Multi-Agent Insurance System

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.	Nandini Sharma	EN22CS301635
2.	Mustafa Shaikh	EN22CS301624
3.	Nishkarsh Sharma	EN22CS301659
4.	Naman Mehta	EN22CS301629
5.	Naman Raghuvanshi	EN22CS301632

Group Name: 09D6

Project Number: AAI-20

Industry Mentor Name:

University Mentor Name: Prof. Nishant Shrivastav

Academic Year: January-June 2026

Problem Statement & Objectives

1. Problem Statement

The insurance industry deals with highly complex policy documents, regulatory clauses, and compliance requirements. Manual analysis of policies is:

- Time-consuming
- Error-prone
- Difficult to scale
- Susceptible to inconsistency

Traditional AI systems struggle with **context overflow**, hallucinations, and lack of traceability when processing large insurance documents.

There is a need for a **collaborative, specialized multi-agent system** that can automate policy research and structured report generation while maintaining accuracy, compliance, and data security.

2. Project Objectives

The main objectives of this project are:

- Design and implement a **Multi-Agent Insurance System**
- Separate responsibilities between specialized agents
- Automate insurance policy research and reporting
- Ensure structured and traceable outputs
- Implement secure local LLM processing
- Maintain workflow state management
- Reduce hallucinations through structured context passing
- Provide a scalable and modular architecture

3. Scope of the Project

The project scope includes:

- Development of a **two-agent collaborative system**
- Integration of:
    - LLM for policy analysis
    - External API for data retrieval
    - MongoDB for persistence
- REST API-based interaction
- Sequential task orchestration using agent framework
- Error handling, retry mechanisms, and logging
- Structured output generation for insurance policy reports

Proposed Solution

The system implements a **Collaborative Multi-Agent Architecture** where tasks are divided between:

- Researcher Agent (Fact extraction & evidence grounding)
- Writer Agent (Synthesis & structured documentation)

Orchestration is handled using **CrewAI**, ensuring sequential execution and automatic context passing.

LLM inference is handled locally using **Ollama** to ensure privacy and PII protection.

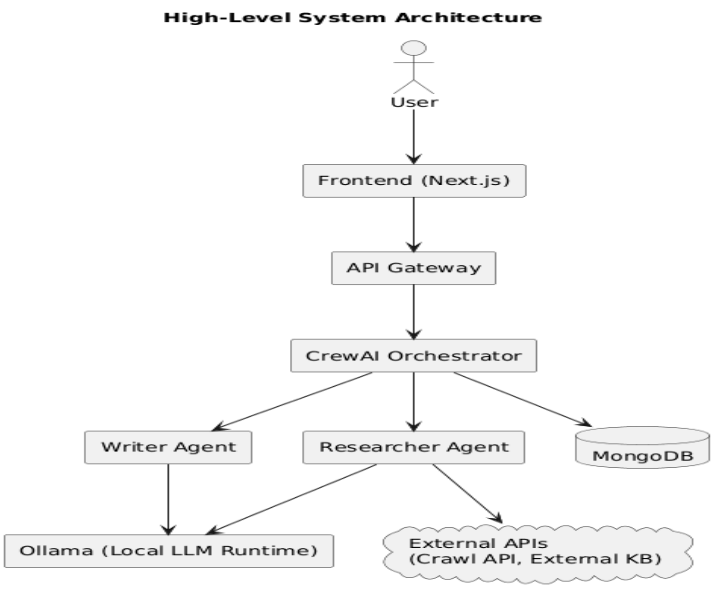
Data persistence is implemented using **MongoDB**.

