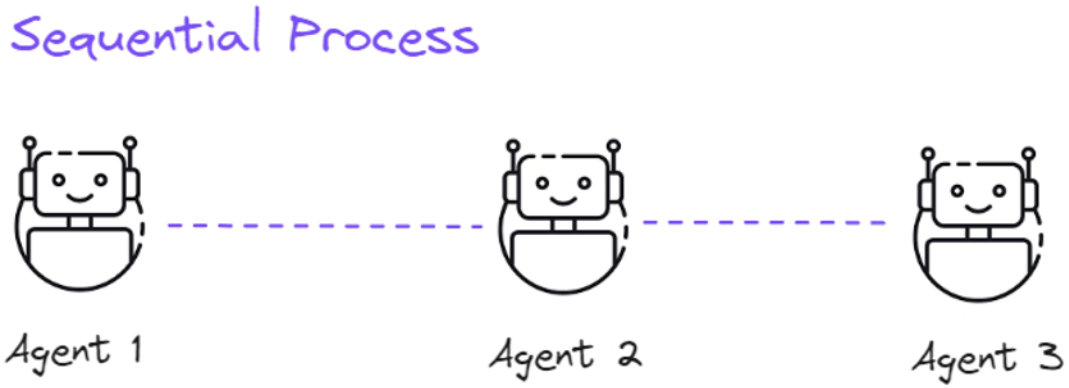
NUFAR GASPAR

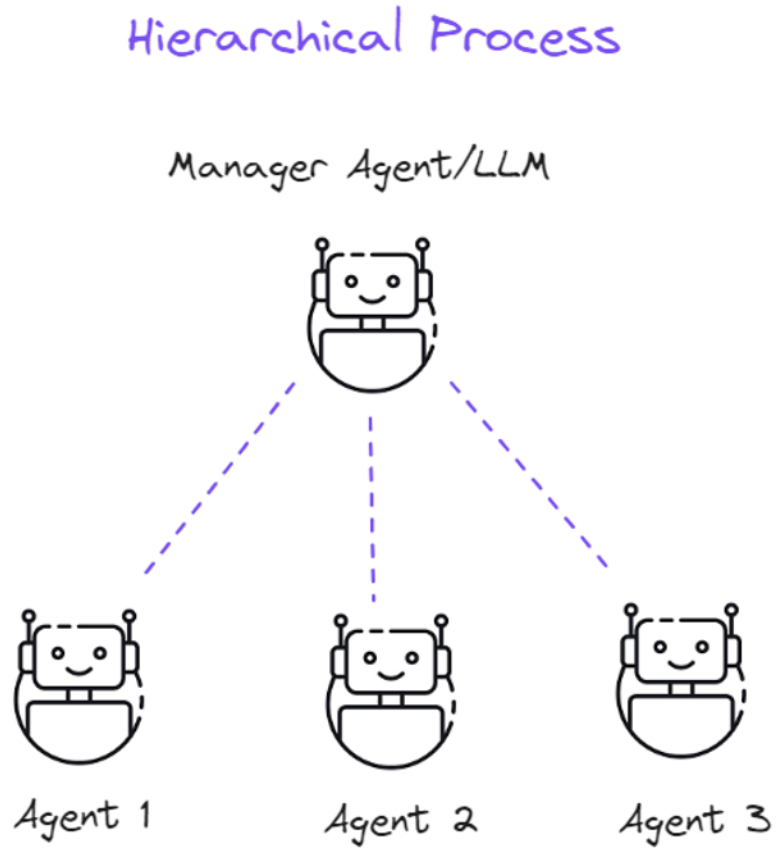
Workshop:
CREW AI Introduction & building a multiagent system

6 steps for building an agent

1. **Clearly define the use case**
   Goals | success metrics | failure tolerance | human escalation
2. **Arrange all relevant data**
   Relevant org data | business process documentation
3. **Define agent actions**
   Access to relevant tools | Agent architecture
4. **Select building methods**
   Build/ buy | Which tools
5. **Build, test, improve**
   Real scenarios | Add guardrails | Refine architecture
6. **Deploy in production**
   Monitor | protect | refine

Multi-agent architectures





Introduction to Crew AI

- Multi-agent platform – agents can interact & collaborate to reach complex goals
- Open-source and paid options
- Python-based, but mostly prompting
- Intuitive and flexible, makes it easy to build
- Each agent has a specialized role & tasks
- Agents can work serially or in parallel; can share info

