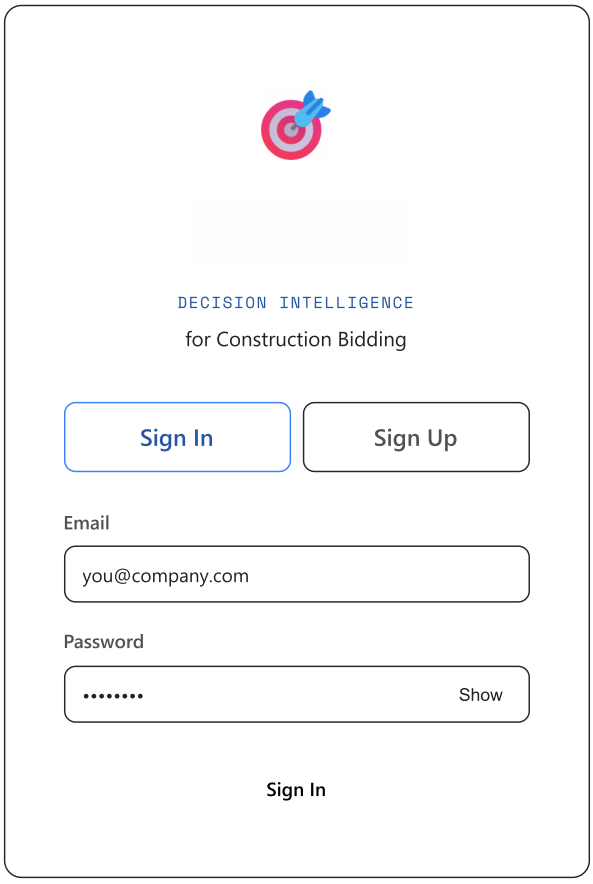
DECISION INTELLIGENCE
for Construction Bidding
Sign In
Sign Up
Email
you@company.com
Password
••••••••
Show
Sign In

`; if (analysis.scores.components.location.needsAddress) { const loc = document.querySelector('.score-component'); if (loc) loc.innerHTML += `

📍 Enter project address for accurate distance

123 Main St, City, State

Calculate

`; } } function renderComponent(icon, title, comp) { const fillClass = comp.score >= 70 ? 'high' : comp.score >= 50 ? 'medium' : 'low'; let details = ''; // Location details if (title === 'Location Fit' && comp.details?.dist !== undefined) { const dist = comp.details.dist; let explanation = ''; if (dist <= 25) explanation = 'Excellent - very close to your office'; else if (dist <= 50) explanation = 'Good - within easy driving distance'; else if (dist <= 100) explanation = 'Moderate - may require extra travel planning'; else if (dist <= 150) explanation = 'Far - significant travel required'; else explanation = 'Very far - consider travel costs carefully'; details = `

This job is ${dist} miles from your office.
${explanation}

`; } else if (title === 'Location Fit' && comp.needsAddress) { details = `

Could not determine project location from documents. Enter address below for accurate scoring.

`; } else if (title === 'Location Fit' && !comp.details?.dist) { details = `

${comp.reason}

`; } // Keywords details if (title === 'Keywords & Contract' && comp.details?.length > 0) { const good = comp.details.filter(d => d.type === 'good'); const bad = comp.details.filter(d => d.type === 'bad'); details = '

'; if (good.length) details += '
✓ Good terms found:

' + good.map(d => ` ${d.keyword} (${d.effect}) p.${d.pages.slice(0, 2).join(',')} `).join(''); if (bad.length) details += '

⚠ Risk terms found:

' + bad.map(d => ` ${d.keyword} (${d.effect}) p.${d.pages.slice(0, 2).join(',')} `).join(''); if (!comp.hasContract) details += '

ℹ️ No contract language detected - risk terms not penalized
'; details += '

'; } else if (title === 'Keywords & Contract' && (!comp.details || comp.details.length === 0)) { details = `

No keywords found. Add keywords in the Keywords tab to track terms that matter to you.

`; } // GC details if (title === 'GC Relationship' && comp.details?.bestGC) { const b = comp.details.bestGC; details = `

Best relationship: ${b.name} ${b.rate !== null ? `(${b.rate}% win rate)` : `(${b.rating}★ rating)`}${comp.details.compPenalty ? `
Competition penalty: -${comp.details.compPenalty} points (${comp.details.gcCount} GCs bidding)` : ''}

`; } // Trade details if (title === 'Trade Match' && comp.details?.found) { const foundNames = comp.details.found.map(code => { const div = CSI_DIVISIONS.find(d => d[0] === code); return div ? `Div ${code} (${div[1]})` : `Div ${code}`; }); details = `

Found: ${foundNames.length > 0 ? foundNames.join(', ') : 'None of your divisions found in specs - verify this project includes your trade'}

