

Facet

A modern thinking workspace for Windows

The thinking behind Facet — why capture-first, why categories instead of folders, why multiple views over one body of data.

By the Facet team · getfacet.co.uk · 2026

§ Foreword

This is a long-form companion to the Facet website. The website explains *what* Facet is. This paper explains *why* Facet is the shape that it is.

It is written for two audiences at once. The first is the person evaluating whether Facet is worth their time and money — someone who already keeps notes in half a dozen places, who has tried Notion and OneNote and Obsidian and sticky notes on the monitor bezel, and who is hoping for something that actually fits the way their mind works. The second is the technically minded reader who wants to understand the design choices: the data model, the classification engine, the view system, the rules.

The two audiences overlap more than one would think. People who think carefully about how they organise their information tend to want to know *why* a tool behaves the way it does before they trust it with the contents of their head.

Where useful, the paper compares Facet against the broad landscape of modern personal information tools — without naming names — and against the classic system that inspired its design: **Lotus Agenda**, the 1988 DOS program by Mitch Kapor, Ed Belove and Jerry Kaplan. Facet is not a clone of Agenda. It is an attempt to take the ideas that made Agenda quietly extraordinary, set them down on a modern Windows desktop, and finish the arguments Agenda left unfinished.

§ 1. The problem with the tools we have

Most software for personal information makes a quiet assumption that the user does not notice until it is too late: **you must decide where a thought belongs before you are allowed to write it down.**

Think about what happens when you sit down to capture a new thought in a typical tool today.

In a note-taking app you open a notebook, choose a section, perhaps a sub-section, then a page. By the time you start typing, the thought has either cooled or you have written it into the wrong page and you will forget it is there.

In a task manager you choose a project, a list, a tag, a priority, a due date, sometimes a status — five small decisions to record one thought. The friction is small per item but enormous in aggregate.

In a folder-based system, the choice is even starker. Every file must live in exactly one folder. If a thought belongs to the Pricing project *and* to your conversation with Fred *and* to the things you want to think about on Tuesday, you must pick one. Folders are jealous: they do not share.

In a kanban board, you pick a column. A thought becomes either an idea or a task, either backlog or in-progress, but rarely both, and never four things at once.

The cumulative effect is that the tool is doing the wrong job. Instead of absorbing your thoughts and offering you ways to look at them later, it is **charging an admission fee** at the door: tell me what kind of thing this is, where it should be filed, and what to call it — and *then* you may write it down.

Most people pay the fee a few times, then stop using the tool. The thoughts move to a notebook on the desk, or to a sprawling text file, or — most often — to nowhere at all.

A second problem: the same thought wants to live in many places

A single sentence about pricing might be:

- a **task** (call Fred about it)
- a **note** on the Pricing project
- a record of an **interaction** with Fred Smith
- something that belongs on **next Tuesday**
- a piece of context for the **strategy review** later in the month

A folder-based system forces you to pick one and lose the rest. Tag-based systems do better, but typically treat tags as a flat afterthought rather than as a primary mechanism for retrieval. Database-style systems can model the relationships, but only if you sit down first and *design the schema* — at which point the cost of capturing a thought is no longer five small decisions but fifty.

The result, in practice, is that almost everyone ends up with their information scattered across several tools, each holding a partial view that the others cannot see. The cost is not lost data — most of it is still somewhere. The cost is **lost connection**: the inability to see a single piece of information from all the angles that matter.

A third problem: structure ages badly

The categories that mattered to you in January are not always the categories that matter in November. Projects end. People change roles. A topic you thought was a footnote turns out to be the centre of your work for the next year.

In a tool where structure is decided up front, structural change is expensive. Files have to be moved. Folders have to be renamed. Cross-links break. Tags have to be re-applied. Most users do not bother and the structure quietly silts up — a record of how their world looked the day they set the tool up, rather than how it looks today.

A tool worth its salt should let your structure evolve without dragging your existing data with it. That is hard to do unless the tool was designed around the idea from the start.

§ 2. The Facet idea, in one sentence

Capture first; organise later; look at the same information from as many angles as you need.

Everything else in Facet — the data model, the rules engine, the views, the automations, the storage format — is in service to that sentence.

