ChatGPT

Omnivore Design System Overhaul Proposal

Executive Summary

Omnivore is an open-source, read-it-later application aimed at power users (researchers, avid readers) as well as casual users. The current design has strong foundations – a distraction-free reader, robust tagging, and dark theme – but it also faces **UX challenges** in balancing information density with clarity, making actions intuitive (especially on mobile), and ensuring consistency and accessibility across the app.

Goals: This proposal outlines a comprehensive design system overhaul to create a **beautiful, functional, and scalable UI**. Key improvements include:

- **Enhanced Visual Hierarchy:** Clear typography and spacing standards to emphasize titles and critical metadata, while de-emphasizing secondary info.
- **Density Controls:** Allow users to switch between compact, comfortable, and spacious views, ensuring power users can scan quickly without overwhelming casual readers [1].
- **Intuitive Actions:** Maintain a clean interface by revealing action buttons on hover or via context menu, with mobile-friendly swipe gestures (implemented carefully to avoid discoverability issues [2]).
- **Improved Labels & Tags:** Visually distinguish system labels (icons for favorites, feeds, etc.) from user-created tags (colored text chips) to reduce confusion.
- **Batch Operations:** Streamline multi-select mode with persistent selections and a floating actions toolbar, enabling efficient batch archiving, tagging, or deleting of items.
- **Progress & State Indicators:** Use subtle progress bars with optional percentage or time indicators for reading progress, and add clear badges or states for “processing”, “completed”, and “archived” items for better feedback.
- **Accessibility & Consistency:** Adhere to WCAG 2.1 AA standards (sufficient color contrast, focus indicators, ARIA labels) and ensure a consistent experience across desktop and mobile (responsive layouts, touch-friendly targets).

By addressing these areas, Omnivore’s interface will serve the needs of knowledge workers and casual readers alike – **information-dense yet elegant**, powerful in features yet easy to use.

Product Context & User Needs

Omnivore Overview: Omnivore lets users save articles, PDFs, and RSS feed items for later reading. It offers full-text search, highlighting/annotation, and sync across devices. Unlike competitors (Pocket, Instapaper, Raindrop.io, Matter), Omnivore is open-source and privacy-focused, targeting users who value control and powerful organization features.

Target Audiences:

- **Knowledge Workers (Researchers, Students):** Save and categorize large volumes of articles. They need efficient organization (labels, folders) and quick retrieval via search.
- **Avid Readers:** Might save 10–100 articles per week. They prioritize a fast interface to triage and read content regularly, often across devices.
- **Casual Users:** Save a few articles per week for leisure. They prefer a simple, uncluttered UI without a steep learning curve.

Key Use Cases:

- **Rapid Triage:** Scanning through a library of 100+ unread items to decide what to read now, later, or archive. The UI must support quick scanning of titles, sources, and maybe topics at a glance, with easy batch actions for organizing.
- **Focused Reading Mode:** Transitioning from the list to a distraction-free reader. The interface chrome should fade away during reading, with clear progress indicators and an easy way to return to the library.
- **Organizing by Topic:** Using folders or labels to group articles (e.g. by research project or interest area). Users should be able to tag or move multiple items, and filter or search by those tags effortlessly.
- **Cross-Device Consistency:** A user might save articles on a phone and read on a desktop. The design should be responsive, with touch-friendly interactions on mobile (since hover doesn't exist on touch), and a consistent visual language so users feel at home on any device.

Current Design Strengths & Gaps

Strengths of the Legacy Design

- **Hover-Action Card Pattern:** On desktop, each article card shows a minimal icon-only action bar (read, 📦 archive, 🏷 label, original link, 🗑 delete) on hover. This keeps the list visually clean while still providing quick actions when needed. It's a familiar pattern that **reduces clutter** and keeps cards compact.
- **Label Chips:** User-created labels are shown as small color-coded chips (e.g. ● **Project X**). The use of a colored dot and text provides a **visual categorization** at a glance, and the consistent small font (11px) and subtle rounded rectangle ensure labels don't overpower the title.
- **Card Information Hierarchy:** The card layout prioritizes critical info:
- A compact metadata line (with icons for flair like ☆ favorite or newsletter, plus timestamp and reading time) at the top – small and muted.
- A 1–2 line **title** in larger, bold text for the article name.
- Secondary info like author and source in regular small text.
- Then any user labels/tags at the bottom.
- An unobtrusive progress bar at the very bottom if the article has been started.

This structure ensures titles stand out, and additional info is available but not in the way. Titles are clamped to 2 lines to avoid overly tall cards. - **Dual Layout Modes:** Users can switch between a grid view (cards ~400px wide, with a 150px thumbnail and metadata below) and a list view (smaller thumbnail to the left, title and info to the right). This flexibility accommodates different browsing styles – grid for visual browsing, list for density. - **Dark Theme Foundation:** The app uses a dark theme by default (#1a1a1a background,

#2a2a2a surface cards). This is great for reducing eye strain during long reading sessions and gives a modern, focused aesthetic. Contrast is handled consistently (light text on dark backgrounds), aligning with many users' preference for dark mode when reading.

Gaps and Pain Points

Despite a solid foundation, several issues hinder the UX:

- **Inconsistent Spacing & Layout:** Margins and padding in the current CSS vary in ways that aren't systematic – some elements are tightly packed while others have extra gaps. Without a design token system, these inconsistencies accumulate and make the interface feel slightly unpolished. We need a consistent spacing scale (e.g. 4px increments) to apply uniformly for harmony.
- **Lack of Visual Emphasis for Key Actions:** The primary action (opening an article to read) is not distinctly highlighted – it's just one of several icons. Meanwhile, secondary actions like archive or delete can visually compete with it. Users might not immediately see how to open an item (especially new users who may not realize clicking the card or the book icon opens the reader). We should establish a clearer primary action styling.
- **Label Overload and Redundancy:** Currently, system labels (or "flairs") like ⭐ or 📰 appear as icons in the metadata line *and* user labels appear as text chips below the title. In some cases, the UI shows an icon *and* the word "Label" which is redundant. There's also no obvious distinction in hierarchy between an official marker (like an article being a newsletter) and a user's organizational tag – they are in separate places, but new users might not grasp the difference quickly.
- **Fixed Information Density:** The single card size/layout might be too roomy for power users (who want to see more items on screen) yet still overwhelming for casual users (who might prefer a simpler list). Without a **density toggle**, one size fits all may actually fit few.
- **Processing Feedback:** When a user saves an article, it goes through a parsing process (to clean content for reader mode). Currently, items in "PROCESSING" state look almost the same as any other item (perhaps with a text indicator). Users can be confused if an article isn't ready to read. Similarly, once parsed ("SUCCEEDED"), there's only a subtle change, and failures are not clearly indicated at all.
- **Non-Touch-Friendly Interactions:** The hover-to-reveal actions don't work on mobile or touchscreens – mobile users have no easy way to quickly archive or tag an item from the list. Also, small icons can be hard to tap. The design needs to adapt to touch input with alternative patterns (like swipe actions or always-visible minimal buttons on mobile).
- **Multi-Select Mode UX:** The current multi-select (batch edit) mode requires tapping a "Select" toggle, then checkboxes appear on each card. It's not obvious and can be clunky. If you accidentally navigate away, you lose the selection. There's room to make batch operations more fluid (e.g. allowing a long-press on a card to initiate selection on mobile, or shift-click ranges on desktop).
- **Reading Progress Clarity:** The thin progress bar on cards changes color as you read more (blue to green). While clever, color alone might not convey "how far" at a glance, and new users may not know what the colors mean. A 20% read article and 90% read article might both just show a partially filled bar with different colors – not immediately obvious without a legend. There is also no explicit "Completed" mark besides a full green bar.
- **State & Status Visibility:** There's no strong visual difference between an article that's unread vs one that's in the archive, or one that failed to parse. All appear as normal list items with maybe minor text differences. This can lead to confusion (e.g., "Did my article save correctly?" or "Where did that article I archived go?").

These gaps present **opportunities for design enhancements** detailed below.

Design Challenges & Proposed Solutions

1. Information Density vs. Readability

Challenge: Power users want to see many items at once for quick triage, whereas others need a more spaced-out layout to comfortably read titles and metadata. The current design has a medium density that may not suit everyone.

Proposed Solutions:

- **Density Settings:** Introduce view options like **Compact, Comfortable, Spacious**. For example:
- *Compact:* minimal padding, no thumbnails (or very small thumbnails), single-line titles. This could show significantly more items per screen – ideal for users like "Tech Tom" who scans a lot of RSS items quickly. (Instapaper's web list exemplifies this approach – it omits thumbnails entirely, allowing more headlines at once [1].)
- *Comfortable:* the current default – medium thumbnail, 1-2 line title, basic metadata.
- *Spacious:* larger thumbnails or even article excerpt snippet, more white space – for those who prefer a richer preview of each item (closer to something like Matter or an RSS reader with previews).
- **User Control with Smart Defaults:** The app can default to Comfortable (balanced) but allow easy switching via a "Density" toggle in the UI (perhaps in the filter bar). Remember the user's choice per device – e.g., a user might use Compact on desktop where they have a big screen, but Spacious on a small tablet for readability.
- **Responsive Adjustments:** Regardless of setting, ensure on very small screens some adaptation occurs. E.g., on phones in compact mode, maybe we hide even more (only titles?). On larger monitors, even comfortable mode could show multiple columns if space allows.
- **Typography Scaling:** To maintain readability in denser views, test the smallest font sizes for metadata. We likely keep body text at 14px minimum for legibility. Title text in compact mode could drop from 16px to 14px if needed, but no smaller. We will use a typographic scale that ensures **readability** even when density increases (no squinting required).
- **Progressive Disclosure of Metadata:** In compact mode especially, we can hide some less critical info until hover or tap. For instance, maybe the author name and labels are hidden in the tightest view, showing only on hover/expand. Or we only show one line of metadata (like just the timestamp or source), and the rest on demand. This way the UI shows the essentials (title) and defers details to interaction, a classic **progressive disclosure** approach.

Why: This flexibility caters to different usage patterns. A researcher with 1000 articles can switch to a no-thumbnail list to power-scroll and find items (similar to an email client or Instapaper's text list). A casual reader can stick with a visual card grid that feels inviting. Providing density options addresses both ends of the spectrum without forcing a one-size-fits-all.

2. Action Discoverability vs. Clutter

Challenge: Omnivore supports 5-7 actions on each item (open/read, archive, add/remove labels, open original link, delete, share, etc.). Displaying all these as icons permanently would clutter the interface; hiding them (current hover approach) can make them **invisible** to new or mobile users.

Proposed Solutions:

- **Hover Reveal on Desktop:** Keep the current hover-to-reveal icon bar for desktop web, as it works well to reduce noise. However, enhance it by:
- Adding a subtle **on-hover highlight** to the entire card to hint it's interactive.
- Possibly showing a **"More" (…) icon** at the end of the metadata line or card corner even when not hovered, as a cue that actions exist. This "more" icon could be persistent and on click opens the action menu. This ensures new users see an affordance for actions without needing to accidentally hover.
- **Tap/Swipe on Mobile:** On touch devices, implement a **swipe gesture** on list items to reveal actions. For example, swiping left could reveal "Archive" and "Delete" (destructive or frequent actions), and swiping right could mark as read or favorite. We will follow mobile conventions (e.g., many email apps use swipe for archive/delete). **Important:** We must provide a visual indicator or tutorial for this, as swipe actions have discoverability issues (users won't know they can swipe without a hint) [2]. One approach is to show a subtle hint (like part of an icon peeking from the side of the card saying "swipe" during first use, or a one-time tooltip). Additionally, provide a fallback: a **long-press** on a card can pop up an action menu (for users who don't like swiping).
- **Always-Visible Primary Action:** Make the primary action (open to read) more obvious. For instance, a clear **"Read" button** or icon could be visible on each card (perhaps on the thumbnail or as a play/read icon overlay). Pocket uses a similar approach where tapping anywhere opens the item, but they also sometimes show a " ► " icon on images. We can overlay a subtle ▶ **icon** on the thumbnail to indicate "Open reader". This reduces ambiguity that clicking the card opens it.
- **Keyboard Accessibility:** Ensure all actions are accessible via keyboard. The hover icons currently might not appear for focus. We will update the card component such that when a card is focused (e.g., tab navigation), it behaves like hover – showing the action icons. Users should be able to tab to an article card, press Enter to open it, or press a shortcut (maybe Alt+A for archive) to perform actions. We can list keyboard shortcuts (like "?" key brings up a cheatsheet).
- **Contextual Menus:** Provide a context menu (right-click on desktop, or a small … menu button on each card) that lists all actions with text labels. This benefits discoverability (users see "Archive" in words, etc.) and accessibility (screen readers can read the menu). It also handles future expansion (if more actions are added, they can live in the menu rather than as new icons).
- **Prioritize Frequent Actions:** Research which actions are most used (likely "Read" and "Archive"). Make those easiest to access (e.g., swipe or single-click). Less common actions (e.g., "Share" or "Move to folder") can be in the overflow menu. This prioritization keeps UI simpler for the common workflows.
- **Avoid Misleading Hover States:** We should only highlight exactly what is clickable. For example, don't highlight the whole card on hover if only certain parts are interactive [3]. In our case, since clicking anywhere on the card (except on specific buttons) will open it to read, it's fine to make the whole card hover-highlight as a clickable unit.

