Zhao Jun Ru - Project Portfolio

PROJECT: VolunCHeer V1.3

Overview

VolunCheer is a desktop application for project managers who wish to keep track of their ongoing / upcoming projects as well as their beneficiary and volunteer pool.

Summary of contributions

- **Major enhancement: implemented the `Project` class whereby it creates a new `Project` object to the VolunCHeer application**
  - What it does: the `Project` class is implemented with relevant functions that ties the Beneficiary pool and Volunteer pool together
  - Features involved: the relevant features include basic functions such as `addProject`, `deleteProject` and `listProject`. The added features are the `complete` feature and the `assign` feature which will be elaborated in further details.
  - Justification: The Project class forms the basis of the VolunCHeer application whereby the aim is to manage and organise projects better and save users form the hustle of paper works.
  - Highlights: The methods implemented involves interactions between the 3 objects namely `Project` `Beneficiary` and `Volunteer`. It provides user with the ability to add and delete `Projects`, assign the specific `Beneficiary` and a list of `Volunteers` to `Project` of interest as well as setting projects status as complete to keep track of ongoing and completed `Projects`.
  - Explanation of features implemented
    1. Assign Beneficiary feature **allows the user to assign a existing `Beneficiary` to the `Project` selected**
       - Justification: A `Beneficiary` is usually attached to a `Project` and the beneficiary details are very important and usually tracked in a separate document / file. Thus we decide to keep Beneficiary information as a separate list and assigned to the project when required such that it can be managed separately.
       - Highlights: This enhancement requires data from both the project list as well as the beneficiary list, there was thus some difficulty trying to implement the command. Many adjustments had to be made to keep the project class and beneficiary class independent and yet synchronised.
    2. Assign Volunteer feature **allows the user to assign a Required number of `Volunteers` to the `Project` selected**
       - Justification: Projects usually require varying number of volunteers, we thus decide to make it flexible by allowing user to key in the required number themselves. Furthermore, should there be any requirement for the `Volunteers` they can filter the `Volunteer` list with our `Map Command` and `Sort` function.

3. Complete project **allows the user to mark a project as completed in the project list** this is indicated by a colour change - to red for the project title.
   - Justification: This is thought to be a useful feature as project managers would want to keep track of what are the ongoing projects while still have a copy of their completed projects.
   - Highlights: This enhancement indicates a change of status of the specified `project`, it is a simple yet useful feature for the user.

- **Minor enhancement**: ListProject and DeleteProject commands are inherited from addressBook and successfully implemented on the project class.
- **Other contributions**:
  - Project management:
    - Manage project submissions and deadlines.
    - Managed setting up of Milestones v1.2 v1.3 and v1.4 #42
    - Managed releases `v1.3` and `v1.4` on GitHub #73 #122 #123
    - Setting up of issues to track progress #36 #74 #76 #110 #119 #120
    - Managed overall merging and integration of project + solve bugs raised. #141
  - Documentation:
    - Did the UserGuide for v1.1: #6
    - Subsequent updates of documents on individual features.
  - Tools:
    - Integrated Ruby + asciidoctor for pdf releases of documentations as well as easy editing of adoc.

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.

