Linear Trend Spotter — Plan of Action

- **Project:** `linear-trend-spotter`
- **Repository:** edwardlthompson/linear-trend-spotter
- **Version Target:** 1.0.0
- **Date:** 2026-02-28
- **Status:** DRAFT
- **Audience:** AI-first, Human-second

Overview

This plan defines a dependency-ordered build sequence to bring the `linear-trend-spotter` codebase into full compliance with the Technical Specification (dated 2026-02-28, v1.0.0). The project is **not** a greenfield build — a functional codebase already exists and is running in production on PythonAnywhere. The work defined here is a structured **alignment pass** that reconciles the live implementation with the newly-authored authoritative specification.

The entire alignment effort is structured into **three milestones**, each containing **two sprints** (six sprints total). Each sprint is sized to fit comfortably within the standard context window of Claude Opus 4.6 (~200K tokens) while maximizing the amount of work accomplished per sprint.

Guiding Constraints

Constraint	Strategy
Production system is live	Changes are non-destructive. No existing functionality is removed until its replacement is verified. Database schema changes are additive (new tables/columns), not destructive (DROP TABLE).
Context window fit	Each sprint references only the spec sections it needs. The full spec is ~1,200 lines; no single sprint requires it all in context.
Minimal milestone count	Three milestones. Work is consolidated aggressively — the only reason to split work across milestones is a hard dependency.
Dependency ordering	Milestone 1 normalizes the foundation (naming, structure, config). Milestone 2 builds on the normalized foundation to align the pipeline and data layer. Milestone 3 adds the notification polish, bot commands, and operational infrastructure.
Sprint sequencing	Within each milestone, sprints are chronologically ordered. Each sprint's deliverables may be consumed by subsequent sprints.
Spec is authoritative	Where the codebase and the spec disagree, the spec is presumed correct. Deviations from this rule are called out explicitly.

Milestone Dependency Chain

```
Milestone 1: Repository Cleanup + Configuration Alignment + Naming Normalization
     │  Sprint 1.1 → Sprint 1.2
     │
     │  provides: Normalized file/module names, aligned config/settings.py,
     │            aligned config/constants.py, cleaned archive, aligned
.env.example,
     │            aligned config_json.example, aligned database file names
     ▼
Milestone 2: Pipeline Alignment + Database Schema + API Client Normalization
     │  Sprint 2.1 → Sprint 2.2
     │
     │  provides: Aligned api/ module names and classes, aligned database/models.py
     │            and database/cache.py, aligned processors/, aligned main.py
     │            pipeline stages, aligned exchange_data/ module
     ▼
Milestone 3: Notifications + Bot Commands + Operational Infrastructure
     │  Sprint 3.1 → Sprint 3.2
     │
     │  provides: Aligned notifications/formatter.py, aligned telegram_bot.py
     │            commands, aligned utils/metrics.py, aligned utils/logger.py,
     │            aligned scheduling scripts, documentation updates
     ▼
   v1.0.0 Spec Compliance Complete
```

Current State Assessment

The following assessment is based on a review of the live codebase files accessible through the project knowledge base, compared against the Technical Specification.

What Already Exists and Broadly Aligns

Component	Status	Notes
`main.py`	☑ Functional	Pipeline orchestrator exists with `run_scanner()`. Stage ordering broadly matches spec.
`scheduler.py`	☑ Functional	File locking via `fcntl.flock`, PID in lock file, stats recording. Closely matches §11.
`telegram_bot.py`	☑ Functional	Long-polling, `/status`, `/list`, `/help` commands present. Has extra `/start` not in spec.
`manage_bot.py`	☑ Functional	Process lifecycle management.
`bot_watchdog.py`	☑ Functional	Cron-driven, checks PID, restarts if down. Matches §11.4.
`config/settings.py`	⚠ Partial	`Settings` class exists but defaults are incomplete vs. spec §9.3.

Component	Status	Notes
`config/constants.py`	⚠ Presumed exists	Exported from `config/__init__.py` as `STABLECOINS`, `EXCHANGE_EMOJIS`, `COIN_MAPPING`.
`api/coinmarketcap.py`	☑ Functional	`CoinMarketCapClient` with `get_all_coins_with_gains()`, `extract_gains()`, `extract_coin_data()`.
`api/coingecko_mapper.py`	☑ Functional	`CoinGeckoMapper` class with `update_mappings()` and `get_stats()`.
`api/chart_img.py`	☑ Functional	`ChartIMGClient` initialized with API key and TradingView mapper.
`api/tradingview_mapper.py`	☑ Functional	`TradingViewMapper` with dedicated `tv_mappings.db`.
`exchange_data/`	☑ Functional	`ExchangeDatabase` and `ExchangeFetcher` exported.
`utils/rate_limiter.py`	⚠ Partial	`RateLimiter` and `CircuitBreaker` exist but lack 429-specific backoff escalation, jitter, and 300s cap per spec §7.5.
`utils/logger.py`	☑ Functional	`setup_logger()` and `app_logger` exported.
`utils/metrics.py`	⚠ Partial	`MetricsCollector` exported but completeness vs. spec §12.3 is uncertain.
`notifications/telegram.py`	☑ Functional	`TelegramClient` present.
`notifications/formatter.py`	⚠ Unknown	Spec requires `MessageFormatter`; verify existence and completeness.
`processors/gain_filter.py`	⚠ Unknown	Spec requires `GainFilter` class; verify.
`processors/uniformity_filter.py`	⚠ Unknown	Spec requires `UniformityFilter` with `calculate()` per §6.2; verify.
`database/models.py`	⚠ Partial	`HistoryDatabase` and `ActiveCoinsDatabase` exist but use `history.db` not `scanner.db`.
`database/cache.py`	⚠ Misaligned	Exports `GeckoCache`/`CoinLoreCache`, not `PriceCache` per spec.

