Yong Han Ching - Project Portfolio

PROJECT: TAble

Overview

TAble is your handy desktop app, optimized for TAs who prefer to work with a Command Line Interface (CLI) while still having the benefits of a Graphical User Interface (GUI)! I have added the functionality for TAs to arrange consults and view all upcoming events in a calendar! The GUI of TAble was created with JavaFX and written in Java, and I have written about 4 kLoC.

Summary of contributions

- **Major enhancement**: Enabled support for **Consults**
  - **What it does**: The user is able to add/edit/delete/list all `Consults` which he has scheduled with a current `Student`.
  - **Highlights**: A separate consult panel that displays all of the user's `Consults` exist for the user to easily keep track of all his consults. Any consults that are overdue are also highlighted in red, to remind the user to delete these outdated consults.
- **Major enhancement**: Added a calendar view for TAble.
  - **What it does**: Added a calendar view that displays all the `Consults`, `Tutorials` and `Reminders` for the user.
  - **Highlights**: This calendar is able to show the number of `Consults`, `Tutorials` and `Reminders` that occurs on a given day in one view, without any information being ommitted.
  - Credits: *{Code for the Calendar Window adapted from https://github.com/SirGoose3432/javafx-calendar/blob/master/src/FullCalendarView.java}*
- **All code contributed**: [RepoSense - Code Report]
- **Other contributions**:
  - Project management:
    - Raised issues that needed to be solved and provided some solutions to these problems. [Issues raised/discussed]
  - Documentation:
    - Contributed in the `Consult` and `Calendar` sections of the User Guide:
    - Contributed in the `Consult` and `Calendar` sections of the Developer Guide: which includes the class diagram for `Consults`, activity diagram for `addConsult` command and a sequence diagram for the creation of the Calendar. More details on these are covered below.
  - Tools:
    - Helped to set up Travis-CI and Netifly.

- Summary of contributions:
  - Created over [35 pull requests] on GitHub
  - Reviewed over [36 pull requests]on GitHub

Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Consults Feature

NOTE	The consults that are overdue will appear in red to highlight that these consults are ready to be deleted!

Adding a consult: `addConsult`

Add a consultation slot at the given time, date and place.
Format: `addConsult student/STUDENT_INDEX beginDateTime/BEGIN_DATE_TIME endDateTime/END_DATE_TIME place/PLACE`

- The 'STUDENT_INDEX' should be referred to by the index as referred to in the student list
- The begin and end date time provided must be in the yyyy-MM-dd HH:mm format
- The location provided should be any valid string i.e, must be alphanumeric
- The consult must be held within the same day, thus the begin and end DateTime objects should have the same date.

Example:

- `addConsult student/1 beginDateTime/2020-03-03 10:00 endDateTime/2020-03-03 12:00 place/Deck`

Editing a consult: `editConsult`

Edit the time, date or place of an existing consultation slot.
Format: `editConsult INDEX [beginDateTime/BEGINDATETIME] [endDateTime/ENDDATETIME] [place/PLACE]`

- Edits the consult at the specified `INDEX`. The index refers to the index number shown in the displayed consultation list. The index **must be a positive integer** 1, 2, 3, …
- At least one of the optional fields must be provided.
- Existing values will be updated to the input values.
- Note that one cannot edit the student involved in the consult, as it is unlikely that the TA will need to change the student at that consult timing.

Example:

- `editConsult 1 beginDateTime/2020-03-03 15:00`
  Edits the beginning time of the 1st consult to be at 2020-03-03, 15:00 hours i.e 3.00 p.m.
- `editConsult 2 place/SR3`
  Edits the place of the 2nd consult to be at SR3.

Listing all consults : `listConsult`

Shows a list of all consultations in TAble.
Format: `listConsult`

Deleting a consult: `deleteConsult`

Removes an existing consultation slot.
Format: `deleteConsult INDEX`

> - Deletes the consultation at the specified `INDEX`.
> - The index refers to the index number shown in the displayed consultation list.
> - The index **must be a positive integer** 1, 2, 3, …

Example:

- `listConsult`
  `deleteConsult 2`
  Deletes the 2nd consultation in TAble.

