Here are the files and their contents:

**.env:**

**.vscode/settings.json:**

{

"IDX.aI.enableInlineCompletion": true,

"IDX.aI.enableCodebaseIndexing": true

}

**README.md:**

# Firebase Studio

This is a NextJS starter in Firebase Studio.

To get started, take a look at src/app/page.tsx.

**components.json:**

{

"$schema": "https://ui.shadcn.com/schema.json",

"style": "default",

"rsc": true,

"tsx": true,

"tailwind": {

"config": "tailwind.config.ts",

"css": "src/app/globals.css",

"baseColor": "neutral",

"cssVariables": true,

"prefix": ""

},

"aliases": {

"components": "@/components",

"utils": "@/lib/utils",

"ui": "@/components/ui",

"lib": "@/lib",

"hooks": "@/hooks"

},

"iconLibrary": "lucide"

}

**next.config.ts:**

import type {NextConfig} from 'next';

const nextConfig: NextConfig = {

/\* config options here \*/

typescript: {

ignoreBuildErrors: true,

},

eslint: {

ignoreDuringBuilds: true,

},

images: {

remotePatterns: [

{

protocol: 'https',

hostname: 'placehold.co', // For placeholder images

port: '',

pathname: '/\*\*',

},

// If AI might return images from various sources, a more permissive rule might be needed.

// However, current flows specify data URIs, so this might not be strictly necessary yet.

// Example for allowing any https source (use with caution):

// {

// protocol: 'https',

// hostname: '\*\*',

// },

],

// For handling data URIs with next/image

dangerouslyAllowSVG: true,

contentSecurityPolicy: "default-src 'self'; script-src 'none'; sandbox;", // if using unoptimized={true} for data URIs, this helps

},

};

export default nextConfig;

**package.json:**

{

"name": "nextn",

"version": "0.1.0",

"private": true,

"scripts": {

"dev": "next dev --turbopack -p 9002",

"genkit:dev": "genkit start -- tsx src/ai/dev.ts",

"genkit:watch": "genkit start -- tsx --watch src/ai/dev.ts",

"build": "next build",

"start": "next start",

"lint": "next lint",

"typecheck": "tsc --noEmit"

},

"dependencies": {

"@genkit-ai/googleai": "^1.8.0",

"@genkit-ai/next": "^1.8.0",

"@hookform/resolvers": "^4.1.3",

"@radix-ui/react-accordion": "^1.2.3",

"@radix-ui/react-alert-dialog": "^1.1.6",

"@radix-ui/react-avatar": "^1.1.3",

"@radix-ui/react-checkbox": "^1.1.4",

"@radix-ui/react-dialog": "^1.1.6",

"@radix-ui/react-dropdown-menu": "^2.1.6",

"@radix-ui/react-label": "^2.1.2",

"@radix-ui/react-menubar": "^1.1.6",

"@radix-ui/react-popover": "^1.1.6",

"@radix-ui/react-progress": "^1.1.2",

"@radix-ui/react-radio-group": "^1.2.3",

"@radix-ui/react-scroll-area": "^1.2.3",

"@radix-ui/react-select": "^2.1.6",

"@radix-ui/react-separator": "^1.1.2",

"@radix-ui/react-slider": "^1.2.3",

"@radix-ui/react-slot": "^1.1.2",

"@radix-ui/react-switch": "^1.1.3",

"@radix-ui/react-tabs": "^1.1.3",

"@radix-ui/react-toast": "^1.2.6",

"@radix-ui/react-tooltip": "^1.1.8",

"@tanstack-query-firebase/react": "^1.0.5",

"@tanstack/react-query": "^5.66.0",

"class-variance-authority": "^0.7.1",

"clsx": "^2.1.1",

"date-fns": "^3.6.0",

"dotenv": "^16.5.0",

"firebase": "^11.7.0",

"genkit": "^1.8.0",

"lucide-react": "^0.475.0",

"next": "15.2.3",

"patch-package": "^8.0.0",

"react": "^18.3.1",

"react-day-picker": "^8.10.1",

"react-dom": "^18.3.1",

"react-hook-form": "^7.54.2",

"recharts": "^2.15.1",

"tailwind-merge": "^3.0.1",

"tailwindcss-animate": "^1.0.7",

"zod": "^3.24.2"

},

"devDependencies": {

"@types/node": "^20",

"@types/react": "^18",

"@types/react-dom": "^18",

"genkit-cli": "^1.8.0",

"postcss": "^8",

"tailwindcss": "^3.4.1",

"typescript": "^5"

}

}

**src/ai/dev.ts:**

import { config } from 'dotenv';

config();

import '@/ai/flows/text-to-sign-translation.ts';

import '@/ai/flows/sign-to-text-translation.ts';

**src/ai/flows/sign-to-text-translation.ts:**

'use server';

/\*\*

\* @fileOverview A sign language to text translation AI agent.

\*

\* - signToTextTranslation - A function that handles the sign language to text translation process.

\* - SignToTextTranslationInput - The input type for the signToTextTranslation function.

\* - SignToTextTranslationOutput - The return type for the signToTextTranslation function.

\*/

import {ai} from '@/ai/genkit';

import {z} from 'genkit';

const SignToTextTranslationInputSchema = z.object({

signDataUri: z

.string()

.describe(

"An image or animation of sign language, as a data URI that must include a MIME type and use Base64 encoding. Expected format: 'data:<mimetype>;base64,<encoded\_data>'."

),

});

export type SignToTextTranslationInput = z.infer<typeof SignToTextTranslationInputSchema>;

const SignToTextTranslationOutputSchema = z.object({

translation: z.string().describe('The text translation of the sign language input.'),

});

export type SignToTextTranslationOutput = z.infer<typeof SignToTextTranslationOutputSchema>;

export async function signToTextTranslation(input: SignToTextTranslationInput): Promise<SignToTextTranslationOutput> {

return signToTextTranslationFlow(input);

}

const prompt = ai.definePrompt({

name: 'signToTextTranslationPrompt',

input: {schema: SignToTextTranslationInputSchema},

output: {schema: SignToTextTranslationOutputSchema},

prompt: `You are an expert in sign language translation. You will receive an image or animation of sign language, and you will translate it to text. Return ONLY the translation, do not include any other commentary.

Sign Language: {{media url=signDataUri}}`,

});

const signToTextTranslationFlow = ai.defineFlow(

{

name: 'signToTextTranslationFlow',

inputSchema: SignToTextTranslationInputSchema,

outputSchema: SignToTextTranslationOutputSchema,

},

async input => {

const {output} = await prompt(input);

return output!;

}

);

**src/ai/flows/text-to-sign-translation.ts:**

'use server';

/\*\*

\* @fileOverview A text-to-sign language translation AI agent.

\* It breaks down input text into a sequence of sign language steps,

\* generating an image and textual guide for each step.

\*

\* - translateTextToSign - A function that handles the text-to-sign translation process.

\* - TranslateTextToSignInput - The input type for the translateTextToSign function.

\* - TranslateTextToSignOutput - The return type for the translateTextToSign function.

\* - SignStep - Represents a single step in the sign language sequence.

\* - HandUsage - Defines the possible hand usage types for a sign.

\*/

import {ai} from '@/ai/genkit';

import {z} from 'genkit';

import type { HandUsage, SignStep } from '@/types';

// Schema for the input text

const TranslateTextToSignInputSchema = z.object({

text: z.string().describe('The text to be translated into sign language.'),

});

export type TranslateTextToSignInput = z.infer<typeof TranslateTextToSignInputSchema>;

// Schema for a single step in the sign sequence (final output structure for each step)

const SignStepSchema = z.object({

sequenceNumber: z.number().describe('The order of this sign in the sequence (e.g., 1, 2, 3).'),

word: z.string().optional().describe('The original word or phrase segment from the input text that this sign represents (e.g., "Hello", "how are you").'),

imageDataUri: z.string().describe("A data URI for the visual representation of this specific sign step. Format: 'data:<mimetype>;base64,<encoded\_data>'."),

handUsage: z.enum(['left\_hand', 'right\_hand', 'both\_hands'])

.describe('Specifies hand usage for this sign: \'left\_hand\', \'right\_hand\', or \'both\_hands\'.'),

textualGuide: z.string().describe('A short textual description of how to perform this sign step, including its sequence number (e.g., "Step 1: Raise right hand, palm out, near shoulder (Hey)").')

});

// Schema for the final output of the translation flow

const TranslateTextToSignOutputSchema = z.object({

signSequence: z.array(SignStepSchema).describe('An array of sign steps representing the translated text, in order.'),

});

export type TranslateTextToSignOutput = z.infer<typeof TranslateTextToSignOutputSchema>;

// Schema for the intermediate step breakdown by the LLM (before image generation for each step)

const SignStepDecompositionSchema = z.object({

originalWordSegment: z.string().describe('The specific word or short phrase segment from the input text that this sign step directly translates (e.g., "Hello", "how are you", "thank you"). This will be used for the "word" field in the final output.'),

signDescriptionForImagePrompt: z.string().describe('A detailed prompt to generate an image for this specific sign segment. This prompt must incorporate all stylistic requirements: sketch art style image focusing only on 3D hands, high-contrast, simple uncluttered background, semi-realistic 3D sketched hands (friendly/neutral avatar if implied), soft warm color palette (blues, greens, soft purples), and strict focus on hand gesture. It should clearly describe handshape, orientation, location, and movement for THIS sign segment.'),

handUsage: z.enum(['left\_hand', 'right\_hand', 'both\_hands'])

.describe('Specify if this sign uses \'left\_hand\', \'right\_hand\', or \'both\_hands\'.'),

textualGuideFragment: z.string().describe('The textual explanation for this specific sign segment of how to perform the sign, suitable for display under the image (e.g., "Raise right hand, palm out, near shoulder"). This will be part of a numbered list.')

});

const TextDecompositionOutputInternalSchema = z.object({

decomposedSteps: z.array(SignStepDecompositionSchema).describe('An array of decomposed sign steps, each with an original word segment, an image generation prompt, hand usage, and textual guide fragment.')

});

// This prompt breaks text into signable steps with descriptions for image generation and textual guides

const textToSignStepsPrompt = ai.definePrompt({

name: 'textToSignStepsPrompt',

input: {schema: TranslateTextToSignInputSchema},

output: {schema: TextDecompositionOutputInternalSchema},

prompt: `You are an expert sign language instructor and linguist. Your task is to break down the input text into a sequence of individual, signable gestures or components following sign language grammar rules where appropriate for the language (assume English/ASL-like structure if not specified). For each component, you must provide:

1. 'originalWordSegment': The specific word or short phrase segment from the input text that this sign step directly translates (e.g., "Hello", "how are you", "thank you"). This will be used for the "word" field in the final output.

2. 'signDescriptionForImagePrompt': This is a specific prompt that will be given to an image generation AI to create a visual for THIS sign step. This prompt \*must\* include all the following stylistic details for the image:

\* Style: Sketch art style image focusing ONLY on 3D hands.

\* Clarity: High-contrast visuals (clear, contrasting backgrounds and foregrounds to make hand shapes easy to see).

\* Background: Simple, uncluttered background to ensure focus is on the hand gesture.

\* Hand Appearance: Semi-realistic 3D sketched hands. If a neutral, friendly avatar is implied by the hands, ensure it's inclusive.

\* Color Palette: If colors are used (e.g. for sketch lines or minimal accents), use soft, warm colors like blues, greens, or soft purples that are calm and inviting. Ensure hand shapes are very easy to see.

\* Focus: Strictly on the hands and their gesture. No other characters or distracting elements.

\* Specificity: The prompt should clearly describe the handshape, orientation, location, and movement for THIS specific sign segment from the input text.

3. 'handUsage': Specify if this sign uses 'left\_hand', 'right\_hand', or 'both\_hands'.

4. 'textualGuideFragment': A concise textual explanation of \*how to perform\* this specific sign segment, suitable for display under the image (e.g., "Raise right hand, palm out, near shoulder"). This will be part of a numbered list.

Input text: "{{{text}}}"

Provide the output as a JSON object adhering to the 'TextDecompositionOutputInternalSchema'. Ensure each sign segment from the input text is represented.

Consider the natural flow and grammar of sign language when breaking down the text. Some words might combine into a single sign, or word order might differ. Provide the most natural and understandable sequence of signs.

Each 'signDescriptionForImagePrompt' must be self-contained and detailed enough for an image model to generate only that step's visual.

Example for "Hello":

{

"decomposedSteps": [

{

"originalWordSegment": "Hello",

"signDescriptionForImagePrompt": "Generate a sketch art style image focusing only on 3D hands. Show a single right hand, open palm facing forward, fingers together and extended upwards, moving slightly away from the forehead in a saluting motion. Style: high-contrast, simple background, semi-realistic 3D sketched hand, soft color palette.",

"handUsage": "right\_hand",

"textualGuideFragment": "Start with your right hand near your forehead, palm open, then move it outward and slightly upward in a small salute motion."

}

]

}

`,

});

export async function translateTextToSign(input: TranslateTextToSignInput): Promise<TranslateTextToSignOutput> {

return translateTextToSignFlow(input);

}

const translateTextToSignFlow = ai.defineFlow(

{

name: 'translateTextToSignFlow',

inputSchema: TranslateTextToSignInputSchema,

outputSchema: TranslateTextToSignOutputSchema,

},

async (input) => {

// 1. Decompose text into sign steps with descriptions

const { output: decompositionResult } = await textToSignStepsPrompt(input);

if (!decompositionResult || !decompositionResult.decomposedSteps || decompositionResult.decomposedSteps.length === 0) {

if (input.text.trim().length > 0) {

console.error("Decomposition failed or returned no steps for input:", input.text, "Result:", decompositionResult);

throw new Error('Failed to decompose text into sign steps. The AI could not break down the provided text.');

}

return { signSequence: [] };

}

const finalSignSequence: SignStep[] = [];

let sequenceNum = 1;

for (const stepDetails of decompositionResult.decomposedSteps) {

// 2. Generate image for each step

const imagePrompt = stepDetails.signDescriptionForImagePrompt;

const { media } = await ai.generate({

model: 'googleai/gemini-2.0-flash-exp',

prompt: [{ text: imagePrompt }],

config: {

responseModalities: ['TEXT', 'IMAGE'],

safetySettings: [

{ category: 'HARM\_CATEGORY\_HATE\_SPEECH', threshold: 'BLOCK\_MEDIUM\_AND\_ABOVE' },

{ category: 'HARM\_CATEGORY\_DANGEROUS\_CONTENT', threshold: 'BLOCK\_MEDIUM\_AND\_ABOVE' },

{ category: 'HARM\_CATEGORY\_HARASSMENT', threshold: 'BLOCK\_MEDIUM\_AND\_ABOVE' },

{ category: 'HARM\_CATEGORY\_SEXUALLY\_EXPLICIT', threshold: 'BLOCK\_MEDIUM\_AND\_ABOVE' },

],

},

});

if (!media || !media.url) {

console.error(`Image generation failed for step: ${stepDetails.textualGuideFragment} with prompt: ${imagePrompt}`);

throw new Error(`Image generation failed for sign corresponding to "${stepDetails.originalWordSegment}".`);

}

finalSignSequence.push({

sequenceNumber: sequenceNum,

word: stepDetails.originalWordSegment,

imageDataUri: media.url,

handUsage: stepDetails.handUsage as HandUsage,

textualGuide: `Step ${sequenceNum}: ${stepDetails.textualGuideFragment}`,

});

sequenceNum++;

}

if (finalSignSequence.length === 0 && decompositionResult.decomposedSteps.length > 0) {

throw new Error('All image generations failed for the decomposed steps, though text decomposition was successful.');

}

return {

signSequence: finalSignSequence,

};

}

);

**src/ai/genkit.ts:**

import {genkit} from 'genkit';

import {googleAI} from '@genkit-ai/googleai';

export const ai = genkit({

plugins: [googleAI()],

model: 'googleai/gemini-2.0-flash',

});

**src/app/globals.css:**

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer base {

:root {

--background: 0 0% 98%; /\* Light Gray for clean background \*/

--foreground: 207 100% 14%; /\* Deep Blue for primary text \*/

--card: 0 0% 100%; /\* White for card elements \*/

--card-foreground: 207 100% 14%; /\* Deep Blue \*/

--popover: 0 0% 100%;

--popover-foreground: 207 100% 14%;

--primary: 207 100% 14%; /\* Deep Blue \*/

--primary-foreground: 0 0% 98%; /\* Light Gray/White for text on primary buttons \*/

--secondary: 0 0% 95%; /\* Slightly off-white, lighter than D6D6D6 \*/

--secondary-foreground: 207 100% 14%; /\* Deep Blue text on secondary \*/

--muted: 0 0% 90%; /\* Neutral gray background elements \*/

--muted-foreground: 207 30% 40%; /\* Muted blue/gray text \*/

--accent: 174 72% 63%; /\* Teal \*/

--accent-foreground: 0 0% 10%; /\* Dark Gray/Black for text on accent \*/

--destructive: 0 84.2% 60.2%;

--destructive-foreground: 0 0% 98%;

--border: 0 0% 84%; /\* Light Gray #D6D6D6 \*/

--input: 0 0% 84%; /\* Light Gray #D6D6D6 for input borders \*/

--ring: 174 72% 53%; /\* Darker Teal for focus rings \*/

--chart-1: 12 76% 61%;

--chart-2: 173 58% 39%;

--chart-3: 197 37% 24%;

--chart-4: 43 74% 66%;

--chart-5: 27 87% 67%;

--radius: 0.5rem;

/\* Sidebar specific theme variables (adjust if sidebar is visually distinct) \*/

--sidebar-background: 207 100% 10%; /\* Darker Deep Blue for sidebar \*/

--sidebar-foreground: 0 0% 90%; /\* Light gray for sidebar text \*/

--sidebar-primary: 174 72% 63%; /\* Teal as primary in sidebar \*/

--sidebar-primary-foreground: 207 100% 10%; /\* Dark blue/black text on sidebar primary \*/

--sidebar-accent: 174 72% 53%; /\* Darker Teal for sidebar accent \*/

--sidebar-accent-foreground: 0 0% 98%; /\* White/Light gray on sidebar accent \*/

