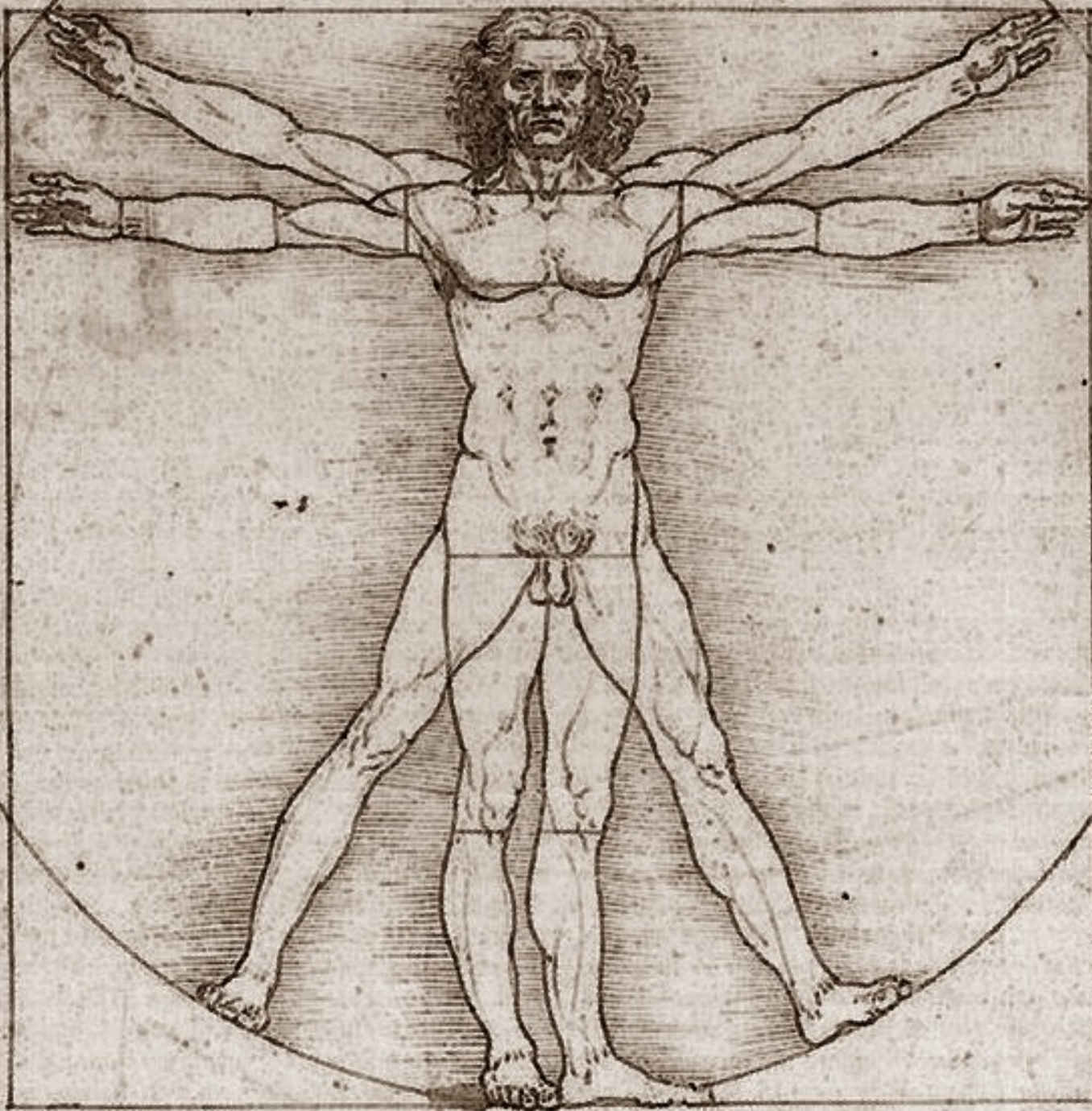


# Vitruvian Brain



# Table of Content

- [Introduction](#)
- [Goals](#)
- [Current Solutions & Flaws](#)
  - [Anki](#)
  - [Supernemo](#)
  - [Obsidian](#)
  - [Roam Research](#)
  - [Vitruvian Brain](#)
- [Principles](#)
- [The 2 Brains](#)
  - [First](#)
  - [Second](#)
- [Fundamentals](#)
- [Incremental Reading](#)
- [Spaced Repetition Scheduling](#)
- [Interest Repetition](#)
- [Interleaving](#)



