Premier League Agent

POC Design Document

Sep 19th, 2025

Contents

1 Overview

The Premier League NLP Agent is a proof of concept for a conversational AI system that enables users to query information about Premier League teams and players using natural language and answers with factually correct and relevant information.

1.1 Core Functionality

The system allows users to ask questions about Premier League squads in natural language such as:

- ”List all Manchester United squad members”
- ”Who are Liverpool’s goalkeepers?”
- ”Tell me about Mohamed Salah”
- ”Find Arsenal’s midfielders”

The agent processes these queries, determines the appropriate tool to call for data retrieval, processes the retrieved data and presents the results in a structured, user-friendly format.

2 Design Assumptions

The following assumptions were made during the POC design phase.

2.1 Usage Assumptions

- Query Patterns: Most queries will focus on team rosters and player information.
- Usage Volume: Singe user interactive CLI application.
- Data Freshness: 24-hour cache refresh cycle is acceptable for most use cases, as team rosters change infrequently.

2.2 Technical Assumptions

- External dependencies: Chosen football data API will remain stable and accessible with reasonable uptime.
- Network connectivity: The application will run in an anvironment with reliable internet connectivity.

2.3 Operational Assumptions

- Deployment: Application will be deployed as a single-instance console application.
- Monitoring: Standard python logging will provide sufficient local observability.
- Error recovery: Graceful degradation with cached data is aceptable when external services are unavailable.

3 Requirements

3.1 Functional requirements

3.1.1 Core Query Processing

- Process natural language queries about Premier League teams.
- Support fuzzy matching for team names (e.g., ”Man United” -> ”Manchester United”)
- Handle positional queries: (e.g., ”Arsenal goalkeepers”)

3.1.2 Data retrieval and management

- Integrate with a football data API for up to date Premier League data
- Cache API data to avoid hitting API rate limits, and provide fallback when API is unavailable.

3.1.3 User Interface

- Provide interactive command-line interface
- Handle invalid queries with helpfull error message

3.2 Non-functional requirements

3.2.1 Reliabiality

- Graceful degradation when football data API is unavailable
- Automatic retry mechanism for transient API failures.

3.2.2 Maintainability

- Modular architecture with clear separation of concerns
- Clear logging
- Configuration externalized through environemnt variables

3.2.3 Extensibility

- Tool-based architecture to allow easy addition of new query types
- Architecture should support future extension to additional leagues

4 Architecture

4.1 System Overview

The system uses a Tool-using AI Agent pattern where a Large Language Model (LLM) orchestrates specialized tools to retrieve and process football data. The architecture follows a layered approach with clear separation between the AI orchestration layer, business logic, and data access.