--sidebar-border: 207 100% 20%; /\* Border for sidebar elements \*/

--sidebar-ring: 174 72% 73%; /\* Lighter Teal for sidebar rings \*/

}

.dark {

--background: 207 30% 10%; /\* Very Dark Blue \*/

--foreground: 0 0% 95%; /\* Light Gray \*/

--card: 207 30% 15%; /\* Dark Blue, lighter than background \*/

--card-foreground: 0 0% 95%;

--popover: 207 30% 15%;

--popover-foreground: 0 0% 95%;

--primary: 174 72% 63%; /\* Teal as primary in dark mode \*/

--primary-foreground: 207 100% 10%; /\* Dark Blue/Black text on Teal \*/

--secondary: 207 30% 20%; /\* Darker Blue \*/

--secondary-foreground: 0 0% 95%;

--muted: 207 30% 25%;

--muted-foreground: 0 0% 60%; /\* Lighter muted text \*/

--accent: 207 100% 25%; /\* Deep Blue, slightly lighter for accent \*/

--accent-foreground: 0 0% 95%; /\* Light Gray text on accent \*/

--destructive: 0 62.8% 30.6%;

--destructive-foreground: 0 0% 98%;

--border: 207 30% 30%; /\* Darker gray/blue \*/

--input: 207 30% 30%;

--ring: 174 72% 73%; /\* Lighter Teal for focus rings \*/

/\* Sidebar specific theme variables for dark mode \*/

--sidebar-background: 0 0% 8%; /\* Even darker for dark sidebar \*/

--sidebar-foreground: 0 0% 80%;

--sidebar-primary: 174 72% 58%;

--sidebar-primary-foreground: 0 0% 10%;

--sidebar-accent: 174 72% 48%;

--sidebar-accent-foreground: 0 0% 95%;

--sidebar-border: 0 0% 15%;

--sidebar-ring: 174 72% 68%;

}

}

@layer base {

\* {

@apply border-border;

}

body {

@apply bg-background text-foreground;

/\* Removed explicit font-family: Arial, Helvetica, sans-serif; as Geist font is applied in layout.tsx \*/

}

}

**src/app/layout.tsx:**

import type {Metadata} from 'next';

import {Geist, Geist\_Mono} from 'next/font/google';

import './globals.css';

import { SidebarProvider } from "@/components/ui/sidebar";

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

import Header from "@/components/Header";

const geistSans = Geist({

variable: '--font-geist-sans',

subsets: ['latin'],

});

const geistMono = Geist\_Mono({

variable: '--font-geist-mono',

subsets: ['latin'],

});

export const metadata: Metadata = {

title: 'SignSpeak',

description: 'Translate text to sign language and vice-versa.',

};

export default function RootLayout({

children,

}: Readonly<{

children: React.ReactNode;

}>) {

return (

<html lang="en" suppressHydrationWarning={true}>

<body className={`${geistSans.variable} ${geistMono.variable} antialiased`}>

<SidebarProvider>

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

<Header />

<main className="flex-grow">

{children}

</main>

<Toaster />

</div>

</SidebarProvider>

</body>

</html>

);

}

**src/app/page.tsx:**

import TranslationInterface from "@/components/TranslationInterface";

export default function HomePage() {

return (

<div className="w-full">

<TranslationInterface />

</div>

);

}

**src/components/Header.tsx:**

import Link from 'next/link';

import LogoIcon from '@/components/icons/LogoIcon';

const Header = () => {

return (

<header className="bg-card border-b border-border shadow-sm">

<div className="container mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">

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

<Link href="/" className="flex items-center gap-2 text-2xl font-bold text-primary">

<LogoIcon className="h-8 w-8 text-accent" />

<span>SignSpeak</span>

</Link>

{/\* Navigation items can be added here if needed \*/}

</div>

</div>

</header>

);

};

export default Header;

**src/components/TranslationInterface.tsx:**

"use client";

import { useState, useEffect } from 'react';

import Image from 'next/image';

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";

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

import { Textarea } from "@/components/ui/textarea";

import { Input } from "@/components/ui/input";

import { Card, CardContent, CardHeader, CardTitle, CardFooter, CardDescription } from "@/components/ui/card";

import { Slider } from "@/components/ui/slider";

import { Label } from "@/components/ui/label";

import { ScrollArea } from "@/components/ui/scroll-area";

import { useToast } from "@/hooks/use-toast";

import { translateTextToSign } from '@/ai/flows/text-to-sign-translation';

import { signToTextTranslation } from '@/ai/flows/sign-to-text-translation';

import type { SavedTranslation, SignStep, HandUsage } from '@/types';

import { getItem, setItem } from '@/lib/localStorage';

import { Camera, Loader2, MessageSquareText, Save, UploadCloud, ZoomInIcon, Hand, Users2, Trash2 } from 'lucide-react';

const SAVED\_TRANSLATIONS\_KEY = 'signspeak-translations';

const getHandUsageDisplay = (handUsage: HandUsage) => {

switch (handUsage) {

case 'left\_hand': return { text: "Left Hand", Icon: Hand };

case 'right\_hand': return { text: "Right Hand", Icon: Hand };

case 'both\_hands': return { text: "Both Hands", Icon: Users2 };

default: return { text: "", Icon: Hand };

}

};

export default function TranslationInterface() {

const [activeTab, setActiveTab] = useState<'text-to-sign' | 'sign-to-text'>('text-to-sign');

const [textInput, setTextInput] = useState<string>('');

const [signImageFile, setSignImageFile] = useState<File | null>(null);

const [signImageDataUri, setSignImageDataUri] = useState<string | null>(null);

const [output, setOutput] = useState<string | SignStep[] | null>(null);

const [isLoading, setIsLoading] = useState<boolean>(false);

const [zoomLevel, setZoomLevel] = useState<number>(100);

const { toast } = useToast();

useEffect(() => {

setOutput(null);

}, [activeTab]);

const handleFileChange = (event: React.ChangeEvent<HTMLInputElement>) => {

const file = event.target.files?.[0];

if (file) {

setSignImageFile(file);

const reader = new FileReader();

reader.onloadend = () => {

setSignImageDataUri(reader.result as string);

setOutput(null);

};

reader.readAsDataURL(file);

} else {

setSignImageFile(null);

setSignImageDataUri(null);

}

};

const handleTextToSign = async () => {

if (!textInput.trim()) {

toast({ title: "Input Required", description: "Please enter text to translate.", variant: "destructive" });

return;

}

setIsLoading(true);

setOutput(null);

try {

const result = await translateTextToSign({ text: textInput });

setOutput(result.signSequence);

} catch (error: any) {

console.error("Text-to-Sign Error:", error);

toast({ title: "Translation Error", description: error.message || "Failed to translate text to sign. Please try again.", variant: "destructive" });

setOutput([]);

} finally {

setIsLoading(false);

}

};

const handleSignToText = async () => {

if (!signImageDataUri) {

toast({ title: "Input Required", description: "Please upload an image of sign language.", variant: "destructive" });

return;

}

setIsLoading(true);

setOutput(null);

try {

const result = await signToTextTranslation({ signDataUri: signImageDataUri });

setOutput(result.translation);

} catch (error: any) {

console.error("Sign-to-Text Error:", error);

toast({ title: "Translation Error", description: error.message || "Failed to translate sign to text. Please try again.", variant: "destructive" });

setOutput('');

} finally {

setIsLoading(false);

}

};

const handleSaveTranslation = () => {

if (!output || (Array.isArray(output) && output.length === 0)) {

toast({ title: "Nothing to Save", description: "Please generate a translation first.", variant: "destructive" });

return;

}

let newTranslation: SavedTranslation;

if (activeTab === 'text-to-sign' && Array.isArray(output)) {

newTranslation = {

id: Date.now().toString(),

type: 'text-to-sign',

input: textInput,

output: output,

timestamp: Date.now(),

inputTextPreview: textInput.substring(0, 30) + (textInput.length > 30 ? "..." : ""),

outputTextPreview: `Sign Sequence (${output.length} steps)`,

};

} else if (activeTab === 'sign-to-text' && typeof output === 'string' && output.trim() !== '') {

newTranslation = {

id: Date.now().toString(),

type: 'sign-to-text',

input: signImageDataUri!,

output: output,

timestamp: Date.now(),

inputTextPreview: "Sign Language Image",

outputTextPreview: output.substring(0, 30) + (output.length > 30 ? "..." : ""),

};

} else {

toast({ title: "Save Error", description: "Could not determine data to save or output is empty.", variant: "destructive" });

return;

}

const savedTranslations = getItem<SavedTranslation[]>(SAVED\_TRANSLATIONS\_KEY) || [];

setItem<SavedTranslation[]>(SAVED\_TRANSLATIONS\_KEY, [newTranslation, ...savedTranslations]);

toast({ title: "Translation Saved", description: "Your translation has been saved locally." });

};

const OutputDisplay = () => {

if (isLoading) {

return (

<div className="flex flex-col items-center justify-center min-h-[200px] h-full p-4">

<Loader2 className="h-12 w-12 animate-spin text-primary" />

<p className="mt-4 text-muted-foreground">Translating...</p>

</div>

);

}

if (!output || (Array.isArray(output) && output.length === 0 && !isLoading) || (typeof output === 'string' && output.trim() === '' && !isLoading) ) {

return (

<div className="flex flex-col items-center justify-center min-h-[200px] h-full p-4 border border-dashed rounded-lg bg-muted/30">

{activeTab === 'text-to-sign' ? <MessageSquareText className="h-16 w-16 text-muted-foreground" /> : <Camera className="h-16 w-16 text-muted-foreground" />}

<p className="mt-4 text-center text-muted-foreground">

{activeTab === 'text-to-sign' ? 'Your sign language sequence will appear here once translated.' : 'Your text translation will appear here once translated.'}

</p>

</div>

);

}

if (activeTab === 'text-to-sign' && Array.isArray(output)) {

return (

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

{output.map((step) => {

const handUsageInfo = getHandUsageDisplay(step.handUsage);

return (

<Card key={step.sequenceNumber} className="overflow-hidden shadow-md bg-card border border-border/70">

<CardHeader className="pb-2 pt-3 px-3 md:pt-4 md:px-4">

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

<CardTitle className="text-sm md:text-base leading-tight">

{step.word ? `Sign for "${step.word}"` : `Sign #${step.sequenceNumber}`}

</CardTitle>

<span className="text-xs md:text-sm font-normal flex items-center text-muted-foreground whitespace-nowrap ml-2">

<handUsageInfo.Icon className="mr-1.5 h-3.5 w-3.5 md:h-4 md:w-4 shrink-0" /> {handUsageInfo.text}

</span>

</div>

{step.word && <CardDescription className="text-xs pt-0.5">Sequence: #{step.sequenceNumber}</CardDescription>}

</CardHeader>

<CardContent className="flex flex-col sm:flex-row gap-3 md:gap-4 items-start p-3 md:p-4">

<div className="relative w-full sm:w-1/2 aspect-[4/3] border rounded-md overflow-hidden shadow-sm bg-muted/20">

<Image

src={step.imageDataUri}

alt={`Sign language for: ${step.textualGuide}`}

layout="fill"

objectFit="contain"

style={{ transform: `scale(${zoomLevel / 100})` }}

data-ai-hint="sign language hand gesture"

unoptimized={step.imageDataUri.startsWith('data:image')}

className="p-1"

/>

</div>

<div className="w-full sm:w-1/2 mt-2 sm:mt-0">

<p className="text-xs md:text-sm text-foreground p-2 md:p-2.5 bg-muted/40 rounded-md shadow-sm whitespace-pre-wrap leading-relaxed">

{step.textualGuide}

</p>

</div>

</CardContent>

</Card>

);

})}

</div>

);

} else if (activeTab === 'sign-to-text' && typeof output === 'string') {

return (

<Card className="bg-card shadow-inner border">

<CardHeader className="pb-3 pt-4">

<CardTitle className="text-base md:text-lg">Translated Text</CardTitle>

</CardHeader>

<CardContent className="p-3 md:p-4">

<p className="text-foreground text-sm md:text-base whitespace-pre-wrap p-3 md:p-4 bg-muted/40 rounded-md min-h-[100px]">{output}</p>

</CardContent>

</Card>

);

}

return null;

};

return (

<div className="container mx-auto py-6 md:py-8 px-2 sm:px-4 lg:px-6 max-w-7xl">

<Tabs value={activeTab} onValueChange={(value) => setActiveTab(value as 'text-to-sign' | 'sign-to-text')} className="w-full">

<TabsList className="grid w-full grid-cols-2 mb-4 md:mb-6 bg-muted p-1 rounded-lg">

<TabsTrigger value="text-to-sign" className="data-[state=active]:bg-accent data-[state=active]:text-accent-foreground data-[state=active]:shadow-md py-2 text-sm md:text-base">

<MessageSquareText className="mr-1.5 md:mr-2 h-4 w-4 md:h-5 md:w-5" /> Text to Sign

</TabsTrigger>

<TabsTrigger value="sign-to-text" className="data-[state=active]:bg-accent data-[state=active]:text-accent-foreground data-[state=active]:shadow-md py-2 text-sm md:text-base">

<Camera className="mr-1.5 md:mr-2 h-4 w-4 md:h-5 md:w-5" /> Sign to Text

</TabsTrigger>

</TabsList>

<div className="flex flex-col md:flex-row md:gap-6 lg:gap-8">

{/\* Left Column: Input Controls \*/}

<div className="w-full md:w-1/2 md:max-w-md lg:max-w-lg xl:max-w-xl flex flex-col mb-6 md:mb-0">

<TabsContent value="text-to-sign" className="mt-0 flex-1">

<Card className="shadow-lg h-full flex flex-col">

<CardHeader>

<CardTitle className="text-lg md:text-xl">Text to Sign Language</CardTitle>

<CardDescription>Enter text to see its sign language representation step-by-step.</CardDescription>

</CardHeader>

<CardContent className="space-y-4 flex-grow">

<Textarea

placeholder="Type your text here (e.g., Hello how are you?)"

value={textInput}

onChange={(e) => setTextInput(e.target.value)}

rows={4}

className="resize-none focus:ring-accent focus:border-accent text-sm md:text-base"

/>

</CardContent>

<CardFooter>

<Button onClick={handleTextToSign} disabled={isLoading} className="w-full bg-primary hover:bg-primary/90 text-primary-foreground text-sm md:text-base py-2.5">

{isLoading ? <Loader2 className="mr-2 h-4 w-4 animate-spin" /> : <MessageSquareText className="mr-2 h-4 w-4" />}

Translate to Sign

</Button>

</CardFooter>

</Card>

</TabsContent>

<TabsContent value="sign-to-text" className="mt-0 flex-1">

<Card className="shadow-lg h-full flex flex-col">

<CardHeader>

<CardTitle className="text-lg md:text-xl">Sign Language to Text</CardTitle>

<CardDescription>Upload an image of sign language to translate it into text.</CardDescription>

</CardHeader>

<CardContent className="space-y-4 flex-grow">

<div>

<Label htmlFor="sign-image-upload" className="block text-xs md:text-sm font-medium text-foreground mb-1.5">Upload Sign Image</Label>

<Input

id="sign-image-upload"

type="file"

accept="image/\*"

onChange={handleFileChange}

className="file:mr-4 file:py-2 file:px-3 file:rounded-md file:border-0 file:text-xs file:font-semibold file:bg-accent file:text-accent-foreground hover:file:bg-accent/90 cursor-pointer focus:ring-accent focus:border-accent text-sm"

/>

{signImageDataUri && (

<div className="mt-3 p-2 border rounded-lg bg-muted/30">

<p className="text-xs font-medium text-foreground mb-1">Preview:</p>

<Image src={signImageDataUri} alt="Sign language preview" width={120} height={90} className="rounded-md object-contain border bg-background" data-ai-hint="sign language" />

</div>

)}

</div>

</CardContent>

<CardFooter>

<Button onClick={handleSignToText} disabled={isLoading || !signImageDataUri} className="w-full bg-primary hover:bg-primary/90 text-primary-foreground text-sm md:text-base py-2.5">

{isLoading ? <Loader2 className="mr-2 h-4 w-4 animate-spin" /> : <UploadCloud className="mr-2 h-4 w-4" />}

Translate to Text

</Button>

</CardFooter>

</Card>

</TabsContent>

</div>

{/\* Right Column: Output Display \*/}

<div className="w-full md:w-1/2 flex flex-col">

<Card className="shadow-lg flex-grow flex flex-col">

<CardHeader>

<CardTitle className="text-lg md:text-xl">Translation Result</CardTitle>

<CardDescription>

{isLoading ? "Processing your translation..." :

!output || (Array.isArray(output) && output.length === 0) || (typeof output === 'string' && output.trim() === '') ?

