# Project: **gemcli Playbooks & Domain Profiles**

## 1) Executive Summary

This initiative equips **gemcli** with a config‑driven “brain” so the app can turn high‑level user goals into **concrete, auditable plans** and then execute them with the right tools—without constant code changes. The approach is built around:

* **Domain Profiles** (JSON files under ~/.gencli/<domain>/\*.json) that set defaults, tool policies, retrieval scopes, evidence rules, and prompts.
* **Playbooks** (JSON recipes) that describe the *sequence of activities* for common outcomes (e.g., “generate a CBSE Class 10 science paper” or “run an NSE swing scan”).
* A light **Planner → Executor → Feedback** loop that maps user *Goals* into a typed task graph (<task, input, output>) with human or automated feedback.

**Why it matters**: predictable results, smarter tool use, lower ops cost, and easy extensibility to new domains (education, coding, markets today; tourism/film tomorrow) by editing JSON files—*not* Python.

## 2) Vision & Outcomes

* **Goal**: Convert a high‑level *Goal* into a sequence of tasks with explicit <goal, premise, plan, execute, feedback> stages—recursively if needed—until success criteria are met.
* **Outcome**: Intelligent, auditable outputs (page‑level citations for PDFs; primary‑source web quotes for markets; code/file‑level references for coding).
* **No hard‑coding**: Tool priorities, retrieval scopes, and evidence policy live in profile/playbook JSON, hot‑reloaded at runtime.

## 3) Core Concepts

### 3.1 Domain Profiles (per domain)

**Location**: ~/.gencli/<domain>/\*.json (examples: education/cbse\_board\_class10.json, finance/nse\_stock\_market.json).

Contain: - version, domain, defaults (language/country/timeframe, evidence policy) - tool\_policy (mode allowlists, priority weights, preferred sources) - retrieval (folder/path scopes, filename boosts, top‑k, optional reranker) - prompts.synthesis\_hint (≤150 tokens): tiny hint we inject to the LLM; **the full JSON is never sent**.

### 3.2 Playbooks (per task)

**Location**: ~/.gencli/<domain>/<task>.json (examples: education/generate\_question\_paper.json, finance/news\_catalyst\_scan.json).

Contain: - name, description, slots (variables), guards (mode allow), evidence\_policy (or inherit from profile) - Ordered steps: each has a tool, args/args\_from\_slots, and an id for wiring outputs to inputs - output.presenter: how to render the result

### 3.3 Planner → Executor → Feedback

* **Planner**: Given a *Goal* + selected Profile + (optional) Playbook, produces a **Runtime Plan** (typed task graph) and **clarifying questions**.
* **Executor**: Runs steps; chooses tools using policy **weights + dynamic signals** (recency, RAG availability, attachments). Your existing **Run progress** panel shows each event.
* **Feedback**: Auto checks (evidence coverage, non‑empty outputs) and human feedback. If failed, planner revises the last K steps (bounded recursion) and resumes.

## 4) Detailed Examples

### Example A — Deploy a simple website (login‑protected) for gemcli

**Goal**: “Create a simple website so users can log in and access gemcli.”

**Recommended path (cost‑benefit)**: Use a **managed front‑door** rather than building auth from scratch.

**Two supported variants**: 1) **Front‑door proxy (fastest)** - Reverse proxy (Nginx/Caddy) or provider access gate in front of Streamlit. - Basic auth/allowlist; HTTPS/TLS; rate limiting; logs. - Best for a small, trusted user list.

1. **Managed auth + thin web**
   * Keep gemcli backend; add a tiny web shell (Next.js/Flask) with **managed auth** (Supabase, Firebase, Clerk, Auth0).
   * Proper sessions/MFA/passwordless, email, roles; deploy to Vercel/Render/Fly; connect via REST/WebSocket to gemcli.

**Playbook sketch (front‑door proxy)**: - **Premise**: domain/subdomain available; Streamlit/gemcli reachable on an internal port; TLS provider or certs ready. - **Plan** (<task, in, out>): 1) clarify\_inputs → ask for domain, user list, auth type. 2) provision\_dns\_tls → verify DNS & certificate issued. 3) configure\_proxy → route https://app.example.com → local service. 4) enable\_auth\_gate → basic auth or provider access policy. 5) harden\_security → rate limit, headers, health checks. 6) smoke\_test → login ok, 401 on bad creds, app reachable. - **Execute**: Steps run with confirmations; secrets entered by you at prompt time. - **Feedback**: If smoke tests fail, auto‑retry with logs; else output a deployment checklist & rollback recipe.

**Playbook sketch (managed auth + thin web)**: - **Premise**: Choose auth provider; target region; expected users; attachment limits. - **Plan**: 1) clarify\_inputs → provider, domain, roles. 2) scaffold\_web → minimal Next.js/Flask UI + login/profile. 3) secure\_backend → JWT verify middleware; rate limits; CORS; .env wiring to gemcli. 4) deploy\_web → Vercel/Render + TLS. 5) smoke\_test → multi‑user session; attachment upload. - **Execute/Feedback**: As above; outputs include URLs, env keys, and a runbook.

**Risks & mitigations**: - Security pitfalls → choose A/B; never hand‑roll crypto. Keep secrets in env; restrict tools in Direct Chat. - Ops drift → playbook prints a runbook, creates health checks and basic alerts.

### Example B — Stock trader workflows (swing trading & portfolio management)

**Goal**: “Scan for India swing setups and manage my portfolio.”

**Premise**: Finance profile selected (~/.gencli/finance/nse\_stock\_market.json), web tools enabled, timeframe and sources set (NSE/BSE/SEBI, company IR). Portfolio CSVs can be attached.

