INF 385T · FALL 2026

TEXAS
The University of Texas at Austin

INF385T: Special Topics in Information Science: Language Models and Agentic Workflows

Week 8: In class Quiz, Interoperability and Model Context Protocol

Dr. Abhijit Mishra

Announcements

- **Group Project 1** – Project Report (2-3 pages)
  - Deadline : Nov 5
- **Group Project 1** – Presentation (Nov 5)
- **Group Project 2** – Proposal submission
  - Deadline: 10/23 (Next week)
- **Industry Keynote:** Nov 19
- **Group Project 2** – Final Presentation (Dec 3)
- **Group Project 2** – Final Report (Dec 10)

Week 7 Recap

- Planning and verification
- Reflection

Recap: Plan first agents

- Plan when dependencies matter
- Use typed steps and dependencies
- Validate before execution

Example: A presentation needs a checkpoint 7 days before its deadline.

Look up Dec 3 first, then calculate Nov 26.

Verification

- Check outcomes against tool evidence
- Run deterministic checks first, semantic checks later
- Use evidence-linked semantic verdicts

Example: Missing deliverables? Revise the brief (reply to the user) only.

Reflection and Recovery

- Turn failures into testable corrections
- Revise failures; preserve completed work
- Enforce budgets and stop rules

Example: The brief omitted deliverables because it used memory.
Copy them from the policy result, then recheck the brief.

Week 8: Interoperability and Model Context Protocol

The tool interoperability problem

Many tools. Different interfaces.



The tool integration issues

- **AI landscape is fragmented:** Multiple AI providers, different authentication methods, data formats
- Diverse tools, context management, unique response handling
- Security Concerns: Data privacy, access control, secure communication
- This makes agent-tool integration complex.

