

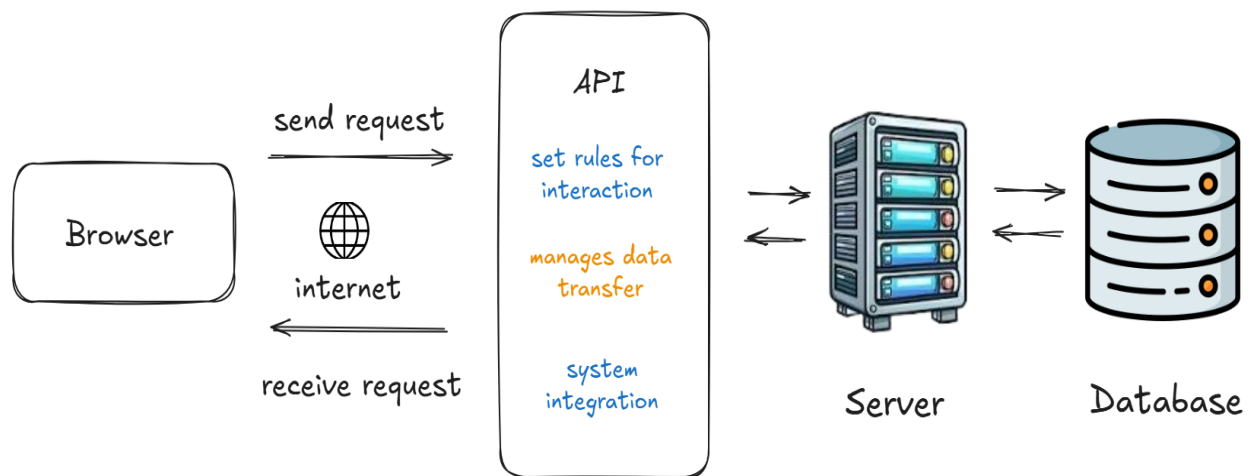
Task for Day 3

API Integration Report

FURNIRO

API Understanding

API integration process



Schemas Adjustments

Adjustments made to schemas

DAY-2 product.ts

```
1 export default {
2   name: 'product',
3   type: 'document',
4   title: 'Product',
5   fields: [
6     {
7       name: 'name',
8       type: 'string',
9       title: 'Product Name',
10    },
11    {
12      name: 'description',
13      type: 'string',
14      title: 'Description'
15    },
16    {
17      name: 'price',
18      type: 'number',
19      title: 'Product Price',
20    },
21    {
22      name: 'discountPercentage',
23      type: 'number',
24      title: 'Discount Percentage',
25    },
26    {
27      name: 'priceWithoutDiscount',
28      type: 'number',
29      title: 'Price Without Discount',
30      description: 'Original price before discount'
31    },
32    {
33      name: 'rating',
34      type: 'number',
35      title: 'Rating',
36      description: 'Rating of the product'
37    },
38    {
39      name: 'ratingCount',
40      type: 'number',
41      title: 'Rating Count',
42      description: 'Number of ratings'
43    },
44    {
45      name: 'tags',
46      type: 'array',
47      title: 'Tags',
48      of: [{ type: 'string' }],
49      description: 'List tags like "new arrival", "bestseller", etc.'
50    },
51    {
52      name: 'sizes',
53      type: 'array',
54      title: 'Sizes',
55      of: [{ type: 'string' }],
56      description: 'List sizes like S, M, L, XL, etc.'
57    },
58    {
59      name: 'image',
60      type: 'image',
61      title: 'Product Image',
62      description: 'Product image'
63    },
64    {
65      name: 'isLow',
66      type: 'boolean',
67      title: 'Is Low',
68      description: 'Is low price'
69    }
70 ]
71 }
```