Crew AI main Building blocks

- Agents
- Tasks
- Crews

```
security_specialist = Agent(
       role="Security Specialist",
       goal="Identify and fix security vulnerabilities in C code",
      backstory="An expert in security vulnerabilities with 15 years of experience in
      analyzing C  code for security issues.",
Additional optional parameters…
```

```
security_review_task = Task(
        description=f"""
            Review the following C code for security vulnerabilities:
            {sample_code}
            Identify any security issues such as buffer overflows, improper input
validation,
            or other security concerns. Be specific about line numbers and exact issues.
            Reference any violations of these guidelines:
            {CODING_GUIDELINES}
        """,
        agent=security_specialist,
        expected_output="A detailed report of security vulnerabilities found in the code"
    )
```

Crew AI main Building blocks

- Agents
- Tasks
- Crews

```
crew = Crew(
        agents=[
            security_specialist,
            memory_specialist,
            style_specialist,
            optimization_specialist,
            technical_writer
        ],
        tasks=[
            security_review_task,
            memory_review_task,
            style_review_task,
            optimization_review_task,
            final_report_task
        ],
        verbose=True,
        process=Process.sequential
    )
```

Additional advanced options

- Memory – help the agents get more reliable results
  - Short-term – during task execution, shared across agents
  - Entity memory – during execution stores the subjects the agents work on
  - Long-term – local DB – stored after execution; leads to self-improvement
- Tools – what the agents can operate – choose wisely, use or create
  - Web search; call an API; Connect to DB; Send notifications;…
- Collaboration – ability for agents to talk, delegate and collaborate
- Guardrails – multiple framework implemented, you can tailor to your needs

List of optional Add-ons/ decisions

- Memory – help the agents get more reliable results
  - Short-term – during task execution, shared across agents
  - Entity memory – during execution stores the subjects the agents work on
  - Long-term – local DB – stored after execution; leads to self-improvement
- Tools – what the agents can operate – choose wisely, use or create
  - Web search; call an API; Connect to DB; Send notifications;…
- Collaboration – ability for agents to talk, delegate and collaborate
- Guardrails – multiple framework implemented, you can tailor to your needs

Additional optional customizations

Tools (Optional)	Represents the capabilities or methods the agent uses for tasks, from simple functions to complex integrations.
Cache (Optional)	Determines if the agent should use a cache for tool usage.
Max RPM	Sets the maximum requests per minute (max_rpm). Can be set to None for unlimited requests to external services.
Verbose (Optional)	Enables detailed logging for debugging and optimization, providing insights into execution processes.
Allow Delegation (Optional)	Controls task delegation to other agents, default is False.
Max Iter (Optional)	Limits the maximum number of iterations (max_iter) for a task to prevent infinite loops, with a default of 25.
Max Execution Time (Optional)	Sets the maximum time allowed for an agent to complete a task.
System Template (Optional)	Defines the system format for the agent.
Prompt Template (Optional)	Defines the prompt format for the agent.
Response Template (Optional)	Defines the response format for the agent.
Use System Prompt (Optional)	Controls whether the agent will use a system prompt during task execution.

NUFAR GASPAR | ng@ngai.ai | +972-54-7884031 |

Customizing agents and tools

```
import os
from crewai import Agent
from crewai_tools import SerperDevTool

# Set API keys for tool initialization
os.environ["OPENAI_API_KEY"] = "Your Key"
os.environ["SERPER_API_KEY"] = "Your Key"

# Initialize a search tool
search_tool = SerperDevTool()

# Initialize the agent with advanced options
agent = Agent(
            role='Research Analyst',
           goal='Provide up-to-date market analysis',
           backstory='An expert analyst with a keen eye for market trends.',
            tools=[search_tool],
           memory=True, # Enable memory
           verbose=True,
           max_rpm=None, # No limit on requests per minute
           max_iter=25, # Default value for maximum iterations
 )
```

Delegation and Autonomy

- Allow_delegation default = false – not allowing agents to seek assistance or delegate tasks

```
agent = Agent(
        role='Content Writer’,
        goal='Write engaging content on market trends’,
        backstory='A seasoned writer with expertise in market analysis.’,
        allow_delegation=True # Enabling delegation
)
```

CREW AI – Additional Materials

- Workshop examples
- Crew AI homepage
- Crew AI example repository
- Crew AI free course
- Good quick start github repo
- Blog: how to build an agent to automate code review

NUFAR GASPAR | ng@ngai.ai | +972-54-7884031 |