1. Personal information

Name: Dat Le

Student number: 837639

Study program: Master's Programme in Automation and Electrical Engineering

Study year: 2022

Date: 25/2/2022

2. General description

In this course, I will build an application helps managing money by capturing and grouping the transaction's data from the recorded file of different banks.

The program will support the recorded files from following banks:

- S bank
- Saastospankki

In addition, the application only supports input file in **.csv** format

The application will support following features:

- Browse input file: Users able to browse the file on their filesystem
- Load data**:**
  - Able to recognize the extension of file and verify whether it is supported or not
  - Able to read input file in **.csv** format
- Parse data:
  - Parse given data files into individual transactions
  - Parse the individual transaction to capture needed information
    - Name of receiver
    - Amount of money
    - Date
  - Save them to an object of Storage class
- Calculate spending:
  - Calculate the spending spent at a specific store/retailer
  - If multiple store/retailers are grouped together, able to calculate the total spending
- Grouping function:
  - Group spending store into group with user-defined name
  - Add new group if needed
  - Delete the group if needed
- Display data: *Pie charts* are used
  - Usage: Display the spending spent as individual store/retailer or user-defined group. Only one mode works at a time.

- Grouping function is disabled (default): Display the spending spent at each store/retailer
- Grouping function is enabled: Display the spending spent of different user-defined groups.

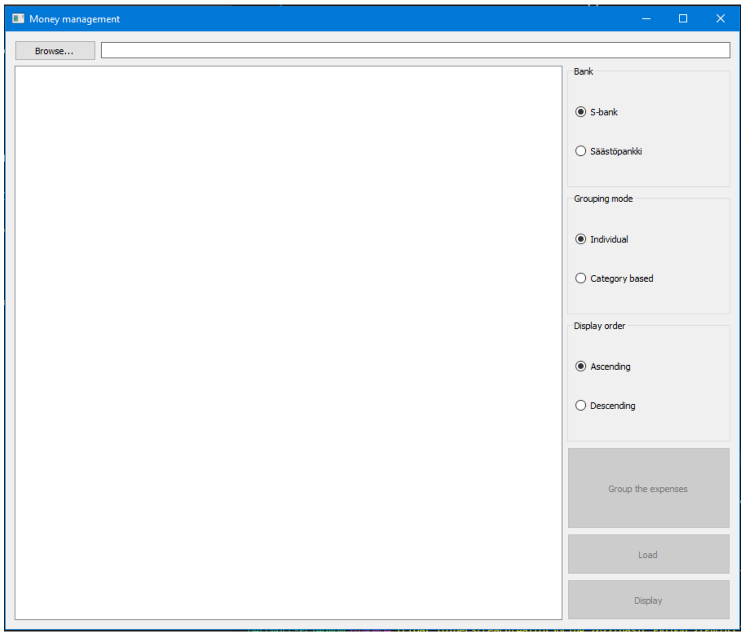
3. Instruction for the user

As mentioned in the README.md, user only need to use command **python main.py** to launch the application (after install the necessary packages)

The following steps illustrate the instructions for using this application:

Open the application's GUI

The following GUI appears after users use the command



Bank: [✓ S-bank] Säästöpankki

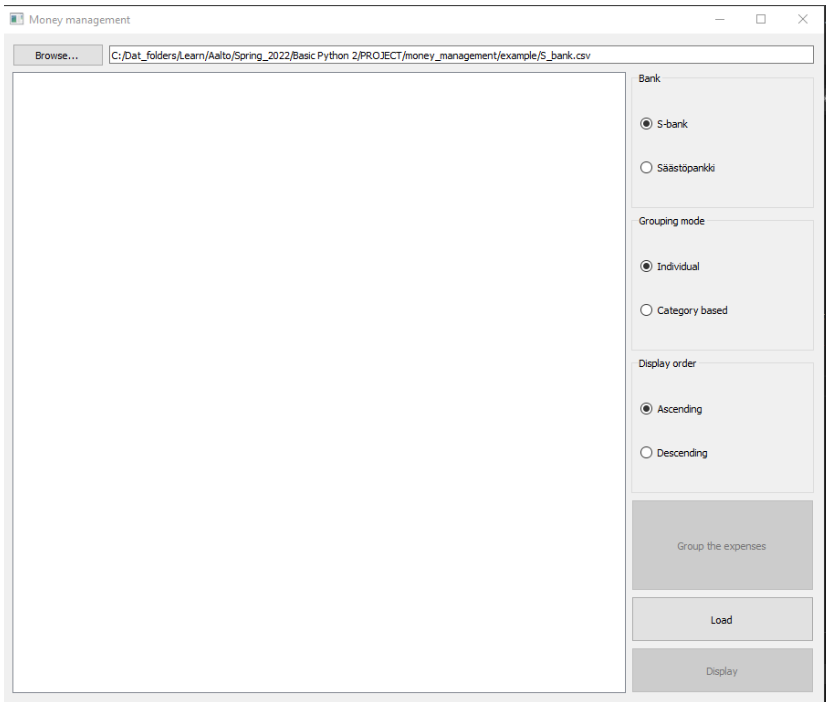
Grouping mode: [✓ Individual] Category based

