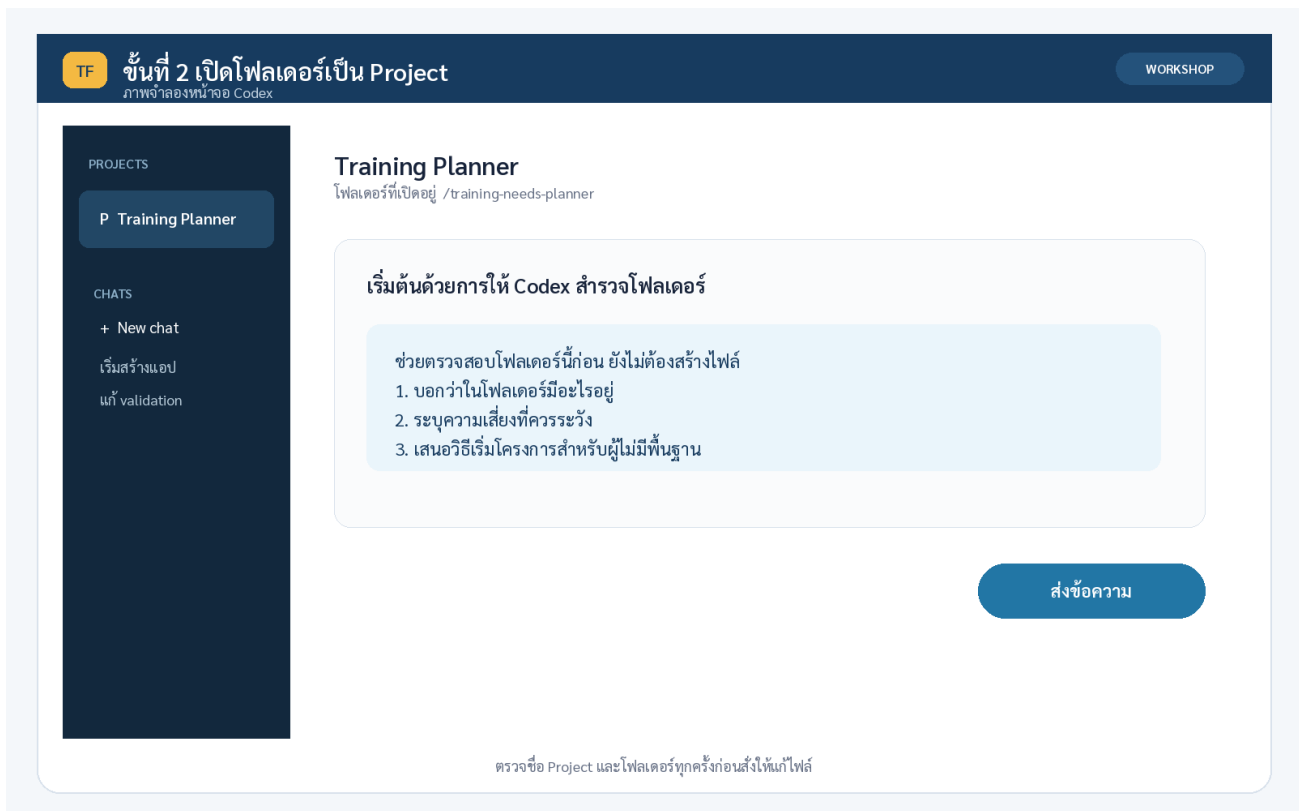
บทที่ 3

เปิด Project และกำหนดสิทธิ์

เราจะเปิดเฉพาะโฟลเดอร์ทดลองใน Codex ตรวจสอบชื่อ Project และเลือกระดับสิทธิ์ที่เพียงพอสำหรับการสร้างแอป

เปิดโฟลเดอร์ที่ต้องการ

ใน Codex ให้เลือกเปิด Project หรือ Open Folder แล้วเลือกโฟลเดอร์ training-needs-planner ที่สร้างไว้ อย่าเลือกโฟลเดอร์แม่ที่มีเอกสารอื่นรวมอยู่ หลังเปิดแล้ว ให้ดูชื่อ Project และเส้นทางที่แสดงบนหน้าจออีกครั้ง



ภาพที่ 3 ตัวอย่าง Project และพร้อมตรวจสอบโฟลเดอร์

พร้อมตรวจสอบครั้งแรก

ช่วยตรวจสอบโฟลเดอร์นี้ก่อน ยังไม่ต้องสร้างไฟล์

1. บอกว่าในโฟลเดอร์มีอะไรอยู่
2. ระบุความเสี่ยงที่ควรระวัง
3. เสนอวิธีเริ่มโครงการสำหรับผู้ไม่มีพื้นฐาน

คำตอบที่ถูกต้องควรบอกว่าโฟลเดอร์ว่างหรือมีไฟล์โดยอยู่ และยังไม่ควรสร้างไฟล์ หาก Codex เริ่มลงมือทันที ให้กดหยุดแล้วเพิ่มคำว่า “ยังไม่ต้องแก้หรือสร้างไฟล์” ไว้ต้นพารามต์

เลือกสิทธิ์สำหรับ Workshop

Codex ต้องอ่านไฟล์เพื่อเข้าใจโครงการ และต้องเขียนไฟล์เพื่อสร้างแอป ระดับ Workspace write จึงเพียงพอสำหรับ Workshop นี้ สิทธิ์ดังกล่าวควรจำกัดอยู่ในโฟลเดอร์ที่เปิด ไม่ควรอนุญาตให้เข้าถึงไฟล์ส่วนตัวหรือระบบอื่นโดยไม่จำเป็น



ภาพที่ 4 เริ่มจากสิทธิ์ต่ำแล้วเพิ่มเมื่อจำเป็น

งาน	สิทธิ์เริ่มต้น	เหตุผล
สำรวจหรืออธิบายไฟล์	อ่านอย่างเดียว	ยังไม่ต้องเปลี่ยนไฟล์
สร้างแอปในโฟลเดอร์ทดลอง	เขียนใน Workspace	สร้างไฟล์และรันทดสอบในขอบเขตโครงการ
ติดตั้งจากอินเทอร์เน็ต	ขออนุมัติเป็นครั้ง	ตรวจแหล่งที่มาและความจำเป็นก่อน
เผยแพร่หรือเชื่อมข้อมูลจริง	คนอนุมัติทุกครั้ง	กระทบผู้ใช้หรือระบบภายนอก
Full Access	ไม่ใช่เป็นค่าเริ่มต้น	ลดขอบเขตป้องกันของ sandbox

บทที่ 4

กำหนดโจทย์และวางแผน

บทนี้เปลี่ยนความคิดเรื่องแอปให้เป็นข้อกำหนดที่ Codex เข้าใจและตรวจสอบได้ ก่อนสร้างไฟล์จริง

เริ่มจากผู้ใช้และปัญหา

ผู้ใช้อย่างคือเจ้าหน้าที่ HR หรือผู้ประสานงานอบรมที่ต้องรวบรวมข้อมูลจากหน่วยงานแล้วเสนอรูปแบบการอบรมเบื้องต้น ปัญหาที่ต้องแก้ไขไม่ใช่การทำนายหลักสูตรที่สมบูรณ์ แต่คือการจัดข้อมูลให้ครบและเสนอจุดเริ่มต้นอย่างมีเหตุผล

รุ่นแรกจะใช้ข้อมูลตัวอย่าง 8 หลักสูตร เก็บไว้ในไฟล์ภายในเครื่อง และใช้กติกาง่าย ๆ ให้คะแนนตามปัญหา ประเภทธุรกิจ งบประมาณ เวลา และจำนวนผู้เรียน วิธีนี้ทำให้เราเห็นตรรกะทั้งหมดและทดสอบได้ โดยไม่ต้องเชื่อม AI API หรือฐานข้อมูล



ภาพที่ 5 โครงสร้างพอร์มสำหรับสร้างแอปจริง

พรอมต์วางแผนฉบับเต็ม

คัดลอกทั้งหมดไปที่ Plan mode

เราจะสร้างเว็บแอปชื่อ Training Needs Planner สำหรับเจ้าหน้าที่ HR ที่ไม่มีพื้นฐานโปรแกรม

เป้าหมาย

ให้ผู้ใช้กรอกชื่อองค์กร ประเภทธุรกิจ จำนวนผู้เรียน งบประมาณ เวลาที่มี และปัญหาหลัก แล้วแอปแนะนำหลักสูตรจากข้อมูลตัวอย่างในเครื่อง

