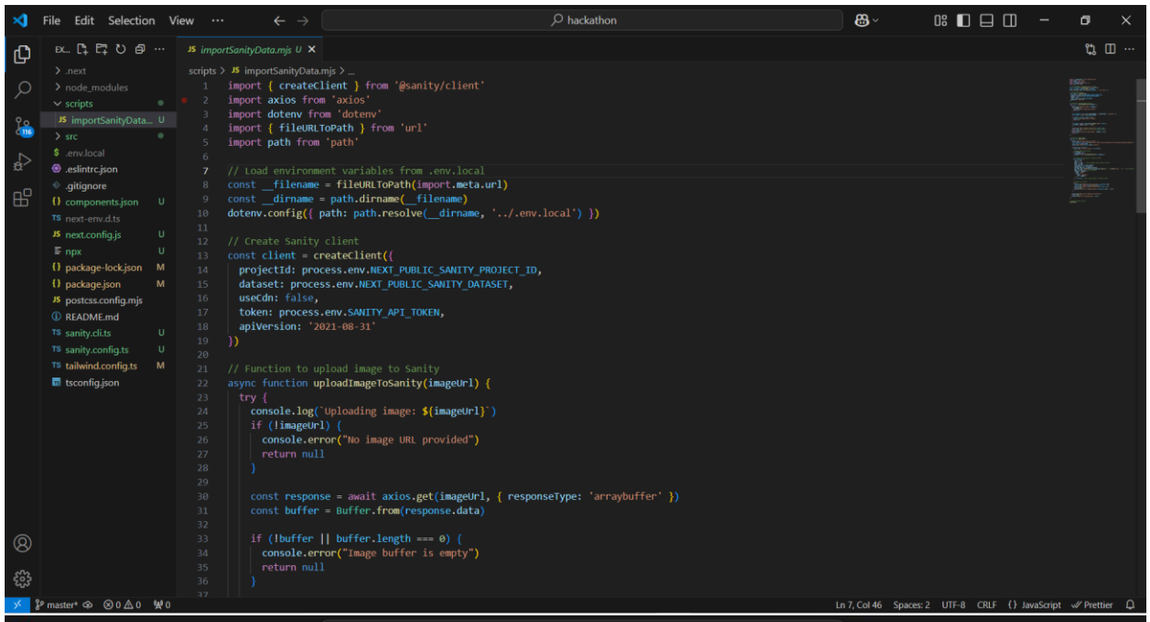
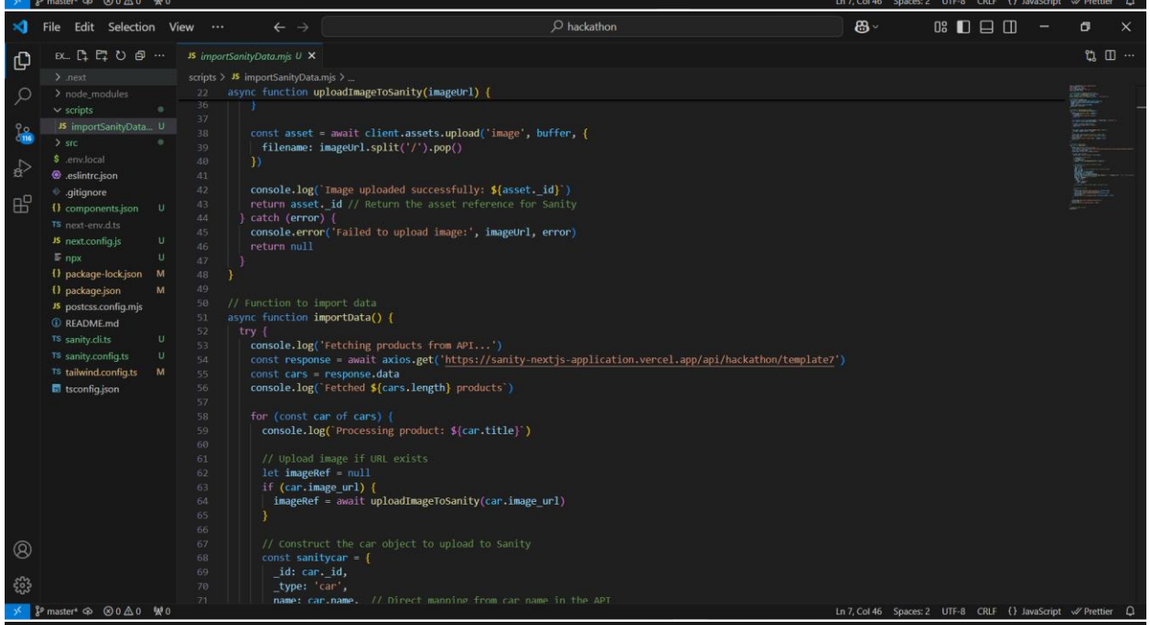
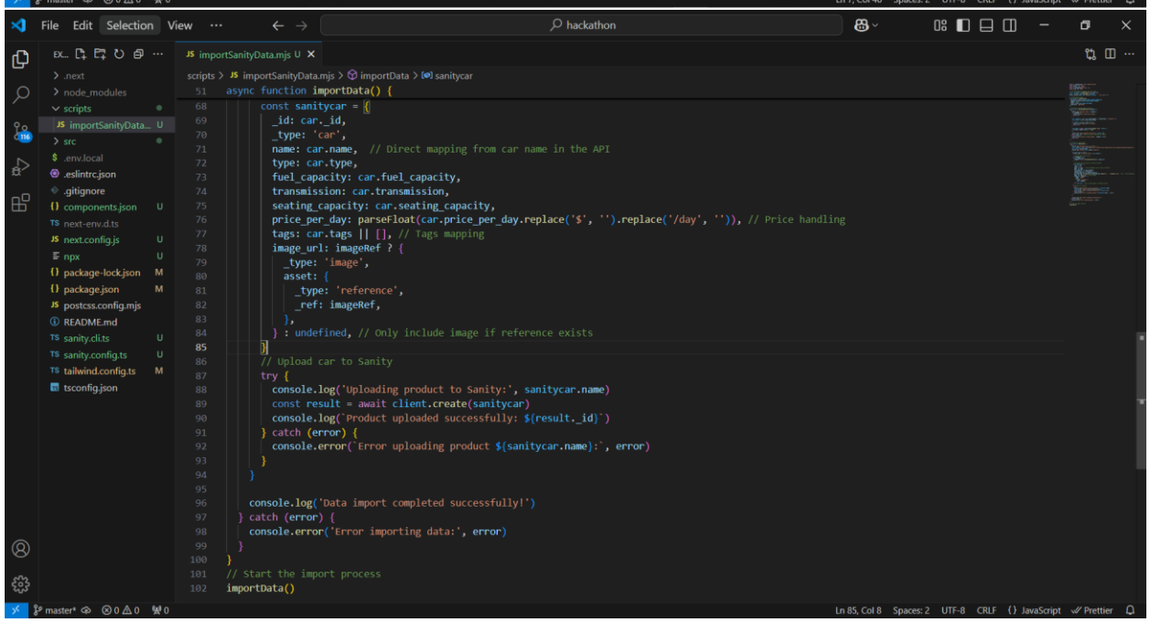