(activeTab === 'text-to-sign' ? 'Your generated sign sequence will appear below.' : 'Your text translation will appear below.') :

(activeTab === 'text-to-sign' ? 'Generated Sign Language Sequence:' : 'Translated Text:')

}

</CardDescription>

</CardHeader>

<CardContent className="flex-grow flex flex-col space-y-4 overflow-hidden p-3 md:p-4">

<ScrollArea className="flex-grow h-0 min-h-[300px] md:min-h-[400px] pr-3 -mr-3">

<OutputDisplay />

</ScrollArea>

{activeTab === 'text-to-sign' && Array.isArray(output) && output.length > 0 && (

<div className="pt-3 border-t mt-auto">

<Label htmlFor="zoom-slider" className="flex items-center mb-1.5 text-xs md:text-sm">

<ZoomInIcon className="mr-1.5 h-3.5 w-3.5 md:h-4 md:w-4 text-muted-foreground"/> Zoom Images ({zoomLevel}%)

</Label>

<Slider

id="zoom-slider"

min={50}

max={150}

step={10}

value={[zoomLevel]}

onValueChange={(value) => setZoomLevel(value[0])}

className="[&>span:first-child]:h-1.5 [&>span:first-child>span]:bg-accent [&>span:last-child]:bg-accent [&>span:last-child]:h-4 [&>span:last-child]:w-4 [&>span:last-child]:border-2"

/>

</div>

)}

</CardContent>

<CardFooter className="mt-auto">

<Button onClick={handleSaveTranslation} disabled={!output || isLoading || (Array.isArray(output) && output.length === 0) || (typeof output === 'string' && output.trim() === '')} variant="outline" className="w-full text-sm md:text-base py-2.5">

<Save className="mr-2 h-4 w-4" /> Save Translation

</Button>

</CardFooter>

</Card>

</div>

</div>

</Tabs>

</div>

);

}

**src/components/icons/LogoIcon.tsx:**

"use client";

import { Languages } from 'lucide-react';

import type { LucideProps } from 'lucide-react';

const LogoIcon = (props: LucideProps) => {

return <Languages {...props} />;

};

export default LogoIcon;

**src/components/ui/accordion.tsx:**

"use client"

import \* as React from "react"

import \* as AccordionPrimitive from "@radix-ui/react-accordion"

import { ChevronDown } from "lucide-react"

import { cn } from "@/lib/utils"

const Accordion = AccordionPrimitive.Root

const AccordionItem = React.forwardRef<

React.ElementRef<typeof AccordionPrimitive.Item>,

React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Item>

>(({ className, ...props }, ref) => (

<AccordionPrimitive.Item

ref={ref}

className={cn("border-b", className)}

{...props}

/>

))

AccordionItem.displayName = "AccordionItem"

const AccordionTrigger = React.forwardRef<

React.ElementRef<typeof AccordionPrimitive.Trigger>,

React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Trigger>

>(({ className, children, ...props }, ref) => (

<AccordionPrimitive.Header className="flex">

<AccordionPrimitive.Trigger

ref={ref}

className={cn(

"flex flex-1 items-center justify-between py-4 font-medium transition-all hover:underline [&[data-state=open]>svg]:rotate-180",

className

)}

{...props}

>

{children}

<ChevronDown className="h-4 w-4 shrink-0 transition-transform duration-200" />

</AccordionPrimitive.Trigger>

</AccordionPrimitive.Header>

))

AccordionTrigger.displayName = AccordionPrimitive.Trigger.displayName

const AccordionContent = React.forwardRef<

React.ElementRef<typeof AccordionPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof AccordionPrimitive.Content>

>(({ className, children, ...props }, ref) => (

<AccordionPrimitive.Content

ref={ref}

className="overflow-hidden text-sm transition-all data-[state=closed]:animate-accordion-up data-[state=open]:animate-accordion-down"

{...props}

>

<div className={cn("pb-4 pt-0", className)}>{children}</div>

</AccordionPrimitive.Content>

))

AccordionContent.displayName = AccordionPrimitive.Content.displayName

export { Accordion, AccordionItem, AccordionTrigger, AccordionContent }

**src/components/ui/alert-dialog.tsx:**

"use client"

import \* as React from "react"

import \* as AlertDialogPrimitive from "@radix-ui/react-alert-dialog"

import { cn } from "@/lib/utils"

import { buttonVariants } from "@/components/ui/button"

const AlertDialog = AlertDialogPrimitive.Root

const AlertDialogTrigger = AlertDialogPrimitive.Trigger

const AlertDialogPortal = AlertDialogPrimitive.Portal

const AlertDialogOverlay = React.forwardRef<

React.ElementRef<typeof AlertDialogPrimitive.Overlay>,

React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Overlay>

>(({ className, ...props }, ref) => (

<AlertDialogPrimitive.Overlay

className={cn(

"fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",

className

)}

{...props}

ref={ref}

/>

))

AlertDialogOverlay.displayName = AlertDialogPrimitive.Overlay.displayName

const AlertDialogContent = React.forwardRef<

React.ElementRef<typeof AlertDialogPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Content>

>(({ className, ...props }, ref) => (

<AlertDialogPortal>

<AlertDialogOverlay />

<AlertDialogPrimitive.Content

ref={ref}

className={cn(

"fixed left-[50%] top-[50%] z-50 grid w-full max-w-lg translate-x-[-50%] translate-y-[-50%] gap-4 border bg-background p-6 shadow-lg duration-200 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[state=closed]:slide-out-to-left-1/2 data-[state=closed]:slide-out-to-top-[48%] data-[state=open]:slide-in-from-left-1/2 data-[state=open]:slide-in-from-top-[48%] sm:rounded-lg",

className

)}

{...props}

/>

</AlertDialogPortal>

))

AlertDialogContent.displayName = AlertDialogPrimitive.Content.displayName

const AlertDialogHeader = ({

className,

...props

}: React.HTMLAttributes<HTMLDivElement>) => (

<div

className={cn(

"flex flex-col space-y-2 text-center sm:text-left",

className

)}

{...props}

/>

)

AlertDialogHeader.displayName = "AlertDialogHeader"

const AlertDialogFooter = ({

className,

...props

}: React.HTMLAttributes<HTMLDivElement>) => (

<div

className={cn(

"flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",

className

)}

{...props}

/>

)

AlertDialogFooter.displayName = "AlertDialogFooter"

const AlertDialogTitle = React.forwardRef<

React.ElementRef<typeof AlertDialogPrimitive.Title>,

React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Title>

>(({ className, ...props }, ref) => (

<AlertDialogPrimitive.Title

ref={ref}

className={cn("text-lg font-semibold", className)}

{...props}

/>

))

AlertDialogTitle.displayName = AlertDialogPrimitive.Title.displayName

const AlertDialogDescription = React.forwardRef<

React.ElementRef<typeof AlertDialogPrimitive.Description>,

React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Description>

>(({ className, ...props }, ref) => (

<AlertDialogPrimitive.Description

ref={ref}

className={cn("text-sm text-muted-foreground", className)}

{...props}

/>

))

AlertDialogDescription.displayName =

AlertDialogPrimitive.Description.displayName

const AlertDialogAction = React.forwardRef<

React.ElementRef<typeof AlertDialogPrimitive.Action>,

React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Action>

>(({ className, ...props }, ref) => (

<AlertDialogPrimitive.Action

ref={ref}

className={cn(buttonVariants(), className)}

{...props}

/>

))

AlertDialogAction.displayName = AlertDialogPrimitive.Action.displayName

const AlertDialogCancel = React.forwardRef<

React.ElementRef<typeof AlertDialogPrimitive.Cancel>,

React.ComponentPropsWithoutRef<typeof AlertDialogPrimitive.Cancel>

>(({ className, ...props }, ref) => (

<AlertDialogPrimitive.Cancel

ref={ref}

className={cn(

buttonVariants({ variant: "outline" }),

"mt-2 sm:mt-0",

className

)}

{...props}

/>

))

AlertDialogCancel.displayName = AlertDialogPrimitive.Cancel.displayName

export {

AlertDialog,

AlertDialogPortal,

AlertDialogOverlay,

AlertDialogTrigger,

AlertDialogContent,

AlertDialogHeader,

AlertDialogFooter,

AlertDialogTitle,

AlertDialogDescription,

AlertDialogAction,

AlertDialogCancel,

}

**src/components/ui/alert.tsx:**

import \* as React from "react"

import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const alertVariants = cva(

"relative w-full rounded-lg border p-4 [&>svg~\*]:pl-7 [&>svg+div]:translate-y-[-3px] [&>svg]:absolute [&>svg]:left-4 [&>svg]:top-4 [&>svg]:text-foreground",

{

variants: {

variant: {

default: "bg-background text-foreground",

destructive:

"border-destructive/50 text-destructive dark:border-destructive [&>svg]:text-destructive",

},

},

defaultVariants: {

variant: "default",

},

}

)

const Alert = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement> & VariantProps<typeof alertVariants>

>(({ className, variant, ...props }, ref) => (

<div

ref={ref}

role="alert"

className={cn(alertVariants({ variant }), className)}

{...props}

/>

))

Alert.displayName = "Alert"

const AlertTitle = React.forwardRef<

HTMLParagraphElement,

React.HTMLAttributes<HTMLHeadingElement>

>(({ className, ...props }, ref) => (

<h5

ref={ref}

className={cn("mb-1 font-medium leading-none tracking-tight", className)}

{...props}

/>

))

AlertTitle.displayName = "AlertTitle"

const AlertDescription = React.forwardRef<

HTMLParagraphElement,

React.HTMLAttributes<HTMLParagraphElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn("text-sm [&\_p]:leading-relaxed", className)}

{...props}

/>

))

AlertDescription.displayName = "AlertDescription"

export { Alert, AlertTitle, AlertDescription }

**src/components/ui/avatar.tsx:**

"use client"

import \* as React from "react"

import \* as AvatarPrimitive from "@radix-ui/react-avatar"

import { cn } from "@/lib/utils"

const Avatar = React.forwardRef<

React.ElementRef<typeof AvatarPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Root>

>(({ className, ...props }, ref) => (

<AvatarPrimitive.Root

ref={ref}

className={cn(

"relative flex h-10 w-10 shrink-0 overflow-hidden rounded-full",

className

)}

{...props}

/>

))

Avatar.displayName = AvatarPrimitive.Root.displayName

const AvatarImage = React.forwardRef<

React.ElementRef<typeof AvatarPrimitive.Image>,

React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Image>

>(({ className, ...props }, ref) => (

<AvatarPrimitive.Image

ref={ref}

className={cn("aspect-square h-full w-full", className)}

{...props}

/>

))

AvatarImage.displayName = AvatarPrimitive.Image.displayName

const AvatarFallback = React.forwardRef<

React.ElementRef<typeof AvatarPrimitive.Fallback>,

React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Fallback>

>(({ className, ...props }, ref) => (

<AvatarPrimitive.Fallback

ref={ref}

className={cn(

"flex h-full w-full items-center justify-center rounded-full bg-muted",

className

)}

{...props}

/>

))

AvatarFallback.displayName = AvatarPrimitive.Fallback.displayName

export { Avatar, AvatarImage, AvatarFallback }

**src/components/ui/badge.tsx:**

import \* as React from "react"

import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const badgeVariants = cva(

"inline-flex items-center rounded-full border px-2.5 py-0.5 text-xs font-semibold transition-colors focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2",

{

variants: {

variant: {

default:

"border-transparent bg-primary text-primary-foreground hover:bg-primary/80",

secondary:

"border-transparent bg-secondary text-secondary-foreground hover:bg-secondary/80",

destructive:

"border-transparent bg-destructive text-destructive-foreground hover:bg-destructive/80",

outline: "text-foreground",

},

},

defaultVariants: {

variant: "default",

},

}

)

export interface BadgeProps

extends React.HTMLAttributes<HTMLDivElement>,

VariantProps<typeof badgeVariants> {}

function Badge({ className, variant, ...props }: BadgeProps) {

return (

<div className={cn(badgeVariants({ variant }), className)} {...props} />

)

}

export { Badge, badgeVariants }

**src/components/ui/button.tsx:**

import \* as React from "react"

import { Slot } from "@radix-ui/react-slot"

import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const buttonVariants = cva(

"inline-flex items-center justify-center gap-2 whitespace-nowrap rounded-md text-sm font-medium ring-offset-background transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 [&\_svg]:pointer-events-none [&\_svg]:size-4 [&\_svg]:shrink-0",

{

variants: {

variant: {

default: "bg-primary text-primary-foreground hover:bg-primary/90",

destructive:

"bg-destructive text-destructive-foreground hover:bg-destructive/90",

outline:

"border border-input bg-background hover:bg-accent hover:text-accent-foreground",

secondary:

"bg-secondary text-secondary-foreground hover:bg-secondary/80",

ghost: "hover:bg-accent hover:text-accent-foreground",

link: "text-primary underline-offset-4 hover:underline",

},

size: {

default: "h-10 px-4 py-2",

sm: "h-9 rounded-md px-3",

lg: "h-11 rounded-md px-8",

icon: "h-10 w-10",

},

},

defaultVariants: {

variant: "default",

size: "default",

},

}

)

export interface ButtonProps

extends React.ButtonHTMLAttributes<HTMLButtonElement>,

VariantProps<typeof buttonVariants> {

asChild?: boolean

}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(

({ className, variant, size, asChild = false, ...props }, ref) => {

const Comp = asChild ? Slot : "button"

return (

<Comp

className={cn(buttonVariants({ variant, size, className }))}

ref={ref}

{...props}

/>

)

}

)

Button.displayName = "Button"

export { Button, buttonVariants }

**src/components/ui/calendar.tsx:**

"use client"

import \* as React from "react"

import { ChevronLeft, ChevronRight } from "lucide-react"

import { DayPicker } from "react-day-picker"

import { cn } from "@/lib/utils"

import { buttonVariants } from "@/components/ui/button"

export type CalendarProps = React.ComponentProps<typeof DayPicker>

function Calendar({

className,

classNames,

showOutsideDays = true,

...props

}: CalendarProps) {

return (

<DayPicker

showOutsideDays={showOutsideDays}

className={cn("p-3", className)}

classNames={{

months: "flex flex-col sm:flex-row space-y-4 sm:space-x-4 sm:space-y-0",

month: "space-y-4",

caption: "flex justify-center pt-1 relative items-center",

caption\_label: "text-sm font-medium",

nav: "space-x-1 flex items-center",

nav\_button: cn(

buttonVariants({ variant: "outline" }),

"h-7 w-7 bg-transparent p-0 opacity-50 hover:opacity-100"

),

nav\_button\_previous: "absolute left-1",

nav\_button\_next: "absolute right-1",

table: "w-full border-collapse space-y-1",

head\_row: "flex",

head\_cell:

"text-muted-foreground rounded-md w-9 font-normal text-[0.8rem]",

row: "flex w-full mt-2",

cell: "h-9 w-9 text-center text-sm p-0 relative [&:has([aria-selected].day-range-end)]:rounded-r-md [&:has([aria-selected].day-outside)]:bg-accent/50 [&:has([aria-selected])]:bg-accent first:[&:has([aria-selected])]:rounded-l-md last:[&:has([aria-selected])]:rounded-r-md focus-within:relative focus-within:z-20",

day: cn(

buttonVariants({ variant: "ghost" }),

"h-9 w-9 p-0 font-normal aria-selected:opacity-100"

),

day\_range\_end: "day-range-end",

day\_selected:

"bg-primary text-primary-foreground hover:bg-primary hover:text-primary-foreground focus:bg-primary focus:text-primary-foreground",

day\_today: "bg-accent text-accent-foreground",

day\_outside:

"day-outside text-muted-foreground aria-selected:bg-accent/50 aria-selected:text-muted-foreground",

day\_disabled: "text-muted-foreground opacity-50",

day\_range\_middle:

"aria-selected:bg-accent aria-selected:text-accent-foreground",

day\_hidden: "invisible",

...classNames,

}}

components={{

IconLeft: ({ className, ...props }) => (

<ChevronLeft className={cn("h-4 w-4", className)} {...props} />

),

IconRight: ({ className, ...props }) => (

<ChevronRight className={cn("h-4 w-4", className)} {...props} />

),

}}

{...props}

/>

)

}

Calendar.displayName = "Calendar"

export { Calendar }

**src/components/ui/card.tsx:**

import \* as React from "react"

import { cn } from "@/lib/utils"

const Card = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn(

"rounded-lg border bg-card text-card-foreground shadow-sm",

className

)}

{...props}

/>

))

Card.displayName = "Card"

const CardHeader = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn("flex flex-col space-y-1.5 p-6", className)}

{...props}

/>

))

CardHeader.displayName = "CardHeader"

const CardTitle = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn(

"text-2xl font-semibold leading-none tracking-tight",

className

)}

{...props}

/>

))

CardTitle.displayName = "CardTitle"

const CardDescription = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn("text-sm text-muted-foreground", className)}

{...props}

/>

))

CardDescription.displayName = "CardDescription"

const CardContent = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div ref={ref} className={cn("p-6 pt-0", className)} {...props} />

))

CardContent.displayName = "CardContent"

const CardFooter = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => (

<div

ref={ref}

className={cn("flex items-center p-6 pt-0", className)}

{...props}

/>

))

CardFooter.displayName = "CardFooter"

export { Card, CardHeader, CardFooter, CardTitle, CardDescription, CardContent }

**src/components/ui/chart.tsx:**

"use client"

import \* as React from "react"

import \* as RechartsPrimitive from "recharts"

import { cn } from "@/lib/utils"

// Format: { THEME\_NAME: CSS\_SELECTOR }

const THEMES = { light: "", dark: ".dark" } as const

export type ChartConfig = {

[k in string]: {

label?: React.ReactNode

icon?: React.ComponentType

} & (

| { color?: string; theme?: never }

| { color?: never; theme: Record<keyof typeof THEMES, string> }

)

}

type ChartContextProps = {

config: ChartConfig

}

const ChartContext = React.createContext<ChartContextProps | null>(null)

function useChart() {

const context = React.useContext(ChartContext)

if (!context) {

throw new Error("useChart must be used within a <ChartContainer />")

}

return context

}

const ChartContainer = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div"> & {

config: ChartConfig

children: React.ComponentProps<

typeof RechartsPrimitive.ResponsiveContainer

>["children"]

}

>(({ id, className, children, config, ...props }, ref) => {

const uniqueId = React.useId()

const chartId = `chart-${id || uniqueId.replace(/:/g, "")}`

return (

<ChartContext.Provider value={{ config }}>

<div

data-chart={chartId}

ref={ref}

className={cn(

"flex aspect-video justify-center text-xs [&\_.recharts-cartesian-axis-tick\_text]:fill-muted-foreground [&\_.recharts-cartesian-grid\_line[stroke='#ccc']]:stroke-border/50 [&\_.recharts-curve.recharts-tooltip-cursor]:stroke-border [&\_.recharts-dot[stroke='#fff']]:stroke-transparent [&\_.recharts-layer]:outline-none [&\_.recharts-polar-grid\_[stroke='#ccc']]:stroke-border [&\_.recharts-radial-bar-background-sector]:fill-muted [&\_.recharts-rectangle.recharts-tooltip-cursor]:fill-muted [&\_.recharts-reference-line\_[stroke='#ccc']]:stroke-border [&\_.recharts-sector[stroke='#fff']]:stroke-transparent [&\_.recharts-sector]:outline-none [&\_.recharts-surface]:outline-none",

className

)}

{...props}

>

<ChartStyle id={chartId} config={config} />

<RechartsPrimitive.ResponsiveContainer>

{children}

</RechartsPrimitive.ResponsiveContainer>

</div>

</ChartContext.Provider>

)

})

