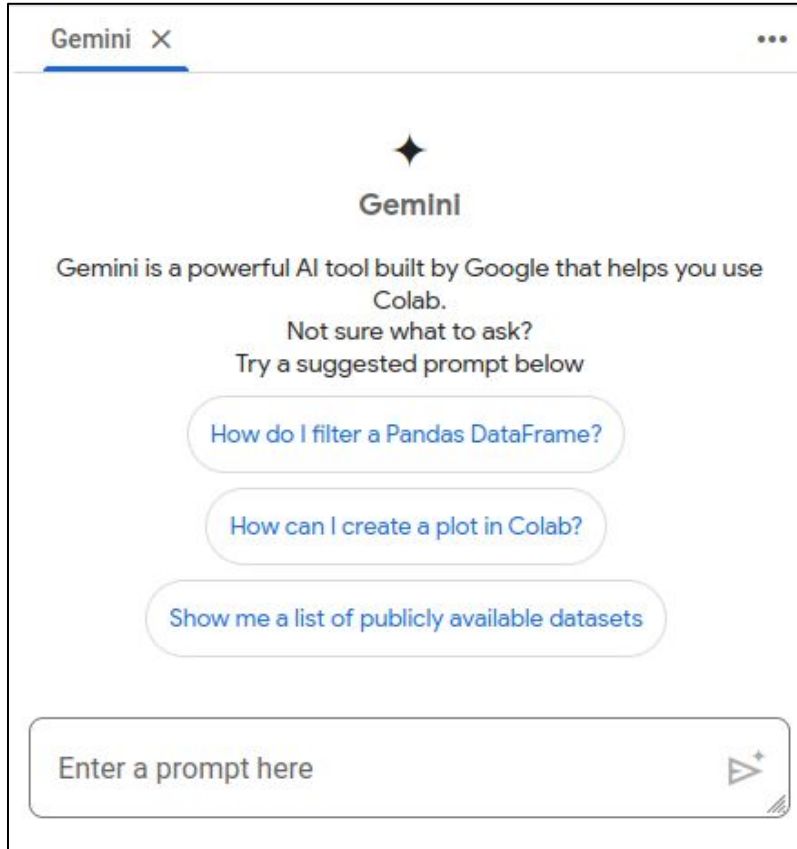


Software project: LLM-Based Agents

Mareike Hartmann
Saarland University, Saarbrücken, Germany

`mareikeh@coli.uni-saarland.de`

Conversational Agents



How can I calculate class centroids for a classification task?

To calculate class centroids for a classification task, you'll first need to have your data in a suitable format, such as a NumPy array or Pandas DataFrame ...

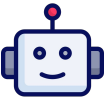
```
import numpy as np

def calculate_centroids(X, y):
    """Calculates class centroids.
    ...
```

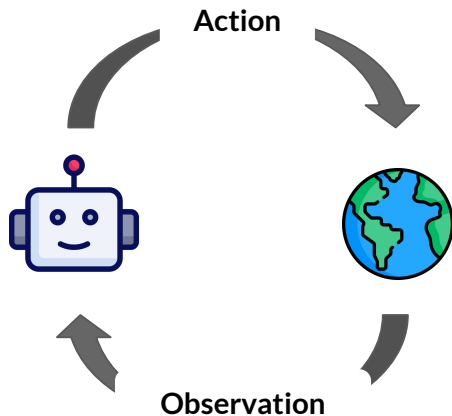


Okay, and how can I use these centroids to classify unseen instances?

Once you have calculated the class centroids, you can classify new instances by finding the centroid closest to the instance...



Interaction With External Environments



(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.

Act 1: `Search[Apple Remote]`

Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the `Front Row media center` program ...

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search `Front Row` next and find what other device can control it.

Act 2: `Search[Front Row]`

Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', '`Front Row (software)`', ...]

Thought 3: Front Row is not found. I need to search `Front Row (software)` .

Act 3: `Search[Front Row (software)]`

Obs 3: Front Row is a discontinued media center software ...