Project Management

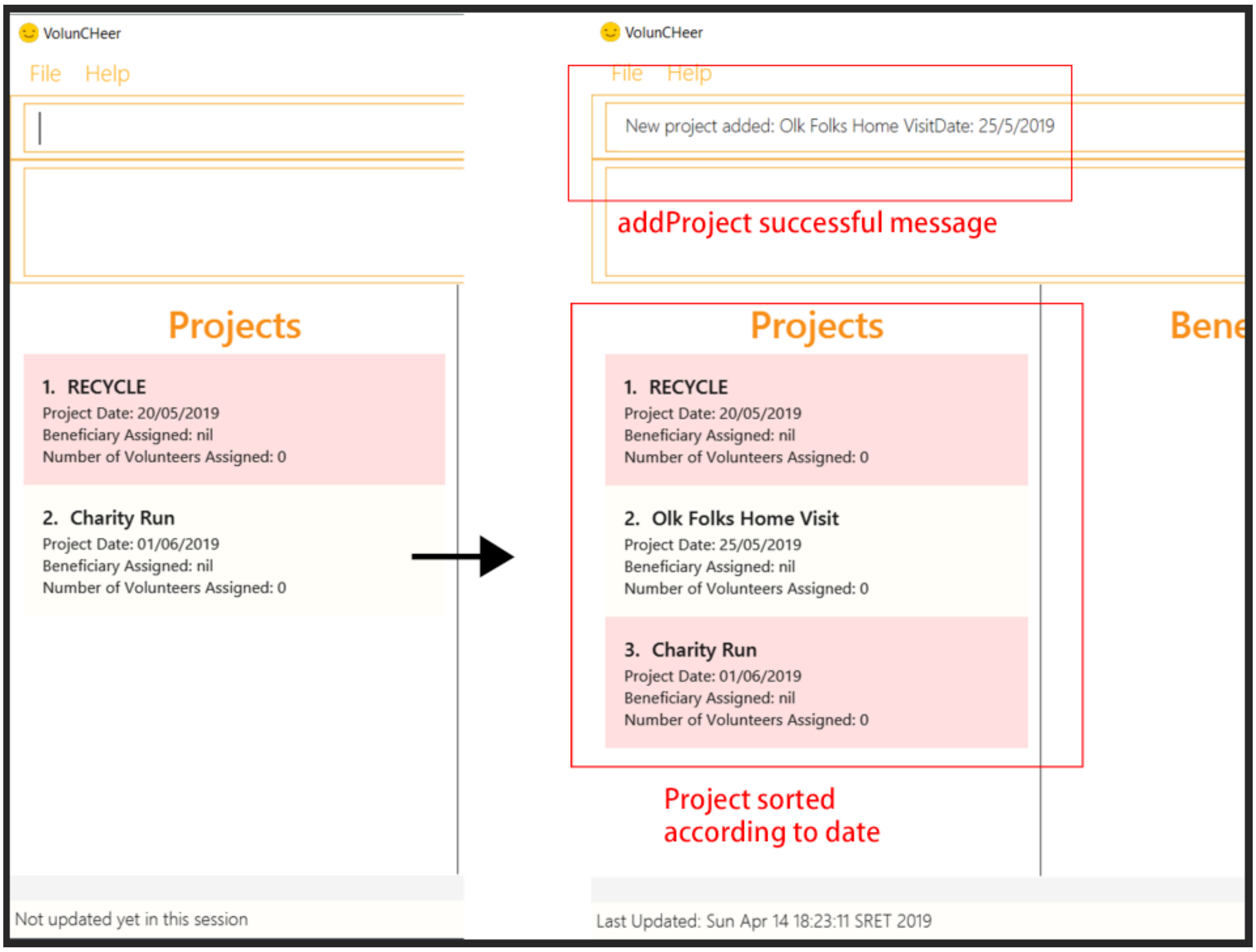
Adding a project: `addProject` / `ap`

One of the first things to do when you use the app is to start adding projects to track, and this is the command to use.

Format: `addProject n/PROJECT_NAME d/DATE`

- Please enter DATE in DD/MM/YYYY format, making sure that the date should be after today.
- Project list does not accept duplicates in Project Titles, so make sure you name everything differently!
- Projects are automatically sorted in ascending date order for easier tracking or Project tasks.

Now let us look at what happens when the command `addProject p/Old Folk Home Visit d/25/05/2019` is entered on screen.



VolunCHeer
File Help
Projects
VolunCHeer
File Help
addProject successful message
Projects
Bene
Project sorted according to date

New project added: Olk Folks Home VisitDate: 25/5/2019

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Charity Run
Project Date: 01/06/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

Not updated yet in this session

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Olk Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

