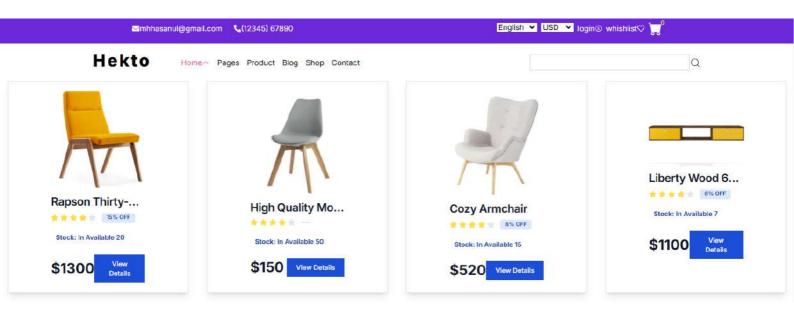


```
import { Rule } from "@sanity/types";
  name: "product",
type: "document",
  title: "Product",
  fields: [
       name: "name",
       type: "string",
       title: "Name",
       validation: (Rule: Rule) => Rule.required().error("Name is required"),
       name: "slug",
type: "slug",
title: "Slug",
description: "This will be used for dynamic routing.",
       options: {
| source: "name", // Set the source of the slug to be the product name
         maxLength: 96, // Maximum length of the slug
         slugify: (input: string) =>
            input
               .toLowerCase() // Convert the input to lowercase
.replace(/\s+/g, "-") // Replace spaces with hyphens
.replace(/[^\w\-]+/g, ""), // Remove any non-word characters (optional)
       validation: (Rule: Rule) => Rule.required().error("Slug is required"),
       name: "image",
       type: "image",
title: "Image",
       options: {
        hotspot: true,
       description: "Upload an image of the product.",
```



















```
type Products = {
 slug: string;
 _id: string;
  name: string;
  imageUrl: string;
  price: number;
  description: string;
  discountPercentage: number;
 stockLevel: number;
};
Codeium: Refactor | Explain | Generate JSDoc | X
const Products = async () => {
  const products = await client.fetch(`
                 *[_type == 'product']
                 { _id,
                   name,
                   "imageUrl": image.asset->url,
                   price,
                   description,
                   discountPercentage,
                     stockLevel,
                      "slug": slug.current }
                      `);
```

```
const query = ` *[_type == 'product' && slug.current == '${slug}']{
  name,
 description,
 price,
  "imageUrl": image.asset->url,
  stockLevel,
  category,
const fetchData: ProductData[] = await client.fetch(query, {
 slug: slug,
});
if (fetchData.length === 0) {
  return <div>Product not found</div>;
//console.log(fetchData.slug)
return (
    {fetchData.map((product: ProductData) => {
      return (
```