WiSe 25/26

Project Advanced Media Technologies: „Middleware for GenAI“

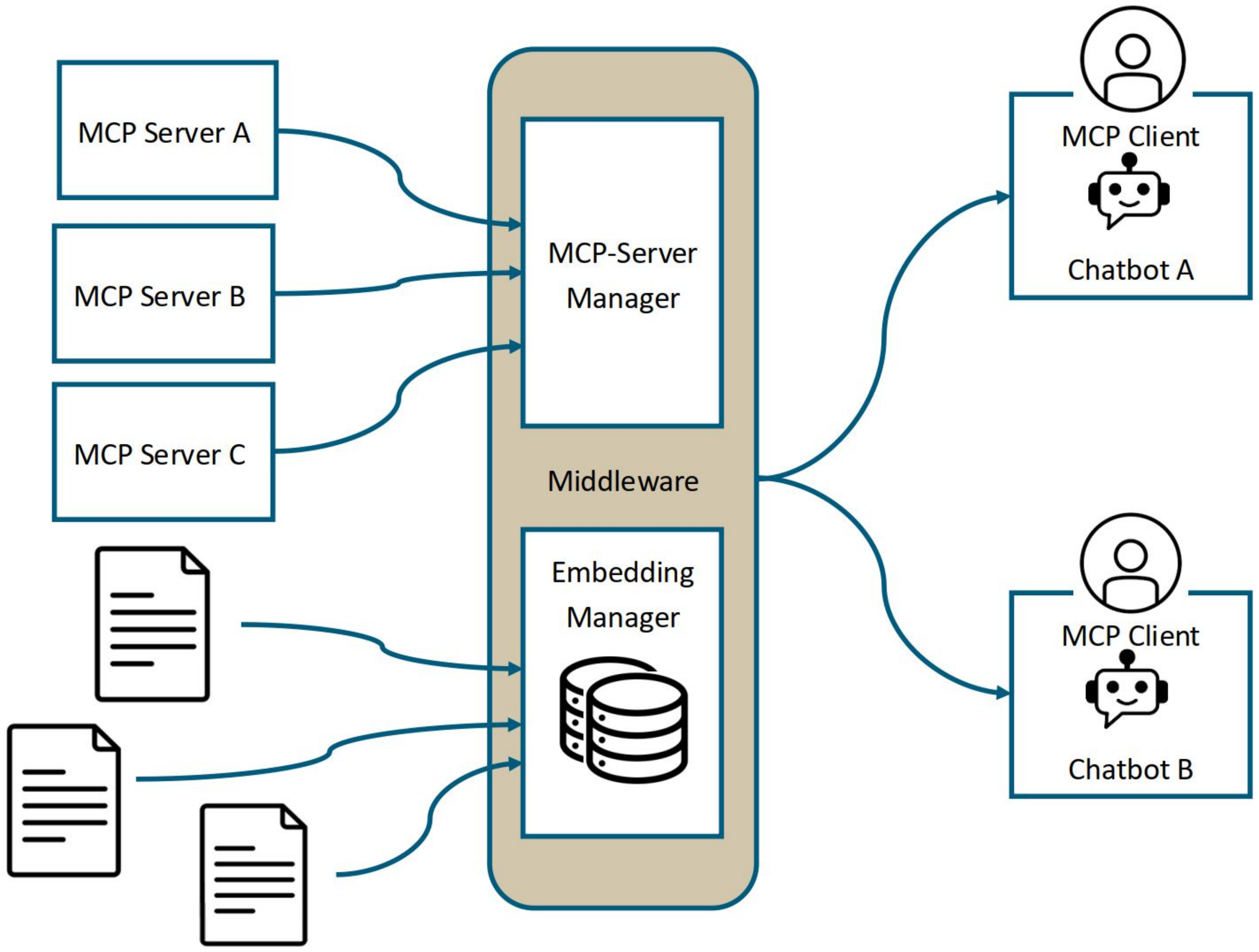
Arno Bock (arnobock_1@campus.tu-berlin.de), Shady Kadry (kadry@campus.tu-berlin.de), George Badour (george.badour@campus.tu-berlin.de)

Problem Statement: State-Of-The-Art GenAI Systems

- **Problems with orchestration of GenAI systems:**
  - Fragmented components
  - Repeated custom integrations (MCP server registration / DB access)
  - <u>Policy control:</u> mostly single-user workflows