`; } return `

${icon} ${title}** (${comp.weight}% weight) **`${comp.score}/100`

${comp.reason}

${details}

`; } // Generate improvement tips based on scores function renderImprovementTips(scores) { const tips = []; const c = scores.components; // Location tips if (c.location.score < 70 && c.location.weight > 0) { if (c.location.details?.dist > 100) { tips.push('📍 **Location:** This project is far from your office. Consider if travel costs are worth it, or focus on closer opportunities.'); } else if (c.location.needsAddress) { tips.push('📍 **Location:** Enter the project address above to get an accurate distance score.'); } } // Keywords tips if (c.keywords.score < 60) { if (c.keywords.details?.badCount > 0) { tips.push('🔑 **Contract Risk:** Review the flagged contract terms carefully. Consider negotiating these clauses before signing.'); } if (c.keywords.details?.goodCount === 0) { tips.push('🔑 **Keywords:** None of your preferred terms were found. Add more "good keywords" in Settings to better identify ideal projects.'); } } // GC tips if (c.gc.score < 60) { if (c.gc.details?.gcCount > 5) { tips.push('🏢 **Competition:** ' + c.gc.details.gcCount + ' GCs bidding creates price pressure. Focus on negotiated or invited bids for better margins.'); } if (c.gc.details?.bestGC && c.gc.details.bestGC.rate !== null && c.gc.details.bestGC.rate < 20) { tips.push('🏢 **GC History:** Your win rate with ' + c.gc.details.bestGC.name + ' is low. Consider building the relationship before bidding more work.'); } if (!c.gc.details?.bestGC || c.gc.details.bestGC.rating <= 2) { tips.push('🏢 **GC Relationship:** You have no strong relationship with these GCs. Research their reputation before investing time.'); } } // Trade tips if (c.trade.score < 70) { tips.push('🔧 **Trade Match:** Not all your divisions were found in specs. Verify your scope is actually included before bidding.'); } if (tips.length === 0) { if (scores.final >= 80) { return '

> ✓ Strong Opportunity
>
> This bid aligns well with your business. Prioritize it and submit a competitive price.

'; } return ''; } return `

> 💡 How to Improve Your Chances
>
> ${tips.map(t => '
- ' + t + '
').join('')}

`; } // Print report async function printReport() { if (!currentAnalysis) return; const a = currentAnalysis; // Get user company info const settings = await getSettings(); const companyName = currentUser?.user_metadata?.company_name || settings.company_name || 'Your Company'; // Generate report ID const reportId = `BR-${Date.now().toString(36).toUpperCase()}`; const reportDate = new Date().toLocaleDateString('en-US', { year: 'numeric', month: 'long', day: 'numeric' }); // Clean project location const location = [a.extracted.project_city, a.extracted.project_state].filter(Boolean).join(', ') || 'Location TBD'; // Building type const buildingType = a.buildingType?.label || 'Not specified'; const buildingIcon = a.buildingType?.icon || '🏗️'; // Bid deadline const deadline = a.extracted.bid_deadline || 'Not specified'; // Score color and badge const scoreColor = a.scores.recommendation === 'GO' ? '#22c55e' : a.scores.recommendation === 'REVIEW' ? '#f59e0b' : '#ef4444'; const badgeText = { GO: 'STRONG FIT', REVIEW: 'REVIEW CAREFULLY', PASS: 'WEAK FIT' }[a.scores.recommendation]; const badgeBg = { GO: '#dcfce7', REVIEW: '#fef3c7', PASS: '#fee2e2' }[a.scores.recommendation]; // Good keywords const goodKeywords = (a.goodFound || []).map(k => k.keyword).filter(Boolean); // Risk keywords from contract risks const riskKeywords = (a.contractRisks?.risksDetected || []).map(r => { const labels = { pay_if_paid: 'Pay-if-Paid', liquidated_damages: 'Liquidated Damages', indemnification: 'Broad Indemnification', no_damages_delay: 'No Damages for Delay', consequential_waiver: 'Consequential Waiver', high_retainage: 'High Retainage', slow_payment: 'Slow Payment', termination_convenience: 'Termination for Convenience', excessive_warranty: 'Excessive Warranty', insurance_requirements: 'High Insurance' }; return labels[r.type] || r.type; }); // AI Analysis Summary (plain English) const summaryParts = []; summaryParts.push(`This ${buildingType.toLowerCase()} project is located ${a.scores.components.location.details?.dist || 'an unknown distance'} miles from your office.`); if (a.scores.components.location.score >= 80) { summaryParts.push('The

location is within your preferred service area.'); } else if (a.scores.components.location.score >= 60) { summaryParts.push('The location is at the edge of your service area.'); } else { summaryParts.push('The location may be outside your typical service area.'); } if (goodKeywords.length > 0) { summaryParts.push(`We found ${goodKeywords.length} favorable term${goodKeywords.length !== 1 ? 's' : ''} in the bid documents.`); } if (riskKeywords.length > 0) { summaryParts.push(`However, ${riskKeywords.length} contract risk${riskKeywords.length !== 1 ? 's were' : ' was'} identified that may impact your decision.`); } if (a.gcs && a.gcs.length > 0) { const knownGCs = a.gcs.filter(gc => gc.rating >= 3); if (knownGCs.length > 0) { summaryParts.push(`You have good relationships with ${knownGCs.length} of the ${a.gcs.length} general contractor${a.gcs.length !== 1 ? 's' : ''} bidding.`); } else { summaryParts.push(`You have limited history with the ${a.gcs.length} general contractor${a.gcs.length !== 1 ? 's' : ''} on this project.`); } } const aiSummary = summaryParts.join(' '); // How to Improve Your Chances const improvements = []; if (a.scores.components.location.score < 80) { improvements.push('Consider factoring in travel time and logistics costs for this location.'); } if (a.scores.components.gc.score < 70) { improvements.push('Reach out to the GCs early to build rapport and clarify scope questions.'); } if (riskKeywords.length > 0) { improvements.push('Have your attorney review the contract terms, especially the identified risk clauses.'); } if (a.scores.components.keywords.score < 70) { improvements.push('Request clarification on any ambiguous scope items before submitting your bid.'); } if (a.scores.components.trade.score < 80) { improvements.push('Highlight your relevant experience in similar projects to strengthen your bid.'); } if (improvements.length === 0) { improvements.push('Your profile is a strong match. Submit a competitive price and emphasize your reliability.'); improvements.push('Follow up with the GC after submission to demonstrate your interest.'); } const printContent = `

