

AI Fullstack Software Development

Exercise

Mini Grocery Application

You are assigned to build a Mini Grocery (Sembako) Application using **Express.js**, **TypeScript**, and **Supabase (PostgreSQL)**, fully containerized using **Docker**.

This application must support two roles: **Visitor (Buyer)** and **Admin (Store Owner)**, each with specific access rights and features. The goal is to implement a clean backend architecture with proper validation, secure access control, and a containerized deployment setup.

Requirements

Tech

Build the backend using Express.js + TypeScript, store data in PostgreSQL (Supabase), and containerize the entire application with Docker (include Dockerfile and optionally docker-compose.yml).

Stack

&

Infrastructure:

User

Implement two roles:

Roles

&

Features:

- Visitor (**Buyer**): can view products, purchase items, and upload payment proof.
- Admin (**Store Owner**): can manage products with full CRUD operations and view store financial history.

Access Control & Validation:

- Enforce role-based access control (RBAC) — only Admins can modify data, while Visitors have read-only access.
- Validate all input data using **Zod**

Standardization

- All API responses must follow a consistent JSON format: { success: boolean, message: string, data?: any }
- Implement comprehensive error handling for invalid input, unauthorized access, and internal server errors.

Thank you

