

Focus, Dashboard, And Progress Model Work Report

#Overview

Today the timer work moved from a sprint-only model toward a more general session flow that can support both focused work and breaks.

The main goal was to start closing the vision gap around 'focus -> break -> continue', while keeping the implementation local to the existing timer route instead of introducing a larger navigation or state-management rewrite.

The work therefore covered both app-side session-model changes and the Supabase function updates needed to make the new flow actually start and finalize sessions correctly.

Later in the same work session, the scope also expanded into the dashboard and the progress presentation on the detail screens so the app better matches the remaining vision-gap plan.

The scope then expanded one step further into first-time-user friction, so the work also covered a guided onboarding path and clearer empty states for new accounts.

Later still, the work expanded beyond the app itself into the signup-confirmation path around account creation. That included auth-screen behavior fixes, a shorter guided-setup timer for quick verification, a minimal confirmation landing page for VPS deployment, Caddy routing, and a less boilerplate-looking confirmation email template.

#ImplementedFeatures

#GeneralSessionModel

Changed the local timer model from a sprint-specific structure into a more general session structure:

- added 'SessionType' in 'lib/types.ts'
- introduced the session types:
 - 'focus'
 - 'short_break'
 - 'long_break'
- replaced the old 'ActiveSprint' shape with 'ActiveSession' in 'lib/asyncStorage.ts'
- stored 'sessionType' together with 'sessionId', 'taskId', 'durationSeconds', and 'endTime'

This means the active timer is no longer assumed to always be a task-linked focus sprint.

#TimerSessionStartAndRestore

Updated the timer screen so it can start and restore different session types:

- replaced sprint-specific storage calls with 'GetActiveSession(...)', 'SaveActiveSession(...)', and 'RemoveActiveSession(...)'
- generalized the timer start path into 'startSession(...)'
- passed 'p_session_type' into the Supabase 'start_sprint_session(...)' RPC
- kept task linkage only for 'focus' sessions
- updated the restore logic so a focus session restores by 'tId', while break sessions restore by 'sessionType'

This gives the existing timer screen enough information to behave differently for focus sessions and break sessions without creating a second timer screen.

#DashboardAndTaskIntegration

Updated the surrounding screens so they understand the new active-session shape:

- updated 'app/task/viewDetailsTask.tsx' to read the new active session model
- updated 'app/(tabs)/index.tsx' so the dashboard card can describe either a focus session or a break session
- made the dashboard open the timer with either a task id or a break-session configuration, depending on what is active

This keeps the rest of the app aligned with the timer change, instead of leaving the new session model isolated to one file.

#DashboardProgressAndHistory

Extended the dashboard so it works more clearly as a study-activity overview:

- added a compact 'Study progress' summary near the top of the dashboard
- showed:
 - 'Focus sessions today'
 - 'Minutes today'
 - 'Minutes this week'
- loaded the summary from 'sprint_sessions' instead of from planning data
- added a 'Recent sessions' section showing:
 - task title when available
 - session type
 - duration
 - final status
 - date and time
- added a small 'Recently completed tasks' section based on recent task completion updates

This moved the dashboard closer to the vision requirement that progress should reflect actual study behavior rather than only task structure.

#DashboardLayoutRestructure

Reworked the order of the dashboard sections so the screen reads more clearly as a home surface:

- kept the active-session card at the top when relevant
- placed 'Study progress' before the task lists
- moved 'Tasks with upcoming deadlines' directly under the progress summary
- pushed 'Recent sessions' and 'Recently completed tasks' lower as secondary context
- made the lower history area work as a side-by-side layout when screen width allows it
- changed the dashboard body to a scrollable layout so the extra sections still fit without clipping

The result is a dashboard that moves from orientation, to next action, to history instead of feeling like a stacked report page.

#ConsistentProgressModel

Aligned the progress language across the detail screens so each layer measures one clear thing:

- on the subject details screen, changed the progress label from 'Assignment Progress' to 'Assignments completed'
- added helper text clarifying that subject progress is based only on completed assignments
- on the assignment details screen, changed the progress label from 'Task Progress' to 'Tasks completed'
- added helper text clarifying that assignment progress is based only on completed tasks
- on the task details screen, separated completion state from study activity
- added a dedicated 'Study activity' block showing:
 - tracked focus time from 'tasks.totalTimeInSeconds'
 - completed focus-session count from 'sprint_sessions'
- added an explicit task status label so completion state is not confused with study effort

This made the meaning of progress more consistent:

- 'Subject' now reads as assignment completion
- 'Assignment' now reads as task completion
- 'Task' now reads as study effort plus completion state
- 'Dashboard' now reads as recent study activity

#FirstTimeSetupAndEmptyStates

Added the first guided setup flow so new users are pushed into one clear study path instead of landing in an empty app:

- added a dedicated 'app/setup.tsx' route for first-time setup
- changed signup so a newly authenticated user is routed to setup instead of directly to the dashboard
- built the setup flow as a strict sequence:
 - create first subject
 - create first assignment
 - create first task
 - start first sprint
- updated the subject, assignment, and task creation screens so they can advance automatically to

the next setup step

- removed the setup-breaking success popups between those guided creation steps
- added short auth-screen explanations describing:
 - what the app does
 - why an account exists
 - that study structure and progress follow the user
- added clearer empty states on the dashboard and subjects screen that point the user into guided setup
- tightened the empty-state copy on subject and assignment details so each one points toward the next required object in the hierarchy

This closes a large part of the first-run friction gap without introducing a separate onboarding system or broader navigation rewrite.