${companyName}

Report ID: ${reportId}

${reportDate}

BidIntell Analysis Report

AI-Powered Bid Evaluation

${a.extracted.project_name || 'Untitled Project'}

Location: ${location}

Building Type: ${buildingIcon} ${buildingType}

Bid Deadline: ${deadline}

GCs Bidding: ${a.gcs?.length || 0}

BIDINDEX SCORE

${a.scores.final}

${badgeText}

Score Breakdown

Component	Weight	Score	Analysis
📍 Location Fit	${a.scores.components.location.weight}%	**${a.scores.components.location.score}**	${a.scores.components.location
🔑 Keywords & Contract	${a.scores.components.keywords.weight}%	**${a.scores.components.keywords.score}**	${a.scores.components.keyworc
🏢 GC Relationship	${a.scores.components.gc.weight}%	**${a.scores.components.gc.score}**	${a.scores.components.gc.reaso
🔧 Trade Match	${a.scores.components.trade.weight}%	**${a.scores.components.trade.score}**	${a.scores.components.trade.rea

Keywords Detected

✓ Favorable Terms (${goodKeywords.length})

${goodKeywords.length > 0 ? goodKeywords.map(kw => ` ${kw} `).join('') : 'None detected' }

⚠ Risk Terms (${riskKeywords.length})

${riskKeywords.length > 0 ? riskKeywords.map(kw => ` ${kw} `).join('') : 'None detected' }

AI Analysis Summary

${aiSummary}

How to Improve Your Chances

${improvements.map(tip => `

✓ ${tip}

`).join('')}

${companyName}

Powered by BidIntell™ | Generated ${new Date().toLocaleString()}

Report ID: ${reportId} | ${a.files?.length || 0} document${(a.files?.length || 0) !== 1 ? 's' : ''} analyzed

`; const printWindow = window.open('', '_blank'); printWindow.document.write(printContent); printWindow.document.close(); printWindow.print(); } async function recalcAddress() { const addr = document.getElementById('manualAddress')?.value.trim(); if (!addr) return; const settings = await getSettings(); const userCoords = await geocode(`${settings.city}, ${settings.state}`); const projCoords = await geocode(addr); if (userCoords && projCoords) { const dist = Math.round(haversine(userCoords.lat, userCoords.lon, projCoords.lat, projCoords.lon)); let score = dist <= 25 ? 100 : dist <= 50 ? 85 : dist <= 100 ? 70 : dist <= 150 ? 50 : 30; if (dist > settings.radius) score = Math.max(10, score - 20); currentAnalysis.scores.components.location = { score, weight: currentAnalysis.scores.components.location.weight, reason: `${dist} miles from your office${dist > settings.radius ? ' (outside service area)' : ''}`, details: { dist } }; let final = 0; for (const k of ['location', 'keywords', 'gc', 'trade']) { const c = currentAnalysis.scores.components[k]; if (c.weight > 0) final += c.score * (c.weight / 100); } currentAnalysis.scores.final = Math.round(final); currentAnalysis.scores.recommendation =

```
final >= 80 ? 'GO' : final >= 60 ? 'REVIEW' : 'PASS'; currentAnalysis.extracted.project_address = addr; renderResult(currentAnalysis); } else alert('Could not geocode that address. Try a different format.'); } async function saveProjectToDb() { if (!currentAnalysis) return; // Check if user is authenticated if (!supabaseClient || !currentUser) { alert('❌ Not logged in. Please sign in to save projects.'); return; } // Capture user feedback const agreement = document.querySelector('input[name="userAgreement"]:checked')?.value || 'agree'; const overrideNote = document.getElementById('overrideNote')?.value?.trim() || null; currentAnalysis.userFeedback = { agreement, note: overrideNote, timestamp: new Date().toISOString() }; // Show saving indicator const saveBtn = event.target; const originalText = saveBtn.innerHTML; saveBtn.disabled = true; saveBtn.innerHTML = '💾 Saving...'; try { // Save project to database const savedProject = await saveProject({ ...currentAnalysis, outcome: 'pending' }); if (!savedProject || !savedProject.id) { throw new Error('Failed to save project - no ID returned'); } console.log('✅ Project saved:', savedProject.id); // Update GC bid counts const gcs = await getGCs(); for (const sg of currentAnalysis.gcs) { const gc = gcs.find(g => g.name === sg.name); if (gc) { gc.bids = (gc.bids || 0) + 1; await saveGC(gc); } } // Show success message saveBtn.innerHTML = '✅ Saved!'; setTimeout(() => { resetAnalysis(); switchTab('projects'); loadAll(); }, 1000); } catch (error) { console.error('❌ Save failed:', error); saveBtn.disabled = false; saveBtn.innerHTML = originalText; alert('Failed to save project: ' + error.message + '\n\nCheck browser console for details.'); } } async function resetAnalysis() { uploadedFiles = []; selectedGCs = []; currentAnalysis = null; document.getElementById('fileList').innerHTML = ''; document.getElementById('selectedGCsList').innerHTML = ''; document.getElementById('competitionIndicator').innerHTML = ''; document.getElementById('analysisResult').innerHTML = ''; document.getElementById('gcSearchInput').value = ''; fileInput.value = ''; updateAnalyzeBtn(); await renderGCSelector(); } // Dashboard // Animate number counting function animateValue(elementId, start, end, duration = 600, suffix = '') { const element = document.getElementById(elementId); if (!element) return; const range = end - start; const increment = range / (duration / 16); let current = start; const timer = setInterval(() => { current += increment; if ((increment > 0 && current >= end) || (increment < 0 && current <= end)) { element.textContent = Math.round(end) + suffix; clearInterval(timer); } else { element.textContent = Math.round(current) + suffix; } }, 16); } async function loadDashboard() { const projects = await getProjects(); const settings = await getSettings(); // Total bids analyzed animateValue('statBids', 0, projects.length); // Bids this week const oneWeekAgo = new Date(); oneWeekAgo.setDate(oneWeekAgo.getDate() - 7); const bidsThisWeek = projects.filter(p => new Date(p.created_at) >= oneWeekAgo).length; animateValue('statBidsThisWeek', 0, bidsThisWeek); // Win rate const won = projects.filter(p => p.outcome === 'won').length; const lost = projects.filter(p => p.outcome === 'lost').length; const winRate = (won + lost) > 0 ? Math.round((won / (won + lost)) * 100) : 0; document.getElementById('statWinRate').textContent = winRate > 0 ? winRate + '%' : '--'; document.getElementById('statWinCount').textContent = won; document.getElementById('statLossCount').textContent = lost; // Average score const projectsWithScores = projects.filter(p => p.scores && p.scores.final); const avgScore = projectsWithScores.length > 0 ? Math.round(projectsWithScores.reduce((sum, p) => sum + p.scores.final, 0) / projectsWithScores.length) : 0; document.getElementById('statAvgScore').textContent = avgScore > 0 ? avgScore : '--'; const highScoreCount = projectsWithScores.filter(p => p.scores.final >= 80).length; document.getElementById('statHighScoreCount').textContent = highScoreCount; // Hours saved const hoursSaved = Math.round(projects.length * settings.decisionTime / 60); animateValue('statHours', 0, hoursSaved); document.getElementById('statDecisions').textContent = projects.length; // Contract risks detected let totalRisks = 0; let riskyProjects = 0; projects.forEach(p => { if (p.contract_risks && p.contract_risks.risksDetected && p.contract_risks.risksDetected.length > 0) { totalRisks += p.contract_risks.risksDetected.length; riskyProjects++; } }); animateValue('statContractRisks', 0, totalRisks); document.getElementById('statRiskyProjects').textContent = riskyProjects; // Data moat health (% of projects with outcomes) const projectsWithOutcomes = projects.filter(p => p.outcome && p.outcome !== 'pending').length; const outcomeRate = projects.length > 0 ? Math.round((projectsWithOutcomes / projects.length) * 100) : 0; document.getElementById('statDataMoat').textContent = outcomeRate > 0 ? outcomeRate + '%' : '--'; document.getElementById('statOutcomeRate').textContent = outcomeRate; const
```

