MooMooMoney - User Guide

By: `AY1920S1-CS2113T-F14-1` Since: `Oct 2019`

1. Introduction

MooMooMoney (MMM) is a program for managing monetary expenditure, made for those who prefer to use a desktop application. More importantly, MMM is optimized for those who prefer to work with a Command Line Interface (CLI). If you can type fast, MMM can help you with your money management faster than traditional GUI apps. Interested? Jump to the Section 2, “Quick Start” to get started. Enjoy!

2. Quick Start

1. Ensure you have Java `11` or above installed in your Computer.
2. Download the latest `MooMooMoney-v1.4.jar` here. (Link will be updated)
3. Copy the file to the folder you want to use as the home folder for MooMooMoney.
4. The application can be run using the provided terminal (Linux/Mac) or Command Prompt/Powershell (Windows) using “java -jar FILENAME.jar”

Figure 1: The current interface of our application

5. Some example commands you can try:
    - `category add shopping`: Creates the category "`shopping`".
    - `budget set c/shopping b/200`: Sets the budget for "`shopping`" to "`$200`"
    - `add c/shopping d/Nike Shoes a/50`: Add the entry `"Nike Shoes"` of the price `"$50"` to the `"shopping"` category.
    - `graph total`: Shows a bar graph of all current categories.
    - `bye` : Exits the app.
6. Refer to Section 3, "Features" for details of each command.

3. Features

Command Format

- The first word in the command is `lower_case` indicating the type of action to be performed, e.g in `view m/10 y/2019` the program understands view as displaying the summary main display for the month.
- Words in `UPPER_CASE` are the parameters to be supplied by the user e.g. in `add c/CATEGORY_NAME`, the user will input the `CATEGORY_NAME` is a name added category.
- These words are preceded by an `identifier/`, which allows the program to understand the user's input, e.g in `delete c/CATEGORY`, the identifier `c/` lets the program know that the `UPPER_CASE` parameter that follows is a category type. Other examples include:

  - `d/` (date)
  - `n/` (name)
  - `b/` (budget)

- The order of the commands does not matter, e.g if the command specifies `n/DESCRIPTION a/AMOUNT`, the same result is given for `a/AMOUNT n/DESCRIPTION` as well.
- Words in `(optional: field)` are optional fields that users can choose to fill.

3.1. Viewing help : `help`

List all valid commands.

Format: `help`

- Shows a list of all commands a user can enter as well as the format for using it.

3.2. Managing categories: `category`

Allows users to see what categories they have, add categories, and delete categories.

Show a list of all current categories: `list`

Format: `list`

- Each category will be displayed with its total spending for the current month giving the user a quick view of the categories he currently has and their index numbers

Examples:

- `list`

Add a new category in which users can include their expenditures: `add`

Format: `category add CATEGORY_NAME`

- Add a new category with the name `CATEGORY_NAME`
- Duplicates are not allowed
- Category names should not contain the symbol “/”
- Category names are not case-sensitive e.g. `food` is the same as `FOOD`

Examples:

- `category add Food`
- `category add Clothes and accessories`

Delete a category: `delete`

Format: `category delete CATEGORY`

- All expenditure under deleted `CATEGORY` will be deleted as well
- The field `CATEGORY` can either be the name of the category or its index number

Examples:

- `category delete 3`
- `category delete Shopping`

3.3. Managing expenditures : `expenditure`

Allows users to add an expenditure, delete an expenditure, and sort the expenditures within each category.

Add an entry for expenditure - what the user has spent money on: `add`

Format: `add n/NAME a/AMOUNT c/CATEGORY (optional: d/DATE)`

- A specified `CATEGORY` has to exist, if not the user will be prompted to add it first.
- Multiple entries of the same expenditure is allowed.
- The order of the parameters does not matter, e.g if the command specifies `add n/DESCRIPTION a/AMOUNT c/CATEGORY`, the same result is given for `add a/AMOUNT n/DESCRIPTION c/CATEGORY` as well.
- The parameter `AMOUNT` must be a number.
- The parameter `DATE` is an optional parameter and must follow the format DD/MM/YYYY. If it is not given, the expenditure will be assigned the current date.

Examples:

- `add n/Laksa a/5.50 c/Food`
- `add c/Shopping n/NikeShoes a/50 d/11/11/2019`

Delete an expenditure: `delete`

Format: `delete i/INDEX c/CATEGORY`

- A specified `CATEGORY` has to exist.
- The order of the parameters does not matter, e.g if the command specifies `delete i/INDEX c/CATEGORY`, the same result is given for `delete c/CATEGORY i/INDEX` as well.
- The parameter `INDEX` must be an integer corresponding to an existing expenditure.

Examples:

- `delete i/2 c/Food`

Sort how expenditures are sorted within each category: `sort`

Format: `sort TYPE`

- The parameter TYPE can be one of three inputs:
- `name`: sorts in alphabetical order
- `cost`: sorts in descending order according to their cost
- `date`: sorts according to their date

Examples:

- `sort name`
- `sort date`

3.4. Manage budget: `budget`

Set budget amount for category.

Format: `budget set c/CATEGORY b/BUDGET`

- `CATEGORY` must be an existing category.
- The budget must be a numerical number with no other symbols and at most 2 decimal places.
- Budget should be placed after a category.
- Multiple categories can be added by specifying more pairs of c/ and b/.
- *CATEGORY* is case insensitive.

- *CATEGORY* should not have a budget set.

Examples:

- `budget set c/Food b/1000 c/Shoes b/500`

  Sets budget amount to be `1000` for the `food` category and budget amount to be `500` for the `shoes` category. Budget amount and category will be displayed.

- `budget set c/Food b/800`

  If budget has already been set, an error will be displayed.