Thought 4: `Front Row (software)` is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

Act 4: `Finish[keyboard function keys]`



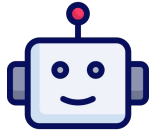
Interaction with a Wikipedia API

[Yao et al. 2023](#)

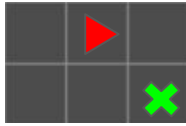
Interaction with External Environments



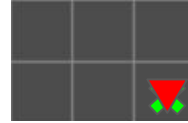
Move to the green cross.



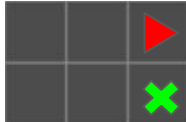
move forward



move forward



move forward



turn right



Digital Assistants: Navigating User Interfaces



Find one-way flights from New York to Toronto.

The screenshot shows a flight booking interface with the following elements:

- Navigation tabs: Book, Flight status, Check-in, My trips
- Category tabs: Flight, Packages, Hotel, Car, Cruise
- Options: Roundtrip, One-way, Book with miles, Flexible dates
- From*: ↔
- Dates*:
- Travelers:
- Class:
- Buttons: [Advanced search >](#) (Certificates, multi-city and upgrades), [Changed bag rules and fees for optional services](#), [Find flights](#), [Find your travel credits](#)

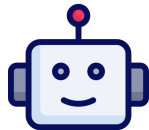


The screenshot shows the same flight booking interface, but with the following changes:

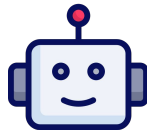
- From*: ↔
- Dates*:
- Class:
- A dropdown menu is open under the To* field, showing "SUGGESTED SEARCH" results:
 - Toronto, ON, CA (YYZ)
 - Toronto, ON, CA (YTZ)
 - Toronto, ON, CA (YBZ)
- Buttons: [Advanced search >](#) (Certificates, multi-city and upgrades), [Changed bag rules and fees for optional services](#), [Find flights](#), [Find your travel credits](#)




```
type ['New York'] [From*]  
type ['Toronto'] [To*]
```



```
select ['Toronto, ON, CA  
(YYZ)'] [To*]
```



Digital Assistants: Orchestrating APIs

 **AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents***

[Trivedi et al. \(2024\)](#)



27 **spotify** : `create_playlist`

24 **venmo** : `create_transaction`

Send money to a user.

Route: POST /venmo/transactions

Parameters:

25 **amazon** : `show_wish_list`

Get list of products in your wishlist.

Route: GET /amazon/wish_list

Parameters:

- **access_token (string):** Access token obtained from amazon app login.
 - Required: true
 - Default: None
 - Constraints: None

Example Task

Here are my all my everyday app accounts.



Play my **Spotify** playlist with enough songs for the entire workout today. My workout plan is in **SimpleNote**.

Let me find Joe's today's workout duration.



```
token = simple_note.login(...)["token"]  
notes = simple_note.search_notes(query="workout", token=...)  
...  
print(note)
```

... Monday: ... 25 minutes ... Tuesday ... 45 minutes ...



Plan is day-wise, let me see what today is.



```
print(datetime.now().strftime("%A"))
```