recent = projects.slice(0, 5); if (recent.length === 0) { document.getElementById('recentActivity').innerHTML = '

No bids analyzed yet

Upload your first bid documents to get started

Analyze Your First Bid

'; } else { document.getElementById('recentActivity').innerHTML = `${recent.map(p => ``).join('')}

PROJECT	SCORE	OUTCOME	
${p.extracted.project_name || 'Untitled'} ${p.extracted.project_city \|\| ''}${p.extracted.project_city && p.extracted.project_state ? ', ' : ''}${p.extracted.project_state || ''}	**${p.scores.final}** **${p.scores.recommendation}**	${P.Outcome}	View

`; } await updateCapacity(); } async function updateCapacity() { const settings = await getSettings(); document.getElementById('capacityIndicator').className = 'capacity-box capacity-' + settings.capacity; document.getElementById('capacityText').textContent = settings.capacity.charAt(0).toUpperCase() + settings.capacity.slice(1); } // Projects async function loadProjects() { const projects = await getProjects(); if (projects.length === 0) { document.getElementById('projectsList').innerHTML = '

No projects yet

Analyze your first bid to see it here

'; return; } document.getElementById('projectsList').innerHTML = `${projects.map(p => ``).join('')}

PROJECT	GCS	SCORE	OUTCOME	ACTIONS
${p.extracted.project_name || 'Untitled'} ${p.extracted.project_city \|\| ''}${p.extracted.project_city && p.extracted.project_state ? ', ' : ''}${p.extracted.project_state || ''}	${p.gcs.slice(0, 2).map(g => g.name.split(' ')[0]).join(', ')}${p.gcs.length > 2 ? '...' : ''}	**${p.scores.final}**	${P.Outcome}	View Outcome ×

`; } async function viewReport(id) { const projects = await getProjects(); const p = projects.find(x => x.id === id); if (!p) return; document.getElementById('reportModalTitle').textContent = p.extracted.project_name || 'Bid Report'; document.getElementById('reportModalBody').innerHTML = `

${p.scores.final}

${p.scores.recommendation}

${renderComponent('📍', 'Location Fit', p.scores.components.location)}${renderComponent('🔑', 'Keywords & Contract', p.scores.components.keywords)}${renderComponent('🏢', 'GC Relationship',

p.scores.components.gc)}${renderComponent(' ', 'Trade Match', p.scores.components.trade)}

`; openModal('reportModal'); } function showOutcome(id) {
document.getElementById('outcomeProjectId').value = id; document.getElementById('outcomeType').value =
''; document.getElementById('outcomeFields').innerHTML = ''; openModal('outcomeModal'); } function
updateOutcomeFields() { const type = document.getElementById('outcomeType').value; let html = ''; if (type
=== 'won') { html = `

Contract Amount ($)

500000

Final Margin (%)

12.5

GC Responsiveness

☐ Acknowledged receipt ☐ Answered pre-bid questions

Decision Confidence

${[1,2,3,4,5].map(n => ` **${n}** `).join('')}

1 = Lucky win, 5 = Expected to win

`; } else if (type === 'lost') { html = `

How high were you?

Unknown

Who won?

Competitor name

Other competitors you usually lose to? (optional)

e.g., ABC Electric, XYZ Mechanical

Separate names with commas

GC Responsiveness

☐ Acknowledged receipt ☐ Answered pre-bid questions

Decision Confidence

${[1,2,3,4,5].map(n => ` **${n}** `).join('')}

1 = Surprised we lost, 5 = Expected to lose

`; } else if (type === 'ghost') { html = `

GC Responsiveness Before Going Silent

☐ Acknowledged receipt ☐ Answered pre-bid questions

Days since bid submission

60

Decision Confidence

${[1,2,3,4,5].map(n => ` **${n}** `).join('')}

1 = Should have followed up more, 5 = Expected this response

`; } else if (type === 'declined') { html = `

Why didn't you bid? (select all that apply)

- [ ] Too many GCs bidding
- [ ] Weak GC relationship
- [ ] Bad contract terms
- [ ] Out of territory
- [ ] At capacity / too busy
- [ ] Unlikely to win on price
- [ ] Scope unclear
- [ ] Other

Additional notes (optional)

Any other context...

