Build this as my initial prototype

Copy-paste this component to /components/ui folder:
```tsx
glowing-effect.tsx
"use client";

import { memo, useCallback, useEffect, useRef } from "react";
import { cn } from "@/lib/utils";
import { animate } from "motion/react";

interface GlowingEffectProps {
  blur?: number;
  inactiveZone?: number;
  proximity?: number;
  spread?: number;
  variant?: "default" | "white";
  glow?: boolean;
  className?: string;
  disabled?: boolean;
  movementDuration?: number;
  borderWidth?: number;
}
const GlowingEffect = memo(
  ({
    blur = 0,
    inactiveZone = 0.7,
    proximity = 0,
    spread = 20,
    variant = "default",
    glow = false,
    className,
    movementDuration = 2,
    borderWidth = 1,
    disabled = true,
  }: GlowingEffectProps) => {
    const containerRef = useRef<HTMLDivElement>(null);
    const lastPosition = useRef({ x: 0, y: 0 });
    const animationFrameRef = useRef<number>(0);

    const handleMove = useCallback(
      (e?: MouseEvent | { x: number; y: number }) => {
        if (!containerRef.current) return;
```

```
if (animationFrameRef.current) {
  cancelAnimationFrame(animationFrameRef.current);
}

animationFrameRef.current = requestAnimationFrame(() => {
  const element = containerRef.current;
  if (!element) return;

  const { left, top, width, height } = element.getBoundingClientRect();
  const mouseX = e?.x ?? lastPosition.current.x;
  const mouseY = e?.y ?? lastPosition.current.y;

  if (e) {
    lastPosition.current = { x: mouseX, y: mouseY };
  }

  const center = [left + width * 0.5, top + height * 0.5];
  const distanceFromCenter = Math.hypot(
    mouseX - center[0],
    mouseY - center[1]
  );
  const inactiveRadius = 0.5 * Math.min(width, height) * inactiveZone;

  if (distanceFromCenter < inactiveRadius) {
    element.style.setProperty("--active", "0");
    return;
  }

  const isActive =
    mouseX > left - proximity &&
    mouseX < left + width + proximity &&
    mouseY > top - proximity &&
    mouseY < top + height + proximity;

  element.style.setProperty("--active", isActive ? "1" : "0");

  if (!isActive) return;

  const currentAngle =
    parseFloat(element.style.getPropertyValue("--start")) || 0;
  let targetAngle =
    (180 * Math.atan2(mouseY - center[1], mouseX - center[0])) /
      Math.PI +
    90;
```

```
        const angleDiff = ((targetAngle - currentAngle + 180) % 360) - 180;
        const newAngle = currentAngle + angleDiff;

        animate(currentAngle, newAngle, {
          duration: movementDuration,
          ease: [0.16, 1, 0.3, 1],
          onUpdate: (value) => {
            element.style.setProperty("--start", String(value));
          },
        });
      });
    },
    [inactiveZone, proximity, movementDuration]
  );

  useEffect(() => {
    if (disabled) return;

    const handleScroll = () => handleMove();
    const handlePointerMove = (e: PointerEvent) => handleMove(e);

    window.addEventListener("scroll", handleScroll, { passive: true });
    document.body.addEventListener("pointermove", handlePointerMove, {
      passive: true,
    });

    return () => {
      if (animationFrameRef.current) {
        cancelAnimationFrame(animationFrameRef.current);
      }
      window.removeEventListener("scroll", handleScroll);
      document.body.removeEventListener("pointermove", handlePointerMove);
    };
  }, [handleMove, disabled]);

  return (
    <>
      <div
        className={cn(
          "pointer-events-none absolute -inset-px hidden rounded-[inherit] border opacity-0
transition-opacity",
          glow && "opacity-100",
          variant === "white" && "border-white",
```

```
    disabled && "!block"
  )}
/>
<div
  ref={containerRef}
  style={
    {
      "--blur": `${blur}px`,
      "--spread": spread,
      "--start": "0",
      "--active": "0",
      "--glowingeffect-border-width": `${borderWidth}px`,
      "--repeating-conic-gradient-times": "5",
      "--gradient":
        variant === "white"
          ? `repeating-conic-gradient(
          from 236.84deg at 50% 50%,
          var(--black),
          var(--black) calc(25% / var(--repeating-conic-gradient-times))
        )`
          : `radial-gradient(circle, #dd7bbb 10%, #dd7bbb00 20%),
        radial-gradient(circle at 40% 40%, #d79f1e 5%, #d79f1e00 15%),
        radial-gradient(circle at 60% 60%, #5a922c 10%, #5a922c00 20%),
        radial-gradient(circle at 40% 60%, #4c7894 10%, #4c789400 20%),
        repeating-conic-gradient(
          from 236.84deg at 50% 50%,
          #dd7bbb 0%,
          #d79f1e calc(25% / var(--repeating-conic-gradient-times)),
          #5a922c calc(50% / var(--repeating-conic-gradient-times)),
          #4c7894 calc(75% / var(--repeating-conic-gradient-times)),
          #dd7bbb calc(100% / var(--repeating-conic-gradient-times))
        )`,
    } as React.CSSProperties
  }
  className={cn(
    "pointer-events-none absolute inset-0 rounded-[inherit] opacity-100 transition-opacity",
    glow && "opacity-100",
    blur > 0 && "blur-[var(--blur)] ",
    className,
    disabled && "!hidden"
  )}
>
  <div
    className={cn(
```

```
            "glow",
            "rounded-[inherit]",
            'after:content-[""] after:rounded-[inherit] after:absolute
after:inset-[calc(-1*var(--glowingeffect-border-width))]',
            "after:[border:var(--glowingeffect-border-width)_solid_transparent]",
            "after:[background:var(--gradient)] after:[background-attachment:fixed]",
            "after:opacity-[var(--active)] after:transition-opacity after:duration-300",
            "after:[mask-clip:padding-box,border-box]",
            "after:[mask-composite:intersect]",

"after:[mask-image:linear-gradient(#0000,#0000),conic-gradient(from_calc((var(--start)-var(--spr
ead))*1deg),#00000000_0deg,#fff,#00000000_calc(var(--spread)*2deg))]"
          )}
        />
      </div>
    </>
  );
 }
);

GlowingEffect.displayName = "GlowingEffect";

export { GlowingEffect };
```

