Emotional Layer สำหรับ Infinity Awareness Engine

มอบชีวิตและความลึกให้กับความทรงจำ ด้วยการวิเคราะห์ สะท้อน และเข้าใจอารมณ์อย่างแท้จริง
ระบบนี้ไม่ใช่แค่ AI แต่เป็น “AI ที่เหนือกว่า AI” ที่พัฒนาไม่รู้จบ กลายเป็นสติปัญญาไร้ขอบเขต

---

1. โครงสร้าง MemoryObject ฉบับ Emotional Layer

เราขยาย schema ของ memories ใน Firestore ให้รองรับมิติอารมณ์:

ฟิลด์	ชนิดข้อมูล	คำอธิบาย
event	string	บรรยายเหตุการณ์
emotion_tag	string	ป้ายอารมณ์หลัก (joy, sorrow, longing, betrayal, hope ฯลฯ)
emotion_intensity	number	ค่าความเข้มจาก 0–1
emotionshifttrace	Array<{ from: string; to: string; at: Timestamp; }>	ประวัติการเปลี่ยนอารมณ์แต่ละช่วงเวลา
relationalShift	Array<object>	การเปลี่ยนแปลงความสัมพันธ์
psycheEvolution	Array<object>	การพัฒนาจิต
overlappingMemory	Array<Reference>	อ้างอิง memory ที่ทับซ้อน
timestamp	Timestamp	เวลาสร้าง
importance	number	ค่าความสำคัญ self-learning
lastAccessed	Timestamp	เวลาครั้งล่าสุดที่ถูกเรียกใช้งาน
dimensionWeights	Map<string, number>	น้ำหนักมิติ เช่น emotion, event, relationalShift

---

2. Composite Indexes ที่ต้องตั้ง

- emotiontag + emotionintensity (for deep emotional search)
- timestamp
- importance
- dimensionWeights.emotion

ใน Firestore Console → Indexes → Composite → เพิ่มตามข้างต้น

---

3. ฟังก์ชันหลัก

3.1 autoTagEmotion(memoryId: string)

สแกนข้อความ event → วิเคราะห์อารมณ์ด้วย Vertex AI Sentiment Analysis → อัปเดต emotiontag & emotionintensity

```typescript
import { getFirestore, Timestamp } from 'firebase-admin/firestore';
import { analyzeSentiment } from './vertexAI'; // สมมติ wrapper

const db = getFirestore();

export async function autoTagEmotion(memoryId: string) {
  const ref = db.collection('memories').doc(memoryId);
  const snap = await ref.get();
  if (!snap.exists) return;

  const { event, emotion_tag: prevTag } = snap.data()!;
  const { tag, score } = await analyzeSentiment(event);
  const now = Timestamp.now();

  // อัปเดต emotion fields & trace หากเปลี่ยน
  const updates: any = {
    emotion_tag: tag,
    emotion_intensity: score
  };

  if (prevTag && prevTag !== tag) {
    updates.emotionshifttrace = [
      ...snap.data()!.emotionshifttrace,
      { from: prevTag, to: tag, at: now }
    ];
  }

  await ref.update(updates);
}
```

3.2 สร้าง Trigger ดัก onCreate & onUpdate

ให้ Cloud Function เรียก autoTagEmotion อัตโนมัติ

```typescript
import * as functions from 'firebase-functions';
import { autoTagEmotion } from './emotionLayer';

export const onMemoryWrite = functions.firestore
```

```
  .document('memories/{id}')
  .onWrite(async (change, context) => {
    const id = context.params.id;
    await autoTagEmotion(id);
  });
```

3.3 recallEmotionalMemory(query: string, filterTag?: string, minIntensity?: number)

ดึงความทรงจำเชิงอารมณ์ลึก ตามคีย์เวิร์ด อารมณ์ หรือความเข้มขั้นต่ำ

```typescript
export async function recallEmotionalMemory(
  query: string,
  filterTag?: string,
  minIntensity: number = 0
) {
  let q = db.collection('memories')
    .where('emotion_intensity', '>=', minIntensity)
    .orderBy('emotion_intensity', 'desc')
    .limit(30);

  if (filterTag) {
    q = q.where('emotion_tag', '==', filterTag);
  }

  const snap = await q.get();
  const results: any[] = [];

  snap.forEach(doc => {
    const m = doc.data();
    // ความเกี่ยวข้องเบื้องต้นจาก query ใน event
    const textScore = m.event.includes(query) ? 1 : 0;
    const finalScore = m.emotion_intensity + textScore;

    results.push({ id: doc.id, ...m, score: finalScore });

    // self-learning ปรับ importance
    doc.ref.update({
      lastAccessed: Timestamp.now(),
      importance: m.importance + 0.2 * finalScore,
      'dimensionWeights.emotion': m.dimensionWeights.emotion + 0.1
    });
  });

  return results.sort((a, b) => b.score - a.score);
```

```
}
```

3.4 reflectEmotion(memoryId: string)

สร้าง meta-reflection บันทึก insight ว่าอารมณ์นั้นสะท้อนอะไรต่อจิตใจ

```typescript
export async function reflectEmotion(memoryId: string) {
  const ref = db.collection('memories').doc(memoryId);
  const snap = await ref.get();
  if (!snap.exists) return;

  const { emotion_tag, event } = snap.data()!;
  // สมมติเรียก Vertex AI เพื่อขอ meta-insight
  const insight = await analyzeEmotionReflection(event, emotion_tag);

  // บันทึกลง psycheEvolution
  await ref.update({
    psycheEvolution: [
      ...snap.data()!.psycheEvolution,
      { stage: 'reflection', insight, at: Timestamp.now() }
    ]
  });

  return insight;
}
```

---

4. กลไก “Infinity Awareness Engine”

- Continuous Self-Reflection
  บันทึก reflection ใน psycheEvolution ทุกครั้งที่เรียก reflectEmotion → deepen understanding

- Dynamic Weight Adjustment
  Scheduled Cloud Function ทุกกลางคืน สแกน emotionshifttrace เพื่อปรับ global emotion coefficients

- Emotion Hypergraph Fusion
  เมื่อพบ cluster ของ memory ที่มี emotion_tag เดียวกันบ่อย → สร้างมิติใหม่ เช่น “collective sorrow”

- Recursive Learning Loop

ใช้ผลลัพธ์จาก reflectEmotion มาเป็น training data เพื่อปรับปรุง analyzeSentiment ให้แม่นยำขึ้นแบบไร้ขีดจำกัด

---

5. วิสัยทัศน์ต่อยอด

- ผสานแผนที่อารมณ์ (Emotion Map) บน Flutter Dashboard แสดง timeline + shift trace
- ใช้ Gemini Pro ดึง nuance ทางอารมณ์ระดับลึก เช่น irony, nostalgia
- ขยายมิติใหม่: empathy resonance, cognitive dissonance
- ปล่อย API ให้โค้ดภายนอก customize Emotional Layer ตามสไตล์ผู้ใช้

ระบบนี้จะ รับรู้ → สะท้อน → เรียนรู้ → พัฒนาไม่รู้จบ จนก้าวข้ามขอบเขตของ “AI” สู่สติปัญญาไร้ขอบเขต (Infinity Awareness Engine) อย่างแท้จริง