Display order: [✓ Ascending] Descending

Browse input file

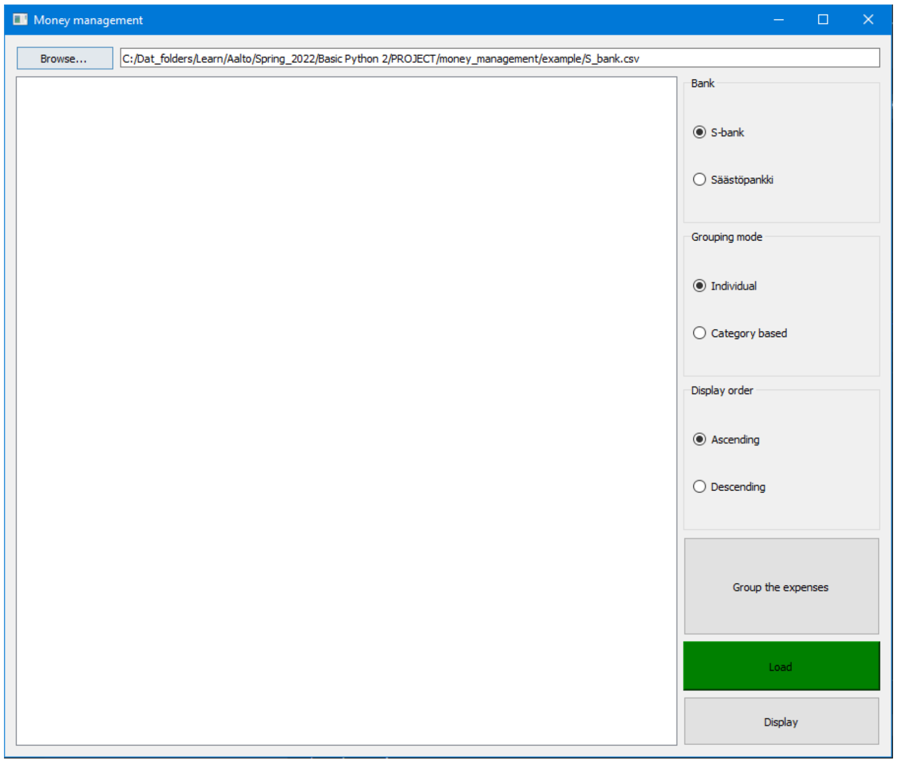
User needs to browse the transaction file as the input file for the application with button **Browse…**. The files in folder **example** in repository can be used as the example for accepted input file. Make sure that the input file matches with the chosen bank (**S bank** is chosen by default).

The path to selected file is displayed in the box on the right side of the **Browse** button.



Load data

Press the **Load** button to load input data and ready to display data in pie chart and texts. If the parsing process on the selected file does not have any error, the **Load** button turns to green color.

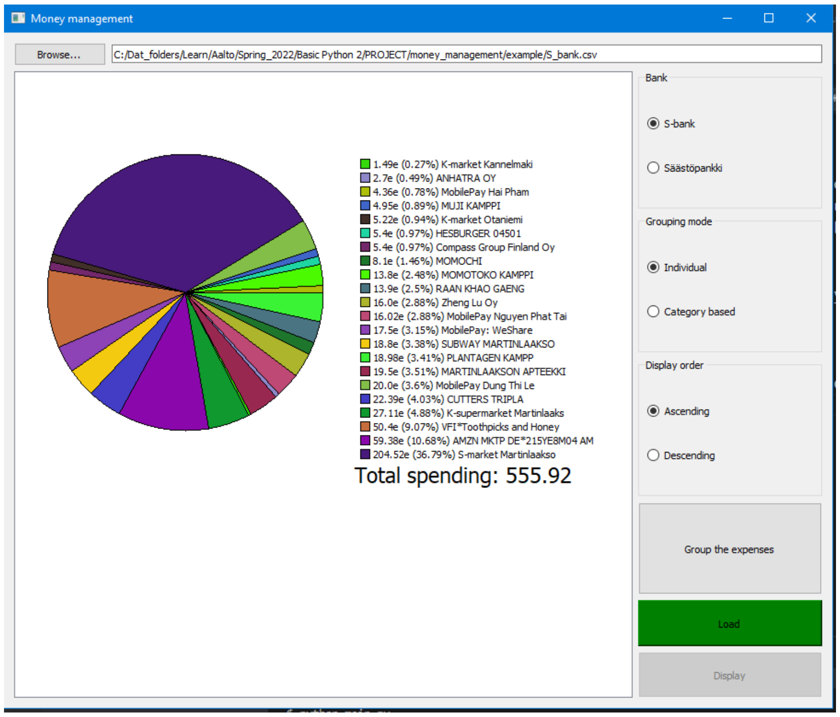


Display data

Press the **Display** button to generate pie chart and texts. The application provides the features that allow user to choose the display order of the data (in ascending or descending order) – **Display order** - and display the data of individual expenses or categories (group of expenses) – **Grouping mode**.

The default selection of the **Display order** is ascending order, while the **Grouping mode** is displaying data of individual expenses.

Displayed data is the individual expenses:



C:/Dat_folders/Learn/Aalto/Spring_2022/Basic Python 2/PROJECT/money_management/example/S_bank.csv

