

W13-P1: investigate jwt token using jwt.io

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3Mi0iJzdXBhYmFzZSIisInJlZiI6ImVzcXdrZWh1aHhzcGJxc2VrdG9mIiwicm9sZSI6ImFub24iLCJpYXQiOjE2ODI0ODc2MTUsImV4cCI6MTk50DA2MzYxNX0.Fw29YhIJikPQ_ojibIDJfJuGlLzk61iBi9D_VEO-z50|
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "iss": "supabase",  
  "ref": "esqwkehuhxspbqsektof",  
  "role": "anon",  
  "iat": 1682487615,  
  "exp": 1998063615  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-s0Zg2gc/;  
)  secret base64 encoded
```

⌚ Signature Verified

SHARE JWT



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImVzcXdrZWh1aHhcGJxc2VrdG9mIiwicm9sZSI6InNlcnZpY2Vfcm9sZSIsImhdCI6MTY4MjQ4NzYxNSwiZXhwIjoxOTk4MDYzMjE1fQ.c8iiPIhDcE4GZ8SrJPItpk7YG60ergiTb-CHuScrxJc
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "iss": "supabase",  
  "ref": "esqwkehuhxspbgsektof",  
  "role": "service_role",  
  "iat": 1682487615,  
  "exp": 1998063615  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-2560Zg2gc/z7UIhFl  
)  secret base64 encoded
```

Signature Verified

[SHARE JWT](#)

W13-P2: Get all products from shop2_xx (RLS not enabled), and from shop_xx (RLS enabled)

Get all products from shop2_xx (RLS not enabled)

The screenshot shows the Postman interface with a successful API call to `https://esqkuehuhxspbksektf.supabase.co/rest/v1/shop2_19?select=*`. The response body contains two product items:

```
268     "name": "Jean Long Sleeve",
269     "cat_id": 5,
270     "price": 40,
271     "remote_url": "https://i.ibb.co/Vpw4x5t/roll-up-jean-shirt.png",
272     "local_url": "/img/mens/roll-up-jean-shirt.png"
273 },
274
275     "id": 40,
276     "name": "Burgundy T-shirt",
277     "cat_id": 5,
278     "price": 25,
279     "remote_url": "https://i.ibb.co/mh3VM1f/polka-dot-shirt.png",
280     "local_url": "/img/mens/polka-dot-shirt.png"
281 }
282 }
```

Get all products from shop_xx (RLS enabled)

