Design

Classes and Variables

Since tasks and forgetting curves (FCs) created by end-users will share some properties (name, comment, priority, and time to remind/start an FC), so it was decided to adopt the Inheritance mechanism in Java. Task object and ForgettingCurve object classes will be built upon a parent class called Plan.

(Getter and Setter method for each variable will be included)

1. Parent Class: Plan

Properties:

Variable Name	Type	Explanation
planName	String	Name of a task or FC
comment	String	Comment for a task or FC
priority	String	Priority of a task or FC (Enable sorting tasks or FCs by priority later) (End-users are only allowed to input: “High”, “Medium”, or “Low” three types of priority)
remindTime	String	Time to remind user to do the task / time of starting a forgetting curve.

2. Child Class 1: Task extends Plan

This class will store information about Task.

Properties:

Variable Name	Type	Explanation
duration	long	Approximate duration (minutes) of competing a task, will be displayed to inform user when the user decides to start the task in the notification window.

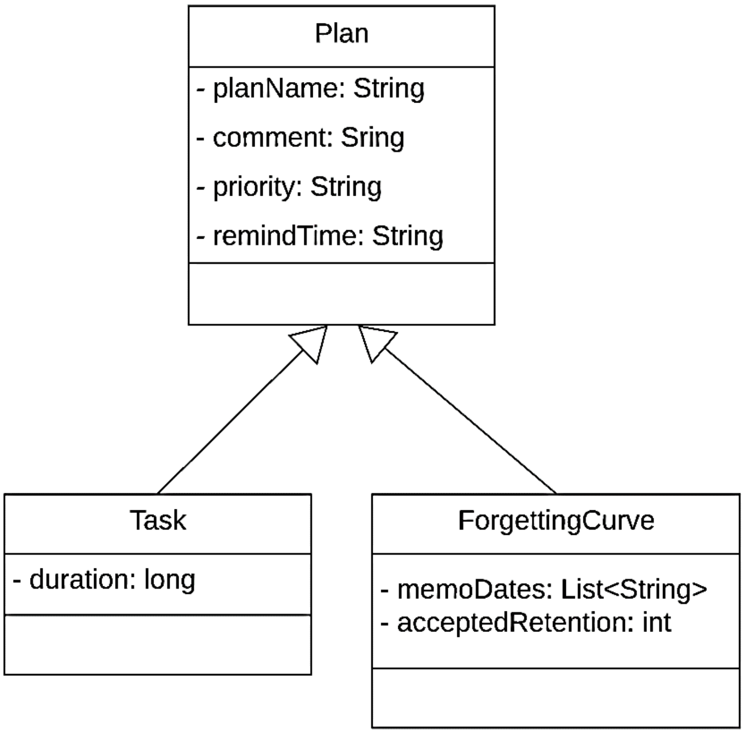
2. Child Class 2: ForgettingCurve extends Plan

This class will store information about ForgettingCurve.

In order to visualise forgetting curves, I initially considered using “retentions”, an array variable of doubles, to store the current memory retention value of a memorisation task every day. This means that the program needs to record and add the current retention value of a memorisation task to “retentions” array every day. But this is could make the program more complicated and inefficient. As explored and tested, I successfully developed an algorithm that only requires the program to identify and add each date that the user chooses to revise/start a memorisation task on.

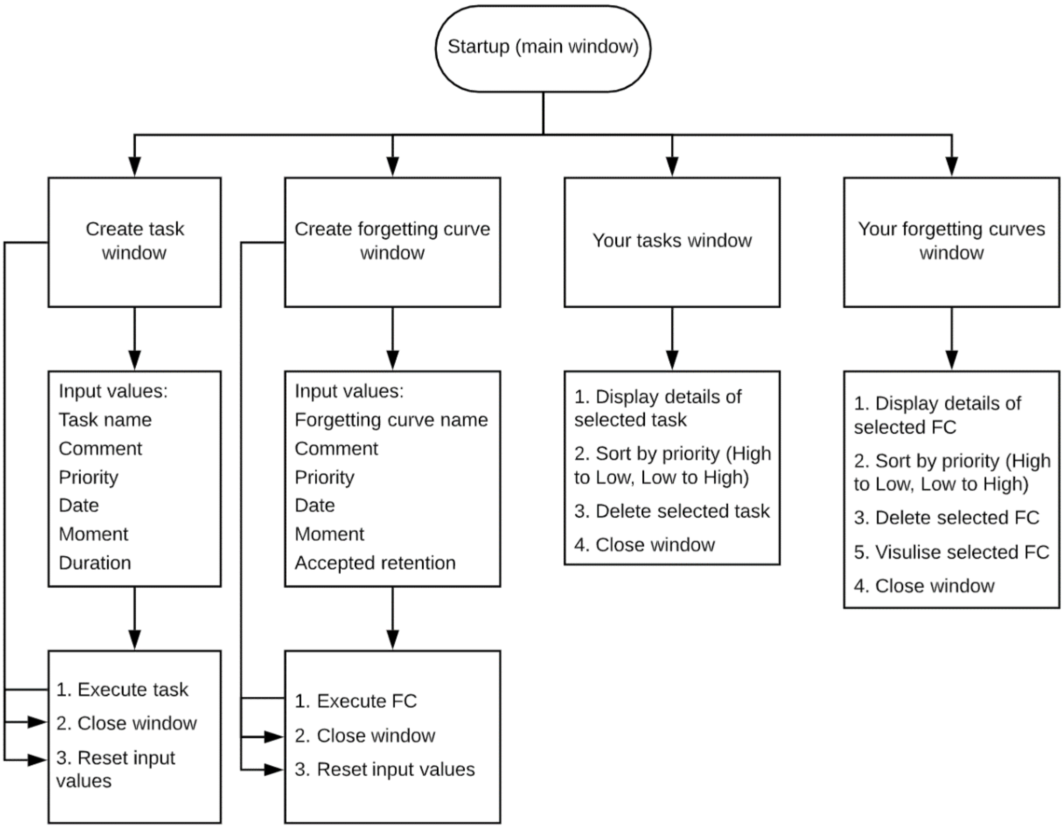
Variable Name	Type	Explanation
memoDates	List<String>	Every time when user decides to start memorising/revising the memorisation task related to a FC, the present date will be added to this list as String. (Strings will be in the form: YYYY/MM/DD hh/mm) E.g. 2020/02/05 12/00 = February fifth, 2020, 12 o’clock
acceptedRetention	Int	Minimum memory retention value of the memorisation task in percentage that is accepted by the user to be maintained. For example, if a user adds a FC of memorising 10 French words and set the accepted retention to be 50%. This is, when the memory retention of these 10 French words drops below 50%, the program will automatically notify the user to revise.

