PROJECT: Habits Tracker Application with Python(OOFP)

Conception Phase

Tourad Baba

Date: 2023-11-07

Habits Tracking Application Concept

Overview

The Habit Tracking Application is designed to empower users to build and maintain healthy habits by providing a feature-rich and user-friendly platform. This concept outlines the technical foundations, main components, user interactions, and data management of the application.

Components

Habit Class:

The "Habit" class is the core component, representing individual habits.

It includes attributes like name, description, periodicity (daily, weekly, monthly), category, creation time, streak count, completion time, and a list of historical completions.

Methods within the Habit class handle adding, removing, updating, and tracking habit progress.

Database (db):

Habit data is stored in "db"

The database maintains two tables: one for habit tracking and another for habit logs.

It offers methods for adding and removing habits, updating logs, fetching habits as choices, getting streak counts, updating habit streaks, resetting logs, and more.

Analytics Module:

The Analytics Module provides deep insights into users' habit-building journeys.

It includes functions for habit analysis, retrieving a list of tracked habits, categorizing habits by periodicity, determining the longest streak across all habits, finding the longest streak for a specific habit and more. The Analytics Module will be implemented using the functional programming paradigm.

Command Line Interface (CLI):

The CLI offers a user-friendly interface for interacting with the application.

Users can create, update, and remove habits, visualize analytics, and perform habit-related tasks.

Data Persistence:

Habit data is persistently stored and retrieved from the database, ensuring users can track their habits across sessions.

User Flow

Creating and Managing Habits:

Users can add new habits, providing details like name, description, periodicity, and category.

Habit names must be unique.

Users can update habit details, such as name, description, category, and periodicity.

Changing periodicity resets the streak and completion time.

Tracking Habit Progress:

Users can mark habits as completed, incrementing their streak counts.

The application automatically records completion times and updates historical completions.

Analytics:

Users can access the Analytics Module to gain insights into their habits.

The application calculates the longest streak, tracks habits by periodicity, calculate habits completion rate and identifies areas where users can improve.

Resetting Habit Streaks:

Users have the option to reset their habit streaks if needed.

The application maintains historical data, even after a reset.

Verifying Habit Streaks:

The application automatically verifies daily, weekly, and monthly streaks to encourage consistency.

Comprehensive Data Management:

Habit data is meticulously managed within the HabitDatabase, ensuring data integrity and providing a seamless user experience.

The database allows users to maintain detailed habit logs, enabling historical analysis and reflection.

Visualization (UML Diagrams):

