Project Requirements

Elements Supply AI

Version 1.0
Prepared by: Upgraver Technologies (Pvt) Ltd
Client: Elements Technik Limited (trading as “Elements Supply AI")

Elements Supply AI is a centralized, audit ready workspace that unifies all engineering documents, digital twins, maintenance manuals, parts catalogs, RFQ / RFP, quotes, orders, supplier risk, and ESG reporting. So operations, procurement, engineering, and sustainability teams work from a single source of truth. The SaaS marketing website will be the entry point, while an authenticated Supplier Portal powers procurement workflows and analytics. The platform is mobile friendly, SEO optimised, and designed to scale with integrations. Supplier authentication via blockchain is explicitly deferred to a later phase and will be covered in a separate proposal.

1. Core Problem We Solve

a. Documents scattered across email, drives, ERPs, CMMS, and vendor portals
b. Low traceability from drawing > quote > PO > delivery > maintenance > ESG proof
c. Delays and cost overruns due to missing specs, wrong parts, or poor supplier fit
d. Fragmented maintenance plans and spare pa forecasting, creating stockouts or excess stock
e. Rising compliance expectations (safety, certifications, and ESG)

- Our solution: an AI and ML centralized platform where every asset/part, connecting all documents, decisions, and data streams end-to-end.

2. Modules at a Glance

a. Document Control Hub: All files in one place with versioning, approvals, watermarks, traceable downloads, and audit trails.
b. Digital Twin Workspace: A living 3D/2D record per asset/part (specs, CAD, manuals, PM plans, sensors, service history).
c. Maintenance Library: In depth repair & maintenance manuals, interactive parts breakdowns, torque charts, exploded views, and checklists.
d. Sourcing & RFQ / RFP: CAD aware RFQs, supplier matching, quote comparison, open bidding, and PO creation.

e. Inventory & Spares Forecasting: AI/ML to set reorder points, predict lead time risk, and suggest “order by" dates.
f. Supplier Risk & ESG: Risk scoring, certificate tracking, corrective actions, and ready to share ESG / Scope-3 support docs.
g. ERP / CMMS Integration: Two-way sync with your systems; ingest sensor/condition data to drive PdM and spare pare kits.
h. Analytics & Copilot: Natural language actions (“Draft a PO", “Compare quotes", “Show ESG gaps") with explainable outputs.

3. Digital Twin

3.1. Key capabilities

3.1.1. Twin Packs (templates) by machine category (e.g., pumps, conveyors, gearboxes, hydraulic power units).
3.1.2. 3D/2D viewer (GLB/STEP/PDF drawings) with tagged parts. Click to open manuals, specs, or replacement part numbers.
3.1.3. RFQ/RFP from Twin: Auto attach current rev, tolerances, finishes, and recent QA notes.
3.1.4. Supplier Proposals to Twin: Quotes, process notes, QA certs, and warranty flow back to the twin.
3.1.5. Maintenance from Twin: PM schedules, work orders, spares consumed, failure events, and root cause analysis.
3.1.6. Exports: One click export of a Twin bundle (PDF datasheet + JSON + CAD) for audits or sharing with external partners.

3.2. Business impact

3.2.1. Eliminates version confusion and rework
3.2.2. Shortens quote cycles. Reduces quality escapes
3.2.3. Creates audit grade traceability for customer, ISO, and ESG demands

4. Maintenance Manuals & Asset Care

4.1. Interactive manuals with step by step procedures, safety notes, tools & torque specs, and embedded parts diagrams.
4.2. Exploded parts views linked to part numbers and inventory availability. “Add to RFQ/PO" in one click.
4.3. Maintenance plans (daily/weekly/monthly/overhaul) with checklists and sign-off trails.
4.4. Downtime & cost capture (labor, parts, external services) and “cost per asset" dashboards.

5. Inventory Forecasting: AI + ML

5.1. Data we use

5.1.1. Demand history (orders/usage), lead times, supplier performance
5.1.2. Twin attributes (material, finish, tolerance, criticality), planned shutdowns
5.1.3. Maintenance signals (PdM trends, upcoming overhauls)

5.2. How it works

5.2.1. For steady items: seasonal/time series models estimate demand.
5.2.2. For spare parts that are ordered only once in a while, we estimate how often orders happen and how big they are, then combine those to give a realistic forecast.
5.2.3. Feature rich ML adds supplier reliability, lead-time variance, and PdM signals to predict realistic safety stock and reorder timing.
5.2.4. The Copilot answers what ifs in plain English: “If we target 97% service level and switch to Supplier B, what’s the new safety stock and cash impact?"

5.3. Outputs

5.3.1. Reorder point, safety stock, suggested order quantity, order by date, and supplier choice recommendations plus 1-click RFQ/PO.

6. Supplier Risk & ESG

6.1. Risk scoring blends on time delivery, defect/return rates, price/lead-time volatility, and service responsiveness.
6.2. Explainable badges (“High risk due to 3 late POs + missing ISO9001 + rising defect trend").
6.3. ESG workspace collects supplier attestations, certificates, and activity logs. Exports a Scope-3 support pack (evidence + calculations) for customers and auditors. When a client asks “prove the suppliers are compliant and show the carbon impact," the platform can produce a single, audit-ready bundle in seconds.

7. Stakeholders & Personas

- Buyer Admin / Requester raises RFQs/RFPs, evaluates quotes, issues POs, tracks orders, uploads/consumes documents, marks payments.
- Supplier Admin / Estimator maintains capabilities, receives RFQs, submits quotes with lead time and attachments. (Open bidding supported.)
- Platform Admin does onboarding/verification, marketplace moderation, dispute support.

- Finance (Buyer/Supplier) does invoice generation (auto/manual), payment status, reports generation.

8. Functional Requirements

8.1. FR-1 Supplier Discovery & Profiles

a. Filters: capability, material, tolerance, finish, location, lead time range, rating threshold, certifications, industries.
b. Sort: match score, lead time, rating, distance, price band.
c. Profiles include capabilities, indicative lead time, rating, certificates, OTIF badge, MOQ, industries; sensitive
d. edits require verification.
e. AI Match ranks suppliers using capability fit + performance (lead time reliability, OTIF, win rate); shows rationale badges.

8.2. FR-2 RFQ / RFP Creation

a. RFQ (fields): part specs, quantity, tolerance/finish, manufacturing method, delivery timeline, location, notes, CAD upload; option to make Open RFQ (Bidding).
b. RFP (fields): problem/objectives, scope, timeline, evaluation criteria, proposal format.
c. AI Assist: detect part type/materials, prefill fields, validate tolerances.
d. If CAD parsing is uncertain, suggestions are flagged and fields remain user editable.

Acceptance:

Given an uploaded STEP file, the form auto-detects “Bracket L-Shaped" and “Aluminum," and requests quantity and delivery time before submission.

8.3. FR-3 Quote Intake (Supplier Side)

a. Direct RFQs and Open RFQs appear in a unified inbox with deadlines/status.
b. Quote form captures unit price, lead time, notes, and attachments (QA certs, datasheets, warranty terms, compliance).
c. AI Quote Generation suggests pricing/lead time from history; suppliers can adjust.

Acceptance:

For RFQ #1049, a supplier submits “USD 280, 5 days" and uploads Specs.pdf; buyer sees it alongside alternatives with timestamps.

8.4. FR-4 Quote Comparison & Award

a. Buyer dashboard to compare quotes by price, lead time, rating, and notes; award and convert to next steps.
b. Open RFQ clearly marks the bidding window and “makes your offer stand out".

8.5. FR-5 Purchase Orders (POs)

a. Creation: both parties may generate POs; auto-creation on award + manual override.
b. Approvals: buyer review/notifications; some cases require Elements team approval.
c. Branding: PDF with Elements Supply AI) brand + buyer brand (“powered by Elements Supply AI").

8.6. FR-6 Invoicing & Payments (Tracking only)

a. Invoice generation from PO (auto) with manual edits supported; supplier branding.
b. Payment status tracking (Paid/Pending/Overdue), records per part/order. No online payment processing.

8.7. FR-7 Orders & Logistics

a. Order states: Pending, Confirmed, In-Production, Shipped/In-Transit, Delivered, Cancelled; quantities, totals, timestamps.
b. Buyer and supplier “Orders" views to track status history.

8.8. FR-8 Document & CAD Management

a. Accepted file types: STEP/IGES/DWG/PDF and others; secure sharing with selected parties.
b. Per order/project, store technical, commercial, QA, logistics, financial, and communication documents as listed by the client.

8.9. FR-9 Reporting & Dashboards

a. Dashboard, advance analytics, supplier performance scorecards/risk alerts (by plan).
b. Exportable reports (CSV/PDF): quote cycle times, win/loss, OTIF, invoice status, inventory forecast snapshots.

8.10. FR-10 Integrations

a. One ERP/MRP integration available at certain plan tiers; generic import/export now, APIs extensible.

8.11. FR-11 Roles & Permissions

a. Role matrix (examples): Buyer Admin/Requester, Supplier Admin/Estimator, Finance, Platform Admin with least-privilege permissions to RFQs, quotes, POs, invoices, orders, and documents.

8.12. FR-12 User & Company Lifecycle Management

a. Company Registration:

   i. A new buyer or supplier registers a company account through a dedicated sign-up wizard.
   ii. Required fields: company name, registration number, tax ID, country, email domain, and primary contact.
   iii. Registration triggers an **admin review/approval workflow**.
   iv. Upon approval, the company becomes an active tenant within the multi-tenant database.

b. User Invitations & Roles:

   i. Company Admins can invite users via email and assign roles (Buyer Admin, Buyer Requester, Supplier Admin, Estimator, Finance).
   ii. Each invitation creates a pending user record with token expiration (48 hours).
   iii. Users can be deactivated, suspended, or reassigned without data loss.

c. Company Switching / Multi-Org Access:

   i. Certain users (consultants or service providers) may belong to multiple companies.
   ii. The UI provides a **"Switch Organization"** menu; permissions are scoped by the active tenant.

d. Domain Verification:

   i. Optional email-domain validation ensures all users under a company share a verified domain (e.g., @*elementstechnik.com*).

e. User Audit Trail:

   i. All join, role change, and deactivation events are logged for compliance.

8.13. FR-13 Billing and Subscription Plans (Stripe Integration)

a. **Purpose:** Implement recurring billing, entitlements, and upgrade/downgrade workflows using **Laravel Cashier (Stripe)**.
b. Plan Tiers

Tier	Price	Intended Audience	Key Feature Limits
Starter	USD 2,400 / month	Single-site engineering firms or small OEMs	Up to 5 users, 10 RFQs per month, 3 GB storage, no ERP integration, basic analytics only.
Growth	USD 4,800 / month	Mid-size manufacturers	Up to 25 users, 100 RFQs per month, 25 GB storage, 1 ERP/CMMS integration, full reporting suite, custom document templates.
Enterprise	Custom quote	Large enterprises / multi-plant operations	Unlimited users & RFQs, unlimited storage (soft limits apply), multi-tenant groups, SSO (SAML/AD), priority support, dedicated instance option.

8.14. FR-14 Feature Gating & Entitlements

a. Each plan defines numeric limits (e.g., rfqs_per_month, users_max, storage_gb) stored in the plans table.
b. Middleware EnsureSubscribed validates active subscription before allowing:
    i. RFQ creation, Quote submission, Document upload, or Integration access.
c. When limits exceed thresholds, users see an upgrade prompt and can redirect to /pricing → Stripe Checkout.
d. Stripe webhooks manage events invoice.payment_succeeded, invoice.payment_failed, and customer.subscription.updated.
e. Grace period = 7 days post failed payment → system enters “read-only” mode until payment resolves.

8.15. FR-15 Billing Portal & Invoices

a. Each company may access /billing/portal (Stripe Customer Portal) to view history, update payment methods, download Stripe-generated invoices, or cancel.
b. Subscription status displayed on the Account → Plan page (active, trialing, past_due, cancelled).
c. Admins can upgrade/downgrade without losing data; features lock/unlock automatically.

8.16. FR-16 Approvals Matrix and Delegations

a. **Purpose:** Introduce configurable, hierarchical approvals for key transactions.
    i. Approval Levels: Up to five levels per company (defined by threshold rules or document type).

ii. Supported Actions: RFQ publishing, PO creation, Change Orders, Invoice approvals, NCR closures.
iii. Delegation: Approvers can assign temporary delegate with start/end date ("on leave" mode).
iv. Tracking: Each approval/rejection records approver, timestamp, comments, and sequence.

8.17. FR-17 Dispute / Return Management

a. Return Requests (RMA): Buyer can raise an RMA against a Delivered order with reason, photos, and proposed resolution.
b. Credit Notes: Finance can issue credit against invoice with link to PO and GRN records.
c. Workflow: Raised → Under Review → Approved/Rejected → Closed.
d. Audit: All RMAs and credits are logged and reflected in supplier performance metrics.