ChartContainer.displayName = "Chart"

const ChartStyle = ({ id, config }: { id: string; config: ChartConfig }) => {

const colorConfig = Object.entries(config).filter(

([, config]) => config.theme || config.color

)

if (!colorConfig.length) {

return null

}

return (

<style

dangerouslySetInnerHTML={{

\_\_html: Object.entries(THEMES)

.map(

([theme, prefix]) => `

${prefix} [data-chart=${id}] {

${colorConfig

.map(([key, itemConfig]) => {

const color =

itemConfig.theme?.[theme as keyof typeof itemConfig.theme] ||

itemConfig.color

return color ? ` --color-${key}: ${color};` : null

})

.join("\n")}

}

`

)

.join("\n"),

}}

/>

)

}

const ChartTooltip = RechartsPrimitive.Tooltip

const ChartTooltipContent = React.forwardRef<

HTMLDivElement,

React.ComponentProps<typeof RechartsPrimitive.Tooltip> &

React.ComponentProps<"div"> & {

hideLabel?: boolean

hideIndicator?: boolean

indicator?: "line" | "dot" | "dashed"

nameKey?: string

labelKey?: string

}

>(

(

{

active,

payload,

className,

indicator = "dot",

hideLabel = false,

hideIndicator = false,

label,

labelFormatter,

labelClassName,

formatter,

color,

nameKey,

labelKey,

},

ref

) => {

const { config } = useChart()

const tooltipLabel = React.useMemo(() => {

if (hideLabel || !payload?.length) {

return null

}

const [item] = payload

const key = `${labelKey || item.dataKey || item.name || "value"}`

const itemConfig = getPayloadConfigFromPayload(config, item, key)

const value =

!labelKey && typeof label === "string"

? config[label as keyof typeof config]?.label || label

: itemConfig?.label

if (labelFormatter) {

return (

<div className={cn("font-medium", labelClassName)}>

{labelFormatter(value, payload)}

</div>

)

}

if (!value) {

return null

}

return <div className={cn("font-medium", labelClassName)}>{value}</div>

}, [

label,

labelFormatter,

payload,

hideLabel,

labelClassName,

config,

labelKey,

])

if (!active || !payload?.length) {

return null

}

const nestLabel = payload.length === 1 && indicator !== "dot"

return (

<div

ref={ref}

className={cn(

"grid min-w-[8rem] items-start gap-1.5 rounded-lg border border-border/50 bg-background px-2.5 py-1.5 text-xs shadow-xl",

className

)}

>

{!nestLabel ? tooltipLabel : null}

<div className="grid gap-1.5">

{payload.map((item, index) => {

const key = `${nameKey || item.name || item.dataKey || "value"}`

const itemConfig = getPayloadConfigFromPayload(config, item, key)

const indicatorColor = color || item.payload.fill || item.color

return (

<div

key={item.dataKey}

className={cn(

"flex w-full flex-wrap items-stretch gap-2 [&>svg]:h-2.5 [&>svg]:w-2.5 [&>svg]:text-muted-foreground",

indicator === "dot" && "items-center"

)}

>

{formatter && item?.value !== undefined && item.name ? (

formatter(item.value, item.name, item, index, item.payload)

) : (

<>

{itemConfig?.icon ? (

<itemConfig.icon />

) : (

!hideIndicator && (

<div

className={cn(

"shrink-0 rounded-[2px] border-[--color-border] bg-[--color-bg]",

{

"h-2.5 w-2.5": indicator === "dot",

"w-1": indicator === "line",

"w-0 border-[1.5px] border-dashed bg-transparent":

indicator === "dashed",

"my-0.5": nestLabel && indicator === "dashed",

}

)}

style={

{

"--color-bg": indicatorColor,

"--color-border": indicatorColor,

} as React.CSSProperties

}

/>

)

)}

<div

className={cn(

"flex flex-1 justify-between leading-none",

nestLabel ? "items-end" : "items-center"

)}

>

<div className="grid gap-1.5">

{nestLabel ? tooltipLabel : null}

<span className="text-muted-foreground">

{itemConfig?.label || item.name}

</span>

</div>

{item.value && (

<span className="font-mono font-medium tabular-nums text-foreground">

{item.value.toLocaleString()}

</span>

)}

</div>

</>

)}

</div>

)

})}

</div>

</div>

)

}

)

ChartTooltipContent.displayName = "ChartTooltip"

const ChartLegend = RechartsPrimitive.Legend

const ChartLegendContent = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div"> &

Pick<RechartsPrimitive.LegendProps, "payload" | "verticalAlign"> & {

hideIcon?: boolean

nameKey?: string

}

>(

(

{ className, hideIcon = false, payload, verticalAlign = "bottom", nameKey },

ref

) => {

const { config } = useChart()

if (!payload?.length) {

return null

}

return (

<div

ref={ref}

className={cn(

"flex items-center justify-center gap-4",

verticalAlign === "top" ? "pb-3" : "pt-3",

className

)}

>

{payload.map((item) => {

const key = `${nameKey || item.dataKey || "value"}`

const itemConfig = getPayloadConfigFromPayload(config, item, key)

return (

<div

key={item.value}

className={cn(

"flex items-center gap-1.5 [&>svg]:h-3 [&>svg]:w-3 [&>svg]:text-muted-foreground"

)}

>

{itemConfig?.icon && !hideIcon ? (

<itemConfig.icon />

) : (

<div

className="h-2 w-2 shrink-0 rounded-[2px]"

style={{

backgroundColor: item.color,

}}

/>

)}

{itemConfig?.label}

</div>

)

})}

</div>

)

}

)

ChartLegendContent.displayName = "ChartLegend"

// Helper to extract item config from a payload.

function getPayloadConfigFromPayload(

config: ChartConfig,

payload: unknown,

key: string

) {

if (typeof payload !== "object" || payload === null) {

return undefined

}

const payloadPayload =

"payload" in payload &&

typeof payload.payload === "object" &&

payload.payload !== null

? payload.payload

: undefined

let configLabelKey: string = key

if (

key in payload &&

typeof payload[key as keyof typeof payload] === "string"

) {

configLabelKey = payload[key as keyof typeof payload] as string

} else if (

payloadPayload &&

key in payloadPayload &&

typeof payloadPayload[key as keyof typeof payloadPayload] === "string"

) {

configLabelKey = payloadPayload[

key as keyof typeof payloadPayload

] as string

}

return configLabelKey in config

? config[configLabelKey]

: config[key as keyof typeof config]

}

export {

ChartContainer,

ChartTooltip,

ChartTooltipContent,

ChartLegend,

ChartLegendContent,

ChartStyle,

}

**src/components/ui/checkbox.tsx:**

"use client"

import \* as React from "react"

import \* as CheckboxPrimitive from "@radix-ui/react-checkbox"

import { Check } from "lucide-react"

import { cn } from "@/lib/utils"

const Checkbox = React.forwardRef<

React.ElementRef<typeof CheckboxPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof CheckboxPrimitive.Root>

>(({ className, ...props }, ref) => (

<CheckboxPrimitive.Root

ref={ref}

className={cn(

"peer h-4 w-4 shrink-0 rounded-sm border border-primary ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 data-[state=checked]:bg-primary data-[state=checked]:text-primary-foreground",

className

)}

{...props}

>

<CheckboxPrimitive.Indicator

className={cn("flex items-center justify-center text-current")}

>

<Check className="h-4 w-4" />

</CheckboxPrimitive.Indicator>

</CheckboxPrimitive.Root>

))

Checkbox.displayName = CheckboxPrimitive.Root.displayName

export { Checkbox }

**src/components/ui/dialog.tsx:**

"use client"

import \* as React from "react"

import \* as DialogPrimitive from "@radix-ui/react-dialog"

import { X } from "lucide-react"

import { cn } from "@/lib/utils"

const Dialog = DialogPrimitive.Root

const DialogTrigger = DialogPrimitive.Trigger

const DialogPortal = DialogPrimitive.Portal

const DialogClose = DialogPrimitive.Close

const DialogOverlay = React.forwardRef<

React.ElementRef<typeof DialogPrimitive.Overlay>,

React.ComponentPropsWithoutRef<typeof DialogPrimitive.Overlay>

>(({ className, ...props }, ref) => (

<DialogPrimitive.Overlay

ref={ref}

className={cn(

"fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",

className

)}

{...props}

/>

))

DialogOverlay.displayName = DialogPrimitive.Overlay.displayName

const DialogContent = React.forwardRef<

React.ElementRef<typeof DialogPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof DialogPrimitive.Content>

>(({ className, children, ...props }, ref) => (

<DialogPortal>

<DialogOverlay />

<DialogPrimitive.Content

ref={ref}

className={cn(

"fixed left-[50%] top-[50%] z-50 grid w-full max-w-lg translate-x-[-50%] translate-y-[-50%] gap-4 border bg-background p-6 shadow-lg duration-200 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[state=closed]:slide-out-to-left-1/2 data-[state=closed]:slide-out-to-top-[48%] data-[state=open]:slide-in-from-left-1/2 data-[state=open]:slide-in-from-top-[48%] sm:rounded-lg",

className

)}

{...props}

>

{children}

<DialogPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70 ring-offset-background transition-opacity hover:opacity-100 focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:pointer-events-none data-[state=open]:bg-accent data-[state=open]:text-muted-foreground">

<X className="h-4 w-4" />

<span className="sr-only">Close</span>

</DialogPrimitive.Close>

</DialogPrimitive.Content>

</DialogPortal>

))

DialogContent.displayName = DialogPrimitive.Content.displayName

const DialogHeader = ({

className,

...props

}: React.HTMLAttributes<HTMLDivElement>) => (

<div

className={cn(

"flex flex-col space-y-1.5 text-center sm:text-left",

className

)}

{...props}

/>

)

DialogHeader.displayName = "DialogHeader"

const DialogFooter = ({

className,

...props

}: React.HTMLAttributes<HTMLDivElement>) => (

<div

className={cn(

"flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",

className

)}

{...props}

/>

)

DialogFooter.displayName = "DialogFooter"

const DialogTitle = React.forwardRef<

React.ElementRef<typeof DialogPrimitive.Title>,

React.ComponentPropsWithoutRef<typeof DialogPrimitive.Title>

>(({ className, ...props }, ref) => (

<DialogPrimitive.Title

ref={ref}

className={cn(

"text-lg font-semibold leading-none tracking-tight",

className

)}

{...props}

/>

))

DialogTitle.displayName = DialogPrimitive.Title.displayName

const DialogDescription = React.forwardRef<

React.ElementRef<typeof DialogPrimitive.Description>,

React.ComponentPropsWithoutRef<typeof DialogPrimitive.Description>

>(({ className, ...props }, ref) => (

<DialogPrimitive.Description

ref={ref}

className={cn("text-sm text-muted-foreground", className)}

{...props}

/>

))

DialogDescription.displayName = DialogPrimitive.Description.displayName

export {

Dialog,

DialogPortal,

DialogOverlay,

DialogClose,

DialogTrigger,

DialogContent,

DialogHeader,

DialogFooter,

DialogTitle,

DialogDescription,

}

**src/components/ui/dropdown-menu.tsx:**

"use client"

import \* as React from "react"

import \* as DropdownMenuPrimitive from "@radix-ui/react-dropdown-menu"

import { Check, ChevronRight, Circle } from "lucide-react"

import { cn } from "@/lib/utils"

const DropdownMenu = DropdownMenuPrimitive.Root

const DropdownMenuTrigger = DropdownMenuPrimitive.Trigger

const DropdownMenuGroup = DropdownMenuPrimitive.Group

const DropdownMenuPortal = DropdownMenuPrimitive.Portal

const DropdownMenuSub = DropdownMenuPrimitive.Sub

const DropdownMenuRadioGroup = DropdownMenuPrimitive.RadioGroup

const DropdownMenuSubTrigger = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.SubTrigger>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubTrigger> & {

inset?: boolean

}

>(({ className, inset, children, ...props }, ref) => (

<DropdownMenuPrimitive.SubTrigger

ref={ref}

className={cn(

"flex cursor-default gap-2 select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent data-[state=open]:bg-accent [&\_svg]:pointer-events-none [&\_svg]:size-4 [&\_svg]:shrink-0",

inset && "pl-8",

className

)}

{...props}

>

{children}

<ChevronRight className="ml-auto" />

</DropdownMenuPrimitive.SubTrigger>

))

DropdownMenuSubTrigger.displayName =

DropdownMenuPrimitive.SubTrigger.displayName

const DropdownMenuSubContent = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.SubContent>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.SubContent>

>(({ className, ...props }, ref) => (

<DropdownMenuPrimitive.SubContent

ref={ref}

className={cn(

"z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-lg data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

))

DropdownMenuSubContent.displayName =

DropdownMenuPrimitive.SubContent.displayName

const DropdownMenuContent = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Content>

>(({ className, sideOffset = 4, ...props }, ref) => (

<DropdownMenuPrimitive.Portal>

<DropdownMenuPrimitive.Content

ref={ref}

sideOffset={sideOffset}

className={cn(

"z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

</DropdownMenuPrimitive.Portal>

))

DropdownMenuContent.displayName = DropdownMenuPrimitive.Content.displayName

const DropdownMenuItem = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Item>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Item> & {

inset?: boolean

}

>(({ className, inset, ...props }, ref) => (

<DropdownMenuPrimitive.Item

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center gap-2 rounded-sm px-2 py-1.5 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50 [&\_svg]:pointer-events-none [&\_svg]:size-4 [&\_svg]:shrink-0",

inset && "pl-8",

className

)}

{...props}

/>

))

DropdownMenuItem.displayName = DropdownMenuPrimitive.Item.displayName

const DropdownMenuCheckboxItem = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.CheckboxItem>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.CheckboxItem>

>(({ className, children, checked, ...props }, ref) => (

<DropdownMenuPrimitive.CheckboxItem

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

className

)}

checked={checked}

{...props}

>

<span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">

<DropdownMenuPrimitive.ItemIndicator>

<Check className="h-4 w-4" />

</DropdownMenuPrimitive.ItemIndicator>

</span>

{children}

</DropdownMenuPrimitive.CheckboxItem>

))

DropdownMenuCheckboxItem.displayName =

DropdownMenuPrimitive.CheckboxItem.displayName

const DropdownMenuRadioItem = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.RadioItem>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.RadioItem>

>(({ className, children, ...props }, ref) => (

<DropdownMenuPrimitive.RadioItem

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none transition-colors focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

className

)}

{...props}

>

<span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">

<DropdownMenuPrimitive.ItemIndicator>

<Circle className="h-2 w-2 fill-current" />

</DropdownMenuPrimitive.ItemIndicator>

</span>

{children}

</DropdownMenuPrimitive.RadioItem>

))

DropdownMenuRadioItem.displayName = DropdownMenuPrimitive.RadioItem.displayName

const DropdownMenuLabel = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Label>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Label> & {

inset?: boolean

}

>(({ className, inset, ...props }, ref) => (

<DropdownMenuPrimitive.Label

ref={ref}

className={cn(

"px-2 py-1.5 text-sm font-semibold",

inset && "pl-8",

className

)}

{...props}

/>

))

DropdownMenuLabel.displayName = DropdownMenuPrimitive.Label.displayName

const DropdownMenuSeparator = React.forwardRef<

React.ElementRef<typeof DropdownMenuPrimitive.Separator>,

React.ComponentPropsWithoutRef<typeof DropdownMenuPrimitive.Separator>

>(({ className, ...props }, ref) => (

<DropdownMenuPrimitive.Separator

ref={ref}

className={cn("-mx-1 my-1 h-px bg-muted", className)}

{...props}

/>

))

DropdownMenuSeparator.displayName = DropdownMenuPrimitive.Separator.displayName

const DropdownMenuShortcut = ({

className,

...props

}: React.HTMLAttributes<HTMLSpanElement>) => {

return (

<span

className={cn("ml-auto text-xs tracking-widest opacity-60", className)}

{...props}

/>

)

}

DropdownMenuShortcut.displayName = "DropdownMenuShortcut"

export {

DropdownMenu,

DropdownMenuTrigger,

DropdownMenuContent,

DropdownMenuItem,

DropdownMenuCheckboxItem,

DropdownMenuRadioItem,

DropdownMenuLabel,

DropdownMenuSeparator,

DropdownMenuShortcut,

DropdownMenuGroup,

DropdownMenuPortal,

DropdownMenuSub,

DropdownMenuSubContent,

DropdownMenuSubTrigger,

DropdownMenuRadioGroup,

}

**src/components/ui/form.tsx:**

"use client"

import \* as React from "react"

import \* as LabelPrimitive from "@radix-ui/react-label"

import { Slot } from "@radix-ui/react-slot"

import {

Controller,

FormProvider,

useFormContext,

type ControllerProps,

type FieldPath,

type FieldValues,

} from "react-hook-form"

import { cn } from "@/lib/utils"

import { Label } from "@/components/ui/label"

const Form = FormProvider

type FormFieldContextValue<

TFieldValues extends FieldValues = FieldValues,

TName extends FieldPath<TFieldValues> = FieldPath<TFieldValues>

> = {

name: TName

}

const FormFieldContext = React.createContext<FormFieldContextValue>(

{} as FormFieldContextValue

)

const FormField = <

TFieldValues extends FieldValues = FieldValues,

TName extends FieldPath<TFieldValues> = FieldPath<TFieldValues>

>({

...props

}: ControllerProps<TFieldValues, TName>) => {

return (

<FormFieldContext.Provider value={{ name: props.name }}>

<Controller {...props} />

</FormFieldContext.Provider>

)

}

const useFormField = () => {

const fieldContext = React.useContext(FormFieldContext)

const itemContext = React.useContext(FormItemContext)

const { getFieldState, formState } = useFormContext()

const fieldState = getFieldState(fieldContext.name, formState)

if (!fieldContext) {

throw new Error("useFormField should be used within <FormField>")

}

const { id } = itemContext

return {

id,

name: fieldContext.name,

formItemId: `${id}-form-item`,

formDescriptionId: `${id}-form-item-description`,

formMessageId: `${id}-form-item-message`,

...fieldState,

}

}

type FormItemContextValue = {

id: string

}

const FormItemContext = React.createContext<FormItemContextValue>(

{} as FormItemContextValue

)

const FormItem = React.forwardRef<

HTMLDivElement,

React.HTMLAttributes<HTMLDivElement>

>(({ className, ...props }, ref) => {

const id = React.useId()

return (

<FormItemContext.Provider value={{ id }}>

<div ref={ref} className={cn("space-y-2", className)} {...props} />

</FormItemContext.Provider>

)

