Product Requirements Document (PRD)

Project Title: Modular ERP Procurement System
Prepared By: Mayuresh Kumar Sinha
Date: October 24, 2025

1. Purpose

To design and implement a scalable, modular ERP Procurement system that streamlines buyer-supplier interactions, supports multi-tenant architecture, and ensures robust data segregation using **`customer_ID`** and **`company_ID`** as primary identifiers.

2. Conceptual Overview

- **Multi-Tenant Architecture:**
  - `customer_ID`: Uniquely identifies the tenant (client organization).
  - `company_ID`: Differentiates internal divisions or subsidiaries within a tenant.
- **Modular Design:**
  - Each module (e.g., RFQ, PO, Invoice, GRN) operates independently but shares core schema attributes.
- **Scalability & Maintainability:**
  - Designed for horizontal scaling and easy onboarding of new customers.
  - Clear separation of concerns across frontend, backend, and data layers.

3. Functional Requirements

3.1 User Roles

Role	Description
Buyer	Initiates procurement requests, manages POs
Supplier	Responds to RFQs, submits invoices
Admin	Manages users, configurations, and compliance
Approver	Reviews and approves procurement documents

3.2 Core Modules

- **Authentication & Access Control**
  - Secure login with role-based access
- **RFQ Management**

    - Create, send, and track RFQs
    - Supplier response tracking
- **Purchase Order (PO)**
    - PO creation, approval, and dispatch
    - PO status lifecycle
- **Invoice & GRN**
    - Invoice submission and reconciliation
    - Goods Receipt Note tracking
- **Audit & Compliance**
    - Activity logs, approval trails
    - Configurable retention policies

---

4. Data Architecture

4.1 Primary Entities

Entity	Key Attributes
Customer	`customer_ID`, name, industry, contact
Company	`company_ID`, `customer_ID`, location, domain
User	`user_ID`, role, `company_ID`, `customer_ID`
RFQ	`rfq_ID`, `company_ID`, `customer_ID`, items
PO	`po_ID`, `company_ID`, `customer_ID`, status
Invoice	`invoice_ID`, `po_ID`, `company_ID`, amount

4.2 Schema Notes

- All transactional tables will include `customer_ID` and `company_ID` for data isolation.
- Foreign key constraints will enforce referential integrity across modules.
- Indexing on `customer_ID`, `company_ID`, and `status` for performance optimization.

5. UI/UX Requirements

- **Login Page:**
    - Clean, branded, responsive design
    - Error handling and password recovery
- **Buyer Dashboard:**
    - Summary of RFQs, POs, invoices
    - Quick actions and alerts
- **Supplier Dashboard:**
    - RFQ responses, PO tracking, invoice submission
- **Admin Panel:**
    - User management, configuration, audit logs

6. Technical Stack

Layer	Technology
Frontend	React.js, html,css, bootstrap
Backend	Flask / FastAPI (Python)
Database	MySQL (validated schema)
Auth	JWT
Hosting	Azure /AWS

---

7. Non-Functional Requirements

- **Security:**
  - Role-based access, encrypted data at rest and in transit
- **Performance:**
  - Sub-second response time for dashboard queries
- **Scalability:**
  - Support for 100+ tenants with isolated data
- **Maintainability:**
  - Modular codebase with clear documentation

8. Success Metrics

- <95% uptime across modules
- <2s average response time
- <1 week onboarding time for new customers
- 100% audit trail coverage for all transactions