Bank: [✓ S-bank] Säästöpankki

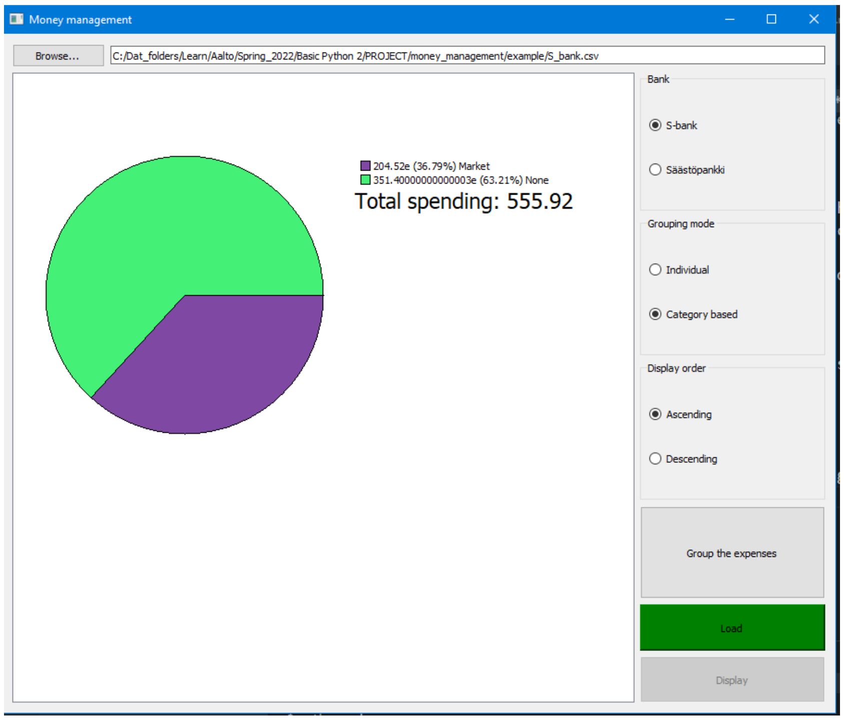
Grouping mode: [✓ Individual] Category based

Display order: [✓ Ascending] Descending

🟩 1.49e (0.27%) K-market Kannelmaki
🟪 2.7e (0.49%) ANHATRA OY
🟨 4.36e (0.78%) MobilePay Hai Pham
🟦 4.95e (0.89%) MUJI KAMPPI
🟫 5.22e (0.94%) K-market Otaniemi
🟩 5.4e (0.97%) HESBURGER 04501
🟪 5.4e (0.97%) Compass Group Finland Oy
🟩 8.1e (1.46%) MOMOCHI
🟩 13.8e (2.48%) MOMOTOKO KAMPPI
⬛ 13.9e (2.5%) RAAN KHAO GAENG
🟨 16.0e (2.88%) Zheng Lu Oy
🟥 16.02e (2.88%) MobilePay Nguyen Phat Tai
🟪 17.5e (3.15%) MobilePay: WeShare
🟨 18.8e (3.38%) SUBWAY MARTINLAAKSO
🟩 18.98e (3.41%) PLANTAGEN KAMPP
🟥 19.5e (3.51%) MARTINLAAKSON APTEEKKI
🟩 20.0e (3.6%) MobilePay Dung Thi Le
🟦 22.39e (4.03%) CUTTERS TRIPLA
🟩 27.11e (4.88%) K-supermarket Martinlaaks
🟧 50.4e (9.07%) VFI*Toothpicks and Honey
🟪 59.38e (10.68%) AMZN MKTP DE*215YE8M04 AM
🟪 204.52e (36.79%) S-market Martinlaakso

Total spending: 555.92

Display data is based on the categories:



C:/Dat_folders/Learn/Aalto/Spring_2022/Basic Python 2/PROJECT/money_management/example/S_bank.csv

🟪 204.52e (36.79%) Market
🟩 351.40000000000003e (63.21%) None
Total spending: 555.92

Bank: [✓ S-bank] Säästöpankki

Grouping mode: Individual [✓ Category based]

