**ReAlign 2.0 UI & Navigation - Implementation Guide**

**Overall Goal:** Implement the new UI and navigation for ReAlign 2.0, focusing on a modern, responsive, and user-friendly experience based on the provided design documents.

**Phase 1: Setup & Core Styling Foundation**

This phase ensures Tailwind CSS is configured with the new brand identity.

* **Step 1.1: Update Tailwind Configuration**
  + **File:** tailwind.config.ts
  + **Action:** Modify the theme.extend.colors and theme.extend.fontFamily sections in tailwind.config.ts to reflect the color palette and typography specified in the "Design Guide" (Section 3: Brand Identity & Tone of Voice) and "Design Brief - Gregory John inspo.docx".
  + **Reference:**
    1. Primary Color: Deep Blue (#1E3A8A)
    2. Secondary Colors: Teal/Green (#2D6A4F, #2CA58D), Muted Orange/Gold (#E76F51, #F4A261)
    3. Neutral Colors: Dark Gray (e.g., #2D3748), Light Gray (e.g., #A0A0A0)
    4. System Colors: Soft Red (e.g., #E63946), Success Green (#2CA58D)
    5. Primary Font: Inter (preferred), Roboto, Open Sans.

**Example tailwind.config.ts additions:** TypeScript  
// tailwind.config.ts

import type { Config } from 'tailwindcss';

const config = {

// ... existing configuration (darkMode, content, prefix)

theme: {

container: {

// ... existing container settings

},

extend: {

colors: {

border: 'hsl(var(--border))',

input: 'hsl(var(--input))',

ring: 'hsl(var(--ring))',

background: 'hsl(var(--background))', // e.g., #FFFFFF or a very light gray

foreground: 'hsl(var(--foreground))', // e.g., #2D3748 (Dark Gray for text)

primary: {

DEFAULT: '#1E3A8A', // Deep Blue

foreground: 'hsl(var(--primary-foreground))', // White or light contrast

},

secondary: {

DEFAULT: '#2D6A4F', // Teal/Green

alt: '#2CA58D', // Alternative Teal/Green

foreground: 'hsl(var(--secondary-foreground))', // White or light contrast

},

accent: {

DEFAULT: '#E76F51', // Muted Orange

alt: '#F4A261', // Muted Gold

foreground: 'hsl(var(--accent-foreground))', // White or dark contrast

},

destructive: {

DEFAULT: '#E63946', // Soft Red

foreground: 'hsl(var(--destructive-foreground))',

},

success: {

DEFAULT: '#2CA58D', // Teal Green

foreground: 'hsl(var(--success-foreground))',

},

muted: {

DEFAULT: 'hsl(var(--muted))', // e.g., a light gray

foreground: 'hsl(var(--muted-foreground))', // a medium gray

},

// ... other existing ShadCN/UI colors (card, popover, etc.)

// Add specific brand colors if not covered by semantic names

'brand-deep-blue': '#1E3A8A',

'brand-teal': '#2D6A4F',

'brand-teal-alt': '#2CA58D',

'brand-orange': '#E76F51',

'brand-gold': '#F4A261',

'brand-dark-gray': '#2D3748',

'brand-light-gray': '#A0A0A0',

},

fontFamily: {

sans: ['Inter', 'Roboto', 'Open Sans', 'system-ui', '-apple-system', 'BlinkMacSystemFont', '"Segoe UI"', 'Helvetica', 'Arial', 'sans-serif'],

// If you have a specific serif or mono font, add it here

},

borderRadius: {

// ... existing borderRadius settings

},

keyframes: {

// ... existing keyframes

},

animation: {

// ... existing animation

}

},

},

plugins: [require('tailwindcss-animate')],

} satisfies Config;

export default config;

* + **Best Practice:** Ensure the hsl variables used by ShadCN/UI in client/src/index.css (or globals.css) are updated to reflect these primary brand colors for semantic consistency if needed, or use the direct hex codes in Tailwind.
* **Step 1.2: Global CSS Styles & Font Imports**
  + **File:** client/src/index.css
  + **Action:**
    1. Ensure the primary font (e.g., Inter) is imported (e.g., via Google Fonts @import or from local assets).
    2. Set the default font-family on the body or html element to your chosen primary font.
    3. Set default text color and background color for the body to align with the Design Guide.