8.18. FR-18 Global Search and Filtering

a. Universal search bar spanning Suppliers, Parts, RFQs, POs, Invoices, and Documents.
b. Advanced filters by status, date range, entity type, and tag.
c. **Saved Searches** for frequently used queries.

8.19. FR-19 Notifications & Reminders

a. Support recurring email and in-app notifications for:
    i. Upcoming RFQ deadlines
    ii. Late deliveries / shipments
    iii. Certificate expiry reminders
    iv. Overdue invoices
b. Daily and weekly digest modes configurable per user.

8.20. FR-20 Data Export & Backup

a. Admins can export complete company data (entities + documents metadata) as ZIP/JSON bundle.
b. Automatic nightly backups to secure storage with 30-day retention.
c. Export includes audit logs and plan metadata for compliance portability.

8.21. FR-21 Localization and Units

a. Multi-language support (JSON translation files).
b. Regional date formats and unit preferences (metric / imperial) stored per company.
c. Currency formatting already covered under FR-9.6.

8.22. FR-22 Audit Trail User Interface

a. Dedicated Audit Log screen (filter by entity, user, action, date).
b. Export to CSV or PDF.
c. “View Changes” modal showing before/after JSON diff for critical fields.

8.23. FR-23 Analytics & Performance KPIs

a. Initial dashboards should include:

   i. RFQ Cycle Time (Average days from creation to award)
   ii. Supplier Response Rate (% RFQs quoted)
   iii. On-Time Delivery Rate
   iv. Quality Score (accepted vs rejected qty)
   v. Spend by Supplier/Category per month
   vi. Forecast Accuracy (MAPE vs actual usage)

b. Each metric is filterable by date range and company unit.

9. Platform Admin Console (System Admin Dashboard)

Purpose: Internal console for platform operators to manage tenants, subscriptions, usage, entitlements, and audits.

9.1. Scope

9.1.1. Tenants (Companies): approve/suspend, view profile, owner, users, storage, regions.
9.1.2. Subscriptions (Stripe): plan, status, next invoice, payment history; upgrade/downgrade; override limits.
9.1.3. Usage & Entitlements: RFQs/month, storage used, active users, integrations; force refresh.
9.1.4. Audit Overview: cross-tenant audit feed with filtering and CSV export.
9.1.5. Optional: Impersonate tenant admin (read-only by default; write only for break-glass role).

9.2. Data Model (new/updated)

9.2.1. platform_admins (id, user_id, role enum: super|support, enabled_bool)
9.2.2. companies add: status enum(active|suspended|pending), region, storage_used_mb, rfqs_monthly_used
9.2.3. company_plan_overrides (company_id, key, value, reason, created_by)
9.2.4. platform_announcements (title, body_md, visible_from, visible_to)

9.3. UI

9.3.1. Companies table → detail drawer (owner, plan, usage, actions).

9.3.2. Subscriptions tab → Stripe invoice list + “Open Billing Portal”.
9.3.3. Usage tab → cards: RFQs this month, storage, active users, API calls.
9.3.4. Audit tab → filterable log; Export CSV.
9.3.5. Actions: Suspend/Activate, Plan Override, Impersonate, Send Announcement.

9.4. Permissions

9.4.1. Role platform:super full access; platform:support read-only + announce.

9.5. Jobs

9.5.1. Nightly job to recompute usage (ComputeTenantUsageJob) and store snapshots.

9.6. Acceptance Tests

9.6.1. AT-20.1: Super admin can Suspend/Activate a company; status reflected in login & APIs.
9.6.2. AT-20.2: Plan override changes enforced immediately (e.g., rfqs_per_month = 200).
9.6.3. AT-20.3: Audit grid filters by company, entity, date and exports CSV accurately.
9.6.4. AT-20.4: Impersonation issues a one-time token and logs start/stop events.
9.6.5. AT-20.5: Usage cards match snapshot table totals within ±1% for the day.

10. Must-add to avoid blockers

10.1. Supplier Onboarding & KYC Verification

10.1.1. Each supplier must maintain a **Company Profile** containing legal name, registration number, tax ID, address, country, phone, email, and website.
10.1.2. Suppliers upload compliance documents (e.g., ISO certificates, business registration, insurance certificates).
10.1.3. Each document has an **expiry date** and triggers an **automatic reminder** 180 days before expiry.
10.1.4. Admins can **approve or reject** suppliers after reviewing KYC documents; only *approved* suppliers can participate in RFQs or submit quotes.
10.1.5. A supplier’s approval status and document validity appear in search results and quote comparisons.

10.2. RFQ Clarifications (Q&A) and Amendments

10.2.1. Every RFQ has a **Clarifications thread** visible to all invited suppliers (or public if “Open Bidding”).

10.2.2. Suppliers can post **questions**; buyers can **reply** or post **RFQ amendments** (e.g., specification changes or new attachments).
10.2.3. All clarifications and amendments are **timestamped** and **emailed** to participants.
10.2.4. Amendments increment an **RFQ version number** and log the reason for traceability.

10.3. Quote Revisions and Withdrawals

10.3.1. Suppliers may **edit or re-submit** a quote any number of times before the RFQ deadline.
10.3.2. Each revision creates a new **revision number** (v1, v2, v3 …) with previous revisions archived but readable by the buyer.
10.3.3. Suppliers can **withdraw** quotes before the deadline. The buyer sees the withdrawal timestamp and reason.

10.4. Line-Item Awards and Split Awards

10.4.1. Buyers can **award individual items** within an RFQ to different suppliers.
10.4.2. Each awarded line creates a separate **Purchase Order (PO)** referencing the parent RFQ.
10.4.3. Non-awarded suppliers automatically receive **"Regret Notice"** emails identifying which items they lost.

10.5. RFQ Deadline and Lifecycle Management

10.5.1. Each RFQ includes: **publish date, submission deadline, bid-opening date, close date**.
10.5.2. Buyers can **extend deadlines** with reason capture; system logs all changes.
10.5.3. Once the deadline passes, submissions lock automatically; no new or revised quotes accepted.
10.5.4. A **lifecycle timeline** displays RFQ creation → clarifications → closing → awards → POs.

10.6. Multi-Currency and Tax Support

10.6.1. RFQs, Quotes, POs, and Invoices store **currency code (ISO 4217)** and **exchange rate note**.
10.6.2. Unit price and total amount fields are currency-specific.
10.6.3. Tax rate (%) and tax amount are captured per document; totals show *net, tax, gross*.
10.6.4. **Incoterms** (EXW, FOB, CIF, DDP etc.) and **Delivery Terms** fields appear in all commercial documents.

10.7. PO Acknowledgement and Change Orders

10.7.1. When a PO is issued, the supplier must **acknowledge** (Accept / Reject / Propose Change).
10.7.2. Proposed changes (e.g., revised price or lead time) create a **Change Order** record linked to the original PO with status history.
10.7.3. Once both parties approve the change, a new **PO Revision Number** is generated (PO-001-R1).

10.8. Goods Receipt & Quality Inspection

10.8.1. Buyers can record **Goods Receipt Notes (GRN)** per PO line (received qty, date, inspector).
10.8.2. Quality Inspection fields: *Accepted Qty, Rejected Qty, Defect Notes, Attachments*.
10.8.3. Rejected items flag the PO line as **"NCR Raised"** (Non-Conformance Report) for corrective action.
10.8.4. Partial receipts accumulate until the total ordered qty is fully received.

10.9. Invoice Reconciliation (Three-Way Match)

10.9.1. The system compares **PO amount, GRN received qty, and Invoice amount**.
10.9.2. Status values:
10.9.2.1. ✅ Matched (within tolerance)
10.9.2.2. ⚠️ Qty Mismatch
10.9.2.3. ⚠️ Price Mismatch
10.9.2.4. ❌ Unmatched
10.9.3. Discrepancies trigger a notification to Buyer Finance role for manual review.

10.10. Audit Trail and Notifications

10.10.1. All critical actions (create, update, delete, award, acknowledge, pay) generate **audit entries**: user, role, timestamp, entity, action, before/after values.
10.10.2. Emails and in-app notifications are dispatched for all workflow events (new RFQ, new quote, amendment, award, PO issue, invoice received, delivery update).
10.10.3. Admins can export the full audit log as CSV.

10.11. Custom Document Templates & Branding

10.11.1. Each company can design **custom templates** for RFQ, Quote, PO, Invoice, and Delivery Note PDFs.
10.11.2. Template editor allows uploading **company logo**, choosing **primary color**, adding **header/footer text**, and customizing **signature blocks**.
10.11.3. The template configuration is stored per company and applied automatically when documents are generated.

10.11.4. Buyers can **preview and download** sample documents to verify branding.
10.11.5. Admin can enforce a **standard layout** if required for multi-tenant consistency.

10.12. Bulk Data Import via CSV

10.12.1. Users can **import large data sets** (e.g., Suppliers, Materials, RFQ Items, Products, Inventory, Price Lists) using structured CSV files.
10.12.2. Each import screen provides a downloadable **template CSV** with column headers, sample values, and validation rules.
10.12.3. The import process performs **server-side validation** and generates a **row-by-row error report** (downloadable CSV or on-screen grid).
10.12.4. Successful rows commit automatically; failed rows can be corrected and re-uploaded.
10.12.5. All imports are logged with file name, timestamp, importer user, row count, success / fail count.

10.13. Notifications and Access Control (Extension)

10.13.1. Notification preferences (Email / In-App / Both) configurable per user and per event type.
10.13.2. Role-based access controls ensure:
10.13.2.1. **Buyers** see their own company's RFQs, Quotes, POs, Invoices.
10.13.2.2. **Suppliers** see only invited or open RFQs and their own quotes/POs.
10.13.2.3. **Finance roles** access Invoice and Payment modules only.
10.13.2.4. **Admins** view all records.

10.14. Performance and Data Integrity

10.14.1. All critical transactions use **DB transactions** to prevent partial writes.
10.14.2. Each entity includes **soft-delete** and **versioning** (revision number, revision reason).
10.14.3. File uploads are virus-scanned (stub hook ok) and stored with access-controlled URLs.
10.14.4. Audit and import logs have retention period configurable by Admin.

11. Development Guard Rails & General Functional Standards

This section defines the **baseline development and interaction rules** that every feature, module, and component of Elements Supply AI must follow.
These standards ensure that the system is launch-ready, maintainable, and consistent across all modules, while allowing fast iteration in early versions.

11.1. Code Architecture Standards

11.1.1. Framework Stack:

- 11.1.1.1. Backend: Laravel 12 (MVC pattern, REST API endpoints, Livewire 3 optional for interactive forms)
- 11.1.1.2. Frontend: React Starter Kit with Tailwind CSS.
- 11.1.1.3. Database: MySQL.

11.1.2. Componentization:

- 11.1.2.1. Every UI element must be created as a reusable React component (e.g., InputField, SelectBox, CheckboxGroup, Table, Toast, Modal, FormCard).
- 11.1.2.2. Components should be generic, configurable, and stored in /resources/js/components/ui/.
- 11.1.2.3. Avoid inline or one-off code duplication.

11.1.3. Folder Structure:

- 11.1.3.1. /app/Models → Eloquent models
- 11.1.3.2. /app/Http/Controllers → API controllers
- 11.1.3.3. /resources/js/pages → React pages per module
- 11.1.3.4. /resources/js/components → Shared components
- 11.1.3.5. /resources/js/hooks → Reusable hooks for API calls, validation, toasts, etc.
- 11.1.3.6. /resources/js/services → Axios or Fetch wrappers for HTTP requests

11.1.4. Coding Practices:

- 11.1.4.1. Follow PSR-12 (Laravel) and Airbnb (React/JSX) style guides.
- 11.1.4.2. No hard-coded strings; use config constants or translation files.
- 11.1.4.3. All functions and props documented via JSDoc/PHPDoc.
- 11.1.4.4. Use environment variables for keys, endpoints, and secrets.

11.2. Form Standards

11.2.1. Validation:

- 11.2.1.1. Every form must implement client-side and server-side validation.
- 11.2.1.2. Required fields marked with “*”.
- 11.2.1.3. Invalid inputs show inline error messages directly beneath the field.
- 11.2.1.4. On submission, forms must re-check validation before sending the request.
- 11.2.1.5. Use shared validation hook (useFormValidation) for consistency.

11.2.2. Error Display Section:

11.2.2.1. Each form must include an error summary area at the top or bottom listing validation errors (especially useful for long forms).
11.2.2.2. Validation errors must scroll into view automatically.