Figure 2. Setting a budget for a nonexistent category

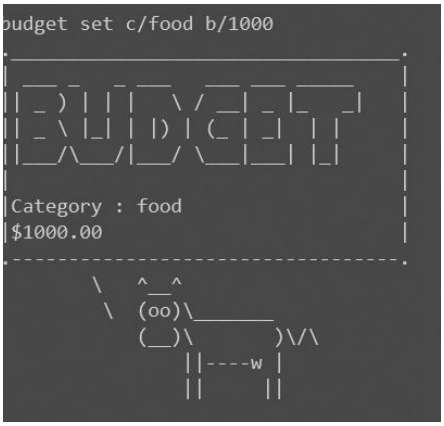

Figure 3. Setting a budget for a valid category

Edit budget amount for a category.

Format: `budget edit c/CATEGORY b/BUDGET`

- `CATEGORY` must be an existing category.

  The budget must be a numerical number with no other symbols.

- `BUDGET` should be placed after a `CATEGORY`.

  Multiple categories can be added by specifying more pairs of c/ and b/.

Examples:

- `budget edit c/Food b/750 c/Shoes b/250`

Change budget amount from previous amount to `750` for the `food` category and budget amount from previous amount to `250` for the `shoes` category. Budget amount and category changed will be displayed.

- `budget edit c/laptop b/100`

  If budget has not been set, an error will be displayed.

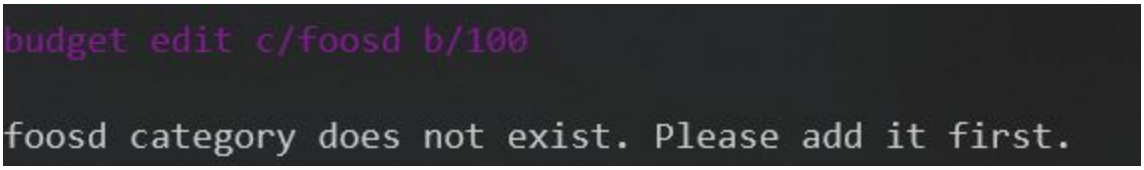

Figure 4. Editing a budget for nonexistent category

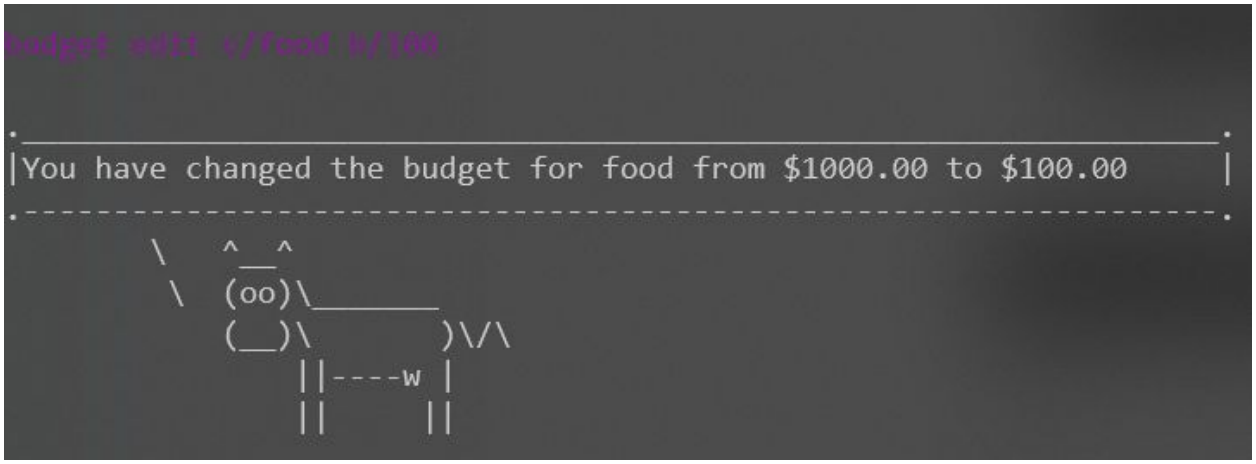

Figure 5. Editing a previously set budget for a valid category

List currently set budget.

Format: `budget list [c/CATEGORY]`

- `CATEGORY` must be an existing category.
- If `CATEGORY` is not specified, budget for all categories will be listed.

Examples:

- `budget list`

  Lists the `budget` for all categories set with a `budget`.

- `budget list c/Food c/Shoes`

  Lists the `budget` for the `food` and `shoes` category.