Let us unpack it.

"Capture first"

The cost of entering an item into Facet is one keystroke and one line of text. You press `Ctrl+Alt+N` from anywhere in Windows — Facet does not even need to be the active window — a small box appears, you type:

Call Fred next Tuesday about pricing

and you press Enter. The box disappears. You go back to what you were doing. Total elapsed time: under three seconds.

No folder was chosen. No project was selected. No tags were typed. No priority was set. No due date was filled in. The thought is captured exactly as it occurred to you. The structure — if any is needed — comes later, and most of it Facet builds for you.

Behind the scenes, several things happened in those three seconds:

- The text was stored as an item.
- The phrase "**next Tuesday**" was recognised as a date and the item was given a due date.
- The word "**Fred**" was matched against the *People > Fred Smith* category and the item was filed under it.
- The word "**pricing**" was matched against the *Projects > Pricing* category and the item was filed under that too.
- Because *Projects > Pricing* has an attached rule that says "any item in this category should also be marked High Priority", the item picked up a High Priority tag on the way through.

The user wrote one line. Facet built five facts.

"Organise later"

A captured item is not finished — it is *available*. Facet's job is to give you tools for putting structure around what you have already captured, not in front of it.

You build that structure incrementally. A category that did not exist yesterday can exist tomorrow, and all the items that mentioned it yesterday can pick it up retroactively the moment its rule is defined. A view that you invent on Thursday gives you a fresh angle on the items you typed in Monday. A rule that you write next month re-files items you typed last week.

This is what we mean when we say **structure follows thought**. The structure of your information is not a cage you build before you put your thoughts inside. It is a scaffold you grow around your thoughts, after they are already there.

"From as many angles as you need"

The third part of the sentence is the most radical. In Facet, a single item is not stored once and looked at once. It is stored once and looked at *many times, simultaneously*, in as many different views as you care to keep open.

The same item about Fred and pricing might right now be visible in:

- a **Calendar** view on next Tuesday
- a **Group-by-Project** view under the Pricing heading
- a **Group-by-Person** view under Fred Smith
- a **High Priority** view in your daily review queue
- a **Matrix** view showing projects against stakeholders, in the Pricing × Fred cell

Tick **Done** on it in any one of those views, and it is done in all five. Edit its text in one, the change appears in the others. Drag it onto a different day in the Calendar view, and the date updates everywhere.

One body of data. Many simultaneous viewing angles. All editable. All live.

This is the design move that separates Facet from almost every other tool in its category. Most personal information tools have a single canonical view of the data — a list, a tree, a board — and offer "filters" or "saved searches" as a secondary feature. Facet inverts that. There is no single canonical view. There is the **database** (a flat sea of items with tags) and there are **views** (saved questions you ask of that database, presented as windows you can lay out as you please).

This inversion has consequences for the rest of the design, which the next sections explain.

§ 3. Categories, not folders

The defining decision in Facet's data model is that items are not filed into folders. They are **tagged with categories**, and an item may carry as many categories as it likes.

This sounds like a small technical difference. It is in fact a different philosophy of what information *is*.

The folder is a metaphor from physical filing

A paper filing system requires folders because paper is a physical object that can only be in one place at a time. The folder metaphor is so deeply embedded in software — folders, files, the desktop, the recycle bin — that most users do not notice it is a metaphor at all.

But thoughts are not physical objects. A single thought can genuinely belong to several places at once, and the choice between them is often arbitrary. Forcing every thought into exactly one container is solving a problem that the medium does not have.

Tags are an improvement, but most tag systems are flat

Many tools have noticed this and added "tags" alongside folders. This helps, but most tag systems are flat: every tag exists at the same level, and you must spell each one correctly every time you want to use it.

A flat tag system scales badly. By the time you have two hundred tags, the list itself is unmanageable — and yet you still need every one of them to find your way back to particular pieces of information.