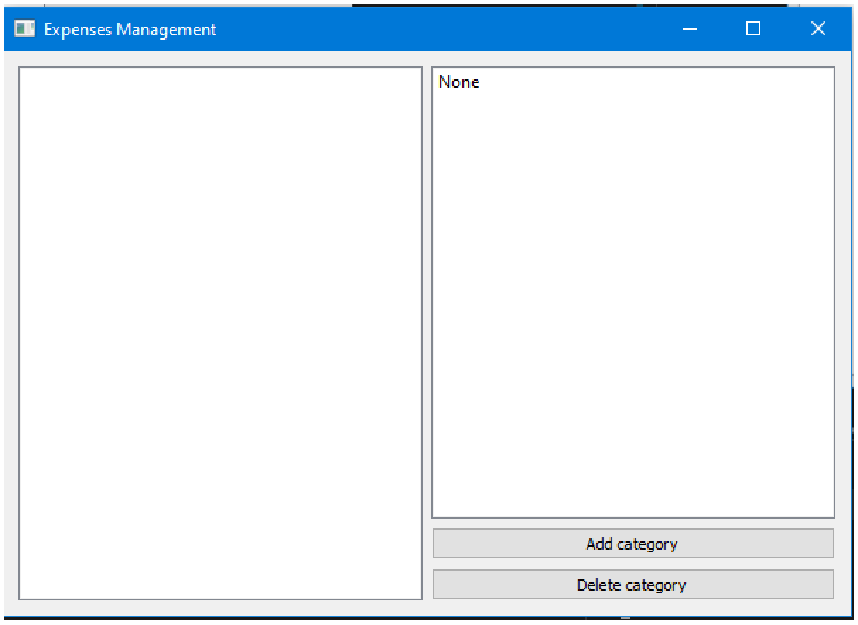
Display order: [✓ Ascending] Descending

Grouping the expenses

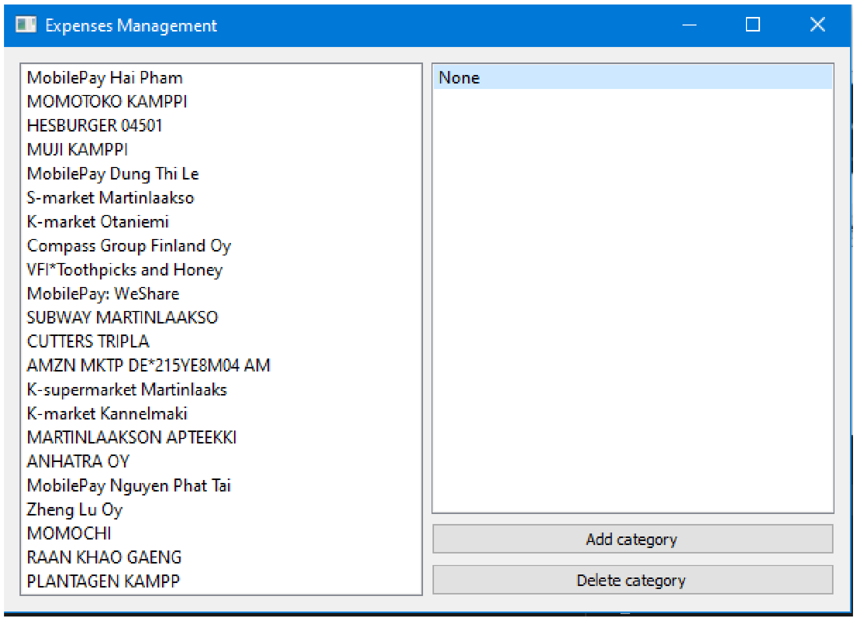
If user want to group the expenses, press the button **Group the expenses**. A new window will appear. The screen on the left side displays the expenses, while the screen on right side displays the available categories. This window provides the functions allow user to add, delete category and modify the category of the expenses.

The default category for all expenses is **None**, which indicates these expenses do not belong to any category.

Please do not delete this category, otherwise the application will crash.

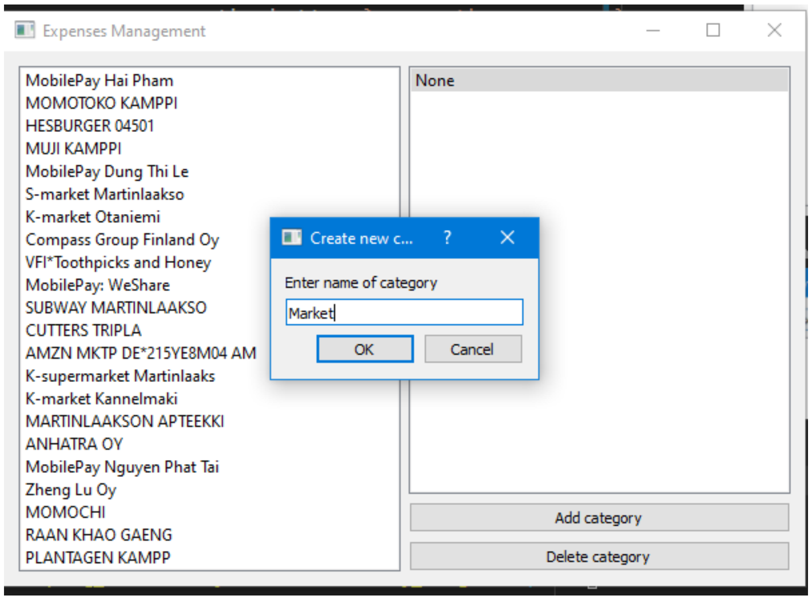


By pressing the category **None**, user can see the expenses belong to this category. The same mechanism is applied on another categories (when user creates a new one with **Add category** button).

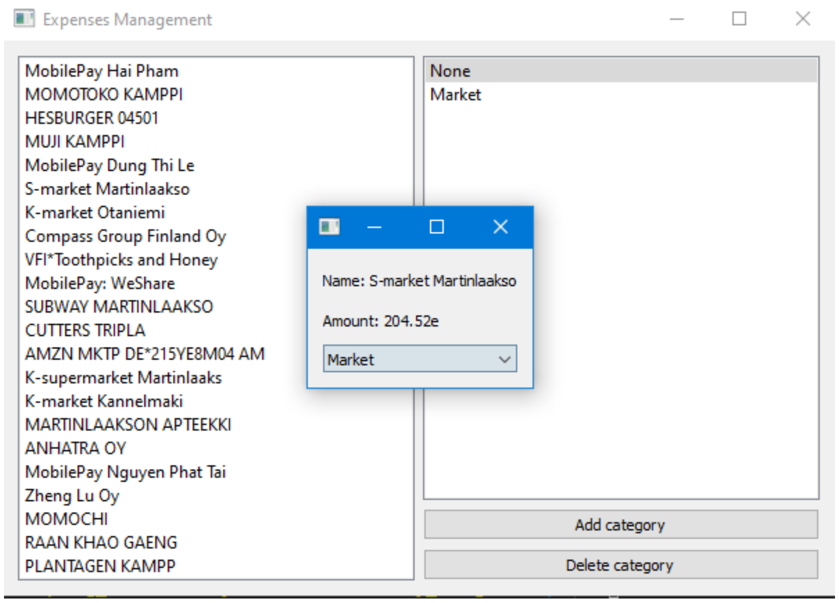


MobilePay Hai Pham
MOMOTOKO KAMPPI
HESBURGER 04501
MUJI KAMPPI
MobilePay Dung Thi Le
S-market Martinlaakso
K-market Otaniemi
Compass Group Finland Oy
VFI*Toothpicks and Honey
MobilePay: WeShare
SUBWAY MARTINLAAKSO
CUTTERS TRIPLA
AMZN MKTP DE*215YE8M04 AM
K-supermarket Martinlaaks
K-market Kannelmaki
MARTINLAAKSON APTEEKKI
ANHATRA OY
MobilePay Nguyen Phat Tai
Zheng Lu Oy
MOMOCHI
RAAN KHAO GAENG
PLANTAGEN KAMPP

