**Basic Division**

| **Role** | **Assigned Tasks** | **Tools** |
| --- | --- | --- |
| **1. Frontend Lead** | - Set up React (with TypeScript + Tailwind CSS)- Create base layout: header, sidebar, form container- Setup routing using React Router- Prepare reusable form/input components | React, Vite/CRA, Tailwind |
| **2. Backend Lead** | - Initialize Node.js + Express + TypeScript- Set up REST API boilerplate (routes, controllers, services)- Setup .env and config structure- Connect to MongoDB Atlas | Node.js, Express, TypeScript |
| **3. Database Lead** | - Create MongoDB Atlas project (on client account)- Define base schema for: Items, Vendors, StockRules, PurchaseOrders- Seed test data for dev use | MongoDB Atlas, Mongoose |
| **4. Auth & Env Setup** | - Setup login page (basic form)- Setup backend user auth routes- Setup JWT-based auth (or session if simpler)- Configure CORS and .env securely | JWT, Express, React |
| **5. DevOps & Hosting Lead** | - Confirm domain & hosting access- Setup frontend & backend deployment pipelines (e.g., Railway, Render, or self-host)- Setup GitHub repo with branches (dev, main) and CI/CD if needed | GitHub, client server |
| **6. Docs & Standards** | - Define API contract (Swagger/Postman)- Create shared Google Doc/Notion for:→ Naming conventions→ Folder structure→ Deployment guide | Notion, Postman |
| **7. QA & Tester** | - Setup Jest/React Testing Library boilerplate (optional at start)- Define test plan- Prepare dummy forms to test UI/backend integration later | Jest, Postman, Swagger |

**Stock rule**

| **Role** |  | **Tasks** |
| --- | --- | --- |
| Frontend |  | - Build UI form for Stock Rule- Implement computed fields: ADR, LOD, SS, ROL, MS- Use state to dynamically calculate fields |
| Backend |  | - Create StockRule schema- Create API endpoints: POST /stock-rule, GET /stock-rules, PUT /stock-rule/:id- Apply calculation logic in backend as fallback |
| DB |  | - Define how stock rules link to item and vendor- Ensure formula fields are stored properly for audit |
| QA |  | - Test if calculations are accurate (mock sales data for ADR)- Validate CRUD ops on stock rules |

**Generate Order**

| **Role** | **Tasks** |
| --- | --- |
| Frontend | - UI form to list reorder candidates (from stock rules)- Select vendor, items, quantities, generate PO |
| Backend | - API to fetch reorderable items (below ROL)- API to create and store Purchase Order |
| DB | - Create PurchaseOrder schema (linked to items and vendor)- Ensure support for partial deliveries later |
| QA | - Simulate PO creation with test stock rules- Check data integrity between PO and StockRule items |

**✅ Bonus Tips**

* **Use environment variables**: For DB URI, JWT secret, etc.
* **Have a dev config and prod config**: Set up .env.development and .env.production.
* **Use Postman to test APIs** before frontend integration.