# Introduction

One of the many problems with self-learning today is the lack of full-featured tools for your studies, current tools, even when they provide all the necessary features to a productive & pleasurable learning session they might have other flaws such as lock-in, complexity, online-only, platform-specific etc.

The purpose of this document is to present the current state-of-art in study/learning/thinking tools, present their respective advantages and disadvantages and the solution currently being worked on, as to address all the flaws where others have "failed", picking the best parts, leaving the bad and trying building a standard that is Simple, Open Source, Plain Text, Robust and Extensible.

To summarize, this is an attempt to create a lock-in free, offline, platform-independent "standard" in which universal learning tools can be build upon.

## Goals

The `Vitruvian Brain` standard has the goal of describing a system in which the following learning tools will be made available:

Zettelkasten, Incremental Reading, Spaced Repetition, Interest Repetition, Incremental Video, Mind Maps, Learning From Text(LFT), Incremental Thinking, Incremental Work, Getting Things Done(GTD), Interleaving and Incremental Writing.

It'll do so by describing a series of procedures, formats and protocols in which \*any\* developer[s] might choose to "implement" or "be compatible with" . This is to make sure the standard is long-lived and does not succumb to the fate of "bit-rot" in which programs tend to have a rather short shelf life. Protocols are much more resilient because they're implementation-independent, so anyone can write an implementation for it, one analogy might be that web browsers come and go all the time, but the HTTP protocol has shown itself to outlive most of its concrete implementations.

## Current Solutions and their flaws

### Anki

Anki is a flashcard spaced repetition application where you can review cards on a exponential schedule, only needing to keep the daily review.

#### Pros

- Popular: A lot of people in medical school and the language learning community have been using anki for ages, so familiarity with them is a plus.
- Community: A lot of pre-made decks and plugins

#### Cons

- Light Lock In: Anki "locks" your files, videos, images, audios and flashcards in a non plain-text format thus "sucking" everything you throw at it and you have to do some work in get it back.
- Complexity: maybe there's too many, unnecessary options which might scare beginners.
- Bad Integration: in order to interact with something like a zettelkasten system you would need to either connect with AnkiWeb(API) or open the app every time(too wlo).

#### Conclusion

Implementing anki into the system I've presented you with would be possible, but we would suffer from integration inefficiencies and the "what it could have been" effect.

## Supermemo

Supermemo is a software program created by Piotr Wozniack. Its most known features are "Incremental Reading" and "Spaced Repetition Flashcards".

### Pros

- Completeness: It has Spaced Repetition( SM17 ), Incremental Reading, Sleep Chart, Knowledge Tree, Neural Review and more.
- Efficiency: It has probably the most efficient interval calculating algorithm(SM17) and Incremental Reading is not implement to the level it has in supermemo, it has clozes, image clozes, topic schedule etc.
- Community: It's considerably big, you can find plugins, workflows, tips, keyboard shortcuts etc.

### Cons

- Complexity: It looks like the central panel for a nuclear reactor station. It takes too much time to get used to the buttons, shortcuts, conventions, nomenclatures etc.
- Platform Dependend: Windows Only( Wine is not always reliable )
- Zettelkasten: I have heard you can implement it on SuperMemo, In my opinion it also suffer from a "what it could have been" effect in my view.
- Lock In: It locks your files into a non-plaintext format.

