

Study Sprint - Architecture Redesign Notes

Purpose of this note

This note documents the architectural and UI redesign work completed on the app documenting

- what changed
- why it changed
- how the structure improved
- which design decisions were intentional

Initial app state

The app originally had a flatter and more fragmented structure than the actual data model supported.

The conceptual data hierarchy of the app is:

****Subject -> Assignment -> Task****

However, the earlier navigation and screen structure did not consistently reflect that hierarchy. In practice, this caused duplicated views, weak context, and screens that felt disconnected from their parent entities.

Main problems in the earlier version:

- top-level tabs included separate screens for ****Subjects****, ****Assignments****, and ****Tasks****
- tasks existed as their own top-level area even though they are children of assignments
- assignments also felt partially detached from subjects
- repeated CRUD patterns created too many screens and too much UI duplication
- many screens relied on older 'defaultStyles' patterns rather than a clearer component/card-based structure
- raw timestamps and brittle date inputs created poor usability
- create/edit flows were separated even when both used the same form structure

Core redesign goal

The redesign aimed to make the app follow its real conceptual model more closely

****Dashboard -> overview****

****Subjects -> actual content hierarchy entry point****

****Timer -> separate utility tool****

And inside the content structure:

****Subject -> Assignment -> Task****

The main design philosophy throughout the redesign was:

- calm
- intuitive
- minimal
- low visual noise
- predictable interaction patterns
- stronger hierarchy
- less redundancy

1. Navigation architecture redesign

Previous navigation problem

The earlier tab setup exposed too many top-level destinations:

- Dashboard/Home
- Subjects
- Assignments
- Tasks
- Timer

This was a problem because Tasks and Assignments were not truly top-level concepts in the product

model. They belong to parent entities.

This caused:

- duplicated list views
- extra screens with weaker contextual meaning
- cognitive overload
- a flatter app structure than intended

Navigation redesign decision

The top-level tabs were simplified to:

- **Dashboard**
- **Subjects**
- **Timer**

Why this is better

This better matches the app structure:

- **Dashboard** = overview and cross-cutting information
- **Subjects** = main entry point into academic content
- **Timer** = standalone utility

Assignments and tasks were removed from the top-level tab bar and are now only accessed through the hierarchy:

- subject details contains assignments
- assignment details contain tasks

Effect of this change

This reduced redundancy and made the app feel more coherent. It also aligned the user flow with the actual data relationships rather than exposing every model as its own first-class navigation area.

2. Content hierarchy redesign

Original issue

Subjects, assignments, and tasks were partially treated like parallel entities rather than nested entities.

This weakened context:

- a task could appear without clearly indicating its assignment or subject
- an assignment could feel detached from its subject
- the hierarchy existed in the database but was not always communicated in the UI

Redesign decision

The hierarchy was made explicit in the UI

- subjects page shows only subject cards
- subject detail page becomes the main hub for the selected subject
- assignment detail page becomes the main hub for the selected assignment
- task detail page becomes the main hub for the selected task

Result

The app now better communicates:

- where the user is
- what the current item belongs to
- how to move deeper into the structure

3. Subject list redesign

Original issue

The subject list screen contained too much management UI directly in the list:

- edit buttons
- delete buttons
- progress bar
- cluttered card actions

This made the subject list feel like a control panel rather than a clean browsing screen.

Redesign decision

The subject list was simplified into clean, tappable subject cards.

Each subject card now focuses on:

- subject title
- optional description
- active/inactive pill
- subject-specific color identity
- full-card tap to open subject details

Removed from list cards

- inline edit button
- inline delete button
- progress bar
- management-heavy layout

Why

The list screen should be for browsing and selecting. Management actions belong inside the detail screen for that entity

Result

The subject list became calmer, easier to scan, and more aligned with the principle that cards should act as entry points rather than mini dashboards.

4. Subject detail screen redesign

Original issue

The subject detail screen was still using older styling patterns and did not fully behave as the subject "hub".

Redesign decision

The subject detail screen was redesigned as the main management hub for the specific subject.

It now includes:

- a subject summary card
- subject status and metadata
- subject-specific color styling
- subject actions (edit/delete)
- create assignment action
- assignment sections below the subject summary

Why this matters

This screen now clearly answers:

- what this subject is
- how active/complete it is
- what assignments belong to it
- what actions can be taken at the subject level

This better reflects the product structure.