Clearing all consults : `clearConsults`

Clears all consultations slots in TAble.
Format: `clearConsults`

Locating consultations by date or place: `findConsult` [coming in v2.0]

Finds consultations whose date or place match any of the given keywords.
Format: `findConsult [DATE] [PLACE]`

> - At least one of the optional fields must be provided
> - If both optional fields are provided, only consults that meet both criterion will be returned
> - The search is case insensitive. e.g `SR1` will match `sr1`
> - The order of the keywords matters. e.g. `find SR1 03-03-2020` will throw an error

Examples:

- `findConsult 03-03-2020`
  Returns all consults on 03-03-2020
- `findConsult SR3`
  Returns any consults that are held at SR3

Calendar Feature

Display the calendar window: `viewCalendar`

Brings up the calendar window. Format: `viewCalendar`

- The calendar will be brought up, displaying the current month and year.
- For each month, the number of tutorials, consults and reminders for each day will be displayed in the calendar.
- Click on the left and right arrows to change the month displayed.

Close the calendar window: `closeCalendar`

Closes the calendar window. Format: `closeCalendar`

- The calendar will be closed, if a current calendar window is displayed.
- Alternatively, clicking the cross can also close the calendar window.

View consults and tutorials on a particular day [Coming in v2.0]

Contributions to the Developer Guide

Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.

Consult feature

TAble allows NUS SoC teaching assistants to track and record all their consultations scheduled with their students.

Implementation

A `Consult` class stores all of the relevant information of consults. The class diagram below shows how all the different components interact to allow the consult feature to function. Note that the XYZConsultCommand and XYZConsultCommandParser refers to all Consult related commands like add, edit, delete etc.