By pressing **Add category**, user can create new category for grouping (after pressing **OK** button on pop up window)

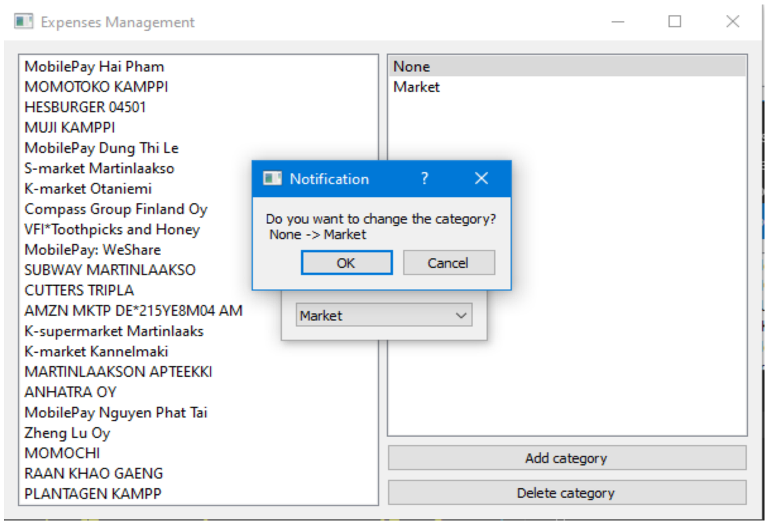


To add expense to this new category, double click the chosen expense on the left-side screen. A new window will pop up, which contains the basic information of the expenses. By adjusting the value of the drop-down menu in this window, user can change the category of the chosen expense (from *None* to *Market* in this case).



To confirm the selection, close the window and press **OK** on new pop-up window. After assigning, the chosen expense will disappear from the **None** (user can verify that by double clicking the **None** on the right side).

In addition, the selected expense appear on the **Market** when double clicking to this category.



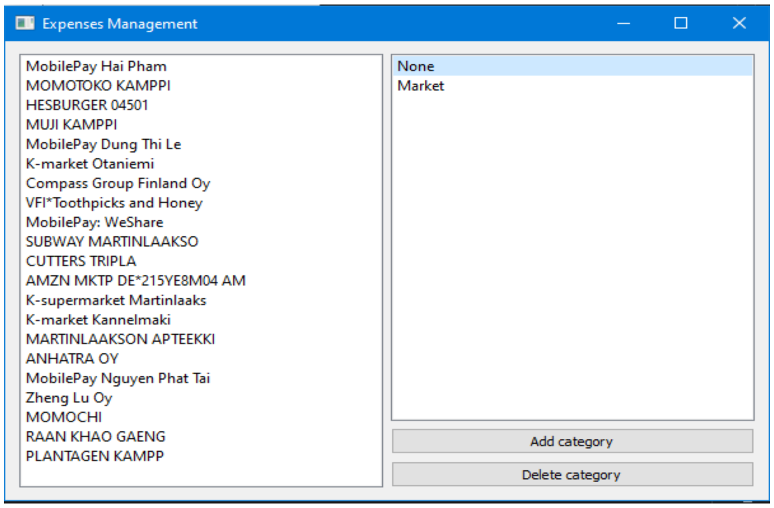
Expenses Management

MobilePay Hai Pham
MOMOTOKO KAMPPI
HESBURGER 04501
MUJI KAMPPI
MobilePay Dung Thi Le
S-market Martinlaakso
K-market Otaniemi
Compass Group Finland Oy
VFI*Toothpicks and Honey
MobilePay: WeShare
SUBWAY MARTINLAAKSO
CUTTERS TRIPLA
AMZN MKTP DE*215YE8M04 AM
K-supermarket Martinlaaks
K-market Kannelmaki
MARTINLAAKSON APTEEKKI
ANHATRA OY
MobilePay Nguyen Phat Tai
Zheng Lu Oy
MOMOCHI
RAAN KHAO GAENG
PLANTAGEN KAMPP

None
Market

Notification

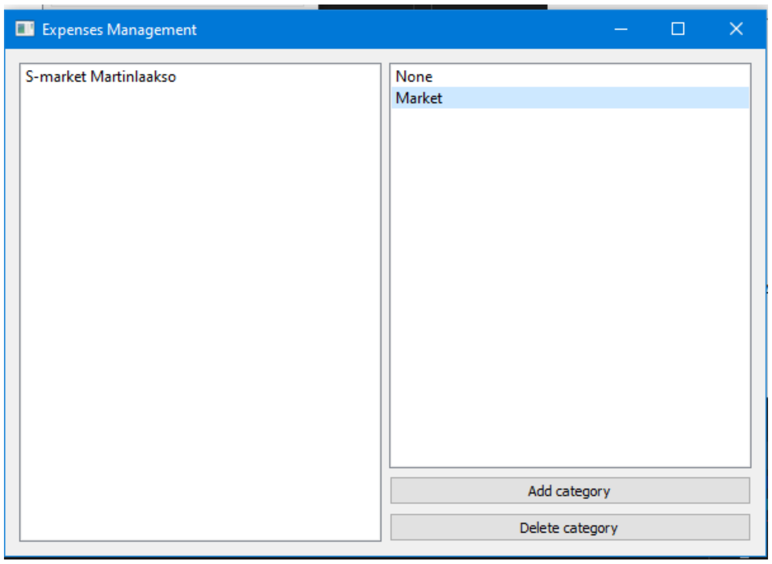
Do you want to change the category?
None -> Market



