

Timer Refactor and Verification Work Report

#Overview

Today the timer screen was worked on with a narrower goal than yesterday: not new interaction features, but cleanup, readability, and making the existing timer flow easier to understand and maintain.

This follows directly from yesterday's state. The April 24 note ended with two follow-up items:

- re-refactor the timer code so it becomes easier to read and work on
- keep the dependency situation stable after the NativeWind version mismatch had broken the screen

Today's work focused on the first of those. The interaction model was kept the same, but the internal structure of 'timer.tsx' was cleaned up so the current hold-to-cancel and focus-mode behavior is easier to inspect without splitting the code into hooks or separate files.

#ImplementedFeatures

#TimerCodeRefactor

Refactored the timer screen structure inside 'app/(tabs)/timer.tsx':

- renamed the component from 'App' to 'TimerScreen'
- renamed unclear callbacks such as the old generic start-animation function into 'startTimerSession'
- grouped the file more clearly into constants, animated values, refs, derived values, actions, render helpers, and JSX
- renamed vague animated/interpolated values to clearer names such as 'startButtonOpacity', 'startButtonTranslateY', and 'pickerOpacity'

This did not change the screen architecture into multiple files. The cleanup stayed local to the timer file so the animation flow is still easy to inspect in one place.

#CleanupHelpers

Extracted repeated timer cleanup work into small local helpers:

- added 'clearCountdownInterval()'
- added 'clearCancelHoldTimeouts()'
- added 'stopRunningAnimations()'
- added 'resetSessionValues()'

Before this, the same interval, timeout, and animation-reset work was spread across multiple callbacks. Pulling it into helpers makes it easier to follow what happens when a session starts, finishes, or is cancelled.

#RenderStructureCleanup

Cleaned up the render section so it is easier to read:

- moved repeated inline layout styles into named 'StyleSheet' entries
- extracted the timer picker item rendering into a local 'renderTimerItem(...)' helper
- kept the JSX order aligned with the visible screen layers: overlay, start button, cancel button, countdown, duration picker, and task details

This mainly improves scanning. The old file worked, but the render section made you jump between inline style objects and animation expressions to understand each layer.

#CommentAndNamingPass

Added a small number of comments only where the code was genuinely hard to follow:

- clarified that 'timerAnimation' owns real timer progress
- clarified that 'cancelOverlayAnimation' is only a temporary visual offset during hold-to-cancel
- clarified that 'startProgressAnimation(fromY)' resumes overlay progress from the current Y position
- clarified why cancel acceleration starts after a short delay

The aim was not to comment every line, but to explain the parts that are hard to infer just by reading the code.

#StateResetTightening

Made the session cleanup more explicit:

- reset 'sessionStartedAtRef' and 'sessionDurationMsRef' when a session ends
- reset cancel-hold flags during session cleanup
- made 'finishTimer()' explicitly clear the countdown interval before running exit animations
- kept the existing unmount cleanup so intervals, timeouts, and running animations are not left behind if the screen disappears mid-session

These are small changes, but they make the timer lifecycle more predictable and reduce the amount of stale mutable state left around after finish or cancel paths.

#LearningNotes

#ReadableCodeVsNewFeatures

Today's timer work was a good reminder that "more maintainable" does not always mean "more abstract."

For this screen, the right cleanup level was:

- better names
- smaller local helpers
- clearer grouping
- a few targeted comments

The wrong cleanup level for the current stage would have been moving the logic into extra hooks or files too early, because that would make it harder to inspect the animation flow while the interaction is still being tuned.

#MutableRefOwnership

The timer file still relies heavily on refs because several parts of the interaction are long-lived and imperative:

- active countdown interval
- running start animation
- running progress animation
- delayed cancel-preview start
- hold-to-cancel completion timeout

The cleanup made this easier to see by separating refs that hold animated values from refs that track mutable timer/session ownership.

#CurrentState

Compared with yesterday, the timer interaction model is mostly the same, but the code behind it is more structured.

The current implementation:

- keeps the red overlay model used yesterday
- keeps 'timerAnimation' as the real progress owner
- keeps 'cancelOverlayAnimation' as the temporary hold-preview layer
- keeps the delayed hold acceleration and release recovery flow
- keeps all timer logic local to 'timer.tsx'
- is now easier to read because repeated cleanup and render logic have been extracted into named local pieces

This means today's work was mainly a recovery and consolidation pass after yesterday's interaction-heavy changes and the earlier dependency-related breakage.

#Verification

Today's static checks passed after the refactor:

```text

```
npm run lint
exited successfully
""
```

```
""text
npx tsc --noEmit
exited successfully
""
```

```
""text
git diff --check -- 'app/(tabs)/timer.tsx'
exited successfully
""
```

There was no new timer commit for today at the time of writing this note. The summary above is based on:

- the current working-tree diff for 'app/(tabs)/timer.tsx'
- the verification commands run after the refactor
- yesterday's note and timer history for context

I did not do a live Expo interaction test inside this note workflow, so runtime behavior is verified statically plus by code review rather than by manually pressing through the UI.

---

### ## #FilesChanged

Main file worked on:

```
""text
app/(tabs)/timer.tsx
""
```

New note added:

```
""text
notes/work-report-timer-2026-04-25.md
""
```

---

### ## #Conclusion

The main timer work today was not adding new features, but making yesterday's feature-rich timer implementation is easier to continue working on.

The result is a timer file that keeps the same focus-mode and hold-to-cancel behavior, while being more readable, more structured, and easier to maintain. The biggest improvement is that the important ideas in the file now have clearer names, clearer ownership, and clearer cleanup paths.

The timer is now considered finished and ready to implement into the rest of the project.