Tech Stack

1. Frontend (User Interface)

- **Primary: React** + **Next.js** (TypeScript)
  Rationale: Fast developer productivity, SEO-capable pages for marketing/docs, strong ecosystem, SSR/SSG for performance and reliability.
- **Mobile (optional): React Native** or **Flutter**
  Rationale: Rapid cross-platform mobile apps for pharmacists/field staff; React Native shares code with React web.

2. Backend (Application Layer & API)

- **Primary: Node.js** with **NestJS** (TypeScript) or **Express** (TypeScript)
  Rationale: Scalable, familiar JS/TS stack end-to-end; NestJS adds structure and DI useful for enterprise features.
- **API style:** RESTful API with well-documented OpenAPI/Swagger; consider GraphQL for flexible client queries.

3. Database & Storage

- **Primary (Transactional): PostgreSQL**
  Rationale: ACID-compliant, strong relational modeling for inventory, batch tracking, prescriptions, and compliance reports.
- **Secondary (Search / Analytics): Elasticsearch** (or PostgreSQL full-text + analytics tools)
  Rationale: Fast search across products, prescriptions, and logs; useful for reporting and audit queries.
- **Object storage: AWS S3** (or equivalent) for backups, reports, audit logs, and attachments (invoice PDFs, scanned prescriptions).

4. Caching & Session Store

- **Redis**
  Rationale: Low-latency caching (product catalog, frequent queries), session store, rate-limiting, and distributed locks (safe stock updates).

5. Authentication & Authorization

- **Auth:** JWT + Refresh tokens, OAuth2 for third-party integrations

- **User roles & RBAC:** Implement role-based access control (pharmacist, manager, admin, auditor).
- **Optional IAM: Auth0**, **Okta**, or **AWS Cognito** for managed identity.

6. Notifications & Communication

- **SMS / OTP / Alerts: Twilio** (or local SMS gateway)
- **Email: SendGrid** / **SES**
- **In-app notifications:** WebSocket or server-sent events (see Realtime below).

7. Realtime & Synchronization

- **Realtime:** WebSockets via **Socket.IO** or managed services (e.g., **Pusher**)
  Use: Multi-device inventory sync, live alerts for low-stock/expiry, collaborative actions across counters.

8. DevOps, Containerization & Hosting

- **Containerization: Docker**
- **Orchestration / Hosting: AWS (ECS/EKS)** or **DigitalOcean App Platform / Kubernetes** depending on scale
- **Infrastructure as Code: Terraform** (or CloudFormation for AWS-specific)
- **Storage & DB hosting:** Managed **RDS (Postgres)** for availability and backups.

9. CI/CD & Release Management

- **CI/CD: GitHub Actions**, **GitLab CI**, or **CircleCI**
  Rationale: Automated testing, builds, container image publishing, and blue/green or canary deploys for minimal downtime.

10. Monitoring, Logging & Observability

- **Monitoring: Prometheus** + **Grafana** (or managed CloudWatch dashboards)
- **Logging / APM: ELK Stack** (Elasticsearch / Logstash / Kibana) or **Datadog / Sentry** for errors and performance tracing.
- **Audit Logging:** Immutable logs for compliance and regulatory audits.

11. Security & Compliance

- **Transport:** TLS everywhere (HTTPS)
- **Secrets Management: HashiCorp Vault** or cloud secret manager
- **Database encryption:** At-rest and in-transit encryption; regular backups with secure storage.

- **Vulnerability scanning:** Dependency scanners (Dependabot / Snyk) and container image scanning.
- **GDPR / HIPAA considerations:** Design data handling & retention policies if dealing with PII/PHI.

12. Testing & Quality Assurance

- **Unit & Integration tests:** Jest + Testing Library (frontend); Jest or Mocha for backend.
- **E2E tests:** Playwright or Cypress.
- **Static analysis / linting:** ESLint, Prettier, TypeScript.

13. Third-Party Integrations

- **Accounting / GST:** Integration with local GST invoicing APIs or exportable GST-compliant reports.
- **Payment / POS:** Optional POS / UPI / Card gateway integrations.
- **Regulatory / Inventory APIs:** Interfaces for government or supplier portals (if required).