Ateneo de Manila University
Department of Quantitative Methods and Information Technology

Project Concept: Statement of Cash Position Generator

We decided to make a program that automatically writes a statement of cash position after you input different business activities into a form, including if it is inflow, outflow, or an operating, financing, or investing activity. The Statement of Cash Position is a daily report of the cash activities on a particular day. It is different from a cash flow such that a minimum accounting period of a cash flow is one month.

Coding the Project Backend

First, we made the code for the backend that includes the functions that connect to MongoDB and the function that separates the numeric values of inflows and outflows, as well as categorizes them into operating, investing, and financing. This function also adds and subtracts values according to the different categories mentioned to get net outflow/inflow for the Operating, Investing, and Financing categories. There were also functions for the deletion/insertion of different entries of cash activity type in MongoDB, and it was just a matter of connecting these to the Graphical User Interface of the app.

Making the GUI

We made a separate python file to house the GUI of the forms needed to insert the data of the different entries into the database. The basic structure of the form is that it will ask for your cash in the bank the previous day, the date of your activity, the name, its outflow or inflow category, then its activity type may it be operating, investing, or financing. There is a function that every time you click to add an entry, a form comes out for you to add another entry. After you are done, the code appends your entries into a list of dictionaries every time you click “Save,” then when you click “Done,” that is the only time the list of dictionaries is appended into the MongoDB database.

After that, another file was made to display the statement of cash position so that the final report could be displayed. We used the functions from the backend to get the entries from the MongoDB database, and then we also used the compute values function from the project backend to get the needed values for the final report. We used the pyqt AddLabel function to make labels and format them in a sort of gridlike fashion (minus the columns) along with the values that were given for the different entries along with the sum of the operating inflow, operating outflow, investing inflow, investing outflow, etc. which was computed by the compute_values function. The net operating, investing, and financing expense displayed by the compute_values function was also made sure to be displayed, as well as the net increase in cash in the bank as well as the cash in the bank from the previous day that we had to input at the beginning of the cycle, as well as computed cash in bank today.

Finalizing the application

A homepage file for the app was made using pyqt on a separate python file, and then another main window file that contains all of the pages of the application, then the pages were arranged into stack layout so that all pages are accessible. The pages were initialized on the main window file by defining the object that will be put into the stack layout so that we can access the page, then when we set up the stack layout, that's what you assign into the main layout of the main window that contains all the pages. Then page switching is governed by a function in the main window that are accessed by the functions in the main window.

The different pages were converted into classes by the __init__ function so that the page will be easily instantiated from the main window, so that per class there is a specific function for the page, and that we have different formats per page. The .self helped by making the different components accessible as classes. The random variables that we put in every page were converted into class attributes. Functions and UI of the pages are put in the classes already. Files were also renamed to make them more descriptive. Functions for the database access were imported per page by converting them into classes as well.

Styling the returns page:

Financial statements are often, if not all the time, presented with underlining under the various numerical values. The underlines were then added by importing QFrame under the scp page which is a different python file. Similar to how the pages were converted into classes, the underlines were put using the __init__(self) function. Once the data is being inputted by the user to generate the Statement of Cash Position on the returns page, we can now see the underlining to separate the Inflow and Outflow, as well as the various categories (Operating, Investing, or Financing). This is a certain indication that a specific procedure in the cash position was completed. Furthermore, a Peso sign was also added at the very end to specify the currency of the transaction and highlight the final cash in bank value.

The alignment of certain texts and underlining were also improved by importing Qt and using the Qt alignment functions. The "Statement of Cash Position" title and the date were further aligned to the center, the amounts were aligned to their right positions, and the horizontal underlinings were adjusted to its bottom alignment. These various stylings were done to follow the usual or proper format of Cash Flow Statements found in the reports of various industries or business establishments.