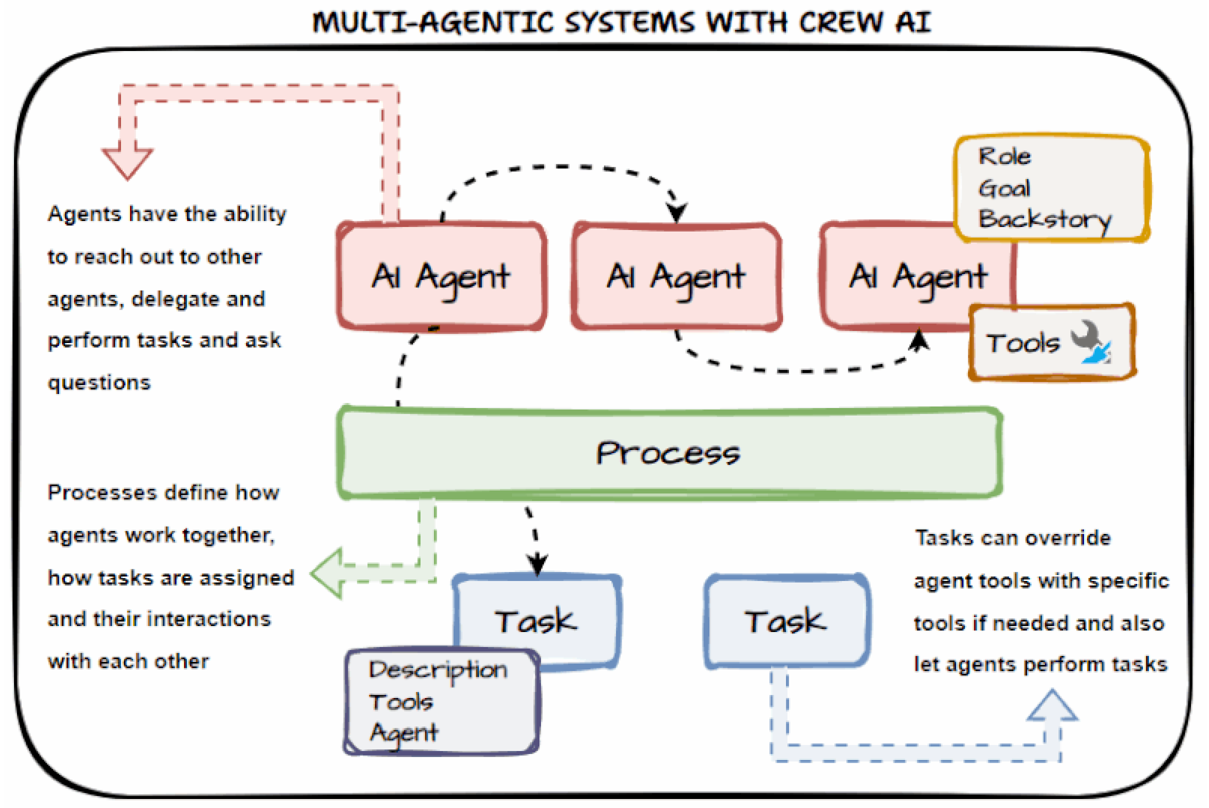
1. Key Features

- Agent Specialization (Researcher & Writer)
- Sequential Task Orchestration
- Automatic Context Passing
- Structured JSON Evidence Generation
- RAG-based Information Retrieval
- Local LLM Processing (Privacy-first)
- Workflow State Management
- Retry with Exponential Backoff
- Circuit Breaker Pattern
- Semantic Caching
- REST API Endpoints
- Structured Report Generation

2. Overall Architecture / Workflow

High-Level Architecture


High-Level System Architecture
User
Frontend (Next.js)
API Gateway
CrewAI Orchestrator
Writer Agent
Researcher Agent
MongoDB
Ollama (Local LLM Runtime)
External APIs (Crawl API, External KB)


MULTI-AGENTIC SYSTEMS WITH CREW AI
Agents have the ability to reach out to other agents, delegate and perform tasks and ask questions
AI Agent
AI Agent
AI Agent
Role
Goal
Backstory
Tools
Process
Processes define how agents work together, how tasks are assigned and their interactions with each other
Task
Description
Tools
Agent
Task
Tasks can override agent tools with specific tools if needed and also let agents perform tasks

System Layers

Client Layer
User submits insurance query

API Layer
Receives request and triggers workflow

Orchestration Layer

- Managed by CrewAI
- Handles task sequencing
- Transfers structured context

Agent Layer

- Researcher Agent
- Writer Agent

Data Layer

- MongoDB collections:

    - workflow_states
    - research_data
    - final_outputs

External Services

- Ollama (LLM inference)
- Crawl API

Workflow Execution Steps

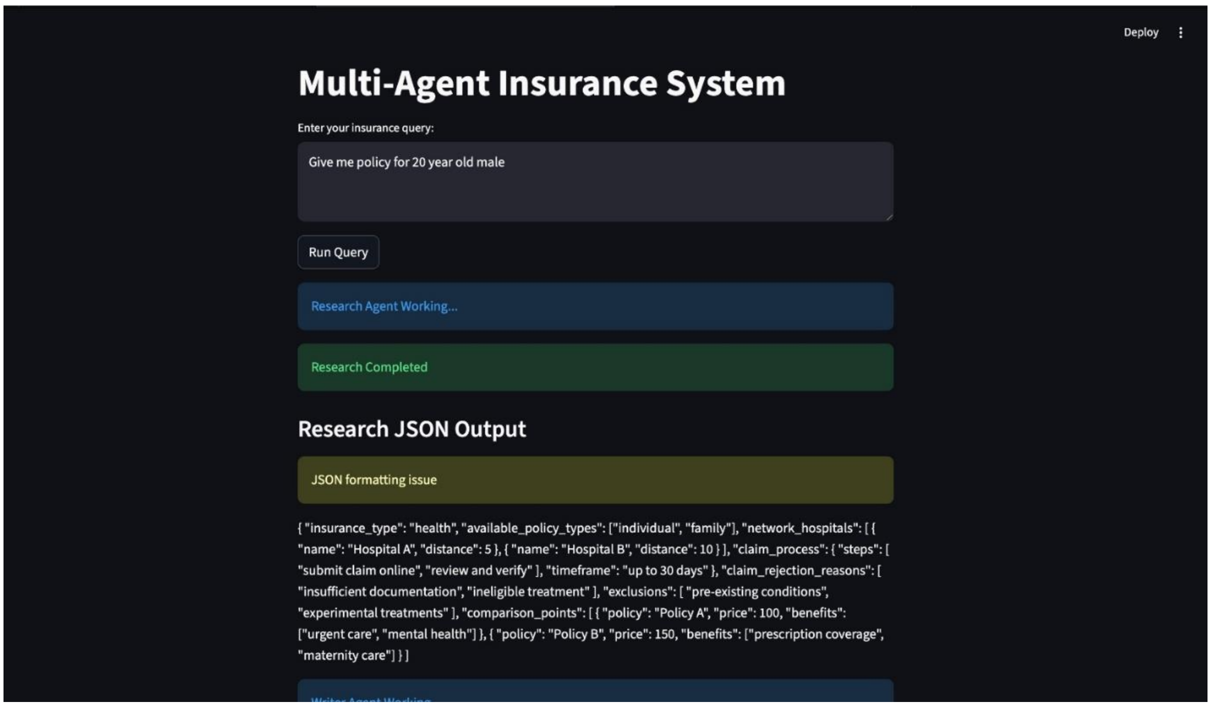
1. User submits insurance query
2. API creates workflow ID
3. Researcher Agent:
    - Queries LLM
    - Fetches external data
    - Validates findings
    - Generates structured JSON
4. Writer Agent:
    - Receives structured context
    - Synthesizes report
    - Applies formatting & compliance
5. Final output stored in database
6. Status updated to “Completed”
7. Structured response returned to user

3. Tools & Technologies Used

Layer	**Technology**
Agent Orchestration	CrewAI
LLM Integration	LangChain
LLM Runtime	Ollama
Database	MongoDB
Backend	FastAPI / Flask
Programming Language	Python 3.9+
External Data	Crawl API
Optional Caching	Redis
Logging	OpenTelemetry
Security	IAM Integration

Results & Output

1. Screenshots / Outputs



Multi-Agent Insurance System

Enter your insurance query:

Give me policy for 20 year old male

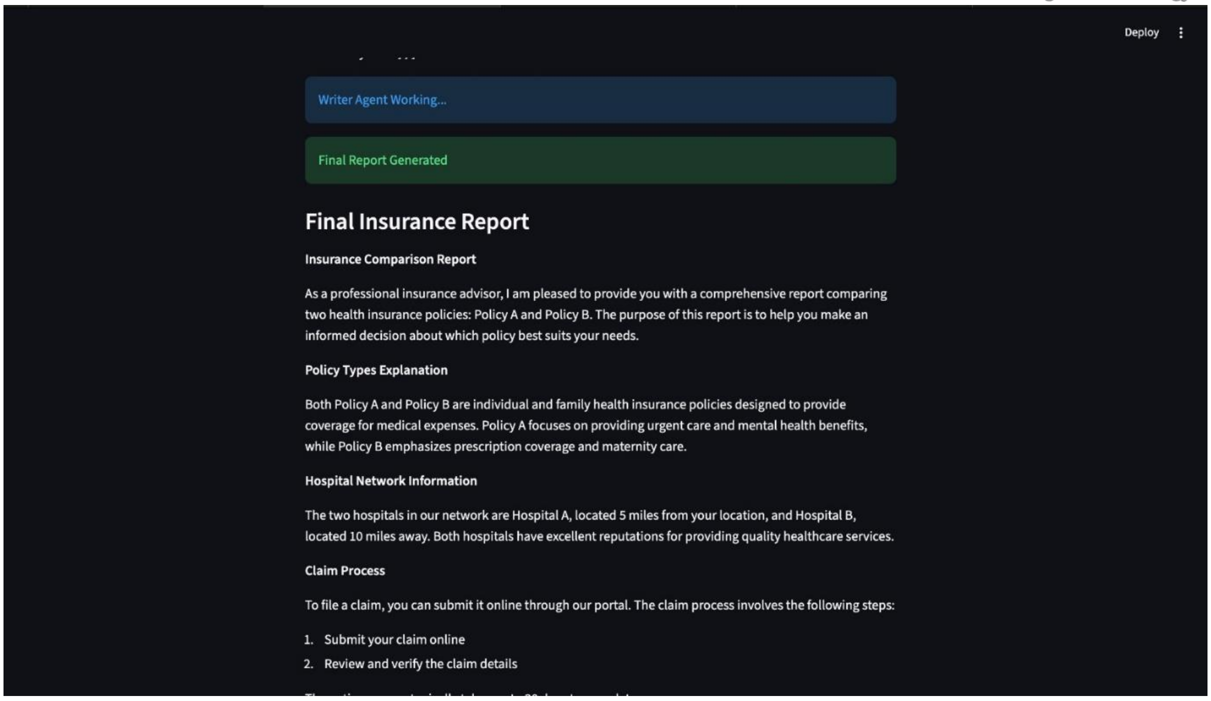
Research Agent Working...

