04 - Conflict Resolution System

# 🧠 Conflict Prevention Strategy

- HQ is the source of truth for Item and Pricing data  
- Shops only send transactional data (Sales, Stock, Returns)  
- All records include a unique `reference\_number` field  
- Modified timestamp is used to compare updates  
- Shops cannot override HQ records directly

# 🔍 Conflict Detection Logic

Before inserting data from Shop:  
- Check if `reference\_number` already exists  
- If yes, compare content and `modified` timestamp  
- If mismatch is found, log conflict in `Sync Log` with status = "Conflict"

# 📋 Common Conflict Types

| Conflict Type | Risk Level | Handling |  
|---------------------------|------------|--------------------------------------|  
| Duplicate Invoice | Medium | Skip or show in Conflict Log |  
| Re-submitted Edited Entry | High | Show in Conflict UI for review |  
| Conflicting Stock Entries | Low | Use `reference\_number` + checks |  
| Master Data Mismatch | ❌ Blocked | Shops can't send Items/Prices |

# 🖥️ Conflict Resolver UI

A custom Page in ERPNext to resolve flagged conflicts:  
- Table with: Shop Name, Doctype, Reference, Status  
- Click to compare Shop vs HQ record (side-by-side JSON)  
- Options: Accept Shop version / Keep HQ version / Merge fields  
- Once resolved, conflict is marked as "Resolved" in Sync Log

# 🧪 Sample Conflict Detection Code

if frappe.db.exists("Sales Invoice", {"reference\_number": doc["reference\_number"]}):  
 existing = frappe.get\_doc("Sales Invoice", {"reference\_number": doc["reference\_number"]})  
 if existing.as\_dict() != doc:  
 # log as conflict  
 frappe.get\_doc({  
 "doctype": "Sync Log",  
 "shop\_name": doc.get("shop\_id"),  
 "document\_type": doc.get("doctype"),  
 "reference\_number": doc["reference\_number"],  
 "status": "Conflict",  
 "error\_message": "Data mismatch detected"  
 }).insert()  
 else:  
 pass # same record, skip