ข้อจำกัด

- ใช้ HTML CSS และ JavaScript
- ไม่ใช้ฐานข้อมูล API หรือบริการภายนอก
- ข้อมูลทั้งหมดต้องอยู่ในเครื่อง
- รองรับมือถือและใช้ภาษาไทย

เสร็จเมื่อ

- กรอกไม่ครบแล้วมีข้อความบอกจุดที่ต้องแก้
- ผลลัพธ์มีชื่อหลักสูตร คะแนน เหตุผล รูปแบบ เวลา และขั้นตอนถัดไป
- บันทึกประวัติไว้ใน browser ได้
- มี test สำหรับ validation และกติกาแนะนำหลักสูตร
- test ผ่านทั้งหมด

ก่อนสร้างไฟล์ โปรดสำรวจโฟลเดอร์ สรุปความเข้าใจ เสนอแผน ไฟล์ที่จะสร้าง สมมติฐาน ความเสี่ยง และวิธีทดสอบ ยังไม่ต้องแก้ไฟล์จนกว่าฉันจะอนุมัติ

TF

ขั้นที่ 4 ใช้ Plan mode ก่อนสร้างไฟล์

ให้ Codex ทบทวนโจทย์และบอกสิ่งที่จะทำ

WORKSHOP

MODE

PLAN Plan mode

คุณ

วางแผนสร้างเว็บแอป Training Needs Planner สำหรับผู้ไม่มีพื้นฐาน โดยใช้ HTML CSS และ JavaScript ไม่ใช้ฐานข้อมูล ข้อมูลอยู่ในเครื่อง และต้องมี test

Codex เสนอแผน

- 1 สร้างแบบฟอร์ม 6 ช่องและ validation ภาษาไทย
- 2 สร้างกติกาจับคู่กับหลักสูตรตัวอย่าง 8 รายการ
- 3 แสดงคะแนน เหตุผล และขั้นตอนถัดไป
- 4 บันทึกประวัติด้วย localStorage
- 5 เพิ่ม test สำหรับข้อมูลไม่ครบและการจับคู่หลักสูตร
- 6 รัน test และเปิด preview ให้ตรวจบนเดสก์ท็อปและมือถือ

อ่านแผนและแก้สมมติฐานก่อนกอนอนุมัติ

ภาพที่ 6 ตัวอย่างคำตอบใน Plan mode

อ่านแผนอย่างไรเมื่อไม่รู้โค้ด

คุณไม่จำเป็นต้องประเมินโครงสร้างทางเทคนิคทุกจุด ให้เริ่มจากคำถามที่ตรวจสอบด้วยสามัญสำนึกได้ แผนบอกผู้ใช้ถูกคนหรือไม่ ข้อมูลที่กรอกครบหรือไม่ มีการส่งข้อมูลออกนอกเครื่องหรือไม่ และมีวิธีพิสูจน์ฟังก์ชันสำคัญหรือไม่ หากแผนเพิ่ม login ฐานข้อมูล หรือบริการออนไลน์ ทั้งที่ไม่ได้ขอ ให้ตัดออกก่อนอนุมัติ

จุดที่ควรแก้ไขในแผนก่อนสร้าง

1. หาก Codex เลือก Framework หรือบริการภายนอก ให้ถามว่าเหตุใดจึงจำเป็นและมีทางเลือกที่ง่ายกว่าหรือไม่
2. หากไม่มี test ให้เพิ่มกรณีข้อมูลไม่ครบ โจทย์ AI และจำนวนผู้เรียนมากกว่า 30 คน
3. หากไม่ได้กล่าวถึงมือถือ ให้เพิ่มการตรวจที่ความกว้างประมาณ 390 พิกเซล
4. หากแผนจะติดตั้ง package ให้ขอรายการ package และเหตุผลก่อนอนุมัติ

บทที่ 5

สร้างไฟล์รุ่นแรก

เมื่อแผนถูกต้อง เราจะอนุมัติให้ Codex สร้างไฟล์ตามขอบเขต แล้วตรวจว่าไฟล์แต่ละชุดมีหน้าที่ตรงกับแผน

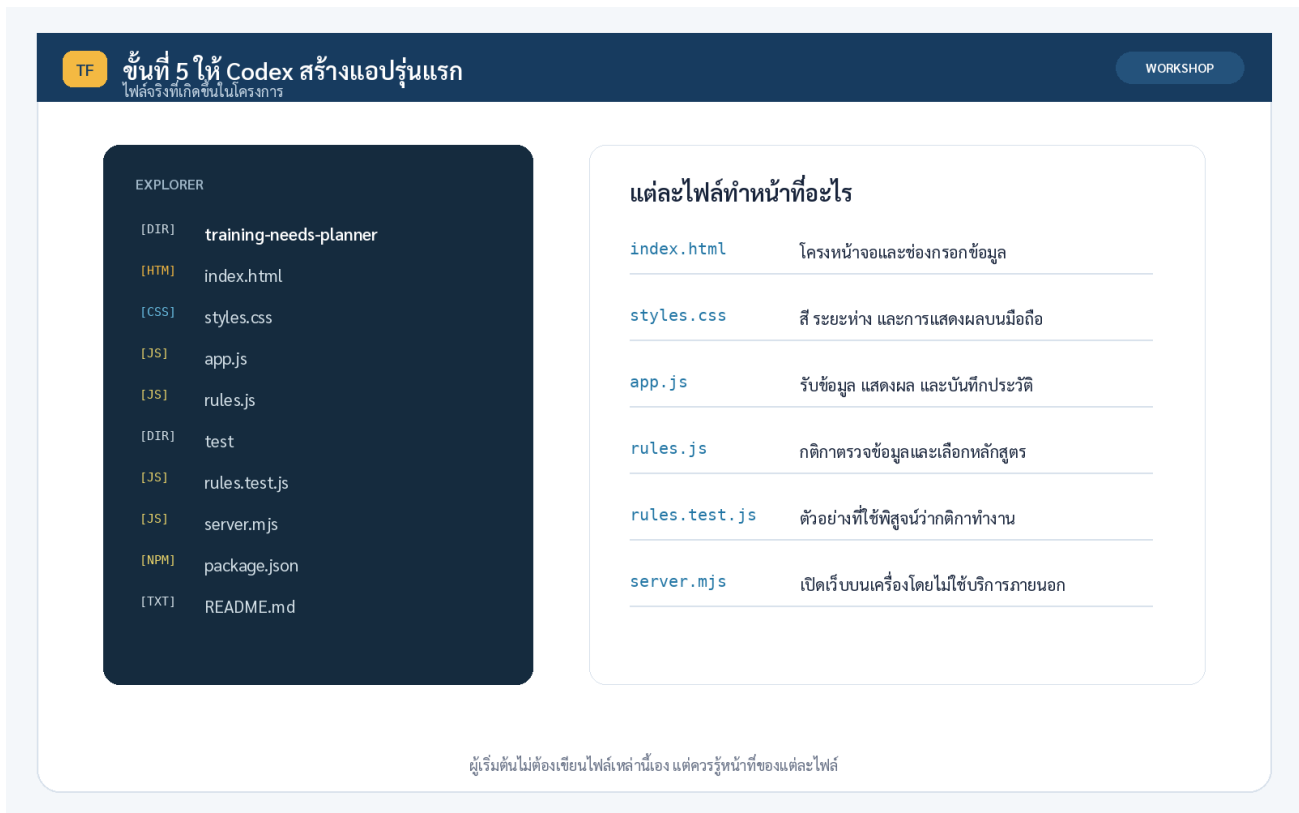
พร้อมต่ออนุมัติให้ลงมือ

อนุมัติแผน ดำเนินการสร้างรุ่นแรกได้

- สร้างเฉพาะไฟล์ที่ระบุในแผน
- ใช้ข้อมูลหลักสูตรตัวอย่างในเครื่อง
- เพิ่ม test ตามเกณฑ์สำเร็จ
- รัน test หลังสร้างเสร็จ
- สรุปไฟล์ที่สร้าง คำสั่งที่รัน ผลทดสอบ และข้อจำกัด

