Chen Xing - Project Portfolio

[github]

PROJECT: NUSCouples

Overview

NUSCouples is a command-line desktop application targeted at couples studying at the National University of Singapore (NUS). It aims to help these couples create and remember new memories during their time in NUS.

Summary of contributions

- **Major enhancement**: added **the ability View/Add/Delete Calendar commands**
  - What it does: What it does is that it will show the Calendar Window when the App is opened. This Calendar have the ability to change the View of the calendar, adding new event(s) to it and deleting the event(s).
  - Justification: This feature improves the product significantly because a user needs a decent looking calendar with some basic features to keep them reminded and manage upcoming events with his/her partner or other miscellaneous events.
  - Highlights: This enhancement does not affect existing commands and commands to be added in future. It required an in-depth analysis of design alternatives. The implementation too was challenging as it required changes to existing commands and external Calendar API.
  - Credits: This features uses CalendarFx API interface, however the interface is in GUI so i manipulated commands to morph into a CLI calendar interface to fit into this project. _
- **Minor enhancement**: added keystrokes to the command box to ease the user by managing the caret to jump from last index to first index and vice versa.
- **Future enhancement**: will be implementing Google Calendar to sync the Calendar in the App currently. ==== Sync the calendar with Google API `[coming in v2.0]`

{explain how the user can authentic with google API and view and retrieve events from Google}

- **Code contributed**: [Functional code] [Test code] *{give links to collated code files}*
- **Other contributions**:

- Project management:
  - Managed releases `v1.3` - `v1.5rc` (3 releases) on GitHub
- Enhancements to existing features:
  - Updated the GUI color scheme (Pull requests #33, #34)
  - Wrote additional tests for existing features to increase coverage from 88% to 92% (Pull requests #36, #38)
- Documentation:
  - Did cosmetic tweaks to existing contents of the User Guide: #14
- Community:
  - PRs reviewed (with non-trivial review comments): #12, #32, #19, #42
  - Contributed to forum discussions (examples: 1, 2, 3, 4)
  - Reported bugs and suggestions for other teams in the class (examples: 1, 2, 3)
  - Some parts of the history feature I added was adopted by several other class mates (1, 2)
- Tools:
  - Integrated a third party library (Natty) to the project (#42)
  - Integrated a new Github plugin (CircleCI) to the team repo

{you can add/remove categories in the list above}

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.

Return to Table of Contents

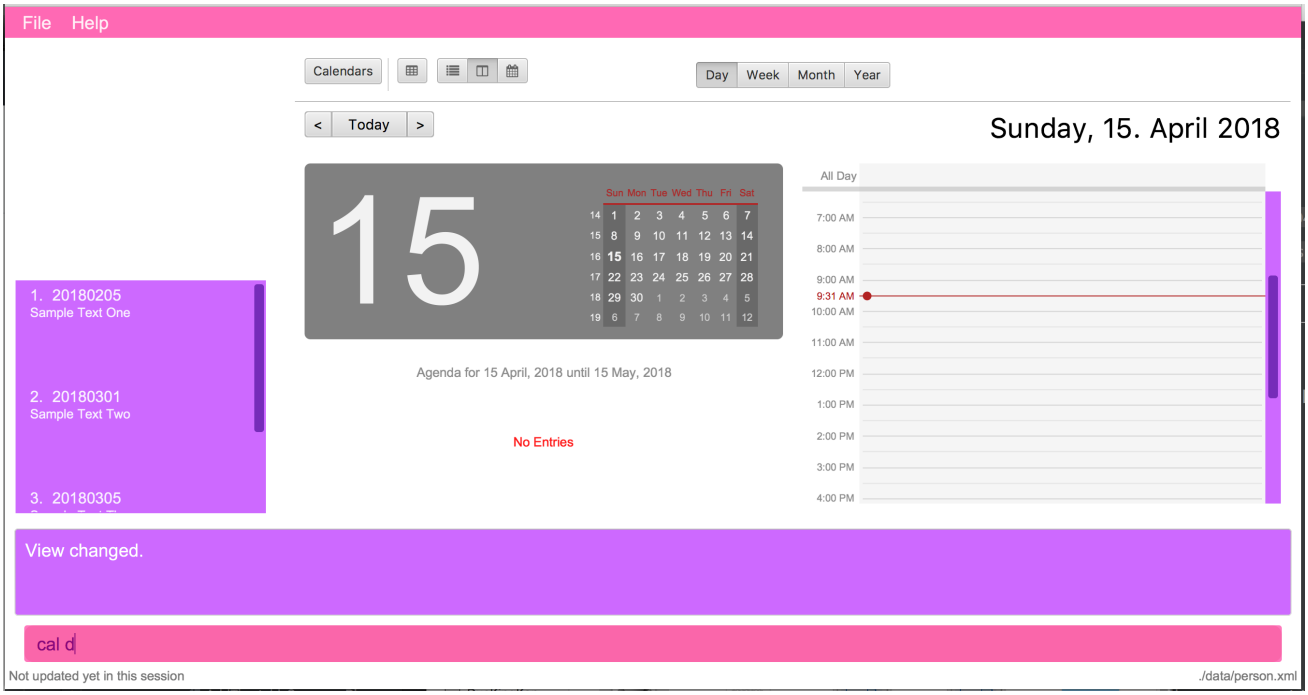
Calendar

When you want to check which are the available dates you are free: `Calendar`

View calendar function.
Format: `cview`
Alias: `cv`

Selecting Different Views
Day: `cal d`
Week: `cal w`
Month: `cal m`
Year: `cal y`

The image below shows a calendar view in day after you have entered `cal d` in the command box. From the calendar, you can see if there is any events on that day.