Why: This balanced approach retains a clean look (no button soup on the screen), yet ensures that whether you use mouse, keyboard, or touch, you can easily find and use the item actions. New users won't be lost, and power users can still swiftly act on items in bulk.

3. Simplifying the Label & Tag System

Challenge: Omnivore has two parallel labeling systems: - System-generated "flair" icons (for special statuses like favorite, pinned, newsletter, etc.). - User-generated labels (tags) with custom colors.

Currently these appear in different places and styles, which can confuse users (e.g., seeing a icon and a colored ● label and not knowing how they differ).

Proposed Solutions:

- **Distinct Visual Treatments:** Maintain the separation but clarify it visually:
- **Flair Icons (System Labels):** Continue to show these as small monochrome (or single-color) icons in the metadata line. They are intentionally subtle, acting as quick glyphs (e.g., ⭐ for favorite). We might add tooltips on hover (e.g., "Favorited" when hovering ⭐) for clarity. Keep these icon-only to save space.
- **User Tags:** Display as colored chips with text (as currently done) below the title or at the bottom of the card. We remove any "🏷 Label:" prefix – just show the tag name with a colored dot or background. For example, a tag might appear as a pill: ● **Project X**. The color and text are enough; no need to literally write "Label".
- **Different Shapes/Styles:** Another idea to differentiate: system labels could be **outlined icons** or a specific color (e.g., always grey or gold icons), whereas user tags use filled color backgrounds. This contrast in style will let experienced users instantly tell system status vs personal categorization.
- **Limit the Display Count:** If an item has many user tags, don't list them all (which could clutter). Show perhaps up to **3 tags**, then "+2 more" if there are more. Clicking the "+2 more" could expand the list or open a details popup. This keeps the card tidy.
- **Clickable Labels:** Make both the flair icons and tag chips clickable for quick filtering. For instance, clicking the newsletter icon could filter the library to show all newsletter items; clicking a **Project X** tag chip filters to that tag. This turns labels into navigation aids and encourages users to use them for organization.
- **Tag Creation UX:** Provide an easy way to add tags from the card. Perhaps one of the hover actions is 🏷 which opens a small tag editor (with autocomplete for existing tags, and option to create new). The current design likely has something like this; we ensure it's accessible on mobile too (maybe via the overflow menu "Add tag").
- **Icon Consistency:** Use a consistent icon set (feather icons or Material icons) for all system labels and actions. For example, the pin and star ⭐ might be replaced with their Material Icon equivalents for a consistent look (or use the emoji-like icons but ensure they match the font style). Consistency in iconography style will make the UI feel more cohesive.

Why: By clearly separating what is a functional/status indicator (flair) versus user-defined category (tag), we reduce cognitive load. Users like Rachel (the researcher) will heavily use tags; Tom (tech) might rely on system markers like feeds; Caroline (casual) might barely use tags at all. The design should accommodate all, making tags useful but not intrusive. Clear tagging also leverages Pocket's strength (unlimited tags for research) [4] [5] while avoiding Instapaper's folder rigidity – thus playing to Omnivore's differentiator of powerful organization.

4. Multi-Selection & Batch Actions

Challenge: Users often need to select multiple items (e.g., to archive a batch of read articles or tag several related items). The current multi-select mode is hidden behind a toggle and can be unintuitive, especially if selections reset unintentionally.

Proposed Solutions:

- **Easy Multi-Select Activation:** Provide multiple entry points into multi-select:
- A "Select" mode toggle button in the top toolbar (as exists).
- Additionally, allow **Shift+Click** on desktop to select ranges (e.g., click first item, Shift+click last item to select all in between, common in email clients).
- Allow **long-press** on a card in mobile to enter selection mode and select that item (a common pattern in mobile list apps).
- **Visible Checkboxes:** When in select mode, show checkboxes on each card (as currently does), but ensure they are easy to tap on mobile (sufficient size). Also allow clicking the card itself in select mode to toggle selection (not opening the article).
- **Persistent Selection:** If the user navigates (e.g., switches tag filter or goes to another folder like Archive), consider preserving the selection state *if it makes sense*, or at least warn before clearing selections. Alternatively, selections could be global (not cleared until the user exits select mode explicitly). This might be complex, so at minimum, provide a clear "Exit multi-select" button that users consciously click to clear selections, rather than automatically clearing on any nav change.
- **Floating Batch Actions Bar:** Once one or more items are selected, show a **floating action bar** at the bottom (on mobile) or top (on desktop, just below the header) summarizing "X items selected" and offering actions like Archive, Delete, Tag, etc. This bar stays visible as the user scrolls. It provides a clear indication they're in multi-select mode and gives one-tap access to batch operations. This is similar to Gmail's interface or other management apps.
- **Select-All and Smart Selection:** Provide a "Select all" option when appropriate. For example, if a user has filtered to a tag or doing a search, a Select All could select all results. Also possibly "Select all unread" or "Select all in this feed" shortcuts, depending on context – these can live in an overflow menu in the multi-select bar.
- **Feedback for Batch Actions:** After a batch operation (say archiving 20 items), give a brief confirmation ("Archived 20 articles") with an undo option via toast. This reassurance helps user trust that the action happened (and undo protects from mistakes).
- **Keyboard Shortcuts:** Support keyboard in multi-select mode too (e.g., hold Shift and use arrow keys to select multiple, or press "A" to archive selected, etc., for power users).

Why: Batch operations should empower users, not frustrate them. Researcher Rachel with 1000 items might periodically clean up hundreds – a smooth multi-select will save her time. By making selection easier and persistent, and showing a clear action bar, we reduce accidental exits and make it obvious how to act on selected items. This contributes to a sense of Omnivore being a professional tool that can handle scale.

5. Reading Progress & Completion

Challenge: Omnivore currently indicates reading progress with a thin bar on each card that fills and changes color (blue/yellow/orange/green) as a percent of the article read. However, without a numeric or textual indicator, users might not know *at a glance* how much is left or what the colors mean. There's also no explicit "Unread vs Finished" marker besides that bar.