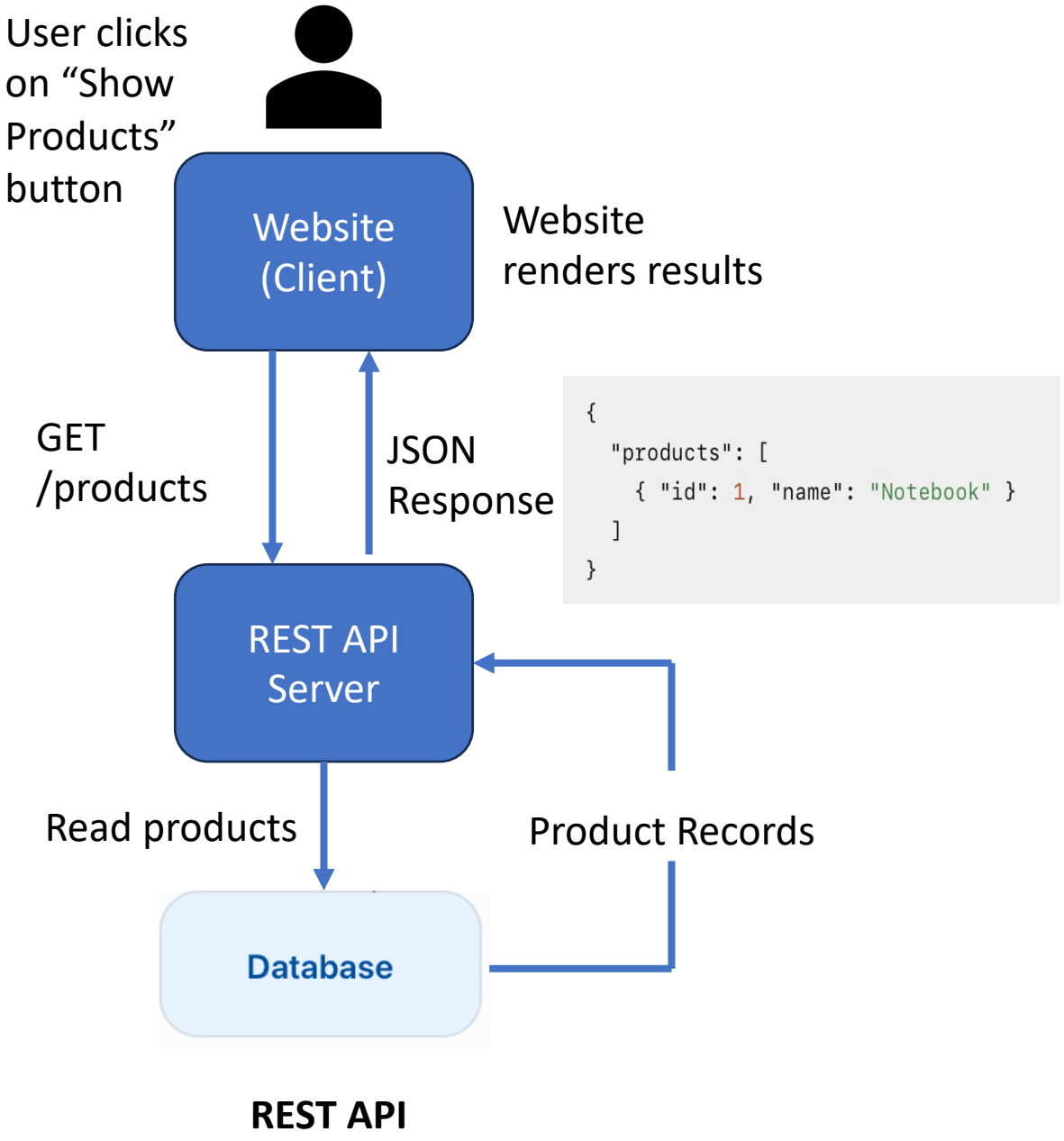


What interoperability means here

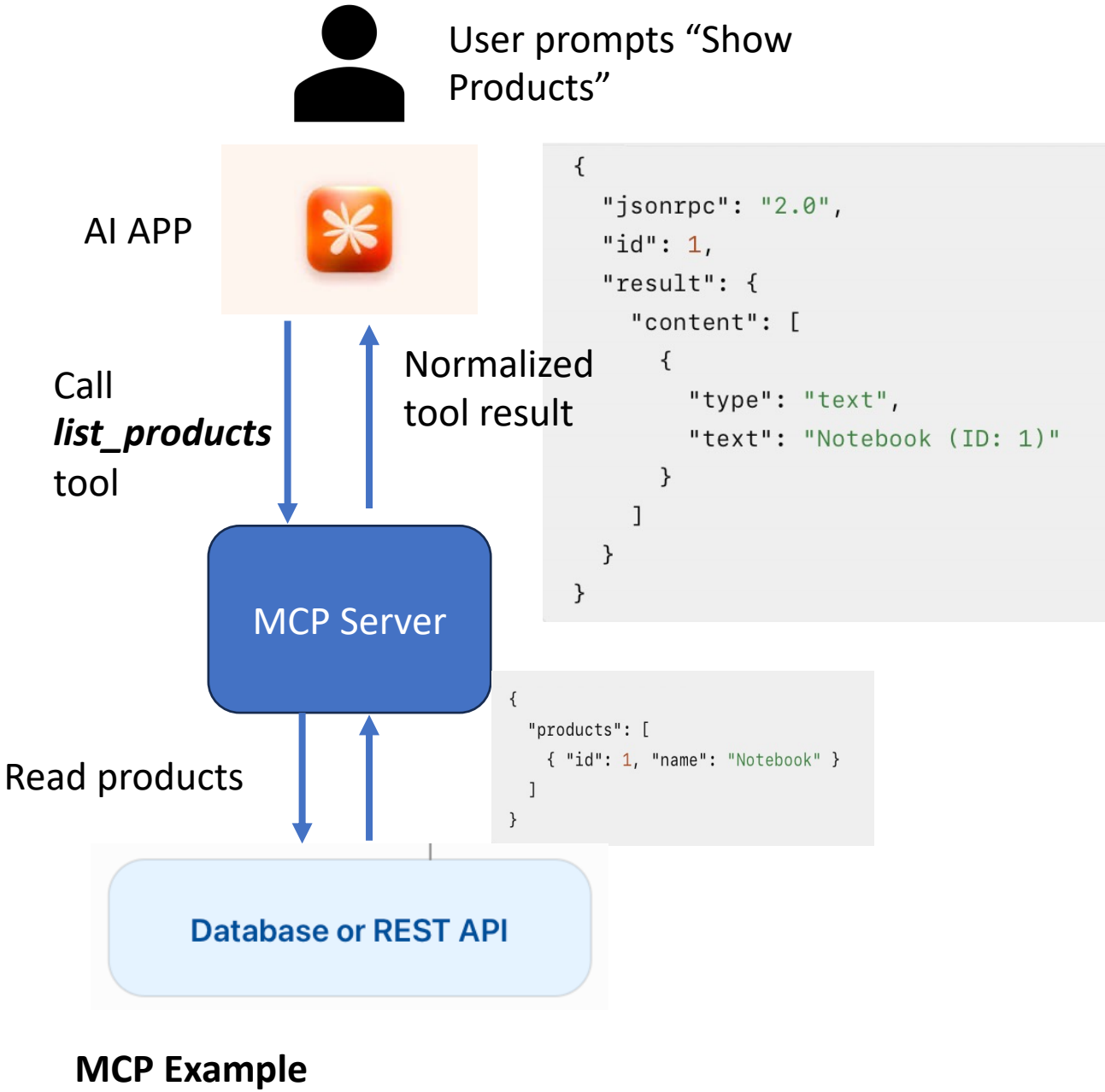
- Shared method and message conventions
- Capability discovery
- Machine-readable schemas
- Standard lifecycle
- Reusable clients, servers, and development tools
- Independent evolution within compatible protocol versions