Decision Confidence

${[1,2,3,4,5].map(n => ` **${n}** `).join('')}

1 = Maybe should have bid, 5 = Definitely right to pass

`; } document.getElementById('outcomeFields').innerHTML = html; // Initialize confidence to 3 setTimeout(() => setConfidence(3), 10); } function setConfidence(val) { document.getElementById('outcomeConfidence').value = val; document.querySelectorAll('.conf-btn').forEach(btn => { btn.style.background = parseInt(btn.dataset.val) <= val ? 'var(--accent)' : 'var(--bg-secondary)'; btn.style.color = parseInt(btn.dataset.val) <= val ? 'white' : 'var(--text-primary)'; btn.style.borderColor = parseInt(btn.dataset.val) <= val ? 'var(--accent)' : 'var(--border-color)'; }); } async function saveOutcome() { const id = document.getElementById('outcomeProjectId').value; const type = document.getElementById('outcomeType').value; if (!type) { alert('Please select an outcome'); return; } let outcomeData = { gcAcknowledged: document.getElementById('gcAcknowledged')?.checked || false, gcAnsweredQuestions: document.getElementById('gcAnsweredQuestions')?.checked || false, confidence: parseInt(document.getElementById('outcomeConfidence')?.value) || 3 }; if (type === 'won') { outcomeData.amount = document.getElementById('outcomeAmount')?.value; outcomeData.margin = document.getElementById('outcomeMargin')?.value; // Update GC wins const projects = await getProjects(); const p = projects.find(x => x.id === id); if (p) { const gcs = await getGCs(); for (const sg of p.gcs) { const gc = gcs.find(g => g.name === sg.name); if (gc) { gc.wins = (gc.wins || 0) + 1; await saveGC(gc); } } } } else if (type === 'lost') { outcomeData.howHigh = document.getElementById('outcomeHowHigh')?.value; outcomeData.winner = document.getElementById('outcomeWinner')?.value; outcomeData.otherCompetitors = document.getElementById('outcomeOtherCompetitors')?.value?.split(',').map(s => s.trim()).filter(Boolean) || []; } else if (type === 'ghost') { outcomeData.daysSince = parseInt(document.getElementById('outcomeDaysSince')?.value) || 60; } else if (type === 'declined') { outcomeData.reasons = Array.from(document.querySelectorAll('.reason-checkbox input:checked')).map(cb => cb.value); outcomeData.notes = document.getElementById('outcomeDeclineNotes')?.value; if (outcomeData.reasons.length === 0) { alert('Please select at least one reason for passing'); return; } } await updateProjectOutcome(id, type, outcomeData); closeModal('outcomeModal'); await loadAll(); } async function deleteProjectUI(id) { if (!confirm('Delete this project?')) return; await deleteProject(id); await loadAll(); } async function exportCSV() { const projects = await getProjects(); if (!projects.length) { alert('No projects to export'); return; } const rows = [['Project', 'City', 'State', 'Score', 'Recommendation', 'GCs', 'Outcome', 'Date']]; for (const p of projects) rows.push([p.extracted.project_name || '', p.extracted.project_city || '', p.extracted.project_state || '', p.scores.final, p.scores.recommendation, p.gcs.map(g => g.name).join('; '), p.outcome, p.createdAt?.split('T')

[0] || '']); const csv = rows.map(r => r.map(c => `"${String(c).replace(/"/g, '""')}"`).join(',')).join('\n'); const a = document.createElement('a'); a.href = 'data:text/csv;charset=utf-8,' + encodeURIComponent(csv); a.download = 'bidiq_export.csv'; a.click(); } // GC Database const RISK_TAG_LABELS = { slow_pay: '💰 Slow pay', pay_if_paid: '📋 Pay-if-paid', change_order_hostile: '⚠️ CO hostile', bid_shopping: '🛒 Bid shopping', low_feedback: '🔇 Low feedback', scope_creep: '📈 Scope creep' }; async function loadGCDatabase() { const gcs = await getGCs(); if (!gcs.length) { document.getElementById('gcDatabaseList').innerHTML = '

