Philip Alexander Boediman - Project Portfolio

PROJECT: MyProject

Background

Our experiences as Computer Science (CS) students who are involved in a lot of projects have motivated us, a team of 5 NUS SoC students, to develop ***MyProject*** - an all-in-one project management application that helps the user manage his/her projects' administrative stuff in a single application. For students, by students, ***MyProject*** mainly targets team leaders w\ who are responsible for directing and keeping track of the projects as well as students who are involved in a lot of projects in general.

Overview

As a start, we were given the source code for a fully functional Addressbook application. The task of our project was to enhance this Addressbook into something useful that can help a particular group of people.

MyProject has various features to aid team leaders in project management. Some of the more prominent features include:

- Keep track of members' performance
- Generate meeting timings
- Keep track of project’s finances
- Record meeting minutes
- Keep track of ongoing and upcoming tasks
- Send out reminders for upcoming meetings and tasks
- Broadcast email for important announcements (My Contributions)

My main role was to design email related features. The following sections illustrate the enhancements that I have made as well as the relevant documentation that I have added to the user and developer guides in relation to these enhancements.

Summary of contributions

This section shows a summary of my coding, documentation and other helpful contribution towards the team project.

Credits to https://javaee.github.io/javamail/ for providing the JavaMail library and the javax.Mail

API. === Code Contributed: Please click the link to see my sample code: Functional code

Major Enhancement - Email feature

Enhancements Implemented:

- `signIn` command
  - lets user signs in using his/her Google account.
- `logOut` command
  - lets user signs out from his/her Google account.
- `sendMail` command
  - sends a personal email to one member in the project / contact list.
- `broadcastMail` command
  - sends a personalised broadcast message to all members in the current project.
- `sendReminder`
  - sends a reminder to all members in the project of the upcoming meeting and tasks which are due in specific number of days defined by user.

Justification

Team leaders often have information that they want to convey to the team members about updates or changes made to the project or remind team members of the upcoming meeting or tasks. There might be several applications in which the project groups are formed in the social media platforms. Team leaders may have to switch between different applications to locate the project group in this platform and this process is oftentimes time-consuming. With this enhancements, team leaders will be able to relay his message to his members in a single application, ***MyProject***, where all the informations on his various projects are stored.

Highlights

These enhancements are extremely useful and use very intuitive command formats which are very similar to the normal GUI that user normally encounters and are very easy to remember. Some of the challenges faced were in understanding how the javax.Mail works which was outside the coverages of the course. These enhancements are extendable as we can further enhance to support more email domains which users wish to use as well as send attachments. Overall, the enhancements can support all the basic one-way mailing features which are sufficient for the optimisation of the team leaders' works.

Secondary Enhancement - Meeting attribute in the project.

Enhancements Implemented:

- `AddProjectMeeting` command
  - lets user adds a meeting to .
- `DeleteProjectMeeting` command
  - lets user signs out from his/her Google account.

Justification

Every projects will definitely have meetings, thus team leaders can easily keep track of upcoming meetings in the projects that he/she has.

Highlights

This enhancements is extremely useful as meeting is a very crucial part in a project's progress. Meetings added by the user will automatically be sorted based on the meeting time (earliest to latest).

Other contributions

- Project Management
  - Tested team members' features and reported bugs for amendment.
  - Implemented Time class which are widely used in the project. #31]
  - Updated the Test classes so that it fits our current model. #201
- Documentation
  - Proof-reading the project's User Guide and Developer Guide and made changes to it that suit our Project.
  - Helped the team update some of the documentations #222 #189

Contributions to the User Guide

We had to provide a user guide for users to familiarise themselves with our application. The user guide consists of easy navigation to certain parts of the user guide, features implemented in our application and a summary of commands available in our application. Below is the section of user guide that I contributed for my features.

Project Meeting

Add a project meeting: `addProjectMeeting` [Checkout]

Adds a new project meeting to the current working project.
Format: `Format: addProjectMeeting [c/ dd/MM/yyyy HHmm] [s/MEETING_DESCRIPTION]`

`dd/MM/yyyy HHmm` refers to the date and time the meeting is to be held.
`MEETING_DESCRIPTION` refers to the purpose of the meeting.

Example:
Let's say that you plan to have a meeting on the 19th November 2019, you want to keep track of this meeting by recording it down in the MyProject app.

