Habit Tracker App

CLI Habit Tracking Application (Python)

Nikita Wokurka

IU DLBDSOOFPP01 — Portfolio Part 2 (Development/Reflection)

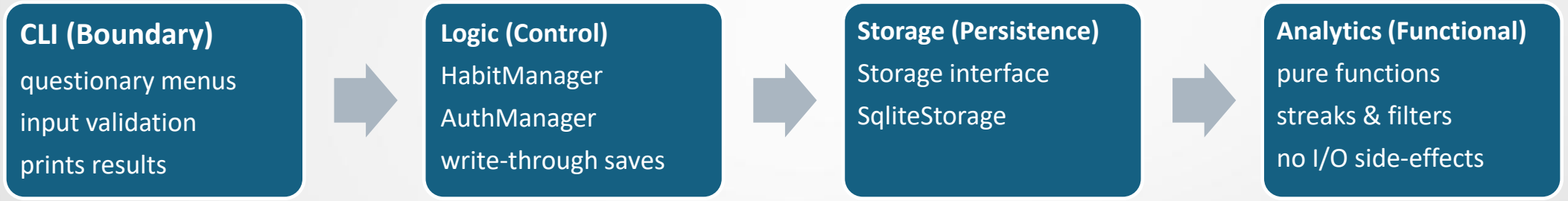
Overview

CLI habit tracking backend (OOP + functional analytics)

- Goal: track daily/weekly habits, record check-offs, and compute streak analytics with authentication
- Scope: CLI-only (no GUI), single-user workflow, modular architecture for future UI/storage swaps
- Python: 3.7+ (implemented with modern Python features)
- Persistence: SQLite by default; JSON as optional alternative backend

Architecture

Separation of concerns: CLI → Services/Logic → Storage + Analytics



- Key rule: UI contains no business logic; analytics functions accept data and return computed results.

Data Model

Habit entity + completion events (suitable for JSON or SQLite)

Habit entity

src > habit_tracker > models > habit.py > ...

You, last month | 1 author (You)

```python
from datetime import datetime
```

You, last month | 1 author (You)

```python
class Habit:
    """Class representing a habit in the habit tracker application."""

    # Initialize a Habit instance
    def __init__(
        self,
        habit_id: int,
        name: str,
        periodicity: str,
        created_date: datetime,
        description: str = None,
        completion_dates: list[datetime] = None
    ):
        self.habit_id = habit_id
        self.name = name
        self.description = description
        self.periodicity = periodicity
        self.completion_dates = completion_dates or []
        self.created_date = created_date

    def log_completion(self, date: datetime):
        """Log the completion of the habit for a specific date."""
        self.completion_dates.append(date)

    def __repr__(self):
        """Return a string representation of the Habit instance."""
        return f"Habit(id={self.habit_id}, name='{self.name}', periodicity='{self.periodicity}')"
```

SqlLite

id	name	description	periodicity	created_date
1	Drink Water	Drink at least 2L	daily	2025-12-21T13:31:51.296605
2	Meditate	10 minutes mindfulness	daily	2025-12-21T13:31:51.301930
3	Read	Read 10 pages	daily	2025-12-21T13:31:51.307430
4	Workout	Exercise 3 times	weekly	2025-12-21T13:31:51.313244
5	Clean Apartment	1 deep clean session	weekly	2025-12-21T13:31:51.318932

Each completion is stored as a timestamped event

id	habit_id	completion_date
1	1	2025-12-21T13:34:14.246906
2	2	2025-12-21T13:34:16.666810
3	3	2025-12-21T13:34:23.742687
4	4	2025-12-21T13:34:25.658077
5	5	2025-12-21T13:34:27.453112

id	username	password_hash	salt
1	default	eh103wXRiK09KAu6l+39MyeQPfsYjgezLgMKVgTIUr4=	/kDpb75Pl73Ot6CNktr+/w==

