**🟢 Final Jira Comment — TDA-18: Manual Location Management & Gold Records**

**Summary**  
Implemented full Admin CRUD system for manual location management with audit and gold-record integration.  
Admins can now manually add, edit, and verify locations via the AdminPanel UI. All admin actions are validated by a secure API key and automatically logged for both auditing and training data enrichment.

**Scope Implemented**

* **Backend**
  + Added /api/v1/admin router with endpoints:
    - POST /admin/locations → manually create and auto-verify new locations
    - PATCH /admin/locations/{id} → edit and verify existing records
    - GET /admin/ping → health check for admin access
  + Integrated CORS and header-based X-Admin-Key authentication.
  + Full audit logging through ai\_logs (action\_type='admin.create'| 'admin.update').
  + Gold-record system implemented in training\_data:
    - Inserts one record per manual action.
    - label\_action='keep', is\_gold\_standard=true.
    - JSON fields input\_data and expected\_output filled with structured data.
* **Frontend**
  + Added **AdminPanel** React component with two forms:
    - “Create New Location” (auto-VERIFIED).
    - “Edit Existing Location” (updates state, confidence, notes).
  + Configurable API base URL & admin key (via environment variable or localStorage).
  + Integrated with Axios and React Router /admin.

**Acceptance Criteria ✅**

* Admin can add new locations with all core fields.  
  → Confirmed, creates location id=3416 with state VERIFIED.
* Admin can edit any existing location.  
  → PATCH endpoint operational (authentication required).
* All manual actions logged for audit.  
  → Verified: ai\_logs entry present.
* Manually added/verified records stored as gold records for LearningBot.  
  → Verified: training\_data row created with correct schema.

**Definition of Done ✅**

* Admin CRUD operations functional (UI + API).
* Audit logging implemented.
* Gold record system working and compliant with DB constraints.
* CORS and schema issues resolved.

**Notes / Next Steps**

* The admin key is currently stored in localStorage for local testing.  
  In production, inject it via **Vite environment variables**:
  1. In your frontend project root, create a file .env.local:
  2. VITE\_ADMIN\_API\_KEY=ADMIN\_KEY\_2025\_!Zy3K9pQ7
  3. Access it in React:
  4. const ADMIN\_API\_KEY = import.meta.env.VITE\_ADMIN\_API\_KEY;
  5. axios.defaults.headers.common['X-Admin-Key'] = ADMIN\_API\_KEY;
  6. Remove any hard-coded keys from code or localStorage.
* PATCH test via curl requires proper variable expansion:
* curl -X PATCH \
* -H "Content-Type: application/json" \
* -H "X-Admin-Key: ${ADMIN\_API\_KEY}" \
* -d '{"notes":"Handmatig getest via PATCH","state":"VERIFIED"}' \
* http://127.0.0.1:8000/api/v1/admin/locations/3416

**Epic Link:** Admin CMS & Human-in-the-Loop  
**Status:** ✅ Completed & Functionally Verified  
**Next Epic (TDA-19):** Data Export & Review Tools (Admin interface for dataset QA)