STEP 39: Requirement-Level Scoring Matrix

Feature Type: Comparison & Analysis Enhancement
Priority: HIGH
Status: ✅ COMPLETE
Version: 1.0
Last Updated: December 2, 2025

Table of Contents

Overview

What is the Scoring Matrix?

The **Requirement-Level Scoring Matrix** is a comprehensive evaluation system that provides a granular, requirement-by-requirement view of how suppliers score across all RFP requirements. Unlike the high-level supplier comparison dashboard, this feature delivers:

- **Granular visibility**: See exactly which requirements each supplier passes, partially meets, or fails
- **Category filtering**: Filter by functional, commercial, legal, security, operational, or other categories
- **Differentiator analysis**: Identify requirements where suppliers differ most
- **Must-have compliance tracking**: Monitor critical requirement compliance
- **Export capabilities**: Export matrix data to CSV for offline analysis

Key Benefits

1. **Transparency**: Complete visibility into supplier compliance at the requirement level
2. **Risk mitigation**: Quickly identify suppliers failing critical requirements
3. **Data-driven decisions**: Make informed choices based on detailed compliance data
4. **Audit trail**: Export and archive compliance matrices for regulatory purposes
5. **Efficiency**: Filter and search through hundreds of requirements quickly

---

Architecture

System Components

```
┌──────────────────────────────────────────────────────────────┐
│                 REQUIREMENT-LEVEL SCORING MATRIX               │
└──────────────────────────────────────────────────────────────┘
                              │
        ┌─────────────────────┼─────────────────────┐
        │                     │                     │
        ▼                     ▼                     ▼
┌───────────────┐      ┌──────────────┐      ┌──────────────┐
│  Data Layer   │      │  API Layer   │      │  UI Layer    │
│               │      │              │      │              │
│ • Prisma      │      │ • GET Matrix │      │ • Matrix     │
│ • Types       │      │ • POST       │      │   View       │
│ • Scoring     │      │   Recompute  │      │ • Filters    │
│   Engine      │      │ • GET Export │      │ • Export     │
└───────────────┘      └──────────────┘      └──────────────┘
        │                     │                     │
        └─────────────────────┼─────────────────────┘
                              │
                              │
                  ┌───────────┴───────────┐
                  │                       │
                  ▼                       ▼
            ┌──────────────┐      ┌──────────────┐
            │ Decision     │      │ Demo         │
            │ Brief        │      │ Scenario     │
            │ Integration  │      │ Data         │
            └──────────────┘      └──────────────┘
```

Technology Stack

- **Database**: PostgreSQL with Prisma ORM
- **Backend**: Next.js API Routes (TypeScript)
- **Frontend**: React with Next.js App Router
- **Styling**: Tailwind CSS
- **Icons**: Lucide React
- **State Management**: React Hooks

Data Model

Database Schema

RFP Model (Updated)

```
model RFP {
  // ... existing fields ...

  // STEP 39: Requirement-Level Scoring Matrix
  scoringMatrixSnapshot Json?  // Cached scoring matrix with requirements vs suppliers
}
```

TypeScript Types

Core Types

```typescript
// Requirement categories
type RequirementCategoryId =
  | "functional"
  | "commercial"
  | "legal"
  | "security"
  | "operational"
  | "other";

// Importance levels
type RequirementImportance =
  | "must_have"
  | "should_have"
  | "nice_to_have";

// Score levels
type RequirementScoreLevel =
  | "pass"
  | "partial"
  | "fail"
  | "not_applicable"
  | "missing";
```

ScoringMatrixSnapshot

The complete scoring matrix snapshot stored in `RFP.scoringMatrixSnapshot`:

```typescript
interface ScoringMatrixSnapshot {
  rfpId: string;
  generatedAt: Date;
  generatedByUserId?: string;

  // All requirements in the matrix
  requirements: ScoringMatrixRequirement[];

  // All scoring cells (requirements × suppliers)
  cells: ScoringMatrixCell[];

  // Summary data for each supplier
  supplierSummaries: ScoringMatrixSupplierSummary[];

  // Configuration used for scoring
  scoringConfig: ScoringConfig;

  // Metadata
  meta: {
    totalRequirements: number;
    totalSuppliers: number;
    version: number;
  };
}
```

ScoringMatrixRequirement

Represents a single requirement (question or clause):

```
interface ScoringMatrixRequirement {
  requirementId: string;
  sourceType: "template_question" | "clause";
  referenceKey: string;
  shortLabel: string;
  longDescription: string;
  category: RequirementCategoryId;
  importance: RequirementImportance;
  defaultWeight: number;
}
```

