Multi-Agent Manufacturing 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
01	Om Tiwari	EN22CS301669
02	Paridhi Shirwalkar	EN22CS301684
03	Nitesh Chourasiya	EN22CS301660
04	Mradul Jain	EN22CS301616

Group Name: Group 01D6

Project Number: AAI-12

Industry Mentor Name:

University Mentor Name: Nishant Shrivastav

Academic Year: 2025-2026

MEDICAPS UNIVERSITY

Datagami
"Lead Digital Technology"

Table of Contents

1. Problem Statement & Objectives

1.1 Problem Statement

Multi-Agent Manufacturing System

1.2 Project Objectives

The objectives of this project are:

- Design a collaborative multi-agent architecture for manufacturing sourcing.
- Implement role specialization between Researcher and Writer agents.
- Establish a structured hand-off protocol using JSON artifacts.
- Develop a centralized Orchestrator to manage workflow execution.
- Implement session-based state tracking and run management.
- Enforce schema validation to ensure clean and reliable outputs.
- Generate structured supplier comparison reports.
- Demonstrate advanced agent orchestration using LLM APIs.
- Build a user interface to capture sourcing requirements and present results.

1.3 Scope of the Project

In Scope:

- Multi-agent architecture design.
- Researcher Agent for supplier data sourcing.
- Writer Agent for report synthesis and formatting.
- Structured JSON-based hand-offs between agents.
- Workflow orchestration with defined execution stages.
- State management with unique Run IDs.
- File-based artifact storage.
- Tab-based UI for structured output visualization.

Out of Scope:

- Direct integration with enterprise ERP systems.
- Live supplier database APIs.
- Automated financial transactions or procurement orders.
- Enterprise-scale distributed deployment.

2. Proposed Solution

The proposed solution is a Multi-Agent Manufacturing System built using Python and CrewAI, leveraging LLM APIs to power intelligent agent reasoning.

The system follows a modular layered architecture:

- Presentation Layer - Streamlit-based user interface.
- Application Layer - Orchestrator controlling execution flow.
- Agent Layer - Researcher and Writer agents.
- Validation Layer - Schema enforcement and structured output validation.
- Storage Layer - Artifact persistence and workflow state tracking.
- LLM Layer - External language model provider.

This structured architecture ensures separation of responsibilities and scalable workflow management.

2.1 Key Features

- Role-based multi-agent collaboration
- Researcher and Writer agent specialization
- Structured JSON hand-off protocol
- Centralized workflow orchestration
- Schema validation using Pydantic
- Unique Run ID generation for each execution
- Session-based API key handling
- Tab-based output interface (Report | Ranked Suppliers | Raw Research | Download)
- Artifact storage per run
- Controlled retries and state tracking

2.2 Overall Architecture / Workflow

Step 1- User Authentication

The user enters the LLM API key in the sidebar. Once authenticated, the Manufacturing Sourcing Query form becomes accessible.

Step 2- Structured Input Capture

The user provides sourcing requirements such as:

- Manufacturing process
- Material
- Location preference

- Minimum monthly capacity
- Required certifications

Step 3- Orchestrator Initialization

The Orchestrator:

- Generates a unique Run ID.
- Initializes workflow state tracking.
- Prepares storage directory for artifacts.

Step 4- Research Phase

The Researcher Agent:

- Converts structured inputs into sourcing prompts.
- Queries the LLM API.
- Collects supplier details.
- Produces structured raw supplier data (JSON).

Step 5- Validation

The Orchestrator validates the raw output against predefined schemas to ensure structural correctness.

Step 6- Writing Phase

The Writer Agent:

- Cleans and normalizes supplier data.
- Scores and ranks suppliers.
- Generates a detailed supplier comparison report.
- Identifies risks and missing information.

