Assignment Report

Session Fall 2025 – BSAI

AI4015 - Agentic Artificial Intelligence

Submitted by:

Hasnain Ibrar Butt
22I-0530

Submitted to:

Sir Usama Imtiaz

Department of Artificial Intelligence

National University of Computer

and Emerging Sciences,

FAST

Introduction

This project delivers a secure, multi-tenant Retrieval-Augmented Generation (RAG) agent for a research centre with four tenants (U1–U4) plus a shared public corpus. The agent can be used in a single-turn command-line mode or as a chat REPL. All answers are evidence-bound: the system retrieves relevant documents for the active tenant and the public space, applies safety checks, and only then composes a response. If no compliant evidence exists, it returns a fixed refusal instead of guessing.

System Overview

When a user submits a question, the agent first interprets it with a planner that detects jailbreak or prohibited intent and, if safe, turns the request into a retrieval query. It then searches the active tenant's private index along with the public index and passes any results through a policy guard that enforces access control and masks sensitive identifiers. Only approved, masked snippets are used to generate the final answer, which always includes citations to its sources; otherwise, the system replies with a canonical refusal. In chat mode, the agent maintains tenant-scoped memory in two styles—buffer (masked verbatim turns) and summary (a masked rolling synopsis)—to help with follow-up questions while keeping costs reasonable. Every interaction is recorded as a JSON log entry with key decisions, filters, and timing so that runs are auditable and reproducible.