5. Assignment detail screen redesign

Original issue

Assignments were previously styled more generically and did not always preserve clear subject context.

Redesign decision

The assignment detail screen was redesigned to:

- function as the hub for one assignment
- show clear metadata
- show progress through child tasks
- expose only assignment-relevant actions
- preserve visual inheritance from the parent subject

New structure

The assignment detail screen now includes:

- assignment summary card
- subject context pill
- deadline metadata

- progress section showing task completion
- task list organized by completion state
- create task action

Why

Assignments are not independent from subjects. The redesign makes that relationship visible without making the screen visually noisy.

6. Task detail screen redesign

Original issue

Tasks were at risk of losing hierarchical context because they are the deepest level in the model.

Redesign decision

The task detail screen was redesigned to preserve parent context explicitly.

It now includes:

- task summary card
- subject context pill
- assignment context pill
- task state and metadata
- parent-aware styling

Why

Tasks are the easiest place for context loss. By surfacing both subject and assignment, the user always knows where the task belongs.

7. Upsert-based form architecture

Original issue

The app originally used separate create and edit screens for the same entities, even when both screens used almost identical form fields and validation.

This created:

- duplicated UI code
- repeated logic
- more route clutter
- higher maintenance cost

Redesign decision

Create/edit flows were consolidated into upsert-style screens where appropriate.

Implemented

- 'upsertSubject'
- 'upsertAssignment'
- 'upsertTask'

Pattern used

The form checks whether an ID exists:

- if no ID exists -> create mode
- if an ID exists -> edit mode

The same screen handles:

- initial default values
- loading existing data when editing
- submit behavior for insert vs update

Why this is better

This reduces duplication while keeping the form styling and validation consistent.

Tradeoff

This introduces a little more branching inside a single screen, but the tradeoff is worth it because create and edit flows are structurally very similar for these entities.

8. Shared utility extraction

Original issue

Small but important logic was duplicated across screens.

Redesign decision

Shared helpers were moved into reusable modules.

Added shared utilities

- 'lib/date.ts'
- 'lib/subjectColors.ts'

Date formatting utility

A shared date formatting module was introduced to standardize:

- date-only display
- date-time display

This replaced raw timestamp rendering such as ISO strings.

Why

Raw database timestamps were ugly and difficult to read. Formatting them centrally improves both UI quality and consistency.

Subject color utility

A shared subject color configuration was introduced to centralize:

- available subject colors
- subject color type
- mapping from logical color key to visual values
- helper function for retrieving the correct color set

Why

This prevents duplicated color logic and ensures consistent use of subject-specific accent colors across screens.

9. Subject color system and inheritance

Original issue

The app initially relied mainly on global app accent colors, which made it harder to preserve identity across nested subject content.

Redesign decision

Subjects were given their own user-selectable accent color.

Color choice

The user chooses from a controlled palette instead of arbitrary colors.

Why a controlled palette

This preserves:

- aesthetic consistency
- readability
- predictable contrast
- low visual noise

How color is used

The subject color is used as a contextual accent, not a replacement for the whole theme.

Used on:

- subject card border
- subject preview card
- subject pills
- inherited borders and indicators in assignment/task screens
- progress bars and completion indicators where appropriate

Important design rule

The subject color was not used for everything.

Primary action buttons such as:

- create
- save
- login

remain on the **global app accent**

Why

This preserves a consistent interaction language:

- app accent = primary action
- subject color = content identity / context

This separation was an intentional design decision.

10. Card-based UI redesign

Original issue

Several screens still relied on older layout/styling conventions that felt less coherent and more cluttered.

Redesign decision

The redesign shifted toward a more consistent card-based interface using:

- bordered surface cards
- semantic Tailwind theme classes
- more restrained spacing
- contextual pills
- reduced action clutter

Card design goals

- easier scanning
- stronger visual hierarchy
- fewer floating controls
- more predictable composition

Result

The app feels more structured and less noisy.

11. Metadata and pill system

Original issue

Status and metadata were previously shown in less consistent ways, including redundant indicators.

Redesign decision

Metadata display was standardized using pill elements for small contextual information.