User Flow

Create → Check-off → Persist → Analytics

Welcome Screen

```
========================
 🏆 Habit Tracker CLI 🏆
========================

? How would you like to use the Habit Tracker today? (Use arrow keys)
 » 🔐 Log into my habit tracker (real data)
   🧪 Try demo with example data (resets every time)
   🚪 Exit application
```

Login

```
? How would you like to use the Habit Tracker today? 🔐 Log into my habit tracker (real data)

🔐 Please log in.

? Password: ************

✅ Login successful!
```

Check-off

```
? Choose an action: Log completion
? Select a habit to log: (id=4) Drinking Water

✅ Habit logged: (id=4) Drinking Water
```

Main menu

```
========================
 🏆 Habit Tracker CLI 🏆
========================

? Choose an action: (Use arrow keys)
 » Add Habit
   Remove Habit
   Log completion
   Analytics
   Exit
```

Habit creation

```
========================
 🏆 Habit Tracker CLI 🏆
========================

? Choose an action: Add Habit
? Enter habit name: Drinking Water
? Choose periodicity: Daily
? Enter description (optional): 2L
```

Habit inspection

```
📋 All habits
Which habit do you want to inspect? 1: Drink Water (daily)

📌 Habit: Drink Water
🆔 ID: 1
🕒 Periodicity: daily
📝 Description: Drink at least 2L
📅 Created: 2025-12-21 13:31
🔥 Longest streak: 1 period(s)
⚡ Current streak: 1 period(s)

✅ Completion dates:
   • 2025-12-21 13:34
```

Habit list

```
? Choose an action: Analytics
? Analytics – choose an option: List all habits

📋 All habits
Which habit do you want to inspect? (Use arrow keys)
 » 1: Drink Water (daily)
   2: Meditate (daily)
   3: Read (daily)
   4: Workout (weekly)
   5: Clean Apartment (weekly)
   Cancel
```

Analytics menu

```
? Choose an action: Analytics
? Analytics – choose an option: (Use arrow keys)
 » List all habits
   List daily habits
   List weekly habits
   Show longest streak (overall)
   Show longest streak for a habit
   Back to main menu
```

Key Features

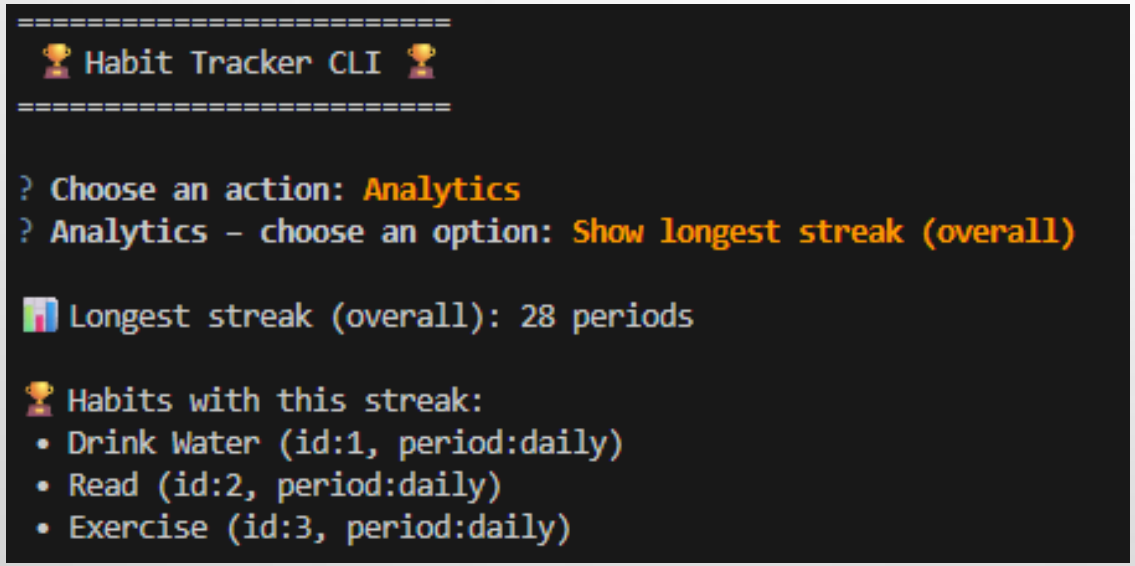
Core requirements implemented

- Supports daily and weekly habits (periodicity is part of the habit definition).
- Check-off logging uses timestamps; history enables streak.
- Streak computation respects periodicity rules (daily vs weekly).
- Predefined fixtures: 5 habits + 4 weeks sample check-offs for quick demo & testing.
- Write-through persistence: user actions are saved immediately.
- Authentication with SHA256 Hash and salt