Proposed Solutions:

- **Progress Bar Enhancement:** Keep the progress bar as a quick visual cue (it's subtle and doesn't take much space). Use a single accent color for the bar (e.g., blue) for simplicity, or continue the gradient

from blue to green but explain it in a legend or onboarding. Consider **adding a small percentage number** next to or overlaying the bar for clarity (e.g., “45%”). This number could appear on hover or constantly if space allows. A user seeing “90%” knows they’re almost done.

- **Alternate Indicator – Time Remaining:** Instead of or in addition to percentage, show an estimate of time left to read (based on article length and reading speed). For example, “3 min left” could display on a nearly finished article. This contextualizes progress in more human terms. Instapaper on web shows reading time estimates for articles [6], and some users might find “3 min left” more motivating than “90%”. We could A/B test whether percentage vs time is more effective for users.
- **“Finished” State:** Clearly mark fully read articles. If an article is 100% read (user scrolled to end), we could:
- Turn the progress bar green (as now) and perhaps replace it with a **checkmark icon** or a label “Read” for a persistent indicator. For example, a small green check could appear next to the title or in the metadata.
- Visually deemphasize read items (e.g., title becomes a lighter color or italicize, similar to how email clients mark read emails in a lighter font). But be careful not to reduce contrast too much for accessibility.
- **In-Progress vs Not Started:** For articles that have some progress but not finished, consider an icon or text like “In Progress” that could show on hover. Perhaps the color of the title could also change or an icon indicator appears when something is partially read. The goal is to differentiate: **Unread (0%)**, **In Progress (1–99%)**, and **Completed (100%)** states clearly.
- **Legend and Consistency:** If using colors (blue→green), ensure colorblind-friendly design by not relying on hue alone. For example, different patterns or icons at milestones: maybe a half-filled circle icon for in-progress, a check for done. Additionally, provide a legend in a help tooltip or guide (so users know green means done).
- **Sync Progress:** Ensure that the progress indicator updates across devices (it sounds like it does sync reading position, which is good). Perhaps show a small sync icon if something is updating. This might be more technical, but just to note: users trust the progress more if it reliably reflects their actual reading.

Why: Reading progress indicators help users manage their saved content – they know what they’ve started and what’s left. It can encourage finishing articles (e.g., seeing “3 min left” might prompt the user to complete it). By making the indicator clearer and distinguishing finished items, we provide a sense of accomplishment and order. Instapaper’s use of dot indicators on mobile gave a quick feel of article length and read portion [6]; Omnivore can achieve similar clarity with bars or text. The improvements ensure that even a quick scan of the library gives the user immediate context on their reading journey.

6. Clear Item States (Processing, Error, Archived)

Challenge: Users encounter various item states: - **Processing:** Article is being fetched/cleaned. - **Failed:** Article could not be processed (perhaps paywalled or network issue). - **Archived:** Article moved to archive (not in main library). - **Trashed/Deleted:** (should be separate view typically). Currently, these states are not obvious in the UI, potentially leaving users wondering what’s happening.

Proposed Solutions:

- **Loading Skeletons:** When new items are added and are in “PROCESSING”, display a **skeleton card** – a placeholder card with shimmering grey blocks instead of text, indicating loading. This is a modern

pattern that reassures the user that content is on its way. Alternatively, show a “Processing…” label or spinner on the card. The card could have a dimmed appearance until ready.

- **Success State:** Once processing is done and the article is ready to read (most cases), simply display it normally. A subtle entrance animation (fade in) can signify it’s become available. We could also flash a brief “✓ Article saved” toast or highlight.
- **Failed State:** If processing fails (e.g., error retrieving content), mark the card with a red outline or icon. For instance, an error **!** icon could appear on the thumbnail or metadata. Also, provide an actionable message: e.g., “**!**Failed to parse. [Retry].” The user could click retry to attempt fetching again. Ensuring failures aren’t silent is crucial so users don’t think the article is available when it isn’t.
- **Archived Items:** In the Archive section, items could have a different style to remind users they’re archived. Perhaps they are semi-transparent or have an “Archived” badge. However, since Archive is a separate view, it might be enough that the section is labeled. Within the main library, archived items should ideally not appear at all (unless we have a combined view with filters). If they do appear (like if someone labels an archived item and it shows up in a label filter), we could tag it with an “Archived” label or use a distinct icon (like a box 📦 icon).
- **Pinned/Favorite States:** These are covered by system labels (☆,). Just ensure they’re visible enough to denote importance (maybe a gold star for favorite).
- **Toast Notifications:** Use brief notifications for state changes – e.g., “Article saved! Parsing content…” then “Article ready to read.” Or after a batch archive: “5 articles archived.” These provide feedback so the user isn’t left guessing.
- **Consistency:** Use the same color language for states across the app (e.g., blue for in-progress actions, green for success, red for errors, yellow/orange for warnings or processing). For example, a processing icon might be yellow, error icon red, success check green. But use these with restraint to avoid a rainbow; mostly the states should be subtle until user needs to see them.

Why: Clear states eliminate uncertainty. If a user saves a link and sees a skeleton card or spinner, they know it’s coming. If something fails, they know to take action or try later instead of endlessly clicking a broken item. Archival status being visible prevents the “where did it go?” confusion. All of this leads to a more trustworthy and user-friendly experience, critical for a tool people rely on to collect important reads.

Visual Hierarchy & Style Guide

To support the above changes, we need a strong underlying visual design system – consistent colors, typography, spacing, and components that make the interface coherent and delightful.

Typography & Content Hierarchy

We will use **Inter** as the primary typeface (as currently used), leveraging its versatility at various weights.

- **Titles (Level 1 info):** Article titles on cards will be ~16px–18px, in a medium or bold weight (600–700). This is the first thing a user sees on a card, so it should have the highest contrast. In dark mode, that means white (#fff) or near-white text on the dark background. One or two lines maximum, with tight line-height (~1.3) to keep it compact yet readable.
- **Metadata & Body (Level 2 info):** This includes author name, source, reading time, dates. Use a smaller size, e.g. 12px–13px, in a regular or medium weight. Color should be a gray/light-gray (#d9d9d9 for important bits, #898989 for secondary bits) to clearly differentiate from titles.

Metadata often appears in a single line (e.g., "⏱5 min • The New York Times"), so use bullet or middot separators and small icons to pack info efficiently.

- **Labels/Chips (Level 3 info):** 11px–12px font, medium weight. These should not stand out more than titles or metadata. When inside a colored tag chip, the text might actually be on a colored background, so ensure contrast (white text on darker colored chip, or black text on light chips if we ever use light mode).
- **Buttons and UI Chrome:** For navigation bars, modals, etc., use 14px–15px for button labels or menu items, since these are interactive elements but not content. Ensure focus state (outline) and perhaps uppercase or strong styling for primary actions.
- **Line Heights:** Titles: ~1.3, Body text (in reader mode, not addressed deeply here) might be 1.5 for readability, but on the cards, most text is short. For any multi-line text (like an excerpt in a spacious view), use ~1.5 line-height.

