Stock maintenance App

Over View:

Aim to create a stock maintenance app which will monitor your inventory and update the stock available and when the stock is running low it will automatically place order to vendors.

Features.

* User can sign in with his email ID
* Ability to talk with bill machine API to automatically update stock or else manually via front end
* User can initially set the in hand products and can set the threshold where a product count should reach in order for the product to be placed at order bucket
* Sales will be recorded throughout the day and the order is placed at EOD or at preferred time set by the user
* Before sending order a confirmation mail is sent to admin to confirm order placement. Admin can modify the order placement
* Dash Board Data =>
  + Products in hand
  + Order Bucket
  + Prediction model data
  + Top selling products
  + Profit or loss
  + Products making good profit , Less selling products
* The fall back component should show a tip about the application whenever it loads

Data base model

1.User model

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| user\_id | UUID / INT | Primary Key |
| name | VARCHAR |  |
| email | VARCHAR | Unique |
| password\_hash | VARCHAR |  |
| role | ENUM | admin, manager, etc. |
| created\_at | TIMESTAMP |  |

**2. Products**

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| product\_id | UUID / INT | Primary Key |
| name | VARCHAR |  |
| sku | VARCHAR | Unique identifier |
| description | TEXT |  |
| price | DECIMAL | Unit price |
| category\_id | FK | Links to categories |
| supplier\_id | FK | Links to suppliers |
| created\_at | TIMESTAMP |  |

**3. Categories**

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| category\_id | UUID / INT | Primary Key |
| name | VARCHAR | e.g., Electronics, Tools |

**4. Suppliers**

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| supplier\_id | UUID / INT | Primary Key |
| name | VARCHAR |  |
| contact\_info | TEXT |  |

**5. Inventory**

Tracks the current stock of each product **per location (optional)**.

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| inventory\_id | UUID / INT | Primary Key |
| product\_id | FK | Links to products |
| location\_id | FK (optional) | For multi-location support |
| quantity | INT | Current stock |
| min\_quantity | INT | Threshold for low stock |
| updated\_at | TIMESTAMP | Last updated |

**6. Stock Movements / Transactions**

Records **inbound, outbound, and manual adjustments**.

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| movement\_id | UUID / INT | Primary Key |
| product\_id | FK |  |
| quantity | INT | Can be positive or negative |
| type | ENUM | purchase, sale, adjustment, transfer |
| note | TEXT | e.g., “Damaged during delivery” |
| created\_by | FK (user\_id) |  |
| created\_at | TIMESTAMP |  |

**7. Purchase Orders**

For managing stock received from suppliers.

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| po\_id | UUID / INT | Primary Key |
| supplier\_id | FK |  |
| status | ENUM | pending, received, cancelled |
| created\_by | FK (user\_id) |  |
| created\_at | TIMESTAMP |  |

**8. PO\_Items (Line Items)**

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| po\_item\_id | UUID / INT | Primary Key |
| po\_id | FK | Links to purchase\_orders |
| product\_id | FK |  |
| quantity | INT |  |
| unit\_price | DECIMAL | Price at time of purchase |

**9. Locations (Optional, for multi-warehouse)**

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| location\_id | UUID / INT | Primary Key |
| name | VARCHAR | e.g., Chennai, Delhi |
| address | TEXT |  |

Architecture (Sample)

┌─────────────┐

│ Frontend │

└─────┬───────┘

↓

┌──────────────────┐

│ API Gateway │ ← JWT auth, routing

└───┬───────┬──────┘

↓ ↓

┌────────────┐ ┌─────────────┐

│ Product │ │ Inventory │

└────┬────────┘ └────┬────────┘

↓ ↓

┌────────────┐ ┌─────────────┐

│ Supplier │ │ Order │

└────────────┘ └─────────────┘

SERVICES

1.User service

Manages user details, Authentication Authorization, Password reset via Mail, Manages Audit log , session logs

2. Supplier Service

3. Notification Service

4.Inventory service

Endpoints:

1. User logs in and JWT is provided to the user by USER SERVICE (GET /API/users/login-user/) this will also verifies whether the user is logged in or not and for verification (GET /API/users/auth-user/). The JWT payload has user ID with that we can fill the home page data accordingly. (UPDATE /API/add-session)
   1. Authorization(Roles: Admin , Staff)
      1. Admin 🡪 Creator of the store capable of doing everything
      2. Staff 🡪 Capable of viewing the data , report inventory bugs to admin , can send request to admin to add products to order bucket.
2. **USER SERVICE 🡪 INVENTORY**

This API call is to get products data for this particular user(GET /API/inventory/get-products, GET /API/inventory/get-seller-hierarchy,