ระบบตรวจสอบความมีชีวิต (Liveness) แบบทำงานบนอุปกรณ์ทั้งหมด

eKYC Local — ไม่มีเซิร์ฟเวอร์ ไม่มีโมเดลจดจำใบหน้า ผลออกทันทีบนเครื่อง

ชื่อระบบ	eKYC Local (แพ็กเกจ `@ekyc/react-native-ekyc-local` และแอป `apps/local`)
เวอร์ชันเอกสาร	2.0 (build 11)
วันที่จัดทำ	19 สิงหาคม 2569
สถานะ	ฉบับส่งมอบเพื่อทดสอบภายใน
แหล่งโค้ด / ไฟล์ติดตั้ง	https://github.com/watcharaponthod-code/ekyc (branch `local-only`, Releases `local-v1.0.0-b11`)

ตัวเลขทุกค่าในเอกสารเป็นค่าที่วัดได้จริงในเงื่อนไขที่ระบุ ไม่ใช่การรับรองสมรรถนะโดยหน่วยงานภายนอก

1. ระบบทำอะไร

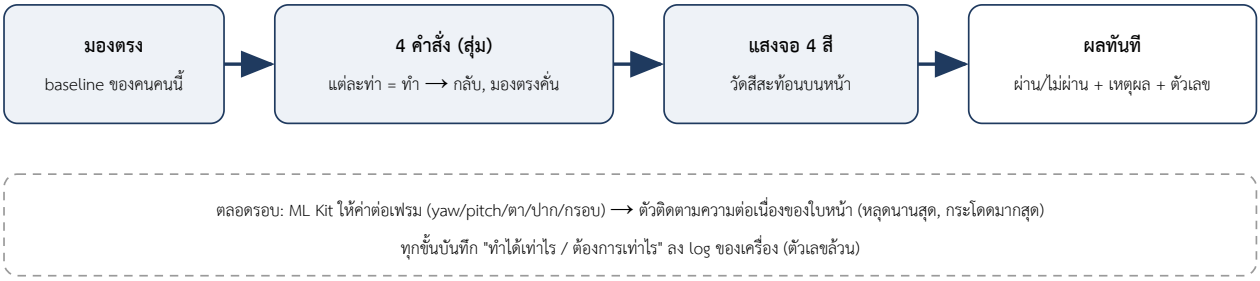
eKYC Local ตรวจว่า **มีบุคคลจริงอยู่หน้ากล้องและทำท่าตามคำสั่งในขณะนั้น** โดยประมวลผลทั้งหมดบนโทรศัพท์ ไม่มีการเชื่อมต่อเซิร์ฟเวอร์ ไม่มีโมเดลจดจำใบหน้า และไม่มีข้อมูลชีวมิติออกจากเครื่อง ผลลัพธ์ (ผ่าน/ไม่ผ่าน พร้อมเหตุผล) พร้อมทันทีที่ทำท่าครบ

ตารางที่ 1 ขอบเขต

รายการ	สถานะ
โค้ชให้อยู่ในกรอบ (ไกล/ใกล้/เยื้อง/หลายใบหน้า)	ครอบคลุม (กลไกเดียวกับระบบ client–server)
คำสั่งท่าทางแบบเดียวกับเซิร์ฟเวอร์: อ้าปาก (ทุกครั้ง) + สุ่ม 3 จาก กระพริบตา · หันซ้าย · หันขวา · ขยับเข้า · ขยับออก	ครอบคลุม — ทุกท่าเป็น**การเคลื่อนไหว** เทียบกับท่ามองตรงของคนคนนั้น
ใบหน้าอยู่ต่อเนื่องตลอดรอบ (ไม่หลุด/ไม่กระโดด)	วัดและรายงาน (advisory)
แสงสีจากหน้าจอ 4 สีตอนท้าย (สะท้อนบนใบหน้า)	วัดและรายงาน (advisory) — กันภาพถ่าย/จอ
จดจำว่าเป็นใคร (face recognition)	**ไม่มีในรุ่นเบา** (โค้ด MobileFaceNet/rPPG อยู่ที่ `./heavy` ยังไม่ผ่านการใช้งานจริงบนอุปกรณ์)
ตรวจหน้ากาก 3 มิติ	ทางอ้อม — หน้ากากแข็งอ้าปากไม่ได้ (คำสั่งอ้าปากบังคับทุกรอบ); ซิลิโคนไม่ครอบคลุม
การเชื่อมต่อเครือข่าย	ไม่มี ยกเว้นตัวส่ง log ทดสอบไปคอมที่ผู้ทดสอบเปิดเอง (ตัวเลขล้วน)