Color Palette & Usage

Omnivore's brand color is a warm **Accent Yellow (#FFD234)** – we can use this sparingly to draw attention to primary actions or highlights.

- **Backgrounds:** Continue with **Dark Theme** as default. Primary background #1a1a1a (for the app background) and secondary #2a2a2a (for card surfaces or modals). Use a slightly lighter variant (#252525 or #2a2a2a) for things like the top bar vs content area to create subtle contrast and layering.
- **Card Surface & Elevation:** Cards can be #2a2a2a or #252525 (just a notch lighter than main background to stand out). We might add a very subtle drop shadow or border on cards for separation – e.g., a 1px border in #333 or a faint shadow on hover to indicate lift.
- **Accent Colors:**
- Use **Omnivore Yellow (#FFD234)** for brand moments – e.g., the Add button ("+ Add") or primary CTAs, and perhaps for selected states or highlights. This color is bright, so use it to call attention, but not for large areas (to avoid eye strain on dark mode).
- **Blue (#4A9EFF)** for actionable highlights – e.g., focus ring, link color, or the progress bar initial color. Blue implies interactivity (links) and is generally good for indicating something is currently active or to be clicked.
- **Green (#4CAF50)** for success indicators (100% read, completed states).
- **Orange/Yellow (#FF9500)** for warnings or high-progress states (this was used in the progress bar gradient).
- **Red (#E85C47 or #8B0000)** for destructive actions (delete) or errors.
- **Text Colors:** Primary text in dark mode: #FFFFFF for most important text (titles). Secondary: #D9D9D9 for key metadata. Tertiary: #898989 for less important info. Muted/disabled: #666666 or #555555 for placeholders or very low priority text.
- **State Colors:** Use subtle colored badges or dots for states if needed. For example, an archived item could have a grey tint. A syncing item maybe blue dot, an error a red dot, etc., but keep these small and iconographic rather than large swaths of color.
- **Contrast:** All text will meet WCAG AA contrast against its background. For example, small text on dark background should be at least 4.5:1. We'll test the gray values for metadata to ensure they are bright enough.

In summary, the color usage will create a **visual hierarchy** where content (article text and titles) pops, while UI chrome (buttons, backgrounds) recede. Bright colors will be used intentionally to guide eyes to the most important actions or statuses.

Spacing & Layout Grid

We adopt a **4px baseline grid** for spacing and sizing: - Small padding/margin: 4px or 8px. - Medium: 12px or 16px. - Large: 20px, 24px, etc., as needed.

For example, card padding might be 12px internally. Gaps between cards might be 16px. Section paddings could be 24px.

Consistent border-radius on elements: use one of a few token values, e.g., 4px for small elements (buttons), 8px for cards (as currently), maybe larger (12px) for modals. This consistency ensures a cohesive look. We'll likely keep the card corner radius at 8px as noted (matches the design token `--radius-lg`).

Component Styles Highlights

- **Library Cards:** We will have two layouts (grid and list) but share style rules. Each card is a dark surface with light text. On hover or focus, it slightly elevates (shadow) and shows action icons. In list view, the thumbnail is left-aligned at 55px square, text on the right. In grid, thumbnail on top with full width, text below. Ensure the thumbnail has a consistent aspect ratio or is cropped if needed. Use a placeholder image or icon for items with no image.
- **Buttons:** Primary button style (for the main call-to-action like "Read" or the global "+ Add") will be filled with the accent color (yellow or perhaps blue) and have clear text. Secondary buttons (like "Archive" in an overflow menu) might just be icon-only or text with no fill (ghost style). Tertiary (less important like "Delete" that we don't want to over-emphasize) can be an icon-only with a subtle color (light gray icon).
- **Icons:** Keep them uniform style (feather icons or Material outlined). Size around 16px (for action icons). Give icons accessible labels (aria-label) since we won't have text labels visible.
- **Nav bars:** The top bar (with search, add, user menu) will span full width, background #2a2a2a, height ~48px. Elements vertically centered. The second-tier filter bar (with toggles for view, sort, multi-select) can be a bit different shade (#1f1f1f) to visually distinguish it as a control strip. Use divider lines or spacing to separate groups of controls.
- **Modal/dialog style:** If any (like confirming delete or editing tags), use the BG Elevated color (#333333) with rounded corners and padding. Text in modals is slightly brighter (since background is darker).

Accessibility Considerations

- **Color Contrast:** As mentioned, all text/background combinations will be checked. E.g., our grey text on dark backgrounds should be bright enough. The colored label chips need sufficient contrast with their text (we may choose automatically black/white text depending on the tag color brightness).
- **Focus States:** Every interactive element (buttons, links, cards) will have a visible focus outline when navigated via keyboard. Likely a 2px outline in our blue (#4A9EFF) or yellow, offset a bit outside the element, to be clearly visible.

- **ARIA Labels:** Icons and buttons will have descriptive labels (e.g., the archive icon would have aria-label="Archive article"). The role of elements like the card (which is clickable) will be clear (maybe treat the whole card as a link to the reader).
- **Screen Reader Order:** Ensure logical ordering in the DOM – e.g., card title should be read before metadata and actions. Possibly mark the title as a heading for easy navigation (maybe not needed, but if it helps SR users skim).
- **Keyboard Shortcuts:** While not purely a visual design concern, it's part of UX. We will document and support shortcuts (e.g., "N" for new item, "F" for search focus, arrow keys to move selection, etc.). This helps power users and also aligns with accessibility (some users with motor issues prefer keyboard).
- **Touch Targets:** Increase the size of tap targets on mobile. Icons that were 16px might be wrapped in a 40px touch area, for example, to prevent frustration. Similarly, spacing out list items so that a finger can comfortably tap one without accidentally hitting another.
- **Motion & Reduction:** Use motion (animations) sparingly and for functional purposes (e.g., a slide in for swipe actions, or a fade for modals). Ensure that any animated elements respect user preferences (if `prefers-reduced-motion` is set, we will minimize animations).
- **Testing:** We will conduct an **accessibility audit** (perhaps using tools or manual testing with screen readers like NVDA/VoiceOver) to catch any issues early, as required by our standard (WCAG 2.1 AA compliance).

Component Redesign Details

This section outlines specific key components with their new design requirements, integrating much of the above into concrete components.