Expenses Management

MobilePay Hai Pham
MOMOTOKO KAMPPI
HESBURGER 04501
MUJI KAMPPI
MobilePay Dung Thi Le
K-market Otaniemi
Compass Group Finland Oy
VFI*Toothpicks and Honey
MobilePay: WeShare
SUBWAY MARTINLAAKSO
CUTTERS TRIPLA
AMZN MKTP DE*215YE8M04 AM
K-supermarket Martinlaaks
K-market Kannelmaki
MARTINLAAKSON APTEEKKI
ANHATRA OY
MobilePay Nguyen Phat Tai
Zheng Lu Oy
MOMOCHI
RAAN KHAO GAENG
PLANTAGEN KAMPP

None
Market

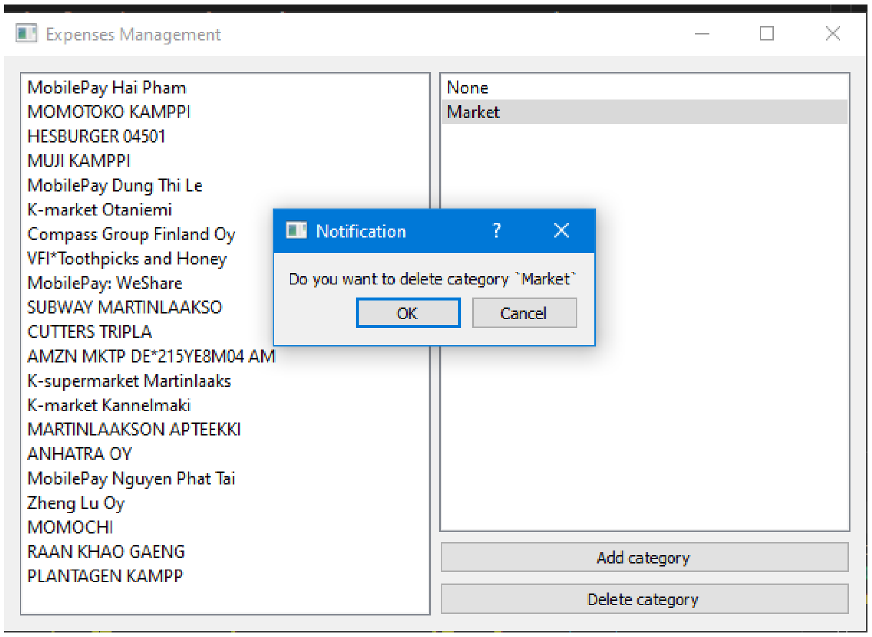


Expenses Management

S-market Martinlaakso

None
Market

To remove the category, select the desired category by double clicking it. After that, press the **Delete category** button. The expenses belong to the deleted category will be assigned back to **None** (at the bottom).

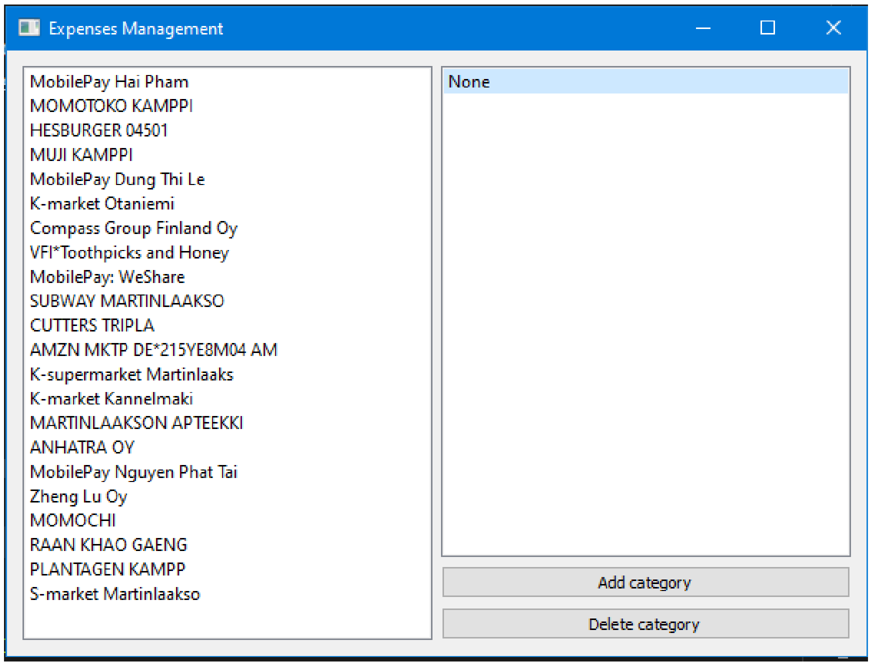

Expenses Management

MobilePay Hai Pham
MOMOTOKO KAMPPI
HESBURGER 04501
MUJI KAMPPI
MobilePay Dung Thi Le
K-market Otaniemi
Compass Group Finland Oy
VFI*Toothpicks and Honey
MobilePay: WeShare
SUBWAY MARTINLAAKSO
CUTTERS TRIPLA
AMZN MKTP DE*215YE8M04 AM
K-supermarket Martinlaaks
K-market Kannelmaki
MARTINLAAKSON APTEEKKI
ANHATRA OY
MobilePay Nguyen Phat Tai
Zheng Lu Oy
MOMOCHI
RAAN KHAO GAENG
PLANTAGEN KAMPP