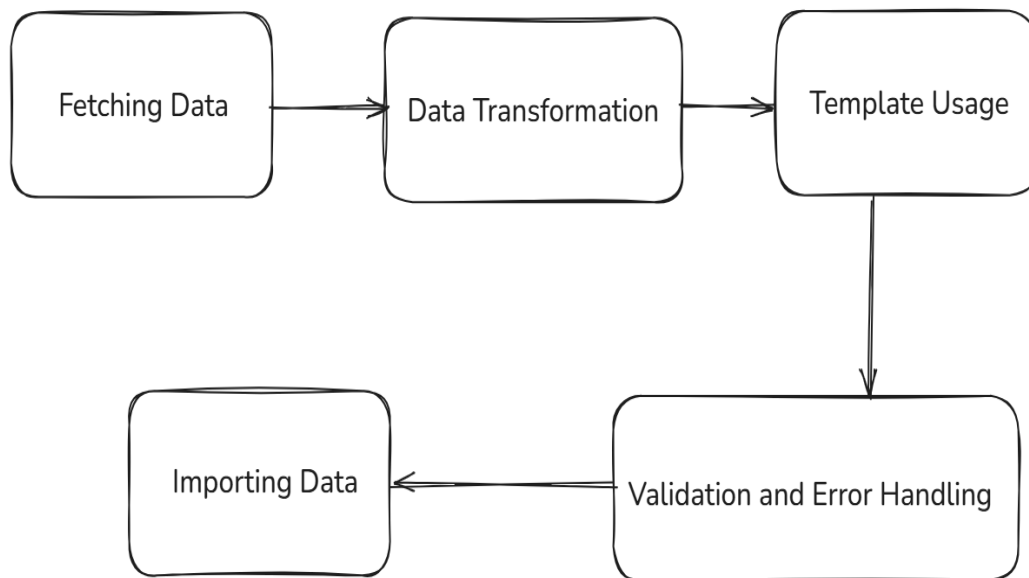
DAY-3 product.ts

```
1 import { defineType } from 'sanity'
2
3 export const product = defineType({
4   name: 'product',
5   title: 'Product',
6   type: 'document',
7   fields: [
8     {
9       name: 'title',
10      title: 'Title',
11      validation: (rule) => rule.required(),
12      type: 'string'
13    },
14    {
15      name: 'description',
16      type: 'text',
17      validation: (rule) => rule.required(),
18      title: 'Description',
19    },
20    {
21      name: 'productImage',
22      type: 'image',
23      validation: (rule) => rule.required(),
24      title: 'Product Image'
25    },
26    {
27      name: 'price',
28      type: 'number',
29      validation: (rule) => rule.required(),
30      title: 'Price',
31    },
32    {
33      name: 'tags',
34      type: 'array',
35      title: 'Tags',
36      of: [{ type: 'string' }]
37    },
38    {
39      name: 'discountPercentage',
40      type: 'number',
41      title: 'Discount Percentage',
42    },
43    {
44      name: 'isLow',
45      type: 'boolean',
46      title: 'Is Low',
47    }
48 ]
49 }
```