Identified Deviations (Spec vs. Codebase)

These deviations are the primary work items for this plan. Each is tagged with an ID for traceability.

ID	Category	Spec Says	Code Has	Severity
DEV-01	File naming	`api/coingecko.py` exports `CoinGeckoClient`	`api/coingecko_optimized.py` exports `CoinGeckoOptimizedClient`	Medium
DEV-02	Database naming	Primary DB is `scanner.db`	Primary DB is `history.db`	Medium
DEV-03	Database naming	Exchange DB is `exchanges.db`	Exchange DB is `exchange_listings.db`	Low
DEV-04	Database naming	Mapping DB is `mappings.db`	Mapping DB is `coingecko_mappings.db`	Low
DEV-05	Cache class	`database/cache.py` exports `PriceCache`	Exports `GeckoCache` / `CoinLoreCache`	Medium
DEV-06	CoinLore removal	CoinLore is not referenced anywhere in spec	Old spec references CoinLore; code may still use it	Medium
DEV-07	Config defaults	§9.3 lists 17 tunable parameters	Code defaults include `SENSITIVITY` (not in spec) and are missing `TOP_COINS_LIMIT`, `ENTRY_NOTIFICATIONS`, `EXIT_NOTIFICATIONS`, `COINGECKO_CALLS_PER_MINUTE`, `CMC_CALLS_PER_MINUTE`, `CACHE_GECKO_ID_DAYS`, `CACHE_EXCHANGE_HOURS`, `CACHE_PRICE_HOURS`, `CIRCUIT_FAILURE_THRESHOLD`, `CIRCUIT_RECOVERY_TIMEOUT`	High
DEV-08	Archive hygiene	`build_mapping_db.py` should be in `.archive/`	Still at repo root	Low
DEV-09	Archive hygiene	`docs/linear-trend-spotter-spec.txt` superseded by `linear-trend-spotter-spec.md`	Still in `docs/`	Low
DEV-10	Repo structure	Spec §3.1 does not include a `docs/` directory	`docs/` exists with `3rd-party-map.json` and old spec	Low

ID	Category	Spec Says	Code Has	Severity
DEV-11	Rate limiter	§7.5 requires 429 backoff escalation (60→120→240, cap 300s), jitter (0–100ms)	`RateLimiter` has basic interval enforcement but no 429-specific escalation or jitter	High
DEV-12	DB schema	§8.1 `active_coins` has `slug` and `cmc_url` columns	Verify these columns exist	Medium
DEV-13	DB schema	§8.1 `price_cache` table in `scanner.db`	Cache may be in separate DB or using different schema	Medium
DEV-14	Top-level files	Spec requires `3rd-party-map.json` at repo root	File is in `docs/3rd-party-map.json`	Low
DEV-15	Top-level files	Spec requires `config_json.example` at repo root	Verify existence and completeness	Medium
DEV-16	Top-level files	Spec requires `.env.example` at repo root	Verify existence and completeness	Medium
DEV-17	CMC limit	§5.2 says 5,000 coins	Old spec said 2,500; code uses `limit=5000` in `main.py` (aligned)	None — Already aligned
DEV-18	Python version	§2.4 says Python 3.10	Old spec said Python 3.13	Info — verify runtime
DEV-19	Notification format	§10.1 and §10.2 define exact caption/message formats	Verify `formatter.py` / inline formatting matches	Medium
DEV-20	Metrics collector	§12.3 defines specific counters and `metrics.json` output	Verify `MetricsCollector` tracks all required counters	Medium
DEV-21	`README.md`	§3.1 says "Public project overview and Telegram invite link"	Verify content	Low

Repository Baseline

The following files and directories already exist in the repository. Implementing agents MUST NOT recreate these from scratch — they must be modified in place or renamed as specified.

Pre-Existing Path	Description
`main.py`	Scanner orchestrator. Modify, do not replace.
`scheduler.py`	Cron entry point. Aligned; minor changes only.
`telegram_bot.py`	Bot handler. Modify for command alignment.
`manage_bot.py`	Bot process manager. Aligned.
`bot_watchdog.py`	Watchdog. Aligned.
`config/settings.py`	Settings class. Modify to align defaults.
`config/constants.py`	Static lookup tables. Verify and align.
`api/coinmarketcap.py`	CMC client. Aligned.
`api/coingecko_optimized.py`	CoinGecko client. Rename to `api/coingecko.py`, rename class.
`api/coingecko_mapper.py`	Mapper. Aligned.
`api/chart_img.py`	Chart-IMG client. Aligned.
`api/tradingview_mapper.py`	TradingView mapper. Aligned.
`exchange_data/exchange_db.py`	Exchange DB. Aligned.
`exchange_data/exchange_fetcher.py`	Exchange fetcher. Aligned.
`notifications/telegram.py`	Telegram client. Aligned.
`notifications/formatter.py`	Message formatter. Verify and align.
`processors/gain_filter.py`	Gain filter. Verify and align.
`processors/uniformity_filter.py`	Uniformity filter. Verify algorithm matches §6.2.
`database/models.py`	DB models. Modify for schema and naming alignment.
`database/cache.py`	Cache layer. Rename class to `PriceCache`, align schema.
`utils/rate_limiter.py`	Rate limiter + circuit breaker. Enhance per §7.5.
`utils/logger.py`	Logging setup. Verify and align.
`utils/metrics.py`	Metrics collector. Verify and align per §12.3.
`update_exchanges.py`	Exchange refresh script. Verify.
`update_mappings.py`	Mapping refresh script. Verify.
`.gitignore`	Already present. Verify covers all runtime files per §3.4.
`.github/copilot-instructions.md`	AI agent directives. Already present.
`linear-trend-spotter-spec.md`	This plan's authoritative source.