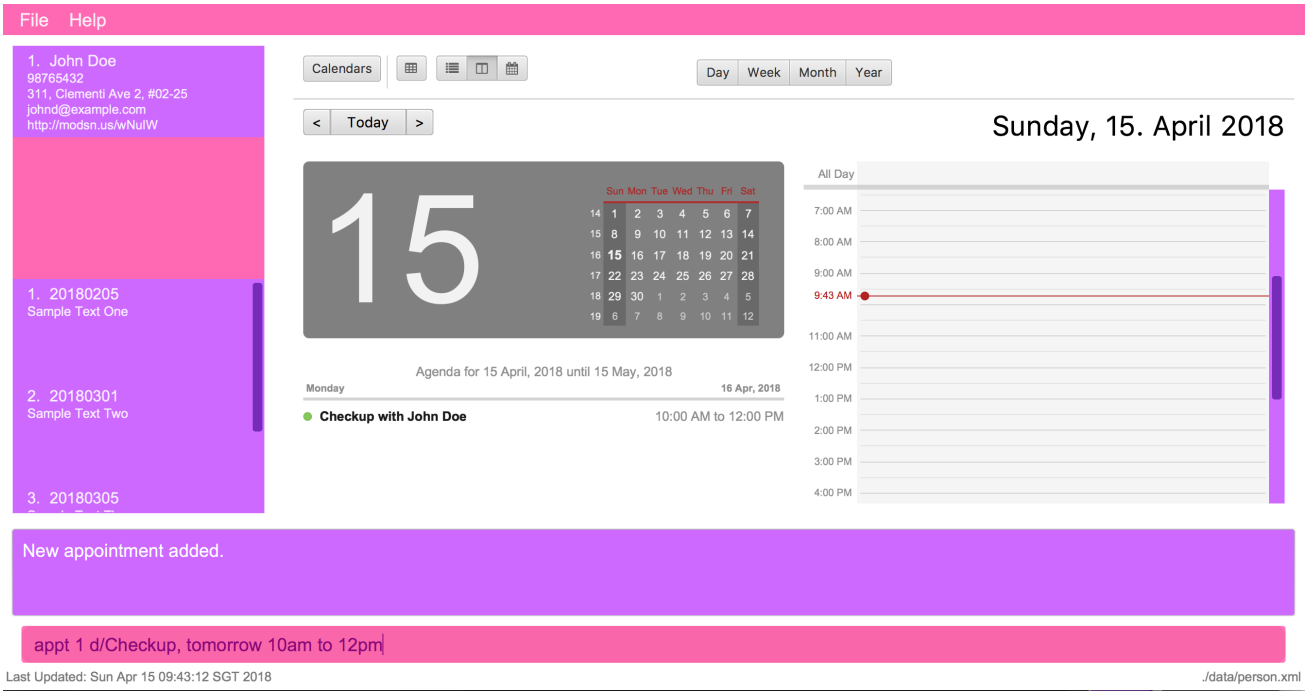
When you have a date with your partner then you decides to add the event to the Calendar after making sure that there are no clashes with your timetable nor schedules: `Appointment`

Adds new event to the calendar.
Format: `appointment`
Alias: `appt`

Following the Format:
To add an event, Description and Time parameters are COMPULSORY fields that are required to enter.
To add Description and time - `d/ + Description, Time`

Example: `appointment d/Lunch, Next Monday 3pm`

The image below shows a calendar view in day after you have entered `appt 1 d/Checkup, tomorrow 10am to 12pm` event. As you can see, the event is nicely populated on the calendar after you have successfully created a new event! Moreover, this Calendar feature allows you to enter multiple events too!! And the footer will show the time when you make changes to the Calendar!!