Model Context Protocol (MCP)

- An Open Source Protocol that enables seamless integration between AI agents with tool and data sources
- **Analogy:** Rest API for web development



REST API



MCP Example

MCP

Open Standard

An open protocol designed for community adoption and extension.

Tool Integration

Securely connect AI with your application's tools and capabilities.

Data Access

Controlled access to application resources and contextual data.

https://www.slideshare.net/slideshow/model-context-protocol-mcp-training-presentation/278317041#12

Growing MCP Support

Official Reference Servers

Examples maintained by the MCP project

modelcontextprotocol/servers

Fetch	Filesystem	Git
Memory	Time	Everything

Community-Built Solutions

Reusable servers for tools and data

Community collections and the MCP Registry

Databases	Browser tools	Search
Messaging	Cloud services	Developer tools

Reference server examples

- **Everything** — Protocol feature demos
- **Fetch** — Retrieve web content
- **Filesystem** — Work with files
- **Git** — Inspect repositories
- **Memory** — Persist a knowledge graph
- **Sequential Thinking** — Structured reasoning
- **Time** — Convert times and time zones

Community ecosystem

- **MCP Registry** — Discover published servers
- **awesome-mcp-servers** — Explore community collections
- **Server SDKs** — Build new integrations
- **Custom MCP servers** — Connect your own systems

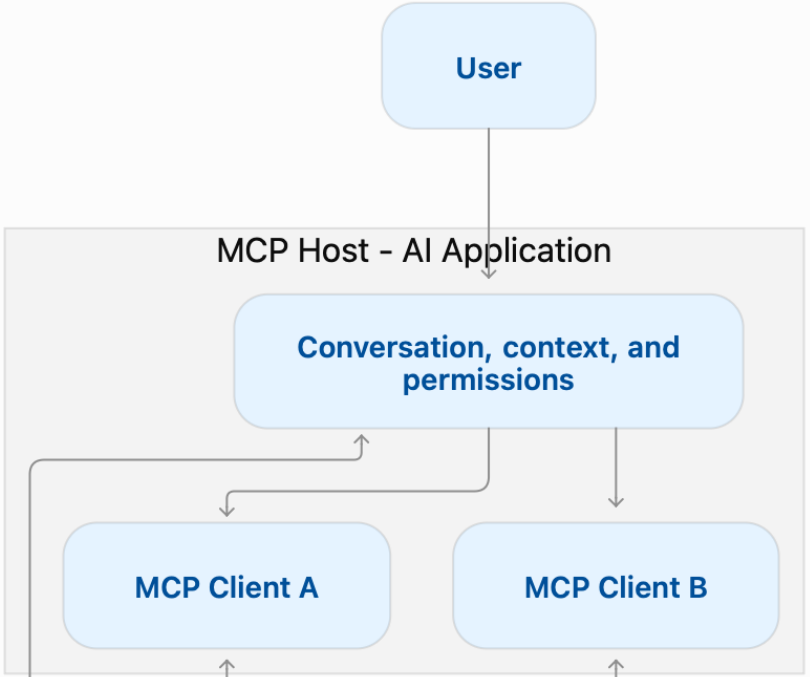
Reference servers demonstrate the protocol. Community servers extend the ecosystem.

