Hotel Management System - User Guide

Welcome to the `HMS+ User Guide`.

Hope you are as excited as we were when we developed this guide for you.

By: `cs2103-ay1819s2-t12-1` Since: `Feb 2019` Licence: `MIT`

1. Introducing HMS+

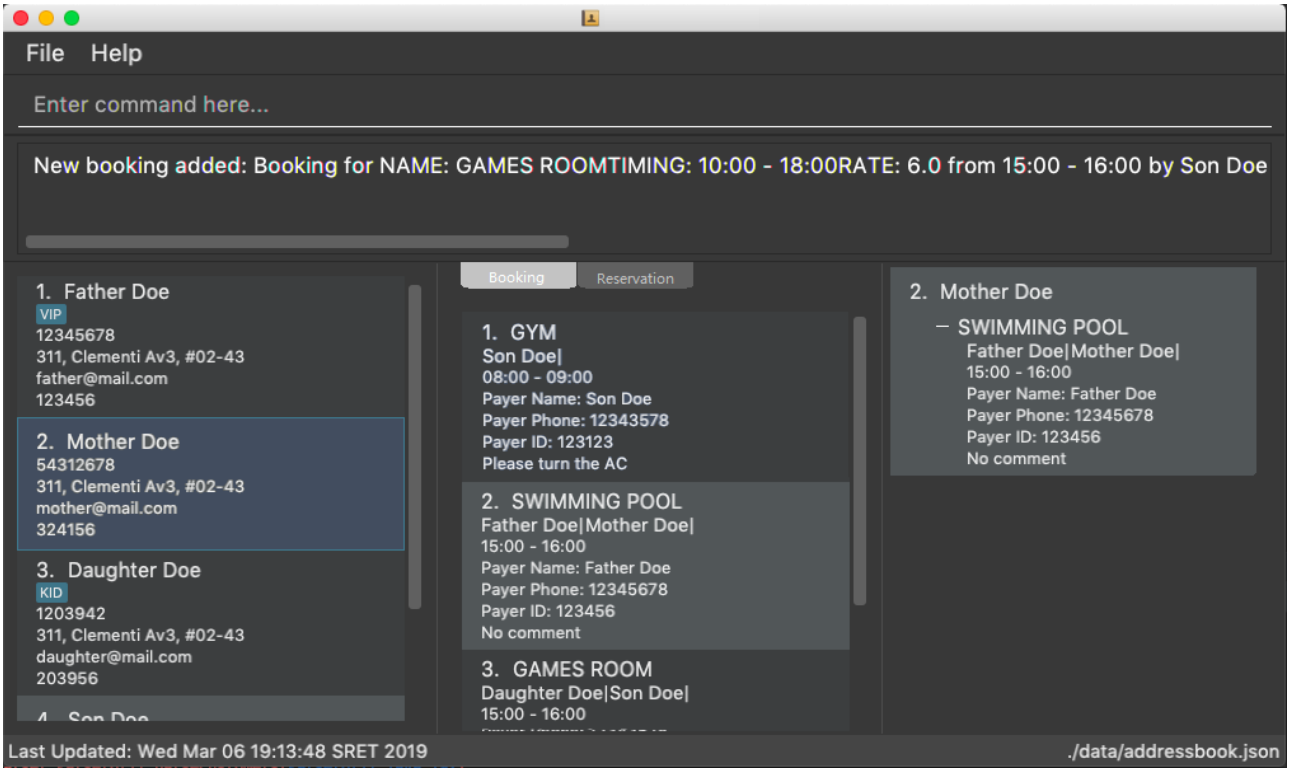
You need a world-class property management system to run your hotel's front desk and back office. We've spent years listening to what independent hoteliers need to take their business to the next level. From before check-in to post check out, we have everything you need to get your customers moving and your hotel running.

HMS+ is for hotels that want to use a single, integrated system to **manage all aspects of their requirements**. As a command line app, HMS is highly customisable to match the needs for different hotels and allows easy scripting for batch processing.

This user guide provides in-depth documentation on the HMS+ installation process, system configuration and management, monitoring and troubleshooting problems. In addition the Section 2, "Getting Started" will provide the basic knowledge of the setup process and how you can get started.

