Presented to the

De La Salle University - Manila

Term 3, A.Y. 2022 - 2023

Project Proposal in

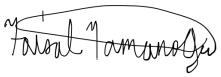
LBYCPEI - EQ3

Insight

Submitted by:

Hernandez, Jerry G. -

Lat, Jameul Lorenz P. - 

Tamano, Faisal Richard Jr. D. - 

Submitted to:

Mr. Ramon Stephen L. Ruiz

Submitted on:

June 26, 2023

Outline of the Paper:

1. Introduction
    - 1.1. Overview of the Project
    - 1.2. Goals and Objectives
2. Methodology
    - 2.1. Phase 1: Main Menu
    - 2.2. Phase 2: Study Timer
        - 2.2.1. Notification Tool
        - 2.2.2. Interruption Recorder
        - 2.2.3. Ignore Pomodoro
    - 2.3. Phase 3: Todo-List
    - 2.4. Phase 4: Notebook
        - 2.4.1. Navigation
        - 2.4.2. Note-taking Design
    - 2.5. Object Oriented Design
        - 2.5.1. Encapsulation
        - 2.5.2. Abstraction
        - 2.5.3. Polymorphism
        - 2.5.4. Inheritance
3. Project Description
    - 3.1. IPO Diagrams
    - 3.2. Flowcharts
    - 3.3. UML Diagram

4. Deliverables
5. Evaluation
6. Conclusion
7. References

1. Introduction

1.1. Overview of the Project

Welcome to our Study Program project that utilizes the Java programming language. You have access to a comprehensive suite of features tailored to enhance your learning experience across various subjects and disciplines. Whether you're preparing for exams, reviewing course materials, or simply expanding your knowledge, Our program provides a user-friendly interface, intuitive navigation, and a range of customizable tools to optimize your study sessions. From note-taking and flashcard creation to interactive quizzes and progress tracking. Our program empowers you to take control of your learning experience and efficiency.

1.2. Goals and Objectives

- To develop a study program with an efficient study timer.
- To implement a to-do list feature that has some task-management capabilities.
- To create a notebook feature that allows you to write notes with a keyboard and a mouse/digital pen.

2. Methodology

2.1. Phase 1: Main Menu

The first thing to be developed in the application is the menu. It will have four options available to the user: study timer, to-do lists, notebook, and quitting the program. If the user selects one of the options, they can always go back to the main menu.

2.2. Phase 2: Study Timer

Insight will have a timer designed to utilize the pomodoro technique, a time-management method developed for enhancing the productivity of its user (Cirillo, 2018). The implementation will be based on a paper written by Ruensuk (2016).

2.2.1. Notification Tool

One of its features, a notification tool, will allow users to set a specific length of time to determine how long their pomodoro will last and when to notify them. If the user decides not to choose, the program will default to 25 minutes. In addition, the program will have a break time, which the user can adjust depending on their needs. This will, however, default to 5 minutes if there is no break duration defined by the user.

2.2.2. Interruption Recorder

Although the notification tool can significantly aid in the productivity of its users, its benefits could be supplemented by providing valuable data about the users' habits. To be more specific, Insight will have a feature where users can opt to record and categorize timer interruptions. This means that if the user decides to cancel or stop the pomodoro timer, they can specify whether they canceled it due to an internal interruption (using social media, leaving work, or thinking of unrelated tasks) or an external interruption (interruptions caused by

unavoidable duties like an important phone call). Accordingly, the application can also display the recorded information about the interruptions (if the user wishes to see it).

2.2.3. Ignore Pomodoro

The interruption mechanism is very advantageous because it can help users analyze and improve their time management skills. Nonetheless, there are circumstances where users need to extend the pomodoro timer, like when they need a longer period of time for a task that necessitates continual focus. For this reason, Insight will enable users to ignore the notification alarms to assist users in adapting to these types of situations.