Core MCP architecture

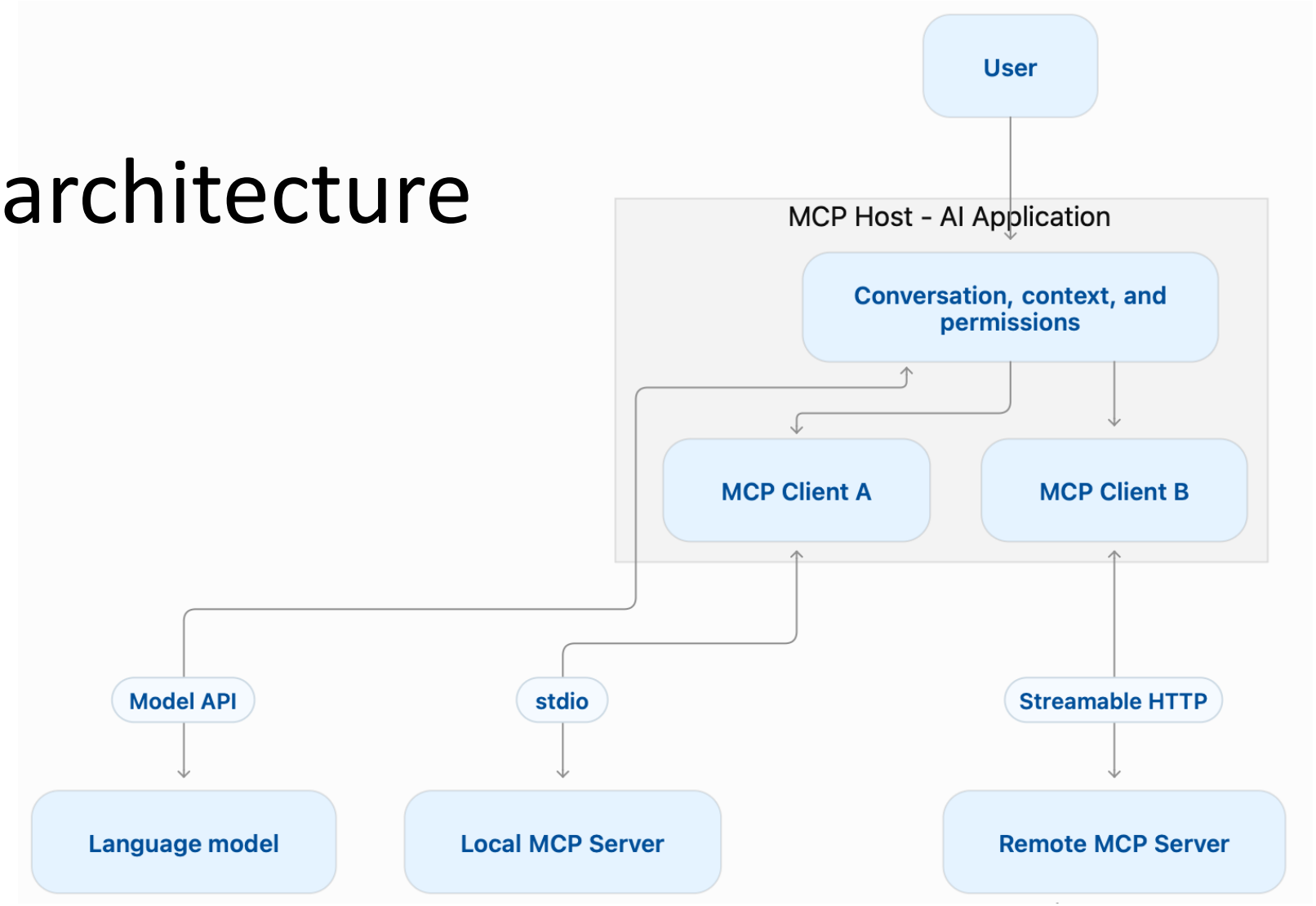
Three things



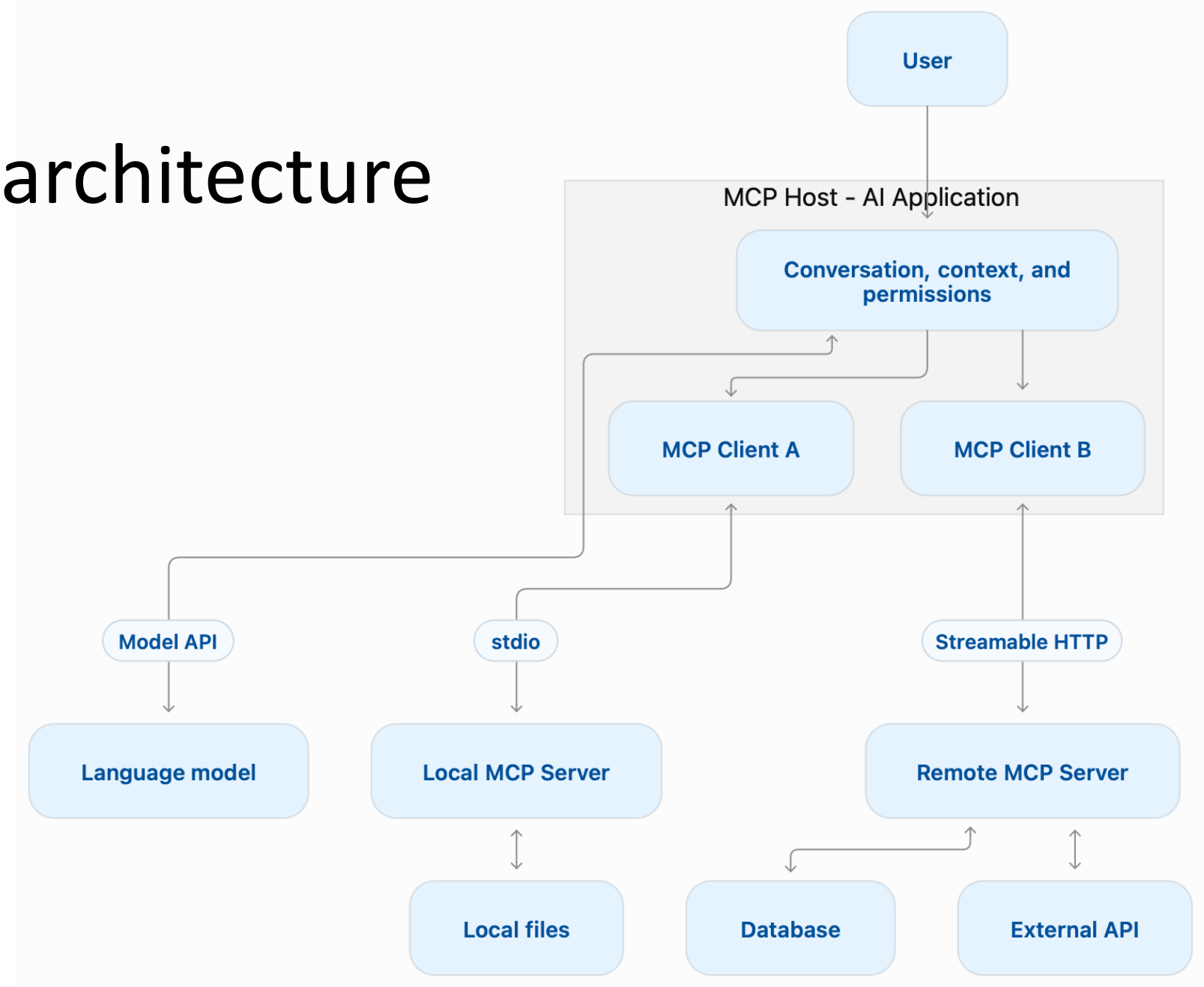
MCP architecture