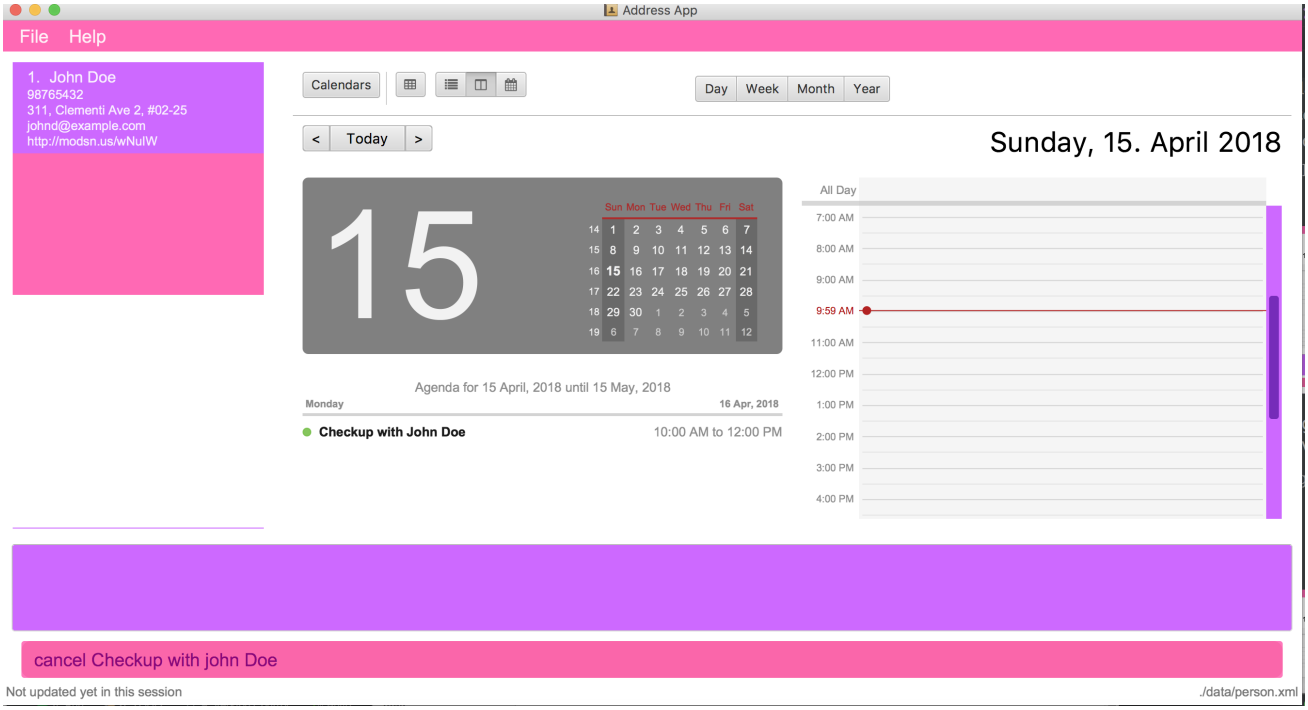
When your partner suddenly cannot make it on the scheduled event: `cancel`

Delete specified event from the calendar.
Format: `cancel`
Alias: Nil

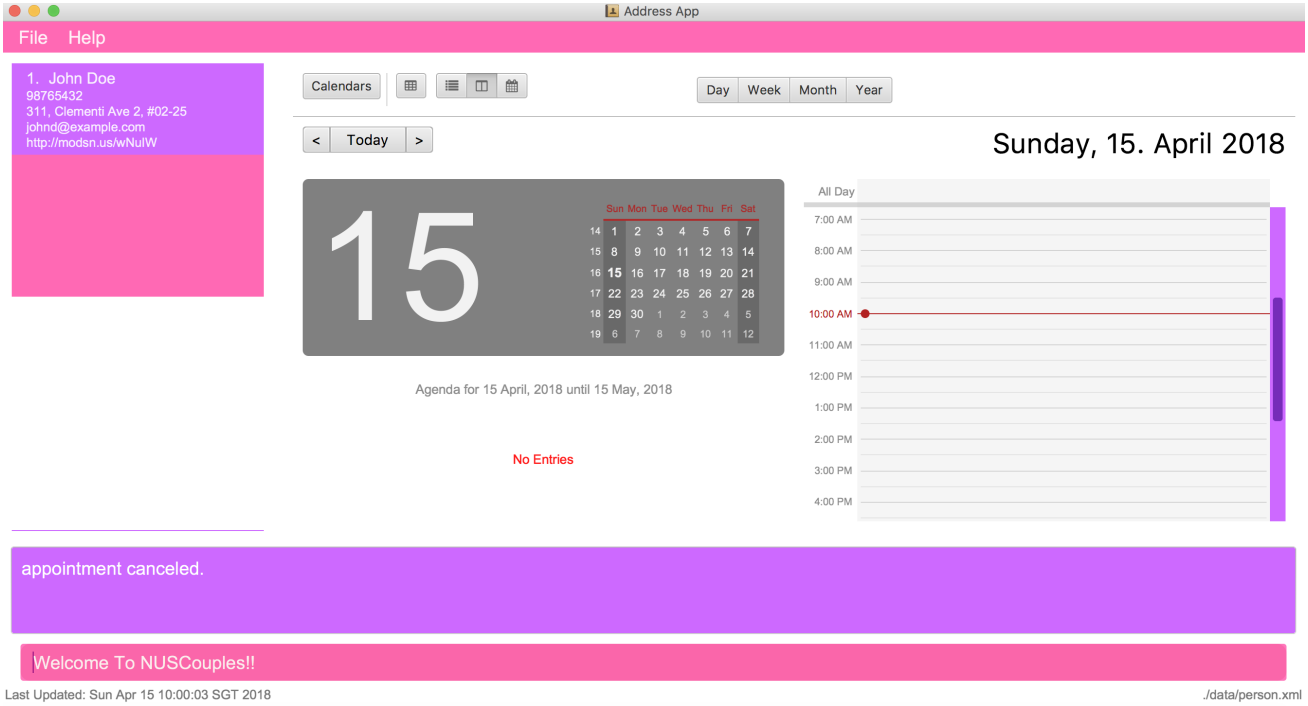
Following the Format:
To Cancel specified event: `Description with Person name`
Example: `cancel Lunch with John Doe`

The images below shows the before and after calendar view after you have executed the cancel appointment commands.
As you can see, after the commands are executed the footer will display the updated time as well and result panel will display `appointment canceled`. When there is no events on that day, the calendar will also display `no entries` too.

