```
'use client';

import { cn } from '@/shared/utils/cn';
import { formatPrice, getTruncatedString } from '@/shared/utils/helpers';
import { ReactNode, useState } from 'react';
import Link from 'next/link';
import { Share2 } from 'lucide-react';
import Avatar from '@/shared/components/avatar';
import ShareModal from './share-modal';
import { trpc } from '@/shared/utils/trpc/trpc';
import { ReviewProps } from '@/shared/components/rating/list-rating';
import { getReviewRating } from '@/shared/components/rating/helper';
import ReactStars from 'react-rating-star-with-type';

interface Props {
  title: string;
  description: string;
  icon: ReactNode;
  className?: string;
  btnLabel?: string;
  input?: number | null;
  output?: number | null;
  isRating?: boolean;
  promptId?: any;
  link: string;
  userId?: string;
  id?: string;
}

const InfoCard = ({
  title,
  description,
  icon,
  className,
  btnLabel,
  input,
  output,
  userId,
  id,
  link,
}: Props) => {
  const reviews = id ? trpc.explorer.getReviews.useQuery(id ?? '') : null;
  const { modelRating } = getReviewRating(
    (reviews?.data as ReviewProps[]) || [],
  );

  const [isButtonHover, setIsButtonHover] = useState(false);
  const [isShowShareModalOpen, setIsShowModalOpen] = useState<boolean>(false);

  const handleShowShareModal = () => setIsShowModalOpen(true);

  const handleCloseModal = () => setIsShowModalOpen(false);
  const renderPrice = (label: string, price: number) => (
    <li className="pricing-unit">
      <span className="font-semibold">{label}</span>
      <span>{formatPrice(price)}/1M Tokens</span>
    </li>
  );

  return (
    <div
      className={cn(
        'relative flex gap-4 p-4 px-3 border border-primary rounded-lg overflow-hidden group',
        className,
      )}
    >
      <div>
        <div className="flex items-center justify-center h-10 bg-primary text-white rounded-lg aspect-square">
          {icon}
        </div>

        {id && (
          <div className="mt-3 relative flex items-center justify-center gap-1 xl:hidden">
            <div className="mb-0.5">
              <ReactStars value={modelRating} isEdit={false} count={1} />
            </div>
            <span className="text-xs font-semibold">{modelRating || 0}/5</span>
          </div>
        )}
      </div>
      <div className="h-4/5 flex flex-col overflow-y-auto no-scrollbar">
        <div className="flex flex-col gap-2 flex-grow">
          <h1 className="text-xl sm:text-2xl font-bold">{title}</h1>

          <Avatar userId={userId} showUsername showBorder />
          <span title={description} className="text-sm">
            {getTruncatedString(description, 100)}
          </span>
        </div>

        {(input || output) && (
          <ul className="p-0 my-3 flex items-center gap-2">
            {input && renderPrice('Input', input)}
            {output && renderPrice('Output', output)}
          </ul>
        )}
      </div>

      {id && (
        <div className="bottom-2 left-4 absolute items-center justify-center gap-1 hidden xl:flex">
          <div className="mb-0.5">
            <ReactStars value={modelRating} isEdit={false} count={1} />
```

```
        </div>
        <span className="text-sm font-semibold">{modelRating || 0}/5</span>
      </div>
    )}

    <div
      className="cursor-pointer hover:opacity-70"
      onClick={handleShowShareModal}
    >
      <svg
        width="95"
        height="25"
        viewBox="0 0 95 25"
        xmlns="http://www.w3.org/2000/svg"
             className="rating-svg absolute right-[150px] xl:right-[120px] bottom-0 scale-x-[2.5]
scale-y-[1.8] fill-[#FB0101]"
      >
        <path d="M21 0H95V25H0L21 0Z" className="fill-[#b42020]" />
      </svg>
      <div className="absolute right-[150px] bottom-0 text-white px-4 py-1">
        <div className="relative flex items-center justify-center gap-2 w-[80px] xl:w-[66px]">
          <span>Share</span>
          <Share2 />
        </div>
      </div>
    </div>
```

```
<Link href={link && link}>
  <div>
    <div className="absolute right-0 bottom-0">
      <svg
        width="95"
        height="25"
        viewBox="0 0 95 25"
        xmlns="http://www.w3.org/2000/svg"
        className="preview-svg scale-x-[2.5] scale-y-[1.8] fill-[#FB0101]"
      >
        <path
          d="M21 0H95V25H0L21 0Z"
          className={`${isButtonHover ? 'fill-[#c95e5e]' : 'fill-[#FB0101]'}`}
        />
      </svg>
    </div>
    <div
      className="absolute right-0 bottom-0 text-white px-4 py-1"
      onMouseEnter={() => setIsButtonHover(true)}
      onMouseLeave={() => setIsButtonHover(false)}
    >
      <div className="relative flex items-center justify-center gap-2 w-[110px]">
        <span>{btnLabel || 'Preview'}</span>
        <svg
          width="16"
```

```
                height="16"
                viewBox="0 0 16 16"
                fill="none"
                xmlns="http://www.w3.org/2000/svg"
              >
                <path
                  fillRule="evenodd"
                  clipRule="evenodd"
                  d="M5.74999 2H4.99999V3.5H5.74999H11.4393L2.21966 12.7197L1.68933
13.25L2.74999 14.3107L3.28032 13.7803L12.4988
4.56182V10.25V11H13.9988V10.25V3C13.9988 2.44772 13.5511 2 12.9988 2H5.74999Z"
                  fill="white"
                />
              </svg>
            </div>
          </div>
        </div>
      </Link>
      <ShareModal
        isOpen={isShowShareModalOpen}
        onClose={handleCloseModal}
        link={link}
      />
    </div>
  );
};
```

```
export default InfoCard;
```