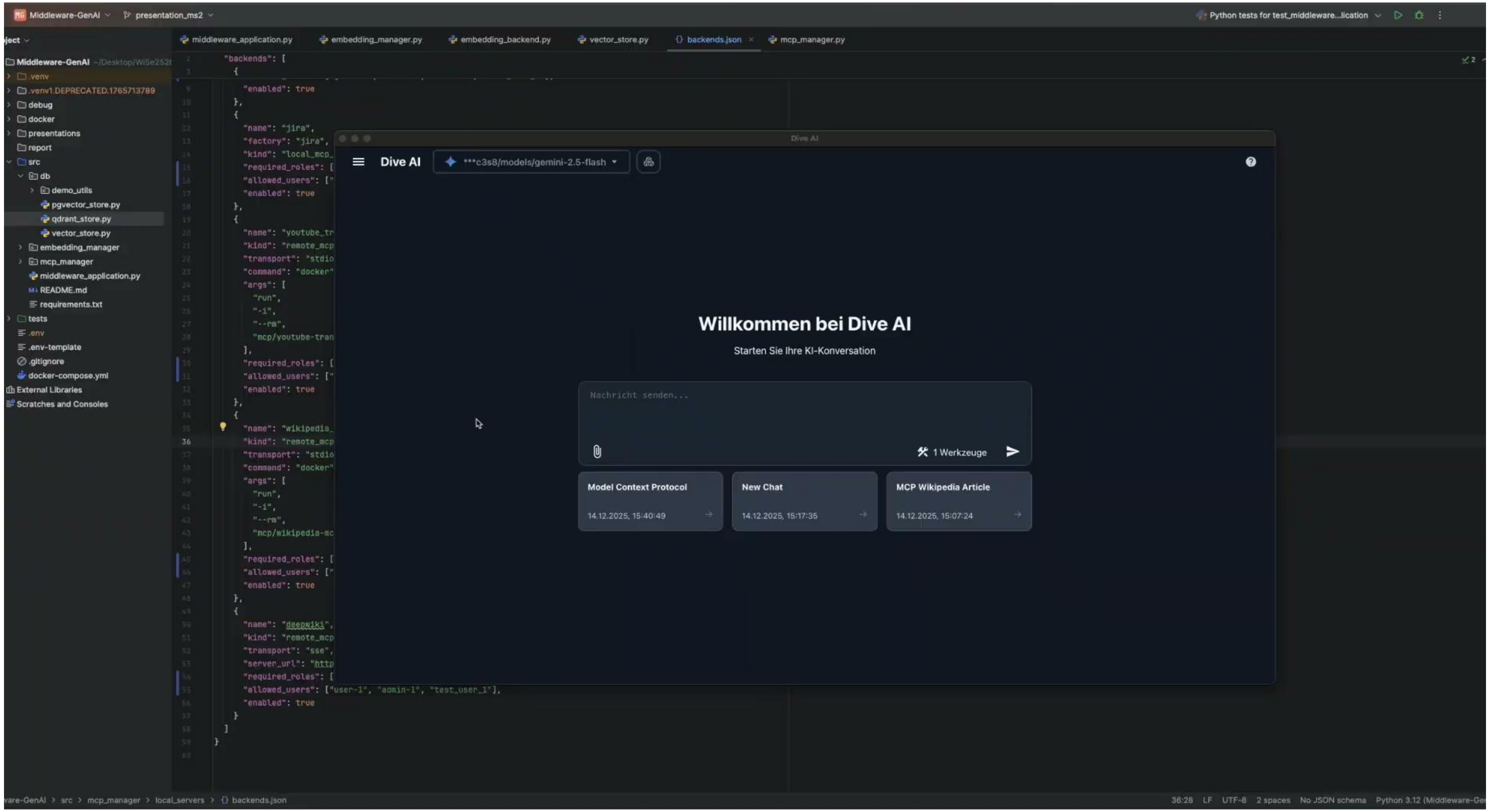
- **Problem statement:** Low reusability of existing GenAI applications <u>due to individual orchestration</u> of:
  - MCP host/server registration
  - Embedding pipelines
  - Database access
  - Policy control mechanisms

Proposed Solution: Middleware Concept



- <u>Eliminates need </u>for **MCP-Clients** to implement:
  - Authorization control
  - MCP host
  - Embedding pipelines
  - Database access/registration

