Zhu Hui - Project Portfolio

PROJECT: AddressBook - Level 4

Overview

This project portfolio page serves to document my contributions to the VolunCHeer project.

VolunCHeer is an open-sourced Command Line Interface (CLI) management application designed for Volunteering Project Managers. We aim to help our target users alleviate the haystack of managing multiple projects, beneficiaries, and volunteers in an efficient and effective manner.

Team T08-1 consists of Zhao Junru, Nguyen Dang Huu, Jeremy Wong Kai Wen and myself. We are a group of friends who have experiences, working with major volunteering organizations. We understand the challenges volunteering project managers faced when dealing with multiple projects, beneficiaries and volunteers and thus, we created VolunCHeer to serve that need.

Terminology

- Volunteer: The volunteers who participate in volunteer organization such as NUS Community Service Club, NUS Vietnamese Community's CIP Committee.
- Beneficiary: Organizations who benefit from volunteering activities such as Old Folk Home, Nursing Home, and Orphanage.
- Project: Projects that are set for volunteers to participate and help the beneficiary.
- Volunteering Project Manager: The one who manages the arrangement of projects, assign volunteers and contact and associate beneficiaries with projects. In this project portfolio, the Volunteering Project Manager is mentioned as the user.

Role

I am in charge of testing for VolunCHeer. If you found any bugs in VolunCHeer, please do not hesitate to contact us directly.

Summary of contributions

- **Major enhancement**: Volunteer Feature
    - What it does: It allows Project Manager to store the data of the volunteers and access specific data easily with a few simple commands.
    - Justification: This feature is one of the basic building blocks of VolunCHeer. Having access to and being able to filter thousands of volunteers' data in matter of seconds would allow project managers to plan their projects efficiently and effectively.
- **Major enhancement**: Password Feature

    - What it does: It prompts for a password verification upon executing the application and prevent unauthorised users from using VolunCHeer.
    - Justification: The data stored in VolunCHeer is important and confidential. I have implemented the password feature because we are concerned about the security of our users and do not wish data stored in our program to be stolen.
    - Highlights: A simple UI with a single password field is used to implement this feature as it is more user friendly compared to using the command line interface.
- **Minor enhancement**: Exit Dialog Box
    - What it does: It prompts the user for a confirmation message when exiting VolunCHeer. It allows the `exit` command to bypass the message.
    - Justification: This feature prevents accidental termination of the program by the user or external program.
- **Code contributed**: [Volunteer] [Password]
- **Other contributions**:
    - UI interface:
        - Ensure that UI is running smoothly

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.

*Volunteer Feature === Volunteer

Adding a volunteer: `addVolunteer` / `av`

As like before, this adds a volunteer to the volunteer pool

Format: `addVolunteer n/NAME y/AGE g/GENDER r/RACE [rg/RELIGION] a/ADDRESS e/EMAIL p/PHONE_NUMBER ec/EMERGENCY_CONTACT [dp/DIETARY_PREFERENCE] [m/MEDICAL_CONDITION] [t/TAG]…`

Alternative Format: `av n/NAME y/AGE g/GENDER r/RACE [rg/RELIGION] a/ADDRESS e/EMAIL p/PHONE_NUMBER ec/EMERGENCY_CONTACT [dp/DIETARY_PREFERENCE] [m/MEDICAL_CONDITION] [t/TAG]…`

- "Add Successful!" message is prompted upon successfully adding a volunteer
- An invalid message will be prompted if a Volunteer with the same exact name is present in the existing database
- Parameters for Religion, Dietary Preference, Medical Condition are optional and set to 'nil' by default

TIP A volunteer can have any number of tags (including 0)

Examples:

- `addVolunteer n/John Doe y/18 g/male r/eurasian rg/christian a/John street, block 123, #01-01 e/johnd@example.com p/98765432 ec/Mary, Mother, 92221111 dp/vegetarian m/asthma`
- `av n/Sarah Soh y/22 g/female r/chinese rg/buddhist a/betsy ave 6, 02-08 e/sarah08@example.com p/92345678 ec/Johnny, Husband, 81234568`

Deleting a volunteer : `deleteVolunteer` \ `dv`

After a volunteer has left, it can be deleted by this command by referencing its index in the list.