**Example client/src/index.css additions (at the top):** CSS  
/\* client/src/index.css \*/

@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700;800&family=Roboto:wght@400;500;700&family=Open+Sans:wght@400;600;700&display=swap');

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer base {

:root {

--background: 0 0% 100%; /\* Light background \*/

--foreground: 222.2 47.4% 11.2%; /\* Dark text, e.g., your brand-dark-gray \*/

/\* ShadCN/UI semantic colors - map to your brand palette \*/

--card: 0 0% 100%;

--card-foreground: 222.2 47.4% 11.2%;

--popover: 0 0% 100%;

--popover-foreground: 222.2 47.4% 11.2%;

--primary: 221 75% 32%; /\* Deep Blue #1E3A8A \*/

--primary-foreground: 210 40% 98%; /\* Light text for on-primary \*/

--secondary: 150 45% 30%; /\* Teal/Green #2D6A4F \*/

--secondary-foreground: 210 40% 98%; /\* Light text for on-secondary \*/

--accent: 24 80% 63%; /\* Muted Orange #E76F51 \*/

--accent-foreground: 210 40% 98%; /\* Light text for on-accent \*/

--muted: 210 40% 96.1%;

--muted-foreground: 215.4 16.3% 46.9%;

--destructive: 0 84.2% 60.2%;

--destructive-foreground: 210 40% 98%;

--border: 214.3 31.8% 91.4%;

--input: 214.3 31.8% 91.4%;

--ring: 221 75% 32%; /\* Deep Blue for focus rings \*/

--radius: 0.5rem;

}

.dark {

/\* Define dark mode colors if needed \*/

}

body {

@apply bg-background text-foreground;

font-family: 'Inter', sans-serif; /\* Or your chosen primary font \*/

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

}

* + **Note:** Review and adjust the HSL values in :root to perfectly match your hex codes from the Design Guide. Many online tools can convert HEX to HSL.

**Phase 2: Authenticated App Navigation (Collapsible Sidebar)**

This is a critical fix. We'll overhaul Sidebar.tsx or create a new component.

* **Step 2.1: Create/Modify Collapsible Sidebar Component - Structure (JSX)**
  + **File:** client/src/components/layout/CollapsibleSidebar.tsx (New or rename/refactor Sidebar.tsx)
  + **Action:** Create the JSX structure for the new sidebar. It should manage an isExpanded state.
  + **Icons:** Use a consistent icon library (e.g., Heroicons, Lucide Icons - which ShadCN often uses). For now, placeholder text or simple SVGs can be used if direct icon components aren't immediately available.

**Example CollapsibleSidebar.tsx structure:** TypeScript  
// client/src/components/layout/CollapsibleSidebar.tsx

import React, { useState } from 'react';

import { NavLink } from 'react-router-dom';

import { Tooltip, TooltipContent, TooltipProvider, TooltipTrigger } from '@/components/ui/tooltip'; // Assuming ShadCN/UI tooltip

// Import icons (e.g., from 'lucide-react' or your chosen library)

import { Home, LayoutList, HardHat, MessageSquareHeart, UserCircle, LogOut, Menu, X } from 'lucide-react'; // Example icons

const navItems = [

{ to: '/app/dashboard', label: 'Dashboard', icon: <Home size={20} /> },

{ to: '/app/tracker', label: 'Tracker', icon: <LayoutList size={20} /> },

{ to: '/app/maker', label: 'Maker', icon: <HardHat size={20} /> },

{ to: '/app/advisor', label: 'Advisor', icon: <MessageSquareHeart size={20} /> },

// --- separator can be handled with margins/borders ---

{ to: '/app/profile', label: 'Profile', icon: <UserCircle size={20} />, bottom: true },

{ to: '/logout', label: 'Logout', icon: <LogOut size={20} />, bottom: true, action: () => { /\* handle logout \*/ console.log('Logout clicked'); } },

];

interface CollapsibleSidebarProps {

isMobileMenuOpen?: boolean;

setIsMobileMenuOpen?: (isOpen: boolean) => void;

}

export const CollapsibleSidebar: React.FC<CollapsibleSidebarProps> = ({ isMobileMenuOpen, setIsMobileMenuOpen }) => {

const [isExpanded, setIsExpanded] = useState(false);

// For mobile/tablet, the Design Guide suggests the sidebar is controlled by a hamburger

// So, isExpanded might be overridden or behave differently based on screen size.

