Lora Chisholm

Web Site Proposed Project Name

WDD 131

Spring 2025

Project Name: Kavia

A group of people sitting around a table with a plan

AI-generated content may be incorrect.

Kavia will be a web site and application designed to help users organize, reflect on, and decide the fate of their unfinished tasks and creative efforts (projects). These may include half-read books, paused passion projects, ideas scribbled in notebooks, gardening half done, or side projects that lost momentum. There also will be a place to do smart goals, and you will be able to set goals, list goals, and delete goals or check them off as completed.

The purpose is to provide a thoughtful, elegant space where users can log these items, review them periodically, and take one of three actions.

1. Resume it
2. Archive it for later
3. Retire and forget it

Unlike a to do list kavia isn’t about productivity hacks. It’s about intentionality, clarity, and mental decluttering, moving forward by letting go or reengaging with purpose.

My Core Features will be:

1. Home Page with at least 1 child page
   1. Home page: Overview of what Kavia is, a featured “unresolved” item, and CTA to manage your personal backlog.
   2. Child Page: Full task dashboard with forms to add, edit or delete entries, sorting and filtering tools.
2. Hosted on GitHub Pages
   1. Will be deployed to a public GitHub repository using HTML/CSS/JS.
3. Significant textual content
   1. Descriptions of each task/project
   2. Explanations for why it was paused
   3. Reflection prompts
   4. Category summaries and decision notes
4. Relevant images
   1. Personal image suggestions: Books, notebooks, tools, etc.
   2. User submitted project
5. Working links
   1. Navigation bar linking Home and Dashboard or other pages
   2. Footer and other buttons with transitions
6. Wireframes matched with CSS layout
   1. CSS layout matches wireframes using the flexbox/Grid, mobile first responsive design.
7. Responsiveness
   1. Viewport aware layout with breakpoint adjustment for phone/tablet/desktop
8. Accessible and readable
   1. Color contrast and type hierarchy will ensure WCAG compliance
      1. Inputs to have proper labels and ARIA support
      2. Lighthouse score of greater than or equal to 96
9. JavaScript Requirements
   1. Functions: modularized actions (addtask, updateStatus, renderList)
   2. DOM interaction: dynamically update lists and details
   3. Conditional branching: sort tasks based on age, urgency, type
   4. Objects: store each entry as a structured object
   5. ES Modules: separate data logic, UI logic and storage logic

**Additional Desired Features (Maybe as time permits):**

1. Soft Reminder System
   1. Instead of deadlines or notification, Kavia gently recommends items that have been paused for a while
   2. Questions like “It’s been 42 days since you last looked at XXXX. Do you want to revisit it?
   3. Uses time difference (based on original entry date + last viewed date)
   4. Gentle, non-pushy.
2. Time aware UX is powerful and rare in student projects, creating meaningful engagement without stress.
3. Guided Decision Flow

Instead of just selecting Resume, Archive, Retire, users go through a 1 question path:

Do I still care about this? Yes🡪 Do I have the time to finish it soon? 🡪 yes 🡪Resume Else 🡪 Archive or Retire

Simple yes/no choices guide user to a decision

Adds a sense of self coaching without pressure

Why it stands out: This turns Kavia into more than a tracker, it feels like a thinking partner.

1. Multi Category Smart Sorting

Instead of just sorting alphabetically or by date, users can sort by:

Category

Effort level

Estimated time remaining

This will , use multiple data properties per task object, advanced sort() logic + UI dropdown or filter buttons, and give control and useful perspectives without clutter

1. Session Based Task Drafting

Allow users to start writing or editing a new task entry, but save it as a draft to not be immediately added to main list.

Useful for pausing mid input and returning later

Drafts are autosave to local storage

Adds a drafts section to manage incomplete entries

Will be like a writing app that adds real world usability for thoughtful users who don’t want to commit instantly.

1. Quick Add Mode

User would be able to press a specific button to open the quick add input field anywhere on the page.

* 1. Uses keydown event listener
  2. Focuses on a minimal input overlay
  3. Captures title and optional category, then inserts task

This will show DOM events, introduce power user interaction style in a clean, minimal way.

1. Dashboard display sidebar: text only
   1. Will show
      1. You have:

-4 Creative items

-2 Technical items

-1 Admin items

-3 gardening items

-3 School items

-5 Personal Care items

-9 Smart Goals

This will be using and calculated by reduce() + map()

Updates in real time as tasks are added/removed/edited

Completely non-graphical, high data clarity, using smart use of array logic, to improve user understanding without adding bulk to capabilities.

1. Undo Recent Action
   1. This is important so a user can change a task status, and offer a small Undo button for 10 seconds.

Uses setTimeout() to clear the undo buffer

Keeps a temporary copy of the last state in memory

And possibility do a subtle banner, like – Task moved to Archive—undo?

This will implement controlled state history like mini versioning, and will show real command of UX friendly js.

Plan Table.

1. Colors: To be determined. (Probably softer and gentler with accents)

10. Layout: To be determined. (Probably more traditional)

The Kavia Plan by Lora Chisholm

|  |  |  |
| --- | --- | --- |
| Layer | File Types | Role in Kavia |
| Markup | .html | Defines the pages (index.html, dashboard.html, site-plan.html) |
| Style | .css | Global and component styles in  /styles/main.css  This will be written with responsive flex/grid layout and accessible color/topography rules |
| Logic | .js & .mjs | Vanilla JavaScript written as ES modules  .app.mjs (entry point)  .data.mjs (task objects & storage helpers)  ui.mjs (rendering and & DOM listeners)  utils.mjs (sort, filter, undo helpers) |
|  |  |  |
|  |  |  |

The point is to have everything run in the browser, with no external frameworks, build tools , or other back end so the project remains lightweight, portable, and easy to host on GitHub pages while still demonstrating modular maintainable JavaScript architecture. I hope I can do it, I am excited to challenge myself!