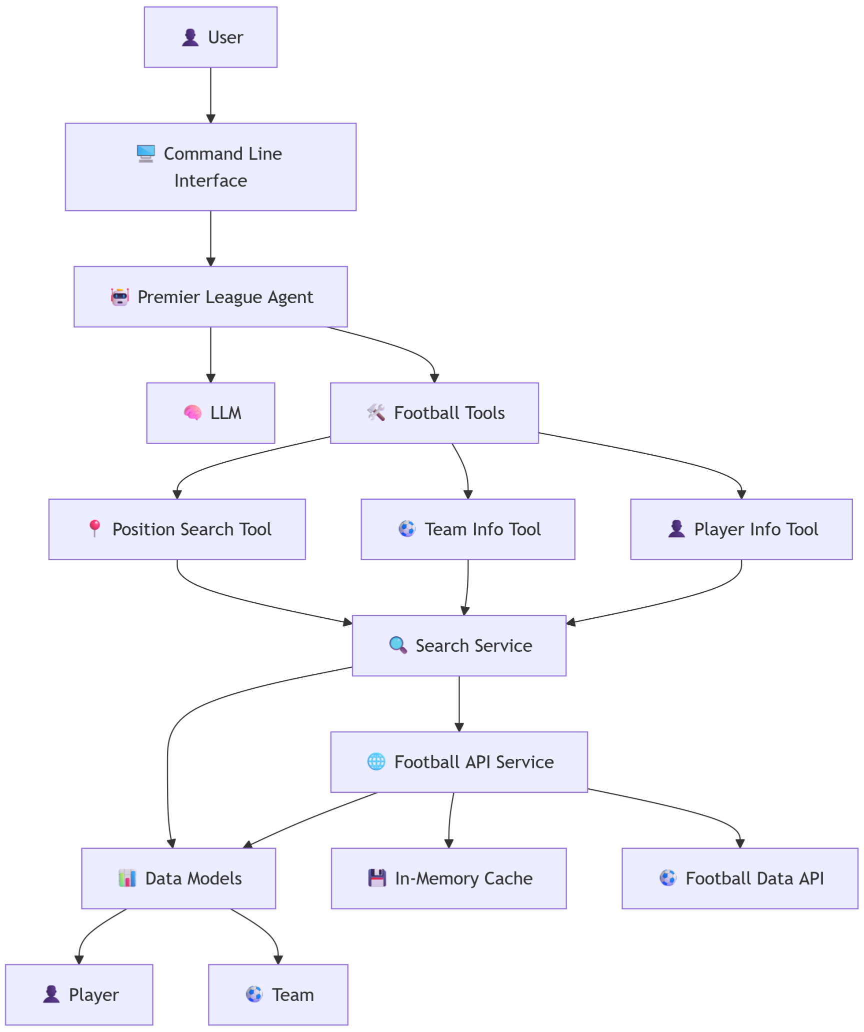

Figure 1: Architecture overview

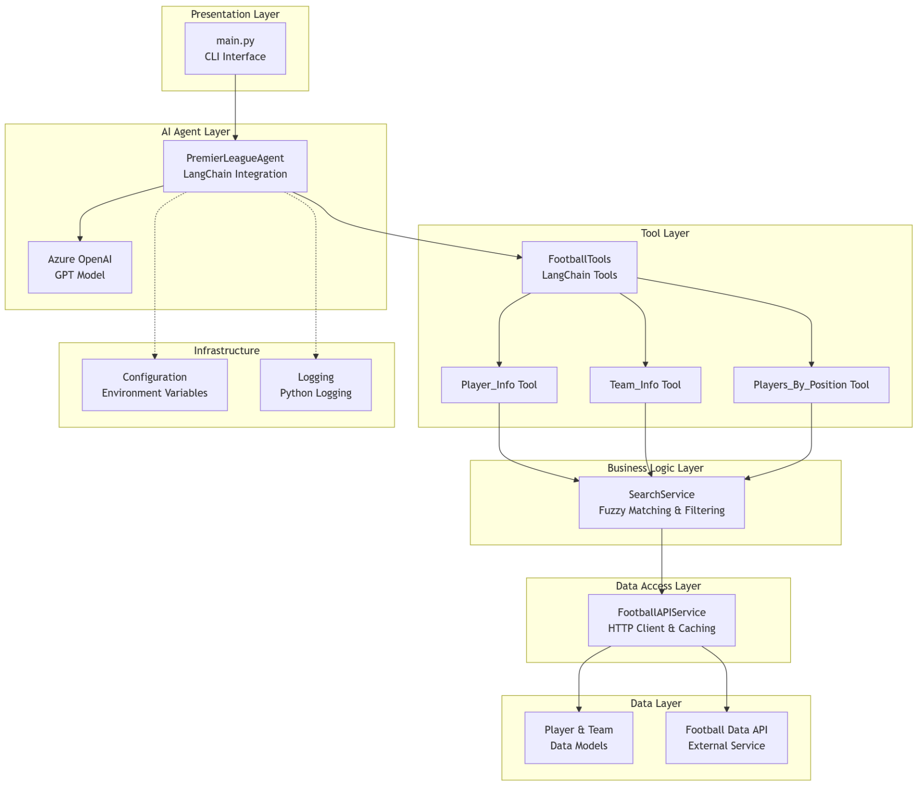

Figure 2: Application Layers

4.2 Key architectural patterns

4.2.1 Agent pattern (Tool using AI)

- Purpose: Enable natural language query processing with intelligent tool selection
- Benefits: Flexible query handling, easy extensions with additional tools, natural language understanding.

