Financial Ghost - User Guide

By: CS2113T-W12-2   Since: September 2019 Licence: NUS

1. Introduction

Financial Ghost is a desktop app to manage finances. It is targeted towards fresh graduates that are responsible for bringing up their own children and for the care for their aging parents in the future. Financial Ghost is also for those who prefer to use a desktop app to manage their finances. More importantly, Financial Ghost is optimized for those who prefer to work with a Command Line Interface (CLI) while still having the benefits of a Graphical User Interface (GUI). If you can type fast, Financial Ghost can get your finance management done faster than traditional GUI apps. Interested? Jump to the Section 2, "Quick Start" to get started. Enjoy!

2. Quick Start

This section serves as a tutorial for you to familiarize yourself with the usage of Financial Ghost. By following the steps below, getting started with Financial Ghost should be a fast and smooth process.

2.1. Installation

1. Ensure you have Java version 1.8 or later installed in your Computer.
2. Download the latest version of FinancialGhost.jar here:
   https://github.com/AY1920S1-CS2113T-W12-2/main/releases
3. Copy the file to the folder you want to use as the home folder for your Financial Ghost.
4. Double-click the file to start the app. You should see the GUI (Graphical User Interface) in a few seconds as shown below.

2.2. User Interface

The mockup of the GUI is shown in the image below.
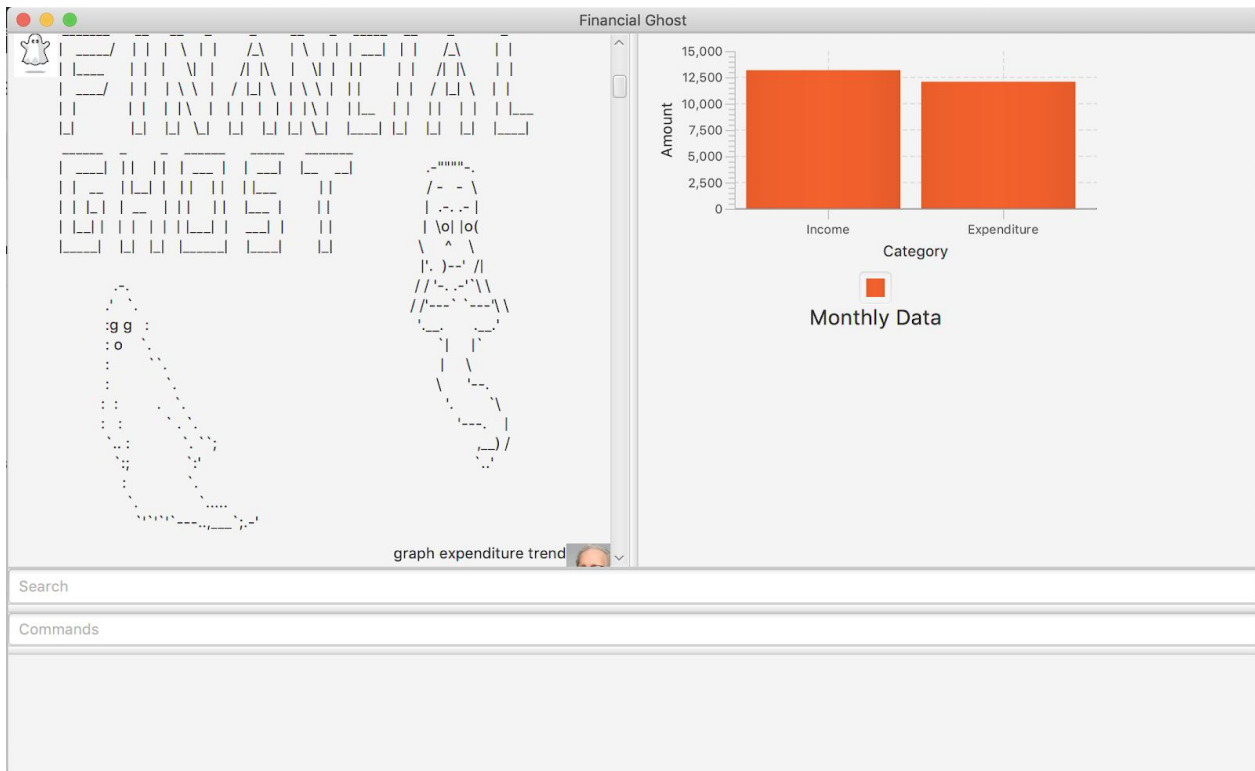
The table below summarises the function of each component.