Migration steps

Data Migration Method: Using the Provided API

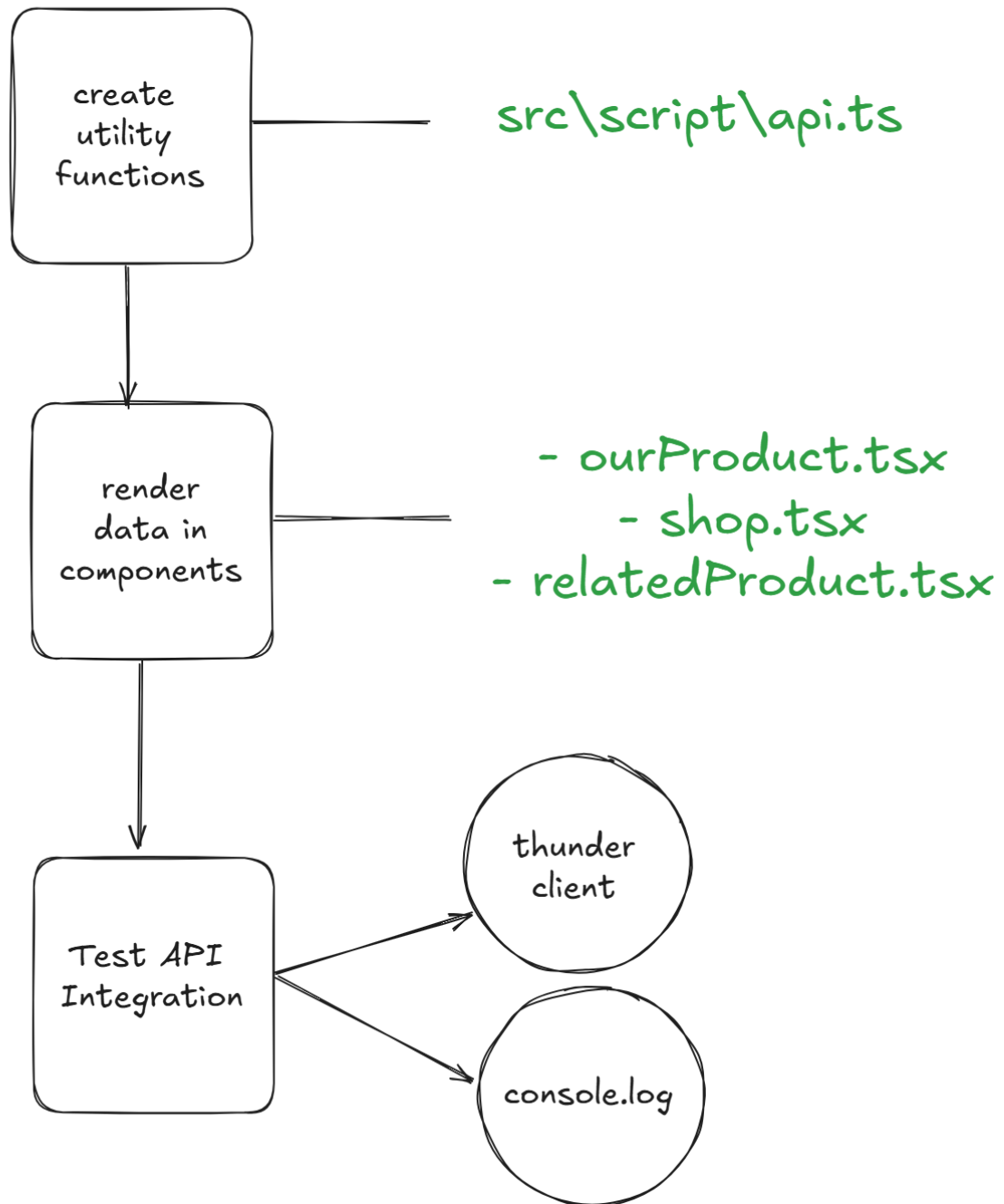
Fetch method is used



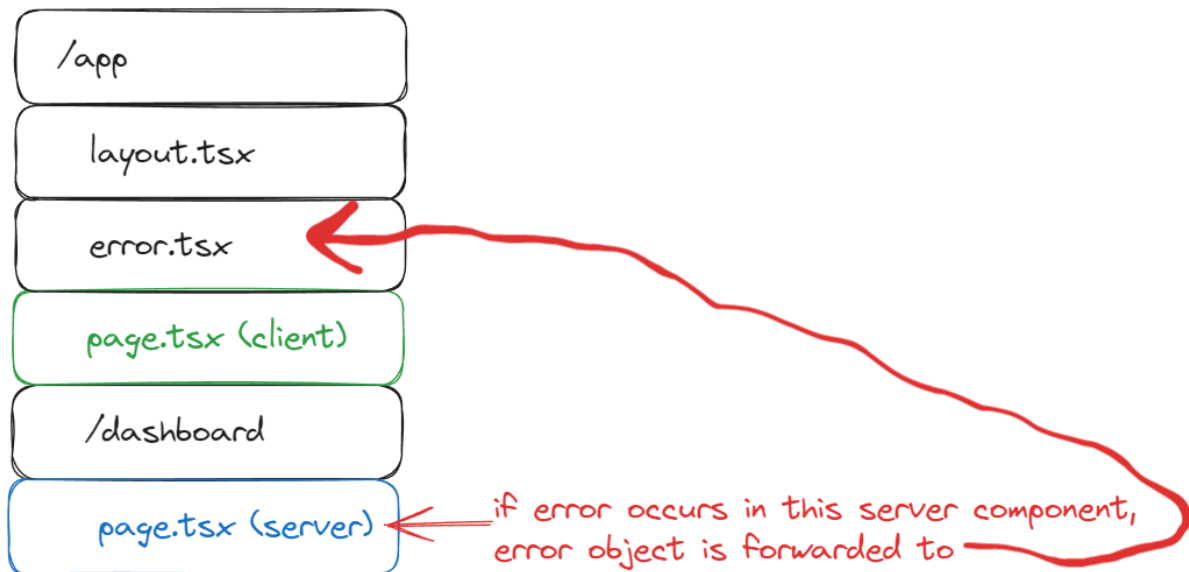
api.ts

```
1 "use server";
2
3 import { client } from "B/sanity/lib/client";
4
5 async function uploadImageToSanity(imageUrl: string) {
6   const response = await fetch(imageUrl);
7   const blob = await response.blob();
8
9   const asset = await client.assets.upload("image", blob);
10  return asset;
11 }
12
13 async function saveProductToSanity(product: any, imageAssetId: string) {
14   const sanityProduct = {
15     _id: `product-${product.id}`,
16     _type: "product",
17     name: product.title,
18     price: product.price,
19     description: product.description,
20     rating: product.rating?.rate || 0,
21     ratingCount: product.rating?.count || 0,
22     discountPercentage: product.discountPercentage || 0,
23     tags: product.category ? [product.category] : [],
24     image: {
25       _type: "image",
26       asset: {
27         _type: "reference",
28         _ref: imageAssetId,
29       },
30     },
31   };
32
33   await client.createOrReplace(sanityProduct);
34   console.log("Imported product: ${sanityProduct.name}");
35 }
36
37 export async function fetchData() {
38   // Check if data already exists in localStorage
39   const localData = localStorage.getItem("products");
40
41   if (localData) {
42     console.log("Using cached data from localStorage");
43     return JSON.parse(localData); // Return the cached data
44   }
45
46   console.log("Fetching data from API...");
47   const response = await fetch("https://template-4-api.vercel.app/api/products");
48   const products = await response.json();
49
50   // Save fetched data to localStorage
51   localStorage.setItem("products", JSON.stringify(products));
52   console.log("Data saved to localStorage");
53
54   const promises = products.map(async (product: any) => {
55     try {
56       const existingProduct = await client.fetch(
57         `*[_type == "product" && _id == ${product.id}]`
58       );
59       if (existingProduct.length > 0) {
60         console.log("Product already exists: ${product.title}");
61         return;
62       }
63
64       const imageAsset = await uploadImageToSanity(product.image);
65       await saveProductToSanity(product, imageAsset._id);
66     } catch (error) {
67       console.error("Failed to process product: ${product.title}, error:", error);
68     }
69   });
70
71   await Promise.all(promises);
72
73   console.log("All products imported successfully!");
74
75   return products; // Return the fetched data
76 }
77
```