3. Charity Run
Project Date: 01/06/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

Last Updated: Sun Apr 14 18:23:11 SRET 2019

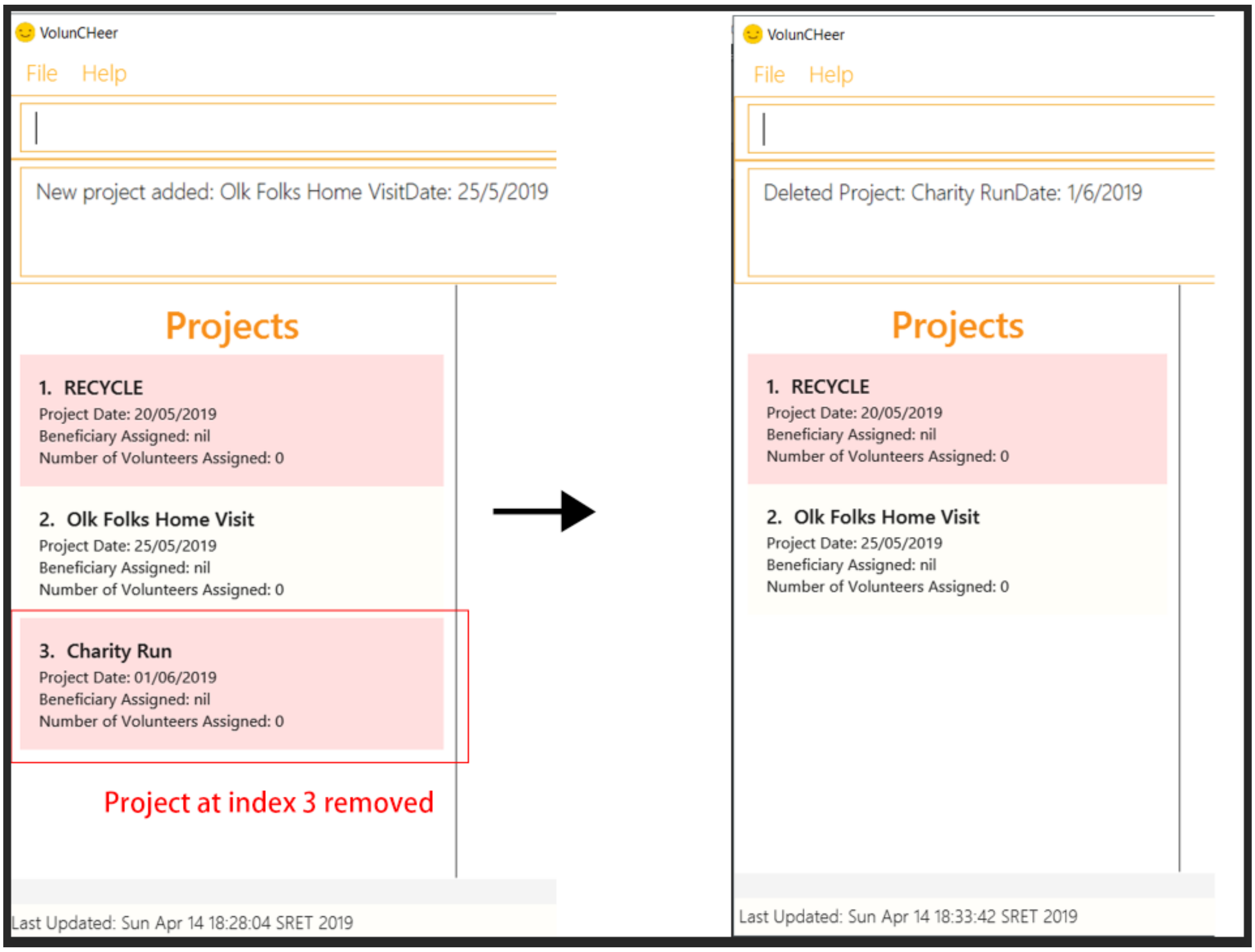
Figure 1. When `addProject p/Old Folk Home Visit d/25/05/2019` *is executed.*

Deleting a project : `deleteProject` / `dp`

When a project is completed or cancelled, VolunCHeer allows you to easily delete it by stating the project order in the list.
Format: `deleteProject INDEX`

- This INDEX refers to the index of the project in the project list. If you are unsure of the order, **PLEASE** use 'listProject' to view all projects and get the correct index. If you delete the wrong projects, please refer to [undo].
- Error message is shown if the INDEX entered is invalid

This is how the project list changes upon execution.


VolunCHeer

File Help

New project added: Olk Folks Home VisitDate: 25/5/2019

Projects

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Olk Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

3. Charity Run
Project Date: 01/06/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

Project at index 3 removed

Last Updated: Sun Apr 14 18:28:04 SRET 2019

VolunCHeer

File Help

Deleted Project: Charity RunDate: 1/6/2019

Projects

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Olk Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

Last Updated: Sun Apr 14 18:33:42 SRET 2019

Figure 2. When `deleteProject 3` *command is executed.*

Listing all projects : `listProject` / `lp`

When you want to take a look at all your projects, this command helps you do so.