Milestone 1 — Repository Cleanup, Configuration Alignment, and Naming Normalization

Goal: Normalize the repository structure, file names, configuration system, and package exports to match the spec. At the end of this milestone, every file is in the correct location with the correct name, all `__init__.py` exports match the spec's key exports table (§3.2), all configuration defaults match §9.3, and archive hygiene is complete.

No behavioral changes to the running pipeline. This milestone is purely structural and nominal.

Sprint 1.1 — Archive Hygiene, Repo Structure, and Top-Level File Alignment

Goal: Move superseded files to `.archive/` with proper naming, relocate misplaced files to their spec-defined locations, verify or create all required top-level template files, and update `.gitignore` to cover all runtime data files per §3.4.

Spec References

The implementing agent for Sprint 1.1 MUST have the following in its context window:

Source	Sections / Files
§3.1 — Top-Level Layout	Complete file tree and purpose table
§3.3 — Archived Artifacts	Archive naming convention, daily increment rules
§3.4 — Runtime Data Files	Files that MUST be gitignored
§9.2 — Secrets Management	`.env.example` template content
§9.3 — Tunable Parameters	All config keys and defaults for `config_json.example`
§9.4 — Configuration File Format	`config.json` structure
§1.5 — Reference Documents	`3rd-party-map.json` location
Current `.gitignore`	Existing content

Deliverables

Action	File	Description
Archive	`build_mapping_db.py` → `.archive/20260228-001-build_mapping_db.py`	Superseded by `api/coingecko_mapper.py` + `update_mappings.py`. **DEV-08.**

Action	File	Description
Archive	`docs/linear-trend-spotter-spec.txt` → `.archive/20260228-001-linear-trend-spotter-spec.txt`	Superseded by `linear-trend-spotter-spec.md`. Same archive group as above. **DEV-09.**
Relocate	`docs/3rd-party-map.json` → `3rd-party-map.json` (repo root)	Spec §3.1 places it at root. Remove `COINAPI_API_KEY` entry (not in spec). **DEV-14.**
Remove	`docs/` directory	After relocating contents, remove the empty `docs/` directory if no other files remain. **DEV-10.**
Create/Verify	`.env.example`	Must contain all 5 variables per §9.2 (4 required + 1 optional).
Create/Verify	`config_json.example`	Must contain all keys per §9.4 with spec defaults.
Update	`.gitignore`	Must cover: `scanner.db`, `exchanges.db`, `mappings.db`, `tv_mappings.db`, `scan.lock`, `scan_stats.json`, `metrics.json`, `trend_scanner.log`, `bot_output.log`, `*.pid`, `config.json`, `.env`. Also retain existing ignores for `__pycache__/`, `*.pyc`, `.idea/`, etc.
Verify	`README.md`	Must exist with at minimum a project overview and Telegram invite link per §3.1. Update if stub.
Verify	`requirements.txt`	Must list `requests` and `python-dotenv` per §15.1.

Steps

1. Identify all files that belong in `.archive/` per the assessment above. Rename using the `YYYYMMDD-NNN-FileName.Extension` convention (§3.3). Use `20260228` as the date stamp and `001` as the shared increment for this batch.
2. Relocate `docs/3rd-party-map.json` to repo root. Remove the `COINAPI_API_KEY` entry from the JSON (CoinAPI is not part of the spec). Verify remaining entries match spec services.
3. Remove the `docs/` directory if it is now empty.
4. Create or verify `.env.example` with exact content per §9.2.
5. Create or verify `config_json.example` with all 17 keys per §9.3–9.4.
6. Update `.gitignore` to include all runtime data files per §3.4. Ensure both old names (`history.db`, `exchange_listings.db`, `coingecko_mappings.db`, `mapping.db`) and new names (`scanner.db`, `exchanges.db`, `mappings.db`) are covered during the transition.
7. Verify `README.md` exists with meaningful content. If it is a stub, add at minimum: project name, one-sentence description, and the Telegram group invite link (`https://t.me/+pmZewVhuEFJjYTIx`).
8. Verify `requirements.txt` lists `requests` and `python-dotenv`.

Exit Criteria

- [ ] `.archive/` contains `20260228-001-build_mapping_db.py` and `20260228-001-linear-trend-spotter-spec.txt`.
- [ ] `3rd-party-map.json` exists at repo root with no `COINAPI_API_KEY` entry.
- [ ] `docs/` directory no longer exists (or is documented as intentionally retained if other files remain).
- [ ] `.env.example` matches §9.2 template exactly.
- [ ] `config_json.example` contains all 17 parameter keys per §9.3.
- [ ] `.gitignore` covers all runtime files per §3.4.
- [ ] `README.md` contains project overview and Telegram invite link.
- [ ] `requirements.txt` lists `requests` and `python-dotenv`.
- [ ] No file outside `.archive/` references `build_mapping_db.py` or `docs/linear-trend-spotter-spec.txt`.