### Conclusion

In terms of raw cognitive power and study efficiency, SuperMemo is not only probably the best tool around, but it quite literally pioneers a lot of the ideas in which we take as best practices today( Spaced Repetition, Incremental Reading, Incremental Learning in general ). But I feel like we could take the best parts of supermemo, turn it into a concrete-implementation-agnostic and let people make their own tools according to it, it might leave to considerable growth in the field of study tools

## Obsidian

Obsidian is a note-taking application where it works with plain-text markdown files, it's very commonly used to implement a zettelkasten system.

### Pros

- Almost Platform Independent: It's available for Windows, MacOS, Linux and mobile.
- No lock in: It's plain text files a folders
- Popular: A lot of people in medical school and the language learning community have been using anki for ages, so familiarity with them is a plus.
- Community: A lot of pre-made decks and plugins
- Extensible: With The Obsidian Plugin API you can write features the tool does have but you want to use.

### Cons

- Files is the lowest level you could go, and block references don't work that well.
- Concrete implementation that might not have everything users need and the possibility of a plugin might be as promising.

### Conclusion

Obsidian comes the closest to an "open" study tool. It does not lock your files in, have zettelkasten, incremental reading and spaced repetition plugins. My only problem with obsidian is the fact that its main "primitive" or "unit of work" is a markdown file, which is fine for most people but I'm pretty sure some people would desire a more "lightweight" way of creating relationship. One example would be a interesting quote, if you want to make this quote a node in the graph/zettelkasten you need to create an entire file for it, which is in my opinion too cumbersome/unecessary, if you collect 100 quotes you'll have 100 more files ... of course you can all put them in a single file, but referencing them from this single file would go "against" the atomicity principle of zettelkasten and each individual one would not be visible in any graph.

## Roam Research

"Roam Research" is a paid online note-taking application where people can access their "graphs" or note online.

### Pros

- Simple: Very slick and intuitive.
- Almost Platform Independent: Works on the browser, it means it's almost universal.
- Blocks: small, cheap and removes the "does this deserve a file?" effect.

### Cons

- Online Only:
- Lock In: Roam has the option to export to json but still too limited.
- Paid: Roam is a paid tool which makes it non-accessible for a lot of people.