#AuthScreenKeyboardHandling

Adjusted the auth screens so text inputs do not stay buried behind the on-screen keyboard:

- updated 'app/login.tsx' so the login content scrolls and shifts upward when the keyboard opens
- updated 'app/createUser.tsx' so the entire create-account content block lifts upward with the keyboard instead of only trying to scroll one input into view
- kept the changes local to the auth screens instead of introducing a broader shared keyboard abstraction

This was aimed specifically at the real usability problem where the password field could end up hidden during login or signup.

#SignupNavigationAndHeaderAlignment

Adjusted the signup screen navigation so it matches the rest of the app more closely:

- removed the temporary in-screen back button experiment from the signup page
- re-enabled the normal stack header for 'createUser' in 'app/_layout.tsx'
- kept signup navigation on the default app-style back arrow instead of a one-off local control

This kept the auth flow visually more consistent with the rest of the route stack.

#GuidedSetupFiveSecondSprint

Changed guided setup so the first sprint can be tested almost immediately:

- updated 'app/setup.tsx' so the setup flow opens the timer with a fixed '5' second duration
- extended 'app/task/timer.tsx' so it can also accept an explicit 'durationSeconds' route param
- kept the rest of the timer behavior unchanged, so the setup-specific shortcut still runs through the same session start, storage, and completion flow as normal timers

This made the first-run path quicker to test without changing the broader timer model back to a special-case setup implementation.

#SignupConfirmationDeployment

Built the first deployable confirmation landing page outside the Expo app:

- added 'deploy/signup-confirmation/site/index.html' as a minimal static confirmation page
- added 'deploy/signup-confirmation/docker-compose.yml' so the page can be served with 'nginx:alpine'
- added a small README for VPS deployment notes and port mapping
- verified the page deployment path together with the external VPS/domain setup already in use

This created a concrete destination URL for signup confirmation emails instead of leaving the email to resolve into a blank or undefined endpoint.

#CaddyAndEmailConfirmationPolish

Finished the external confirmation flow around signup:

- corrected the Caddy reverse-proxy target from container port '8080' to '80' for the 'nginx' confirmation container
- confirmed that the confirmation page then resolved correctly behind the existing Caddy-plus-Docker setup
- replaced the original bare confirmation email body with a cleaner branded HTML email using the existing '{{ .ConfirmationURL }}' placeholder

This moved the signup confirmation flow from a functional but rough setup into something that is both deployable and presentable.

#PostSessionBreakFlow

Added the first real post-session flow in the timer UI:

- after a completed focus session, the timer now shows:
 - 'Start short break'
 - 'Skip break'
- starting the break reopens the same timer route in 'short_break' mode
- after a completed short break, the timer now shows:
 - 'Continue with same task'
 - 'Back to dashboard'
- passed 'returnTaskId' through the route so the timer can return the user to the original task after the break

This is the first implementation of an actual study loop rather than a timer that simply ends and disappears.

#BreakTimerPresentation

Adjusted the timer UI so break sessions read more clearly:

- added a fixed-duration block for break sessions instead of showing the normal duration picker
- used a fixed 5-minute short-break duration for the first implementation
- kept the focus-session picker unchanged
- made the break start button match the existing 'Start Sprint' button styling, but show only 'Start'
- removed the bug where picker or pre-start break elements remained visible on top of the running break session

This keeps the first break flow minimal and visually consistent with the existing timer screen.

#SupabaseFunctionAlignment

Adjusted the Supabase side so the new app flow could actually run:

- updated 'start_sprint_session(...)' to accept 'p_session_type'
- allowed break sessions to start with 'taskId = null'
- aligned the SQL with the real table schema using:
 - 'sessionId'
 - 'taskId'
 - 'userId'
 - 'sessionType'
 - 'countedIntoTaskTotal'
- corrected function-return behavior so the app receives the created session id in the shape it expects
- kept finalize logic so only 'focus' sessions contribute to 'tasks.totalTimeInSeconds'

Without this database alignment, the app-side session model would compile but still fail when starting real sessions.

#ProblemsAndSetbacks

#SchemaMismatch

The main blocker today was that the first SQL version assumed table columns that did not exist in the real Supabase schema.

The actual 'sprint_sessions' table already contained:

- 'sessionId'
- 'taskId'
- 'userId'
- 'plannedDuration'
- 'startedAt'
- 'endedAt'
- 'elapsedSeconds'
- 'status'
- 'countedIntoTaskTotal'

- 'sessionType'

But it did not contain 'createdAt' or 'updatedAt', so the first function version failed at runtime.