Sprint 1.2 — Configuration Alignment, Module Renaming, and Package Export Normalization

Goal: Align `config/settings.py` defaults with §9.3, rename `api/coingecko_optimized.py` to `api/coingecko.py` and its class to `CoinGeckoClient`, rename the cache class in `database/cache.py` from `GeckoCache`/`CoinLoreCache` to `PriceCache`, update all `__init__.py` exports to match §3.2, and update all import sites throughout the codebase to reference the new names.

Spec References

The implementing agent for Sprint 1.2 MUST have the following in its context window:

Source	Sections / Files
§3.2 — Source Package Layout	Key exports table for all sub-packages
§9.1 — Configuration Architecture	Settings class design
§9.3 — Tunable Parameters	All 17 parameter keys with types and defaults
§7.2 — CoinGecko	Client naming and behavior
§8.1 — Primary Database	Database naming
§14.2 — Caching Strategy	Cache class naming
All `__init__.py` files	Current export lists
`main.py`	Import statements referencing renamed modules
`telegram_bot.py`	Import statements referencing renamed modules
`config/settings.py`	Current defaults and properties
`api/__init__.py`	Current exports
`database/__init__.py`	Current exports

Deliverables

File	Nature of Change

File	Nature of Change
`config/settings.py`	**DEV-07.** Add all missing default config keys from §9.3: `TOP_COINS_LIMIT`, `ENTRY_NOTIFICATIONS`, `EXIT_NOTIFICATIONS`, `COINGECKO_CALLS_PER_MINUTE`, `CMC_CALLS_PER_MINUTE`, `CACHE_GECKO_ID_DAYS`, `CACHE_EXCHANGE_HOURS`, `CACHE_PRICE_HOURS`, `CIRCUIT_FAILURE_THRESHOLD`, `CIRCUIT_RECOVERY_TIMEOUT`. Remove `SENSITIVITY` (not in spec). Add corresponding `@property` accessors. Update `db_paths` to use spec-defined names: `scanner.db` (was `history.db`), `exchanges.db`, `mappings.db`.
`api/coingecko_optimized.py` → `api/coingecko.py`	**DEV-01.** Rename file. Rename class from `CoinGeckoOptimizedClient` to `CoinGeckoClient`.
`api/__init__.py`	Update exports: `CoinGeckoClient` (was `CoinGeckoOptimizedClient`), remove old import.
`database/cache.py`	**DEV-05.** Rename class(es) from `GeckoCache`/`CoinLoreCache` to `PriceCache`. Remove any CoinLore-specific logic. **DEV-06.**
`database/__init__.py`	Update exports to `PriceCache` (was `GeckoCache`/`CoinLoreCache`).
`main.py`	Update all imports: `CoinGeckoClient` (was `CoinGeckoOptimizedClient`), `PriceCache` (was `CoinLoreCache`/`GeckoCache`). Update any database path references if they used old names.
`telegram_bot.py`	Update imports for renamed cache class if used directly.
`config/__init__.py`	Verify exports match: `settings`, `STABLECOINS`, `EXCHANGE_EMOJIS`. Remove `COIN_MAPPING` if it is not referenced in spec.

Steps

1. **Config defaults alignment.** Open `config/settings.py`. In `_get_default_config()`, add all missing keys from §9.3 with their spec-defined default values. Remove `SENSITIVITY`. Ensure `USE_14D_FILTER` remains (it is referenced in §16 as a future consideration flag). Add `@property` accessors for all new keys that don't already have them.
2. **Database path alignment.** Update the `db_paths` property in `Settings` to return: `'history'` → `BASE_DIR / 'scanner.db'`, add `'exchanges'` → `BASE_DIR / 'exchanges.db'`, add `'mappings'` → `BASE_DIR / 'mappings.db'`. Retain backward compatibility: if the old-named files exist on disk but the new-named files don't, the first scan after this change should still work. Consider adding a migration note or alias.
3. **CoinGecko client rename.** Rename `api/coingecko_optimized.py` to `api/coingecko.py`. Inside the file, rename the class from `CoinGeckoOptimizedClient` to `CoinGeckoClient`. Update `api/__init__.py`.
4. **Cache class rename.** In `database/cache.py`, rename `GeckoCache`/`CoinLoreCache` to `PriceCache`. Remove any CoinLore-specific logic or endpoints. Update `database/__init__.py`.

5. **Import cascade.** Search every `.py` file for imports of the old names and update them. Key files: `main.py`, `telegram_bot.py`, all `__init__.py` files.
6. **Verify all `__init__.py` exports** match the §3.2 key exports table.

Exit Criteria

- [ ] `config/settings.py _get_default_config()` returns all 17 keys from §9.3 (plus `USE_14D_FILTER`).
- [ ] `from config.settings import settings; print(settings.top_coins_limit)` returns `5000`.
- [ ] `api/coingecko.py` exists and exports `CoinGeckoClient`.
- [ ] `api/coingecko_optimized.py` no longer exists.
- [ ] `database/cache.py` exports `PriceCache`.
- [ ] No `.py` file in the repo imports `CoinGeckoOptimizedClient`, `GeckoCache`, or `CoinLoreCache`.
- [ ] All `__init__.py` exports match §3.2.

Milestone 2 — Pipeline Alignment, Database Schema, and API Client Enhancement

Goal: Align the 10-stage pipeline implementation in `main.py` with the spec, normalize database schemas to match §8, enhance the rate limiter with 429 escalation and jitter per §7.5, and verify that the gain filter and uniformity filter logic matches §5.5 and §6.2 respectively.

Depends on: Milestone 1 (all naming is normalized, all config defaults are present).