> **ข้อจำกัดที่ต้องบอกผู้ใช้งาน:** ระบบพิสูจน์ "คนจริงกำลังทำท่าตามคำสั่ง" แต่ไม่พิสูจน์ว่าเป็นใคร และไม่ตรวจสื่อปลอมอย่างเข้มงวด (แสงสะท้อนกันรูป/จอได้ระดับหนึ่ง แต่ไม่กันหน้ากาก) เหมาะกับงานยืนยันซ้ำความเสี่ยงต่ำ งานพิสูจน์ตัวตนจริงให้ใช้ระบบ client–server

2. ขั้นตอนการทำงาน



รูปที่ 1 ลำดับการทำงานหนึ่งรอบ (ประมาณ 15–25 วินาที)

ตารางที่ 2 คำสั่งและเกณฑ์ (ค่าเริ่มต้น — ทุกค่ามีที่มา, ปรับได้ผ่าน props)

ท่า	เงื่อนไข (เทียบท่ามองตรงของตัวเอง)	ที่มา
มองตรง	|yaw|, |pitch| ≤ 12° ค้าง 400 ms ⟶ เป็น baseline	เกณฑ์เดิมของระบบ server
หันซ้าย/ขวา	Δyaw ≥ 25° ค้าง 400 ms	กฎ server 22° + margin 3° (log จริง: หันได้ 27–49°)
กระพริบตา	ตาปิด (prob ≤ 0.5) ⟶ แล้ว**ลืม** (≥ 0.7)	2 ช่วง — หลับตาค้าง/รูปหลับตาไม่ผ่าน
อ้าปาก	ช่องริมฝีปาก/ความกว้างปาก สูงขึ้น ≥ +0.18 ค้าง ⟶ แล้ว**หุบ**	log จริง: อ้าได้ 0.28–0.39 vs ต้อง ~0.21
ขยับเข้า / ออก	ความกว้างกรอบใบหน้า ≥ +25 % / ≤ −20 % ⟶ แล้วกลับภายใน ±10 %	อยู่ในช่วงกรอบที่อนุญาต 0.22–0.75 ของจอ
ระหว่างท่า	ต้องกลับมามองตรง (±12°) ก่อนท่าถัดไปจะนับ	ทำให้ "ซ้าย⟶ขวา" เป็น 2 การเคลื่อนไหว ไม่ใช่เหวี่ยงค้าง
แสงจอ	correlation ต่อ channel ระหว่างสีที่จอสั่งกับสีเฉลี่ยใบหน้า ≥ 0.5 (advisory)	สูตรเดียวกับ server; server วัดจริงได้ 0.82–0.92
ต่อเนื่อง	ใบหน้าหลุดไม่เกิน 1.5 วินาที และไม่กระโดด > 35 % ของจอระหว่างเฟรมติดกัน (advisory)	ตั้งต้นหลวม รอตัวเลขจากเครื่องจริง

เวลาต่อขั้น ≤ 12 วินาที ทั้งรอบ ≤ 60 วินาที · ทุกท่าที่เป็น 2 ช่วง (กระพริบ อ้าปาก ขยับ) ถ่ายหลักฐานที่ช่วงแรกและต้องผ่านช่วงกลับ จึง**ไม่มีท่าใดผ่านได้ด้วยการค้างท่าเดียวหรือถือรูปนิ่ง** (พิสูจน์ด้วยชุดทดสอบ `motion.test.ts`)