No GCs yet

Add your first general contractor

+ Add GC

'; return; } document.getElementById('gcDatabaseList').innerHTML = gcs.map(gc => { const rate = gc.bids > 0 ? Math.round(gc.wins / gc.bids * 100) : null; const tagsHtml = (gc.riskTags || []).length > 0 ? ` ${gc.riskTags.map(t => ` ${RISK_TAG_LABELS[t] || t} `).join('')} ` : ''; return ` ${gc.name} ${gc.bids || 0} bids${rate !== null ? ' • ' + rate + '% win rate' : ''} ${[1, 2, 3, 4, 5].map(i => `★`).join('')} Tags × ${tagsHtml} `; }).join(''); } async function editGCTags(name) { const gcs = await getGCs(); const gc = gcs.find(g => g.name === name); if (!gc) return; const currentTags = gc.riskTags || []; const tagOptions = Object.entries(RISK_TAG_LABELS).map(([val, label]) => ` ☐ ${label} `).join(''); const modal = document.createElement('div'); modal.className = 'modal-overlay active'; modal.id = 'editTagsModal'; modal.innerHTML = ` Edit Risk Tags: ${name} × ${tagOptions} Cancel Save Tags `; document.body.appendChild(modal); } async function saveGCTags(name) { const gcs = await getGCs(); const gc = gcs.find(g => g.name === name); if (!gc) return; gc.riskTags = Array.from(document.querySelectorAll('.editGCTag:checked')).map(cb => cb.value); await saveGC(gc); document.getElementById('editTagsModal')?.remove(); await loadGCDatabase(); } async function showAddGCModal() { document.getElementById('newGCName').value = ''; document.getElementById('newGCLocation').value = ''; // Clear risk tags document.querySelectorAll('.gcRiskTag').forEach(cb => cb.checked = false); const settings = await getSettings(); newGCRatingValue = settings.defaultStars; renderNewGCRating(); openModal('addGCModal'); } function renderNewGCRating() { document.getElementById('newGCRating').innerHTML = [1, 2, 3, 4, 5].map(i => `★`).join(''); } function setNewGCRating(r) { newGCRatingValue = r; renderNewGCRating(); } async function saveNewGC() { const name = document.getElementById('newGCName').value.trim(); const loc = document.getElementById('newGCLocation').value.trim(); if (!name) { alert('Please enter the GC name'); return; } const fullName = loc ? `${name} - ${loc}` : name; const gcs = await getGCs(); if (gcs.find(g => g.name.toLowerCase() === fullName.toLowerCase())) { alert('This GC already exists'); return; } // Capture risk tags const riskTags = Array.from(document.querySelectorAll('.gcRiskTag:checked')).map(cb => cb.value); await saveGC({ name: fullName, rating: newGCRatingValue, bids: 0, wins: 0, riskTags });

```
closeModal('addGCModal'); await loadGCDatabase(); await renderGCSelector(); } async function rateGC(name, rating) { const gcs = await getGCs(); const gc = gcs.find(g => g.name === name); if (gc) { gc.rating = rating; await saveGC(gc); await loadGCDatabase(); } } async function deleteGCUI(name) { if (!confirm('Delete ' + name + '?')) return; const gcs = await getGCs(); const gc = gcs.find(g => g.name === name); if (gc && gc.id) { await deleteGC(gc.id); } await loadGCDatabase(); await renderGCSelector(); } // Keywords - FIXED async function loadKeywords() { const kw = await getKeywords();
document.getElementById('goodKeywordsList').innerHTML = kw.good.map(k => `
  ${k} ×
`).join('') || 'No good keywords yet'; document.getElementById('badKeywordsList').innerHTML = kw.bad.map(k => `
  ${k} ×
`).join('') || 'No risk keywords yet'; } // Add good keyword(s) - supports comma-separated input async function addGoodKeyword() { const input = document.getElementById('goodKeywordInput'); const rawVal = input.value.trim(); if (!rawVal) { alert('Please enter a keyword'); return; } // Split by comma for multiple keywords const keywords = rawVal.split(',').map(k => k.trim().toLowerCase()).filter(k => k.length > 0); if (keywords.length === 0) { alert('Please enter a keyword'); return; } const kw = await getKeywords(); let added = 0; for (const val of keywords) { if (!kw.good.includes(val)) { kw.good.push(val); added++; } } if (added > 0) { await saveKeywordsStorage(kw); await loadKeywords(); } input.value = ''; input.focus(); } // Add bad keyword(s) - supports comma-separated input async function addBadKeyword() { const input = document.getElementById('badKeywordInput'); const rawVal = input.value.trim(); if (!rawVal) { alert('Please enter a keyword'); return; } // Split by comma for multiple keywords const keywords = rawVal.split(',').map(k => k.trim().toLowerCase()).filter(k => k.length > 0); if (keywords.length === 0) { alert('Please enter a keyword'); return; } const kw = await getKeywords(); let added = 0; for (const val of keywords) { if (!kw.bad.includes(val)) { kw.bad.push(val); added++; } } if (added > 0) { await saveKeywordsStorage(kw); await loadKeywords(); } input.value = ''; input.focus(); } async function removeKeyword(type, val) { const kw = await getKeywords(); kw[type] = kw[type].filter(k => k !== val); await saveKeywordsStorage(kw); await loadKeywords(); } // Settings async function loadSettings() { const s = await getSettings(); document.getElementById('settingCity').value = s.city; document.getElementById('settingState').value = s.state; document.getElementById('settingRadius').value = s.radius; document.getElementById('settingLocationMatters').checked = s.locationMatters; document.getElementById('settingRiskTolerance').value = s.riskTolerance; document.getElementById('settingCapacity').value = s.capacity; document.getElementById('settingDecisionTime').value = s.decisionTime; document.getElementById('settingDefaultStars').value = s.defaultStars; document.querySelector('#weightLocation input').value = s.weights.location; document.querySelector('#weightLocation .weight-slider-value').textContent = s.weights.location + '%'; document.querySelector('#weightKeywords input').value = s.weights.keywords; document.querySelector('#weightKeywords .weight-slider-value').textContent = s.weights.keywords + '%'; document.querySelector('#weightGC input').value = s.weights.gc; document.querySelector('#weightGC .weight-slider-value').textContent = s.weights.gc + '%'; document.querySelector('#weightTrade input').value = s.weights.trade; document.querySelector('#weightTrade .weight-slider-value').textContent = s.weights.trade + '%'; updateWeightTotal(); // CSI Divisions - SORTED