Format: `listProject`

Assigning a Beneficiary to Project: `assignB`

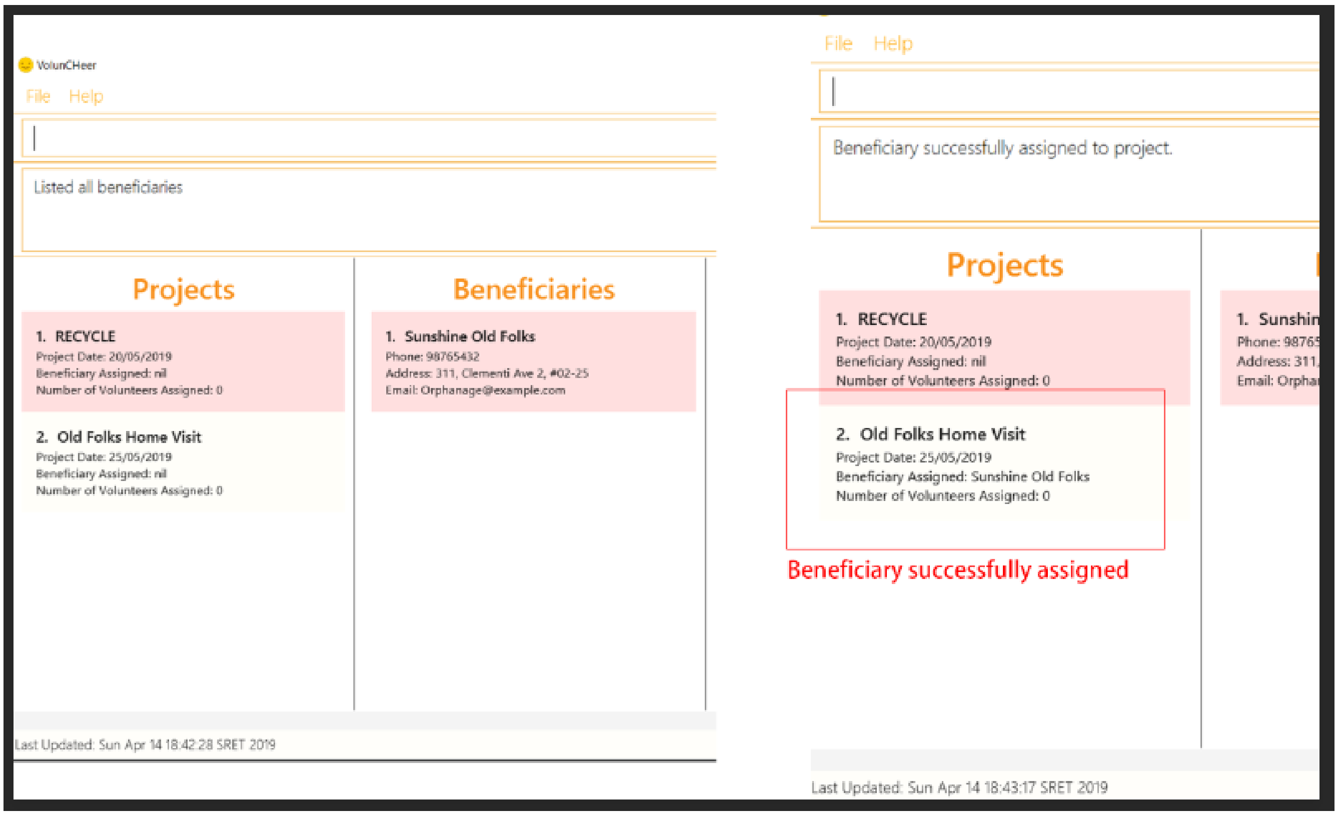
Projects are generally associated with certain beneficiaries. VolunCHeer allows you to attach them easily with this command. It assigns the Beneficiary at the provided INDEX to the Project with ProjectTitle indicated.

Format: `assignBeneficiary p/PROJECT_TITLE i/INDEX`

- The assigned Beneficiary can then be seen under the Project card as shown below.
- There can be only one beneficiary for each project, however, one beneficiary can be assigned to multiple projects.

IMPORTANT The index **must be a positive integer** `1, 2, 3, …`

After assigning a beneficiary, the project will have its data updated as seen below.



Listed all beneficiaries

Projects

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Old Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

Beneficiaries

1. Sunshine Old Folks
Phone: 98765432
Address: 311, Clementi Ave 2, #02-25
Email: Orphanage@example.com

Last Updated: Sun Apr 14 18:42:28 SRET 2019

Beneficiary successfully assigned to project.

Projects

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Old Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: Sunshine Old Folks
Number of Volunteers Assigned: 0

Beneficiary successfully assigned

1. Sunshin
Phone: 98765
Address: 311,
Email: Orphan

Last Updated: Sun Apr 14 18:43:17 SRET 2019

Figure 3. When `assignB p/Old Folks Home Visit i/1` *command executed*

TIP Use listBeneficiary to view a full list of Beneficiary to assign. Use summaryBeneficiary command to view the Projects attached to each Beneficiary.

