

A description of the UI implementation within Neorg's GTD

Most Important Design Goals

- ZERO FRICTION - user interfaces should provide as little friction as possible when capturing a note of any kind.
- Fast response times - the interface should be very responsive and database operations should not be felt.
- Dynamic data - notes and ideas should be representable in many formats - the form of a list, a calendar, a graph.
- Ensure that migrating from "small" to "larger" thoughts is unobtrusive.
- Provide several calendar views and graphs to visualize data.
- All data should be stored in plain-text `.norg` files, such that it can easily be version controlled and sent across devices.
- Understand the nuances and pain points of other note taking solutions (points back to the "zero friction" bullet point).

Step 1: Capturing

The capturing process involves having an idea or new task and writing it down somewhere where it can be reviewed later.

An important thing to note is that the process of capturing **does not necessarily imply** the process of annotating. Annotating is adding metadata like contexts, timeframes, associations (unique to Neorg) and projects. Those are part of the Next Actions ontology, and are discussed later. They *can* be also part of the capturing process (something we call **preliminary information**), but for most cases they are not a requirement.

The capture mechanism is bound to `<LocalLeader><LocalLeader>` by default. It can be executed on *any* file, including non-norg files.

The User Interface

The user interface for capturing a note should be hilariously simple at first glance. All the user should see is a text box with a prompt to press ? for more information. The expanded information should include the basic capture syntax.

The most basic capture syntax includes:

- Regular text - the normal content of the captured note.

- `#work` - a context, some circumstance that you can attach to the note itself.
- `@jeremy` - an association, an entity that is bound to the note in some way.
- `!1` - urgency, starting from 1 to infinity. One is the most urgent.
- `&tickler` - binds the content of your note to a specific ontology alongside the default `&inbox` ontology.
- `~1{y[ear]|mo[nth]|w[EEK]|d[ay]|h[our]|m[inute]}` - the estimated amount of time you expect the task to take you. The content in square brackets is optional. Recommended to give yourself a rough estimate without setting a specific due date.

Default Captured Information

When pressing the "capture" keybind, a default collection of information should be collected and stored alongside the note, ready for the user to see when they are going through the processing phase.

This set of information includes:

- The exact date (down to the minute) the note was captured. Seconds should be omitted, as we would like to minimize cognitive overhead.
- The name of the file and the line number where you got the idea - this should in most cases help as a context where you got your "aha" moment.

Extra Captured Information

The user should be able to highlight some text in their current buffer before pressing the capture keybind to attach the text (with appropriate syntax highlighting) to the note.

The User Experience

Punctuation tends to be a serious giveaway to the "tone" of a note. It could be sensible for Neorg to notice that a note ends with a question mark, and therefore should by default put the note inside the `maybe/someday` ontology. If the user during the processing phase changes their mind and does not decide to throw the item into the `someday` ontology then the item can be happily pulled from under the rug.

Interesting Details

If a note contains preliminary information, it will first be put into the appropriate container as a "volatile" task. It will behave just like any other task or idea, but it will *still* be presented to the user during the processing phase. It's during the processing phase that the user can then fully commit to their initial

decisions or alter the metadata of the task to repurpose it and move the task to a different container.

Other Capturing Mechanisms

Apart from this "quick capture", it would also make complete sense to support whole file captures, external resource captures (URLs, PDFs), etc. each with their own UIs.

Ontologies

Before any further explanations are made it's important to understand *ontologies* as a concept. Think of an ontology as a container with some distinct properties. For starters, a single ontology contains any number of elements of the same type. The type consists of the following:

- A text field (required)
- A **derives** clause (optional) - this merges the current ontology with another template ontology. Useful to derive e.g. a **calendar** ontology which will force you to provide all of the data necessary to display a calendar.
- A **display** clause (required) - how to display the data in the ontology, can be one of **list**, **graph**, **calendar**, **table**.
- Any set of keywords: **key** : **Date|Text|File**. Some keywords can be marked as required, while others optional.

In Neorg, **every GTD list is simply an ontology**. This includes the inbox, the next actions, tickler file, calendar, you name it. Neorg binds your captured task to an ontology under the hood. Moving a task from the inbox to the next actions list is equivalent to changing the ontology from **&inbox** to **&next-actions**, and letting the type system verify if all variables and data are in the right place.

The Power of Data

Given ontologies are just generic containers, their data can be bent at the user's will. Data can be displayed as tables, lists, calendars and others given the correct queries.

