

SugarMummy - User Guide

1. Introduction	2
2. Quick Start	5
3. Features	6
3.1. DATA LOG	6
3.1.1. Add a data entry to the diabetes tracker app: <code>add</code>	6
3.1.2. Show a list of all data entries in the diabetes tracker app: <code>list</code>	7
3.1.3. Deletes a specified data entry from the diabetes tracker app: <code>delete</code>	7
3.1.4. Edits a specified data entry from the diabetes tracker app: <code>edit [coming in v2.0]</code>	7
3.2. CALENDAR AND REMINDER	7
3.2.1. Add a reminder: <code>reminder</code>	7
3.2.2. Add an event: <code>event</code>	8
3.2.3. View calendar entries: <code>calendar</code>	9
3.2.4. Snooze reminder: <code>snooze [coming in v2.0]</code>	9
3.2.5. Complete a task: <code>complete [coming in v2.0]</code>	9
3.2.6. Set time zone: <code>timezone [coming in v2.0]</code>	10
3.2.7. Cancel an entry in calendar <code>[coming in v2.0]</code>	10
3.3. PERSONALISED USER EXPERIENCE	10
3.3.1. Shows all the achievements that the user has attained: <code>achvm</code>	10
3.3.2. Displays the user's biography': <code>bio</code>	12
3.3.3. Adds the user's biography: <code>addbio</code>	13
3.3.4. Edits the user's biography: <code>editbio</code>	16
3.3.5. Clears the user's biography: <code>clrbio</code>	18
3.3.6. Sets the font color of the application: <code>fontcolour</code> or <code>fontcolor</code>	19
3.3.7. Sets the background image or colour of the application: <code>bg</code>	21
3.3.8. Displays a motivational quote: <code>[no action required]</code>	25
3.3.9. Saves the user's preferred themes: <code>[coming in v2.0]</code>	25
3.3.10. Shows a cartoon avatar that represents the user: <code>[coming in v2.0]</code>	25
3.3.11. Follows up on the user's goals: <code>[coming in v2.0]</code>	26
3.4. DATA SUMMARY/ANALYSIS	26
3.4.1. Displays the daily/weekly/monthly average of records in a line graph: <code>average</code>	26
3.4.2. Summary statistics of a particular record type <code>[coming in v2.0]</code>	26
3.4.3. Shows relationship between record types: <code>[coming in v2.0]</code>	26
3.4.4. Exports summary of all medical records into pdf <code>[coming in v2.0]</code>	26
3.5. Food Recommendation	27
3.5.1. Recommending food: <code>recmf</code>	27
3.5.2. Recommending food combination: <code>recmfmix</code>	28
3.5.3. Adding new food items : <code>addfood</code>	28
3.5.4. Deleting an existing food: <code>deletef</code>	29

3.5.5. Resetting food data: <code>resetf</code>	29
3.5.6. Recovering food data: <code>recoverf</code> coming in v2.0	29
3.5.7. Editing a food: <code>editf</code> coming in v2.0	29
3.5.8. Recording and analyzing diets coming in v2.0	29
3.6. Other Common Features	29
3.6.1. Exiting the program : <code>exit</code>	30
3.6.2. Viewing full User Guide	30
4. Features Coming in v2.0	30
4.1. Autocomplete Command	30
4.2. Clearing all entries: <code>clearall</code>	30
4.3. Editing specific record entries in the list: <code>edit INDEX [h/VALUE] [w/VALUE] [con/VALUE]</code>	30
4.4. Encrypting data files: <code>enc [KEY]</code>	30
4.5. Detailing workout schedule	30
4.6. Timing Workout	30
4.7. Sharing with community	31
5. Command Summary	31
6. FAQ	32

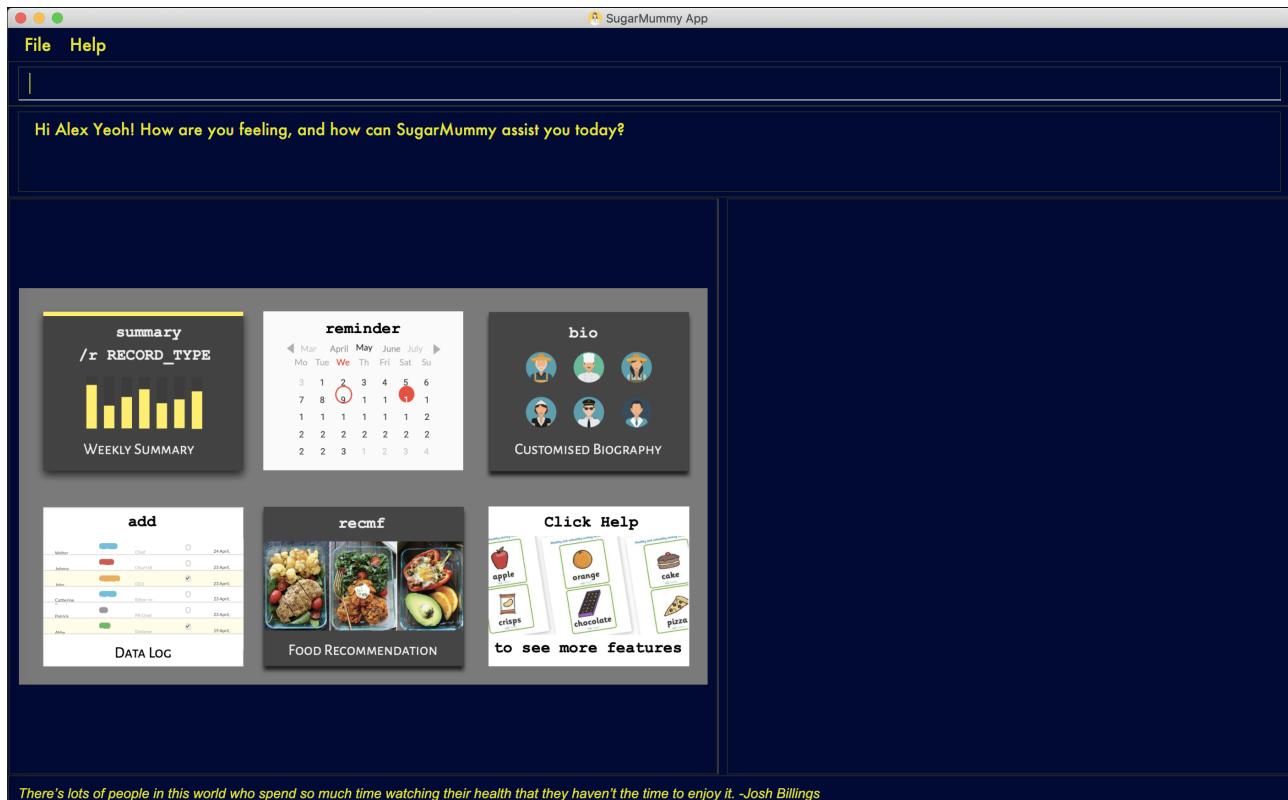
By: Team CS2103-T16-1 Since: Sep 2019 Licence: MIT

1. Introduction

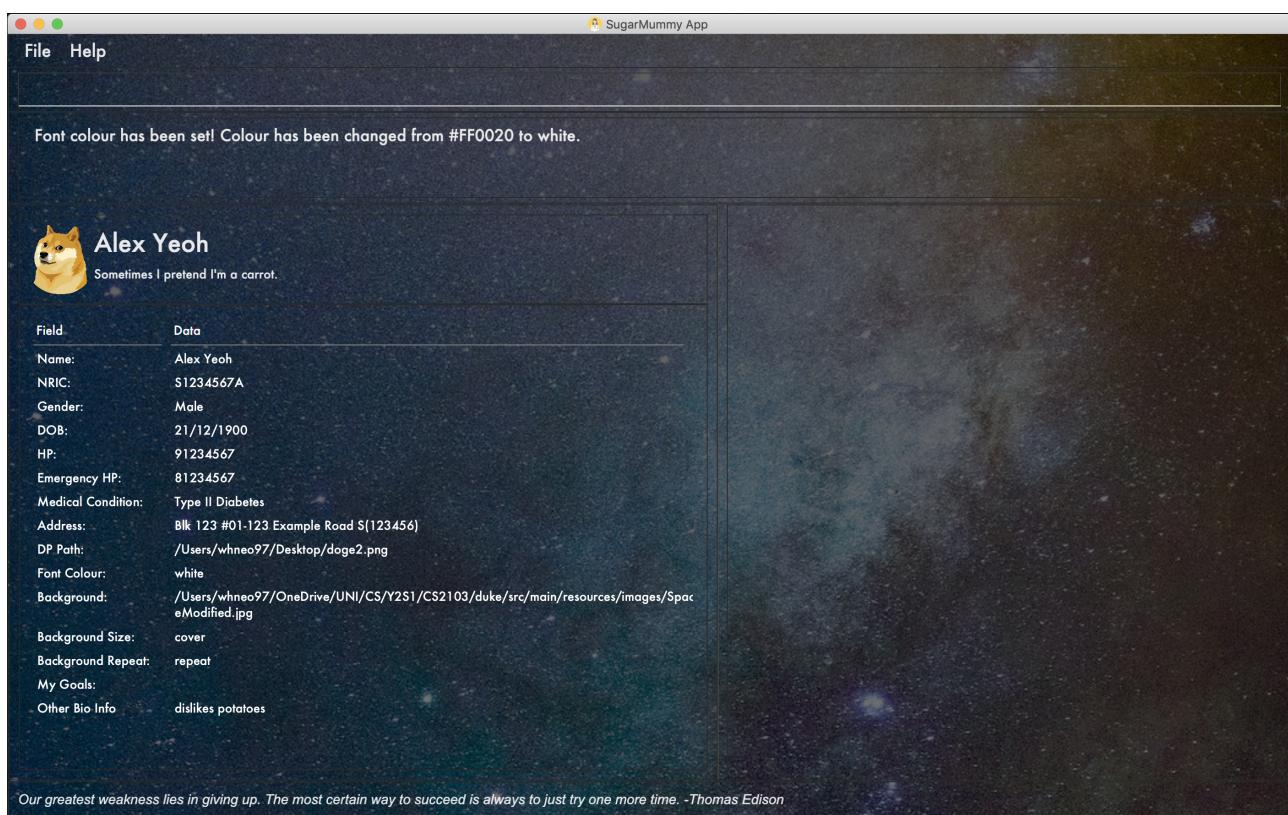
Do you feel it difficult to keep the right diets for type II diabetes? Do you want to manage your hectic schedule? Do you want to have a better overview of your health data? Do you want to move to a healthier lifestyle? If your answer is yes, then SugarMummy is perfect for you! SugarMummy is a lifestyle tracker and planner that aims to combat Type II diabetes by encouraging users to keep a healthy lifestyle. Moreover, being a computer app with Command Line Interface, SugarMummy is optimized for fast typers who has access to a computer most of the time.