11.2.3. Loading & Submit States:

11.2.3.1. All submit buttons disable during network calls and show a spinner.
11.2.3.2. Prevent double submissions or race conditions.

11.2.4. Toast & Feedback:

11.2.4.1. After every successful action, show a success toast.
11.2.4.2. For any failed action, show an error toast with the reason (from API response or fallback message).
11.2.4.3. Toasts should be globally available, positioned top-right, dismissible, and automatically disappear after 4–5 seconds.

11.2.5. Accessibility:

11.2.5.1. Forms must be navigable via keyboard.
11.2.5.2. Every field labeled with for and id attributes.
11.2.5.3. Focus returns to the first invalid field on failed submission.

11.3. API & State Handling Standards

11.3.1. HTTP Requests:

11.3.1.1. Use a unified HTTP service (/resources/js/services/api.js) wrapping Axios/Fetch for consistent headers and error handling.
11.3.1.2. Include global interceptors for 401 (unauthorized), 404 (not found), and 500 (server) errors.
11.3.1.3. Implement exponential backoff on retry for transient errors.

11.3.2. API Feedback:

11.3.2.1. All API calls must return standardized JSON with status, message, and data.
11.3.2.2. Frontend must interpret these values for toast & UI states.
11.3.2.3. Log errors to the console only in dev mode.

11.3.3. State Management:

11.3.3.1. Use React Context or Zustand for global state (auth, user, company, notifications).
11.3.3.2. Keep module state isolated; no cross-module state pollution.

11.3.4. Response Caching:

11.3.4.1. Use Laravel cache for static lists (materials, incoterms, currencies).
11.3.4.2. In frontend, use SWR or query caching for recent data where appropriate.

11.4. Notification & Communication Standards

11.4.1. Push Notifications:

11.4.1.1. Implement in-app notification center (bell icon) using WebSocket / Pusher / Laravel Echo.
11.4.1.2. Notifications must appear in real time without page reload.
11.4.1.3. Include type (info, success, warning, error), title, and description.

11.4.2. Email Notifications:

11.4.2.1. Each event triggers an email based on templates (RFQ Created, Quote Submitted, PO Awarded, Invoice Paid).
11.4.2.2. Emails must be responsive, branded, and stored as Blade templates.

11.4.3. Smooth Experience:

11.4.3.1. No duplicate alerts; mark read/unread with proper UX.
11.4.3.2. Push and email notifications must be consistent and synchronized (once per event).

11.5. UI Behavior Guard Rails

11.5.1. Every interactive element (button, icon, checkbox, link) must provide visual feedback (hover/focus/active).
11.5.2. All destructive actions (delete, cancel, reject) require confirmation modals.
11.5.3. Empty states must display meaningful illustration + CTA (e.g., “No RFQs yet – Create your first RFQ”).
11.5.4. Responsive Design: All screens must work seamlessly across ≥ 1280 px desktop, ≥ 768 px tablet, ≥ 360 px mobile.
11.5.5. Performance: Each page load < 2 s on average Wi-Fi; lazy-load large data tables.
11.5.6. Tables: Sortable, paginated, with sticky headers and column filters.
11.5.7. Modals: ESC and outside-click must close them gracefully.
11.5.8. Tooltips: Use for truncations and help texts; must never block primary actions.
11.5.9. Timeline and Status Bars: All lifecycle changes (RFQ → Quote → PO → Delivery) visualized via consistent timeline component.

11.6. Reusability & DRY Principle

11.6.1. Shared Component Library: Maintain /resources/js/components/ui/ with all base UI elements (forms, modals, tables, cards).
11.6.2. Utility Functions: Common date/time, currency, and formatters stored in /utils/helpers.js.
11.6.3. Do Not Repeat: No duplicate code across modules; reuse and extend existing components.
11.6.4. Styling: Use Tailwind classes or component-level SCSS modules only – no inline CSS.

11.7. Error Handling & Fallbacks

11.7.1. Global Error Boundary: React error boundary to catch UI crashes and display user-friendly message.
11.7.2. Network Fallback: If offline, show “Connection lost” banner with auto-retry.
11.7.3. Graceful Fallbacks: If AI Assist or 3D Viewer is down, show “Feature temporarily unavailable.”
11.7.4. Empty API Response: Show placeholder cards instead of blank space.

11.8. Development Priorities & Simplification

11.8.1. Launch First, Harden Later: The primary goal is a working, usable MVP — not maximum security.
11.8.2. Security De-scoping (for now): Use standard Laravel auth, CSRF, and HTTPS only. Do not over-complicate with JWT rotation or zero-trust flows yet.
11.8.3. Focus on User Flow: Each module must work from start to finish without dead ends or broken links.
11.8.4. Performance Over Perfection: Prefer fast, readable code to over-engineered solutions.
11.8.5. Readable Code: Variable and function names must be self-explanatory. No abbreviations or obscure logic.
11.8.6. Testing Light: Basic feature tests and manual UAT for launch; full unit tests in later phase.

11.9. Deployment & Environment Consistency

11.9.1. Environment Variables: All configurations (API keys, URLs, Stripe IDs) in .env.
11.9.2. Build Consistency: CI/CD pipeline (auto-deploy to staging and production with branch triggers).
11.9.3. Logging: Centralized logs via Laravel Log + Monolog (file & Slack notifications for errors).
11.9.4. Monitoring: Basic uptime checks and error alerts enabled before launch.
11.9.5. Backup Automation: Nightly DB and file backups to secure storage.

11.10. General User Experience Assurance

11.10.1. Smooth Navigation: No page reloads for core actions – use AJAX or React state updates.
11.10.2. Progress Indicators: Show spinners or loading bars during long operations.
11.10.3. Undo Options: Provide undo for non-destructive actions (e.g., archive, mark read).
11.10.4. Consistent Terminology: Use standardized labels across all modules (“RFQ,” “PO,” “Invoice,” “Order”).
11.10.5. User Context: Breadcrumbs and page titles must always reflect the current location and entity ID.
11.10.6. Accessibility: All interactive elements must be reachable by keyboard and screen readers.
11.10.7. No Unexpected Navigation: Warn before navigating away with unsaved changes.

11.11. Readiness Criteria

The application is considered “Launch Ready” only when:

11.11.1. All forms validate correctly with clear error messages.
11.11.2. All major actions show toasts (success/error/warning).
11.11.3. Notifications (pusher + email) work consistently.
11.11.4. CRUD operations function end-to-end without console errors.
11.11.5. Reusable components exist for every UI element.
11.11.6. Pages load in under 2 seconds on standard Wi-Fi.
11.11.7. No blocking security issues or data loss bugs exist.
11.11.8. QA sign-off from Buyer and Supplier roles is complete.

12. AI & ML Features

12.1. Design principles

12.1.1. Auditable by design. Every AI prompt, suggestion, and action is logged with user, time, and context so you can review who did what, when.
12.1.2. Human in the loop. AI proposes; users decide. Suggestions are always editable and reversible.
12.1.3. Graceful fallback. If AI is slow/unavailable, forms still work and show “hint unavailable."

12.2. RFQ Assist

12.2.1. Auto-prefill. Reads titles, descriptions, and attached files (PDF drawings, images, CAD filenames) to prefill fields like material, finish, process, quantity, and target lead time.

12.2.2. Gap & conflict checks. Flags missing items (e.g., tolerance not set) and conflicts (e.g., anodize + stainless) with clear, editable suggestions.
12.2.3. Why does it think so: Shows a brief rationale (e.g., “Detected ‘6061-T6’ from drawing note and file name").
12.2.4. User control. All fields remain editable; nothing is auto submitted without a person reviewing.

12.3. Supplier Matching

12.3.1. Fit over price. Ranks suppliers by capability, material, finish, location, and past outcomes (on time delivery, quality notes), plus other configured parameters (e.g., min order qty).
12.3.2. Rationale badges. Each match shows why it was picked (“5 similar jobs, avg 6-day lead time, 0 recent defects").
12.3.3. Filter aware. Respects user filters (country, certification) and plan rules.

12.4. Quote Assist

12.4.1. For suppliers: Suggests a price/lead time band based on similar past jobs on the platform, so suppliers can sanity check before submitting.
12.4.2. For buyers: Highlights outliers (too high/low) and calls out unusual lead times or MOQs with a short explanation.

12.5. Cost Band Estimator

12.5.1. Reasonable range: Estimates a target price band using process, material, region, and historical platform data.
12.5.2. Negotiation aid: Helps buyers spot inflated quotes and helps suppliers avoid under quoting.

12.6. CAD / Drawing Intelligence

12.6.1. Extracts basic hints from filenames and drawing text (material calls, finishes, common tolerance tags) to speed up RFQs.
12.6.2. Find similar parts: Surfaces “similar parts" previously quoted or ordered so teams can reuse the right vendors and reference prices.
12.6.3. Human check: Complex GD&T still requires human review at launch.

12.7. Digital Twin Intelligence

12.7.1. Auto link: Quotes, process notes, QA certificates, and warranty terms are attached to the part/asset’s Twin automatically.
12.7.2. Action prompts: Suggests follow ups like warranty reminders, re-order of wear parts, or adding a supplier to the preferred list based on outcomes.

12.8. Forecasting & Inventory Insights

12.8.1. Lead time awareness: Builds supplier lead time variance into reorder points so recommendations reflect reality, not just averages.
12.8.2. Clear outputs: Proposes safety stock, order by dates, and suggested order quantities with a one line “why."
12.8.3. What ifs: Simple scenario testing (“If we switch to Supplier B, how do order by dates change?").

12.9. Supplier Risk (Predictive)

12.9.1. Multi signal view: Combines delivery variance, defect/return rates, and certificate/terms gaps to score risk.
12.9.2. Explainable badges: “Medium risk: 2 late POs, ISO9001 expired, defect trend."
12.9.3. Mitigation tips: Suggests actions (increase lead time, add secondary supplier, request cert update).

12.10. ESG Packs

12.10.1. One click evidence bundle: Assembles a Scope-3 support pack from the digital twin/BOM and supplier inputs: method used, emission factors referenced, basic calculations, and linked evidence.
12.10.2. Share ready. Exports PDF/CSV for customers and auditors; shows data sources and assumptions.

12.11. Copilot

12.11.1. Read & guide: Natural language answers to common questions:
    a. “Summarize quotes on RFQ-1049."
    b. “What’s overdue this week?"
12.11.2. Action with approval: Drafts operations that need sign off (RBAC enforced):
    a. “Draft a PO to Supplier C for 200 units at $275, 6-day lead time."
12.11.3. Traceable. All Copilot actions are logged; high-impact actions always require approval.

12.12. Learning Loop

12.12.1. Continuously improves: Feeds accepted quotes, delivery results, defects and cost bands, and forecasts.
12.12.2. Quality visible. Shows model accuracy (e.g., MAPE/MAE) and data freshness, so teams know when to trust a signal or collect more data.

13. Client Inputs & Resources Needed

13.1. Supplier Information

13.1.1. Supplier list (Excel is fine): name, country, contact, what they can do (processes), materials, finishes.
13.1.2. Certificates: PDFs like ISO or safety/environment certificates, with expiry dates.
13.1.3. Performance: on time delivery %, quality issues, notes.

13.2. Past Activity (to load sample data)

13.2.1. RFQs: titles/descriptions, due dates, attached files (drawings/CAD/PDFs).
13.2.2. Quotes received: supplier, price, lead time, date, any notes.
13.2.3. Purchase Orders (POs): PO number, items, promised date, terms.
13.2.4. Invoices: invoice number, amount, status (Paid/Pending/Overdue).
13.2.5. Shipments: tracking numbers and delivered dates.

13.3. Parts & Items (for ordering and forecasting)

13.3.1. Item list: part/SKU number, description, unit, preferred supplier, typical lead time.
13.3.2. Stock rules: minimum / maximum stock levels.
13.3.3. BOM: which parts belong to which product.

13.4. Drawings & Manuals

13.4.1. Files to store: CAD/drawings (STEP/IGES/DWG/PDF) and any repair / maintenance manuals.
13.4.2. Good file names help: e.g., BRACKET_6061_RevC.pdf. (Strictly no space)

13.5. Maintenance & Assets (for maintenance features)

13.5.1. Machine list: name, model, serial, location.
13.5.2. Planned checks: weekly/monthly tasks, checklists, safety notes.
13.5.3. Downtime history: when it stopped, why, how long, parts used.

13.6. ESG & Compliance

13.6.1. Policies or statements: supplier codes of conduct, environment policies.
13.6.2. Certificates & audits: related PDFs with expiry dates.
13.6.3. Carbon data: material weights, shipping distances, any calculation notes.