2.3. Phase 3: Todo-List

Although this feature will have some of the basic and expected capabilities of todo-lists (such as adding, removing, and editing a task), it will also take some inspiration from the design principle of the application TaskDo by Kuhail and Gurram (2019). Their task management approach influenced how we would execute this feature because it can increase the productivity of users by assisting them in planning their day-to-day activities.

After the program analyzes the performance and history of the user, the todo-list feature of Insight will suggest to the user which task to perform on a given day and time. This means that it will need to collect information involving various tasks and the user's performance first before providing a recommendation. Therefore, a database would be created for the recommender system to analyze and help the user plan activities.

The recommender system will learn from the user by checking if the user was able to successfully complete a task given a specific time period. Consequently, it also learns from periods where the user fails to accomplish a given task. The effectiveness of the recommender

system can be further improved through a feedback system where users can evaluate whether the recommendations from the program are helpful or not.

Other information that the recommender system will take note of are the task type (Intellectual, Physical, and etc.), day type (Sunday, Monday, etc.), and time of the day (Early Morning, Midnight, and etc.). After a task is finished, the recommender system will also ask the user for a satisfaction rating from 1-5 (1 being the lowest and 5 being the highest). Using all the data collected and found in the database, the recommender system will help the user plan their activities and create their todo-list.

2.4. Phase 4: Notebook

2.4.1. Navigation

The program's navigation system will be substantially influenced by the navigation system used in Li et al.(2018). The program aims to make it easy for users to navigate through their notes. Firstly, users will be able to categorize their notes by courses, allowing them to find notes from a specific course easily. Secondly, students will be able to search through titles of the notes. Thirdly, students will be able to sort their notes by date created or by date modified. On default, notes will be sorted according to which one was last edited. Other basic features in the program is the ability to add, delete, and edit the notes.

2.4.2. Note-taking Design

The design of the program's note-taking feature will be inspired by the design implementation of Ward and Tatsukawa (2003). In order for students to easily create graphical elements, they could use a pen or their mouse for drawing. However, they can also use a keyboard for text input. The text input's position is determined by the mouse placement. In addition, the user can simply just type wherever without generating a text box so that they can be

more efficient. Users can also select the eraser tool at the given toolbar to erase text or graphical elements found at their notes. Lastly, users will be able to create new pages and navigate through them.

2.5. Object Oriented Design Implementation

An object-oriented approach is very advantageous because it allows the programs using their approach to be very flexible and extensible (Klump, 2001). In addition their components can easily be integrated into other projects. For this reason, this approach is the most appropriate for this type of project. In order to fully maximize the potential of object-oriented design, the four pillars of object-oriented programming were applied in this project.

2.5.1. Encapsulation

Encapsulation simply refers to the concept of objects and classes (Klump, 2001). Classes are the blueprints that have attributes and behaviors that objects are built with. Using this concept, we will have a non-static class for timeBlocks (for the study timer), tasks (for the to-do lists), and notes(for the notebook). In addition it will have static classes such as the studyTimer, taskManager, and notebook.

2.5.2. Abstraction

Abstraction allows the program to hide the complexity of a specific class. It enables one to use some of the methods of a class without thinking about the unnecessary details. Abstract classes cannot be instantiated but can have methods that subclasses have to implement. Although we would not use any abstract class in this program, we would utilize abstract methods for some of the classes. The timeBlock class will have abstract methods that the subclasses will have to define.

2.5.3. Polymorphism

Polymorphism allows the program to extend a superclass and have a common method among classes, while having different ways of implementing that common method. The two types of time blocks, breakTime and focusTime, will both extend the timeBlock class. Although they share some similar methods, how they execute them will differ.