Format: `deleteVolunteer INDEX`
Alternative Format: `dv INDEX`

- Deletes the volunteer at the specified `INDEX`.
- The index refers to the index number shown in the displayed volunteer list.
- The index **must be a positive integer** 1, 2, 3, …
- Error message is shown if the given index is invalid

Examples:

- `listVolunteer`
  `deleteVolunteer 2`
  Deletes the 2nd volunteer in the volunteer list.
- `findVolunteer Betsy`
  `dv 1`
  Deletes the 1st volunteer in the searched volunteer list.

TIP	Use the list volunteers commands to check the correct index of the volunteer to be deleted

Editing a volunteer : `editVolunteer` \ `ev`

Similar to beneficiary, we can update volunteer particulars by the given index.

Format: `editVolunteer INDEX [n/NAME] [y/AGE] [g/GENDER] [r/RACE] [rg/RELIGION][p/PHONE] [a/ADDRESS] [e/EMAIL] [ec/EMERGENCYCONTACT] [dp/DIETARYPREFERENCE] [mc/MEDICALCONDITION] [t/TAG]…`

Alternative Format: `ev INDEX [n/NAME] [y/AGE] [g/GENDER] [r/RACE] [rg/RELIGION][p/PHONE] [a/ADDRESS] [e/EMAIL] [ec/EMERGENCYCONTACT] [dp/DIETARYPREFERENCE] [mc/MEDICALCONDITION] [t/TAG]…`

- Edits the volunteer at the specified `INDEX`. The index refers to the index number shown in the displayed volunteer 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.
- When editing tags, the existing tags of the volunteer will be removed i.e adding of tags is not cumulative.
- You can remove all the volunteer's tags by typing `t/` without specifying any tags after it.

Examples:

- `editVolunteer 1 p/91234567 e/johndoe@example.com`
  Edits the phone number and email address of the 1st volunteer to be `91234567` and `johndoe@example.com` respectively.
- `ev 2 n/Betsy Crower t/`
  Edits the name of the 2nd volunteer to be `Betsy Crower` and clears all existing tags.

Locating volunteers by name: `findVolunteer` \ `fv`

Searching for volunteers works similarly to beneficiaries.

Format: `find KEYWORD [MORE_KEYWORDS]`

Alternative Format: `fv KEYWORD [MORE_KEYWORDS]`

- The search is case insensitive. e.g `hans` will match `Hans`
- The order of the keywords does not matter. e.g. `Hans Bo` will match `Bo Hans`
- Only the name is searched.
- Only full words will be matched e.g. `Han` will not match `Hans`
- volunteers matching at least one keyword will be returned (i.e. `OR` search).
- e.g. `Hans Bo` will return `Hans Gruber`, `Bo Yang`

Examples:

- `findVolunteer John`
  Returns `john` and `John Doe`
- `fv Betsy Tim John`
  Returns any volunteer having names `Betsy`, `Tim`, or `John`

Listing all volunteers : `listVolunteer` \ `lv`

Shows a list of all volunteers in the volunteer pool.

Format: `listVolunteer`

Alternative Format: `lv`

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.

*Password Feature === Password Feature ==== Current Implementation A password is required to grant access to the project manager at the start of the program. It serves as a security mechanism to protect the important data within.

At the start of the program, the UI is initialised by `UiManager`. A password window will pop out and prompt the user to enter the password.