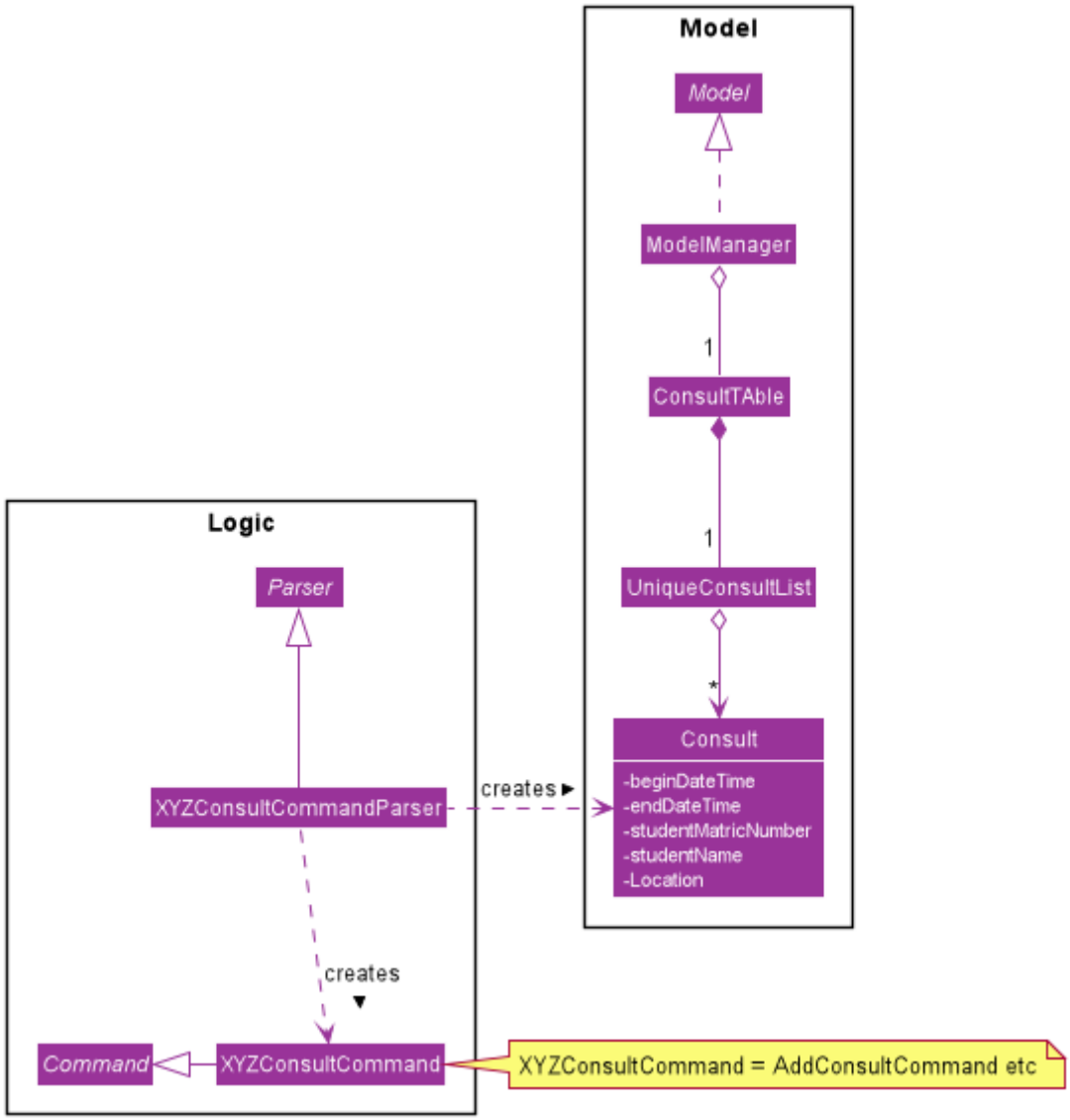

Figure 1. Class diagram of 'Consults' feature

A consultation slot is represented by the `Consult` class which contains 5 key attributes, the `beginDateTime`, `endDateTime`, `location`, `studentMatricNumber` and `studentName`. The `beginDateTime` and `endDateTime` attributes are represented using Java's `LocalDateTime` class. The `studentMatricNumber` and `studentName` attributes are classes that belong to a certain `Student`. The `studentMatricNumber` is used to uniquely identify the `Student` involved in the consultation, while the studentName is used in the front end of the application, displayed to the TA.

The `XYZConsultCommand` class represent classes that extend the abstract class `Command` and allows the TA to add, edit, delete, view and clear consultations added to TAble. These `XYZConsultCommandS` are created by the respective `XYZConsultCommandParserS`.

A `ConsultTAble` that has a `UniqueConsultList` stores and manages all `ConsultS`. When a `Consult` is added to the ConsultTAble, there will first be a check that the `Consult` does not already exist in the `UniqueConsultList`.

Given the class diagram and the understanding of how the `Consult` class interacts with other classes, let us examine how an `addConsult` command behaves, by using the following activity diagram of the system.

