MEDICAPS UNIVERSITY

Datagami
“Lead Digital Technology”

Project - High Level Design

on Autonomous Legal Researcher 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	Saransh Pillai	EN22CS301870
2	Sandeep Rathore	EN22CS301861
3	Shreya Kabra	EN22CS301929
4	Shivanshi Shrivastava	EN22CS301915
5	Shivangi Patidar	EN22CS301910

Group Name:Group 02D8

Project Number:AAI-37

Industry Mentor Name:

University Mentor Name:Prof. Pradeep Baniya

Academic Year:2025-26

Table of Contents

1. Introduction

1.1 Scope of the Document

This document presents the **High-Level Design (HLD)** for the *Autonomous Legal Researcher Agent*.
The purpose of this document is to describe the **overall system architecture, major components, data flow, external interfaces, tools, APIs, and quality attributes** of the system.

This HLD serves as a **blueprint for system implementation**, while detailed class-level logic, algorithms, and code structures are addressed separately in the **Low-Level Design (LLD)**.

1.2 Intended Audience

This document is intended for:

- Faculty evaluators and academic reviewers
- Software architects and system designers
- Developers implementing agent-based AI systems
- Students studying Agentic AI system design

1.3 System Overview

The Autonomous Legal Researcher Agent is a **goal-oriented Agentic AI system** designed to assist users in conducting legal research efficiently.

The system:

- Accepts legal research queries from users via a simple text-based interface
- Uses **LangGraph** to orchestrate LLM-driven planning and agent control flow
- Searches trusted legal and government sources using **SerpAPI**
- Extracts legal content from web pages using **BeautifulSoup** (static scraping) and **Selenium** (dynamic scraping)
- Summarizes extracted legal information using **Large Language Models (LLMs)**

- Stores research outputs in a **Vector Database (e.g., ChromaDB / FAISS)** for semantic retrieval and reuse

Unlike traditional chatbots, the system exhibits **controlled autonomy**, enabling it to plan, execute, and adapt research steps within predefined boundaries.

2. System Design

2.1 Application Design

The system adopts a **layered, agent-based application design** that integrates architectural structure and execution logic.

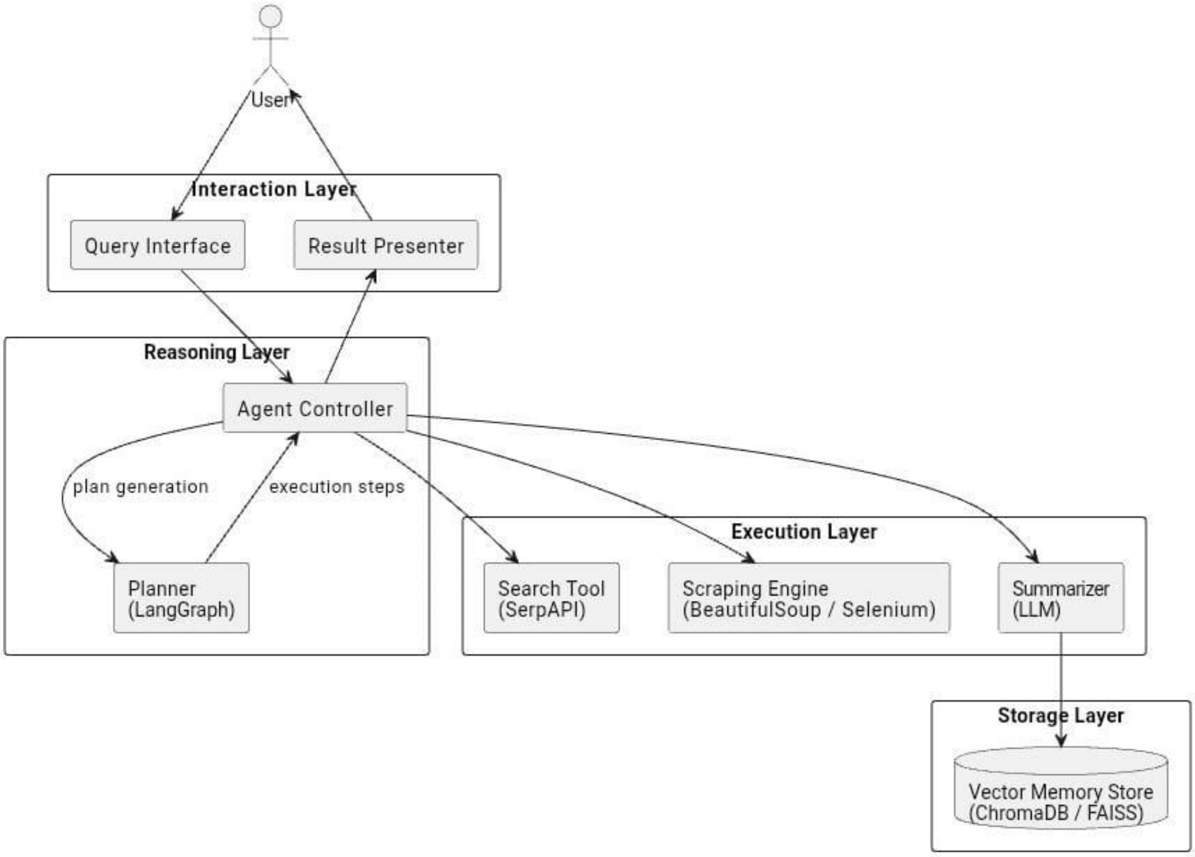
The application is organized into the following layers:

- **Interaction Layer:** Handles user input and result presentation
- **Reasoning Layer:** Uses LangGraph to manage agent planning and control flow
- **Execution Layer:** Performs search, scraping, and summarization tasks
- **Storage Layer:** Maintains long-term semantic memory using a vector database

At the core of the application is the **Agent Controller**, which orchestrates:

- Planner invocation
- Tool execution sequencing
- Data persistence and response generation

This design ensures modularity, scalability, and maintainability.

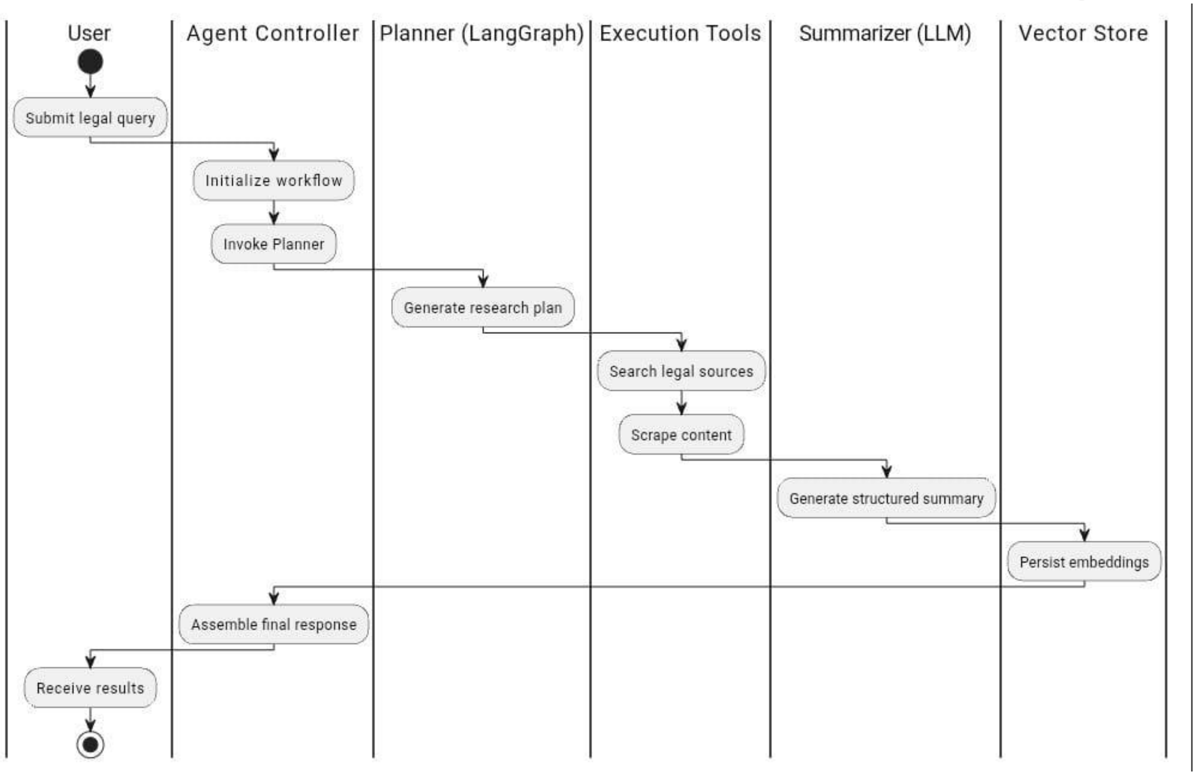


2.2 Process Flow

The high-level execution flow of the system is as follows:

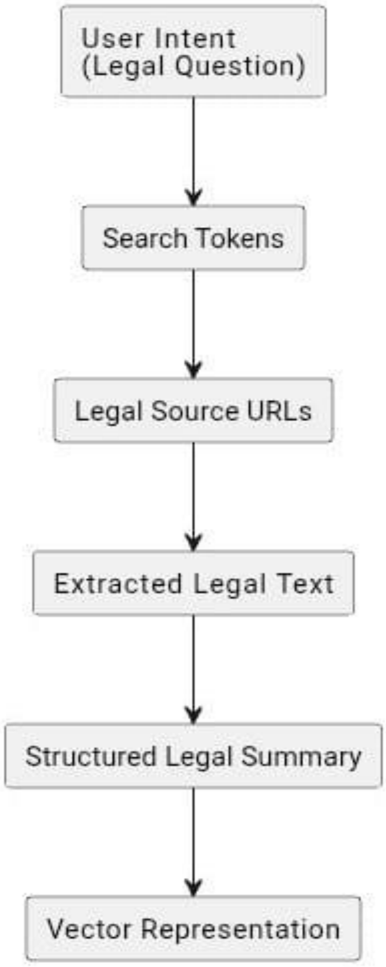
1. User submits a legal research query
2. Agent Controller initiates the research workflow
3. Planner (LangGraph) generates a structured research plan
4. Search Tool (SerpAPI) identifies relevant legal and government sources
5. Scraper extracts legal content using BeautifulSoup and Selenium
6. Summarizer (LLM) produces a structured legal summary
7. Knowledge Store (Vector Database) persists the research output
8. Final response is delivered to the user

This flow demonstrates the **autonomous yet controlled** nature of the agent.



2.3 Information Flow

- **Input:** User legal query
- **Intermediate Artifacts:**
  - Search keywords
  - Source URLs
  - Extracted legal text
  - Generated summaries
  - Vector embeddings
- **Output:** Structured legal summary with source references

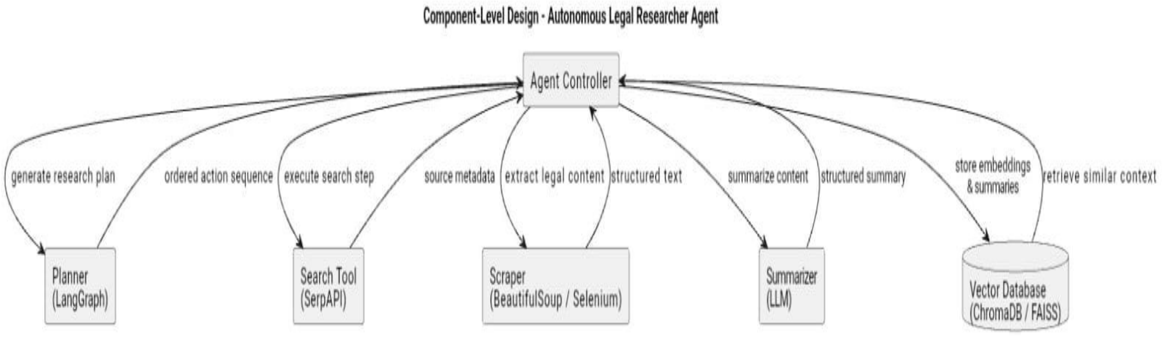


2.4 Component Design

The system is composed of the following high-level components:

- **Agent Controller:**
  Manages the complete lifecycle of query execution and coordinates all subsystems.
- **Planner (LangGraph):**
  Interprets user intent and generates an ordered sequence of research actions.
- **Search Tool (SerpAPI):**
  Identifies authoritative legal and government sources relevant to the query.
- **Scraper (BeautifulSoup & Selenium):**
  Extracts clean and structured legal text from static and dynamic web pages.
-

- **Summarizer (LLM):**
  Converts complex legal content into concise, structured summaries.
- **Vector Database (ChromaDB / FAISS):**
  Stores semantic representations of research data and enables similarity-based retrieval.



