Financial Ghost - User Guide

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

- 1. Introduction
- 2. Features(v1.1)
    - 2.1. Add a ToDos Task to List: `todo`
    - 2.2. Add a Deadline Task to List: `deadline`
    - 2.3. Add an Event Task to List: `event`
    - 2.4. Add a Period Task to List: `period`
    - 2.5. Add a Fixed Duration Task to List: `duration`
    - 2.6. Add a Multiple Event to List: `multiEvent`
    - 2.7. Listing all the Tasks: `list`
    - 2.8. Mark a task as done: `done`
    - 2.9. Locating tasks by name: `find`
    - 2.10. Deleting a task: `delete`
    - 2.11. Change the type or the date of an existing task: `reschedule`
    - 2.12. Choose the confirmed date of a Multiple Event: `choose`
    - 2.13. Get the current day's reminders: `today reminders`
    - 2.14. Get upcoming reminders: `upcoming reminders`
    - 2.15. Get past reminders: `past reminders`
    - 2.16. View schedule for a specific day: `schedule`
    - 2.17. Find a free time slot based on a given specific time: `free-time`
    - 2.18. Exiting the program: `bye`
    - 2.19 Saving the data
- 3. Features (Coming in v1.4)
    - 3.1 Add income: `income`
    - 3.2 Add expenditure: `spent`
    - 3.3 Detailed financial report for a specific month: `finance`
    - 3.4 Table of expenditure categories: `category`
    - 3.5 Login: `login`
    - 3.6 Help Page: `helppage`
    - 3.7 Suggested Correction Exception
    - 3.8 Input stages
    - 3.9 Timing Shortcuts: `now ytd`
    - 3.10 Undo: `undo`
    - 3.11 Search for Items: `find`
    - 3.12 Short term goal setting: `goal-short`
    - 3.13 Check goal progress: `check goals`
    - 3.14 Complete set goal: `done goal`
    - 3.15 Loan tracking (for small loans): `lent borrowed settled`
    - 3.16 Loan tracking (for large loans): `lent borrowed`

