PROJECT DOCUMENT - TIME MANAGEMENT

1. Personal Information

- Title: Time Management
- Name: Nguyen Duc Tam
- Student Number: 906968
- Study Program: Digital Systems and Design
- Study Year and Date: 2021-2024

2. General description and difficulty level

2.1. General Description

Time Management is a program that allows the users to measure the time of their activities (same mechanism with stopwatch). The program has an easy-to-use graphical user interface (GUI), and it also offers different functionality and flexibility for users' need and favor. All these actions below can be done in the GUI, which makes the program extremely easy to use for the users.

- Users can measure the time of up to 5 activities simultaneously.
- The basic operations include start/resume, stop/pause, reset, restart the timer of the activities.
- The users can also edit the activity name or delete the activity from the program.
- Besides, the user can choose the time expression they want, export data, move to next day and choose any day they want to measure the time.
- The unique features of the program are Statistics, Pomodoro and Help:
- Statistics can show the overview of the program data, some descriptive data and the visualization from the time measurement.
- Pomodoro allows the users to freely set their Pomodoro reminder, and sends the notification about working/resting based on Pomodoro technique.
- Help feature shows the information and guidance to the users.

2.2. Difficulty Level - Hard

The difficulty level of the project is **hard**. The final implemented project satisfies the following requirements:

- Graphical user interface
- Creation of an activity based on user input
- The amount of added activities can be maxed out at 5
- Taking time for activities
- The activity that is being timed should be distinct
- Multitasking, multiple courses can be timed at the same timer
- Picking a new activity to time interrupts and saves the previous activity’s timer

- Timer's time is shown on screen
- Switching courses/replacing can be done with the graphical interface
- Print time management data of current day
- Print time management data from previous days from a file
- Saving time management data into a file
- Unit Tests for at least part of the program