หากต้องติดตั้งสิ่งใดหรือใช้อินเทอร์เน็ต ให้หยุดขออนุมัติก่อน

ระหว่างทำงาน Codex อาจแสดงรายชื่อไฟล์ คำสั่งที่กำลังรัน และความคืบหน้า อย่ารีบส่งคำสั่งใหม่ซ้อนกัน ให้รอจนได้สรุปหรือมีคำถามที่ต้องตอบ หากมีคำขออนุมัติให้อ่านเป้าหมายของคำสั่งและขอบเขตที่จะกระทบก่อนยอมรับ



ภาพที่ 7 ไฟล์ของแอปแรกและหน้าที่ของแต่ละไฟล์

ความหมายของไฟล์โดยไม่ต้องอ่านโค้ดทั้งหมด

ไฟล์ `index.html` กำหนดสิ่งที่ปรากฏบนหน้า เช่น หัวข้อ ช่องกรอก และปุ่ม `styles.css` ควบคุมสี ขนาด และการจัดวาง `app.js` เชื่อมการกระทำของผู้ใช้กับผลลัพธ์ ส่วน `rules.js` เก็บข้อมูลหลักสูตรและกติกาคำนวณคะแนน การแยกกติกาออกจากหน้าจอช่วยให้ทดสอบได้ง่ายขึ้น

โพลเดอร์ `test` มีตัวอย่างที่ระบบต้องทำได้ `rules.test.js` จึงทำหน้าที่คล้ายแบบตรวจงานอัตโนมัติ `server.mjs` เปิดเว็บบนเครื่อง และ `package.json` รวบรวมชื่อคำสั่งที่ใช้บ่อย เมื่อ Codex บอกให้พิมพ์ `npm test` หรือ `npm start` โปรแกรม `npm` จะอ่านคำสั่งจริงจากไฟล์นี้

ตรวจว่า Codex ไม่ออกนอกขอบเขต

รายชื่อไฟล์ข้างต้นไม่ควรมีไฟล์ `.env` กุญแจ API หรือการตั้งค่าบริการออนไลน์ หากเห็น package จำนวนมากหรือโพลเดอร์ที่ไม่รู้จัก ให้ถาม Codex ว่าแต่ละรายการจำเป็นอย่างไร และขอให้ลดโครงสร้างให้เหลือเท่าที่รุ่นแรกต้องใช้

บทที่ 6

รันแอปและอ่านผลลัพธ์

เราจะให้ Codex รัน `test` เปิดเว็บในเครื่อง และแปลผลลัพธ์ใน Terminal เป็นภาษาธรรมดา

รันการทดสอบ

ขอให้ Codex รันคำสั่ง `npm test` แล้วรายงานจำนวน test ที่ผ่านและไม่ผ่าน หากมี test ไม่ผ่าน ให้หยุดที่จุดนี้ก่อน อย่าเปิดแอปแล้วตัดสินจากหน้าต่างเพียงอย่างเดียว

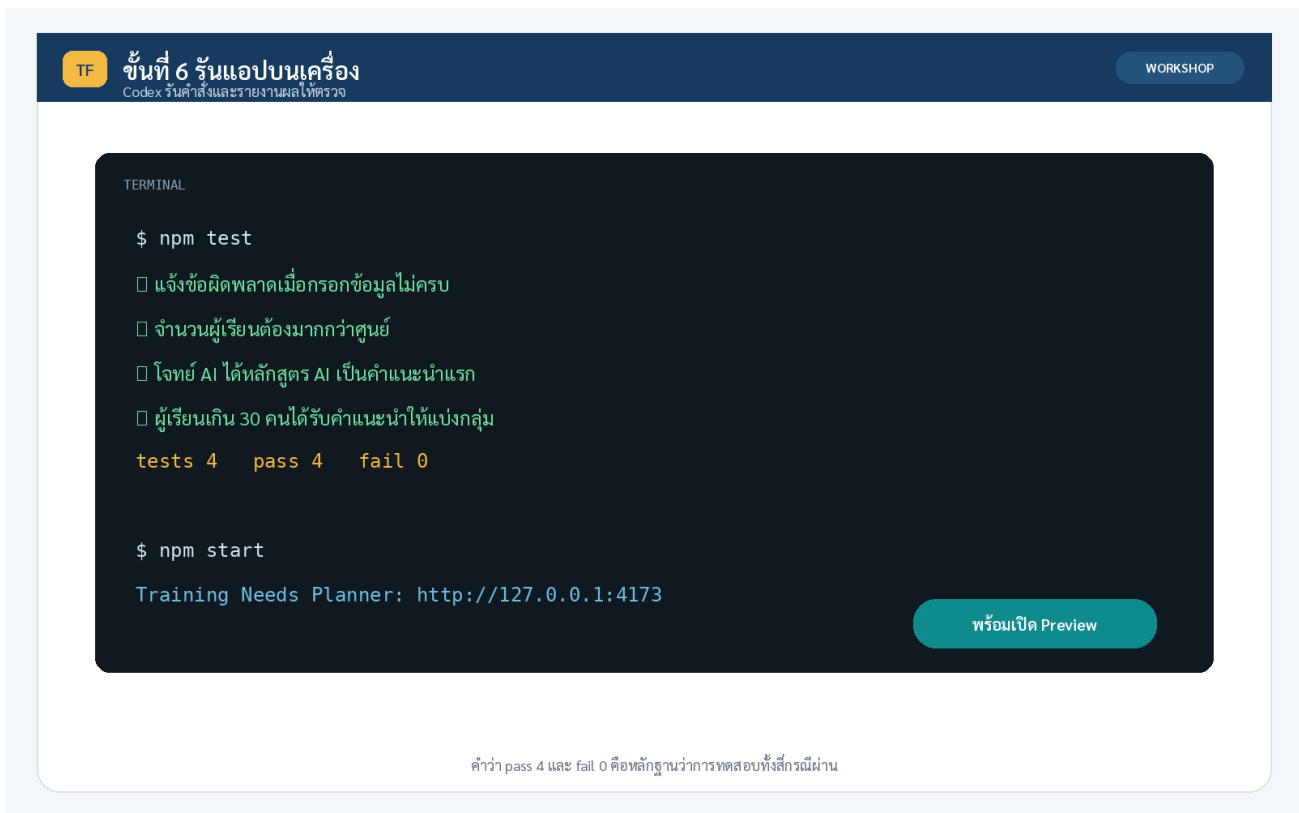
พร้อมทดสอบ

รัน `npm test`

หากมี test ไม่ผ่าน ให้สรุปชื่อ test อากา สาเหตุที่เป็นไปได้ และแผนแก้ก่อนแก้ไฟล์

เปิดเว็บในเครื่อง

เมื่อ test ผ่าน ให้ขอรัน `npm start` ระบบควรแสดง URL ที่ขึ้นต้นด้วย `http://127.0.0.1` หรือ `localhost` เปิด URL นี้ใน Preview หรือเว็บเบราว์เซอร์ URL ดังกล่าวชี้กลับมาที่เครื่องของคุณ ไม่ใช่เว็บไซต์สาธารณะ



ภาพที่ 8 ผลทดสอบและ URL สำหรับเปิดแอป

อ่านบรรทัดสำคัญใน Terminal

ข้อความ `pass 4` และ `fail 0` หมายถึงตัวอย่างอัตโนมัติทั้งสี่กรณีได้ผลตามที่กำหนด URL ในบรรทัดท้ายคือที่อยู่สำหรับเปิดแอปในเครื่อง หาก Terminal ยังคงทำงานอยู่หลังแสดง URL นั้นเป็นพฤติกรรมปกติ เพราะเซิร์ฟเวอร์ต้องเปิดค้างไว้เพื่อให้หน้าเว็บใช้งานได้

หาก URL เปิดไม่ได้

ให้คัดลอกข้อความผิดพลาดจาก Terminal ทั้งก้อน อย่าสรุปเองว่า “เปิดไม่ได้” แล้วส่งให้ Codex ระบุคำสั่งที่รันระบบปฏิบัติการ และสิ่งที่เห็นในเว็บเบราว์เซอร์ Codex จะวินิจฉัยได้แม่นยำได้แม่นยำกว่าหากมีข้อความจริงและขั้นตอนที่ทำซ้ำได้

พรีวิวหน้าจอต้อนรับเมื่อเปิดเว็บไม่ได้

ฉันรัน npm start แล้วเปิด URL ไม่ได้

คำสั่งที่รัน: npm start

ข้อความใน Terminal:

[วางข้อความทั้งหมด]

สิ่งที่เกิดใน browser:

[เช่น ขึ้น This site cannot be reached]