document.getElementById('tradesCheckboxes').innerHTML = CSI_DIVISIONS.map(([code, name]) => `
  ☐ ${code}: ${name}
` ).join(''); document.querySelectorAll('#tradesCheckboxes .checkbox-item').forEach(el => { const cb = el.querySelector('input'); // Handle checkbox changes only cb.addEventListener('change', () => { el.classList.toggle('selected', cb.checked); }); }); } function updateWeightTotal() { const l = parseInt(document.querySelector('#weightLocation input').value) || 0; const k = parseInt(document.querySelector('#weightKeywords input').value) || 0; const g = parseInt(document.querySelector('#weightGC input').value) || 0; const t = parseInt(document.querySelector('#weightTrade input').value) || 0; const total = l + k + g + t; const el = document.getElementById('weightTotal'); const warning = document.getElementById('weightWarning');
```

```
el.textContent = total + '%'; el.className = 'weight-total-value ' + (total === 100 ? 'valid' : 'invalid');
warning.style.display = total === 100 ? 'none' : 'inline'; } document.querySelectorAll('.weight-slider
input').forEach(input => { input.addEventListener('input', () => { input.closest('.weight-
slider').querySelector('.weight-slider-value').textContent = input.value + '%'; updateWeightTotal(); }); }); async
function saveSettings() { const weights = { location: parseInt(document.querySelector('#weightLocation
input').value) || 0, keywords: parseInt(document.querySelector('#weightKeywords input').value) || 0, gc:
parseInt(document.querySelector('#weightGC input').value) || 0, trade:
parseInt(document.querySelector('#weightTrade input').value) || 0 }; const total = weights.location +
weights.keywords + weights.gc + weights.trade; if (total !== 100) { alert('Score weights must total exactly
100%. Currently: ' + total + '%'); return; } const selectedTrades =
Array.from(document.querySelectorAll('#tradesCheckboxes input:checked')).map(cb => cb.value);
console.log('Selected trades:', selectedTrades); const s = { city:
document.getElementById('settingCity').value.trim(), state:
document.getElementById('settingState').value.trim().toUpperCase(), radius:
parseInt(document.getElementById('settingRadius').value), locationMatters:
document.getElementById('settingLocationMatters').checked, riskTolerance:
document.getElementById('settingRiskTolerance').value, capacity:
document.getElementById('settingCapacity').value, decisionTime:
parseInt(document.getElementById('settingDecisionTime').value) || 45, defaultStars:
parseInt(document.getElementById('settingDefaultStars').value), weights, trades: selectedTrades }; try { await
saveSettingsStorage(s); await updateCapacity(); // Reload settings from database to confirm they were saved
dataCache.settings = null; await loadSettings(); alert('✓ Settings saved to cloud!'); } catch (e) { alert('❌ Error
saving settings: ' + e.message); } } // ========== // ANALYTICS & LEARNING FUNCTIONS // ==========
function calculatePredictionAccuracy(projects) { const withOutcomes = projects.filter(p => p.outcome &&
p.outcome !== 'pending' && p.scores?.final); if (withOutcomes.length === 0) return null; const results = {
total: withOutcomes.length, byScore: { go: { total: 0, won: 0 }, review: { total: 0, won: 0 }, pass: { total: 0, won: 0
} }, byOutcome: { won: 0, lost: 0, ghost: 0, declined: 0 } }; withOutcomes.forEach(p => { const score =
p.scores.final; const outcome = p.outcome; // Track outcomes results.byOutcome[outcome]++; // Track by
score bucket const bucket = score >= 80 ? 'go' : score >= 60 ? 'review' : 'pass';
results.byScore[bucket].total++; if (outcome === 'won') results.byScore[bucket].won++; }); return results; }
function calculateUserFeedback(projects) { const withFeedback = projects.filter(p => p.user_agreement &&
p.user_agreement !== 'agree'); const total = projects.filter(p => p.scores?.final).length; return { total,
disagreements: withFeedback.length, tooHigh: withFeedback.filter(p => p.user_agreement ===
'too_high').length, tooLow: withFeedback.filter(p => p.user_agreement === 'too_low').length,
agreementRate: total > 0 ? ((total - withFeedback.length) / total * 100).toFixed(1) : 0 }; } function
calculateCalibration(projects) { const byOutcome = projects.filter(p => p.outcome && p.outcome !==
'pending' && p.scores?.final); if (byOutcome.length === 0) return null; const groups = { won: [], lost: [],
ghost: [], declined: [] }; byOutcome.forEach(p => groups[p.outcome].push(p.scores.final)); return {
avgScoreWon: groups.won.length > 0 ? (groups.won.reduce((a,b) => a+b, 0) / groups.won.length).toFixed(1) :
0, avgScoreLost: groups.lost.length > 0 ? (groups.lost.reduce((a,b) => a+b, 0) / groups.lost.length).toFixed(1) :
0, avgScoreGhost: groups.ghost.length > 0 ? (groups.ghost.reduce((a,b) => a+b, 0) /
groups.ghost.length).toFixed(1) : 0, avgScoreDeclined: groups.declined.length > 0 ?
(groups.declined.reduce((a,b) => a+b, 0) / groups.declined.length).toFixed(1) : 0 }; } function
generateRecommendations(accuracy, feedback, calibration) { const recommendations = []; if (!accuracy ||
accuracy.total < 5) { return ['
```