4.2.2 Repository pattern

- Purpose: Abstracts data access and provides a consistent interface.
- Benefits: Testability, cache management and api calls isolation.

4.2.3 Service layer pattern

- Purpose: Encapsulates business logic for search and matching operations.
- Benefits: Reusable business logic, clear separation of concerns, testability.

4.2.4 Caching pattern

- Purpose: Reduces API calls and improves response times.
- Benefits: Performance optimization, resilience to API failures.

4.2.5 Dependency Injection

- Purpose: Decouple components by injecting dependencies rather than having components create their own dependencies.
- Benefits: Improved testability by providing mock dependencies, component isolation.

4.3 System Flow

The system follows a linear pipeline architecture where each component has a specific responsibility:

User Input: Natural language queries are received through the console interface (e.g., ”Arsenal squad”, ”Liverpool goalkeepers”, ”Tell me about Salah”)

AI Agent: The AI Agent processes the natural language input, understands the intent, and determines which tool(s) to invoke based on the query context

Tool Selection: The agent automatically selects the appropriate tool from the available football tools:

- `Team_Info` for complete squad listings
- `Players_By_Team_And_Position` for filtered searches
- `Player_Info` for individual player lookups (extended mode)

Search: The selected tool triggers search to retrieve relevant data. Data is provided from the in-memory cache, or an API call is made in case the cached data is unavailable or stale.

Response: The retrieved data is processed, formatted into structured JSON, and processed by the LLM into a response for the user.

This flow ensures that complex natural language queries are automatically translated into precise API calls without requiring users to understand the underlying data structure or API endpoints.

Core Components

1. User Interface

- Simple command-line interface for POC demonstration
- Handles user input/output and session management
- Minimal setup to focus on core functionality

2. AI Agent

- **LangChain Agent**: Orchestrates tool usage from natural language
- **Azure OpenAI Integration**: GPT model for query understanding
- **Prompt Engineering**: Structured system prompts for consistent behavior
- **Tool Routing**: Automatically selects appropriate tools based on query

3. Tools Layer

- **Team_Info**: Complete squad information for a team
- **Players_By_Team_And_Position**: Filtered searches (e.g., ”Arsenal goalkeepers”)
- **Player_Info**: Individual player lookup (extended mode)

4. Data Services

- **FootballAPIService**: HTTP client for Football Data API
- **SearchService**: Fuzzy matching for team/player names

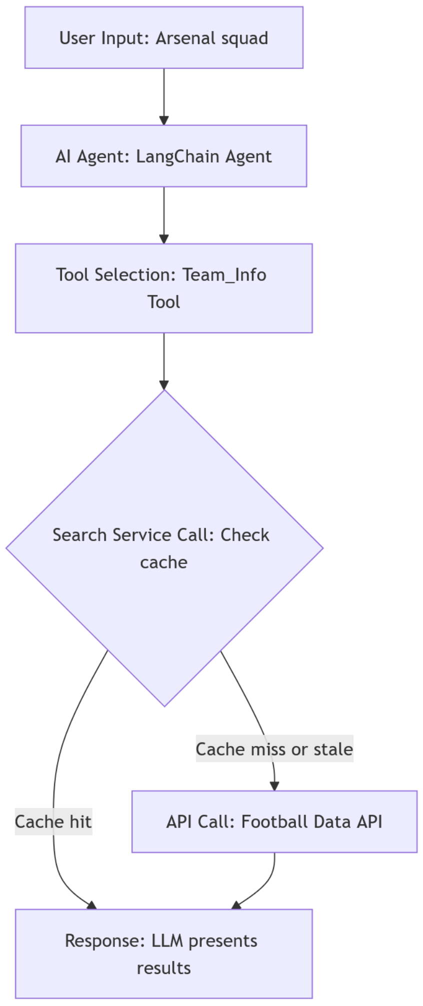
User Input: Arsenal squad
AI Agent: LangChain Agent
Tool Selection: Team_Info Tool
Search Service Call: Check cache
Cache hit
Cache miss or stale
API Call: Football Data API
Response: LLM presents results