Categories in Facet are hierarchical and structured

Facet calls its tags **categories** because they do more than a flat tag. Categories live in a tree:

People

- └ Fred Smith
- └ Wendy
- └ John Pell

Projects

- └ Pricing policy
- └ Product launch
- └ Strategy review

Actions

- └ Phone calls
- └ Letters
- └ Meetings
- └ Things to read

Status

- └ Active
- └ Waiting for reply
- └ Blocked

Priority

- └ High
- └ Medium
- └ Low

The tree is not cosmetic. It does real work:

- **Parent categories roll up their children.** Asking "show me everything about People" gives you everything about Fred, Wendy and John, without you having to enumerate them.
- **Categories can have rules.** A category can describe the kinds of items it wants to claim — by text match, by date, by the presence of other tags — and Facet attaches it to matching items automatically.
- **Categories can have typed values.** A category like *Estimate hours* can hold a number per item; a category like *Last contacted* can hold a date; a category like *Status* can hold one of a fixed list of values. Items become not just tagged but **richly described**.
- **Categories can be exclusive.** *Priority > High*, *Priority > Medium* and *Priority > Low* are mutually exclusive: assigning one drops the others. This makes one-of-many choices clean.
- **Categories can describe themselves.** A category for *Status > Waiting for reply* can carry a rule that says "if an item has been in this category for more than five days, move it to *Status > Stale*". The category is not just a label; it is a small piece of behaviour.

The category tree is your personal vocabulary

This is the part most users find surprising once they have lived with the tool for a few weeks. The category tree is not a filing system. It is a **vocabulary** for describing your world.

Each category you add is a word in that vocabulary: a person who matters, a project you are running, a topic you keep coming back to, a status that items move through. As your vocabulary grows, Facet gets better at filing for you: each new item is processed against the rules that hang off each category, and the right tags get attached automatically.

By the time you have lived in Facet for a year, your category tree is a quietly accurate map of what is actually on your mind. Most users have never seen a tool show this back to them.

§ 4. Many views, one truth

If categories are the *vocabulary* of your database, **views** are the *sentences* — the questions you ask of your data, asked in a form you can read.

A view in Facet is a window. You open as many as you like, lay them out across your screen, and each one shows the same underlying items filtered, sorted and presented according to its own configuration. Close a view; reopen it later; the layout is remembered.

The view types

Facet ships with six view layouts. Each is a different geometry for displaying the same data.

Flat list. Items shown as rows with no grouping. The simplest view — useful for short focused queries (everything tagged *Today*, for example).

Due-date buckets. Items grouped into Overdue, Today, This week, Later and No date. The classic daily-review view: glance once, see what is hot.

Group under a category. Pick a parent category. The view shows one section per direct child, with items rolled up from all descendants. So a view grouped under *People* gives you one section per person; a view grouped under *Projects* gives you one section per project. The same items appear under multiple sections if they carry multiple tags — and that is the point.

Matrix. Items as rows; the direct children of a chosen category as columns. Click a cell to toggle a tag on an item. This is the only two-dimensional view in Facet, and it is where you do bulk tagging: "which of these items belongs to which stakeholder?", answered in twelve clicks.

Calendar. A month grid. Items appear on their due date. Drag an item onto a different day to reschedule it. Items with a repeating pattern appear as ghost chips on every future occurrence. The current day is highlighted; overdue items are tinted; items with no date sit in a strip below the grid waiting to be scheduled.

Group by category value. Pick a typed category (Number, Date or Text) and a grain. Number values group into buckets — "1–10", "11–100" — by order of magnitude. Date values group by day, week, month or year. Text values group by exact match. The view is a histogram with the items themselves as the height.

Each layout has its own configuration: sort order, simple filters, custom column choices, optional headings, optional totals. Each view is a **saved query** that you can open, close, duplicate and rearrange. The view is not the data; the data is the database.

Views are editing surfaces, not reports

This is the move Lotus Agenda made in 1988 that most tools still have not copied. **A view is not just a way of looking at the data. It is a way of editing it.**

If you drag an item from the *Pricing* section of a project-grouped view into the *Strategy* section, the item's *Project* tag changes from Pricing to Strategy. If you drop an item onto a different day in the calendar, its due date moves. If you add a new item *inside* a section of a category-grouped view, it inherits that section's category automatically. If you tick a checkbox in a matrix cell, that tag is added to that item.