Assigning one or more Volunteers to Project: `assignV`

We also provide an easy method to assign a specific number of volunteers to the indicated Project

Format: `assignVolunteer p/PROJECT_TITLE rv/REQUIRED_NUMBER_OF_VOLUNTEERS`

TIP The number of volunteers assigned to the Project can be seen under the Project card as shown below.

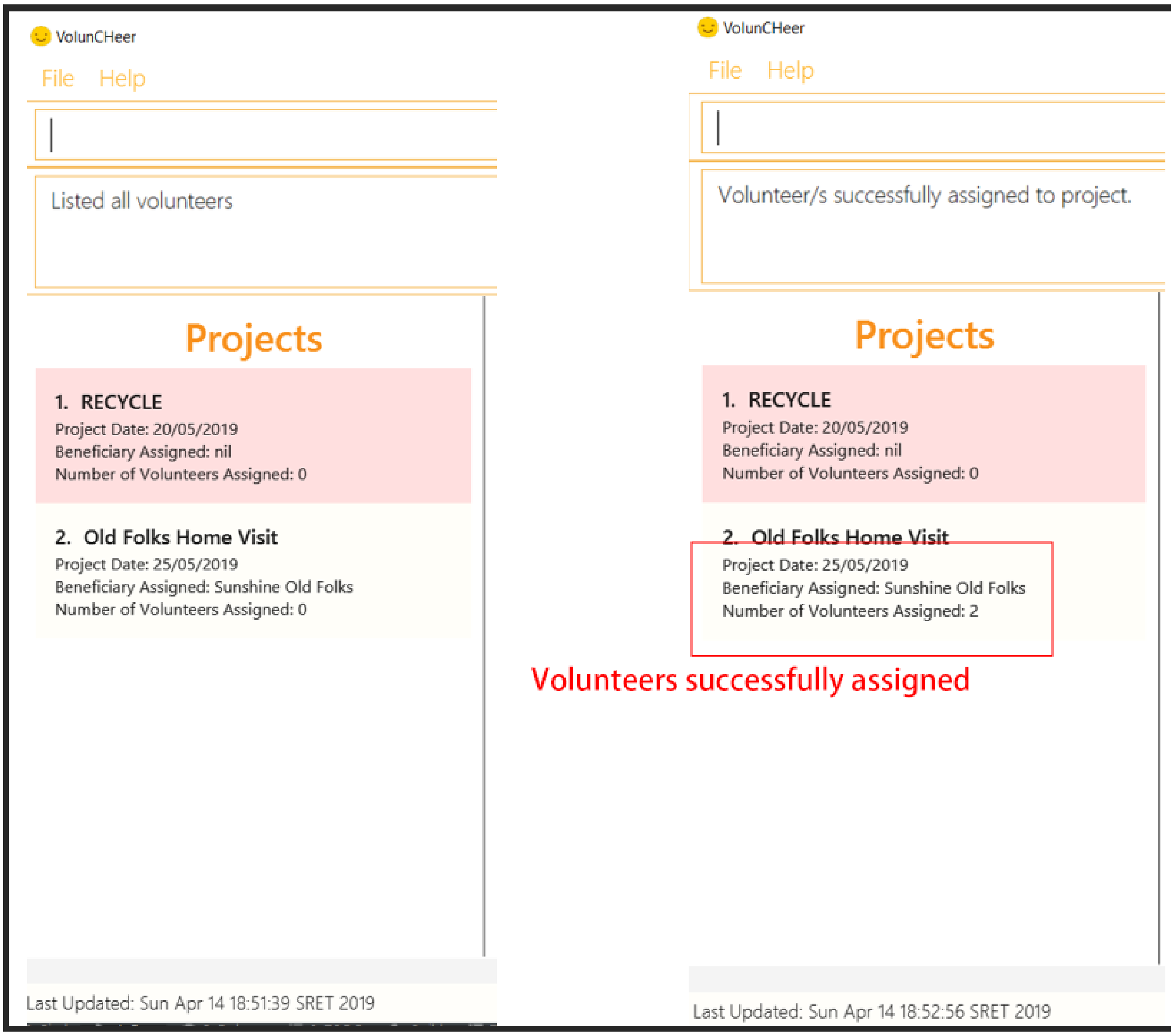

VolunCHeer
File
Help
Listed all volunteers
Projects
1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0
2. Old Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: Sunshine Old Folks
Number of Volunteers Assigned: 0
Last Updated: Sun Apr 14 18:51:39 SRET 2019
VolunCHeer
File
Help
Volunteer/s successfully assigned to project.
Projects
1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0
2. Old Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: Sunshine Old Folks
Number of Volunteers Assigned: 2
Volunteers successfully assigned
Last Updated: Sun Apr 14 18:52:56 SRET 2019