2. Getting Started

1. Ensure you have Java version `9` or later installed in your Computer.
2. Download the latest `hms.jar` here.
3. Copy the file to the folder you want to use as the home folder for your Hotel Management System.
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:
   - **`listcustomers`** : lists all customers
   - **`addcustomer`**`n/John Doe p/98765432 e/johnd@example.com id/887 a/John street, block 123, #01-01` : adds a customer named `John Doe` to the HMS+ database.
   - **`deletecustomer`**`3` : deletes the 3rd customer shown in the current list
   - **`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 in curly brackets separated by a slash (/) are interchangeable (entering either of those items have the same effect) e.g. `{addcustomer/ac}` are three aliases of the same command.
- 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`

Effect: Displays a help list, which lists all the commands that can be used.
Format: `help`

3.2. Adding a customer : `addcustomer`, `ac`

Effect: Adds a customer to the customer database.
Format: `{addcustomer/ac} n/NAME [p/PHONE_NUMBER] [e/EMAIL] [id/IDENTIFICATIONNO] [a/ADDRESS] [t/TAG]…`

> **TIP** A person can have any number of tags (including 0)

Examples:

- `addc n/John Doe p/98765432 dob/28/05/1999 e/johnd@example.com id/552526 a/John street, block 123, #01-01`
- `addc n/Betsy Crowe e/betsy.crowe@example.com p/123456 id/345252`

3.3. Listing all customers : `listcustomers`, `lc`

Effect: Displays a customer list, which lists all customers in the customer database.
Format: `{listcustomers/lc}`

3.4. Editing a customer : `editcustomer`, `ec`

Effect: Edits the fields of an existing customer in the customer database.
Format: `{editcustomer/ec} INDEX [n/NAME] [p/PHONE] [e/EMAIL] [id/IDENTIFICATIONNO] [a/ADDRESS] [t/TAG]…`

- Edits the customer at the specified INDEX. The index refers to the index number shown in the displayed customer list. The index must be a positive integer.
- At least one of the optional fields must be provided. Otherwise, nothing will be changed.
- Existing values will be updated to the input values.
- When editing tags, the existing tags of the customer will be removed, i.e. adding of tags is not cumulative.
- You can remove all the customer's tags by typing `t/` without specifying any tags after it.

Examples:

- `listc`, then `editc 1 p/91234567 e/johndoe@example.com`
  Edits the phone number and email address of the 1st customer to be 91234567 and johndoe@example.com respectively.
- `listc`, then `edit 2 n/Betsy Crower t/`
  Edits the name of the 2nd customer to be `Betsy Crower` and clears all existing tags.

3.5. Finding customers by name: `findname`, `fn`

Effect: Displays a customer list, which consists of customers whose names contain any of the given keywords.
Format: `{findname/fn} 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 full words will be matched, e.g. `Han` will not match `Hans`
- Persons matching at least one keyword will be returned (i.e. `OR` search). e.g. `Hans Bo` will return Hans Gruber and Bo Yang

Examples:

- `find John`
  Returns John Cena and John Doe
- `find Betsy Tim John`
  Returns any person having names Betsy, Tim, or John

3.6. Deleting a customer : `deletecustomer`, `deletec`

Effect: Deletes a customer from the customer database.
Format: `{deletecustome/deletec} INDEX`

- Deletes the customer at the specified index. The index refers to the index number shown in the displayed customer list. The index must be a positive integer.

Examples:

- `listcustomers`, then `deletecustomer 2`
  Deletes the 2nd person of the customer database.
- `findname Betsy`, then `deletecustomer 1`
  Deletes the 1st customer in the customer list returned by the `findname` command.

3.7. Reserving a room : `add-reservation`, `ar [coming in v2.0]`

Effect: Adds a booking for a room associated with certain customers.
Format: `{add-reservation/ar} r/ROOM_TYPE d/START_DATE-END_DATE i/INDEX_OF_CUSTOMER [i/MORE_INDICIES]… [c/COMMENTS]`

- `ROOM_TYPE` is a positive integer. Which number corresponds to which actual type is defined by the user.
- `START_DATE` and `END_DATE` follows the `DAY.MONTH` format.
- `COMMENTS` can contain any text without slash (/).

Examples:

- `listcustomers`, then `ar r/1 d/20.5-25.5 i/15`
  Adds a booking of Room Type 1, one customer from the complete customer list, from 20 May to 25 May.
- `findname Jack Rose`, then `add-reservation r/2 d/14.2-15.2 c/1 c/2`
  Adds a booking of Room Type 3, two customers from the search result of Jack and Rose, from 14 Feb to 15 Feb.