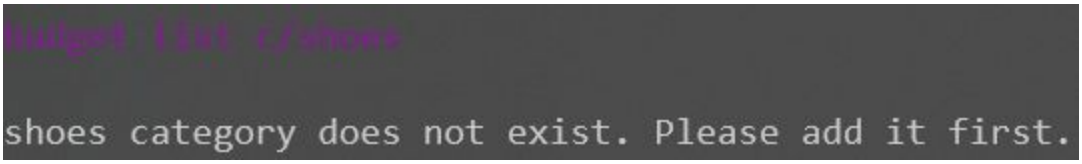

Figure 6. Listing a budget for a nonexistent category

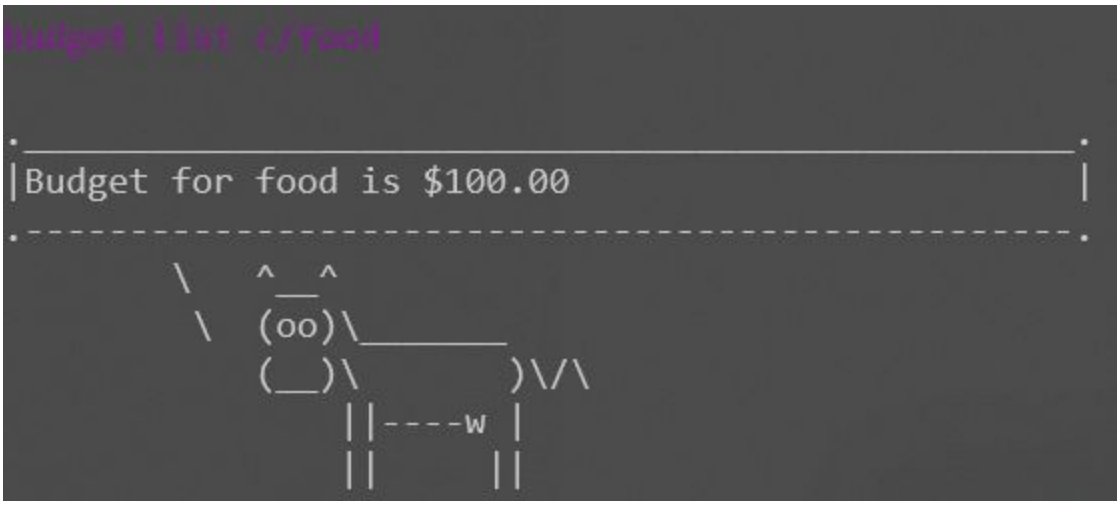

Figure 7. Listing a budget for a valid category

Figure 8. Listing budgets for all categories.

View the savings for each category

Format: `budget savings [c/CATEGORY] s/STARTMONTHYEAR [e/ENDMONTHYEAR]`

- `CATEGORY` must be an existing category.
- `STARTMONTHYEAR` and `ENDMONTHYEAR` should be a month and year value in this format: `02/2019` (February 2019). `ENDMONTHYEAR` should be a month and year after `STARTMONTHYEAR`.
- If there are no expenditures in the category, savings will equal budget.

Examples:

- `budget savings c/Food s/02/2019 e/10/2019`

  Views the total `savings` (budget set for that category - total expenditure of that month) for the `food` category from `February 2019` to `October 2019`.

- `budget savings s/01/2019`

  Views the total savings (budget set – total expenditure) for all categories for January 2019.

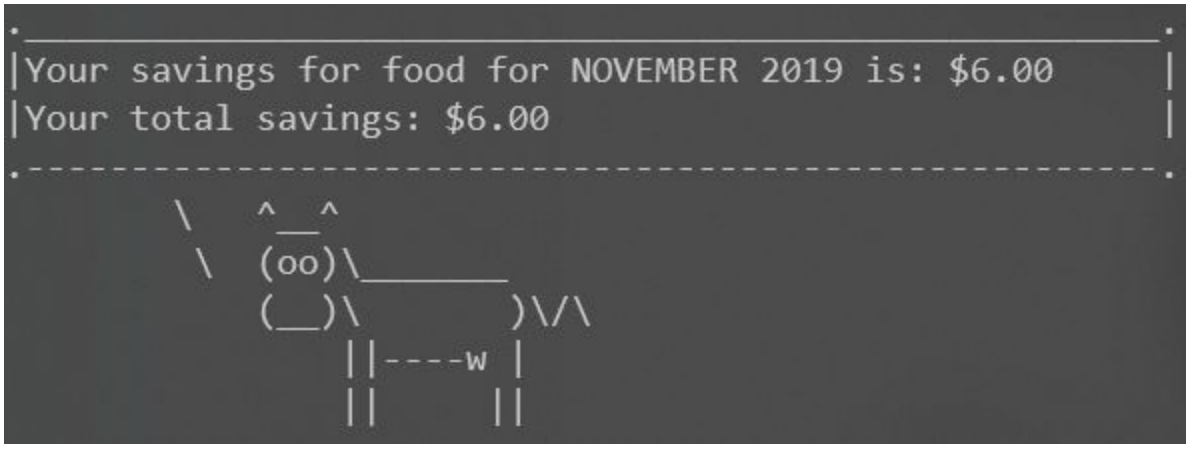

Figure 9. Showing the savings for food in November 2019

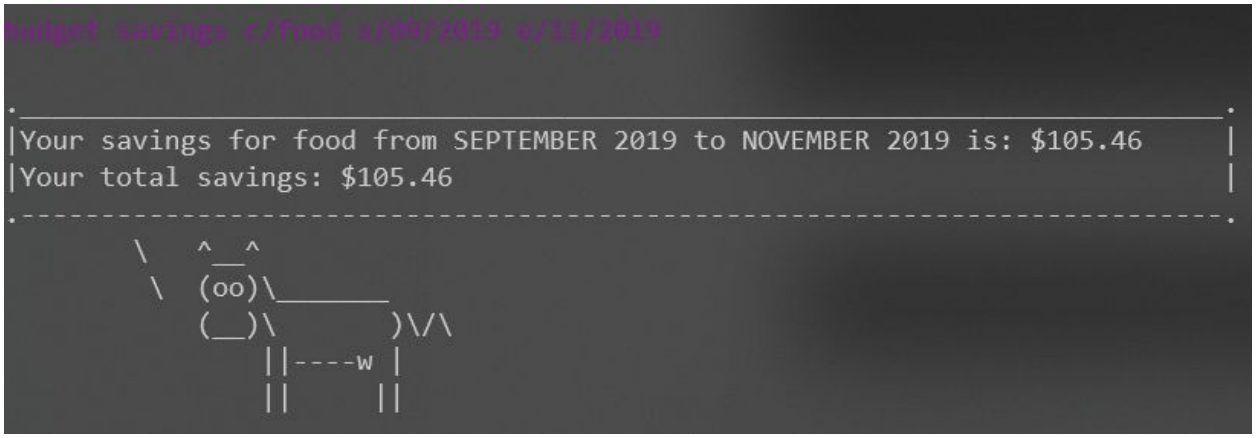

Figure 10. Showing the savings for food within the three months.

