```tsx
import React, { useState, useRef } from 'react';
import { Play, Check, Loader2 } from 'lucide-react';
import { Button } from '../spread_sheet_swarm/ui/button';

interface AnimatedRunButtonProps {
  onRun: () => Promise<void>;
}

const AnimatedRunButton: React.FC<AnimatedRunButtonProps> = ({ onRun }) => {
  const [isRunning, setIsRunning] = useState<boolean>(false);
  const [isComplete, setIsComplete] = useState<boolean>(false);
  const buttonRef = useRef(null);
  const handleClick = async (e: any) => {
    if (isRunning) return;

    // Create ripple effect
    const button: any = buttonRef.current;
    const circle = document.createElement('span');
    const diameter = Math.max(button.clientWidth, button.clientHeight);
    const radius = diameter / 2;

    circle.style.width = circle.style.height = `${diameter}px`;
    circle.style.left = `${e.clientX - button.offsetLeft - radius}px`;
    circle.style.top = `${e.clientY - button.offsetTop - radius}px`;
    circle.classList.add('ripple');
```

```
  const ripple = button.getElementsByClassName('ripple')[0];
  if (ripple) {
    ripple.remove();
  }
  button.appendChild(circle);

  // Start the task
  setIsRunning(true);
  setIsComplete(false);

  try {
    await onRun();
    setIsComplete(true);

    // Reset after showing completion state
    setTimeout(() => {
      setIsComplete(false);
      setIsRunning(false);
    }, 2000);
  } catch (error) {
    setIsRunning(false);
    setIsComplete(false);
  }
};

return (
```

```
<>
  <style jsx>{`
    .button-wrapper {
      position: relative;
    }

    .ripple {
      position: absolute;
      border-radius: 50%;
      transform: scale(0);
      animation: ripple 600ms linear;
      background-color: rgba(255, 255, 255, 0.7);
    }

    .progress-bar {
      position: absolute;
      bottom: 0;
      left: 0;
      height: 2px;
      width: 100%;
      background-color: rgb(147 197 253);
      border-radius: 9999px;
      animation: progress 2s ease-in-out infinite;
    }

    @keyframes ripple {
```

```
    to {
      transform: scale(4);
      opacity: 0;
    }
  }

  @keyframes progress {
    0% {
      transform: translateX(-100%);
    }
    100% {
      transform: translateX(100%);
    }
  }

  @keyframes pop-in {
    0% {
      transform: scale(0);
      opacity: 0;
    }
    100% {
      transform: scale(1);
      opacity: 1;
    }
  }
`}</style>
```

```
<div className="button-wrapper">
  <Button
    ref={buttonRef}
    onClick={handleClick}
    disabled={isRunning}
    className={`
      relative overflow-hidden transition-all duration-300
      ${isComplete ? 'bg-green-500 hover:bg-green-600' : ''}
      ${isRunning ? 'bg-blue-500 hover:bg-blue-600' : ''}
    `}
  >
    <div className="flex items-center space-x-2">
      {isComplete ? (
        <Check className="w-4 h-4 animate-pop-in" />
      ) : isRunning ? (
        <Loader2 className="w-4 h-4 animate-spin" />
      ) : (
        <Play className="w-4 h-4" />
      )}
      <span>{isComplete ? 'Complete' : isRunning ? 'Running...' : 'Run Task'}</span>
    </div>
  </Button>

  {isRunning && <div className="progress-bar" />}
</div>
```

```
    </>
  );
};

export default AnimatedRunButton;
```