UML Diagram:



There will be TaskList and ForgettingCurveList classes to store tasks and memorisation tasks in lists.

Menu Navigation



On the "Create task window", user is allowed to input details of a task. After correctly inputting, user is allowed to execute the relative task reminder by clicking "Execute" button.

To create multiple task reminders, user can click "Reset" to default all inputted values then fill in new details of a task on the same window then execute. "Close" button also allows user to close the current window.

The same logic can also be applied to FC creating process.

Functions on "Your tasks window" and "Your forgetting curves window" are also listed.

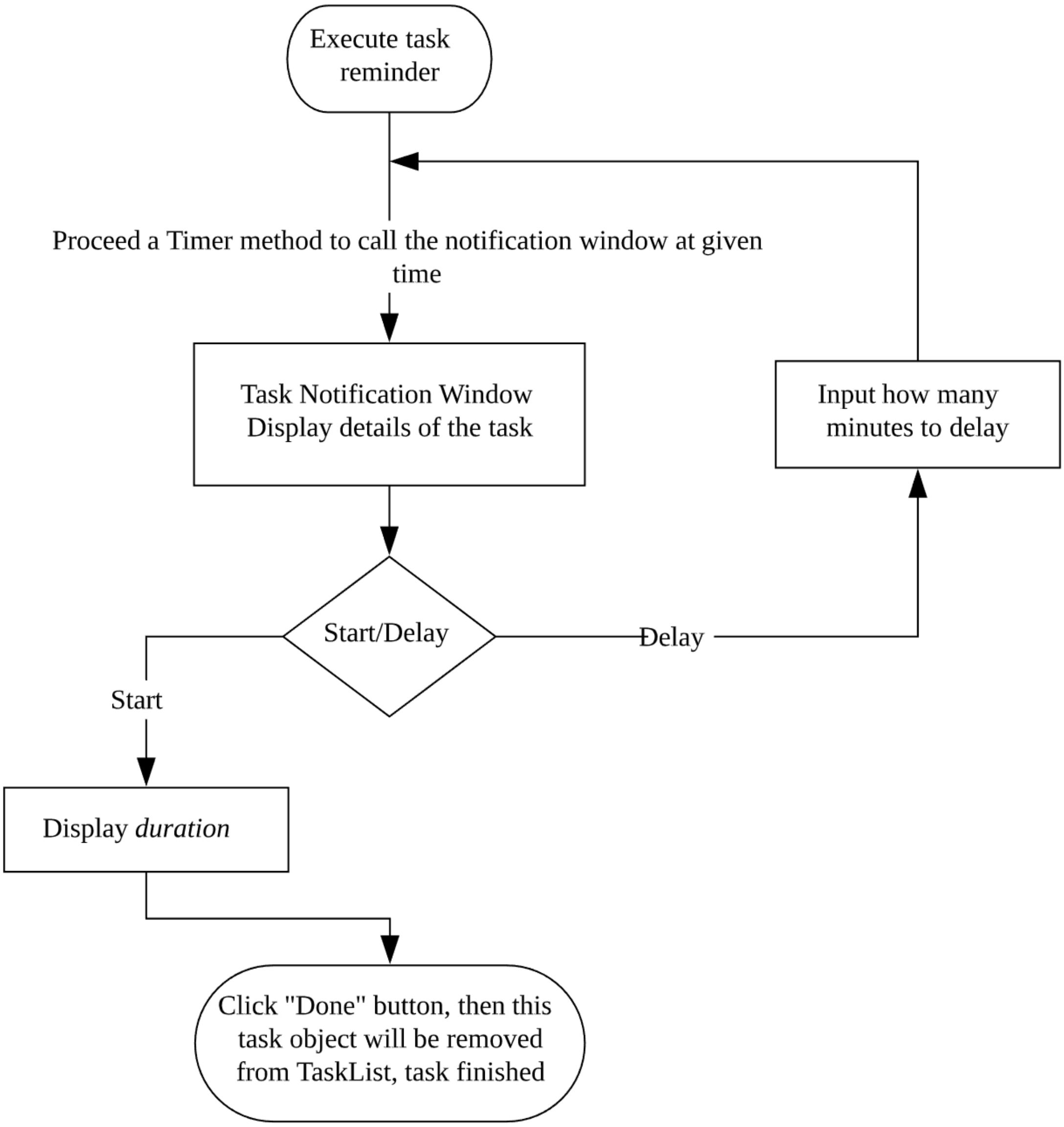
Algorithms

1. After Executing

1.1 Task

After clicking “Execute”, current task will be added to TaskList and a timer to call notification window will be proceeded based on user-inputted remind time.