Research Completed

Research JSON Output

JSON formatting issue

```
{ "insurance_type": "health", "available_policy_types": ["individual", "family"], "network_hospitals": [ {
"name": "Hospital A", "distance": 5 }, { "name": "Hospital B", "distance": 10 } ], "claim_process": { "steps": [
"submit claim online", "review and verify" ], "timeframe": "up to 30 days" }, "claim_rejection_reasons": [
"insufficient documentation", "ineligible treatment" ], "exclusions": [ "pre-existing conditions",
"experimental treatments" ], "comparison_points": [ { "policy": "Policy A", "price": 100, "benefits":
["urgent care", "mental health"] }, { "policy": "Policy B", "price": 150, "benefits": ["prescription coverage",
"maternity care"] } ]
```



Writer Agent Working...

Final Report Generated

Final Insurance Report

Insurance Comparison Report

As a professional insurance advisor, I am pleased to provide you with a comprehensive report comparing two health insurance policies: Policy A and Policy B. The purpose of this report is to help you make an informed decision about which policy best suits your needs.

Policy Types Explanation

Both Policy A and Policy B are individual and family health insurance policies designed to provide coverage for medical expenses. Policy A focuses on providing urgent care and mental health benefits, while Policy B emphasizes prescription coverage and maternity care.

Hospital Network Information

The two hospitals in our network are Hospital A, located 5 miles from your location, and Hospital B, located 10 miles away. Both hospitals have excellent reputations for providing quality healthcare services.

Claim Process

To file a claim, you can submit it online through our portal. The claim process involves the following steps:

1. Submit your claim online
2. Review and verify the claim details

The system successfully generates:

- Structured JSON research outputs
- Formatted insurance coverage reports
- Workflow status logs
- Stored research evidence with traceability

2. Reports / Dashboards / Models

The system produces:

- Policy Coverage Summary Reports
- Structured JSON Output Files
- Research Confidence Scores
- Workflow Logs
- Quality Metrics:
  - Completeness Score
  - Accuracy Score
  - Readability Score

MongoDB stores:

- Research evidence
- Final formatted outputs
- Workflow state transitions

3. Key Outcomes

- Successfully implemented a **collaborative multi-agent system**
- Reduced hallucination through task isolation
- Achieved structured, traceable outputs
- Ensured PII-safe processing using local LLM
- Built scalable modular architecture
- Implemented error handling & retry strategy
- Enabled state persistence and resume capability
- Demonstrated practical application of Agentic AI in Insurance

Conclusion

The Multi-Agent Insurance System demonstrates how **Agentic AI architectures** can transform complex insurance workflows.

By dividing responsibilities between:

- Researcher Agent (Fact extraction)
- Writer Agent (Synthesis & formatting)

the system significantly improves:

- Accuracy
- Traceability
- Modularity
- Scalability
- Compliance

The use of CrewAI for orchestration, Ollama for secure local LLM inference, and MongoDB for state persistence enabled us to build a production-ready architecture blueprint.

This project strengthened our understanding of:

- Multi-agent coordination
- RAG pipelines
- Workflow orchestration
- LLM integration
- State management
- Error handling strategies
- Secure AI system design

Future Scope & Enhancements

1. Add More Specialized Agents

- Compliance Validator Agent
- Fraud Detection Agent
- Quality Assurance Agent

2. Scalability Enhancements

- Horizontal scaling with microservices
- Load balancing
- Message queue (Kafka / RabbitMQ)

3. Performance Optimization

- Advanced semantic caching
- Async processing
- Database sharding

4. Monitoring & Observability

- Distributed tracing
- Centralized log aggregation

5. Security Enhancements

- OAuth2 / JWT Authentication
- Role-based Access Control
- End-to-End encryption

6. Feature Extensions

- PDF export generation
- Real-time dashboard
- Multi-language policy analysis
- Regulatory auto-updates
- Agent learning with feedback loop