Screenshot: The Calendar View before you enter the Cancel Command



Screenshot: The Calendar View after you enter the Cancel Command



Accessibility

Unique KeyStrokes in NUSCouples.
Move Cursor to front: `Shift Ctrl`
Move Cursor to behind: `Shift Alt`
Move Cursor to behind (MAC USERS): `Shift Option`

Return to Table of Contents

Sync the calendar with Google API `[coming in v2.0]`

{explain how the user can authentic with google API and view and retrieve events from Google}

|calendar, cal |Viewing your current calendar |`calendar` | cal w

|appointment, appt |Adding a new event to your current calendar |`appointment` | appt 1 d/Checkup, tomorrow 10am to 12pm

|cancel |Deleting an event from your current calendar |`cancel` |

|exit, ex |Exits *NUSCouples* |`exit` |

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.

[Proposed] Calendar Google API Feature

Proposed Implementation (Appearing in V2.0)

The Calendar Viewer mechanism is facilitated by `Google Calendar API` and reside in the `ModelManager`. It supports viewing/add/editing/deleting capability that modifies the state of *NUSCouples*. Firstly, it uses OAuth 2.0 endpoints to allow users to share specific data with the application while keeping their usernames, passwords, and other information private. For example, an application can use OAuth 2.0 to obtain permission from users to store files in their Google Drives which sync to the calendar. This implementation requires the user to connect to the internet because *NUSCouples* needs to open the system browser and supply a local redirect URI to handle responses from Google's authorization server.

Basic steps

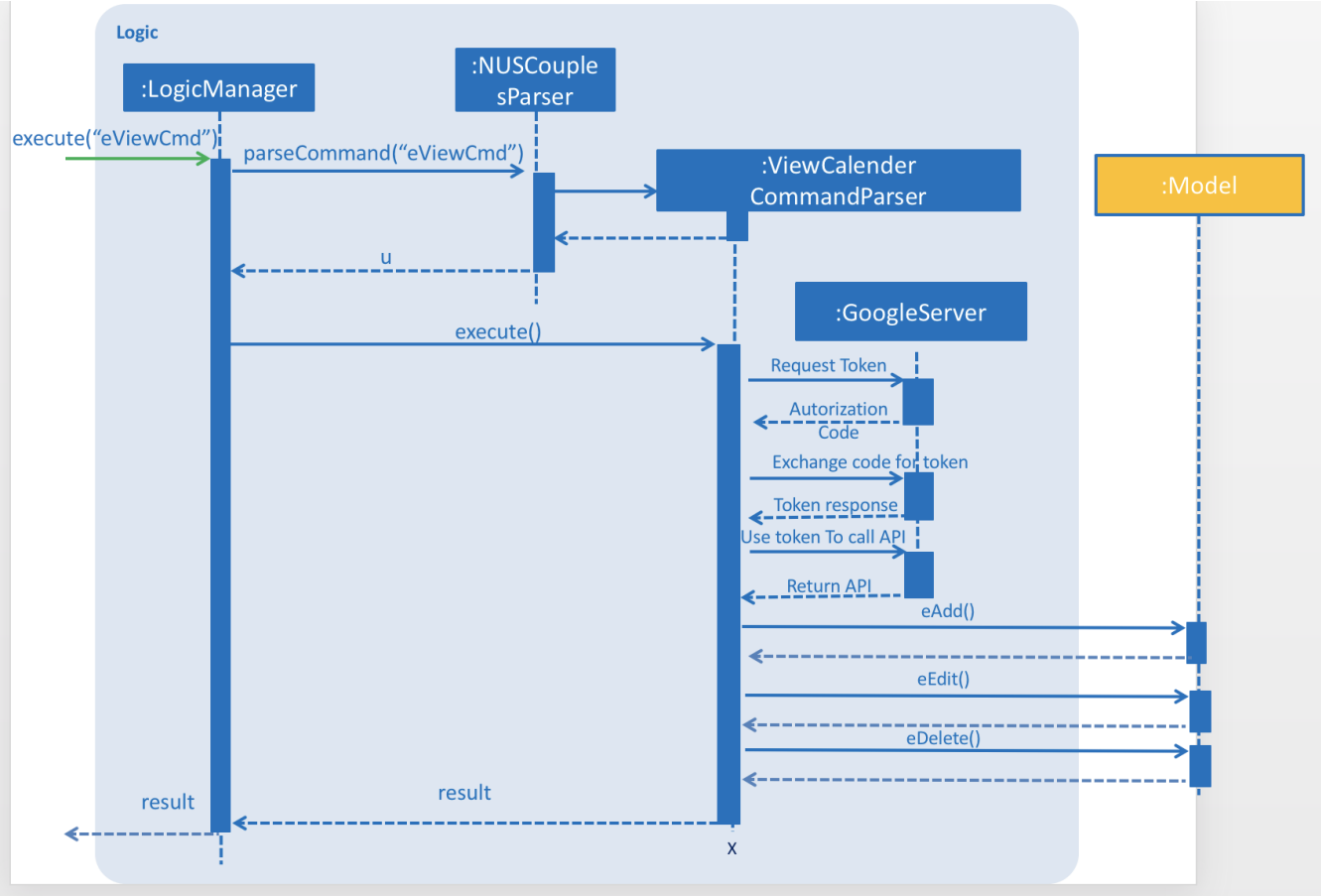
All applications follow a basic pattern when accessing a Google API using OAuth 2.0. At a high level, this are the four steps:

No.	Description
Step 1:	Obtain OAuth 2.0 credentials from the Google API Console. Visit the Google API Console to obtain OAuth 2.0 credentials such as a client ID and client secret that are known to both Google and your application. The set of values varies based on what type of application you are building. For example, a JavaScript application does not require a secret, but a web server application does.

No.	Description
Step 2:	Obtain an access token from the Google Authorization Server. Before your application can access private data using a Google API, it must obtain an access token that grants access to that API. A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the scope parameter. There are several ways to make this request, and they vary based on the type of application you are building. For example, a JavaScript application might request an access token using a browser redirect to Google, while an application installed on a device that has no browser uses web service requests. Some requests require an authentication step where the user logs in with their Google account. After logging in, the user is asked whether they are willing to grant the permissions that your application is requesting. This process is called user consent. If the user grants the permission, the Google Authorization Server sends your application an access token (or an authorization code that your application can use to obtain an access token). If the user does not grant the permission, the server returns an error. It is generally a best practice to request scopes incrementally, at the time access is required, rather than up front. For example, an app that wants to support purchases should not request Google Wallet access until the user presses the “buy” button; see Incremental authorization.
Step 3:	After an application obtains an access token, it sends the token to a Google API in an HTTP authorization header. It is possible to send tokens as URI query-string parameters, but we don’t recommend it, because URI parameters can end up in log files that are not completely secure. Also, it is good REST practice to avoid creating unnecessary URI parameter names. Access tokens are valid only for the set of operations and resources described in the scope of the token request. For example, if an access token is issued for the Google+ API, it does not grant access to the Google Contacts API. You can, however, send that access token to the Google+ API multiple times for similar operations.

No.	Description
Step 4:	Refresh the access token, if necessary. Access tokens have limited lifetimes. If your application needs access to a Google API beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows your application to obtain new access tokens

The sequence diagram below shows interactions within the `Logic` Component for Outh 2.0 endpoints:



Design Considerations

Aspect: Implementation of Google Calendar feature

- **Alternative 1 (current choice):** Data are not save locally.
  - Pros: User does not worry about getting data lost
  - Cons: User unable to retrieve the Calendar if internet is not connected
- **Alternative 2:** Save Data locally
  - Pros: User does not worry their Calendar is unable to connected to Google.
  - Cons: The latest Calendar events might not have been synchronized.

Aspect: Using Open-source or proprietary Calendar API

- **Alternative 1 (current choice):** Using Google API (open source)
    - Pros: I will learnt more even if I failed at the end of the project and Google API is more versatile
    - Cons: Tedious to implement it.
- **Alternative 2:** Using Restful API (proprietary)
    - Pros: Easier to implement due to everything is assisted.
    - Cons: Restrictive, need more money for more features to add on.