// We'll use a simpler hover for desktop first.

const handleMouseEnter = () => !isMobileMenuOpen && setIsExpanded(true); // Don't expand on hover if mobile menu is explicitly open

const handleMouseLeave = () => !isMobileMenuOpen && setIsExpanded(false);

const sidebarWidth = isExpanded ? 'w-60' : 'w-16'; // Tailwind classes for width

const mobileBaseClasses = "fixed inset-y-0 left-0 z-50 bg-primary text-primary-foreground transform transition-transform duration-300 ease-in-out";

// Desktop/Tablet - Hover-based or fixed mini

// Mobile - Hamburger controlled slide-in (using Sheet component later)

return (

<TooltipProvider delayDuration={0}>

{/\* Desktop Sidebar \*/}

<aside

className={`hidden md:flex flex-col ${sidebarWidth} bg-primary text-primary-foreground transition-all duration-300 ease-in-out space-y-2 py-4`}

onMouseEnter={handleMouseEnter}

onMouseLeave={handleMouseLeave}

>

<div className="px-4 mb-4"> {/\* Logo or App Name \*/}

<NavLink to="/app/dashboard" className="flex items-center">

<img src="/realign-logo-icon.svg" alt="ReAlign Icon" className={`h-8 ${isExpanded ? 'mr-2' : 'mx-auto'}`} />

{isExpanded && <span className="font-bold text-xl">ReAlign</span>}

</NavLink>

</div>

<nav className="flex-grow flex flex-col justify-between">

<ul className="space-y-1 px-2">

{navItems.filter(item => !item.bottom).map((item) => (

<li key={item.label}>

<Tooltip>

<TooltipTrigger asChild>

<NavLink

to={item.to}

onClick={item.action}

className={({ isActive }) =>

`flex items-center p-2 rounded-md hover:bg-primary-foreground/20 ${isActive ? 'bg-primary-foreground/10 font-semibold' : ''}`

}

>

<span className={`flex-shrink-0 ${isExpanded ? 'mr-3' : 'mx-auto'}`}>{item.icon}</span>

{isExpanded && <span className="truncate">{item.label}</span>}

</NavLink>

</TooltipTrigger>

{!isExpanded && <TooltipContent side="right" className="bg-primary text-primary-foreground border-primary-foreground/20">{item.label}</TooltipContent>}

</Tooltip>

</li>

))}

</ul>

<ul className="space-y-1 px-2 mt-auto"> {/\* Bottom items \*/}

{navItems.filter(item => item.bottom).map((item) => (

<li key={item.label}>

<Tooltip>

<TooltipTrigger asChild>

<NavLink

to={item.to}

onClick={item.action}

className={({ isActive }) =>

`flex items-center p-2 rounded-md hover:bg-primary-foreground/20 ${isActive ? 'bg-primary-foreground/10 font-semibold' : ''}`

}

>

<span className={`flex-shrink-0 ${isExpanded ? 'mr-3' : 'mx-auto'}`}>{item.icon}</span>

{isExpanded && <span className="truncate">{item.label}</span>}

</NavLink>

</TooltipTrigger>

{!isExpanded && <TooltipContent side="right" className="bg-primary text-primary-foreground border-primary-foreground/20">{item.label}</TooltipContent>}

</Tooltip>

</li>

))}

</ul>

</nav>

</aside>

</TooltipProvider>

);

};

* + **Place realign-logo-icon.svg in client/public/ folder.** (You'll need to create/source this icon).
* **Step 2.2: Sidebar Responsive Behavior (Mobile/Tablet Hamburger)**
  + **File:** client/src/components/layout/AppShell.tsx and modify CollapsibleSidebar.tsx
  + **Action:** For mobile/tablet, the sidebar will be controlled by a hamburger button in a top bar. We'll use ShadCN's Sheet component for the slide-out effect.
  + **Install lucide-react if not already:** npm install lucide-react

**Modify CollapsibleSidebar.tsx to accept props for mobile state and render content for the Sheet:** TypeScript  
// client/src/components/layout/CollapsibleSidebar.tsx (additions/modifications)

// ... (imports and navItems as before) ...

