Conception Phase

1. Project Purpose

The purpose of this project is to build a Habit Tracking App in Python, with a primary focus on backend functionality. A “habit” is a task that must be completed by the user at regular intervals.

2. Core Functionality

2.1. Habit Creation. Users can create new habits by providing a name, a description (text), and a periodicity (frequency within a time period).

2.2. Task Completion. Each habit must be checked off at least once during its defined period. If a user fails to complete the task in time, the habit is considered broken.

2.3. Event Logging. The system records a timestamp for every occurrence of each tracked action (task/event within the habit).

2.4. Streaks. If a user completes a habit for *x* consecutive periods without breaking it, the application records a streak of *x* periods.

2.5. Analytics. The app shall be able to:

- List all currently tracked habits.
- List habits grouped by the same periodicity.
- Return the longest streak across all habits.
- Return the longest streak for a specific habit.

2.6. User Interface. All primary user interactions occur via a Python-based CLI. Through guided prompts, users can create habits, complete tasks, view analytics, and exit.

3. Additional Enhancements

To extend the standard specification and increase complexity, we introduce the following deviations:

3.1. Physical Button Integration. As an addition to CLI, a physical button is introduced. The button supports multiple signals (short press, long press, rotation, etc.). Each unique interaction maps to a defined action (event). Events can be triggered manually via the CLI or automatically via the button.

3.2. Positive Streaks. The concept of streak is understood as positive event: Completing at least the planned number of events in each period (e.g., eat vegetables ≥ 3 times per day). No negative streaks (avoiding an action beyond a threshold in each period) are allowed.

3.3. System Uptime & Data Persistence. Persist all event logs and habit data locally (e.g., SQLite or file-based storage) to prevent data loss across restarts. On restart, the program restores all habit states, including last-checked timestamps and current streaks.

3.4. Integration. The program provides interfaces for the physical button.

4. User Flow. The flow of action may look like the following:

4.1. User creates a new habit (name, description, periodicity).

4.2. User triggers events via CLI or physical button; the system recognizes each event.

4.3. Events are stored in the database.

4.4. The analytics module calculates streaks.

5. Technical Realization

5.1. Language & Version: Python 3.13 or later.

5.2. Database: SQLite3 to store habit definitions, event timestamps, descriptions, and periodicities.

5.3. CLI as an interface for the user (using *questionary* or similar library).

5.4. Core Class: Habit (or Counter). Keeps attributes for name, description, period type, and period count. Streak for the given habit shall be defined as positive or negative. Class includes the methods for adding events, incrementing counts, persisting to, and retrieving from the database.

5.5. Analytical Module: provides analytics (counts the number of events for a given habit, shows all habits, list habits sharing the same periodicity, counts the information on streaks for a specific habit or all habits).

5.6. Main Module: Implements the user interface for habit management, task completion, analytics, and exit flows.

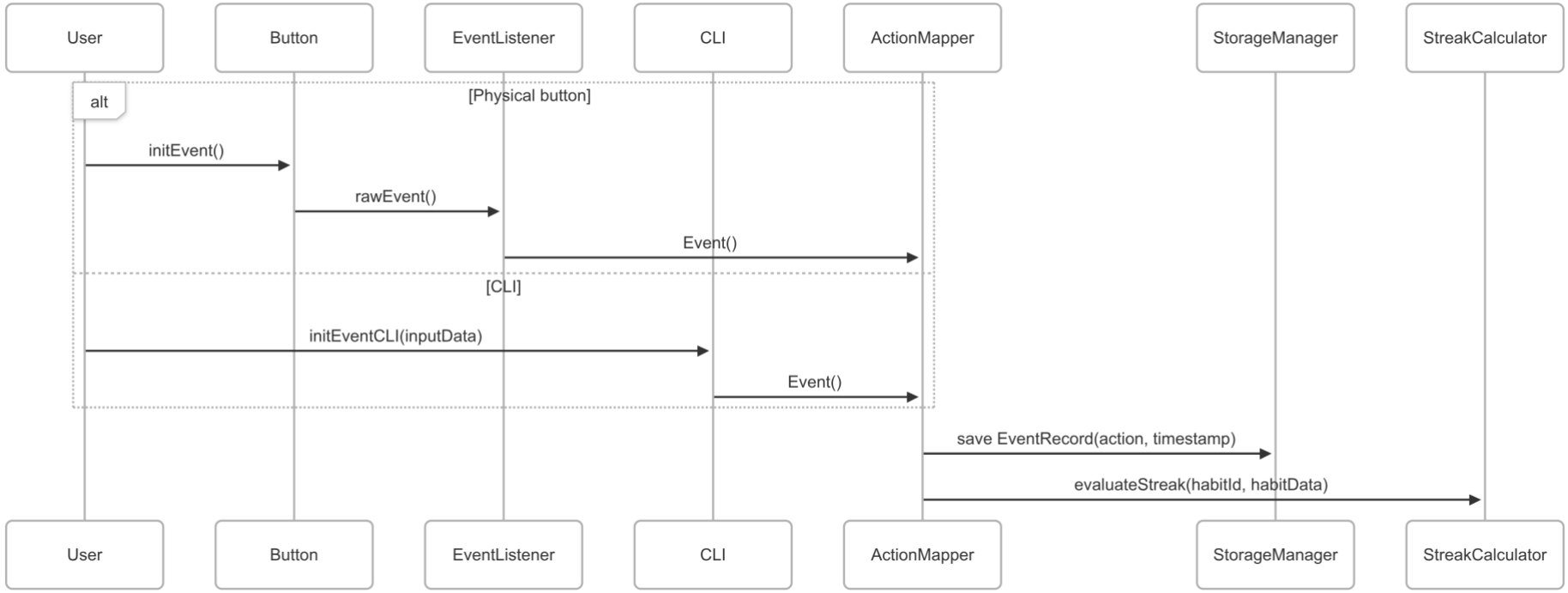
6. Documentation

6.1. README.md with project overview and installation instructions.

6.2. requirements.txt listing all external dependencies.

7. UML Sequence Diagram

The following UML Sequence Diagram illustrates the flow of operations within the system for the given habit (after its creation).


User
Button
EventListener
CLI
ActionMapper
StorageManager
StreakCalculator
alt
[Physical button]
initEvent()
rawEvent()
Event()
[CLI]
initEventCLI(inputData)
Event()
save EventRecord(action, timestamp)
evaluateStreak(habitId, habitData)
User
Button
EventListener
CLI
ActionMapper
StorageManager
StreakCalculator