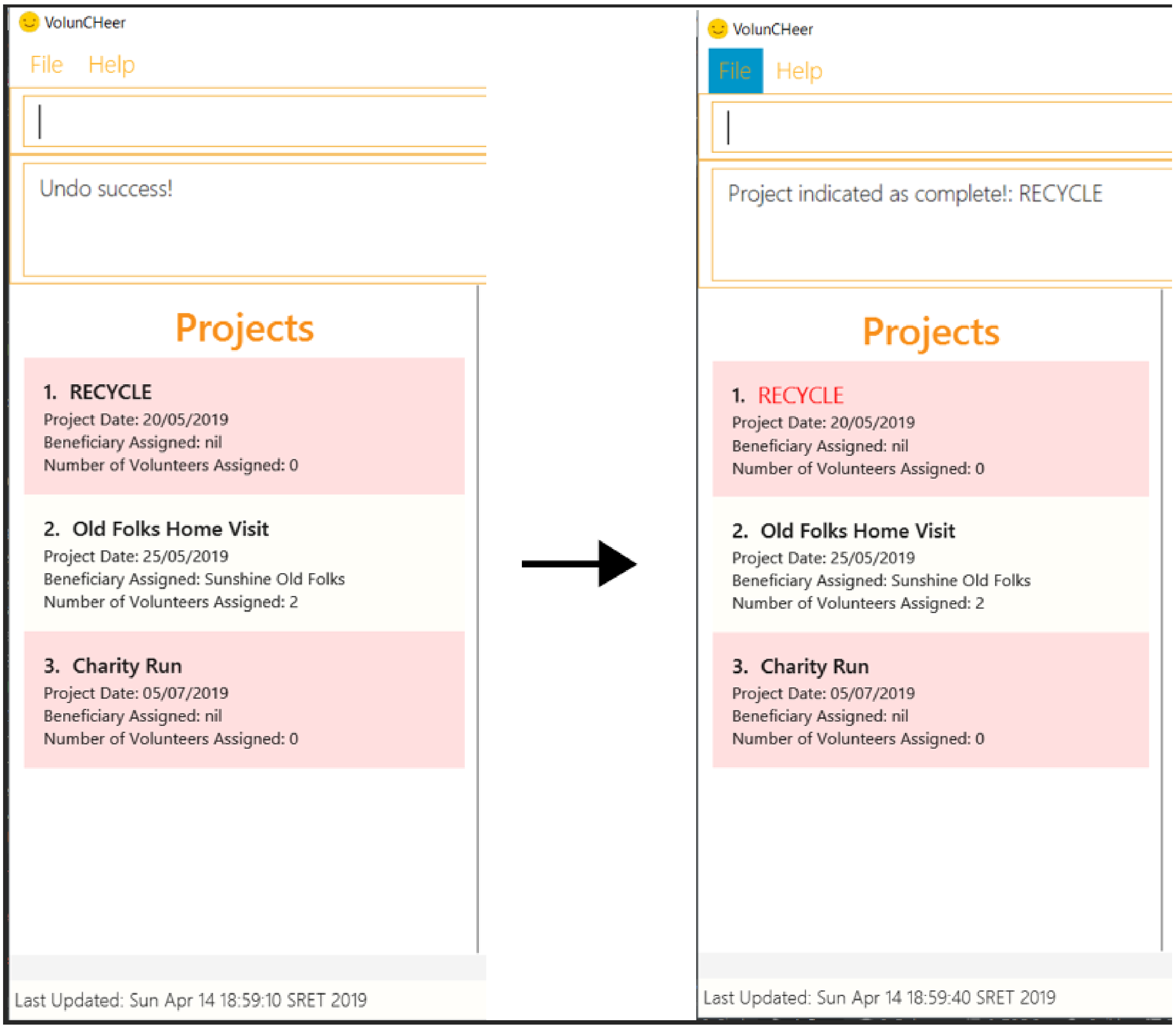
Figure 4. When `assignV p/Old Folks Home Visit rv/2` *is executed.*

TIP Use the commands listed in [map] to filter out the desired list of volunteers.