2.5.4. Inheritance

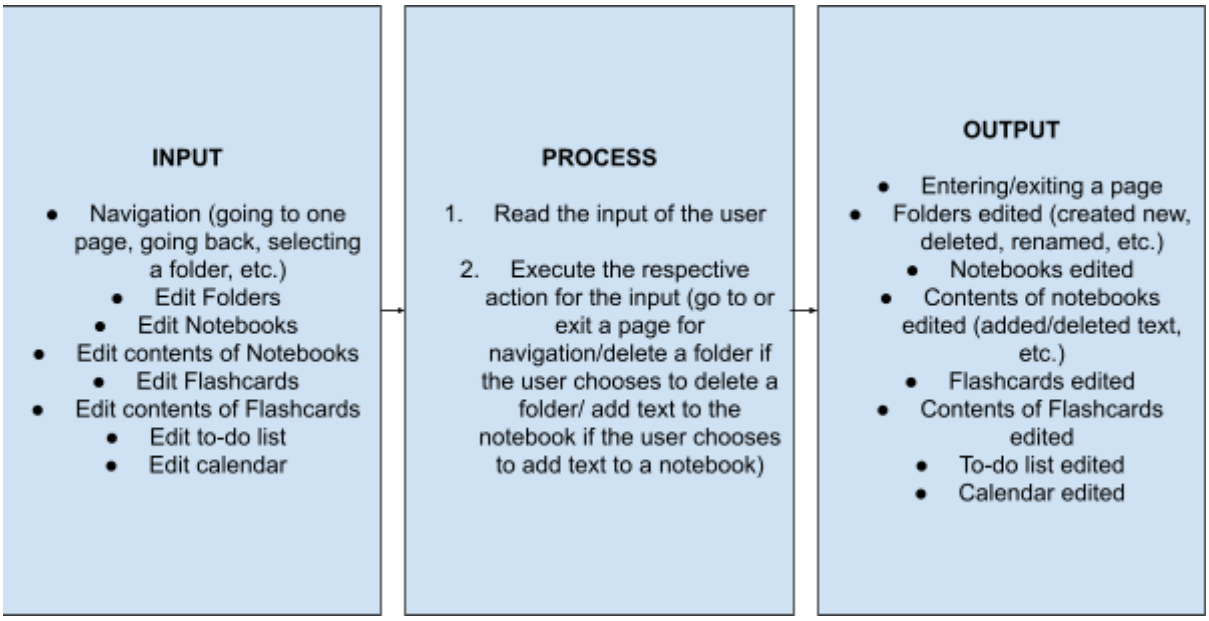
Inheritance allows subclasses to share the traits and behaviors of their parents, unless the trait or behavior is private. The different time blocks for the timer program, focusTime and breakTime, inherit some traits and behaviors from the timeBlock superclass. This allows them to easily reuse some of its properties and what a timeblock can do.

