Session 8 Flow (90–120 min)

1.

Revisit Agents (10 min)

- Quick recap of **Session 6 & 7**:
  - Agents = reasoning + action.
  - Tools = how agents act.
  - MCP = standard "USB port" for tool integration.
- Slide reminder: **ReAct loop** (Plan → Act → Observe → Reflect).

2.

Intro to Agentic RAG (15 min)

- RAG alone = knowledge from docs.
- Agents alone = action with tools.
- **Agentic RAG = knowledge + action in one loop.**
- Show the diagram (the one you liked):
  - Query → Agent → Plan → (RAG retriever OR MCP tool) → Reflect → Answer.
- Emphasize **enterprise fit**: GitLab, Confluence, Jira, internal DBs.

3.

MCP Refresher (10 min)

- Why MCP is critical: security, portability, reusability.
- Tools are just APIs, MCP makes them **plug-and-play**.
- Example: GitLab MCP server with search_code, list_mrs, get_file.

4.

Architecture Walkthrough (15 min)

- Present the **Mermaid diagrams** (System, Sequence, Security).
- Highlight flow:
  - **RAG path** = Chroma + Sentence-Transformers.
  - **MCP path** = Files MCP + GitLab MCP.
  - **LLM-only path** for fallback.
- Show **security guardrails**: allowlists, read-only tokens, logging.

5.

Live Demo (30 min)

Run the upgraded **Agentic RAG Enterprise Project**:

- **RAG example:**

  "Summarize the onboarding policy."

  → See retrieval chunks in UI.

- **GitLab example:**

  "Search for login in project 123 (main)."

  → Trace shows MCP tool call → GitLab API.

- **Files MCP example:**

  "List files under ./rag-service/data."

- **LLM-only example:**

  "What is the capital of France?"

  → Goes direct to LLM fallback.

- Emphasize **timeline trace in UI** (plan, tool call, results).

6.

Wrap-Up (10 min)

- Agentic RAG = **bridge** between knowledge and action.
- Demos showed:
  - Chroma retriever (knowledge).
  - GitLab MCP (enterprise system).
  - Unified orchestrator + UI trace.
- Next step in roadmap: **Orchestration frameworks** (LangGraph, Prefect, etc.).

🔦 Tips to Position Yourself as Expert

- Use enterprise language: “governance,” “allowlists,” “audit logs,” “scalability.”
- Drop references to real-world use: GitLab MR review bots, Merchant onboarding, Knowledge copilots.
- During demo, pause after each query to explain **why the orchestrator routed it that way**.
- End with: *“You’ve now seen how we can plug real enterprise systems into AI copilots in a safe and standardized way.”*

Governance

- **Scenario:** Explaining *why MCP is safer* than ad-hoc tool integrations.
- **Where to say it:** Right after introducing MCP servers.
- **Phrase:**

  “MCP gives us governance — central control over which tools agents can call, with predictable schemas and monitoring hooks.”

✅

Allowlists

- **Scenario:** Presenting the **GitLab MCP server**.
- **Where to say it:** When showing .env with ALLOWED_PROJECTS=123,456.
- **Phrase:**

  “We don’t just let agents loose on the whole GitLab — we enforce project-level allowlists so only approved repos are accessible.”

📜

Audit Logs

- **Scenario:** Describing **enterprise observability & compliance**.
- **Where to say it:** While explaining your **Security & Ops Mermaid diagram**.
- **Phrase:**

“Every MCP tool call — what tool, what args, which user — is logged. These audit logs can feed into SIEM systems, ensuring compliance and traceability.”

📈

Scalability

- **Scenario:** Wrapping up the demo with forward-looking vision.
- **Where to say it:** In the **Wrap-Up / Q&A**.
- **Phrase:**

  “Because the architecture is modular — RAG retrievers, MCP servers, orchestrator — it scales horizontally. We can add Jira MCP, Confluence MCP, or new retrievers without breaking the system.”

✅ By sprinkling these terms in those **specific points of your session**, you’ll sound not only like someone who understands the tech, but also like an **enterprise architect** who’s thought through governance, compliance, and scale.