13.7. How to Send Us the Files

13.7.1. Preferred: shared drive link (or we'll provide a secure folder).
13.7.2. Formats: Excel/CSV for lists; PDF/STEP/IGES/DWG/PNG/JPG for documents.
13.7.3. Clean data helps: remove duplicates and give each record a clear ID (e.g., supplier ID, RFQ ID).

14. Digital Twin Packs (3D) - Client Inputs

14.1. What we need from you

14.1.1. Pack categories

14.1.1.1. Examples: Pumps, Gearboxes, Conveyors, Hydraulic Power Units (HPU), Valves, Motors, Bearings, Blades/Knives, Custom Jigs/Fixtures.
14.1.1.2. For each category, confirm if it's a standalone part or an assembly (with sub-parts).

14.1.2. Representative items per category

14.1.2.1. 2 - 3 real examples per pack to model (the most common ones).
14.1.2.2. For each example: part number, name, short description, where it's used.

14.1.3. Geometry & measurements - Provide any one of these (more is better)

14.1.3.1. CAD/Drawing files (STEP/IGES/DWG/PDF).
14.1.3.2. Manufacturer datasheet (with dimensions).
14.1.3.3. Photos + a simple sketch with key dimensions (length/width/height, hole diameters, pitch, thickness, flange OD/ID, bolt circle, etc.).
14.1.3.4. Tolerances & surface finish if critical.

14.1.4. Assembly/BOM

14.1.4.1. Exploded view or list of sub-parts (name, part number, quantity, where it fits).
14.1.4.2. What sub-parts are wear parts (likely to be replaced) vs structural.

14.1.5. Where the part belongs

14.1.5.1. Which machine or line uses it (e.g., "HPU-0042 on Press Line A").
14.1.5.2. Any compatibility rules (e.g., "Blade type B only fits Chipper model X").

14.1.6. Materials & finishes

14.1.6.1. Material grade (e.g., 6061-T6, EN8), coating (e.g., black anodize), hardness/heat treatment.

14.1.7. Variants

14.1.7.1. Sizes or options (e.g., 25mm / 30mm bore; left/right-hand; voltage; thread type).

14.1.8. Maintenance notes

14.1.8.1. Basic PM/CM tasks (e.g., “Replace filter every 3 months," “Grease bearing weekly").
14.1.8.2. Safety notes, torque specs if you have them.

14.1.9. Photos/labeling conventions

14.1.9.1. A few clear photos (front, side, top; include a scale if possible).
14.1.9.2. How you name files/parts. Any specific pattern

14.2. Who does what (quick roles)

14.2.1. Client: choose categories, provide examples, upload files/measurements, confirm reviews.
14.2.2. Our team: model 3D/2D, map fields, build Twin Pack templates, set up RFQ/PO actions, and publish.

We will not guess part details. The client provides categories, examples, and measurements. We convert them into easy to use 3D Twin Packs that let teams identify parts, attach documents, and create RFQs/POs in one click.

15. UI / UX Design Standards

15.1. This section defines the complete visual language, layout structure, interaction principles, and accessibility rules for Elements Supply AI. These standards ensure that the user experience is consistent, efficient, and aligned with enterprise-grade (SAP-like) product design philosophies while remaining modern and lightweight.

15.2. React Starter Kit Theme & Component Consistency

The front-end implementation must strictly follow the existing React Starter Kit style and architecture provided with the Laravel 12 + React setup.

15.2.1. Design Frameworks

15.2.1.1. Primary UI framework: Tailwind CSS (for utility-based styling).
15.2.1.2. Component library: shadcn/ui (for forms, modals, toasts, and interactive components).
15.2.1.3. Icons: Iconify.
15.2.1.4. Layout engine: React Starter Kit default page and dashboard layouts.

15.2.2. Implementation Rules

15.2.2.1. Use Existing Layouts and Screens:

15.2.2.1.1. Reuse the existing layouts, navigation, and authentication screens included in the React Starter Kit installation.
15.2.2.1.2. Extend or modify them only when functionally necessary.
15.2.2.1.3. When modified, maintain consistent spacing, typography, padding, and component structure as the

15.2.2.2. Follow Established Visual Theme:

15.2.2.2.1. Maintain the color palette, typography scale, border radius, shadows, and animations that ship with the Starter Kit.
15.2.2.2.2. Do not introduce third-party themes or excessive custom styling outside the Tailwind + shadcn/ui ecosystem.
15.2.2.2.3. Any new pages or components must visually blend seamlessly with existing ones.

15.2.2.3. Component Reuse and Consistency:

15.2.2.3.1. Use the shared components/ui/ directory for all reusable atoms and molecules (buttons, inputs, modals, cards, tables).
15.2.2.3.2. When creating new components, follow the naming conventions, prop structure, and class usage patterns of shadcn/ui.
15.2.2.3.3. Avoid inline CSS or ad-hoc Tailwind classes that break consistency.

15.2.2.4. Layout Modifications:

15.2.2.4.1. Minor modifications (e.g., sidebar links, dashboard cards) are permitted if required by functional changes.
15.2.2.4.2. All layout updates must retain the same **grid system, top bar height, sidebar width, and responsive breakpoints** used in the Starter Kit.

15.2.2.5. Theming Rules:

15.2.2.5.1. The default dark/light mode toggle (if present) should continue to function with all new pages.
15.2.2.5.2. All custom components must inherit theme variables (colors, fonts) from Tailwind's configuration.

15.2.2.6. Screen and Page Creation:

15.2.2.6.1. When building new screens (e.g., RFQ, Quotes, Orders, Admin Console), use the existing layout wrappers

(DashboardLayout, AuthLayout, MainLayout) for structural consistency.

15.2.2.6.2. Page headers, breadcrumbs, and action buttons must follow existing placement and alignment patterns.

15.2.2.7. Accessibility and Responsiveness:

15.2.2.7.1. All new UI elements must remain responsive across mobile, tablet, and desktop using the same breakpoint logic defined in the Starter Kit.

15.2.2.7.2. Maintain accessibility semantics (ARIA, tab navigation, focus outlines) from shadcn/ui defaults.

15.2.3. Acceptance Tests

15.2.3.1. AT-UI-01: All new screens use the same typography, spacing, and color palette as the Starter Kit baseline.

15.2.3.2. AT-UI-02: Modified layouts maintain consistent grid, sidebar, and toolbar structure.

15.2.3.3. AT-UI-03: No external CSS frameworks or conflicting themes introduced.

15.2.3.4. AT-UI-04: All components use shadcn/ui or are derived directly from its base styles.

15.2.3.5. AT-UI-05: UI remains responsive and consistent across devices (mobile/tablet/desktop).

15.3. Design Philosophy

15.3.1. Form Follows Function:

15.3.1.1. The platform prioritizes information density, clarity, and traceability over decorative or playful design.

15.3.1.2. Every UI element should serve a functional purpose.

15.3.2. SAP-Inspired Efficiency:

15.3.2.1. Focus on business workflows—data tables, object pages, approval panels, timelines, and wizards.

15.3.2.2. Minimal animations, direct affordances (buttons, inputs, modals), and compact layouts.

15.3.3. Consistency Over Customization:

15.3.3.1. Use a **single shared component library** across all modules (RFQ, Quotes, POs, Documents, Analytics).

15.3.3.2. Avoid one-off styling; all pages inherit global design tokens.

15.3.4. Predictability:

15.3.4.1. Controls appear in the same place on every page.
15.3.4.2. Users should never need to guess how to perform an action.

15.3.5. Enterprise Neutral:

15.3.5.1. Colors and typography must remain professional and neutral to accommodate client branding overlays.

15.4. Color System

Category	Purpose	Default Color	Usage
Primary	Brand/Action	#0E2230	Buttons, links, icons, titles
Accent	Highlights/Active states	#0073B1	Selection, hover, focus
Success	Positive feedback	#1A9E55	Status: Paid, Delivered, OK
Warning	Attention required	#E5A50A	Status: Pending, Overdue
Error	Critical or invalid	#D92D20	Validation errors, failed uploads
Background	Page	#F5F7FA	Overall layout background
Card/Panel	Containers	#FFFFFF	Tables, modals, forms
Borders	Dividers	#E1E4E8	Input outlines, table rows
Text Primary	Titles/labels	#0E2230	Headings, primary text
Text Secondary	Metadata/hints	#6B7280	Descriptions, timestamps

15.4.1. Use **only Tailwind CSS color tokens** mapped to these values.
15.4.2. High contrast (min. 4.5:1) is mandatory for all text/background pairs.
15.4.3. Dark mode optional, but ensure all colors can invert gracefully.

15.5. Typography

Element	Font Family	Weight	Size	Usage
Headings (H1–H3)	Manrope / Poppins	600–700	24px–18px	Page & section titles
Body Text	Manrope	400	16px	Default text
Small Text / Metadata	Manrope	400	14px	Captions, labels
Button / Action Text	Manrope	500	15px	Buttons, tags

15.5.1. Line height = 1.5× font size.
15.5.2. Paragraph spacing = 1.25× line height.
15.5.3. All text left-aligned, except numerical data (right-aligned).
15.5.4. Headings use consistent capitalization (Sentence case only).

15.6. Layout System

15.6.1. Grid

15.6.1.1. 12-column responsive grid using Tailwind's flex + grid utilities.
15.6.1.2. Max content width: 1440px, centered.
15.6.1.3. Gutter spacing: 24px desktop / 16px tablet / 8px mobile.
15.6.1.4. Padding inside sections: 32px top/bottom, 24px sides.

15.6.2. Page Archetypes

15.6.2.1. List Page (Table View)

15.6.2.1.1. Sticky header with page title and primary actions (e.g., "Create RFQ").
15.6.2.1.2. Filters and search bar top-aligned above the table.
15.6.2.1.3. Table: striped rows, hover highlight, 44–48px row height.
15.6.2.1.4. Pagination at bottom right.

15.6.2.2. Object Page (Detail View)

15.6.2.2.1. Header: key metadata (ID, title, status, supplier, total).
15.6.2.2.2. Tabs below header: Overview, Attachments, Timeline, Audit Log.
15.6.2.2.3. Right-side panel for quick actions (Approve, Print, Export).

15.6.2.3. Wizard (Multi-Step Form)

15.6.2.3.1. Stepper at top with "Next / Back" navigation.
15.6.2.3.2. Auto-save draft on step change.
15.6.2.3.3. Validation per step, not global.

15.6.2.4. Dashboard (KPI/Analytics)

15.6.2.4.1. 2–3 KPIs per row.
15.6.2.4.2. Chart style: minimalist (bar, line, pie only).
15.6.2.4.3. All widgets exportable as PNG or CSV.

15.6.2.5. Timeline View

15.6.2.5.1. Chronological status flow (e.g., Order: Pending → Shipped → Delivered).
15.6.2.5.2. Each event shows timestamp, user, role, and attachments.

15.7. Components Library

Each module must reuse components from a central library.
Below outlines the full atomic component set.

15.7.1. Atoms

- 15.7.1.1. Buttons (primary, secondary, danger, ghost)
- 15.7.1.2. Icons (Iconify icon set, 16px/20px)
- 15.7.1.3. Inputs (text, number, date, select, file, textarea)
- 15.7.1.4. Checkbox / Toggle / Radio
- 15.7.1.5. Tooltip / Badge / Tag
- 15.7.1.6. Avatar / Initial bubble
- 15.7.1.7. Spinner / Skeleton loader

15.7.2. Molecules

- 15.7.2.1. Form group (label + input + helper text + error message)
- 15.7.2.2. Card (with header + content + footer)
- 15.7.2.3. Modal / Drawer
- 15.7.2.4. Notification Toast
- 15.7.2.5. Search bar
- 15.7.2.6. Tabs
- 15.7.2.7. Stepper (for wizard)
- 15.7.2.8. Empty state card (icon + text + CTA)
- 15.7.2.9. Table row (configurable columns)

15.7.3. Organisms

- 15.7.3.1. Data Table (sortable, paginated, with filter row)
- 15.7.3.2. Master-Detail Split Panel
- 15.7.3.3. Form Wizard (multi-step component)
- 15.7.3.4. Timeline View
- 15.7.3.5. Comment/Q&A Thread
- 15.7.3.6. File Manager (grid + list toggle)
- 15.7.3.7. KPI Card
- 15.7.3.8. Audit Log Viewer

15.8. Interaction & Behavior Rules

15.8.1. Feedback & Loading

- 15.8.1.1. All async actions show a spinner (centered or inline).
- 15.8.1.2. Toast appears for: success ✅, error ❌, warning ⚠, info **i**.
- 15.8.1.3. Loading states use skeleton placeholders, not blank screens.

15.8.2. Validation

15.8.2.1. Inline error message below field: red text (#D92D20).
15.8.2.2. Tooltip hints for formatting help.
15.8.2.3. “Next” or “Submit” disabled until validation passes.