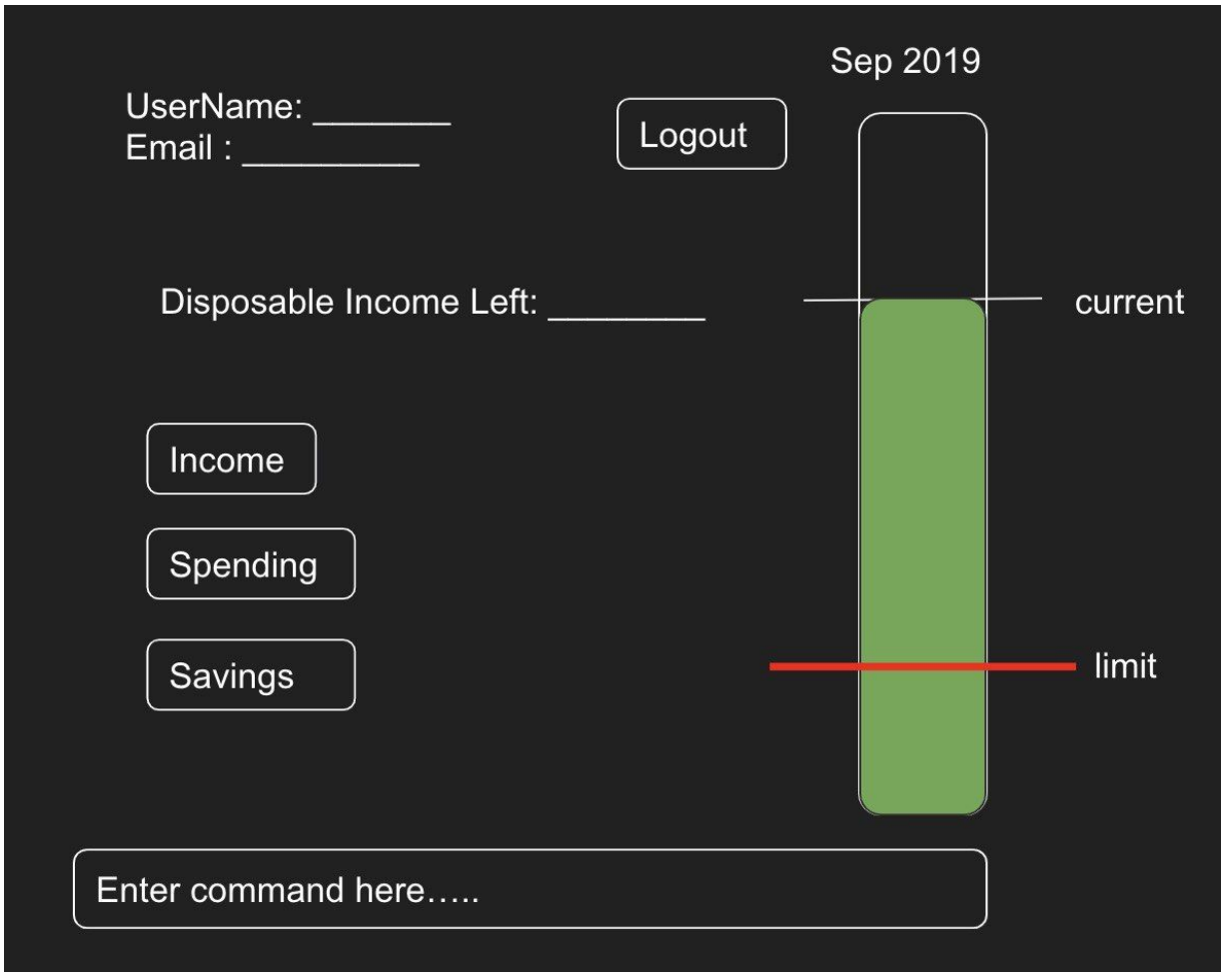
    - 3.17 Shared expenditures: `spentSplit`
    - 3.18 Instalments tracking: `instalment`
    - 3.19 Check Instalments: `check instalments`
    - 3.20. Recurring Instalments
    - 3.21 Recurring Bills: `bills`
    - 3.22. Reminder to Pay Bills
    - 3.23. Create A Bank Account Tracker: `bank-account`
    - 3.24. Income and Expenditure for specific bank account: `income spent`
    - 3.25. Internal Transfer: `deposit withdraw Transfer`
    - 3.26. Check balance: `check-balance`
- 4. Features (Coming in v2.0)
    - 4.1 Check goal savings progress:`check goals savings`
    - 4.2 Goal planner commit goal: `commit goal`
    - 4.3 Fix-time deposit: `time-deposit`
    - 4.4 Monthly finance in-out trend: `trend`
    - 4.5. Command Auto-complete

1. Introduction

In Phase 1, we built Duke as the code base for Financial Ghost. Duke is for those who prefer to use a desktop app for managing tasks. More importantly, Duke 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, Duke can get your management tasks done faster than traditional GUI apps.

In Phase 2, Duke is morphed to Financial Ghost to be 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. The app also operates with a Command Line Interface (CLI) with a Graphical User Interface (GUI).

The mockup of the GUI is shown in the image below.



2. Features (v1.1)

2.1. Add a ToDos Task to List : `todo`

Adds a ToDos Task to the Duke Task list

Format: `todo [Desc]`

Examples:

- `todo borrow book`
- `todo get money`

2.2. Add a Deadline Task to List: `deadline`

Adds a Deadline Task with the deadline date to the Duke Task list

Format: `deadline [Desc] /by d/m/yyyy HHmm`

Examples:

- `deadline Duke User Guide /by 17/09/2019 2359`
- `deadline Duke Developer Guide /by 17/09/2019 2359`

2.3. Add an Event Task to List: `event`

Adds an Event Task to the Duke Task list detailing the time and duration of the event

Format: `event [Desc] /at d/m/yyyy HHmm to d/m/yyyy HHmm`

Examples:

- `event code jam /at 6/6/2011 1100 to 7/7/2011 1400`
- `event Circles.Hack /at 12/09/2019 1200 to 21/9/2019 1400`

2.4. Add a Period Task to List: `period`

Adds a Period Task to the Duke Task list for tasks to be done within a specified period of time