// Add Sheet components from ShadCN

import { Sheet, SheetContent, SheetTrigger, SheetClose } from "@/components/ui/sheet";

interface NavItemProps {

to: string;

label: string;

icon: React.ReactNode;

action?: () => void;

isExpanded?: boolean; // Used for mobile sheet where it's always expanded

onClick?: () => void; // For SheetClose

}

const NavItem: React.FC<NavItemProps> = ({ to, label, icon, action, isExpanded = true, onClick }) => (

<li>

<NavLink

to={to}

onClick={() => {

if (action) action();

if (onClick) onClick(); // Close sheet on click

}}

className={({ isActive }) =>

`flex items-center p-3 rounded-md hover:bg-primary-foreground/20 ${isActive ? 'bg-primary-foreground/10 font-semibold' : ''}`

}

>

<span className="mr-3 flex-shrink-0">{icon}</span>

{isExpanded && <span className="truncate">{label}</span>}

</NavLink>

</li>

);

// SidebarContent for reuse in desktop and mobile sheet

const SidebarContent: React.FC<{isSheet?: boolean; closeSheet?: () => void;}> = ({ isSheet, closeSheet }) => (

<div className={`flex flex-col h-full ${isSheet ? 'bg-primary text-primary-foreground p-4' : ''}`}>

{!isSheet && ( /\* Desktop only logo area \*/

<div className="px-4 mb-4">

<NavLink to="/app/dashboard" className="flex items-center">

{/\* Placeholder for ReAlign logo \*/}

<img src="/realign-logo-icon.svg" alt="ReAlign Icon" className={`h-8 ${isExpanded ? 'mr-2' : 'mx-auto'}`} />

{isExpanded && <span className="font-bold text-xl">ReAlign</span>}

</NavLink>

</div>

)}

{isSheet && ( /\* Mobile only logo area \*/

<div className="mb-6 text-center">

<NavLink to="/app/dashboard" onClick={closeSheet} className="inline-flex items-center">

<img src="/realign-logo-icon.svg" alt="ReAlign Icon" className="h-10 mr-2" />

<span className="font-bold text-2xl text-primary-foreground">ReAlign</span>

</NavLink>

</div>

)}

<nav className="flex-grow flex flex-col justify-between">

<ul className="space-y-1">

{navItems.filter(item => !item.bottom).map((item) => (

isSheet ? <SheetClose asChild key={item.label}><NavItem {...item} isExpanded={true} /></SheetClose>

: <NavItemWithTooltip key={item.label} item={item} isExpanded={isExpandedStateForDesktop} /> // isExpandedStateForDesktop needs to be passed

))}

</ul>

<ul className="space-y-1 mt-auto pt-4 border-t border-primary-foreground/10"> {/\* Bottom items \*/}

{navItems.filter(item => item.bottom).map((item) => (

isSheet ? <SheetClose asChild key={item.label}><NavItem {...item} isExpanded={true} /></SheetClose>

: <NavItemWithTooltip key={item.label} item={item} isExpanded={isExpandedStateForDesktop} />

))}

</ul>

</nav>

</div>

);

// Helper for desktop with tooltip logic

let isExpandedStateForDesktop = false; // This needs to be managed by the desktop sidebar's state

const NavItemWithTooltip: React.FC<{item: typeof navItems[0], isExpanded: boolean}> = ({ item, isExpanded }) => (

<li>

<Tooltip>

<TooltipTrigger asChild>

<NavLink

to={item.to}

onClick={item.action}

className={({ isActive }) =>

`flex items-center p-2 rounded-md hover:bg-primary-foreground/20 ${isActive ? 'bg-primary-foreground/10 font-semibold' : ''}`

}

>

<span className={`flex-shrink-0 ${isExpanded ? 'mr-3' : 'mx-auto'}`}>{item.icon}</span>

{isExpanded && <span className="truncate">{item.label}</span>}

</NavLink>

</TooltipTrigger>

{!isExpanded && <TooltipContent side="right" className="bg-primary text-primary-foreground border-primary-foreground/20">{item.label}</TooltipContent>}

</Tooltip>

</li>

);

export const CollapsibleSidebar: React.FC = () => {

const [isDesktopExpanded, setIsDesktopExpanded] = useState(false);

const [isMobileSheetOpen, setIsMobileSheetOpen] = useState(false);

isExpandedStateForDesktop = isDesktopExpanded; // Update the shared state for NavItemWithTooltip

const handleDesktopMouseEnter = () => setIsDesktopExpanded(true);

const handleDesktopMouseLeave = () => setIsDesktopExpanded(false);

const desktopSidebarWidth = isDesktopExpanded ? 'w-60' : 'w-16';

return (

<TooltipProvider delayDuration={0}>

{/\* Desktop Sidebar \*/}

<aside

className={`hidden md:flex flex-col ${desktopSidebarWidth} bg-primary text-primary-foreground transition-all duration-300 ease-in-out fixed inset-y-0 left-0 z-40`}

onMouseEnter={handleDesktopMouseEnter}

onMouseLeave={handleDesktopMouseLeave}

>

<SidebarContent />

</aside>

{/\* Mobile/Tablet Hamburger + Sheet \*/}

<div className="md:hidden fixed top-0 left-0 right-0 z-50 bg-background/80 backdrop-blur-sm h-16 flex items-center px-4 border-b"> {/\* Mobile Top Bar \*/}