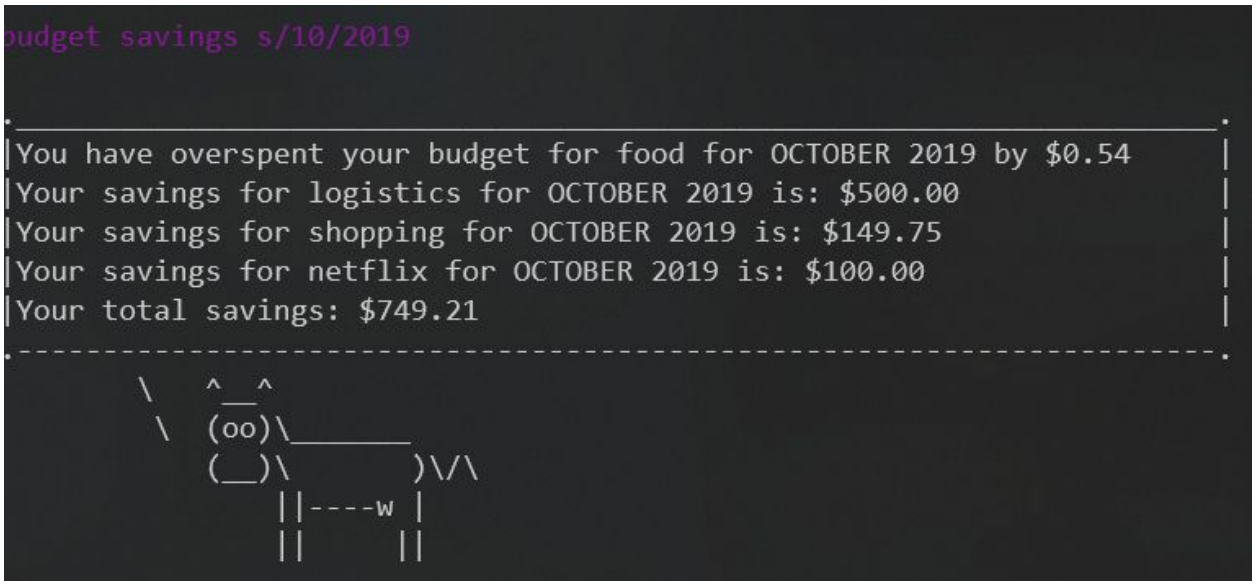

Figure 11. Showing the savings for all categories for October 2019

3.5. View Main Display : `view`

View the list of transactions based on category or all transactions.

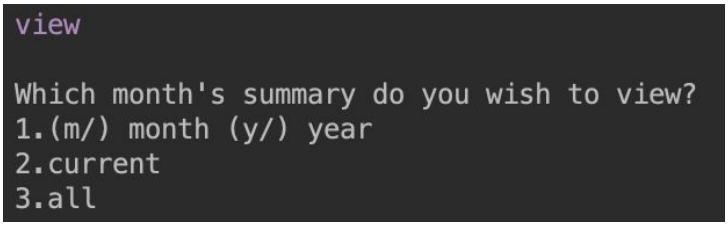

Figure 12. Showing the 3 different view commands

Format: `view m/MONTH y/YEAR`

- View all expenditures from all categories within the `MONTH` and `YEAR` specified

Examples:

- `view m/12 y/2019`
  Displays a table showing the expenditures within each category for December 2019

```
view m/12 y/2019

.-----------------.-------------------------------------------------------------------------------------------------------------------.
|Month: December  |                                                <Categories>                                                       |
|Year: 2019       |-----------------.-----------------.-----------------.-----------------.-----------------------------------------|
|                 |food             |transport        |entertainment    |logistics        |accomodation                             |
|                 |-----------------.-----------------.-----------------.-----------------.-----------------------------------------|
|                 |pizza    |$35.00 |van      |$22.00 |movie    |$12.20 |         |$      |hostel   |$546.00                |
|                 |ramen    |$14.50 |         |$      |skating  |$30.20 |         |$      |         |                       |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
|Total:           |$49.50           |$22.00           |$42.40           |$0.00            |$546.00                                  |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
|Budget:          |$500.00          |$300.00          |$300.00          |$200.00          |$600.00                                  |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
|Savings:         |$450.50          |$278.00          |$257.60          |$200.00          |$54.00                                   |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
```

Figure 13. View specific month and year summary

Format: `view current`

- Views all expenditures from all categories within the current `MONTH` and `YEAR` as not specified

Examples:

- `view current`
  Displays a table showing the expenditures within each category for November 2019

```
view current

.-----------------.-------------------------------------------------------------------------------------------------------------------.
|Month: November  |                                                <Categories>                                                       |
|Year: 2019       |-----------------.-----------------.-----------------.-----------------.-----------------------------------------|
|                 |food             |transport        |entertainment    |logistics        |accomodation                             |
|                 |-----------------.-----------------.-----------------.-----------------.-----------------------------------------|
|                 |laksa    |$5.00  |bus      |$3.40  |karaoke  |$20.30 |glue     |$1.90  |         |$                      |
|                 |curry    |$3.50  |taxi     |$24.50 |         |$      |papaer   |$8.00  |         |                       |
|                 |fishball |$2.20  |heli     |$60.00 |         |$      |         |       |         |                       |
|                 |sushi    |$12.50 |         |$      |         |       |         |       |         |                       |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
|Total:           |$23.20           |$87.90           |$20.30           |$9.90            |$0.00                                    |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
|Budget:          |$500.00          |$300.00          |$300.00          |$200.00          |$600.00                                  |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
|Savings:         |$476.80          |$212.10          |$279.70          |$190.10          |$600.00                                  |
.-----------------.-----------------.-----------------.-----------------.-----------------.-----------------------------------------.
```

Figure 14. View current month’s summary

Format: `view all`

- View all expenditures from all categories from all the entries stored regardless of time

Examples:

- `view all`
  Displays a table showing all the expenditures within all categories

Figure 15. View all months and all years summary

3.6. Display data as a graph: `graph`

Display data in a visual graph format.