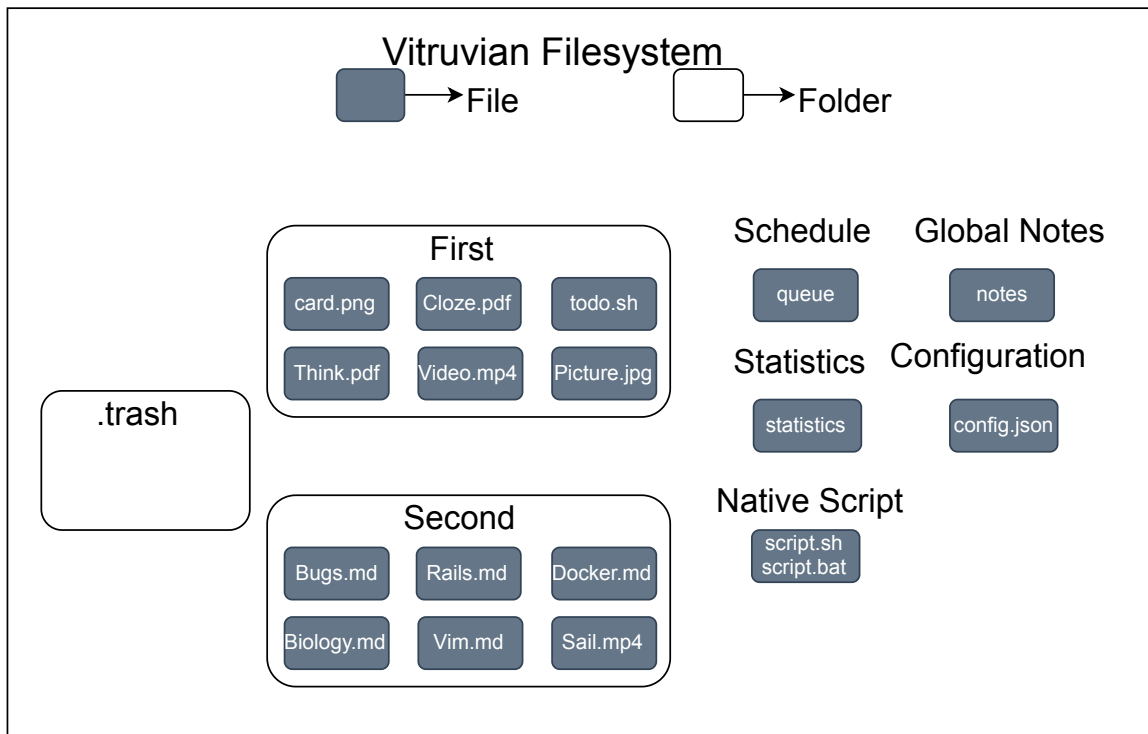
### Conclusion

Roam is probably the best option between all of the presented here but ... It's not free. So that's a no.



## Vitruvian Brain

`Vitruvian Brain` Standard is a series of documentation and practices in which developers might implement learning applications. By following those design choices it'll ease the cognitive load of inventing everything from scratch, they can have a battle-tested framework for study tools and focus on the delivery of said tools, instead of discussing which solutions are the best one.



### Pros

- No lock in: It's just files and folder.
- Platform Independent: Since it's a standard and not a concrete implementation we might implement it in: C++, python, bash, Web, iOS, Windows, Linux, MacOS, Embedded etc.
- Good Primitive: Bullets, files and folders, everything else is built atop those basic primitives.
- Bullets: small, cheap and remove the "does this deserve a file?" effect.
- No "Software Rot": Since it's not even a concrete implementation, there's nothing to rot here, protocols outlive programs by the decades, that's the intent here.

### Cons

- To be added ...

### Conclusion

The very purpose of the `Vitruvian Brain` standard is to learn from those previous tools, take their best points, mitigate their weaknesses and make an open standard from a hypothetical study tool with the features that provide the most benefit( Spaced Repetition, Incremental Reading, Incremental Writing, Zettelkasten, Interest Repetition, Interleaving etc. )

## Principles

The `Vitruvian Brain` will takes inspiration from the `Unix Philosophy` such as:

`Do one thing and do it well`

`Plain text is the ultimate interface`: We don't make use of `JSON`, things such as configuration are also non-json plaintext.

Since this is not a piece of software but a standard, anyone might build a client that operates with the standard.