<Sheet open={isMobileSheetOpen} onOpenChange={setIsMobileSheetOpen}>

<SheetTrigger asChild>

<button className="p-2 rounded-md text-foreground hover:bg-muted">

<Menu size={24} />

<span className="sr-only">Open Menu</span>

</button>

</SheetTrigger>

<SheetContent side="left" className="w-72 p-0 bg-primary text-primary-foreground border-r-0">

<SidebarContent isSheet={true} closeSheet={() => setIsMobileSheetOpen(false)} />

</SheetContent>

</Sheet>

<div className="ml-4 font-bold text-lg text-primary">ReAlign</div> {/\* App name in mobile top bar \*/}

</div>

</TooltipProvider>

);

};

* + **Styling:** Ensure the SheetContent is styled with the primary blue background and light text, as per the Design Guide.
  + **Note:** The isExpandedStateForDesktop variable is a bit of a hack to pass the state to NavItemWithTooltip. A more robust solution might involve context or drilling the prop directly if SidebarContent is further refactored. For AI implementation, this direct approach might be simpler to start with.
* **Step 2.3: Integrate Sidebar and Top Bar into AppShell.tsx**
  + **File:** client/src/components/layout/AppShell.tsx
  + **Action:** Replace the old Sidebar usage with CollapsibleSidebar. Adjust the main content area's padding/margin to account for the sidebar width.

**Example AppShell.tsx:** TypeScript  
// client/src/components/layout/AppShell.tsx

import React from 'react';

import { Outlet } from 'react-router-dom';

import { CollapsibleSidebar } from './CollapsibleSidebar';

import { Toaster } from "@/components/ui/toaster"; // Assuming you have a toaster

export function AppShell() {

return (

<div className="flex h-screen bg-background">

<CollapsibleSidebar />

{/\* Main content area \*/}

{/\* Desktop: needs left padding equal to collapsed sidebar width \*/}

{/\* Mobile: needs top padding equal to mobile header height \*/}

<main className="flex-1 overflow-y-auto pt-16 md:pt-4 md:pl-16"> {/\* Adjust md:pl-16 to match collapsed sidebar width \*/}

<div className="p-4 md:p-6 lg:p-8"> {/\* Inner padding for content \*/}

<Outlet />

</div>

</main>

<Toaster />

</div>

);

}

* + **Crucial:** The md:pl-16 on the main element needs to exactly match the collapsed width of the desktop sidebar (e.g., w-16 is 4rem = 64px. If using 1rem=16px, then pl-16 is correct. Adjust as needed). The pt-16 is for the mobile top bar.

**Phase 3: Public Website UI & Navigation**

This involves creating a separate layout and header/footer for public-facing pages.

* **Step 3.1: Create PublicLayout.tsx**
  + **File:** client/src/components/layout/PublicLayout.tsx (New)
  + **Action:** This component will wrap public pages and include PublicHeader and PublicFooter.

**Example PublicLayout.tsx:** TypeScript  
// client/src/components/layout/PublicLayout.tsx

import React from 'react';

import { Outlet } from 'react-router-dom';

import { PublicHeader } from './PublicHeader';

import { PublicFooter } from './PublicFooter';

import { Toaster } from "@/components/ui/toaster";

export function PublicLayout() {

return (

<div className="flex flex-col min-h-screen bg-background">

<PublicHeader />

<main className="flex-1">

<Outlet />

</main>

<PublicFooter />

<Toaster />

</div>

);

}

* **Step 3.2: Create PublicHeader.tsx Component**
  + **File:** client/src/components/layout/PublicHeader.tsx (New)
  + **Action:** Implement the header with logo, navigation links (Home, About, How It Works, Solutions, Contact, Login, Register), and responsive hamburger menu.
  + **Styling:** Use Tailwind CSS. Colors and fonts should align with the Design Guide.