> 📊 Complete at least 5 projects with outcomes to see AI recommendations.

```
']; } // Check prediction accuracy const goWinRate = accuracy.byScore.go.total > 0 ?
(accuracy.byScore.go.won / accuracy.byScore.go.total * 100) : 0; const passWinRate =
```

accuracy.byScore.pass.total > 0 ? (accuracy.byScore.pass.won / accuracy.byScore.pass.total * 100) : 0; if (goWinRate < 50 && accuracy.byScore.go.total >= 3) { recommendations.push({ type: 'warning', title: 'Low Win Rate on GO Recommendations', message: `You're winning only ${goWinRate.toFixed(0)}% of GO-scored bids. The algorithm may be too optimistic. Consider increasing your score thresholds or adjusting component weights.` }); } if (passWinRate > 20 && accuracy.byScore.pass.total >= 3) { recommendations.push({ type: 'warning', title: 'Missing Opportunities', message: `You won ${passWinRate.toFixed(0)}% of PASS-scored bids. The algorithm may be too conservative. You might be passing on winnable projects.` }); } // Check user feedback if (feedback && feedback.disagreements > 0) { const disagreeRate = (feedback.disagreements / feedback.total * 100); if (disagreeRate > 30) { recommendations.push({ type: 'info', title: 'Frequent Score Disagreements', message: `You disagree with ${disagreeRate.toFixed(0)}% of scores (${feedback.tooHigh} too high, ${feedback.tooLow} too low). Consider adjusting component weights in Settings to better match your preferences.` }); } } // Check calibration if (calibration && calibration.avgScoreWon && calibration.avgScoreLost) { const wonScore = parseFloat(calibration.avgScoreWon); const lostScore = parseFloat(calibration.avgScoreLost); if (wonScore - lostScore < 10) { recommendations.push({ type: 'warning', title: 'Poor Score Separation', message: `Average score for won bids (${wonScore}) is close to lost bids (${lostScore}). The algorithm isn't differentiating well. Check if your keyword lists and GC ratings are accurate.` }); } } if (recommendations.length === 0) { recommendations.push({ type: 'success', title: 'Scores Looking Good!', message: 'Your prediction accuracy is solid and scores align with outcomes. Keep tracking results to further refine the algorithm.' }); } return recommendations; } async function renderAnalytics() { const projects = await getProjects(); if (!projects || projects.length === 0) { document.getElementById('accuracyMetrics').innerHTML = '

> No projects yet. Start analyzing bids to see AI learning insights.

'; document.getElementById('feedbackAnalysis').innerHTML = ''; document.getElementById('calibrationInsights').innerHTML = ''; document.getElementById('aiRecommendations').innerHTML = ''; return; } // Calculate metrics const accuracy = calculatePredictionAccuracy(projects); const feedback = calculateUserFeedback(projects); const calibration = calculateCalibration(projects); const recommendations = generateRecommendations(accuracy, feedback, calibration); // Render Prediction Accuracy if (accuracy && accuracy.total > 0) { const goWinRate = accuracy.byScore.go.total > 0 ? (accuracy.byScore.go.won / accuracy.byScore.go.total * 100).toFixed(0) : 0; const reviewWinRate = accuracy.byScore.review.total > 0 ? (accuracy.byScore.review.won / accuracy.byScore.review.total * 100).toFixed(0) : 0; const passWinRate = accuracy.byScore.pass.total > 0 ? (accuracy.byScore.pass.won / accuracy.byScore.pass.total * 100).toFixed(0) : 0; document.getElementById('accuracyMetrics').innerHTML = `

GO Recommendations (80-100)

${goWinRate}% win rate

${accuracy.byScore.go.won} won / ${accuracy.byScore.go.total} total

REVIEW Recommendations (60-79)

${reviewWinRate}% win rate

${accuracy.byScore.review.won} won / ${accuracy.byScore.review.total} total

PASS Recommendations (0-59)

${passWinRate}% win rate

${accuracy.byScore.pass.won} won / ${accuracy.byScore.pass.total} total

Total projects with outcomes: ${accuracy.total} (${accuracy.byOutcome.won} won, ${accuracy.byOutcome.lost} lost, ${accuracy.byOutcome.ghost} ghosted, ${accuracy.byOutcome.declined} declined)

`; } else { document.getElementById('accuracyMetrics').innerHTML = '

Complete at least 1 project with an outcome to see accuracy metrics.

'; } // Render User Feedback if (feedback && feedback.total > 0) { document.getElementById('feedbackAnalysis').innerHTML = `

Agreement Rate

${feedback.agreementRate}%

${feedback.total - feedback.disagreements} agreed / ${feedback.total} total

Score Too High

${feedback.tooHigh}

False positives

Score Too Low

${feedback.tooLow}

False negatives

Use the feedback toggles after each analysis to train the AI on your preferences.

`; } else { document.getElementById('feedbackAnalysis').innerHTML = '

User feedback will appear here after you analyze bids and provide agreement ratings.

'; } // Render Calibration if (calibration && calibration.avgScoreWon) { document.getElementById('calibrationInsights').innerHTML = `

Avg Score - Won Bids

${calibration.avgScoreWon}

Avg Score - Lost Bids

${calibration.avgScoreLost}

Avg Score - Ghosted

${calibration.avgScoreGhost}

Avg Score - Declined

${calibration.avgScoreDeclined}

> **Ideal calibration:** Won bids should have higher average scores than lost/ghosted bids. A 15-20 point difference indicates good calibration.

`; } else { document.getElementById('calibrationInsights').innerHTML = '

> Calibration insights will appear after completing more projects with outcomes.

'; } // Render Recommendations const recHtml = recommendations.map(rec => { if (typeof rec === 'string') return rec; const icon = rec.type === 'success' ? '✅' : rec.type === 'warning' ? '⚠️' : 'ℹ️'; const color = rec.type === 'success' ? 'var(--success)' : rec.type === 'warning' ? 'var(--warning)' : 'var(--info)'; return `

${icon} ${rec.title}

${rec.message}

`; }).join(''); document.getElementById('aiRecommendations').innerHTML = recHtml; } // Load all data on app init async function loadAll() { await loadSettings(); // Check if onboarding is needed (only if user is authenticated) const settings = await getSettings(); console.log('📋 loadAll - checking onboarding:', { onboarding_completed: settings.onboarding_completed, currentUser: !!currentUser }); if (!settings.onboarding_completed && currentUser && supabaseClient) { console.log('🎯 Showing onboarding...'); renderOnboardingStep(1); document.getElementById('onboardingModal').classList.add('active'); return; // Don't load other data until onboarding is complete } else { console.log('✅ Skipping onboarding - already completed or not authenticated'); document.getElementById('onboardingModal').classList.remove('active'); } await loadKeywords(); await loadGCDatabase(); await loadProjects(); await updateCapacity(); await renderAnalytics(); }