Step 62 - Phase 1: Data & Constraints Assessment

Existing Prisma Models Analysis

Models Available for Supplier Portal (NO SCHEMA CHANGES):

1. Supplier RFP List Data

- **Source Model**: `RFP` + `SupplierContact` + `SupplierResponse`
- **Join Strategy**:
- Find all `SupplierContact` records where `portalUserId` = current user ID
- For each, fetch the related `RFP` and `SupplierResponse`
- **Available Fields**:
- `rfpId`: RFP.id
- `title`: RFP.title
- `buyerCompanyName`: RFP.company.name (via relation)
- `stage`: RFP.stage (enum: INTAKE, QUALIFICATION, DISCOVERY, DRAFTING, PRICING_LEGAL_REVIEW, EXEC_REVIEW, SUBMISSION, DEBRIEF, ARCHIVED)
- `status`: RFP.status (string)
- `submissionDeadline`: RFP.submissionEnd
- `qnaEndDate`: RFP.askQuestionsEnd
- `demoWindowStart`: RFP.demoWindowStart
- `demoWindowEnd`: RFP.demoWindowEnd
- `supplierStatus`: Computed from SupplierContact.invitationStatus + SupplierResponse.status + SupplierResponse.submittedAt
- `outcomeStatus`: SupplierResponse.awardOutcomeStatus (“recommended” | “shortlisted” | “not_selected” | “declined”)
- **Computed Flags**:
- `hasPendingQuestions`: Count of SupplierQuestion where status = PENDING
- `hasPendingUploads`: Check if required attachments missing (best effort)
- `isOverdue`: Compare submissionEnd with current date and SupplierResponse.status

2. Supplier Requirements/Questions Data

- **Source Model**: `RFP.requirementGroups` (JSON field) + `SupplierResponse.structuredAnswers` (JSON field)
- **Strategy**:
- Parse RFP.requirementGroups to extract requirements list
- Match against SupplierResponse.structuredAnswers to determine answered/unanswered status
- Match against SupplierResponseAttachment to check for uploaded docs
- **Available Fields**:
- requirementId (from JSON structure)
- title/question text (from JSON structure)
- category/subcategory (from JSON structure)
- answered: boolean (check if key exists in structuredAnswers)

- hasUploadedDoc: boolean (check SupplierResponseAttachment count)
- lastUpdatedAt: SupplierResponse.updatedAt
- **CRITICAL**: Do NOT expose:
- autoScoreJson from SupplierResponse
- weights, scoringType from requirements
- Any AI reasoning or scoring metadata

3. Supplier Documents Data

- **Source Model**: `SupplierResponseAttachment`
- **Filter**: WHERE supplierResponseId = (current supplier's response)
- **Available Fields**:
- documentId: id
- fileName: fileName
- fileType: fileType
- fileSize: fileSize
- uploadedAt: createdAt
- uploadedBy: Derive from SupplierContact name/email via SupplierResponse relation
- attachmentType: attachmentType (enum)
- description: description

4. Supplier Submission Preview Data

- **Source Model**: `RFP` + `SupplierResponse` + `SupplierResponseAttachment`
- **Strategy**:
- Fetch RFP.requirementGroups (questions)
- Fetch SupplierResponse.structuredAnswers (answers)
- Fetch SupplierResponseAttachment list (documents)
- Combine into read-only preview format
- **CRITICAL**: Do NOT expose:
- autoScoreJson
- overrides (buyer overrides from Step 61)
- comments (buyer comments from Step 61)
- Any scoring, weighting, or internal metadata

5. Supplier Outcome Data

- **Source Model**: `SupplierResponse` + `RFP`
- **Available Fields**:
- outcomeStatus: SupplierResponse.awardOutcomeStatus
- outcomeDate: RFP.awardDecidedAt (if available)
- simpleOutcomeMessage: Generate from awardOutcomeStatus
- **Logic**:
- If awardOutcomeStatus = "recommended": "You have been recommended for this RFP"
- If awardOutcomeStatus = "shortlisted": "You have been shortlisted"
- If awardOutcomeStatus = "not_selected": "You were not selected for this RFP"
- If awardOutcomeStatus = "declined": "This opportunity was declined"
- If null: "Decision pending" or "In Review"
- **CRITICAL**: Do NOT expose:

- RFP.awardSnapshot (contains detailed scoring)
- RFP.decisionBriefSnapshot (buyer-internal decision brief)
- RFP.scoringMatrixSnapshot (full scoring matrix)
- RFP.comparisonNarrative (AI comparison)
- Other suppliers' data or rankings

6. Supplier Q&A Data

- **Source Model**: `SupplierQuestion` + `SupplierBroadcastMessage`
- **Strategy**:
- SupplierQuestion: Show only where supplierContactId = current supplier
- SupplierBroadcastMessage: Show all for this RFP (public broadcasts)
- **Available Fields** (SupplierQuestion):
- id
- question
- answer
- status (PENDING/ANSWERED)
- askedAt
- answeredAt
- **Available Fields** (SupplierBroadcastMessage):
- id
- message
- createdAt
- createdBy (user ID, can resolve to name if needed)

Data Constraints Summary

✅ SAFE TO EXPOSE (Supplier-Facing):

- RFP title, description (if not marked confidential), buyer company name
- Timeline dates (submissionEnd, askQuestionsEnd, demoWindow dates)
- RFP stage (mapped to user-friendly labels)
- Supplier's own invitation status, submission status, submission timestamp
- Supplier's own answers, documents, questions
- High-level outcome status (awarded/not selected/in review)
- Public broadcast messages

❌ NEVER EXPOSE (Buyer-Internal):

- `autoScoreJson` - AI/rule-based scoring results
- `overrides` - Buyer manual overrides
- `comments` - Buyer evaluation comments
- `scoringMatrixSnapshot` - Full scoring matrix
- `decisionBriefSnapshot` - Executive decision brief
- `comparisonNarrative` - AI comparison narrative
- `awardSnapshot` - Detailed award decision data
- `opportunityScore` - Buyer-internal opportunity scoring
- `internalNotes` - Any buyer notes

- Other suppliers’ names, data, or rankings
- Requirement weights, scoring types, must-have flags (can show labels but not internal metadata)

Implementation Strategy

Backend Endpoints (Phase 2):

1. Create supplier-scoped service functions in `lib/services/supplier-rfp.service.ts`
2. Implement strict role + identity checking in all endpoints
3. Use Prisma `select` to explicitly choose safe fields
4. Transform/filter JSON fields to remove internal metadata
5. Return 404 (not 403) for unauthorized access to avoid information leakage

Frontend Pages (Phases 3-4):

1. Reuse existing UI components where possible (cards, tables, tabs)
2. Create new supplier-specific components in `components/supplier/`
3. Integrate with existing Supplier Work Inbox from Step 54
4. Use server components for initial data fetch + auth checks
5. Use client components for tabs, filters, interactive elements

Activity Logging (Phase 5):

1. Add new event types to `lib/activity-types.ts`
2. Log all supplier portal access events
3. Include rfpId, supplierId, timestamp in all logs

Security Enforcement:

1. All endpoints: Check `session.user.role === 'supplier'`
2. All RFP access: Verify `SupplierContact.portalUserId === session.user.id`
3. All data transformations: Strip buyer-internal fields
4. All errors: Return generic 404 to avoid information leakage

Graceful Degradation Plan

If expected data is missing:
- **No requirementGroups**: Show empty state “No requirements defined”
- **No structuredAnswers**: Show all requirements as “Not Answered”
- **No attachments**: Show empty state “No documents uploaded”
- **No awardOutcomeStatus**: Show “Decision pending”
- **No timeline dates**: Show “Not specified”

Next Steps

Proceed to Phase 2: Implement backend endpoints with strict supplier scoping and data filtering.