Return to Table of Contents

Calendar Viewer feature

Proposed Implementation

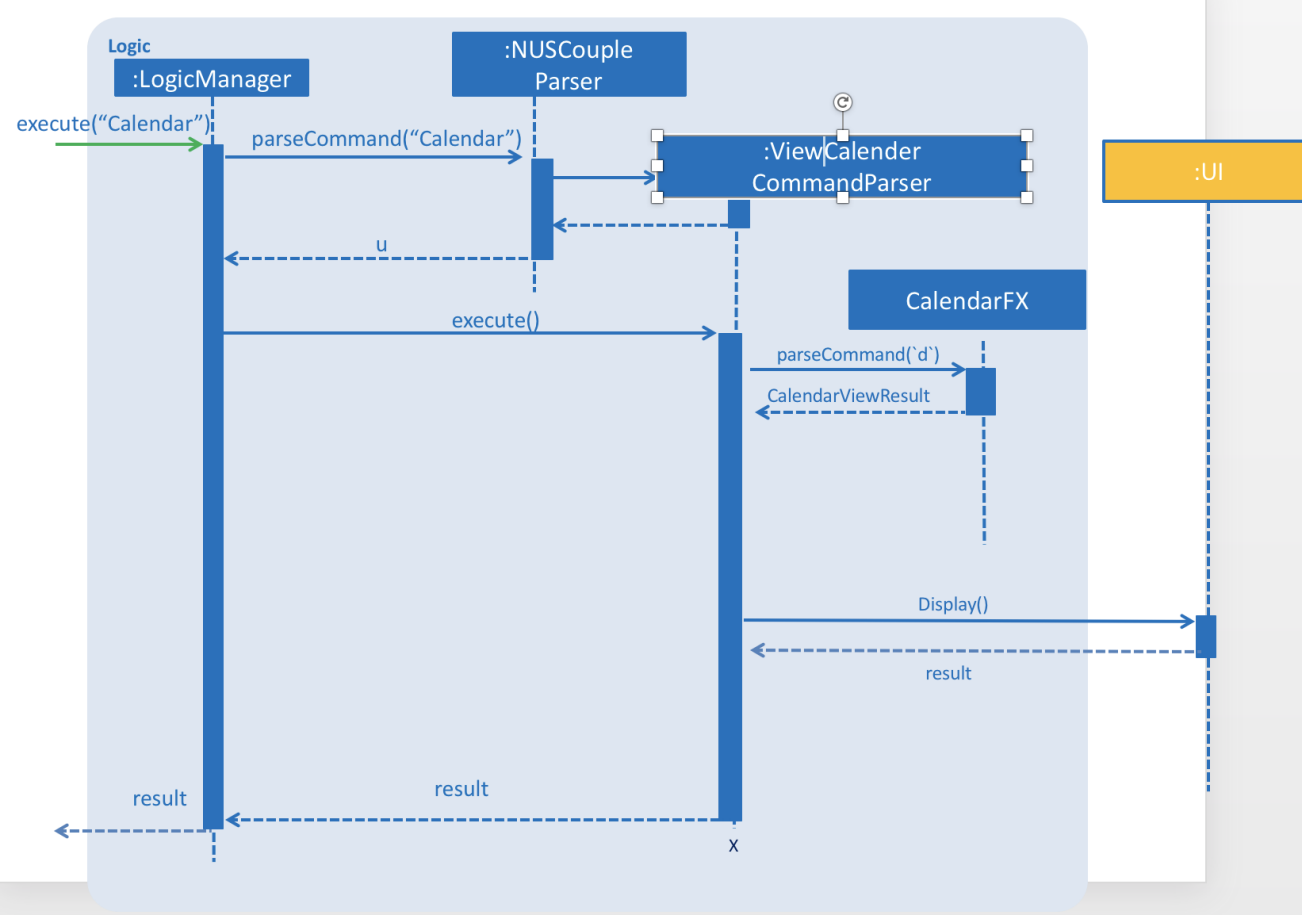
The Calendar Viewer feature is implemented by 'BrowserPanel', which will reside in 'UI'.

The idea of implementing this Calendar Viewer feature that sits ontop of Browser panel UI is to give the user a first look at the upcoming events first before going on to other features on the App.

Users are able to select different views such as in Days, Weeks, Months or Years by adding a prefix 'c', 'w', 'm', 'y' after adding 'Cal' or 'Calendar' behind.

This Calendar Interface is created and designed by CalendarFX. Through their GUI interface, i manipulate and massage the codes to allow user to enter commands to change the views since this is a CLI interface APP.

The sequence diagram below shows interactions within the `Logic` Component for the `execute' API call.



Design Considerations

Aspect: Implementation of Calendar View

- **Alternative 1 (current choice):** Display only current month event.
  - Pros: Easier to implement and Neater rather than displaying more than 1 mth.
  - Cons: Need to input cmd to filter through other month.
- **Alternative 2:** Don't display any month until user defines.
  - Pros: More interaction.
  - Cons: The UI will be blank at initial stage which is ugly.

Aspect: Calendar View performance

- **Alternative 1 (current choice):** Requires RAM of at least 6GB and above.
  - Pros: Faster retrieval and display the Calendar out
  - Cons: Need to buy more memory
- **Alternative 2:** Don't display Year.
  - Pros: Reduce latency
  - Cons: User can't add event to next year

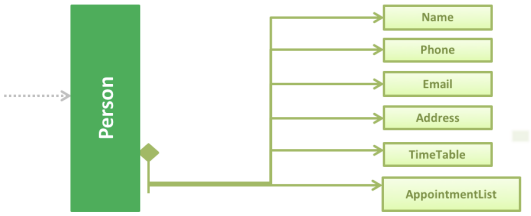
Calendar Add Appointment feature

Proposed Implementation

The Calendar Add Appointment feature is implemented by 'Appointment', which will reside in 'Model'.

