

Timer UI and Countdown Work Report

#Overview

Today the standalone timer screen was developed further before wiring it into the task system.

The main focus was improving the timer interaction and learning how the React Native animation flow works. The timer is still being treated as its own tab for now, with placeholder task data used in place of real task integration.

#ImplementedFeatures

#TaskInformationPlaceholder

Added placeholder task information to the timer screen:

- Placeholder task name
- Placeholder task description
- Fade-in animation when the timer starts
- Fade-out animation when the timer finishes

This prepares the timer UI for the later task integration, where the placeholder values can be replaced by real task data.

#AdjacentTimerFade

Updated the timer duration selector so adjacent numbers fade away when the timer starts:

- The centered selected value remains visible
- Neighboring values fade out during the active timer state
- Neighboring values are intended to fade back in after the timer finishes

This was implemented by separating the normal picker opacity from the active timer opacity and combining them with 'Animated.add' and 'Animated.multiply'.

#MeasuredTimerHeight

Started adjusting the timer overlay to use the measured screen/container height:

- Added 'containerHeight'
- Added 'onLayout' to measure the actual timer screen area
- Updated timer overlay movement to use the measured container height

This was done because the full window height does not always match the visible tab screen area when headers, tab bars, or safe areas are involved.

#CountdownDisplay

Added countdown display logic:

- Added 'timeRemaining'
- Added 'selectedIndex'
- Added 'formatTime(totalSeconds)'
- Converted the selected timer value into a 'MM:SS' display while running
- Added 'TIMER_UNIT_IN_SECONDS' so timer values can behave as seconds during development and minutes later

Current development behavior:

- 'TIMER_UNIT_IN_SECONDS = 1'
- Selecting '5' means a 5-second timer

Planned production behavior:

- 'TIMER_UNIT_IN_SECONDS = 60'
- Selecting '5' means a 5-minute timer

#CountdownFadeControl

Started separating countdown visibility from the rest of the timer UI:

- Added 'countdownAnimation'

- Added 'showCountdownText'
- Began separating the 'MM:SS' countdown fade from the button and picker fade
- Fixed the nested animation callback syntax after adding the countdown fade-out flow

The goal is for the countdown text to fade out first, then for the button and adjacent timer values to fade back in after the countdown is gone.

#LearningNotes

#ReactState

Worked with several pieces of state:

- 'duration' stores the selected timer value
- 'isRunning' tracks whether the timer is active
- 'timeRemaining' stores the countdown value
- 'selectedIndex' identifies which duration is selected
- 'showCountdownText' controls whether the selected item renders as 'MM:SS'
- 'containerHeight' stores the measured height of the timer screen

Important distinction:

- State values trigger re-renders when changed
- Animated values drive smooth visual changes without normal React state updates on every animation frame

#Hooks

Clarified where hooks are allowed:

- 'useState', 'useRef', 'useEffect', and 'useCallback' must be called inside the component
- Hooks must not be placed inside callbacks, conditionals, loops, or event handlers
- 'useEffect' dependency arrays must be inside the 'useEffect(...)' call

One key bug came from an effect without a proper dependency array. Because the countdown updates state every second, the effect ran every second and reset the red overlay position.

#AnimationFlow

The timer now uses multiple animated values:

- 'timerAnimation' controls the red overlay movement
- 'buttonAnimation' controls the start button and inactive timer value visibility
- 'taskDetailsAnimation' controls the placeholder task information
- 'countdownAnimation' controls the 'MM:SS' countdown visibility

The main lesson was that one animation value should not control too many unrelated visual states. Separate animation values make it easier to control the order of fade-out and fade-in transitions.

#Verification

The timer file syntax issue around the end of the 'animation' callback was fixed.

Current lint result:

```
““text
npm run lint
exited successfully
““
```

The previous parse error was caused by mismatched closing braces/parentheses near the nested '.start(...)' callbacks at the end of the animation sequence.

The remaining behavior to confirm is the final transition order:

- 'MM:SS' countdown should fade out
- selected text should switch back to the normal timer value while hidden
- adjacent timer values should fade back in
- start button should fade back in

#FilesChanged

Main file worked on:

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

New note added:

```
““text
notes/work-report-timer-2026-04-22.md
““
```

#Conclusion

The timer UI moved from a basic animated duration selector toward a more complete timer experience. It now has placeholder task information, a ‘MM:SS’ countdown concept, measured layout support, and separate animation values for different UI elements.

The syntax error at the end of the animation callback has been fixed and lint now passes. The remaining immediate work is to finish confirming the final fade-out/fade-in ordering so the countdown disappears cleanly before the picker and start button return.