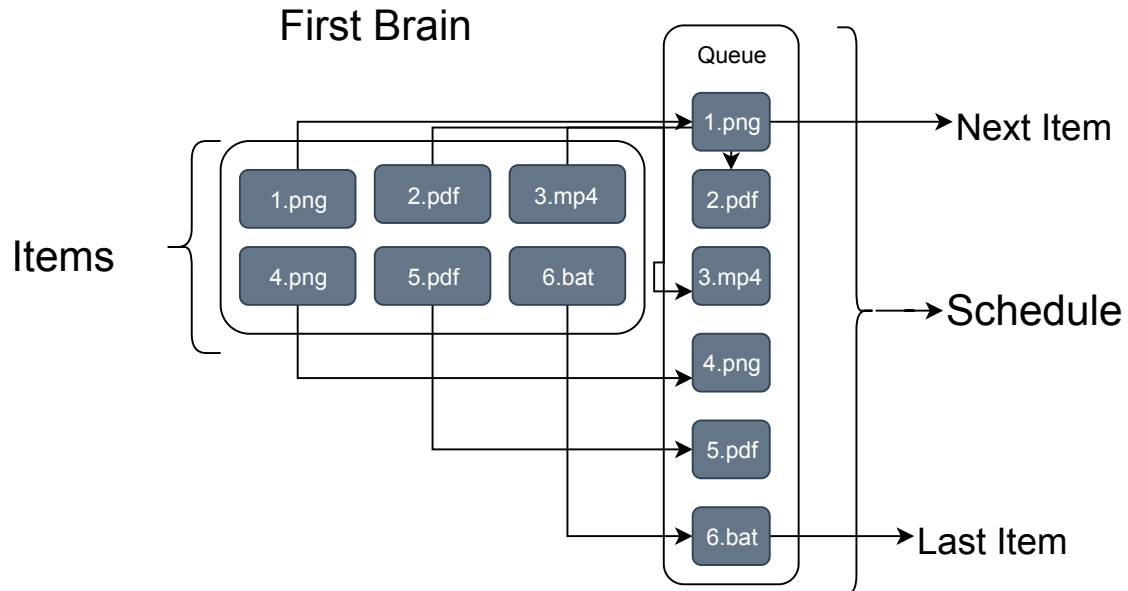
## The 2 Brains

The Vitruvian Framework has the concept of "two brains". The First Brain and the Second Brain.

The First Brain is for everything you want to see in the future by a time schedule. Like Todos, Books, Articles, Prompts, Writing Prompts, Videos, Inspirations, Reminders et.

The Second Brain is for References, wikipedia, Writing, catalog and thinking-trail.

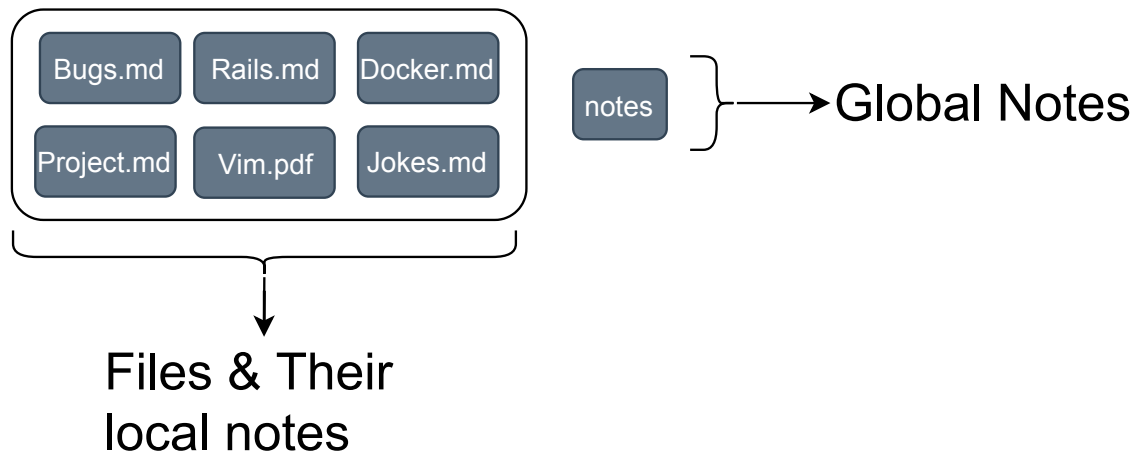
### First Brain



Unless you only plan to work in the future, this is probably the most important one, this is where the "study" part of the framework runs on.

To summarize, the first brain is merely a folder with many files inside of it and a queue to schedule the review of those files. So if you drop a .pdf inside the first brain folder, you'll not review it today, but when the scheduler arranges it for you. This is good for the "stress-less" nature of the study session, you just bind to a keyboard shortcut and press it to study.

## Second Brain



The second brain is more of a wikipedia/reference/catalog/quotes/catalog where you store things that you don't need to be reminded of on a regular basis.

# Fundamentals

## Bullets

Bullets are the "Working Blocks" of a Vitruvian Brain system, bullet are simple lines in the following format:

@id: @whitespace @content

To make "hierarchical tags" use the format #@parent/child e.g #programming/javascript

Those bullets are stored in normal plaintext files with extensions such as ".txt" and ".md" such as:

The *Bullet Id* is the current timestamp including seconds when the bullet was made, this works as a timestamp and also as an local **unique ID**, considering you can't possibly create 2 bullets in the same second. All of the bullets are considered to be unique, if one is malformed, like, without an ID, it'll not try to correct it.

The bullet format follows the following format:

- YYMMDDhhmmss
  - YYYY: Year
  - MM: Month[s]
  - DD: Day[s]
  - hh: Hour[s]
  - mm: Minute[s]
  - ss: Second[s]

All single-digit values will have a leading 0 so: 1 -> 01, 2 -> 02, 9 -> 09 etc.