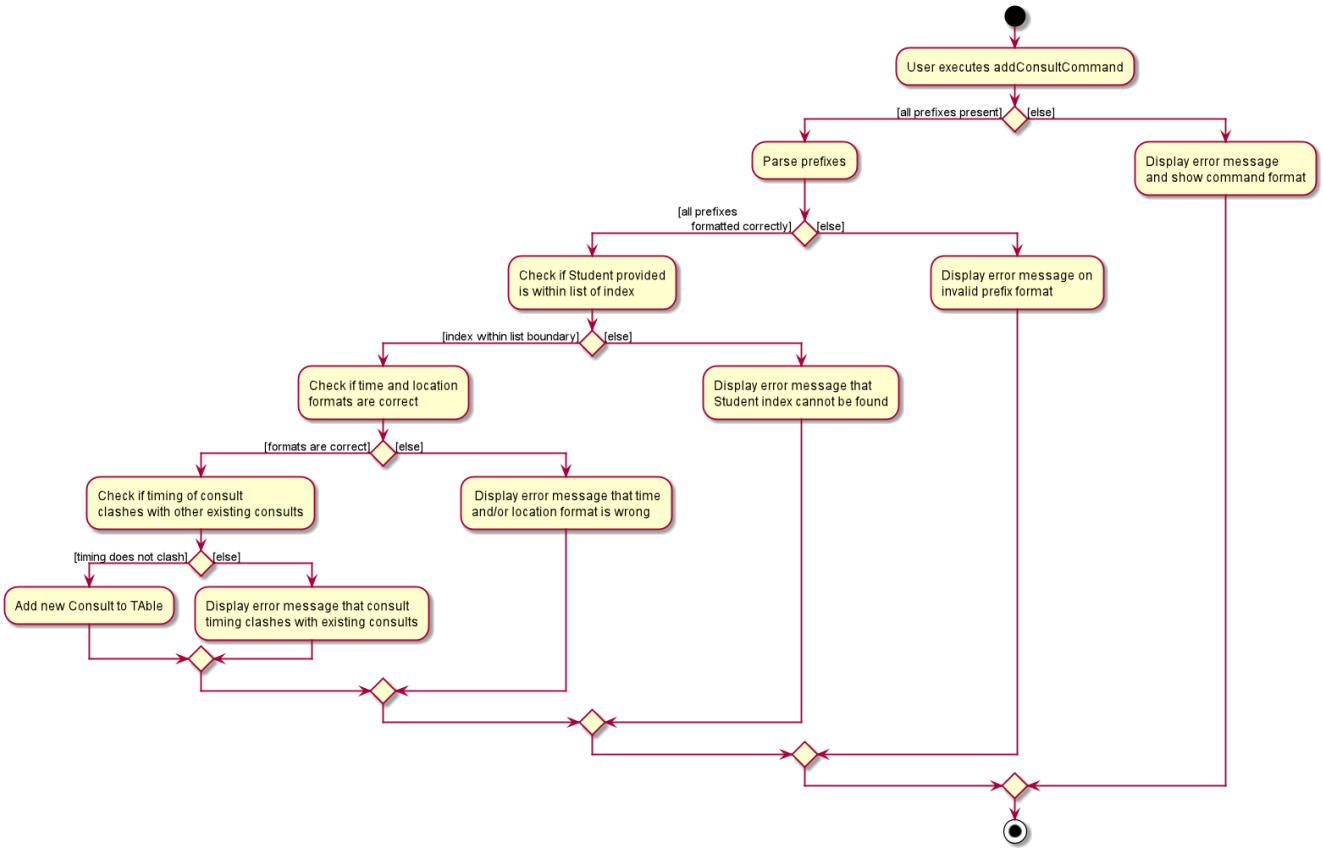

Figure 2. Activity diagram for the command addConsult in the `Consult` feature

The workflow above shows how a `Consult` is added and the various checks that occurs at the different points throughout the workflow.

To summarize the above activity diagram, there are 5 key checks which TAble checks for when the user is adding a `consult`. Firstly, TAble checks if all 4 prefixes are present in the command. Then, TAble checks if all prefixes present are formatted correctly. Following which, TAble will check if each of the data passed for the prefixes are in the right format. Firstly, it checks if the `Student` index provided is within the range of the `Student` list. Then, it checks if the timing and `Location` provided are in the correct format. Lastly, and most importantly, it checks if the `Consult` timing provided clashes with other existing `Consult` timings.

Design Considerations

Consult storing Matric Number and Name

	Alternative 1 (Current Choice): Make the consult store the matric number and name of the student that is attending the consult.	**Alternative 2 :** Make the consult store the entire Student class that is attending the consult.
Pros	It is easier to implement and coupling will be reduced as `Student` will not be directly related to the `consult` class.	It is easier to visualise as the entire `Student` related to the `consult` will be stored.
Cons	It is not as intuitive as currently, the `Consult` class only stores certain attributes of the `Student` that is attending the `consult`.	It will be harder to test due to the high degree of coupling.