#FunctionReturnShape

Another blocker was the shape of the return value from 'start_sprint_session(...)'.

Even after the insert worked, the app still showed:

- 'Session could not be created.'

The issue was not the insert itself, but that the returned value shape did not match what 'getSessionId(...)' was looking for on the app side.

This had to be corrected so the RPC returned the created session id in a directly readable object shape.

#PauseUIScreenOverlap

The first version of the break UI had presentation bugs:

- the pause start button text looked cramped and awkward
- pre-start pause UI stayed visible after the break actually started
- picker or fixed-duration elements overlapped the running break session

This was corrected by hiding pre-start break UI while the timer is running and by reverting the pause start button back to the same visual model as the existing sprint start button.

#ConfirmationRoutePortMismatch

The external signup-confirmation deployment initially failed behind Caddy with 'HTTP ERROR 502'.

The actual issue was not the Docker network arrangement itself, but that the reverse proxy was targeting 'signup-confirmation:8080' even though the 'nginx' container listens internally on port '80'.

Changing the upstream target to the real container port fixed the route.

#CurrentState

The timer flow now goes further than the previous sprint-only model.

The app now supports:

- starting a 'focus' session tied to a task
- starting a 'short_break' session with no task linkage
- storing and restoring the active session with its type
- showing a post-focus decision between taking a break or skipping it
- returning from a completed short break into the same task flow or back to the dashboard
- keeping break sessions out of task time totals

At this point, the app has the first working version of the focus-and-break loop described in the vision plan, even though the cycle logic and long-break offer are not implemented yet.

The dashboard also now gives a clearer answer to:

- 'What have I done today?'
- 'What should I work on next?'

And the detail screens now separate planning completion from study activity more explicitly, which makes the app easier to read without having to infer what each progress bar means.

For a brand-new user, the app also no longer drops straight into a generic empty state after account creation. There is now a clearer route from signup to:

- first subject
- first assignment
- first task
- first sprint

That makes the hierarchy feel more guided and less like a blank structure the user has to interpret alone.

The signup path also now has a more complete confirmation loop around it:

- the auth screens behave more safely when the mobile keyboard opens
- guided setup can launch a very short first sprint for fast verification
- the confirmation email can point to a real public landing page
- that landing page has a working Docker/Caddy deployment path on the VPS
- the email itself no longer looks like a raw boilerplate template

#Verification

During today's work, the following behaviors were verified through implementation checks and runtime iteration:

- the new session model compiles across timer, dashboard, task details, and local storage
- 'start_sprint_session(...)' now succeeds after the Supabase function updates
- the timer can start using the new session-based flow
- break sessions no longer leave the picker or fixed-duration setup visible on top of the running timer
- the dashboard compiles with the new progress-summary, recent-session, and recent-completion sections
- the task details screen compiles with a new 'sprint_sessions'-based completed-session count
- the subject and assignment detail screens now label completion metrics more explicitly
- the new guided setup route compiles and links correctly with the subject, assignment, and task creation flow
- the login and signup screens compile after the keyboard-handling adjustments
- the guided setup route now opens the timer with an explicit 5-second fixed duration
- the deployable signup-confirmation page was brought up behind the VPS Caddy setup after correcting the upstream container port from '8080' to '80'
- the confirmation email template was updated to a cleaner HTML version while keeping '{{ .ConfirmationURL }}' as the actual confirmation link placeholder

Static verification also passed:

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

```
npm run lint
exited with existing warning only in:
- app/task/timer.tsx
""
```

I did not run a live interactive app test for the later dashboard and progress-model changes. That part of the verification is static rather than runtime-confirmed.

#FilesChanged

Main app files worked on:

```
""text
app/task/timer.tsx
app/task/viewDetailsTask.tsx
app/(tabs)/index.tsx
app/(tabs)/subjects.tsx
app/setup.tsx
app/subject/viewDetailsSubject.tsx
app/subject/upsertSubject.tsx
app/assignment/viewDetailsAssignment.tsx
app/assignment/upsertAssignment.tsx
app/task/upsertTask.tsx
app/createUser.tsx
app/login.tsx
app/_layout.tsx
lib/asyncStorage.ts
lib/types.ts
deploy/signup-confirmation/docker-compose.yml
deploy/signup-confirmation/site/index.html
```

deploy/signup-confirmation/README.md
""

New note added:

""text
notes/work-report-timer-2026-05-03.md
""

#Conclusion

The main result today was not just a timer change, but a broader step toward closing the remaining vision gaps around study flow and progress clarity.

The app now has:

- a session model that can represent both focused work and breaks
- the first concrete 'focus -> break -> continue' path from the vision plan
- a dashboard that reflects recent study effort more directly
- detail screens that use more explicit and consistent progress meanings
- a first guided onboarding path that leads a new user from signup to their first workable sprint path
- more usable auth screens when entering credentials on mobile
- a complete basic signup-confirmation flow that now reaches a real deployed landing page and a cleaner confirmation email

The remaining work in this area is now less about inventing the model from scratch and more about extending, polishing, and live-validating the pieces that are already in place.