VolunCHeer - User Guide

By: `CS2113T T08-01` Since: `Feb 2019` Licence: `MIT`

1. Introduction

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.

You can also use VolunCHeer to filter out suitable volunteers based on their data stored in the system. VolunCHeer is The VolunCHeer Application is for project managers or directors who require a system to **keep track of their ongoing projects and the volunteer pool**. VolunCheer is optimised for users who prefer a **Command Line Interface (CLI)** while still being able to view important data on the **Graphical User Interface(GUI)**. VolunCHeer will save you the need for multiple documents and folders just to store volunteer/beneficiary information. It also removes the need to track large amount of volunteer data to fit selection criterias using traditional methods such as Excel.

Wish to know more? Click on Section 2. "Quick Start" to get started.

1.1. Overview

Shown below is a quick overview of our VolunCHeer application.

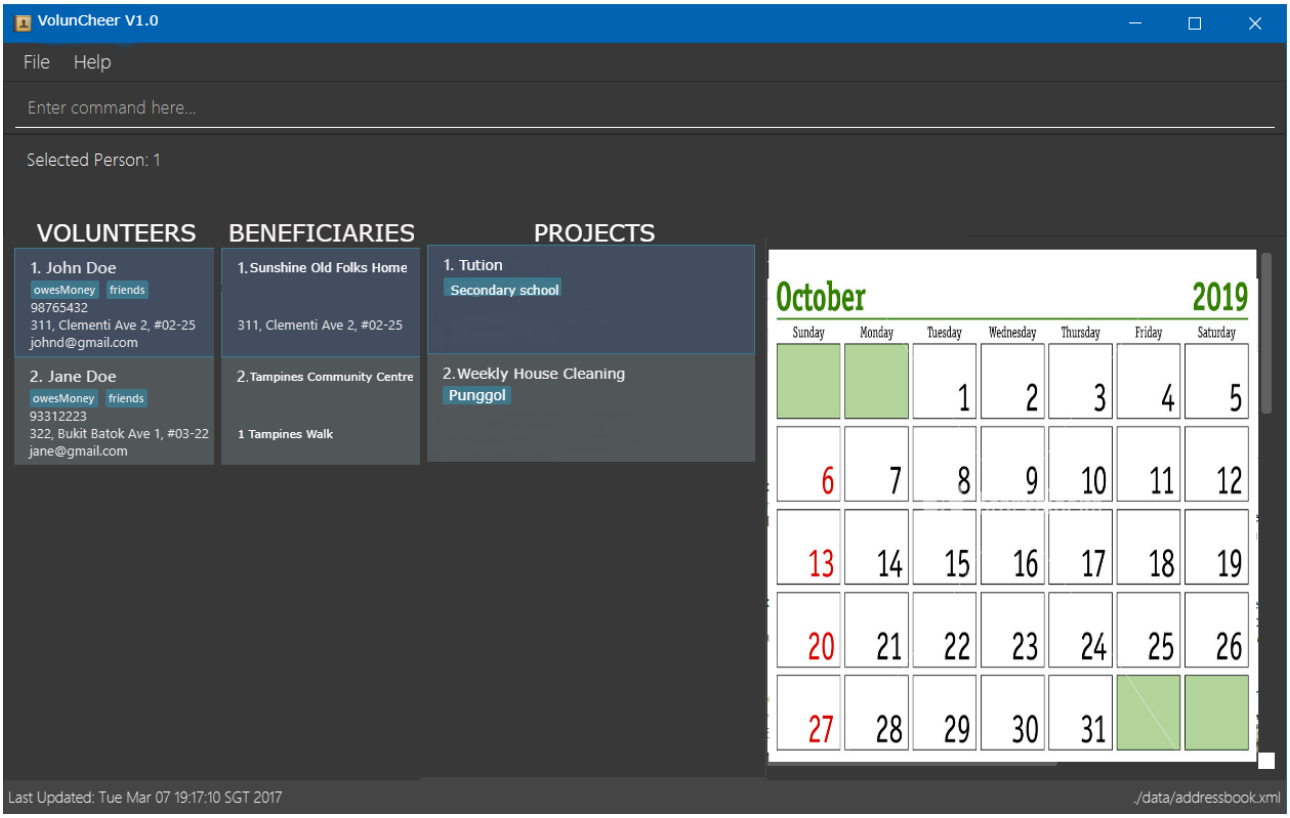
+ image::Ui.png[width="790"]

+

1. Command Box: This is where you type your commands.
2. Command Result: This shows the result of command execution, useful tips and error messages are also shown here.
3. Project List: This shows the list of projects currently in VolunCHeer.
4. Beneficiary List: This shows the list of beneficiaries currently in VolunCHeer.