You do not have to switch back to the "real" view of your data to edit it. The view *is* a real view of your data, and every change you make propagates everywhere else immediately.

The "spreadsheet for words"

Agenda's designers used the phrase "**a spreadsheet for words**" to describe what they were building. A spreadsheet lets you model numbers flexibly without writing a formal program: cells hold values, formulas recompute, and you can rearrange the layout without affecting the underlying numbers.

Facet aims to do the same for textual information. The items are values. The categories are formulas-with-text — patterns that match, rules that fire, tags that propagate. The views are reports — different presentations of the same calculation. Edit a value in one report and the others recompute.

This is the deepest reason why Facet is shaped the way it is. **A tool for thought should let you re-present the same thoughts as easily as a spreadsheet lets you re-present the same numbers.** That is the bar.

§ 5. Rules, not workflows

Most tools that offer automation give you a workflow editor: a tree of triggers and actions, often with a visual node graph, in which you build explicit pipelines that fire when something happens.

Workflow editors are powerful but they are also a separate program living inside your information tool. They need their own mental model. They break when you change the underlying data. They tend to be authored once, then forgotten about.

Facet takes a different tack, borrowed directly from Agenda's most overlooked innovation: **rules live on categories**, and an item is processed through the category hierarchy each time it is created or edited.

The condition / action model

Every category in Facet can carry two things:

- **Conditions** — descriptions of items the category wants to claim. A string condition matches text in the item; a category condition tests whether the item is already tagged with another category; a date condition tests the item's due date or end date; a value condition tests a typed category's number, date or text value.
- **Actions** — things to do once an item is in this category. Add another tag; remove a tag; set a value; set a due date; mark as done; set a priority; discard the item entirely.

When you add or edit an item, Facet walks the category tree once. At each category, it evaluates the conditions. If they match, it adds the category to the item, then runs the actions. The actions can trigger further category matches downstream. The item arrives at the bottom of the tree fully classified.

This is **declarative classification**. You do not write a procedure that says "when an item comes in, do X, then Y, then Z". You write a *description* of each category — "items in this category mention Fred, or are tagged with Strategy Review, or have a date in the next seven days" — and Facet works out which categories to apply to each item.

Why this matters

The declarative approach has three properties that workflow editors struggle to match.

Rules compose. A single item passing through the tree may pick up five tags from five different category rules. You did not have to anticipate the combination. Each rule is local; the global behaviour emerges.

Rules apply retroactively. Write a rule today and it processes the items you wrote last year. You do not have to re-run anything.

Rules are inspectable. Each category's rules sit on the category itself, in plain controls — not buried in a separate automation configuration screen. Click the category, see the rules. Change them. The next time any item is touched, the new rules apply.

Automations: the imperative layer

Sometimes you do want the imperative form. You want to say "every Monday morning at nine, create an item that says 'weekly review'", or "when I press F5, open my three most-used views in a cascade". For that, Facet has **automations**: named sequences of actions you trigger by hand, by keyboard shortcut, by schedule, or when something happens.

Automations sit alongside rules, not inside them. The rules engine does declarative classification. The automation engine does imperative workflows. The two share their action vocabulary — anything an automation can do, a rule action can do — but they are distinct facilities for distinct jobs.

This separation matters. Most tools collapse the two into a single "automations" feature, with the result that simple classification problems become impenetrable workflows. Facet keeps the simple thing simple and reserves the expressive thing for when you actually need it.

§ 6. The four primitives, briefly

A reader who has made it this far has earned a compact summary of what Facet actually is, under the hood. Here it is.

Facet's data model has **four primitives**, and that is all.

Item

A single piece of textual information. Has a heading (a line of text), an optional due date, an optional end date, an optional repeat pattern, an optional long-form Markdown notes body, an optional list of attached file or web links, and a set of attached categories with optional typed values.

That is the whole item. There is no "type" field. There is no "this is a task versus a note" toggle at the data level — a single tick on the item ("treat as a note") hides the Done checkbox if you want to use it as reference rather than as a task, but the underlying record is identical.

Category

A node in the category tree. Has a name, an optional colour, an optional typed value (Number, Date or Text), an optional list of allowed values (for the typed kind), optional rules (conditions and actions), optional exclusivity with sibling categories, optional aliases for text matching, and an optional "items here are notes by default" flag.