3.8. Listing all reservations : `list-reservations`, `lr [coming in v2.0]`

Effect: Displays a booking list, which lists one of: 1. all the bookings in the booking database; 2. the bookings associated with certain customers; 3. the bookings that contains a certain date.
Format: `{list-reservations/lr} [i/INDEX_OF_CUSTOMER]… [d/DATE]`

- If the indices are provided, the command shows only the bookings associated with any of the customers. The index refers to the index number shown in the displayed customer list. The index must be a positive integer.
- If a date is provided, the command shows only the bookings that span across that date. The date should follow the `DD.MM` format.
- Indicies and dates can be provided at the same time. The command will then show only the bookings associated with the customers that covers the date.

Examples:

- `listc`, then `lr i/2`
  Lists all the bookings under the name of the 2nd customer.
- `lr d/05.12`
  Lists all the bookings that spans across the date 12 May.

3.9. Editing a room reservation : `edit-reservation`, `er` `[coming in v2.0]`

Effect: Edits the fields of an existing booking in the booking database.
Format: `{edit-reservation/er} INDEX [r/ROOM_TYPE] [d/START_DATE-END_DATE] [c/COMMENTS]`

- Edits the booking at the specified index. The index refers to the index number shown in the displayed booking list. The index must be a positive integer.
- At least one of the optional fields must be provided. Otherwise, nothing will be changed.
- Changing the associated customers is forbidden because that may lead to billing issues. If that is desired, delete the existing booking and create a new one.
- Existing values will be updated to the input values.
- When editing comments, the existing comments of the booking will be removed, i.e adding of comments is not cumulative.
- You can remove all the booking's comments by typing `c/` without specifying any tags after it.

Examples:

- `listb`, then `er 1 r/3`
  Edits the room type of the 1st booking to be Type 3.
- `listb`, then `er 2 d/14.2-14.3 c/`
  Edits the date of the 2nd booking to be from 14 Feb to 14 Mar and clears all existing comments.

3.10. Deleting reservations: `delete-reservation`, `dr` `[coming in v2.0]`

Effect: Deletes a booking from the booking database.
Format: `{delete-reservation/dr} INDEX`

> - Deletes the reservation at the specified index. The index refers to the index number shown in the displayed booking list. The index must be a positive integer.

Examples:

- `listbookings`, then `deletebooking 2`
  Deletes the 2nd person of the booking database
- `listb i/2`, then `deletebooking 1`
  Deletes the 1st booking in the results of the listb command.

3.11. Booking a service: `add-booking`,`ab`

Effect: Adds a service associated with certain customers.
Format: `{add-booking/ab} s/SERVICE_NAME :/START_TIME-END_TIME $/PAYER INDEX [c/CUSTOMER INDEX] [com/COMMENTS]`

> - `SERVICE_TYPE` is a string. Which corresponds to which service type is defined by the user.
> - `START_TIME` and `END_TIME` follows the `HH 24-hour` format.
> - `COMMENTS` can contain any text without slash (`/`).

Examples:

- `listc`, then `add-booking s/GYM :/20-23 $/2 Adds a booking for service GYM, for the 2nd customer from the complete customer list, from 20:00 to 23:00 if the service is available.
- `findn Jack Rose`, then `add-booking s/GYM h/14-15 $/1 c/2` Adds a booking of service GYM for customer index 2 payed by customer index 1, from 14:00 to 15:00.

3.12. Listing all booked services: `listbookings`,`lb`

Effect: Displays a booking list, which lists all the bookings made till now. Format: `{listbookings/lb}`

3.13. Editing a booked service: `edit-booking`, `eb`

Effect: Edits the fields of a booking in the database.
Format: `{edit-booking/eb} INDEX [s/SERVICE_NAME] [:/START_TIME - END_TIME] [p/PAYER INDEX] [c/CUSTOMER INDICES] [com/COMMENTS]`

- Edits the booking at the specified index. The index refers to the index number shown in the displayed booking list. The index must be a positive integer.
- At least one of the optional fields must be provided. Otherwise, nothing will be changed.
- Existing values will be updated to the input values.
- When editing comments, the existing comments of the booking will be removed, i.e adding of comments is not cumulative.
- You can remove all the booking's comments by typing `com/` without specifying any tags after it.

Examples:

- `lb`, then `eb 1 s/GYM` Edits the service type of the 1st booking to be GYM.
- `lb`, then `edit-booking 2 :/14-15 c/` Edits the timing of the 2nd booking to be 14:00 - 15:00 and clears all existing comments.