15.8.3. Hover & Focus States

15.8.3.1. Hover: light blue background (#E6EEF2).
15.8.3.2. Focus: 2px outline (#0073B1).
15.8.3.3. Active: pressed effect (scale 0.98).

15.8.4. Modals & Drawers

15.8.4.1. Modal size: small (400px), medium (720px), large (1024px).
15.8.4.2. Always dismissible via “X” and ESC key.
15.8.4.3. No nested modals allowed (use drawers or tabs).

15.8.5. Navigation & Breadcrumbs

15.8.5.1. Top nav with product switcher and search.
15.8.5.2. Left sidebar: collapsible, icons + labels, persistent highlight on active route.
15.8.5.3. Breadcrumbs appear on every page (Home / RFQs / RFQ #123).
15.8.5.4. Quick action toolbar (Export / New / Filter / More) top-right.

15.8.6. Scroll & Pagination

15.8.6.1. Virtual scroll for large tables (>200 rows).
15.8.6.2. Pagination control bottom-right (Prev, Next, 1…10).
15.8.6.3. Infinite scroll allowed only for activity feeds.

15.8.7. Notifications

15.8.7.1. In-app notification bell (header) with unread count.
15.8.7.2. “View All” opens full history grouped by date.
15.8.7.3. Clicking a notification deep-links to its record.

15.9. Accessibility & Keyboard Navigation

15.9.1. Every component must be operable by keyboard:
15.9.1.1. Tab = focus next
15.9.1.2. Shift+Tab = focus previous
15.9.1.3. Enter = confirm
15.9.1.4. ESC = cancel/dismiss
15.9.2. Use ARIA labels on all interactive elements.
15.9.3. Ensure focus order follows visual layout.
15.9.4. All icons require text equivalents (alt or aria-label).

15.10. Empty States & Error Handling

Situation	Message Style	Example
No Data	Illustration + title + CTA	“No RFQs yet. Create your first RFQ.”
Validation Error	Inline red text + icon	“Delivery date cannot be earlier than today.”
System Error	Modal alert + retry button	“Something went wrong. Please try again.”
404 / Access Denied	Full-page message + home link	“You don’t have permission to view this page.”

15.11. File Upload & Preview

15.11.1. Accepted file types: .pdf, .docx, .xlsx, .stp, .iges, .dwg, .jpg, .png.
15.11.2. Max size: 50 MB per file.
15.11.3. Drag & drop zone with progress bar.
15.11.4. On hover: “View”, “Download”, and “Delete” icons.
15.11.5. PDF and images preview inline; CAD shows thumbnail + metadata.

15.12. Mobile Design Standards

15.12.1. Mobile menu as bottom navigation (Home, RFQs, Orders, Docs, Profile).
15.12.2. Use collapsible accordions instead of tables.
15.12.3. Floating “+” button for primary actions.
15.12.4. Swipe left to reveal contextual options (Edit, Delete).
15.12.5. Responsive touch area ≥44px.
15.12.6. Input forms auto-scroll to next field on submit.

15.13. Theming & White Labeling

15.13.1. Each company’s logo, colors, and footer text configurable under Settings → Branding.
15.13.2. PDF templates and emails automatically adapt to theme.
15.13.3. Default Elements Supply AI branding always retained in footer unless white-label license applied.

15.14. Icons & Visual Language

15.14.1. Use IconifyIcons (16px inline, 20px toolbar).
15.14.2. Status indicators:
15.14.2.1. 🟢 = Success / Active
15.14.2.2. 🟠 = Warning / Pending
15.14.2.3. 🔴 = Error / Rejected
15.14.2.4. ⚪ = Neutral / Draft
15.14.3. Avoid emoji or 3D icons in production UI (only internal mockups).

15.15. Microinteractions & Motion

15.15.1. Duration: 150–250ms max.
15.15.2. Easing: ease-in-out.
15.15.3. Use for hover, button press, expand/collapse, modal open/close.
15.15.4. Never animate data tables or core transactional elements.

15.16. UX Writing & Label Standards

15.16.1. Tone: professional, clear, neutral.
15.16.2. Avoid jargon; spell out abbreviations once (e.g., “PO (Purchase Order)”).
15.16.3. Button labels = action verbs (“Create RFQ”, “Submit Quote”, “Download PDF”).
15.16.4. Error messages: human-readable, not technical.
15.16.5. Use title case for UI buttons and labels.

15.17. Onboarding & Help

15.17.1. Step-by-step guided onboarding for new users:
15.17.1.1. Highlight key modules (RFQ, PO, Docs).
15.17.1.2. Show sample data for demonstration.
15.17.2. “Help” icon opens side drawer with contextual documentation or link to Knowledge Base.
15.17.3. Chatbot (optional) provides contextual hints using natural language.

15.18. Dashboard Design

15.18.1. Minimal 3-row layout:
15.18.1.1. Row 1: KPIs (Spend, RFQs, Orders, On-time Delivery %)
15.18.1.2. Row 2: Open Tasks (Approvals, Pending Invoices)
15.18.1.3. Row 3: Recent Activity Feed
15.18.2. Use consistent widget sizes (300–400px width).
15.18.3. Allow drag-and-drop repositioning for Enterprise tier.

15.19. User Feedback & Ratings

15.19.1. After key actions (RFQ creation, Quote submission, Order closure), prompt short feedback modal:
15.19.1.1. “How was this process?” ★★★★☆
15.19.1.2. Optional comment field.
15.19.2. Aggregate UX metrics visible only to Platform Admin for continuous improvement.

15.20. Design Governance

15.20.1. All new pages/components must pass **UX Review Checklist**:
15.20.1.1. Alignment with spacing grid.
15.20.1.2. Accessibility test.

15.20.1.3. Component reuse compliance.
15.20.1.4. Consistent button placement.
15.20.1.5. Responsive validation.
15.20.2. Design system managed via version control.
15.20.3. Updates require version increment and changelog entry.

16. UI Development Instructions for Copilot

This section defines the exact implementation rules and constraints for VS Code Copilot (and other AI-assisted code generation tools) when generating user interface components, screens, layouts, and elements for the Elements Supply AI application. The objective is to maintain a unified, professional UI consistent with the React Starter Kit, prevent redundant code, and ensure visual and behavioral consistency throughout the platform.

16.1. General Guidelines

16.1.1. Primary Stack

16.1.1.1. Frontend framework: React (TypeScript)
16.1.1.2. Styling: Tailwind CSS
16.1.1.3. Component library: shadcn/ui
16.1.1.4. Icons: Iconify

16.1.2. Theme Consistency

16.1.2.1. Always inherit **Tailwind configuration** (colors, spacing, typography, border-radius, shadows).
16.1.2.2. Avoid introducing new color codes or inline styles unless explicitly defined in Tailwind config.
16.1.2.3. Use **dark/light mode** toggles that already exist in the Starter Kit.

16.1.3. Component Creation Rules

16.1.3.1. Always prefer composition over duplication.
16.1.3.2. Reuse existing shadcn/ui components and pass props for variations.
16.1.3.3. Every form must use Form, Input, Select, and Button components from shadcn/ui.
16.1.3.4. Modals, alerts, tables, and toasts must also use standardized shadcn/ui components.
16.1.3.5. Place new shared components only under /ui/, and prefix with clear functional names.

16.2. Code Generation Behavior

16.2.1. Follow React Starter Kit structure and import existing layouts and base components.
16.2.2. Use TypeScript interfaces for props; no implicit any types.
16.2.3. Use functional components with hooks (useState, useEffect, useQuery, etc.).
16.2.4. Implement proper loading, empty, and error states using shared UI patterns.
16.2.5. Maintain accessibility (ARIA attributes, keyboard focus, semantic HTML).
16.2.6. Use Tailwind utility classes from the Starter Kit — never inline CSS or styled-components.
16.2.7. Generate reusable skeleton loaders and empty-state placeholders for all data-driven screens.
16.2.8. Keep UI text minimal and professional; follow tone defined in “UI/UX Writing Standards”.

16.3. Visual and Interaction Consistency

16.3.1. Page Layouts

16.3.1.1. All new pages must use an existing layout (Dashboard, Auth, Main).
16.3.1.2. Primary actions (e.g., “Create RFQ”, “Submit Quote”) appear at the top-right of the header bar.
16.3.1.3. Secondary actions go under a “More” (⋮) menu.
16.3.1.4. Breadcrumbs appear at the top-left for navigation consistency.

16.3.2. Tables and Lists

16.3.2.1. Use a shared DataTable component for all tabular data.
16.3.2.2. Include pagination controls and bulk action support.
16.3.2.3. Default sorting: created_at desc.

16.3.3. Forms and Wizards

16.3.3.1. Multi-step processes (e.g., RFQ creation, Quote submission) use Stepper components.
16.3.3.2. Show inline validation with Tailwind text-red-500 styles and error messages beneath fields.
16.3.3.3. Include success/error toasts on all submit actions.

16.3.4. Toasts and Notifications

16.3.4.1. All alerts, confirmations, and status messages must use the shared Toast component.
16.3.4.2. Toasts appear in the top-right and auto-dismiss after 4 seconds.

16.3.5. Loading and Empty States

16.3.5.1. Use skeleton loaders (gray shimmer placeholders) while fetching data.
16.3.5.2. Empty screens display an illustration + title + CTA button.

16.3.6. Accessibility

16.3.6.1. Ensure focus rings, keyboard tab navigation, and ARIA roles are present.
16.3.6.2. Buttons must be reachable and operable via keyboard.
16.3.6.3. Use readable contrast (WCAG AA+) for all text and icons.

16.4. UI Code Structure & Standards

16.4.1. Naming Rules

16.4.1.1. Components: PascalCase (e.g., RfqForm.tsx)
16.4.1.2. Hooks: camelCase starting with use (e.g., useFetchQuotes.ts)
16.4.1.3. CSS utilities: kebab-case in Tailwind (e.g., bg-slate-100)
16.4.1.4. Routes: kebab-case (e.g., /rfqs, /purchase-orders)

16.4.2. Code Quality

16.4.2.1. Use ESLint + Prettier formatting.
16.4.2.2. Maintain clean imports and avoid dead code.
16.4.2.3. Enforce consistent prop order and indentation (2 spaces).
16.4.2.4. Avoid unnecessary console logs in production.

16.5. UI Development Guard Rails

16.5.1. No duplicated components across modules; all shared components live in /ui/.
16.5.2. Every interactive element must have hover/focus states.
16.5.3. Do not bypass layout wrappers; all pages must inherit from an approved layout.
16.5.4. Keep component size manageable (≤200 lines per file; split when needed).
16.5.5. No direct DOM manipulation — use React state/hooks.
16.5.6. Always use React Router for navigation (no window.location redirects).
16.5.7. Use axios or fetch from /services/api.ts — no ad-hoc HTTP calls.
16.5.8. When uncertain, prefer simplicity and follow existing Starter Kit patterns.

16.6. Acceptance Tests

16.6.1. AT-UI-06: All new screens visually match the React Starter Kit theme and shadcn/ui component style.
16.6.2. AT-UI-07: All actions use consistent spacing, fonts, and button placements.
16.6.3. AT-UI-08: No external CSS frameworks or inconsistent styling introduced.

16.6.4. AT-UI-09: Components compile without lint or type errors.
16.6.5. AT-UI-10: UI remains responsive and accessible at all breakpoints.

17. Notifications UX (Push + Email + Read State)

Purpose: Ensure real-time, reliable notifications with consistent read/unread state across channels.

17.1. Events (minimum)

17.1.1. RFQ created/invitation; Quote submitted/withdrawn; Award/PO issued; GRN posted; Invoice status changed; Plan over-limit.

17.2. Data Model

17.2.1. notifications (id, company_id, user_id, type, title, body, entity_type, entity_id, channel enum: push|email|both, read_at, meta json)
17.2.2. user_notification_prefs (user_id, event_type, channel, digest enum:none|daily|weekly)

17.3. Realtime & Email

17.3.1. Websocket via Echo/Pusher for push; queued mail for email; single event produces both when configured.

17.4. UI

17.4.1. Header bell with badge; Inbox panel (Today / Earlier) with Mark all as read.
17.4.2. Clicking an item deep-links to the record; read state toggles immediately (optimistic).

17.5. Acceptance Tests

17.5.1. AT-21.1: When a Quote is submitted, target buyer receives both push & email (if prefs=both) within 5s/2m respectively.
17.5.2. AT-21.2: Mark as read in the bell syncs with the Notifications page; page refresh persists.
17.5.3. AT-21.3: Changing preference to “daily digest” suppresses immediate emails and sends one summary at 18:00 local.
17.5.4. AT-21.4: Duplicate notifications are deduped per event id; never double-send.