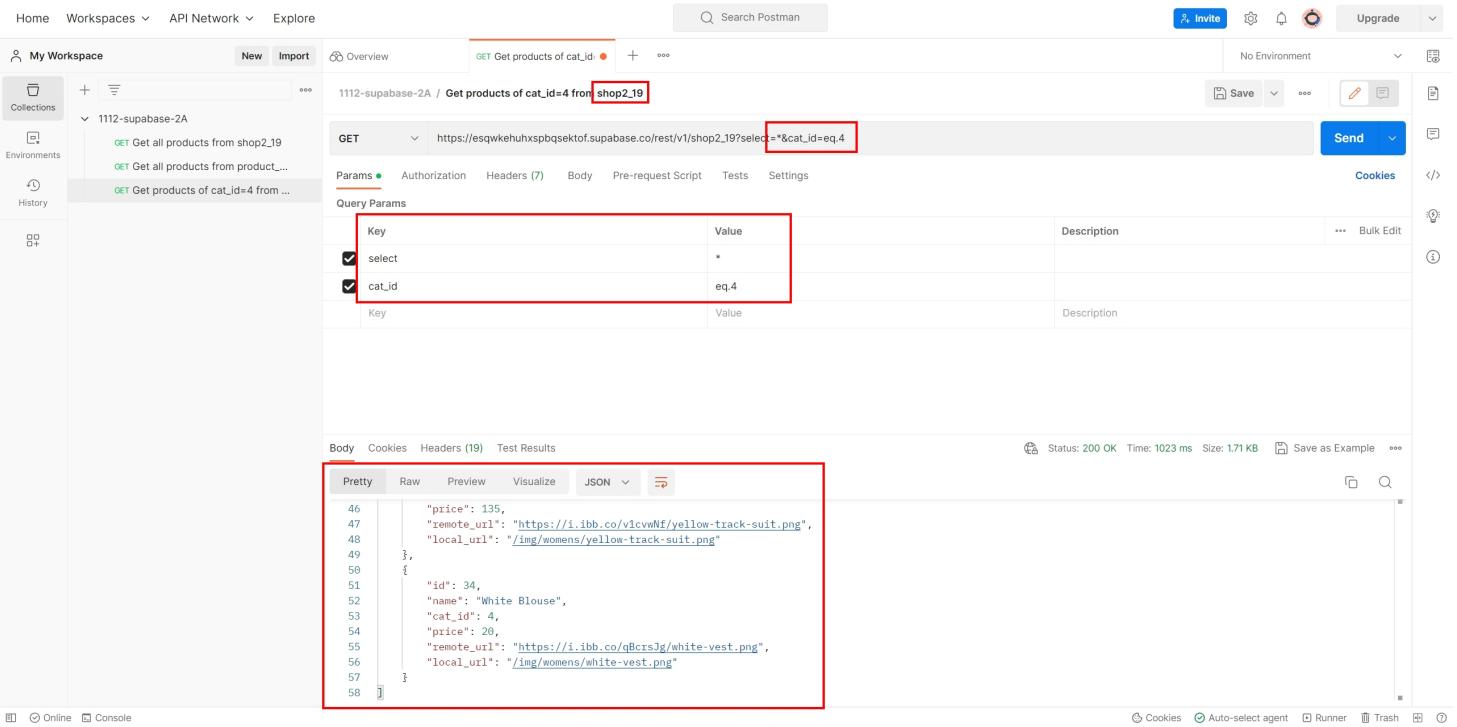
The screenshot shows the Postman interface with a successful API call to `https://esqkuehuhxspbksektf.supabase.co/rest/v1/product_xx?select=*`. The response body contains three product items:

```
42     "img_url": "/img/hats/blue-beanie.png"
43 },
44 {
45     "pid": 7,
46     "pname": "Brown Cowboy",
47     "cat_id": 1,
48     "price": 35,
49     "img_url": "/img/hats/brown-cowboy.png"
50 },
51 {
52     "pid": 8,
53     "pname": "Grey Brim",
54     "cat_id": 1,
55     "price": 25,
56     "img_url": "/img/hats/grey-brim.png"
57 }
58 }
```

A red box highlights the "secret" key in the JWT Settings panel, which is described as having the ability to bypass Row Level Security.

W13-P3: Get products of cat_id=4 from shop2_xx (RLS not enabled), and from shop_xx (RLS enabled)