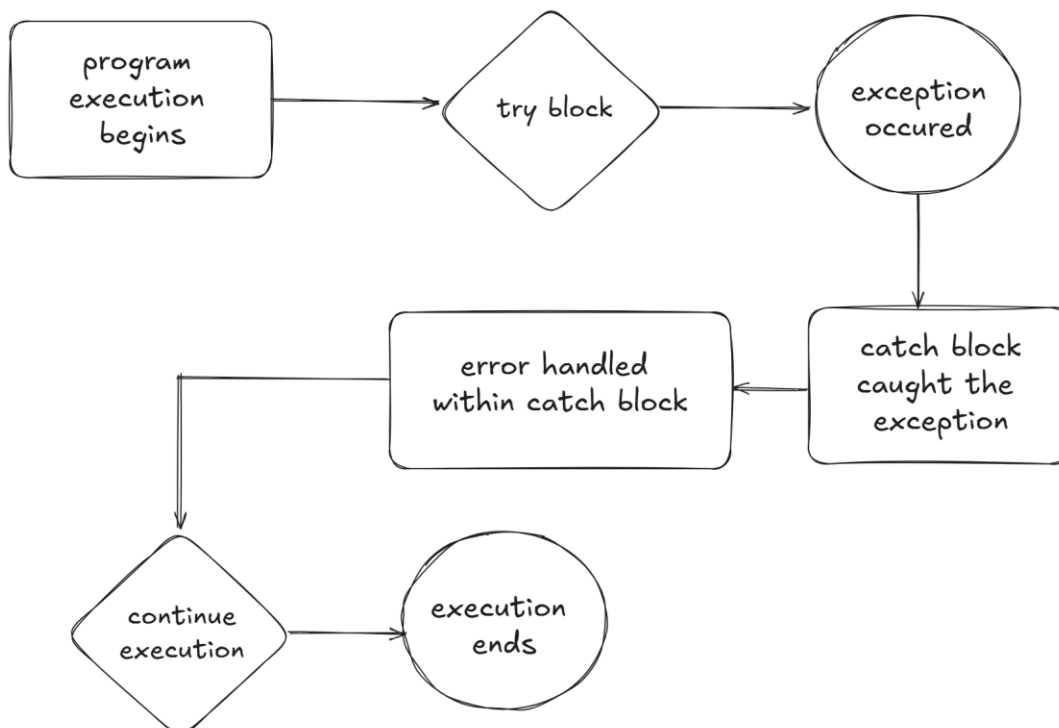
API Integration in Next.js



Error Handling



Error Handling



Checklist for day3

API-understanding



Schema Validation



Data Migration



API Integration in Next.js:



Submission Preparation

