

Project Title: ERP Management System (MERN Stack)

Client Overview:

A mid-sized company needs a full-featured ERP web application to manage business operations like **Products, Customers, Suppliers, Sales Orders, Purchase Orders, Goods Receipt Notes (GRN), Invoices**, and **User Roles** (Admin, Sales, Purchase, Inventory, etc.).

Project Objective:

To build a full-stack ERP system using **React (frontend)**, **Node.js + Express (backend)**, and **MongoDB (database)**. The app will support secure user authentication (JWT), role-based access control, responsive UI, and CRUD operations for key business entities.

Frontend Requirements (React.js):

1. UI/UX Design:

- Responsive layout using **React + Material-UI / Bootstrap** •

Sidebar/Topbar navigation with icons

- Pages:

- Dashboard (metrics, charts)
- Product Management (Add/Edit/Delete/Search)
- Customer/Supplier Directory
- Sales & Purchase Orders
- GRN Form
- Invoice Generation
- User Management (Admin)
- Login/Register Pages

2. Routing (React Router v6):

- /dashboard, /products, /customers, /sales-orders, /grn, /invoice, /admin, etc.
- Protected routes using token-based guards

3. State Management:

- **Redux Toolkit** or **Context API** for managing:
 - Auth state
 - User roles
 - Global messages (toast alerts)
 - Loading states

4. Authentication & Authorization:

- Token-based login (JWT)
- Role-based route protection (Admin, Sales, Purchase, etc.)
- Logout and session expiration handling

5. UI Features:

- Toast alerts (react-toastify)
- Search, filter, pagination for tables
- Chart.js / Recharts for dashboards
- Form validation (Formik + Yup or React Hook Form)
- PDF invoice export (jsPDF)

Backend Requirements (Node.js + Express.js):

1. Core API Modules:

Module Endpoints

Users	POST /api/register
	POST /api/login

GET /api/users (Admin only)

Module Endpoints

Products GET, POST, PUT, DELETE /api/products

Customers / Suppliers GET, POST, PUT, DELETE /api/customers /
/api/suppliers

Sales Orders GET, POST, PUT /api/sales-orders **Purchase**

Orders GET, POST, PUT /api/purchase-orders

GRN (Goods Receipt

Note) POST /api/grn (linked to Purchase Order) **Invoices** POST

/api/invoices, GET all, GET by ID **2. Security:**

- JWT for auth
- Middleware for role-based access
- Passwords hashed with **bcrypt**

3. Database Design (MongoDB + Mongoose):

Collections:

- users (name, email, password, role)
- products (title, SKU, price, stock, reorderLevel)
- customers, suppliers (name, contact, address)
- salesOrders (customer, products, status, totalPrice) •
purchaseOrders (supplier, products, status)
- grns (linked to purchaseOrders)
- invoices (linked to salesOrders)

API Features:

- **Pagination & Filtering:** Add ?page=, ?search= params •

Global Error Handling: Central error middleware

• **Validation:** Mongoose + Joi/express-validator • **API**

Documentation: Swagger or Postman collection

Project Timeline:

Week Milestone

Week 1 Project setup, Auth (register/login), Role-based access

Week 2 Product & User CRUD APIs + React views Week 3 Sales

& Purchase Order flows, dashboard, GRN module Week 4

Invoicing, final integration, reports, PDF generation Week 5

Testing, error handling, deployment (Vercel + Render)

Deliverables:

- Fully functional **MERN-based ERP web app**
- REST API for all business entities
- Secure JWT login & role-based route protection •
- Responsive UI for desktop/tablet/mobile
- Swagger/Postman API docs
- Test coverage (Jest + React Testing Library)

Evaluation Criteria:

Area Criteria

Functionality All ERP modules working end-to-end **Code**

Quality Reusable components, modular Express routes

Security JWT, Bcrypt, role-based permissions

Area Criteria

UI/UX Professional, mobile-first layout

Performance Paginated APIs, optimized queries

Extras Optional features (invoices, charts, real-time features)

Tech Stack Summary:

Category Technology

Frontend React, React Router, Redux Toolkit, Axios, Material-UI

Backend Node.js, Express.js

Database MongoDB + Mongoose

Authentication JWT + bcrypt

UI Enhancements Chart.js, react-toastify, Formik/Yup

Testing Jest (Backend), React Testing Library (Frontend)

Deployment Vercel (Frontend), Render/Heroku (Backend)

Optional Enhancements:

- Stripe/PayPal Payment Gateway
- Inventory Alerts via Email
- Admin Analytics Dashboard (real-time charts)
- Export reports to Excel/PDF
- File upload (e.g., product images, invoice attachment)

This Full-Stack ERP Management System is designed to deliver a

comprehensive, scalable, and user-centric business solution built entirely on modern web technologies (React, Node.js, Express, and MongoDB). Tailored to streamline and automate core operational processes—including product management, customer records, sales and purchase orders, inventory tracking (GRN), and invoicing—this solution provides end-to-end visibility and control for growing businesses.

The system emphasizes secure access with robust user authentication and role-based authorization, ensuring that data access is tightly controlled and workflows are protected. With a responsive and intuitive user interface, it enhances productivity across departments, whether on desktop or mobile. This project not only reflects engineering best practices in full-stack development, but also aligns with real-world ERP demands—offering clients a high-performance, reliable, and future-ready business platform.