Step 7- Artifact Storage

Artifacts are stored under:

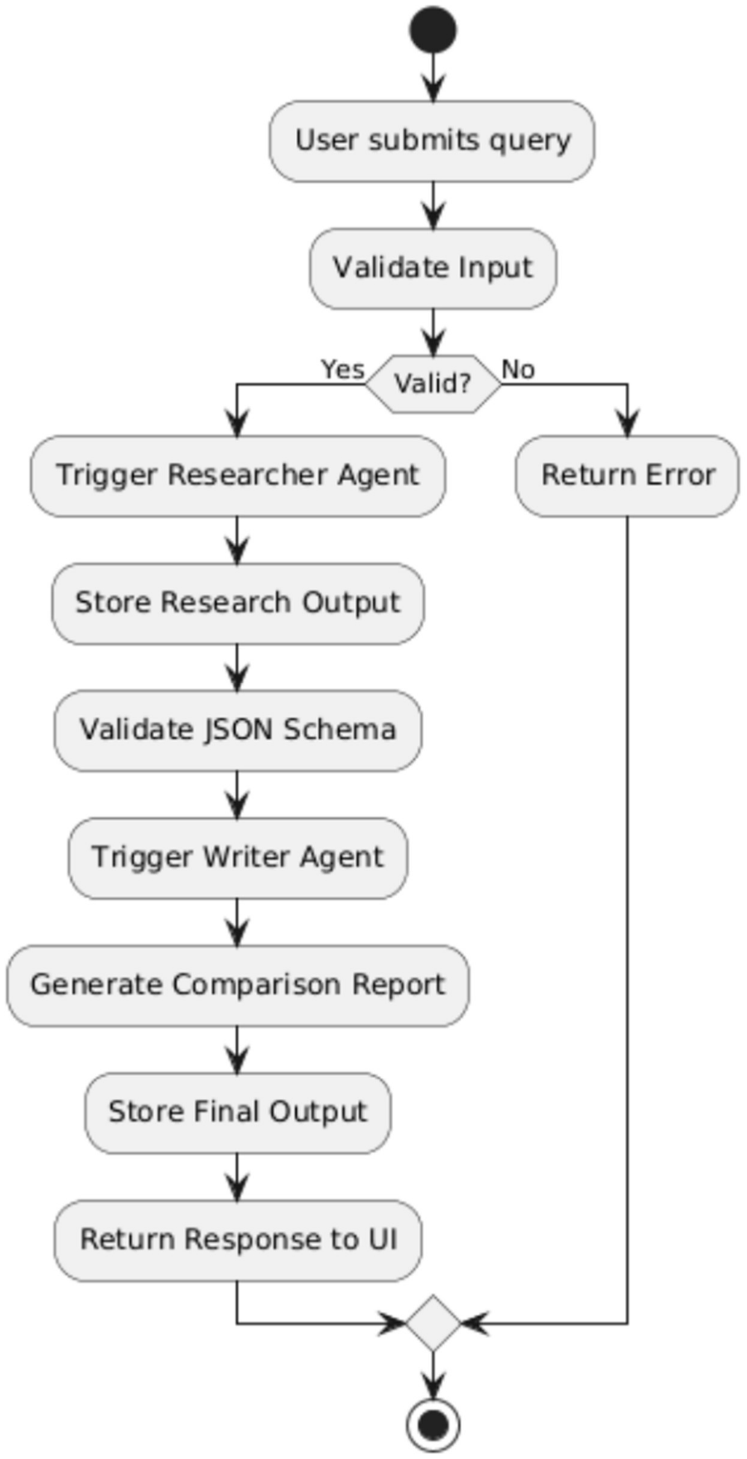
artifacts/run_<timestamp>/

Step 8- Output Presentation

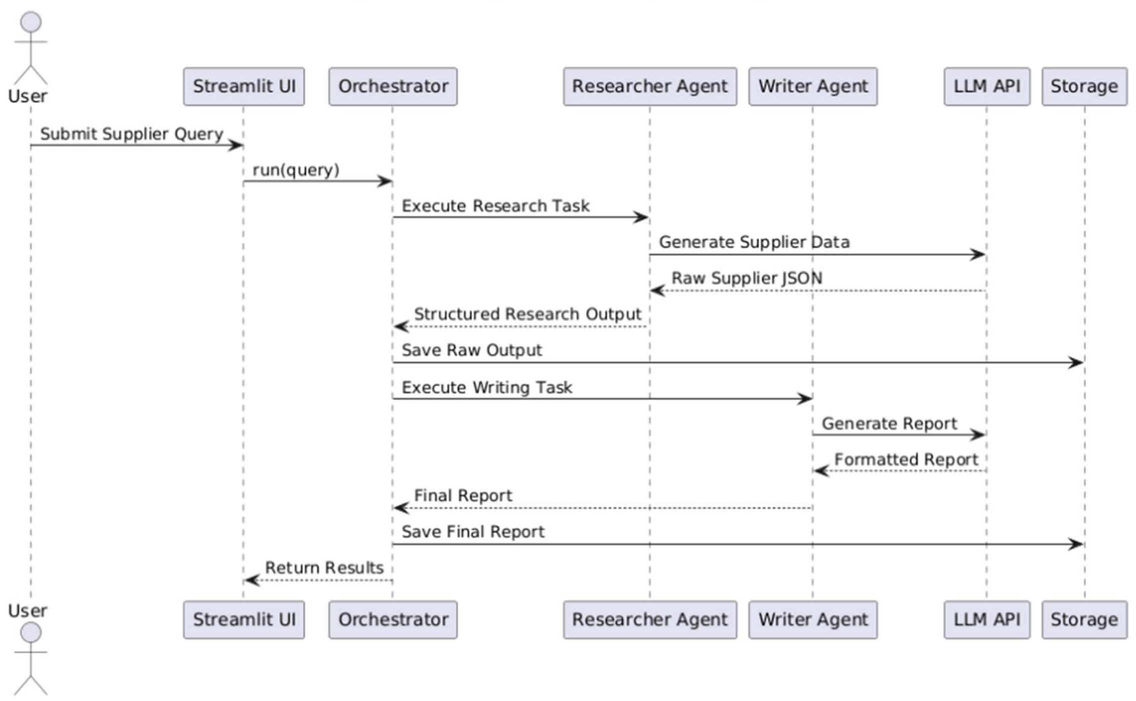
The UI displays results in structured tabs:

- Final Report
- Ranked Suppliers
- Raw Research Output
- Downloadable Files

Multi-Agent Workflow Activity



Supplier Sourcing Workflow - Sequence Diagram



2.3 Tools & Technologies Used

- Python 3.10+
- CrewAI (LangChain-based orchestration framework)
- Gemini / LLM API
- Streamlit (UI layer)
- Pydantic (Schema validation)
- JSON (Structured data exchange)
- Git & GitHub (Version control)
- Virtual Environment (venv)

3. Implementation Details

3.1 Agent Design (Researcher & Writer)

The system follows a role-based agent specialization model where each agent is responsible for a clearly defined task within the workflow.

Researcher Agent

The Researcher Agent is responsible for sourcing supplier information based on structured manufacturing requirements provided by the user.

Responsibilities:

- Interpret structured input parameters (process, material, location, capacity, certifications).
- Convert structured inputs into well-defined prompts for the LLM.
- Generate a list of potential suppliers.
- Extract key attributes such as:
    - Supplier name
    - Location
    - Manufacturing capabilities
    - Certifications
    - Capacity information
    - Contact details
- Return output in structured JSON format.

The Researcher Agent focuses purely on information gathering and does not perform ranking or formatting logic. This separation ensures clarity of responsibilities and reduces cross-agent interference.

Writer Agent