2. Quick Start

1. Ensure you have Java version `9` or later installed in your Computer.
2. Download the latest `VolunCHeer.jar` here.
3. Copy the file to the folder you want to use as the home folder for our VolunCHeer Application.
4. Double-click the file to start the app. The GUI should appear in a few seconds.



5. Type the command in the command box and press `Enter` to execute it.
   e.g. typing **`help`** and pressing `Enter` will open the help window.
6. Some example commands you can try:
   - **`list`** : lists all contacts
   - **`addProject`** n/Project Sunshine d/20190320: adds a project named "Project Sunshine" in the project list.
   - **`deleteProject2`** : deletes the 2nd project portfolio in the current list of projects.
   - **`exit`** : exits the app
7. Refer to Section 3, “Features” for details of each command.

3. Features

Command Format

- Words in `UPPER_CASE` are the parameters to be supplied by the user e.g. in `add n/NAME`, `NAME` is a parameter which can be used as `add n/John Doe`.
- Items in square brackets are optional e.g `n/NAME [t/TAG]` can be used as `n/John Doe t/friend` or as `n/John Doe`.
- Items with … after them can be used multiple times including zero times e.g. `[t/TAG]…` can be used as (i.e. 0 times), `t/friend`, `t/friend t/family` etc.
- Parameters can be in any order e.g. if the command specifies `n/NAME p/PHONE_NUMBER`, `p/PHONE_NUMBER n/NAME` is also acceptable.

3.1. Viewing help : `help`

Format: `help`

3.2. Adding a project: `addProject`

Adds a new project
Format: `addProject n/PROJECT_NAME d/DATE`

- Please enter DATE in yyyymmdd format.
- Project Title should not have duplicates in the project list.

Examples:

- `addProject n/Charity Run d/081219`

3.3. Adding a volunteer: `addVolunteer`

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]…`

- "Add Successful!" message is prompted upon successfully adding a volunteer
- Otherwise, if the volunteer already exists, show edit option to update portfolio
- Dietary Preference, Medical Condition is 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/nil a/John street, block 123, #01-01 e/johnd@example.com p/98765432 ec/Mary, Mother, 92221111 dp/vegetarian m/asthma`
- `addVolunteer 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

3.4. Adding a beneficiary: `addBeneficiary`

Adds a beneficiary to the list of Beneficiaries
Format: `addBeneficiary n/NAME a/ADDRESS e/EMAIL p/PHONE_NUMBER `

- "New beneficiary added: Orphanage Phone: 98765432 Email: Orphanage@example.com Address: 311, Clementi Ave 2, #02-25Attached Project List: []" message is prompted upon successfully adding a beneficiary

Examples:

- `addBeneficiary n/Orphanage p/98765432 e/Orphanage@example.com a/311, Clementi Ave 2, #02-25`

Consideration:

- The beneficiary will be used to assign to a project, this means that the project will benefit this beneficiary, i.e. Orphanage Home, Nursing home, etc.
- When add a new beneficiary, the project lists assigned to it will be empty. You can assign projects to it by assign command stated below.

3.5. Editing a beneficiary: `editBeneficiary`

Edits a beneficiary to the list of Beneficiaries
Format: `editBeneficiary INDEX (must be a positive integer) [n/NAME] [p/PHONE] [e/EMAIL] [a/ADDRESS] `

> - "Edited Beneficiary: Old Folk Home Phone: 91234567 Email: Orphanage@example.com Address: 311, Clementi Ave 2, #02-25Attached Project List: []" message is prompted upon successfully editing a beneficiary

Examples:

- `editBeneficiary 1 n/Old Folk Home p/91234567`

Consideration:

- When a beneficiary is edited, the data of the beneficiary in its attached projects is in sync, meaning that that data is automatically updated in the mentioned projects.

3.6. Listing all beneficiary: `listBeneficiary`

Shows a list of all Beneficiaries in the beneficiary pool.
Format: `listBeneficiary`

Consideration: * The command can be used to get back to full list after several commands changing the list.

3.7. Locating beneficiaries by name: `findBeneficiary`

Finds beneficiaries whose names contain any of the given keywords.
Format: `findBeneficiary KEYWORD [MORE_KEYWORDS]`

- The search is case insensitive. e.g `orphanage` will match `Orphanage`
- The order of the keywords does not matter. e.g. `Orphanage Nursing` will match `Nursing Orphanage`
- Only the name is searched.
- Only full words will be matched e.g. `Orphan` will not match `Orphanage`
- beneficiaries matching at least one keyword will be returned (i.e. `OR` search). e.g. `Orphanage Nursing` will return `Orphanage Rainbow`, `Nursing Home`

Examples:

- `find Nursing`
  Returns `Nursing Home` and `Nursing Center`

3.8. Listing all projects : `listProject`

Shows a list of all projects.
Format: `listProject`

3.9. Listing all volunteers : `list`

Shows a list of all volunteers in the volunteer pool.
Format: `list`

3.10. Editing a project : `editProject`

Edits an existing project
Format: `editProject PROJECT_NAME [n/NAME] [d/DATE]…`

- Edits the project at the specified 'PROJECT_NAME'.
- Existing values will be updated to the input values.
- When editing tags, the existing tags of the project will be removed i.e adding of tags is not cumulative.
- Project's tags can be removed by typing `t/` without specifying any tags after it.
- `n/` is invalid as 'PROJECT_NAME' cannot be removed unless with deleteProject command.