Sprint 2.1 — Database Schema Alignment and Rate Limiter Enhancement

Goal: Align the SQLite schemas for `scanner.db` (§8.1), `exchanges.db` (§8.2), and `mappings.db` (§8.3) with the spec. Enhance `utils/rate_limiter.py` with the full §7.5 behavior: 429-specific backoff escalation (60→120→240s, cap 300s), jitter (0–100ms), and minimum interval enforcement per service.

Spec References

The implementing agent for Sprint 2.1 MUST have the following in its context window:

Source	Sections / Files
§8.1 — Primary Database — `scanner.db`	`active_coins`, `scan_history`, `price_cache` table definitions
§8.2 — Exchange Listings Database — `exchanges.db`	`exchange_listings`, `exchange_metadata`, `listing_cache` table definitions
§8.3 — Mapping Database — `mappings.db`	`symbol_mapping`, `mapping_metadata` table definitions
§7.5 — Rate Limit Strategy	Full rate limiter behavioral spec
§13.1 — API Failures	Retry strategy, circuit breaker behavior

Source	Sections / Files
§13.2 — Database Errors	SQLite locked retry strategy
`database/models.py`	Current `HistoryDatabase`, `ActiveCoinsDatabase` implementations
`database/cache.py`	Current `PriceCache` implementation (renamed in Sprint 1.2)
`exchange_data/exchange_db.py`	Current `ExchangeDatabase` implementation
`api/coingecko_mapper.py`	Current mapper DB schema
`utils/rate_limiter.py`	Current `RateLimiter` and `CircuitBreaker` implementations

Deliverables

File	Nature of Change
`database/models.py`	**DEV-02, DEV-12.** Align `CREATE TABLE` statements with §8.1. Ensure `active_coins` has all spec columns including `slug` and `cmc_url`. Ensure `scan_history` has all spec columns and indexes. Handle migration: use `ALTER TABLE ADD COLUMN` for any missing columns to avoid data loss on existing production DB.
`database/cache.py`	**DEV-13.** Align `price_cache` table schema with §8.1. Ensure `coin_id`, `prices`, `uniformity_score`, `gains_30d`, `cache_date` columns match spec. Implement cache TTL check against `CACHE_PRICE_HOURS` config.
`exchange_data/exchange_db.py`	**DEV-03.** Update file path reference to `exchanges.db`. Align `exchange_listings`, `exchange_metadata`, and `listing_cache` table definitions with §8.2.
`api/coingecko_mapper.py`	**DEV-04.** Update file path reference to `mappings.db`. Align `symbol_mapping` and `mapping_metadata` table definitions with §8.3.
`utils/rate_limiter.py`	**DEV-11.** Enhance `RateLimiter`: add 429-specific backoff escalation (consecutive 429s double wait: 60→120→240s, capped at 300s; counter resets on success). Add jitter (random 0–100ms per call). Add `record_success()` and `record_429()` methods if not already present. Enhance `CircuitBreaker` to use `CIRCUIT_FAILURE_THRESHOLD` and `CIRCUIT_RECOVERY_TIMEOUT` from config.

Steps

1. **Audit current schemas.** Read `database/models.py`, `database/cache.py`, `exchange_data/exchange_db.py`, and `api/coingecko_mapper.py`. Document the current `CREATE`

   `TABLE` statements.
2. **Diff against spec.** Compare each table definition against §8.1–8.3. List missing columns, missing indexes, and naming differences.
3. **Implement schema alignment in `database/models.py`.** For `active_coins`: add any missing columns from §8.1 (`slug`, `cmc_url`, `gecko_id`, etc.) using `ALTER TABLE ADD COLUMN` wrapped in try/except (column may already exist). For `scan_history`: add missing columns and indexes. Do NOT use `DROP TABLE` — the production DB has live data.
4. **Implement schema alignment in `database/cache.py`.** Align `price_cache` table with §8.1. Implement the 6-hour TTL check using `CACHE_PRICE_HOURS` from settings.
5. **Implement schema alignment in `exchange_data/exchange_db.py`.** Align table definitions with §8.2.
6. **Implement schema alignment in `api/coingecko_mapper.py`.** Align table definitions with §8.3.
7. **Enhance `utils/rate_limiter.py`.** Add the 429 escalation logic to `RateLimiter`. Add the jitter calculation (`random.uniform(0, 0.1)` added to each wait). Update `CircuitBreaker.__init__` to accept `failure_threshold` and `recovery_timeout` from config rather than hardcoded values.
8. **Verify all database file path references** use the spec-defined names from the updated `Settings.db_paths`.

Exit Criteria

- [ ] `database/models.py` creates `active_coins` with all 14 columns from §8.1.
- [ ] `database/models.py` creates `scan_history` with all columns and both indexes from §8.1.
- [ ] `database/cache.py` creates `price_cache` with all 5 columns from §8.1.
- [ ] `exchange_data/exchange_db.py` creates all 3 tables from §8.2.
- [ ] `api/coingecko_mapper.py` creates both tables from §8.3.
- [ ] `RateLimiter` implements 429 escalation (60→120→240→300s cap) with jitter.
- [ ] `CircuitBreaker` reads thresholds from config.
- [ ] No `ALTER TABLE` statement fails on an existing production database.

Sprint 2.2 — Pipeline Stage Alignment and Processor Verification

Goal: Align `main.py` pipeline stages with §5.2–5.11. Verify `processors/gain_filter.py` matches §5.5 (gain thresholds: 7d >7%, 30d >30%; stablecoin exclusion). Verify `processors/uniformity_filter.py` implements the exact algorithm in §6.2. Verify `main.py` data flow enrichment matches §4.3.

