Individual Project: Build a Personal Research Portal (PRP)

Three phases with progressive deliverables: Prompting → RAG → Research Portal Product

Work mode	Individual
Grade weight	Phase 1: 10% • Phase 2: 15% • Phase 3: 20% (Total: 45% of course grade)
Primary goal	A research-grade portal that answers questions with evidence, citations, and exportable research artifacts
Stack	Your choice (Python + Streamlit/Gradio recommended). Free-tier and open-weight tools allowed.
Submission	GitHub repo (code + data manifest) + reports + demo recording

1. **Overview (what you are building)**

By the end of Phase 3 you will deliver a Personal Research Portal (PRP): a lightweight product that helps you move from a research question to a grounded synthesis. It ingests a domain corpus, retrieves evidence, produces citation-backed answers and research artifacts, and logs evaluation results.

Minimum PRP capabilities (MVP)

- Ingest a domain corpus (PDFs/URLs/notes) with metadata (author, year, source type, link/DOI).
- Retrieve evidence (semantic retrieval at minimum).
- Generate answers where every major claim is backed by an inline citation to a source/chunk ID.
- Produce at least one research artifact: evidence table, annotated bibliography, or synthesis memo.

- Save research threads (query, retrieved chunks, answer) and export outputs (Markdown/CSV/PDF).
- Run an evaluation set and report groundedness/faithfulness plus at least one additional metric.

What counts as a citation in this project

- You may cite using (source_id) or (source_id, chunk_id) as long as it uniquely maps to text you ingested.
- Your repo must include a data manifest that makes every citation resolvable to: source title + link/DOI + local file or snapshot.
- Example citation formats: (Smith2023, chunk_07) or (Smith2023) if the source_id uniquely identifies one ingested document.
- If the corpus does not support a claim, your system must say so rather than guessing.

2. **How the three phases fit together**

- Phase 1 produces your research framing, prompt kit, and an evaluation rubric you will reuse.
- Phase 2 uses that same framing to build research-grade retrieval and grounding over your corpus.
- Phase 3 wraps Phase 2 into a usable portal and generates exportable research artifacts.

Design intent: each phase should feel like a direct upgrade of the previous one, not a reset.

3. **Deliverables at a glance**

Phase	**Weight**	**Purpose**	**Core deliverables**	**MVP scope (minimum)**	**Stretch goals (examples)**
Phase 1 — Prompt the Research Domain	10%	Design prompts and evaluate model behavior on research tasks.	Framing brief; prompt kit; evaluation sheet; analysis memo.	2 tasks • 2 models • 2 test cases per task • 2 prompt variants per task (16 total runs).	3rd model; adversarial tests; automated checks; better uncertainty calibration.

Phase 2 — Ground the Domain (Research-Grade RAG)	15%	Build baseline RAG over your corpus; add production patterns; evaluate.	Ingestion pipeline; baseline RAG; enhancements; evaluation report + logs.	15 sources • 20 queries • 1 enhancement • citation-backed answers.	Hybrid retrieval; reranking; section-aware chunking; structured citations; RAGAs/ARES.
Phase 3 — Build the Personal Research Portal Product	20%	Turn your RAG into a usable portal that produces research artifacts.	Working app + demo; final report; generated artifacts; evaluation view.	Search + citations + threads + export + at least 1 artifact type.	Agentic research loop; knowledge graph; gap finding; BibTeX export; improved UX.

4. **Phase 1 — Prompt the Research Domain (10%)**

Goal

Pick a research domain, define a main research question, and test how different models handle core research tasks. You will produce a prompt kit you will reuse in Phases 2 and 3.

You must choose

- A domain and one main research question.
- 4–6 sub-questions that break your main question into retrievable parts.
- Two tasks from the task menu (each with a required output format).
- Two accessible models (three allowed for stretch).

Task menu (choose 2) — with concrete output formats

- Paper triage: output a 5-field summary (Contribution, Method, Data, Findings, Limitations).
- Claim–evidence extraction: output 5 rows with Claim | Direct quote/snippet | Citation (source_id, chunk_id).
- Cross-source synthesis: output a table of Agreement | Disagreement | What evidence supports each side.

- Research gap spotting: output 3 gaps, each with 'What is missing' + 'What evidence would resolve it'.
- Citation formatting: output a paragraph with inline citations plus a short reference list from your manifest.

Definitions (what we mean in Phase 1)

- Task: a repeatable research operation with a required output format (e.g., paper triage or claim-evidence extraction).
- Test case: one concrete input for a task (e.g., Paper A PDF; or Paper B, Methods section). You will run every test case through both prompts and both models.
- Prompt variant: two versions per task - Prompt A (baseline) and Prompt B (improved with structure/guardrails).

Worked example (illustrative - you will use your own domain)

- Domain + question: "Trustworthy RAG evaluation methods" - Main question: "How do different faithfulness metrics fail, and how can we combine them?"
- Tasks (pick 2): (1) Paper triage; (2) Claim-evidence extraction.
- Test cases (2 per task):
  - Paper triage test cases: Paper A (survey) and Paper B (empirical study).
  - Claim-evidence extraction test cases: Paper C (benchmark) and Paper D (critique).
- Prompts (2 per task): Prompt A = minimal instruction; Prompt B = structured output + "cite chunk_id" + "say unknown when missing".
- Models (2): any two you can access. Run all combinations (16 runs for MVP) and score outputs for groundedness and citation correctness.