3.14. Find a booked service contains payer: `findbookingcontainspayer`, `fbcp`

Effect: Displays a booking list, which is paid by the customer whose identification number is the same as the given number Format: `{findbookingcontainspayer/fbcp} PAYER_IDENTIFICATION_NUMBER`

- The searching is done in the whole booking list.

Examples: * `fbcs 12345678`
Returns any booking which is paid by the customer with identification number 12345678

3.15. Deleting a booked service: `deletebooking`, `delb`

Effect: Deletes a booking from the database. Format: `{deletebooking/delb} INDEX`

- Deletes the booking at the specified index. The index refers to the index number shown in the displayed room service list. The index must be a positive integer.

Examples: * `lb`, then `deletebooking 2` Deletes the 2nd booking of the booking database

3.16. Generate customer's bill : `generatebill`, `gb [coming in v2.0]`

Effect: Generates the bill for the customer based on his room reservations plus service bookings less the amount the customer has already paid+ Format: `{generatebill/gb} INDEX`

- Generates the bill for the customer at the specified index.

Examples: * `listc`, then `gb 2` Generates the bill for the second customer.

3.17. Listing entered commands : `history`

Effect: 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.

3.18. Undoing previous command : `undo`

Effect: Restores the address book to the state before the previous *undoable* command was executed.
Format: `undo`

NOTE	Undoable commands: those commands that modify HMS's content (`addc`, `deletecustomer`, `editc`, etc.).

Examples:

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

3.19. Redoing the previously undone command : `redo`

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

Examples:

- `deletecustomer 1`
  `undo` (reverses the `deletecustomer 1` command)
  `redo` (reapplies the `deletecustomer 1` command)
- `deletecustomer 1`
  `redo`

The `redo` command fails as there are no `undo` commands executed previously.

- `deletecustomer 1`
  `clear`
  `undo` (reverses the `clear` command)
  `undo` (reverses the `deletecustomer 1` command)
  `redo` (reapplies the `deletecustomer 1` command)
  `redo` (reapplies the `clear` command)

3.20. Clearing all entries : `clearcustomers`

Effect: Clears all entries from the customer database.
Format: `clearcustomers`

3.21. Exiting the program : `exit`

Effect: Exits the program.
Format: `exit`

3.22. Saving the data

The HMS+ data file is saved in the hard disk automatically after any command that changes the data.
There is no need to save manually.

3.23. Encrypting data files `[coming in v2.0]`

{explain how the user can enable/disable data encryption}

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 HMS folder.

5. Command Summary

- **Help** : `help`
- **Add Customer** : `{addcustomer/addc/ac} n/NAME [p/PHONE_NUMBER] [e/EMAIL] [id/IDENTIFICATIONNO] [a/ADDRESS] [t/TAG]…`
- **List Customers** : `{listcustomers/listc/lc}`
- **Edit Customer** : `{editcustomer/editc/ec} INDEX [n/NAME] [p/PHONE] [e/EMAIL] [id/IDENTIFICATIONNO] [a/ADDRESS] [t/TAG]…`
- **Find Customer by name** : `{findname/findn/fn} KEYWORD [MORE_KEYWORDS]`

- **Delete Customer** : `deletecustomer INDEX`
- **Reserve room** : `{addservice/adds/as} s/SERVICE_TYPE h/START_TIME-END_TIME i/INDEX [i/MORE_INDICES] [c/COMMENTS]`
- **List room reservations** : `{listservices/lists/ls} [i/INDEX_OF_CUSTOMER]… [h/START_TIME - END_TIME]`
- **Edit room reservaitons** : `{editservice/edits/es} INDEX [s/SERVICE_TYPE] [h/START_TIME - END_TIME] [c/COMMENTS]`
- **Delete room reservation** : `deleteservice INDEX`
- **Book services of hotel** : `{addbooking/addb/ab} r/ROOM_TYPE d/START_DATE-END_DATE i/INDEX_OF_CUSTOMER [i/MORE_INDICIES]… [c/COMMENTS]`
- **List services already booked** : `{listbookings/listb/lb} [i/INDEX_OF_CUSTOMER]… [d/DATE]`
- **Edit services already booked** : `{editbooking/editb/eb} INDEX [r/ROOM_TYPE] [d/START_DATE-END_DATE] [c/COMMENTS]`
- **Delete service already booked** : `deletebooking INDEX`
- **Generate bill** : `{generatebill/gb} INDEX`
- **History** : `history`
- **Undo** : `undo`
- **Redo** : `redo`
- **Clear customers** : `clearcustomers`
- **Clear room reservations** : `clearreservations`
- **Clear room services** : `clearbookings`