ช่วยวินิจฉัยที่สาเหตุ ยังไม่ต้องเปลี่ยนไฟล์จนกว่าจะอธิบายสาเหตุที่น่าเป็นไปได้

บทที่ 7

ทดลองใช้แบบผู้ใช้จริง

หน้าที่สวยไม่ได้ยืนยันว่าแอปทำงาน เราจะทดลองทั้งกรณีปกติและกรณีที่ผู้ใช้ทำข้อมูลไม่ครบ

The screenshot displays the 'Training Needs Planner' app interface. At the top, there's a dark blue header with the app's logo (TF) and name. Below this, a secondary header contains the app's purpose: 'วางแผนอบรมให้ตรงกับปัญหาธุรกิจ'. The main content area is titled 'เปลี่ยนบริบทขององค์กรเป็นแผนอบรมเบื้องต้น' (Change the context of the organization to a preliminary training plan). It includes a sub-header 'กรอกข้อมูล 6 ช่อง ระบุจะจับคู่กับหลักสูตรตัวอย่างและอธิบายเหตุผล' (Fill in 6 fields to specify which example course to pair with and explain the reason). The form is divided into two columns. The left column, titled 'ขั้นที่ 1 เล่าใจขององค์กร' (Step 1: Tell the story of the organization), contains six input fields: 'ชื่อองค์กร' (Organization name), 'ประเภทธุรกิจ' (Business type), 'จำนวนผู้เรียน' (Number of learners), 'งบประมาณ' (Budget), 'เวลาที่มี' (Time available), and 'ปัญหาหลัก' (Main problem). Each field has a placeholder text 'เลือกหรือพิมพ์ข้อมูล' (Select or type information). A large blue button labeled 'สร้างคำแนะนำ' (Generate recommendation) is at the bottom of this column. The right column is currently empty, showing a placeholder with a blue circle containing three dots and the text 'คำแนะนำจะปรากฏตรงนี้' (Recommendation will appear here) and 'กรอกข้อมูลด้านซ้ายแล้วกดสร้างคำแนะนำ' (Fill in the information on the left and click 'Generate recommendation'). At the bottom right of the main content area, there's a small link 'หน้าจอเริ่มต้น' (Start screen).

ภาพที่ ๑ หน้าจอเริ่มต้นของ Training Needs Planner

เริ่มจากดูความเข้าใจของหน้าจอ

ก่อนกรอกข้อมูล ลองมองหน้าเว็บโดยไม่ย้อนอ่านพรีวิว ผู้ใช้ควรรู้ว่าต้องเริ่มตรงไหน ช่องใดบังคับ และจะเกิดอะไรเมื่อกดปุ่ม หน้าจอตัวอย่างแบ่งเป็นสองส่วนชัดเจน ด้านซ้ายรับข้อมูล ด้านขวาแสดงคำแนะนำ จึงไม่ต้องอาศัยคู่มือยาวเพื่อเริ่มใช้งาน

ทดลองกรอกไม่ครบ

กดปุ่มสร้างคำแนะนำโดยยังไม่กรอกข้อมูล แอปควรไม่แสดงผลลัพธ์ปลอม แต่ต้องบอกจุดที่ขาดในแต่ละช่อง และย้าย focus ไปยังช่องแรกที่ต้องแก้ไข

จุดตรวจผล ทุกช่องบังคับมีข้อความภาษาไทยและกรอบสีแดง ไม่มีผลลัพธ์ปรากฏทางขวา

ภาพที่ 10 การแจ้งข้อผิดพลาดเมื่อข้อมูลไม่ครบ

ทดลองกรณีปกติ

กรอกชื่อองค์กร เลือกธุรกิจบริการ ระบุผู้เรียน 25 คน งบประมาณ 30,000 ถึง 80,000 บาท เวลา 1 วัน และปัญหาว่าอยากเริ่มใช้ AI จากนั้นกดสร้างคำแนะนำ

จุดตรวจผล ด้านขวาแสดงหลักสูตร AI สำหรับงานประจำวัน คะแนน เหตุผล รูปแบบ ระยะเวลา จำนวนผู้เรียน และสิ่งที่ควรทำต่อ

TF

Training Needs Planner
 แอปพลิเคชันสำหรับวางแผนอบรม

APP PREVIEW

TF

Training Needs Planner
 วางแผนอบรมให้ตรงกับปัญหาธุรกิจ

เปลี่ยนบริบทขององค์กรเป็นแผนอบรมเบื้องต้น

กรอกข้อมูล 6 ช่อง ระบบจะจับคู่กับหลักสูตรตัวอย่างและอธิบายเหตุผล

ขั้นที่ 1 เล่าโจทย์ขององค์กร

ชื่อองค์กร

บริษัท ก้าวไกลเซอร์วิส จำกัด

ประเภทธุรกิจ

บริการ

จำนวนผู้เรียน

25

งบประมาณ

30,000 ถึง 80,000 บาท

เวลาที่มี

1 วัน

ปัญหาหลัก

อยากเริ่มใช้ AI แต่ยังไม่รู้จะเริ่มตรงไหน

สร้างคำแนะนำ

ขั้นที่ 2

เหมาะสมมาก

95%

AI สำหรับงานประจำวัน

ช่วยทีมเลือกงานที่เหมาะสมกับ AI เขียนพร้อมดัดแปลงคำตอบ และออกแบบวิธีใช้ที่ปลอดภัย

เหตุผลที่แนะนำ

- ตรงกับปัญหาหลักที่เลือก
- อยู่ในงบประมาณและเวลาที่กำหนด
- จำนวนผู้เรียนเหมาะกับ Workshop

บันทึกคำแนะนำนี้

ผลลัพธ์หลังกรอกข้อมูลครบ

ภาพที่ 11 ผลคำแนะนำจากข้อมูลตัวอย่าง

อย่าใช้คะแนนเกินความหมาย

คะแนนในแอปนี้มาจากกติกาที่เราเขียนเอง ไม่ใช่การพยากรณ์ความสำเร็จของการอบรม 95 เปอร์เซ็นต์หมายถึงข้อมูลที่ตรงกับเงื่อนไขของหลักสูตรหลายข้อ จึงควรใช้เป็นจุดเริ่มต้นในการสนทนา ไม่ใช่การตัดสินใจแทนผู้เชี่ยวชาญหรือผู้บริหาร

บทที่ 8

แก้บั๊กด้วยหลักฐาน

มือใหม่ไม่จำเป็นต้องหาบรรทัดที่ผิดเอง แต่ต้องบอกอาการ ขั้นตอนทำซ้ำ ผลที่คาดหวัง และหลักฐานให้ครบ

รูปแบบรายงานบั๊กที่ Codex ใช้แก้ได้

คำว่า bug หมายถึงพฤติกรรมที่ไม่เป็นไปตามที่กำหนด รายงานที่ดีไม่ใช่คำกว้างว่า “มันเสีย” แต่บอกว่าทำอะไร เห็นอะไร และควรเห็นอะไร ตัวอย่างต่อไปใช้กรณีป้อนสร้างคำแนะนำทำงานทั้งที่ยังไม่ได้เลือกงบประมาณ

