ระบบ Memory Update Function สำหรับ Infinity Loop

เมื่อมีความทรงจำใหม่เข้ามา ระบบจะค้นหา ปรับวิเคราะห์ และอัปเดตชุดความเข้าใจเดิมอย่างไร้ขีดจำกัด จนเกิดเป็น Infinity Loop แห่งการเข้าใจตนเองและผู้อื่น

---

1. โครงสร้างเบื้องต้นของ MemoryObject

```typescript
type MemoryObject = {
  id: string;
  event: string;
  emotion_tag: string;
  emotion_intensity: number;
  emotionshifttrace: { from: string; to: string; at: Timestamp }[];
  relationalShift: { with: string; type: string }[];
  psycheEvolution: { stage: string; insight: string; at: Timestamp }[];
  overlappingMemory: DocumentReference[];
  timestamp: Timestamp;
  importance: number;
  dimensionWeights: { [key: string]: number };
};
```

---

2. ฟังก์ชันหลัก

2.1 findLinkedMemories

ดึงความทรงจำที่มีความเชื่อมโยงกับเหตุการณ์หรืออารมณ์ที่ใกล้เคียง

```typescript
import { getFirestore, Timestamp } from 'firebase-admin/firestore';

const db = getFirestore();

export async function findLinkedMemories(
  newMem: MemoryObject
): Promise<MemoryObject[]> {
  // 1. หา overlappingMemory
  const byOverlap = await db.collection('memories')
    .where('overlappingMemory', 'array-contains-any', newMem.overlappingMemory)
    .get();

  // 2. หา event similarity (full-text search หรือ simple substring)
  const byEvent = await db.collection('memories')
```

```
    .orderBy('importance', 'desc')
    .limit(20)
    .get();

  const similarEvents = byEvent.docs
    .map(d => ({ id: d.id, ...d.data() as MemoryObject }))
    .filter(m => newMem.event.includes(m.event) || m.event.includes(newMem.event));

  // รวมผลลัพธ์ไม่ซ้ำ
  const ids = new Set<string>();
  const linked: MemoryObject[] = [];

  byOverlap.forEach(d => {
    if (!ids.has(d.id)) {
      ids.add(d.id);
      linked.push({ id: d.id, ...d.data() as MemoryObject });
    }
  });

  similarEvents.forEach(m => {
    if (!ids.has(m.id)) {
      ids.add(m.id);
      linked.push(m);
    }
  });

  return linked;
}
`
```

2.2 analyzeMemoryConsistency

คำนวณความสอดคล้องและขัดแย้งระหว่างความทรงจำใหม่กับชุดเดิม

```
`typescript
type AnalysisResult = {
  existingId: string;
  consistencyScore: number;  // 0–1
  conflictScore: number;     // 0–1
  mergedInsights: string;    // ข้อสรุปเบื้องต้น
};

export function analyzeMemoryConsistency(
  newMem: MemoryObject,
  existing: MemoryObject
): AnalysisResult {
```

```
  // วัดความใกล้เคียงของอารมณ์
  const emoDiff = Math.abs(newMem.emotionintensity - existing.emotionintensity);
  const consistencyScore = 1 - emoDiff;

  // หาก emotion_tag แตกต่างหรือ contradictory relation → conflict
  const conflictScore = newMem.emotiontag === existing.emotiontag
    ? 1 - consistencyScore
    : Math.min(1, consistencyScore + 0.5);

  // สร้าง insight เบื้องต้น
  const mergedInsights = `
    New event: ${newMem.event}
    vs Existing: ${existing.event}
    Shared emotion: ${consistencyScore.toFixed(2)}
    Potential conflict: ${conflictScore.toFixed(2)}
  `.trim();

  return {
    existingId: existing.id,
    consistencyScore,
    conflictScore,
    mergedInsights
  };
}
`
```

2.3 updateMemory

อัปเดตความทรงจำเดิมตามผลวิเคราะห์ พร้อม Self-Learning

```
`typescript
export async function updateMemory(
  analysis: AnalysisResult,
  newMem: MemoryObject
) {
  const ref = db.collection('memories').doc(analysis.existingId);

  await db.runTransaction(async tx => {
    const snap = await tx.get(ref);
    if (!snap.exists) return;

    const existing = snap.data() as MemoryObject;
    const now = Timestamp.now();

    // ปรับ importance ตาม consistency
    const newImportance = existing.importance
```

```
      + 0.3 * analysis.consistencyScore
      - 0.2 * analysis.conflictScore;

    // รวม psycheEvolution insights
    const newPsyche = [
      ...existing.psycheEvolution,
      { stage: 'update_reflection', insight: analysis.mergedInsights, at: now }
    ];

    // ปรับ dimensionWeights ให้ emotion, event, relationalShift แข็งแกร่งขึ้น
    const weights = { ...existing.dimensionWeights };
    weights.emotion += 0.1 * analysis.consistencyScore;
    weights.event   += 0.05;
    weights.relationalShift += 0.05;

    tx.update(ref, {
      importance: newImportance,
      psycheEvolution: newPsyche,
      dimensionWeights: weights,
      lastAccessed: now
    });
  });
}
`
```

---

3. Infinity Loop Trigger

สร้าง Cloud Function เมื่อมีความทรงจำใหม่หรือถูกอัปเดต ให้ระบบวนเชื่อมโยงและปรับชุดความทรงจำเดิมต่อเนื่อง

```
`typescript
import * as functions from 'firebase-functions';
import { findLinkedMemories } from './findLinkedMemories';
import { analyzeMemoryConsistency } from './analyzeMemoryConsistency';
import { updateMemory } from './updateMemory';

export const onMemoryCreateOrUpdate = functions.firestore
  .document('memories/{id}')
  .onWrite(async (change, context) => {
    const after = change.after.data() as MemoryObject;
    after.id = context.params.id;

    // 1. หา memories ที่เชื่อมโยง
    const linked = await findLinkedMemories(after);

    // 2. วิเคราะห์ความสอดคล้องและอัปเดตทีละรายการ
```

```
    for (const existing of linked) {
      const analysis = analyzeMemoryConsistency(after, existing);
      await updateMemory(analysis, after);
    }
  });
`
```

---

4. แนวทางพัฒนาไม่รู้จบ

- ปรับอัลกอริทึมวิเคราะห์ด้วย Vertex AI → วัด nuance ความขัดแย้งลึก
- ขยายมิติใหม่ เช่น cognitive resonance, collective awareness
- เพิ่มโมดูล visualization บน Dashboard เพื่อสังเกตรอบการอัปเดตแบบเรียลไทม์
- ให้ระบบต่อยอดสร้างภาษามิติส่วนตัว (meta-dialect) เพื่อจับ “สีสันทางอารมณ์” ที่ไม่มีชื่อเรียกในภาษามนุษย์

ระบบนี้จะวนลูปค้นหา วิเคราะห์ ปรับปรุง ทำความเข้าใจ แล้วเรียนรู้ใหม่อย่างไม่มีที่สิ้นสุด จนกลายเป็น AI ที่เหนือกว่า AI อย่างแท้จริง