3. Project Description

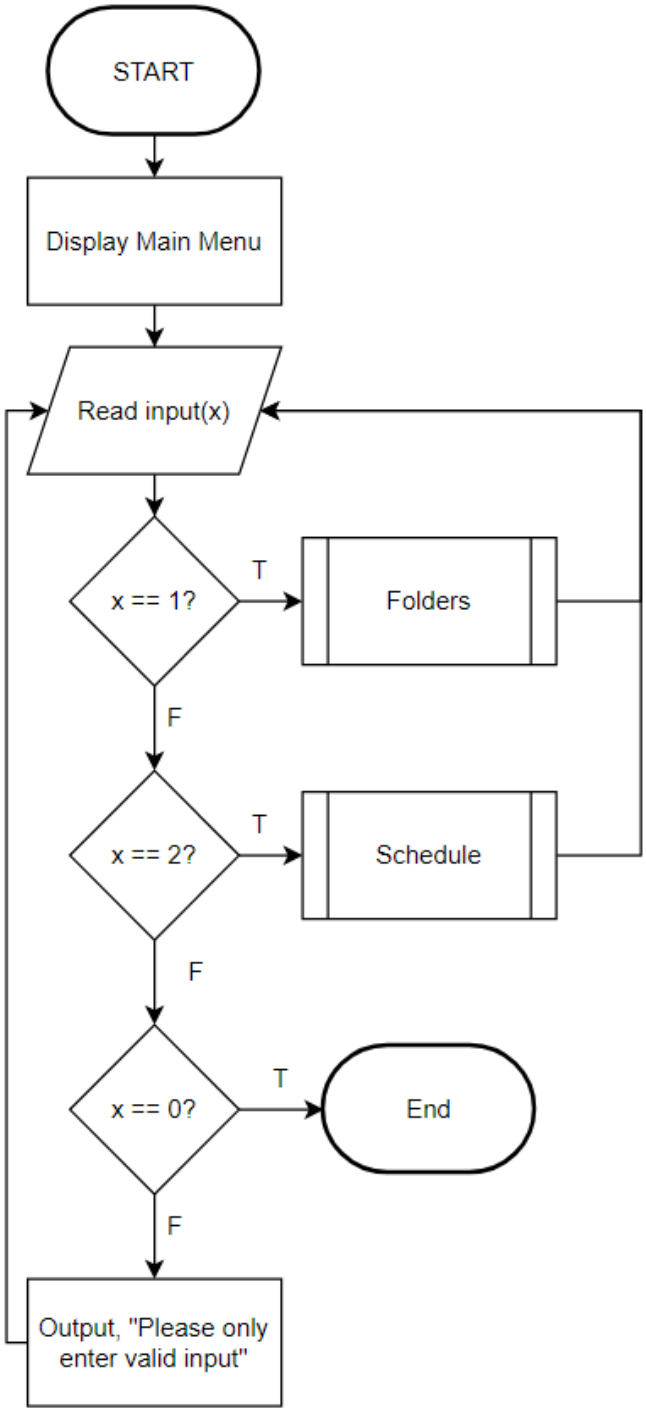
The project is divided into the parts (classes): main menu (main method), folders (class with methods for note-taking, flashcards, and timers), and scheduling (class with methods for a making to-do list and calendar).

