National University of Computer & Emerging Sciences
Karachi Campus

OptiChain

Smart Supply Synchronization System (4S)

Project Proposal
Database Systems
Section: 5J

Group Members:
23k-2001 Muzammil Siddiqui
23k-0532 Rayyan Ali
23k-0817 Hassaan Mustufa

Project Overview

Project Description:

Organizations with multiple branches often struggle with resource management because each branch operates independently with its own inventory. This can lead to duplication of procurement, wastage of surplus materials, and increased costs. The proposed system, Smart Supply Synchronization System (4S), will integrate decentralized branch databases with a centralized database to improve visibility of stock across all branches.

When a branch receives an order, the system will check the central database to determine whether the required items are available as surplus in another branch and whether transferring them would be cheaper than local procurement/production. This ensures cost-effective and efficient resource utilization across the organization.

Application Type:

The proposed application will be a web-application with a centralized dashboard for managers and branch-level access for requisition operations.

Functional Requirements

Key Features:

- Branch Inventory Management: Each branch maintains its own inventory and requisition records.
- Central Database Integration: Synchronization of all branch databases to a central hub for consolidated visibility.
- Smart Requisition Engine: Real-time check of surplus availability in other branches with cost comparison (inter-branch transfer vs. local procurement).

- Order Processing: Branches can create, update, and fulfill orders with support for all CRUD operations.
- Decision Support Reports: Generate insights on procurement patterns, wastage, surplus trends, and cost savings.

User Roles:

- Admin:
    - Manage system-wide settings.
    - Monitor requisition flow and access reports.
    - Oversee synchronization between central and branch databases.
- Branch Manager:
    - Manage local inventory (add, update, delete, view).
    - Initiate requisition requests and process orders.
    - Approve or reject inter-branch transfers.
- Staff/Employee:
    - Record incoming stock and outgoing requisitions.
    - View available inventory and request approvals from the manager.

Technical Specifications

Tech Stack:

- Frontend: HTML, CSS, Tailwind, Bootstrap, JavaScript
- Backend: Python, OAuth (Optional) for authentication
- Database: PostgreSQL (branch-level + central instances).
- APIs: Fast API to connect front-end and backend; SQLAlchamy to connect backend and Database
- Deployment (optional): Vercel (Front-end), Onrender (Back-end), Supabase (Database)

Development Plan

Timeline:

- **Week 1–2:** Requirement analysis, ERD design, schema normalization.
- **Week 3–4:** Database setup for branch and central instances, backend API development.
- **Week 5–6:** Frontend development (Interface dashboard + role-based access).
- **Week 7:** Integration of smart requisition engine (cost comparison + transfer checks).
- **Week 8:** Testing (unit + integration) and preparation of reports.
- **Week 9:** Final deployment and documentation.

Challenges & Risks:

- Data Synchronization: Ensuring accurate and real-time updates between branch and central databases.
  Mitigation: Use scheduled API sync and conflict-resolution rules.

- Cost Calculation Accuracy: Differences in logistics, time delays, and transfer feasibility may affect decision-making.
  Mitigation: Build configurable cost parameters per branch.

- Scalability: Handling multiple branches with heavy data load.
  Mitigation: Optimize queries, use indexing, and design scalable database schema.