HCLI - Habit Tracker CLI

Development Phase Overview

Alejandro Moral Aranda

March 2, 2025

Introduction

- **HCLI** is a CLI-based habit tracker designed to help users manage daily or weekly habits.
- It uses several Python libraries:
  - `typer` for CLI command handling.
  - `rich` for enhanced terminal output.
  - `json` for data persistence.
- The codebase is structured into components for configuration, habit management, and analytics.

Application Architecture

- **ConfigManager:** Manages configuration files and paths.
- **HabitTracker:** Handles core functionalities:
  - Loading/saving habit data.
  - Adding/checking habits.
  - Calculating streaks and generating analytics.
- **Typer CLI:** Integrates commands and maps user input to functionalities.

ConfigManager: Handling Application Configurations

- Manages file paths for data and user configuration.
- Ensures that the configuration file (`config.json`) exists.
- Loads and saves configuration data using JSON.
- Key responsibilities:
  - Initialize with default values.
  - Load configuration from disk.
  - Save configuration to disk.
  - Set and get configuration parameters.

HabitTracker: Core Functionality

- Loads habit data and user information.
- Provides methods for:
  - Adding a new habit (`add_habit`).
  - Checking a habit (`check_habit`).
  - Listing habits, calculating streaks, and generating summaries.
- Key responsibilities:
  - Initialize with configuration data.
  - Load habit data from JSON files.
  - Save habit data to JSON files.
  - Handle user interactions and data integrity.

Managing Habits: Add and Check Methods

- **add_habit:** Registers a new habit with its periodicity.
- **check_habit:** Marks a habit as completed (optionally for a given date).
- Key steps:
  - Validate habit name and periodicity.
  - Store habit details in memory.
  - Save habit data to JSON file.
  - Provide user feedback.

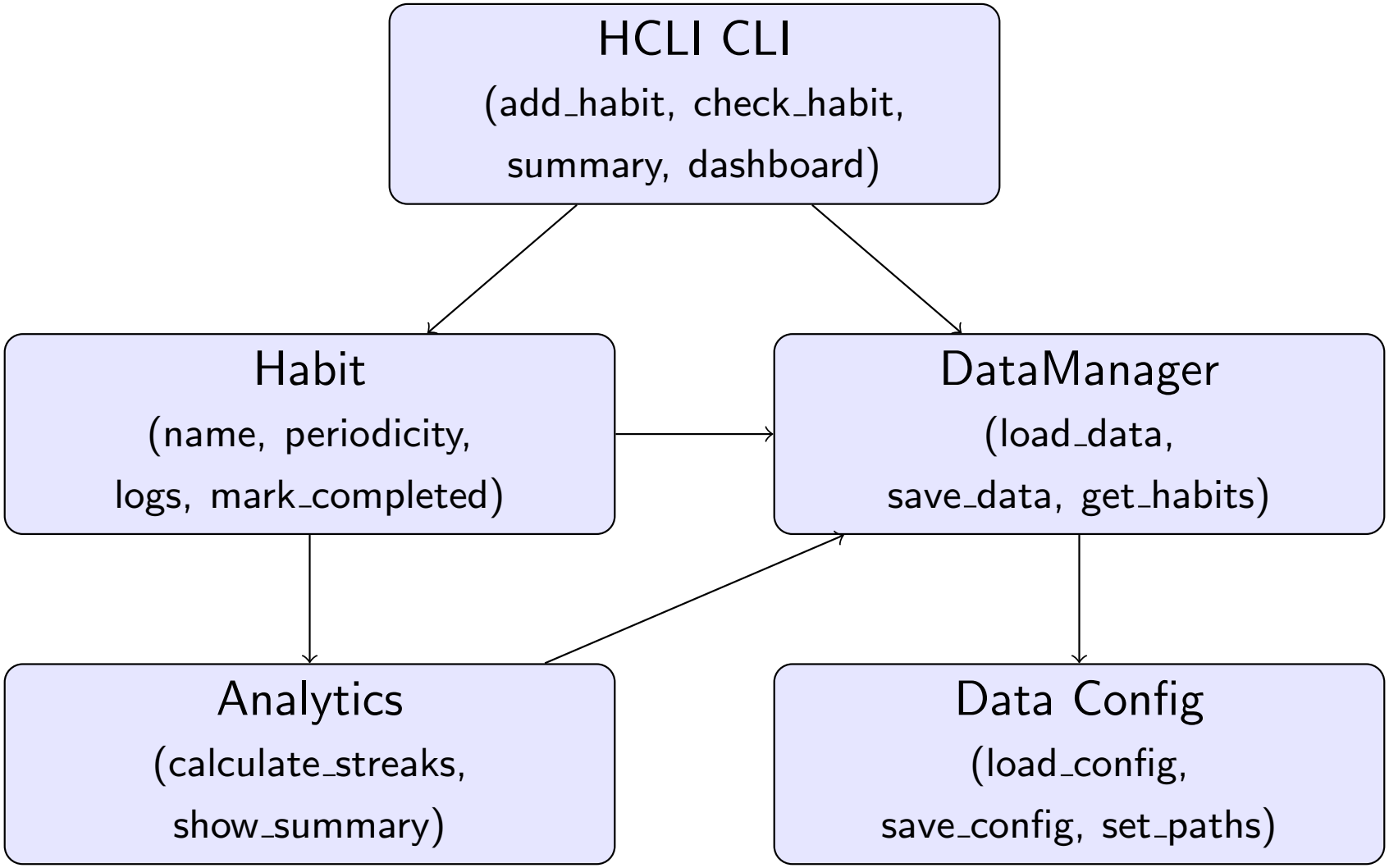
Analyzing Habit Data: Streaks and Summary