18. Analytics Permissions & Data Guarding

Purpose: Ensure dashboards respect **role** and **plan**; prevent leakage across tenants or roles.

18.1. Access Rules

18.1.1. Buyer Admin/Requester: sees only their company data.
18.1.2. Supplier roles: see only their own quotes, POs, on-time metrics for them (no buyer-internal spend).
18.1.3. Finance: financial dashboards only (AP/AR, invoice aging).
18.1.4. Platform Admin: cross-tenant system metrics only (never tenant PII unless impersonating).

18.2. Plan Gating

18.2.1. Starter: operational widgets only (counts, recent activity).
18.2.2. Growth: adds trend charts, supplier scorecards.
18.2.3. Enterprise: full suite + export + custom widgets; can schedule reports.

18.3. Technical

18.3.1. All analytics queries tenant-scoped by company_id at repo/service layer.
18.3.2. PII masking in charts: supplier emails/usernames never shown—use display names or IDs.
18.3.3. Cached materialized views recomputed hourly; plan gating applied at query layer.

18.4. UI

18.4.1. Dashboard tiles show a lock icon with tooltip when gated by plan.
18.4.2. Export buttons hidden for non-eligible plans/roles.

18.5. Acceptance Tests

18.5.1. AT-24.1: Supplier viewing “Spend by Supplier” sees only their own totals or a gated screen.
18.5.2. AT-24.2: Starter plan cannot export CSV/PDF; Growth can; Enterprise can schedule.
18.5.3. AT-24.3: When plan upgrades, gated widgets unlock without logout; downgrade re-locks after webhook event.

19. Cross-Cutting Guard Rails & Consistency Rules

These rules apply to every module, page, API, and component. They exist to prevent duplication, reduce code size, enforce consistency, and guarantee smooth UX and reliable operations.

19.1. No Duplication (DRY) – Functions, Modules, Routes

19.1.1. Rules

19.1.1.1. Single Source of Truth: Any shared behavior (validation, toasts, date/price formatting, API calls) must live in a single reusable utility or component.
19.1.1.2. Route Uniqueness: Route names and paths must be unique. Routes must be declared in a central registry (backend + frontend) to prevent collisions.
19.1.1.3. Dedup Checks in CI: CI must fail on duplicated code > 30 lines or 85% similarity.

19.1.2. Enforcement

19.1.2.1. Frontend: ESLint “no-duplicate-imports”, “no-duplicate-case”; JSCPD (or npx jscpd) for copy/paste detection.
19.1.2.2. Backend: PHPStan/Larastan + rector sets; Pest architectural tests to forbid controller-to-controller calls.

19.1.2.3. Route registry files:

19.1.2.3.1. Laravel: /routes/web.php, /routes/api.php with a route name prefix per module.
19.1.2.3.2. React: /resources/js/routes/index.ts exporting a ROUTES object (single source).

19.1.3. Acceptance Tests

19.1.3.1. AT-25.1: JSCPD score ≤ 3% duplication in PR.
19.1.3.2. AT-25.2: No route duplication (CI route linter passes).
19.1.3.3. AT-25.3: Shared helpers imported from /utils or /services, not re-implemented locally.

19.2. Minimize Code & Keep It Simple

19.2.1. Rules

19.2.1.1. Prefer composition over inheritance in React components.
19.2.1.2. Keep files short: functions ≤ 50 lines; components ≤ 200 lines (split if larger).
19.2.1.3. Avoid premature abstractions; extract only when reused ≥2 places.

19.2.2. Enforcement

19.2.2.1. ESLint complexity cap ("complexity": ["error", 10]), max-lines-per-function, max-lines-per-file.
19.2.2.2. PHP CS Fixer + Rector to simplify code.
19.2.2.3. CI fails if complexity or file length thresholds are exceeded.

19.2.3. Acceptance Tests

19.2.3.1. AT-25.4: Lint/format passes with zero warnings.
19.2.3.2. AT-25.5: No component exceeds agreed thresholds (CI gate).

19.3. Launch-Ready Feature Criteria

19.3.1. Rules

19.3.1.1. Every feature must:

19.3.1.1.1. a) Validate inputs (client + server)
19.3.1.1.2. b) Use shared toasts (success/error)
19.3.1.1.3. c) Update UI state immediately (optimistic or after re-fetch)
19.3.1.1.4. d) Create an audit entry
19.3.1.1.5. e) Include empty state & loading skeleton
19.3.1.1.6. f) Have at least one CSV export if relevant (tables)

19.3.2. Acceptance Tests

19.3.2.1.1. AT-25.6: For each feature PR, checklist includes validation, toast, state update, audit, skeleton, empty state. QA blocks merge if any missing.

19.4. Notifications: Push + Email (Right User, Right Time)

19.4.1. Rules

19.4.1.1. All business events emit a single domain event → dispatcher fan-out to push/email per user prefs.
19.4.1.2. No double sends: dedupe by event id + recipient.
19.4.1.3. All notifications deep-link to the entity.

19.4.2. Enforcement

19.4.2.1. Central NotificationService only; no direct mail/push in feature code.
19.4.2.2. “Mark as read” syncs bell + inbox + mobile.

19.4.3. Acceptance Tests

19.4.3.1. AT-25.7: Event → push within 5s; email within 2m; read state consistent after refresh.
19.4.3.2. AT-25.8: Same event only notifies once per user.

19.5. Smooth Authentication

19.5.1. Rules

19.5.1.1. Standard Laravel auth + CSRF; remember-me optional.
19.5.1.2. Sign-in < 2 steps; error messages human-readable.

19.5.1.3. On sign-in, restore last organization + last visited module (if available).
19.5.1.4. 401/419 → unified re-auth dialog, preserves unsaved state if possible.

19.5.2. Acceptance Tests

19.5.2.1. AT-25.9: Re-auth returns user to the original page/action.
19.5.2.2. AT-25.10: Invalid credentials show clear inline error; no stack traces.

19.6. State Updates & Data Freshness

19.6.1. Rules

19.6.1.1. Mutations must **optimistically update** UI (or re-fetch on success) and handle rollback on error.
19.6.1.2. Lists invalidate caches of affected queries (RFQs, Quotes, POs).
19.6.1.3. Detail pages refresh header stats after actions.

19.6.2. Enforcement

19.6.2.1. Use a standard useMutation/useQuery wrapper (SWR/React Query pattern or custom hooks).
19.6.2.2. No manual setState against stale copies; always go through the query cache.

19.6.3. Acceptance Tests

19.6.3.1. AT-25.15: After create/update/delete, the list and detail reflect changes without full page reload.
19.6.3.2. AT-25.16: Failed mutation rolls back optimistic UI and shows error toast.

19.7. Consistency Across Pages/Components

19.7.1. Rules

19.7.1.1. All pages use shared UI tokens (colors, spacing, typography) and standard page archetypes (List, Object, Wizard, Timeline, Dashboard).
19.7.1.2. Primary actions placed top-right; secondary under “⋮”.
19.7.1.3. Confirmation modals for destructive actions; same copy system-wide.

19.7.2. Acceptance Tests

19.7.2.1. AT-25.17: Heuristic scan — action placements are consistent on RFQ/Quote/PO/Invoice pages.

19.7.2.2. AT-25.18: Button labels follow verb style (“Create RFQ”, “Submit Quote”, “Download PDF”).

19.8. Routing & Naming Conventions

19.8.1. Rules

19.8.1.1. Backend routes: kebab-case, versioned for API (e.g., /api/v1/rfqs).
19.8.1.2. Frontend routes: centralized ROUTES object with typed params.
19.8.1.3. Naming: Models (Singular, PascalCase), tables (snake_case plural), components (PascalCase), files (kebab-case).

19.8.2. Acceptance Tests

19.8.2.1. AT-25.19: /api/v1/* only; no ad-hoc or unversioned APIs.
19.8.2.2. AT-25.20: Route params are validated (numeric ids, UUIDs) at controller request level.

19.9. Error Handling & Observability

19.9.1. Rules

19.9.1.1. All API responses: { status, message, data } with appropriate HTTP codes.
19.9.1.2. Global error boundary in React; friendly fallback screens.
19.9.1.3. Log levels: error/warn/info; P1 alerts ping Slack/Email.

19.9.2. Acceptance Tests

19.9.2.1. AT-25.21: 4xx shows inline messages; 5xx shows retry & support link.
19.9.2.2. AT-25.22: Audit created for all create/update/delete (before/after snapshot).

19.10. HTTP Status & Error Envelope

19.10.1. All API endpoints must return a uniform response envelope:

```
{
  "status": "success" | "error",
  "message": "Human-readable explanation",
  "data": { },
  "errors": {
    "field_name": ["Error message…"]
  }
}
```

Code	Symbolic Name	Meaning
200	OK	Successful operation
201	Created	Resource created
204	NoContent	Deletion or empty response
400	BadRequest	Malformed input
401	Unauthorized	Auth required
403	Forbidden	Access denied
404	NotFound	Resource missing
409	DuplicateEntity	Conflict / already exists
422	ValidationError	Field-level failures
429	RateLimited	Too many requests
500	ServerError	Unhandled exception

19.11. Pagination Defaults

19.11.1. All list endpoints must support the standard pagination contract: ?page=1&per_page=25
19.11.2. Default: page=1, per_page=25
19.11.3. Maximum: per_page=100
19.11.4. Response Envelope: { data:[], meta:{ total, page, per_page, last_page } }
19.11.5. Sort Order: default created_at DESC; must accept sort_by and direction parameters.
19.11.6. Empty states and loaders should handle pagination gracefully in UI.

19.12. Search Indexes (FULLTEXT Columns)

19.12.1. Copilot must create FULLTEXT indexes and search scopes on these columns:

Table	Columns	Notes
suppliers	name, capabilities	Used in Directory search & filters
rfqs	title, material, method	Buyer global search
documents	filename, mime	Document manager lookup

quotes	note	Quick filter inside RFQ detail
audit_logs	entity_type, action	Admin console search
notifications	title, body	Inbox search

19.12.2. Search queries must be tenant-scoped (company_id) and safe from SQL injection by using Laravel Query Builder search scopes.

19.13. Email & Push Notification Templates

19.13.1. All notification events use pre-defined Blade templates with matching subject lines.

19.13.2. Templates reside in resources/views/emails/<module>/.

Event	Template File	Email Subject	Push Title
RFQ Invitation Sent	emails/rfq/invitation.blade.php	New RFQ Invitation – {{ rfq.title }}	RFQ Invitation Received
Quote Received	emails/quote/received.blade.php	Quote Received for {{ rfq.title }}	New Supplier Quote Submitted
RFQ Awarded / PO Issued	emails/po/awarded.blade.php	Purchase Order {{ po.po_number }} Awarded	PO Awarded to You
Invoice Paid	emails/invoice/paid.blade.php	Invoice {{ invoice.invoice_number }} Marked Paid	Payment Received
Plan Over Limit	emails/billing/overlimit.blade.php	Plan Limit Reached – Action Required	Usage Limit Warning
System Announcement	emails/platform/announcement.blade.php	Important Notice from Elements Supply AI	Platform Announcement

19.13.3. Each email template must have a corresponding notification class and Blade view.

20. Shared Guard Rails (for Copilot & Devs)

20.1. All new endpoints must return { status, message, data } and meaningful HTTP codes.
20.2. Every create/update/delete logs an audit record with before/after snapshot.
20.3. All screens use shared UI components (inputs, selects, tables, toasts, modals).
20.4. Every async action → spinner + toast (success/error).
20.5. Permissions checked in policy layer + route middleware; never only in the UI.
20.6. Add seeders & factories for happy-path demo data (admins, companies, keys, notifications).

20. Acceptance Test Checklists

20.1. Supplier Discovery & Profiles

a. Given capability filters (process, material, location), when a buyer searches, then only matching, verified suppliers are listed.
b. Given a supplier profile, when capabilities/certificates are updated, then changes require approval if flagged and appear in the audit log.
c. Given a supplier card, when ‘Invite to RFQ’ is clicked, then the RFQ wizard opens prefilled with the supplier selected.
d.

20.2. RFQ / RFP Creation

a. Given a CAD file (STEP/IGES) is uploaded, when the wizard runs, then part type/material suggestions display and required fields validate.
b. Given an RFQ marked ‘Open’, when submitted, then it is visible to eligible suppliers until the bidding deadline.
c. Given an RFP, when published, then suppliers can submit proposals with attachments and structured fields (price, lead time, more).

20.3. Quote Intake (Supplier)

a. Given a supplier receives an RFQ, when opening, then the quote form shows price/lead time fields and attachment uploads.
b. Given AI quote assist is enabled, when loading the form, then suggested price/lead time values populate with an edit option.
c. Given a submission, when saved, then the buyer sees the quote in comparison view with timestamp and status.

