Clippie: Your personal clipboard manager

Project Implemented by: Dany Sigha, Tina Chen, Nawaf Khaled Otaishan, Min Jee Lisa Moon

Table of Contents **2**

CHAPTER ONE

A Brief Description of Clippie

This project is a software application called Clippie with a graphical user interface component developed using python's PyQt5 framework. It is a clipboard manager for laptops and desktops. The project will consist of two main components: the visual interface and the sqlite3 database. The software application's main functions include the ability to copy, paste, delete, search, set "shelf-life", group cards, and set a password. The general purpose of the project is to:

- Improve the workflow of users
- Keep record of potentially important and sensitive information of users
- Provide a hassle-free way to swiftly store information

CHAPTER TWO

The Design Process

We used the Agile methodology to develop Clippie. The project is not very critical. Occasional mistakes in the software application are tolerable. The project is dynamic due to the limitations of the team’s ability to implement the desired features.

2.1 Iterative Development

Division of tasks:

- Dany and Nawaf co-worked on the implementations of the CardRenderer, CardObject, MainWindow, and ClipboardManager classes
- Nawaf also worked on the implementation of the card class
- Dany was in charge of developing the visual interface for the home page of the system.
- Tina worked on the implementation of the settings and login pages
- Lisa worked on the establishment of the database
- Everyone worked on the data accessor object class for their respective interfaces

Issues:

- Unfamiliarity with the PyQt5 framework meant that there was a learning curve which we each overcame at a different pace. Our progress was thus very slow at the beginning of the project.

- Parts of the project were developed faster than others, which led to the development of several data access object classes. This undermined our ability to test our code early, and eventually led to code duplication.

- Being new to this level of rigor for code maintenance and documentation has induced a strenuous refactoring workload to make the code readable and reduce code smells.

- With iterative development we had some tech debts when implementing some features that we haven’t addressed (grab clipboard class for windows)

2.2 Refactoring

In order to reduce code smells, refactoring was necessary to reduce the number of parameters our functions were using, and remove the parameters they did not need. To solve this issue we passed class instances as parameters in other classes' constructors. In some cases as many as 9 parameters were compressed into a single object.

We introduced the data accessor object for communication from interface to database. The data accessor object retrieves functions from the ClipboardManager_Db.py file to create connections to the database. We also used a decorator class to execute the password validity check function before storing the new password to our database. We greatly simplified our code so that it’s easier to follow and understand by creating more classes to organize implementations.

2.3 Testing Issues

We initially did not do any automatic testing for any of the interfaces. All the testing was done manually where we tried out different inputs to see if the outputs fulfilled our requirements. Thus, it was not till the very last week that our team developed unit tests to test the functionalities of the widgets on the interfaces.

2.4 Collaborative Development

Processes that worked:

Towards the end of the semester we organized several pair programming sessions to clean up the code and solve issues that we had trouble tackling individually. This led us to delegate tasks and it improved our workflow.

Issues:

Our team consists of members that lived in different time zones:

- New York GMT-4
- Korea GMT+9
- Saudi Arabia GMT+3

Working remotely in different time zones affected our communication and prevented some of us from attending every iteration meeting. Furthermore, our lack of experience with Github led to some inefficiencies such as: delays in pushes or pulls.

CHAPTER THREE

Requirements and Specifications

3.1 Requirements and Specifications

<u>User Stories and Use Cases that our System Follows:</u>

User wants to Delete card

a. Use cases involved: delete
b. Preconditions:
    i. If the system is locked by the user, the system must be unlocked.
    ii. There are one or more cards/datas inside the clipboard.
c. Main Flow:
    i. If the user selects the card that they want to delete, the user can press on the delete button to delete the card from the clipboard or to remove from the group only [S1-5]
    ii. [E1]
d. Sub Flow:
    i. [S1] User sees a card that he or she wants to delete.
    ii. [S2] The user presses the delete button located on the corner of the application.
    iii. [S3] The card gets deleted.
e. Alternative Flows:
    i. [E1]Nothing happens.