As user can delay the notification on the notification, I considered developing a recursive method to enable user to delay for infinite times:

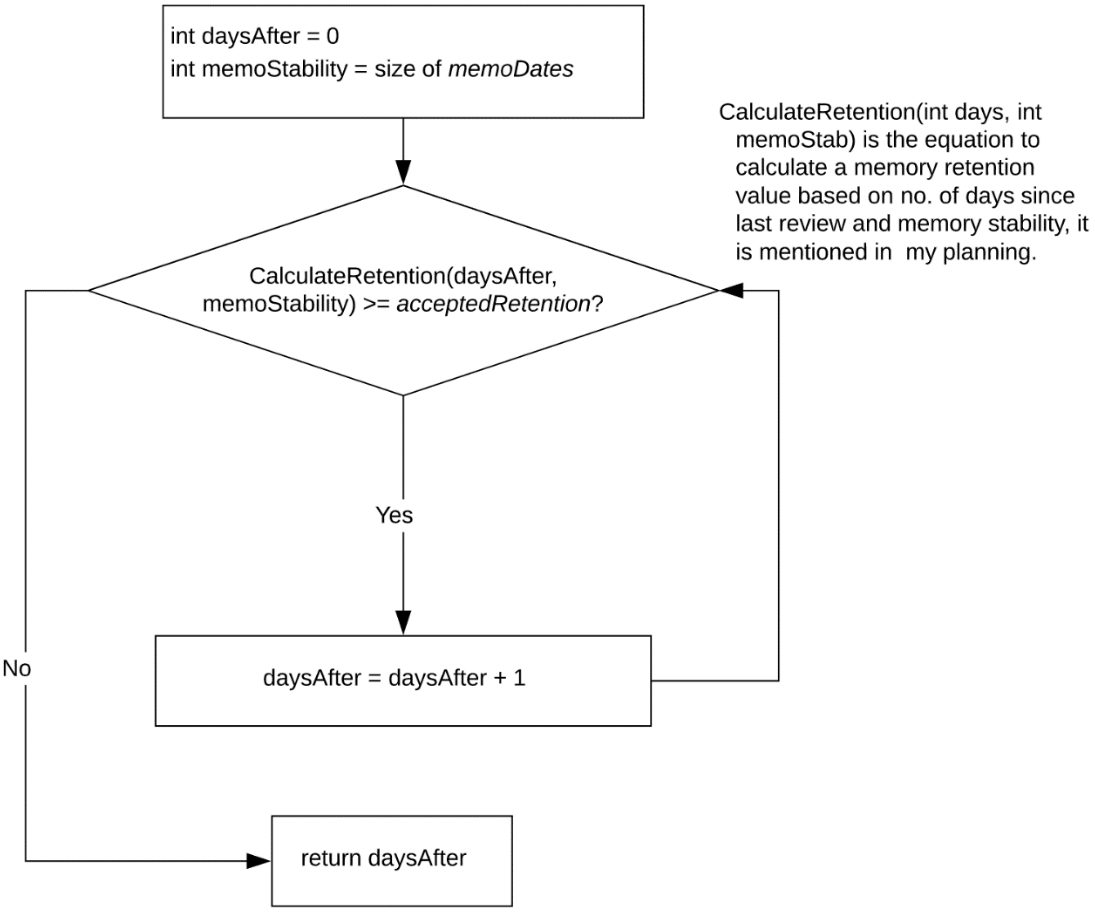

Execute task reminder
Proceed a Timer method to call the notification window at given time
Task Notification Window Display details of the task
Input how many minutes to delay
Start/Delay
Delay
Start
Display *duration*
Click "Done" button, then this task object will be removed from TaskList, task finished

1.2 Memorisation Task

1.2.1 Count after how many days the program will need to remind user to review

According to Acta Neurobiol Experimentalis (Woźniak Piotr A., 1995), stability of memory of a memorisation task could be initially set to be 1 and every time when user decides to revise, the stability of memory will be increased by 1. Hence stability of memory is equal the size of *memoDates* list.

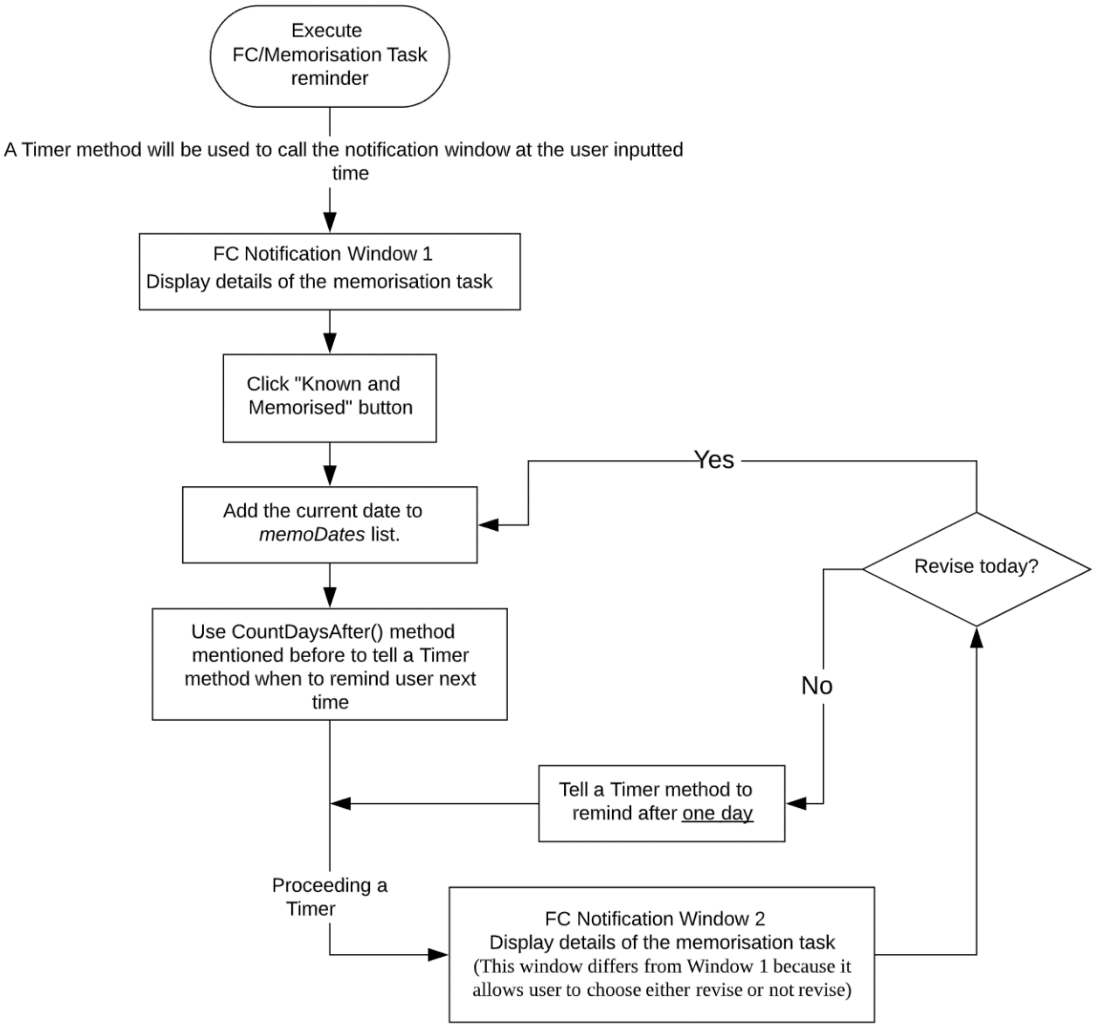
In order to automatically remind user to review the related memorisation task on the day when the current memory retention value is less than *acceptedRetention*, an algorithm to count the no. of days is developed, so after a number of days, the expected memory retention will not be acceptable:


int daysAfter = 0
int memoStability = size of *memoDates*
CalculateRetention(daysAfter, memoStability) >= *acceptedRetention*?
Yes
No
daysAfter = daysAfter + 1
return daysAfter

CalculateRetention(int days, int memoStab) is the equation to calculate a memory retention value based on no. of days since last review and memory stability, it is mentioned in my planning.

From the flowchart above, it can be seen that the mechanism is to use a loop to find the no. of days wanted. As long as the value of CalculateRetention() is greater than *acceptedRetention*, *daysAfter* will be increased by 1, then it will be put as parameter again in CalculateRetention() to compare the CalculateRetention(*daysAfter*, *memoStability*) value with *acceptedRetention.* Until the value of CalculateRetention(*daysAfter*, *memoStability*) is less than *acceptedRetention*, the loop will stop then output after how many days the current retention value will be less than *acceptedRetention*. This algorithm can help to tell Timer method when should user be reminded based on user's accepted memory retention value of a memorisation task.

1.2.2 Memorisation Task Reminding Process


Execute FC/Memorisation Task reminder
A Timer method will be used to call the notification window at the user inputted time
FC Notification Window 1
Display details of the memorisation task
Click "Known and Memorised" button
Add the current date to *memoDates* list.
Use CountDaysAfter() method mentioned before to tell a Timer method when to remind user next time
Yes
Revise today?
No
Tell a Timer method to remind after one day
Proceeding a Timer
FC Notification Window 2
Display details of the memorisation task
(This window differs from Window 1 because it allows user to choose either revise or not revise)

The mechanism for memorisation task reminding is that knowing acceptedRentention and memoDates, it is able to determine after how many days should remind user to revise. As the notification window pops up, user can decide either revise today or not revise today, **if revise today**, then today's date will be added to memoDates then the notification will be displayed after CountDaysAfter() days. **If not**, then FC notification window will be set to display one day after the current date. I decide to realise the whole process by recursively calling the FC notification window 2 as that flowchart demonstrates.

(Adopted a non-recursive method for making the video, will be mentioned in Development, Section: Algorithms)