You might use plain-text for the content of the bullet or the standard markdown flavor, markdown links can be either `[[Technology](https://example.org)]` or `[[Technology]]`. Any embedding, or dynamic content will be implemented on software, this gives enough flexibility for your writing without any specialized program, but remember that IDs are still needed, and when you open an actual software implementation, you'll see the images, videos, sounds rendered from either embedded like `![[Video.mp4]]` etc.

Global bullets are inside a file called **notes** while any local bullet is in your file such as **Javascript.md** or **Docker.md**

Bullets can be in any file such as `Javascript.md` `Journal.md` or `Medicine Summary.md`. But it is assumed you have a "main" file in which most of your bullets will reside in( the default target location ). By default this file is simply called `Vitruvian Brain`.

You can modify the default location for the bullets if you want.

## Incremental Reading

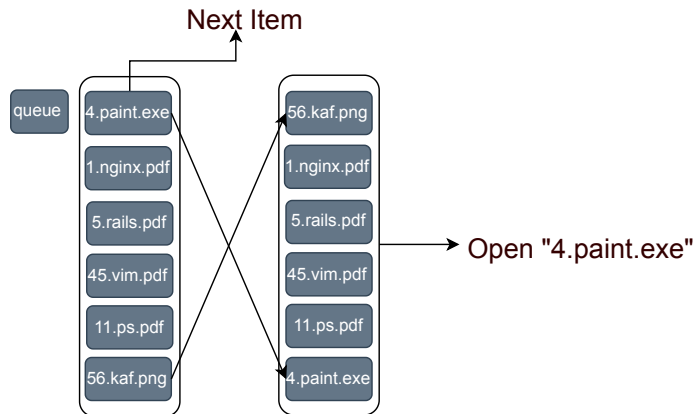
Incremental Reading is like normal PDF reading but instead of processing everything in one go, you "break" the text into smaller extracts and then those extracts will become flashcards. Incremental Reading is a type of Incremental Learning where instead of cramming/blocking to study you instead study in many "small sessions" in order to increase attention, retention and overall reading pleasure.

Vitruvian Brain implements incremental reading with normal files such as ".png", ".pdf", ".epub" etc.

Our Incremental Reading has 2 parts, one is the "first" folder, the other is the "queue" file. The folder will store your articles/books/cartoons/mangas/textbooks/dictionaries while your queue will schedule when those files should be shown in a study session. The "incremental" part here is that by implementing a screenshot system we can crop up a book into many different screenshots and possibly transform those images into something else like a PDF.