User wants to store a text, image, or url by copying it

f. **Use cases involved: Copy**
g. Preconditions:
    i. If the system is locked by the user, the system must be unlocked.
h. Main Flow:
    i. If the user *copies* text, image, or other datas, the exact same data will pop up in the clipboard as a form of a card.[S1][S2]
    ii. User does not copy anything.[E1]
i. Sub Flow:
    i. [S1]User copies an item.
    [S2]The data will be copied into the clipboard.
4. Alternative Flows:
    1. [E1] Nothing happens.

User wants to set a password.

j. **Use cases involved: Set Password**
k. Preconditions:
    i. Clippie has been established prior to setting lock.
l. Main Flow:
    i. The user decides to protect the application with a password. User chooses the “Set Password” option.[S1]

    ii. The user may choose not to lock the application by not setting the password [E1]

m. Sub Flow:
    i. [S1] The application locks and a password is requested from the system for the user to input.

n. Alternative Flows:
    i. [E1] The user does not lock the application. Nothing happens.

User wants to change the password.

o. **Use cases involved: disable password, change password, forgot password**

p. Preconditions:
    i. Clippie has a password set up from prior.

q. Main Flow:
    i. The user decides to change the current password on file.[S1] The user decides to disable the password.[E1] The user decides to choose the forgot password option. [E3]

r. Sub Flow:
    i. [S1] The user is prompted to change the password after entering a new password and the password on file is changed.

s. Alternative Flows:
    i. [E1] The user does not lock the application. Nothing happens.
    ii. [E2] The application prompts the user to enter the current password to disable the password.
    iii. [E3] The application sends an email to the user with a temporary password.

User wants to hide the content of the card

t. **Use cases involved: toggle visibility**

u. Preconditions:
    i. Multiple cards have already been stored by the user prior.

v. Main Flow:
    i. The user may decide to hide the card by pressing the toggle visibility button located at the corner of the card. The chosen card's contents get hidden. [S1][S3]

w. Sub Flow:
    i. [S1] The user presses the toggle visibility button located at the corner of the card.

x. Alternative Flows:
    i. [E1] Card is shown in the open area if it is not hidden by the user.

User wants to favorite a card.

y. **Use cases involved: "favorite" card feature**

z. Preconditions:
    i. Cards are created prior

aa. Main Flow:
    i. The user may choose to select a card as "favorite" which will be shown in a "favorites" folder [S1], else the user can choose to not use this function at all [E1]

bb. Sub Flows:
    i. The selected card will be added to the "favorites" folder

cc. Alternative Flow:
    i. Nothing gets added to the "favorite" folder

User wants access to locked application

dd. **Use cases involved: unlock**

ee. Preconditions:

   i. The system is locked by the user.

ff. Main Flow:

   i. The user is prompted to enter the password in order to access the system. [S1]If the system is not locked, the user will not have to input a password to access the system. [E1]

gg. Subflow:

   i. [S1] Once the password is entered, the system unlocks and the user can now access the system and all its functions.