Spec References

The implementing agent for Sprint 2.2 MUST have the following in its context window:

Source	Sections / Files
§4.3 — Data Flow	Progressive dict enrichment scheme
§5.1 — Pipeline Overview	All 10 stages table
§5.2 — Stage 1 through §5.11 — Stage 10	Detailed stage specs
§6.1 — Purpose through §6.3 — Score Interpretation	Full uniformity algorithm
§5.5 — Stage 4 — Gain Filter	Gain thresholds and stablecoin list

Source	Sections / Files
`main.py`	Current pipeline implementation
`processors/gain_filter.py`	Current gain filter implementation
`processors/uniformity_filter.py`	Current uniformity filter implementation
`config/constants.py`	Current stablecoin list

Deliverables

File	Nature of Change
`processors/gain_filter.py`	Verify and align. Must implement `GainFilter` class (or function) with: 7d >7%, 30d >30% thresholds. Must exclude stablecoins from `config/constants.py`. If thresholds are hardcoded, that is acceptable per spec ("currently hardcoded in `GainFilter`" — §16).
`processors/uniformity_filter.py`	Verify and align. Must implement `UniformityFilter` with `calculate()` method matching the exact 5-step algorithm in §6.2: normalize to cumulative %, compute ideal line, compute total deviation, compute max deviation, apply `100 × (1 - √(min(normalized, 1)))`. Return `(uniformity_score, total_gain_pct)`. Score 0 if `total_gain ≤ 0`.
`config/constants.py`	Verify `STABLECOINS` list includes at minimum: USDT, USDC, DAI, BUSD, TUSD, USDP, GUSD per §5.5.
`main.py`	Align pipeline stage ordering and data flow with §5.2–5.11. Ensure each stage enriches the coin dict as described in §4.3. Ensure Stage 9 (Entry/Exit) performs the set-difference described in §5.10. Ensure Stage 10 uses TradingView mapper with MEXC → Kraken → Coinbase priority per §5.11.

Steps

1. **Read `processors/gain_filter.py` in full.** Compare against §5.5. Verify: both 7d >7% and 30d >30% thresholds are checked. Verify stablecoins are excluded. Fix any discrepancy.
2. **Read `processors/uniformity_filter.py` in full.** Compare against §6.2. Verify the 5-step algorithm: cumulative percentage normalization, ideal line calculation, total deviation, max deviation, `100 × (1 - √(min(normalized, 1)))` transformation. Verify score is 0 when `total_gain ≤ 0`. Fix any discrepancy.
3. **Read `config/constants.py` `STABLECOINS`.** Verify it includes all coins listed in §5.5. Add any missing entries.
4. **Read `main.py` `run_scanner()` in full.** Map each code block to the 10 pipeline stages. Verify stage ordering matches §5.1. Verify data enrichment matches §4.3. Verify entry/exit logic matches §5.10. Verify notification delivery matches §5.11 (chart priority: MEXC → Kraken → Coinbase). Document and fix any deviations.

5. **Verify `main.py` uses `ENTRY_NOTIFICATIONS` and `EXIT_NOTIFICATIONS` config flags** (added in Sprint 1.2) to control whether notifications are sent.

Exit Criteria

- ☐ `processors/gain_filter.py` enforces both 7d >7% and 30d >30% thresholds.
- ☐ `processors/gain_filter.py` excludes all stablecoins listed in §5.5.
- ☐ `processors/uniformity_filter.py` implements the exact 5-step algorithm from §6.2.
- ☐ `processors/uniformity_filter.py` returns score 0 for `total_gain ≤ 0`.
- ☐ `main.py` pipeline executes all 10 stages in the correct order per §5.1.
- ☐ `main.py` data flow enrichment matches §4.3 (each stage adds the documented keys).
- ☐ `main.py` entry/exit detection matches §5.10 (set-difference logic).
- ☐ Chart generation uses MEXC → Kraken → Coinbase priority per §5.11.

Milestone 3 — Notifications, Bot Commands, and Operational Infrastructure

Goal: Align notification message formatting with §10.1–10.2, align bot commands with §10.3, complete the metrics collector per §12.3, verify logging architecture per §12.1, and finalize the operational scripts and documentation.

Depends on: Milestone 2 (pipeline is fully aligned, schemas are normalized).

Sprint 3.1 — Notification Formatting and Telegram Bot Command Alignment

Goal: Align `notifications/formatter.py` message templates with §10.1 (entry) and §10.2 (exit). Align `telegram_bot.py` commands with §10.3. Remove the `/start` command (not in spec) or keep it as a harmless alias for `/help`.

Spec References

The implementing agent for Sprint 3.1 MUST have the following in its context window:

Source	Sections / Files
§10.1 — Entry Notifications	Exact caption format with emojis, HTML structure, CMC URL, exchange volume lines
§10.2 — Exit Notifications	Exact exit message format
§10.3 — Telegram Bot Commands	`/status`, `/list`, `/help` — command names and response descriptions
§5.11 — Stage 10	Notification delivery flow
`notifications/formatter.py`	Current message formatting
`notifications/telegram.py`	Current Telegram client
`telegram_bot.py`	Current bot command handlers

Deliverables

File	Nature of Change
`notifications/formatter.py`	**DEV-19.** Verify or create `MessageFormatter` class with `format_entry()` and `format_exit()` methods. `format_entry()` must produce the exact HTML caption from §10.1 including: ◍ header with CMC link, gains section, uniformity score, exchange volumes with emojis (▧ Coinbase, 🐙 Kraken, ▨ MEXC). Volume lines must show `No volume` instead of `$0` or `N/A` when coin is not traded on an exchange. `format_exit()` must produce the exact format from §10.2 with timestamp, ◍ header, and CMC link.
`telegram_bot.py`	Align commands with §10.3. `/status`: current number of active coins, last scan time, scan duration (read from `scan_stats.json` or `metrics.json`). `/list`: list all currently qualified coins with uniformity scores. `/help`: list available commands. Decide on `/start`: either remove or alias to `/help`.
`main.py`	Update notification delivery in Stage 10 to use `MessageFormatter.format_entry()` and `MessageFormatter.format_exit()` if not already doing so.