Format: `period [Desc] /from d/m/yyyy HHmm /to d/m/yyyy HHmm`

Examples:

- `period code jam /at 6/6/2011 1100 /to 7/7/2011 1400`
- `period Circles.Hack /at 12/09/2019 1100 /to 21/9/2011 1400`

2.5. Add a Fixed Duration Task to List: `duration`

Adds a Fixed Duration Task to the Duke Task list for tasks to be done that requires a certain duration

Format: `duration [Desc] /needs [Desc]`

Examples:

- `duration Math practice paper /needs 2 hours`
- `duration Wedding Dinner /needs 4 hours`

2.6. Add a Multiple Event to List: `multiEvent`

Adds an event with several possible dates to the Duke Task list, with the functionality to finalise on one date in the future

Format:

`multiEvent event_name /at d/m/yyyy HHmm to d/m/yyyy HHmm /or d/m/yyyy HHmm to d/m/yyyy HHmm /or ...`

- User is able to add an indefinite number of possible dates as long as they are in the correct format

`Choose [task index] [date choice index]`

- Task index refers to the index number shown in the displayed task list
- Date choice index starts from 1 and should not exceed the number of possible dates for the given multiEvent

Examples:

- `multiEvent peer review session /at 1/12/2009 2232 to 1/12/2009 2234 /or 2/12/2009 2232 to 2/12/2009 2234 /or 3/12/2009 2250 to 3/12/2009 2359`
  adds an event “peer review session” with 3 possible dates
- `choose 3 2`
  Finalises the second date for a multiEvent with the index of 3 in the tasklist
  After the command, it would be finalised to this:
  `multiEvent peer review session /at 2/12/2009 2232 to 2/12/2009 2234`

2.7. Listing all the Tasks : `list`

Shows a list of all tasks in Duke.

Format: `list`

2.8. Mark a task as done: `done`

When a task is completed this command can be used to mark the task as done.

Format: `done [index]`

- Marks the task as done at the specified [index].
- [index] refers to the task as shown in the Duke Task List
- [index] must be a positive integer e.g 1,2,3.

Examples:

- `done 1`

2.9. Locating tasks by name: `find`

Finds a Task which description contains any of the given keywords.

Format: `find [keywords]`

- The search is case insensitive. e.g `help` will match `Help`
- The order of the keywords does not matter. e.g. `Help me` will match `me Help`
- Only the description name is searched.
- Tasks matching at least one keyword will be returned. e.g. `Help` will return `Help me, I need Help` , `Help Desk`
-

Examples:

- `find book`
- `find money`

2.10. Deleting a task: `delete`

Deletes an existing task in the list

Format: `delete [index]`

- Deletes the task at the specified `[index]`.
- The index refers to the index number shown in the displayed task list.
- The index must be a positive integer 1, 2, 3.

Examples:

- `delete 2`
  The second task in the list will be removed.
- `delete 5`
  The fifth task on the list will be removed.

2.11. Change the type or the date of an existing task: `reschedule`

Changes the type and/or the date of an existing task. When the runs the command, Duke will delete the original task and append the new one in the list.

Format: `reschedule new_task_type [index_of_the_old_task] new_date`

- Change the type and/or the date of an old task.
- The index refers to the index number shown in the displayed task list.
- The index must be a positive integer 1, 2, 3.

Examples:

- `reschedule event 1 /at 12/9/2019 1200 /to 12/9/2019 1400`
  Change the first task in the list as an event with the new time slot.
- `Reschedule todo 4`
  Change the fourth task in the list to be a todo task.

2.12. Get Reminders for the : `reminders today`

Displays the reminders for the current day

Format: `reminders today`

- Displays today's deadline, event and period tasks

Examples:

- `Reminders today`

2.13. Get upcoming reminders: `reminders upcoming`

Displays all upcoming reminders with respect to the current day

Format: `reminders upcoming`

- Displays upcoming deadline, event and period tasks

Examples:

- `reminders upcoming`

2.14. Get past reminders: `reminders past`

Displays the past reminders with respect to the current day

Format: `reminders past`

- Displays past deadline, event and period tasks

Examples:

- `reminders past`

2.15. View Schedule for a specific day: `schedule`

Displays the schedule for the specified day

Format: `schedule d/mm/yyyy`

- Displays deadline, event and period tasks for the specified day

Examples:

- `Schedule 14/3/2019`

2.16. Find a free time slot based on a given specific time: `free-time`

Displays the nearest free time slot based on the given specific time and the length of the desired duration.