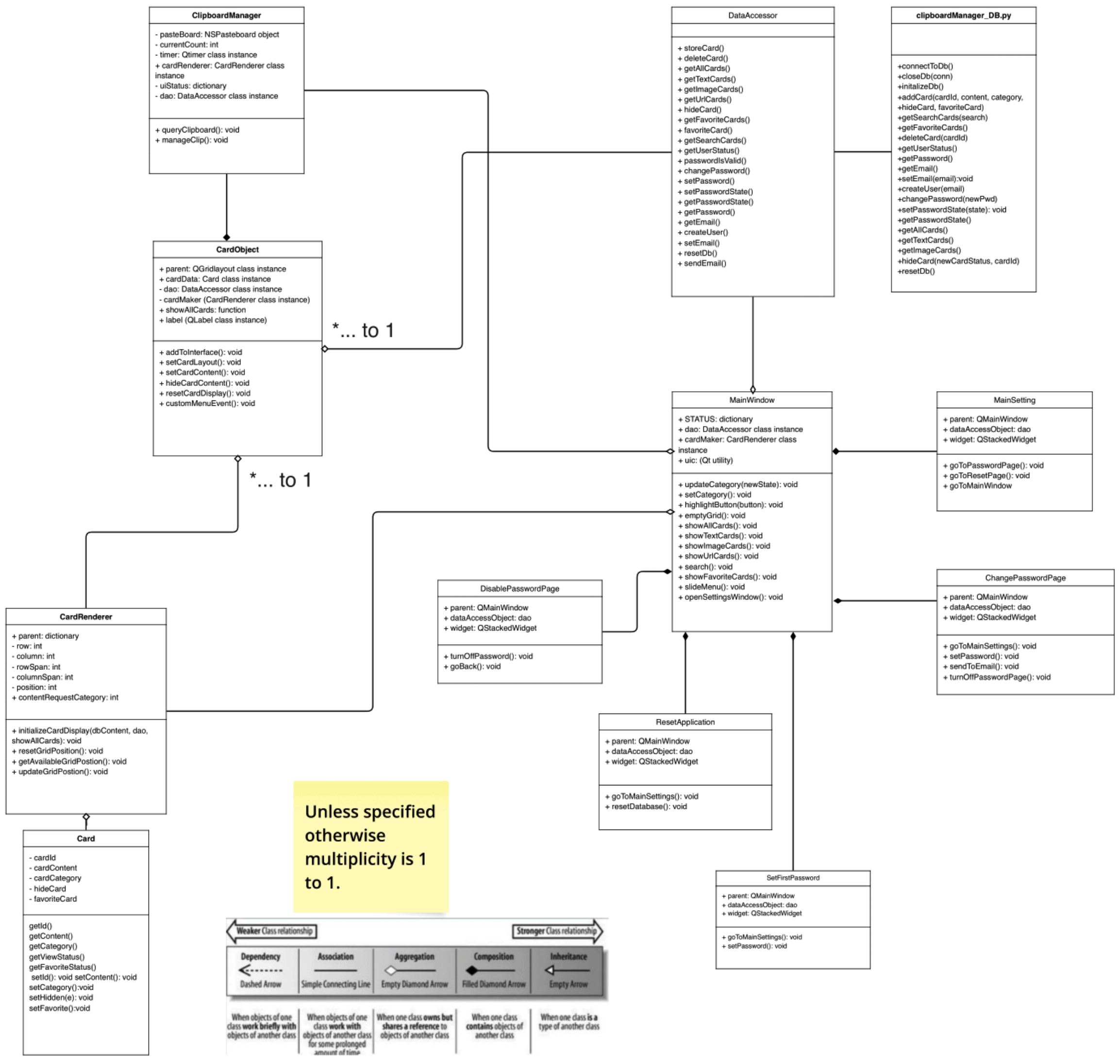
hh. Alternative Flow:

   i. [E1] Users have direct access to the system without the need of entering a password.