**Example PublicHeader.tsx:** TypeScript  
// client/src/components/layout/PublicHeader.tsx

import React, { useState } from 'react';

import { NavLink }\_from 'react-router-dom';

import { Button } from '@/components/ui/button';

import { Sheet, SheetContent, SheetTrigger, SheetClose } from '@/components/ui/sheet';

import { Menu, X } from 'lucide-react'; // Icons

const publicNavLinks = [

{ to: '/', label: 'Home' },

{ to: '/about', label: 'About Us' },

{ to: '/how-it-works', label: 'How It Works' },

{ to: '/solutions', label: 'Solutions' },

{ to: '/contact', label: 'Contact Us' },

];

export function PublicHeader() {

const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

return (

<header className="sticky top-0 z-50 w-full border-b bg-background/95 backdrop-blur supports-[backdrop-filter]:bg-background/60">

<div className="container flex h-16 items-center justify-between">

{/\* Logo \*/}

<NavLink to="/" className="flex items-center space-x-2">

<img src="/realign-logo-full.svg" alt="ReAlign Logo" className="h-8" /> {/\* Create realign-logo-full.svg \*/}

</NavLink>

{/\* Desktop Navigation \*/}

<nav className="hidden md:flex items-center space-x-6">

{publicNavLinks.map(link => (

<NavLink

key={link.label}

to={link.to}

className={({isActive}) =>

`text-sm font-medium transition-colors hover:text-primary ${isActive ? 'text-primary' : 'text-muted-foreground'}`

}

>

{link.label}

</NavLink>

))}

</nav>

<div className="hidden md:flex items-center space-x-2">

<Button variant="ghost" asChild>

<NavLink to="/login">Login</NavLink>

</Button>

<Button asChild>

<NavLink to="/register">Register</NavLink>

</Button>

</div>

{/\* Mobile Navigation Trigger \*/}

<div className="md:hidden">

<Sheet open={isMobileMenuOpen} onOpenChange={setIsMobileMenuOpen}>

<SheetTrigger asChild>

<Button variant="ghost" size="icon">

<Menu className="h-6 w-6" />

<span className="sr-only">Toggle menu</span>

</Button>

</SheetTrigger>

<SheetContent side="left" className="w-full max-w-xs bg-background p-6">

<div className="flex flex-col space-y-5">

<div className="flex justify-between items-center mb-4">

<NavLink to="/" onClick={() => setIsMobileMenuOpen(false)} className="flex items-center space-x-2">

<img src="/realign-logo-full.svg" alt="ReAlign Logo" className="h-7" />

</NavLink>

<SheetClose asChild>

<Button variant="ghost" size="icon"><X className="h-6 w-6" /></Button>

</SheetClose>

</div>

{publicNavLinks.map(link => (

<SheetClose asChild key={link.label}>

<NavLink

to={link.to}

className={({isActive}) =>

`block py-2 text-lg font-medium transition-colors hover:text-primary ${isActive ? 'text-primary' : 'text-foreground'}`

}

>

{link.label}

</NavLink>

</SheetClose>

))}

<hr />

<SheetClose asChild>

<Button variant="outline" className="w-full" asChild><NavLink to="/login">Login</NavLink></Button>

</SheetClose>

<SheetClose asChild>

<Button className="w-full" asChild><NavLink to="/register">Register</NavLink></Button>

</SheetClose>

</div>

</SheetContent>

</Sheet>

</div>

</div>

</header>

);

}

* + **Place realign-logo-full.svg in client/public/ folder.** (You'll need to create/source this).
* **Step 3.3: Create PublicFooter.tsx Component**
  + **File:** client/src/components/layout/PublicFooter.tsx (New)
  + **Action:** Implement a simple footer with copyright and links to Privacy Policy, Terms of Service.

**Example PublicFooter.tsx:** TypeScript  
// client/src/components/layout/PublicFooter.tsx

import React from 'react';

import { NavLink }\_from 'react-router-dom';

export function PublicFooter() {

const currentYear = new Date().getFullYear();

return (

<footer className="border-t bg-background">

<div className="container py-8 text-center md:text-left">

<div className="md:flex md:items-center md:justify-between">

<p className="text-sm text-muted-foreground">

&copy; {currentYear} ReAlign. All rights reserved.