- **streaks:** Computes current and longest streaks for each habit.
- **summary:** Provides an overview including total habits, check-ins, and pending habits.
- Key steps:
  - Sort habit logs by date.
  - Calculate streaks based on periodicity.
  - Generate a summary report.
  - Highlight pending habits.

Connecting Commands with Typer

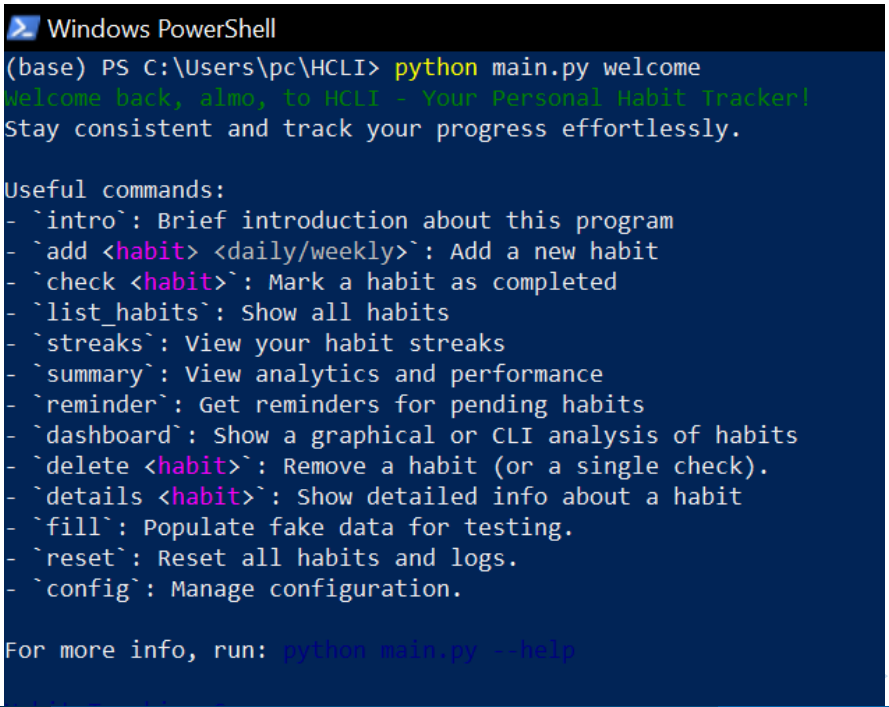
- Uses the `typer` library to define CLI commands.
- Each command (e.g., `add`, `check`, `list_habits`) maps to a corresponding method in `HabitTracker`.
- Key steps:
  - Define command options and arguments.
  - Map commands to functions in `HabitTracker`.
  - Handle exceptions and provide user feedback.

System Architecture Diagram



Visualizing Habit Data: Dashboard

- The `dashboard` method displays a graphical or ASCII-based chart of habit progress.
- Utilizes `rich` for ASCII dashboards and `matplotlib` for graphical views.
- Key steps:
  - Fetch habit data from the DataManager.
  - Generate ASCII bars or matplotlib plots.
  - Display the dashboard in the terminal.



Next Steps & Conclusion

- **Enhancements:**
  - Expand test coverage.
  - Refine dashboard visualizations.
  - Optimize data storage and performance.
- **Conclusion:**
  - HCLI has evolved into a robust CLI tool.
  - Continuous improvements will further refine the user experience.

Thank You!