Format: `graph c/CATEGORY m/MONTH y/YEAR` or `graph total`

- User selects the category and month to be displayed as a bar graph.
- The parameter `MONTH` is to be entered as a number from 1 - 12, corresponding to each of the 12 months of the year (i.e `1` = January, `2` = February, etc)
- The parameter `YEAR` is to be entered as a numerical value (eg. `2017, 2018, 2019,` etc) for the intended year to be displayed
- If no parameter for `m/MONTH` or `y/YEAR` is entered, i.e command is `graph c/CATEGORY`, a graph of the current month and year for the chosen category will be displayed by default.
- Entering `graph total` displays a bar graph for all categories instead of expenditure of a particular category, for the current month and year.

Examples:

- `graph c/food m/1 y/2019`

  Displays a bar graph of all the expenditure in the category `Food` for the month of January, 2019

- `graph c/food`

  Displays a bar graph of all the expenditure in the category `Food` for the current month and year.

- `graph total`

  Displays a bar graph of all categories

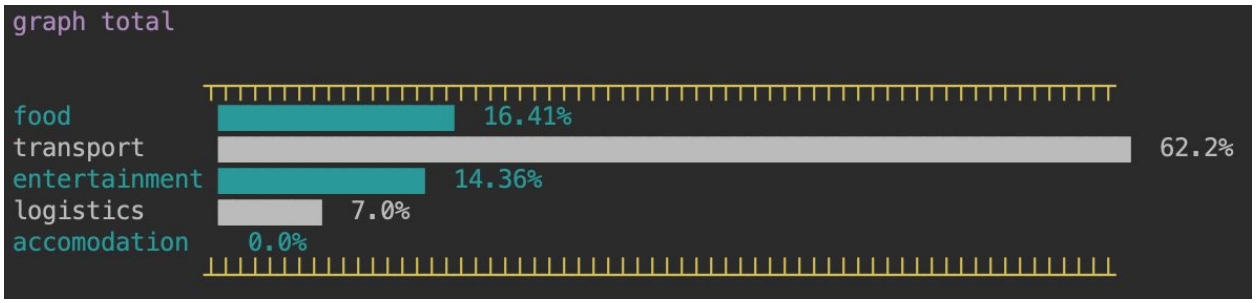

Fig 16.1 View of graph display

3.7. Setting a scheduled payment : `schedule`

Set a payment or bill in advance.

Format: `schedule d/DATE a/AMOUNT n/DESCRIPTION`

- User sets the payment in advance by providing a DATE, AMOUNT and DESCRIPTION input as shown in Fig 17.1.

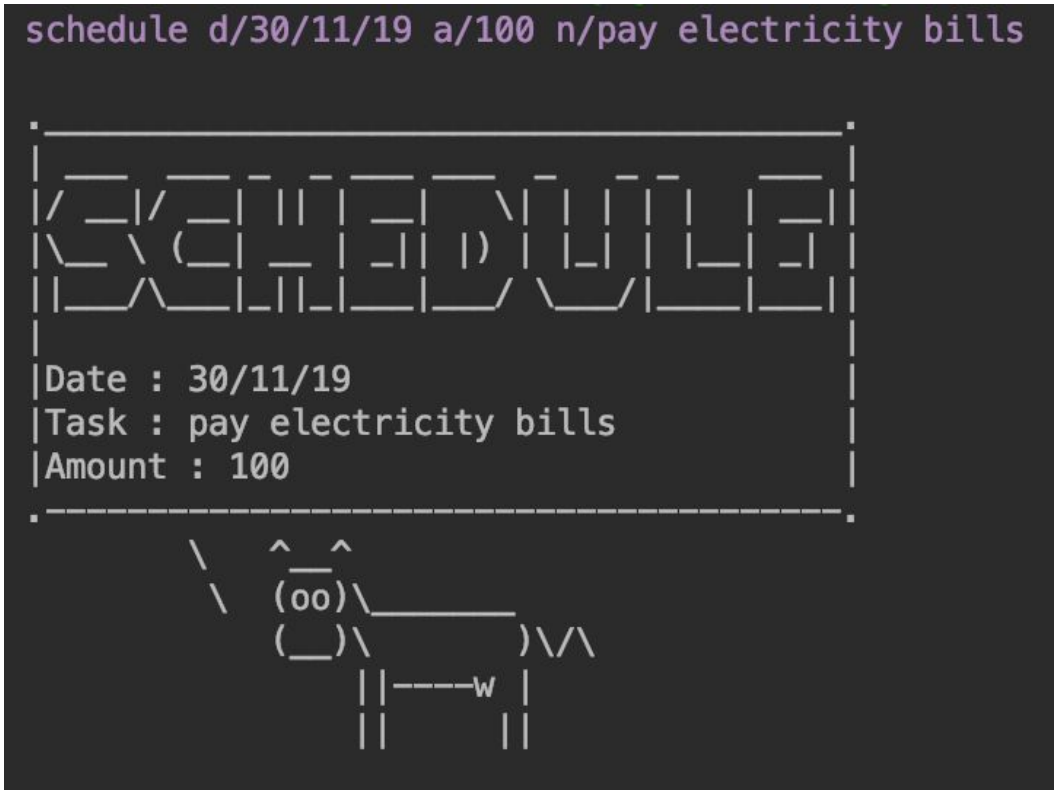

Fig 17.1. Adding a scheduled payment successfully

- The DATE parameter should be entered in the format dd/mm/yy (i.e 31st October 2019 = 31/10/19) as shown in Fig 17.2.