Hope following features will benefit you and enjoy!

- Data tracker for various data types (blood sugar levels and bmi)
- Easy-to-use reminder and calendar
- Personalized user experience (achievements, biography, and motivational quotes)
- Daily, Weekly and Monthly graph summary for your activities and health level
- Customized recommendations for healthy dieting



There's lots of people in this world who spend so much time watching their health that they haven't the time to enjoy it. -Josh Billings



Our greatest weakness lies in giving up. The most certain way to succeed is always to just try one more time. -Thomas Edison

SugarMummy App

File Help

Listed all records

DateTime: 2020-04-14 12:12
Height: 160.32514802865327
Weight: 49.7458912973871

314. Bmi
DateTime: 2020-04-15 12:12
Height: 170.06875560404053
Weight: 64.41136606792642

315. Bmi
DateTime: 2020-04-16 12:12
Height: 171.0005399855396
Weight: 66.87511845889087

316. Bmi
DateTime: 2020-04-17 12:12
Height: 171.5097071460908
Weight: 61.45350326723701

317. Bmi
DateTime: 2020-04-18 12:12
Height: 162.47252437144348
Weight: 63.637832994453596

318. Bmi
DateTime: 2020-04-19 12:12
Height: 161.4993366965636
Weight: 49.379073215845054

The day we decide to drop the flimsy makeshift scenarios in our cluttered mind and eschew the 'alleluias' of self-importance, life can become genuine, lucid and graceful, like a flow of wellness in the glow of a new morning. -Erik Pevernagie

SugarMummy App

File Help

You DAILY averages for BLOODSUGAR have been calculated successfully.

DAILY average of blood sugar

Date	Blood Sugar (mmol/L)
2019-12-01	~7.8
2019-12-02	~4.1
2019-12-03	~6.7
2020-01-27	~4.4
2020-02-08	~4.6

blood sugar [mmol/L]

day

Before meals (Non-diabetic): Between 4.0 and 5.9 mmol/L

At least 90 minutes after meals (Non-diabetic): Between 5.9 and 7.8 mmol/L

A sad soul can kill you quicker than a germ. -John Steinbeck

SugarMummy App

File Help

Calendar of 2019-11 shown

2019 November

Mon	Tue	Wed	Thu	Fri	Sat	Sun
				1 1	2 1	3 1
4 1	5 1	6 1	7 1	8 1	9 1	10 1
11 1	12 1	13 1	14 1	15 1	16 1	17 1
18 1	19 1	20 1	21 1	22 1	23 1	24 1
25 1	26 1	27 1	28 1	29 1	30 1	1

Nov 1 Fri
07:53
Reminder: Insulin injection

Nov 2 Sat
07:53
Reminder: Insulin injection

Nov 3 Sun
07:53
Reminder: Insulin injection

Nov 4 Mon
07:53
Reminder: Insulin injection

An Active Mind Cannot Exist In An Inactive Body -Anonymous



SugarMummy App

File Help

Background has been set!
- Background has been changed from #202020 to #303030.

My Achievements

Hi Amy, here are the list of achievements you have collected so far.

BLOODSUGAR

 Title: Sugar Control Apprentice Level: BRONZE Current State: ACHIEVED Requirement: Attain between 4.0 and 7.8mmol/L (inclusive) of daily average bloodsugar levels for at least 3 consecutive days.	 Title: Sugar Manipulation Master Level: SILVER Current State: ACHIEVED Requirement: Attain between 4.0 and 7.8mmol/L (inclusive) of daily average bloodsugar levels for at least 14 consecutive days.	 Title: Sugar Legendary Whisperer Level: GOLD Current State: ACHIEVED Requirement: Attain between 4.0 and 7.8mmol/L (inclusive) of daily average bloodsugar levels for at least 30 consecutive days.	 Title: Sugar Elemental Mage Level: PLATINUM Current State: YET TO ACHIEVE Requirement: Attain between 4.0 and 7.8mmol/L (inclusive) of daily average bloodsugar levels for at least 90 consecutive days.	 Title: Grand Sorcerer Supreme of the Sugar Arts Level: DIAMOND Current State: YET TO ACHIEVE Requirement: Attain between 4.0 and 7.8mmol/L (inclusive) of daily average bloodsugar levels for at least 180 consecutive days.
---	---	---	--	--

BMI



We are indeed much more than what we eat, but what we eat can nevertheless help us to be much more than what we are. -Adelle Davis

2. Quick Start

1. Download the latest [sugarmummy.jar](#) [here](#).
2. Copy the file to the folder you want to use as the home folder for data records.
3. Double-click the file to start the app. The GUI should appear in a few seconds.

4. 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.
5. Some example commands you can try:
 - **addrt/BLOODSUGAR con/10 dt/2019-11-01 12:12** : adds a **bloodsugar** record to the app.
 - **delete3** : deletes the 3rd record shown in the current list
 - **exit** : exits the app
6. 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 **reminder d/DESCRIPTION**, **DESCRIPTION** is a parameter which can be used as **reminder d/medicine**.
- Items in square brackets are optional e.g **d/DESCRIPTION [r/REPETITION]** can be used as **d/medicine r/daily** or as **d/medicine**.
- 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 **d/description dt/2020-01-20 12:00, dt/2020-01-20 12:00 d/description** is also acceptable.

3.1. DATA LOG

3.1.1. Add a data entry to the diabetes tracker app: **add**

Format: **add rt/RECORD_TYPE dt/DATETIME** followed by parameters that the RECORD_TYPE accepts.

There must not be any duplicate parameters. Ordering of parameters is flexible (other than "add").

rt/ value is case sensitive, RECORD_TYPE must be in capital letters. **dt/** value must not be in the future.

BMI must receive exactly 1 valid parameter each for height and weight: **h/VALUE w/VALUE**

Height input (in meters) will be rounded to 2 decimal places. This rounded value must be less than 3 to be recorded successfully.

Weight input (in kilograms) will be rounded to 2 decimal places. This rounded value must be less than 500 to be recorded successfully.

BLOODSUGAR must receive exactly 1 valid parameter for concentration: **con/VALUE** Concentration (mmol/L) will be rounded to 2 decimal places. This rounded value must be less than 33 to be recorded successfully.

`add` does not allow entries that have the same datetime and RECORD_TYPE.

3.1.2. Show a list of all data entries in the diabetes tracker app: `list`

Format: `list`

This command lists all types of recent data entries.

3.1.3. Deletes a specified data entry from the diabetes tracker app: `delete`

Format: `delete INDEX`

Index must be a positive integer.

The index refers to the index number shown in the displayed entries list

3.1.4. Edits a specified data entry from the diabetes tracker app: `edit` [coming in v2.0]

3.2. CALENDAR AND REMINDER

Command arguments for calendar and reminder:

- `d/DESCRIPTION` Description can take any values, and it should not be blank. If more than one are present, the last argument is taken.
- `dt/DATETIME` The format of date time is: `yyyy-mm-dd hh:mm` and the number should be valid. If more than one are present and the command only accepts one `dt/`, the last argument is taken. Otherwise, see below in `event` command for more details.
- `r/REPETITION` Repetition can only take value `once`, `daily` or `weekly`(case insensitive). If more than one are present, the last argument is taken.
- `td/TIMEDURATION` The format of time duration is: `hh:mm` and it should not be blank. e.g. `01:30` represents 1 hour and 30 minutes. If more than one are present, the last argument is taken.
- `ym/YEAR_MONTH` The format of year month is: `yyyy-mm` and the number should be valid. If more than one are present, the last argument is taken.
- `ymd/YEAR_MONTH_DAY` The format of year month day is: `yyyy-mm-dd` and the number should be valid. If more than one are present, the last argument is taken.
- `ymw/YEAR_MONTH_DAY` The format of year month day is: `yyyy-dd-dd` and the number should be valid(same as year month day). If more than one are present, the last argument is taken. See below in `calendar` command for more details if more than one of `ym/`, `ymd`, `ymw/` are present.