Session 8 – Presenter Cheat Sheet

1. Revisit Agents

- *"Agents are not just answering, they **plan, act, and reflect**."*
- *"Tools are how agents act — and with MCP, tools become plug-and-play."*

2. Intro to Agentic RAG

- *"RAG = knowledge, Agents = action. **Agentic RAG = both, in one loop**."*
- *"The orchestrator decides: do I retrieve knowledge, call a tool, or just answer?"*

3. MCP Refresher

- *"MCP is the governance layer — like a USB standard for AI tools."*
- *"It enforces **schemas, allowlists, and logs** — so we can trust what tools are being called."*

4. Architecture

- *"Here's the flow: User → Orchestrator → either RAG, MCP, or LLM-only."*
- *"Notice how GitLab MCP is read-only, allowlisted, and logged — that's enterprise compliance built in."*

5. Live Demo (say before each query)

- *"Now let's try a RAG query — retrieving policy docs."*
- *"Here's a GitLab query — search for 'login' in project 123."*
- *"This one is a local file MCP query."*
- *"Finally, a general knowledge query goes to LLM-only."*

(While the UI shows the **timeline**:)

- *"See how the orchestrator shows its plan, the tool call, the raw results, and the final synthesis."*

6. Wrap-Up

- *"We've now seen how **knowledge + action** are combined in a safe, governed way."*
- *"Governance via MCP, allowlists on projects, audit logs on tool calls."*
- *"This architecture is scalable — we can plug in Jira, Confluence, Vault, or any enterprise system the same way."*

Here are **two clear reasons** (keep them sharp for your session):

1

Uncontrolled External Execution

- **Why it's a concern:**

  MCP servers expose tools that agents can call at runtime.

  If not tightly governed, an LLM could trigger **file access, code execution, or API calls** beyond intended scope.

- **How security sees it:**

  *"This opens potential vectors for **data leakage** or **privilege escalation**, unless every tool is strictly sandboxed and allowlisted."*

2

Compliance & Audit Gaps

- **Why it's a concern:**

MCP tooling is still new. Enterprises don't yet have **standardized audit trails, RBAC policies, or SIEM connectors** for MCP traffic.

- **How security sees it:**

  "Without mature logging, monitoring, and approval workflows, MCP tool calls can't meet compliance frameworks like SOC2, ISO27001, PCI-DSS."

👉 You can then add:

*"That's why in today's demo, MCP is scoped to **read-only, allowlisted GitLab projects and local file paths**, with output caps and logging — the very guardrails security would expect before approving wider MCP adoption."*

🔴 Why MCP is restricted today (your org's security stance)

1. **Uncontrolled external execution**
   - MCP lets LLMs trigger tools at runtime.
   - Without strict sandboxing, this risks **data leakage** or **privilege escalation**.
2. **Compliance & audit gaps**
   - MCP is new; lacks mature **audit trails, RBAC, SIEM connectors**.
   - Hard to prove compliance for **SOC2, ISO27001, PCI-DSS**.

🟢 How MCP can become enterprise-safe

1. **Governance & allowlists**
   - Limit tools to **read-only, allowlisted systems** (e.g., GitLab projects).
   - Enforce **schema validation, rate limits, and output caps**.
   - Wrap every MCP call with **audit logging** → SIEM.
2. **Controlled deployment**
   - Run MCP servers **inside the secure VPC/Kubernetes cluster**, behind internal auth.
   - Add **RBAC policies** so only approved agent workflows can call sensitive tools.
   - Integrate with **existing enterprise monitoring** (Splunk, Azure Monitor, etc.).

✅ That way, you position yourself as both:

- **Realistic** (why it's blocked today).
- **Forward-looking** (how it can be adopted safely).

💡 Analogy for MCP Security

"Think of MCP like a USB port for AI. Just as you can plug in any USB device to your laptop — a keyboard, a flash drive, or even something malicious — MCP lets an AI agent plug into any tool. IT doesn't block USBs because they're bad, they block them until there are policies: device control, encryption, monitoring. MCP is in the same stage — powerful, but it needs those guardrails before enterprises allow it."

That makes it **memorable** and shows you understand both **the risk** and **the governance path forward**.