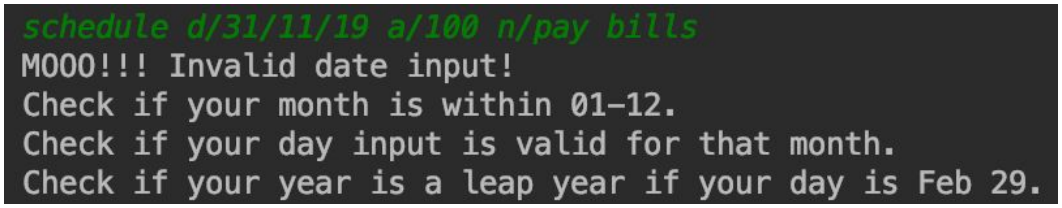

Fig 17.2 Adding a scheduled payment with invalid date

- The AMOUNT parameter must be entered as a number (i.e 4, 3.30, 1050, etc) as shown in FIg 17.3.

Fig 17.3 Adding a scheduled payment with invalid amount

- The DESCRIPTION parameter allows spaces (i.e pay bills, bills)

- Once a scheduled payment has been set, the user will be prompted when he runs the app on that day automatically. The program should display user's due payment in the following format as shown in the Fig 18 below.

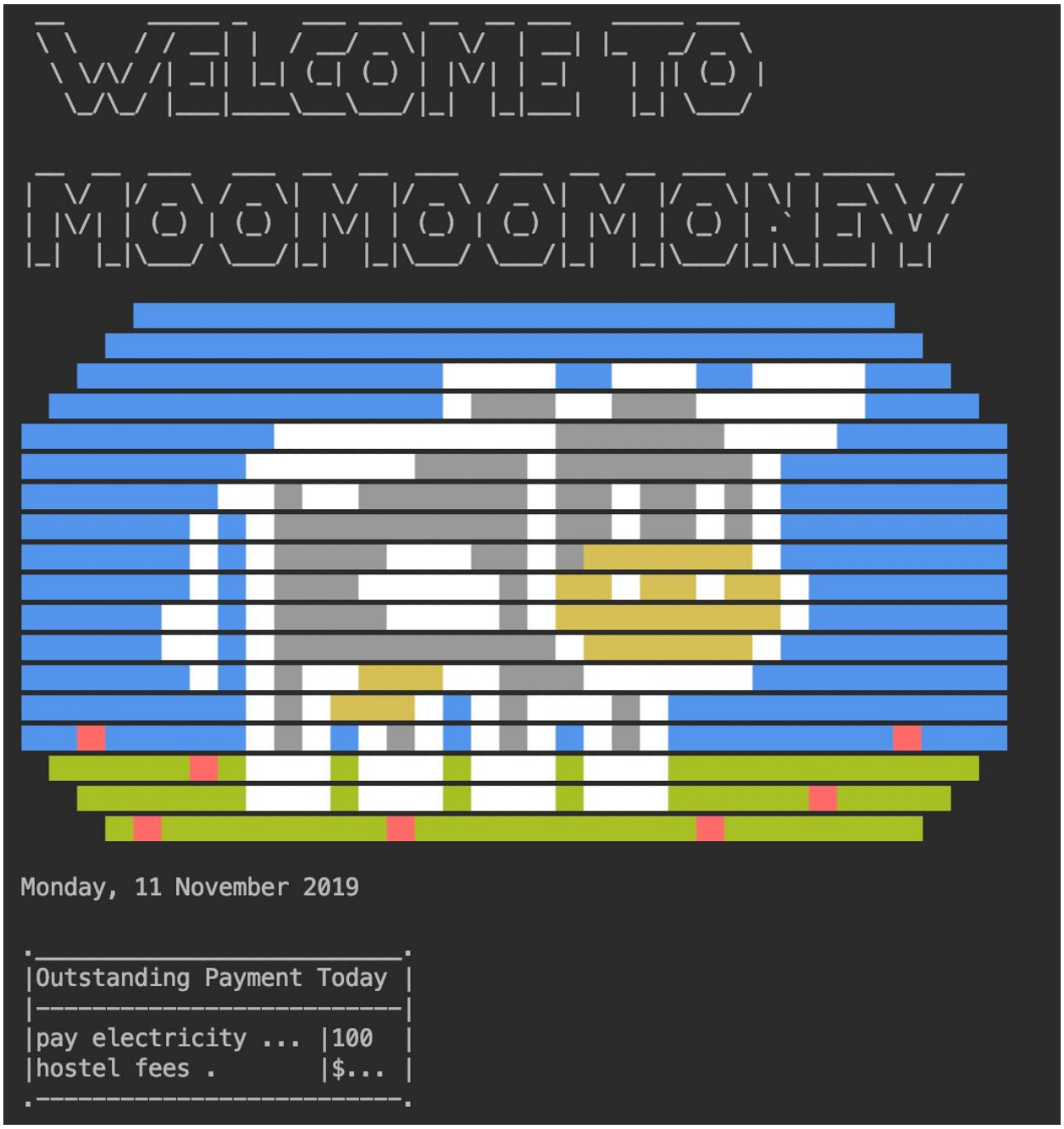

Fig 18 Display of outstanding payments

Examples:

- `schedule d/01/11/19 a/50 n/pay school fees`

Set a payment of $50 to pay school fees on 1st November 2019

- `schedule d/23/01/20 a/102.50 n/electricity bill`

  Set a payment of $102.50 to pay electricity bill on 23rd January 2020

- `schedule list`

  Shows list of scheduled payments sorted in order of earliest date.