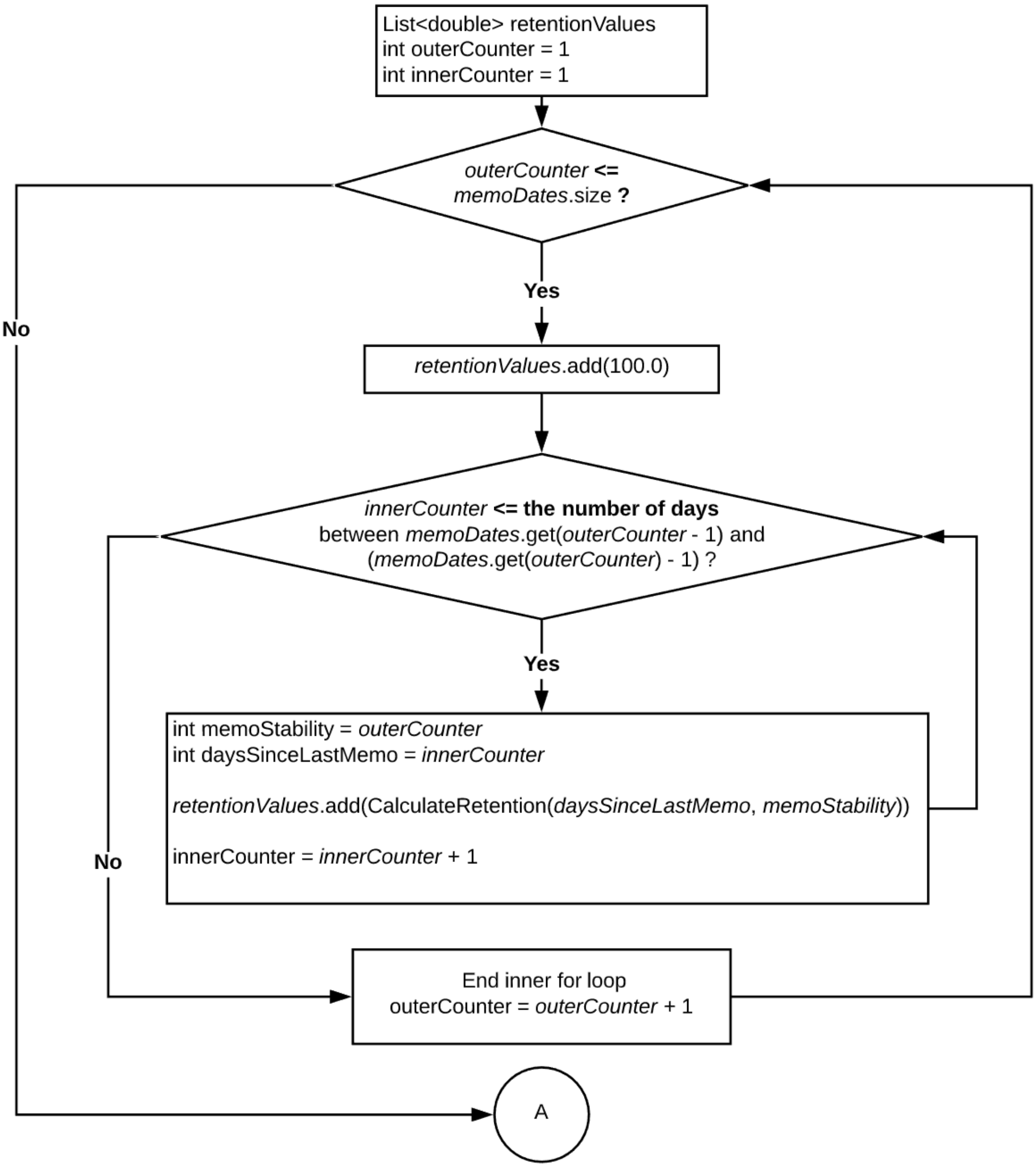
2. Displaying

2.2 Visualising Forgetting Curves

I have developed an algorithm to generate all memory retention values of a memorisation task from the starting day to the current day by only adopting *memoDates* list.

This algorithm is divided into 2 parts.

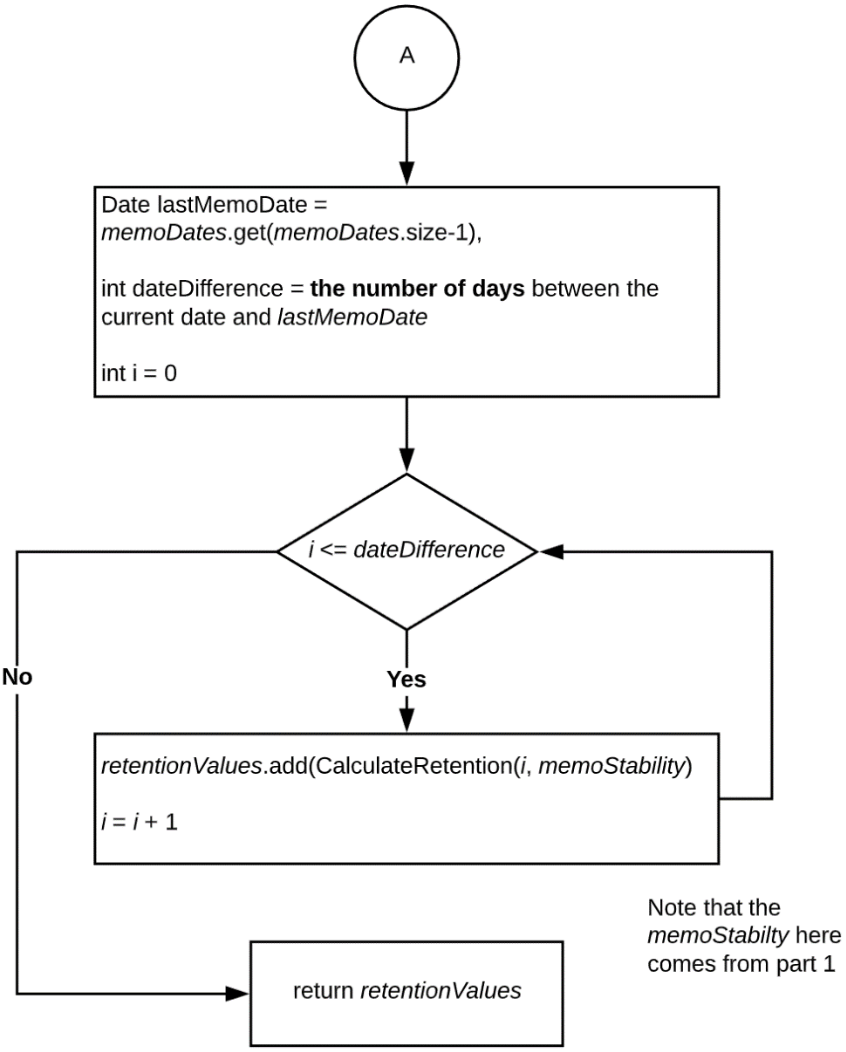
Part 1:


List<double> retentionValues
int outerCounter = 1
int innerCounter = 1
outerCounter <= memoDates.size ?
Yes
No
retentionValues.add(100.0)
innerCounter <= the number of days between memoDates.get(outerCounter - 1) and (memoDates.get(outerCounter) - 1) ?
Yes
No
int memoStability = outerCounter
int daysSinceLastMemo = innerCounter
retentionValues.add(CalculateRetention(daysSinceLastMemo, memoStability))
innerCounter = innerCounter + 1
End inner for loop
outerCounter = outerCounter + 1
A

This part aims to get retention values from the FC starting date to the day before the last date revised (last object in *memoDates* list), it splits all days between the starting date and the last date revised into periods, the stability of memory in each period is equal and the number of days in each period is known. So it can get retention values period by period. The inner loop is for getting retentions of days in one period, the outer loop is for getting retention values of all periods.

E.g., if there are date 1 (start), date 2, and date 3 in *memoDates*, period 1 is from date 1 to the day before date 2, here the memory stability is 1, so retention values in period 1 are able to be obtained, the next period is from date 2 to the day before date 3, stability increases by 1, again, retention values can be obtained. Then all retention values of these two periods can be obtained.

Part 2:


A
Date lastMemoDate =
memoDates.get(memoDates.size-1),
int dateDifference = the number of days between the current date and lastMemoDate
int i = 0
i <= dateDifference
No
Yes
retentionValues.add(CalculateRetention(i, memoStability)
i = i + 1
return retentionValues

Note that the *memoStabilty* here comes from part 1

This part aims to get retention values between the current day and the last day revised and it has only one period, so it uses one loop to get all retentions.

