CAPSTONE DAY – OVERVIEW

PURPOSE OF THE CAPSTONE

- Consolidate learning by building a small project.
- Encourage teamwork and creativity.
- Push the **accelerator** closer to production for at least one use case.

FORMAT & SCHEDULE

Suggested 3–4 hour structure:

1. **Intro & recap (20–30 min)**
   - Remind everyone of Days 1–3.
   - Clarify expectations and scoring (if any).
2. **Team formation and project selection (20–30 min)**
   - 2–4 people per team.
   - Pick or adapt a project idea (see project ideas file).
3. **Build time (2–2.5 h)**
   - Implement a small but end-to-end solution.
4. **Demos and feedback (30–60 min)**
   - 5–10 min per team.
   - Explain goal, architecture, demo, lessons learned.

EXPECTATIONS FOR TEAMS

Teams should:

- Use both environments where possible (Ollama + watsonx).
- For a “production-ready” flavour, build on top of:
  - The `accelerator` service (API + tools + UI), and/or
  - RAG & governance notebooks from `labs-src`.

Produce:

- A working notebook or small service.
- A short explanation (slides or markdown) that covers:
  - Problem statement.
  - Architecture (how they used RAG, agents, orchestration).
  - Limitations and next steps.

ASSESSMENT CRITERIA (OPTIONAL)

If you choose to “score” or rank projects, consider:

- **Clarity of problem statement**
  - Is it clear what the assistant or system is trying to solve?
- **Technical implementation**
  - Does it run end-to-end?
  - Does it use RAG and/or agents in a meaningful way?
- **Use of workshop topics**
  - RAG (Day 2).
  - Orchestration / agents (Day 3).
  - Evaluation / governance (optional but encouraged).
- **Presentation & storytelling**
  - Is the demo easy to follow?
  - Do they reflect on what worked / didn’t?

DELIVERABLES

Each team should provide:

- A repo or folder (or branch) with:
  - Code / notebooks.
  - Instructions to run (README).
- A demo plan:
  - 2–3 main use cases to show.
  - Known limitations.

Optional:

- PRs or patches against `accelerator` (if your org uses a shared repo).

LINKING TO PROJECT IDEAS

You can reference the **Capstone Project Ideas** page for inspiration.

Highlight projects that explicitly:

- Extend `retriever.py`, `pipeline.py`, `eval_small.py`, `ui/app.py`.
- Combine the two environments (local vs watsonx).
- Include some governance / evaluation flavour.

Encourage teams to adapt scope to fit the available time.

SUGGESTED FACILITATION TIPS

- Keep teams small (2–4 people) so everyone can contribute.
- Encourage early “thin slice” demos:
  - Simple but working RAG / agent path first.
  - Add polish only if time allows.
- Have a floating **“help desk”** (one or two facilitators) for debugging.

The capstone is about **learning and collaboration**, not perfection.

CAPSTONE PROJECT IDEAS

HOW TO USE THIS LIST

- Pick one idea or merge aspects from several.
- Adapt scope to fit 3–4 hours of work.
- Feel free to leverage code from:
  - Day 1–3 labs.
  - `labs-src` notebooks.
  - `accelerator` Python modules & notebooks.

1. WORKSHOP FAQ ASSISTANT (RAG OVER THE COURSE)

- **Description**
  - Build an assistant that can answer questions about the workshop itself:
    - Agenda, labs, files, concepts.
- **Required pieces**
  - Corpus = workshop docs, READMEs, and lab guides.
  - RAG in both:
    - Ollama env.
    - watsonx env.
  - Optional: deploy via `accelerator` service.
- **Helpful accelerator files**
  - `rag/retriever.py`, `rag/pipeline.py`, `rag/prompt.py`
  - `tools/chunk.py`, `tools/embed_index.py`
  - `service/api.py`, `ui/app.py`
- **Suggested stretch goals**
  - Side-by-side comparison (Ollama vs watsonx).
  - Evaluation with `tools/eval_small.py`.

2. INTERNAL POLICY / HR HELPER

- **Domain**
  - Synthetic HR/policy docs.
- **Build**
  - RAG with safety-aware prompting and refusal patterns.
- **Use accelerator as**
  - A properly configured microservice for HR policy queries.
- **Helpful reference notebooks**
  - RAG examples in `labs-src`.
  - Governance notebooks for policy checks.
- **Stretch goals**
  - Confidence indicator (e.g. high/medium/low).
  - Refusal or handoff for out-of-scope questions.
  - Governance metrics via Evaluation Studio.

3. RAG DEBUGGER & EVALUATOR

- **Focus**
  - Evaluation rather than a new use case.
- **Build**
  - Harness that compares different RAG settings/backends:
    - `k` values.
    - Embedding models.
    - Retriever strategies.
- **Accelerator integration**
  - Implement `tools/eval_small.py` to call `/ask`.
  - Use `notebook:Analyze_Log_and_Feedback.ipynb` to explore logs.
- **Helpful reference notebooks**
  - RAG notebooks in `labs-src`.
  - Governance evaluation notebook.
- **Stretch goals**
  - Simple dashboards (e.g. Streamlit or notebooks).
  - Export to governance as evaluation datasets.

4. TEAM KNOWLEDGE HUB BOT

- **Corpus**
  - Short articles written by team members about their domains.
- **Build**
  - RAG assistant that attributes answers to authors and links to sources.
- **Use accelerator to**
  - Host the service with a tailored `ui/app.py` Streamlit interface.
- **Helpful reference notebooks**
  - `labs-src/use-watsonx-chroma-and-langchain-to-answer-questions-rag.ipynb`
  - Accelerator ingestion notebooks.
- **Stretch goals**
  - User authentication hooks.
  - Session-aware UI and feedback collection.

5. ORCHESTRATED ASSISTANT WITH TOOLS

- **Build**
  - Agent in watsonx env that chooses between tools:
    - Accelerator RAG API.
    - Calculator.
    - Possibly others (ticket creation, CRM, etc.).
- **Helpful files**
  - `service/api.py`, `service/deps.py` (as the RAG microservice).
  - Agent notebook from Day 3.
  - Governance notebooks (`labs-src`).
- **Stretch goals**
  - Logging suitable for governance.
  - Simple “session view” for conversations and tool calls.

CUSTOM PROJECT IDEAS

Encourage participants to propose:

- Domain-specific assistants (e.g., finance, support, internal engineering docs).
- Integrations with existing datasets or APIs.
- Extensions to accelerator:
  - Additional endpoints (e.g. `/ingest`, `/batch-ask`).
  - Batch scoring APIs.
  - Admin UI for corpus management.

TIPS FOR SUCCESS

- Start from existing lab notebooks and accelerator scripts.
- Aim for a clear end-to-end story: **data → RAG → API/UI → evaluation**.
- Limit scope to something demoable within the time.
- Focus on one or two *new* ideas, not everything at once.