3.8. Notifications for budget balance

Prompts user when user adds an expenditure to tell user if budget has exceeded or is going to reach its limit automatically. After user adds an expenditure, the user will also be notified of the balance left.

- User adds an expenditure as mentioned in Section 3.4.
- User will be notified on the current budget balance and if the user is reaching or has reached the limit. Fig 19 displays the various examples of how the user will be notified.

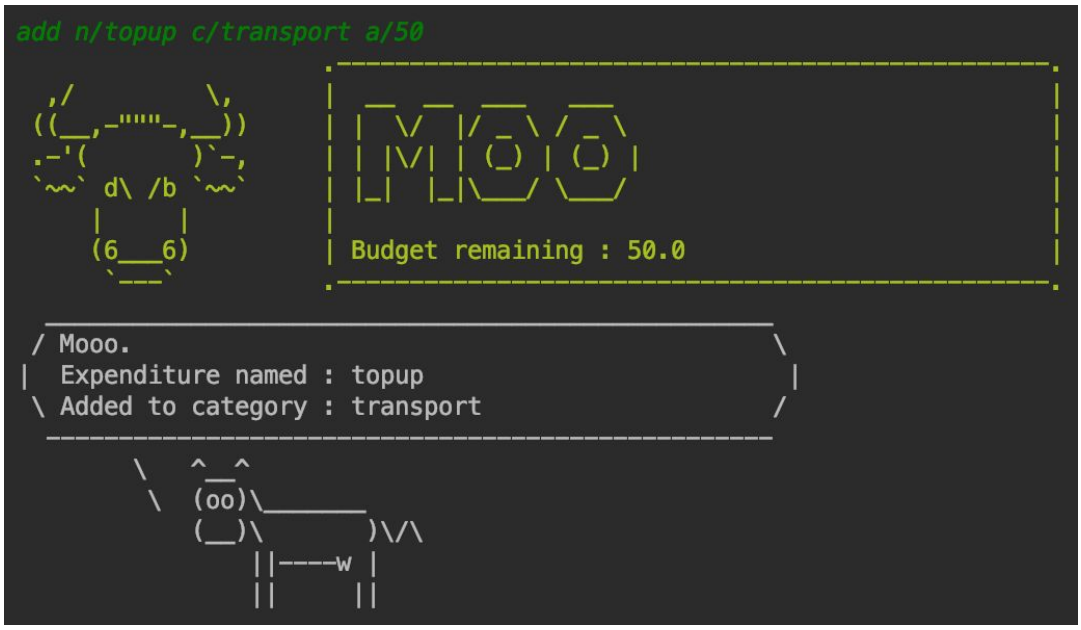

Fig 19.1. Notification when budget balance limit is not exceeded

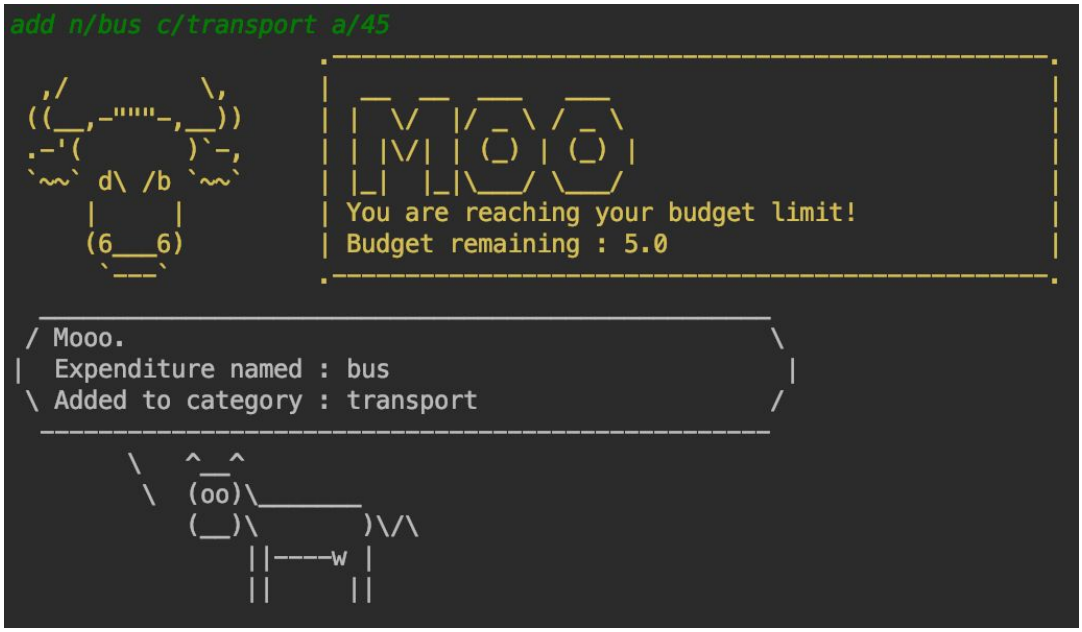

Fig 19.2. Notification when user is reaching the budget limit

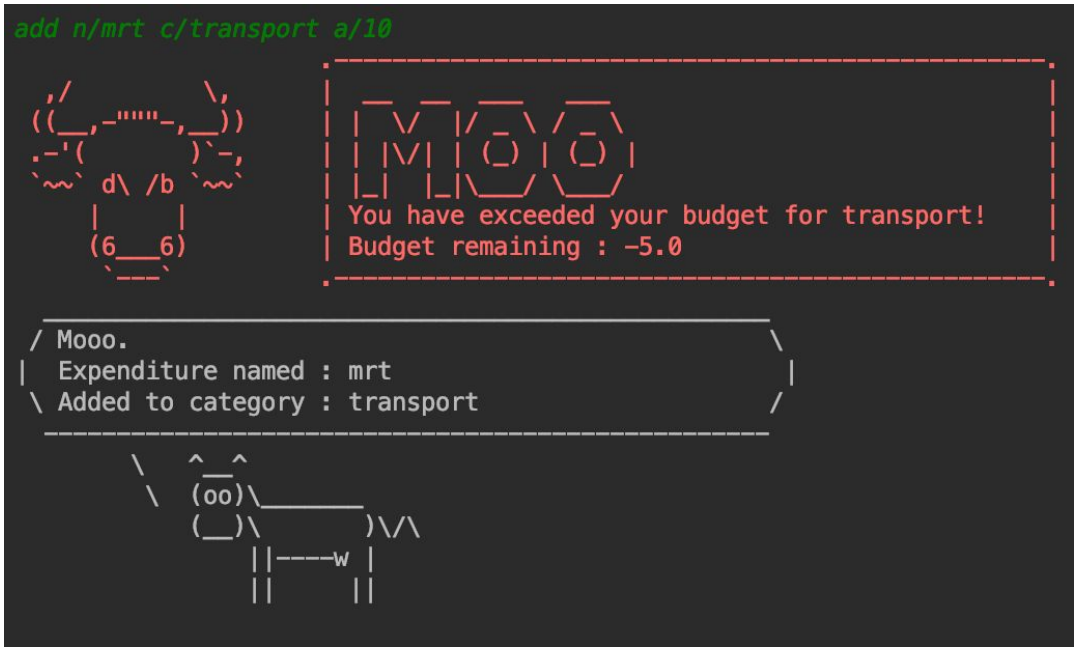

Fig 19.3. Notification when user exceeds the budget

3.9. Exiting the program : `bye`

Exits the program.

Format: `bye`

- Alternatively, users can also use `exit` to exit the program.

3.10. Moo : `moo`

The program will return a quirky response back to the user.

Format: `moo`

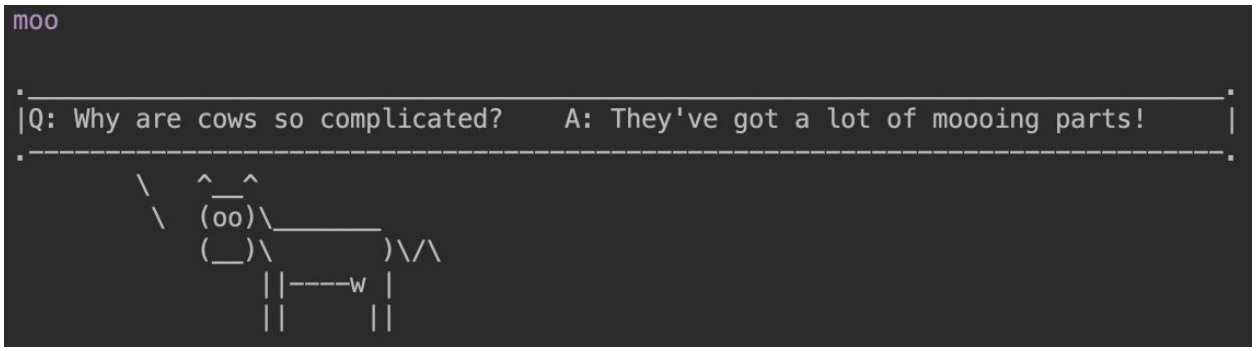

Figure 19.4. Moo response

3.11. Saving the data

All data are saved automatically to a file on the hard disk after any command that changes data is ran. As such, there is no need to do any manual saving.

3.12. Calculator `[coming in v2.0]`

A calculator will be available in the application to allow users to calculate their expenditure or savings without requiring an external program.

Format: `calculator`

3.13. Data Encryption `[coming in v2.0]`

Due to the need to store financial information, data encryption will be implemented to ensure that non-authenticated users will not be able to view the data.

