The Strategic Architecture

Instead of one giant "universal" app, we are building two specialized clients for the same backend.

Phase 1: The Foundation (Day 1)

Goal: Get a blank screen running on your phone with the correct tooling.

1. **Initialize Project (TypeScript + Expo):**
   Bash

   ```
   npx create-expo-app@latest nora-mobile --template blank-typescript
   ```

2. **Install NativeWind (Tailwind):**
   - Setup nativewind and tailwindcss immediately so you can copy-paste your web classes.
   - *Why:* Keeps your design language consistent with almost zero effort.

4. **Install Core Navigation:**
   Bash

   ```
   npm install @react-navigation/native @react-navigation/native-stack @react-navigation/bottom-tabs
   ```

5. npx expo install react-native-screens react-native-safe-area-context

Phase 2: The "Brain" Transplant (Day 2-3)

Goal: Move the logic without moving the UI.

1. Copy-Paste Logic:
   Manually copy these folders from Web to Mobile:

    - src/services/ (API calls)
    - src/utils/ (Formatters, helpers)
    - src/context/ (Auth provider, global state)
    - src/types/ (TypeScript interfaces)
2. The "Search & Replace" Audit:
   You must find and replace browser-specific code in these files:
    - localStorage $\rightarrow$ SecureStore (for Auth Tokens).
    - localStorage $\rightarrow$ AsyncStorage (for non-sensitive settings).
    - window.location $\rightarrow$ Remove this (Navigation handles redirection).
    - cookies $\rightarrow$ Mobile apps rarely use cookies; ensure your API accepts Bearer tokens in headers.

Phase 3: The Skeleton (Day 4)

Goal: Create the empty rooms before filling them with furniture.

1. **Build the Navigation Tree:**
    - **AuthStack:** Login, Signup (No tabs visible).
    - **AppTabs:** Home, Journal, Profile (The main interface).
    - **ModalStack:** Audio Player, Settings (Screens that slide up from the bottom).
2. **Create Placeholder Screens:**
    - Create files for LoginScreen.tsx, HomeScreen.tsx, etc., containing just a <View><Text>Screen Name</Text></View>.
    - Wire them up in App.tsx.

Phase 4: UI Migration (Days 5-14)

Goal: Porting the visual layer.

1. The "Atomic" Components (First):
   Migrate your small UI kit first.
   - Button.tsx: HTML <button> $\rightarrow$ <TouchableOpacity>
   - Input.tsx: HTML <input> $\rightarrow$ <TextInput>
   - Card.tsx: <div> with shadow $\rightarrow$ <View> with elevation (Android) / shadow props (iOS).
2. **The Screens (Priority Order):**
   - **Priority A (Auth):** Login/Signup. High confidence builder.
   - **Priority B (Read-Only):** History lists, Profile.
   - **Priority C (The Core):** The Audio Recording screen. This requires the most work.

Phase 5: The "Native" Audio Feature (Days 15+)

Goal: Rebuilding the recording experience.

1. **Install expo-av:**
   - This is the standard library. Try to implement the recording logic here first.
2. **The Visualizer Challenge:**
   - *Challenge:* Web uses AudioContext to draw waveforms. expo-av only gives you amplitude metering (volume levels).
   - *Solution:* Use the metering data to drive a simple animation (bars going up and down).
   - *Backup Plan:* If this isn't smooth enough, we install react-native-audio-recorder-player and run npx expo prebuild.

Phase 6: Polish & Build

1. **Assets:** Create icon.png and splash.png.
2. **Configuration:** Update app.json with your Bundle ID (e.g., com.nora.app).

3. **Build:**
   Bash

   ```
   eas build --profile preview --platform ios
   ```

Comparison: What changed from the original plan?

Feature	**Old Plan**	**New Recommended Plan**
Code Strategy	Migrate everything at once.	**Copy logic first, rewrite UI.**
Web App	Maybe kill it?	**Keep it alive.** Separate codebase.
Styling	Undecided.	**NativeWind (Mandatory).**
Auth Storage	localStorage.	**expo-secure-store (Mandatory).**
Audio	Web Audio API.	**expo-av with Prebuild option.**