



UNIVERSITY OF GHANA

(All rights reserved)

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCE

SEMESTER 1 2025/2026 ACADEMIC YEAR

COURSE SYLLABUS

**Course Code and Title: DCIT 403 – Designing Intelligent Agent**

**Credits: 3**

**LAB MANUAL**

**(Python-Based Practical Laboratory)**

**Primary Project: Disaster Response & Relief Coordination System**

**Programming Language: Python**

**Agent Framework: SPADE (Smart Python Agent Development Environment)**

**Online Laboratory Platform: GitHub Codespaces**

**1. INTRODUCTION**

This laboratory manual accompanies the course Intelligent Agent Systems and provides structured, hands-on experience in the design, modeling, simulation, and implementation of intelligent multi-agent systems using Python.

All laboratory sessions are delivered through an online development and simulation environment based on GitHub Codespaces, ensuring a uniform, browser-accessible, and reproducible setup for all students. This approach eliminates local installation inconsistencies while supporting realistic multi-agent execution.

The laboratories are organized around a single capstone-driven practical project, the Disaster Response and Relief Coordination System, which evolves incrementally throughout the semester. Each laboratory session introduces specific technical competencies that directly contribute to the completion of the final multi-agent system.

Students will apply:

- Agent-oriented design principles
- Belief–Desire–Intention (BDI) style reasoning
- The Prometheus agent-oriented software engineering methodology

Progression is from simple autonomous agents to a coordinated, distributed, and evaluated multi-agent system operating under uncertainty.

## **2. LABORATORY OBJECTIVES**

Upon successful completion of the laboratory component, students will be able to:

1. Implement intelligent software agents using Python and the Smart Python Agent Development Environment (SPADE)
2. Model agent perception, goals, events, and actions within a dynamic environment
3. Design and execute inter-agent communication using the Foundation for Intelligent Physical Agents – Agent Communication Language (FIPA-ACL)
4. Apply the Prometheus methodology to real-world, distributed problem domains
5. Build coordinated multi-agent systems for dynamic and uncertain environments
6. Evaluate agent autonomy, coordination effectiveness, and system robustness

## **3. TOOLS AND SOFTWARE REQUIREMENTS**

Mandatory Tools (Provided via GitHub Codespaces)

- Python 3.9 or higher
- SPADE (Smart Python Agent Development Environment)
- Extensible Messaging and Presence Protocol (XMPP) Server
  - Prosody XMPP Server *or* ejabberd XMPP Server
- Prometheus Design Tool (PDT)

Supporting Tools

- Draw.io or StarUML for Agent Unified Modeling Language (AUML) diagrams
- Git for distributed version control
- Visual Studio Code or PyCharm (accessed through Codespaces)

## **4. PROJECT OVERVIEW**

### **Disaster Response & Relief Coordination System**

#### **Problem Description**

Following a disaster event such as flooding, earthquakes, or fire outbreaks, emergency response operations must be coordinated rapidly under conditions of uncertainty, partial information, and limited resources. Centralized control systems are often unavailable, unreliable, or overloaded during such events.

This project models a **decentralized intelligent multi-agent system** in which autonomous agents collaborate to:

- Detect disaster events
- Assess damage severity
- Allocate rescue and response tasks
- Manage limited relief and logistical resources

The system emphasizes **distributed decision-making, coordination, and resilience**.

### Core Agent Types and Responsibilities

Agent	Responsibility
SensorAgent	Detects disaster events and reports environmental conditions
RescueAgent	Performs rescue operations
LogisticsAgent	Manages supplies and relief items
CoordinatorAgent	Assigns tasks, sets priorities, and coordinates agents

## 5. LAB STRUCTURE

Each laboratory session consists of:

- **Objective**
- **Background**
- **Practical Tasks**
- **Deliverables**
- **Assessment Criteria**

## LAB SESSIONS

### LAB 1: ENVIRONMENT AND AGENT PLATFORM SETUP

#### Objective

To configure the Python agent development environment and deploy a basic agent.

#### Background

The Smart Python Agent Development Environment (SPADE) enables the creation of intelligent agents using asynchronous behaviors and message-based interaction over the Extensible Messaging and Presence Protocol (XMPP).

#### Practical Tasks

1. Launch the provided GitHub Codespaces environment
2. Verify Python and SPADE installation
3. Start the embedded XMPP server
4. Create agent credentials
5. Implement and execute a basic SPADE agent

#### **Deliverables**

- Screenshot of a running agent in GitHub Codespaces
- Python source code
- Environment setup report ( $\frac{1}{2}$  page)

## **LAB 2: PERCEPTION AND ENVIRONMENT MODELING**

#### **Objective**

To implement agent perception of environmental and disaster-related events.

