Assistant Completion 32-Hour Sprint **(This is max time, You can get done and submit prior to this is you are done-Helps with better overall score.)**: Integration Push (Owners: Nilesh, Noopur, Parth, Chandresh)

Goal: complete core integration points with Seeya + Sankalp so the pipeline flows end-to-end:

Message → Summarize (/api/summarize) → Create Task (/api/process_summary) → Respond (/api/respond) → Recall (/api/search_similar) → Coach feedback (/api/coach_feedback) → Metrics.

Deadline: 32 hours from start.

Deliverables (shared)

1. New API endpoints (integrated into assistant-live-demo repo):
    - POST /api/respond → produce and store response.json.
    - POST /api/search_similar → return related past summaries/tasks.
    - POST /api/coach_feedback → store coach feedback & score.
    - GET /api/metrics → return service metrics (counts/latency/errors).
2. DB schema additions (SQLite assistant_demo.db):
    - responses(response_id, task_id, user_id, response_text, tone, status, timestamp)
    - embeddings(id, item_type, item_id, vector_blob, timestamp) (vector storage simplified; can be JSON/text for now)
    - coach_feedback(id, summary_id, task_id, response_id, score, comment, timestamp)
    - metrics(id, endpoint, status_code, latency_ms, timestamp)
3. Streamlit tester updates:
    - Show Response card after Task.
    - Show Related Past Context from /api/search_similar.
    - Add Feedback tab to POST to /api/coach_feedback.
    - Add Metrics tab reading /api/metrics.
4. Tests:
    - tests/test_integration.py updated to exercise new endpoints and validate DB writes.

5. VALUES.md entries added by each owner (≤150 words each) addressing Humility, Gratitude, Honesty.

Who Does What (owners & integration points)

Noopur — Responder integration (primary)

- Implement /api/respond.
    - Input: task.json (from Sankalp) or {task_id}; call into existing responder_agent.py.
    - Output: response.json saved to DB.
    - Tone/safety checks via safety_filter.py (if flagged, set status="flagged").
- Update Streamlit: display Response card and add “Send Response” simulation button.
- Unit tests for response generation + safety flags.

Integration points: read tasks table, write responses table.

Chandresh — EmbedCore & Recall (primary)

- Implement embedding index + /api/search_similar.
    - Accept message.json or summary_id; return top-3 similar summaries/tasks with similarity scores.
    - Store embeddings for new summaries (on /api/summarize write hook) — coordinate with Seeya to invoke embedding write or poll DB.
- Provide simple similarity using SentenceTransformer or a placeholder cosine on mocked vectors. Persist vectors in embeddings table.
- Add small script rebuild_embeddings.py for reindexing.
- Unit tests for search_similar and embedding storage.

Integration points: subscribes to summaries writes; returns context to Streamlit UI.

Parth — CoachAgent & Feedback (primary)

- Implement /api/coach_feedback.
    - Input: {summary_id, task_id, response_id, scores: {clarity, relevance, tone}, comment}
    - Compute score (aggregate) and persist in coach_feedback.

    - Optionally send a reward update to Chandresh endpoint (e.g., /api/rl_reward) — simple POST hook.

- Build simple auto-scoring rules (clarity from summary length/key phrases; relevance via similarity score from Chandresh).
- Update Streamlit Feedback tab to accept ratings and display coach logs.
- Unit tests to assert feedback persistence and coach score logic.

Integration points: reads summary/task/response; writes coach feedback; optionally notifies RL.

Nilesh — Metrics, Logging, Execution Tracking (primary)

- Implement middleware or small wrapper logging every API call to metrics table (endpoint, status, latency).
- Build /api/metrics summarizing:
    - total_messages, total_summaries, total_tasks, total_responses, avg_latency_ms, error_rate.
- Add Streamlit Metrics tab that fetches /api/metrics.
- Ensure logs are appended on success/error for /api/respond, /api/search_similar, /api/coach_feedback.
- Provide a short monitoring_readme.md with commands to fetch metrics and tail logs.

Integration points: every service logs to metrics; test harness calls metrics endpoint.

Step-by-step Schedule (32 hours)

Hour 0 — Kickoff (0–0.5h)

- Quick sync call (15–30 minutes): confirm DB type (SQLite), repo branch, and where to push. Agree on port and API base (/api/*).
- Create a new branch alpha-integration in repo and push skeleton endpoints.