Analytics

Questions answered (pure functions) + sample CLI output

- List all habits
- List habits by periodicity (daily / weekly)
- Longest streak overall (across all habits)
- Longest streak for a selected habit



Tech Choices

Why these tools?

- SQLite: lightweight, zero-config DB; supports structured queries and reliable persistence.
- JSON (optional): simple file portability for quick experiments.
- questionary: user-friendly interactive CLI menus (less error-prone than raw input()).
- pytest: fast unit tests for streak rules and analytics correctness.
- stdlib: sqlite3/json/datetime + hashlib for secure password hashing (if auth enabled).

How to Run

Clean install + common commands (copy/paste)

⚙️ Installation (Step-by-Step)

1️⃣ Clone the repository

```
git clone https://github.com/n1wo/habit-tracker.git
```

```
cd habit-tracker
```

2️⃣ Create and activate a virtual environment

Windows (PowerShell):

```
python -m venv .venv
```

```
.venv\Scripts\activate
```

macOS/Linux:

```
python -m venv .venv
```

```
source .venv/bin/activate
```

3️⃣ Install the project (runtime dependencies)

Install the project in editable mode so Python automatically finds the src/ layout:

```
pip install -e .
```

(Optional) Install development & test dependencies

```
pip install -r requirements-dev.txt
```

🚀 Usage

Run the app

From the project root:

```
python -m habit_tracker
```

The CLI will guide you through:

- 🔐 First-time password setup (stored securely)
- 📋 Main menu for creating, viewing, and analyzing habits
- ✅ Marking habits as completed
- 🏆 Viewing streak analytics

Example flow:

```
> Add new habit
> Mark habit as completed
> View all habits
> Analyze longest streak
```

Tests & Quality

Confidence in streak rules + pure analytics

- Unit tests cover: habit creation, check-off behavior, streak calculations, analytics filters.
- Edge cases handled: duplicate check-offs in same period; empty completion history; mixed daily/weekly.
- Storage is abstracted → tests can use InMemoryStorage to isolate logic.
- Docstrings + README ensure reproducible setup and usage.

```
(.venv) PS C:\Users\nikit\OneDrive\Documents\GitHub\habit-tracker> pytest
============================= test session starts =============================
platform win32 -- Python 3.13.7, pytest-9.0.1, pluggy-1.6.0
rootdir: C:\Users\nikit\OneDrive\Documents\GitHub\habit-tracker
configfile: pytest.ini
testpaths: tests
collected 64 items

tests\test_analytics.py .........................                        [ 40%]
tests\test_auth_manager.py ...........                                   [ 57%]
tests\test_example_data_factories.py ....                                [ 64%]
tests\test_habit.py ..                                                   [ 67%]
tests\test_habit_manager.py ...........                                  [ 84%]
tests\test_sql_store.py .....                                            [ 92%]
tests\test_sql_store_user.py ...                                         [ 96%]
tests\test_user.py ..                                                    [100%]

============================= 64 passed in 0.69s ==============================
```

Reflection

What worked, trade-offs, limits, next steps

- Worked well: clean separation (UI/logic/storage/analytics) → easier testing and maintenance.
- Trade-offs: CLI-first UX keeps scope tight; SQLite adds structure vs JSON simplicity.
- Known limits: single user; basic reporting; no reminders/notifications; minimal visualization.
- Next steps: richer analytics (missed periods, trends), export CSV, multi-user profiles, optional web UI.