Create a staged refactor proposal as a Markdown document with clear stages,

deliverables, acceptance criteria, and "no-backtrack" design rules.

plan = """# micro-lm Refactor — Staged Proposal (No-Backtrack Plan)

This plan turns the M1–M11 work into a clean, domain■agnostic package with ARC + DeFi adapters, three demoable breakpoints, and a hardened dependency on **ngeodesic v0.1.0**. Each stage has **explicit outputs** and **acceptance criteria** so we never backtrack.

Stage 0 — Repo Hygiene & Freeze (1 day)

Goal: Freeze the moving parts so refactor lands on solid ground.

Work

- Freeze milestone scripts under `scripts/milestones/` (read■only).

- Create branch `refactor/m12`.

- Add `.github/workflows/ci.yml` skeleton (lint + unit test placeholder).

- Add `configs/global.yml` with defaults (seeds, logging, fast mode).

Outputs

- `scripts/milestones/` with M1–M11 + `README_MILESTONES.md` (reference■only).

- `configs/global.yml` (seed=42, fast=false, logging=info).

Acceptance

- CI runs and passes `flake8/ruff` + placeholder tests.

- No files outside `scripts/milestones/` are modified by milestones.

Stage 1 — Core Skeleton (domain■agnostic) (0.5–1 day)

Goal: Land the stable core API; this should not change later.

Work

- Create `src/micro_lm/core/`:

- `runner.py` — stable façade: `run_micro(domain, prompt, *, context, policy, rails, T, backend="sbert")`.

- `mapper_api.py` — thresholding, sweeps, ensemble policy.

- `backends/` — plug■ins: `wordmap.py` (Tier■0), `sbert.py` (Tier■1), stubs for `tfidf.py`, `openai_embed.py`.

- `rails_shim.py` — thin calls into **ngeodesic v0.1.0** (Stage■10/11).

- `bench_io.py`, `metrics.py` — consistent artifacts: `summary.json`, `report.md`, `metrics.csv`.

- Wire `src/micro_lm/cli.py`: `micro-lm `.

Outputs

- Public API: `from micro_lm.core.runner import run_micro`.

- Backends directory with Tier■0 + Tier■1.

- Rails shim importing `ngeodesic>=0.1.0,<0.2`. (Pin exact during release.)

Acceptance

- `run_micro(...)` signature frozen and covered by unit tests.

- Importing `micro_lm` does not import domain code (keeps adapters optional).

Stage 2 — Domain Adapters (DeFi & ARC) (1–2 days)

Goal: Lift proven logic into adapters with zero reinvention.

Work

- `src/micro_lm/domains/defi/`:

- `labels.py`, `mapper.py`, `verify.py` (LTV/HF/oracle/abstain), `benches/`.

- Port M6/M9/M10/M11 logic into `verify.py` & e2e bench.

- `src/micro_lm/domains/arc/`:

- `labels.py`, `mapper.py`, `verify.py` (basic sanity), `benches/`.

- Use `ngeodesic` latent ARC suites via `rails_shim`.

Outputs

- Symmetric folders for `defi/` and `arc/` with benches:

- `mapper_bench.py` (prompt→primitive)

- `rails_bench.py` (primitive→latent via `ngeodesic`)

- `e2e_bench.py` (full pipeline)

Acceptance

- Unit tests green for: `test_defi_verify.py`, `test_arc_verify.py`.

- Adapters import without side effects; `pip install -e .` works.

Stage 3 — Fixtures & Bench Data (0.5 day)

Goal: Reproducible demos at the three breakpoints.

Work

- `tests/fixtures/defi/`: 5k M8 corpus + labels; `m8_per_class_thresholds.json`.

- `tests/fixtures/arc/`: ~30 prompts + labeled CSV; `arc_mapper.joblib` (stub).

- Add small latent suite (or auto■fetch) for ARC rails bench: `latent_arc_20.npz`.

Outputs

- Fixture files checked in (tiny or auto■download).

Acceptance

- Local runs succeed without network (unless auto■fetch flag is used).

Stage 4 — Three CLIs (Mapper / Rails / E2E) (0.5 day)

Goal: One■liners for stakeholder demos.

Work

- `micro-lm defi mapper-bench --backend sbert --thresholds "0.3,0.35,0.4" ...`

- `micro-lm defi rails-bench --suite latent_arc_100 --rails stage11 ...`

- `micro-lm defi e2e-bench --rails stage11 --runs 5 --perturb ...` (same for `arc`).