</p>

<nav className="mt-4 md:mt-0 flex justify-center space-x-4">

<NavLink to="/privacy-policy" className="text-sm text-muted-foreground hover:text-primary">

Privacy Policy

</NavLink>

<NavLink to="/terms-of-service" className="text-sm text-muted-foreground hover:text-primary">

Terms of Service

</NavLink>

</nav>

</div>

</div>

</footer>

);

}

**Phase 4: Routing & Basic Page Placeholders**

* **Step 4.1: Update Routing in client/src/App.tsx**
  + **File:** client/src/App.tsx
  + **Action:** Define routes for public pages using PublicLayout and authenticated app pages using AppShell.

**Example App.tsx structure:** TypeScript  
// client/src/App.tsx

import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';

import { AuthProvider, useAuth } from './context/AuthContext'; // Assuming AuthContext exists

import { AppShell } from './components/layout/AppShell';

import { PublicLayout } from './components/layout/PublicLayout';

// --- Public Page Placeholders ---

const HomePage = () => <div className="container py-10"><h1>Homepage</h1><p>Public content...</p></div>;

const AboutPage = () => <div className="container py-10"><h1>About Us</h1></div>;

const HowItWorksPage = () => <div className="container py-10"><h1>How It Works</h1></div>;

const SolutionsPage = () => <div className="container py-10"><h1>Solutions</h1></div>;

const ContactPage = () => <div className="container py-10"><h1>Contact Us</h1></div>;

const PrivacyPolicyPage = () => <div className="container py-10"><h1>Privacy Policy</h1></div>;

const TermsPage = () => <div className="container py-10"><h1>Terms of Service</h1></div>;

// --- Auth Page Placeholders (can reuse existing ones if they fit) ---

import LoginPage from './pages/Login'; // Existing: vibecode1/realignmvp/ReAlignMvp-bb0fbf87dcbcc7decbbb02a2ffeabc726b5257dd/client/src/pages/Login.tsx

import RegisterPage from './pages/RegisterNegotiator'; // Use as general register: vibecode1/realignmvp/ReAlignMvp-bb0fbf87dcbcc7decbbb02a2ffeabc726b5257dd/client/src/pages/RegisterNegotiator.tsx

// ... import other auth pages like ResetPassword, UpdatePassword

// --- App Page Placeholders (can reuse existing ones or create new) ---

import DashboardPage from './pages/Dashboard'; // Existing: vibecode1/realignmvp/ReAlignMvp-bb0fbf87dcbcc7decbbb02a2ffeabc726b5257dd/client/src/pages/Dashboard.tsx

const TrackerPage = () => <div><h2>Tracker Module</h2></div>; // Placeholder

const MakerPage = () => <div><h2>Maker Module</h2></div>; // Placeholder

const AdvisorPage = () => <div><h2>Advisor Module</h2></div>; // Placeholder

const ProfilePage = () => <div><h2>User Profile</h2></div>; // Placeholder

import NotFoundPage from './pages/not-found'; // Existing: vibecode1/realignmvp/ReAlignMvp-bb0fbf87dcbcc7decbbb02a2ffeabc726b5257dd/client/src/pages/not-found.tsx

// ProtectedRoute HOC

const ProtectedRoute = ({ children }: { children: JSX.Element }) => {

const { user, isLoading } = useAuth(); // Assuming useAuth provides user and loading state

if (isLoading) return <div>Loading authentication...</div>; // Or a spinner component

return user ? children : <Navigate to="/login" replace />;

};

// UnProtectedRoute HOC (for /login, /register - redirect if already logged in)

const UnProtectedRoute = ({ children }: { children: JSX.Element }) => {

const { user, isLoading } = useAuth();

if (isLoading) return <div>Loading authentication...</div>;

return !user ? children : <Navigate to="/app/dashboard" replace />;

};

function App() {

return (

<BrowserRouter>

<AuthProvider> {/\* Wrap with AuthProvider \*/}

<Routes>

{/\* Public Pages \*/}

<Route path="/" element={<PublicLayout />}>

<Route index element={<HomePage />} />

<Route path="about" element={<AboutPage />} />

<Route path="how-it-works" element={<HowItWorksPage />} />

<Route path="solutions" element={<SolutionsPage />} />

<Route path="contact" element={<ContactPage />} />

<Route path="privacy-policy" element={<PrivacyPolicyPage />} />

<Route path="terms-of-service" element={<TermsPage />} />

<Route path="login" element={<UnProtectedRoute><LoginPage /></UnProtectedRoute>} />

<Route path="register" element={<UnProtectedRoute><RegisterPage /></UnProtectedRoute>} />