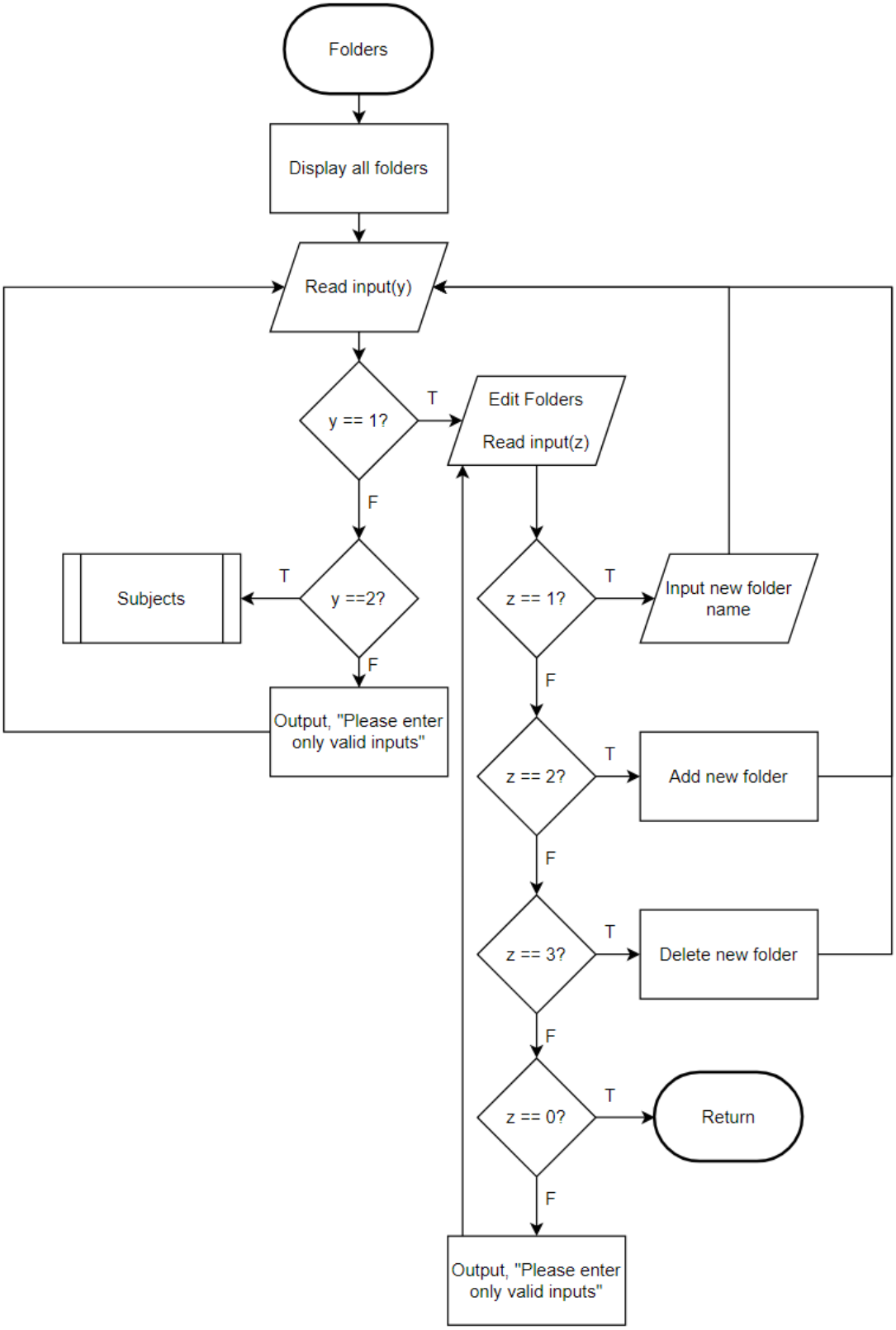
The program will first show the main menu where the user can either go to their folders or their schedule. Inside the folders are the subjects that the user has created as well as a choice to make new subjects. Each subject can have notebooks, flashcards, and more folders which are all subject to the preference of the user (the user may remove or add more notebooks, flashcards, or folders). For the scheduling, the user will have access to their to-do list and the calendar of the week. The calendar will show all of the activities that the user has entered along with the dates and times that the user has entered. Both the to-do list and the calendar are to be changed by the user. At the start of the program, the program will be empty (No folders, no to-do list, etc.). The user will create the folders for each subject and add notes to their to-do list themselves.