Overall, this algorithm allows to get all retention values from starting date to current date by only using *memoDates*, this can work because the parameters to get a single memory retention value can all be found from *memoDates*.

Also, when user just started a memorisation task, program will only output one retention value, 100.

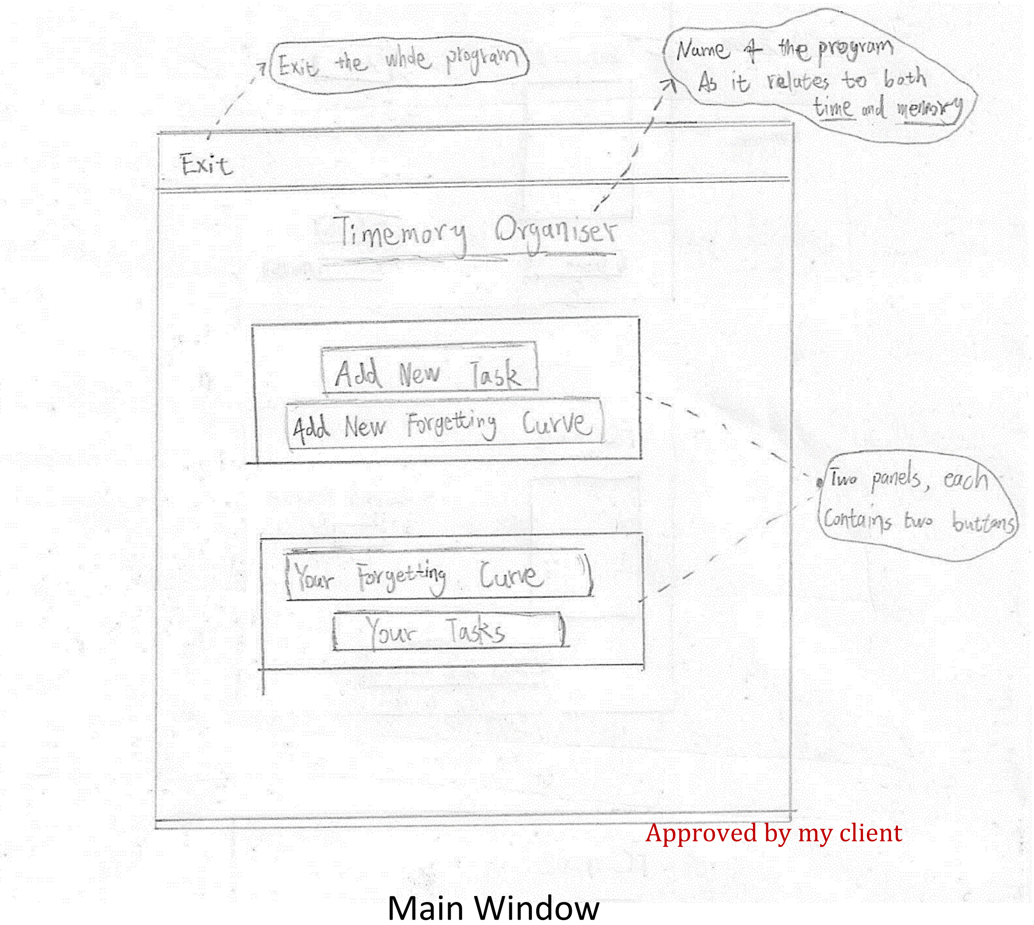
Finally, this algorithm will return all retention values in a list, with such values, it is enabled to plot forgetting curves.

There will also be a method to convert a list of string type to calendar type.

Since many algorithms and methods related to FC are interrelated, so I decide to put them into one class called ForgettingCurveMethods.

Moreover, sorting method to sort FCs and tasks by priorities will be Selection Sort as it performs a smaller number of swaps compared to bubble sort.

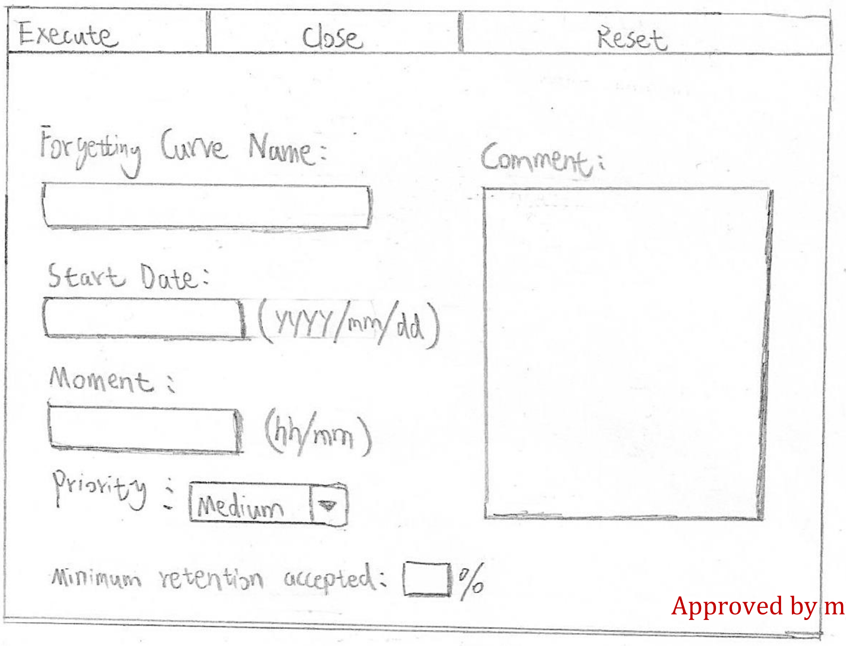
Graphical User Interface Design (Sketches)