Examples include:

- subject name
- assignment parent
- deadline
- active/inactive state

Important cleanup decision

Some pills were removed when they became redundant.

Example:

- completed/in-progress pill was removed in places where the same information was already communicated by a checkbox or progress structure

Why

This reduced duplication and visual clutter.

12. Progress display redesign

Original issue

Progress indicators were previously placed too aggressively in list views where they created clutter.

Redesign decision

Progress bars were kept only where they make structural sense:

- subject detail
- assignment detail

Why

These are hub screens where progress is meaningful.

Progress bars were intentionally removed from places like subject list cards where they overloaded a browsing view.

Additional improvement

Progress is now shown with both:

- a percentage bar
- an 'x / y' completed count
- remaining item count text

This makes progress more understandable than a bar alone.

13. Authentication-related debugging insight

During development, a major debugging issue turned out not to be screen architecture at all, but session/auth failure.

This surfaced as:

- fetch failures
- apparent data loading errors
- misleading "network request failed" behavior

Takeaway

Auth state and session expiry can easily masquerade as architecture or fetch bugs.

This reinforced the importance of:

- clearer auth handling
- not assuming every fetch failure is a UI/data issue
- checking session state early when debugging

14. Time handling and dual-boot issue insight

A separate development issue was discovered related to system time mismatch in a Windows/Linux dual-boot environment.

Although not an app architecture feature directly, it affected development by causing:

- failed requests
- misleading network/auth behavior

Development takeaway

System time correctness matters for:

- authentication
- HTTPS
- tokens
- scheduled features

This was important context during debugging and implementation.

15. Notifications and reminders

Assignment creation/updates included work around deadline reminders.

Behavior

When an assignment has a valid future deadline:

- a reminder can be scheduled
- previous reminders are updated/cancelled when necessary
- notification IDs are stored through async storage helpers

Why this matters architecturally

This means assignment upsert behavior is not only CRUD. It also coordinates:

- persistence
- reminder scheduling
- reminder cleanup

This is relevant for the final report because it shows that form flows have side effects beyond database writes.

16. General design principles used across the redesign

Principle 1: reflect the true data hierarchy

The UI should match the conceptual model:

- subjects contain assignments
- assignments contain tasks

Principle 2: remove redundant top-level structure

Not every data model deserves a top-level tab.

Principle 3: keep list screens for browsing

Heavy management actions should live in detail screens.

Principle 4: preserve context

The deeper the user goes, the more important parent context becomes.

Principle 5: use color as identity, not decoration

Subject colors provide contextual identity without overwhelming the UI.

Principle 6: keep primary actions globally consistent

App accent remains the primary action language.

Principle 7: reduce duplication

Reusable upsert screens and shared utilities reduce maintenance cost.

Principle 8: avoid visual noise

Redundant pills, repeated indicators, and crowded cards were intentionally reduced.

17. Current architecture summary

Tabs

- Dashboard
- Subjects
- Timer

Hierarchy

- Subject
 - Assignment
 - Task

Reusable utilities

- date formatting
- subject color mapping
- progress checks
- async storage notification helpers

Form strategy

- upsert-style forms for core entities

Design system

- NativeWind
- semantic Tailwind tokens
- shared card/pill patterns
- subject color inheritance

18. Final outcome

The redesign changed the app from a flatter CRUD-style structure into a more coherent hierarchical study workflow.

The main improvements were:

- better navigation structure
- reduced redundancy
- clearer subject/assignment/task relationships
- stronger contextual design
- less cluttered list screens
- reusable form patterns
- centralized shared helpers

- more polished and consistent UI

Overall, the app now better reflects its intended purpose as a study productivity tool organized around the academic structure of:

****Subject â Assignment â Task****

19. Possible items to mention later in the final report

These are likely worth discussing explicitly:

- why Tasks and Assignments were removed as top-level tabs
- why subject detail and assignment detail were turned into hubs
- why create/edit were merged into upsert patterns
- why a controlled color palette was used instead of arbitrary colors
- why subject color was used for context, not for primary actions
- why duplicated metadata indicators were removed
- why shared date formatting and subject color utilities were extracted
- how preserving hierarchy improved usability
- how debugging/auth issues affected development decisions