ScoringMatrixCell

Represents how a specific supplier scores on a specific requirement:

```
interface ScoringMatrixCell {
  requirementId: string;
  supplierId: string;
  scoreLevel: RequirementScoreLevel;
  numericScore: number;
  justification?: string;
}
```

ScoringMatrixSupplierSummary

Aggregated scores for a single supplier:

```
interface ScoringMatrixSupplierSummary {
  supplierId: string;
  supplierName: string;
  overallScore: number;
  weightedScore: number;
  categoryScores: Array<{
    category: RequirementCategoryId;
    score: number;
    weightedScore: number;
  }>;
  mustHaveCompliance: {
    total: number;
    passed: number;
    failed: number;
  };
}
```

API Endpoints

1. GET Matrix

Endpoint: `GET /api/dashboard/rfps/[id]/comparison/matrix`

Description: Retrieves the scoring matrix for an RFP. Returns cached snapshot if available, otherwise computes on-the-fly.

Access: Buyer users only (must own RFP’s company)

Response:

```json
{
  "success": true,
  "matrix": {
    "rfpId": "...",
    "generatedAt": "2025-12-02T...",
    "requirements": [...],
    "cells": [...],
    "supplierSummaries": [...],
    "scoringConfig": {...},
    "meta": {...}
  }
}
```

Error Responses:

- `401 Unauthorized`: User not authenticated
- `403 Forbidden`: User not authorized (not a buyer or doesn't own RFP)
- `404 Not Found`: RFP not found
- `500 Internal Server Error`: Failed to generate matrix

2. POST Recompute

Endpoint: `POST /api/dashboard/rfps/[id]/comparison/matrix/recompute`

Description: Forces a recomputation of the scoring matrix. Useful when supplier responses or requirements change.

Access: Buyer users only (must own RFP's company)

Request Body (Optional):

```json
{
  "scoringConfigOverrides": {
    "defaultWeights": {
      "functional": 1.0,
      "commercial": 0.9
    }
  }
}
```

Response:

```json
{
  "success": true,
  "message": "Scoring matrix recomputed successfully",
  "matrix": {...}
}
```

Activity Log: Creates `comparison_matrix_recomputed` event

3. GET Export

Endpoint: `GET /api/dashboard/rfps/[id]/comparison/matrix/export`

Description: Exports the scoring matrix to CSV format with optional filters.

Access: Buyer users only (must own RFP's company)

Query Parameters:

- `category`: Filter by category (e.g., “functional”, “commercial”, “all”)
- `onlyDifferentiators`: `true` to show only requirements where suppliers differ
- `onlyFailedOrPartial`: `true` to show only failed or partial requirements
- `searchTerm`: Search term to filter requirements

Example:

```
GET /api/dashboard/rfps/[id]/comparison/matrix/export?category=functional&onlyDiffer-
entiators=true
```

Response: CSV file download

CSV Format:

```
"Requirement ID","Category","Importance","Short Label","Description","Supplier A -
Score","Supplier B - Score","Supplier A - Justification","Supplier B - Justification"
"REQ-001","functional","must_have","Multi-channel Sup-
port","...","pass","partial","Supports all channels","Social media in beta"
```

Activity Log: Creates `comparison_matrix_exported` event

UI Components

Matrix View Page

Location: `/app/dashboard/rfps/[id]/scoring-matrix/page.tsx`

Route: `/dashboard/rfps/[id]/scoring-matrix`

Features

1. **Tab Navigation**
   - **Matrix View**: Tabular view of requirements × suppliers
   - **Supplier Summaries**: Card-based summary view

2. **Filter Panel**
   - Category dropdown (All, Functional, Commercial, Legal, Security, Operational, Other)
   - Search bar for requirements
   - “Only Differentiators” toggle
   - “Only Failed/Partial” toggle
   - Reset filters button

3. **Action Buttons**
   - **Filters**: Toggle filter panel
   - **Export CSV**: Download filtered matrix as CSV
   - **Recompute**: Force matrix regeneration

4. **Matrix Table**
   - Sticky header and first column
   - Color-coded score badges

  - Requirement categories with icons
  - Importance badges (must-have, should-have, nice-to-have)
  - Hover tooltips for justifications

5. **Supplier Summary Cards**
  - Overall score (unweighted)
  - Weighted score (with importance factoring)
  - Must-have compliance indicators
  - Category breakdown charts

Score Level Styling

- **Pass**: Green badge with checkmark icon
- **Partial**: Amber badge with minus icon
- **Fail**: Red badge with X icon
- **Not Applicable**: Gray badge with question icon
- **Missing**: Light gray badge with alert icon

Category Icons

- **Functional**: TrendingUp icon
- **Commercial**: DollarSign icon
- **Legal**: FileText icon
- **Security**: Shield icon
- **Operational**: Settings icon
- **Other**: Briefcase icon

Integrations

1. Decision Brief Integration

The scoring matrix is integrated into the Decision Brief composer to provide requirement-level insights.

