Habit Tracking App: Abstract

1. Concept

The *Habit Tracking App* was developes as a command-line interface (CLI) tool aimed at helping users build and maintain positive routines by tracking their adherence to self-defined habits. At its core, the app focuses on fostering behavioral consistency through regular task completion within user-specified timeframes (daily, weekly, or monthly).

Primary Goals

- Enable users to define habits by name, description, and periodicity.
- Track the successful completion of habits through timestamped events.
- Provide analytical insights on performance streaks and habit adherence.
- Offer an intuitive CLI-based user experience suitable for automation and quick interactions.

The system was envisioned to support physical button inputs as an alternative trigger for logging events, adding a hardware-software interaction component (separate module). However, demonstrating this feature within the scope of this project is not feasible, as it would have required the user to possess a compatible physical button device.

2. Architecture and Functional Design

The app is structured around Python backend, organized into the components:

a) **main.py**: Serves as the CLI entry point, guiding users through options such as creating habits, completing tasks, analyzing data, and exiting the app.

b) **db.py**: Implements data storage using SQLite3. It manages two tables:

   - counter: stores metadata for each habit: name, description, period type, and required completions per period.
   - tracker: records the timestamped completion events linked to each habit.

c) **counter.py**: encapsulates the Counter class representing a habit entity. The class holds habit's metadata (name, description, periodicity), tracks in-memory count and implements __str__() to render summary. The module also includes methods to manage the habit lifecycle: creation and deletion.

d) **analyse.py**: responsible for turning raw habit-completion data into meaningful metrics and streak calculations**.**

e) **test_project.py**: contains unit tests using *pytest,* validating the correctness of habit CRUD operations, database integrity and analytical computations.

3. Core Functionality

The Habit Tracker supports all standard CRUD operations:

- Creates new habits with customizable name, description, and frequency.
- Reads existing habits and associated metadata.
- Updates the informaton on the events (manual or automatic timestamps).
- Deletes: habits and all their associated completion events.

Each completion event is recorded in the database, which is then used by the analytics module to: list all existing habbits, group them by periodicity or calculate current (habit specific) or longest (among all habits) streaks.

The streak system uses only *positive streaks*—periods where the completion count meets or exceeds the required threshold. There are no negative streaks (e.g., tracking avoidance behavior).

4. Predefined data

To support immediate exploration of the app's features, the application comes preloaded with five sample habits: two daily, two weekly, and one monthly, along with example tracking data (exact events on each habit with the timestamp). This dataset allows user to see how the application works and instantly analyze current progress: observe how streak calculations function across varying periodicities.

5. Current Status and Next Steps

With the implementation of its database schema, domain logic, and CLI interface, the Habit Tracking App has evolved into a fully functional and testable prototype. It is currently ready for user feedback, usability trials, and potential enhancement (e.g., GUI frontend, mobile interface, physical device integration).