Current Progress (What You've Built)

✅ Employee (Requester) Module:

- Employees can log in (right now no email verification restriction yet).
- They can browse the stationery catalog (pens, notebooks, gadgets, etc.).
- They can add to cart and submit requests for approval.
- My Orders/My Requests page shows order status (Pending, Approved, Delivered).

✅ Seller (Store Manager/Retailer) Module:

- Sellers can view all orders made by employees.
- They can update order status → "Processing → Shipped → Delivered."
- Sellers receive requester details (name, address, etc.).
- Sellers can add/manage products.
- Seller dashboard shows:
  - Total earnings
  - Total orders
  - Product list and management

✅ Admin Panel:

- Multi-vendor setup — Admin can approve/remove sellers/stores.
- Admin dashboard includes:
  - Total products
  - Total stores
  - Total revenue

🧩 Project Structure Summary

User Role	Capabilities	Dashboard Summary
Employee (Requester)	Browse catalog, add to cart, submit request, view order status	My Orders page
Seller / Store Manager	Manage products, view incoming requests, update delivery status	Total orders, earnings, product list
Admin	Approve/remove stores, view global statistics	Total stores, products, and revenue overview

---

⚙️ What's Missing / To-Do (based on your message)

Here are areas we can focus next:

1. Secure Login → restrict login to company email (e.g., @company.com).
2. Request Approval Flow → add intermediate approval before seller sees the request (if required).
3. Notification System → when status changes (Processing → Delivered), employee gets notified.

🧩 Full Tech Stack Analysis

1. Frontend

⚙️ *Framework:*

- Next.js
  - You're building your web app using Next.js (a React-based framework).
  - It allows both server-side rendering (SSR) and static site generation (SSG).
  - This helps improve SEO, performance, and faster page loading for your E-Shop.

---

2. Backend / Server

⚙️ *Runtime:*

- Node.js
  - Core backend runtime for your project.
  - Handles all server-side logic, API routes (e.g., /api/inngest, /api/stripe), and integrations.

🧰 *Framework (inferred):*

- Next.js API Routes (acts as backend)
  - You're not using a separate Express server; Next.js API routes handle backend endpoints.

---

3. Database

🗄️ *Neon Database*

- A PostgreSQL cloud database platform.
- Manages structured data like:
  - Employees / Users
  - Orders / Requests
  - Sellers / Stores
  - Products / Inventory
  - Admin details
- Neon is serverless and integrates easily with Prisma ORM or direct SQL.

---

4. Authentication & User Management

🔐 *Clerk*

- Provides authentication (sign-up, login, sessions, profile management).
- Can enforce company email domain restrictions later (@company.com).
- Allows role-based access: Employee, Seller, Admin.

---

5. Image Management

🖼️ *ImageKit*

- Used to store and optimize product images (pens, notebooks, gadgets).
- Offers CDN delivery for fast image loading and transformation (resizing, cropping).

---

6. Workflow Automation & Webhooks

⚡ Inngest

- Event-driven workflow automation tool.
- Triggers background jobs (e.g., sending emails or updating status when order placed).
- Webhooks link Clerk (auth), Stripe (payments), and app events (order updates).

---

7. AI Integration

🤖 *Google Gemini API*

- Enables AI-based features such as:
  - Product recommendations
  - Smart inventory management

---

9. Deployment

🌐 *Vercel*

- Hosting platform for your Next.js app.

---

10. Version Control

📜 *GitHub*

- Used to push and maintain your project code repository.
- Allows collaboration, issue tracking, and version management.

🧠 Summary: Architecture Overview

Layer	Tool / Tech	Purpose
Frontend	Next.js, React, Tailwind	UI for employees, sellers, admins
Backend	Node.js (Next.js API routes)	Server logic, APIs, webhooks
Database	Neon (PostgreSQL)	Stores user, product, and order data
Auth	Clerk	Secure login & role management
Images	ImageKit	Product image hosting & optimization
Automation	Inngest	Event-triggered workflows
AI	Gemini API	Smart recommendations, analytics
Hosting	Vercel	Deployment and live hosting
Version Control	GitHub	Code repository management

🚀 How It All Connects

- User (Employee) logs in via Clerk.
- Next.js Frontend displays products (from Neon DB).
- Employee places an order → API route stores it in DB.
- Seller Dashboard (Next.js page) shows order details (from DB).
- Seller updates status → Inngest workflow notifies employee.
- Admin oversees all stores & analytics (from aggregated data).
- Deployed & live on Vercel with ImageKit serving optimized images.