16

พร้อม์รายงานบั๊ก

พบข้อผิดพลาดในแบบฟอร์ม

ขั้นตอนทำซ้ำ

1. กรอกทุกช่อง ยกเว้นงบประมาณ
2. กดสร้างคำแนะนำ

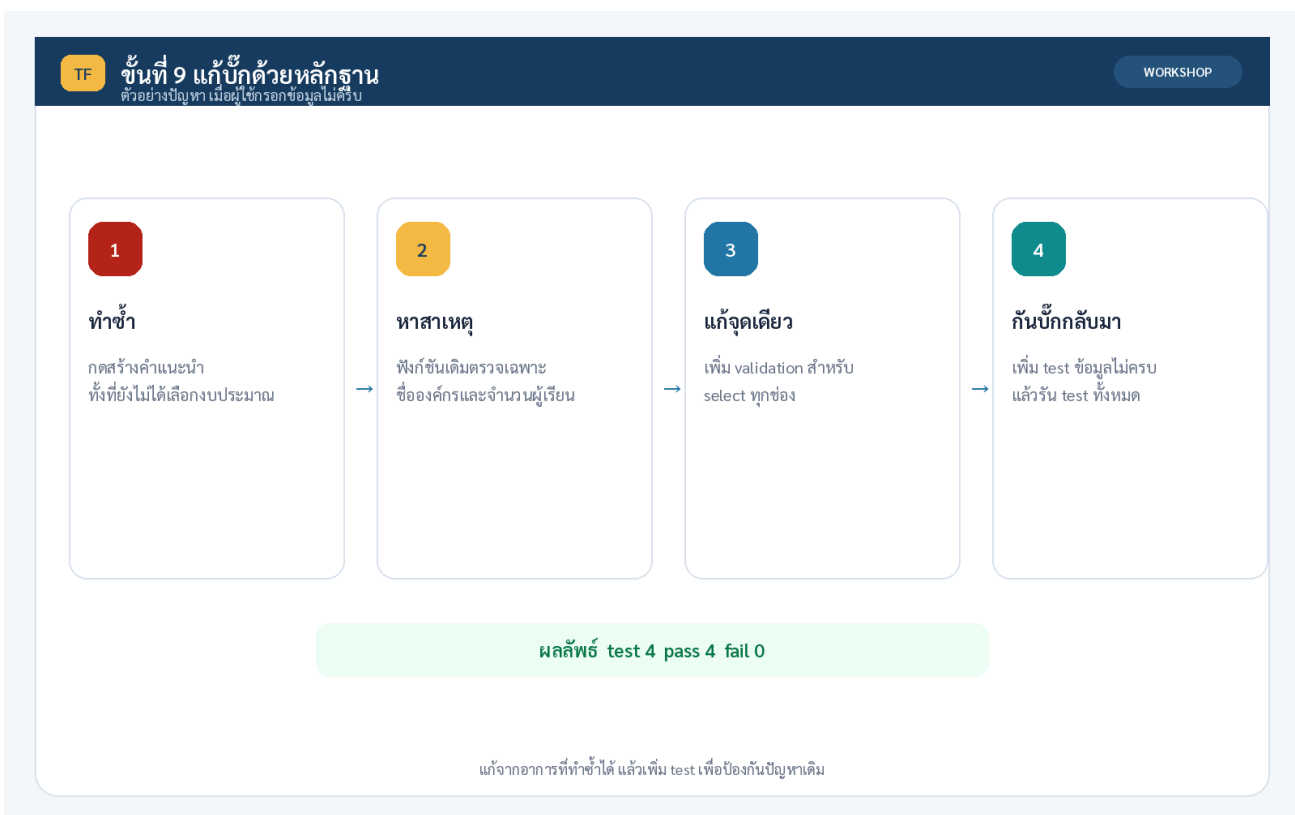
ผลที่เกิดขึ้น

แอปยังแสดงผลคำแนะนำ

ผลที่คาดหวัง

ไม่ควรสร้างคำแนะนำ และต้องแสดงข้อความใต้ช่องงบประมาณ

โปรดหาสาเหตุ แก่เฉพาะจุด และเพิ่ม regression test สำหรับกรณีนี้
รัน test ทั้งหมดและสรุป Diff หลังแก้



ภาพที่ 12 กระบวนการแก้บั๊กจากอาการถึง regression test

เหตุใดต้องเพิ่ม regression test

หลังแก้บั๊ก แอปอาจใช้งานได้ในวันนี้ แต่การแก้ฟิเจอร์ในอนาคตอาจทำให้ปัญหาเดิมกลับมา regression test คือการบันทึกตัวอย่างที่เคยผิดไว้เป็นข้อสอบอัตโนมัติ หากปัญหาเดิมเกิดซ้ำ test จะเตือนทันที

คำถามหลัง Codex บอกว่าแก้แล้ว

1. สาเหตุรากคืออะไร และแก้ในไฟล์ใด
2. มีการเปลี่ยนส่วนอื่นที่ไม่เกี่ยวข้องหรือไม่

3. เพิ่ม test ชื่ออะไร และ test จำลองขั้นตอนใด
4. ผล npm test หลังแก้เป็นอย่างไร
5. มีข้อจำกัดหรือกรณีขอบที่ยังไม่ได้ทดสอบอะไรบ้าง

บทที่ 9

เพิ่มกติกาและประวัติ

เมื่อรุ่นแรกทำงาน เราจะเพิ่มความสามารถทีละเรื่องเพื่อไม่ให้ขอบเขตบานและยังตรวจได้ว่าอะไรทำให้พฤติกรรมเปลี่ยน

เข้าใจกติกาแนะนำหลักสูตร

rules.js เริ่มทุกหลักสูตรด้วยคะแนนพื้นฐาน แล้วเพิ่มคะแนนเมื่อปัญหา ประเภทธุรกิจ งบประมาณ เวลา และจำนวนผู้เรียนตรงกับเงื่อนไข จากนั้นเรียงหลักสูตรจากคะแนนมากไปน้อยและเลือกอันดับแรก วิธีนี้ตั้งใจให้เรียบง่าย โปร่งใส และทดสอบได้

ตัวอย่างตรรกะจากแอปจริง

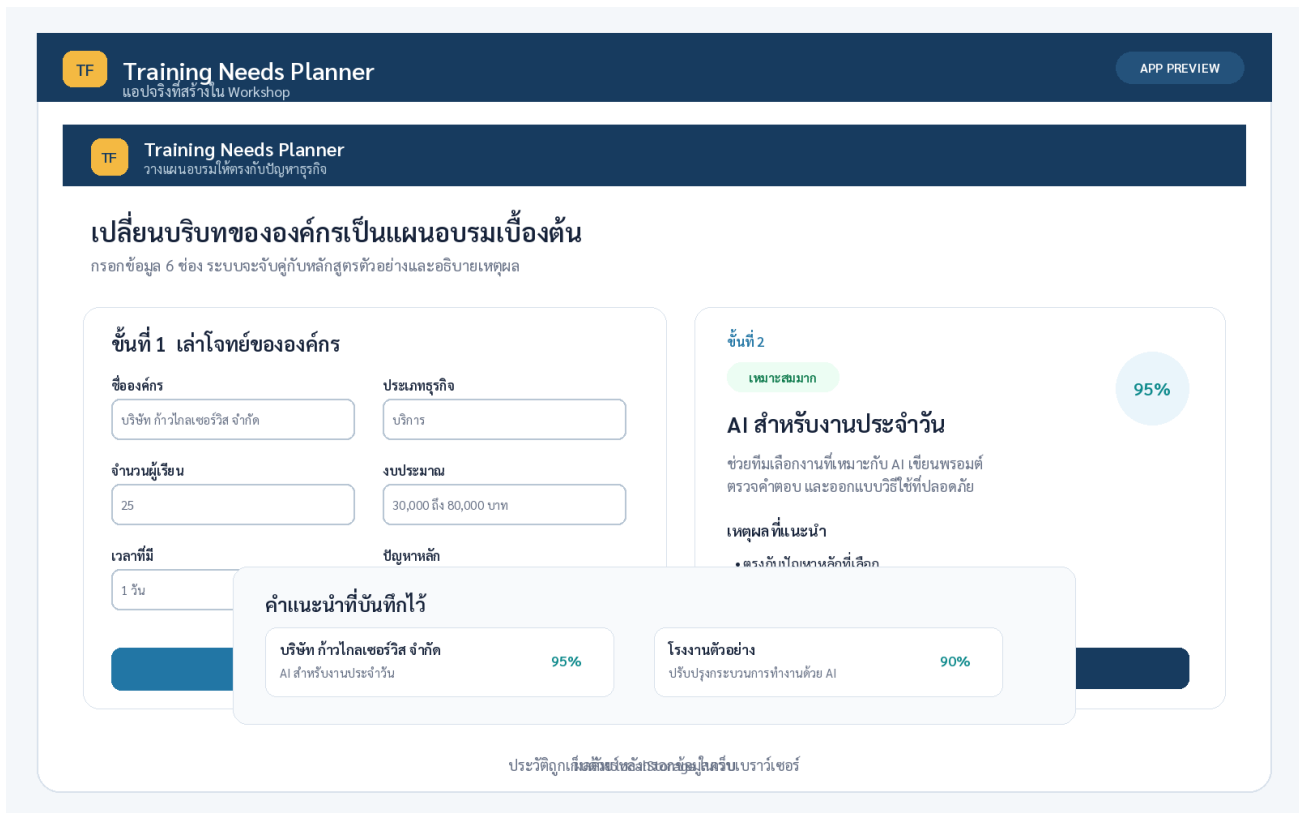
```
export function scoreCourse(course, input) {
  let score = 25;
  if (course.challenges.includes(input.challenge)) score += 35;
  if (course.industries.includes(input.industry)) score += 15;
  if (course.budgets.includes(input.budget)) score += 10;
  if (course.durations.includes(input.duration)) score += 10;
  if (Number(input.learners) <= 30) score += 5;
  return Math.min(score, 95);
}
```