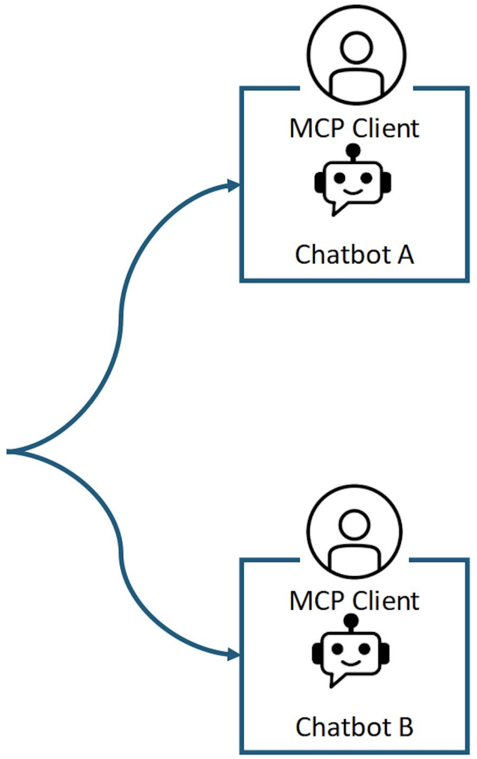
Demo: Current Implementation



```
  "backends": [
    {
      "enabled": true
    },
    {
      "name": "jira",
      "factory": "jira",
      "kind": "local_mcp_
      "required_roles": [
      "allowed_users": ["
      "enabled": true
    },
    {
      "name": "youtube_tr
      "kind": "remote_mcp
      "transport": "stdio
      "command": "docker"
      "args": [
        "run",
        "-i",
        "--rm",
        "mcp/youtube-tran
      ],
      "required_roles": [
      "allowed_users": ["
      "enabled": true
    },
    {
      "name": "wikipedia_
      "kind": "remote_mcp
      "transport": "stdio
      "command": "docker"
      "args": [
        "run",
        "-i",
        "--rm",
        "mcp/wikipedia-mc
      ],
      "required_roles": [
      "allowed_users": ["
      "enabled": true
    },
    {
      "name": "deepwiki",
      "kind": "remote_mcp
      "transport": "sse",
      "server_url": "http
      "required_roles": [
      "allowed_users": ["user-1", "admin-1", "test_user_1"],
      "enabled": true
    }
  ]
}
```

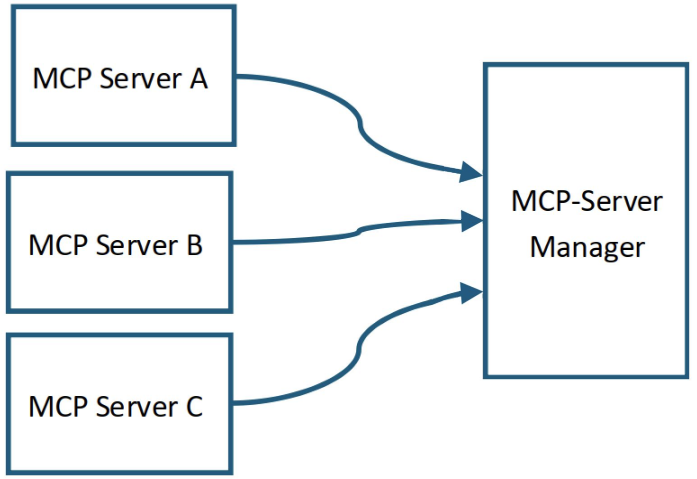
Middleware Concept: Requirements

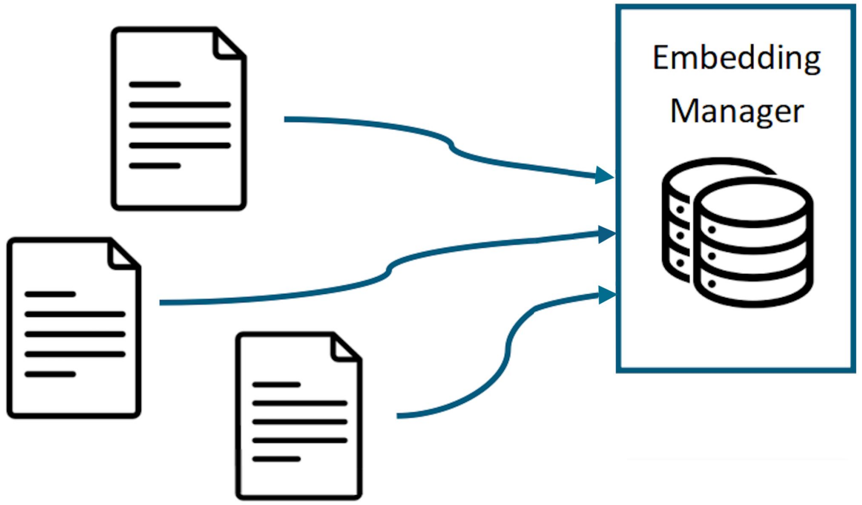
- **MCP-Clients** *(LLM chatbots)*
  - <u>Receive prompts</u> through user interface ✅
  - Pass to middleware and <u>await response</u> ✅
  - Answer based on retrieved information ✅
- **Middleware**
  - Authorize user and parse prompt 🟨
  - Communicate with MCP-Server manager/Embedding manager ✅
  - Return information back to MCP-Client ✅