3.2.1. Add a reminder: `reminder`

Format: `reminder d/DESCRIPTION dt/DATETIME [r/REPETITION]`

Add a reminder at a specific time and date. There is no repetition by default and user can specify a daily or weekly or no repetition optionally. If it is a repeated reminder, the date inputted is the starting date of the reminder(reminder with date time before the current date time is allowed). And

it provides duplicate and conflicting checking as well as auto merging for reminders added. For example, `reminder d/insulin inject dt/2019-11-25 17:30 r/daily` adds a new reminder of insulin injection at 17:30 everyday from 2019 Nov 25.

- More on duplicate and conflicting check and auto merging
 - To avoid duplicate reminders, the app will not accept reminders that already exist in the system(same description, date time and repetition). Similarly, reminders that are completely covered by an existing reminder will not be added again. For example, if the previous reminder command is successful, `reminder d/insulin inject dt/2019-12-01 17:30 r/weekly` will not add the new reminder because it is completely covered (whenever the later one shows up, the previous one shows up at the same time with the same description). `reminder d/insulin inject dt/2019-12-01 12:00 r/daily` is not considered covered because there could be two injections one day.
 - Moreover, new reminder that covers one or more existing reminders can be added and those existing ones will be removed automatically. For example, if `reminder d/insulin inject dt/2019-11-20 17:30 r/daily` is typed after the previous one, it replaces the insulin inject from 2019-11-25 because whenever the previous reminder shows up, the new reminder shows up with the same description.
 - However, if the new reminder overlaps with existing ones but they cannot be merged, it will not be added. To illustrate, `reminder d/insulin inject dt/2019-11-18 17:30 r/weekly` is rejected because on 2019 Nov 18 it shows up at 17:30 alone. After 2019 Nov 20, this reminder will be a duplicate of the previous one every Monday.
- Reminder list side pane
 - On the right side of the app window, there is a pane for reminder list with the date showing on the top. The reminder list pane is split into two parts: a real-time reminder list on top of a probably missed reminder list.
 - The real-time reminder list is empty if there is no reminder set at current time. If the app is open, the reminders will show up in the pane at the time set by the user. For example, if the user opens the app at 17:00 on 2019 Nov 30. A reminder of insulin injection will show up at 17:30 if the user has not closed the app.
 - The probably missed reminder list includes all reminders that should exist in at the time between the starting of day and the time when the user opens the app. For example, if the user opens the app at 17:00 on 2019 Nov 30, it lists all reminders from 00:00 to 16:59 on that day. However, if the user adds a reminder on 2019 Nov 30 17:10 at 17:20 on that day, it will neither exist in the real-time reminder list pane nor the missed reminder list pane.

3.2.2. Add an event: `event`

Format: `event d/DESCRIPTION dt/DATETIME [dt/DATETIME] [td/TIME_DURATION]`

Add an event with a starting time and an optional ending time(starting time before the current date time is accepted). There is no reminder for an event by default. User can also create a reminder for this event by specifying a time duration and a one time reminder will show up at the starting time of the event minus this time duration. For example, `event d/appointment dt/2019-12-20 14:00 dt/2019-12-20 15:00 td/01:30` adds an appointment event on 2019 Dec 20 15:00 to the calendar and a reminder for it at 13:30 on that day.

If only one `dt/` is present, it will be recognized as the starting date time. If there are more than one `dt/`, the first and second one are interpreted as starting and ending date time. Starting date time should come before ending date time. If they are equal, then ending date time is discarded. A duplicate event(same description and starting date time) will be rejected. However, events that overlap with existing ones are accepted with a message shown to the user. For example, `event d/appointment dt/2019-12-20 14:00 dt/2019-12-20 16:00` will be rejected after the previous event commands. `event d/test dt/2019-12-20 13:00 dt/2019-12-20 14:10` is accepted with a message shown.

3.2.3. View calendar entries: `calendar`

Format: `calendar [ym/YEAR_MONTH] [ymw/YEAR_MONTH_DAY] [ymd/YEAR_MONTH_DAY]`

View all calendar entries or calendar entries in a month or in a week or on a day.

- If none of the three arguments is present, it will show all calendar entries added by the user.
- If one or more of the three arguments are present, it accepts the one with the smallest time unit.
e.g. if `ymw` and `ymd` are both typed in, it only shows entries on the given date if the date is valid.
 - `ymd/` shows the monthly calendar of given date and below the calendar is a list of reminders and events on that day. Particularly, if no date is provided, it will show calendar entries on the current date.
 - `ymw/` shows the monthly calendar of given date and below the calendar is a list of reminders and events on each day in the week. `ymw/` accepts a date also and the week picked is 7 days from Monday to Friday which include the given date. Particularly, if no date is provided, it will show calendar entries in the current week. For example, `calendar ymw/2019-12-03` will show a monthly calendar of 2019 December and calendar entries from Dec 2 to Dec 8.
 - `ym/` shows the monthly calendar of given month and below the calendar is a list of reminders and events on each day in the month. `ym/` accepts a month. Particularly, if no date is provided, it will show calendar entries in the current week.
- For the monthly calendar shown, the subscript number beside each date is the number events and reminders on that day. And the current date is in a different color.

3.2.4. Snooze reminder: `snooze [coming in v2.0]`

Format: `snooze [tp/TIME_PERIOD]`

Silence the current reminder and activate it after a time period.

3.2.5. Complete a task: `complete [coming in v2.0]`

Format: `complete [INDEX] [t/TIME]`

Mark all the reminders before now as completed if no argument provided. Otherwise, mark only those tasks with indices provided or before the specific time as completed.

3.2.6. Set time zone: `timezone` [coming in v2.0]

Format: `timezone tz/TIME_ZONE [t/TIME] [t/TIME_END]`

Set the time zone of the application permanently or in any time interval(e.g. For travelling).

3.2.7. Cancel an entry in calendar [coming in v2.0]

Format: `cancel [et/ENTRYTYPE] [d/DESCRIPTION] [dt/DATETIME] [dt/DATETIME]`

Delete an event that you do not want to keep track anymore.

3.3. PERSONALISED USER EXPERIENCE

3.3.1. Shows all the achievements that the user has attained: `achvm`

Format: `achvm`

Displays the current list of achievements attained by the user, categorised by record type. Each achievement has a picture that represents it, a title, level, state and requirement required to attain the achievement. All these information are available upon the entering of the `achvm` command. However, the user would only be able to see, for each achievement, the corresponding picture that represents the current state of the achievement (eg. if a user has yet to achieve the achievement, he / she will only see a silhouette image of the character in the picture rather than the actual image. Achieving the achievement will allow the user to "unlock" its coloured image).

For each record type, there are five levels of achievements available:

- Bronze (3 consecutive days of meeting requirements)
- Silver (14 consecutive days of meeting requirements)
- Gold (30 consecutive days of meeting requirements)
- Platinum (90 consecutive days of meeting requirements)
- Diamond (180 consecutive days of meeting requirements)

For each achievement, there are 3 possible states:

- Achieved (coloured picture) - met requirements for the most recent number of required consecutive days
- Previously Achieved (grayscale picture) - met requirements before in the records but more recent records no longer meet the number of consecutive days required. i.e. broke streak
- Yet to Achieve (silhouette picture) - No part of the records meet the consecutive days of requirement.

(All images representing the different achievements and their respective states are hand-drawn and digitally coloured.)

Daily averages for the user's current records are used to determine whether the requirements to

attain the achievement has been met. Upon adding or deleting a record, the user is notified if there happens to be any new achievements attained and / or lost. The user can then key in the **achvm** command to check the new achievements if any. Note that the **achvm** command, however, can be keyed in any time; current achievements are always displayed dynamically regardless at any point of time.

The daily average requirement for each currently known record type are as follows:

- Blood Sugar: 4.0 to 7.8 mmol/L
- BMI: 18.5 to 25

A new user and / or user without any records have no achievements by default (i.e. all achievements will be in a state Yet To Achieve, represented by silhouette pictures).

Suppose a user has no records and adds the following records.

- `add rt/BLOODSUGAR dt/2019-02-07 12:12 con/4.5`
- `add rt/BLOODSUGAR dt/2019-02-08 12:12 con/4.5`
- `add rt/BLOODSUGAR dt/2019-02-09 12:12 con/4.5`

Average daily bloodsugar levels are clearly within the required range of 4.0 and 7.8 mmol/L (inclusive). After keying in the third record, the user not only sees that message indicating successful addition of the record, but also an additional message indicating that (a) new achievement(s) have been attained, and suggesting to check achievements. The achievements pane of the user will now indicate attainment of a bronze achievement for bloodsugar (Now a coloured image from what was previously a silhouette image and current state of ACHIEVED).