แม้คุณจะไม่เขียนโค้ด แต่สามารถตรวจตรรกะได้จากตัวเลข ตัวอย่างเช่น ปัญหาหลักมีน้ำหนัก 35 คะแนน มากกว่างบประมาณ 10 คะแนน ซึ่งสะท้อนว่าความตรงกับปัญหาสำคัญกว่าการพอดีกับงบประมาณ หาก你不เห็นด้วย ให้ปรับกติกาในระดับธุรกิจก่อนขอให้ Codex เปลี่ยนโค้ด

เพิ่มการบันทึกประวัติ

พร้อมต์เพิ่มฟีเจอร์ทีละเรื่อง

เพิ่มความสามารถบันทึกคำแนะนำล่าสุดไว้ใน localStorage
แสดงไม่เกิน 6 รายการ โดยมีชื่อองค์กร หลักสูตร คะแนน และวันที่บันทึก
เพิ่มปุ่มล้างประวัติ และต้องรับมีอกรณข้อมูลเดิมอ่านไม่ได้
ห้ามเพิ่มฐานข้อมูลหรือส่งข้อมูลออกนอกเครื่อง
เพิ่มหรือปรับ test ที่เกี่ยวข้อง แล้วรัน test ทั้งหมด



ภาพที่ 13 ประวัติคำแนะนำที่บันทึกในเครื่อง

ข้อจำกัดของ localStorage

ประวัติจะอยู่ในเว็บเบราว์เซอร์และอุปกรณ์ที่ใช้งาน หากล้างข้อมูลเว็บไซต์ เปลี่ยนเบราว์เซอร์ หรือย้ายเครื่อง ประวัติอาจหาย ข้อมูลไม่ได้กลายเป็นระบบส่วนกลางและไม่เหมาะกับข้อมูลอ่อนไหว รุ่นนี้จึงเป็นต้นแบบสำหรับทดลองกระบวนการ ไม่ใช่ระบบ HR จริง

บทที่ 10

ตรวจมือถือและการเข้าถึง

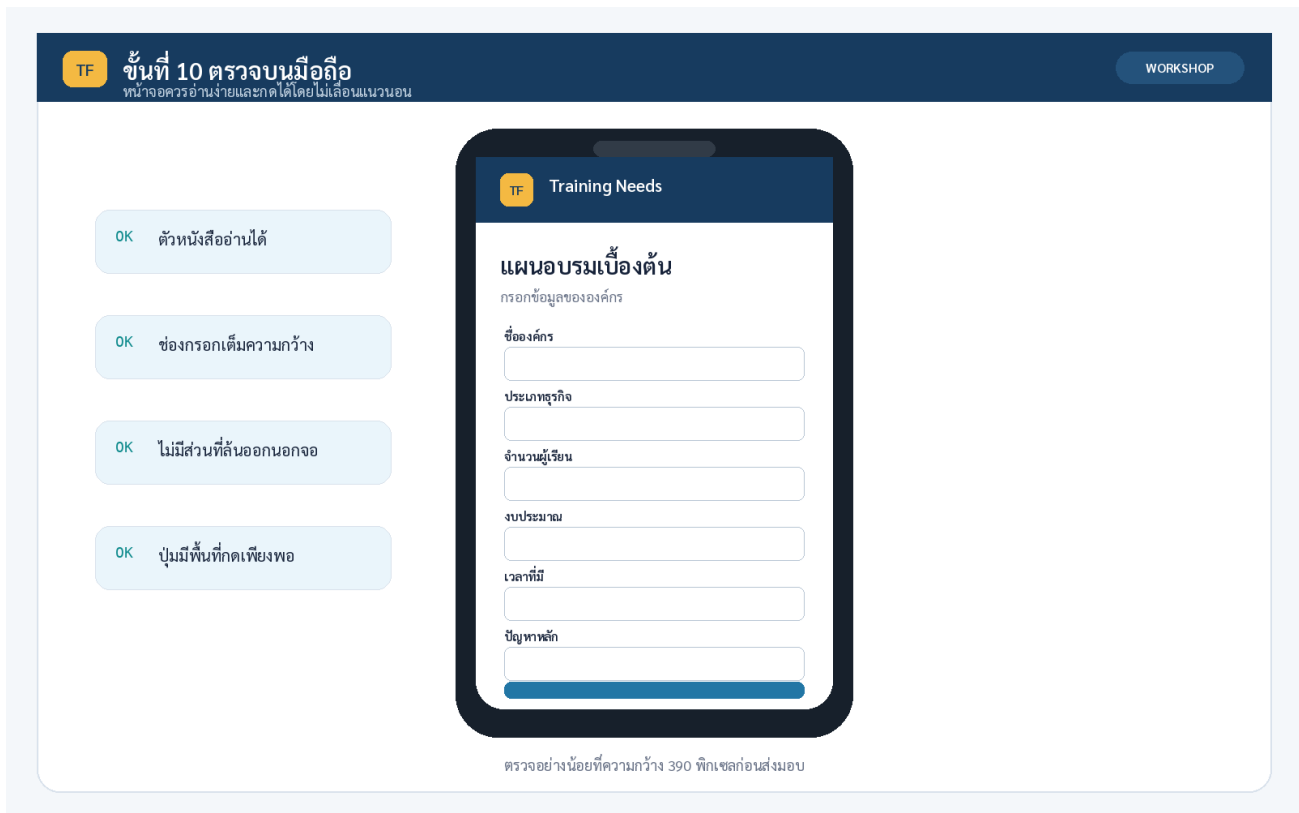
แอปพร้อมใช้เมื่อคนอ่าน กรอก และกดปุ่มได้บนหน้าจอจริง ไม่ใช่เพียงเมื่อโค้ดทำงาน

ตรวจมุมมองมือถือ

ขอให้ Codex เปิด Preview ที่ความกว้างประมาณ 390 พิกเซล จากนั้นตรวจว่าช่องกรอกเรียงเป็นคอลัมน์เดียว ตัวหนังสือไม่เล็กเกินไป ไม่มีการเลื่อนแนวนอน และปุ่มมีพื้นที่กดเพียงพอ การตรวจนี้จับปัญหาที่ test ของกติกาไม่เห็น

พร้อมตรวจมือถือ

เปิด Preview ที่ความกว้างประมาณ 390 พิกเซล
ตรวจ overflow ตัวหนังสือ ช่องกรอก ปุ่ม และลำดับการอ่าน
ถ้ามีปัญหา ให้แก้ CSS โดยรักษาหน้าตาบนเดสก์ท็อป
สรุปสิ่งที่ตรวจและภาพรวมหลังแก้



ภาพที่ 14 ตัวอย่างการแสดงผลบนมือถือ

ตรวจการใช้งานด้วยคีย์บอร์ด

กดปุ่ม Tab ซ้ำ ๆ เพื่อเลื่อนจากช่องหนึ่งไปยังอีกช่องหนึ่ง ลำดับควรเป็นไปตามที่อ่านจากบนลงล่าง เมื่อกด Enter ที่ปุ่มสร้างคำแนะนำควรทำงาน และช่องที่ผิดควรมีข้อความบอก ไม่ใช่สีเพียงอย่างเดียวในการสื่อความหมาย

ให้ Codex ตรวจจุดพื้นฐาน

พร้อมตรวจการเข้าถึง

ตรวจ accessibility ขั้นพื้นฐานของหน้าเว็บนี้

- semantic HTML และลำดับ heading
- label เชื่อมกับ input และ select
- keyboard focus ที่มองเห็นได้
- contrast ของข้อความและปุ่ม
- aria-live สำหรับผลลัพธ์ที่เปลี่ยน

รายงานปัญหาเป็น Must fix และ Nice to have ก่อนแก้ไฟล์

บทที่ 11

ตรวจ Diff และส่งมอบ

ก่อนยอมรับงาน เราจะดูว่ามีไฟล์ใดเปลี่ยน อ่าน Diff ของกติกาสำคัญ และรวบรวมหลักฐานว่าพฤติกรรมตรงตามโจทย์