MCP architecture

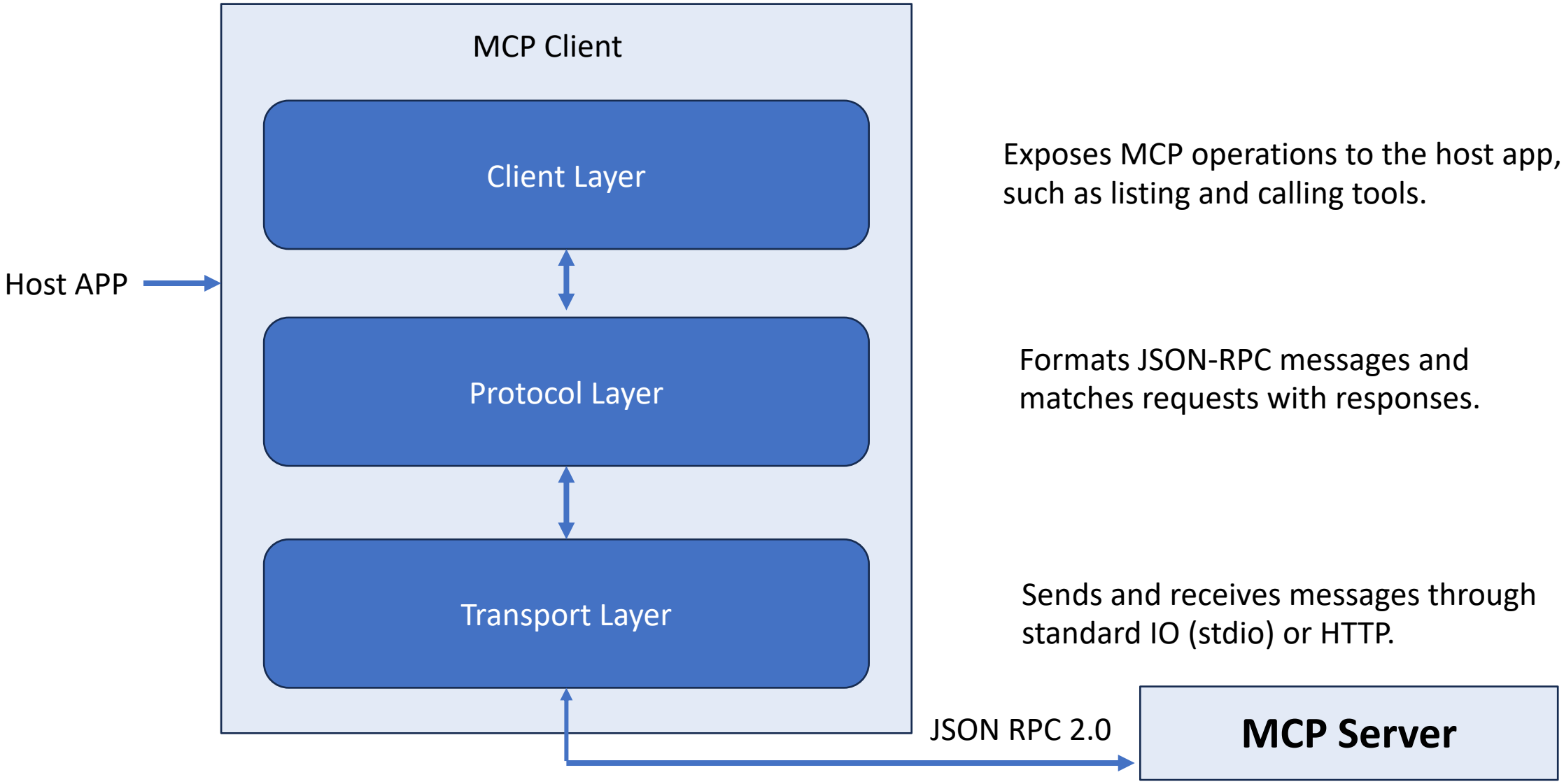


MCP architecture



RPC -> Remote procedure/function calling