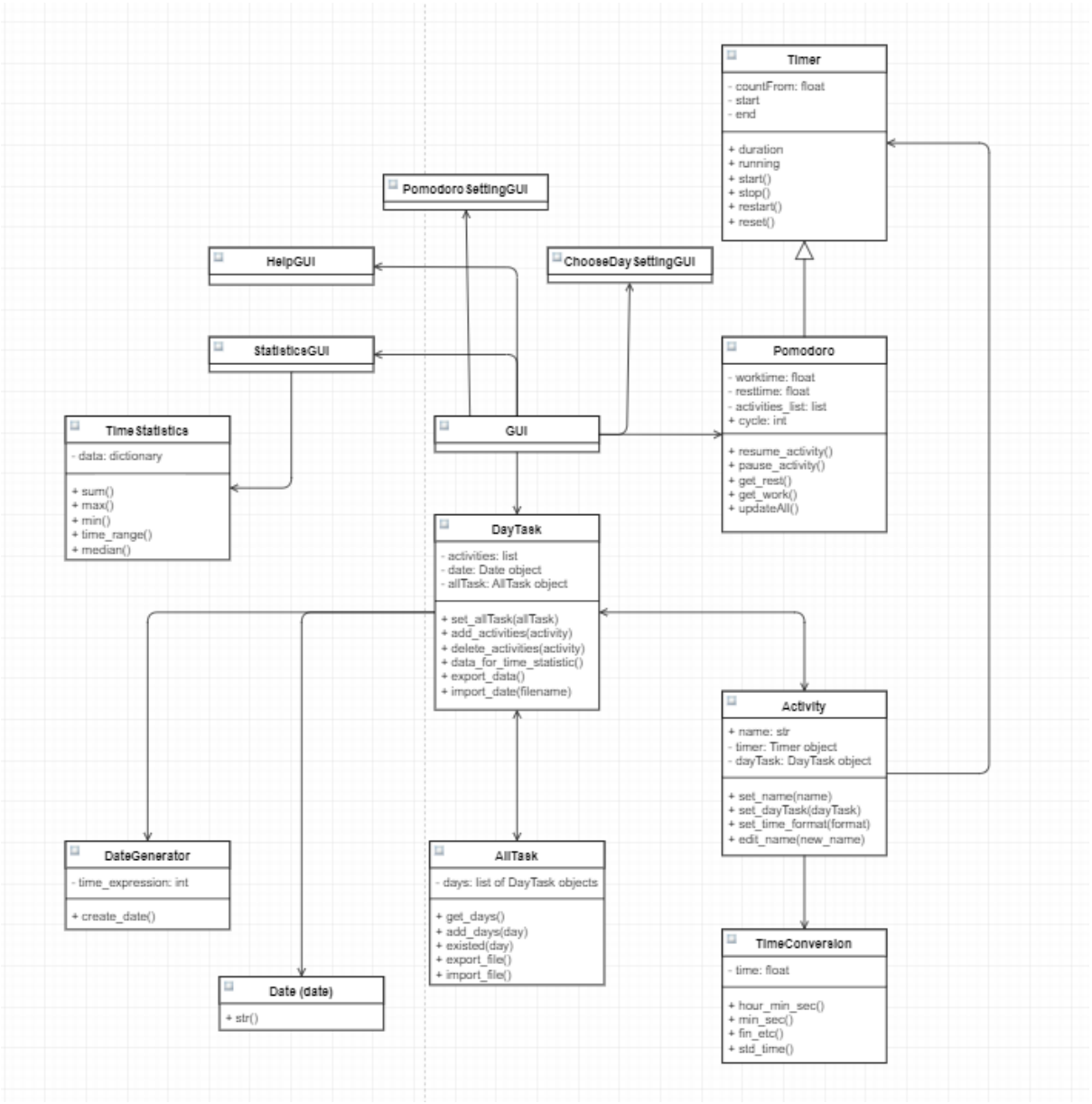
3. Instructions for the user

Instructions about installing and usage of the program have been documented carefully in the README.md file at the root of the project folder. Instructions can be found there.

4. External libraries

- The Python built-in libraries used in the project includes:
    - datetime
    - time
- External libraries used in the project are:
    - PyQt5 (used for creating the GUI of the program)
    - pyqtgraph (agreed with the TA, used for creating graphs for function Statistics of the program
    - numpy (This library is not allowed, and also not used in the project, but it is a required library for installing pyqtgraph library)
    - According to the dependencies management in the virtual environment of the project, the users need to import these packages besides original Python: numpy, PyQt5, PyQt5-Qt5, PyQt5-sip, PyQtChart, PyQtChart-Qt5, pyqtgraph. These dependencies are automatically downloaded along with PyQt5 and pyqtgraph libraries, so they are not external libraries that are not allowed.

5. Structure of the program



The program can be divided into 2 parts: frontend (what the users see in the GUI) and backend (how the program “brain” works behind the scene).

- The main class used for the frontend is class GUI. It creates the main GUI window of the program, where all the main information is shown for the users. Besides class GUI, there are also other classes that creates other part of the GUI:
    - Class ChooseDaySettingGUI: open a window for the users so that they can choose which day they want to record the activities time.
    - Class PomodoroSettingGUI: create a window for the users to change the setting of the Pomodoro Reminder feature of the program, and activate the reminder if they want to use it.
    - Class HelpGUI: create a window to show the instructions for the user of how to use the program.

    - Class StatisticsGUI: create a window to show the statistics of the activities timer, and show the visualization of the data using pie charts.
- The main class used for the backend is class DayTask. Each DayTask object directly connects to an instance of class GUI, and all the information shown in the GUI comes from this class. This class also connects to other class to build the program backend altogether:
    - Class AllTask: this class works like a central database of the program. It contains all the information of every day which the users use the program.
    - Class Activity: this class is equivalent to an activity that the user wants to measure the timer. It contains the name and the timer of the activity, as well as some method to manipulate the name and timer of that activity.
    - Class Timer: this class creates the timer for each activity, as well as for the Pomodoro reminder.
    - There are other classes like TimeStatistics, TimeConversion, DateGenerator, Date that are used for other features of the program. The details about their usage can be found in the corresponding .py file of each class.

6. Algorithms

- Basic converter in different time evaluation units (e.g., Mins - Hours - Days - Weeks - Credits).
- Some statistics math formula to sum up the timer data of the user in a day:
    - how many percentages a single task takes up
    - total time spent
    - average time per task
    - maximum time for a task in that day
    - minimum time for a task in that day
    - range from the minimum time to maximum time
    - median of the time of the tasks

7. Data Structures

- Python predefined “list”
- Python predefined “dictionary”

8. Files and file formats

- File will be used when the user wants to save the timer of the tasks.
- File will be loaded if the user wants to see the data from previous days.
- The format of the files used in the program is CSV. A CSV (comma-separated values) file is a text file in which information is separated by commas.
- All the files can be found in folder Code/time_data from the repository of the project.
- These files are automatically created and updated by the program itself, and the users never need to open these files. However, these files can be used by TAs to create test data for the program.
- There are 2 main types of files that readers need to differentiate:

- The “central database” of the program: file “all_task.csv”. An example of the file can be seen from the screenshot below. Note: the date format must be DD-MM-YYYY so that the program can execute the files.

all_data.csv M

time_data > all_data.csv

```
NOTE: Exported file is in CSV type and is best viewed with Excel. For visualizing data, please use Stats button in the app instead.
Date,Activity,Time (in second)
20-04-2022,activity_1,19.795814990997314
20-04-2022,activity_2,8.253262281417847
20-04-2022,activity_3,19.476365089416504
20-04-2022,Y2,0.0
14-04-2022,activity_4,28.568395137786865
14-04-2022,activity_5,118.4763650894165
14-04-2022,activity_6,18.476365089416504
21-04-2022,Y2,313.1271872520447
21-04-2022,Linear Algebra,70.80209374427795
21-04-2022,Database,67.69905948638916
21-04-2022,Work,72.86384201049805
22-04-2022,Database,5.964285612106323
22-04-2022,Linalg,54.520495653152466
27-04-2022,heelooo,10.866657733917236
07-04-2022,Work,26.16886568069458
07-04-2022,Interactive Code Block,23.541445016860962
07-04-2022,Database,21.219514846801758
11-04-2022,hello wrod,13.998448371887207
11-04-2022,GUI,12.378118991851807

