Agentic-LM — Step-by-step tasks (minimal daily pushes + values)

Goal: deliver a working Uniguru-LM prototype (KB-grounded NLP + Vaani TTS) integrated into BHIV/Gurukul in a short sprint. Focus people: Nipun, Nisarg, Vijay, Karthikeya, Vedant, Rishabh. Two main days of focused work (Day 1 = build; Day 2 = integrate, test, deploy). Every day ends with a mandatory 3-line reflection (Humility / Gratitude / Honesty) — 5 pts each.

Quick summary (one line)

Make a retrieval-first, indigenous NLP composer (templates + n-gram + tiny GRU), expose / compose, add Vaani TTS, wire to Gurukul UI, log feedback for RL.

Day 0 — Prep (30–60 minutes, everyone)

Goal: align envs, access and minimal smoke test.

Actions (each person):

- Confirm access: repo, NAS mount path, Qdrant URL, Mongo URL, Vaani credentials, dev server details.
- Pull the latest v1-BHIV_CORE and create branch uniguru-jugaad/<initial>.
- Share one-line readiness in the team channel: Ready: <name> — Qdrant OK / NAS path OK / Vaani OK.

Deliverable: all members reply Ready in the group with statuses.

Integration prompt for group:

- Ask: “Do you have access to NAS:/vedabase and QDRANT_URL?”
- Tell: “I will use branch uniguru-jugaad/<initial>; please not push to main.”

Day 1 — Build core pieces (minimum viable push by EOD)

Nisarg — Composer (templates + n-gram + tiny GRU scaffold)

Daily goal (small win): composer.compose(extractive_answer, lang) -> grounded draft text.

- Implement templates (explain/compare/example) and an n-gram scorer to smooth templates.
- Add a simple GRU stub file (composer/gru.py) with train script but optional run — ensure composer uses n-gram fallback if GRU not ready.
- Enforce grounding: any sentence must have at least one token overlap with top_chunks (simple check).

Deliverable: composer/compose.py function plus unit test tests/test_compose.py.

Integration check: composer callable via local import and responds in EN/HI.

What to ask others:

- To Nipun: “Send sample top_chunks schema; composer will expect that exact shape.”
- To Vijay: “I’ll need composer callable by the API; approved function signature: compose(trace_id, extractive_answer, top_chunks, lang)?”

What to ask others:

- To Vijay: “Will trace_id be stable across retry? I’ll store it in UI state.”
- To Karthikeya: “Play audio on click; do you need any extra pre-signed URL handling?”

Day 1 end — Minimum Viable Push:

- All services started on dev host, simple end-to-end call returns composed text and (mock/real) audio URL.
- Daily reflection (each member — 3 short lines):

    1. Humility: one limit encountered.
    2. Gratitude: one help/tool you appreciated.
    3. Honesty: one incomplete or risky shortcut.

Day 2 — Integrate, RL hooks, QA, and deploy test

Nisarg — strengthen grounding & policy hook

Goal: add grounding verification and template selection policy (epsilon simple).

- If grounding fails, auto-fallback to more extractive template (shorter, cite heavy).
- Record chosen template_id as action in RL log.

Deliverable: composer writes {template_id} and grounded: true|false into trace.

What to ask others:

- To Shashank (if present): “What reward thresholds do we trigger policy update on?”

What to ask others:

- To Nipun/Nisarg: “When feedback arrives, what minimal data do you need to update policy?”

Smoke tests and acceptance (team)

- Run 10 test queries (EN + HI) through UI, confirm:
  - final_text present, citations shown,
  - audio plays (or shows pending),
  - feedback flow records reward in Mongo,
  - composer grounded:true for >90% cases.
- If any failure, tag owner and fix.

Deliverable: smoke_results.md with pass/fail per test and trace_ids.

End of Day 2 deliverable: deployable dev instance; Gurukul testers can try with 10–50 users; logs and feedback collection live.

Final Reflection (each team member; 3 lines): Humility / Gratitude / Honesty — saved to repo reflections/<name>.md and added to PR description.

Integration matrix — who depends on who (one line each)

- Vijay ← Nipun, Nisarg, Karthikeya: API calls composer + TTS.
- Nipun ← BHIV bucket / NAS: needs read path and correct doc metadata.
- Nisarg ← Nipun: composer must get chunk format and citation offsets.
- Karthikeya ← Vijay: needs API call pattern for TTS invocation.
- Vedant/Rishabh ← Vijay/Karthikeya: require trace_id, audio_url, final_text fields.

Message to teammate:

Nisarg → Nipun:

Nipun — composer needs sentence offsets and token overlap. Can you include sentence boundaries in top_chunks? Example chunk schema: {text, sentences:[{s, start, end}], source}.

Scoring & values enforcement (mandatory)

- At end of each day each person submits a 3-line reflection (Humility / Gratitude / Honesty) saved to reflections/<name>_<date>.md. This is required before merging work into uniguru-jugaad/*. Each reflection is 15 pts total and will be combined with technical QA to form daily acceptance.

Quick checklist to start now

1. Everyone confirm access and branch.
2. Nipun spin up Qdrant test call and share sample JSON.
3. Nisarg scaffold composer templates file and push stub.
4. Vijay stand up uniguru_lm FastAPI skeleton.
5. Karthikeya provide TTS stub endpoint.
6. Vedant add test UI button; Rishabh add feedback UI.