Examples:

- `editProject Charity Run d/20190301`
  Edits the date of the project to be '20190301'.

3.11. Editing a volunteer : `editVolunteer`

Edits an existing volunteer in the volunteer list.
Format: `edit 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.
- `editVolunteer 2 n/Betsy Crower t/`
  Edits the name of the 2nd volunteer to be `Betsy Crower` and clears all existing tags.

3.12. Locating volunteers by name: `find`

Finds volunteers whose names contain any of the given keywords.
Format: `find 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:

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

3.13. Deleting a project : `deleteProject`

Deletes the specified project from the application.
Format: `deleteProject INDEX`

- Deletes the project specified with 'PROJECT_TITLE'.
- The PROJECT_TITLE should match exactly, use 'listProject' to view all projects if unsure.
- Error message is shown if the PROJECT_TITLE entered is invalid

3.14. Deleting a volunteer : `deleteVolunteer`

Deletes the specified volunteer from the volunteer list.
Format: `delete 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:

- `list`
  `delete 2`
  Deletes the 2nd volunteer in the volunteer list.
- `find Betsy`
  `delete 1`
  Deletes the 1st volunteer in the results of the `find` command.

3.15. Selecting a volunteer : `select`

Selects the volunteer identified by the index number used in the displayed volunteer list.
Format: `select INDEX`

- Selects the volunteer and loads the Google search page 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, …`

Examples:

- `list`

`select 2`
Selects the 2nd volunteer in the volunteer list.

- `find Betsy`
  `select 1`
  Selects the 1st volunteer in the results of the `find` command.

3.16. Assigning a beneficiary to project: 'assign'

Assigns a beneficiary identified by the index number used in the displayed beneficiary list to a project matched by project title entered. Format: `assign PROJECT_TITLE, i/INDEX`

- Assigns the beneficiary with index = "INDEX" to the project with title "PROJECT_TITLE".
- Project attached will be shown on the specific beneficiary
- There can be only one beneficiary for each project, however, one beneficiary can be assigned to multiple projects.
- The index **must be a positive integer** `1, 2, 3, …`

3.17. Mark project as complete: 'complete' --- coming soon

Marks project with "PROJECT_TITLE" as complete Format: `complete PROJECT_TITLE`

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

3.18. Assigning mapping index to each volunteer : `map`

Assigns the volunteers with points 3, 2, 1 according to the selection criteria set by the user. Format: `map t/(POINTS)(CRITERIA) t/(POINTS)(CRITERIA) t/(POINTS)(CRITERIA)`

- There are three types of tags, the age of volunteer (y/), race (r/) and medical condition (m/).
- There can be at most 3 tags and at least 1 tag as the selection criteria.
- Each volunteer is tagged with the final PRIORITY_SCORE based on the points used for sorting later on.
- The age criteria has comparators >,<,= which relate to the age given afterwards.
- See examples below for a clearer picture.

Examples:

- `map y/3>18 r/2chinese m/1NIL` Gives volunteers above the AGE of 18 3 points, RACE chinese 2

points and MEDICAL_CONDITION of NIL 1 point.

- `map m/3NIL` Only gives volunteers with no MEDICAL_CONDITION 3 points.

3.19. Sort volunteers according to PRIORITY_SCORE : `sort`

Sorts the volunteers from highest PRIORITY_SCORE to lowest PRIORITY_SCORE. Format: `sort`

- The map function should be called before sort to generate the PRIORITY_SCORE
- Volunteers with PRIORITY_SCORE of 0 will not be sorted in any particular order
- Selection of the volunteers based on the selection criteria can be done after they are sorted

3.20. Select multiple volunteers from sorted list : `extract` (Coming in V1.4)

Format: `extract [a][b]`

- Call extract after sort function to extract [b]-[a] number of volunteers.

Examples:

*`extract [1][20]` Extracts the first 20 volunteers in the sorted list. *`extract [5][15]` Extracts volunteer number 5 to 15 in the list.

3.21. Listing entered commands : `history`

Lists all the commands that you have entered in reverse chronological order.
Format: `history`

NOTE	Pressing the <code>↑</code> and <code>↓</code> arrows will display the previous and next input respectively in the command box.