```

- The files for data from each day users use the program: file in the format “DD-MM-YYYY.csv”. An example of the file can be seen below:

21-04-2022.csv U

time_data > 21-04-2022.csv

```
NOTE: Exported file is in CSV type and is best viewed with Excel. For visualizing data, please use Stats button in the app instead. WARNING: DO NOT edit the exported CSV file!
Activity,Time (in second)
Y2,313.1271872520447
Linear Algebra,70.80209374427795
Database,67.69905948638916
Work,72.86384201049805

```

9. Testing plan

- Test whether the graphical interface is working correctly or not, the buttons function correctly or not, and test the correctness of information shown on the GUI.
- Different functionality tests on the GUI (e.g., does the start/stop function works well, the function saves timer data into file, the function reads input file, function chooses another day, etc.)
- Test max 5 activity can be added
- Unittest for TimeConversion and TimeStatistics methods (These are tested separately to make sure the algorithms and math work correctly).
- Besides that, there are also some draft files used for testing.

10. The known shortcomings and flaws in the program

Currently, when a user clicks Reset or Restart button, the timer of the activity will be reset to 0, and there is no way the user can undo the action. The only theoretical way is that they already have a saved file of that activity before, and after they exit the program, they click don't save. This way sounds indeed complicated, and it will likely never happen in real life. The same situation happens with the Delete button: there is no undo method.

Now, the program shows a notification when the user clicks the reset, restart, delete button. However, I believe that it will be more convenient if there is an undo method.

One solution I can think of is using a backup file. The data in the file is the same as in the backend, but will not change when the reset, restart, delete button are clicked, and will be exported as a separate file in the backup folder.

11. 3 best and 3 worst areas

- 3 best areas:
    - The main window. All the features of the program can be done with the GUI, and the timer of the activities are conveniently shown live on the GUI.
    - Click the Statistics button: when there is at least one activity with the timer on, this button will show a window that contains the tables about the statistics of the data, as well as the chart for visualizing the data.
    - Click the Pomodoro button: The button will show a window to ask for the setting of Pomodoro Reminder from the user. After that, the program will have a Pomodoro Reminder running inside and will periodically show the reminder of resting/working.
    - Click the Help button: the button will show a window that has the information about manual instruction of how to use the program
- 3 worst areas:
    - The main window UI experience is quite poor since there is no decoration with colors.
    - TimeCoversion button with FinnishECTS: if the timer is not working long enough, the timer shown will be 0.000.
    - The files in the time_data are very sensitive. The program may not run if the users incorrectly modify the data in the file.

12. Changes to the original plan

According to the original plan, I would have started coding around 10.03, but in fact, I started on around 17-18.03 due to working on Round 6 too long. Otherwise, the documentation of the project also took longer than I expected: I needed to work on it for around 2-3 days.

13. Realized order and scheduled

- 28.02: first meeting with TA, ask questions
- 17.03: start coding the program (I need to finish round 5 & 6 first to learn things, then I'll start coding the project)
- 25.03: checkpoint 1. Add some main classes: Activity, Timer, Date, TimeConversion

- 15.04: checkpoint 2. 60-70% of the project is completed. Backend is almost completed, GUI is visible and basically usable, but there are many bugs in the program.
- 27.04: The coding part of the project is done.
- 05.05: The documentation of the project is done.
- 06.05 14:00: deadline. Upload all the code and the documentation on Gitlab before the deadline

14. Assessment of the final result

In general, I am satisfied with the final result of my project. Basically, it satisfies all the requirements of the Hard level, and it also has some special unique features. The final program is also user-friendly and very easy to use. There are also no bugs or unknown shortcomings so far.

I think that the backend of the project is the strongest part. The OOP design and implementation is very concrete and precise. I think that the classes in my project connect and work together really efficiently. The weakest part is the export data and the files (which is explained in depth in section 10).

I think that the program's structure can easily be changed, and can also be expanded. Different classes and the methods inside them are well-commented so that the developers can easily get the overall flow of the program. The most concerning part in terms of changing and expanding is the GUI class. I think it is too long and I was a bit greedy to put everything into one class. Maybe it will be much better and scalable if I splits the GUI into smaller classes.

15. Literature references and links

- Basics in Programming Y2 (Course material): https://plus.cs.aalto.fi/y2/2022/
- PyQt5 Documentation: https://doc.qt.io/
- datetime: https://docs.python.org/3/library/datetime.html
- pyqtgraph: https://pyqtgraph.readthedocs.io/en/latest/
- Stopwatch in Python (for class Timer): https://github.com/ravener/stopwatch.py
- PyQt5 GUI tutorial: https://zetcode.com/gui/pyqt5/

16. Attachments

- Installation Demo Video: https://drive.google.com/file/d/1fO7vrSXogasl0xaH150uoZU_gDmbMlfl/view?usp=sharing
- Usage Demo Video: https://drive.google.com/file/d/1QVH-oTRmjlxNso6arwjS1qDMs0lU3OQf/view?usp=sharing