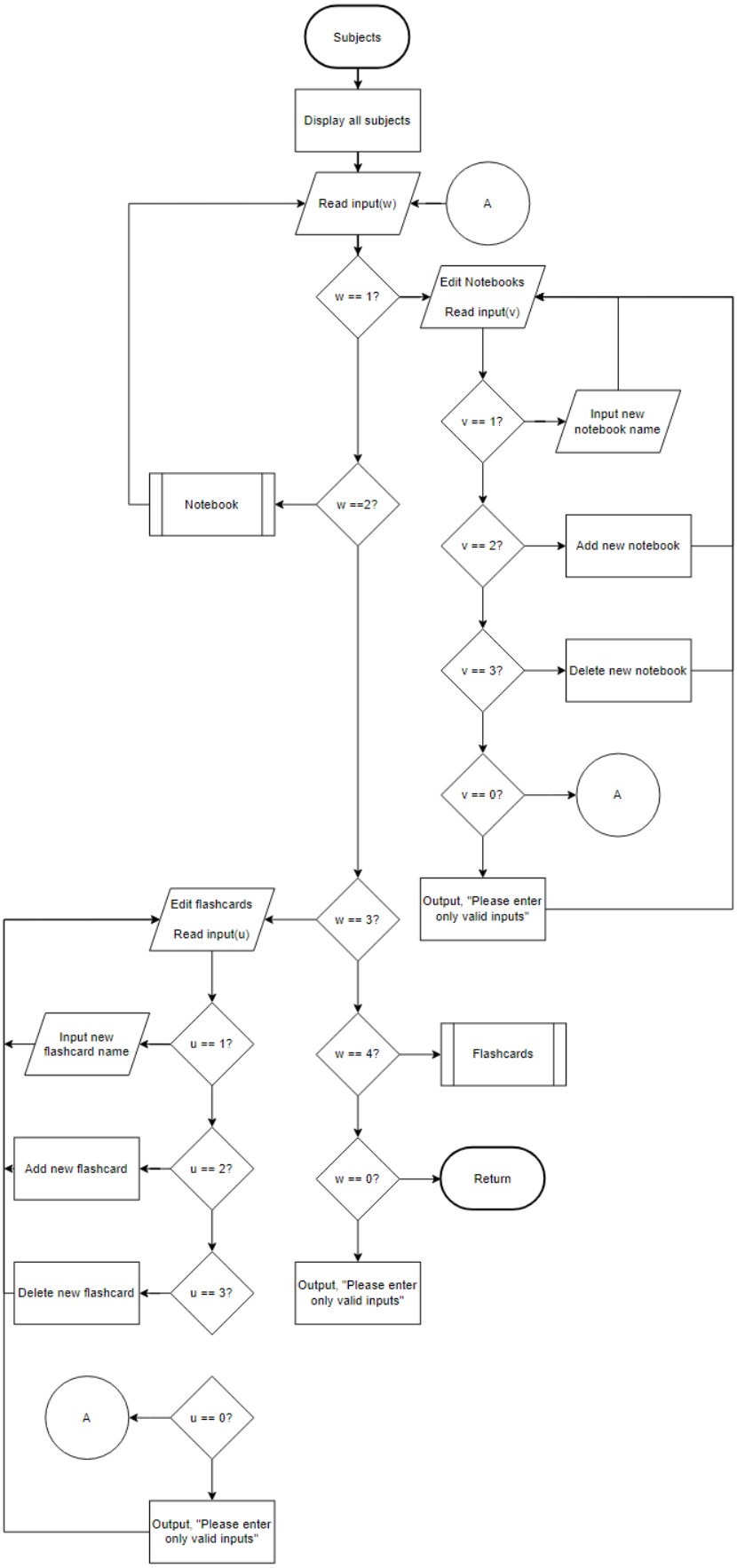
3.1. IPO Diagrams

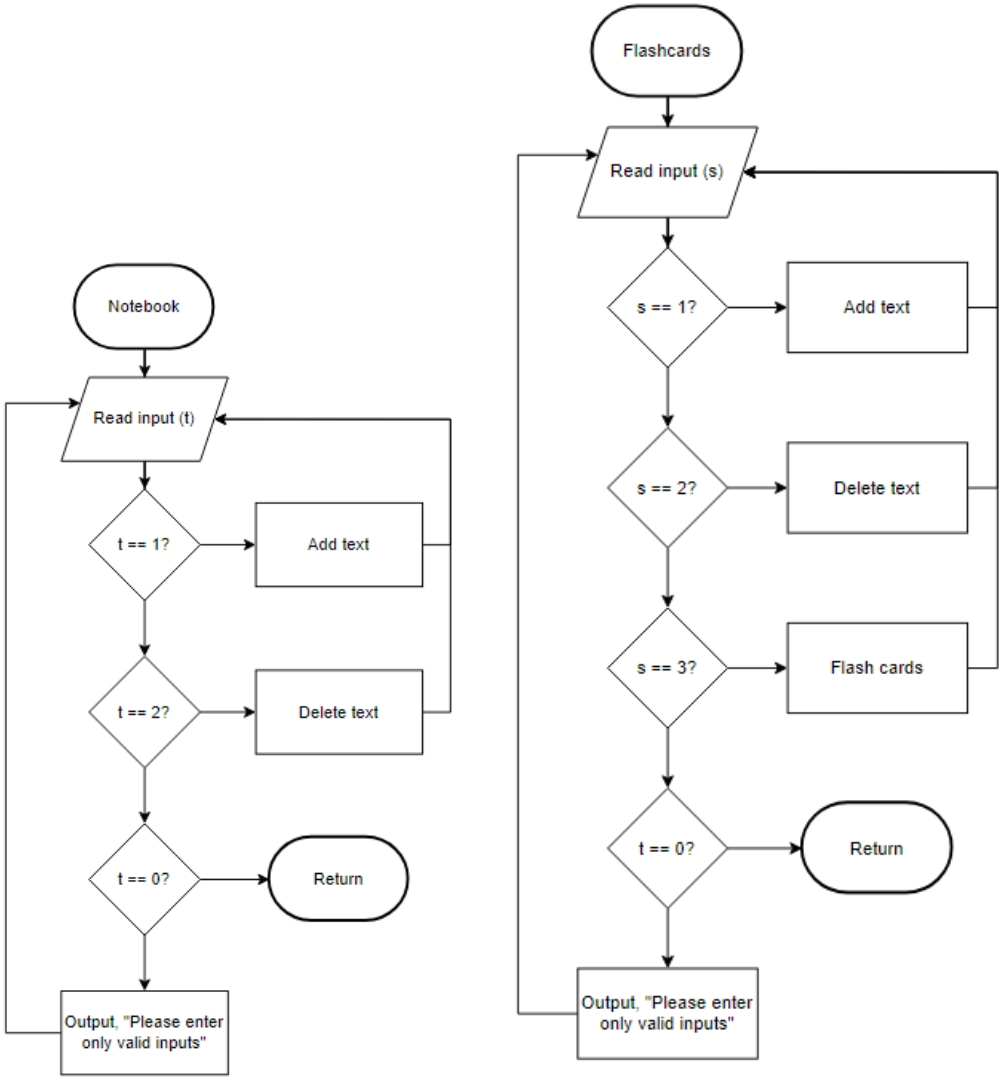


3.2. Flowcharts



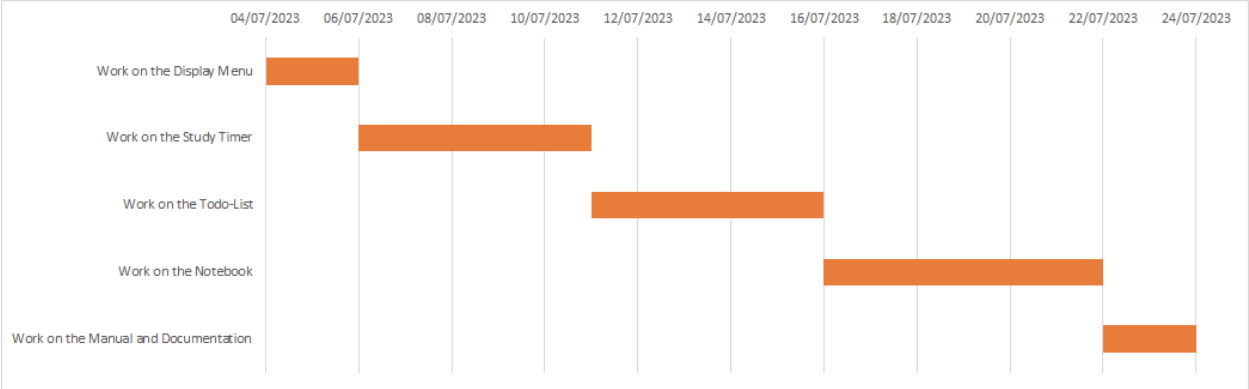

Folders
Display all folders
Read input(y)
y == 1?
T
Edit Folders
Read input(z)
F
Subjects
T
y ==2?
F
Output, "Please enter only valid inputs"
z == 1?
T
Input new folder name
F
z == 2?
T
Add new folder
F
z == 3?
T
Delete new folder
F
z == 0?
T
Return
F
Output, "Please enter only valid inputs"