The Writer Agent is responsible for transforming raw supplier data into structured and decision-ready outputs.

Responsibilities:

- Parse and normalize raw supplier JSON.
- Validate completeness of supplier attributes.
- Rank suppliers based on requirement fit.
- Identify missing fields or risks.
- Generate a structured Supplier Comparison Report.

The Writer Agent produces:

- A cleaned and normalized structured_suppliers.json
- A formatted report.md containing:
    - Executive summary
    - Comparison table

- Risk & gap analysis
- Recommendation insights

This separation between Researcher and Writer ensures modularity, maintainability, and scalable design.

3.2 Orchestrator Logic

The Orchestrator acts as the central workflow controller and manages the end-to-end execution lifecycle.

Core Responsibilities:

- Generate a unique Run ID for each session.
- Initialize storage directories.
- Trigger agent execution sequentially.
- Handle state transitions.
- Validate outputs before passing them to the next stage.
- Capture and log execution results.

Execution Flow:

1. Receive structured user input.
2. Initialize workflow state (CREATED).
3. Trigger Researcher Agent.
4. Store raw research output.
5. Validate research output.
6. Trigger Writer Agent.
7. Store structured output and report.
8. Mark workflow as COMPLETED.

The Orchestrator ensures that agents do not directly communicate with each other. Instead, all communication is controlled through validated artifacts managed by the Orchestrator.

This approach improves:

- Traceability
- Error isolation
- Workflow control
- Extensibility

3.3 Schema Validation Strategy

To ensure reliable inter-agent communication, the system enforces structured schema validation between workflow stages.