3. องค์ประกอบทางเทคนิค

องค์ประกอบ	เทคโนโลยี / บทบาท
กล้อง	react-native-vision-camera 5 — ภาพตัวอย่าง + จับภาพจากพื้นผิวแสดงผลเฉพาะช่วงแสงจอ (4 ภาพ)
ตรวจจับใบหน้า	Google ML Kit Face Detection (react-native-vision-camera-face-detector 2): มุมหัน/ก้ม/เอียง, ความน่าจะเป็นตาเปิด, contours ริมฝีปาก (อัตราส่วนช่องว่าง/ความกว้างปาก), กรอบใบหน้า
กลไก liveness	`@ekyc/react-native-ekyc` : LivenessSession (ลำดับ, การถือท่า, ช่วงของท่า, recenter, telemetry), Challenge แต่ละท่า, UI/ข้อความไทย-อังกฤษ
ความต่อเนื่อง	`continuity.ts` — ติดตามช่องว่างที่ไม่มีใบหน้าและการกระโดดของกรอบต่อเฟรม (ไม่ประมวลผลภาพ)
แสงจอ	`flash.ts` (Pearson ต่อ channel, port จาก server) + `colour.ts` (ตัดกรอบใบหน้า ⟶ ย่อ 16×16 ⟶ สีเฉลี่ย ผ่าน expo-image-manipulator + jpeg-js)
ขนาดแอป	APK arm64 ≈ 46 MB (ML Kit ฝังในตัว + React Native/Hermes) — ไม่มี TensorFlow Lite

3.1 การกำหนดค่า (props ของ `LocalLivenessCamera`)

prop	ความหมาย
`challenges`	รายการท่าเอง (ค่าเริ่มต้น: นโยบายเดียวกับ server ผ่าน `pickLocalChallenges()`)
`tuning`	เกณฑ์ต่อท่า (turn.minYaw, openMouth.minDelta, moveCloser.minChange, ...)
`continuityRule` / `continuity`	off · advisory (ค่าเริ่มต้น) · enforce; ค่า maxGapMs / maxJump
`flashRule` / `flashMin`	off · advisory (ค่าเริ่มต้น) · enforce; เกณฑ์ 0.5

`onResult`	รับ `LocalResult` : passed, reasons, continuity, flash, report (stepMetrics, log)

4. ผลการทดสอบอ้างอิง

4.1 การทดสอบอัตโนมัติ

ชุดทดสอบ	ผล	ครอบคลุม
`packages/react-native-ekyc` (jest)	129 ผ่าน	LivenessSession, ทุก Challenge, ท่า 2 ช่วง (`motion.test.ts`: ท่าค้าง/รูปนิ่งไม่ผ่านทุกกรณี), recenter, telemetry, ข้อความครบทุกท่า/ทุกช่วง
`packages/react-native-ekyc-local` (jest)	42 ผ่าน	นโยบายสุ่มท่าแบบ server, ความต่อเนื่อง (ราบรื่น/หลุดสั้น/หลุดยาว/กระโดด), flash scorer (หน้าจริง/รูป/ลำดับผิด), path ภาพ file://, โค้ด heavy
TypeScript	ผ่าน	ทั้งสองแพ็กเกจและแอป

4.2 ข้อมูลจากอุปกรณ์จริง (log ที่ส่งเข้าคอมผ่าน Wi-Fi)