})

FormItem.displayName = "FormItem"

const FormLabel = React.forwardRef<

React.ElementRef<typeof LabelPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root>

>(({ className, ...props }, ref) => {

const { error, formItemId } = useFormField()

return (

<Label

ref={ref}

className={cn(error && "text-destructive", className)}

htmlFor={formItemId}

{...props}

/>

)

})

FormLabel.displayName = "FormLabel"

const FormControl = React.forwardRef<

React.ElementRef<typeof Slot>,

React.ComponentPropsWithoutRef<typeof Slot>

>(({ ...props }, ref) => {

const { error, formItemId, formDescriptionId, formMessageId } = useFormField()

return (

<Slot

ref={ref}

id={formItemId}

aria-describedby={

!error

? `${formDescriptionId}`

: `${formDescriptionId} ${formMessageId}`

}

aria-invalid={!!error}

{...props}

/>

)

})

FormControl.displayName = "FormControl"

const FormDescription = React.forwardRef<

HTMLParagraphElement,

React.HTMLAttributes<HTMLParagraphElement>

>(({ className, ...props }, ref) => {

const { formDescriptionId } = useFormField()

return (

<p

ref={ref}

id={formDescriptionId}

className={cn("text-sm text-muted-foreground", className)}

{...props}

/>

)

})

FormDescription.displayName = "FormDescription"

const FormMessage = React.forwardRef<

HTMLParagraphElement,

React.HTMLAttributes<HTMLParagraphElement>

>(({ className, children, ...props }, ref) => {

const { error, formMessageId } = useFormField()

const body = error ? String(error?.message ?? "") : children

if (!body) {

return null

}

return (

<p

ref={ref}

id={formMessageId}

className={cn("text-sm font-medium text-destructive", className)}

{...props}

>

{body}

</p>

)

})

FormMessage.displayName = "FormMessage"

export {

useFormField,

Form,

FormItem,

FormLabel,

FormControl,

FormDescription,

FormMessage,

FormField,

}

**src/components/ui/input.tsx:**

import \* as React from "react"

import { cn } from "@/lib/utils"

const Input = React.forwardRef<HTMLInputElement, React.ComponentProps<"input">>(

({ className, type, ...props }, ref) => {

return (

<input

type={type}

className={cn(

"flex h-10 w-full rounded-md border border-input bg-background px-3 py-2 text-base ring-offset-background file:border-0 file:bg-transparent file:text-sm file:font-medium file:text-foreground placeholder:text-muted-foreground focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 md:text-sm",

className

)}

ref={ref}

{...props}

/>

)

}

)

Input.displayName = "Input"

export { Input }

**src/components/ui/label.tsx:**

"use client"

import \* as React from "react"

import \* as LabelPrimitive from "@radix-ui/react-label"

import { cva, type VariantProps } from "class-variance-authority"

import { cn } from "@/lib/utils"

const labelVariants = cva(

"text-sm font-medium leading-none peer-disabled:cursor-not-allowed peer-disabled:opacity-70"

)

const Label = React.forwardRef<

React.ElementRef<typeof LabelPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof LabelPrimitive.Root> &

VariantProps<typeof labelVariants>

>(({ className, ...props }, ref) => (

<LabelPrimitive.Root

ref={ref}

className={cn(labelVariants(), className)}

{...props}

/>

))

Label.displayName = LabelPrimitive.Root.displayName

export { Label }

**src/components/ui/menubar.tsx:**

"use client"

import \* as React from "react"

import \* as MenubarPrimitive from "@radix-ui/react-menubar"

import { Check, ChevronRight, Circle } from "lucide-react"

import { cn } from "@/lib/utils"

function MenubarMenu({

...props

}: React.ComponentProps<typeof MenubarPrimitive.Menu>) {

return <MenubarPrimitive.Menu {...props} />

}

function MenubarGroup({

...props

}: React.ComponentProps<typeof MenubarPrimitive.Group>) {

return <MenubarPrimitive.Group {...props} />

}

function MenubarPortal({

...props

}: React.ComponentProps<typeof MenubarPrimitive.Portal>) {

return <MenubarPrimitive.Portal {...props} />

}

function MenubarRadioGroup({

...props

}: React.ComponentProps<typeof MenubarPrimitive.RadioGroup>) {

return <MenubarPrimitive.RadioGroup {...props} />

}

function MenubarSub({

...props

}: React.ComponentProps<typeof MenubarPrimitive.Sub>) {

return <MenubarPrimitive.Sub data-slot="menubar-sub" {...props} />

}

const Menubar = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Root>

>(({ className, ...props }, ref) => (

<MenubarPrimitive.Root

ref={ref}

className={cn(

"flex h-10 items-center space-x-1 rounded-md border bg-background p-1",

className

)}

{...props}

/>

))

Menubar.displayName = MenubarPrimitive.Root.displayName

const MenubarTrigger = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.Trigger>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Trigger>

>(({ className, ...props }, ref) => (

<MenubarPrimitive.Trigger

ref={ref}

className={cn(

"flex cursor-default select-none items-center rounded-sm px-3 py-1.5 text-sm font-medium outline-none focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",

className

)}

{...props}

/>

))

MenubarTrigger.displayName = MenubarPrimitive.Trigger.displayName

const MenubarSubTrigger = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.SubTrigger>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubTrigger> & {

inset?: boolean

}

>(({ className, inset, children, ...props }, ref) => (

<MenubarPrimitive.SubTrigger

ref={ref}

className={cn(

"flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[state=open]:bg-accent data-[state=open]:text-accent-foreground",

inset && "pl-8",

className

)}

{...props}

>

{children}

<ChevronRight className="ml-auto h-4 w-4" />

</MenubarPrimitive.SubTrigger>

))

MenubarSubTrigger.displayName = MenubarPrimitive.SubTrigger.displayName

const MenubarSubContent = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.SubContent>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.SubContent>

>(({ className, ...props }, ref) => (

<MenubarPrimitive.SubContent

ref={ref}

className={cn(

"z-50 min-w-[8rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

))

MenubarSubContent.displayName = MenubarPrimitive.SubContent.displayName

const MenubarContent = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Content>

>(

(

{ className, align = "start", alignOffset = -4, sideOffset = 8, ...props },

ref

) => (

<MenubarPrimitive.Portal>

<MenubarPrimitive.Content

ref={ref}

align={align}

alignOffset={alignOffset}

sideOffset={sideOffset}

className={cn(

"z-50 min-w-[12rem] overflow-hidden rounded-md border bg-popover p-1 text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

</MenubarPrimitive.Portal>

)

)

MenubarContent.displayName = MenubarPrimitive.Content.displayName

const MenubarItem = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.Item>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Item> & {

inset?: boolean

}

>(({ className, inset, ...props }, ref) => (

<MenubarPrimitive.Item

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm px-2 py-1.5 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

inset && "pl-8",

className

)}

{...props}

/>

))

MenubarItem.displayName = MenubarPrimitive.Item.displayName

const MenubarCheckboxItem = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.CheckboxItem>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.CheckboxItem>

>(({ className, children, checked, ...props }, ref) => (

<MenubarPrimitive.CheckboxItem

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

className

)}

checked={checked}

{...props}

>

<span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">

<MenubarPrimitive.ItemIndicator>

<Check className="h-4 w-4" />

</MenubarPrimitive.ItemIndicator>

</span>

{children}

</MenubarPrimitive.CheckboxItem>

))

MenubarCheckboxItem.displayName = MenubarPrimitive.CheckboxItem.displayName

const MenubarRadioItem = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.RadioItem>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.RadioItem>

>(({ className, children, ...props }, ref) => (

<MenubarPrimitive.RadioItem

ref={ref}

className={cn(

"relative flex cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

className

)}

{...props}

>

<span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">

<MenubarPrimitive.ItemIndicator>

<Circle className="h-2 w-2 fill-current" />

</MenubarPrimitive.ItemIndicator>

</span>

{children}

</MenubarPrimitive.RadioItem>

))

MenubarRadioItem.displayName = MenubarPrimitive.RadioItem.displayName

const MenubarLabel = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.Label>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Label> & {

inset?: boolean

}

>(({ className, inset, ...props }, ref) => (

<MenubarPrimitive.Label

ref={ref}

className={cn(

"px-2 py-1.5 text-sm font-semibold",

inset && "pl-8",

className

)}

{...props}

/>

))

MenubarLabel.displayName = MenubarPrimitive.Label.displayName

const MenubarSeparator = React.forwardRef<

React.ElementRef<typeof MenubarPrimitive.Separator>,

React.ComponentPropsWithoutRef<typeof MenubarPrimitive.Separator>

>(({ className, ...props }, ref) => (

<MenubarPrimitive.Separator

ref={ref}

className={cn("-mx-1 my-1 h-px bg-muted", className)}

{...props}

/>

))

MenubarSeparator.displayName = MenubarPrimitive.Separator.displayName

const MenubarShortcut = ({

className,

...props

}: React.HTMLAttributes<HTMLSpanElement>) => {

return (

<span

className={cn(

"ml-auto text-xs tracking-widest text-muted-foreground",

className

)}

{...props}

/>

)

}

MenubarShortcut.displayname = "MenubarShortcut"

export {

Menubar,

MenubarMenu,

MenubarTrigger,

MenubarContent,

MenubarItem,

MenubarSeparator,

MenubarLabel,

MenubarCheckboxItem,

MenubarRadioGroup,

MenubarRadioItem,

MenubarPortal,

MenubarSubContent,

MenubarSubTrigger,

MenubarGroup,

MenubarSub,

MenubarShortcut,

}

**src/components/ui/popover.tsx:**

"use client"

import \* as React from "react"

import \* as PopoverPrimitive from "@radix-ui/react-popover"

import { cn } from "@/lib/utils"

const Popover = PopoverPrimitive.Root

const PopoverTrigger = PopoverPrimitive.Trigger

const PopoverContent = React.forwardRef<

React.ElementRef<typeof PopoverPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof PopoverPrimitive.Content>

>(({ className, align = "center", sideOffset = 4, ...props }, ref) => (

<PopoverPrimitive.Portal>

<PopoverPrimitive.Content

ref={ref}

align={align}

sideOffset={sideOffset}

className={cn(

"z-50 w-72 rounded-md border bg-popover p-4 text-popover-foreground shadow-md outline-none data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

</PopoverPrimitive.Portal>

))

PopoverContent.displayName = PopoverPrimitive.Content.displayName

export { Popover, PopoverTrigger, PopoverContent }

**src/components/ui/progress.tsx:**

"use client"

import \* as React from "react"

import \* as ProgressPrimitive from "@radix-ui/react-progress"

import { cn } from "@/lib/utils"

const Progress = React.forwardRef<

React.ElementRef<typeof ProgressPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof ProgressPrimitive.Root>

>(({ className, value, ...props }, ref) => (

<ProgressPrimitive.Root

ref={ref}

className={cn(

"relative h-4 w-full overflow-hidden rounded-full bg-secondary",

className

)}

{...props}

>

<ProgressPrimitive.Indicator

className="h-full w-full flex-1 bg-primary transition-all"

style={{ transform: `translateX(-${100 - (value || 0)}%)` }}

/>

</ProgressPrimitive.Root>

))

Progress.displayName = ProgressPrimitive.Root.displayName

export { Progress }

**src/components/ui/radio-group.tsx:**

"use client"

import \* as React from "react"

import \* as RadioGroupPrimitive from "@radix-ui/react-radio-group"

import { Circle } from "lucide-react"

import { cn } from "@/lib/utils"

const RadioGroup = React.forwardRef<

React.ElementRef<typeof RadioGroupPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Root>

>(({ className, ...props }, ref) => {

return (

<RadioGroupPrimitive.Root

className={cn("grid gap-2", className)}

{...props}

ref={ref}

/>

)

})

RadioGroup.displayName = RadioGroupPrimitive.Root.displayName

const RadioGroupItem = React.forwardRef<

React.ElementRef<typeof RadioGroupPrimitive.Item>,

React.ComponentPropsWithoutRef<typeof RadioGroupPrimitive.Item>

>(({ className, ...props }, ref) => {

return (

<RadioGroupPrimitive.Item

ref={ref}

className={cn(

"aspect-square h-4 w-4 rounded-full border border-primary text-primary ring-offset-background focus:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50",

className

)}

{...props}

>

<RadioGroupPrimitive.Indicator className="flex items-center justify-center">

<Circle className="h-2.5 w-2.5 fill-current text-current" />

</RadioGroupPrimitive.Indicator>

</RadioGroupPrimitive.Item>

)

})

RadioGroupItem.displayName = RadioGroupPrimitive.Item.displayName

export { RadioGroup, RadioGroupItem }

**src/components/ui/scroll-area.tsx:**

"use client"

import \* as React from "react"

import \* as ScrollAreaPrimitive from "@radix-ui/react-scroll-area"

import { cn } from "@/lib/utils"

const ScrollArea = React.forwardRef<

React.ElementRef<typeof ScrollAreaPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.Root>

>(({ className, children, ...props }, ref) => (

<ScrollAreaPrimitive.Root

ref={ref}

className={cn("relative overflow-hidden", className)}

{...props}

>

<ScrollAreaPrimitive.Viewport className="h-full w-full rounded-[inherit]">

{children}

</ScrollAreaPrimitive.Viewport>

<ScrollBar />

<ScrollAreaPrimitive.Corner />

</ScrollAreaPrimitive.Root>

))

ScrollArea.displayName = ScrollAreaPrimitive.Root.displayName

const ScrollBar = React.forwardRef<

React.ElementRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>,

React.ComponentPropsWithoutRef<typeof ScrollAreaPrimitive.ScrollAreaScrollbar>

>(({ className, orientation = "vertical", ...props }, ref) => (

<ScrollAreaPrimitive.ScrollAreaScrollbar

ref={ref}

orientation={orientation}

className={cn(

"flex touch-none select-none transition-colors",

orientation === "vertical" &&

"h-full w-2.5 border-l border-l-transparent p-[1px]",

orientation === "horizontal" &&

"h-2.5 flex-col border-t border-t-transparent p-[1px]",

className

)}

{...props}

>

<ScrollAreaPrimitive.ScrollAreaThumb className="relative flex-1 rounded-full bg-border" />

</ScrollAreaPrimitive.ScrollAreaScrollbar>

))

ScrollBar.displayName = ScrollAreaPrimitive.ScrollAreaScrollbar.displayName

export { ScrollArea, ScrollBar }

**src/components/ui/select.tsx:**

"use client"

import \* as React from "react"

import \* as SelectPrimitive from "@radix-ui/react-select"

import { Check, ChevronDown, ChevronUp } from "lucide-react"

import { cn } from "@/lib/utils"

const Select = SelectPrimitive.Root

const SelectGroup = SelectPrimitive.Group

const SelectValue = SelectPrimitive.Value

const SelectTrigger = React.forwardRef<

React.ElementRef<typeof SelectPrimitive.Trigger>,

React.ComponentPropsWithoutRef<typeof SelectPrimitive.Trigger>

>(({ className, children, ...props }, ref) => (

<SelectPrimitive.Trigger

ref={ref}

className={cn(

"flex h-10 w-full items-center justify-between rounded-md border border-input bg-background px-3 py-2 text-sm ring-offset-background placeholder:text-muted-foreground focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 [&>span]:line-clamp-1",

className

)}

{...props}

>

{children}

<SelectPrimitive.Icon asChild>

<ChevronDown className="h-4 w-4 opacity-50" />

</SelectPrimitive.Icon>

</SelectPrimitive.Trigger>

))

SelectTrigger.displayName = SelectPrimitive.Trigger.displayName

const SelectScrollUpButton = React.forwardRef<

React.ElementRef<typeof SelectPrimitive.ScrollUpButton>,

React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollUpButton>

>(({ className, ...props }, ref) => (

<SelectPrimitive.ScrollUpButton

ref={ref}

className={cn(

"flex cursor-default items-center justify-center py-1",

className

)}

{...props}

>

<ChevronUp className="h-4 w-4" />

</SelectPrimitive.ScrollUpButton>

))

SelectScrollUpButton.displayName = SelectPrimitive.ScrollUpButton.displayName

const SelectScrollDownButton = React.forwardRef<

React.ElementRef<typeof SelectPrimitive.ScrollDownButton>,

React.ComponentPropsWithoutRef<typeof SelectPrimitive.ScrollDownButton>

>(({ className, ...props }, ref) => (

<SelectPrimitive.ScrollDownButton

ref={ref}

className={cn(

"flex cursor-default items-center justify-center py-1",

className

)}

{...props}

>

<ChevronDown className="h-4 w-4" />

</SelectPrimitive.ScrollDownButton>

))

SelectScrollDownButton.displayName =

SelectPrimitive.ScrollDownButton.displayName

const SelectContent = React.forwardRef<

React.ElementRef<typeof SelectPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof SelectPrimitive.Content>

>(({ className, children, position = "popper", ...props }, ref) => (

<SelectPrimitive.Portal>

<SelectPrimitive.Content

ref={ref}

className={cn(

"relative z-50 max-h-96 min-w-[8rem] overflow-hidden rounded-md border bg-popover text-popover-foreground shadow-md data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0 data-[state=closed]:zoom-out-95 data-[state=open]:zoom-in-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

position === "popper" &&

"data-[side=bottom]:translate-y-1 data-[side=left]:-translate-x-1 data-[side=right]:translate-x-1 data-[side=top]:-translate-y-1",

className

)}

position={position}

{...props}

>

<SelectScrollUpButton />

<SelectPrimitive.Viewport

className={cn(

"p-1",

position === "popper" &&

"h-[var(--radix-select-trigger-height)] w-full min-w-[var(--radix-select-trigger-width)]"

)}

>

{children}

</SelectPrimitive.Viewport>

<SelectScrollDownButton />

</SelectPrimitive.Content>

</SelectPrimitive.Portal>

))