Hours 0.5–8 — Phase A (8 hours)

- Noopur: scaffold /api/respond endpoint, basic responder hook, DB write to responses.
- Chandresh: scaffold embedding storage function and embeddings table; implement a placeholder similarity function.
- Parth: scaffold /api/coach_feedback endpoint and feedback table.
- Nilesh: implement basic metrics logger middleware and metrics table; ensure endpoint exists.

Deliverable by hour 8: endpoints exist (stubs) and DB tables created; basic tests for DB writes running.

Hours 8–18 — Phase B (10 hours)

- Noopur: implement safety filter integration and response status logic; add unit tests for flagged content.
- Chandresh: implement /api/search_similar returning nearest 3 items; connect to summaries writes (either hook or polling).
- Parth: implement auto-scoring logic using Chandresh similarity; persist coach feedback; basic Streamlit UI for feedback.
- Nilesh: complete metrics capture for these endpoints; add simple aggregation logic.

Deliverable by hour 18: functional /api/respond, /api/search_similar, /api/coach_feedback, metrics logging; Streamlit tabs partially updated.

Hours 18–26 — Phase C (8 hours)

- Joint work: integrate with Seeya/Sankalp pipeline:
    - Confirm Seeya's /api/summarize triggers embedding write or run a script to index existing summaries.
    - Ensure Sankalp's /api/process_summary yields task.json that Noopur can consume.
- Update Streamlit: Response card appears automatically after Task; Related Past Context shown; Feedback actions visible.
- Run and fix integration issues; add DB constraints if necessary; iterate.

Deliverable by hour 26: full pipeline end-to-end in local environment: Message → Summary → Task → Respond → Similar Context → Coach Feedback → Metrics.

Hours 26–32 — Phase D (6 hours)

- Write/complete tests: update tests/test_integration.py to exercise new endpoints end-to-end.

- Each owner writes short VALUES.md entry and pushes to branch.
- Final bug fixes, README updates (how to run metrics and how to test).
- Prepare 60–90s demo clip or screenshots and push.

Deliverable by hour 32: passing integration test, updated Streamlit demo, DB populated for demo, VALUES.md committed.

Acceptance Criteria (what “done” looks like)

- POST /api/respond returns response.json and writes to responses table.
- POST /api/search_similar returns top-3 related items with similarity scores.
- POST /api/coach_feedback stores feedback and returns aggregated coach score.
- GET /api/metrics returns non-empty metrics for endpoints exercised.
- Streamlit tester shows Response card, Related Past Context and Feedback actions; Metrics tab shows counts.
- tests/test_integration.py passes locally.
- Each participant committed VALUES.md with Humility, Gratitude, Honesty reflections (required).

Quick API Contracts (copy into repo)

/api/respond (POST)

Input: task.json (see earlier contract)

Output:

```
{ "response_id":"r123", "task_id":"t123",
"response_text":"...", "tone":"polite", "status":"ok",
"timestamp":"ISO8601" }
```

/api/search_similar (POST)

Input: { "summary_id":"s123" } or message.json

Output:

```
{ "related":[ { "item_type":"summary", "item_id":"s456",
"score":0.87, "text":"..." }, ... ] }
```

/api/coach_feedback (POST)

Input:

```
{ "summary_id":"s123", "task_id":"t123",
"response_id":"r123",
  "scores":{"clarity":4,"relevance":5,"tone":4},
"comment":"..." }
```

Output: { "feedback_id":"f123", "score":13, "stored":true }

/api/metrics (GET)

Output: sample

```
{ "total_messages":123, "total_responses":45,
"avg_latency_ms":120, "error_rate":0.02 }
```

Testing & Run commands (for README)

- Start API: uvicorn api.main:app --reload --port 8000
- Start Streamlit: streamlit run streamlit/demo_streamlit.py
- Run tests: pytest -q tests/test_integration.py or python tests/test_integration.py
- Inspect DB: sqlite3 assistant_demo.db "select * from responses;"

Small operational notes

- Use feature branch alpha-integration and push frequently; everyone merges only after tests pass locally.
- Keep commits small and include short messages: e.g., noopur: add /api/respond and write response table.
- If embedding model is slow, use a lightweight text hashing or mini-vector placeholder for the sprint and mark as TODO.