Mark project as complete: 'complete'

Once a `project` is done, you can mark it as complete to distinguish it from your other projects. Simply provide an INDEX to indicate which project you would like to complete.

Format: `complete i/INDEX`



Undo success!

Projects

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Old Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: Sunshine Old Folks
Number of Volunteers Assigned: 2

3. Charity Run
Project Date: 05/07/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

Last Updated: Sun Apr 14 18:59:10 SRET 2019

Project indicated as complete!: RECYCLE

Projects

1. RECYCLE
Project Date: 20/05/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

2. Old Folks Home Visit
Project Date: 25/05/2019
Beneficiary Assigned: Sunshine Old Folks
Number of Volunteers Assigned: 2

3. Charity Run
Project Date: 05/07/2019
Beneficiary Assigned: nil
Number of Volunteers Assigned: 0

Last Updated: Sun Apr 14 18:59:40 SRET 2019

Figure 5. When `complete i/1` *command is executed*

NOTE Once marked as complete, project title will be displayed in red colour font

== 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.

=== Project Complete Feature The complete feature allows users to indicate a project as completed.

==== Implementation To facilitate the complete feature, an association with a new `Complete` class is added to the `Project` class:

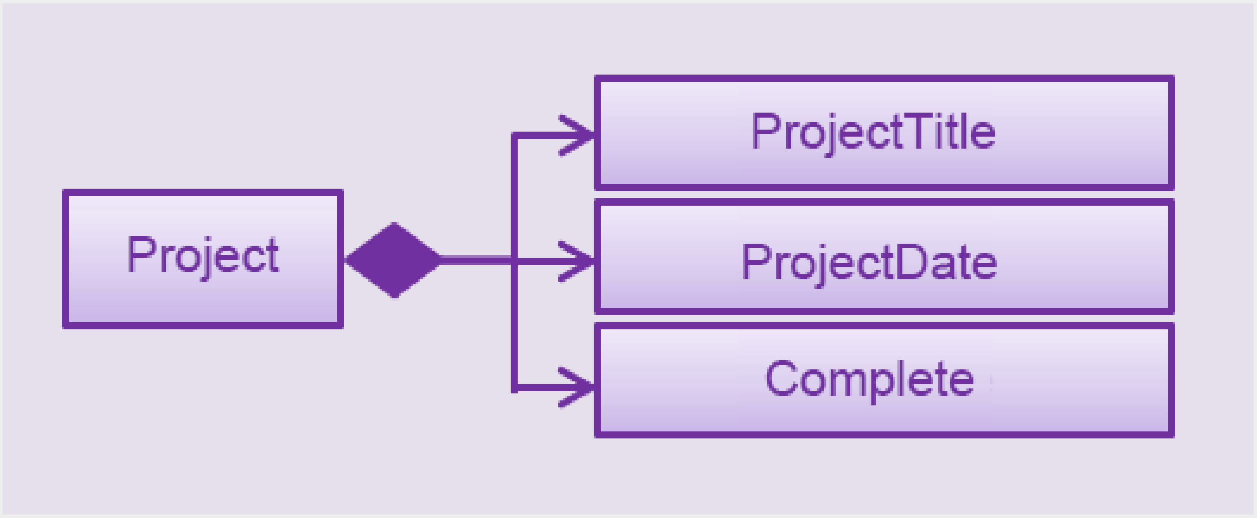


Figure 6. Structure of the attributes of a Project in the Model component.

The diagram shows that the Project class is associated with the Complete class.

The following sequence diagram shows how the complete command works:

[CompleteSequenceDiagram] | *CompleteSequenceDiagram.png*

Figure 7. Figure Sequence diagram for the complete command.

1. The `CompleteCommandParser` parses the user input to obtain the target project index and constructs a ne `CompleteCommand` with this index.
2. The logic portion of the complete command will be executed by the `CompleteCommand` method. To mark a Project object as complete:
    1. The **CompleteCommand()** method creates a `targetProject` based on the provided project index.
    2. In the **executeCommandResult()** method then creates a `editedProject` with `Complete` attribute set to “true”. The `editedProject` is created with ProjectBuilder as shown below:

```
    Project editedProject = new
ProjectBuilder(targetProject).withComplete(true).build();
```

3. In the executeCommandResult() method

```
model.setProject(targetProject, editedProject)
```

is called to replace `Project`’s complete attribute from “false” to “true” in the VolunCHeer in-memory.