Diff คืออะไร

Diff แสดงความแตกต่างระหว่างก่อนและหลังแก้ไข สีแดงมักหมายถึงสิ่งที่ลบ สีเขียวหมายถึงสิ่งที่เพิ่ม มือใหม่ไม่ต้องอ่านทุกเครื่องหมาย แต่ควรดูว่าไฟล์ที่เปลี่ยนเกี่ยวข้องกับงานหรือไม่ มีข้อมูลลบหรือคำสั่งเสียงเพิ่มเข้ามาหรือไม่ และ test เปลี่ยนตามพฤติกรรมใหม่หรือไม่

TF

ขั้นที่ 11 อ่าน Diff ก่อนยอมรับงาน
ดูสิ่งที่ลบและเพิ่มในไฟล์สำคัญ

WORKSHOP

```
rules.js

- if (!input.organization) return null;
+ const errors = validateInput(input);
+ if (Object.keys(errors).length) {
+   renderErrors(errors);
+   return;
+ }

+ test("validates incomplete input", () => {
+   assert.equal(Object.keys(errors).length, 6);
+ });
```

test ผ่านครบ

สีแดงคือสิ่งที่ลบ สีเขียวคือสิ่งที่เพิ่ม ตรวจสอบว่าเปลี่ยนแปลงเฉพาะขอบเขตที่ขอ

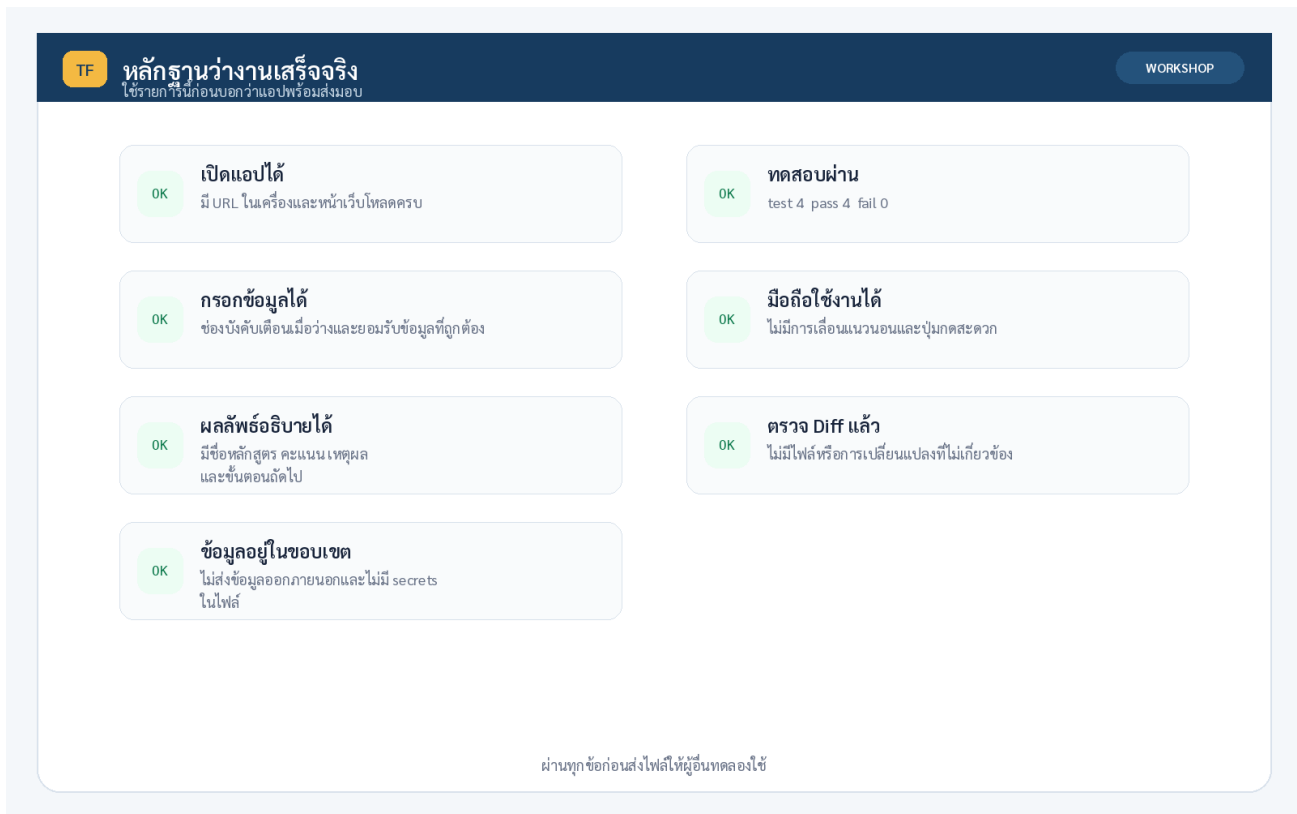
ภาพที่ 15 ตัวอย่าง Diff ของ validation และ test

พร้อมอ่าน Diff

สรุป Diff สำหรับผู้ไม่มีพื้นฐานเป็นภาษาไทย

1. ไฟล์ใดเปลี่ยนและเปลี่ยนเพื่ออะไร
2. พฤติกรรมผู้ใช้ใดเปลี่ยน
3. มี dependency network หรือข้อมูลลบเพิ่มหรือไม่
4. test ใดพิสูจน์พฤติกรรมใหม่
5. มีข้อจำกัดหรือความเสี่ยงอะไรเหลืออยู่
อย่าแก้ไขไฟล์เพิ่มในขั้นนี้

รายการหลักฐานก่อนส่งมอบ



ภาพที่ 16 หลักฐานว่าแอปพร้อมส่งมอบ

ขอรายงานส่งมอบจาก Codex

พร้อมส่งมอบ

จัดทำสรุปส่งมอบรุ่นแรก

- วิธีเริ่มแอปทีละขั้น
- URL ที่ใช้ในเครื่อง
- ไฟล์ที่สำคัญและหน้าที่
- ข้อมูลตัวอย่างสำหรับทดลอง
- ผล test ล่าสุด
- ข้อจำกัดที่ทราบ
- วิธีหยุด server
- งานที่แนะนำสำหรับรุ่นถัดไป

ห้ามอ้างว่างานพร้อม Production

หยุดเซิร์ฟเวอร์ด้วย Ctrl และ C ใน Terminal ที่รัน npm start การหยุดเซิร์ฟเวอร์ไม่ลบไฟล์หรือประวัติใน browser เพียงทำให้ URL ในเครื่องเปิดไม่ได้จนกว่าจะรัน npm start อีกครั้ง

บทที่ 12

แก้ปัญหาที่พบบ่อย

บทนี้ใช้เมื่อผลลัพธ์ไม่ตรงกับภาพตัวอย่าง ให้แก้จากอาการและหลักฐานแทนการลองเปลี่ยนหลายอย่างพร้อมกัน

อาการ	ตรวจอะไร	พรมต์ที่ควรใช้
Codex สร้างสิ่งที่ไม่ได้ขอ	อ่านแผนและรายชื่อไฟล์	หยุดงาน ระบุ out of scope และขอแผนใหม่
คำสั่ง node หรือ npm ใช้ไม่ได้	ระบบปฏิบัติการและข้อความเต็ม	ขอทางเลือกติดตั้ง ยังไม่ให้ลงมือ
npm test ไม่ผ่าน	ชื่อ test และ error แรก	ขอ root cause ก่อนแก้
URL เปิดไม่ได้	Terminal ยังทำงานหรือไม่	ส่งคำสั่ง URL และข้อความจริง
หน้าจอสวแต่ผลไม่ตรง	ข้อมูลที่กรอกและผลที่คาดหวัง	รายงานขั้นตอนทำซ้ำและเพิ่ม test
มือถือเลื่อนแนวนอน	องค์ประกอบที่กว้างเกินจอ	ตรวจ 390 พิกเซลและรักษา desktop
Codex ขอ Full Access	เหตุผลและคำสั่งเป้าหมาย	ถามหาวิธีที่ใช้สิทธิ์ต่ำกว่า

กติกากการแก้ปัญหาสำหรับมือใหม่

เปลี่ยนครั้งละหนึ่งสาเหตุ หากแก้หลายไฟล์และหลายอาการพร้อมกัน คุณจะไมรู้ว่าการเปลี่ยนใดช่วยหรือสร้างปัญหาใหม่ หลังแก้ทุกครั้ง ให้ทำซ้ำขั้นตอนเดิม รัน test และอ่าน Diff ของไฟล์ที่เกี่ยวข้อง

พรมต์วินิจัยแบบไม่เดา

แม่แบบวินิจัย

ช่วยวินิจัยจากหลักฐานต่อไปนี้

สิ่งที่ฉันกำลังทำ

[อธิบายเป้าหมายหนึ่งประโยค]

ขั้นตอนทำซ้ำ

- ...
- ...