Component-Level Design - Autonomous Legal Researcher Agent

2.5 Key Design Considerations

- **Modularity:** Each component performs a single well-defined function
- **Loose Coupling:** Components interact via interfaces rather than tight dependencies
- **Explainability:** Execution steps can be traced and audited
- **Controlled Autonomy:** Automated decision-making within safe operational limits

2.6 API Catalogue

API / Tool Name	Purpose
LangGraph	Agent planning, reasoning, and control flow
LLM API	Query understanding and summarization
SerpAPI	Retrieval of legal and government sources
BeautifulSoup	Static HTML content extraction
Selenium	Dynamic web scraping for JS-heavy portals
Vector DB API	Semantic storage and retrieval of research data

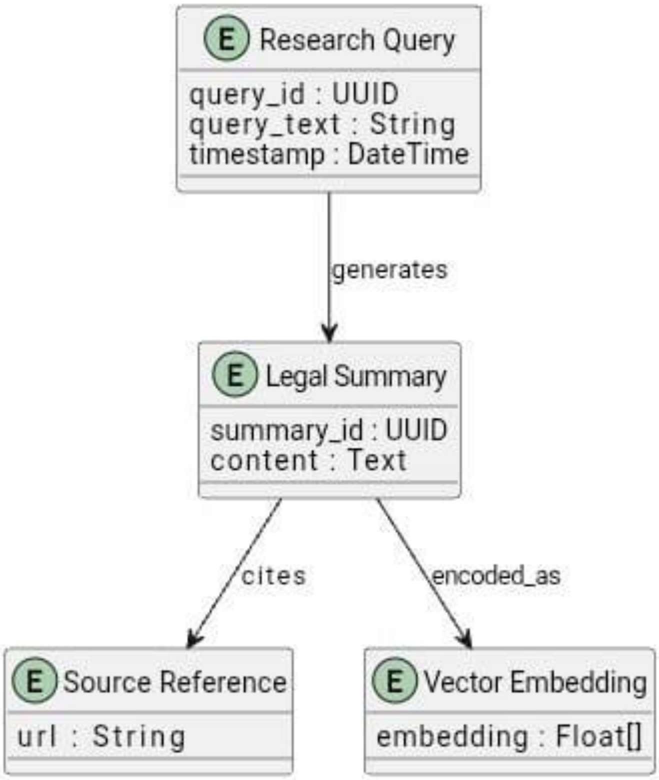
External Dependencies

- LangGraph framework
- LLM service provider
- SerpAPI search service
- Government and judicial legal portals (read-only access)

3. Data Design

3.1 Data Model Overview

Field	**Description**
Query	User-submitted legal question
Summary	Generated structured legal summary
Sources	Referenced legal URLs
Embeddings	Vector representation of summaries
Timestamp	Date and time of execution



3.2 Data Access Mechanism

- **Phase 1:** File-based text storage for simplicity
- **Phase 2:** Vector database storage (ChromaDB / FAISS) using embeddings for semantic retrieval

3.3 Data Retention Policies

- Data retained for academic learning and system improvement
- No personal or sensitive user information stored
- Stored data used solely for research reference

3.4 Data Migration

- Gradual migration from text storage to vector-based storage
- Ensures backward compatibility and data integrity

4. Interfaces

- **User Interface:** Minimal text-based interface for query submission and result display
- **Backend Interface:** Python-based internal APIs for inter-module communication

5. State and Session Management

- Stateless request handling for scalability
- Long-term context maintained via the Vector Database

6. Caching Strategy

- Frequently accessed summaries cached locally
- Reduces redundant processing and LLM invocations
- Improves overall system responsiveness

7. Non-Functional Requirements

7.1 Security Considerations

- Access limited to trusted and authoritative sources
- Read-only interaction with external systems
- No modification or redistribution of source content

7.2 Performance and Scalability

- Optimized search and summarization workflows
- Modular architecture supports horizontal scaling
- Efficient API usage under academic constraints

8. References

[1] LangChain, “LangGraph Documentation,” 2024. [Online]. Available: https://python.langchain.com/docs/langgraph

[2] SerpAPI, “SerpAPI Documentation,” 2024. [Online]. Available: https://serpapi.com

[3] L. Richardson, “BeautifulSoup Documentation,” 2024. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/

[4] SeleniumHQ, “Selenium WebDriver Documentation,” 2024. [Online]. Available: https://www.selenium.dev

[5] Chroma, “ChromaDB Documentation,” 2024. [Online]. Available: https://www.trychroma.com

[6] Facebook AI Research, “FAISS: A Library for Efficient Similarity Search,” 2024. [Online]. Available: https://github.com/facebookresearch/faiss

[7] TrueLaw AI, “Legal Autonomous Agents Framework,” 2024. [Online]. Available: https://www.truelaw.ai/blog/legal-ai-autonomous-agents-frameworks-for-compound-legal-ai-systems