Project - High Level Design

On

Sports Planning Assistant Agent

Course Name: Agentic AI

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
1	Priyanshi Gandhe	EN22CS301764
2	Priyanshu Sengar	EN22CS301770
3	Rudra Yadav	EN22CS301831
4	Prithviraj Singh Tomar	EN22CS301757
5	Pranjali Sharma	EN22CS301733

Group Name: 07D7

Project Number: AAI-30

Industry Mentor Name:

University Mentor Name: Prof. Ajeet Singh Rajput

Academic Year: 2026

1. Introduction

Agentic Artificial Intelligence (Agentic AI) represents an advanced paradigm in artificial intelligence where systems are designed to act autonomously, reason over goals, plan multi-step actions, and adapt dynamically to changing environments. Unlike traditional rule-based or reactive AI systems, agentic systems are goal-driven and capable of decomposing complex objectives into manageable sub-problems while maintaining contextual awareness and decision continuity.

In the domain of sports management and analytics, planning activities such as match analysis, player performance evaluation, tournament scheduling, and training preparation involve multiple interdependent tasks, constrained resources, and time-sensitive decisions. Manual planning in such environments is often inefficient, error-prone, and difficult to scale. The **Sports Planning Assistant Agent** addresses these challenges by applying Agentic AI principles to automate sports-related planning tasks through intelligent reasoning, dependency management, and execution scheduling.

This project focuses on engineering a **Planner Agent** capable of receiving high-level sports goals and autonomously generating structured, optimized, and executable plans in a simulated sports management environment.

1.1 Scope of the Document

This High Level Design document defines the **architectural blueprint** of the Sports Planning Assistant Agent. It covers the system's overall architecture, component-level design, process and information flow, data design, interfaces, and non-functional requirements.

The scope includes:

- Agent-based system architecture
- Core planning and reasoning components
- Data and state management strategies
- Interface and API-level interactions
- Security, performance, and scalability considerations

The following aspects are **out of scope**:

- Detailed algorithmic implementation

- Low-level code design
- UI wireframes
- Deployment and DevOps configurations

1.2 Intended Audience

This document is intended for:

- Computer Science and Engineering (CSE) students
- Project guides and academic evaluators
- System architects and AI researchers
- Review committees involved in project assessment

Readers are expected to have a foundational understanding of software engineering principles, artificial intelligence concepts, and system architecture.

1.3 System Overview

The **Sports Planning Assistant Agent** is an agent-based intelligent system designed to autonomously plan and manage complex sports-related tasks. The system accepts a high-level goal, such as *"Generate Match Statistics"*, and transforms it into a sequence of executable subtasks through structured reasoning.

At a high level, the system performs the following functions:

- Interprets abstract user goals
- Decomposes goals into actionable tasks
- Resolves dependencies between tasks
- Validates resource availability via mock tools
- Generates an optimized execution schedule
- Monitors execution and produces structured outputs

The system embodies agentic properties such as autonomy, goal-orientation, adaptability, and deliberative planning, making it suitable for dynamic sports planning scenarios.

2. System Design

2.1 Application Design

The system follows an **agent-based modular architecture** inspired by layered and microservice design principles. At the core lies a Planner Agent that orchestrates multiple specialized components, each responsible for a distinct cognitive or operational function.

This architectural style is chosen because:

- Agent-based design aligns naturally with goal-driven AI behavior
- Modularity enables independent evolution of components
- Clear separation of concerns improves maintainability
- The system can be extended to multi-agent collaboration in the future

The architecture logically separates perception, reasoning, planning, execution, and monitoring layers while maintaining cohesive agent control.

2.2 Process Flow

The end-to-end workflow of the system proceeds as follows:

1. The user submits a high-level sports planning goal.
2. The Planner Agent initializes a new planning session.
3. The Goal Interpreter analyzes and contextualizes the goal.
4. The Task Decomposition Engine breaks the goal into subtasks.
5. The Dependency Manager establishes execution order.
6. The Resource Validator checks availability using mock tools.
7. The Scheduler generates an optimized execution plan.
8. The Execution Monitor tracks task completion.
9. The Output Generator produces the final structured result.

This flow demonstrates a complete **sense–think–plan–act–observe** loop characteristic of Agentic AI systems.

2.3 Information Flow

Information flow within the system is coordinated centrally by the Planner Agent. User inputs are progressively transformed from abstract goals into concrete data structures representing tasks, dependencies, and schedules. Each component consumes and produces structured information that feeds into the next stage of reasoning or execution.

Bidirectional communication ensures that execution feedback and validation results can dynamically influence planning decisions, enabling adaptive behavior.

2.4 Components Design

Planner Agent

Acts as the central decision-making entity. It maintains global context, invokes sub-components, manages state transitions, and ensures goal completion.

Goal Interpreter

Transforms high-level user goals into structured internal representations by identifying intent, scope, and constraints.

Task Decomposition Engine

Breaks complex goals into atomic, actionable subtasks using hierarchical planning principles.

Dependency Manager

Analyzes relationships among tasks and enforces correct execution order by identifying prerequisites and constraints.

Scheduler

Generates an optimized execution plan that respects dependencies, resource constraints, and efficiency goals.

Execution Monitor

Tracks task progress, updates execution status, and detects anomalies or failures.

Output Generator

Aggregates results into a structured, human-readable format suitable for sports planning insights.