To add a project meeting:

1. Type `addProjectMeeting c/19/11/2019 1300 s/DIscussion on version 2` and press enter to execute it.Note that this project meeting will be held after the 4th meeting on 16/11/2019 1700 and before the 5th meeting on 29/11/2019 1300. [addProjectMeetingPPP1]
2. Success message with the respective information about the meeting will be displayed. The meeting added will be automatically sorted according to the dates and times in ascending order. Thus, the new meeting added will be placed as number 5 and the previous meeting with number 5 will be move to number 6.

Email Features

Tired of switching between applications? We got you covered, below you will find some commands which support sending emails right here within the application.

NOTE	Do keep in mind that the user Account in our current version is only compatible with Gmail Account and please ensure that the Access to less secure app in the security setting is enabled before signing in!

Sign in to your account: `signIn`

Signs in to the your email account.
Format: `signIn ac/ACCOUNT_EMAIL_ADDRESS pa/PASSWORD`

`ACCOUNT_EMAIL_ADDRESS` refers to the sender's/user's email address. `PASSWORD` refers to the password to the sender's/user's email address.

Example:

- `signIn ac/example@gmail.com pa/12345678`

IMPORTANT	Please Turn on the access to less secure app in your account's security setting.

NOTE	This command is required to be executed before the remainder of the email commands can be executed. The correctness of the email address used and the password will be checked.

Send reminder: `sendReminder`

Prerequisites: Checkout to a project using the checkout command.

Sends a reminder to all members from the current working project of the upcoming Meeting and Task that is due.
Format: `sendReminder d/DURATION`

`DURATION` is the number of days from the current time within which the Tasks are due and The Meetings are held.

Example:
Let's say that you have several meetings and task for the coming week, and you want to remind your group of the upcoming meetings and the tasks that are supposed to be done by the week.

Instead of typing all the meetings and tasks for the coming week all over again, you can just easily send these lists to their email addresses from the application.

To send reminder:

1. Type `sendReminder d/7` and press enter to execute it.Note that the date at the time this screenshot was taken is 10/11/2019, thus, only meeting meetings 2-4 and task 1-3 will be sent as reminders to the members. [sendReminderPPP1]
2. A success message will be displayed in the box saying "Reminders have been sent!"
3. Members will receive an email with the tasks due and meetings happening within the next 7 days.

NOTE	We will not check the correctness of the members' email addresses

Contributions to the Developer Guide

Below is the section of developer guide that I contributed for my features. They showcase the complexity of my features as well as my thought process while trying to implement the features.

Meeting feature

Description of feature

Within every project, there are meetings to be held at certain time. The diagram below shows the class diagram of the meeting class and how it interacts with the model.

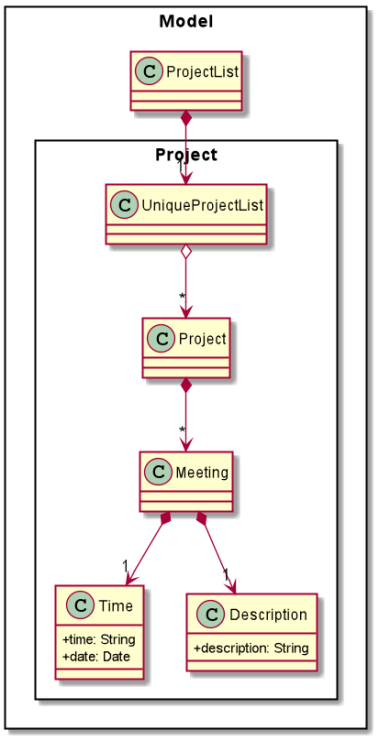

Figure 1. Class Diagram for Meeting

As shown above, each project stores multiple meetings in a list. These meetings are automatically sorted based on the time in ascending order. Here are some meeting related commands that can be executed by the app.

1. addProjectMeeting - adds a meeting into the project model.
2. deleteProjectMeeting- removes a meeting from the project model based on index specified by user

Details

As seen in figure 1, each meeting consists of 2 parameters namely description and time to show what is the meeting about and the date and time of the meeting respectively.

Sending Reminder feature

Description of feature

Sends reminder for tasks and meetings that are due in the number of days given by the user input.

The basic implementation uses javax.Mail to send email to other email addresses. The Mailer class has static method sendEmail which is responsible for sending all kinds of email to a given recipient(s). User's email account information is obtained from the Model class to send the emails. Currently, only gmail server has been made available for use in sending the emails.