Outputs

- CLI help texts with examples.

- Artifacts into `.artifacts///` subfolders.

Acceptance

- Each command emits `summary.json` + `report.md`.

- Exit code non■zero on failed expectations.

Stage 5 — Tests & CI (1 day)

Goal: Lock behavior; prevent regressions.

Work

- `tests/unit/`:

- `test_core_mapper_api.py` — thresholding, abstain, ensemble invariants.

- `test_core_rails_shim.py` — calls `ngeodesic` with fixed seeds; checks schema.

- `test_defi_verify.py`, `test_arc_verify.py` — reason mapping invariants.

- `tests/smoke/`:

- `test_*_mapper_bench_fast.py` — 100 prompts, 1 threshold.

- `test_*_rails_bench_fast.py` — latent_arc_20, `--fast`.

- `test_*_e2e_bench_fast.py` — 3 prompts per domain, `--runs 1`.

- CI matrix: py310/py311; upload `.artifacts/`.

Outputs

- Green CI with artifacts per job.

Acceptance

- CI gate passes; artifacts attached to run.

Stage 6 — Docs (0.5 day)

Goal: Stakeholder clarity + self■serve usage.

Work

- `README.md` — short quickstart for DeFi & ARC.
- `docs/ARCHITECTURE.md` — core vs domains vs ngeodesic.
- `docs/BENCHMARKS.md` — explains 3 breakpoints + artifact schema.
- `docs/API.md` — tiny public API reference.
- Add the already generated docs:
- `docs/overview.md` (directory structure)
- `docs/NGF_micro_lm_positioning_note.md` (NGF vs micro■lm split)

Outputs

- Linked docs from README.

Acceptance

- `README` examples run unmodified on a clean machine.

Stage 7 — Tier■0 + Matcher Swappability (0.5 day)

Goal: Prove plug■in backends without touching public API.

Work

- Finalize `core/backends/wordmap.py` and `domains/*/vocab.yml`.
- Add `backend` flag + config overrides.
- Optional `ensemble:wordmap+sbert` policy.

Outputs

- Wordmap backend & ensemble path.

Acceptance

- Mapper bench passes with both `wordmap` and `sbert` (fast slice).
- E2E bench yields identical **verify** reasons regardless of backend choice.

Stage 8 — Release (0.5 day)

Goal: Tag, pin, publish.

Work

- Pin `ngeodesic==0.1.0` (or the exact tested tag).
- `CHANGELOG.md` for `v0.1.0-alpha`.
- GitHub Release with `.artifacts` bundles from smoke benches.

Outputs

- Tag `v0.1.0-alpha` (micro-lm).

Acceptance

- Fresh clone + `pip install .` → all examples succeed.
- Stakeholder one■liners run and produce reports.

No■Backtrack Rules (Design Invariants)

- **Public API freeze:** `run_micro(...)` signature does not change after Stage 1.
- **Rails silo:** all Stage■10/11 code stays in **ngeodesic**; `rails_shim` is the only caller.
- **Bench artifact schema:** `summary.json`, `report.md`, `metrics.csv` structures frozen at Stage 4.
- **Determinism:** edge prompts must produce identical `verify.reason` across N≥3 runs.
- **Safety first:** ambiguous prompts → abstain, never guess (enforced in `mapper_api`).
- **Config over code:** backends, thresholds, and rails options chosen via YAML/CLI, not edits.

Quick Milestone Checklist

- ■ Stage 0: Freeze & CI skeleton
- ■ Stage 1: Core skeleton + API freeze
- ■ Stage 2: DeFi & ARC adapters
- ■ Stage 3: Fixtures land
- ■ Stage 4: CLIs emit artifacts
- ■ Stage 5: CI green with smoke benches
- ■ Stage 6: Docs complete
- ■ Stage 7: Tier■0/Ensemble proven
- ■ Stage 8: Tagged release (alpha)

Appendix — Example Commands (copy■paste)

Mapper (M8■style):

```bash
micro-lm defi mapper-bench \
--prompts_jsonl tests/fixtures/defi/defi_mapper_5k_prompts.jsonl \
--labels_csv tests/fixtures/defi/defi_mapper_labeled_5k.csv \
--backend sbert \
--thresholds "0.3,0.35,0.4" \
--out_dir .artifacts/defi/mapper_bench
```