Format: `free-time [the-length-of-the-duration] hrs /on d/mm/yyyy HHmm`

- The length of the duration is counted by hours

Examples:

- `free-time 4 hrs /on 6/6/2011 1300`

2.17. Exiting the program: `bye`

Exits the program.

Format: `bye`

2.18. **Saving the file**

Duke Tasklist data is saved in the hard disk automatically after any command that changes the data.

There is no need to save manually.

3. Features [coming in v1.4]

Below are future implementations that we Plan to release in version 1.4.

3.1. Add income:`income` [Basic Tracking]

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

`income view all`

- Views all income sources listed

`income delete [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` [Basic Tracking]

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] /at d/mm/yyyy`

- Adds an expenditure to total expenditure list

`expenditure view all`

- Views all past expenditures

`expenditure delete [index]`

- Deletes selected expenditure from expenditure list by index

Examples:

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

Extension Type: [Basic Tracking]

3.3. Detailed financial report for specific month: `finance`

Shows a detailed financial report for a specific month.

Format: `finance /for [month]`

- `[month]` should be ranged from 1 to 12

Examples:

- `finance /for 10` Returns a financial report of October.
- `finance /for 5` Returns a financial report of May.

Extension Type: [C-Statistics]

3.4. Table of expenditure categories: `category` [C-Statistics]

Shows a table of the expenditure of the respective categories.

Format: `category /for [month]`

- [month] should be ranged from 1 to 12

Examples:

- `Histogram /for 5` Returns a histogram chart of expenditure and income for May.

Extension Type: [C-Statistics]

3.5. 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]

3.6. 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]

3.7. 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]

