# Golf Rules Assistant Project Context Summary

## Project Overview

Building a voice-activated golf rules assistant that helps golfers understand rules while on the course. The system combines a structured golf rules database with vector search and LLM interpretation to provide conversational rule guidance.

## Current System Architecture

### Core Components Built

1. **Golf Rules Database** (`golf_rules_data.py`)
   - Comprehensive USGA/R&A rules structured as Python dictionaries
   - Hierarchical organization (sections → rules → sub-rules)
   - Rich metadata: keywords, examples, conditions, situational explanations
   - ~200+ rules with detailed conditions and examples
2. **Vector Search Engine** (`vector_search.py`)
   - Uses sentence transformers for semantic search
   - Supports both standard and LLM-optimized search modes
   - Compression features for cost-efficient LLM usage (65-70% savings)
   - Cached embeddings for performance
3. **Hybrid AI System** (`golf_rules_hybrid.py`)
   - Production-ready assistant combining vector search + LLM interpretation
   - Cost optimization with compressed context
   - Usage logging and monitoring
   - Conversational rule interpretations

### Recent Rules Expansion

Just expanded Rules 16.3 (Embedded Ball) and 16.4 (Lifting Ball to Check Conditions) with full conditional details matching the existing detailed structure.

## Strategic Pivot: B2B Club-Focused Approach

### New Business Model

- **Target**: Golf clubs as customers (not individual consumers)

- **Product**: Plugin/service for existing club member apps
- **Value Prop**: Premium member service with club-specific local rules
- **Revenue**: SaaS subscriptions per club ($200-1000/month tiers)

### Why This Approach

- Eliminates need for GPS course detection
- Simplified technical scope (one club's rules per instance)
- Predictable B2B revenue vs uncertain consumer adoption
- Direct relationship with rule authorities (head pros)
- Leverages existing member app infrastructure

## Planned Local Rules Integration

### Multi-Tenant Architecture

```python
CLUB_INSTANCE = {
    "club_id": "club_name_id",
    "club_name": "Club Name",
    "local_rules": [...],  # Club-specific rules only
    "course_info": {...},
    "branding": {...},     # Logo, colors, terminology
    "features_enabled": {...}
}
```

### Rule Precedence Logic

1. Tournament-specific local rules (highest)
2. Permanent local rules (hole-specific)
3. Course-wide local rules
4. Official Rules of Golf (lowest)

### Modified Query Workflow

```
Voice Input → Text Query → Local Rules Search (FIRST) →
Official Rules Search (if needed) → Rule Precedence Resolution →
LLM Interpretation with Local Context → Voice Response
```

## Current Development Status

### What's Working

- Complete official rules database with vector search
- Production-ready hybrid AI system
- Cost-optimized LLM integration
- Conversational rule interpretations

### Next Steps - Target Club MVP

- **Real Partner**: Has access to a golf club, head pro interested
- **Local Rules**: Has club's local rules database (93KB) ready to integrate
- **Goal**: Build single-club MVP for testing and demo

## Technical Implementation Plan

### Phase 1: Data Integration (Week 1)

1. Parse club's local rules into structured format
2. Create club-specific database with their rules
3. Test search functionality with actual club rules

### Phase 2: Basic Interface (Week 2)

1. Build simple web interface for testing
2. Implement voice input (if needed)
3. Test with real club scenarios

### Phase 3: Integration Ready (Week 3)

1. Create API endpoints for club app integration
2. Build integration documentation
3. Demo with head pro

## Key Technical Decisions Needed

1. **Local Rules Format**: Need to see club's actual local rules structure
2. **Integration Method**: API integration vs standalone widget vs white-label app
3. **Interface Type**: Web app, voice interface, or both

4. **Club App Platform**: What platform is their existing member app built on?

## Files Structure

```
golf_rules_assistant/
├── golf_rules_data.py      # Complete official rules database
├── vector_search.py        # Semantic search engine
├── golf_rules_hybrid.py    # Production AI system
└── [TO BUILD]
    ├── clubs/
    │   ├── club_manager.py     # Multi-tenant management
    │   ├── local_rules_db.py   # Club-specific rules
    │   └── club_config.py      # Club customization
    ├── api/
    │   ├── club_api.py         # Club-specific endpoints
    │   └── member_auth.py      # Integration with club apps
    └── admin/
        ├── club_onboarding.py  # New club setup
        └── rules_editor.py     # Club staff rule editing
```

## Immediate Next Actions

1. **Analyze club's local rules format** (have 93KB file ready)
2. **Design local rules data structure** to match their format
3. **Modify existing search system** to prioritize local rules
4. **Build single-club MVP interface**
5. **Test with head pro scenarios**

## Questions to Resolve

1. What format are the club's local rules in? (PDF, Word, spreadsheet, etc.)
2. How does the club want members to access this? (app integration, web link, etc.)
3. What specific scenarios does the head pro say members ask about most?
4. What platform is their existing member app built on?

---

**Current Status**: Ready to begin MVP development with real club partner. Need to analyze their local rules format and build club-specific integration.