IN THE NEXT PAGES

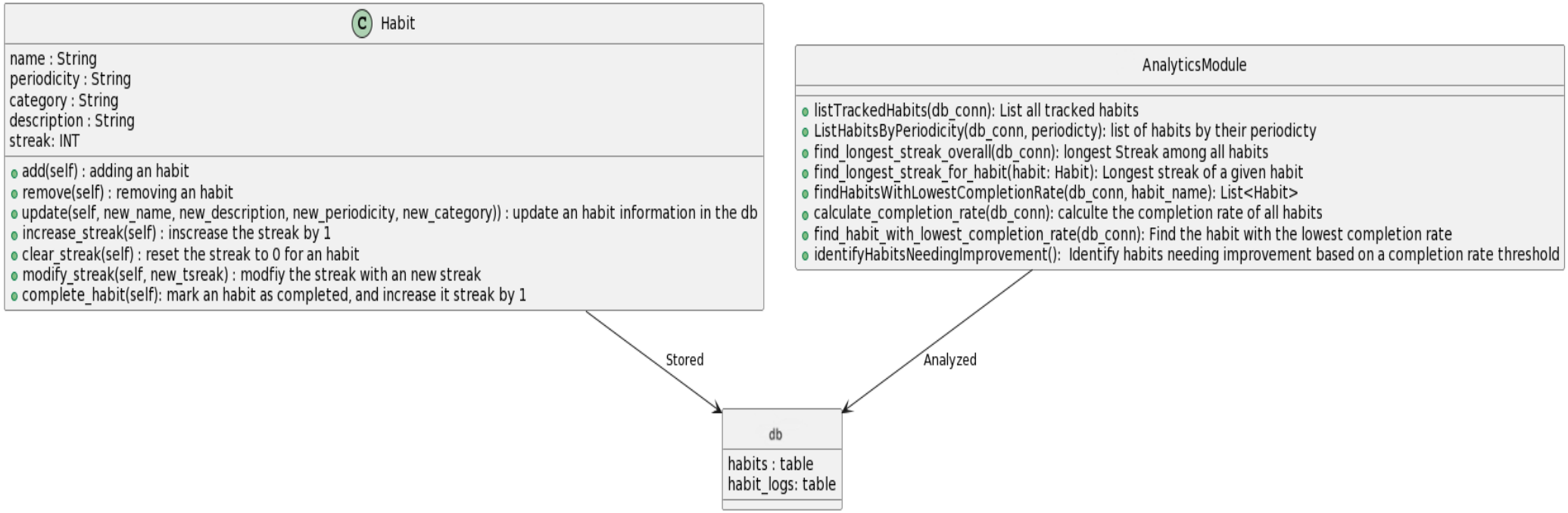
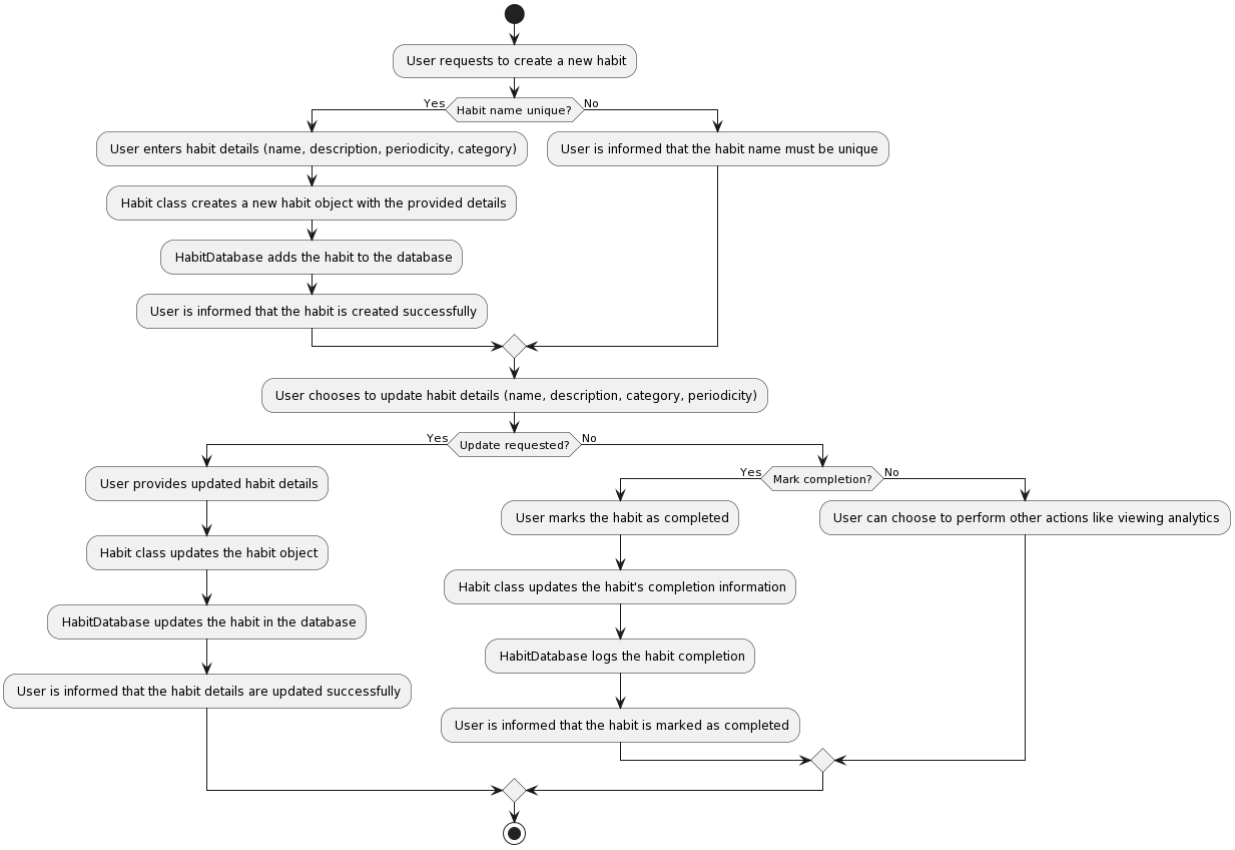

Habit
name : String
periodicity : String
category : String
description : String
streak: INT
add(self) : adding an habit
remove(self) : removing an habit
update(self, new_name, new_description, new_periodicity, new_category)) : update an habit information in the db
increase_streak(self) : inscrease the streak by 1
clear_streak(self) : reset the streak to 0 for an habit
modify_streak(self, new_tsreak) : modfiy the streak with an new streak
complete_habit(self): mark an habit as completed, and increase it streak by 1
AnalyticsModule
listTrackedHabits(db_conn): List all tracked habits
ListHabitsByPeriodicity(db_conn, periodicty): list of habits by their periodicty
find_longest_streak_overall(db_conn): longest Streak among all habits
find_longest_streak_for_habit(habit: Habit): Longest streak of a given habit
findHabitsWithLowestCompletionRate(db_conn, habit_name): List<Habit>
calculate_completion_rate(db_conn): calculte the completion rate of all habits
find_habit_with_lowest_completion_rate(db_conn): Find the habit with the lowest completion rate
identifyHabitsNeedingImprovement(): Identify habits needing improvement based on a completion rate threshold
Stored
Analyzed
db
habits : table
habit_logs: table

Diagram of some actions the user can perform



By following this comprehensive concept, The aim is to build a sophisticated habit tracking application that not only assists users in forming healthy habits but also provides in-depth analytics to enhance their habit-building experience.