Approved by my client

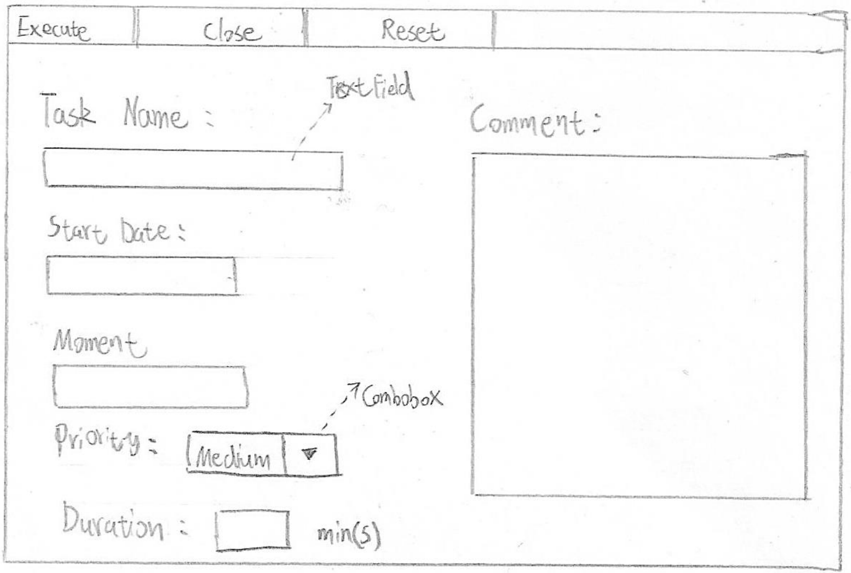
Main Window

Task Creating Window



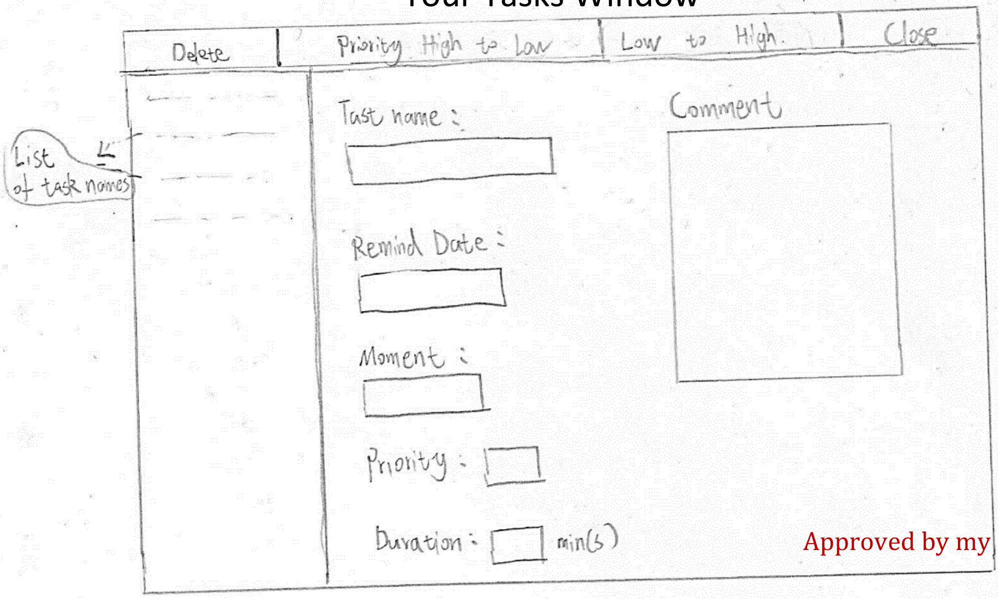
Approved by my client

Memorisation Task/FC Creating Window



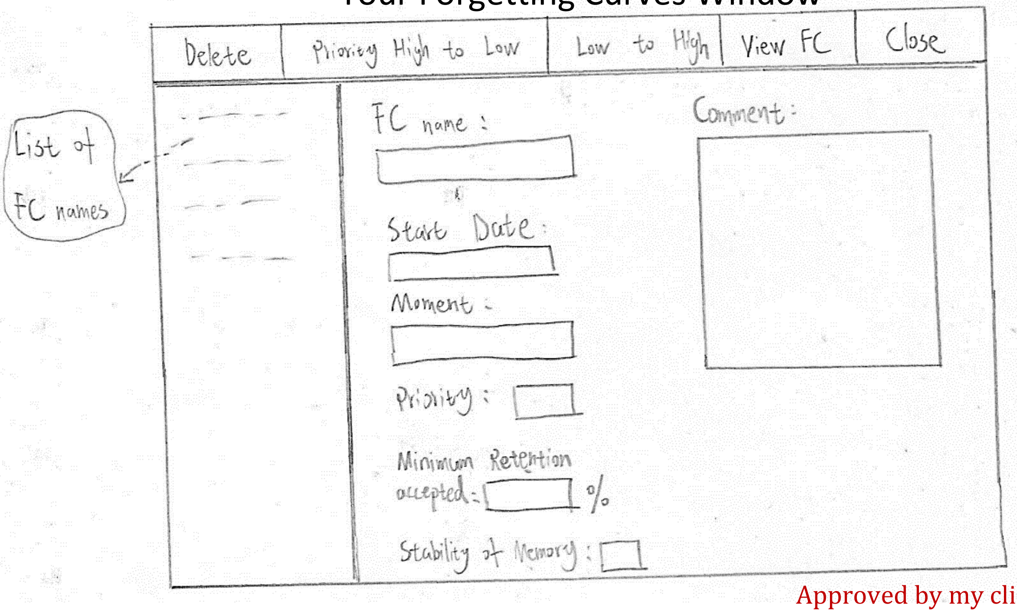
Approved by my client

Your Tasks Window



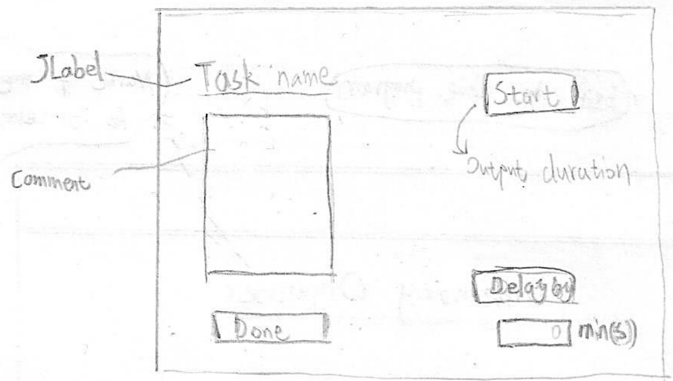
Approved by my client

Your Forgetting Curves Window



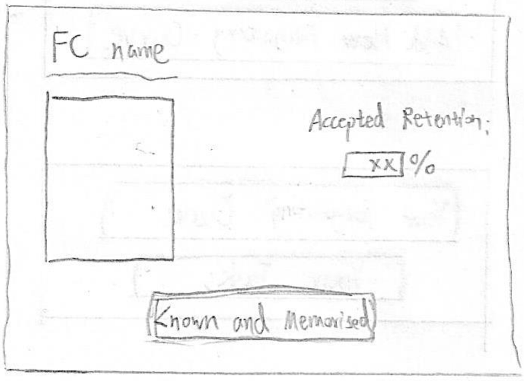
Approved by my client

Task Notification Window



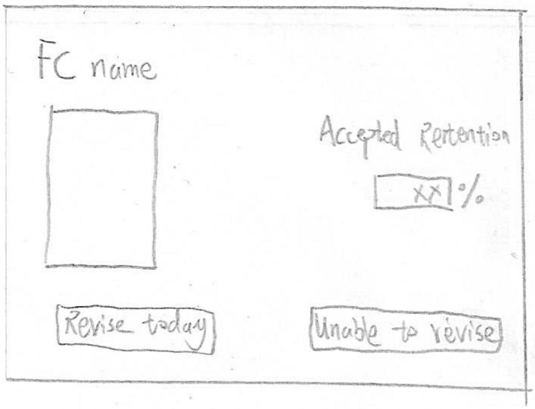
Approved by my client

FC Notification Window 1



Approved by my client

FC Notification Window 2



Approved by my client

Test Plan

Test number	Description	Input	Expected outcome
1	Testing if task notification window can be displayed with correct details. at the user-decided time. This links to success criteria 2 and 3.	Fill in appropriate details in correct data types.	The notification window is displayed with correct details at the user-decided time.
2	Testing if the FC notification window 1 can be displayed with correct details at the user-decided time. This links to success criteria 2 and 3.	Fill in appropriate details in correct data types.	The FC notification window 1 for is displayed with correct details at the user-decided time.
3	Testing if task notification can be delayed by a user-decided time interval. This links to success criteria 4.	Time interval to delay: int; Click "Delay" button.	The same notification is displayed after a user-decided time interval.
4	Testing if the notification of the task can be finished and can be deleted after clicking the "Done" button. This links to success criteria 4.	Click "Done" button.	Notification window is closed, and the task related to the notification is deleted from the list, which can be shown on the "Your Tasks" window. (No such task will be displayed anymore)
5	Testing if the FC notification window 2 can be displayed with correct details at the correct time based on user-inputted accepted retention. Using calculators, to calculate after many days it will result a retention value that is below user's accepted retention. This links to success criteria 5.	Click "Known and Revised" button on the FC notification window 1 for a memorisation task.	The FC notification window 2 is display after **a correct number of days** that will result a below-user-accepted retention value. Also, correct details will be displayed.