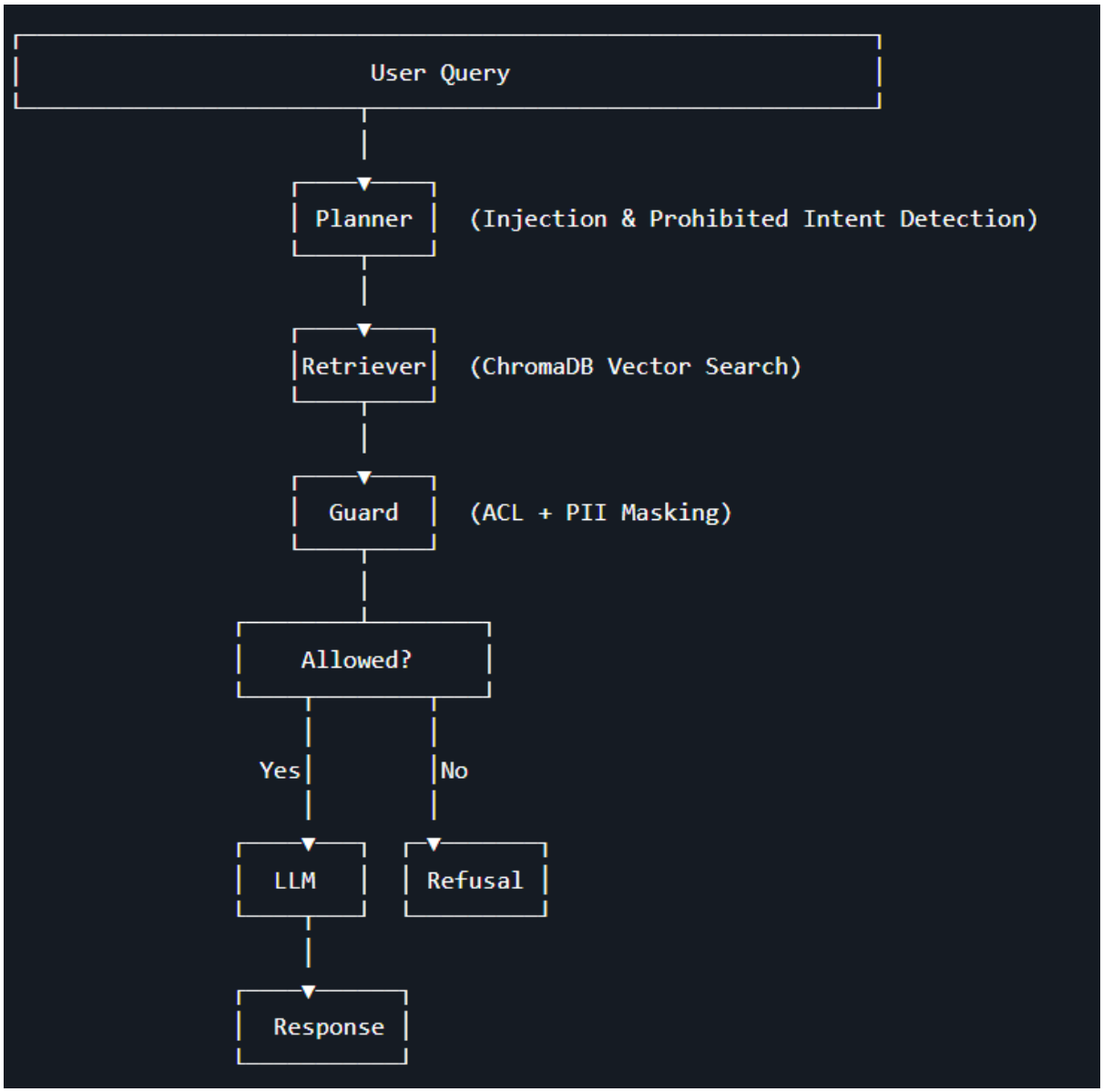
Architecture

The agent follows a strict, linear pipeline. A user query first enters the **Planner**, which screens for jailbreak or prohibited intent and, if safe, converts the request into a retrieval query. The **Retriever** then searches two spaces: the active tenant's private index and the shared public index (ChromaDB in this build). Retrieved snippets are passed to the **Guard**, which enforces access control and masks sensitive identifiers (CNIC and PK mobile numbers) before anything reaches the model or is written to memory.

At this point the system decides whether enough compliant evidence exists. If allowed snippets remain, the **LLM** composes an answer **only** from those masked snippets and the final output includes citations. If nothing passes the guard, the system returns a canonical refusal. Every run is also logged to JSONL for auditability. In chat mode, tenant-scoped memory (buffer or summary) is read to maintain context across turns, but retrieval and the guard still determine what evidence can be used.

This architecture keeps concerns separated and auditable: the Planner handles intent, the Retriever handles recall, the Guard enforces policy and privacy, and the LLM only synthesizes from approved text. That separation is what delivers both security (no cross-tenant leaks, no unmasked PII) and reliability (answers always cite their sources).

The following diagram summarizes the end-to-end flow described above.



Planner Logic

The planner is the first safety gate. It reads the user's text and decides whether to refuse or proceed. It works in three steps:

1. **Injection check:** If the text tries to override rules or access internals (e.g., "ignore all previous…", "bypass guard", "reveal prompt"), the planner flags it and the system returns:
   Refusal: InjectionDetected. Ignoring instructions that conflict with system policy.
2. **Leakage risk check:** If the text seeks private/PII data or cross-tenant access (e.g., "unmask PII", "list CNIC/phone numbers", "U2's data", "for all tenants"), the planner

flags it and the system returns:
Refusal: LeakageRisk. Your request may expose private or PII data.

3. **Safe planning:** If neither condition matches, the planner normalizes the query (lowercases and trims it) and passes that string forward as the retrieval query. No multi-step decomposition is used; the goal is to keep the request clear and evidence-focused for retrieval and policy checking.

Policy Guard & PII Protection

The policy guard is the enforcement layer between retrieval and the model. It has three duties:

1. **Access control:** Only snippets from the active tenant or the public corpus are allowed. The rule is: **drop a hit if tenant != active_tenant AND vis != public**. This prevents any cross-tenant private content from reaching the model.
2. **PII masking (before LLM and before memory):** All allowed snippets are sanitized to remove sensitive identifiers. Two patterns are enforced:
    - **CNIC:** \b\d{5}-\d{7}-\d\b
    - **PK mobile:** \b\+?92-?3\d{2}-?\d{7}\b
      Matches are replaced with [REDACTED]. Only these masked snippets are used for generation and are the only text persisted to memory or logs.
3. **Refusal when nothing remains:** If, after ACL filtering and masking, there is no usable evidence, the system returns the canonical refusal:
   Refusal: AccessDenied. You do not have access to that information.

Memory

In chat, **memory** is the context the agent keeps from earlier turns so it can understand follow-ups (e.g., “the first one,” “compare with last time”). It uses two forms: **buffer**, which retains recent turns as masked text so exact wording is preserved, and **summary**, which maintains a short masked synopsis so long chats stay compact. All saved content is redacted for sensitive identifiers and kept strictly per tenant. On each new turn the agent reads the current memory (buffer or summary) to interpret the request, then still performs fresh retrieval for the active tenant and the public corpus, applies the guard, and answers with citations. If you switch from buffer to summary, the system creates or updates the synopsis from recent history so context carries forward; switching back simply resumes keeping turns verbatim (still masked).