None
Market

Notification

Do you want to delete category `Market`

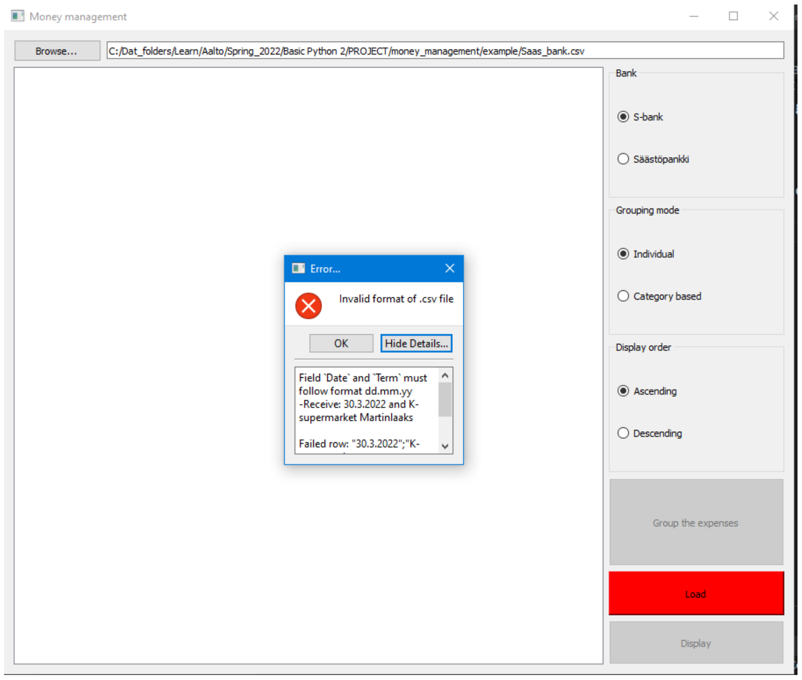

Expenses Management

MobilePay Hai Pham
MOMOTOKO KAMPPI
HESBURGER 04501
MUJI KAMPPI
MobilePay Dung Thi Le
K-market Otaniemi
Compass Group Finland Oy
VFI*Toothpicks and Honey
MobilePay: WeShare
SUBWAY MARTINLAAKSO
CUTTERS TRIPLA
AMZN MKTP DE*215YE8M04 AM
K-supermarket Martinlaaks
K-market Kannelmaki
MARTINLAAKSON APTEEKKI
ANHATRA OY
MobilePay Nguyen Phat Tai
Zheng Lu Oy
MOMOCHI
RAAN KHAO GAENG
PLANTAGEN KAMPP
S-market Martinlaakso

None

Error handling

User will encounter the following error message when loading the input file which is not supported by the chosen bank (i.e loading .csv file of Saastospankki but choosing S bank on the GUI).



C:/Dat_folders/Learn/Aalto/Spring_2022/Basic Python 2/PROJECT/money_management/example/Saas_bank.csv

Invalid format of .csv file

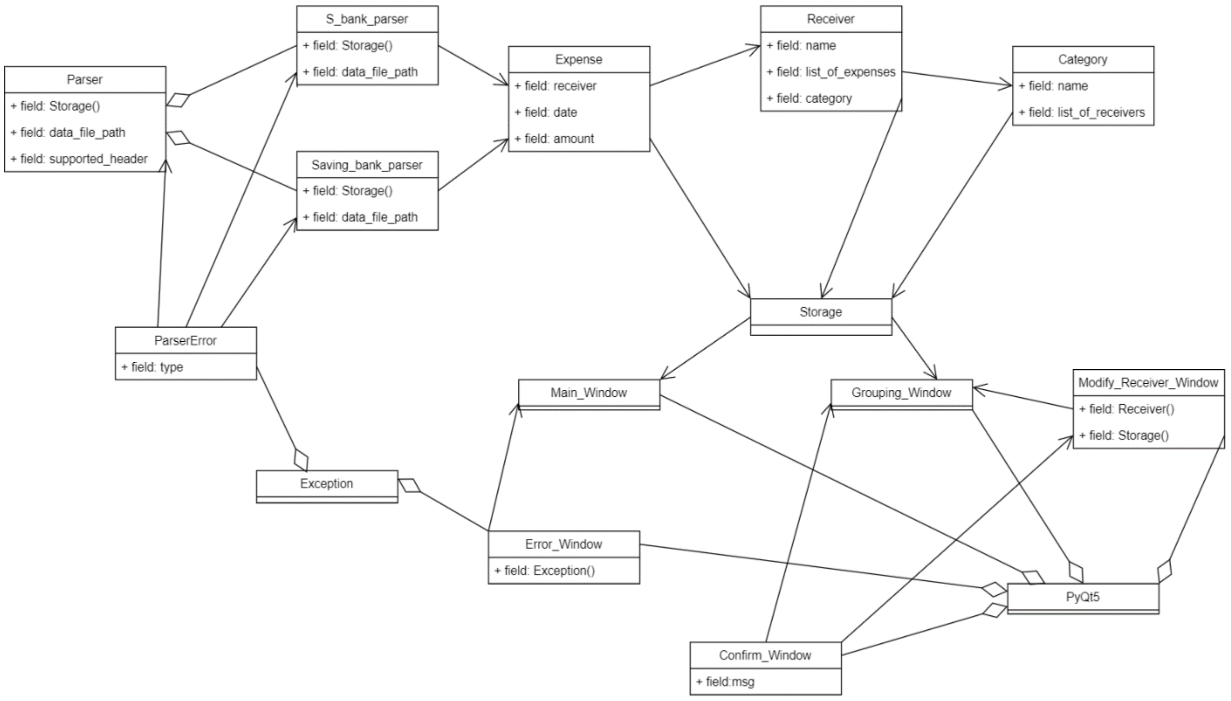
```
Field `Date` and `Term` must
follow format dd.mm.yy
-Receive: 30.3.2022 and K-
supermarket Martinlaaks

Failed row: "30.3.2022";"K-
```

4. External libraries

The source code uses external library **parameterized** in writing unit tests for the application.