#### **Background**

Agents must sense their environment to guide decision-making and react to changes.

#### **Practical Tasks**

1. Implement a simulated disaster environment
2. Create a SensorAgent that periodically monitors conditions
3. Generate and log disaster events such as damage severity levels

#### **Deliverables**

- SensorAgent code
- Event logs
- Brief explanation of percepts

## **LAB 3: GOALS, EVENTS, AND REACTIVE BEHAVIOR**

#### **Objective**

To model agent goals and event-triggered behavior.

#### **Background**

Agents respond to internal and external events while pursuing goals.

#### **Practical Tasks**

1. Define rescue and response goals

2. Trigger events from sensor reports
3. Implement reactive behavior using Finite State Machine (FSMs)

#### **Deliverables**

- FSM diagram
- Python implementation
- Execution trace

### **LAB 4: AGENT COMMUNICATION USING FIPA-ACL**

#### **Objective**

To enable inter-agent communication.

#### **Background**

Coordination in multi-agent systems depends on standardized message exchange using the Foundation for Intelligent Physical Agents – Agent Communication Language (FIPA-ACL).

#### **Practical Tasks**

1. Implement ACL message exchange between agent
2. Use INFORM and REQUEST performatives
3. Parse incoming messages and trigger agent actions

#### **Deliverables**

- Message logs
- Agent communication code

### **LAB 5: COORDINATION AND TASK DELEGATION**

#### **Objective**

To coordinate rescue and response tasks among agents.

#### **Background**

Task allocation is a core MAS challenge.

#### **Practical Tasks**

1. Implement CoordinatorAgent
2. Assign rescue tasks to RescueAgents
3. Handle task acceptance, rejection, and reassignment

#### **Deliverables**

- Coordination protocol description
- Running multi-agent demo

## LAB 6: PROMETHEUS SYSTEM SPECIFICATION

### Objective

To formally specify the disaster response system using the Prometheus methodology.

### Background

Prometheus provides structured agent-oriented design.

### Practical Tasks

1. Define system-level goals
2. Identify operational scenarios
3. Specify percepts, actions, and data

### Deliverables

- Goal hierarchy diagram
- Scenario descriptions
- Interface specifications

## LAB 7: AGENT ARCHITECTURAL DESIGN

### Objective

To design the agent architecture.

### Practical Tasks

1. Identify agent types
2. Allocate responsibilities
3. Create agent descriptors

### Deliverables

- Architectural diagram
- Agent descriptors

## LAB 8: INTERACTION AND PROTOCOL DESIGN

### Objective

To define formal interaction protocols.

### **Practical Tasks**

1. Convert scenarios into AUML diagrams
2. Define message sequences
3. Validate protocol completeness

### **Deliverables**

- AUML interaction diagrams

## **LAB 9: CAPABILITY-BASED AGENT DESIGN**

### **Objective**

To decompose agents into capabilities.

### **Practical Tasks**

1. Identify agent capabilities
2. Assign plans to capabilities
3. Define triggering events

### **Deliverables**

- Capability diagrams
- Process descriptions

## **LAB 10: BDI-STYLE PLAN IMPLEMENTATION**

### **Objective**

To implement plans and context conditions.

### **Practical Tasks**

1. Implement plans as behaviors
2. Define context conditions
3. Handle plan failure

### **Deliverables**

- Python plan code
- Execution logs

## **LAB 11: SYSTEM INTEGRATION**

### **Objective**

To integrate all agents into a working MAS.

### **Practical Tasks**

1. Deploy all agents
2. Test communication and coordination
3. Debug failures

### **Deliverables**

- Integrated system demo
- Test report

## **LAB 12: SYSTEM EVALUATION**

### **Objective**

To evaluate system performance and robustness.

### **Practical Tasks**

1. Simulate multiple disaster scenarios
2. Measure response time
3. Evaluate coordination effectiveness

### **Deliverables**

- Evaluation report
- Performance metrics

## **LAB 13: DEMONSTRATION AND PRESENTATION**

### **Objective**

To demonstrate the completed system.

### **Practical Tasks**

1. Live system demonstration
2. Explain design decisions
3. Answer evaluation questions

### **Deliverables**

- Final project submission
- Presentation slides

## **6. ASSESSMENT CRITERIA**

<b>Criterion</b>	<b>Weight</b>
Functional Correctness	40%
Agent Design Quality	25%
Coordination & Communication	20%
Documentation	15%

## **7. ACADEMIC INTEGRITY**

All submissions must represent original work. Plagiarism or unauthorized collaboration will be penalized according to university regulations.

## **8. CONCLUSION**

This lab manual equips students with **practical, industry-relevant skills** in intelligent agent systems while maintaining strong theoretical foundations through agent-oriented design methodologies.