Habit Tracker App

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
- 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	Drinking Water	2l	daily	2025-12-17T15:03:47.061523
2	Running	5km	weekly	2025-12-17T15:04:17.503878

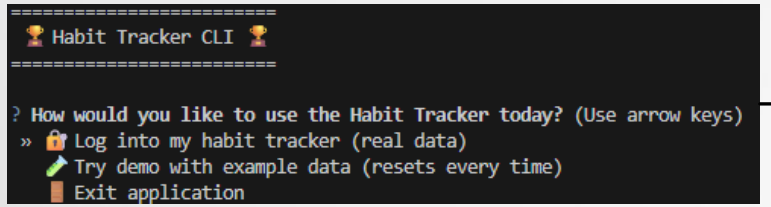
id	habit_id	completion_date
1	1	2025-12-17T15:06:00.564245

id	username	password_hash	salt
1	default	eh103wXRiK09KAu6l+39MyeQPfsYjgezLgMKVgTlUr4=	/kDpb75Pl73Ot6CNktr+/w==

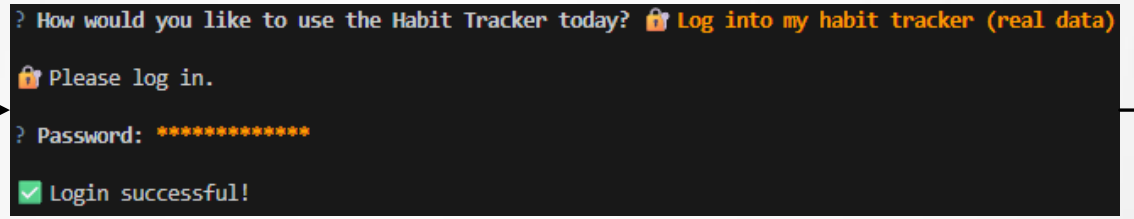
User Flow

Create → Check-off → Persist → Analytics

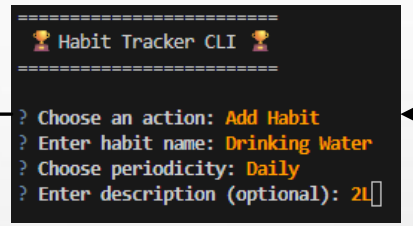
Welcome screen



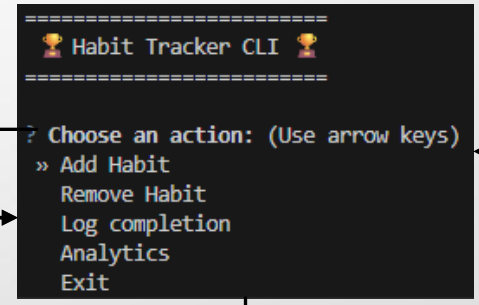
login



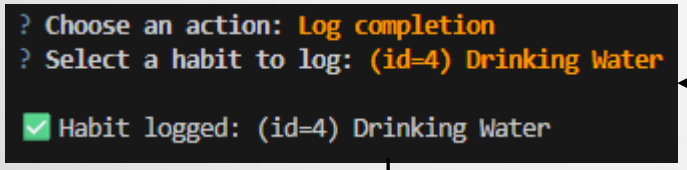
habit creation



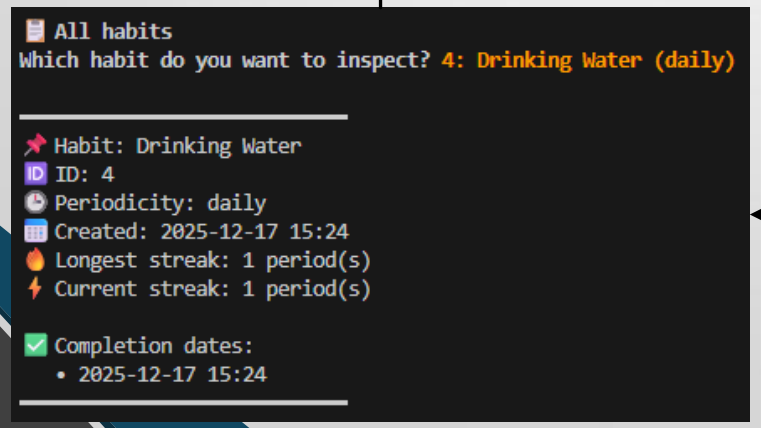
main menu



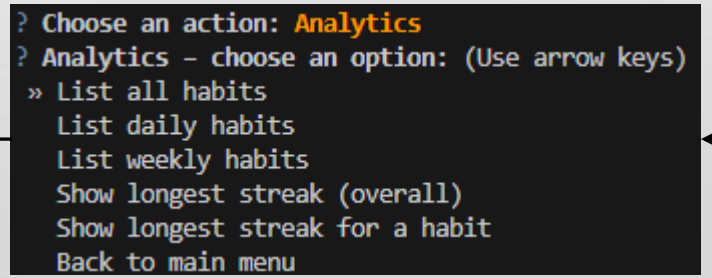
Check-off



Habit inspection



analytics menu



Key Features

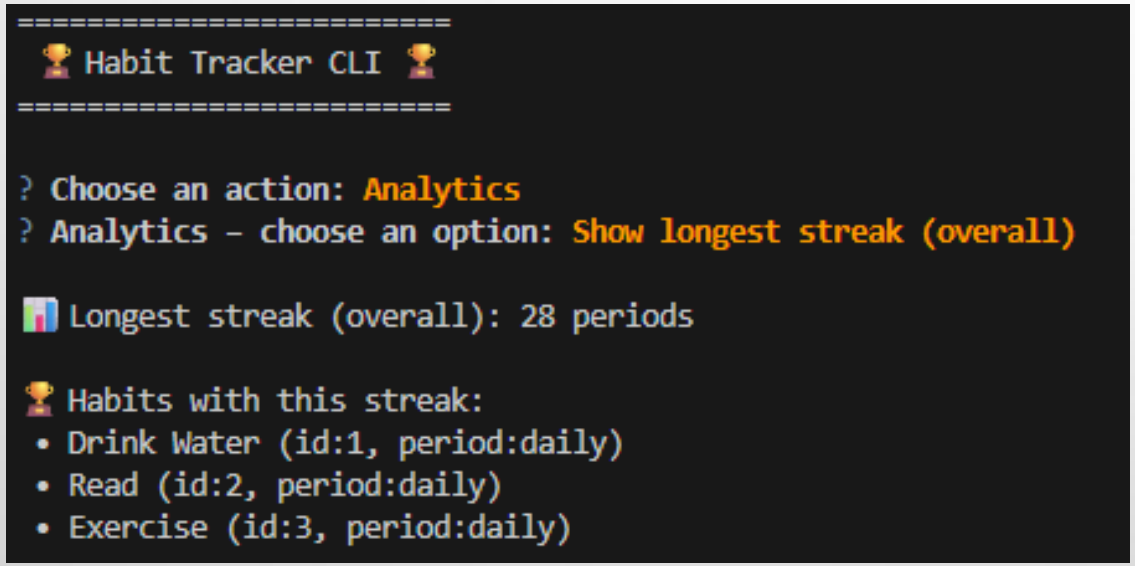
Core requirements implemented

- Supports daily and weekly habits (periodicity is part of the habit definition).
- Check-off logging uses timestamps; history enables streak and trend analysis.
- 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.

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/<your-username>/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 dependencies

Install required packages:

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

Or use an **editable install** so Python automatically finds your `src/` folder:

```
pip install -e .
```

🚀 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

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.