### Reading/Reviwing Items:

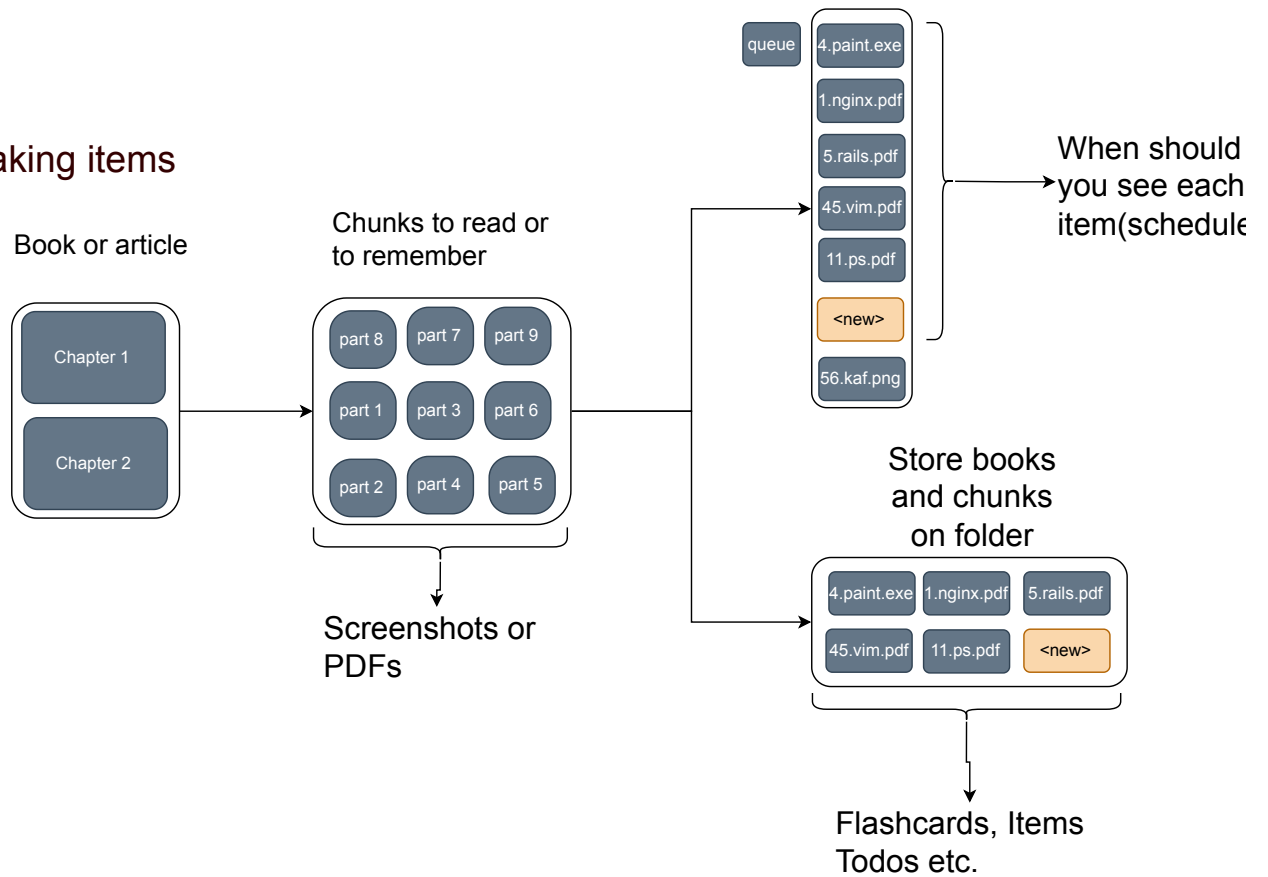
#### Reading / Reviewing Items



### Writing/Making Items:



## Making items



One added benefit is that since it's a "file generic" scheduler you can put anything in there, scripts to start your project, photo editing files to start editing file immediately, a mind map executable for you to work on a certain tree, if it opens then you can use it.

## Spaced Repetition & Scheduling Algorithm

The way the framework define the "spaced repetition algorithm" is different than the commonly implemented like SM-2 or SM-18 is that the way the algorithm usually works is that it requires a "daily drill", or a daily amount of cards you need to review in order for the algorithm to be effective.

This standard uses a "scheduling queue" and not "specific" dates for review in order to avoid the need of having a "daily drill" for several reasons. Such as when you're tired or just forgets to review your flashcards, or when in a specific time when you have enough energy/interest to review around 100 items about programming, you'll only review 10. To summarize, daily drills can be demotivating due to item accumulations and opportunity loss when you wanted to review more items but the strict algorithm would not allow you to.

Depending if the item is being reviewed because of motivation or because of time is important to the algorithm. If it's a time-only review, then a normal exponential scheduling will work just as fine. But when we review an item "Before" its schedule date due to momentum, the scheduling date will not be the same.

It's also important not to "burden" users with accumulating reviews, which might create displeasure in retaining information and that should not be the case, we'll allow for a more forgiving algorithm than Anki's modified SM2 Algorithm.

Which means users might skip a day without being heavily punished, we're here to maximize the pleasure of learning and retention, not punish users for missing a review day.