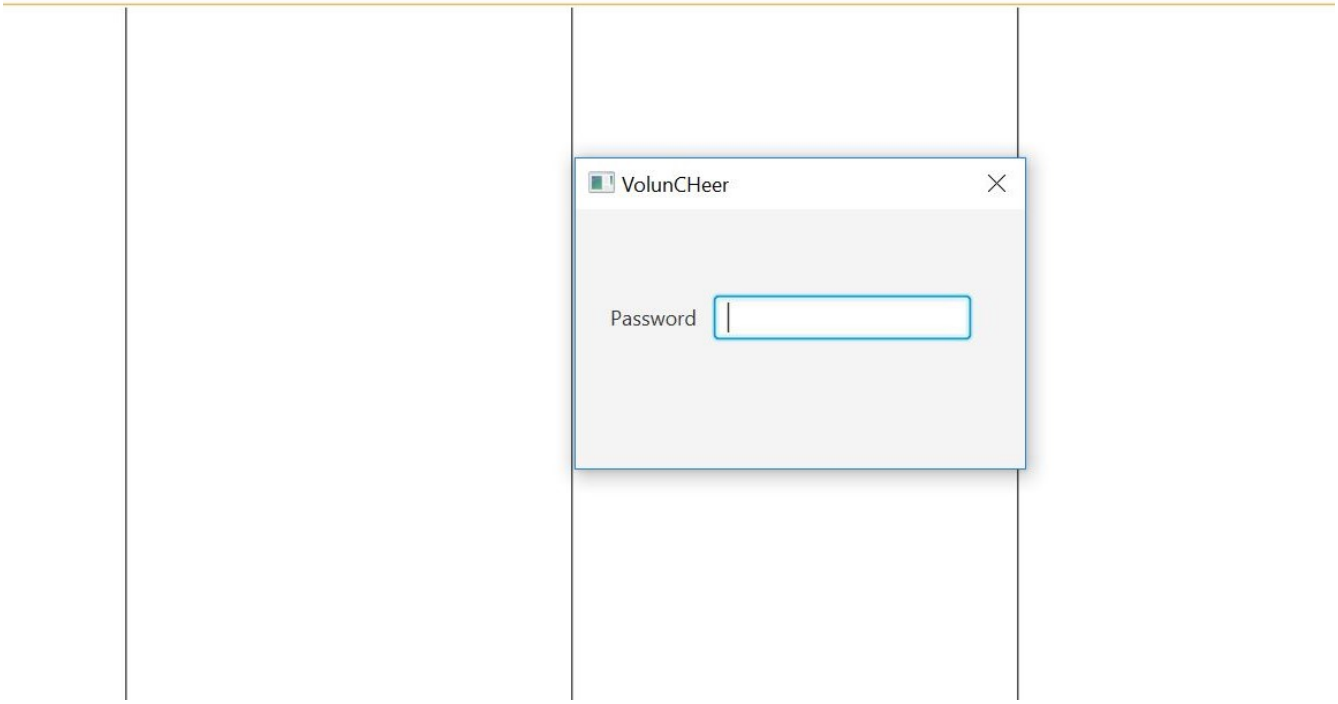

Figure 1. Program prompting user for password

The password feature uses `ValidatePassword` to check if the user input matches the password stored in the program.

If the password matches, it will set boolean `user` to be true. `UiManager` will thus instantiate `fillInnerParts()`, displaying the hidden data to the project manager.

Figure 2. Data displayed to user upon entering the correct password

If the password is entered incorrectly for five times, the program will terminate itself. A sequence diagram is shown below to illustrate how it works in an abstract level.

[ArchitectureDiagram password] | *ArchitectureDiagram_password.png*

Figure 3. Sequence diagram of Password Feature

Design Considerations

Aspect	Alternatives	Pros (+)/ Cons(-)
Implementation of Synchronization	**UI with a password field. (current choice)**	+ : It is simple and user friendly. It serves its purpose as a offline application for a single user. - : The current alternative does not allow the user to change password.
	Using Command Line Interface	+ : It allows more flexibility to implement more parameters such as custom security questions to improve its security. - : It may not be as user friendly as the UI.