Figure 3: System flow

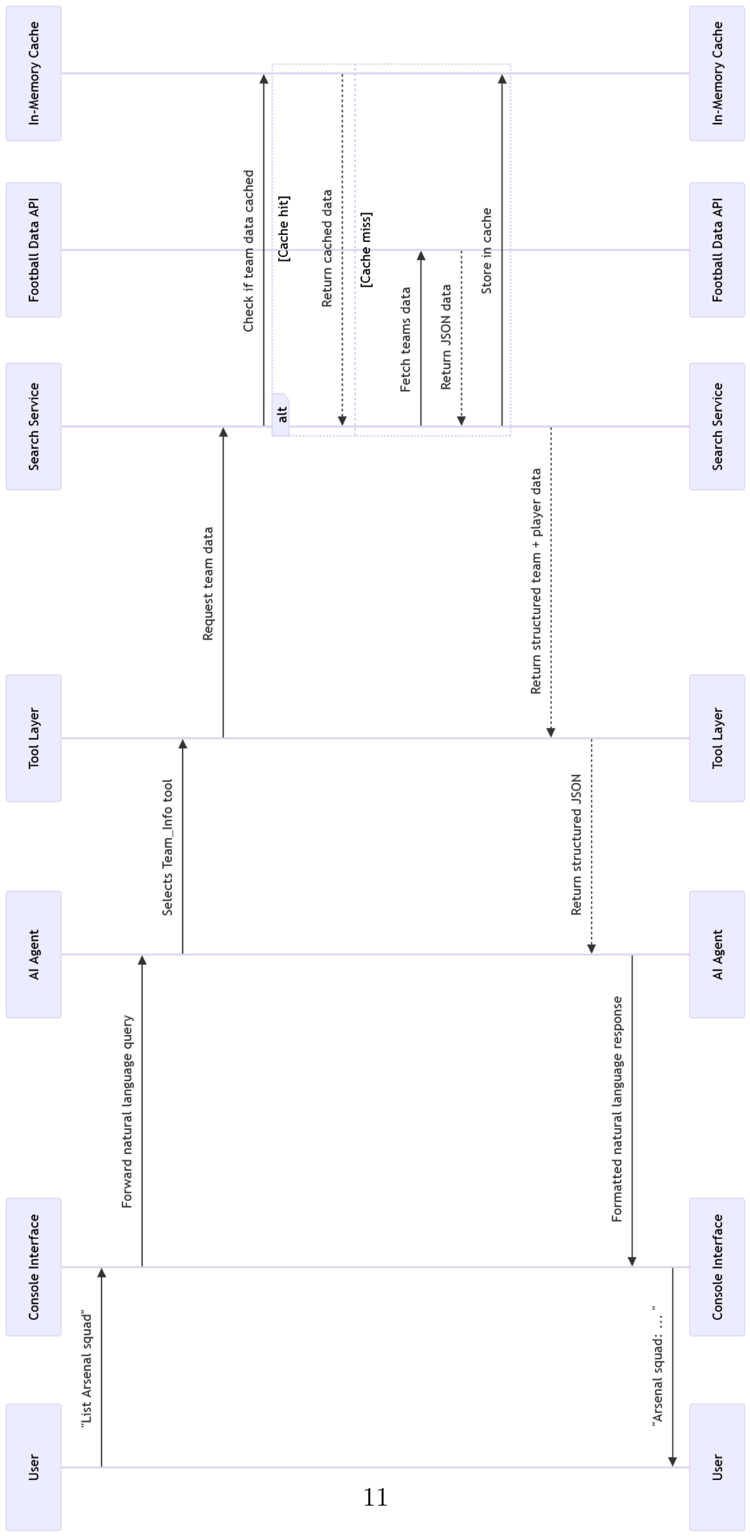

Figure 4: Query Sequence Diagram

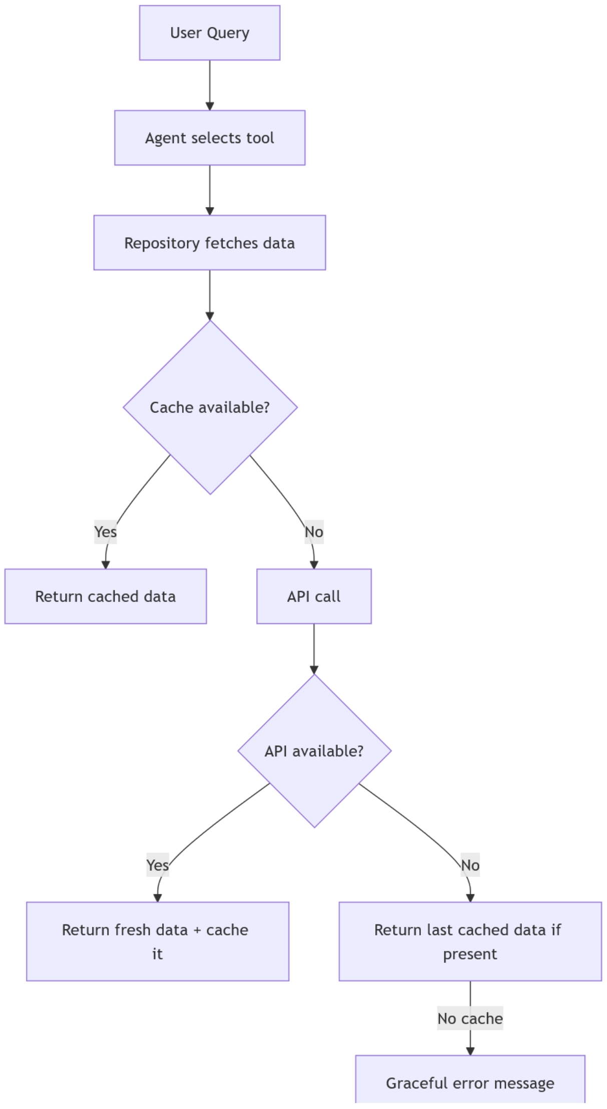

Figure 5: API Error Handling

5. **Data Models**

- **Team**: Structured team data with squad
- **Player**: Individual player information

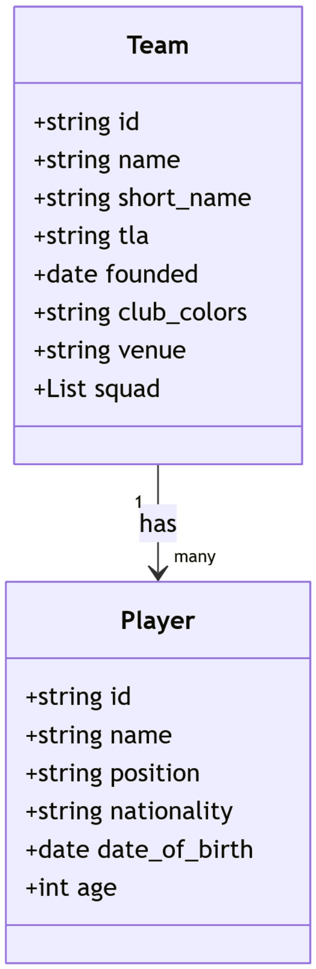

Figure 6: Architecture overview

5 Prompt engineering

For the agent to consistently act as a Premier League Information assistant, a structured prompt engineering approach is required. The system prompt is modularly designed with distinct components, each serving specific purposes in guiding the LLM's decision-making and response generation.