NOTE	User Email Account Information is stored through the signIn command.

Details

SendReminder takes in a single integer as parameter. The integer will be the duration in days from the current times in which the meetings and tasks are due.

The following sequence diagram shows the process of sending reminder to the project members.

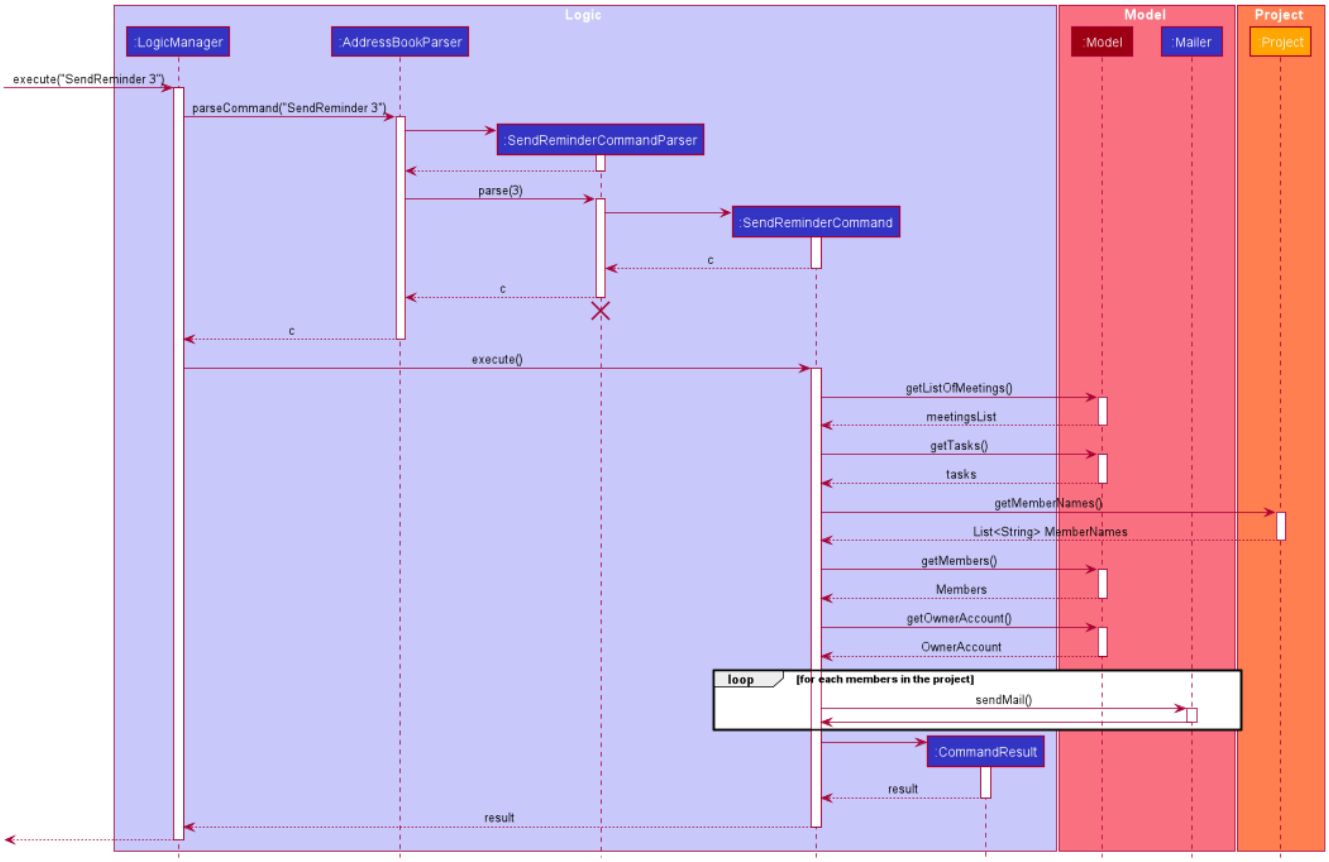

Figure 2. Sequence Diagram for sendReminder

These are the stages shown in figure 2.

