Build this as my initial prototype

Copy-paste this component to /components/ui folder:
```tsx
splite.tsx
'use client'

import { Suspense, lazy } from 'react'
const Spline = lazy(() => import('@splinetool/react-spline'))

interface SplineSceneProps {
  scene: string
  className?: string
}

export function SplineScene({ scene, className }: SplineSceneProps) {
  return (
    <Suspense
      fallback={
        <div className="w-full h-full flex items-center justify-center">
          <span className="loader"></span>
        </div>
      }
    >
      <Spline
        scene={scene}
        className={className}
      />
    </Suspense>
  )
}

demo.tsx
'use client'

import { SplineScene } from "@/components/ui/splite";
import { Card } from "@/components/ui/card"
import { Spotlight } from "@/components/ui/spotlight"

export function SplineSceneBasic() {
  return (
    <Card className="w-full h-[500px] bg-black/[0.96] relative overflow-hidden">
      <Spotlight
        className="-top-40 left-0 md:left-60 md:-top-20"
```

```
        fill="white"
      />

      <div className="flex h-full">
        {/* Left content */}
        <div className="flex-1 p-8 relative z-10 flex flex-col justify-center">
          <h1 className="text-4xl md:text-5xl font-bold bg-clip-text text-transparent
bg-gradient-to-b from-neutral-50 to-neutral-400">
            Interactive 3D
          </h1>
          <p className="mt-4 text-neutral-300 max-w-lg">
            Bring your UI to life with beautiful 3D scenes. Create immersive experiences
            that capture attention and enhance your design.
          </p>
        </div>

        {/* Right content */}
        <div className="flex-1 relative">
          <SplineScene
            scene="https://prod.spline.design/kZDDjO5HuC9GJUM2/scene.splinecode"
            className="w-full h-full"
          />
        </div>
      </div>
    </Card>
  )
}
```

Copy-paste these files for dependencies:
```tsx
aceternity/spotlight
import React from "react";
import { cn } from "@/lib/utils";

type SpotlightProps = {
  className?: string;
  fill?: string;
};

export const Spotlight = ({ className, fill }: SpotlightProps) => {
  return (
    <svg
      className={cn(
```

```
      "animate-spotlight pointer-events-none absolute z-[1]  h-[169%] w-[138%] lg:w-[84%]
opacity-0",
       className
     )}
     xmlns="http://www.w3.org/2000/svg"
     viewBox="0 0 3787 2842"
     fill="none"
   >
     <g filter="url(#filter)">
       <ellipse
         cx="1924.71"
         cy="273.501"
         rx="1924.71"
         ry="273.501"
         transform="matrix(-0.822377 -0.568943 -0.568943 0.822377 3631.88 2291.09)"
         fill={fill || "white"}
         fillOpacity="0.21"
       ></ellipse>
     </g>
     <defs>
       <filter
         id="filter"
         x="0.860352"
         y="0.838989"
         width="3785.16"
         height="2840.26"
         filterUnits="userSpaceOnUse"
         colorInterpolationFilters="sRGB"
       >
         <feFlood floodOpacity="0" result="BackgroundImageFix"></feFlood>
         <feBlend
           mode="normal"
           in="SourceGraphic"
           in2="BackgroundImageFix"
           result="shape"
         ></feBlend>
         <feGaussianBlur
           stdDeviation="151"
           result="effect1_foregroundBlur_1065_8"
         ></feGaussianBlur>
       </filter>
     </defs>
   </svg>
 );
```

```
};

```
```tsx
ibelick/spotlight
'use client';
import React, { useRef, useState, useCallback, useEffect } from 'react';
import { motion, useSpring, useTransform, SpringOptions } from 'framer-motion';
import { cn } from '@/lib/utils';

type SpotlightProps = {
  className?: string;
  size?: number;
  springOptions?: SpringOptions;
};

export function Spotlight({
  className,
  size = 200,
  springOptions = { bounce: 0 },
}: SpotlightProps) {
  const containerRef = useRef<HTMLDivElement>(null);
  const [isHovered, setIsHovered] = useState(false);
  const [parentElement, setParentElement] = useState<HTMLElement | null>(null);

  const mouseX = useSpring(0, springOptions);
  const mouseY = useSpring(0, springOptions);

  const spotlightLeft = useTransform(mouseX, (x) => `${x - size / 2}px`);
  const spotlightTop = useTransform(mouseY, (y) => `${y - size / 2}px`);

  useEffect(() => {
    if (containerRef.current) {
      const parent = containerRef.current.parentElement;
      if (parent) {
        parent.style.position = 'relative';
        parent.style.overflow = 'hidden';
        setParentElement(parent);
      }
    }
  }, []);

  const handleMouseMove = useCallback(
    (event: MouseEvent) => {
```

```
      if (!parentElement) return;
      const { left, top } = parentElement.getBoundingClientRect();
      mouseX.set(event.clientX - left);
      mouseY.set(event.clientY - top);
    },
    [mouseX, mouseY, parentElement]
  );

  useEffect(() => {
    if (!parentElement) return;

    parentElement.addEventListener('mousemove', handleMouseMove);
    parentElement.addEventListener('mouseenter', () => setIsHovered(true));
    parentElement.addEventListener('mouseleave', () => setIsHovered(false));

    return () => {
      parentElement.removeEventListener('mousemove', handleMouseMove);
      parentElement.removeEventListener('mouseenter', () => setIsHovered(true));
      parentElement.removeEventListener('mouseleave', () =>
        setIsHovered(false)
      );
    };
  }, [parentElement, handleMouseMove]);

  return (
    <motion.div
      ref={containerRef}
      className={cn(
        'pointer-events-none absolute rounded-full
bg-[radial-gradient(circle_at_center,var(--tw-gradient-stops),transparent_80%)] blur-xl
transition-opacity duration-200',
        'from-zinc-50 via-zinc-100 to-zinc-200',
        isHovered ? 'opacity-100' : 'opacity-0',
        className
      )}
      style={{
        width: size,
        height: size,
        left: spotlightLeft,
        top: spotlightTop,
      }}
    />
  );
}
```

```
```tsx
shadcn/card
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<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLHeadingElement>
>(({ className, ...props }, ref) => (
  <h3
    ref={ref}
    className={cn(
      "text-2xl font-semibold leading-none tracking-tight",
      className,
```
```

```
    )}
    {...props}
  />
))
CardTitle.displayName = "CardTitle"

const CardDescription = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLParagraphElement>
>(({ className, ...props }, ref) => (
  <p
    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 }

```

Install these NPM dependencies:
```bash
```

@splinetool/runtime, @splinetool/react-spline, framer-motion
```
```