File Edit Selection View
hackathon
importSanityData.mjs U
scripts > importSanityData.mjs > ...
1 import { createClient } from '@sanity/client'
2 import axios from 'axios'
3 import dotenv from 'dotenv'
4 import { fileURLToPath } from 'url'
5 import path from 'path'
7 // Load environment variables from .env.local
8 const __filename = fileURLToPath(import.meta.url)
9 const __dirname = path.dirname(__filename)
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') })
12 // Create Sanity client
13 const client = createClient({
14 projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15 dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16 useCdn: false,
17 token: process.env.SANITY_API_TOKEN,
18 apiVersion: '2021-08-31'
19 })
21 // Function to upload image to Sanity
22 async function uploadImageToSanity(imageUrl) {
23 try {
24 console.log(`Uploading image: ${imageUrl}`)
25 if (!imageUrl) {
26 console.error("No image URL provided")
27 return null
28 }
30 const response = await axios.get(imageUrl, { responseType: 'arraybuffer' })
31 const buffer = Buffer.from(response.data)
33 if (!buffer || buffer.length === 0) {
34 console.error("Image buffer is empty")
35 return null
36 }
master* 0 0 0
Ln 7, Col 46 Spaces: 2 UTF-8 CRLF {} JavaScript Prettier


File Edit Selection View
hackathon
importSanityData.mjs U
scripts > importSanityData.mjs > ...
22 async function uploadImageToSanity(imageUrl) {
38 const asset = await client.assets.upload('image', buffer, {
39 filename: imageUrl.split('/').pop()
40 })
42 console.log(`Image uploaded successfully: ${asset._id}`)
43 return asset._id // Return the asset reference for Sanity
44 } catch (error) {
45 console.error('Failed to upload image:', imageUrl, error)
46 return null
47 }
48 }
50 // Function to import data
51 async function importData() {
52 try {
53 console.log('Fetching products from API...')
54 const response = await axios.get('https://sanity-nextjs-application.vercel.app/api/hackathon/template7')
55 const cars = response.data
56 console.log(`Fetched ${cars.length} products`)
58 for (const car of cars) {
59 console.log(`Processing product: ${car.title}`)
61 // Upload image if URL exists
62 let imageRef = null
63 if (car.image_url) {
64 imageRef = await uploadImageToSanity(car.image_url)
65 }
67 // Construct the car object to upload to Sanity
68 const sanitycar = {
69 _id: car._id,
70 _type: 'car',
master* 0 0 0
Ln 7, Col 46 Spaces: 2 UTF-8 CRLF {} JavaScript Prettier


File Edit Selection View
hackathon
importSanityData.mjs U
scripts > importSanityData.mjs > importData > sanitycar
51 async function importData() {
68 const sanitycar = {
69 _id: car._id,
70 _type: 'car',
71 name: car.name, // Direct mapping from car name in the API
72 type: car.type,
73 fuel_capacity: car.fuel_capacity,
74 transmission: car.transmission,
75 seating_capacity: car.seating_capacity,
76 price_per_day: parseFloat(car.price_per_day.replace('$', '').replace('/day', '')), // Price handling
77 tags: car.tags || [], // Tags mapping
78 image_url: imageRef ? {
79 _type: 'image',
80 asset: {
81 _type: 'reference',
82 _ref: imageRef,
83 },
84 } : undefined, // Only include image if reference exists
85 }
86 // Upload car to Sanity
87 try {
88 console.log('Uploading product to Sanity:', sanitycar.name)
89 const result = await client.create(sanitycar)
90 console.log(`Product uploaded successfully: ${result._id}`)
91 } catch (error) {
92 console.error(`Error uploading product ${sanitycar.name}:`, error)
93 }
94 }
96 console.log('Data import completed successfully!')
97 } catch (error) {
98 console.error('Error importing data:', error)
99 }
100 }
101 // Start the import process
102 importData()
master* 0 0 0
Ln 85, Col 8 Spaces: 2 UTF-8 CRLF {} JavaScript Prettier