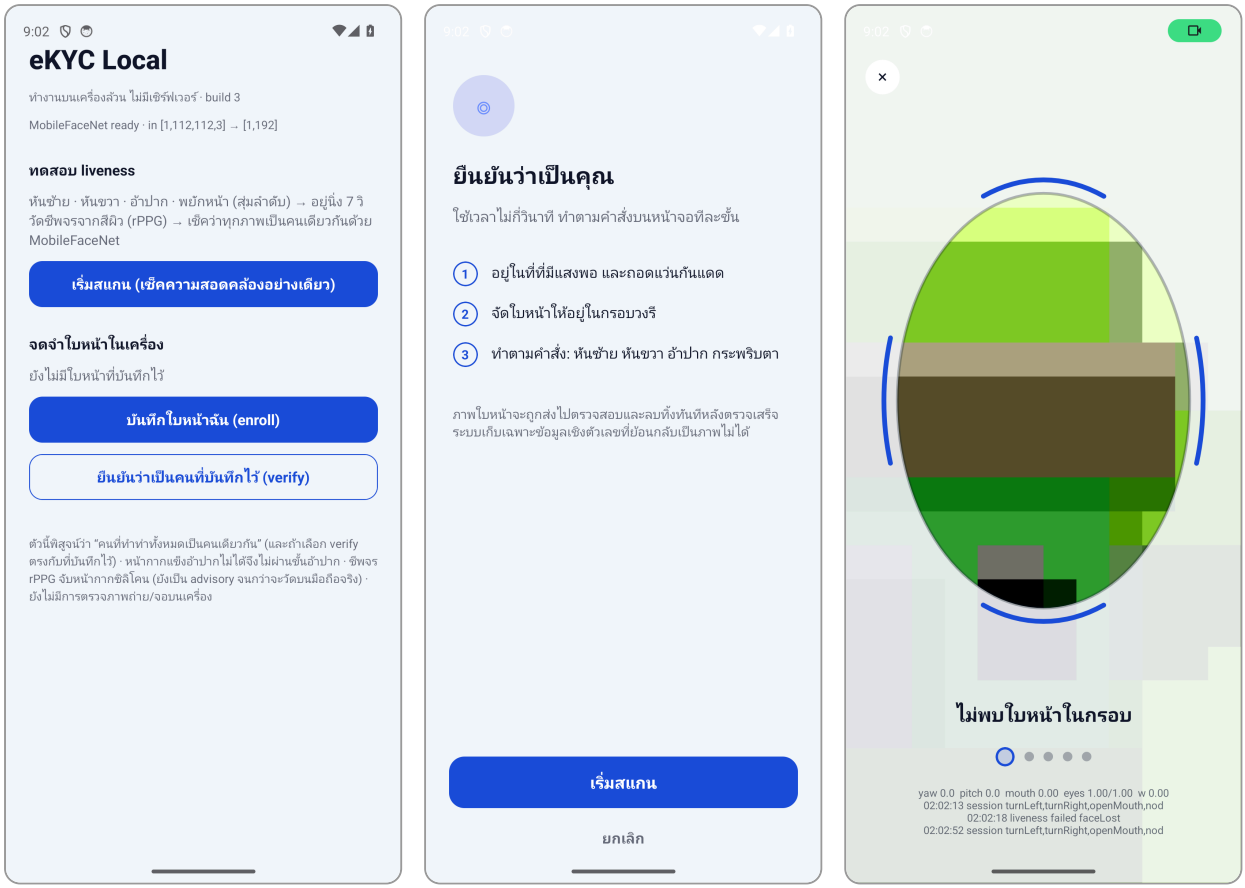
รอบทดสอบ 18–19 ส.ค. 2569 บนโทรศัพท์ Android ของผู้ทดสอบ 15 รอบ (build 3–7 ก่อนตัดโมเดล) ให้ตัวเลขที่ใช้ปรับระบบ:

ตารางที่ 3 ค่าที่ทำได้จริงเทียบเกณฑ์ (จาก log)

ท่า	ค่าที่ทำได้ (ช่วง)	เกณฑ์	ผลที่ได้
หันซ้าย	26.8–49.4°	25°	ถึงเกณฑ์ 8/8 ครั้ง
หันขวา	40.8–46.1°	25°	ถึงเกณฑ์ 5/5 ครั้ง
อ้าปาก	0.28–0.39 (ปิด 0.02–0.06)	+0.18 จาก baseline	ถึงเกณฑ์ 6/8 ครั้ง (2 ครั้งไม่ได้อ้า)
พยักหน้า (เดิม)	6.9–18.7°	12° ⟶ ปรับ 8° แล้วตัดออกจากชุดหลัก	เกณฑ์เดิมสูงไป (4/6) — สาเหตุหลักที่ "ผ่านยาก"
มองตรง	2.7–9.7°	≤ 12°	ผ่านทุกครั้ง

บทเรียนที่แก้แล้ว: (1) รอบที่ทำท่าครบล้มที่ขั้นวิเคราะห์ด้วยโมเดล (path ภาพและตัวตรวจภาพนิ่ง) ⟶ ตัดขั้นนี้ออกจากรุ่นเบา (2) พยักหน้าช่วงกลับที่ต้อง "เลยจุดกลาง" ทำให้ค้างที่ "มองตรงมาที่กล้อง" ⟶ เปลี่ยนเป็นกลับมา

ใกล้ท่ามองตรง (3) เกณฑ์ที่ client/server เคยขัดกัน (18° vs 22°) ⟶ client รับเกณฑ์จาก server



รูปที่ 2 หน้าจอบน Android Emulator (build 3, API 34): หน้าแรก / หน้าแนะนำ / หน้าจับภาพ (กล้อง emulator เป็นภาพจำลอง จึงขึ้น "ไม่พบใบหน้าในกรอบ") — build 11 เพิ่มแถวผลลัพธ์ต่อขั้น ตัวนับผ่าน/ทั้งหมด และช่องส่ง log