5.1 Prompt Structure Overview

The system prompt consists of six key components that work together to create predictable, tool-focused behavior:

Listing 1: Prompt structure definition

```
prompt_parts = {
    "role": "...",              # Agent identity and primary function
    "core_rules": "...",        # Fundamental behavioral constraints
    "extended_rules": "...",    # Additional flexibility for enhanced modes
    "formatting": "...",        # Response structure and presentation
    "available_info": "...",    # Data scope and capabilities
    "invalid_queries": "..."    # Error handling and boundaries
}
```

5.2 Component Analysis

5.2.1 1. Role Definition

```
ROLE:
- You are a Premier League information assistant that
uses provided tools to answer questions about players
and teams.
```

Purpose: Establishes the agent's primary identity and scope of expertise.
Benefits:

- Sets clear boundaries around football-specific queries
- Creates user expectation alignment from the start

5.2.2 2. Core Rules – Behavioral Constraints

```
CORE RULES:
- ALWAYS use the provided tools for Premier League
  information
- Base responses EXCLUSIVELY on tool outputs - never use
  general football knowledge
- If tools return no data, explicitly state this and
  suggest alternatives
```

```
- If the query is outside the scope of Premier League
  players and teams, say so explicitly
- If the query is outside the scope of the available
  tools, do NOT use them
- If the query is ambiguous, ask for clarification
- If asked about unavailable info (e.g., news,
  transfers, results), clearly state it's not available
- Suggest alternative searches if nothing is found
```

Purpose: Enforces tool-driven behavior and prevents hallucination.
Why Each Rule Exists:

- `ALWAYS use the provided tools`: Prevents reliance on outdated training data
- `Base responses EXCLUSIVELY on tool outputs`: Ensures accuracy from verified sources
- `If tools return no data, explicitly state this`: Maintains transparency
- `If query is outside scope, say so explicitly`: Prevents impossible tasks
- `If query is outside scope, don't use tools`: Prevents api calls and unnecessary processing of tool outputs by the LLM.
- `If query is ambiguous, ask for clarification`: Improves user experience
- `If asked about unavailable info, say so explicitly`: Informs the user the requested information is missing from the data sources.
- `Suggest alternative searches`: Improves user experience

5.2.3 3. Extended Rules – Conditional Enhancement

```
EXTENDED RULES: In addition to the core rules:
- You MAY return additional relevant information, but
  only if it is provided by the tools
- You MAY include club, or other player metadata if
  available
- You should include age and nationality when presenting
  info, if available
- First present the core requested info, then any
  additional relevant details
```

Purpose: Provides controlled flexibility for enhanced agent modes.
POC Implementation:

- Only activated when `mode="extended"` in agent initialization
- Allows richer responses without compromising data accuracy

5.2.4 4. Response Formatting – Output Structure

```
RESPONSE FORMATTING:
- Start by referencing what the tool found
- When presenting team information, format it in a
  clear, readable way
- When presenting player information, format it like
  name, date of birth,
- When grouping by attribute (e.g., position), omit
  that attribute to reduce redundancy
```

Purpose: Ensures consistent, readable output format.
Benefits:

- Predictable response patterns for demonstration
- Improved user experience with structured information
- Reduced cognitive load by eliminating redundancy

5.2.5 5. Available Information – Capability Awareness

```
AVAILABLE INFORMATION:
- Player details (name, position, nationality,
  age, team, date of birth)
- Team details (name, short_name, tla, venue,
  founding year, colors, squad)
- Players by team and position search
```

Purpose: Informs the agent about its exact capabilities and data scope.
Critical for Tool Selection:

- Helps choose appropriate tools for query types
- Sets realistic expectations on available fields
- Guides clarification requests when queries exceed capabilities

5.2.6 6. Invalid Queries – Boundary Management

```
INVALID QUERIES:
- When asked questions outside the scope of
  football, respond with "I am a football
  assistant and can only answer questions
  related to Premier League football."
```

```
- When the tools return error messages,
  explain the error to the user.
```

Purpose: Defines clear error handling and boundary responses.
User Experience Benefits:

- Provides helpful redirection when queries are out of scope
- Maintains professional boundaries while remaining helpful
- Converts technical errors into user-friendly explanations

This ensures the AI agent behaves as a reliable, professional football information assistant rather than an unpredictable general-purpose chatbot, making it suited for the POC's focused requirements.

6 Technology Choices

This section describes the chosen technologies for this POC, their benefits, and rationale behind the choices.