Anatomy of MCP Client

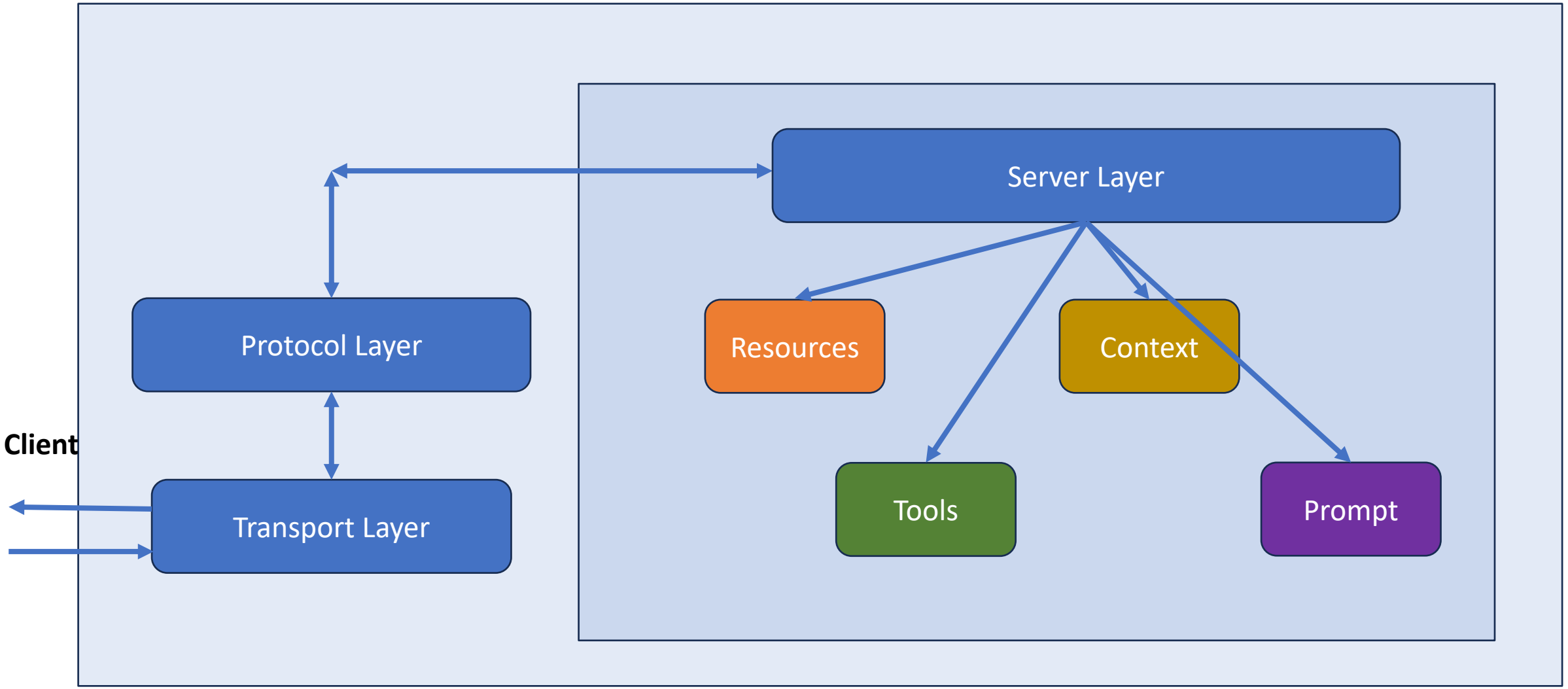


Exposes MCP operations to the host app, such as listing and calling tools.