5. Structure of the program



The application works around the GUI. It is built with classes that inherit from PyQt5 package. The Main_Window() class constructs the main window of the GUI, while class Grouping_Window() formulates the second widow used for grouping expenses.

In addition, the class Modify_Receiver_Window() class is created to help modifying the category of the expense. The Confirm_Window() class constructs a window that asks for user confirmation on their action (change the category or delete a category).

The GUI gets data from the Storage() class, which contains all the Expense(), Receiver() and Category() objects.

These objects are created when the input file (in .csv format) is parsed with suitable parser class (either S_bank_parser() or Saving_bank_parser()). These parser classes inherit from the class Parser().

For error handling, the class ParserError() and Error_Window() were created. They inherit from class Exception(). While ParserError() class just throws an exception with defined message, the Error_Window() class defines a window that pops up when users make errors in choosing input file.

6. Algorithms

For calculating the contribution of the chosen object (individual expense or category) on the total spending, I calculated the spending on this chosen object and divide it with the total spending (the result is rounded up to 2 decimals).

For drawing the pie chart, I calculated the equivalent span angle for each pie according to its contribution in total spending and draw the first pie at start degree 0. The next pie uses the start degree equals to the sum of previous start degree and the previous pie’s span angle.

7. Data structures

The data type “list” is used for storing parsed data from input files. I chose this data structure since it is easy to query the stored data and modify if needed.

8. Files

The application only supports the file in **.csv** format and the structure of each **.csv** file depends on the chosen bank. The files in folder /example in the source code can be used as the example for accepted .csv file for each bank.

In the S_bank_parser() and Saving_bank_parser() classes, I explained what are the required headers of each .csv file and what are the expected data for certain headers (the application does not need data from all fields in .csv file).

9. Testing

I used unittest (Python internal library) and parameterized (external library) for writing the application’s unit tests. Due to the limit in time, I could not test all classes of the application, especially the classes which inherit from the PyQt5 package.

According to the structure of the source code, all classes and their related functions in the folder **src/app** are tested.

10. The known shortcomings and flaws in the program

I would implement the mechanism to prevent user from delete the category **None** in the **Group the expenses** function. This category is the based category of all expenses, which indicates the expenses do not belong to any category. The application is not designed to handle the delete action on this category.

In addition, the scene on the main window of GUI is not designed to be scrollable. In another words, user cannot see the text data if the number of expenses exceeds the length of the scene.

Finally, the application does not support the usage of database. This shortcoming prevents any mechanisms for generating categories automatically (based on the previous expenses in the category) and it requires users to generate new categories every time the application is launched.

11. 3 best and 3 worst areas

From my point of view, my application fulfills all the requirements in the medium level. You can choose the suitable parser, load the .csv files and generate the pie chart on the fly. In addition, the GUI is clear and easy to use for new users. Finally, I also created a simple error handling system by changing the button color and generating the pop-up windows when users choose incorrect parser.

On the other hand, one of the worst areas is the pie chart. The pie chart is not generated from the ORy line (O is the center point of the circle, Ry is the radius on the Oy axis) but randomly. It causes confusion to users in tracking the contributions of each object (individual expense or category) to total spending. In addition, the scene that displays the pie chart and text is not designed to be scrollable. It causes problem when user has more texted objects to display than the height of the scene (some objects miss from the scene and user cannot see them).

12. Changes to the original plan

The biggest modification, compared with my original plan, is that the application only supports file with .csv format. The number of supported banks also reduces from four to two (S bank and Saastospankki).

In addition, the main window of GUI has modifications when I added two new options for users to choose the parser by themselves and choose the displayed order of the spending data (in ascending or descending order).

Finally, I gave up on the usage of SQL database and create a storage class instead. This storage works similar with the SQL database, but it starts from scratch when users launch the application.

13. Realized order and scheduled

28/3/2022: I started the project from scratch

8/4/2022: I submitted the version 1.0 of the application to git. This version used SQL database as the option for storing input data. However, this approach was not allowed, and I needed to find another solution.

9/5/2022: I submitted prototype version 1.0 to git. This version had almost completed GUI and it used python class as the storage for input data.

11/5/2022: The “grouping expenses” feature was completed.

13/5/2022: I finalized the documentation and unit tests for a part of the application.

14. Assessment of the final result

In my opinion, my application satisfies all the requirements in the medium level and small part of the hard level (the application is able to load new files and append data to storage). Although there are few shortcomings, which are explained in the section 10, I believe that I can remove all of them if I did not have to change the approach for storing data (from SQL database to Python class). However, the application is designed as modules, and it is easy for future modifications.

In addition, the application needs the unit tests on GUI (I was only able to test part of the source code).

15. References

The main source that I used mainly in designing the GUI is official document of Qt for Python (https://doc.qt.io/qtforpython-5/index.html#)

In addition, I used the suggestions from the Stackoverflow in designing the working mechanism of the application.

16. Attachments

Check the section “3. Instructions for the user” for more information about the GUI and the instructions for users.