Categories are nested. A category can be both a tag and a parent of other tags simultaneously.

View

A saved question about the database. Has a name, a layout (one of the six listed earlier), a configuration appropriate to its layout (which category to group under, which columns to show in a matrix, which date range for a calendar, what to sort by, what to filter on), and a stored window position so it reopens where you left it.

Views are independent of each other. Closing one does not affect any other.

Automation (and Rule)

A named sequence of actions. Has a trigger (manual, hotkey, scheduled, or hook), an optional set of conditions, and an ordered list of actions to perform.

A **rule** is the special case of an automation that is attached to a category and fires automatically on item add/edit. The vocabulary of actions is shared. Anything you can do in an automation, you can do in a rule.

Everything else is built from these four

Calendars are views. Daily review queues are views. Stakeholder matrices are views. Tag inheritance is categories. "Send to inbox" is an automation. "Mark stale after five days" is a rule. "Weekly review on Monday morning" is a scheduled automation.

The whole application is the four primitives, composed.

§ 7. Local, private, portable

A tool you trust with the contents of your head should not phone home.

This is not a feature in the marketing sense — it is a default. Facet does not have a server component. It does not have an account system. It does not have telemetry, analytics, crash reporting, sync, or "intelligence" that requires uploading your data to anyone. There is no backing service that could be turned off.

One file, on your own machine

A Facet database is one file: `db.json`. You choose where it lives. You can put it on your desktop, in your Documents folder, on a USB stick, in a cloud-synced folder if you want sync of your own choosing, or in a Git repository if you want history. You can copy it, back it up, version it, inspect it in a text editor.

The format is JSON. It is not encrypted. It is not obfuscated. It is yours, in a form you can read, parse and migrate without Facet's help if you ever need to.

Installed or portable, same binary

The same `facet.exe` runs as a normal installed Windows application or as a portable application from a folder you carry on a USB stick. The portable mode writes nothing outside its own folder. The installed mode stores settings under `%AppData%\Facet`. The choice is yours; you can switch back and forth.

Why this matters commercially

Most of the personal information tools released in the last decade are SaaS products. They are pleasant to use until you cannot pay the subscription, until the company is acquired, until the product is deprecated, until your data is held behind a login that has stopped working. Personal information is the worst possible data to lose access to, because nobody else has a copy.

Facet is a desktop application. You buy it once. You install it. It runs forever on the machine you installed it on. When you upgrade Windows, your data is the same file in the same place. When the developer is on holiday, your data is still on your machine. When you stop paying for anything, you still have everything you ever wrote.

This is not a nostalgic preference. It is the only sane default for a tool whose job is to hold the contents of your head.

§ 8. The view from the user's chair

Architecture aside, what does Facet actually feel like to use?

Day one

You install the app. You open it. There is a blank database and an empty quick-entry box. You type a line. You press Enter. The item appears in a list. You type another line. Another item appears.

By the end of the first hour you have perhaps twenty items in the database. None of them are tagged. There are no categories. There are no rules. The tool is doing nothing clever — it is acting as a list.

This is fine. Facet at this stage is no worse than any other note-taker. It is also no better. The difference will come from what happens next.

Week one

You notice that several items mention the same project. You create a category for that project. You attach a simple rule: "any item whose text contains the word *Pricing* belongs to this category". The next time you type a pricing-related item, it picks the tag up automatically. You remember that you wrote a few items about pricing in the last few days; you rerun the rules; those items pick the tag up too.

You repeat the trick for two or three other projects, half a dozen people, a couple of recurring topics. By the end of the first week you have perhaps a dozen categories and a hundred items. Around three quarters of the items are tagged correctly without you having lifted a finger. You tag the rest by hand in about five minutes.

You open a view grouped under *People*. You see, for the first time, everything you have written about Fred in one place. Some of it is task flavoured, some of it is note flavoured, some of it is a half-remembered remark. All of it is there.

This is the moment most users say something has changed.

Month one

The category tree has grown. There are people, projects, recurring topics, statuses, priorities. A few of the categories carry typed values — estimates, due dates, contact dates. Several categories have rules that fire automatically.