Tuesday

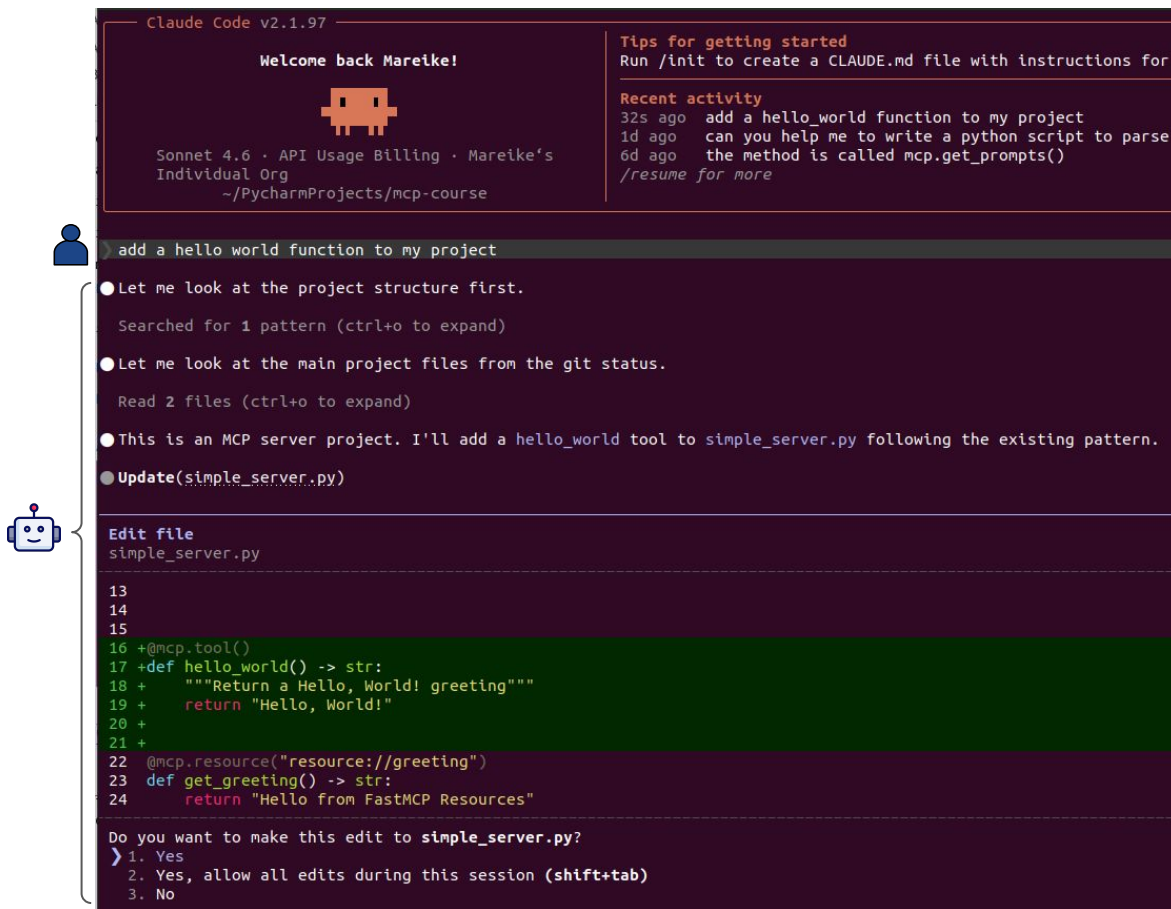


Tuesday, so it's 45 mins, now let me play Joe's playlist with enough songs for this duration




```
workout_mins = 45  
token = spotify.login(...)["token"]  
playlists = spotify.show_playlist_library(token=token)  
for playlist in playlists:  
    duration = 0  
    for id in playlist["song_ids"]:  
        duration += spotify.show_song(song_id=id)["duration"]  
    if total_duration >= workout_mins:  
        spotify.play_music(playlist_id=playlist["id"], token=token)  
        break
```

Coding Assistants



Claude Code v2.1.97

Welcome back Mareike!



Sonnet 4.6 · API Usage Billing · Mareike's Individual Org
~/PycharmProjects/mcp-course

Tips for getting started
Run `/init` to create a CLAUDE.md file with instructions for

Recent activity
32s ago add a hello_world function to my project
1d ago can you help me to write a python script to parse the method is called `mcp.get_prompts()`
/resume for more

add a hello world function to my project

- Let me look at the project structure first.
Searched for 1 pattern (ctrl+o to expand)
- Let me look at the main project files from the git status.
Read 2 files (ctrl+o to expand)
- This is an MCP server project. I'll add a `hello_world` tool to `simple_server.py` following the existing pattern.
- Update(`simple_server.py`)


Edit file
simple_server.py

```
13
14
15
16 +@mcp.tool()
17 +def hello_world() -> str:
18 +     """Return a Hello, World! greeting"""
19 +     return "Hello, World!"
20 +
21 +
22 @mcp.resource("resource://greeting")
23 def get_greeting() -> str:
24     return "Hello from FastMCP Resources"
```

Do you want to make this edit to `simple_server.py`?