Command Bar	The Text Bar on the bottom with prompt text "Commands" inside is where the Commands is being typed by the user.
Command Result Panel	As seen on top left, this is the panel where the result of the command entered by the user is shown.
Search Bar	The Search Bar is right on top of the command bar with the prompt text "Search" inside the bar. This is where searching of all the finances will be done.
Account panel	This is the panel on the right. This will be where the information of the account will be shown, whether it is through searching by the user or through command by the user.

2.3. Executing Commands

Below are some steps to teach the user how to use the application.

1. Type the Commands (Case Sensitive) in the Command Bar and press `Enter` to execute it.
2. Some examples you can try:
   - `list all instalments`: List out all the instalments of the user in the Account Panel
   - `lent Max Chan /amt 200 /on 9/10/2019`: Record the lending of $200 to a person named Max Chan

For the details of each command, you may refer to the Features Section below.

3. Features [v1.4]

Below are implementations that we are releasing in version 1.4.

3.1. Add income:`income`

Allows the user to track total income for a holistic evaluation of total money inflow

Done with commands to add and edit income sources

Format: `income [description] /amt [amount in dollars] /at d/mm/yyyy`

- Adds an income source to list of total income

`list all income`

- Views all income sources listed

`delete income [index]`

- Deletes selected income source from income list by index

Examples:

- `income Teaching Assistant /amt 480 /at 28/09/2019`
- `income delete 3`

Extension Type: [Basic Tracking]

3.2. Add expenditure: `spent`

Allows users to track total expenditure for a holistic evaluation of total money outflow

Done with commands to view, add and edit past expenditure

Format: `spent [description] /amt [amount in dollars] /cat [category] /at d/mm/yyyy`

- Adds an expenditure to total expenditure list

`list all expenditure`

- Views all past expenditures

`delete expenditure [index]`

- Deletes selected expenditure from expenditure list by index

Examples:

- `spent Menya Sakura /amt 20 /cat food /at 15/08/2019`
- `delete expenditure 2`

Extension Type: [Basic Tracking]

3.3. Financial Report for The Current Month: `graph monthly report`

Shows the total amount of income and expenditure for the current month.

Format: `graph monthly report (pie_chart/histogram/line_graph)`

- `(pie_chart/histogram/line_graph)` can be omitted then Financial Ghost will show the default graph for the monthly report which is histogram.

Examples:

- `graph monthly report`
- `graph monthly report pie_chart`

Extension Type: [C-Statistics]

3.4. Income/Expenditure Categories: `graph income/expenditure trend`

Shows a graph of the distribution of income/expenditure based on the type.

Format: `graph income/expenditure trend (pie_chart/histogram/line_graph)`
- `(pie_chart/histogram/line_graph)` can be omitted then Financial Ghost will show the default graph for the category trend which is line graph.

Examples:

- `graph income trend`
- `graph expenditure histogram`

Extension Type: [C-Statistics]

3.5. Graph for Financial Status: `graph financial status`

Shows a graph of near 3 months' income and expenditure until the given date.

Format: `graph financial status /until [date]`

Examples:

- `graph finance status /until 12/12/2019`
- `graph finance status /until 5/7/2019`

Extension Type: [C-Statistics]

3.6. Timing Shortcuts: `- now`
`- ytd`
`- tmr`
`- lstwk`
`- nxtwk`
`- lstmth`
`- nxtmth`
`- lstyr`
`- nxtyr`

Adds the shortcut to input certain timings when adding items

- Users can choose to input shortcuts to specific times instead of the actual date for add commands
- Shortcuts for timings such as now, tomorrow and yesterday can be chosen

Examples:

- ```
  [input] spent New glasses /amt 150 /by now
  [UI] Got it! I've added the following expenditure
          [E]$150.0 New glasses (on: <today's date>)
  ```
- ```
  [input] spent Koi green tea macchiato /amt 4 /by ytd
  [UI] Got it! I've added the following expenditure
          [E]$4.0 Koi green tea macchiato (on: <yesterday's date>)
  ```

Extension Type: [C-Friendlier Syntax]

3.7. Undo: `undo`

The user is able to undo up to 5 latest commands they have issued to the program.