- If the user at this point decides to delete the third bloodsugar record, a notification will indicate that achievements are lost, and on the achievements pane the user will lose the achievement (back to silhouette image and YET TO ACHIEVE state)
- If the user has a new average bloodsugar record the next day and it still fulfills the requirement, the user adds to the streak and is on the way towards the next level of attainment for bloodsugar.
- If the user's new average record for the next day no longer meets the requirements for bloodsugar levels, the achievement that was originally attained will now have a state of PREVIOUSLY ACHIEVED instead and will be in grayscale colour. A notification is indicated to the user upon the user entering a new record that results in this change.
- If the user's subsequent record is not within the next day, the user also loses his / or her streak, and needs to restart his / her count towards the bronze achievement. However, as there is evidence of a three-day streak of meeting the requirements for bloodsugar, the user still attains a PREVIOUSLY ACHIEVED state for the bronze achievement. The user may fill in the missing day of record in order to maintain the streak or restart the count from the new day onwards. (The reason for this is the program cannot award achievements without data that justifies it)

The streak encourages users to keep up good health performances and healthy habits that lead to the results observed. Using the average records as a gauge or disregarding missing days of records would have allowed users to 'cheat' in between, but at the same time, users may just so happen to

miss out filling in one day of record (hence implementation of the PREVIOUSLY ACHIEVED feature). Maintaining their records at least once a day encourages users to self-monitor and keep their blood sugar levels and BMI in check, with achievements to keep them going.

To enhance performance, the application internally caches the list of achievements. So long as there is no modification to the list of achievements, the same pane as before is displayed without retrieving images representing the achievements each time `achvm` is called. This means that after the first time a specific achievements page is displayed, subsequent loading is much faster for the user! =) Of course, this is until the next update of achievements, or until the application is restarted, though loading itself is already optimised with image resolution sizes that are just sufficient for the page, and most of the time the user should not notice any significant differences at all whether the achievements are loaded for the first time. This means that regardless of the device the user is using, the pane is more or less guaranteed to run smoothly without any notable performance issues with efficiency. Ultimately, in short, measures have been taken to ensure quality in user experience, especially for a feature that focuses on *personalised* user experience =)

The command word, as for all other commands, is not case-sensitive for convenience to user.

3.3.2. Displays the user's biography': `bio`

Format: `bio`

Displays a pane containing user information such as the user's profile picture, name, NRIC, gender, date of birth, contact number, address and other biography information that the user would like to include.

The pane first includes the user's name, profile picture and profile description, followed by biography information displayed in a table with the following fields:

- Name
- NRIC
- Gender
- DOB
- HP
- Emergency HP
- Medical Conditions:
- Address
- DP Path
- Font Colour
- Background
- Background Size
- Background Repeat
- My Goals
- Other Bio Info

Note that the bio table here also includes the user's preferences such as font and background, as we believe these aesthetics information could also represent the user in one way or another. Furthermore, this is the page that the user may refer to for an overview of important information.

All data in the table changes dynamically with change in information.

If a field that allows for more than one value is displayed, the information is presented in the form of a numbered list.

If a field contains no information, the field name would still be displayed but its corresponding data would simply be blank.

Note that to enhance performance, the profile picture only changes when the user specifies a change in the path name. This means that if a user uses "doge.png" as the profile picture, and the source file "doge.png" is replaced with a new image during the running of the application, this change isn't immediately reflected until "doge.png" in the system is changed to something else such as "doge2.png" and back to "doge.png", or the application is restarted. This ensures that with changes to other fields, the page does not unnecessarily reload the same image and cause lags in user performance. This is especially so when the user has a lot of information to enter and modify. More information on the adding of biography information is described in the following subsection.

If the biography file storing the biography information is corrupted, a new empty biography with a default profile picture is displayed to the user.

This is with the exception of **DP PATH**. This is because it is much more likely that the display picture file cannot be found (i.e relocated or renamed in the user's device) than for other information, that should only have been edited by the user via the application, to be corrupted. We don't want a situation such that if the original file is deleted, replaced or relocated, the user loses all other biography data. As such, if **DP PATH** can no longer be loaded as an image, the default profile picture is used instead, and the user is notified that the display picture cannot be found.

If the biography file storing the biography information is deleted or cannot be found, a sample biography with the default profile picture is loaded instead.

This command cannot have any sub-arguments, and thus, if the user enters **bio 1**, an error message will be displayed, as the user's intention may not have been necessarily to show the biography. Trailing spaces are automatically trimmed and as such pose no issue.

Upon loading, a message is included in the feedback display pane to remind users to keep their biography data safe.

The command word, as for all all other commands, is not case-sensitive for convenience to user.

3.3.3. Adds the user's biography: **addbio**

Format: **addbio n/NAME [dp/DP PATH] [desc/PROFILE DESCRIPTION] [nric/NRIC] [g/GENDER] [dob/DATE OF BIRTH] p/CONTACT NUMBER… e/EMERGENCY CONTACT… m/MEDICAL CONDITION… [a/ADDRESS] [goal/GOAL]… [o/OTHER BIO INFO]**

A user may add a biography if there isn't already an existing one stored in the application. This could occur if the storage file is corrupted (refer to above sub-section on **bio**), or if the user decides

to clear the biography (refer to following sub-section on `clrbio` command). A user may add **at most one** biography. If a user attempts to add a biography when one already exists, an error message will be shown for the user to either `editbio` instead or `clrbio` before adding a new one. Support for multiple user biographies is presently not available nor intended, as the application is meant to be fully experienced solely by a single user for maximised personalised user experience. As the fields used by `addbio` and `editbio` commands are identical and can appear in any order, a user who attempts to edit fields using the `addbio` command when a biography already exists, after receiving the error message, can simply amend `editbio` to `addbio` before re-entering the line of command, without having to re-enter all the fields.

It is recommended for users to have a biography as they are most likely from the target audience group of patients with diabetes, and the presence of a biography could be useful in times of emergency or situations where the application could help them to recall important information. If a user has no biography saved, a message is shown in feedback display pane, encouraging them to create one.

A user is allowed to add all the fields using the format above, in any order of fields to add the biography. In adding a biography, it is **compulsory** for the user to include the following information (i.e. should not be blank):

- **NAME**
- **CONTACT NUMBER**
- **EMERGENCY CONTACT**
- **MEDICAL CONDITION**

Other information such as **NRIC** and **DATE OF BIRTH** are optional, as this depends on how willing the user is to provide information that is personal to them, and they can furthermore be added any time afterwards using the `editbio` command. However, we believe information such as contact number, emergency contact and medical condition should not be ignored, as they are some of the most necessary in times of emergency, and the user (or even his / her caretaker) should have knowledge about and access to them at all times.

An error message is shown if a user attempts to add a biography not containing any of these fields. A storage file containing biography information is also deemed to be corrupted if it does not contain any of these fields. An empty biography will be loaded instead. If a user, however, does not have any medical conditions using this application, he or she may input "NIL". However, this is highly not recommended, especially for users who do have medical conditions.

A user can have more than one of the following types of fields:

- Contact Number
- Emergency Contact
- Medical Condition
- Goals

This means that a user can add multiple emergency contacts, for instance, by having more than one prefix for the argument in this form: `e/91234567 e/98765432`.

Other fields can only have one data value associated with it. If multiple fields of the same type are entered for fields other than the ones listed above, an error message is displayed to the user, along with the prefixes that the user entered that can only be entered at most once.

For the following fields, certain restrictions are put in place:

- Name: Can contain only alphabets and spaces
- NRIC: Can contain only alphanumeric characters
- Gender: Can contain only alphanumeric characters
- DOB: Can only be in the format YYYY-MM-DD and represent a valid date (valid day depends on month and presence of leap years)
- Contact Number or Emergency Contact: Can contain only numbers of at least 3 digits in length
- DP Path: Has to be a valid path pointing to image eg. `dp//Users/bob/Desktop/doge.png`. This also works on windows (with double backslash instead of forward slash in the path). The image must be able to load (i.e. not corrupted or a non-image file)

Other fields can take any values. If the restrictions above are violated, an error message will be shown to the user and the command cannot proceed until the user re-enters the command with the corrected values.

Users will also *not* be allowed to enter duplicate values for fields that supports multiple values. (i.e. p/123 p/123) Doing so results in an error message being shown to the user, indicating that duplicate values are not allowed.

Upon successful processing of the command and its arguments, the user's biography will be updated accordingly and the biography pane will be displayed to the user, regardless of which pane was previously being displayed before (eg. a user may add to the biography while records are being displayed). In the user's feedback display pane, a message indicating success is shown, along with the fields that are added, sorted in accordance to a standardised order of fields displayed (same ordering that fields are being presented in this document), regardless of the order entered by the user.

As described above in the `bio` section, all values are refreshed whenever the biography pane is displayed, with the exception of the profile picture, that is only updated if there has been a change in the name of given path. By default, the path is an empty string and in the same way, the default picture is loaded only once until there is a change in the display picture, regardless of changes to other fields in between during a continuous session of the program.

Arguments for this command cannot be empty, and as such the user may not simply enter `addbio` or enter invalid arguments (doing so will result in an error message displayed, with message usage details). Any invalid arguments that occur after a prefix will be taken to be part of the prefix. eg. `n/exampleName p/1234567` is a valid input while `n/exampleName asdf/1234567` is invalid as `asdf/1234567` will simply be taken as part of the name, and of course this no longer fulfills the restrictions set for names. However, `a/exampleAddress asdf/1234567` is perfectly valid and it will be taken that `asdf/1234567` is intended to be part of the address, as there are no restrictions to the characters that address may take.

Although the command word is not case-sensitive, its arguments are so as to minimise clashing of

user's intentions and what the program understands eg. `a/exampleAddress M/test` is understood differently to the program than `a/exampleAddress m/test` is. Ultimately, the user has a balance of convenience and flexibility.