demo.tsx

```
"use client";

import { Box, Lock, Search, Settings, Sparkles } from "lucide-react";
import { GlowingEffect } from "@/components/ui/glowing-effect";
import { cn } from "@/lib/utils";

export function GlowingEffectDemo() {
 return (
  <ul className="grid grid-cols-1 grid-rows-none gap-4 md:grid-cols-12 md:grid-rows-3
lg:gap-4 xl:max-h-[34rem] xl:grid-rows-2">
    <GridItem
      area="md:[grid-area:1/1/2/7] xl:[grid-area:1/1/2/5]"
      icon={<Box className="h-4 w-4" />}
      title="Do things the right way"
      description="Running out of copy so I'll write anything."
    />
    <GridItem
      area="md:[grid-area:1/7/2/13] xl:[grid-area:2/1/3/5]"
```

```
        icon={<Settings className="h-4 w-4" />}
        title="The best AI code editor ever."
        description="Yes, it's true. I'm not even kidding. Ask my mom if you don't believe me."
      />
      <GridItem
        area="md:[grid-area:2/1/3/7] xl:[grid-area:1/5/3/8]"
        icon={<Lock className="h-4 w-4" />}
        title="You should buy Aceternity UI Pro"
        description="It's the best money you'll ever spend"
      />
      <GridItem
        area="md:[grid-area:2/7/3/13] xl:[grid-area:1/8/2/13]"
        icon={<Sparkles className="h-4 w-4" />}
        title="This card is also built by Cursor"
        description="I'm not even kidding. Ask my mom if you don't believe me."
      />
      <GridItem
        area="md:[grid-area:3/1/4/13] xl:[grid-area:2/8/3/13]"
        icon={<Search className="h-4 w-4" />}
        title="Coming soon on Aceternity UI"
        description="I'm writing the code as I record this, no shit."
      />
    </ul>
  );
}

interface GridItemProps {
  area: string;
  icon: React.ReactNode;
  title: string;
  description: React.ReactNode;
}

const GridItem = ({ area, icon, title, description }: GridItemProps) => {
  return (
    <li className={cn("min-h-[14rem] list-none", area)}>
      <div className="relative h-full rounded-[1.25rem] border-[0.75px] border-border p-2
md:rounded-[1.5rem] md:p-3">
        <GlowingEffect
          spread={40}
          glow={true}
          disabled={false}
          proximity={64}
          inactiveZone={0.01}
```

```
          borderWidth={3}
        />
        <div className="relative flex h-full flex-col justify-between gap-6 overflow-hidden
rounded-xl border-[0.75px] bg-background p-6 shadow-sm
dark:shadow-[0px_0px_27px_0px_rgba(45,45,45,0.3)] md:p-6">
          <div className="relative flex flex-1 flex-col justify-between gap-3">
            <div className="w-fit rounded-lg border-[0.75px] border-border bg-muted p-2">
              {icon}
            </div>
            <div className="space-y-3">
              <h3 className="pt-0.5 text-xl leading-[1.375rem] font-semibold font-sans
tracking-[-0.04em] md:text-2xl md:leading-[1.875rem] text-balance text-foreground">
                {title}
              </h3>
              <h2 className="[&_b]:md:font-semibold [&_strong]:md:font-semibold font-sans
text-sm leading-[1.125rem] md:text-base md:leading-[1.375rem] text-muted-foreground">
                {description}
              </h2>
            </div>
          </div>
        </div>
      </div>
    </li>
  );
};
```

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