Format: `undo`

Entering `undo` reverts commands to the previous state before the command was implemented. This affects the following types of commands:

- Commands that add to the account database
- Commands that delete an entry from the account database
- Commands that alter an entry from the account database

Commands like `list` or `graph` that do not involve making changes to the database will print an error message: "`Command can't be undone!`".

If program prints message: "`No commands to undo!`" it could mean that there were no commands prior to the undo command were issued to the program, or undo has been called more than 5 times in succession.

Extension Type:[C- Undo]

3.8. Search for items: `find`

The user is able to search for items grouped according to their respective types. The keyword/s are to be entered in the search bar. The search results are case sensitive.

Format: `[keyword/s]`

Extension Type: [C- BetterSearch]

3.9. Goal setting: `goal`

Allow the user to track their goals by computing the recommended amount of savings per month to reach their goals.

Format: `goal [desc] /amt [cost] /by [d/M/yyyy] /priority [priority level]`

Examples:

- `goal buy HDB /amt 100000 /by 9/12/2030 /priority HIGH`
- `goal buy Lambo /amt 150000 /by 10/12/2040 /priority MEDIUM`
- `goal buy Boat /amt 200000 /by 11/12/2050 /priority LOW`

Extension Type: [D-Goals]

3.10. List and check goal progress: `list goals`

Lists and checks for the progress the user has made towards their set goals e.g percentage to completion and savings required per month vs actual savings per month

Format: `list goals`

Extension Type: [D-Goals]

3.11. Complete set goal: `done goal`

Allows the user to mark their goal as complete and automatically converts goal into the specified month's expenditure.

Returns the remaining goals and shows the user the updated progress the user has made towards their goals.

Format: `done goal [index]`

Examples:

- `done goal 1`

Extension Type: [D-Goals]

3.12. Goal planner: `commit goal`

Allows the user to plan his finances before he commits to spending money to complete their set goal.

Returns the remaining goals if he had completed the specified goals and shows the user the goals that he can complete with their remaining amount of money.

Format: `commit goal [index]...`

Examples:

- `commit goal 1`
- `commit goal 1,2,3`

Extension Type: [D-Goals]

3.13. Loan Tracking: `lent borrowed settled`

Allows the user to keep track of the money borrowed/lent from/to other parties e.g friends/colleagues/organisations.

Includes command to indicate status of debt (settled/unsettled)

If outgoing loans are settled, loan is added to expenditure automatically

If incoming loans are settled, loan is added to income automatically

Format:

- `lent [other party] /amt [cost] /on [date]`

- `borrowed [other party] /amt [cost] /on [date]`

- `paid [amount] /to [other party]` (for settling incoming loans)

- `received [amount] /from [other party]` (for settling outgoing loans)

Examples:

- `lent Max Chan /amt 200`

- `borrowed Daniel Chan /amt 100`

- `received 6 /from Max Chan`

- `Paid 25 /to Daniel Chan`

Extension Type: [D-Loans]

3.14. Shared Expenditures: `split settle`

Allows the user to add expenditures meant to be split among others

Individual debts settled will automatically be added to income

Format:

- `split [description] /amt [amount in dollars] /on [date] /with [other party]`
  `and [other party] and …`
- `settled [index] [index/name of party]`

Examples:

- `spentSplit PGP Mala /amt 40 /on 27/09/2019 /with Sean Chan and Abhijit and`
  `Max Chan`

- `spentSplit 8 Abhijit`

Extension Type: [D-Loans]

3.15. Check Archive: `check income/expenditure`

Allows the user to check their income or expenditure records for previous months

Also allows users to check the income and expenditure for the month

Format:

- `check income [month] [year]`
- `check expenditure [month] [year]`
- `list month income`
- `list month expenditure`

Examples:

- `check income 10 2019`

- `check expenditure 5 2005`

Extension Type: [C-Archive]

3.16. Instalments Tracking: `add instalment`

Allows the user to keep track of the uncompleted payments that they have to settle via recurring monthly instalments.

Format: `add instalment [desc] /amt [cost] /within [number of months] months /from [d/M/yyyy] @[annual interest rate]%`

- The output will list all the instalments with the added instalment.

Examples:

- `add instalment mortgage /amt 100000 /within 200 months /from 12/12/2010 @6%`

Extension Type: [D-instalments]

3.17. Check Instalments: `delete instalments`

Allows the user to check all of the uncompleted instalments that they have to settle via recurring monthly instalments. Also shows a percentage bar at the same time.