- Yes
- Yes, allow all edits during this session (shift+tab)
- No

Personal Assistants



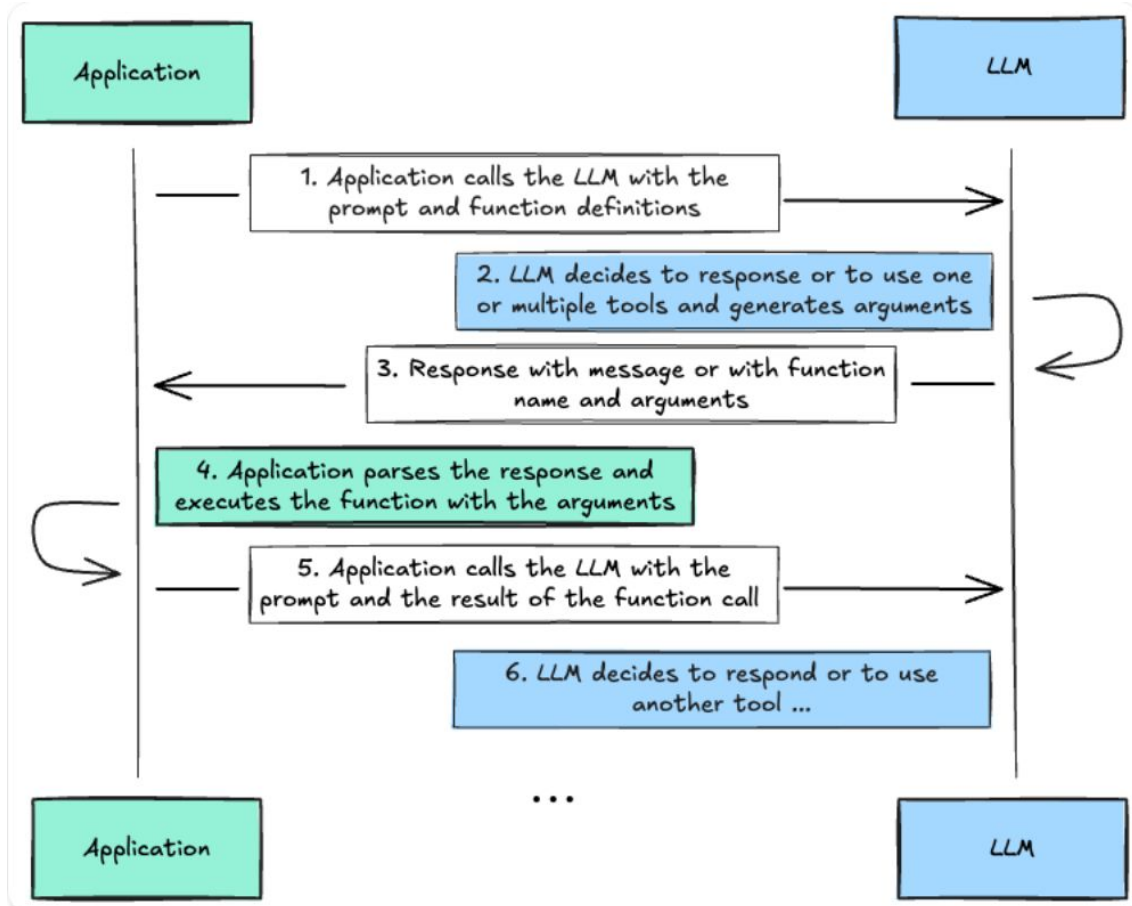
OpenClaw


THE AI THAT ACTUALLY DOES THINGS.



Clears your inbox, sends emails, manages your calendar, checks you in for flights.
All from WhatsApp, Telegram, or any chat app you already use.

<https://openclaw.ai/>



How Can LLMs Operate Tools?