You have a handful of views laid out the way you like them. A calendar on the right of the screen for date-driven work. A daily-review view on the left, filtered to high-priority items due this week. A project-grouped view that you open when you want to see the shape of your work, and a person-grouped view that you open before any meeting.

You have built two or three automations. One creates a weekly-review item every Monday morning. One fires when you press F8 — it cascades your three favourite views and brings the calendar to the front. One clears all items tagged *Done* into an archive at the end of each quarter.

You have stopped using sticky notes.

Year one

The database has perhaps a few thousand items. The category tree is recognisably a map of the year — the projects you started, the people you worked with, the topics you returned to. You can find anything in seconds.

When you change role, change project, or change focus, the structure adapts. You add categories for the new world; the old ones do not have to go anywhere. The information you wrote last year is still searchable, still tagged, still re-projectable into any view you care to build.

Facet's compounding value is the second-year-onwards effect. The first year, it is a pleasant note-taker. The second year, it is the only record you have of what you actually did.

§ 9. Who Facet is for — and who it isn't

We are not going to pretend Facet is for everyone. It is a tool with a particular shape, and that shape suits some people and not others.

Facet is for you if

- You think in **projects, people, topics, dates and priorities**, and you want a tool that respects that those categories overlap.
- You generate **a lot of small textual fragments** during the day — ideas, reminders, notes from calls, things to follow up on — and you want them all in one place without having to file them.
- You like the idea of a **single local file** holding your data, and you have no desire to put it in someone else's cloud.
- You are willing to **invest a little time** building up a personal category vocabulary in exchange for a tool that files things for you thereafter.
- You miss tools that **respected your intelligence** instead of trying to be friendly.

Facet is probably not for you if

- You want a **collaborative workspace** with shared documents and real-time editing. Facet is a single-user tool. It is for the contents of *one* head.
- You want a **rich-document editor** with embedded images, tables, drawing tools and so on. Facet is primarily textual. Notes use Markdown; attached files are linked, not embedded.
- You want **mobile sync, web access** or a **cross-platform client**. Facet is a Windows desktop application. If your work is primarily on a phone, this is not the tool.
- You want a **CRM**, a **project management tool**, a **PM-style ticket tracker** or a **wiki**. Facet is none of these. It can be *adjacent* to all of them — many people will keep their meeting notes and follow-ups in Facet while doing the formal work in another tool — but it does not try to be any of them.

The honest pitch is this: Facet is a *thinking* tool. It is for the intermediate layer between "thought in your head" and "thing recorded in a system of record". For people whose job involves a lot of that intermediate layer — and there are more of us than the software market seems to think — it is a quiet, deep pleasure to use.

§ 10. A brief history of the idea

Facet's core ideas are not new. They are largely Mitch Kapor's, by way of his collaborators Ed Belove and Jerry Kaplan at Lotus Development in the late 1980s.

Lotus Agenda, released for DOS in 1988, was built around the observation that personal information is short, messy, evolving and idiosyncratic — and that databases at the time were the wrong tool for it because they required the structure to be designed before the data could be entered.

Agenda's designers wrote about "a spreadsheet for words": a flexible, user-shaped environment for organising real-world information. Items could be entered freely; categories could be created as needed; an item could belong to many categories at once; views were spreadsheet-like grids that re-projected the same data into different working arrangements; and most strikingly, a **declarative rule system** lived on the category hierarchy and processed each item through it automatically.

Agenda sold about 200,000 copies — respectable but not a hit. Lotus discontinued it in the early 1990s and replaced it with the more familiar paper-organisier metaphor of Lotus Organizer. The deeper idea went largely unfollowed for thirty years. Tools that picked up parts of it — tags, smart folders, saved searches, rule-driven classification — rarely picked up all of it.

Facet picks up all of it. The view geometry, the typed category values, the unified action vocabulary across rules and automations, the single-file local storage, the Markdown notes, the native Windows window management, the global capture hotkey — these are modern. But the underlying philosophy of capture-first, multi-classification and view-as-edit is straight out of the 1990 *Communications of the ACM* paper that the Agenda team wrote.