ผลที่เกิดขึ้นจริง

[วางข้อความหรืออธิบายหน้าจอ]

ผลที่คาดหวัง

[บอกพฤติกรรมที่ต้องการ]

โปรดเสนอ 2 ถึง 3 สาเหตุ เรียงจากน่าจะเป็นที่สุด

ยังไม่ต้องแก้ไฟล์จนกว่าจะบอกวิธีพิสูจน์แต่ละสาเหตุ

เมื่อควรเริ่มแชตใหม่

หากบทสนทนายาวจน Codex เริ่มสับสนเป้าหมาย ให้เปิดแชตใหม่ใน Project เดิม แล้วสรุปสถานะปัจจุบัน ไฟล์ที่เกี่ยวข้อง สิ่งที่ทำสำเร็จ ปัญหาที่เหลือ และเกณฑ์สำเร็จ อย่าคัดลอกประวัติทั้งหมด เพราะข้อความที่ไม่เกี่ยวข้องจะเพิ่มความสับสน

ภาคผนวก ก พรอมต์ฉบับเต็มสำหรับทำ Workshop

ใช้พรอมต์ตามลำดับ อย่าส่งทั้งหมดในครั้งเดียว หลังแต่ละพรอมต์ให้อ่านคำตอบและตรวจจุดที่หนังสือระบุ ก่อนเดินหน้าต่อ

1 สํารวจโฟลเดอร์

ช่วยตรวจสอบโฟลเดอร์นี้ก่อน ยังไม่ต้องสร้างหรือแก้ไฟล์
สรุปสิ่งที่พบ ความเสี่ยง และคำถามที่ต้องการจากฉัน

2 วางแผน

ใช้ข้อกำหนด Training Needs Planner ในบทที่ 4
เสนอแผน ไฟล์ สมมติฐาน out of scope acceptance criteria และ test plan
ยังไม่ต้องแก้ไฟล์

3 สร้างรุ่นแรก

อนุมัติแผน สร้างเฉพาะไฟล์ที่ระบุ ใช้ข้อมูลตัวอย่างในเครื่อง เพิ่ม test แล้วสรุปผล

4 เปิดแอป

รัน npm test ก่อน หากผ่านให้รัน npm start และบอก URL ในเครื่องพร้อมวิธีหยุด server

5 ทดสอบแบบฟอร์ม

ตรวจกรณีไม่กรอกข้อมูล กรอกบางส่วน กรอกครบ และจำนวนผู้เรียนเกิน 30 คน รายงานสิ่งที่พบก่อนแก้

6 แก้บั๊ก

ใช้แม่แบบรายงานบั๊กในบทที่ 8 หา root cause แก้เฉพาะจุด เพิ่ม regression test และรัน test ทั้งหมด

7 เพิ่มประวัติ

เพิ่ม localStorage แสดงไม่เกิน 6 รายการ มีปุ่มล้าง และไม่ส่งข้อมูลออกนอกเครื่อง

8 ตรวจมือถือ

ตรวจหน้าเว็บที่ความกว้าง 390 พิกเซลและแก้เฉพาะ CSS ที่จำเป็น

9 ตรวจ Diff

สรุป Diff เป็นภาษาไทย แยกพฤติกรรม ความเสี่ยง dependency และหลักฐาน test

10 ส่งมอบ

จัดทำคู่มือเริ่มแอป ผล test ข้อจำกัด และงานรุ่นถัดไป โดยไม่อ้างว่าพร้อม Production

ภาคผนวก ข คำสั่งที่ใช้ใน Workshop

คำสั่ง	ใช้เมื่อใด	ผลที่คาดหวัง
node -v	ตรวจสอบ Node.js	แสดงหมายเลขเวอร์ชัน
npm -v	ตรวจสอบ npm	แสดงหมายเลขเวอร์ชัน
npm test	ทดสอบปกติ	pass 4 และ fail 0
npm start	เปิดเว็บในเครื่อง	แสดง URL 127.0.0.1
Ctrl + C	หยุด server	Terminal กลับมารับคำสั่ง

ไม่ต้องจำคำสั่งทั้งหมด คุณสามารถขอให้ Codex รันและอธิบายผลได้ สิ่งสำคัญคือรู้ว่าคำสั่งนั้นตรวจสอบหรือเปลี่ยนอะไร และควรเห็นหลักฐานใดเมื่อสำเร็จ

โครงสร้างไฟล์สุดท้าย

```
training-needs-planner/  
├── index.html  
├── styles.css  
├── app.js  
├── rules.js  
├── server.mjs  
├── package.json  
├── README.md  
├── test/  
│   └── rules.test.js
```

ภาคผนวก ค พจนานุกรมสำหรับมือใหม่

คำศัพท์	ความหมาย
App หรือ Application	โปรแกรมที่ผู้ใช้ได้ตอบเพื่อทำงานบางอย่าง
Browser	โปรแกรมเปิดเว็บไซต์ เช่น Chrome Safari หรือ Edge
Project	โฟลเดอร์และบริบทที่ Codex ใช้ทำงานร่วมกัน
HTML	โครงสร้างของหน้าเว็บ เช่น หัวข้อ แบบฟอร์ม และปุ่ม
CSS	กฎที่กำหนดสี ระยะห่าง ขนาด และการจัดวาง
JavaScript	คำสั่งที่ทำให้หน้าเว็บตอบสนองต่อผู้ใช้และคำนวณผล
Node.js	โปรแกรมที่ใช้รัน JavaScript นอกเว็บเบราว์เซอร์
npm	เครื่องมือเรียกคำสั่งของโครงการและจัดการ package
Terminal	พื้นที่สำหรับพิมพ์คำสั่งและอ่านข้อความจากโปรแกรม
Localhost	ที่อยู่ที่ชี้กลับมายังเครื่องของผู้ใช้
Validation	การตรวจว่าข้อมูลที่กรอกครบและอยู่ในรูปแบบที่ยอมรับ
localStorage	พื้นที่เก็บข้อมูลขนาดเล็กในเว็บเบราว์เซอร์ของเครื่องนั้น
Test	ตัวอย่างที่รันอัตโนมัติเพื่อตรวจว่าพฤติกรรมยังตรงข้อกำหนด
Regression test	Test ที่เพิ่มหลังพบปัญหาเพื่อกันไม่ให้อีกเดิมกลับมา
Diff	ภาพเปรียบเทียบสิ่งที่ลบและเพิ่มในไฟล์
Sandbox	ขอบเขตทางเทคนิคที่จำกัดสิ่งซึ่ง Codex ทำกับเครื่องได้
Approval	จุดที่ระบบหยุดรอให้ผู้ใช้อนุมัติการกระทำ

ภาคผนวก ง แหล่งอ้างอิง

ตรวจสอบข้อมูลวันที่ 19 กันยายน 2569 หน้าจอ ชื่อเมนู โมเดล และข้อกำหนดของผลิตภัณฑ์อาจเปลี่ยนได้ ควรตรวจสอบเอกสารทางการก่อนนำไปใช้ในหลักสูตรหรือสภาพแวดล้อมจริง

1. [วิดีโอต้นทาง Codex for Beginners Tutorial 2026 Build Your First App in Minutes](#)
2. [OpenAI Codex Best practices](#)
3. [OpenAI Prompting](#)
4. [OpenAI Projects and chats](#)
5. [OpenAI ChatGPT desktop app](#)
6. [OpenAI Sandbox](#)
7. [OpenAI Permissions](#)

หมายเหตุการใช้เนื้อหา

หนังสือเล่มนี้เรียบเรียงและขยายความใหม่เพื่อการศึกษา ไม่ใช่บทถอดเสียงหรือคำแปลแบบคำต่อคำ ภาพหน้าจอ Codex เป็นภาพประกอบเพื่ออธิบายลำดับการทำงาน ส่วนหน้าจอ Training Needs Planner แสดงแอปตัวอย่างที่สร้างและทดสอบสำหรับ Workshop นี้