Student’s Check-in

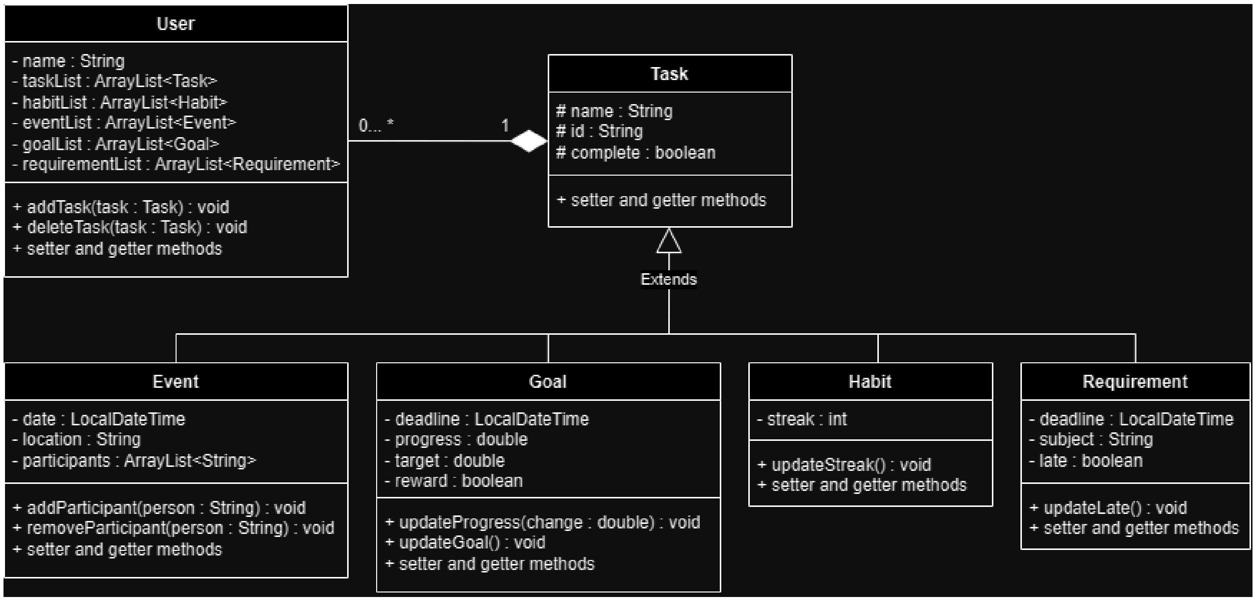
Reasons for Choosing our Topic

Based on our previous experiences, we noticed that students, including ourselves, fail to finish their assigned tasks before the deadline. Often, it becomes hard for students to keep track of their many assignments from different subjects alongside their activities that occur outside of school. Due to this, a lot of students have difficulty balancing their social, personal and academic lives.

We had three proposals to choose from. The **Student’s Music Library**, the **Mini Calendar**, and the **Requirements Checklist**. As a group, we had discussed during our first quarter proposal if there was a way we could combine these proposals as they had some overlap.

To solve this issue, we combined two of our projects, namely the Requirements Checklist and the Mini Calendar. In doing so, our app is able to create a list of requirements with specified due dates and subjects based on user input and visualize them in calendar form. Furthermore, the addition of events and habits gives students the ability to organize their personal lives and focus on their self-growth. All in all, we hope that our project entitled **Student’s Check-in** will live up to its goal of improving the time management skills and productivity of our fellow students.

UML Diagram


User
- name : String
- taskList : ArrayList<Task>
- habitList : ArrayList<Habit>
- eventList : ArrayList<Event>
- goalList : ArrayList<Goal>
- requirementList : ArrayList<Requirement>
+ addTask(task : Task) : void
+ deleteTask(task : Task) : void
+ setter and getter methods
0... *
1
Task
# name : String
# id : String
# complete : boolean
+ setter and getter methods
Extends
Event
- date : LocalDateTime
- location : String
- participants : ArrayList<String>
+ addParticipant(person : String) : void
+ removeParticipant(person : String) : void
+ setter and getter methods
Goal
- deadline : LocalDateTime
- progress : double
- target : double
- reward : boolean
+ updateProgress(change : double) : void
+ updateGoal() : void
+ setter and getter methods
Habit
- streak : int
+ updateStreak() : void
+ setter and getter methods
Requirement
- deadline : LocalDateTime
- subject : String
- late : boolean
+ updateLate() : void
+ setter and getter methods

CRC Cards

User	
• Has a name • Has a habit list • Has an event list • Has a requirement list • Has a task list • Can add or remove tasks • Checks requirements (Sets Requirements as completed or not) • Can select task • Can sort a Task-list	• Task • Event • Habit • Requirement • Goal

Task	Event, Habit, Requirement, Goal
• Has a name • Has an id • Can be completed • Has a date and time • Can be renamed • Can be rescheduled • Can be added or deleted from a User list • Can send a reminder	• User • Event • Habit • Goal • Requirement

Event	Task
• Has the same responsibilities as the Task Class, except in a separate list • Automatically checked once date and time have passed • Can have "mini-tasks" • Mini-tasks can be checked as completed or not	• User • Task • Habit • Requirement • Goal

Habit	Task
• Has a progress meter • Progress resets daily (or by an interval specified by the user) • Has a daily target • Has "mini-habits" • Has a simple checkbox • Does not have a specified date or time • Does not appear on Calendar • Has a streak that increases by 1 if checked during the interval • Otherwise, streak resets to 0	• User • Task • Event • Requirement • Goal

Requirement	Task
• Has a subject • Has a quarter • Can be set as completed or not • Can be late if past the due date • Can check its due date	• User • Task • Event • Habit • Goal

Goal	Task
• Has progress, target, and a date • Appears on the calendar • User can add or subtract to the progress at any time • Reward specified to be positive or negative • If positive, goal will be completed when progress >= target • If negative, goal will be completed when progress <= target	• User • Task • Event • Requirement • Habit

Scenarios

Scenario #1: School Day

1. User creates two Requirements and adds them to their list.
2. User finishes one of the Requirements and it is marked as complete.
3. User failed to complete the other Requirement before its deadline and it is marked as Late.

Scenario #2: Reunion

1. User creates an Event scheduling an upcoming family reunion and adds it to their list.
2. User invites two people as participants to the Event.
3. Once the Event's date passes, it is marked as complete.

Scenario #3: Workout Plan

1. User creates a Habit to exercise everyday.
2. User creates Goal g1 with a positive reward to finish 10 push-ups.
3. User creates Goal g2 with a negative reward to lose 10.0 kilograms of weight.
4. User adds these Tasks to their list.
5. User updates Goal g1 with 8 push-ups.
6. User updates Goal g2 with losing 12.5 kilograms of weight and it is marked as complete.
7. User checks their Habit to exercise and it is marked as complete for the day.
8. Once a day has passed, the Habit's streak is increased to 1 and it is unchecked.

Exceptions

TaskNotFoundException

- Occurs when the User attempts to delete a Task that isn't in their TaskList.
- Usually impossible to encounter in the application and is only meant to act as a failsafe.

DayOutOfBoundsException

- Occurs if the User attempts to input a day that is beyond the accepted range for the specific month (i.e. January 32, 2024).
- The error is displayed below the textbox and the User is forced to input a valid day before they can create the Task.

YearOutOfBoundsException

- Occurs if the User attempts to input a negative year or a year that is beyond the maximum positive integer limit 999999999 of the LocalDateTime class.
- The error is displayed below the textbox forcing the User to choose a year within the valid range before they can create the Task.