This means that, unlike GTD's lists, ontologies are not as tightly intertwined and are more flexible as a result. This means that any specific list (e.g. the calendar) can be duplicated or its output modified (you can have several different calendars which are just different ontologies with their **displays** set to **calendar**).

The GTD implementation in Neorg works because Neorg provides a **default set of ontologies** (with appropriate names like **&inbox**, **&calendar** etc.) crafted to mimic the GTD workflow in the most efficient way possible.

Processing

Processing is a very important part of your workflow. It involves going through each capture in the inbox and deciding what the next course of action for that task should be. You perform processing whenever you have some spare time to do some side tasks.

Processing View

There are a few ways you can process tasks, these are "one by one", "list view" and "calendar view":

- One by one - self explanatory, displays one task at a time, allowing you to make a conscious decision for each task. Packed with one-key keybinds for most tasks (send to maybe/someday ontology, send to tickler file, archive).
- List view - for when you know you had some important things to do but do not want to sift through every SINGLE capture you've made. Most urgent tasks should be displayed in red.
- Calendar view - since tasks are first sorted by urgency and then date created (ascending), it is also possible to display your thoughts in the form of a calendar. Especially useful if your tasks are often time sensitive.

The order in which tasks are presented should be:

- Most urgent first, or
- Oldest first

Types of Actions

Depending on what you would like to do with the task you may opt for a set of different actions:

- Add to the maybe/someday ontology, e.g. "learn Ithkuil".
- Convert to an **action** and move to Next Actions. By **action** we mean an actual PHYSICAL thing you can do, instead of a hypothetical. E.g. "plan cake lottery" -> "e-mail Arthur and Camille and remind them to bake their cakes".
- If the task can be done in 2 minutes, DO NOW - the editor will enter a state of "doing a task", and Neorg will remember across sessions that a task is still in the process of being done.
- Move to Next Actions.
- Delegate to a timeframe/date - this will open up the calendar view with all your other tasks and allow you to select when you want to assign the task to be done.

- Delegate to "waiting for" ontology - if your task is being blocked by another person or task you may move it to the "waiting for" ontology. If you have your associations and tasks set up correctly Neorg will also be able to auto-detect when this task is no longer being "blocked".
- Move to a project (if the project doesn't exist, it will be created, else the task will be appended to the end of the project).
- Archive the task.
- Place the task in the trash.

"Waiting for"

Talk about automatic unblocking of a task if your associations and tasks are set up correctly. The unblocked task is moved back into the inbox.

Also talk about how waiting for's can have expected unblock dates (expected due dates) as well as expected start dates just like a regular task. Neorg will then notify you if that task is overdue or was done faster than you expected (useful if you want Neorg to tell you that coworker Jimmy has been slacking on the side project for the past 2 weeks now).

UI/UX

- Each type of action should have a single mnemonic keybind bound to it regardless of the processing view.
- The default view should be a popup box in the middle of the screen with basic actions. Pressing ?<action-button> should display help related to that action.
- The user should additionally be able to press the u button to undo the last action and place the task back into the inbox.

Common Metadata Operations

When processing your tasks you should also verify preliminary metadata or assign metadata to your tasks.

During this stage, you may press any of the keybinds to apply any metadata. Press C to add a space separated list of contexts, and press <CR> to apply. Press A to assign the note to a specific association.

Contrary to other GTD implementations, the user should also be able to press **Es** (estimated start date) or **Ed** (estimated due date) to set a fuzzy date for when they expect the project to be complete or started. Not every task has a strict deadline, and you definitely do not want to see tasks that actually **do** have a deadline shown with the same urgency as tasks that only have an estimated deadline.

CONTEXTS vs ASSOCIATIONS A *context* is a general tag that you apply to your note - this includes the tools you might need to complete the task, where the task needs to be completed, or a category that your task falls under.

An *association* is just like a context but it has **permanent metadata** attached to it. This is especially useful for people or locations. You may create an association to a coworker (think of it as a profile) with their name and surname as permanent metadata. Then, when you're in a meeting with them, you may bring up your list of associations with that person to see what you wanted to discuss with them.