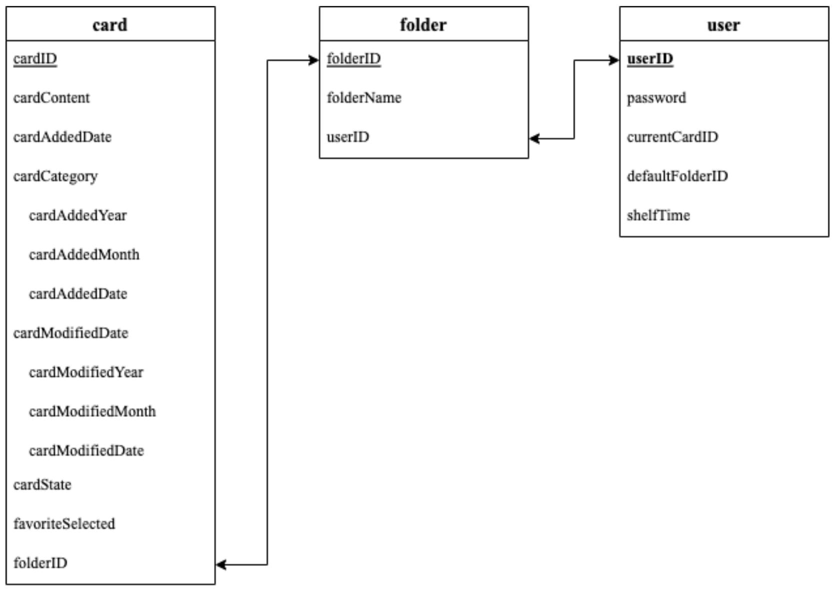
CHAPTER FOUR

Architecture and Design

01. The class diagram



02. The database model



03. An overview of our classes

Name	Functionality
`class UI(QMainWindow)`	Serves as the main window for our application. All clipboard content is displayed here.
`class CardRenderer`	Keeps track of the position of the cards in the grid layout on the main window. Contains the method that first adds cards to the interface when the program starts.
`class DataAccessor`	Acts as the intermediary between the interface and the database.
`class ClipboardManager`	Keeps track of the changes in the clipboard every second.
`class CardObject`	Creates the card labels on the interface one at a time whenever it is requested to do so.

class Card	Creates instances of a card object that stores the info of cards in the database on the fly.
clipBoardManager_DB.py (not a class)	Allows our program to store data beyond a single iteration.

04. About our Framework:

We chose PyQt5 because our team members all had experience using python. Moreover, PyQt5 is a tool of choice for cross platform development. Our use of the PyQt5 GUI framework heavily influenced our design. For instance, our interfaces use Qwidgets to represent our buttons, fields, cards and other interactive parts of our interface. Most of our classes are connected to our Main Window class and they are called in the Main Window's infinite loop.

However, we had numerous challenges and limitations when working with this framework. For example, the Qt grid layout structure does not match the requirements of our dynamic card storage system. To resolve this issue, we re-initialize the grid by querying the database when a card is removed or added to the system.

Moreover, we encountered several issues regarding scaling of images and icons on our interface, This led to pixelated visuals which did not reflect the quality of the media files we worked with. This issue can be attributed to our lack of experience with the framework. It was resolved by referring back to Qt documentation and extensive research.

Lastly, designing interfaces on PyQt5 can be done manually or via a visual interface. Due to our lack of experience with the framework, we were initially unable to leverage this feature of PyQt5, leading to inefficiencies.

CHAPTER FIVE

Reflections and Lessons Learned

Tina:

Everything regarding the software developing process, from architecture to using GUI developers, to establishing a database was new to me prior to developing this application. I've had to learn about how to set up and query from the database, how to add functionalities to buttons after generating a ui file, and how to organize my code so that they are easier to understand and manage. In addition, I really liked the Model View Controller framework presented during class because our application followed that framework and that made the implementation of our application much more organized and clean. My biggest takeaway from this experience is to always make your code easy to understand (using variable and function names that make sense to what they represent), keep the formatting consistent, and that refactoring and simplifying the code makes future edits a lot easier and manageable.

Nawaf:

This was an interesting project to tackle, and I think I am satisfied with what we were able to do given the time constraints and the limitations of being online and communicating with teammates across the world. This project challenged my skills in python, problem solving and time management. It was the first time I participated in a project of this size, but the learning process and the new experience gained by using the PyQt5 framework was fun and interesting. In terms of shortcomings, I wished we had better formatted our code, and made good comments on it during the development of the project. I wish we made the project more easier to distribute but due to time constraints we couldn't unfortunately. All in all, everyone in the team did their best to make this project ready for the presentation, and I wish them all the best.

Dany:

I am very proud of this project and what my team has accomplished. I think we all have grown as individuals and I am glad that this specific project was completed as it was my idea. I learned about code smells and the importance of writing clean and readable code. After we were introduced to these concepts, it was striking to me to see how much code smells my own code had. Furthermore, I was introduced to testing and its importance for the first time. It was particularly interesting and challenging to develop adequate and meaningful tests for a visual interface. Lastly, I learned a lot about the importance of git because working on a team of four people in three different time zones, it was essential to our collaboration.