Library Item Card

Features & Layout: Each card encapsulates an article in the library. It must display at minimum: Title, source/author, relevant metadata (date added, reading time), labels, and affordances for actions.

- **Thumbnail:** A preview image (if available) or a fallback icon if not. Size ~150px wide (grid) or 55px (list). Rounded corners (same radius as card). The thumbnail can also host an overlay icon (like a play/read ►).
- **Metadata Row:** Top of card (or top-left in list layout). This will include small icons and text, e.g., " 2 days ago • 5 min read". We use a small clock or time icon for "5 min". If the item is from an RSS feed or newsletter, an icon (or) might appear here. This row is in a muted style (12px, #898989). It may also include system flair icons (☆ if favorited, etc).
- **Title:** Below metadata (or to the right of thumbnail in list mode). 1–2 lines, 700 weight, 16px, truncating overflow. The title is the focal point. We ensure it has enough contrast and maybe slightly larger than current for emphasis. Clicking the title (or anywhere on card) opens the article.
- **Author/Source:** After title (especially in grid layout, likely on a new line under title). E.g., "*The New York Times*" in 12px italic or regular. We might prefix with author if available, e.g., "Jane Doe – The New York Times". This text is secondary.
- **Labels/Tags:** One line showing up to 3 colored chips for user tags, if any. If none, this line collapses. If an article is pinned or special, we might show a pin icon here or in metadata instead.
- **Progress Bar:** Along the very bottom of the card, a 4px high line spanning the width (except maybe some padding from edges). It's subtle (perhaps grey background with colored fill). If 0%, maybe a

grey empty bar or no bar at all until started. If 100% read, a full bar (green) and maybe a check icon overlay on the left end.

- **Hover Actions (Desktop):** On hover, a semi-transparent dark toolbar appears either overlaying the bottom of the card or at the top right corner. Contains icon buttons: Read (if we choose to show it), Archive, Label, Share, Delete, and an overflow if needed. These icons are typically light gray circles or just icons with no background, highlighted on hover.
- **Selected State:** If in multi-select mode and this item is selected, the card background could change (e.g., to #3A3A3A) or get a colored border (blue outline). The checkbox is shown (checked). We ensure the progress bar or other info doesn't conflict visually with selection state.

States to consider: - *Unread (new)*: Perhaps the title is bold vs normal for unread? Or an icon dot indicating unread. (Pocket does something similar by not marking anything, whereas email apps bold unread. We might or might not bold titles for unread to give an extra hint.) - *Hovered*: Card slightly raises with a shadow, actions visible. - *Focused (keyboard)*: Similar to hover – show actions, plus a focus ring around the card. - *Active (being clicked)*: brief press animation (scale 0.98 or so). - *Processing*: Card is dim with a spinner overlay or skeleton (as discussed). - *Error*: Card might have an error icon in metadata or instead of thumbnail. - *Archived*: If shown at all in main list, maybe an "Archived" text or icon.

Responsive Behavior: On small mobile screens, we might switch to a one-column list automatically. The card might simplify: maybe hide the thumbnail for a cleaner list if screen is very small (or provide an option). The actions may always be in an overflow menu on mobile (since hover is not applicable).

Action Buttons & Menus

We have several categories of actions, each with distinct styling:

- **Primary Action – "Open/Read":** This might be the card itself (clickable area) rather than a button. But we ensure it's the most obvious. On some UIs, we could place a prominent button like a "Read now" on the card, but that might clutter. Instead, we ensure clicking the card is clearly the main action (with the whole card highlight on hover).
- **Primary Global Actions:** The top bar "+ Add" button (to add new content manually) should stand out – e.g., a bright blue or yellow filled button with a plus icon. This is an important CTA for users to bring content in.
- **Secondary Actions (common per item):** Archive, Mark as unread/read, etc. These appear as icons. Style: outline or ghost buttons (no heavy fill). On hover they might get a slight highlight or a tooltip with text.
- **Tertiary (destructive or less-used):** Delete, for example, could be colored red or kept hidden in overflow to prevent accidental taps. If shown, it might be just a trash icon that turns red on hover.
- **Overflow Menu (…):** When clicked, reveals a context menu with text like "Share via…", "Move to folder…", etc. Style that menu in the dark theme (dark background, light text, 8px rounded corners, subtle shadow). Menu items height ~32px each for easy tapping. Include icons in the menu for each action for familiarity.
- **Tooltips:** Hovering on any icon button should show a tooltip (e.g., hover on 🏷 shows "Label (L)", indicating you can press L key as shortcut). This helps discovery and learning of shortcuts.

Consistency & Feedback: All buttons should use the same active/hover states (like lighten background on hover in dark mode, or a slight scale). After an action is clicked, provide feedback (like if you click Archive,

maybe the icon could momentarily change to an undo or a checkmark, and the card could disappear from the list or grey out). This ties into the toast notifications of actions.

Labels & Tags

As discussed: - Render user tags as colored chips with text. We'll use the color palette provided (red, orange, yellow, green, blue, purple, pink, gray) for user to choose tag colors. The chips will use a semi-transparent version of the color for the background (to avoid too much vibrance on dark mode) and a solid border in that color for definition. For example, a Blue tag might be a slightly muted blue background with a brighter blue border and white text. - Size of chips: roughly height 20px (small pills), padding 4px horizontal. - The ● symbol (colored dot) can prefix the tag name or we can just use a solid round background. Either way, something like ● **Design**. - System labels (flairs) will remain just icons, placed in the metadata line without text. They should have accessible names (aria-label like "Newsletter" for). - Possibly we add a legend or reference in the UI for what each flair means (maybe in a help doc or tooltip on hover of the icon we say "This is a newsletter article automatically added via email"). - The user can manage tags via a side panel or settings. But in the UI, adding a tag likely via the hover action opens a small dialog listing existing tags with checkboxes and an input to add new – ensure this dialog is keyboard navigable and not too cluttered.

Progress Indicator

The progress bar at 4px height spans the width of the content area of the card. It's placed at the bottom, maybe with a tiny radius on corners.

- Use CSS transitions to animate it smoothly as progress updates (if we live update).
- If we include text (percentage or time), we could overlay it on the right end of the bar or place it next to the bar (e.g., "45%").
- Completed articles: instead of a bar, perhaps just a green check icon on the card (for a cleaner look). Or a full green bar with the check at the end.
- If article not started: maybe no bar visible at all to reduce clutter (or a grayed out bar outline to show something is there).
- Testing needed: If many bars on screen, does it look too busy? Possibly keep them low-contrast (e.g., dark gray background bar with a softly colored foreground) so they don't scream visually.