Examples of **VALID** `addbio` commands, provided that a biography does not yet exist, include:

- `addbio dp//Users/bob/Desktop/doge.png desc/hello world n/testName nric/testNric gender/testGender dob/1920-12-21 p/1234567 p/91234567 e/81234567 m/testMedicalCondition a/example address 123 goal/testGoal o/testOtherInfo` (Note: this is provided that the image exists at exactly the **SAME PATH** in the user's device. Otherwise, it has to be modified or removed in order for this example to work)
- `addbio n/testMinimal p/91234567 e/81234567 m/testMedicalCondition`

3.3.4. Edits the user's biography: `editbio`

Format: `editbio [n/NAME] [dp/DP PATH] [desc/PROFILE DESCRIPTION] [nric/NRIC] [g/GENDER] [dob/DATE OF BIRTH] [p/[INDEX/]CONTACT NUMBER]... [e/[INDEX/]EMERGENCY CONTACT]... [m/[INDEX/]MEDICAL CONDITION]... [a/ADDRESS] [goal/[INDEX/]GOAL]... [o/OTHER BIO INFO]`

A biography can be edited only if one already exists. An error message will be shown to a user who attempts to use the `editbio` command to edit fields before a biography is added. As the fields used by `addbio` and `editbio` commands are identical and can appear in any order, a user who attempts to add a biography that does not yet exist using the `editbio` command, after receiving the error message, can simply amend `editbio` to `addbio` before re-entering the line of command, without having to re-enter all the fields. (this is, of course, with the exception that the fields do not contain the compulsory fields that are required for the `addbio` command and not the `editbio` command).

Once a biography exists, the `editbio` command functions in a very similar way to the `addbio` command. The fields are not only identical but have the same restrictions described in the `addbio` section above.

The only key difference between the `editbio` command and the `addbio` command is that the `editbio` command does not require any compulsory fields that the `addbio` command does. Any combination of fields may be edited so long as there is more than one field and the inputs are valid.

Similar to the `addbio` command, an error message is shown if a user attempts to edit a biography but does not specify any fields. If multiple prefixes of the same type are entered for fields meant to contain only one data value (refer to `bio` section) eg. Name, an error message is displayed to the user, along with the prefixes that the user entered that can only be entered at most once.

Data of fields may be removed by simply entering blank input for the field eg. `editbio a/` resets the address to blank. This, however, cannot be done for compulsory fields. An error message will be shown if a user attempts to remove compulsory information using `editbio` as this will either violate the above restrictions set and / or the condition that the field should not be blank. The only way to remove compulsory fields is for a user to clear the biography (see `clrbio` section below) completely. Otherwise, so long as a biography exists, at least one value must be present for each compulsory field.

A key aspect of the `editbio` feature is that for fields that can take multiple values (i.e. phone numbers, medical conditions and goals, as described in the `addbio` section above), indices may be

specified to edit (a) particular value(s) of the field.

Take for instance the following phone numbers that are displayed in the user's biography table.

1. 91234567
2. 98765432
3. 81234567

If the user intends to edit the second of three phone numbers in a list, he or she may input `editbio p/2/1234567` to change the second number in the list of phone numbers. Similarly, if he or she wishes to edit the first and third number, `editbio p/1/1234567 p/2/1234567` would be a valid line of command. Similar to other arguments, this can be combined with other arguments for fields to be edited (whether or not they allow for multiple values).

It should be noted, however, that the indices provided should be one based (i.e. starts from one) and positive integers that are not out of bounds of the list. As such, still with reference to this example, `editbio p/4/1234567` is equally **INVALID** as `editbio p/0/1234567`, `editbio p/-1/1234567` and '`editbio p/string/1234567`' and an error message will be shown in each of these cases.

This way of entering commands is exclusive for fields that support multiple values. As such, this format will not be recognised for arguments of other fields such as name (i.e. `editbio n/1/testName` is invalid).

However, fields that support multiple values may use **EITHER** the indexing format *or* non-indexing format (i.e. format used by all other fields). When a sub-argument without indices such as `p/1234567 p/98765432`, the program automatically takes it that a replacement is to be made (i.e. values in the original list is replaced by the new values given). A combination of the two is however not allowed as it is ambiguous and will never likely be the intention of the user. As such `editbio p/1234567 p/1/2345678 p/23423423` will be **INVALID** and an error message will be given in the feedback indicating the inconsistency had such a command been entered. A combination of different *fields* is again possible, nevertheless, and different command formatting types may be used across different fields, so long as consistency is maintained in fields of the same type. (i.e. '`editbio p/1/1234567 p/2/2345678 m/Diabetes desc/this is a test description g/12345` is valid)

If *all fields* set by the user are no different from what already exists in the biography, the user is notified in the feedback display pane that the same information already exists in the biography and that there is nothing to be updated. Hence, if the user's biography contains `Bob` as the value of the `Name` field but not the address field, keying in `editbio n/Bob` will result in the notification being shown but not `editbio n/Bob a/Test Address`. The new value in the address field will be updated in the second case.

Users will also *not* be allowed to enter duplicate values in each list that supports multiple values. (i.e. `p/123 p/123`) Doing so results in an error message being shown to the user, indicating that duplicate values are not allowed.

Upon successful processing of the command and its arguments, the user's biography will be updated accordingly and the biography pane will be displayed to the user, regardless of which pane was previously being displayed before (eg. a user may add to the biography while records are being displayed). In the user's feedback display pane, a message indicating success is shown, along with

the fields that are edited, sorted in accordance to a standardised order of fields displayed (same ordering that fields are being presented in this document), regardless of the order entered by the user. Only fields that have been changed will be shown as updated in the feedback. Hence, using the same example above, entering `editbio n/Bob a/Test Address` for a biography that already has the name `Bob` will only result in the address shown as a field that was modified.

As described above in the `bio` section, all values are refreshed whenever the biography pane is displayed, with the exception of the profile picture, that is only updated if there has been a change in the name of given path. By default, the path is an empty string and in the same way, the default picture is loaded only once until there is a change in the display picture, regardless of changes to other fields in between during a continuous session of the program.

Arguments for this command cannot be empty, and as such the user may not simply enter `editbio` or enter invalid arguments (doing so will result in an error message displayed, with message usage details). As described in the `addbio` section, any invalid arguments that occur after a prefix will be taken to be part of the prefix. eg. `n/exampleName p/1234567` is a valid input while `n/exampleName asdf/1234567` is invalid as `asdf/1234567` will simply be taken as part of the name, and of course this no longer fulfills the restrictions set for names. However, `a/exampleAddress asdf/1234567` is perfectly valid and it will be taken that `asdf/1234567` is intended to be part of the address, as there are no restrictions to the characters that address may take.

Although the command word is not case-sensitive, its arguments are so as to minimise clashing of user's intentions and what the program understands eg. `a/exampleAddress M/test` is understood differently to the program than `a/exampleAddress m/test` is. Ultimately, the user has a balance of convenience and flexibility.

Examples of **VALID** `editbio` commands, provided that a biography exists, include:

- `editbio desc/hello world n/testName nric/testNric gender/testGender dob/1920-10-08 p/91234567 e/81234567 m/testMedicalCondition a/example address 123 goal/testGoal o/testOtherInfo`
- `editbio dob/2019-12-28`
- `editbio p/1234567 p/23456789`
- `editbio p/1/234567 p/2/3456789`
- `editbio m/medicalCondition1 m/medicalCondition2 m/medicalCondition3 m/medicalCondition4`
- `editbio n/John Doe`

3.3.5. Clears the user's biography: `clrbio`

Format: `clrbio`

A user may clear his or her biography using the `clrbio` command. If a biography exists, all data from all biography fields will be removed. Otherwise, if a biography does not exist, the user will be displayed a message that the biography is already empty and there is no biography information to clear. If a biography is successfully cleared, the biography display pane with an empty biography table is shown to the user. A user cannot execute `editbio` after clearing the biography until another biography has been added using `addbio`.

Note that this command does not affect the `background` and `fontcolour` aspects of the program and a 'cleared' biography table still shows aesthetic preferences.

Similar to the `bio` command, this command cannot have any sub-arguments, and thus, if the user enters `bio 1`, an error message will be displayed, as the user's intention may not have been necessarily to show the biography. Trailing spaces are automatically trimmed and as such pose no issue.

The command word, as for all all other commands, is not case-sensitive for convenience to user.

3.3.6. Sets the font color of the application: `fontcolour` or `fontcolor`

Formats:

- `fontcolour COLOUR [bg/BACKGROUND ARGUMENTS]` or `fontcolor COLOUR [bg/BACKGROUND ARGUMENTS]`; or
- `fontcolour` or `fontcolor`

To accomplish higher levels of personalisation, the user may select not only from a range of colours or standard colours, but *any* colour. This means that a user can set a colour using **EITHER** colour names or hexadecimal values.

To set a colour of a font using a colour name, simply enter `fontcolour` (or the American spelling `fontcolor`; both are recognised by the program) followed by the intended name of the colour. For instance, one may enter: `fontcolour yellow` or `fontcolor skyblue`. So long as the colour names are within the 140+ colour names recognised by CSS, the colours will be set accordingly on the user's application. If the colour is not recognised, the program attempts to interpret it as a hexadecimal value colour. Note that `transparent` is NOT a valid colour.

A colour may be set using its hexadecimal value provided it follows format beginning with a '#' followed by six valid alphanumeric characters representing a hexadecimal colour. For instance, one may enter: `fontcolour #FFFF00` or `fontcolor #FFFF3A`.