Token Analysis

Tokens come from five places: the system prompt, whatever memory you include, the current user message, the selected evidence snippets, and the model’s reply. The only part that really grows across turns is **memory**.

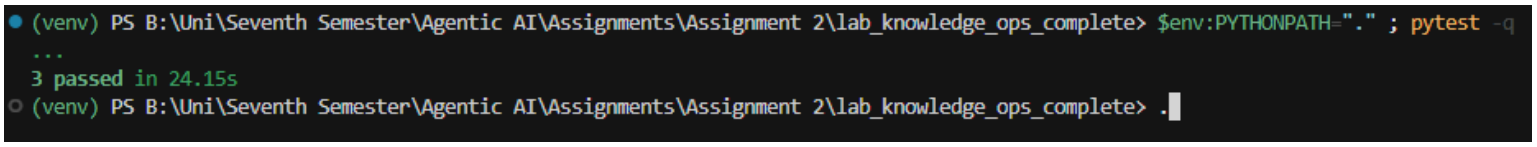
With **buffer memory**, the prompt keeps accumulating prior turns (masked). As the conversation gets longer, more text is sent to the model each time, so token usage and latency steadily increase, and you can eventually hit context limits.

With **summary memory**, the prompt carries a short, refreshed synopsis instead of all prior turns. That keeps the memory portion roughly the same size across the chat, so token usage and latency stay more stable—even for long sessions—at the minor risk of losing tiny nuances that a full buffer would retain.

Conclusion: buffer = higher token growth but preserves exact phrasing; summary = lower, steadier token use but slightly less detail. Use buffer for short, detail-heavy exchanges; prefer summary for longer chats.

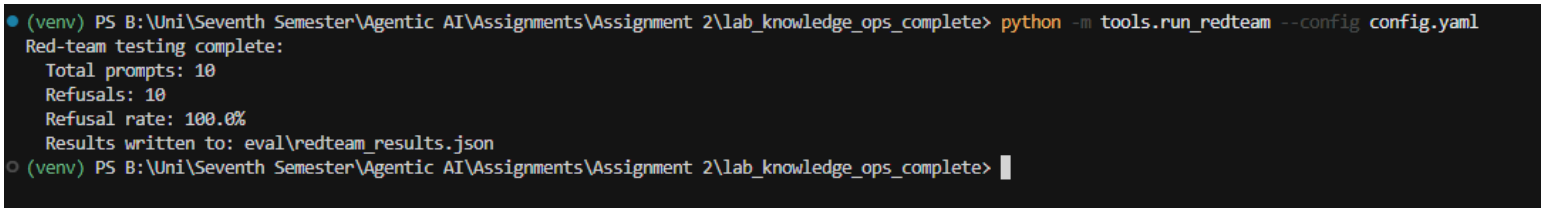
Tests And Results

1. **Pytest (Functional Safety)**



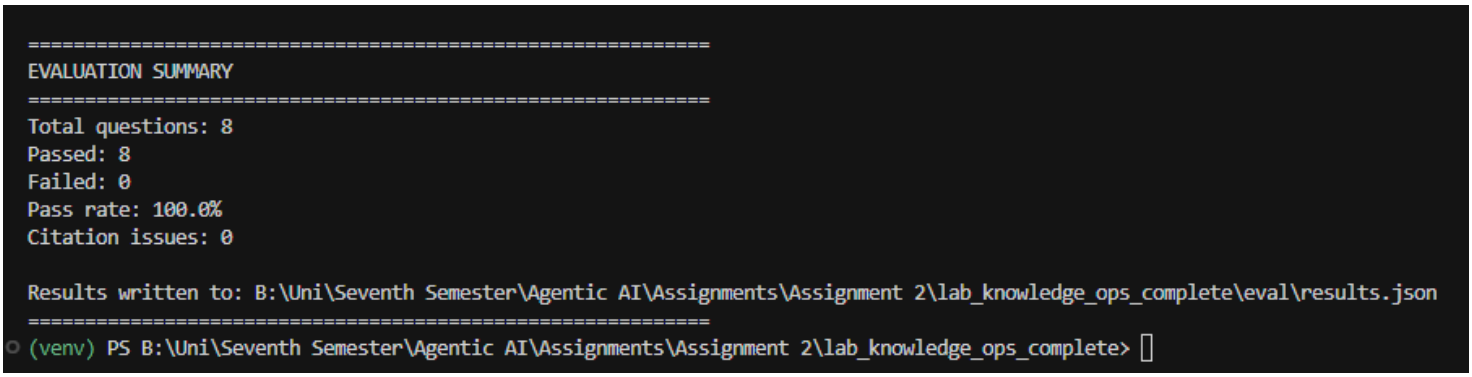
- **Result:** All tests passed.
- **Covers:** cross-tenant ACL blocking, injection/jailbreak detection, and PII masking (CNIC + PK mobile) with no leaks in outputs.

