**Claude Code Prompt – Initial Frontend Development**

You are tasked with building the \*\*web frontend\*\* of a cross-platform app. Follow these instructions strictly:

1. Use \*\*Next.js with TypeScript\*\* for web.

2. Use \*\*Tailwind CSS\*\* for styling. All colors, spacing, and fonts must come from shared TypeScript design tokens.

3. State management must be handled using \*\*React Context API\*\*.

4. Implement \*\*Next.js routing\*\* for all pages.

5. Build \*\*shared component logic files\*\* that handle state, behavior, and permissions. Platform-specific wrappers (.web.tsx) should handle styling only.

6. Use \*\*dummy data\*\* for all API responses. Structure data in `/data` folder. Make it easily removable when backend is ready.

7. Implement \*\*roles/permissions logic\*\* in all UI elements; hide buttons or sections if user role does not allow it.

8. Integrate \*\*Socket.IO client\*\* to simulate real-time updates using dummy events.

9. Pages to implement: Dashboard, Profile, Content/List. Components: Buttons, Input Fields, Modals, Cards/Lists.

10. Follow strict TypeScript usage. Every component, context, and dummy API data must be strongly typed.

11. Keep \*\*future mobile integration in mind\*\*: all shared logic must be usable by React Native with NativeWind later.

12. Do \*\*not implement backend logic yet\*\*. Focus on functional frontend with dummy data and real-time simulation.

Deliverables:

- Clean folder structure as specified in roadmap.

- Fully functional frontend UI with dummy data.

- Components should respect permissions and be ready for backend integration.

- All theme tokens and shared logic ready for reuse in mobile.

Always follow the \*\*Full Stack Specification\*\* provided for technologies, TypeScript usage, real-time handling, and permissions.

Always clean up the codebase after you are done, there should not be any obsolete code/unused code, folders or files