Feature Request: Contextual Merchant Intelligence for Sales Role-Play & Coaching

Objective

Enhance the Sales Role-Play Coach and Get Coaching features to be dynamically informed by real merchant data. The AI should understand who the merchant actually is—not just their business type—by gathering intelligence in the background without slowing down the user experience.

Desired Outcome

When a sales rep opens Role-Play or Coaching for a specific merchant (e.g., "Peking Chinese Restaurant"), the AI persona should behave like that actual business owner would, informed by:

1. **Everything in the merchant/deal record** - business name, type, location, stage, notes, contact info, lead temperature
2. **Their website** (if available in the record) - scraped for menu/services, pricing signals, hours, owner name, unique selling points, recent news or promotions
3. **Prior meeting transcriptions** (if recordings exist) - how the owner actually talks, their communication style, objections they've raised, questions they asked, their decision-making patterns, key concerns

Technical Requirements

Background Processing

- Website scraping and analysis must happen asynchronously—do NOT block the role-play UI
- Cache intelligence data per merchant so repeat sessions are instant
- Gracefully handle missing data (no website URL, no recordings, etc.)

Intelligence Gathering

Determine the best approach to:

- Fetch and parse merchant websites for relevant business intelligence
- Extract insights from meeting transcription text (communication patterns, objections, interests)
- Synthesize this data into a structured context object

AI Context Injection

The role-play and coaching AI calls should receive enriched context including:

- Merchant business profile (from scraping + record data)
- Owner persona details (from transcripts if available)
- Conversation history and known objections
- Current pipeline stage and deal history

Data Model Considerations

You may need to:

- Add fields to store cached merchant intelligence
- Create a background job/worker pattern for scraping
- Store extracted insights from transcriptions

User Experience

- Role-play should launch immediately with basic merchant info
- Enhanced intelligence loads in background and enriches the session
- Visual indicator if enhanced data is still loading vs. ready
- If website scraping fails, continue with available data—never block the feature

Scenarios to Support

1. **Cold Approach** - AI knows the business from website research, acts like a skeptical owner who's never met the rep
2. **Objection Handling** - AI uses actual objections from past transcripts if available
3. **Closing** - AI references prior conversations and known concerns
4. **Follow-up Visit** - AI remembers what was discussed (from transcripts)
5. **Get Coaching** - Advice is tailored to this specific merchant's situation and history

Example Context Flow

```
Rep opens role-play for "Peking Chinese Restaurant"
├── Immediate: Load deal record data (name, type, stage, notes, temperature)
├── Background: Fetch website → extract menu, hours, pricing, owner name
├── Background: Load any meeting transcripts → extract communication patterns
└── Inject all available context into AI system prompt
```

Quality Bar

The role-play should feel like practicing on the ACTUAL merchant, not a generic "Chinese restaurant owner." If we have a transcript where the owner said "I'm happy with my current processor and they're my cousin," the AI should use that relationship objection. If the website shows they've been in business since 1985, the AI should act like an established owner who's seen sales reps come and go.

Implementation Approach

Analyze the existing codebase and determine:

- Where role-play/coaching AI calls are made
- How merchant/deal data is currently accessed
- Best pattern for background processing in this stack
- How to structure the intelligence cache
- Optimal way to enrich AI prompts with gathered context

Make architectural decisions that fit the existing patterns in the codebase. Prioritize reliability and user experience over feature completeness—it's better to have solid website intelligence than broken transcript analysis.

Start Here

1. Find the role-play and coaching components/routes
2. Trace how they currently get merchant context
3. Design the intelligence gathering and caching layer
4. Implement background fetching that doesn't block UI
5. Enrich the AI prompts with gathered intelligence
6. Test with a real merchant record