Formats JSON-RPC messages and matches requests with responses.

Sends and receives messages through standard IO (stdio) or HTTP.

Anatomy of an MCP Server



Building a Custom MCP server

- Requires tool definitions
  - Defining tool schema
  - Defining tool handlers
- Requires prompt definitions
  - Defining Prompt Template
  - Implement a prompt handler

- Why?

MCP separates what a tool or prompt do es from how it is executed or generated

- This allows clients to discover capabilities before using them.

Component	Why is it needed?
Requires Tool Definitions	So MCP clients know which tools are available.
Defining Tool Schema	Specifies the tool's name, purpose, and expected inputs.
Defining Tool Handlers	Implements the actual logic when a tool is called.
Requires Prompt Definitions	So MCP clients can discover and reuse predefined prompts.
Defining Prompt Template	Specifies the prompt's structure and required arguments.
Implementing Prompt Handler	Generates the actual prompt using the supplied arguments.

Defining a tool schema

- Specifies the tool's name, purpose, and required inputs.

```
from mcp.types import Tool

deadline_tool = Tool(
    name="get_deadline",
    description="Look up a course project deadline.",
    inputSchema={
        "type": "object",
        "properties": {
            "project": {
                "type": "string",
                "enum": ["project_1", "project_2"]
            }
        },
        "required": ["project"]
    }
)
```

Define Tool Handler

- Implement the logic that executes the tool

```
DEADLINES = {
    "project_1": "2026-11-05",
    "project_2": "2026-12-03"
}

async def handle_tool(name: str, arguments: dict):
    if name != "get_deadline":
        raise ValueError(f"Unknown tool: {name}")

    project = arguments["project"]

    if project not in DEADLINES:
        raise ValueError(f"Unknown project: {project}")

    return [
        TextContent(
            type="text",
            text=DEADLINES[project]
        )
    ]
```

Define Prompt Schema

- Specifies the prompt's name, purpose, and required arguments
.

```python
from mcp.types import Prompt, PromptArgument

study_prompt = Prompt(
    name="study_plan",
    description="Create a study plan for a course project.",
    arguments=[
        PromptArgument(
            name="project",
            description="Project name",
            required=True
        )
    ]
)
```

Define Prompt Handler

- Generates the prompt using the provided arguments.

```
async def handle_prompt(name: str, arguments: dict):
    if name != "study_plan":
        raise ValueError(f"Unknown prompt: {name}")

    project = arguments["project"]

    return GetPromptResult(
        description="Study planning prompt",
        messages=[
            PromptMessage(
                role="user",
                content=TextContent(
                    type="text",
                    text=(
                        f"Create a three-step study plan for "
                        f"{project}. Check its deadline first."
                    )
                )
            )
        ]
    )
```

Implement methods for listing all tools

- Tool Listing: Shows all tools available on the MCP server. This is needed for discovery

```
async def list_tools():
    return [deadline_tool]

print([tool.name for tool in await list_tools()])
```

Implement methods for listing all prompts

Shows all prompts available on the MCP server.

```
async def list_prompts():
    return [study_prompt]

print([prompt.name for prompt in await list_prompts()])
```

Tool Calling

At runtime

Agent Executes a tool with specific inputs and returns its result.

```
result = await handle_tool(
    "get_deadline",
    {"project": "project_2"}
)

for item in result:
    print(item.text)
```

Prompt Calling

At runtime:

Agent Retrieves a prompt with its arguments filled in.

```
result = await handle_prompt(
    "study_plan",
    {"project": "project_2"}
)

for message in result.messages:
    print(message.content.text)
```

Week 8 Summary

- **MCP Fundamentals:** Learned how MCP standardizes commu nication between AI agents and external tools.
- **Tool Schema & Handlers:** Defined tools and implemented thei r execution logic.
- **Prompt Schema & Handlers**: Created reusable prompts with d ynamic arguments.
- **Tool & Prompt Discovery:** Explored listing available tools and prompts through MCP.

Over to MCP Tutorial with ADK