==== Design Considerations

Aspect	Alternatives	Pros (+)/ Cons(-)
Implementation of ‘CompleteCommand’	**Add a Complete attribute to Project (current choice) -Completed projects indicated “Red”**	+ : It is easy to tag complete status as an attribute to the `Project` as we can make use of current implementations such as model.setProject(Project,Project) that sets the `Project`’s complete attribute to “true”. - : Unable to have a observable list of complete projects.
	Create a new CompletedProjectList that consists of all the complete projects, a listComplete command to show all completed tasks..	+ : Will use less memory (e.g. for deleteVolunteer, just save the volunteer being deleted). - : We must ensure that the implementation of each individual command are correct.

=== Assign Feature Assigning a Beneficiary / VolunteerList to Project.

==== Implementations Since the implementation of commands AssignBeneficiary and AssignVolunteer are similar, we will describe the implementation of AssignBeneficiary command only and provide the difference between the two.

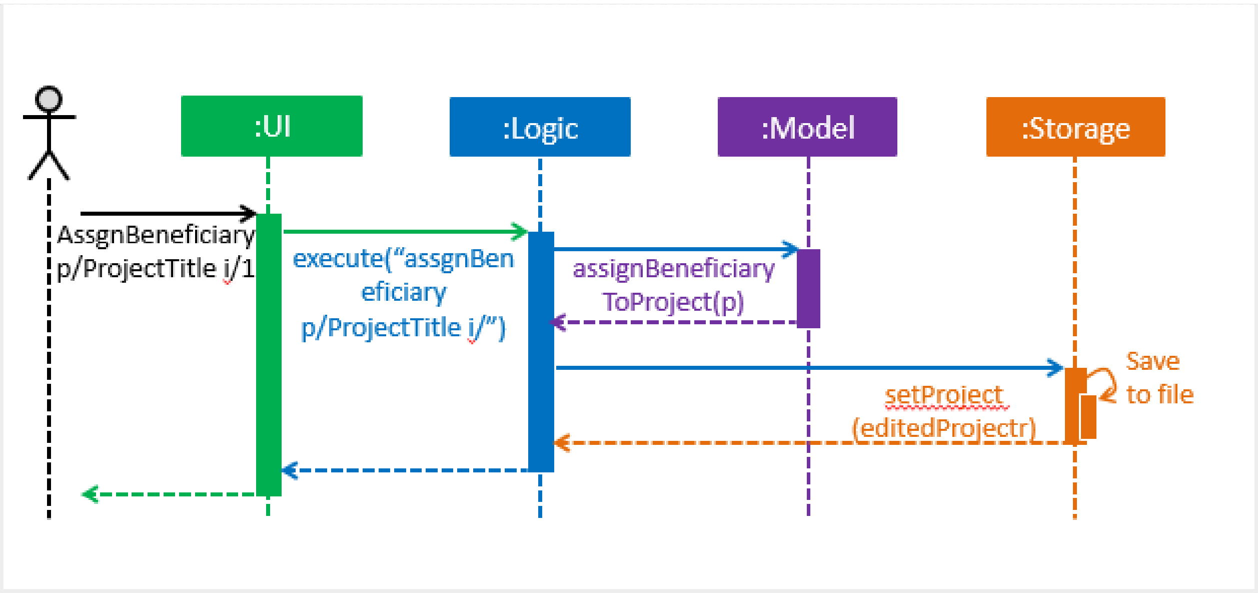

:UI
:Logic
:Model
:Storage
AssgnBeneficiary p/ProjectTitle i/1
execute("assgnBeneficiary p/ProjectTitle i/")
assignBeneficiary ToProject(p)
setProject (editedProjectr)
Save to file

Figure 8. Sequence diagram to show how the AssignBeneficiaryCommand works.

1. The **AssignBeneficiaryCommand(ProjectTitle, Index)** takes in the targetProject's projectTitle attribute and targetBeneficiary's index.
2. The **executeCommandResult()** method
    1. Sets up projectToAssign by calling a predicate to compare with the `ProjectTitle` in `FilteredProjectList`:

    ```
    model. getFilteredProjectList(). filtered(equalProjectTitle).get(0);
    ```

    2. **updateBeneficiary(model)** methods updates the `Beneficiary` object so that ProjectTitle is tracked within the Beneficiary class.
    3. editedProject is created using ProjectBuilder to take in the `Beneficiary` assigned. The following method is called to store the `Project` in VolunCHeer with specific `Beneficiary` attached to it.

    ```
    model.setProject(projectToAssign, editedProject)
    ```