Habit Tracking Application

Phase 2 (Development) Presentation

Kiran Ravada

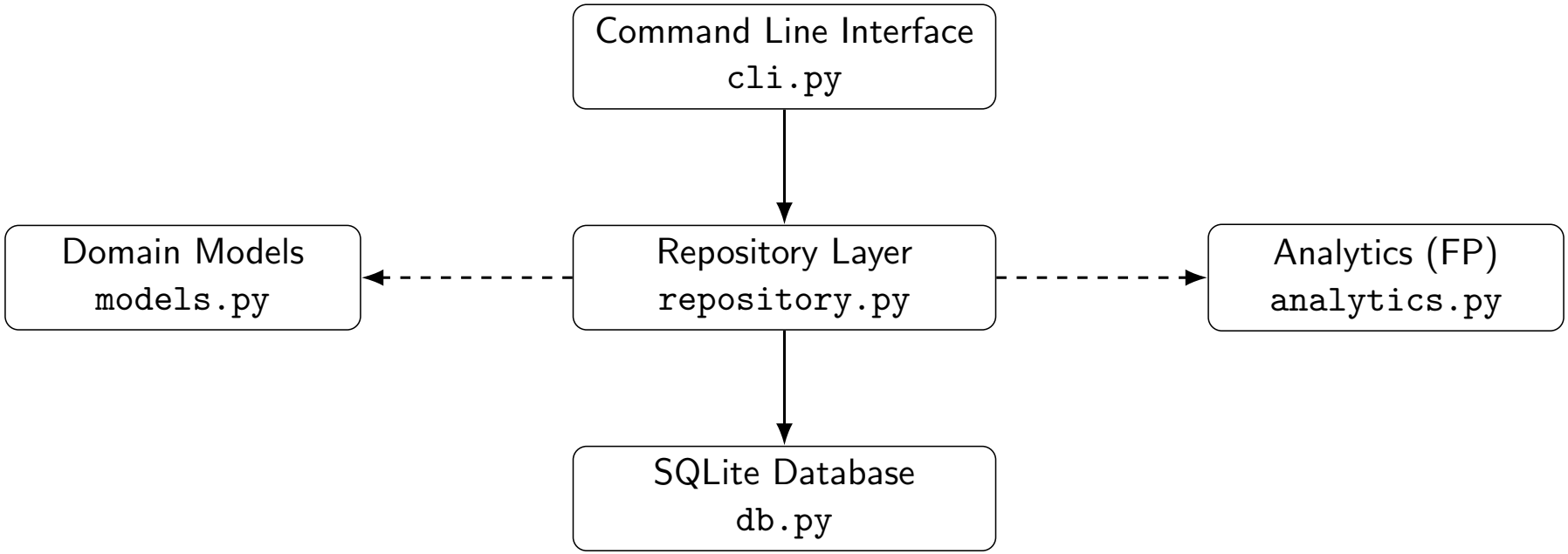
IU Internationale Hochschule

February 20, 2026

Project Overview & Phase-1 Summary

- Goal: backend habit tracking application in Python (CLI-based)
- OOP: domain model (Habit, Completion events) + repository layer
- FP: analytics functions for streaks and filtering
- Persistence: SQLite database (habits + completion event log)
- Phase 1 deliverables: architecture concept, data model, streak logic definition

Current Architecture (Updated Diagram)



Main flow: CLI → Repository → Database. Repository uses Models and Analytics.

Habit Class & Event Log Structure

Habit (OOP domain entity)

- `id`, `name`, `task`, `periodicity` (daily/weekly), `created_at`

Completion (Event log record)

- `habit_id`, `completed_at`
- One habit $\rightarrow$ many completion events (history)

Reasoning: Event log supports streak computation and auditability (when completions happened).

Storage Approach (SQLite) & Data Flow

Storage choice: SQLite (DB)

- Tables:
  - `habits(id, name UNIQUE, task, periodicity, created_at)`
  - `completions(id, habit_id FK, completed_at)`
- Write path: CLI → Repository → INSERT/DELETE in DB
- Read path: Repository → load habits/completions → analytics (pure functions)

CLI Workflow: Create → Check-off → View

Create habit

- `python -m habits.cli add --name "Workout" --task "20 min exercise" --period daily`

Check-off completion

- `python -m habits.cli check --name "Workout"`

View completion history

- `python -m habits.cli history --name "Workout"`

Other commands: `list`, `delete`, `seed`, `seed-data`, `analyze`

Analytics Functions (FP) – Design & Status

Functional programming approach

- Use of `map`, `filter`, and pure helper functions
- Analytics is read-only (no DB writes)

Implemented analytics

- List habits by periodicity
- Longest streak per habit
- Longest streak overall

Streak logic

- Daily habits: consecutive calendar days
- Weekly habits: consecutive ISO calendar weeks
- Gaps break the streak

Predefined Habits + 4-Week Test Data

Predefined habits

- `seed` command creates 5 habits (daily + weekly mix)

4-week example data

- `seed-data` inserts realistic completion events across the last 4 weeks
- Purpose: reproducible demo dataset and easier testing of analytics/streaks

Testing Strategy & Early Test Results

Testing approach

- Framework: `pytest`
- Unit tests focus on streak logic correctness:
  - Daily consecutive completions
  - Daily streak with gaps
  - Weekly streak across ISO weeks

Result

- Tests executed successfully (e.g., `3 passed`)

Progress vs Original Plan

Original plan (Phase 1)

- CLI-based interaction
- Modular structure (separate files)
- Persistent storage + streak analytics
- Unit tests + dummy data

Current progress (Phase 2)

- Modular codebase implemented (models/repo/db/analytics/cli)
- SQLite persistence in place
- CLI workflow implemented and tested with seed data
- Analytics and pytest tests implemented

Next Steps Toward Phase 3

- Finalize portfolio: polish documentation and explanations
- Strengthen test suite (more edge cases, optional DB integration tests)
- Minor refactoring/cleanup for readability and maintainability
- Ensure final submission package is clean (no `.venv`, no `habits.db`)