3.22. Undoing previous command : `undo`

Restores the VolunCHeer application to the state before the previous *undoable* command was executed.
Format: `undo`

NOTE	Undoable commands: those commands that modify the VolunCHeer application's main content (`addProject`, `addVolunteer`, `delete`, `edit` and `clear`).

Examples:

- `delete 1`
  `list`
  `undo` (reverses the `delete 1` command)
- `select 1`
  `list`
  `undo`
  The `undo` command fails as there are no undoable commands executed previously.
- `delete 1`
  `clear`
  `undo` (reverses the `clear` command)
  `undo` (reverses the `delete 1` command)

3.23. Redoing the previously undone command : `redo`

Reverses the most recent `undo` command.
Format: `redo`

Examples:

- `delete 1`
  `undo` (reverses the `delete 1` command)
  `redo` (reapplies the `delete 1` command)
- `delete 1`
  `redo`
  The `redo` command fails as there are no `undo` commands executed previously.
- `delete 1`
  `clear`
  `undo` (reverses the `clear` command)
  `undo` (reverses the `delete 1` command)
  `redo` (reapplies the `delete 1` command)
  `redo` (reapplies the `clear` command)

3.24. Clearing all entries : `clear`

Clears all entries from the specific list requested by user.
Format: `clear`

3.25. Export data file: `export`

Exports the saved data in a csv file.
Format: 'export'

- Supports export of volunteer list
- Supports export of project data
- Application shows successful export message once exported

3.26. Import data file: `import`

Imports csv file saved in local folder.
Format: 'import FILE_DIRECTORY'

- The application finds the local file and extrapolate the data
- Supports import of volunteer data
- Application shows successful import message once imported

3.27. Exiting the program : `exit`

Exits the program.
Format: `exit`

3.28. Saving the data

All data for the application are saved in the hard disk automatically after any command that changes the data.
There is no need to save manually.

3.29. Attendance taking `[coming in v2.0]`

Track attendance of the volunteers and award frequent volunteers with certificates or promote to team leader.

3.30. Manage funding and sponsorships `[coming in v2.0]`

Manage funds and sponsors for individual projects and track project spending.

4. FAQ

Q: How do I transfer my data to another Computer?
A: Install the app in the other computer and overwrite the empty data file it creates with the file that contains the data of your previous VolunCHeer application folder.

5. Command Summary

- **AddProject** `addProject n/PROJECT_TITLE d/DATE b/BENEFICIARY [t/TAG]…`
  e.g. `addProject n/Charity Run d/081219 b/Sunshine Old Folks Home`
- **AddVolunteer** `addVolunteer n/NAME y/AGE a/ADDRESS e/EMAIL p/PHONE_NUMBER g/EMERGENCY_CONTACT r/RACE d/DIETARY_PREFERENCE m/MEDICAL CONDITION [t/TAG]…`
  e.g. `addVolunteer n/John Doe y/18 a/John street, block 123, #01-01 e/johnd@example.com p/98765432 g/98292998 r/chinese d/vegetarian m/asthma`
- **AddBeneficiary** `addBeneficiary n/NAME a/ADDRESS e/EMAIL p/PHONE_NUMBER v/VOLUNTEERS_REQUIRED [t/TAG]…`
  e.g. `addBeneficiary n/Sunshine Old Folks Home a/sunshine ave 5 e/sunny@oldfolks.sg p/67580392 v/20`
- **List** : `list`
- **EditProject** `editProject PROJECT_NAME [n/NAME] [d/DATE] [b/BENEFICIARY] [t/TAG]…`
  e.g. `editProject Charity Run d/010319`
- **EditVolunteer** `edit INDEX [n/NAME] [p/PHONE] [e/EMAIL] [a/ADDRESS] [t/TAG]…`
  e.g. `editVolunteer 1 p/91234567 e/johndoe@example.com`
- **Find** : `find KEYWORD [MORE_KEYWORDS]`
  e.g. `find James Jake`
- **DeleteProject** : `delete PROJECT_TITLE` e.g. `delete Charity Run`
- **DeleteVolunteer** : `delete INDEX`
  e.g. `delete 3`
- **Select** : `select INDEX`
  e.g.`select 2`
- **Map** `map t/SELECTION t/SELECTION t/SELECTION`
  e.g. `map y/18 > r/chinese m/NIL`
- **Sort** `sort`
- **Extract** `extract VOLUNTEERS_REQUIRED+` e.g. `extract 20`
- **History** : `history`
- **Undo** : `undo`
- **Redo** : `redo`
- **Clear** : `clear`
- **Export** : `export`
- **Import** : `import`
- **Exit** * : `exit`
- **Help** : `help`