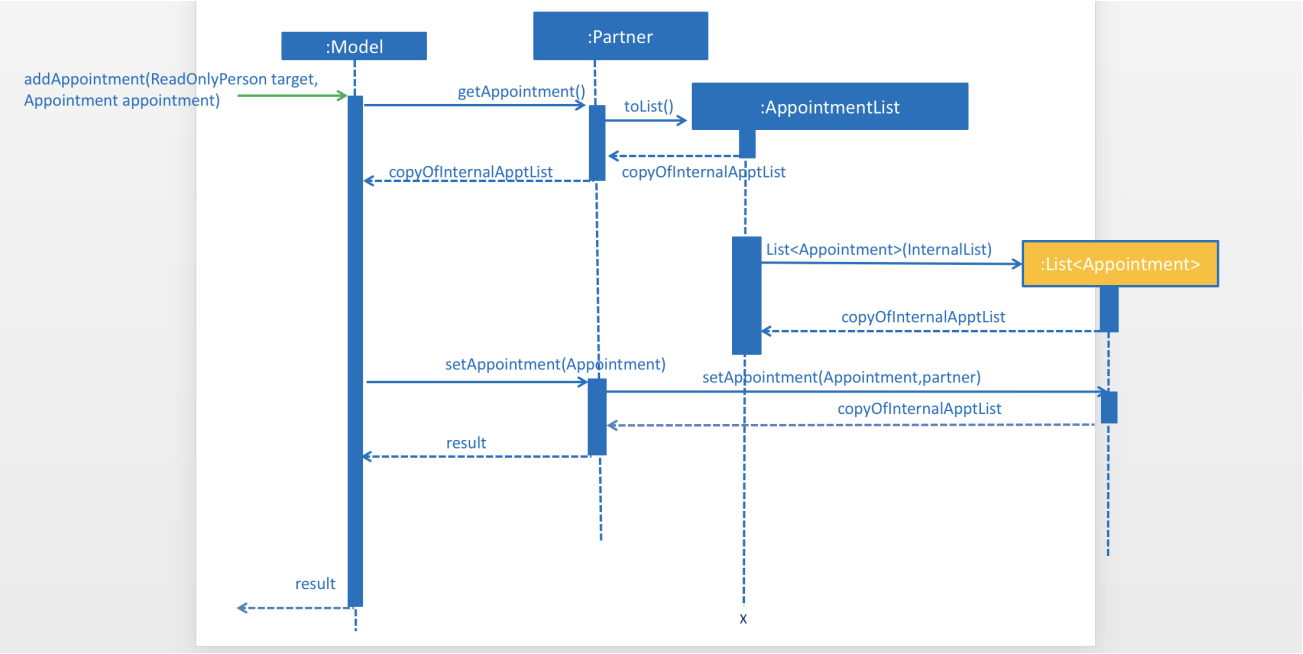
The idea of implementing Add Appointment feature is to allow the user to add his/her event on the Calendar.

The AppointmentList class holds an internal list that holds the Appointment class. Appointments are made up of 3 internal variables.



- Description: It Holds a string about the appointment. This String will be used to identify which Appointment.
- Start Date: It Holds the starting time of the appointment. Used for sorting appointments and UI.
- End Date: Holds the end time of the appointment. Used for UI.

The sequence diagram below shows interactions within the `Model` Component for the `AddAppointment' API call.



Design Considerations

Aspect: Implementation of Add Event View

- **Alternative 1 (current choice):** Display in Calendar UI.
  - Pros: Easier to implement and Neater.
  - Cons: Doesn't display all events listed on the partner.
- **Alternative 2:** Create a List to display all events.
  - Pros: User can have an overview of all the events listed
  - Cons: The UI will be blank at initial stage which is ugly.

Aspect: Allow duplicate events

- **Alternative 1 (current choice):** On same time of the same day.
  - Pros: User can plan which one is more important to attend
  - Cons: Ambiguous as there are multiple similar events on the same time frame
- **Alternative 2:** Restrict user to add same event on the same time frame.
  - Pros: More neat looking
  - Cons: User cant compare or manage well