2.5 Key Design Considerations

- **Scalability:** Modular design allows additional planning domains and tasks.
- **Extensibility:** New tools and planners can be integrated without redesign.
- **Fault Tolerance:** Execution monitoring enables graceful handling of failures.
- **Modularity:** Components operate independently with well-defined interfaces.
- **Agent Autonomy:** The system minimizes external intervention during planning and execution.

2.6 API Catalogue

The system exposes conceptual APIs such as:

- Goal Submission API for user interaction
- Task Planning API for internal reasoning
- Resource Validation API for tool checks
- Execution Status API for monitoring
- Result Retrieval API for output delivery

Each API defines clear input/output contracts and supports loose coupling between components.

3. Data Design

Data design plays a critical role in the Sports Planning Assistant Agent, as the effectiveness of agentic reasoning depends on structured, consistent, and context-aware data representations. The system must support dynamic planning, dependency tracking, and execution monitoring, all of which require well-defined data entities and controlled access mechanisms.

3.1 Data Model

The data model is conceptual in nature and designed to represent the cognitive and operational state of the Planner Agent. Instead of traditional transactional data, the system primarily manages planning-oriented data objects.

Key conceptual entities include:

- Goal Entity: Represents the high-level objective provided by the user, including goal description, priority, constraints, and expected outcome.
- Task Entity: Defines individual actionable steps derived from the goal. Each task contains attributes such as task ID, task type, prerequisites, execution status, and estimated effort.
- Dependency Entity: Captures relationships between tasks, ensuring that execution order respects logical and resource-based constraints.
- Resource Entity: Represents abstract or simulated resources such as datasets, computation modules, or analysis tools validated through mock interfaces.
- Schedule Entity: Stores the ordered execution plan generated by the Scheduler, including timestamps and sequencing information.
- Agent State Entity: Maintains contextual information such as current planning phase, execution progress, and historical decisions.

This structured data model enables the agent to reason, plan, and adapt dynamically.

3.2 Data Access Mechanism

Data access within the system is centrally coordinated by the Planner Agent to preserve consistency and avoid race conditions. Components do not directly manipulate shared data; instead, they interact through controlled interfaces.

The data access mechanism follows these principles:

- Read-only access for interpretation and validation components
- Controlled write access during planning and execution stages
- Immutable historical logs for auditing and debugging
- Context-aware data sharing across planning phases

This approach aligns with agentic system design by maintaining a single source of truth while supporting autonomous reasoning.

3.3 Data Retention Policies

Data retention policies are designed based on the lifecycle of planning sessions.

- Ephemeral Data: Temporary planning artifacts such as intermediate task decompositions and partial schedules are discarded after session completion.
- Session Data: Agent state and execution logs are retained for the duration of a session to support monitoring and replanning.
- Persistent Data: Final outputs and summary results may be stored for evaluation, reporting, or future reference.

These policies balance performance efficiency with traceability and reproducibility.

3.4 Data Migration

Although the system initially operates in a simulated environment, it is designed to support future data expansion. Versioned schemas and backward-compatible data representations allow seamless migration when new planning attributes or entities are introduced.

4. Interfaces

Interfaces define how users, tools, and future agents interact with the Sports Planning Assistant Agent. A well-designed interface layer ensures loose coupling, extensibility.

User-to-Agent Interface

This interface enables users to submit high-level sports planning goals and receive structured outputs. It abstracts system complexity and presents a simplified interaction model.

Agent-to-Tool Interface

The agent communicates with mock tools to validate resource availability, simulate data access, or verify execution feasibility. These interfaces mimic real-world integrations without external dependencies.

Agent-to-Agent Interface (Future Scope)

The architecture supports future multi-agent collaboration, where specialized agents (e.g., analytics agent, scheduling agent) can coordinate through standardized protocols.

5. State and Session Management

State and session management is fundamental to agentic behavior. The Planner Agent maintains a persistent internal state that captures reasoning context, planning history, and execution progress.

Key managed states include:

- Goal interpretation state
- Task decomposition state
- Dependency resolution state
- Execution monitoring state

Session continuity ensures that the agent can pause, resume, or adapt plans without losing contextual awareness. This enables dynamic replanning, a hallmark of intelligent agents.

6. Caching

Caching mechanisms are incorporated to improve efficiency and responsiveness, particularly during repeated or similar planning requests.

Caching strategies include:

- Task Plan Cache: Stores previously generated task decompositions for common goals.
- Resource Validation Cache: Avoids redundant mock tool calls.
- Intermediate Reasoning Cache: Retains partial planning results during multi-step reasoning loops.

Caching reduces computational overhead and supports faster decision-making without compromising correctness.

7. Non-Functional Requirements

Non-functional requirements ensure that the system meets quality standards beyond functional correctness.

7.1 Security Aspects

Although the system operates in a simulated environment, security considerations remain essential. The design incorporates:

- Input validation to prevent malformed or ambiguous goals
- Controlled access to planning APIs
- Data integrity checks for execution results
- Role-based access for administrative operations

These measures ensure robustness and future readiness for real-world deployment.

7.2 Performance Aspects

Performance requirements focus on planning efficiency and scalability.

Key performance goals include:

- Low response time for goal interpretation
- Efficient task decomposition and scheduling
- Ability to handle multiple concurrent planning requests
- Predictable execution timelines under load

The modular agent-based design ensures that performance bottlenecks can be isolated and optimized independently.

8. References

The system design is informed by established research and literature in artificial intelligence and autonomous planning:

- Artificial Intelligence: A Modern Approach, Stuart Russell & Peter Norvig
- Research on intelligent agents and deliberative planning systems
- Academic studies on Agentic AI architectures
- Literature on AI-driven sports analytics and decision-support systems