Coordination of a Multi-Agent System for Emergency Response

Team 05

January 11, 2025

Overview

- Task 1 focused on environment and agent design
- Task 2 explored coordination mechanisms
- Now, we integrate these into a practical implementation
- Implementation utilizes **CrewAI** and **Ollama**
- In this presentation, we will cover:
  - Agents and their tasks
  - Crews module and data models
  - Database and tools
  - Scripts
  - Testing
  - Integration in `main.py` and data folder

Agents and Tasks

- For each crew, we define agents and tasks using `.yaml` files.
- Each crew is represented as a Python class, with agents and tasks defined as methods within the class.
- Agents are instantiated with configuration settings, and tasks are created by linking them with specific tools and data models.

Example Code: Medical Services Crew

```
hospital_coordinator:
  role: Hospital Coordinator
  goal: >
    ...
  backstory: >
    ...
  allow_delegation: false
  verbose: true
  llm: ollama/llama3.1
  temperature: 0.4
  max_tokens: 800
```

```
rank_hospitals:
  description: >
  ...
  expected_output: >
  ...
  agent: hospital_coordinator
  context:
    - fetch_hospital_information
```

Crews Module

Data Models

Database and Tools

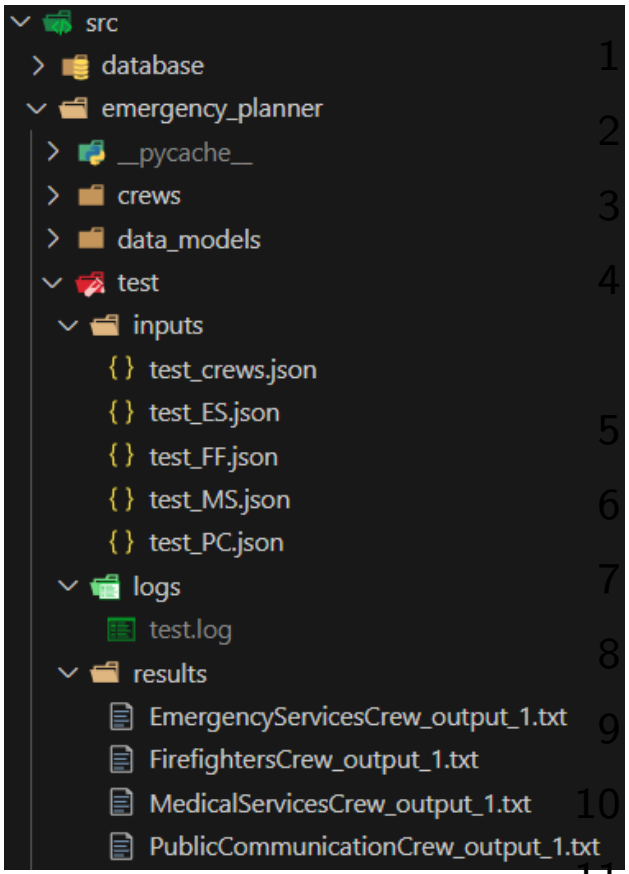
GPS Tools

Scripts

emergency_planner/test.py

```
def process_crew_test(crew_name: str, crew_inputs: Dict[str, Any], test_index:
     int) -> None:
    ...
    crew = instantiate_crew(crew_name)
    logger.info("Agents loaded")
    for agent in crew.crew().agents:
        logger.info(f"Role: {agent.role}")
    ...
    for task in crew.crew().tasks:
        result = task.execute_sync(agent=task.agent, context=task.context, tools=
            task.tools)
    ...
    final_result = crew.crew().kickoff(inputs=crew_inputs)
    ...
def main() -> None:
    ...
    test_cases = load_test_cases(JSON_FILE)
    for i, test_case in enumerate(test_cases, start=1):
        ...
        # Input model validation
        ...
        process_crew_test(crew_name, crew_inputs, i)
    ...
```

emergency_planner/test/



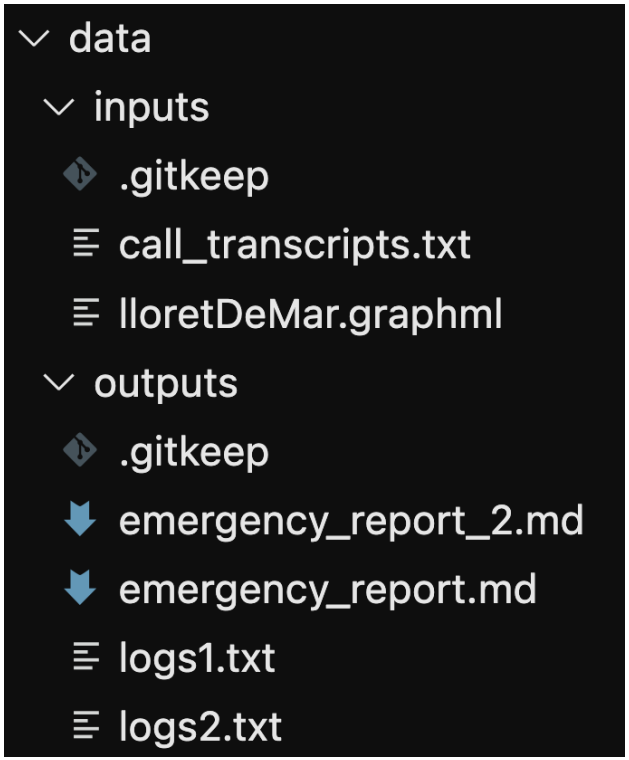
Listing 1: Input Example

```
{
"test_cases": [
    {
        "crew_to_test": "
            EmergencyServicesCrew",
        "crew_inputs": {
            "transcript":
                "A fire of electrical..."
        }
    }
]
}
```

emergency_planner/main.py

```
class EmergencyPlannerFlow(Flow[EmergencyPlannerState]):
    @start()
    def get_call_transcript(self): ...

    @listen(get_call_transcript)
    def emergency_services(self): ...

    @listen(emergency_services)
    def firefighters(self): ...

    @listen(emergency_services)
    def medical_services(self):
        if not self.state.call_assessment.medical_services_required:
            return
        ...

    @listen(or_(and_(firefighters, medical_services), "
         retry_public_communication"))
    def public_communication(self): ...

    @router(public_communication)
    def check_approval(self): ...
```

data/




```
@start()
def get_call_transcript(self):
    with open(EMERGENCY_CALL_TRANSCRIPTS_FILENAME, "r"
        ) as f:
        self.state.call_transcript = f.readlines()[
            TRANSCRIPT_INDEX]
```

```
@listen("save full emergency report")
def save_full_emergency_report(self):
    full_emergency_report = f"""
# Emergency Report

## Call Transcript
{self.state.call_transcript}
...
"""
with open(EMERGENCY_REPORT_FILENAME, "w") as f:
    f.write(full_emergency_report)
```

Thank you!

Questions?