The Next Actions

The Next Actions is the list of items you visit when you're **going** to get things done. Instead of hypotheticals or things that need to be expanded on, you go through each item in the Next Actions ontology and get each item done. This ontology displays the general items of the Next Actions as well as tasks with upcoming due dates. By default, all items in a Next Actions have an urgency of 10. Items that are part of the calendar (with marked due dates) have their urgency equal to the amount of days left for the task (remember, 1 is the most urgent).

The Next Actions is displayed in whole, allowing you to choose what you would like to work on first. Items with the highest urgency are displayed first, with every other item sorted by the estimated time that task will take, otherwise by date created.

Facilities for Time Management

When you press <CR> to begin working on a task, Neorg enters a "task pursuit" mode. It will provide live information about the due date, the time spent daily on the task, completion rate (if you are working on a project). The `:Neorg task` command should display this information in a single popup, with the following abilities:

- Mark the task as complete.
- Temporarily pause - pause the current task (to go eat lunch, sleep, or perform other intermediary activities). Closing all Neovim/client instances will automatically signal to the GTD process to pause the task.
- Cancel the task (with a reason) - equivalent of moving to the archive.
- "Change your mind" - move to the `maybe/someday` ontology.
- Postpone the task (to a different date).
- Block the task (move to the "waiting for" ontology).

Pomodoro Integration

This wouldn't be the job of GTD itself, but a pomodoro plugin could hook into the currently active task, provide metrics like "if you spend 40 minutes a day on this task it should be complete by x" and the usual pomodoro features.

Statusline Integration

Neorg should provide statusline functions to display GTD progress, tasks left and other important bits of information.

Maybe/someday ontology

The m/s ontology is for thoughts or ideas that you are unsure about or do not have time to currently pursue. They do not clog up your Next Actions nor the inbox, allowing you to keep the things you actually have to do clean.

Items are moved here during the capturing phase or during the processing phase. The contents of this ontology is reviewed ideally at the end of each week during the review process, ensuring all of your cool ideas are still under your consideration.

Contrary to other GTD implementations, the m/s ontology **also allows you to have projects within it**. It's common to have an idea or concept that you might do someday, but you would like to have planned out in advance. It's also an interesting way of procrastinating I guess :p

After a task receives a defined or estimated due/start date it will then be moved into your regular `&projects` ontology where you can proceed to work on the project.

Archive

TODO.

Trigger List

A trigger list is a list of general errands that you do frequently. You go through the list and think to yourself whether you have forgotten about anything from that given topic. Neorg will provide three different trigger lists by default: a personal trigger list, a student trigger list and a work trigger list.

Below is an example snippet of a personal trigger list. If you ever remember anything about the presented topic capture it to the inbox as soon as possible:

- Projects started, not completed
- Projects that need to be started

- Commitments/promises to others
- Significant Other
- Family
- Friends
- Classmates
- Borrowed items
- Projects: other organizations
- Service
- Civic
- Volunteer
- Communications to make/get
- Family
- Friends
- Professional
- Initiate or respond to:
- Phone calls
- Text messages
- Voicemail
- E-mail
- Snail mail
- Social Media
- Upcoming events
- Special occasions
- Birthdays
- Anniversaries
- Weddings
- Graduations
- Holidays
- Travel
- Weekend trips
- Vacations
- Social events
- Cultural events
- Sporting events
- Things to do
- Places to go
- People to meet/invite
- Local attractions
- Administration
- Financial
- Bills
- Banks
- Investments
- Loans
- Taxes

User Experience

Neorg will display trigger lists in a UI that will allow you to immediately add captures, closing the gap and making the mental model of a trigger list easier to deal with.

Review Process

TODO.

Motivating the User

Neorg may opt to urge the user to get their tasks done every now and again (obviously under a configuration option). When the user has a few tasks to sift through, let them know they have not a lot left to go. If the user has a lot of tasks in their inbox, urge them by saying that sifting through just a few captures per day will quickly amount to serious progress. Do not let the user's tasks feel stale.

Displays

Tasks are sometimes difficult to visualize, especially once they grow greatly. Below are some concepts for making the visualization of tasks easier.

Display All Tasks Related to a Context

The following display could show all of the tasks that are related to a given context, a set of contexts (A or B) or a union of contexts (A and B) in the form of a graph of connections.

Display Task Dependencies

The following display would be shown in the form of a mindmap. Projects would branch out into tasks, and tasks that depend on other tasks would be marked with a one-way arrow.

General Considerations

We should absolutely establish a single ruleset for how we set our keybinds. Currently they are a bit over the place.