SelectContent.displayName = SelectPrimitive.Content.displayName

const SelectLabel = React.forwardRef<

React.ElementRef<typeof SelectPrimitive.Label>,

React.ComponentPropsWithoutRef<typeof SelectPrimitive.Label>

>(({ className, ...props }, ref) => (

<SelectPrimitive.Label

ref={ref}

className={cn("py-1.5 pl-8 pr-2 text-sm font-semibold", className)}

{...props}

/>

))

SelectLabel.displayName = SelectPrimitive.Label.displayName

const SelectItem = React.forwardRef<

React.ElementRef<typeof SelectPrimitive.Item>,

React.ComponentPropsWithoutRef<typeof SelectPrimitive.Item>

>(({ className, children, ...props }, ref) => (

<SelectPrimitive.Item

ref={ref}

className={cn(

"relative flex w-full cursor-default select-none items-center rounded-sm py-1.5 pl-8 pr-2 text-sm outline-none focus:bg-accent focus:text-accent-foreground data-[disabled]:pointer-events-none data-[disabled]:opacity-50",

className

)}

{...props}

>

<span className="absolute left-2 flex h-3.5 w-3.5 items-center justify-center">

<SelectPrimitive.ItemIndicator>

<Check className="h-4 w-4" />

</SelectPrimitive.ItemIndicator>

</span>

<SelectPrimitive.ItemText>{children}</SelectPrimitive.ItemText>

</SelectPrimitive.Item>

))

SelectItem.displayName = SelectPrimitive.Item.displayName

const SelectSeparator = React.forwardRef<

React.ElementRef<typeof SelectPrimitive.Separator>,

React.ComponentPropsWithoutRef<typeof SelectPrimitive.Separator>

>(({ className, ...props }, ref) => (

<SelectPrimitive.Separator

ref={ref}

className={cn("-mx-1 my-1 h-px bg-muted", className)}

{...props}

/>

))

SelectSeparator.displayName = SelectPrimitive.Separator.displayName

export {

Select,

SelectGroup,

SelectValue,

SelectTrigger,

SelectContent,

SelectLabel,

SelectItem,

SelectSeparator,

SelectScrollUpButton,

SelectScrollDownButton,

}

**src/components/ui/separator.tsx:**

"use client"

import \* as React from "react"

import \* as SeparatorPrimitive from "@radix-ui/react-separator"

import { cn } from "@/lib/utils"

const Separator = React.forwardRef<

React.ElementRef<typeof SeparatorPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof SeparatorPrimitive.Root>

>(

(

{ className, orientation = "horizontal", decorative = true, ...props },

ref

) => (

<SeparatorPrimitive.Root

ref={ref}

decorative={decorative}

orientation={orientation}

className={cn(

"shrink-0 bg-border",

orientation === "horizontal" ? "h-[1px] w-full" : "h-full w-[1px]",

className

)}

{...props}

/>

)

)

Separator.displayName = SeparatorPrimitive.Root.displayName

export { Separator }

**src/components/ui/sheet.tsx:**

"use client"

import \* as React from "react"

import \* as SheetPrimitive from "@radix-ui/react-dialog"

import { cva, type VariantProps } from "class-variance-authority"

import { X } from "lucide-react"

import { cn } from "@/lib/utils"

const Sheet = SheetPrimitive.Root

const SheetTrigger = SheetPrimitive.Trigger

const SheetClose = SheetPrimitive.Close

const SheetPortal = SheetPrimitive.Portal

const SheetOverlay = React.forwardRef<

React.ElementRef<typeof SheetPrimitive.Overlay>,

React.ComponentPropsWithoutRef<typeof SheetPrimitive.Overlay>

>(({ className, ...props }, ref) => (

<SheetPrimitive.Overlay

className={cn(

"fixed inset-0 z-50 bg-black/80 data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=open]:fade-in-0",

className

)}

{...props}

ref={ref}

/>

))

SheetOverlay.displayName = SheetPrimitive.Overlay.displayName

const sheetVariants = cva(

"fixed z-50 gap-4 bg-background p-6 shadow-lg transition ease-in-out data-[state=open]:animate-in data-[state=closed]:animate-out data-[state=closed]:duration-300 data-[state=open]:duration-500",

{

variants: {

side: {

top: "inset-x-0 top-0 border-b data-[state=closed]:slide-out-to-top data-[state=open]:slide-in-from-top",

bottom:

"inset-x-0 bottom-0 border-t data-[state=closed]:slide-out-to-bottom data-[state=open]:slide-in-from-bottom",

left: "inset-y-0 left-0 h-full w-3/4 border-r data-[state=closed]:slide-out-to-left data-[state=open]:slide-in-from-left sm:max-w-sm",

right:

"inset-y-0 right-0 h-full w-3/4 border-l data-[state=closed]:slide-out-to-right data-[state=open]:slide-in-from-right sm:max-w-sm",

},

},

defaultVariants: {

side: "right",

},

}

)

interface SheetContentProps

extends React.ComponentPropsWithoutRef<typeof SheetPrimitive.Content>,

VariantProps<typeof sheetVariants> {}

const SheetContent = React.forwardRef<

React.ElementRef<typeof SheetPrimitive.Content>,

SheetContentProps

>(({ side = "right", className, children, ...props }, ref) => (

<SheetPortal>

<SheetOverlay />

<SheetPrimitive.Content

ref={ref}

className={cn(sheetVariants({ side }), className)}

{...props}

>

{children}

<SheetPrimitive.Close className="absolute right-4 top-4 rounded-sm opacity-70 ring-offset-background transition-opacity hover:opacity-100 focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:pointer-events-none data-[state=open]:bg-secondary">

<X className="h-4 w-4" />

<span className="sr-only">Close</span>

</SheetPrimitive.Close>

</SheetPrimitive.Content>

</SheetPortal>

))

SheetContent.displayName = SheetPrimitive.Content.displayName

const SheetHeader = ({

className,

...props

}: React.HTMLAttributes<HTMLDivElement>) => (

<div

className={cn(

"flex flex-col space-y-2 text-center sm:text-left",

className

)}

{...props}

/>

)

SheetHeader.displayName = "SheetHeader"

const SheetFooter = ({

className,

...props

}: React.HTMLAttributes<HTMLDivElement>) => (

<div

className={cn(

"flex flex-col-reverse sm:flex-row sm:justify-end sm:space-x-2",

className

)}

{...props}

/>

)

SheetFooter.displayName = "SheetFooter"

const SheetTitle = React.forwardRef<

React.ElementRef<typeof SheetPrimitive.Title>,

React.ComponentPropsWithoutRef<typeof SheetPrimitive.Title>

>(({ className, ...props }, ref) => (

<SheetPrimitive.Title

ref={ref}

className={cn("text-lg font-semibold text-foreground", className)}

{...props}

/>

))

SheetTitle.displayName = SheetPrimitive.Title.displayName

const SheetDescription = React.forwardRef<

React.ElementRef<typeof SheetPrimitive.Description>,

React.ComponentPropsWithoutRef<typeof SheetPrimitive.Description>

>(({ className, ...props }, ref) => (

<SheetPrimitive.Description

ref={ref}

className={cn("text-sm text-muted-foreground", className)}

{...props}

/>

))

SheetDescription.displayName = SheetPrimitive.Description.displayName

export {

Sheet,

SheetPortal,

SheetOverlay,

SheetTrigger,

SheetClose,

SheetContent,

SheetHeader,

SheetFooter,

SheetTitle,

SheetDescription,

}

**src/components/ui/sidebar.tsx:**

"use client"

import \* as React from "react"

import { Slot } from "@radix-ui/react-slot"

import { VariantProps, cva } from "class-variance-authority"

import { PanelLeft } from "lucide-react"

import { useIsMobile } from "@/hooks/use-mobile"

import { cn } from "@/lib/utils"

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

import { Input } from "@/components/ui/input"

import { Separator } from "@/components/ui/separator"

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

import { Skeleton } from "@/components/ui/skeleton"

import {

Tooltip,

TooltipContent,

TooltipProvider,

TooltipTrigger,

} from "@/components/ui/tooltip"

const SIDEBAR\_COOKIE\_NAME = "sidebar\_state"

const SIDEBAR\_COOKIE\_MAX\_AGE = 60 \* 60 \* 24 \* 7

const SIDEBAR\_WIDTH = "16rem"

const SIDEBAR\_WIDTH\_MOBILE = "18rem"

const SIDEBAR\_WIDTH\_ICON = "3rem"

const SIDEBAR\_KEYBOARD\_SHORTCUT = "b"

type SidebarContext = {

state: "expanded" | "collapsed"

open: boolean

setOpen: (open: boolean) => void

openMobile: boolean

setOpenMobile: (open: boolean) => void

isMobile: boolean

toggleSidebar: () => void

}

const SidebarContext = React.createContext<SidebarContext | null>(null)

function useSidebar() {

const context = React.useContext(SidebarContext)

if (!context) {

throw new Error("useSidebar must be used within a SidebarProvider.")

}

return context

}

const SidebarProvider = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div"> & {

defaultOpen?: boolean

open?: boolean

onOpenChange?: (open: boolean) => void

}

>(

(

{

defaultOpen = true,

open: openProp,

onOpenChange: setOpenProp,

className,

style,

children,

...props

},

ref

) => {

const isMobile = useIsMobile()

const [openMobile, setOpenMobile] = React.useState(false)

// This is the internal state of the sidebar.

// We use openProp and setOpenProp for control from outside the component.

const [\_open, \_setOpen] = React.useState(defaultOpen)

const open = openProp ?? \_open

const setOpen = React.useCallback(

(value: boolean | ((value: boolean) => boolean)) => {

const openState = typeof value === "function" ? value(open) : value

if (setOpenProp) {

setOpenProp(openState)

} else {

\_setOpen(openState)

}

// This sets the cookie to keep the sidebar state.

document.cookie = `${SIDEBAR\_COOKIE\_NAME}=${openState}; path=/; max-age=${SIDEBAR\_COOKIE\_MAX\_AGE}`

},

[setOpenProp, open]

)

// Helper to toggle the sidebar.

const toggleSidebar = React.useCallback(() => {

return isMobile

? setOpenMobile((open) => !open)

: setOpen((open) => !open)

}, [isMobile, setOpen, setOpenMobile])

// Adds a keyboard shortcut to toggle the sidebar.

React.useEffect(() => {

const handleKeyDown = (event: KeyboardEvent) => {

if (

event.key === SIDEBAR\_KEYBOARD\_SHORTCUT &&

(event.metaKey || event.ctrlKey)

) {

event.preventDefault()

toggleSidebar()

}

}

window.addEventListener("keydown", handleKeyDown)

return () => window.removeEventListener("keydown", handleKeyDown)

}, [toggleSidebar])

// We add a state so that we can do data-state="expanded" or "collapsed".

// This makes it easier to style the sidebar with Tailwind classes.

const state = open ? "expanded" : "collapsed"

const contextValue = React.useMemo<SidebarContext>(

() => ({

state,

open,

setOpen,

isMobile,

openMobile,

setOpenMobile,

toggleSidebar,

}),

[state, open, setOpen, isMobile, openMobile, setOpenMobile, toggleSidebar]

)

return (

<SidebarContext.Provider value={contextValue}>

<TooltipProvider delayDuration={0}>

<div

style={

{

"--sidebar-width": SIDEBAR\_WIDTH,

"--sidebar-width-icon": SIDEBAR\_WIDTH\_ICON,

...style,

} as React.CSSProperties

}

className={cn(

"group/sidebar-wrapper flex min-h-svh w-full has-[[data-variant=inset]]:bg-sidebar",

className

)}

ref={ref}

{...props}

>

{children}

</div>

</TooltipProvider>

</SidebarContext.Provider>

)

}

)

SidebarProvider.displayName = "SidebarProvider"

const Sidebar = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div"> & {

side?: "left" | "right"

variant?: "sidebar" | "floating" | "inset"

collapsible?: "offcanvas" | "icon" | "none"

}

>(

(

{

side = "left",

variant = "sidebar",

collapsible = "offcanvas",

className,

children,

...props

},

ref

) => {

const { isMobile, state, openMobile, setOpenMobile } = useSidebar()

if (collapsible === "none") {

return (

<div

className={cn(

"flex h-full w-[--sidebar-width] flex-col bg-sidebar text-sidebar-foreground",

className

)}

ref={ref}

{...props}

>

{children}

</div>

)

}

if (isMobile) {

return (

<Sheet open={openMobile} onOpenChange={setOpenMobile} {...props}>

<SheetContent

data-sidebar="sidebar"

data-mobile="true"

className="w-[--sidebar-width] bg-sidebar p-0 text-sidebar-foreground [&>button]:hidden"

style={

{

"--sidebar-width": SIDEBAR\_WIDTH\_MOBILE,

} as React.CSSProperties

}

side={side}

>

<div className="flex h-full w-full flex-col">{children}</div>

</SheetContent>

</Sheet>

)

}

return (

<div

ref={ref}

className="group peer hidden md:block text-sidebar-foreground"

data-state={state}

data-collapsible={state === "collapsed" ? collapsible : ""}

data-variant={variant}

data-side={side}

>

{/\* This is what handles the sidebar gap on desktop \*/}

<div

className={cn(

"duration-200 relative h-svh w-[--sidebar-width] bg-transparent transition-[width] ease-linear",

"group-data-[collapsible=offcanvas]:w-0",

"group-data-[side=right]:rotate-180",

variant === "floating" || variant === "inset"

? "group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)\_+\_theme(spacing.4))]"

: "group-data-[collapsible=icon]:w-[--sidebar-width-icon]"

)}

/>

<div

className={cn(

"duration-200 fixed inset-y-0 z-10 hidden h-svh w-[--sidebar-width] transition-[left,right,width] ease-linear md:flex",

side === "left"

? "left-0 group-data-[collapsible=offcanvas]:left-[calc(var(--sidebar-width)\*-1)]"

: "right-0 group-data-[collapsible=offcanvas]:right-[calc(var(--sidebar-width)\*-1)]",

// Adjust the padding for floating and inset variants.

variant === "floating" || variant === "inset"

? "p-2 group-data-[collapsible=icon]:w-[calc(var(--sidebar-width-icon)\_+\_theme(spacing.4)\_+2px)]"

: "group-data-[collapsible=icon]:w-[--sidebar-width-icon] group-data-[side=left]:border-r group-data-[side=right]:border-l",

className

)}

{...props}

>

<div

data-sidebar="sidebar"

className="flex h-full w-full flex-col bg-sidebar group-data-[variant=floating]:rounded-lg group-data-[variant=floating]:border group-data-[variant=floating]:border-sidebar-border group-data-[variant=floating]:shadow"

>

{children}

</div>

</div>

</div>

)

}

)

Sidebar.displayName = "Sidebar"

const SidebarTrigger = React.forwardRef<

React.ElementRef<typeof Button>,

React.ComponentProps<typeof Button>

>(({ className, onClick, ...props }, ref) => {

const { toggleSidebar } = useSidebar()

return (

<Button

ref={ref}

data-sidebar="trigger"

variant="ghost"

size="icon"

className={cn("h-7 w-7", className)}

onClick={(event) => {

onClick?.(event)

toggleSidebar()

}}

{...props}

>

<PanelLeft />

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

</Button>

)

})

SidebarTrigger.displayName = "SidebarTrigger"

const SidebarRail = React.forwardRef<

HTMLButtonElement,

React.ComponentProps<"button">

>(({ className, ...props }, ref) => {

const { toggleSidebar } = useSidebar()

return (

<button

ref={ref}

data-sidebar="rail"

aria-label="Toggle Sidebar"

tabIndex={-1}

onClick={toggleSidebar}

title="Toggle Sidebar"

className={cn(

"absolute inset-y-0 z-20 hidden w-4 -translate-x-1/2 transition-all ease-linear after:absolute after:inset-y-0 after:left-1/2 after:w-[2px] hover:after:bg-sidebar-border group-data-[side=left]:-right-4 group-data-[side=right]:left-0 sm:flex",

"[[data-side=left]\_&]:cursor-w-resize [[data-side=right]\_&]:cursor-e-resize",

"[[data-side=left][data-state=collapsed]\_&]:cursor-e-resize [[data-side=right][data-state=collapsed]\_&]:cursor-w-resize",

"group-data-[collapsible=offcanvas]:translate-x-0 group-data-[collapsible=offcanvas]:after:left-full group-data-[collapsible=offcanvas]:hover:bg-sidebar",

"[[data-side=left][data-collapsible=offcanvas]\_&]:-right-2",

"[[data-side=right][data-collapsible=offcanvas]\_&]:-left-2",

className

)}

{...props}

/>

)

})

SidebarRail.displayName = "SidebarRail"

const SidebarInset = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"main">

>(({ className, ...props }, ref) => {

return (

<main

ref={ref}

className={cn(

"relative flex min-h-svh flex-1 flex-col bg-background",

"peer-data-[variant=inset]:min-h-[calc(100svh-theme(spacing.4))] md:peer-data-[variant=inset]:m-2 md:peer-data-[state=collapsed]:peer-data-[variant=inset]:ml-2 md:peer-data-[variant=inset]:ml-0 md:peer-data-[variant=inset]:rounded-xl md:peer-data-[variant=inset]:shadow",

className

)}

{...props}

/>

)

})

SidebarInset.displayName = "SidebarInset"

const SidebarInput = React.forwardRef<

React.ElementRef<typeof Input>,

React.ComponentProps<typeof Input>

>(({ className, ...props }, ref) => {

return (

<Input

ref={ref}

data-sidebar="input"

className={cn(

"h-8 w-full bg-background shadow-none focus-visible:ring-2 focus-visible:ring-sidebar-ring",

className

)}

{...props}

/>

)

})

SidebarInput.displayName = "SidebarInput"

const SidebarHeader = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div">

>(({ className, ...props }, ref) => {

return (

<div

ref={ref}

data-sidebar="header"

className={cn("flex flex-col gap-2 p-2", className)}

{...props}

/>

)

})

SidebarHeader.displayName = "SidebarHeader"

const SidebarFooter = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div">

>(({ className, ...props }, ref) => {

return (

<div

ref={ref}

data-sidebar="footer"

className={cn("flex flex-col gap-2 p-2", className)}

{...props}

/>

)

})

SidebarFooter.displayName = "SidebarFooter"

const SidebarSeparator = React.forwardRef<

React.ElementRef<typeof Separator>,

React.ComponentProps<typeof Separator>

>(({ className, ...props }, ref) => {

return (

<Separator

ref={ref}

data-sidebar="separator"

className={cn("mx-2 w-auto bg-sidebar-border", className)}

{...props}

/>

)

})

SidebarSeparator.displayName = "SidebarSeparator"

const SidebarContent = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div">

>(({ className, ...props }, ref) => {

return (

<div

ref={ref}

data-sidebar="content"

className={cn(

"flex min-h-0 flex-1 flex-col gap-2 overflow-auto group-data-[collapsible=icon]:overflow-hidden",

className

)}

{...props}

/>

)

})

SidebarContent.displayName = "SidebarContent"

const SidebarGroup = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div">

>(({ className, ...props }, ref) => {

return (

<div

ref={ref}

data-sidebar="group"

className={cn("relative flex w-full min-w-0 flex-col p-2", className)}

{...props}

/>

)

})

SidebarGroup.displayName = "SidebarGroup"

const SidebarGroupLabel = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div"> & { asChild?: boolean }

>(({ className, asChild = false, ...props }, ref) => {

const Comp = asChild ? Slot : "div"

return (

<Comp

ref={ref}

data-sidebar="group-label"

className={cn(

"duration-200 flex h-8 shrink-0 items-center rounded-md px-2 text-xs font-medium text-sidebar-foreground/70 outline-none ring-sidebar-ring transition-[margin,opa] ease-linear focus-visible:ring-2 [&>svg]:size-4 [&>svg]:shrink-0",

"group-data-[collapsible=icon]:-mt-8 group-data-[collapsible=icon]:opacity-0",

className

)}

{...props}

/>

)

})

SidebarGroupLabel.displayName = "SidebarGroupLabel"

const SidebarGroupAction = React.forwardRef<

HTMLButtonElement,

React.ComponentProps<"button"> & { asChild?: boolean }

>(({ className, asChild = false, ...props }, ref) => {

const Comp = asChild ? Slot : "button"

return (

<Comp

ref={ref}

data-sidebar="group-action"

className={cn(

"absolute right-3 top-3.5 flex aspect-square w-5 items-center justify-center rounded-md p-0 text-sidebar-foreground outline-none ring-sidebar-ring transition-transform hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 [&>svg]:size-4 [&>svg]:shrink-0",

// Increases the hit area of the button on mobile.