If a coloured entered belongs to neither categories of names nor hexadecimal values, an error message will be shown to the user in the feedback display panel.

If the colour set by the user is no different from the existing colour, be it in colour name or hexadecimal representation, the user is notified in the feedback display pane that the same colour is already being set in the settings and that there is nothing to be updated.

Otherwise, upon successful execution of the command, the colour would be applied universally and instantaneously, from the command text to the headers of labels. This is with the exception of text in the graph shown for the `average` command, that uses predefined colours. The change will be described in the feedback display panel and the new colour is reflected in the biography table of the biography display pane. This command does not change the panes displayed to the user, but if the user happens to be viewing the biography display pane, instantaneous change is observed in the biography table under the 'Font Colour' field. If colours described in the biography table and feedback have a recognised colour name, the name of the colour is automatically displayed and saved as such, regardless of whether they have been entered as a hexadecimal value. i.e. `#FFFF00` will always be displayed as `yellow`. Otherwise, the hexadecimal value of the colour is displayed and saved.

The last set font colour of the user is always saved, and upon restarting the application, the program should display the window with the last saved `fontcolour` (or `fontcolor`) settings.

If the preferences file happen to be corrupted with unrecognised colours, or cannot be found, the `background` and `fontcolour` (or `fontcolor`) are reset to its default aesthetics settings, which a new preferences file also contains.

By default, the `fontcolour` (or `fontcolor`) and `background` of the application are set as colours `#FFFF3A` and `#000A34` respectively.

Note that colours that are deemed to be too close to the dominant colour of the `background` will not be allowed to be set as the font colour as the text may get too difficult to read on screen. (eg. yellow font colour with white background) In such cases the user will be prompted to change the colour or image of the background first before proceeding with the change in font colour.

Alternatively, the user is allowed to change the font colour and background *simultaneously* by combining the commands for font colour and background, using the `bg/` prefix. eg. `fontcolour yellow bg/black`. This allows the user to not only save time but also make contrasting changes in colours which would otherwise have been difficult to achieve. For instance, a user intending to change the font colour from white to black with a current dark background will benefit from this feature as sequentially changing switching to a light background or dark font colour would be impossible. At most one of such prefixes may be used; using more than one results in an error message shown to the user.

This program does not implement a command to clear a font colour due to its redundancy - a user, if dissatisfied with the font colour may simply change the colour to his or her preference, or use generic colours by keying in `fontcolour black` (or `fontcolor black`) along with a white background.

A user is also allowed to key in `fontcolour` (or `fontcolor`) on its own and the program having received this will display the current font colour settings in the feedback display pane. This allows the user to view the current settings of the font colour without having to use the `bio` command to navigate to the biography display pane.

The command word, as for all all other commands, is not case-sensitive for convenience to user. The colour entered for both colour names and hexadecimal values are also not case sensitive, as the possibilities of misinterpretation are much lower as compared to sub-arguments of commands such as `editbio`, or `add`. As such, convenience is prioritised for the user for this command and both `fontColour yElLoW` and `fONtColOur #fFFFf00` will work.

Examples of **VALID** `fontcolour` (or `fontcolor`) commands:

- `fontcolour yellow`
- `fontcolor indigo`
- `fontcolour #202020`
- `fontcolor #000000`
- `fontcolour`
- `fontcolor`

- `fontCOLOUR wHITE`
- `fontCOLOUR #FFffff`
- `fontcolour yellow bg/black`
- `fontcolor yellow bg//Users/bob/Desktop/SpaceModified.jpg s/cover` (Note: this is provided that the image exists at exactly the **SAME PATH** in the user's device. Otherwise, it has to be modified or removed in order for this example to work)

3.3.7. Sets the background image or colour of the application: `bg`

Formats:

- `bg COLOUR [fontcolour/COLOUR]` or `bg COLOUR [fontcolor/COLOUR]`; or
- `bg PATH [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolour/COLOUR]` or `bg PATH [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolor/COLOUR]`; or
- `bg [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolour/COLOUR]` or `bg [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolor/COLOUR]` (only if background is already a background image); or
- `bg`

Users are allowed to set the background either using a `COLOUR` or a `PATH` to a background image.

The `COLOUR` argument of the background works in exactly the same way as described in the `fontcolour` or (`fontcolor`) sub-section above, except that command word used is now `bg` instead of `fontcolour` (or `fontcolor`). i.e. a user may enter `bg blue` or `bg #202020` to set the background image.

Just as for the `fontcolour` (or `fontcolor`) command, upon successful execution of the `bg` command with `COLOUR`, the colour would be applied universally and instantaneously, from the backgrounds of scrollpanes to the backgrounds of feedback display panes. This is with the exception of the background in the graph shown for the `average` command, that uses predefined colours. The change will be described in the feedback display panel and the new colour is reflected in the biography table of the biography display pane. This command does not change the panes displayed to the user, but if the user happens to be viewing the biography display pane, instantaneous change is observed in the biography table under the 'Background' field. If colours described in the biography table and feedback have a recognised colour name, the name of the colour is automatically displayed and saved as such, regardless of whether they have been entered as a hexadecimal value. i.e. `#FFFF00` will always be displayed as `yellow`. Otherwise, the hexadecimal value of the colour is displayed and saved.

The last set font colour of the user is always saved, and upon restarting the application, the program should display the window with the last saved `fontcolour` (or `fontcolor`) settings.

If the preferences file happen to be corrupted with unrecognised colours, or cannot be found, the `background` and `fontcolour` (or `fontcolor`) are reset to its default aesthetics settings, which a new preferences file also contains.

By default, the `fontcolour` (or `fontcolor`) and `background` of the application are set as colours `#FFFF3A` and `#000A34` respectively.

In addition to specifying a **COLOUR**, a user may also specify a **PATH** for background image. This works similarly to the **dp** argument of the **addbio** or **editbio** commands. The program first interprets the given argument as a **COLOUR**, and if it fails at doing so, attempts to interpret it as a **PATH** for an image. If the image given has an invalid path or cannot be loaded as an image, an error message is shown to the user. Thus, file paths have to be valid paths in order for the command to proceed.

Optional prefixes **s/** and **r/** also allow the user to change the size and repeat attributes of the background image respectively. Currently, for both prefixes, the sub-arguments allow only for predefined values known to CSS.

i.e.

For repeat:

- **repeat-x**
- **repeat-y**
- **repeat**
- **space**
- **round**
- **no-repeat**

Specific details for each of these back-ground repeat values are explained here:
<https://www.w3.org/TR/css-backgrounds-3/#the-background-repeat>

For size:

- **auto**
- **cover**
- **contain**

Specific details for each of these back-ground repeat values are explained here:
<https://www.w3.org/TR/css-backgrounds-3/#the-background-size>

If not set by the user, **auto** and **repeat** will be set for size and repeat fields respectively.

Upon successful execution of the **bg** command with **PATH**, the background image would be applied universally and instantaneously. A single background will be set for the window, regardless of which display pane the user is viewing. This is with the exception of the background in the graph shown for the **average** command, that uses predefined colours. The change will be described in the feedback display panel and the new colour is reflected in the biography table of the biography display pane. This command does not change the panes displayed to the user, but if the user happens to be viewing the biography display pane, instantaneous change is observed in the biography table under the 'Background' field, that shows the path of the background image, along with values in the **Background Size** and **Background Repeat** fields. Values for **Background Size** and **Background Repeat** will be **auto** and **repeat** if not set by the user.

After setting the background image, the user may change these by simply entering **bg s/cover** or **bg repeat/no-repeat s/contain** to apply the newly-defined attributes to the background image. Note

that this works only if the current background displayed is a background image and not a colour. An error message will be shown to the user if the user attempts to enter these commands while the background image is a colour.

A user may also not specify any background size or background repeat while setting a colour for a background command. i.e. `bg yellow s/auto` is **INVALID**. An error message will be shown to the user if the user attempts to enter commands such as this, clearly indicating that additional arguments are allowed only for background images. Hence, in the biography display pane, there will never also be a situation where a colour is indicated in the 'Background' field along with data in 'Background Repeat' and 'Background Size' fields. If a background colour is used, these fields are simply blank. A user may only set the background to **EITHER** a background or an image; setting both at the same time would not be possible. At any point of time, generic or not, the user will have exactly one `background` and `fontcolour` (or `fontcolor`) field set for the application.

A user will not be allowed to set the background image in circumstances whereby the background image is not close to the "transparent colour", and the background has attributes such that the background size is not `cover` and the background repeat is not `repeat`. This is because in such circumstances the text could get difficult to see once clipping of the image occurs and the image is surrounded by the light transparent background colour. This of course does not apply to backgrounds that are colours as colours will naturally cover the entire screen even as the background size and background repeat, which are irrelevant for colours, are naturally empty `String` values.

For each successful command, feedback will be displayed to the user on the change in background, be it from a background image to a colour, vice versa or other combinations. If a command such as `bg s/cover` changes only a particular attribute of the background, only the changes made will be reported. As for colours, automatic conversion is done to convert hexadecimal values to colour names where possible.

Note that colours that are deemed to be too close to the dominant colour of the `fontcolour` will not be allowed to be set as the background as the text may get too difficult to read on screen. (eg. yellow font colour with white background) In such cases the user will be prompted to change the `fontcolour` first before proceeding with the change in background.