20.4. Quote Comparison & Award

a. Given at least two quotes, when viewing comparison, then price, lead time, rating, and notes columns appear sortable.

b. Given a selected quote, when ‘Award’ is confirmed, then the PO creation flow triggers (auto or manual).
c. Given an open RFQ deadline passes, when no quotes exist, then the RFQ auto closes and notifies the requester.

20.5. Purchase Orders

a. Given an awarded quote, when auto PO is configured, then a PO PDF is generated with dual branding and unique number.
b. Given manual PO mode, when drafted and submitted, then approvers receive notifications as per the approval matrix.
c. Given a PO, when exported, then PDF and CSV formats are downloadable

20.6. Invoicing & Payment Tracking

a. Given a PO exists, when ‘Generate Invoice’ is clicked, invoice data pre-fills from the PO and is editable before issue.
b. Given an invoice status update, when marked Paid/Pending/Overdue, then order and finance dashboards reflect the new status.
c. Given plan limits, when invoice count exceeds allowance, then the system blocks creation with an upgrade prompt

20.7. Orders & Logistics Tracking

a. Given a PO is confirmed, when the supplier updates status to In Production or Shipped, then the order timeline shows the change and sends alerts.
b. Given a tracking number is added, when saved, then it is visible to the buyer with a carrier link.
c. Given a delivery is confirmed, when marked Delivered, then cycle time metrics update.

20.8. Document & CAD Management

a. Given a project/order, when a user uploads a document, then file type validation runs and permissions restrict access to authorised parties.
b. Given a CAD file, when preview is requested, then a lightweight viewer or download is available.
c. Given retention rules, when a project closes, then documents remain accessible per retention policy.

20.9. Reporting & Dashboards

a. Given user role and plan, when loading dashboards, then the correct widgets (spend, supplier performance, cycle times) appear.
b. Given an export request, when CSV/PDF is chosen, then the file downloads with applied filters.
c. Given period filters, when changed, then charts/tables refresh accordingly.

20.10. Integrations (Foundations)

a. Given CSV import templates, when valid files are uploaded, then records are created or with error reporting for rejects.
b. Given API tokens, when used, then endpoints authenticate and rate limits apply.
c. When a given plan includes ERP/MRP integration, when connected, then data sync jobs are visible with logs.

20.11. Roles & Permissions

a. Given RBAC rules, when a user without permission tries to access a module, then a denied message appears and is audited.
b. Given a new role is assigned, when the user logs in, then navigation reflects allowed modules only.
c. Given an audit requirement, when sensitive actions occur (award, cancel, delete), then audit entries record actor, time, and context.

21. End to End User Journey

21.1. Upload drawings/CAD/manuals: platform creates or updates the Twin.
21.2. Need a spare part: From the manual/exploded view, add the part to RFQ/PO.
21.3. Supplier quotes arrive: platform ranks by fit, risk, lead time, and price.
21.4. Approve PO: goods received; QA docs stored on the digital twin.
21.5. Maintenance performed: spares consumed; downtime and costs logged.
21.6. Forecast updates safety stock: Copilot suggests “order-by" dates and supplier mix.
21.7. ESG pack compiles evidence for audits/customer requests.

22. Security, Governance, and Scale

22.1. RBAC by role, project, and supplier
22.2. Approvals on every material action (min/max changes, PO creation, ESG release)
22.3. Full audit trails (who saw/downloaded/approved what, when)
22.4. API first architecture for ERP/CMMS integrations

23. Why We Stand Out

23.1. Competitors focus on quoting or machining. We anchor everything: documents, sourcing, and maintenance, on a digital twin that lives beyond the purchase moment.
23.2. Manuals that drive transactions: Interactive manuals and exploded views link directly to parts lists, RFQs, and POs.
23.3. Maintenance integrated procurement: work orders automatically pre stage spares and suggest supplier switches before failure.

23.4. Explainable AI: Every recommendation (forecast, supplier, price band) shows why, with drill through to evidence (POs, QA docs, telemetry).
23.5. ESG by design: Supplier evidence collection, certificate aging alerts, and auto assembled ESG support packs, ready to share with customers.
23.6. Open architecture: Real ERP/CMMS, not just CSV imports, so the platform slots into existing operations without heavy change management.
23.7. Role based Copilot: Natural language commands that actually do work (draft RFQs/POs, compare quotes, generate ESG packs), with approvals and full audit trails.

24. Competitive Comparison

Legend: ✔ Yes | ◑ Partial/typical | X Not core / not typical

Capability	Protolabs	Xometry	SAP Ariba	GEP	Elements Supply AI
Central Document Control Hub (versioning, approvals, audit)	✔	✔	✔	✔	✔
Digital Twin per asset/part (3D viewer, lifecycle)	X	X	X	✔	✔
Interactive Maintenance Manuals linked to parts & inventory	X	X	X	✔	✔
Exploded parts with click to RFQ/PO	X	X	✔	✔	✔
Maintenance Plans	X	X	X	◑	✔
CAD aware RFQ/RFP with supplier matching	✔	✔	◑	✔	✔
Quote comparison by fit/lead/price/risk with explanations	✔	✔	✔	✔	✔
AI/ML Forecasting (safety stock, order by, lead time risk)	◑	✔	✔	◑	✔
Supplier Risk (OTD, defects, term compliance) with badges	◑	◑	✔	✔	✔
ESG Workspace (certs, Scope-3 support packs)	✔	✔	◑	✔	✔

ERP/CMMS two way sync (items, POs, WOs, costs)	◑	X	✔	✔	✔
Role based Copilot that drafts RFQs/POs & ESG packs	X	◑	◑	◑	✔

26. Database Plan (MySQL + Laravel 12)

This plan defines the canonical data model for Elements Supply AI.
All tables are **tenant-scoped** by `company_id` unless stated as platform-wide.

26.1 Conventions (Copilot must follow)

- **Engine/Charset:** InnoDB, `utf8mb4`, `utf8mb4_unicode_ci`.
- **Naming:** tables = `snake_case` plural; PK = `id` (BIGINT UNSIGNED, AI); FKs = `{singular}_id`.
- **Timestamps:** `created_at`, `updated_at`; soft deletes: `deleted_at` on business tables.
- **Multitenancy:** include `company_id` (BIGINT UNSIGNED, FK→`companies.id`) on all tenant data.
- **Enums:** use Laravel `enum()` (or string with CHECK) with constants in Eloquent models.
- **Auditing:** every create/update/delete emits an `audit_logs` row (before/after JSON).
- **Files:** store metadata in `documents` with object-morph; binaries in S3/local storage.
- **Indexes:** always index FKs; add composite indexes for common queries; add FULLTEXT where noted.

26.2 Core Platform Tables

26.2.1 `companies` (platform tenant)

- `id` BIGINT PK
- `name` VARCHAR(160) **unique**
- `slug` VARCHAR(160) **unique**
- `status` ENUM('pending','active','suspended')
- `region` VARCHAR(64)
- `owner_user_id` BIGINT FK→`users.id` NULL
- Usage: `rfqs_monthly_used` INT DEFAULT 0, `storage_used_mb` INT DEFAULT 0
- Billing: `stripe_id` VARCHAR(191) NULL, `plan_code` ENUM('starter','growth','enterprise') NULL
- `trial_ends_at` TIMESTAMP NULL
- Timestamps, Soft delete

Indexes: (`status`), (`plan_code`), (`owner_user_id`)

26.2.2 `users` (platform-wide)

- `id` BIGINT PK
- `company_id` BIGINT FK→`companies.id` NULL *(platform admins have NULL)*
- `name` VARCHAR(160)
- `email` VARCHAR(191) **unique**

- `password` VARCHAR(255)
- `role` ENUM('buyer_admin','buyer_requester','supplier_admin','supplier_estimator','finance','platform_super','platform_support')
- `last_login_at` TIMESTAMP NULL
- Timestamps, Soft delete

Indexes: (`company_id`,`role`)

26.2.3 `company_user` (optional multi-org membership)

- `id` BIGINT PK
- `company_id` BIGINT FK→`companies.id`
- `user_id` BIGINT FK→`users.id`
- `role` ENUM(…same as above…)
- Unique: (`company_id`,`user_id`)
- Timestamps

26.2.4 Billing (Laravel Cashier default)

Create via `php artisan vendor:publish --tag=cashier-migrations`
Includes `subscriptions`, `subscription_items`, etc.
Link at **company level** (Company uses `Billable` trait).
Add `company_id` to `subscriptions` if you switch to company-centric billing.

26.3 Directory & Supplier

26.3.1 `suppliers`

- `id` BIGINT PK
- `company_id` BIGINT FK *(buyer's tenant that manages directory entry; optional global)*
- `name` VARCHAR(160)
- `country` VARCHAR(2), `city` VARCHAR(120)
- `email` VARCHAR(191), `phone` VARCHAR(60)
- `website` VARCHAR(191) NULL
- `status` ENUM('pending','approved','rejected','suspended')
- `capabilities` JSON *(methods, materials, certs)*
- `rating_avg` DECIMAL(3,2) DEFAULT 0.00
- Timestamps, Soft delete

Indexes: (`company_id`,`status`), FULLTEXT(`name`,`city`,`capabilities`)

26.3.2 `supplier_documents`

- `id` BIGINT PK
- `supplier_id` BIGINT FK
- `type` ENUM('iso','tax','insurance','registration','other')
- `document_id` BIGINT FK→`documents.id`
- `expires_at` DATE NULL
- `status` ENUM('valid','expiring','expired')

- Timestamps

Indexes: (`supplier_id`,`type`), (`expires_at`)

26.4 Sourcing: RFQ / Quotes / Awards

26.4.1 `rfqs`

- `id` BIGINT PK
- `company_id` BIGINT FK *(buyer)*
- `created_by` BIGINT FK→`users.id`
- `title` VARCHAR(200)
- `type` ENUM('ready_made','manufacture')
- `material` VARCHAR(120) NULL
- `method` VARCHAR(120) NULL
- `tolerance_finish` VARCHAR(120) NULL
- `incoterm` VARCHAR(8) NULL
- `currency` CHAR(3) DEFAULT 'USD'
- `open_bidding` TINYINT(1) DEFAULT 0
- Lifecycle dates: `publish_at` DATETIME NULL, `due_at` DATETIME NULL, `close_at` DATETIME NULL
- `status` ENUM('draft','open','closed','awarded','cancelled') DEFAULT 'draft'
- `version` INT DEFAULT 1

- Timestamps, Soft delete

Indexes: (`company_id`,`status`,`due_at`), FULLTEXT(`title`)

26.4.2 `rfq_items`

- `id` BIGINT PK
- `rfq_id` BIGINT FK
- `line_no` INT
- `part_name` VARCHAR(160)
- `spec` TEXT NULL
- `quantity` INT
- `uom` VARCHAR(16) DEFAULT 'pcs'
- `target_price` DECIMAL(12,2) NULL
- Unique: (`rfq_id`,`line_no`)

26.4.3 `rfq_invitations`

- `id` BIGINT PK
- `rfq_id` BIGINT FK
- `supplier_id` BIGINT FK
- `invited_by` BIGINT FK→`users.id`
- `status` ENUM('invited','accepted','declined')

- Unique: (`rfq_id`,`supplier_id`)

26.4.4 `rfq_clarifications` *(Q&A / amendments)*

- `id` BIGINT PK
- `rfq_id` BIGINT FK
- `user_id` BIGINT FK
- `kind` ENUM('question','answer','amendment')
- `message` TEXT
- `attachment_id` BIGINT FK→`documents.id` NULL
- `rfq_version` INT *(increment when amendment)*
- Timestamps

Indexes: (`rfq_id`,`kind`)

26.4.5 `quotes`

- `id` BIGINT PK
- `company_id` BIGINT FK *(buyer for scoping)*
- `rfq_id` BIGINT FK
- `supplier_id` BIGINT FK
- `submitted_by` BIGINT FK→`users.id`

- Money: `currency` CHAR(3), `unit_price` DECIMAL(12,2), `min_order_qty` INT NULL
- `lead_time_days` INT
- `note` TEXT NULL
- `status` ENUM('draft','submitted','withdrawn','awarded','lost') DEFAULT 'submitted'
- `revision_no` INT DEFAULT 1
- Timestamps, Soft delete

Unique: (`rfq_id`,`supplier_id`,`revision_no`)
Indexes: (`rfq_id`,`supplier_id`,`status`)

26.4.6 `quote_items`

- `id` BIGINT PK
- `quote_id` BIGINT FK
- `rfq_item_id` BIGINT FK
- `unit_price` DECIMAL(12,2)
- `lead_time_days` INT
- `note` VARCHAR(255) NULL
- Unique: (`quote_id`,`rfq_item_id`)