Reason for choosing Alternative 1: Due to the time constraint of this project, our group has decided to choose alternative 1, as it not only reduces coupling, but is sufficient for us to uniquely identify the `Student` participating in the `consult` as the `MatricNumber` would be unique.

Sorting the consults

	Alternative 1 (Current Choice): Automatically sorts the consult list by the time of the consult, whenever a consult is added	**Alternative 2 :** Sorts the consult list only when user enters a sort command
Pros	It is intuitive for the consult list to be sorted according to their date and time.	User can decide how they want the list to be sorted (e.g. by alphabetical order for location).
Cons	User are not able to sort the list according to their preferences.	User have to type the sort command each time a new consult is added.

Reason for choosing Alternative 1: We believe that automatically sorting the consult list provides a better user experience as the user can immediately see the upcoming consults in the panel, without having to type a sort command.

Calendar feature

TAble allows NUS SoC teaching assistants to bring up a calendar view of all their upcoming consults, tutorials and reminders.

Implementation

The key command which users can call is the `viewCalendar` command.

Most functions to display the calendar window are called within the `CalendarWindow` class, as it is the main class that drives the calendar view. The `CalendarWindow` class extends `UiPart<Stage>`, and represents the calendar using a `GridPane` as the skeleton of the calendar, with individual dates being populated by a `CalendarDay` class which extends `UiPart<Region>`. The `CalendarDay` has a `StackPane` that displays the number of `ConsultS`, `TutorialS` and `ReminderS` each day.

The following sequence diagram demonstrates how the Calendar is initialised, each time the TAble app is set up.