1. Parses the input to obtain the duration.
2. Goes to `Model` to get the the Meetings and Tasks.
3. Obtains list of tasks and meetings which are due in the duration time.
4. Goes to `Project` to get the Members names.
5. Goes to `Model` to get the members in the project.
6. Goes to `Model` to get the OwnerAccount information.
7. Sends email to all the members about the upcoming meetings and tasks in the project.
8. Display the success message.

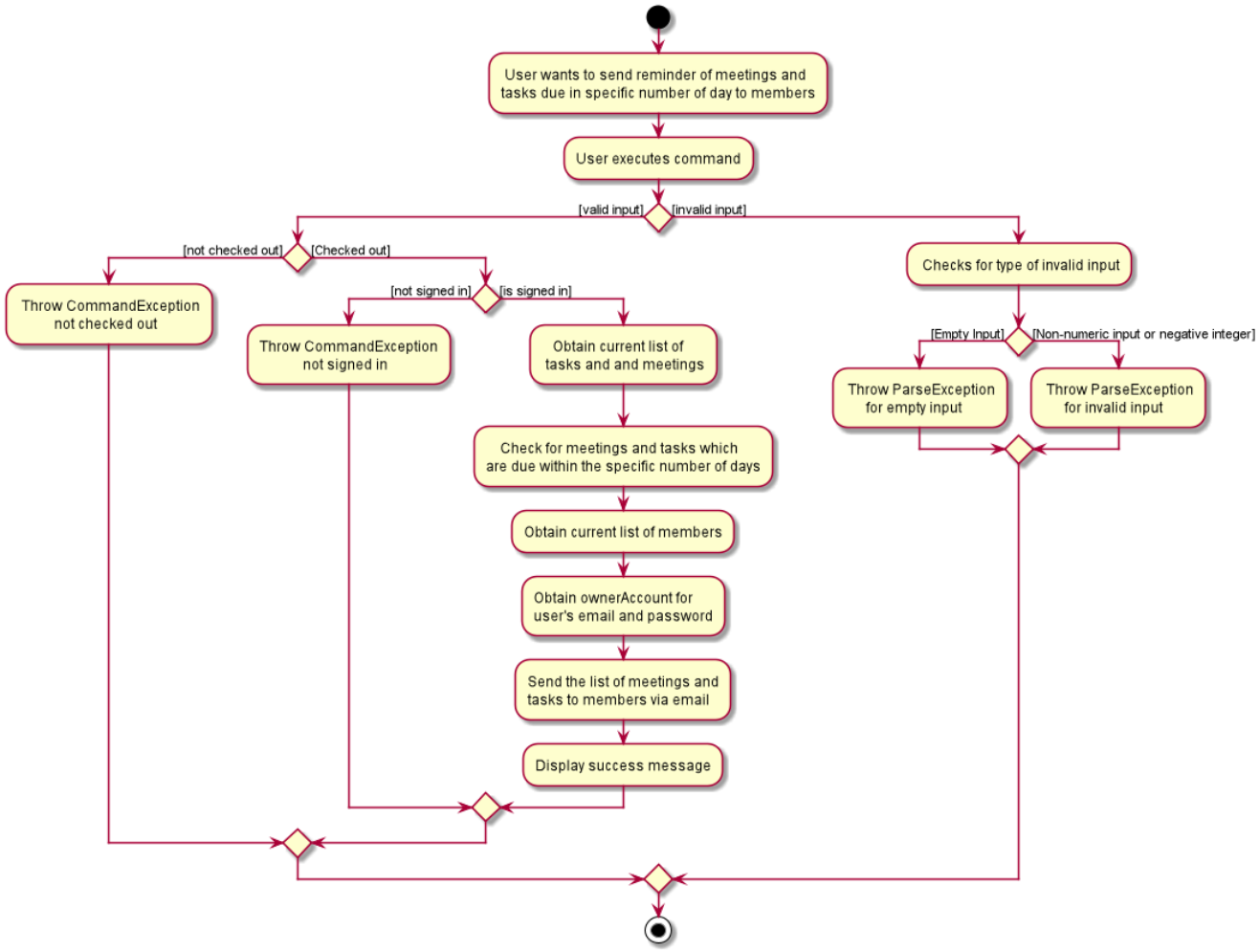

Figure 3. Activity Diagram for sendReminder

The diagram above shows how `sendReminder` works. There are 2 possible error messages for invalid input. Firstly, if the user inputs nothing as the duration and input cannot be empty error message is shown. Secondly, an error will be shown if the user non-numeric or negative or zero number as the duration of time. For a valid input, user will be required to checkout then signIn first before executing the command.