{/\* Add other public auth routes like forgot-password, reset-password \*/}

</Route>

{/\* Authenticated App Pages \*/}

<Route path="/app" element={<ProtectedRoute><AppShell /></ProtectedRoute>}>

<Route path="dashboard" element={<DashboardPage />} />

<Route path="tracker" element={<TrackerPage />} />

<Route path="maker" element={<MakerPage />} />

<Route path="advisor" element={<AdvisorPage />} />

<Route path="profile" element={<ProfilePage />} />

{/\* Add other app-specific routes here \*/}

</Route>

{/\* Logout Route (can be a simple component that calls logout and redirects) \*/}

<Route path="/logout" element={<div>Logging out...</div> /\* Implement actual logout logic \*/} />

{/\* Catch-all Not Found Page \*/}

<Route path="\*" element={<NotFoundPage />} />

</Routes>

</AuthProvider>

</BrowserRouter>

);

}

export default App;

* + **Note:** The ProtectedRoute and UnProtectedRoute are essential. AuthProvider and useAuth from client/src/context/AuthContext.tsx will need to correctly manage authentication state. The Logout route should trigger the logout function from AuthContext.
* **Step 4.2: Create Basic Page Components (Placeholders if not existing)**
  + **Files:** Create new files in client/src/pages/ for any public pages that don't exist (e.g., HomePage.tsx, AboutPage.tsx).
  + **Action:** For now, these can be simple functional components returning a div with an h1 tag.

**Example HomePage.tsx:** TypeScript  
// client/src/pages/HomePage.tsx

import React from 'react';

import { Button } from "@/components/ui/button";

import { Link } from "react-router-dom";

export default function HomePage() {

return (

<div className="container mx-auto px-4 py-16 text-center">

<h1 className="text-4xl md:text-5xl font-bold text-primary mb-6">

Take Control of Your Mortgage Situation

</h1>

<p className="text-lg md:text-xl text-muted-foreground mb-10 max-w-2xl mx-auto">

ReAlign empowers you to navigate loss mitigation with clarity and confidence. Automate paperwork, get expert guidance, and stay organized.

</p>

<div className="space-x-4">

<Button size="lg" asChild>

<Link to="/register">Get Started Free</Link>

</Button>

<Button size="lg" variant="outline" asChild>

<Link to="/how-it-works">Learn More</Link>

</Button>

</div>

{/\* TODO: Add sections for Features, How it works summary, Testimonials as per Design Guide \*/}

</div>

);

}

**Phase 5: Apply UI Elements and Best Practices Continuously**

* **Step 5.1: Iteratively Implement UI Elements**
  + **Action:** As pages and components are built or refined beyond placeholders:
    - Refer to the "Design Guide" Section 5: "UI Components & Interaction Patterns" and the original "Design Brief - Gregory John inspo.docx".
    - Use the ShadCN/UI components from client/src/components/ui/ (e.g., button.tsx, card.tsx, input.tsx, label.tsx, form.tsx) as the foundation for forms, buttons, cards, etc.
    - Style these components using Tailwind utility classes to match the brand identity (colors, typography, spacing).
    - For example, primary buttons should use the bg-primary and text-primary-foreground classes.
* **Step 5.2: General Best Practices for AI Implementation:**
  + **Component Reusability:** Encourage the creation of small, reusable components for elements like form fields with labels, specific card layouts, etc.
  + **Accessibility (A11y):**
    - Use semantic HTML elements (<nav>, <main>, <aside>, <header>, <footer>, <button>).
    - Ensure all interactive elements are keyboard accessible and have clear focus indicators (Tailwind's default focus rings with ring-primary should work well).
    - Add aria-labels where necessary, especially for icon-only buttons.
    - Ensure sufficient color contrast (refer to Design Guide).
  + **Responsiveness:** Continuously test the UI on different screen sizes (use browser developer tools). Ensure layouts adapt gracefully, content is readable, and interactive elements are easily tappable/clickable.
  + **Code Clarity:** Generate well-formatted, commented code where necessary, especially for complex logic or state management.
  + **State Management:** Use React's useState and useContext for local and shared state as appropriate. For more complex global state, the existing setup (likely React Query + AuthContext) should be leveraged.
  + **Error Handling & Loading States:** Implement basic loading indicators (e.g., skeletons or spinners from ShadCN/UI) and user-friendly error messages for data fetching or form submissions.
  + **Iterative Refinement:** After this initial UI scaffolding, further iterations will be needed to build out the detailed functionality of each module (Tracker, Maker, Advisor).