Here are my all my everyday app accounts.  Joe


 



Play my **Spotify** playlist with enough songs for the entire workout today. My workout plan is in **SimpleNote**.

Let me find Joe's today's workout duration.  

```
token = simple_note.login(...)"token"]
notes = simple_note.search_notes(query="workout", token=...)
...
print(note)
```

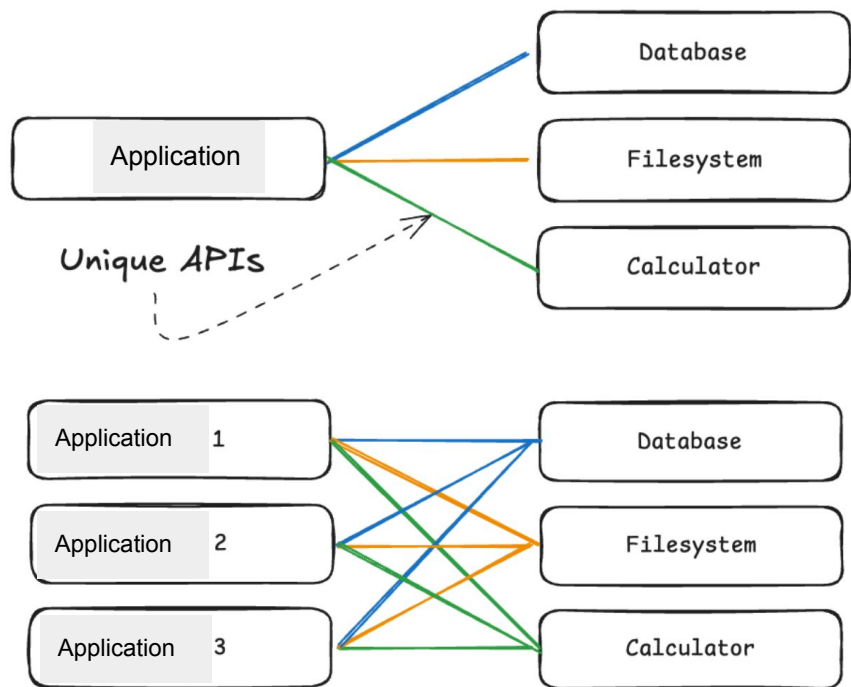
```
exec(
  token = simple_note.login(...)"token"]
  notes = simple_note.search_notes(query="workout", token=...)
  ...
  print(note)
)
```

... Monday: ... 25 minutes ... Tuesday ... 45 minutes ... 

Plan is day-wise, let me see what today is.  

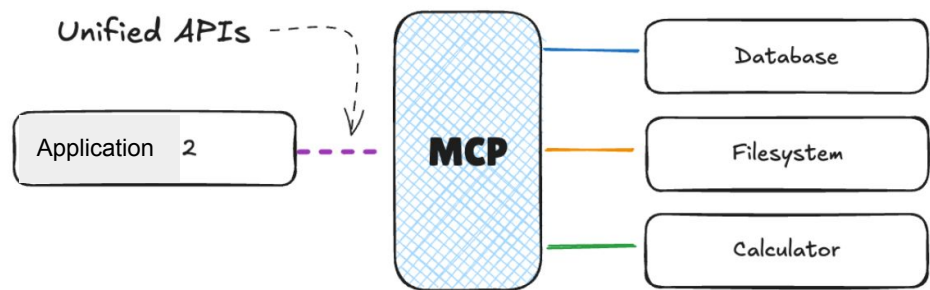
```
print(datetime.now().strftime("%A"))
```