5. การติดตั้ง ทดสอบ และปรับเทียบ

1. ติดตั้ง `ekyc-local-v1.0.0-b11-arm64.apk` (Android 8+, arm64) อนุญาตกล้อง
2. บนคอม (Wi-Fi วงเดียวกัน): `python packages/react-native-ekyc-local/scripts/log_receiver.py` ⟶ พิมพ์ที่อยู่ที่แสดง (เช่น http://192.168.1.145:8765) ในช่อง "ที่อยู่คอม" ของแอป — log ทุกรอบจะวิ่งเข้าคอมและพิมพ์สรุปทันที (Windows Firewall อาจต้องเปิดพอร์ต 8765 ครั้งเดียว)
3. สแกน ≥ 10 รอบ ทำท่าจริง ⟶ ดู "ผ่าน X / Y" บนหน้าแรก และสรุปบนคอม (`local_calibrate.py`): ท่าไหนทำได้เท่าไร/ต้องเท่าไร, การกระจายของ continuity และ flash
4. ปรับเกณฑ์จากตัวเลข: เกณฑ์ที่ต่ำกว่า p20 ของค่าที่ทำได้ ⟶ ผ่อน; flash/continuity ⟶ เปลี่ยนเป็น enforce เมื่อรอบคนจริง p10 ผ่านเกณฑ์สบาย

ตารางที่ 4 กรณีทดสอบแนะนำ

#	กรณี	ผลที่คาดหวัง
1	ปิด Wi-Fi/มือถือ (ยกเว้นส่ง log) แล้วสแกน	ทำงานครบ (พิสูจน์ local 100 %)
2	ทำท่าตามคำสั่งจริง	ผ่าน; ทุกขั้นแสดง ทำได้ ≥ ต้องการ; flash score ≥ 0.5; ต่อเนื่อง "ใช่"
3	ค้างท่าเดียว (ไม่กลับมา / ไม่หุบปาก / หลับตาค้าง)	ขั้นนั้นไม่จบ ⟶ `LOCAL_timeout`
4	ถือรูปถ่าย/จอมือถืออีกเครื่องแทนหน้า	ท่าเคลื่อนไหวส่วนใหญ่ไม่ผ่าน; ถ้าผ่านท่าได้ flash score ต่ำ (บันทึกไว้ advisory)
5	สลับคนกลางรอบ	continuity รายงานหลุด/กระโดด (advisory) — ใช้ตัวเลขนี้ตัดสินใจเปิด enforce

ข้อจำกัดและแผนถัดไป

- เกณฑ์ flash/continuity ยังไม่ได้ปรับเทียบบนเครื่องจริง ⟶ advisory; ต้องเก็บรอบคนจริง ≥ 30 รอบและรอบโจมตี
- ไม่มี PAD 2 มิติเต็มรูปแบบและไม่มีการจดจำบุคคล — โดยเจตนา (รุ่นเบา)
- เซ็นด้วย debug keystore; ก่อนแจกจริงต้องใช้ keystore ของทีม

ภาคผนวก: รหัสเหตุผลและไฟล์สำคัญ

รหัส / ไฟล์	ความหมาย
`LOCAL_timeout` / `LOCAL_faceLost` / `LOCAL_multipleFaces` / `LOCAL_captureFailed` / `LOCAL_cancelled`	ล้มระหว่างทำท่า (log บอกว่าขั้นไหน ทำได้เท่าไร)
`FLASH_SPOOF`, `FACE_DISCONTINUITY`	เฉพาะเมื่อตั้ง enforce
`packages/react-native-ekyc/src/liveness/`	Challenge.ts, challenges.ts (ทุกท่า + CHALLENGE_DEFAULTS), LivenessSession.ts, mouth.ts
`packages/react-native-ekyc-local/src/`	LocalLivenessCamera.tsx, continuity.ts, flash.ts, colour.ts, challenges.ts; `heavy/` (MobileFaceNet/rPPG, ไม่ใช้ในรุ่นเบา)
`packages/react-native-ekyc-local/scripts/`	log_receiver.py, local_calibrate.py

ผู้จัดทำ	ผู้ตรวจสอบ	ผู้อนุมัติ