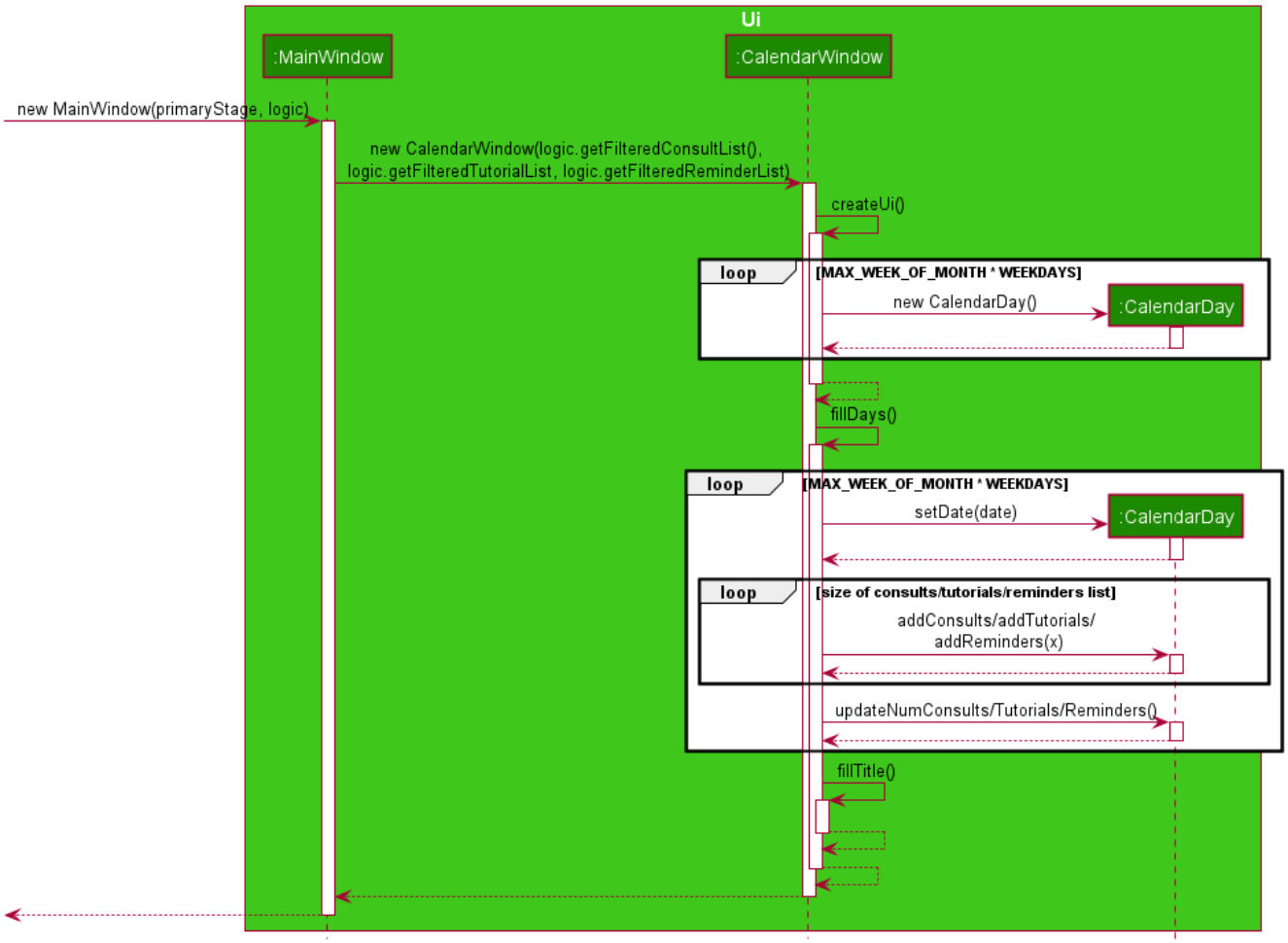

Figure 3. Sequence diagram of creation of Calendar

The above sequence diagram illustrates how the Calendar is initialised when the TAble app is start up. Thus, whenever the user inputs the `viewCalendar` command, this calendar window is brought up, similar to how the Help window is brought up as well. The calendar window brought up will show the calendar view of the current month and year.

The calendar window also stores the `UniqueConsultList`, `UniqueTutorialList` and the `UniqueReminderList`. As seen in the sequence diagram above, the calendar window checks for the consults, tutorials and reminders in the 3 lists, to see if they are on a particular date. If they occur on that date, the `StackPane` in `CalendarDay` will be updated to reflect the corresponding number of consult, tutorial and reminders. As seen in the screenshot below, the number of reminders, tutorials and consults will not clash, as they take up the top, middle and bottom rows respectively.