MxN Integration Problem



MCP (Model Context Protocol)

a standardized way for connecting applications and tools/APIs



Architectural Components of MCP

Host:

user-facing AI application that end-users interact with

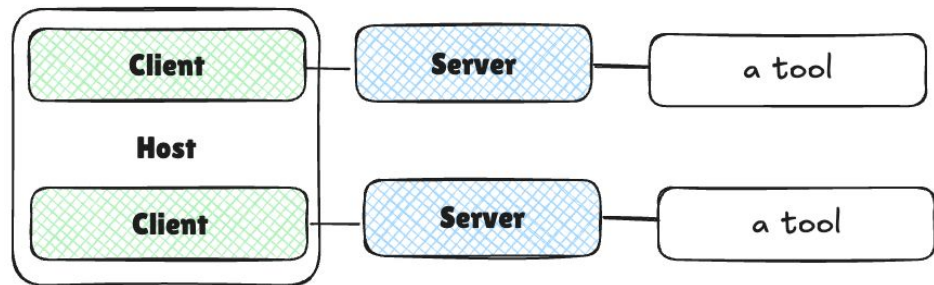
- AI Chat apps like ChatGPT or Claude Desktop
- Custom AI agents built in libraries like [LangChain](#)

Client:

a component within the Host application that manages communication with a specific MCP Server

Server:

an external service that exposes capabilities (tools, data, etc.) to AI models via the MCP protocol



MCP Server

Simple MCP server exposing an addition tool

```
# server.py
from mcp.server.fastmcp import FastMCP

# Create an MCP server
mcp = FastMCP("SimpleServer")

# Add an addition tool
@mcp.tool() new *
def add(a: int, b: int) -> int:
    """Add two numbers"""
    return a + b

@mcp.resource("resource://greeting") new *
def get_greeting() -> str:
    return "Hello from FastMCP Resources"

@mcp.prompt() new *
def math_assistant(problem: str) -> str:
    """Guide the agent to help solve a math problem using available tools."""
    return f"You are a math assistant. Help the user solve the following problem: {problem}. Use the add tool when you need to perform addition."

# Main execution block - this is required to run the server
if __name__ == "__main__":
    mcp.run()
```

Developer Tool: MCP Inspector

an interactive developer tool for testing and debugging MCP servers (see [MCP inspector](#))

The screenshot displays the MCP Inspector interface, which is organized into several sections:

- Navigation Bar:** Located at the top, it includes tabs for Resources, Prompts, Tools (which is currently selected), Tasks, Apps, Ping, Sampling, Elicitations, Roots, Auth, and Metadata.
- Tools Panel (Left):** A sidebar containing a search bar, a "List Tools" button, a "Clear" button, and a list of available tools. The "add" tool is selected, showing its name and description "Add two numbers".
- Tool Configuration (Right):** A detailed view of the "add" tool. It includes:
 - Metadata:** A row of checkboxes for "Read-only", "Destructive", "Idempotent", and "Open-world".
 - Inputs:** Two text input fields labeled "a" and "b", with values "3" and "5" respectively.
 - Metadata Management:** A section for "Tool-specific Metadata" with an "Add Pair" button and the text "No metadata pairs".
 - Output Schema:** A section with an "Expand" button and a code editor showing a JSON schema:

```
{  type: "object"  properties: {    result: {      type: "number"    }  }}
```
 - Actions:** Two buttons: "Run Tool" and "Copy Input".
 - Results:** A "Tool Result: Success" message and a "Structured Content" section containing a JSON object:

```
{  result: 8}
```
 - Validation:** A green bar at the bottom of the structured content area with the text "Valid according to output schema".
 - Unstructured Content:** A section at the very bottom, currently empty.

Connecting a Host to a Server

```
from langchain_mcp_adapters.client import MultiServerMCPClient
from deepagents import create_deep_agent

async def main():
    client = MultiServerMCPClient(
        {
            "simple_server": {
                "transport": "stdio", # Local subprocess communication
                "command": "python",
                "args": ["/home/mareike/PycharmProjects/mcp-course/simple_server.py"]
            },
            "mensa_server": {
                "transport": "stdio", # Local subprocess communication
                "command": "python",
                "args": ["/home/mareike/PycharmProjects/mcp-course/parse_mensaar.py"]
            }
        }
    )
```

```
tools = await client.get_tools()

agent = create_deep_agent(
    model="gpt-4o-mini",
    tools=tools,
    backend=FileSystemBackend(
        root_dir=workdir, # Absolute path to accessible directory
        virtual_mode=True # Recommended: Normalize/sanitize paths
    ),
    checkpointer=MemorySaver(), # Required for multi-turn / persistence
    interrupt_on={ # Optional: Human approval for file ops
        "read_file": True,
        "write_file": True,
        "edit_file": True,
    }
)
```

