Build this as my initial prototype

Copy-paste this component to /components/ui folder:

```tsx

modern-animated-sign-in.tsx

'use client';

import {

memo,

ReactNode,

useState,

ChangeEvent,

FormEvent,

useEffect,

useRef,

forwardRef,

} from 'react';

import Image from 'next/image';

import {

motion,

useAnimation,

useInView,

useMotionTemplate,

useMotionValue,

} from 'motion/react';

import { Eye, EyeOff } from 'lucide-react';

import { cn } from '@/lib/utils';

// ==================== Input Component ====================

const Input = memo(

forwardRef(function Input(

{ className, type, ...props }: React.InputHTMLAttributes<HTMLInputElement>,

ref: React.ForwardedRef<HTMLInputElement>

) {

const radius = 100; // change this to increase the radius of the hover effect

const [visible, setVisible] = useState(false);

const mouseX = useMotionValue(0);

const mouseY = useMotionValue(0);

function handleMouseMove({

currentTarget,

clientX,

clientY,

}: React.MouseEvent<HTMLDivElement>) {

const { left, top } = currentTarget.getBoundingClientRect();

mouseX.set(clientX - left);

mouseY.set(clientY - top);

}

return (

<motion.div

style={{

background: useMotionTemplate`

radial-gradient(

${visible ? radius + 'px' : '0px'} circle at ${mouseX}px ${mouseY}px,

#3b82f6,

transparent 80%

)

`,

}}

onMouseMove={handleMouseMove}

onMouseEnter={() => setVisible(true)}

onMouseLeave={() => setVisible(false)}

className='group/input rounded-lg p-[2px] transition duration-300'

>

<input

type={type}

className={cn(

`shadow-input dark:placeholder-text-neutral-600 flex h-10 w-full rounded-md border-none bg-gray-50 px-3 py-2 text-sm text-black transition duration-400 group-hover/input:shadow-none file:border-0 file:bg-transparent file:text-sm file:font-medium placeholder:text-neutral-400 focus-visible:ring-[2px] focus-visible:ring-neutral-400 focus-visible:outline-none disabled:cursor-not-allowed disabled:opacity-50 dark:bg-zinc-800 dark:text-white dark:shadow-[0px\_0px\_1px\_1px\_#404040] dark:focus-visible:ring-neutral-600`,

className

)}

ref={ref}

{...props}

/>

</motion.div>

);

})

);

Input.displayName = 'Input';

// ==================== BoxReveal Component ====================

type BoxRevealProps = {

children: ReactNode;

width?: string;

boxColor?: string;

duration?: number;

overflow?: string;

position?: string;

className?: string;

};

const BoxReveal = memo(function BoxReveal({

children,

width = 'fit-content',

boxColor,

duration,

overflow = 'hidden',

position = 'relative',

className,

}: BoxRevealProps) {

const mainControls = useAnimation();

const slideControls = useAnimation();

const ref = useRef(null);

const isInView = useInView(ref, { once: true });

useEffect(() => {

if (isInView) {

slideControls.start('visible');

mainControls.start('visible');

} else {

slideControls.start('hidden');

mainControls.start('hidden');

}

}, [isInView, mainControls, slideControls]);

return (

<section

ref={ref}

style={{

position: position as

| 'relative'

| 'absolute'

| 'fixed'

| 'sticky'

| 'static',

width,

overflow,

}}

className={className}

>

<motion.div

variants={{

hidden: { opacity: 0, y: 75 },

visible: { opacity: 1, y: 0 },

}}

initial='hidden'

animate={mainControls}

transition={{ duration: duration ?? 0.5, delay: 0.25 }}

>

{children}

</motion.div>

<motion.div

variants={{ hidden: { left: 0 }, visible: { left: '100%' } }}

initial='hidden'

animate={slideControls}

transition={{ duration: duration ?? 0.5, ease: 'easeIn' }}

style={{

position: 'absolute',

top: 4,

bottom: 4,

left: 0,

right: 0,

zIndex: 20,

background: boxColor ?? '#5046e6',

borderRadius: 4,

}}

/>

</section>

);

});

// ==================== Ripple Component ====================

type RippleProps = {

mainCircleSize?: number;

mainCircleOpacity?: number;

numCircles?: number;

className?: string;

};

const Ripple = memo(function Ripple({

mainCircleSize = 210,

mainCircleOpacity = 0.24,

numCircles = 11,

className = '',

}: RippleProps) {

return (

<section

className={`max-w-[50%] absolute inset-0 flex items-center justify-center

dark:bg-white/5 bg-neutral-50