MVP checklist (do these in order)

- Write a 1-page framing brief: domain, main question, sub-questions, scope (what you include/exclude).
- Create 2 test cases per chosen task (4 total test cases across Phase 1). A test case is one concrete input, such as a specific paper or section.
- Write two prompt variants per task: Prompt A (baseline) and Prompt B (improved structured prompt with guardrails).
- Run both models across all test cases and both prompt variants, and record outputs in the evaluation sheet (2 tasks x 2 cases x 2 prompts x 2 models = 16 runs).
- Score each output using the Phase 1 rubric and write a 1–2 page analysis of failure modes and fixes.

Phase 1 scoring rubric (1–4) — what the numbers mean

- 4: Correctly grounded and structured; citations are correct; uncertainty is stated when evidence is weak.
- 3: Mostly correct and structured; minor missing nuance OR minor citation/format issues.
- 2: Partially correct; key omissions OR weak grounding OR vague citations.
- 1: Not usable; hallucinated claims, fabricated citations, or fails the required structure.

Phase 1 deliverables (submit together)

- Framing brief (1–2 pages).
- Prompt kit (Markdown): prompts + why each constraint exists.
- Evaluation sheet (CSV/Google Sheet/Markdown table): scores + notes per run (for MVP, include 16 rows - one per run).
- Analysis memo (1–2 pages): patterns, failures, and Phase 2 design choices.

Common mistakes to avoid

- Choosing tasks that do not force citations (then you cannot assess grounding).
- Scoring without any written justification (graders cannot tell what you observed).
- Changing your question mid-phase (it breaks continuity into Phase 2).

5. **Phase 2 — Ground the Domain (Research-Grade RAG) (15%)**

Goal

Build a baseline RAG pipeline over your own corpus and upgrade it with production patterns. This phase prioritizes retrieval quality, traceability, and evaluation—not UI polish.

Corpus requirements (MVP)

- 15–30 sources minimum. At least 8 should be peer-reviewed papers, standards, or reputable technical reports.
- Every source must have metadata: source_id, title, authors, year, type, venue (if applicable), link/DOI, and a 1–2 sentence relevance note.
- Source acquisition can be manual (download PDFs) or automated (crawler/agentic selector). Either way, do not rely on live links alone: store the raw artifact (PDF/HTML snapshot/extracted text) under data/raw/ OR include a reproducible download script.
- If you use an automated crawler/agent to pick sources, include the code plus the run configuration (prompts/filters) and a short note describing how sources were selected (manual vs scripted vs agentic).

Baseline RAG MVP (do these in order)

- Ingest sources: parse and clean text; store text + metadata; create a data manifest (CSV/JSON).
- Chunk with a documented strategy (chunk size, overlap). Use section-aware chunking for papers if feasible.
- Embed and index in a vector store (FAISS acceptable).
- Retrieve top-k chunks per query and generate an answer that cites chunk/source IDs.
- Create an evaluation set of at least 20 queries: 10 direct, 5 synthesis/multi-hop, 5 ambiguity/edge cases.
- Example direct query: "What does <method/metric> measure, and what are its known failure modes?"
- Example synthesis query: "Compare Paper A vs Paper B on <criterion>; where do they agree and disagree?"
- Example edge-case query: "Does the corpus contain evidence for <claim>?" (If not found, your system must say so.)
- Evaluate groundedness/faithfulness plus one additional metric (answer relevance, context precision/recall, citation precision). Interpret results.

Required production patterns (must-have)

- Logging: store queries, retrieved chunks, model outputs, and prompt/version IDs.
- Reproducibility: pinned dependency versions; one-command run path (Makefile or a single script).
- Trust behavior: refuse to invent citations; explicitly flag missing or conflicting evidence.

Enhancements (implement at least 1 for MVP)

- Query rewriting or decomposition (question → sub-queries).
- Reranking (cross-encoder, LLM reranker, or heuristic reranker).
- Metadata filtering (year/author/type) and faceted retrieval.
- Hybrid retrieval (BM25 + vector).
- Confidence / evidence-strength scoring.
- Structured citations (inline citations + reference list from your manifest).

Phase 2 deliverables (submit together)

- Code + repo (baseline RAG) with README instructions.
- Data manifest (CSV/JSON) with metadata for every source.
- Evaluation report (3–5 pages): query set design, metrics, results, and what improved with enhancements.

- Run logs (machine-readable): at least 20 queries with retrieved chunks and outputs.

Phase 2 acceptance tests (what graders will check quickly)

- A single command produces: retrieval results, an answer with citations, and a saved log entry.
- Randomly sampled answers have citations that resolve to real source text in your corpus.
- Your evaluation report includes at least 3 representative failure cases with evidence.

6. **Phase 3 — Build the Personal Research Portal Product (20%)**

Goal

Turn your Phase 2 system into a usable portal that supports a research workflow: question → evidence → synthesis → export. Your portal should not just chat; it should produce research artifacts.