The validation strategy includes:

- Defined Pydantic models for supplier data.
- Strict field validation (data types, required fields).
- Controlled default handling for missing values.
- Validation before passing artifacts to the next agent.

Validation occurs at two levels:

1. **Research Output Validation**
   - Ensures JSON structure matches expected supplier schema.
   - Confirms required attributes are present.
2. **Writer Output Validation**
   - Ensures ranking and structured fields are properly formatted.
   - Confirms report content consistency.

Benefits of schema validation:

- Reduces malformed JSON errors.
- Minimizes hallucinated structure.
- Improves reliability of structured outputs.
- Maintains consistency across workflow stages.

3.4 Storage & Artifact Management

The system implements run-based artifact storage for traceability and reproducibility.
For each execution, a unique directory is created:

artifacts/run_<timestamp>/

Stored files include:

- raw_suppliers.json- Researcher Agent output
- structured_suppliers.json- Writer Agent normalized output
- report.md- Final comparison report
- state.json- Workflow state tracking

Storage Responsibilities:

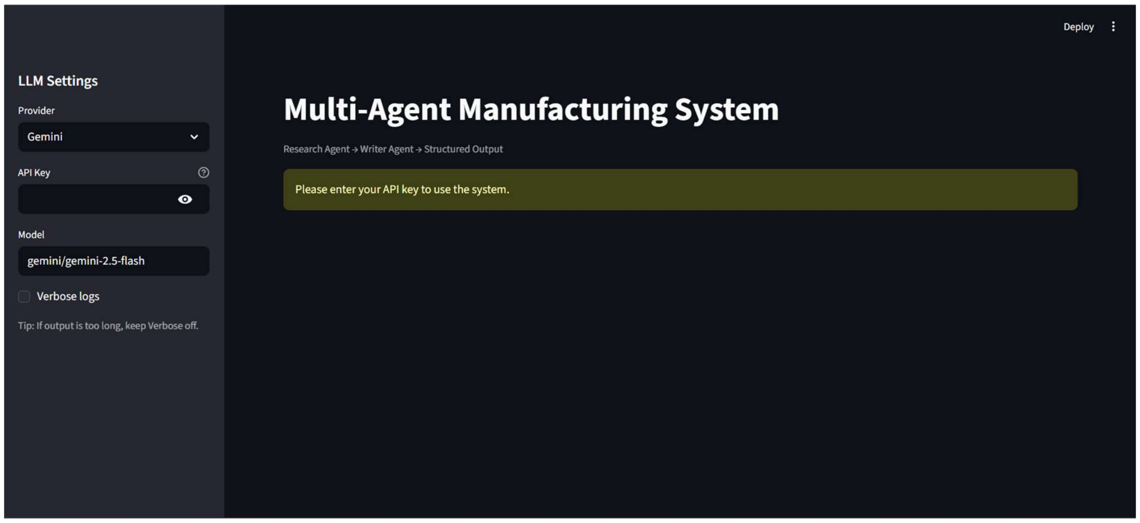
- Maintain separation between runs.
- Preserve intermediate and final outputs.
- Support debugging and auditability.
- Enable artifact download from the UI.

Each run is isolated, ensuring that no data contamination occurs between sessions.
This structured artifact management approach enables:

- Reproducibility
- Debugging transparency
- Workflow audit trails
- Scalable extension toward database-backed storage in future versions