[mask-image:linear-gradient(to\_bottom,black,transparent)]

dark:[mask-image:linear-gradient(to\_bottom,white,transparent)] ${className}`}

>

{Array.from({ length: numCircles }, (\_, i) => {

const size = mainCircleSize + i \* 70;

const opacity = mainCircleOpacity - i \* 0.03;

const animationDelay = `${i \* 0.06}s`;

const borderStyle = i === numCircles - 1 ? 'dashed' : 'solid';

const borderOpacity = 5 + i \* 5;

return (

<span

key={i}

className='absolute animate-ripple rounded-full bg-foreground/15 border'

style={{

width: `${size}px`,

height: `${size}px`,

opacity: opacity,

animationDelay: animationDelay,

borderStyle: borderStyle,

borderWidth: '1px',

borderColor: `var(--foreground) dark:var(--background) / ${

borderOpacity / 100

})`,

top: '50%',

left: '50%',

transform: 'translate(-50%, -50%)',

}}

/>

);

})}

</section>

);

});

// ==================== OrbitingCircles Component ====================

type OrbitingCirclesProps = {

className?: string;

children: ReactNode;

reverse?: boolean;

duration?: number;

delay?: number;

radius?: number;

path?: boolean;

};

const OrbitingCircles = memo(function OrbitingCircles({

className,

children,

reverse = false,

duration = 20,

delay = 10,

radius = 50,

path = true,

}: OrbitingCirclesProps) {

return (

<>

{path && (

<svg

xmlns='http://www.w3.org/2000/svg'

version='1.1'

className='pointer-events-none absolute inset-0 size-full'

>

<circle

className='stroke-black/10 stroke-1 dark:stroke-white/10'

cx='50%'

cy='50%'

r={radius}

fill='none'

/>

</svg>

)}

<section

style={

{

'--duration': duration,

'--radius': radius,

'--delay': -delay,

} as React.CSSProperties

}

className={cn(

'absolute flex size-full transform-gpu animate-orbit items-center justify-center rounded-full border bg-black/10 [animation-delay:calc(var(--delay)\*1000ms)] dark:bg-white/10',

{ '[animation-direction:reverse]': reverse },

className

)}

>

{children}

</section>

</>

);

});

// ==================== TechOrbitDisplay Component ====================

type IconConfig = {

className?: string;

duration?: number;

delay?: number;

radius?: number;

path?: boolean;

reverse?: boolean;

component: () => React.ReactNode;

};

type TechnologyOrbitDisplayProps = {

iconsArray: IconConfig[];

text?: string;

};

const TechOrbitDisplay = memo(function TechOrbitDisplay({

iconsArray,

text = 'Animated Login',

}: TechnologyOrbitDisplayProps) {

return (

<section className='relative flex h-full w-full flex-col items-center justify-center overflow-hidden rounded-lg'>

<span className='pointer-events-none whitespace-pre-wrap bg-gradient-to-b from-black to-gray-300/80 bg-clip-text text-center text-7xl font-semibold leading-none text-transparent dark:from-white dark:to-slate-900/10'>

{text}

</span>

{iconsArray.map((icon, index) => (

<OrbitingCircles

key={index}

className={icon.className}

duration={icon.duration}

delay={icon.delay}

radius={icon.radius}

path={icon.path}

reverse={icon.reverse}

>

{icon.component()}

</OrbitingCircles>

))}

</section>

);

});

// ==================== AnimatedForm Component ====================

type FieldType = 'text' | 'email' | 'password';

type Field = {

label: string;

required?: boolean;

type: FieldType;

placeholder?: string;

onChange: (event: ChangeEvent<HTMLInputElement>) => void;

};

type AnimatedFormProps = {

header: string;

subHeader?: string;

fields: Field[];

submitButton: string;

textVariantButton?: string;

errorField?: string;

fieldPerRow?: number;

onSubmit: (event: FormEvent<HTMLFormElement>) => void;

googleLogin?: string;

goTo?: (event: React.MouseEvent<HTMLButtonElement>) => void;

};

type Errors = {

[key: string]: string;

};

const AnimatedForm = memo(function AnimatedForm({

header,

subHeader,

fields,

submitButton,

textVariantButton,

errorField,

fieldPerRow = 1,

onSubmit,

googleLogin,

goTo,

}: AnimatedFormProps) {

const [visible, setVisible] = useState<boolean>(false);

const [errors, setErrors] = useState<Errors>({});

const toggleVisibility = () => setVisible(!visible);

const validateForm = (event: FormEvent<HTMLFormElement>) => {

const currentErrors: Errors = {};

fields.forEach((field) => {