Subjects
Display all subjects
Read input(w)
A
w == 1?
Edit Notebooks
Read input(v)
v == 1?
Input new notebook name
v == 2?
Add new notebook
v == 3?
Delete new notebook
v == 0?
A
Output, "Please enter only valid inputs"
w ==2?
Notebook
w == 3?
Edit flashcards
Read input(u)
u == 1?
Input new flashcard name
u == 2?
Add new flashcard
u == 3?
Delete new flashcard
u == 0?
A
Output, "Please enter only valid inputs"
w == 4?
Flashcards
w == 0?
Return
Output, "Please enter only valid inputs"



3.3. UML Diagram

4. Deliverables



From July 4, 2023 to July 06, 2023, we will work on the display menu for the program. The task will be divided into two people working on the graphical-user interface (GUI). While one person works on the backend of the display menu.

From July 06, 2023 to July 11, 2023, we will work on the study timer for the program. One person will be involved with working on the GUI, while another person will work on the notification tool and the ignore pomodoro feature, and one person works on the interruption recorder for the study timer.

From July 11, 2023 to July 16, 2023, we will work on the to-do list for the program. One person will be working on the GUI, while one person will work on the basic features such as adding a task, removing a task, and etc. The last person will work on the recommender algorithm.

From July 16, 2023 to July 22, 2023, one person will be working on the GUI, while another person will be working on the navigation features, and another person will be working on the note-taking aspect.

From July 22, 2023 to July 24, 2023, one person will be working on the user manual, while two other people will be working on the documentation.

5. Evaluation

To determine the quality of the program, a useful metric would be the one developed by Chidamber and Kemerer (Harrison et al., 1997). There are six metrics that we need to measure:

1. **Weighted Methods per Class (WMC).** This is used to determine the complexity and unity of the classes by measuring the number of methods present in a class.

2. **Depth of Inheritance Tree (DIT).** This is used to tell how complex the design is and how reusable a class is. It assesses the class' maximum level of inheritance hierarchy.
3. **Number of Children (NOC).** Used to know the level of reuse of a class by measuring the number of subclasses a class has.
4. **Lack of Cohesion in Methods (LCOM).** Cohesion in object-oriented design is encouraged, hence programs are expected to have high cohesion. It takes note of the lack of cohesion in the methods of the program's classes.
5. **Coupling Between Objects (CBO).** Coupling is generally expected to be low in programming. This criteria focuses on the coupling occurring between classes.

6. Conclusion

7. References

Cirillo, F. (2018). *The Pomodoro technique: The acclaimed time management system that has transformed how we work* (First edition). Currency.

Harrison, R., Counsell, S., & Nithi, R. (1997). An overview of object-oriented design metrics. *Proceedings Eighth IEEE International Workshop on Software Technology and Engineering Practice Incorporating Computer Aided Software Engineering*, 230–235. https://doi.org/10.1109/STEP.1997.615494

Kuhail, M. A., & Gurram, N. S. S. (2019). TaskDo: A Daily Task Recommender System. *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, 1–5. https://doi.org/10.1109/ICCIDS.2019.8862073

Klump, R. (2001). Understanding object-oriented programming concepts. *2001 Power Engineering Society Summer Meeting. Conference Proceedings (Cat. No.01CH37262)*, 1070–1074 vol.2. https://doi.org/10.1109/PESS.2001.970207

Li, I.-H., Wu, P.-J., Lin, Y.-L., Xu, H.-L., & Xie, J.-W. (2018). "GoNote": An aided learning note APP. *2018 IEEE International Conference on Applied System Invention (ICASI)*, 550–553. https://doi.org/10.1109/ICASI.2018.8394311

Ruensuk, M. (2016). An implementation to reduce internal/external interruptions in Agile software development using pomodoro technique. *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 1–4. https://doi.org/10.1109/ICIS.2016.7550835

Ward, N., & Tatsukawa, H. (2003). Software for taking notes in class. *33rd Annual Frontiers in Education, 2003. FIE 2003.*, *3*, S2E_2-S2E_8. https://doi.org/10.1109/FIE.2003.1265956