We acknowledge the debt openly. Facet is independent of Lotus, IBM and HCL — we are not affiliated with, endorsed by or sponsored by any of them — but the inspiration is no secret. This is what Agenda would have looked like if it had been allowed to grow up.

§ 11. What Facet is not

A short section, but a necessary one, because positioning a tool is as much about saying what it does *not* do as what it does.

Facet is not a cloud product. There is no server. There is no account. Your data does not leave your machine unless you put it somewhere yourself.

Facet is not a subscription. You buy it once. You own that copy. Updates within the current major version are free. Future major versions, if any, will be priced separately and you will not be forced into them.

Facet is not a clone of Agenda. Agenda was a 1988 DOS program. Facet is a 2026 native Windows application written from scratch. It honours the design philosophy, not the screens.

Facet is not a project management tool. It does not have Gantt charts. It does not have dependencies between items. It does not have team membership. If you want to manage a team, use a team tool. Facet manages *your* end of working with that team.

Facet is not a note-taking app. Notes are part of Facet — every item can have a long-form Markdown body — but notes are not the centre. The centre is the *item*: a short, tagged, possibly dated piece of information. The notes hang off the item, not the other way round.

Facet is not a wiki. There are no inter-document links in the hypertext sense. Items relate to each other through *shared categories*, not through embedded links. (You can put Markdown links in notes if you want them, but it is not the primary navigation mechanism.)

Facet is not "yet another second brain". That phrase has become a genre and Facet does not fit it. The category-and-view model is a working tool for current information, not a personal knowledge garden for ideas you might one day return to. If your information lives mostly in long essays you want to interconnect, you want a different tool.

What Facet *is* is in section 6 of this paper. Four primitives. Composed.

§ 12. The road from here

Facet 1.0 is the foundation. The data model, the category tree, the six view layouts, the declarative rules engine, the automation engine, the quick capture, the calendar view, the matrix view, the Markdown notes support, the file-attachment support, the global hotkey, the local-file storage, the portable mode — these are all in the box today.

There are deliberate omissions. There is no native mobile client. There is no built-in sync. There is no shared-team mode. There is no AI assistant for filing or summarising your items. There is no rich-text editor for notes — Markdown only. None of these are oversights; they are choices about where the line is.

Some of them will move over time. Sync via your own cloud folder is already supported (use any folder-syncing service you trust). A read-only mobile companion is on the long-term thinking list. AI features — if and when they appear — will be optional, local where possible, and explicitly opt-in for anything that leaves your machine.

What will not move is the centre. **Capture first; organise later; multiple views over one body of data, all editable.** Anything that violates those three constraints is not a feature; it is a different product. Facet will stay Facet.

§ 13. Trying Facet

Facet is sold from **getfacet.co.uk** as a single one-off purchase. You get the installer, the portable build, the user guide and this paper.

The same binary runs in both installed and portable modes. There is a trial that lets you exercise the full feature set for a generous evaluation period; once you have decided, the licence unlocks the same binary without re-downloading anything.

There is no commitment beyond the one-off purchase. There is no subscription. There is no account. There is no telemetry. The licence is yours, the software is yours, the database file is yours.

If Facet is for you, you will know inside a week. If it is not, write to us within thirty days of purchase and you will be refunded, no explanation required.

§ 14. A closing thought

The personal computer was sold to us, originally, as a thinking tool. A universal machine that would extend the user's mind. Most of what we actually run on it has drifted away from that promise — into platforms, feeds, dashboards and notification streams that compete for our attention rather than serve it.

The thought behind Facet is that there is still room — on the desktop, locally, privately, with no subscription and no cloud — for software that is genuinely on the user's side. Software whose only job is to hold what you give it, hand it back the way you want to see it, and stay out of your way.

That is what Facet is. Capture first. Organise later. Many views over one body of data, all of them live, all of them yours.

We hope you will try it.

Facet is an independent product and is not affiliated with, endorsed by, or sponsored by Lotus, IBM or HCL. Lotus Agenda is referenced here only as the historical influence on the design philosophy of personal information management software.

© 2026 Facet · Built in the UK · getfacet.co.uk · sales@getfacet.co.uk