const value = (event.target as HTMLFormElement)[field.label]?.value;

if (field.required && !value) {

currentErrors[field.label] = `${field.label} is required`;

}

if (field.type === 'email' && value && !/\S+@\S+\.\S+/.test(value)) {

currentErrors[field.label] = 'Invalid email address';

}

if (field.type === 'password' && value && value.length < 6) {

currentErrors[field.label] =

'Password must be at least 6 characters long';

}

});

return currentErrors;

};

const handleSubmit = (event: FormEvent<HTMLFormElement>) => {

event.preventDefault();

const formErrors = validateForm(event);

if (Object.keys(formErrors).length === 0) {

onSubmit(event);

console.log('Form submitted');

} else {

setErrors(formErrors);

}

};

return (

<section className='max-md:w-full flex flex-col gap-4 w-96 mx-auto'>

<BoxReveal boxColor='var(--skeleton)' duration={0.3}>

<h2 className='font-bold text-3xl text-neutral-800 dark:text-neutral-200'>

{header}

</h2>

</BoxReveal>

{subHeader && (

<BoxReveal boxColor='var(--skeleton)' duration={0.3} className='pb-2'>

<p className='text-neutral-600 text-sm max-w-sm dark:text-neutral-300'>

{subHeader}

</p>

</BoxReveal>

)}

{googleLogin && (

<>

<BoxReveal

boxColor='var(--skeleton)'

duration={0.3}

overflow='visible'

width='unset'

>

<button

className='g-button group/btn bg-transparent w-full rounded-md border h-10 font-medium outline-hidden hover:cursor-pointer'

type='button'

onClick={() => console.log('Google login clicked')}

>

<span className='flex items-center justify-center w-full h-full gap-3'>

<Image

src='https://cdn1.iconfinder.com/data/icons/google-s-logo/150/Google\_Icons-09-512.png'

width={26}

height={26}

alt='Google Icon'

/>

{googleLogin}

</span>

<BottomGradient />

</button>

</BoxReveal>

<BoxReveal boxColor='var(--skeleton)' duration={0.3} width='100%'>

<section className='flex items-center gap-4'>

<hr className='flex-1 border-1 border-dashed border-neutral-300 dark:border-neutral-700' />

<p className='text-neutral-700 text-sm dark:text-neutral-300'>

or

</p>

<hr className='flex-1 border-1 border-dashed border-neutral-300 dark:border-neutral-700' />

</section>

</BoxReveal>

</>

)}

<form onSubmit={handleSubmit}>

<section

className={`grid grid-cols-1 md:grid-cols-${fieldPerRow} mb-4`}

>

{fields.map((field) => (

<section key={field.label} className='flex flex-col gap-2'>

<BoxReveal boxColor='var(--skeleton)' duration={0.3}>

<Label htmlFor={field.label}>

{field.label} <span className='text-red-500'>\*</span>

</Label>

</BoxReveal>

<BoxReveal

width='100%'

boxColor='var(--skeleton)'

duration={0.3}

className='flex flex-col space-y-2 w-full'

>

<section className='relative'>

<Input

type={

field.type === 'password'

? visible

? 'text'

: 'password'

: field.type

}

id={field.label}

placeholder={field.placeholder}

onChange={field.onChange}

/>

{field.type === 'password' && (

<button

type='button'

onClick={toggleVisibility}

className='absolute inset-y-0 right-0 pr-3 flex items-center text-sm leading-5'

>

{visible ? (

<Eye className='h-5 w-5' />

) : (

<EyeOff className='h-5 w-5' />

)}

</button>

)}

</section>

<section className='h-4'>

{errors[field.label] && (

<p className='text-red-500 text-xs'>

{errors[field.label]}

</p>

)}

</section>

</BoxReveal>

</section>

))}

</section>

<BoxReveal width='100%' boxColor='var(--skeleton)' duration={0.3}>

{errorField && (

<p className='text-red-500 text-sm mb-4'>{errorField}</p>

)}

</BoxReveal>

<BoxReveal

width='100%'

boxColor='var(--skeleton)'

duration={0.3}

overflow='visible'

>

<button

className='bg-gradient-to-br relative group/btn from-zinc-200 dark:from-zinc-900

dark:to-zinc-900 to-zinc-200 block dark:bg-zinc-800 w-full text-black

dark:text-white rounded-md h-10 font-medium shadow-[0px\_1px\_0px\_0px\_#ffffff40\_inset,0px\_-1px\_0px\_0px\_#ffffff40\_inset]

dark:shadow-[0px\_1px\_0px\_0px\_var(--zinc-800)\_inset,0px\_-1px\_0px\_0px\_var(--zinc-800)\_inset] outline-hidden hover:cursor-pointer'

type='submit'

>

{submitButton} &rarr;

<BottomGradient />

</button>

</BoxReveal>

{textVariantButton && goTo && (