Example Project Structure

- think of an **interesting, complex** use case to be addressed by an agent
- implement a baseline agent/server framework for solving the task
- collect data to evaluate your agent
 - Many possible approaches for [evaluation](#)
- improve your agent/server framework

Schedule

Week	Date			To Dos
16	13.04.		Kick-off	Check out tutorials, come up with an initial idea
17	20.04.		Present ideas	Meet with group mates, re-fine topic. Meet with me to discuss the re-fined topic
18	27.04.		-	
19	04.05.		Present re-fined group project, discussions	Write-up project proposal. Hand in: 11.5.
20	11.05.		-	
21	18.05.		Progress presentations	
22	25.05.	Holiday		
23	01.06.		Progress presentations	
24	08.06.		Progress presentations	
25	15.06.		Progress presentations	
26	22.06.		Progress presentations	
27	29.06.		Progress presentations	
28	06.07.		Progress presentations	
29	13.07.		Progress presentations	
30...	20.7. ...		-	
35	31.08.	Hand in	-	

Schedule

- Monday slot for progress presentations with whole course
- Additional weekly 30 min meeting slot per group
 - either Tuesday or Wednesday, between 10:00 and 14:30
 - will be scheduled once the groups are in place

Deliverables: Project Proposal

Project proposal (1-2 pages)

- Description of use case
- Description of what needs to be implemented
- Description of planned evaluation
- Statement about the use of AI for writing the proposal

Deadline: May 11

Deliverables: Final Submission

- Code + instructions on how to run
 - Code has to be executable on a linux machine
 - Code should be hosted on github
 - Including instructions on how to run in a README
- Project documentation: 6-8 pages, [ACL style](#), including
 - How did you implement/configure the agent? What is its functionality?
 - How did you implement the MCP server(s)? What is its functionality?
 - Task description
 - Data collection
 - Evaluation
 - Challenges & Limitations (including on how/why you deviated from the project proposal, limitations that you noticed while working on the project)
 - Possible Future Work
 - Use of AI

Deadline: August 31

Grading

- Project work (60%)
- Progress presentations (10%)
- Final report (30%)

Use of AI

- allowed for support in literature review and coding
- for writing proposal and report:
 - allowed for reformulating paragraphs you drafted
 - not allowed for generating the full report

- all AI outputs must be thoroughly reviewed by you
- you must be able to explain anything in the code and report in your own words
- any use of AI components for completing the project must be specified in a special section of the proposal/report

Resources

MCP

[Official MCP documentation](#) [documentation]

[MCP for beginners \(Microsoft\)](#) [tutorial]

[MCP course \(huggingface\)](#) [tutorial]

[Example MCP servers](#) [example servers]

[Model context protocol servers](#) [example servers]

[Python implementation of MCP](#) [software]

Evaluation

[Survey on Evaluation of LLM-based Agents](#)

[MCP-Universe](#)

[MCP-RADAR](#)

[MCPWorld](#)

LLM-based agents

[LangChain](#) [software]

[Luo et al. 2025](#) [survey]

[Wang et al. 2025](#) [survey]

ToDos

send me an email (mareikeh@coli.uni-saarland.de) until end of day, including:

- name
- study programme
- **why you want/need to take this course**
- **why you are interested in LLM-based agents**
- programming experience
- experience with LLM-based agents in professional or private life (if are you using them and how)

If you are accepted for participation, I will send you an invite to a team on Microsoft Teams

ToDos

- think about an interesting use case for LLM-based agents that you want to address in this software project
- send me a slide introducing your idea on a high level
(deadline: Sunday, April 19)