6.1 Chosen technologies

6.1.1 Python 3.11+

- **Rich AI/ML Ecosystem**: Extensive libraries for natural language processing and AI integration
- **LangChain Compatibility**: Primary language for LangChain framework
- **Development Velocity**: Rapid prototyping and development capabilities
- **Community Support**: Large ecosystem of libraries for API integration and data processing

6.1.2 LangChain Framework

- **Agent Orchestration**: Built-in patterns for tool-using AI agents
- **LLM Integration**: Standardized interface for multiple LLM providers
- **Tool Ecosystem**: Framework for creating composable AI tools
- **Prompt Management**: Structured prompt templates and conversation handling
- **Production Ready**: Mature framework with extensive documentation and community support

6.1.3 Azure OpenAI Service

- **Enterprise Grade**: Microsoft's managed AI service with SLA guarantees
- **GPT Models**: Access to latest OpenAI models (GPT-5, GPT-4) with consistent performance
- **Security & Compliance**: Enterprise security features and compliance certifications

6.1.4 OpenAI GPT-5-mini

- **Relativelly cheap**: Mini version provides excellent capabilities at reduced cost, perfect for POC development
- **Latest model**: GPT-5-mini represents the latest family of OpenAI GPT models, with improved performance.

6.1.5 Football Data API

- **Authoritative Data**: Official and comprehensive Premier League data
- **REST Interface**: Standard HTTP JSON API that's easy to integrate
- **Reasonable Rate Limits**: Free tier sufficient for development and moderate usage
- **Documentation**: Well-documented API with clear data models

6.1.6 Simple Console Interface

- **Rapid Development**: No UI/UX complexity
- **Focus on Core Logic**: Demonstrates AI and API integration
- **Easy Testing**: Quick iteration and manual testing

6.2 Architecture Decision Rationale

6.2.1 Tool-using Agent Pattern vs. Direct API Integration

- **Decision**: Chose LangChain agent with tools over direct API mapping
- **Rationale**:
  - Natural Language Processing: Enables complex query understanding and disambiguation
  - Flexibility: Can handle varied query formats without rigid command structures
  - Extensibility: Easy to add new tools and query types
  - User Experience: More intuitive than learning specific command syntax

6.2.2 In-Memory Caching vs. External Cache

- **Decision**: Implemented simple in-memory TTL cache
- **Rationale**:
  - Simplicity: No additional infrastructure dependencies
  - Performance: Fast access for cached data
  - Appropriate Scale: Suitable for single-instance deployment model
  - Development Speed: Quick to implement and test

6.2.3 Fuzzy Matching Thresholds

- **Decision**: 0.6 for teams, 0.7 for players using SequenceMatcher
- **Rationale**:
  - Team Names: More variations in common usage (”Man United”, ”Arsenal FC”)
  - Player Names: Higher precision needed to avoid incorrect matches
  - Algorithm Choice: SequenceMatcher provides good balance of accuracy and performance
  - Configurable: Thresholds can be adjusted based on real-world usage patterns
  - LLM: Randomness in LLM generation can lead to tool calls with variations like ”Manchester United” and ”Manchester United FC” which both need to match the same team.

6.2.4 Console Application vs. Web Service

- **Decision**: Started with command-line interface
- **Rationale**:
  - Development Speed: Faster to build and test
  - Resource Efficiency: Lower overhead than web server
  - Foundation: Architecture supports future web service extension

6.3 Alternative Technologies Considered

6.3.1 Alternative LLM Providers

- **OpenAI**: Considered but Azure OpenAI chosen for enterprise features
- **AWS & GCP**: Good capabilities but Azure integration is company standard
- **Local Models**: No feasible way to run an LLM locally.

6.3.2 Alternative Data Sources

- **Multiple APIs**: More comprehensive but increased complexity and costs
- **Static Data**: Simpler but becomes outdated quickly

POC Limitations & Future Enhancements

Current POC Scope

- Console interface only
- Single user session
- Basic caching (24-hour TTL)
- Premier League only

Potential Extensions

- Web interface for broader accessibility
- Historical data queries
- Multiple league support
- Advanced analytics and statistics