Format: `delete instalments [index]`

- The output will list all the instalments without the deleted instalment.

Examples:

- `delete instalments 3`

Extension Type: [D-instalments]

3.18. Check Instalments: `list all instalments`

Allows the user to check all of the uncompleted instalments that they have to settle via recurring monthly instalments. Also shows a percentage bar at the same time.

Format: `list all instalments`

- The output will list all the instalments.

Examples:

- `list all instalments`
- `[output]`

1. car (200 per month until 12/12/2030)

2. laptop (50 per month until 2/3/2020)

Extension Type: [D-instalments]

3.19. Recurring Instalments

Auto deduct monthly as a part of Expenditure until the last month and from there on it will terminate. Notify the user monthly when it deducts and when it ends. Auto remove it when it is done.

Extension Type: [D-instalments]

3.20. Create A Bank Account Tracker `bank-account`

Allows the user to create a bank account tracker to track the balance and interest rate of the bank account.

Format: `bank-account [description] /amt [initial amount of money] /at [initial date] /rate [interest rate]`

- Create a bank account tracker to track and manage the bank accounts

Examples:

- [input] `bank-account OCBC /amt 0 /at 27/7/2017 /rate 0.005`
- [input] `bank-account DBS /amt 100 /at 14/8/2018 /rate 0`

Extension Type: [D-Bank]

3.21. Delete A Bank Account Tracker: `Delete bank-tracker`

Allows the user to delete the bank tracker they do not want based on the index of the bank account tracker in the list.

Format: `delete bank-tracker [index of the bank account tracker]`

- The index should be a positive number and small then the total number of trackers currently existing.
- The user may need to use list command to check the index of the tracker first.

Examples:

- [input] delete bank-tracker 1
- [input] delete bank-tracker 3

Extension Type: [D-Bank]

3.22. List Bank Account Trackers: `list bank trackers`

Allows the user to check all the bank account trackers recorded in Financial Ghost.

Format: `list bank trackers`

- The index of the tracker in the list can be used to run the delete bank-tracker command

Examples:

- List bank trackers

Extension Type: [D-Bank]

3.23. Internal Transfer: `deposit/withdraw`

Allows the user to keep track of deposit and withdraw money. When the user deposit money, the money will also be counted as a new income and same for withdrawing the money.

Format: `deposit/withdraw [amount] [account_description] /at [date]`

Examples:

- `deposit 200 OCBC /at 28/9/2019`
- `withdraw 50 DBS /at 25/9/2019`

Extension Type: [D-Bank]

3.24. Check balance: `check-balance`

Allows the user to check the balance in their accounts at a specific date after the initial date based on the current statistics.

Format: `check-balance [account_description] /at [future_date]`

Examples:

- `check-balance OCBC /at 12/3/2020`

Extension Type: [D-Bank]

3.25. Exiting the program: `bye`

Exits the program.

Format: `bye`

3.26. Saving the file

Financial Ghosts writes data to the hard disk automatically after any command that changes the data.

There is no need to save manually.

4. Features [coming in 2.0]

Below are future implementations that we Plan to release in version 2.0.

4.1. Login: `login`

The user needs password to access this software.

- If it is the first time to use Financial Ghost, the software will require the user to setup his/her account.
- After the first login, the password will be required to access Financial Ghost.

Examples:

- `[output] What is the password?`

Extension Type: [C-Security]

4.2. Input Stages

UI prompts the user for commands with the correct syntax, guiding the user and allowing them to input commands in stages

- User can just input the command type, and the UI would proceed to prompt the user with the proper syntax for the next input according to the command given.
- Gives users the choice to input commands in one-shot or in stages

Examples:

- ```
  [User] spent
  [UI] What did you spend on?
  [User] Pinky Pasta
  [UI] How much was it?
  [User] 6.50
  [UI] When did you buy it? Please input as: d/mm/yyyy
  [User] 23/09/2019
  [UI] Got it! I've added the following expenditure
        [E]$6.5 Pinky Pasta (on: 23/09/2019)
  ```

Extension Type: [C-Friendlier Syntax]

4.3. Fix-time deposit: `time-deposit`

Allows the user to keep track of time deposit with different interest rates. Financial Ghost will record and also calculate the amount of money in the future.