Design considerations

Aspect: Data structure to support the sendReminder commands

- **Alternative 1:** Storing List<Person> in the Project for members.
  - Pros: Easy to implement. Do not need to look for the person object in the addressBook from the List of String of members' names in the project.
  - Cons: Introduces coupling and may cause unwanted bugs due to cyclic dependencies as Project contains Person and Person contains Project.
- **Alternative 2:** Stores members as List<String> of members names.
  - Pros: Reduces coupling and and eliminates cyclic dependencies between Project and Person object..
  - Cons: Have to hash the members object in the addressBook by names and go through the List<String> of members' names one by one to get the Person object of the member. Harder

to implement.

Sign In feature

Description of feature

Signs in using a Google account for mailing purposes.

The basic implementation uses javax.Mail to check for the validity of the email address and password. User's email account information is stored in the Model class through this command.

NOTE	In this version, only gmail server has been made available for use in signing in and sending emails.

Details

signIn takes in two inputs, first is the email address of the user (in gmail), second is the password to the email address.

The following sequence diagram shows the process of signing in to the user's account.

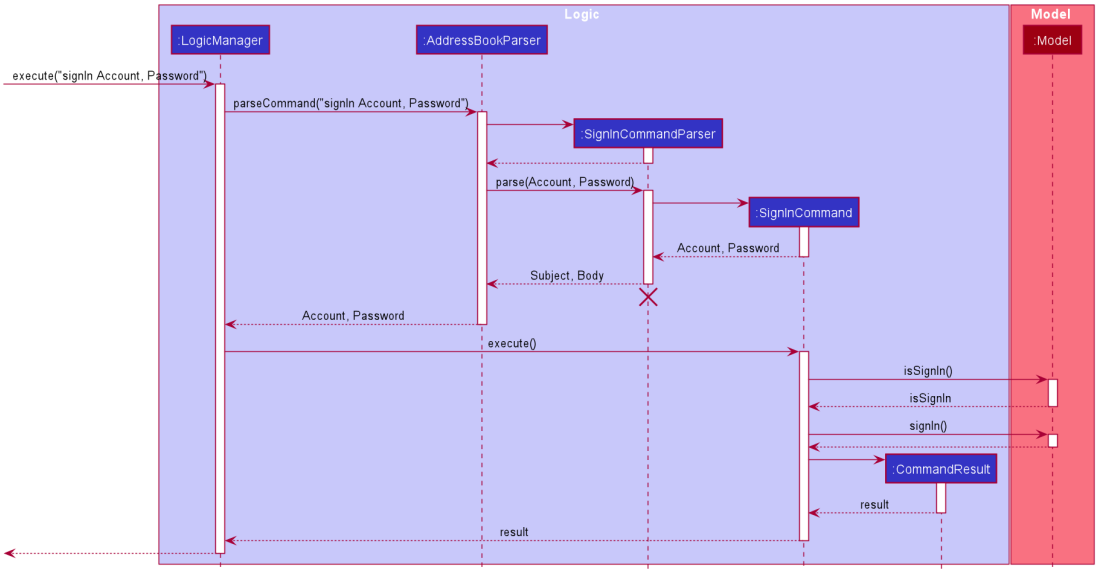

Figure 4. Sequence Diagram for signIn command

These are the stages shown in figure 4.

1. Parses the input to obtain the Email address and Password.
2. Goes to `Model` to check if user has signed in.
3. Creates OwnerAccount with the address and password.
4. Goes to `Model` to store the Email address and Password as `OwnerAccount`.
5. Display the success message.

Design considerations

Aspect: Eliminating signIn command

- **Alternative 1 (current choice):** Lets user Signs In using their own Gmail Account.
    - Pros: Allows users to user their own Gmail Account and lets members(recipients) know email sent by the team leaders.
    - Cons: Expose users' password when users are trying to sign in. Users will need to change the account security settings to let less secure app access the account. Compromises account security.
- **Alternative 2:** Hardcode an account in the application that is responsible for all mailing command.
    - Pros: Do not expose users' password and Users do not need to change their Account security settings to send the emails.
    - Cons: Members(recipient) receiving email from the users' may not know the origin of the email. This may lead to confusion for the project’s members.