Steps

1. **Read `notifications/formatter.py` in full.** If `MessageFormatter` class does not exist, create it. If formatting is inline in `main.py` or `notifications/telegram.py`, extract it into `MessageFormatter`.
2. **Align `format_entry()` output** with the exact template in §10.1. Pay special attention to: the `<a href>` HTML link wrapping the coin symbol and name, the exchange emoji mapping (§10.1), and the "No volume" substitution rule.
3. **Align `format_exit()` output** with §10.2. Include the `{timestamp}` field (ISO 8601 or human-readable).
4. **Read `telegram_bot.py` command handlers.** Verify `/status` returns: active coin count, last scan time, and scan duration. If last scan time is read from `scan_stats.json`, verify the file path matches the stats file written by `scheduler.py`. Verify `/list` shows uniformity scores alongside coin names/symbols. Verify `/help` lists all three commands.
5. **Handle `/start` command.** The spec does not include it. Two options: (a) remove it, or (b) keep it as a friendly alias that sends the same response as `/help`. Option (b) is preferred for UX — Telegram users instinctively send `/start` when they first interact with a bot.
6. **Update `main.py` Stage 10** to use `MessageFormatter` methods if it currently formats messages inline.

Exit Criteria

- ☐ `notifications/formatter.py` exports `MessageFormatter` with `format_entry()` and `format_exit()`.
- ☐ Entry notification caption matches §10.1 format exactly (including emojis, HTML, volume substitution).
- ☐ Exit notification message matches §10.2 format exactly.

- [ ] `/status` command returns active count, last scan time, and scan duration.
- [ ] `/list` command includes uniformity scores.
- [ ] `/help` lists `/status`, `/list`, `/help`.

Sprint 3.2 — Metrics, Logging, and Operational Finalization

Goal: Align `utils/metrics.py` with §12.3 (all required counters, `metrics.json` output, `/status` integration). Verify `utils/logger.py` matches §12.1 (two-handler setup, format strings, rotation settings). Verify scheduling scripts match §11. Final documentation pass.

Spec References

The implementing agent for Sprint 3.2 MUST have the following in its context window:

Source	Sections / Files
§12.1 — Logging Architecture	Two-handler setup, format strings, rotation (10 MB × 5 backups)
§12.2 — Log Files	`trend_scanner.log`, `bot_output.log`
§12.3 — Scan Metrics	All required counters: total fetched, eliminated per stage, API calls per service, cache hit/miss, wall-clock per stage, total duration
§11.1 — Scheduled Tasks	Cron schedule table
§11.2 — Scan Locking	File lock behavior
§14.1 — Scan Profile	Expected performance characteristics
`utils/metrics.py`	Current implementation
`utils/logger.py`	Current implementation
`scheduler.py`	Current implementation
`update_exchanges.py`	Current implementation
`update_mappings.py`	Current implementation

Deliverables

File	Nature of Change
`utils/metrics.py`	**DEV-20.** Align `MetricsCollector` with §12.3. Must track: total coins fetched from CMC, coins eliminated at each of the 10 filter stages, API calls made to each service (CMC, CoinGecko, Chart-IMG, Telegram), cache hit/miss ratios (price cache, exchange volume cache), wall-clock time per pipeline stage, total scan duration. Must write to `metrics.json` after each scan.

File	Nature of Change
`utils/logger.py`	Verify: `setup_logger(name, log_file)` creates two handlers — console (`StreamHandler(sys.stdout)`, `INFO`, simple `%(message)s` format) and file (`RotatingFileHandler`, `DEBUG`, detailed `%(asctime)s - %(name)s - %(levelname)s - %(message)s` format, 10 MB max, 5 backups). Fix if different. Verify `app_logger` is pre-configured for `trend_scanner.log`.
`scheduler.py`	Verify: cron entry point matches §11.1–11.2. Lock file path, PID write, lock release with unlink. Stats file. No changes expected — already well-aligned.
`update_exchanges.py`	Verify: uses `ExchangeDatabase` and `ExchangeFetcher` from `exchange_data/`. Uses updated DB path.
`update_mappings.py`	Verify: uses `CoinGeckoMapper` from `api/`. Uses updated DB path.
`main.py`	Integrate metrics collection: call `metrics.record_stage()` (or equivalent) at each pipeline stage boundary. Call `metrics.save()` at scan completion.
`linear-trend-spotter-spec.md`	No changes to the spec itself — this plan exists alongside it.
`README.md`	Final content pass. Ensure it contains: project name and description, Telegram invite link, setup instructions (mentioning `.env` and `config.json`), and a brief explanation of how the scanner works.