The **queue(schedule)** is basically a list of items, that is, items, in which they have an "interest point" assigned to them. This "interest point" will define the item's probability, as well as its average frequency. This is due to the fact that this list of items( the queue ) is not static, it's dynamic and it's constantly changing, every time you read an item, the queue will "shift" and item, or remove its first items, and add it to the end of the list. This way, in order to see an item twice( without any modification ) you would need to read every item on the queue in order to see the other item again.

The **queue(schedule)** is a list of your interest that will act in all 3 areas: Consumption, Retention and Application

For the "Consumption" area, this will be for items you want to read, watch or consume in general. This is where most of the learning will take place, since this is where you're consuming new content instead of remembering old content or re-thinking concepts you already know( meditation ).

Some examples of items that are in the "consumption" area are:

- News about an industry you would like to know
- A book on neuroscience
- A blog about how machine learning backpropagation works.
- An infographic you want to integrate into your life
- A 10 seconds video you saw online about how a guy lubricates his joint and you want to incorporate this knowledge, behavior and habit to yourself
- A 7 seconds video where you clipped from an artist performing a keyboard-shortcut on a software program you also happen to use, in order for you to start using the shortcut yourself.
- For the "Retention" area, this is the items you want to have a memory for life, this might be
- Some examples for the "Retention" area are:

A 7 seconds video where you clipped from an artist performing a keyboard-shortcut on a software program you also happen to use, in order for you to start using the shortcut yourself.

Different from the "Spaced Repetition" scheduling algorithm, the "Interest Repetition" scheduling algorithm is not about hard time, or "dates". It's more about ratios and probabilities.

While a "Spaced Repetition algorithm" might schedule your items like: - day(1) -> day(3) -> day(10) -> day(30) -> day(80)

A interest repetition has no concept of "days" or "dates", it has the notion of "rations" and "probability". Since every item in the "Interest Repetition Queue" has a "interest point" which ranges from -99 to 99. -99 means that is has a 99% probability of being "ignored" while reading your queue. while 99 means it has a 99% change of going up one step every time you read your queue.

This means that the queue is "self-regulating". Items will move up and down, being popped and shifted based on their interest point.

Give the exponential nature of the schedule, you might put any file there and it'll open either when you want, or when the scheduler makes time for it. This might be used for painting, music, editing, incremental work, incremental documentation and more.

## Interest Repetition

Interest Repetition is an upgrade to Spaced Repetition, that is, the scheduling algorithm that schedules reviews based on the forgetting curve where each consecutive correct review will schedule the next review exponentially longer. This way, you might review an item around only 5 times in your lifetime(1day, 1week, 1month, 1year, 50years) while retaining the memory forever.

Interest Repetition will take into account the current "mood" of the users while showing and scheduling new cards. Then adjusts the card schedules accordingly. When "raw", it's the same as an "Spaced Repetition" algorithm where the interval between the days in exponential

## Interleaving

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## **Main Files**

- notes
  - Global Bullets, this is the file where most of your bullets will go.
- queue
  - It's the scheduled consumption queue for your interests, it might include things such as: Videos, Audios, images, PDFs, scripts for automation, scripts for plugins etc.
- statistics
  - It's the file with read/write/delete/qpop/qshift/qscan including the time you spend, so you can know what you read, what extract you took, what you deleted, how much time you spend reading, how much time you spend working, how much time you spend on a certain script or plugin, did you spend today more on the topic of "Biology" or "Clinical Pathology" etc.

## **Main Folders**

- first
  - It's the folder for the actual "Interest Repetition" files, such as videos, images, PDFs, HTML, Scripts etc.
- second
  - Second Brain, Zettelkasten, References, Files etc.
- .trash
  - Trash folder



## Folder Structure Example

- brain/
  - notes
  - queue
  - statistics
  - first/
    - 890123.video.mp4
    - 123124.pathology.pdf
    - 380921.pathology.pdf
    - 453809.mtDNA.pdf
    - 32879.human-anatomy.pdf
  - second/
    - 890123.video.mp4
    - 123124.pathology.pdf
    - 380921.pathology.pdf
    - 453809.mtDNA.pdf
    - 32879.human-anatomy.pdf

