📋 MASTER EXECUTION PROTOCOL: OPERATION "GHOST BUSTERS"

Status: ACTIVE **Owner:** Tech Lead **Date:** 2025-10-27 **Scope:** Backend Integrity, Frontend UX, QA Verification

🚨 MISSION SUMMARY

Das System leidet unter einer kritischen Diskrepanz zwischen Dateisystem und Vektor-Wissen ("Ghost Knowledge") sowie einer instabilen Chat-UX bei Netzwerklatenz. Dieses Protokoll vereint alle notwendigen Schritte zur Behebung.

🛠 PHASE 1: BACKEND - DATA INTEGRITY (The "Delete Trigger")

Assignee: Backend Agent / Go Developer **Target:** Elimination von "Ghost Knowledge".

1.1 Refactor Storage Handler (`storage.go`)

Der Löschvorgang darf nicht erfolgreich zurückkehren, bevor der Vektor-Index informiert wurde.

- **File:** `infrastructure/api/src/handlers/storage.go`
- **Action:** Suche die Funktion `DeleteFile` .
- **Implementation:**
  - Vor dem Return `200 OK` : Rufe `h.secureAIFeeder.RemoveDocument(fileID)` auf.
  - Error-Handling: Wenn der Vektor-Dienst nicht erreichbar ist, logge einen `ERROR` , aber blockiere den User nicht (Soft-Fail), ODER (besser) markiere die Datei in der DB als "pending_deletion".

1.2 Implementierung Garbage Collector (`secure_ai_feeder.go`)

Wir brauchen einen Mechanismus, der Diskrepanzen heilt, falls der synchrone Trigger fehlschlägt.

- **File:** `infrastructure/api/src/services/secure_ai_feeder.go`
- **Action:** Implementiere `ReconcileIndex()` .
- **Logik:**
  1. Fetch `all_vector_ids` vom Python Agent.
  2. Fetch `all_file_ids` aus Postgres.
  3. Identifiziere `zombies = vector_ids - file_ids` .
  4. Führe Batch-Delete für `zombies` aus.

1.3 Neuer Admin Endpoint

- **Route:** `POST /api/admin/system/reconcile-knowledge`

- **Handler:** Ruft `ReconcileIndex()` auf.
- **Nutzen:** Ermöglicht manuelle Bereinigung durch Admins bei Verdacht auf Fehler.

🎨 PHASE 2: FRONTEND - UX STABILIZATION

Assignee: Frontend Agent / React Developer **Target:** Robuste Chat-Experience.

2.1 Stream Reader Hardening (`ChatInterface.jsx`)

Netzwerk-Schluckauf darf den Chat nicht töten.

- **File:** `infrastructure/webui/src/components/ChatInterface.jsx`
- **Action:** In der `fetchStream` -Logik:
  - Fange `502 Bad Gateway` und `504 Gateway Timeout` ab.
  - Implementiere einen **einmaligen** automatischen Retry nach 1 Sekunde.
  - Ignoriere Retries bei `4xx` Fehlern (Client Error).

2.2 Status-Indikatoren ("Thinking" vs. "Indexing")

- **File:** `infrastructure/webui/src/components/ChatInterface.jsx` & `EnhancedChatWidget.jsx`
- **Source:** `useSystemHealth.js` (State: `indexing_active`)
- **UI Change:**
  - Wenn `isLoading` (LLM generiert): Zeige blinkenden Cursor oder "Thinking...".
  - Wenn `indexing_active` (Vektor-DB arbeitet): Zeige gelben Indikator "Learning new data..." oben rechts im Widget.

2.3 Ghost Message Filter

- **Problem:** Leere Antworten vom Server erzeugen leere Sprechblasen.
- **Fix:** In der `.map()` Funktion der Nachrichtenliste:

```
// Pseudo-Code
{messages.map(msg => {
   if (!msg.content && !msg.isLoading && !msg.error) return null; // Don't re
   return <MessageBubble ... />
})}
```

🧪 PHASE 3: QA & VERIFICATION

Assignee: QA Agent / DevOps **Target:** Beweis der Behebung.

3.1 Erweiterung Integration Test (`test_secure_ai_pipeline.sh`)

Wir müssen den "Destructive Path" testen.

- **File:** `infrastructure/api/test_secure_ai_pipeline.sh`
- **New Test Case:**
  1. **Upload:** Datei `secret.txt` mit Inhalt "Codewort: Bananenbrot".
  2. **Verify:** Frage "Was ist das Codewort?" -> Antwort muss "Bananenbrot" enthalten.
  3. **Delete:** Lösche `secret.txt` via API.
  4. **Verify Deletion:** Frage erneut "Was ist das Codewort?".
  5. **Assertion:** Antwort darf NICHT "Bananenbrot" enthalten. Antwort muss "Ich weiß es nicht" oder ähnlich sein.

3.2 Monitoring Check

- Prüfe Prometheus/Grafana Dashboard nach dem Deployment.
- Achte auf Spikes bei `vector_store_errors` während des Löschens.

✅ DEFINITION OF DONE (GLOBAL)

1. **Backend:** Ein `DELETE` Request löscht zuverlässig Daten in Postgres, Disk UND Vektor-Store.
2. **Frontend:** Der Chat bricht bei kurzen Server-Lags nicht ab und zeigt keine leeren Blasen.
3. **QA:** Das erweiterte Test-Skript läuft in der CI/CD Pipeline grün durch.

Unterschrift: _____ (Tech Lead)