Location: `/lib/decision-brief/composer.ts`

New Field: `matrixSummary: DecisionBriefMatrixSummary | null`

MatrixSummary Structure:

```
interface DecisionBriefMatrixSummary {
  hasMatrix: boolean;
  totalRequirements: number;
  totalSuppliers: number;
  mustHaveComplianceBySupplier: Array<{
    supplierId: string;
    supplierName: string;
    passedCount: number;
    totalCount: number;
    compliancePercentage: number;
  }>;
  topDifferentiatorRequirements: string[];
}
```

Usage in Decision Brief:

- Shows must-have compliance for each supplier
- Highlights top 5 differentiator requirements
- Provides quick overview without full matrix detail

2. Portfolio Overview Integration

The scoring matrix can be referenced in portfolio-level analytics to show requirement compliance trends across multiple RFPs.

Future Enhancement: Aggregate requirement compliance across all RFPs in a portfolio to identify:
- Common failure patterns
- Supplier strengths/weaknesses across requirements
- Requirement categories that need improvement

Demo Mode

Demo Data

The demo scenario includes a precomputed scoring matrix with:

- **8 requirements** across functional, commercial, security, and operational categories
- **4 suppliers** (Acme, Northwind, Contoso, Fabrikam)
- **32 scoring cells** (8 requirements × 4 suppliers)
- **Realistic scoring patterns** showing varying compliance levels

Demo Suppliers:

1. **Acme Connect Solutions**: Perfect 100% score (all pass)
2. **Northwind Voice Systems**: 75% score (some partial, some fail)
3. **Contoso Cloud Communications**: Perfect 100% score (all pass)
4. **Fabrikam Unified Solutions**: 50% score (mix of pass, partial, fail)

Demo Requirements:

- Multi-channel Support (functional, must-have)
- AI-Powered Routing (functional, must-have)
- Real-time Analytics Dashboard (functional, should-have)
- Transparent Pricing Model (commercial, must-have)
- SOC 2 Type II Compliance (security, must-have)
- End-to-End Encryption (security, must-have)
- 99.99% Uptime SLA (operational, must-have)
- 24/7 Technical Support (operational, should-have)

Testing Demo:

1. Log in as demo buyer: `diane.demo@cloudstack.com` / `demo123`
2. Navigate to primary RFP: "Unified Communications & Contact Center RFP – 2025"
3. Click "Scoring Matrix" in navigation or go to `/dashboard/rfps/[id]/scoring-matrix`
4. Explore filters, export CSV, and recompute functionality

Security

Access Control

1. **Buyer-Only Access**: Only users with `role = "buyer"` can access scoring matrix
2. **Company Scoping**: Users can only view matrices for RFPs belonging to their company
3. **RFP Ownership Verification**: System verifies user created or owns the RFP

Data Protection

1. **Snapshot Caching**: Matrix snapshots are cached in database to prevent recalculation on every request
2. **Controlled Regeneration**: Only authenticated buyers can trigger recomputation
3. **Export Logging**: All exports are logged in activity log for audit trail

API Security

- All endpoints use Next.js `getServerSession` for authentication
- Role-based access control (RBAC) enforced at API level
- Company-scoped queries prevent cross-company data leakage

Usage Guide

For Buyers

Viewing the Scoring Matrix

1. Navigate to RFP detail page
2. Click **“Scoring Matrix”** in the navigation menu
3. View the matrix in either:
   - **Matrix View**: Full tabular view
   - **Supplier Summaries**: Card-based summary

Filtering Requirements

1. Click **“Filters”** button to open filter panel
2. Apply filters:
   - **Category**: Select specific category or “All”
   - **Search**: Type keywords to find specific requirements
   - **Only Differentiators**: Show only requirements where suppliers differ
   - **Only Failed/Partial**: Show only problematic requirements
3. Click **“Reset Filters”** to clear all filters

Exporting to CSV

1. Apply desired filters (optional)
2. Click **“Export CSV”** button
3. CSV file downloads with filtered data
4. Open in Excel, Google Sheets, or other spreadsheet software

Recomputing the Matrix

When to recompute:
- New supplier response submitted

- Template or clauses updated
- Scoring configuration changed

How to recompute:
1. Click **“Recompute”** button
2. Wait for regeneration (usually < 5 seconds)
3. Matrix updates with latest data

Understanding Scores

Score Levels:

- **Pass (✓)**: Supplier fully meets requirement
- **Partial (~)**: Supplier partially meets requirement (50% credit)
- **Fail (✗)**: Supplier does not meet requirement
- **Not Applicable (?)**: Requirement doesn’t apply to this supplier
- **Missing (!)**: No response data found