Calendar Cancel Appointment feature

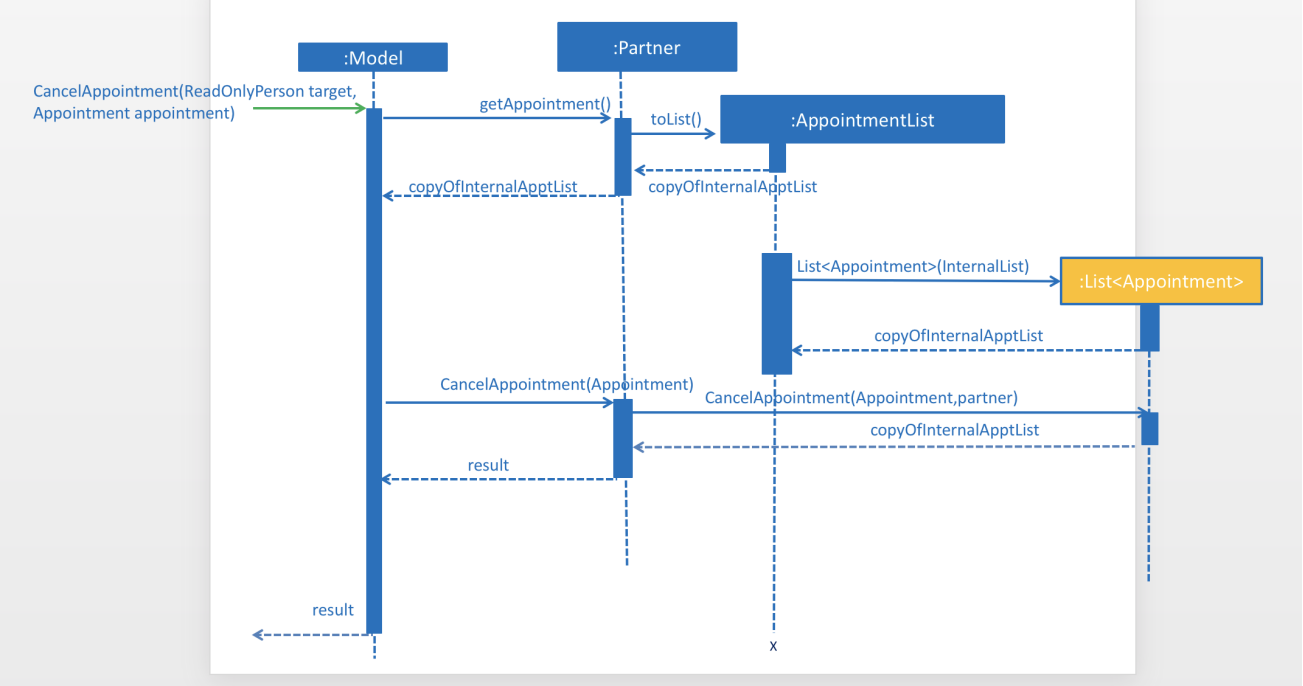
Proposed Implementation

The Calendar Cancel Appointment feature is implemented by 'Appointment', which will reside in 'Model'.

The idea of implementing a Cancel Appointment feature is to allow the user to remove his/her event on the Calendar when needed.

When the user calls to cancel an appointment in the Calendar, the model will ask AppointmentList to return a list of Appointments. However, the AppointmentList will only return a mutable copy of the AppointmentList back to the caller, as a practice of Defensive Programming.

The sequence diagram below shows interactions within the `Model` Component for the `CancelAppointment' API call.



Design Considerations

Aspect: Implementation of Add Event View

- **Alternative 1 (current choice):** Display in Calendar UI.
  - Pros: Easier to implement and Neater.
  - Cons: Have to cancel one by one and thus calendar will refresh over and over again.
- **Alternative 2:** Create a List to display all events.
  - Pros: User can have an overview of all the events listed to cancel
  - Cons: The UI will be blank at initial stage which is ugly.

|Chen Xing |Scheduler: This app allows user to schedule//delete/view planned meetings |Accessibility: Reduce the effort when user enters command on the command box through custom keystrokes.
Tracker: The system will update the time on the footer to show when the changes have been made.

Appendix A: Use Cases

(For all use cases below, the **System** is the `NUSCouples` app and the **Actor** is the `user`, unless specified otherwise)

Use case: Authenticate User with Google

MSS

1. User are required to generate and download their credential from google API credentials: Google Dashboard and import into project resource directory.
2. NusCouples use the credential to authenticate with Google API using Auth2.0.

Use case ends.

Extensions

1a. The partner already has an existing google calendar hosted in google.

1a1. NUSCouples redirects to google calendar account to authenticate using the user credential.

1a2. User confirms change.

Use case resumes at step 2.

1b. The given credential is invalid.

1b1. NUSCouples shows an error message and close.

Use case ends.

Use case: View Calendar of User

MSS

1.User inside browser panel enters the command to view calendar at different view(s). 2.NUSCouples displays the Calendar

+ Use case ends.

Extensions

1a. There is an existing user in NUSCouples.

1a1. *NUSCouples* request to display Calendar of day/week/month/year from CalendarFX.

1a2. *NUSCouples* populate the calendar on the browser panel

Use case resumes at step 2.

Use case ends.

Use case: Add Event on Calendar

MSS

1.User inside browser panel enters the command to add appointment. 2.NUSCouples update the appointment list. 3.NUSCouples displays the Calendar of all the events in the appointment list and update the change of event on the footer.

Use case ends.

Extensions

1a. User didn’t specify date of event

1a1. *NUSCouples* shows an error

2a. NUSCouples add the new appointment into the list

Use case ends.

Use case: Delete Event on Calendar

MSS

1.User inside browser panel enters the command to delete appointment. 2.NUSCouples update the appointment list. 3.NUSCouples displays the Calendar of all the events in the appointment list.

Use case ends.

Extensions

1a. User didn't specify date of event

1a1. *NUSCouples* shows an error

2a. NUSCouples cannot find the appointment into the list

2a1. NUSCouples displays error message

Use case ends.

Use case: Keyboard Accessibility

MSS

1.User enters a long command but want's to add something at the front or back 2.User press Shift Ctrl (move cursor to the front) Shift Alt (move cursor to the back)

Use case ends.

Extensions

1a. User is using MAC

1a1. enter `option` key instead of `Alt` key

2a. User doesn't have keystroke Shift/Alt/Ctrl.

2a1. This feature can't be used

Use case ends.

PROJECT: PowerPointLabs

{Optionally, you may include other projects in your portfolio.}