**Playbook 1: nse\_swing\_scan** - **Plan**: 1) web\_search (discovery): catalysts <45 days (results, order wins, approvals, pledges, mgmt changes, policy). Output 8–12 tickers with reasons & sources. 2) web\_fetch (evidence): for top 6, fetch **primary sources** (NSE/BSE filings, IR press releases, presentations); quote short excerpts with dates. 3) price\_history (technicals): compute 20/50 EMA, distance to 52‑week high, ATR(14), volume vs 20‑day avg; RS vs NIFTY 50 & sector. 4) synthesize\_trades: propose 3–5 swing trades with entries/ATR stops/targets, 2–6 weeks horizon, 1% risk per trade; “invalidate if…” clause. - **Evidence policy**: Primary sources preferred; table of top evidence per ticker; mark secondary/blog sources clearly. - **Output**: Overview bullets → Candidates table → Trade proposals → Evidence tables.

**Playbook 2: portfolio\_review** - **Premise**: User uploads holdings CSVs; RAG‑OFF is fine. - **Plan**: 1) load\_portfolio (attachments): parse weights, sectors, betas. 2) market\_context (web): last 30–45d macro/sector notes (RBI, CPI, GST/PLI, policy moves). 3) risk\_checks: concentration (top names & sectors), drawdown vs thresholds, liquidity flags. 4) rebalance\_suggestions: suggest trims/adds (number & %) given constraints (turnover, taxes optional, min lot sizes). 5) action\_list: orders/alerts with rationale & links. - **Evidence policy**: web primary where applicable; otherwise calculations are shown inline (tables). - **Output**: Portfolio snapshot → Risks → Suggested actions → Evidence links.

**Benefits**: repeatability, auditability, and fewer wasted web calls (the plan is tight and cites sources).

## 5) High‑Level Work & Estimated Timeline

Assumes current gemcli codebase with tools, RAG, progress panel, streaming policy; 1–2 engineers; low‑risk increments.

**Phase 0 — Design & Schemas (0.5–1 day)** - Define **Profile** and **Playbook** JSON schemas (reserved keys, validation rules). - Decide initial set: education/cbse\_board\_class10.json, education/generate\_question\_paper.json, finance/nse\_stock\_market.json, finance/nse\_swing\_scan.json.

**Phase 1 — Config Loader & Registry (0.5 day)** - JSONC parser (strip //), schema validation, hot‑reload on file changes. - Merge order: base profile → task/playbook → runtime slots.

**Phase 2 — Planner (1–2 days)** - LLM prompt to produce a **Runtime Plan** (typed task graph) + clarifications. - Plan validator: enforce mode allowlists, tool existence, I/O wiring. - Clarification UI: list questions → collect answers → re‑plan.

**Phase 3 — Executor Bridge (0.5–1 day)** - Map Runtime Plan steps to your existing execute\_plan and progress events. - Auto checks (evidence coverage %, non‑empty outputs); bounded re‑plan.

**Phase 4 — Profiles & Playbooks Authoring (1–2 days)** - Author and test the four starters (education & finance pairs). - Add retrieval scopes and evidence rules; tune top‑k/reranker.

**Phase 5 — UX Polish (0.5 day)** - Profile/Playbook selector; active banner; results presenters (paper, trades, tables).

**Total**: **~3.5 to 6 days** elapsed, incremental and shippable per phase.

**Stretch goals** (later): multi‑tenant profiles; rate limits per user; domain‑specific presenters; optional “final‑answer streaming” in LLM Tools/Agent.

## 6) Prerequisites

* **Operational**: domain/DNS (for website), TLS certificates or provider; basic infra (Docker or hosting).
* **Accounts/Keys**: LLM provider, vector DB (if used), web data sources (NSE/BSE allowlisted domains), optional auth provider (Supabase/Firebase/Clerk/Auth0) for managed auth path.
* **Data**: organized project folders for education PDFs; optional holdings CSVs for portfolio.
* **App**: current gemcli with tools enabled, Run progress UI, streaming policy (Direct Chat only), and RAG working.

## 7) Risks & Mitigations

* **Security pitfalls** (auth, secrets) → prefer front‑door/managed auth; store secrets in env; enforce mode tool allowlists.
* **Config sprawl** → JSON schemas, validate command, and a Profiles/Playbooks registry page.
* **Plan quality** → plan validator, clarifying questions, bounded re‑plan, evidence coverage checks.
* **Web dependence** → prefer primary sources; timeouts and retries in web\_fetch; cache results per run.

## 8) Success Criteria & KPIs

* **Adoption**: # of runs using Profiles/Playbooks; time‑to‑first‑use < 5 minutes.
* **Quality**: ≥90% runs include required evidence (PDF page cites or primary links).
* **Efficiency**: ≤2 re‑plans per run on average; fewer 0‑hit retrievals.
* **Maintainability**: New domain or task delivered by adding JSON only (no Python edits) ≥80% of the time.

## 9) Next Steps

1. Approve the schemas (Profiles + Playbooks) and the four starter files (education & finance).
2. Implement loader/validator (Phase 1) and wire the Planner to emit Runtime Plans (Phase 2).
3. Ship the first two playbooks; measure KPIs; iterate on weights/scopes.

This document is intended for both **managers** (scope, timeline, risks) and **architects** (schemas, flow, integration points). It can serve as the working brief for implementation.