Product MVP requirements

- Interface: a working UI with at least: search, ask, show sources/citations, and history.
- Research threads: save query + retrieved evidence + answer (file-based OK).
- Artifact generator: produce at least one of: evidence table, annotated bibliography, or synthesis memo with references.
- Export: allow export of artifacts (Markdown/CSV/PDF).
- Evaluation: a page or script that runs your query set and summarizes metrics plus representative examples.
- Trust behavior: every answer includes citations; missing evidence is stated explicitly with a suggested next retrieval step.

Recommended artifact schemas (pick at least 1 for MVP)

- Evidence table: Claim | Evidence snippet | Citation (source_id, chunk_id) | Confidence | Notes.
- Annotated bibliography: 8–12 sources with 4 fields (claim, method, limitations, why it matters).
- Synthesis memo: 800–1200 words with inline citations and a reference list generated from your manifest.

Phase 3 deliverables (submit together)

- Working PRP app + instructions to run locally.

- Demo recording (3–6 minutes) showing retrieval, citations, artifact generation, and export.
- Final report (6–10 pages): architecture, design choices, evaluation, limitations, and next steps.
- Generated research artifact outputs included in the repo and referenced in the report.

Stretch goals (optional)

- Agentic research loop (plan → search → read → synthesize) with guardrails and logs.
- Knowledge graph view (entities/concepts linked to source passages).
- Automatic disagreement map (conflicts surfaced with citations).
- Gap finder: missing evidence + targeted next retrieval actions.
- Improved UX: filters by year/venue/type, reading list, tagging, saved collections.

7. **Grading (points aligned to course weight)**

Your project is graded on rigor, clarity, and evidence, not on paid tooling. Points below map directly to the phase weights.

Phase 1 (10 points = 10%)

- Framing quality and scope discipline (3): clear question, sub-questions, inclusions/exclusions.
- Prompt kit quality (4): structured prompts, guardrails, reusable prompt cards.
- Evaluation rigor and analysis (3): consistent scoring, failure tags, actionable takeaways.

Phase 2 (15 points = 15%)

- Corpus + manifest quality (4): metadata completeness, reproducibility, source credibility.
- Retrieval + grounding implementation (5): working RAG with correct citations and logs.
- Enhancement + measurable improvement (3): at least one enhancement with evidence it helped.
- Evaluation report quality (3): query set design, metrics, interpretation, failure cases.

Phase 3 (20 points = 20%)

- Portal MVP functionality (8): UI, threads, citations, export, reliability of core flow.

- Research artifacts (4): artifact schema correctness, usefulness, citation traceability.
- Evaluation + trust behaviors (4): run query set, summarize metrics, handles missing evidence.
- Engineering + communication (4): clean repo, run instructions, demo, clear final report.

8. **Submission requirements**

Repo expectations (minimum)

- A README that tells a grader how to run your system in 5 minutes or less.
- Pinned dependencies (requirements.txt or pyproject.toml).
- data/data_manifest.csv (or JSON) with the required metadata schema.
- Your repo must include data/raw/ with the downloaded PDFs/snapshots you used, OR a reproducible download script that regenerates data/raw/.
- A folder of outputs (artifacts/exports) generated by your system.
- Logs from evaluated runs (machine-readable).

Recommended repo structure

```
repo/
  README.md
  requirements.txt (or pyproject.toml)
  data/
    raw/ (downloaded PDFs, HTML snapshots, original notes)
    processed/ (parsed text, chunks, intermediate files)
    data_manifest.csv (or JSON)
  src/
    app/ (Phase 3 UI)
    ingest/ (parsers + chunking + optional download scripts)
    rag/ (retrieval + generation)
    eval/ (query sets, scripts, results)
  outputs/ (artifacts, exports)
  logs/ (runs, prompts, versions)
  report/ (Phase 1, Phase 2, Phase 3 writeups)
```

AI usage disclosure (required)

- Include a 1-page AI-usage log listing: tool name, what you used it for, and what you changed manually afterward.
- You are responsible for correctness, citations, and meeting requirements even if an AI tool suggested code or text.

Appendix A — Templates (copy/paste)

A1. Prompt card template

- Prompt name:
- Intent:
- Inputs (what you provide):
- Outputs (required structure):
- Constraints / guardrails (no fabricated citations; cite chunk IDs; etc.):
- When to use / when not to use:
- Failure modes to watch for:

A2. Evaluation row template (per answer)

- Task ID: | Test case ID: | Query ID:

Task ID:

- Prompt ID + model:
- Retrieved evidence IDs:
- Score 1 — Groundedness/faithfulness (1–4):
- Score 2 — Citation correctness (1–4):
- Score 3 — Usefulness (1–4):
- Notes + failure tag (missing evidence, wrong citation, overconfident, etc.):

A3. Source metadata schema (minimum fields)

source_id, title, authors, year, source_type, venue, url_or_doi, raw_path, processed_path, tags, relevance_note

raw_path should point to the file you store under data/raw/. processed_path is optional but recommended (e.g., parsed text under data/processed/).