3.14. Data Analytics `[coming in v2.0]`

As we have access to a user's expenditure data, we would apply some data analytics to predict a user's future expenditure or the areas where the user should cut down spending on.

4. FAQ

Q: How do I transfer my data to another Computer?

A: A folder called **data** will be created in the same location as the jar file, transfer the folder to another computer in the same location as the jar file.

Q: Is it possible to edit the data that is saved?

A: Yes, the data will be stored in the folder **data** in 4 separate text files. For example, **budget.txt** is in this format:

CATEGORY
AMOUNT

Q: Is an internet connection required?

A: No, all files are stored locally on the hard disk and as such, it is important that you backup the files as necessary.

5. Command Summary

- **Help** : `help`
- **List Categories** : `list`
- **Add Categories** : `category add CATEGORY`

e.g. `category add Food`

- **Delete Categories** : `category delete CATEGORY`
  e.g. `category delete 3`
- **Add Expenditure** : `add n/NAME a/AMOUNT c/CATEGORY (optional: d/DATE)`
  e.g. `add n/Nasi Lemak a/3 c/Food d/2019-11-20`
- **Delete Expenditure** : `delete i/INDEX c/CATEGORY`
  e.g. `add i/2 c/Food`
- **Sort Expenditure** : `sort TYPE`
  e.g. `sort name`
- **Manage Budget** : `budget set c/CATEGORY a/AMOUNT`
- e.g. `budget set c/Food b/100`
- **View Display**: `view m/MONTH y/YEAR`
- **Displaying Graph**: `graph c/CATEGORY m/MONTH y/YEAR`
  e.g. `graph c/Food m/10 y/2019`
- **Schedule Payments** : `schedule d/DATE a/AMOUNT n/DESCRIPTION`
- **Exiting the program** : `bye`
- **Moo** : `moo`