# Planning

## 1. The Scenario

My client, Lucy Z., is an IB (M20) student in an international school. As an IB student with many works to do, she always struggles to manage the time to do her IB works. As she expressed, the major problem that makes her unable to focus on her works is that she loves watching series on her computer, and once she was fascinated by some series, she cannot stop watching. Over and over, she realised that she has wasted too much time. Moreover, she has to memorise many academic terms and explanations for her Economics and Chemistry tests, but during the tests she said that she always forgets the terms which she has already memorised. After interviewing with her, I got to know that she needs a computer program that can remind and tell her what to do at a specific time and she can personalise such plans. This could distract her from keeping watching series. Also, she wants to have something that can aid her with memorisation. Moreover, she expects the program to have a friendly user interface and can be easily managed. [1]

## 2. The Rationale

For the reminders, I considered using some Timer methods in some programming language so that reminders can be called at pre-decided times adopting these methods. I also need to make friendly graphical user interfaces for my client to input details of tasks (plans) and able to see detail outputs when reminders pop up.

For aiding Lucy with memorisation, I did a massive amount of related research and consulted with the psychology teacher in our school. Thereafter, I got to know that Forgetting Curve can be used to hypothesises the decline of memory retention in time. This curve approximately shows how information is lost in human brains over time when there is no attempt to revise it. The reason why it is appropriate for my solution is because memory retention values on a curve can be calculated by using computing algorithms.

The most widely-used and simplest equation[2] for approximating forgetting is:

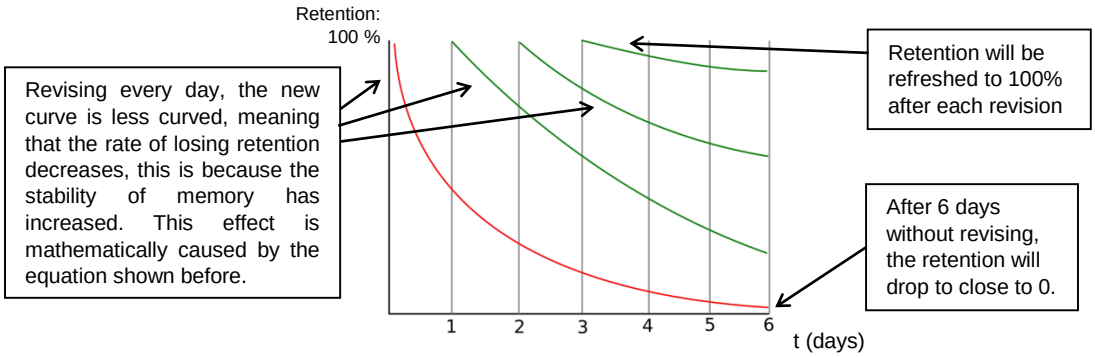
$$\mathrm{R} = \mathrm{e}^{-t/S}$$

Where R is retrievability (a measure of how easy it is to retrieve a piece of information from memory), S is stability of memory (determines how fast R falls without revising and it increases as the number of times of revising increases), t is time.

[1] See Appendices – Appendix A – Interview.

[2] Woźniak Piotr A., Gorzelańczyk Edward J. and Murakowski Janusz A. 1995. "Two components of long-term memory.". Acta Neurobiol Experimentalis. 55(4). Pp 301-305.

Figure 1 - A typical representation of forgetting curves with explanations




Source of the image: Praveen S., 2017. "Ebbinghaus Forgetting Curve". *Psychestudy*. Available at: https://www.psychestudy.com/cognitive/memory/ebbinghaus-forgetting-curve (Accessed: 04/01/2020)

However, this method may not be very accurate for estimating memory retentions, but this does enable to see how one's memory retention of some information loses over time, also providing the user an awareness to revise. Thus, I decided to adopt this method to visualise user's memory retentions of some specific information over time and remind user to revise when appropriate.

Finally, I chose to use Java, an object-oriented language. It has various open-source libraries for me to program efficiently. It has timer class that can be used to remind users. The memorisation task (forgetting curve) could also share some variables with users' inputted tasks, meaning that Java's Inheritance feature could be adopted. Also, it allows to create different classes for various needed algorithms and methods whilst maintaining an understandable flow of logic. Moreover, it can be used across all platforms and should extensions be required, these could easily be implemented. Swing as a GUI toolkit in Java was also decided to use since it has many sophisticated set of GUI components, helping me to build user interfaces easier.

Word count: 535

## 3. Success Criteria

1. Program will provide a friendly interface for end-users and can be easily managed.

2. Program will allow end-users to create executable normal tasks or memorisation tasks after inputting correct details of tasks.

3. When the current time is at user-inputted remind time, task/memorisation task notifications will pop up,

4. Program will allow end-users to either delay the notifications by certain time or choose to mark the tasks as finished tasks (will be deleted from backend data).

5. Program will analyse end-users' inputted ideal and acceptable memory retentions of memorisation tasks. Then, the system will automatically remind them when to revise based on their accepted retentions over time (current retention less than accepted retention).

6. When a notification window that advises end-users to revise their memorisation tasks pop up, end-users will be allowed to either choose to revise the memorising task on the day that the program advises them to or simply ignore the advice (Still will remind when appropriate).

7. Program will display a list of current tasks and a list of forgetting curves (memorisation tasks), when one is selected, its details will be displayed.

8. Program will allow forgetting curves to be visible for the users to view.

9. Program will allow the end-users to delete executing tasks or memorisation tasks.

10. Program will allow the end-users to sort executing tasks or memorisation tasks by priority (High to Low, Low to High).

11. Program will automatically send warning messages to end-users when errors occur.