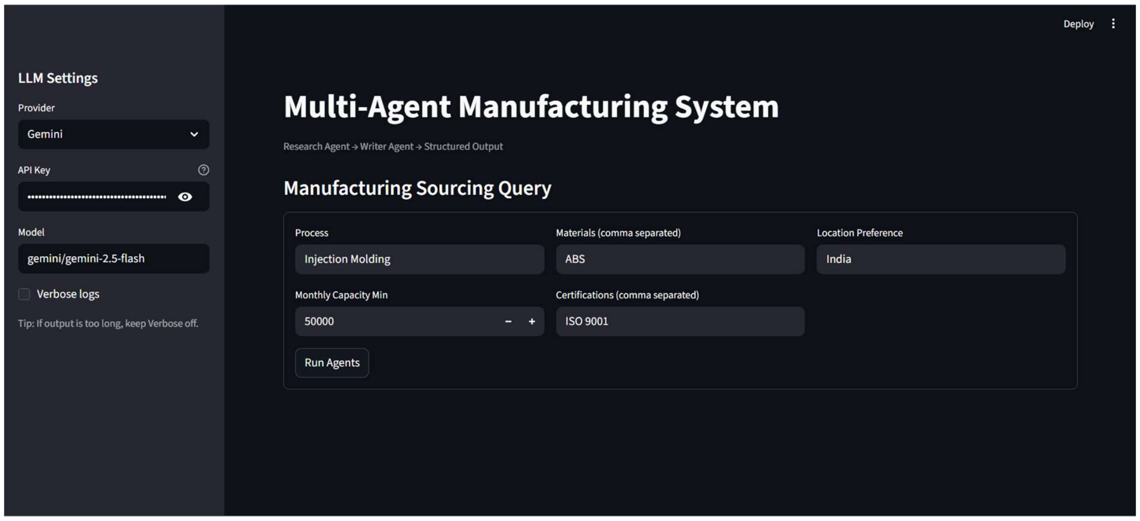
4. Results & Output

4.1 Screenshots / Outputs


Deploy
LLM Settings
Provider
Gemini
API Key
Model
gemini/gemini-2.5-flash
Verbose logs
Tip: If output is too long, keep Verbose off.
Multi-Agent Manufacturing System
Research Agent → Writer Agent → Structured Output

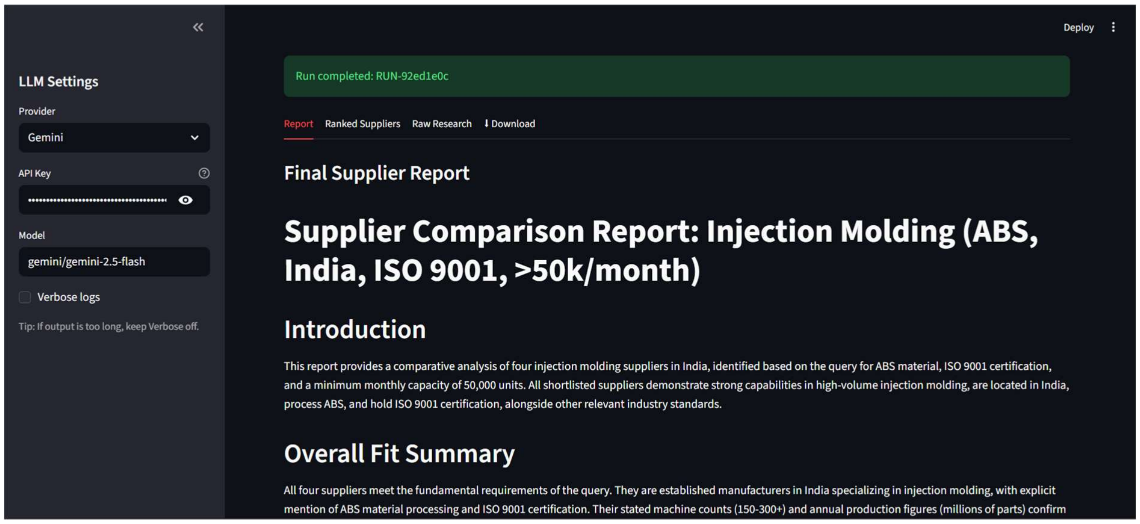
Please enter your API key to use the system.

Figure 3: Initial UI state prompting for LLM API key before accessing the sourcing workflow.


Deploy
LLM Settings
Provider
Gemini
API Key
Model
gemini/gemini-2.5-flash
Verbose logs
Tip: If output is too long, keep Verbose off.
Multi-Agent Manufacturing System
Research Agent → Writer Agent → Structured Output
Manufacturing Sourcing Query
Run Agents

Process: Injection Molding	Materials (comma separated): ABS	Location Preference: India
Monthly Capacity Min: 50000	Certifications (comma separated): ISO 9001	

Figure 4: Structured Manufacturing Sourcing Query form with configurable parameters.



LLM Settings

Provider: Gemini

API Key: ••••••••••••••••••••••••••••••••••••

Model: gemini/gemini-2.5-flash

Verbose logs

Tip: If output is too long, keep Verbose off.