Navigation & Layout Components

Omnivore's navigation consists of a few layers: - **Top App Bar:** Contains search, add button, profile menu. We will ensure search input is prominent (center or left). The search bar could even expand when focused. The "+ Add" stands out to encourage adding content. The user menu (account avatar) sits on the right. - **Filters/Actions Bar:** Right below top bar, provides filtering and view options. For example: a dropdown to filter by label or feed, the toggle for list/grid view, the multi-select toggle, and sort order. These controls should be visually grouped – e.g., all view-related controls together. Use icons with tooltips: e.g., [≡] for list and [⊞] for grid, that toggle each other. A label filter might show an icon 🏷 and the current filter name. Sort button shows maybe "Date ▼". - **Folder Tabs or Sections:** Possibly a third layer (depending on design) for switching between Library (Inbox), Archive, Favorites, etc. For instance, tabs or a sidebar. In a minimal approach, we could have a horizontal tab list: **Inbox (unread)** | **Archive** | **Trash**, etc., below the filter bar. The active tab is highlighted (underline or background). - Alternatively, a sidebar (drawer) on desktop could list these sections and also show a list of tags/folders for navigation. However, the brief did not explicitly

mention a persistent sidebar, it mentioned “Folder Tabs” as Tier 3. Likely they want horizontal tabs for key sections to save space in a dense UI. - **Responsive Nav:** On mobile, the multi-tier nav might collapse into a single top bar with a menu. Possibly a hamburger menu for sections (Inbox, Archive), and the search might become an icon that opens a search overlay to save space. We’ll design mobile nav carefully to still allow filtering by tags (maybe via a filter icon that opens a filter sheet).

By standardizing these nav components with consistent height (48px each tier), distinct background shades, and clear labels/icons, users can intuitively navigate and control their view.

User Research & Validation Plan

No design system is complete without user feedback. We propose the following research and testing to validate our design changes:

Usability Testing (Qualitative)

Conduct moderated or unmoderated tests with 5-10 users representing our personas (power users and casual):

Key Scenarios & Questions:

- **Card Clarity:** Present users with the new card design. Ask: *“What information can you gather about this article at a glance?”* We expect answers about title, source, time, etc. If users miss key info or find it too much, that’s noted. (Goal: ensure quick scanning is possible.)
- **Action Discoverability:** Without prompting, observe how users attempt to archive or delete an item. Do they find the hover icons or swipe gesture? Ask: *“How would you archive multiple articles?”* This reveals if the multi-select entry is clear. Also: *“Did you notice the actions available for each item? What actions do you see?”* (Goal: actions should be easily findable.)
- **Label Understanding:** Show an item with a star icon and a couple tag chips. Ask: *“What do these symbols or colored tags mean? How would you use them?”* We want to ensure users distinguish system vs user labels. If not, maybe more differentiation or onboarding is needed.
- **Information Density Preference:** Show the same set of articles in Compact vs Spacious mode. Which do they prefer and why? Ask: *“Is there too much or too little information on these cards?”* Also *“Do you find the thumbnail useful or distracting?”* (Goal: find the optimal default and confirm the need for density toggle.)
- **Reading Progress Indicator:** If possible, show cards at various progress states. Ask: *“How can you tell if an article is unread, partially read, or finished?”* See if they interpret the progress bar or checkmarks correctly. Also *“Would seeing a percentage or time remaining be useful to you?”* (Goal: verify if our indicator communicates clearly.)
- **Multi-Select & Batch:** Have the user select multiple items (in a prototype). Note if they figure out Shift+click or the select toggle. Ask: *“How would you delete all articles in this list?”* (Goal: ensure batch actions are discoverable and easy.)
- **Mobile Interactions:** On a touchscreen prototype, see if they naturally try swiping or tapping the overflow for actions. Ask for their feedback: *“Was it easy to perform actions on an item on your phone?”* (Goal: validate swipe and long-press patterns.)

- **General Visual Appeal:** Ask for subjective feedback: *"What's your impression of this interface? Does it feel overwhelming, or clean, or modern?"* We want to ensure the design is pleasing since that affects adoption.

We will record these sessions and extract pain points for iteration.

A/B Testing (Quantitative)

Once implemented, we can run A/B tests in the live app to measure the impact of certain design decisions:

- **A/B Test 1: Action Button Visibility** – Compare the current hover-only design (Variant A) vs. a variant where a primary action button (e.g., a persistent "Open" or "Archive" button) is always visible on the card (Variant B), vs. possibly a variant with an always-visible "…" menu icon (Variant C). **Metrics:** Time to perform an action (from seeing an item to archiving it), number of actions taken, and user preference via survey. This will inform if making an action persistent improves speed or if it just adds clutter.
- **A/B Test 2: Label Display Mode** – Test icon-only vs text labels. For example, Variant A: system labels and user tags both icon+text (which could mean system icons have a small label too, or all tags show as text), Variant B: the proposed design (icons for system, chips for user), Variant C: a text-only approach (no icons, e.g., "Favorite" spelled out – likely too verbose). **Metrics:** Do users correctly identify item status (favorite, etc.) and find items via tags? We could track usage of tagging features or ask which design they find cleaner.
- **A/B Test 3: Card Density Default** – Some users get a default Compact view vs others Spacious. **Metrics:** How many articles do they click through? Is retention or engagement higher when more items are shown (compact) or when a richer preview is shown (spacious)? This can guide what default to use for new users. Also monitor if users toggle the setting.
- **A/B Test 4: Progress Indicator Format** – Variant A: Color bar only (no text), Variant B: bar + percentage number, Variant C: bar + "time left" label, Variant D: an icon/badge approach ("In Progress" text or icon, "✓ Done" for finished). **Metrics:** Does inclusion of text impact reading completion rates (perhaps seeing time left motivates finishing)? Do users in a survey understand their progress better in one variant? Also check UI preference (which variant do users find least cluttered vs most informative).

We will run these tests for a statistically significant period and use the results to fine-tune the design system.

Analytics to Monitor

Beyond tests, ongoing metrics can tell us if the design is successful:

- **Engagement with Actions:** Track frequency of key actions (per user per week: how often they archive, tag, favorite, delete). After redesign, expected easier access could increase these if users were previously unaware of them.
- **Article Completion Rate:** What percentage of saved articles get opened and read (to at least 90% progress)? If progress indicators and better organization help, we might see a higher completion rate (users actually read what they save, which is a success sign).