"after:absolute after:-inset-2 after:md:hidden",

"group-data-[collapsible=icon]:hidden",

className

)}

{...props}

/>

)

})

SidebarGroupAction.displayName = "SidebarGroupAction"

const SidebarGroupContent = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div">

>(({ className, ...props }, ref) => (

<div

ref={ref}

data-sidebar="group-content"

className={cn("w-full text-sm", className)}

{...props}

/>

))

SidebarGroupContent.displayName = "SidebarGroupContent"

const SidebarMenu = React.forwardRef<

HTMLUListElement,

React.ComponentProps<"ul">

>(({ className, ...props }, ref) => (

<ul

ref={ref}

data-sidebar="menu"

className={cn("flex w-full min-w-0 flex-col gap-1", className)}

{...props}

/>

))

SidebarMenu.displayName = "SidebarMenu"

const SidebarMenuItem = React.forwardRef<

HTMLLIElement,

React.ComponentProps<"li">

>(({ className, ...props }, ref) => (

<li

ref={ref}

data-sidebar="menu-item"

className={cn("group/menu-item relative", className)}

{...props}

/>

))

SidebarMenuItem.displayName = "SidebarMenuItem"

const sidebarMenuButtonVariants = cva(

"peer/menu-button flex w-full items-center gap-2 overflow-hidden rounded-md p-2 text-left text-sm outline-none ring-sidebar-ring transition-[width,height,padding] hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 active:bg-sidebar-accent active:text-sidebar-accent-foreground disabled:pointer-events-none disabled:opacity-50 group-has-[[data-sidebar=menu-action]]/menu-item:pr-8 aria-disabled:pointer-events-none aria-disabled:opacity-50 data-[active=true]:bg-sidebar-accent data-[active=true]:font-medium data-[active=true]:text-sidebar-accent-foreground data-[state=open]:hover:bg-sidebar-accent data-[state=open]:hover:text-sidebar-accent-foreground group-data-[collapsible=icon]:!size-8 group-data-[collapsible=icon]:!p-2 [&>span:last-child]:truncate [&>svg]:size-4 [&>svg]:shrink-0",

{

variants: {

variant: {

default: "hover:bg-sidebar-accent hover:text-sidebar-accent-foreground",

outline:

"bg-background shadow-[0\_0\_0\_1px\_hsl(var(--sidebar-border))] hover:bg-sidebar-accent hover:text-sidebar-accent-foreground hover:shadow-[0\_0\_0\_1px\_hsl(var(--sidebar-accent))]",

},

size: {

default: "h-8 text-sm",

sm: "h-7 text-xs",

lg: "h-12 text-sm group-data-[collapsible=icon]:!p-0",

},

},

defaultVariants: {

variant: "default",

size: "default",

},

}

)

const SidebarMenuButton = React.forwardRef<

HTMLButtonElement,

React.ComponentProps<"button"> & {

asChild?: boolean

isActive?: boolean

tooltip?: string | React.ComponentProps<typeof TooltipContent>

} & VariantProps<typeof sidebarMenuButtonVariants>

>(

(

{

asChild = false,

isActive = false,

variant = "default",

size = "default",

tooltip,

className,

...props

},

ref

) => {

const Comp = asChild ? Slot : "button"

const { isMobile, state } = useSidebar()

const button = (

<Comp

ref={ref}

data-sidebar="menu-button"

data-size={size}

data-active={isActive}

className={cn(sidebarMenuButtonVariants({ variant, size }), className)}

{...props}

/>

)

if (!tooltip) {

return button

}

if (typeof tooltip === "string") {

tooltip = {

children: tooltip,

}

}

return (

<Tooltip>

<TooltipTrigger asChild>{button}</TooltipTrigger>

<TooltipContent

side="right"

align="center"

hidden={state !== "collapsed" || isMobile}

{...tooltip}

/>

</Tooltip>

)

}

)

SidebarMenuButton.displayName = "SidebarMenuButton"

const SidebarMenuAction = React.forwardRef<

HTMLButtonElement,

React.ComponentProps<"button"> & {

asChild?: boolean

showOnHover?: boolean

}

>(({ className, asChild = false, showOnHover = false, ...props }, ref) => {

const Comp = asChild ? Slot : "button"

return (

<Comp

ref={ref}

data-sidebar="menu-action"

className={cn(

"absolute right-1 top-1.5 flex aspect-square w-5 items-center justify-center rounded-md p-0 text-sidebar-foreground outline-none ring-sidebar-ring transition-transform hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 peer-hover/menu-button:text-sidebar-accent-foreground [&>svg]:size-4 [&>svg]:shrink-0",

// Increases the hit area of the button on mobile.

"after:absolute after:-inset-2 after:md:hidden",

"peer-data-[size=sm]/menu-button:top-1",

"peer-data-[size=default]/menu-button:top-1.5",

"peer-data-[size=lg]/menu-button:top-2.5",

"group-data-[collapsible=icon]:hidden",

showOnHover &&

"group-focus-within/menu-item:opacity-100 group-hover/menu-item:opacity-100 data-[state=open]:opacity-100 peer-data-[active=true]/menu-button:text-sidebar-accent-foreground md:opacity-0",

className

)}

{...props}

/>

)

})

SidebarMenuAction.displayName = "SidebarMenuAction"

const SidebarMenuBadge = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div">

>(({ className, ...props }, ref) => (

<div

ref={ref}

data-sidebar="menu-badge"

className={cn(

"absolute right-1 flex h-5 min-w-5 items-center justify-center rounded-md px-1 text-xs font-medium tabular-nums text-sidebar-foreground select-none pointer-events-none",

"peer-hover/menu-button:text-sidebar-accent-foreground peer-data-[active=true]/menu-button:text-sidebar-accent-foreground",

"peer-data-[size=sm]/menu-button:top-1",

"peer-data-[size=default]/menu-button:top-1.5",

"peer-data-[size=lg]/menu-button:top-2.5",

"group-data-[collapsible=icon]:hidden",

className

)}

{...props}

/>

))

SidebarMenuBadge.displayName = "SidebarMenuBadge"

const SidebarMenuSkeleton = React.forwardRef<

HTMLDivElement,

React.ComponentProps<"div"> & {

showIcon?: boolean

}

>(({ className, showIcon = false, ...props }, ref) => {

// Random width between 50 to 90%.

const width = React.useMemo(() => {

return `${Math.floor(Math.random() \* 40) + 50}%`

}, [])

return (

<div

ref={ref}

data-sidebar="menu-skeleton"

className={cn("rounded-md h-8 flex gap-2 px-2 items-center", className)}

{...props}

>

{showIcon && (

<Skeleton

className="size-4 rounded-md"

data-sidebar="menu-skeleton-icon"

/>

)}

<Skeleton

className="h-4 flex-1 max-w-[--skeleton-width]"

data-sidebar="menu-skeleton-text"

style={

{

"--skeleton-width": width,

} as React.CSSProperties

}

/>

</div>

)

})

SidebarMenuSkeleton.displayName = "SidebarMenuSkeleton"

const SidebarMenuSub = React.forwardRef<

HTMLUListElement,

React.ComponentProps<"ul">

>(({ className, ...props }, ref) => (

<ul

ref={ref}

data-sidebar="menu-sub"

className={cn(

"mx-3.5 flex min-w-0 translate-x-px flex-col gap-1 border-l border-sidebar-border px-2.5 py-0.5",

"group-data-[collapsible=icon]:hidden",

className

)}

{...props}

/>

))

SidebarMenuSub.displayName = "SidebarMenuSub"

const SidebarMenuSubItem = React.forwardRef<

HTMLLIElement,

React.ComponentProps<"li">

>(({ ...props }, ref) => <li ref={ref} {...props} />)

SidebarMenuSubItem.displayName = "SidebarMenuSubItem"

const SidebarMenuSubButton = React.forwardRef<

HTMLAnchorElement,

React.ComponentProps<"a"> & {

asChild?: boolean

size?: "sm" | "md"

isActive?: boolean

}

>(({ asChild = false, size = "md", isActive, className, ...props }, ref) => {

const Comp = asChild ? Slot : "a"

return (

<Comp

ref={ref}

data-sidebar="menu-sub-button"

data-size={size}

data-active={isActive}

className={cn(

"flex h-7 min-w-0 -translate-x-px items-center gap-2 overflow-hidden rounded-md px-2 text-sidebar-foreground outline-none ring-sidebar-ring hover:bg-sidebar-accent hover:text-sidebar-accent-foreground focus-visible:ring-2 active:bg-sidebar-accent active:text-sidebar-accent-foreground disabled:pointer-events-none disabled:opacity-50 aria-disabled:pointer-events-none aria-disabled:opacity-50 [&>span:last-child]:truncate [&>svg]:size-4 [&>svg]:shrink-0 [&>svg]:text-sidebar-accent-foreground",

"data-[active=true]:bg-sidebar-accent data-[active=true]:text-sidebar-accent-foreground",

size === "sm" && "text-xs",

size === "md" && "text-sm",

"group-data-[collapsible=icon]:hidden",

className

)}

{...props}

/>

)

})

SidebarMenuSubButton.displayName = "SidebarMenuSubButton"

export {

Sidebar,

SidebarContent,

SidebarFooter,

SidebarGroup,

SidebarGroupAction,

SidebarGroupContent,

SidebarGroupLabel,

SidebarHeader,

SidebarInput,

SidebarInset,

SidebarMenu,

SidebarMenuAction,

SidebarMenuBadge,

SidebarMenuButton,

SidebarMenuItem,

SidebarMenuSkeleton,

SidebarMenuSub,

SidebarMenuSubButton,

SidebarMenuSubItem,

SidebarProvider,

SidebarRail,

SidebarSeparator,

SidebarTrigger,

useSidebar,

}

**src/components/ui/skeleton.tsx:**

import { cn } from "@/lib/utils"

function Skeleton({

className,

...props

}: React.HTMLAttributes<HTMLDivElement>) {

return (

<div

className={cn("animate-pulse rounded-md bg-muted", className)}

{...props}

/>

)

}

export { Skeleton }

**src/components/ui/slider.tsx:**

"use client"

import \* as React from "react"

import \* as SliderPrimitive from "@radix-ui/react-slider"

import { cn } from "@/lib/utils"

const Slider = React.forwardRef<

React.ElementRef<typeof SliderPrimitive.Root>,

React.ComponentPropsWithoutRef<typeof SliderPrimitive.Root>

>(({ className, ...props }, ref) => (

<SliderPrimitive.Root

ref={ref}

className={cn(

"relative flex w-full touch-none select-none items-center",

className

)}

{...props}

>

<SliderPrimitive.Track className="relative h-2 w-full grow overflow-hidden rounded-full bg-secondary">

<SliderPrimitive.Range className="absolute h-full bg-primary" />

</SliderPrimitive.Track>

<SliderPrimitive.Thumb className="block h-5 w-5 rounded-full border-2 border-primary bg-background ring-offset-background transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50" />

</SliderPrimitive.Root>

))

Slider.displayName = SliderPrimitive.Root.displayName

export { Slider }

**src/components/ui/switch.tsx:**

"use client"

import \* as React from "react"

import \* as SwitchPrimitives from "@radix-ui/react-switch"

import { cn } from "@/lib/utils"

const Switch = React.forwardRef<

React.ElementRef<typeof SwitchPrimitives.Root>,

React.ComponentPropsWithoutRef<typeof SwitchPrimitives.Root>

>(({ className, ...props }, ref) => (

<SwitchPrimitives.Root

className={cn(

"peer inline-flex h-6 w-11 shrink-0 cursor-pointer items-center rounded-full border-2 border-transparent transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 focus-visible:ring-offset-background disabled:cursor-not-allowed disabled:opacity-50 data-[state=checked]:bg-primary data-[state=unchecked]:bg-input",

className

)}

{...props}

ref={ref}

>

<SwitchPrimitives.Thumb

className={cn(

"pointer-events-none block h-5 w-5 rounded-full bg-background shadow-lg ring-0 transition-transform data-[state=checked]:translate-x-5 data-[state=unchecked]:translate-x-0"

)}

/>

</SwitchPrimitives.Root>

))

Switch.displayName = SwitchPrimitives.Root.displayName

export { Switch }

**src/components/ui/table.tsx:**

import \* as React from "react"

import { cn } from "@/lib/utils"

const Table = React.forwardRef<

HTMLTableElement,

React.HTMLAttributes<HTMLTableElement>

>(({ className, ...props }, ref) => (

<div className="relative w-full overflow-auto">

<table

ref={ref}

className={cn("w-full caption-bottom text-sm", className)}

{...props}

/>

</div>

))

Table.displayName = "Table"

const TableHeader = React.forwardRef<

HTMLTableSectionElement,

React.HTMLAttributes<HTMLTableSectionElement>

>(({ className, ...props }, ref) => (

<thead ref={ref} className={cn("[&\_tr]:border-b", className)} {...props} />

))

TableHeader.displayName = "TableHeader"

const TableBody = React.forwardRef<

HTMLTableSectionElement,

React.HTMLAttributes<HTMLTableSectionElement>

>(({ className, ...props }, ref) => (

<tbody

ref={ref}

className={cn("[&\_tr:last-child]:border-0", className)}

{...props}

/>

))

TableBody.displayName = "TableBody"

const TableFooter = React.forwardRef<

HTMLTableSectionElement,

React.HTMLAttributes<HTMLTableSectionElement>

>(({ className, ...props }, ref) => (

<tfoot

ref={ref}

className={cn(

"border-t bg-muted/50 font-medium [&>tr]:last:border-b-0",

className

)}

{...props}

/>

))

TableFooter.displayName = "TableFooter"

const TableRow = React.forwardRef<

HTMLTableRowElement,

React.HTMLAttributes<HTMLTableRowElement>

>(({ className, ...props }, ref) => (

<tr

ref={ref}

className={cn(

"border-b transition-colors hover:bg-muted/50 data-[state=selected]:bg-muted",

className

)}

{...props}

/>

))

TableRow.displayName = "TableRow"

const TableHead = React.forwardRef<

HTMLTableCellElement,

React.ThHTMLAttributes<HTMLTableCellElement>

>(({ className, ...props }, ref) => (

<th

ref={ref}

className={cn(

"h-12 px-4 text-left align-middle font-medium text-muted-foreground [&:has([role=checkbox])]:pr-0",

className

)}

{...props}

/>

))

TableHead.displayName = "TableHead"

const TableCell = React.forwardRef<

HTMLTableCellElement,

React.TdHTMLAttributes<HTMLTableCellElement>

>(({ className, ...props }, ref) => (

<td

ref={ref}

className={cn("p-4 align-middle [&:has([role=checkbox])]:pr-0", className)}

{...props}

/>

))

TableCell.displayName = "TableCell"

const TableCaption = React.forwardRef<

HTMLTableCaptionElement,

React.HTMLAttributes<HTMLTableCaptionElement>

>(({ className, ...props }, ref) => (

<caption

ref={ref}

className={cn("mt-4 text-sm text-muted-foreground", className)}

{...props}

/>

))

TableCaption.displayName = "TableCaption"

export {

Table,

TableHeader,

TableBody,

TableFooter,

TableHead,

TableRow,

TableCell,

TableCaption,

}

**src/components/ui/tabs.tsx:**

"use client"

import \* as React from "react"

import \* as TabsPrimitive from "@radix-ui/react-tabs"

import { cn } from "@/lib/utils"

const Tabs = TabsPrimitive.Root

const TabsList = React.forwardRef<

React.ElementRef<typeof TabsPrimitive.List>,

React.ComponentPropsWithoutRef<typeof TabsPrimitive.List>

>(({ className, ...props }, ref) => (

<TabsPrimitive.List

ref={ref}

className={cn(

"inline-flex h-10 items-center justify-center rounded-md bg-muted p-1 text-muted-foreground",

className

)}

{...props}

/>

))