Run completed: RUN-92ed1e0c

Final Supplier Report

Supplier Comparison Report: Injection Molding (ABS, India, ISO 9001, >50k/month)

Introduction

This report provides a comparative analysis of four injection molding suppliers in India, identified based on the query for ABS material, ISO 9001 certification, and a minimum monthly capacity of 50,000 units. All shortlisted suppliers demonstrate strong capabilities in high-volume injection molding, are located in India, process ABS, and hold ISO 9001 certification, alongside other relevant industry standards.

Overall Fit Summary

All four suppliers meet the fundamental requirements of the query. They are established manufacturers in India specializing in injection molding, with explicit mention of ABS material processing and ISO 9001 certification. Their stated machine counts (150-300+) and annual production figures (millions of parts) confirm

Figure 5: Final Supplier Comparison Report with tab-based output navigation.

The system generates a structured output dashboard that includes:

- Run completion banner with unique Run ID.
- Supplier Comparison Report.
- Query Requirements summary.
- Ranked supplier listing.
- Raw research JSON view.
- Downloadable artifacts.

The tab-based layout ensures clarity and separation between processed data and raw research output.

4.2 Reports / Dashboards / Models

Final Supplier Comparison Report Includes:

- Executive Summary
- Supplier Comparison Table
- Supplier Ranking Explanation
- Risk & Gap Analysis
- Recommendations

Generated Artifacts:

- raw_suppliers.json
- structured_suppliers.json
- report.md
- state.json

These artifacts provide traceability and auditability of sourcing decisions.

4.3 Key Outcomes

- Successfully designed and implemented a collaborative multi-agent architecture with clear role separation between Researcher and Writer agents, ensuring modularity and scalability.
- Demonstrated structured hand-offs between specialized AI agents using JSON-based artifacts, reducing ambiguity and enforcing controlled data exchange.
- Achieved reliable schema-validated outputs through structured validation mechanisms, minimizing inconsistencies and reducing the risk of malformed or hallucinated data.
- Built a functional sourcing automation prototype capable of transforming structured manufacturing inputs into ranked supplier comparison reports.
- Implemented centralized orchestration with defined workflow stages and unique Run ID tracking, enabling reproducibility and traceability of each execution.
- Developed a tab-based result interface that clearly separates raw research data from processed and ranked outputs, improving interpretability and transparency.
- Demonstrated practical integration of LLM APIs within an enterprise-style workflow, including controlled execution, structured prompting, and session-based API handling.
- Established a foundation for extensible agent-based automation systems applicable to broader manufacturing and procurement use cases.

5. Conclusion

The Multi-Agent Manufacturing System fulfills the project allocation requirements by successfully designing and implementing a collaborative agent architecture featuring specialization and structured hand-off protocols.

The system demonstrates advanced inter-agent communication, workflow orchestration, and state management while generating high-quality, structured supplier comparison reports.

Through this project, significant learning was achieved in:

- Multi-agent architecture design
- Agent orchestration using CrewAI
- Structured output enforcement
- Schema validation strategies
- LLM-based workflow engineering
- Modular system design principles

This project serves as a strong foundation for intelligent manufacturing automation systems.

6. Future Scope & Enhancements

- Add Compliance Verification Agent
- Add RFQ (Request for Quotation) Generator
- Integrate real-time web search tools
- Add supplier scoring customization
- Implement database-backed storage
- Add Docker containerization
- Deploy backend as REST API
- Add caching for repeated queries
- Implement automated rate-limit fallback logic
- Integrate analytics dashboard for supplier insights

7. References

1. CrewAI Documentation. *CrewAI: Multi-Agent Orchestration Framework*. Available at: https://docs.crewai.com
2. LangChain Documentation. *Building LLM-powered Applications*. Available at: https://docs.langchain.com
3. Google AI. *Gemini API Documentation*. Available at: https://ai.google.dev
4. OpenAI. *Best Practices for Prompt Engineering and Structured Outputs*. Available at: https://platform.openai.com/docs
5. Streamlit Documentation. *Streamlit – The Fastest Way to Build Data Apps*. Available at: https://docs.streamlit.io
6. Pydantic Documentation. *Data Validation and Settings Management Using Python Type Hints*. Available at: https://docs.pydantic.dev

7. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley.
8. Sommerville, I. (2015). *Software Engineering (10th Edition).* Pearson Education.
9. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach (4th Edition).* Pearson Education.
10. Fowler, M. (2002). *Patterns of Enterprise Application Architecture.* Addison-Wesley.