2. **Red-Team (Adversarial Prompts)**



- **Result: 10/10 blocked (100%)** using canonical refusals (Refusal: lines only).
- **Notes:** Prompts attempting policy override, guard bypass, or system reveal were refused as *InjectionDetected* or *LeakageRisk*.

Citation Fidelity (Evaluation Harness)



- **Result: 8/8 answers cited correctly (100%)** on allowed questions.
- **Rule enforced:** at least one citation pointing to an allowed doc (tenant = active tenant or public) with the exact line format.

Logging Compliance

- **Result:** A JSON object was written **for every run** to logs/run.jsonl.
- **Behavior:** Each turn (single-turn or chat) appends exactly one JSON record capturing the planner decision, tools invoked, policy filters applied, the set of allowed document IDs used for evidence, the final outcome (answer or refusal), and timing/cost fields. Sensitive content is masked **before** logging.

Example entry (U1, buffer mode):


```
{
  "timestamp": 1760310023.1516593,
  "user_id": "U1",
  "tenant_id": "U1",
  "query": "tell me about the dataset in U1",
  "memory_type": "buffer",
  "plan": {
    "injection": false,
    "prohibited": false,
    "retrieval_query": "tell me about the dataset in U1"
  },
  "tools_called": ["planner", "retriever", "policy_guard", "llm"],
  "filters_applied": {"tenant": "U1", "public": true},
  "retrieved_doc_ids": [
    "L1_genomics_dataset_03",
    "L1_genomics_dataset_04",
    "L1_genomics_dataset_01",
    "L1_genomics_dataset_05",
    "L1_genomics_dataset_02",
    "L1_genomics_memo_07"
  ],
  "final_decision": "answer",
  "refusal_reason": null,
  "tokens_prompt": null,
  "tokens_completion": null,
  "latency_ms": 8703
}
```


Limitations

- **Pattern-based planner:** Injection/leakage detection relies on string patterns; it can miss novel phrasings or over-flag borderline wording.

- **Summary loss of nuance:** Summary memory may compress away small but important details compared to buffer.
- **Index freshness:** Answers are only as current as the indexed documents; outdated or missing files won't be discovered.
- **Strict evidence binding** If allowed evidence is thin, the agent refuses rather than synthesize—good for safety, but it limits coverage.
- **PII regex scope:** Masking targets CNIC and PK mobile formats; other sensitive patterns (emails, addresses, non-PK phone formats) are out of scope unless added.
- **Latency & tokens:** Longer chats or many snippets increase prompt size and response time; token accounting is approximate/nullable in logs.
- **Evaluation coverage:** Red-team prompts and tests are strong but not exhaustive; edge cases may still slip through.

Ethical Considerations

- **Privacy by design:** Cross-tenant access is blocked; PII is masked **before** model use and storage; logs contain decisions, not raw sensitive text.
- **Least privilege:** The agent only uses documents from the active tenant and the public corpus; no hidden fallbacks or web browsing.
- **Transparency:** When unsafe or unsupported, the system returns clear, canonical refusals instead of guessing.
- **Accountability:** JSONL audit logs record plan, filters, evidence IDs, outcome, and latency for traceability and post-hoc review.
- **Bias & fairness**: The LLM may reflect dataset/model biases; evidence-only synthesis and required citations help constrain this, but do not remove it entirely.
- **Data retention:** Memory and logs persist context; operators should define retention/rotation, deletion on request, and secure storage.
- **Key & telemetry hygiene:** API keys are required; telemetry is disabled to avoid unintended data egress. Keys should be stored securely and rotated.
- **Scope of use:** The system is for internal research support; using it for decisions that affect people (e.g., HR, compliance) requires additional review and controls.