26.5 Purchasing: PO / Change Orders / Orders

26.5.1 `purchase_orders`

- `id` BIGINT PK
- `company_id` BIGINT FK *(buyer)*
- `rfq_id` BIGINT FK
- `quote_id` BIGINT FK
- `po_number` VARCHAR(40) **unique**
- Commercials: `currency` CHAR(3), `incoterm` VARCHAR(8) NULL, `tax_percent` DECIMAL(5,2) NULL
- `status` ENUM('draft','sent','acknowledged','confirmed','cancelled') DEFAULT 'sent'
- `revision_no` INT DEFAULT 0
- `pdf_document_id` BIGINT FK→`documents.id` NULL
- Timestamps, Soft delete

Indexes: (`company_id`,`status`), (`rfq_id`,`quote_id`)

26.5.2 `po_lines`

- `id` BIGINT PK
- `purchase_order_id` BIGINT FK
- `rfq_item_id` BIGINT FK NULL *(for mapping to RFQ)*
- `line_no` INT
- `description` VARCHAR(200)
- `quantity` INT
- `uom` VARCHAR(16)

- `unit_price` DECIMAL(12,2)
- `delivery_date` DATE NULL
- Unique: (`purchase_order_id`,`line_no`)

26.5.3 `po_change_orders`

- `id` BIGINT PK
- `purchase_order_id` BIGINT FK
- `proposed_by_user_id` BIGINT FK
- `reason` VARCHAR(255)
- `changes_json` JSON *(list of field changes)*
- `status` ENUM('proposed','accepted','rejected') DEFAULT 'proposed'
- `po_revision_no` INT *(applied revision number when accepted)*
- Timestamps

26.5.4 `orders` (execution status)

- `id` BIGINT PK
- `company_id` BIGINT FK
- `purchase_order_id` BIGINT FK
- `supplier_id` BIGINT FK

- `status` ENUM('pending','in_production','in_transit','delivered','cancelled') DEFAULT 'pending'
- `tracking_number` VARCHAR(80) NULL
- `timeline` JSON *(array of {status, at, user_id, note})*
- Timestamps

Indexes: (`company_id`,`status`), (`supplier_id`)

26.6 Receiving & Quality

26.6.1 `grns` *(Goods Receipt Notes)*

- `id` BIGINT PK
- `company_id` BIGINT FK
- `purchase_order_id` BIGINT FK
- `received_by` BIGINT FK→`users.id`
- `received_at` DATETIME
- `note` VARCHAR(255) NULL
- Timestamps

26.6.2 `grn_lines`

- `id` BIGINT PK
- `grn_id` BIGINT FK
- `po_line_id` BIGINT FK

- `received_qty` INT
- `accepted_qty` INT
- `rejected_qty` INT
- `inspection_note` VARCHAR(255) NULL
- `ncr_flag` TINYINT(1) DEFAULT 0
- Unique: (`grn_id`,`po_line_id`)

26.6.3 `ncrs` *(Non-Conformance Reports)*

- `id` BIGINT PK
- `company_id` BIGINT FK
- `po_line_id` BIGINT FK
- `raised_by` BIGINT FK→`users.id`
- `status` ENUM('raised','in_review','corrective_action','verified','closed') DEFAULT 'raised'
- `reason` VARCHAR(255)
- `documents_json` JSON NULL
- Timestamps

26.7 Invoicing & Match

26.7.1 `invoices`

- `id` BIGINT PK
- `company_id` BIGINT FK
- `purchase_order_id` BIGINT FK
- `supplier_id` BIGINT FK
- `invoice_number` VARCHAR(60)
- `currency` CHAR(3)
- `subtotal` DECIMAL(12,2)
- `tax_amount` DECIMAL(12,2) DEFAULT 0
- `total` DECIMAL(12,2)
- `status` ENUM('pending','paid','overdue','disputed') DEFAULT 'pending'
- `document_id` BIGINT FK→`documents.id` NULL
- Timestamps, Soft delete

Indexes: (`company_id`,`status`), (`supplier_id`)

26.7.2 `invoice_lines`

- `id` BIGINT PK
- `invoice_id` BIGINT FK
- `po_line_id` BIGINT FK NULL
- `description` VARCHAR(200)
- `quantity` INT

- `uom` VARCHAR(16)
- `unit_price` DECIMAL(12,2)

26.7.3 `invoice_matches`

- `id` BIGINT PK
- `invoice_id` BIGINT FK
- `po_id` BIGINT FK
- `grn_id` BIGINT FK NULL
- `result` ENUM('matched','qty_mismatch','price_mismatch','unmatched')
- `details` JSON
- Timestamps

26.8 Documents & Media

26.8.1 `documents` *(polymorphic)*

- `id` BIGINT PK
- `company_id` BIGINT FK NULL *(platform docs can be NULL)*
- Polymorph: `documentable_type` VARCHAR(160), `documentable_id` BIGINT
- `kind` ENUM('rfq','quote','po','invoice','grn','ncr','supplier','template','other')
- `path` VARCHAR(255)

- `filename` VARCHAR(191)
- `mime` VARCHAR(120)
- `size_bytes` BIGINT
- `hash_sha256` CHAR(64)
- `version` INT DEFAULT 1
- Timestamps, Soft delete

Indexes: (`documentable_type`,`documentable_id`), (`company_id`,`kind`)

26.9 Notifications & Preferences

26.9.1 `notifications`

- `id` BIGINT PK
- `company_id` BIGINT FK
- `user_id` BIGINT FK
- `type` VARCHAR(120) *(event key)*
- `title` VARCHAR(160), `body` TEXT
- `entity_type` VARCHAR(160), `entity_id` BIGINT
- `channel` ENUM('push','email','both') DEFAULT 'both'
- `read_at` DATETIME NULL
- `meta` JSON
- Timestamps

26.9.2 `user_notification_prefs`

- `id` BIGINT PK
- `user_id` BIGINT FK
- `event_type` VARCHAR(120)
- `channel` ENUM('push','email','both','none') DEFAULT 'both'
- `digest` ENUM('none','daily','weekly') DEFAULT 'none'
- Unique: (`user_id`,`event_type`)

26.10 API & Webhooks

26.10.1 `api_keys`

- `id` BIGINT PK
- `company_id` BIGINT FK
- `name` VARCHAR(120)
- `token_hash` CHAR(64)
- `scopes` JSON
- `last_used_at` DATETIME NULL
- `revoked_at` DATETIME NULL
- `created_by` BIGINT FK→`users.id`
- Timestamps

26.10.2 `webhooks`

- `id` BIGINT PK
- `company_id` BIGINT FK
- `url` VARCHAR(255)
- `event_filters` JSON
- `secret` VARCHAR(191)
- `active` TINYINT(1) DEFAULT 1
- Timestamps

26.11 Analytics & Governance

26.11.1 `usage_snapshots`

- `id` BIGINT PK
- `company_id` BIGINT FK
- `date` DATE
- `rfqs_count` INT
- `quotes_count` INT
- `pos_count` INT
- `storage_used_mb` INT
- Unique: (`company_id`,`date`)

26.11.2 `audit_logs`

- `id` BIGINT PK
- `company_id` BIGINT FK NULL *(platform actions may be NULL)*
- `user_id` BIGINT FK NULL
- `entity_type` VARCHAR(160), `entity_id` BIGINT
- `action` ENUM('create','update','delete','award','acknowledge','status_change','login','impersonate')
- `before` JSON NULL, `after` JSON NULL
- `ip` VARCHAR(64), `ua` VARCHAR(255)
- `created_at` TIMESTAMP

Indexes: (`company_id`,`entity_type`,`entity_id`), (`action`,`created_at`)

26.11.3 `retention_holds`

- `id` BIGINT PK
- `company_id` BIGINT FK
- `entity_type` VARCHAR(160), `entity_id` BIGINT
- `reason` VARCHAR(255)
- `active` TINYINT(1) DEFAULT 1
- Timestamps
- Unique: (`company_id`,`entity_type`,`entity_id`,`active`)

26.11.4 `company_plan_overrides`

- `id` BIGINT PK
- `company_id` BIGINT FK
- `key` VARCHAR(80) *(e.g., rfqs_per_month, users_max, storage_gb)*
- `value` VARCHAR(80)
- `reason` VARCHAR(255)
- `created_by` BIGINT FK→`users.id`
- Timestamps
- Unique: (`company_id`,`key`)

26.12 Indexing & Search Guidance

- Always index (`company_id`, `status`) on transactional tables.
- Add FULLTEXT on `suppliers.name`, `suppliers.capabilities`, `rfqs.title`.
- Composite examples:
  - `quotes (rfq_id, supplier_id, status)`
  - `purchase_orders (company_id, status)`
  - `orders (supplier_id, status)`
  - `notifications (user_id, read_at)`
- For analytics speed, consider **materialized views** (or summary tables) for monthly counts.

26.13 Cascade & Referential Rules

- **ON DELETE CASCADE** from parent → children where safe:
  - `rfqs` → `rfq_items`, `rfq_invitations`, `rfq_clarifications`, `quotes` (or soft-delete preferred)
  - `quotes` → `quote_items`
  - `purchase_orders` → `po_lines`, `po_change_orders`, `orders`
  - `grns` → `grn_lines`
  - `invoices` → `invoice_lines`, `invoice_matches`
- **Restrict** when legal record must persist (e.g., invoices). Prefer **soft delete** for business entities.

26.14 Migration Order (generate in this sequence)

1. `companies`, `users`, `company_user`
2. Cashier billing tables
3. `suppliers`, `supplier_documents`, `documents`
4. `rfqs`, `rfq_items`, `rfq_invitations`, `rfq_clarifications`
5. `quotes`, `quote_items`
6. `purchase_orders`, `po_lines`, `po_change_orders`, `orders`
7. `grns`, `grn_lines`, `ncrs`

8. `invoices`, `invoice_lines`, `invoice_matches`
9. `notifications`, `user_notification_prefs`
10. `api_keys`, `webhooks`
11. `usage_snapshots`, `audit_logs`, `retention_holds`, `company_plan_overrides`

(Copilot: create factories/seeders immediately after each domain group.)

26.15 Seeders (minimum)

- `CompanySeeder` (1 demo tenant + plan_code)
- `UserSeeder` (buyer admin, supplier estimator, finance)
- `SupplierSeeder` (8 suppliers with capabilities JSON)
- `RfqSampleSeeder` (3 RFQs, items, invitations)
- `QuoteSampleSeeder` (quotes from 3 suppliers, with items)
- `PoSampleSeeder` (1 PO + lines + order timeline)
- `GrnInvoiceSeeder` (1 GRN with partial/accepted/rejected; 1 Invoice + match)
- `NotificationSeeder` (recent events)
- `ApiKeySeeder` (scoped key)

26.16 Acceptance Checks (DB)

- AC-DB-01: All FK constraints valid; inserts/updates fail on invalid FK.

- AC-DB-02: Multitenancy enforced — **no** row without correct `company_id` on tenant tables.
- AC-DB-03: Soft deletes present on RFQ/Quote/PO/Invoice/Supplier and cascade to children through application logic.
- AC-DB-04: Unique constraints prevent duplicates: `po_number`, (`rfq_id`,`line_no`), (`quote_id`,`rfq_item_id`), (`supplier_id`,`type` in docs), (`company_id`,`key` in overrides).
- AC-DB-05: Indexes exist for FK columns and composite lookups listed in §26.12.
- AC-DB-06: Document links valid; `documents.documentable_*` matches existing rows.
- AC-DB-07: Audit rows created on create/update/delete with non-empty `after` (and `before` when update/delete).
- AC-DB-08: Retention: records older than policy are marked archived or removed by job (test on sandbox clock).
- AC-DB-09: Cashier tables present; subscription for company can be created and queried.

26.17 Laravel Model Notes (for Copilot)

- Add `BelongsToCompany` trait to tenant models to auto-scope by `company_id`.
- Define relationships:
  - `Rfq hasMany RfqItem, hasMany Quote, hasMany RfqClarification`
  - `Quote belongsTo Rfq, belongsTo Supplier, hasMany QuoteItem`
  - `PurchaseOrder belongsTo Rfq, belongsTo Quote, hasMany PoLine, hasMany PoChangeOrder, hasOne Order`

    - `Invoice belongsTo PurchaseOrder, hasMany InvoiceLine, hasOne InvoiceMatch`
    - `Document morphTo documentable`
- Use `casts` for JSON fields (`capabilities`, `timeline`, `changes_json`, `meta`).

Approval

By approving via email, the client acknowledges that they have reviewed and agreed to the requirements outlined in this document. The client affirms that all details provided accurately reflect their needs and expectations, and they authorize the service provider to proceed with the project based on the specified requirements.