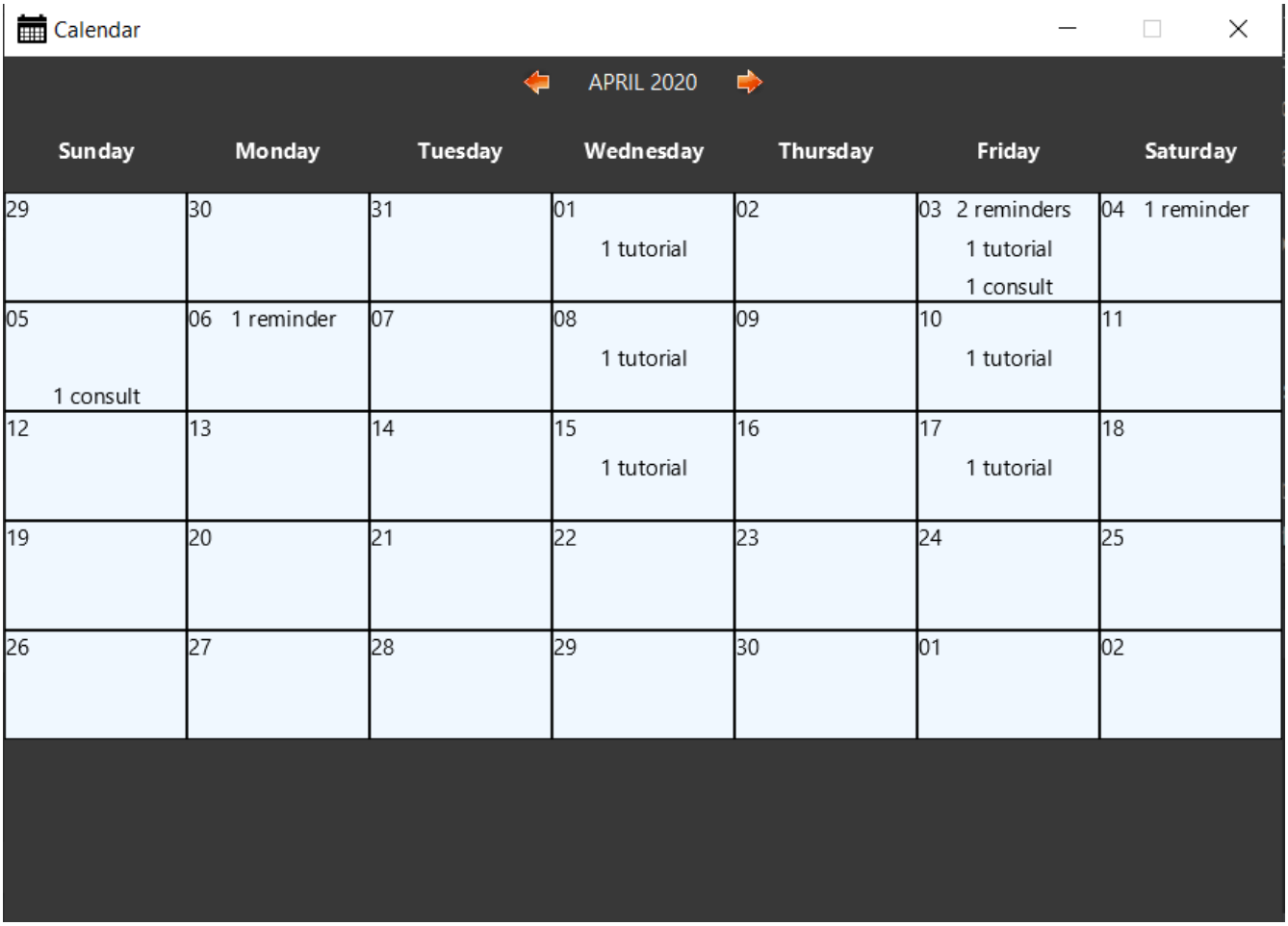

Figure 4. Screenshot of how the number of reminders, tutorials and consults are displayed on the same day

Design Considerations

Displaying calendar as a separate window

	Alternative 1 (Current Choice): Displaying the calendar view as a separate window.	**Alternative 2 :** Having a calendar view permanently by the side of TAble.
Pros	It is clearer and less visually clunky for the user, as information overload is prevented.	It is more intuitive, as the user can always view the calendar at the side for his upcoming schedules.
Cons	Lower user experience as the user has to type in a command to view the calendar.	The UI may be too cluttered as there are already alot of information being displayed currently.

Reason for choosing Alternative 1: We believe that our current implementation of the UI has all of TAble's core features, and the calendar view can and should be implemented as a separate window to prevent visual clutter. The command to view the calendar window command is also short and intuitive to remember, thus the user experience would likely not be significantly affected.

Use Cases

Use case: Edit consult (U03)

MSS

1. User requests to update either the time, date or place of an existing consult
2. System updates the existing consult according to the the User's request

   Use case ends.

Extensions

1a. The consult does not exist.

   1a1. System shows an error message that the consult does not exist.

   Use case ends.

1b. The time, date or place provided is in the wrong format.

   1b1. System shows an error message that time, date or place must be provided in the correct format.

   Use case ends.

Use case: Delete consult (U04)

MSS

1. User requests to delete an existing consult
2. System removes the existing consult from the database

   Use case ends.

Extensions

1a. The consult does not exist.

   1a1. System shows an error message that the consult does not exist.

   Use case ends.

1b. The user enters the delete consult command in the wrong format.

   1b1. System shows an error message that the wrong delete consult command format was given.

   Use case ends.