Format: `time-deposit [amount] [account_description] /from [start_date] /to [end_date]`

Examples:

- `time-deposit 200 OCBC /from 28/9/2019 /to 28/3/2020`
- `time-deposit 50 DBS /from 25/9/2019 /to 25/3/2020`

Extension Type: [D-Bank]

4.4. Monthly in-out trend: `trend`

Displays histogram of the financial statement for the selected year while displaying the average amount of money in and money out for the past 12 months.

Format: `trend [year]`

- The `[year]` must be a year after 2010.

Examples:

- `trend 2019`

Extension Type: [C-Statistics]

4.5. Command Auto-complete

The UI will suggest the appropriate command and command format to use based on the characters entered by the user

- After the first login, the password will be required to access Financial Ghost.

Examples:

- `[input] income -> [suggested command] income [description] /amt [amount in dollars] /at d/mm/yyyy`
- `[input] fi -> [suggested command] find`

Extension Type: [C-Help]

4.6. Suggested Corrected Exception

When the input string cannot be recognized, an exception will be thrown and the error messages will be prompted in order to guide the user.

Extension Type: [C-Help]

4.7. Help Page

General Help page. When the user types in the wrong command. Exception will be thrown to them and suggest them to look up the help page.

Format: `helppage`

- Throws out a help page.

Extension Type: [C-Help]

5. FAQ

Q: How do I fix the data file in the case that the file becomes corrupted?
A: Navigate to the root directory of Financial Ghost folder located in your computer. Go the the data file located in the filepath: FinancialGhost/data/moneyAccount.txt and delete the line containing the error displayed in the error message.

Q: What if I wanted to change my monthly expected expenditure?

6. Command Summary

- Add Income: `income [desc] /amt [amount in dollars] /payday d/mm/yyyy`
  `e.g income Teaching Assistant /amt 480 /payday 28/09/2019`
- Add Expenditure: `spent [desc] /amt [amount in dollars] cat/ [category] /at`
  `d/mm/yyyy`
- `e.g spent Menya Sakura /amt 20 /at 15/08/2019`
- Delete Income: `delete income [index]`
- Delete Expenditure: `delete expenditure [index]`
- List Full Income History: `list all income`
- List Full Expenditure History: `list all expenditure`
- List Current Month Income History: `list month income`
- List Current Month Expenditure History: `list all expenditure`
- List Past Month Income History: `check income [month] [year]`
- List Past Month Expenditure History: `check expenditure [month] [year]`
- Add Goal: `goal [desc] /amt [cost] /by [d/M/yyyy] /priority [priority level]`
- Delete Goal: `delete goal [index]`
- List Goals: `list goals`
- Complete Goal: `done goal [index]`
- Commit Goal: `commit goal [index]...`
- Add Instalment: `add instalment [desc] /amt [cost] /within [number of months]`
  `months /from [d/M/yyyy] @[annual interest rate]%`
- Delete Instalment: `delete instalment [index]`
- List Instalments: `list all instalments`
- Add Incoming Loan: `lent [other party] /amt [cost]`
- Add Outgoing Loan: `borrowed [other party] /amt [cost]`
- Delete Loan: `delete loan [index]`
- List Loans: `list all loans`
- List Incoming Loans: `list incoming loans`
- List Outgoing Loans: `list outgoing loans`
- Settle Incoming Loan: `paid [amount] /to [other party]`
- Settle Outgoing Loan: `received [amount] /from [other party]`
- Share Expenditure: `split [desc] /amt [amount in dollars] /on d/m/yyyy`
  `/parties [other party] and [other party] and ...`
- Settle Split Expenditure: `settle [index] [index/name of party]`
- Create A Bank Account Tracker: `bank-account [desc] /amt [initial amount of`
  `money] /at [initial date] /rate [interest rate]`
- Delete A Bank Account Tracker: `delete bank-tracker [index of the tracker]`
- Predict Balance: `check-balance [desc] /at [the future date]`
- Deposit: `Deposit [amount] [desc] /at [date]`
- WIthdraw: `Withdraw[amount] [desc] /at [date]`
- List Bank Tracker: `list bank trackers`
- Generate Current Month's Financial Report: `graph monthly report`

- Generate The Graph of Expenditure/Income Categories: `graph income/expenditure trend`
- Generate financial status: `graph financial status /until [date]`
- Undo: `undo`
- Login: `login`
- Help Page: `helppage`
- Timing Shortcuts: `now ytd tmr lstwk nxtwk lstmth nxtmth lstyr nxtyr`
- Exiting the program: `bye`