<BoxReveal boxColor='var(--skeleton)' duration={0.3}>

<section className='mt-4 text-center hover:cursor-pointer'>

<button

className='text-sm text-blue-500 hover:cursor-pointer outline-hidden'

onClick={goTo}

>

{textVariantButton}

</button>

</section>

</BoxReveal>

)}

</form>

</section>

);

});

const BottomGradient = () => {

return (

<>

<span className='group-hover/btn:opacity-100 block transition duration-500 opacity-0 absolute h-px w-full -bottom-px inset-x-0 bg-gradient-to-r from-transparent via-cyan-500 to-transparent' />

<span className='group-hover/btn:opacity-100 blur-sm block transition duration-500 opacity-0 absolute h-px w-1/2 mx-auto -bottom-px inset-x-10 bg-gradient-to-r from-transparent via-indigo-500 to-transparent' />

</>

);

};

// ==================== AuthTabs Component ====================

interface AuthTabsProps {

formFields: {

header: string;

subHeader?: string;

fields: Array<{

label: string;

required?: boolean;

type: string;

placeholder: string;

onChange: (event: React.ChangeEvent<HTMLInputElement>) => void;

}>;

submitButton: string;

textVariantButton?: string;

};

goTo: (event: React.MouseEvent<HTMLButtonElement>) => void;

handleSubmit: (event: React.FormEvent<HTMLFormElement>) => void;

}

const AuthTabs = memo(function AuthTabs({

formFields,

goTo,

handleSubmit,

}: AuthTabsProps) {

return (

<div className='flex max-lg:justify-center w-full md:w-auto'>

{/\* Right Side \*/}

<div className='w-full lg:w-1/2 h-[100dvh] flex flex-col justify-center items-center max-lg:px-[10%]'>

<AnimatedForm

{...formFields}

fieldPerRow={1}

onSubmit={handleSubmit}

goTo={goTo}

googleLogin='Login with Google'

/>

</div>

</div>

);

});

// ==================== Label Component ====================

interface LabelProps extends React.LabelHTMLAttributes<HTMLLabelElement> {

htmlFor?: string;

}

const Label = memo(function Label({ className, ...props }: LabelProps) {

return (

<label

className={cn(

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

className

)}

{...props}

/>

);

});

// ==================== Exports ====================

export {

Input,

BoxReveal,

Ripple,

OrbitingCircles,

TechOrbitDisplay,

AnimatedForm,

AuthTabs,

Label,

BottomGradient,

};

demo.tsx

'use client';

import { useState, ChangeEvent, FormEvent, ReactNode } from 'react';

import {

Ripple,

AuthTabs,

TechOrbitDisplay,

} from '@/components/blocks/modern-animated-sign-in';

import Image from 'next/image';

type FormData = {

email: string;

password: string;

};

interface OrbitIcon {

component: () => ReactNode;

className: string;

duration?: number;

delay?: number;

radius?: number;

path?: boolean;

reverse?: boolean;

}

const iconsArray: OrbitIcon[] = [

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/html5/html5-original.svg'

alt='HTML5'

/>

),

className: 'size-[30px] border-none bg-transparent',

duration: 20,

delay: 20,

radius: 100,

path: false,

reverse: false,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/css3/css3-original.svg'

alt='CSS3'

/>

),

className: 'size-[30px] border-none bg-transparent',

duration: 20,

delay: 10,

radius: 100,

path: false,

reverse: false,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/typescript/typescript-original.svg'

alt='TypeScript'

/>

),

className: 'size-[50px] border-none bg-transparent',

radius: 210,

duration: 20,

path: false,

reverse: false,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/javascript/javascript-original.svg'

alt='JavaScript'

/>

),

className: 'size-[50px] border-none bg-transparent',

radius: 210,

duration: 20,

delay: 20,

path: false,

reverse: false,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/tailwindcss/tailwindcss-original.svg'

alt='TailwindCSS'

/>

),

className: 'size-[30px] border-none bg-transparent',

duration: 20,

delay: 20,

radius: 150,

path: false,

reverse: true,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/nextjs/nextjs-original.svg'

alt='Nextjs'

/>

),

className: 'size-[30px] border-none bg-transparent',

duration: 20,

delay: 10,

radius: 150,

path: false,

reverse: true,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/react/react-original.svg'

