Node.js Assignment

Assignment Title:

"Multi-Tenant Invoice Management System (Mini ERP API)"



Objective

Build a multi-tenant REST API using Node.js, Express, and MongoDB/PostgreSQL that allows organizations to manage clients, create invoices, and track payments.

Problem Statement

You're building a mini-invoicing ERP system for multiple companies to manage their clients and invoices securely. Each company has its own users and data isolated.

X Core Requirements

1. Authentication & RBAC

- JWT-based auth
- Roles: Admin, Manager, Accountant
- Role-permission checks on sensitive actions
- Admin can invite users to their company (email simulation)

2. Multi-Tenancy Support

A user belongs to one organization

• Each organization manages: Clients Invoices Users Design should isolate data per organization (tenant). 3. Invoice Module • Create invoice: Add client (name, email, address) o Add invoice details: items[], amount, due date, status, PDF attachment • List all invoices (with filters: status, date range) • Send invoice email to client (mock/send using nodemailer) Update and delete invoice 4. File Upload (PDF) • Allow user to upload a PDF invoice (via Multer or similar) • Store file securely and return URL

5. Dashboard Summary API

- Return counts and totals:
 - Total invoices this month

- Total paid/unpaid
- Total clients
- o Filtered per organization

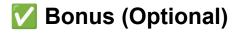
m Tech Requirements

Stack:

- Node.js + Express
- MongoDB (or PostgreSQL with Prisma)
- JWT Auth + RBAC
- Multer for file upload
- Nodemailer or email mocker
- Optional: Redis or Job Queue

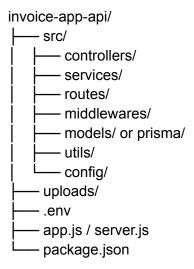
Dev Requirements:

- Use .env and dotenv for config
- Use services, controllers, middleware pattern
- Use async/await
- Secure coding best practices
- Proper validation (Joi or similar)



- Swagger or Postman collection
- Email reminders for unpaid invoices (simulate via cron or job queue)
- Use Redis for caching dashboard stats
- Implement Stripe payment flow for invoices

Suggested Folder Structure



Submission Instructions

- GitHub repo with:
 - Code and folder structure
 - Postman or Swagger documentation
 - .env.example
 - README with setup, features, and API usage
- Deployed link (if possible)

Evaluation Criteria

Area	Weight
Auth, RBAC, and Security	25%
Clean Code & API Design	20%
File Handling & Email Feature	15%
Multi-Tenant Architecture	20%
Validation, Testing, Docs	10%
Bonus (Docker, Redis, Cron)	10%