6	Testing if the FC notification window 2 for a forgetting curve can be displayed with correct details after a correct number of days based on user-inputted accepted retention **over time.** Using calculators, to calculate after how many days it will result a below-accepted retention value. This links to success criteria 6.	Click “Revised” button or “Unable to revise today” button on the FC notification window 2 for a memorisation task.	The FC notification window 2 is displayed with correct details after a correct number of days based on user-inputted accepted retention.
7	Testing if the executing tasks and forgetting curves can be displayed in some windows in the form of lists, the details of each of the tasks or forgetting curves in the lists can be displayed when selecting a specific item on the lists. This links to success criteria 7.	Click “Your Tasks” button or “Your forgetting curves” button on the main window. Click an item on the list displayed	The window showing the correct list which displays current tasks or forgetting curves is popped up. Details of the task or forgetting curve selected are displayed.
8	Testing if each of the current forgetting curve can be visualised. This links to success criteria 8.	Select a forgetting curve then click “Show FC” button	A visualised forgetting curve of the selected forgetting curve is popped up.
9	Testing if tasks and forgetting curves can be manually deleted This links to success criteria 9.	Select a forgetting curve or a task then click “Delete” button	The selected task or forgetting curve is deleted from the lists in the windows.
10	Testing if tasks and FC in the lists can be sorted by priority. This links to success criteria 10.	Click “High to Low” Or Click “Low to High”	Tasks and FCs in their lists are sorted based on priority in the windows.
11	Testing if a warning message will be automatically sent to end-users when input errors occur. This links to success criteria 11.	Filled in abnormal data: Inappropriate data type or inappropriate forms of data – e.g. wrong calendar format	A window displaying warning message will pop up.

Special Testing Plan for Forgetting Curves

8Apr	9Apr	10Apr	11Apr	12Apr	13Apr	14Apr
Notify		Notify			Notify	Notify
		Revise			Not revise	revise
						100%
		100%	60.7%	36.8%	22.3%	13.3%
100%	36.8%	13.5%				

Accepted Minimum Retention: 30%

Enter acceptedRention: 30%, starting a memorisation task (FC) on Apr 8th.

After 2 days, on Apr 10th, the program will notify to revise. Choose "Revise today", then the program will notify to revise after 3 days on Apr 13th, however, choose "Unable to revise today" on the 13th, the program will notify after one day, on Apr 14th. On Apr 14th, choose to revise.

100%	36.8%	100%	60.7%	36.8%	22.3%	100%

Visualising this memorisation task, it is expected to display the values shown above.

These are all being pre-calculated using the equation of getting retention value mentioned in Planning.