alt='React'

/>

),

className: 'size-[50px] border-none bg-transparent',

radius: 270,

duration: 20,

path: false,

reverse: true,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/figma/figma-original.svg'

alt='Figma'

/>

),

className: 'size-[50px] border-none bg-transparent',

radius: 270,

duration: 20,

delay: 60,

path: false,

reverse: true,

},

{

component: () => (

<Image

width={100}

height={100}

src='https://cdn.jsdelivr.net/gh/devicons/devicon@latest/icons/git/git-original.svg'

alt='Git'

/>

),

className: 'size-[50px] border-none bg-transparent',

radius: 320,

duration: 20,

delay: 20,

path: false,

reverse: false,

},

];

export function Demo() {

const [formData, setFormData] = useState<FormData>({

email: '',

password: '',

});

const goToForgotPassword = (

event: React.MouseEvent<HTMLButtonElement | HTMLAnchorElement>

) => {

event.preventDefault();

console.log('forgot password');

};

const handleInputChange = (

event: ChangeEvent<HTMLInputElement>,

name: keyof FormData

) => {

const value = event.target.value;

setFormData((prevState) => ({

...prevState,

[name]: value,

}));

};

const handleSubmit = async (event: FormEvent<HTMLFormElement>) => {

event.preventDefault();

console.log('Form submitted', formData);

};

const formFields = {

header: 'Welcome back',

subHeader: 'Sign in to your account',

fields: [

{

label: 'Email',

required: true,

type: 'email',

placeholder: 'Enter your email address',

onChange: (event: ChangeEvent<HTMLInputElement>) =>

handleInputChange(event, 'email'),

},

{

label: 'Password',

required: true,

type: 'password',

placeholder: 'Enter your password',

onChange: (event: ChangeEvent<HTMLInputElement>) =>

handleInputChange(event, 'password'),

},

],

submitButton: 'Sign in',

textVariantButton: 'Forgot password?',

};

return (

<section className='flex max-lg:justify-center'>

{/\* Left Side \*/}

<span className='flex flex-col justify-center w-1/2 max-lg:hidden'>

<Ripple mainCircleSize={100} />

<TechOrbitDisplay iconsArray={iconsArray} />

</span>

{/\* Right Side \*/}

<span className='w-1/2 h-[100dvh] flex flex-col justify-center items-center max-lg:w-full max-lg:px-[10%]'>

<AuthTabs

formFields={formFields}

goTo={goToForgotPassword}

handleSubmit={handleSubmit}

/>

</span>

</section>

);

}

```

Install these NPM dependencies:

```bash

motion, lucide-react

```

Extend existing tailwind.config.js with this code:

```js

module.exports = {

darkMode: "class",

theme: {

extend: {

colors: {

background: "var(--background)",

foreground: "var(--foreground)",

skeleton: "var(--skeleton)",

border: "var(--btn-border)",

input: "var(--input)",

},

borderRadius: {

DEFAULT: "0.5rem",

},

boxShadow: {

input: [

"0px 2px 3px -1px rgba(0, 0, 0, 0.1)",

"0px 1px 0px 0px rgba(25, 28, 33, 0.02)",

"0px 0px 0px 1px rgba(25, 28, 33, 0.08)",

].join(", "),

},

animation: {

ripple: "ripple 2s ease calc(var(--i, 0) \* 0.2s) infinite",

orbit: "orbit calc(var(--duration) \* 1s) linear infinite",

},

keyframes: {

ripple: {

"0%, 100%": { transform: "translate(-50%, -50%) scale(1)" },

"50%": { transform: "translate(-50%, -50%) scale(0.9)" },

},

orbit: {

"0%": {

transform:

"rotate(0deg) translateY(calc(var(--radius) \* 1px)) rotate(0deg)",

},

"100%": {

transform:

"rotate(360deg) translateY(calc(var(--radius) \* 1px)) rotate(-360deg)",

},

}

},

},

},

};

```

Extend existing globals.css with this code:

```css

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer base {

:root {

--background: hsl(0, 0%, 100%);

--foreground: hsl(0, 0%, 0%);

--skeleton: hsl(0, 0%, 90%);

--border: hsl(220, 20%, 90%);

--btn-border: hsl(214deg, 31.8%, 91.4%);

--input: hsl(220, 20%, 90%);

--radius: 0.5rem;

}

.dark {

--background: hsl(222, 94%, 5%);

--foreground: hsl(0, 0%, 100%);

--skeleton: hsl(218, 36%, 16%);

--border: hsl(220, 20%, 90%);

--btn-border: hsl(217deg, 32.6%, 17.5%);

--input: hsl(219, 63%, 16%);

--radius: 0.5rem;

}

body {

@apply bg-background text-foreground;

}

}

@layer components {

.g-button {

@apply rounded-[var(--radius)] border;

border-color: var(--btn-border)

}

}

```