TabsList.displayName = TabsPrimitive.List.displayName

const TabsTrigger = React.forwardRef<

React.ElementRef<typeof TabsPrimitive.Trigger>,

React.ComponentPropsWithoutRef<typeof TabsPrimitive.Trigger>

>(({ className, ...props }, ref) => (

<TabsPrimitive.Trigger

ref={ref}

className={cn(

"inline-flex items-center justify-center whitespace-nowrap rounded-sm px-3 py-1.5 text-sm font-medium ring-offset-background transition-all focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 data-[state=active]:bg-background data-[state=active]:text-foreground data-[state=active]:shadow-sm",

className

)}

{...props}

/>

))

TabsTrigger.displayName = TabsPrimitive.Trigger.displayName

const TabsContent = React.forwardRef<

React.ElementRef<typeof TabsPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof TabsPrimitive.Content>

>(({ className, ...props }, ref) => (

<TabsPrimitive.Content

ref={ref}

className={cn(

"mt-2 ring-offset-background focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2",

className

)}

{...props}

/>

))

TabsContent.displayName = TabsPrimitive.Content.displayName

export { Tabs, TabsList, TabsTrigger, TabsContent }

**src/components/ui/textarea.tsx:**

import \* as React from 'react';

import {cn} from '@/lib/utils';

const Textarea = React.forwardRef<HTMLTextAreaElement, React.ComponentProps<'textarea'>>(

({className, ...props}, ref) => {

return (

<textarea

className={cn(

'flex min-h-[80px] w-full rounded-md border border-input bg-background px-3 py-2 text-base ring-offset-background placeholder:text-muted-foreground focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50 md:text-sm',

className

)}

ref={ref}

{...props}

/>

);

}

);

Textarea.displayName = 'Textarea';

export {Textarea};

**src/components/ui/toast.tsx:**

"use client"

import \* as React from "react"

import \* as ToastPrimitives from "@radix-ui/react-toast"

import { cva, type VariantProps } from "class-variance-authority"

import { X } from "lucide-react"

import { cn } from "@/lib/utils"

const ToastProvider = ToastPrimitives.Provider

const ToastViewport = React.forwardRef<

React.ElementRef<typeof ToastPrimitives.Viewport>,

React.ComponentPropsWithoutRef<typeof ToastPrimitives.Viewport>

>(({ className, ...props }, ref) => (

<ToastPrimitives.Viewport

ref={ref}

className={cn(

"fixed top-0 z-[100] flex max-h-screen w-full flex-col-reverse p-4 sm:bottom-0 sm:right-0 sm:top-auto sm:flex-col md:max-w-[420px]",

className

)}

{...props}

/>

))

ToastViewport.displayName = ToastPrimitives.Viewport.displayName

const toastVariants = cva(

"group pointer-events-auto relative flex w-full items-center justify-between space-x-4 overflow-hidden rounded-md border p-6 pr-8 shadow-lg transition-all data-[swipe=cancel]:translate-x-0 data-[swipe=end]:translate-x-[var(--radix-toast-swipe-end-x)] data-[swipe=move]:translate-x-[var(--radix-toast-swipe-move-x)] data-[swipe=move]:transition-none data-[state=open]:animate-in data-[state=closed]:animate-out data-[swipe=end]:animate-out data-[state=closed]:fade-out-80 data-[state=closed]:slide-out-to-right-full data-[state=open]:slide-in-from-top-full data-[state=open]:sm:slide-in-from-bottom-full",

{

variants: {

variant: {

default: "border bg-background text-foreground",

destructive:

"destructive group border-destructive bg-destructive text-destructive-foreground",

},

},

defaultVariants: {

variant: "default",

},

}

)

const Toast = React.forwardRef<

React.ElementRef<typeof ToastPrimitives.Root>,

React.ComponentPropsWithoutRef<typeof ToastPrimitives.Root> &

VariantProps<typeof toastVariants>

>(({ className, variant, ...props }, ref) => {

return (

<ToastPrimitives.Root

ref={ref}

className={cn(toastVariants({ variant }), className)}

{...props}

/>

)

})

Toast.displayName = ToastPrimitives.Root.displayName

const ToastAction = React.forwardRef<

React.ElementRef<typeof ToastPrimitives.Action>,

React.ComponentPropsWithoutRef<typeof ToastPrimitives.Action>

>(({ className, ...props }, ref) => (

<ToastPrimitives.Action

ref={ref}

className={cn(

"inline-flex h-8 shrink-0 items-center justify-center rounded-md border bg-transparent px-3 text-sm font-medium ring-offset-background transition-colors hover:bg-secondary focus:outline-none focus:ring-2 focus:ring-ring focus:ring-offset-2 disabled:pointer-events-none disabled:opacity-50 group-[.destructive]:border-muted/40 group-[.destructive]:hover:border-destructive/30 group-[.destructive]:hover:bg-destructive group-[.destructive]:hover:text-destructive-foreground group-[.destructive]:focus:ring-destructive",

className

)}

{...props}

/>

))

ToastAction.displayName = ToastPrimitives.Action.displayName

const ToastClose = React.forwardRef<

React.ElementRef<typeof ToastPrimitives.Close>,

React.ComponentPropsWithoutRef<typeof ToastPrimitives.Close>

>(({ className, ...props }, ref) => (

<ToastPrimitives.Close

ref={ref}

className={cn(

"absolute right-2 top-2 rounded-md p-1 text-foreground/50 opacity-0 transition-opacity hover:text-foreground focus:opacity-100 focus:outline-none focus:ring-2 group-hover:opacity-100 group-[.destructive]:text-red-300 group-[.destructive]:hover:text-red-50 group-[.destructive]:focus:ring-red-400 group-[.destructive]:focus:ring-offset-red-600",

className

)}

toast-close=""

{...props}

>

<X className="h-4 w-4" />

</ToastPrimitives.Close>

))

ToastClose.displayName = ToastPrimitives.Close.displayName

const ToastTitle = React.forwardRef<

React.ElementRef<typeof ToastPrimitives.Title>,

React.ComponentPropsWithoutRef<typeof ToastPrimitives.Title>

>(({ className, ...props }, ref) => (

<ToastPrimitives.Title

ref={ref}

className={cn("text-sm font-semibold", className)}

{...props}

/>

))

ToastTitle.displayName = ToastPrimitives.Title.displayName

const ToastDescription = React.forwardRef<

React.ElementRef<typeof ToastPrimitives.Description>,

React.ComponentPropsWithoutRef<typeof ToastPrimitives.Description>

>(({ className, ...props }, ref) => (

<ToastPrimitives.Description

ref={ref}

className={cn("text-sm opacity-90", className)}

{...props}

/>

))

ToastDescription.displayName = ToastPrimitives.Description.displayName

type ToastProps = React.ComponentPropsWithoutRef<typeof Toast>

type ToastActionElement = React.ReactElement<typeof ToastAction>

export {

type ToastProps,

type ToastActionElement,

ToastProvider,

ToastViewport,

Toast,

ToastTitle,

ToastDescription,

ToastClose,

ToastAction,

}

**src/components/ui/toaster.tsx:**

"use client"

import { useToast } from "@/hooks/use-toast"

import {

Toast,

ToastClose,

ToastDescription,

ToastProvider,

ToastTitle,

ToastViewport,

} from "@/components/ui/toast"

export function Toaster() {

const { toasts } = useToast()

return (

<ToastProvider>

{toasts.map(function ({ id, title, description, action, ...props }) {

return (

<Toast key={id} {...props}>

<div className="grid gap-1">

{title && <ToastTitle>{title}</ToastTitle>}

{description && (

<ToastDescription>{description}</ToastDescription>

)}

</div>

{action}

<ToastClose />

</Toast>

)

})}

<ToastViewport />

</ToastProvider>

)

}

**src/components/ui/tooltip.tsx:**

"use client"

import \* as React from "react"

import \* as TooltipPrimitive from "@radix-ui/react-tooltip"

import { cn } from "@/lib/utils"

const TooltipProvider = TooltipPrimitive.Provider

const Tooltip = TooltipPrimitive.Root

const TooltipTrigger = TooltipPrimitive.Trigger

const TooltipContent = React.forwardRef<

React.ElementRef<typeof TooltipPrimitive.Content>,

React.ComponentPropsWithoutRef<typeof TooltipPrimitive.Content>

>(({ className, sideOffset = 4, ...props }, ref) => (

<TooltipPrimitive.Content

ref={ref}

sideOffset={sideOffset}

className={cn(

"z-50 overflow-hidden rounded-md border bg-popover px-3 py-1.5 text-sm text-popover-foreground shadow-md animate-in fade-in-0 zoom-in-95 data-[state=closed]:animate-out data-[state=closed]:fade-out-0 data-[state=closed]:zoom-out-95 data-[side=bottom]:slide-in-from-top-2 data-[side=left]:slide-in-from-right-2 data-[side=right]:slide-in-from-left-2 data-[side=top]:slide-in-from-bottom-2",

className

)}

{...props}

/>

))

TooltipContent.displayName = TooltipPrimitive.Content.displayName

export { Tooltip, TooltipTrigger, TooltipContent, TooltipProvider }

**src/hooks/use-mobile.tsx:**

import \* as React from "react"

const MOBILE\_BREAKPOINT = 768

export function useIsMobile() {

const [isMobile, setIsMobile] = React.useState<boolean>(false);

const [hasMounted, setHasMounted] = React.useState(false);

React.useEffect(() => {

setHasMounted(true);

const mql = window.matchMedia(`(max-width: ${MOBILE\_BREAKPOINT - 1}px)`);

const onChange = () => {

setIsMobile(mql.matches);

};

setIsMobile(mql.matches);

mql.addEventListener("change", onChange);

return () => mql.removeEventListener("change", onChange);

}, []);

if (!hasMounted) {

return false;

}

return isMobile;

}

**src/hooks/use-toast.ts:**

"use client"

// Inspired by react-hot-toast library

import \* as React from "react"

import type {

ToastActionElement,

ToastProps,

} from "@/components/ui/toast"

const TOAST\_LIMIT = 1

const TOAST\_REMOVE\_DELAY = 1000000

type ToasterToast = ToastProps & {

id: string

title?: React.ReactNode

description?: React.ReactNode

action?: ToastActionElement

}

const actionTypes = {

ADD\_TOAST: "ADD\_TOAST",

UPDATE\_TOAST: "UPDATE\_TOAST",

DISMISS\_TOAST: "DISMISS\_TOAST",

REMOVE\_TOAST: "REMOVE\_TOAST",

} as const

let count = 0

function genId() {

count = (count + 1) % Number.MAX\_SAFE\_INTEGER

return count.toString()

}

type ActionType = typeof actionTypes

type Action =

| {

type: ActionType["ADD\_TOAST"]

toast: ToasterToast

}

| {

type: ActionType["UPDATE\_TOAST"]

toast: Partial<ToasterToast>

}

| {

type: ActionType["DISMISS\_TOAST"]

toastId?: ToasterToast["id"]

}

| {

type: ActionType["REMOVE\_TOAST"]

toastId?: ToasterToast["id"]

}

interface State {

toasts: ToasterToast[]

}

const toastTimeouts = new Map<string, ReturnType<typeof setTimeout>>()

const addToRemoveQueue = (toastId: string) => {

if (toastTimeouts.has(toastId)) {

return

}

const timeout = setTimeout(() => {

toastTimeouts.delete(toastId)

dispatch({

type: "REMOVE\_TOAST",

toastId: toastId,

})

}, TOAST\_REMOVE\_DELAY)

toastTimeouts.set(toastId, timeout)

}

export const reducer = (state: State, action: Action): State => {

switch (action.type) {

case "ADD\_TOAST":

return {

...state,

toasts: [action.toast, ...state.toasts].slice(0, TOAST\_LIMIT),

}

case "UPDATE\_TOAST":

return {

...state,

toasts: state.toasts.map((t) =>

t.id === action.toast.id ? { ...t, ...action.toast } : t

),

}

case "DISMISS\_TOAST": {

const { toastId } = action

// ! Side effects ! - This could be extracted into a dismissToast() action,

// but I'll keep it here for simplicity

if (toastId) {

addToRemoveQueue(toastId)

} else {

state.toasts.forEach((toast) => {

addToRemoveQueue(toast.id)

})

}

return {

...state,

toasts: state.toasts.map((t) =>

t.id === toastId || toastId === undefined

? {

...t,

open: false,

}

: t

),

}

}

case "REMOVE\_TOAST":

if (action.toastId === undefined) {

return {

...state,

toasts: [],

}

}

return {

...state,

toasts: state.toasts.filter((t) => t.id !== action.toastId),

}

}

}

const listeners: Array<(state: State) => void> = []

let memoryState: State = { toasts: [] }

function dispatch(action: Action) {

memoryState = reducer(memoryState, action)

listeners.forEach((listener) => {

listener(memoryState)

})

}

type Toast = Omit<ToasterToast, "id">

function toast({ ...props }: Toast) {

const id = genId()

const update = (props: ToasterToast) =>

dispatch({

type: "UPDATE\_TOAST",

toast: { ...props, id },

})

const dismiss = () => dispatch({ type: "DISMISS\_TOAST", toastId: id })

dispatch({

type: "ADD\_TOAST",

toast: {

...props,

id,

open: true,

onOpenChange: (open) => {

if (!open) dismiss()

},

},

})

return {

id: id,

dismiss,

update,

}

}

function useToast() {

const [state, setState] = React.useState<State>(memoryState)

React.useEffect(() => {

listeners.push(setState)

return () => {

const index = listeners.indexOf(setState)

if (index > -1) {

listeners.splice(index, 1)

}

}

}, [state])

return {

...state,

toast,

dismiss: (toastId?: string) => dispatch({ type: "DISMISS\_TOAST", toastId }),

}

}

export { useToast, toast }

**src/lib/localStorage.ts:**

export function getItem<T>(key: string): T | null {

if (typeof window === 'undefined') return null;

try {

const item = window.localStorage.getItem(key);

return item ? (JSON.parse(item) as T) : null;

} catch (error) {

console.error('Error reading from localStorage', key, error);

return null;

}

}

export function setItem<T>(key: string, value: T): void {

if (typeof window === 'undefined') return;

try {

window.localStorage.setItem(key, JSON.stringify(value));

} catch (error) {

console.error('Error writing to localStorage', key, error);

}

}

export function removeItem(key: string): void {

if (typeof window === 'undefined') return;

try {

window.localStorage.removeItem(key);

} catch (error) {

console.error('Error removing from localStorage', key, error);

}

}

**src/lib/utils.ts:**

import { clsx, type ClassValue } from "clsx"

import { twMerge } from "tailwind-merge"

export function cn(...inputs: ClassValue[]) {

return twMerge(clsx(inputs))

}

**src/types/index.ts:**

export type HandUsage = 'left\_hand' | 'right\_hand' | 'both\_hands';

export interface SignStep {

sequenceNumber: number;

word?: string; // The original word/phrase this sign corresponds to

imageDataUri: string;

handUsage: HandUsage;

textualGuide: string;

}

export interface SavedTranslation {

id: string;

type: 'text-to-sign' | 'sign-to-text';

input: string; // For text-to-sign, this is text. For sign-to-text, this is the image data URI.

output: string | SignStep[]; // Updated to support SignStep array for text-to-sign

timestamp: number;

inputTextPreview?: string;

outputTextPreview?: string;

}

**tailwind.config.ts:**

import type { Config } from "tailwindcss";

export default {

darkMode: ["class"],

content: [

"./src/pages/\*\*/\*.{js,ts,jsx,tsx,mdx}",

"./src/components/\*\*/\*.{js,ts,jsx,tsx,mdx}",

"./src/app/\*\*/\*.{js,ts,jsx,tsx,mdx}",

],

theme: {

extend: {

colors: {

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

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

card: {

DEFAULT: 'hsl(var(--card))',

foreground: 'hsl(var(--card-foreground))'

},

popover: {

DEFAULT: 'hsl(var(--popover))',

foreground: 'hsl(var(--popover-foreground))'

},

primary: {

DEFAULT: 'hsl(var(--primary))',

foreground: 'hsl(var(--primary-foreground))'

},

secondary: {

DEFAULT: 'hsl(var(--secondary))',

foreground: 'hsl(var(--secondary-foreground))'

},

muted: {

DEFAULT: 'hsl(var(--muted))',

foreground: 'hsl(var(--muted-foreground))'

},

accent: {

DEFAULT: 'hsl(var(--accent))',

foreground: 'hsl(var(--accent-foreground))'

},

destructive: {

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

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

},

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

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

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

chart: {

'1': 'hsl(var(--chart-1))',

'2': 'hsl(var(--chart-2))',

'3': 'hsl(var(--chart-3))',

'4': 'hsl(var(--chart-4))',

'5': 'hsl(var(--chart-5))'

},

sidebar: {

DEFAULT: 'hsl(var(--sidebar-background))',

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

primary: 'hsl(var(--sidebar-primary))',

'primary-foreground': 'hsl(var(--sidebar-primary-foreground))',

accent: 'hsl(var(--sidebar-accent))',

'accent-foreground': 'hsl(var(--sidebar-accent-foreground))',

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

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

}

},

borderRadius: {

lg: 'var(--radius)',

md: 'calc(var(--radius) - 2px)',

sm: 'calc(var(--radius) - 4px)'

},

keyframes: {

'accordion-down': {

from: {

height: '0'

},

to: {

height: 'var(--radix-accordion-content-height)'

}

},

'accordion-up': {

from: {

height: 'var(--radix-accordion-content-height)'

},

to: {

height: '0'

}

}

},

animation: {

'accordion-down': 'accordion-down 0.2s ease-out',

'accordion-up': 'accordion-up 0.2s ease-out'

}

}

},

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

} satisfies Config;

**tsconfig.json:**

{

"compilerOptions": {

"target": "ES2017",

"lib": ["dom", "dom.iterable", "esnext"],

"allowJs": true,

"skipLibCheck": true,

"strict": true,

"noEmit": true,

"esModuleInterop": true,

"module": "esnext",

"moduleResolution": "bundler",

"resolveJsonModule": true,

"isolatedModules": true,

"jsx": "preserve",

"incremental": true,

"plugins": [

{

"name": "next"

}

],

"paths": {

"@/\*": ["./src/\*"]

}

},

"include": ["next-env.d.ts", "\*\*/\*.ts", "\*\*/\*.tsx", ".next/types/\*\*/\*.ts"],

"exclude": ["node\_modules"]

}