Steps

1. **Read `utils/metrics.py` in full.** Compare counter inventory against §12.3. Add any missing counters. Ensure `save()` method writes to `metrics.json` in the format expected by the `/status` bot command.
2. **Read `utils/logger.py` in full.** Verify the two-handler architecture, format strings, and rotation settings match §12.1. Fix any discrepancy.
3. **Read `scheduler.py` in full.** Verify it aligns with §11.1–11.2. No major changes expected.
4. **Read `update_exchanges.py` and `update_mappings.py`.** Verify they reference the correct database paths (updated in Sprint 1.2). Verify they use the correct module imports.
5. **Integrate metrics into `main.py`.** At each pipeline stage boundary, record the stage name, survivor count, API calls made, and wall-clock duration. At scan completion, call `metrics.save()`.
6. **Final `README.md` pass.** Add or update: project description per §2.2, Telegram group link, setup instructions, brief pipeline overview. Keep it concise — the spec is the detailed reference.
7. **Final integration check.** Mentally walk through the full pipeline from `scheduler.py` through `main.run_scanner()` through all 10 stages to notification delivery. Verify every import, every config access, every DB path, and every class name is consistent with the spec.

Exit Criteria

- [ ] `utils/metrics.py` tracks all counters listed in §12.3.
- [ ] `metrics.json` is written after each scan with the complete counter set.
- [ ] `utils/logger.py` creates two-handler loggers matching §12.1.
- [ ] `trend_scanner.log` uses `RotatingFileHandler` with 10 MB max and 5 backups.
- [ ] `update_exchanges.py` and `update_mappings.py` use correct DB paths and imports.

- [ ] `main.py` records metrics at each pipeline stage boundary.
- [ ] `README.md` contains project overview, Telegram invite, and setup instructions.
- [ ] A full mental walkthrough of `scheduler.py` → `main.run_scanner()` → all 10 stages → notification delivery encounters no naming inconsistencies, missing imports, or broken config accesses.

Complete Change Manifest

All files created, modified, relocated, or archived across all three milestones.

Milestone 1 — Repository Cleanup and Naming (2 sprints)

Sprint 1.1 — Archive and Structure:

```
.archive/20260228-001-build_mapping_db.py          (archived from root)
.archive/20260228-001-linear-trend-spotter-spec.txt (archived from docs/)
3rd-party-map.json                                  (relocated from docs/)
.env.example                                        (created or verified)
config_json.example                                 (created or verified)
.gitignore                                          (updated)
README.md                                           (verified or updated)
requirements.txt                                    (verified)
```

Sprint 1.2 — Config and Naming:

```
config/settings.py                                  (modified — defaults,
db_paths, properties)
api/coingecko.py                                    (renamed from
api/coingecko_optimized.py)
api/__init__.py                                     (updated exports)
database/cache.py                                   (class rename to PriceCache)
database/__init__.py                                (updated exports)
config/__init__.py                                  (verified exports)
main.py                                             (updated imports)
telegram_bot.py                                     (updated imports)
```

Milestone 2 — Pipeline and Schema Alignment (2 sprints)

Sprint 2.1 — Schema and Rate Limiter:

```
database/models.py                                  (schema alignment, migration
logic)
database/cache.py                                   (schema alignment, TTL logic)
exchange_data/exchange_db.py                        (schema alignment, path
update)
api/coingecko_mapper.py                             (schema alignment, path
update)
```

```
utils/rate_limiter.py                               (429 escalation, jitter,
config integration)
```

Sprint 2.2 — Pipeline and Processors:

```
processors/gain_filter.py                           (verified or aligned)
processors/uniformity_filter.py                     (verified or aligned)
config/constants.py                                 (stablecoin list verified)
main.py                                             (pipeline stage alignment)
```

Milestone 3 — Notifications, Metrics, and Finalization (2 sprints)

Sprint 3.1 — Notifications and Bot:

```
notifications/formatter.py                          (aligned MessageFormatter)
telegram_bot.py                                     (command alignment)
main.py                                             (Stage 10 formatter
integration)
```

Sprint 3.2 — Metrics, Logging, and Finalization:

```
utils/metrics.py                                    (counter alignment,
metrics.json)
utils/logger.py                                     (verified or aligned)
scheduler.py                                        (verified)
update_exchanges.py                                 (path verification)
update_mappings.py                                  (path verification)
main.py                                             (metrics integration)
README.md                                           (final content)
```

Approximate total: ~25 files touched across 3 milestones / 6 sprints.

Risk Notes

Risk	Mitigation
Production DB migration. Renaming `history.db` → `scanner.db` and adding columns could break the live system if done carelessly.	Use `ALTER TABLE ADD COLUMN` wrapped in try/except. Provide a transition period where both old and new DB names are supported. Never use `DROP TABLE`.
CoinLore remnants. The old spec referenced CoinLore; some code paths may still invoke it.	Sprint 1.2 removes the cache class. Sprint 2.2 audits `main.py` for any remaining CoinLore calls.

Risk	Mitigation
Rename cascade. Renaming `CoinGeckoOptimizedClient` → `CoinGeckoClient` touches every file that imports it.	Sprint 1.2 handles this systematically with a grep-and-replace pass across all `.py` files.
Rate limiter behavioral change. Adding 429 escalation and jitter changes API call timing.	The changes make the system *more conservative* (slower when rate-limited), not less. This is safe to deploy without staged rollout.
Metrics integration in `main.py`. Adding `metrics.record()` calls at each stage boundary increases `main.py` complexity.	The calls are lightweight (counter increments and `time.time()` snapshots). Use the existing `timed_block()` context manager from `utils/metrics.py` where possible.
Bot command `/start` removal. Users who have previously interacted with the bot may send `/start` expecting a response.	Keep `/start` as an alias for `/help` rather than removing it entirely.