RMA Web Application - Comprehensive Documentation

CACHE SYSTEM OVERVIEW

Where Data is Stored

1. **SQLite Database File: `rma_cache.db`**
    - **Location**: Same directory as `rma_web.py` script
    - **Accessibility**:
        - ✅ **Single User**: Only accessible to the user running the Streamlit app
        - ❌ **NOT shared** between different workspace users
        - ❌ **NOT synced** across different machines
        - Each user/machine has their own separate cache file
2. **Streamlit Cache (`@st.cache_data`)**
    - **Location**: In-memory (RAM)
    - **Duration**: 3600 seconds (1 hour) or until cleared
    - **Accessibility**: Only for the current Streamlit session
    - **Resets**: When app restarts or cache is manually cleared

Database Schema

Table: `rmas`

```sql
CREATE TABLE rmas (
    rma_id TEXT PRIMARY KEY,          -- Unique RMA identifier
    store_url TEXT,                   -- Store shopify URL
    status TEXT,                      -- Pending/Approved/Received
    created_at TEXT,                  -- ISO timestamp
    json_data TEXT,                   -- Full RMA JSON payload
    last_fetched TEXT,                -- When we last updated this RMA
    courier_status TEXT,              -- Tracking status (if available)
    courier_last_checked TEXT         -- When courier was last checked
)
```

Table: `sync_log`

```sql
```

```
CREATE TABLE sync_log (
    scope TEXT PRIMARY KEY,          -- Format: "{store_url}_{status}"
    last_sync TEXT                   -- ISO timestamp of last sync
)
```

BUTTON FUNCTIONS

🔄 Sync All Data (Primary Red Button)

What it does:

1. Clears Streamlit cache
2. For each store (Diesel, Hurley, Jeep, Reebok, Superdry):
    - For each status (Pending, Approved, Received):
        - Calls ReturnGO API to get list of RMAs
        - For each RMA in the list:
            - Fetches full RMA details
            - Saves to SQLite database
3. Updates sync timestamps
4. Reloads the page to show new data

When to use:

- First time opening the app
- When you want the latest data from ReturnGO
- After making changes in ReturnGO portal

Expected behavior:

- Shows progress bar
- Displays "Syncing {Store} - {Status}... (X/Y RMAs)"
- Takes several minutes for all stores
- Shows toast notification when complete

🔄 Update All (Secondary Button)

What it does:

1. Clears Streamlit's in-memory cache (`@st.cache_data`)

2. Clears resource cache (`@st.cache_resource`)
3. Removes the table state from session
4. Reloads the page

When to use:

- Data looks stale but you don't want to re-sync from API
- UI isn't updating after changes
- Table is behaving oddly

What it DOESN'T do:

- Does NOT fetch new data from ReturnGO
- Does NOT update the database
- Only refreshes the display from existing database

🗑️ Reset Cache (Secondary Button)

What it does:

1. **Deletes ALL data** from SQLite database
2. Clears Streamlit caches
3. Resets filter state
4. Reloads the page

When to use:

- Something is corrupted in the database
- You want to start completely fresh
- Testing/debugging

⚠️ WARNING:

- This deletes all cached RMA data
- Next sync will take longer as it re-downloads everything

Store-Specific Status Buttons (e.g., "PENDING 5")

What it does:

1. Sets filter to specific store + status

2. Triggers a sync ONLY for that store + status combination
3. Updates the table to show only matching records

When to use:

- Quick refresh of a specific store's data
- Checking updates for one status
- Faster than full sync

API REQUEST SETUP

How API Calls Work

1. **Authentication**

```
python

headers = {
    "x-api-key": MY_API_KEY,      # Lowercase version
    "X-API-KEY": MY_API_KEY,      # Uppercase version (both for compatibility)
    "x-shop-name": store_url      # Which store to query
}
```

2. **Endpoint Structure**

```
python

# List RMAs with pagination
GET {api_url}/rmas?status={status}&pagesize=500&cursor={cursor}

# Get RMA details
GET {api_url}/rma/{rma_id}

# Update tracking
PUT {api_url}/shipment/{shipment_id}/tracking

# Add comment
POST {api_url}/rma/{rma_id}/comments
```

3. **Pagination Flow**

```
Page 1: GET /rmas?status=Pending&pagesize=500
        ↓
     Response contains: { "rmas": [...], "next_cursor": "abc123" }
        ↓
Page 2: GET /rmas?status=Pending&pagesize=500&cursor=abc123
        ↓
     Response contains: { "rmas": [...], "next_cursor": "def456" }
        ↓
Continue until next_cursor is null
```

4. **Rate Limiting**
   - Session configured with retry logic
   - Retries on 500, 502, 503, 504 errors
   - 0.1 second delay between detail fetches
   - No explicit rate limit handling (relies on API)

DATA FLOW

Sync Operation Flow

```
1. User clicks "Sync All Data"
      ↓
2. Clear Streamlit cache
      ↓
3. For each Store × Status combination:
      ↓
4. Call API: GET /rmas?status=X
      ↓
5. Get list of RMA IDs
      ↓
6. For each RMA ID:
      ↓
7. Call API: GET /rma/{id}
      ↓
8. Save to SQLite: INSERT/UPDATE rmas table
      ↓
9. Update sync_log table
      ↓
10. Reload page (st.rerun())
      ↓
11. Load data from SQLite
```

```
    ↓
12. Display in table
```

Display/Read Flow

```
1. Page loads
    ↓
2. Call get_all_active_from_db() [CACHED for 1 hour]
    ↓
3. Read from SQLite: SELECT * FROM rmas WHERE status IN (...)
    ↓
4. Parse JSON data for each RMA
    ↓
5. Process into table rows
    ↓
6. Apply filters (store, status, search)
    ↓
7. Display in st.dataframe()
```

CURRENT ISSUE: Data Not Appearing

Root Cause

The `get_all_active_from_db()` function has `@st.cache_data(ttl=3600)`, which means:

- After first load, it caches the result for 1 hour
- When sync completes and does `st.rerun()`, it loads the CACHED (old) data
- New data is in the database but not shown

Why "Update All" Doesn't Work

- It clears cache and reruns
- But if database is empty, there's nothing to show
- You need to SYNC first, THEN the data appears

Solution

Need to clear the cache BEFORE st.rerun() in the sync function, or remove the cache decorator from get_all_active_from_db().

MULTI-USER WORKSPACE CONSIDERATIONS

If Multiple Users Run This App:

1. **Each user has separate database file**
   - User A's `rma_cache.db` ≠ User B's `rma_cache.db`
   - No data sharing between users
2. **Each user makes their own API calls**
   - API key is shared (from secrets)
   - But each user's sync is independent
3. **For shared workspace:**
   - Consider using shared database (PostgreSQL like levis_web.py)
   - Or network-shared SQLite file (not recommended)
   - Or cloud storage solution

TROUBLESHOOTING

"Table is empty after sync"

→ Cache issue - click "Update All" after sync completes

"403 errors when syncing"

→ API key issue or store access permissions

"Database locked"

→ Multiple instances running or crash during write → Solution: Close all instances, delete `rma_cache.db`, restart

"Sync runs forever"

→ API pagination issue or network problem
→ Check logs for errors

"Old data showing"

→ Cache not cleared properly
→ Click "Reset Cache" to force refresh