- **Growth of Organizing Behavior:** Measure how many labels are created and applied. If our tag UI is improved, more users might start tagging (especially casual ones who didn't before). Also track number of items with tags, etc.
- **Multi-Select Usage:** Count how often users enter multi-select mode and perform batch actions. An increase would show the feature is more discoverable/usable now.
- **Navigation patterns:** See if users utilize new density settings, switching views or filters more. Also cross-device usage – e.g., does a user comfortably move from mobile to desktop (we might see shorter mobile sessions for saving, longer desktop sessions for reading).
- **Performance metrics:** Page load and list scroll performance with large libraries. The new design should not slow things down; in fact optimizing spacing and maybe virtualization of list could help. We'll monitor if any rendering changes impacted performance (time to interactive, frame rate during scroll with 1000 items, etc.).

By combining qualitative feedback and quantitative data, we'll iterate on the design system to ensure it truly improves the user experience as intended.

Implementation & Next Steps

Design Tokens and Systemization

Implementing this overhaul will involve creating a robust **design token** structure for Omnivore: - Define global variables for spacing (e.g., `--space-1: 4px`, `--space-2: 8px`, etc.), colors (`--color-bg-primary`, `--color-text-secondary`, etc.), font sizes (`--font-size-title`, etc.), and radii. - These tokens will be documented (perhaps updating `DESIGN-TOKENS.md` with the new values and usage guidelines). - Use a consistent CSS-in-JS approach (since Omnivore uses Stitches), replacing hardcoded values with tokens. This will resolve the inconsistent spacing issue noted and make future theming (e.g., a light theme or custom themes) much easier. - Clean up any legacy styles that conflict, unify on one layout grid system.

Technical Feasibility & Constraints

We should verify technical considerations: - The front-end is a modern Vite-based app (likely React/Next.js). The new components (cards, menus, etc.) should be implemented as reusable components in the codebase (e.g., a `<LibraryCard>` component with props for article data, which uses subcomponents for metadata, tags, etc.). - The design should account for **internationalization**: text lengths in other languages might be longer. Ensure our truncation and spacing can handle it (especially for metadata line and button labels). - **Performance:** Adding thumbnails and images could slow things if not handled (but it seems current design already has them). Use lazy-loading for images outside of viewport. The skeletons for loading should replace large spinners that might have been used. - **Backward compatibility:** Ensure that if some users still use an older version or if not all changes can roll out at once, nothing breaks. But since it's a web app, likely we deploy at once. - **Testing:** We will write unit/integration tests for interactive behavior (like multi-select logic, swipe gestures – possibly using Jest/React Testing Library and browser testing for swipe). - Work with developers to ensure keyboard nav is fully implemented (tabindex ordering, event handlers for key shortcuts, etc.).

Deliverables and Timeline

We plan to deliver the following artifacts for implementation: - **High-fidelity Mockups:** Screens illustrating the library in various states (default view, compact vs spacious, mobile vs desktop, multi-select mode on, etc.). Also, detailed mockups of components like the action menu, tag editor dialog, and reader view toolbar if needed. These will be done in Figma and shared with the team. - **Interactive Prototype:** A clickable prototype demonstrating key interactions (hover on card, swipe on mobile, open menu, etc.) to help developers and stakeholders experience the flow. - **Design Specification Document:** This document (which you are reading) serves as a basis, and we will add any additional implementation notes – e.g., exact hex codes, pixel measurements, example CSS for a card component, etc. - **Contribution Guidelines Update:** Since this is an open-source project, we will update the `CONTRIBUTING.md` or design system docs to guide future contributors on using the design system – e.g., "use existing components, don't introduce new colors outside the palette, follow accessibility guidelines," etc. - **Accessibility Report:** An audit report ensuring the new UI meets standards, with any remaining issues noted for fixing.

Timeline: We'll proceed with a phased approach: 1. **Design Finalization (1-2 weeks):** Incorporate feedback from maintainers and community on these proposals. 2. **Implementation of Core Components (2-4 weeks):** Build the new card component, navigation bars, and implement new styles/tokens. Launch behind a feature flag if needed. 3. **Testing & Iteration (2 weeks):** Do the user testing and A/B tests as outlined, gather data. 4. **Refinement and Full Launch (1 week):** Polish any rough edges from testing feedback (e.g., if users struggled with swipe, adjust it or add a tutorial), then roll out to all users. 5. **Post-launch Monitoring (ongoing):** Watch analytics, fix bugs, and plan any further enhancements (maybe bring some conceptual features like AI-assisted tagging or advanced filters in future once the foundation is solid).

Conclusion

The proposed design system for Omnivore aims to respect the **content-first** philosophy – making the user's saved articles the star of the show – while equipping power users with efficient tools to manage and navigate large libraries. By addressing key pain points (density, discoverability, consistency) and grounding the design in proven UX principles (visual hierarchy, progressive disclosure, feedback loops), Omnivore can significantly enhance its user experience.

Importantly, this redesign keeps in mind **future scalability** (a tokenized system and modular components) and **inclusivity** (accessible for all users, keyboard and screen-reader friendly). It also learns from the competition: adopting strengths like Pocket's robust tagging and Instapaper's clean reading interface, and then going further with Omnivore's unique open-source ethos.

We're excited to collaborate with the community on bringing this design to life. With careful implementation and user-centric iteration, Omnivore will not only match its competitors in usability and polish, but potentially surpass them by catering to the needs of serious readers and researchers. The end result will be a **delightful, intuitive, and powerful reading platform** that remains welcoming to newcomers and deeply satisfying to veteran users.

Let's build a **beautifully functional** Omnivore, together – one that rewards our users' curiosity and respects their need for focus and organization. We believe the changes outlined here will make Omnivore a joy to use, whether you have 5 articles or 5,000.

Next steps: Review this proposal, provide feedback, and start implementation on the highest-priority items (we suggest tackling the library card and navigation components first, as they have the broadest impact). We look forward to input from contributors and users to refine these ideas and ensure Omnivore continues to grow as a community-driven project.

---

1 6 Pocket vs. Instapaper. I've been using the wrong service all... | by Marius Masalar | Adventures in Consumer Technology | Medium
https://medium.com/adventures-in-consumer-technology/read-it-later-pocket-vs-instapaper-78e7db35277f

2 Using Swipe to Trigger Contextual Actions - NN/G
https://www.nngroup.com/articles/contextual-swipe/

3 design patterns - Hover on a card with clickable links - User Experience Stack Exchange
https://ux.stackexchange.com/questions/140491/hover-on-a-card-with-clickable-links

4 5 Instapaper vs. Pocket: Which is best? [2025] | Zapier
https://zapier.com/blog/instapaper-vs-pocket/