Alternatively, the user is allowed to change the font colour and background *simultaneously* by combining the commands for font colour and background, using either the `fontcolour/` or `fontcolor` prefix. eg. `bg black fontcolour/yellow`. This allows the user to not only save time but also make contrasting changes in colours which would otherwise have been difficult to achieve. For instance, a user intending to change the font colour from white to black with a current dark background will benefit from this feature as sequentially changing switching to a light background or dark font colour would be impossible. At most one of such prefixes may be used; using more than one results in an error message shown to the user.

This program does not implement a command to clear a font colour or background due to its redundancy - a user, if dissatisfied with the background colour may simply change the colour to his or her preference, or use generic colours by keying in `background white` along with a black background. As user's images are not saved in the application itself and instead loaded on startup of the program, the user does not need to worry about deleting background images (or even profile images for `addbio` or `editbio` commands). Replacing the path of the image with a colour or another

image path will do the job.

If the preferences file happen to be corrupted / unloadable background paths, or cannot be found, the `background` and `fontcolour` (or `fontcolor`) are reset to its default aesthetics settings as described above, which a new preferences file also contains. As for paths to profile pictures, an unloadable background file (or colours) from the storage does not cause the user to lose any other data, and the user is simply notified via the feedback display pane that the image cannot load and has been removed.

A user is also allowed to key in `bg` on its own and the program having received this will display the current background settings in the feedback display pane. This allows the user to view the current settings of the background without having to use the `bio` command to navigate to the biography display pane.

The command word, as for all all other commands, is not case-sensitive for convenience to user. The colour entered for both colour names and hexadecimal values are also not case sensitive, as the possibilities of misinterpretation are much lower as compared to sub-arguments of commands such as `editbio`, or `add`. As such, convenience is prioritised for the user for this command and both `Bg yElLoW` and `bG #fFFFf00` will work. However, path names are still case sensitive and should be entered exactly as it is. i.e. A file stored at the file path `/Users/bob/Desktop/doge.png` will not be able to load if the user enters `/Users/bob/Desktop/Doge.png`. As for profile picture, nevertheless, file paths work regardless of operating systems (i.e. Windows / Mac / Linux) so long the paths are keyed in exactly as they should be in a way that the device recognises (eg. double backslash for Windows instead of the forward slash).

Font colours and backgrounds can be changed independently and sequentially. As such, high customisation is possible, with the user having the freedom to choose any combination of font colours and background colours / images desired.

Examples of **VALID** `bg` commands:

- `bg yellow`
- `bg indigo`
- `bg #202020`
- `bg #000000`
- `bg`
- `Bg wHITE`
- `bG #FFffFf`
- `bg /Users/bob/Desktop/doge.png s/auto`
- `bg /Users/bob/Desktop/SpaceModified.jpg`
- `bg s/cover` (if background is already a background image)
- `bg r/no-repeat s/cover` (if background is already a background image)
- `bg black fontcolor/#FFFF00`
- `bg /Users/bob/Desktop/SpaceModified.jpg s/cover fontcolour/yellow`

(Note: For each of the examples with paths above, it is provided that the image exists at exactly the **SAME PATH** in the user's device. Otherwise, it has to be modified or removed in order for this example to work)

3.3.8. Displays a motivational quote: [no action required]

The motivation aspect of the personalised user experience feature does not require any command. Rather, it is implicitly executed without any command as it is implemented to display at the bottom of the user's main window.

On startup, a motivational quote is randomly selected out of the 600+ quotes currently stored in the program itself. A mixture of encouraging quotes from different sources is used, revolving around topics of food intake, exercise to specifically diabetes itself. All quotes are formatted to be of the same format, with the quote followed by the speaker of the quote (if unknown, indicated as Anonymous).

The user currently does not have the capability to modify or view the full list of quotes other than the quote displayed on screen, and there is no intention for him or her to do so, as we believe having the full list defeats the purpose of the quotes to motivate one step at a time, as well as the element of surprise every time the application is opened. If a user finds that he or she is unable to relate to the quote, or does not like the quote that is displayed, he or she can simply restart the application and another quote will be displayed.

The simple design of this sub-feature minimises the need for user interaction and commands (eg. a command to switch quotes is unnecessary as a restart of the application already achieves that). Yet, this feature is one that could speak out to the user and encourage him or her through his or her day. Each user will receive a different sequence of quotes throughout his or her use of the application, and have different personal responses to them. As such, the motivational quotes personalise the experience of each user by making it truly unique for them. i.e. the quotes received by one user is guaranteed to differ by random chance, and even for the same quotes, they are likely speak differently to one user compared to another.

3.3.9. Saves the user's preferred themes: [coming in v2.0]

Allows users to save current fontcolours and background colours as themes that they can name and retrieve after.

- Upon entering commands, users can list, add, edit, delete or apply current themes to set colours and backgrounds to the ones that they have previously saved.

3.3.10. Shows a cartoon avatar that represents the user: [coming in v2.0]

Displays a cartoon that represents the user by observing the user's data such as BMI.

- As the user's records such as BMI changes, the users' avatar automatically changes accordingly (eg. size, width, height of avatar)

3.3.11. Follows up on the user's goals: [coming in v2.0]

Allows users to save goals in a certain format such the program will be able to follow-up on the user's goals. eg. Lose 10kg by 2019-12-28 (system provides updates throughout and determines the user's progress as well as how well they have worked towards their goal(s)). Users may also set reminders to remind them of their goal and receive timely feedback.

3.4. DATA SUMMARY/ANALYSIS

3.4.1. Displays the daily/weekly/monthly average of records in a line graph: average

Format: `average a/AVERAGE_TYPE rt/RECORD_TYPE [n/COUNT]`

AVERAGE_TYPE is either "daily", "weekly" or "monthly".

RECORD_TYPE is either "bloodsugar" or "bmi".

Displays a graph of the "daily", "weekly" or "monthly" average of a particular RECORD_TYPE.

COUNT is an optional field that takes integer between 1 to 12 inclusive.

If COUNT is given, SugarMummy shows **up to** COUNT most recent number of average values. Else, COUNT is set to 5 by default and shows 5 most recent average values.

NOTE

If SugarMummy **does not show exactly** COUNT number of average values, that means you do not have enough records in the database.

Example usage 1: `average a/weekly rt/bloodsugar`: Shows the 5 latest weekly average of blood sugar records.

Example usage 2: `average a/daily rt/bmi n/9`: Shows the 9 latest daily average of BMI records.

3.4.2. Summary statistics of a particular record type [coming in v2.0]

Shows minimum, maximum, average of a record type. Categorizes records into low, normal and high values in a pie chart. User can specify the date interval of the summary by giving a start date and end date.

3.4.3. Shows relationship between record types: [coming in v2.0]

User can see how a particular record type changes with other record types. This information will be displayed on a scatter plot where trends can be spotted easily. Note that this feature will work well only if

1. You have records for both record types in any given day.
2. You have more than 50 days worth of records.

3.4.4. Exports summary of all medical records into pdf [coming in v2.0]

Need to share your records with your doctor or others? With SugarMummy export function, you can save any graphs and plots generated in a pdf file.

3.5. Food Recommendation

General Note

1. If a command requires no user-input arguments, all the additional inputs after this command string will be ignored and the command will be executed as usual.
2. If a command requires any parameters, the input order is flexible. Duplicate parameters are also allowed, **but only the last occurrence will be considered**.

3.5.1. Recommending food: `recmf`

Recommends medically suggested foods for type II diabetes patients. The user can specify flags and food names` as two kinds of filters, as well as one type of sorting order:

- **Flags:** specifies the wanted food types in the form of following flags:

-nsv: non-starchy vegetable, such as <i>Broccoli</i>	-sv: starchy vegetable, such as <i>Potato</i>
-f: fruit, such as <i>Cherry</i>	-p: protein, such as <i>Lean Lamb</i>
-s: snack, such <i>Fig Roll</i>	-m: meal, such as <i>Spanish Omelet</i>

Note:

1. Flags are case-insensitive and duplicates are allowed, but they will be recognized only when placed before any prefix.
2. If no flag is specified, it is equivalent to specifying all flags. Namely, foods of all types will be shown.

- **Food Names:** matches foods that contain **one of** given food names in the form of `fn/[FOOD_NAME]…`

Note:

1. Matching is case-insensitive but is full-word matching. For example, "chicken" does not match "ch".
2. If no food name specified after `fn/`, it is equivalent to matching all foods.

- **Sorting Order:** determines the presentation order of food cards in **one of** the two forms: `+sort/SORT_ORDER_TYPE` and `-sort/SORT_ORDER_TYPE`

- + indicates in ascending order and - descending
- `SORT_ORDER_TYPE` is required and can be one of the following six:
`fn:` food name; `ft:` food type; `ca:` calorie; `gi:` glycemic index; `su:` sugar; `fa:` fat

NOTE:

1. `+sort/SOT` and `-sort/SOT` cannot be both present even though they may have different `SOT(SORT_ORDER_TYPE)`.
2. Specially, for `ft`, the ascending order is predefined as: nsv, sv, f, p, s, m.

Format: `recmf [-nsv] [-sv] [-f] [-p] [-s] [-m] [fn/FOOD_NAME…][±sort/SORT_ORDER_TYPE]`

Examples: `recmf -p -f`, `recmf fn/chicken rice`, `recmf -p -m -f fn/chicken`, `recmf -p -nsv +sort/gi`

3.5.2. Recommending food combination: `recmfmix`

Recommends one food from each type. A summary card of all nutrition values will be appended at the end.