Middleware Concept: Requirements

- **MCP-Server Manager**
  - Manage internal MCP-Server registry ✅
  - Route prompts to relevant servers *(capability-based)*✅
  - Register new servers through admin user only 🟥
- **Embedding Manager**
  - Embed prompts based on pipeline registry 🟥
  - Upload data through admin user only 🟨
  - Registers/Manages different vector databases 🟥
  - Performs database session control *(user-based)* ✅





Next Steps: Routing

- **Embedding router:** pick pipeline by
  - Modality
  - Corpus
  - Fast vs quality
  - User entitlements

- **Database router:**
  - map requests to backend + tenant/corpus collection; optionally query multiple DBs

- **Industry patterns:**
  - **Embeddings:** rule-based routing first, “smart routing” later
  - **DBs:** per-tenant indexes or federated retrieval with merge/rerank.

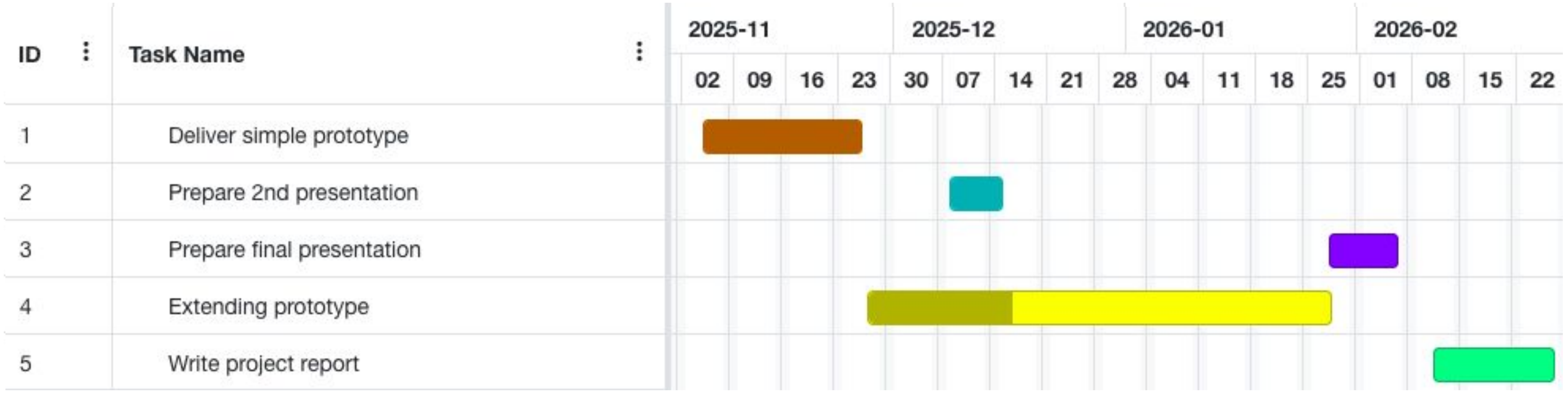
Next Steps: Authentication / UI

- **Auth upgrade:**
  - move from “user-id-in-query” to token-based identity (API key/JWT)

- **UI question:**
  - Dive works for demo
  - But real login likely needs a minimal admin/user UI or gateway

- **Open questions:**
  - Dynamic MCP discovery
  - Tool visibility (no enumeration)
  - Multi-user concurrency

Project Schedule



ID	Task Name
1	Deliver simple prototype
2	Prepare 2nd presentation
3	Prepare final presentation
4	Extending prototype
5	Write project report

Thanks for your attention!

Any questions or feedback?