3.8. 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]
  ```

3.9. Timing Shortcuts: `now ytd`

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]

3.10. Undo**:** `undo`

The user is able to undo the latest command they implemented.

Format: `undo`

- Undos commands adding or deleting to the user's account database

Examples:

- `undo`

Extension Type**:**[C- Undo]

3.11. Search for items: `find`

The user is able to search for items under the respective categories they are saved in as well as searching for a particular field of an item such as the description or month field.

Format: `find [keyword] /type [type of item] /field [desc]`

If /:type and /field are empty it will just match the keyword to all types and fields.

Examples:

- `Find sushi /type food /field desc`

Extension Type: [C- BetterSearch]

3.12. Short term goal setting: `goal-short`

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

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

Examples:

- `goal-short buy HDB /amt 100000 /by 9/12/2030 /priority high`
- `goal-short buy Lambo /amt 100000 /by 10/12/2040 /priority medium`
- `goal-short buy Boat /amt 100000 /by 11/12/2050 /priority low`

Extension Type: [D-Goals]

3.13. Check goal progress: `check goals`

Checks for the progress the user has made towards their set goals e.g percentage to completion

Format: `check goals`

Extension Type: [D-Goals]

3.14. 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]

3.15. Loan Tracking (For Small Loans): `lent borrowed settled`

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

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]`
- `borrowed [other party] /amt [cost]`
- `settled [index]`

Examples:

- `lent Max Chan /amt 200`
- `borrowed Daniel Chan /amt 100`
- `settled 6`

Extension Type: [D-Loans]

3.16. Loan Tracking (For Large Loans): `lent borrowed`

Allows the user to keep track of debts with interest from/to other parties e.g organisations/banks.

Interest can be indicated as simple or compounded

Format:

- `lent [other party] /amt [cost] /interest [percentage/simple increment] /per [month/year]`
- `borrowed [other party] /amt [cost] /interest [percentage/simple increment] /per [month/year]`

Examples:

- `borrowed DBS /amt 10000 /interest 5% /per month`
- `lent Chan Bro's Pte Ltd /amt 50000 /interest 500 /per month`

Extension Type: [D-Loans]

3.17. Shared Expenditures: `spentSplit`

Allows the user expenditures meant to be split among others

Individual debts settled will automatically be added to income

Format:

- `spentSplit [description] /amt [amount in dollars] /parties [other party] and [other party] and … /at d/mm/yyyy`
- `spentSplit settled [index] [index of party]`

Examples:

- `spentSplit PGP Mala /amt 40 /parties Sean Chan and Abhijit and Max Chan /at 27/09/2019`
- `spentSplit 8 2`

Extension Type: [D-Loans]

3.18. Instalments Tracking: `instalment`

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

Format:

- `instalment [desc] /amt [cost] /until [d/M/yyyy]`

Examples:

- `instalment mortgage /amt 100000 /until 31/12/2050`

Extension Type: [D-instalments]

3.19. Check Instalments: `check instalments`

Allows the user to check all of the uncompleted instalments that they have to settle via recurring monthly instalments.

Format: `check instalments or check instalments [d/M/yyyy]`

- `The output will list all the uncompleted payments with the individual amounts.`
- `If the user append a date after the command, it will provide the list and the amount needs to be paid in that month.`

Examples:

- `Check instalments`
- `[output]`

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

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

- `check instalments 16/9/2020`
- `[output]`

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

  `Instalments at 16/9/2020: 200`

Extension Type: [D-instalments]

3.20. 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.

Extension Type: [D-instalments]

3.21. Recurring Bills: `bills`

Allows the user to keep track of the recurring payments that they have to periodically process.

Format: `bills [desc] /amt [cost] /every [month]`

- `[month] refers to the frequency of billing.`

Examples:

- `Electricity bill /amt 2000 /every 1`
- `Road Tax /amt 4000 /every 4`

Extension Type: [D-Bills]

3.22. Reminder to Pay Bills

Program will remind the user a week in advance about the outstanding bills that they have to pay at the start of a session (if any).

Examples:

- `Hello ____! You have outstanding bill(s):`
    - `Road Tax /amt 4000 /every 4`

Extension Type: [D-Bills]

3.23. Create A Bank Account Tracker

Allows the user to keep track of the places where they spend their money.

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`
- `[UI line1] Got it. New bank account tracker has been recorded.`
- `[UI line2]  Account: OCBC, Balance: 0, Initial Date: 27/7/2017 Rate: 0.005`
- `[input] Bank-account DBS /amt 100 /at 14/8/2018 /rate 0`
- `[UI line1] Got it. New bank account tracker has been recorded.`
- `[UI line2]  Account: DBS, Balance: 100, Initial Date: 14/8/2017 Rate: 0`

Extension Type: [D-Bank]

3.24. Income and Expenditure for specific bank account**:** `income spent`

Allows the user to add income or deduct money from the account directly.

Format:

- `income [description] /amt [amount in dollars] /at d/mm/yyyy /to [account_description]`
- `spent [description] /amt [amount in dollars] /at d/mm/yyyy /from [account_description]`
- Just append “/to [description]” or “/from [description]” in the standard income/expenditure command. Without appending this, the transaction will be considered as a cash transaction.

Examples:

- `income intern salary /amt 1000 /at 2/12/2019 /to OCBC`
- `spent dinner /amt 10 /at 9/29/2019 /from DBS`

Extension Type: [D-Bank]

3.25. Internal Transfer: `deposit withdraw Transfer`

Allows the user to keep track of deposit money, withdraw money or the internal transfer between bank accounts

Format:

- `deposit [amount] [account_description] /at [date]`
- `withdraw [amount] [account_description] /at [date]`
- `Transfer [amount] /from [account_description] /to [account_description] /at [date]`

Examples:

- `deposit 200 OCBC /at 28/9/2019`
- `withdraw 50 DBS /at 25/9/2019`
- `Transfer 1000 /from DBS /to OCBC /at 1/10/2019`

Extension Type: [D-Bank]

3.26. 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]

4. Features [coming in 2.0]

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

4.1. Check Goal savings progress: `check goals savings`

Checks for the progress the user has made towards their set goals e.g savings required per month vs actual savings per month.

Format: `check goals savings`

Extension Type: [D-Goals]

4.2. 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]

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]