Importance Levels:

- **Must-Have**: Critical requirement (weight = 1.0)
- **Should-Have**: Important but not critical (weight = 0.6-0.9)
- **Nice-to-Have**: Optional requirement (weight < 0.6)

Weighted Scoring:

- **Overall Score**: Unweighted average (all requirements equal)
- **Weighted Score**: Considers importance and category weights
- **Must-Have Penalty**: -10 points per failed must-have requirement

Analyzing Supplier Compliance

Red Flags:

- Supplier with < 80% must-have compliance
- Multiple failed must-have requirements
- Low functional or security category scores

Differentiators:

- Requirements where only one supplier passes
- High-importance requirements with varied scores
- Category gaps between top suppliers

Making Decisions:

1. Filter to must-have requirements only
2. Identify suppliers failing critical requirements
3. Compare weighted scores (not just overall scores)
4. Review differentiator requirements
5. Export matrix for stakeholder review

Development Notes

Scoring Engine Logic

Requirement Extraction

1. Parse `appliedTemplateSnapshot` for template questions
2. Parse `appliedClausesSnapshot` for linked clauses
3. Map sections/clauses to categories
4. Assign importance based on weight or mandatory flag

Cell Scoring

1. Extract `extractedRequirementsCoverage` from supplier response
2. Match requirements by `requirementId` or `referenceKey`
3. Determine score level:
   - `fully_addressed` → pass (1.0)
   - `partially_addressed` → partial (0.5)
   - `not_applicable` → not_applicable (0.0)
   - Missing → missing (0.0)
4. Store justification from supplier response

Supplier Summary Calculation

1. **Overall Score**: Simple average of all numeric scores
2. **Weighted Score**: Average weighted by requirement importance and category
3. **Category Scores**: Scores grouped by category
4. **Must-Have Compliance**: Count of passed/failed must-have requirements

Default Scoring Config

```
{
  defaultWeights: {
    functional: 1.0,
    commercial: 0.9,
    legal: 0.95,
    security: 1.0,
    operational: 0.8,
    other: 0.6
  },
  mustHavePenalty: 10,
  partialFactor: 0.5
}
```

Performance Considerations

1. **Caching**: Matrix snapshots cached in database
2. **Lazy Loading**: Matrix computed only when requested
3. **Incremental Updates**: Consider incremental updates for large RFPs
4. **Pagination**: Future enhancement for RFPs with > 100 requirements

Error Handling

1. **Missing Data**: Returns null or empty arrays gracefully
2. **Parsing Errors**: Logs errors and continues with available data
3. **API Failures**: Returns 500 with error details

4. **Snapshot Corruption**: Falls back to recomputation

Testing

Unit Tests:
- Test requirement extraction logic
- Test cell scoring logic
- Test supplier summary calculations
- Test filter application

Integration Tests:
- Test API endpoints with mock data
- Test authentication and authorization
- Test CSV export formatting

E2E Tests:
- Test full user flow (view → filter → export → recompute)
- Test with demo data
- Test error states

Future Enhancements

Phase 2 (Planned)

1. **Custom Weighting UI**: Allow buyers to adjust category weights
2. **Requirement Grouping**: Group requirements by section/subsection
3. **Heatmap Visualization**: Color-coded heatmap view
4. **Comparison Mode**: Side-by-side comparison of 2-3 suppliers
5. **Requirement Notes**: Add buyer notes to specific requirements
6. **Historical Tracking**: Track requirement compliance over time

Phase 3 (Future)

1. **AI-Generated Insights**: Automatic identification of gaps and risks
2. **Benchmarking**: Compare supplier scores against industry benchmarks
3. **What-If Analysis**: Simulate score changes with different weights
4. **Mobile View**: Responsive mobile-optimized matrix view
5. **Collaborative Review**: Multi-user review and commenting
6. **PDF Export**: Generate formatted PDF reports

Changelog

Version 1.0 (December 2, 2025)

Initial Release:

- ✅ Database schema update (RFP.scoringMatrixSnapshot)
- ✅ TypeScript types for matrix data structures
- ✅ Scoring engine with 3 core functions

- ✅ API endpoints (GET matrix, POST recompute, GET export)
- ✅ UI components (matrix view, filters, export)
- ✅ Decision brief integration
- ✅ Demo mode with precomputed snapshot
- ✅ Activity log integration
- ✅ Comprehensive documentation

Support

For questions or issues:

1. **Documentation**: Review this guide
2. **Demo Mode**: Test with demo account
3. **Code Comments**: Review inline code documentation
4. **Activity Log**: Check activity log for debugging

License

End of Documentation