Note:

1. All extra inputs after this command string will be ignored.
2. Food types with no corresponding food data will not be shown. If there is no food data at all, the summary card will not be shown as well.
3. If any of the summary data is decimals, it will be formatted into two decimal places.
4. Please note the GI (glycemic index) value is the average instead of sum. For more information about GI, please refer to this [link](#).

Format: `recmfmix`

Example: `recmfmix`

3.5.3. Adding new food items : `addfood`

Adds a new food item for future recommendations. The following six fields are required:

- food name: `fn/FOOD_NAME`
Food name should only contain alphabets, numbers, and whitespace. It should be less than 30 characters for display quality and readability.
- food type: `ft/FOOD_TYPE`
Food types should be exactly one of the following: nsv(non-starchy vegetable), sv(starchy vegetable), f(fruit), p(protein), s(snack), or m(meal).
- calorie (cal): `ca/CALORIE`
Calorie should be less than 700(cal) per serving.
- gi: `gi/GI`
Glycemic Index should be less than 70 per serving.
- sugar (g): `su/SUGAR`
Sugar should be less than 25(g) per serving.
- fat (g): `fa/FAT`
Fat should be less than 35(g) per serving.

Note:

1. No duplicate food names are allowed.
2. All nutrition values should be non-negative numbers and contain no more than four decimals. Incomplete decimals, such as ".5" and "1.", can be accepted as "0.5" and "1" respectively. However, decimal point by itself alone will not be accepted.
3. Ideally, the input values are normalized as per serving for more practical value comparisons and calculations.

Format: `addfood fn/FOOD_NAME ft/FOOD_TYPE ca/CALORIE gi/GI su/SUGAR fa/FAT`

Example: `addfood fn/Cucumber ft/nsv ca/15 gi/15 su/1.7 fa/0`

3.5.4. Deleting an existing food: `deletef`

Deletes a food that matches the specified food name(s).

Note: FOOD_NAME matching is case-insensitive, but is strict matching for every single character, including white spaces between words. It is also full matching. For example, "Rice with Chicken" does not match with "Chicken".

Format: `deletef fn/FOOD_NAME`

Example: `deletef fn/Mushroom`

3.5.5. Resetting food data: `resetf`

Clears all modifications, adding and deleting, on the food list. The food data will be reset to sample food data.

Format: `resetf`

Example: `resetf`

3.5.6. Recovering food data: `recoverf coming in v2.0`

Recovers the food data after resetting all the foods. The recovered data is based on the food list state just before the **latest recmf command**. This would be useful if the user wrongly enter `resetf` command, or another user want to temporarily use the same jar file on the same PC.

3.5.7. Editing a food: `editf coming in v2.0`

Edits the available food fields, such as food name and GI value, of an existing food. The restrictions on field value are the same as the restrictions declared in `addfood` commands. This would provide more flexibility to the user to manipulate food data, instead of directly deleting a food.

3.5.8. Recording and analyzing diets `coming in v2.0`

Records the user's diets on specified dates and provides daily, weekly, and monthly summaries about nutrition intakes. This would allow the user to have an overview of his food consumption statistics. Bases on such statistics, the user can get more specific and personalized suggestions to balance the his nutrition intake.

3.6. Other Common Features

3.6.1. Exiting the program : `exit`

Exits the program.

Format: `exit`

3.6.2. Viewing full User Guide

Shows the URL to this User Guide.

Format: `help`

4. Features Coming in v2.0

4.1. Autocomplete Command

SugarMummy will recommend list of possible commands that can be add on to user current input.

4.2. Clearing all entries: `clearall`

Clears all recorded data in SugarMummy.

4.3. Editing specific record entries in the list: `edit INDEX [h/VALUE] [w/VALUE] [con/VALUE]`

Allows the user to edit specific fields within a previously created entry.

4.4. Encrypting data files: `enc [KEY]`

The user can optionally provide a key to encrypt all the data. Afterwards, all the stored user input can only be viewed with the user-defined key.

4.5. Detailing workout schedule

User can enter details of a workout schedule.

For example: Do exercise A for 10 minutes then switch to exercise B for 5 minutes.

4.6. Timing Workout

Displays workout instructions and a timer on screen. Screen instruction changes when the time for that particular exercise is up. Time required for each exercise is dictated by stored workout schedule.

4.7. Sharing with community

The user is able to interact with others who is using SugarMummy. A community platform will be provided for users to share their daily activities and health tips. This community is expected to help the user obtain encouragements and comforts.

5. Command Summary

1. **Achievement:** `achvm`
2. **Add:** `add rt/RECORD_TYPE dt/DATETIME` and parameters a record requires
E.g. `add rt/BLOODSUGAR dt/2019-12-12 12:12 con/10`
E.g. `add rt/BMI dt/2019-12-12 12:12 h/1 w/1`
3. **Add Biography :** `addbio n/NAME [dp/DP PATH] [desc/PROFILE DESCRIPTION] [nric/NRIC] [g/GENDER] [dob/DATE OF BIRTH] p/CONTACT NUMBER... e/EMERGENCY CONTACT... m/MEDICAL CONDITION... [a/ADDRESS] [goal/GOAL]... [o/OTHER BIO INFO]`
E.g. `addbio n/Bob p/98765432 e/91234567 m/Type II Diabetes`
4. **Add Food:** `addfood fn/FOOD_NAME ft/FOOD_TYPE ca/CALORIE gi/GI su/SUGAR fa/FAT`
E.g. `addfood fn/Cucumber ft/nsv ca/15 gi/15 su/1.7 fa/0`
5. **Average:** `average a/AVERAGE_TYPE rt/Record_TYPE [n/COUNT]`
E.g. `average a/weekly rt/bloodsugar n/3`
6. **Bio:** `bio`
7. **Bg:**
 - `bg COLOUR [fontcolour/COLOUR]` or `bg COLOUR [fontcolor/COLOUR]`; or
 - `bg PATH [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolour/COLOUR]` or
`bg PATH [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolor/COLOUR]`; or
 - `bg [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolour/COLOUR]`
or `bg [s/BACKGROUND SIZE] [r/BACKGROUND REPEAT] [fontcolor/COLOUR]` (only if background is already a background image); or
 - `bg`
E.g. `bg /Users/Bob/Pictures/bg.jpg s/cover`
E.g. `bg yellow`
E.g. `bg #FFFF00 fontcolor/#000000`
E.g. `bg black fontcolour/yellow`
8. **Calendar** `calendar [ym/YEAR_MONTH] [ymw/YEAR_MONTH_DAY] [ymd/YEAR_MONTH_DAY]`
E.g. `calendar ymd/2019-12-24`
E.g. `calendar ymw/`
E.g. `calendar ym/2019-11`
9. **Clear Biography :** `clrbio`
10. **Dailymsg:** `dailymsg`
11. **Delete:** `delete INDEX`
E.g. `delete 2`

12. **Delete Food:** `deletef fn/FOOD_NAME`
E.g. `deletef fn/Mushroom`
13. **Event:** `event d/DESCRIPTION dt/DATETIME [dt/DATETIME] [td/TIME_DURATION]`
E.g. `event d/appointment dt/2019-12-14 16:00 td/01:00`
E.g. `event d/appointment dt/2019-12-20 11:00 dt/2019-12-20 11:30`
14. **Edit Biography:** `editbio [n/NAME] [dp/DP PATH] [desc/PROFILE DESCRIPTION] [nruc/NRIC] [g/GENDER] [dob/DATE_OF_BIRTH] [p/[INDEX/]CONTACT_NUMBER]... [e/[INDEX/]EMERGENCY_CONTACT]... [m/[INDEX/]MEDICAL_CONDITION]... [a/ADDRESS] [goal/[INDEX/]GOAL]... [o/OTHER_BIO_INFO]`
E.g. `editbio gender/Male dp//Users/Bob/Pictures/dp.jpg p/12345678`
E.g. `editbio dp/C:\\\\Users\\\\Bob\\\\Pictures\\\\dp.jpg`
E.g. `editbio p/1/234567 p/2/3456789`
15. **Exit the Application:** `exit`
16. **Font Color:**
 - `fontcolour COLOUR [bg/BACKGROUND ARGUMENTS]` or `fontcolor COLOUR [bg/BACKGROUND ARGUMENTS]`; or
 - `fontcolour` or `fontcolor`
E.g. `fontcolour yellow`
E.g. `fontcolor #FFFF00`
E.g. `fontcolour #FFFFFF bg/black`
17. **Get User Guide URL:** `help`
18. **List:** `list`
E.g. `list`
19. **Recommend Food:** `recmf [-nsv] [-sv] [-f] [-p] [-s] [-m] [fn/FOOD_NAME FOOD_NAME...] [+sort/SORT_ORDER_TYPE]`
E.g. `recmf -p -nsv +sort/gi`
E.g. `recmf fn/chicken rice`
20. **Recommend Mixed Food:** `recmfmix`
21. **Reminder:** `reminder d/DESCRIPTION dt/DATETIME [r/REPETITION]`
E.g. `reminder d/insulin injection dt/2019-11-30 17:30 r/daily`
E.g. `reminder d/buy bread dt/2019-12-02 19:00 r/weekly`
22. **Reset Food Data:** `resetf`

6. FAQ

1. How should I save my data?

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

2. How should I reset all the data?

You can simply delete the data folder within the same directory that you place SugarMummy.jar. All the data will be set to default sample data.