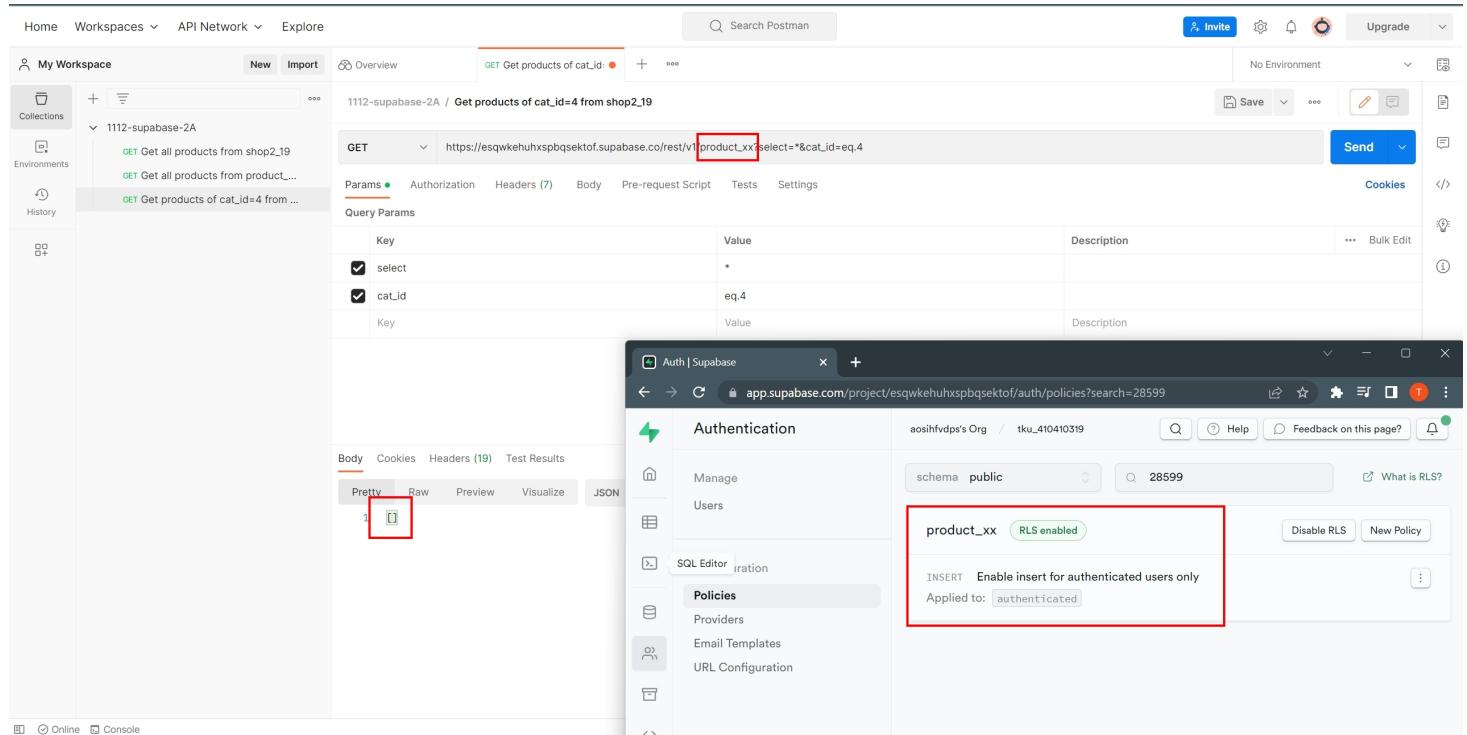
Get products of cat_id=4 from shop2_xx (RLS not enabled)



The screenshot shows the Postman interface with a collection named "1112-supabase-2A". A GET request is made to the endpoint `https://esqwkeuhxspbqsekto.supabase.co/rest/v1/shop2_19?select=*&cat_id=eq.4`. The "Query Params" table is highlighted with a red box, showing "select" with value "*" and "cat_id" with value "eq.4". The "Body" tab shows the JSON response:

```
46   "price": 135,
47   "remote_url": "https://i.ibb.co/v1cvwNf/yellow-track-suit.png",
48   "local_url": "/img/womens/yellow-track-suit.png"
49 },
50 {
51   "id": 34,
52   "name": "White Blouse",
53   "cat_id": 4,
54   "price": 20,
55   "remote_url": "https://i.ibb.co/qBcrsJg/white-vest.png",
56   "local_url": "/img/womens/white-vest.png"
57 }
```

Get products cat_id=4 from shop2_xx (RLS not enabled)



The screenshot shows the Postman interface with the same collection and endpoint. The "Query Params" table is highlighted with a red box, showing "select" with value "*" and "cat_id" with value "eq.4". The "Body" tab shows an empty JSON response.

An inset window titled "Auth | Supabase" shows the Supabase Auth Policies page. A policy for the "product_xx" schema is selected, labeled "RLS enabled". The policy details are:

```
INSERT Enable insert for authenticated users only
Applied to: authenticated
```

Get products cat_id=4 from shop_xx (RLS enabled)

The screenshot shows a Postman workspace with a collection named '1112-supabase-2A'. A GET request is made to the URL `https://esqwkkehuhxspbqsektof.supabase.co/rest/v1/product_xx?select=*&cat_id=4`. The 'Auth' tab is selected, showing 'apikey' and 'Authorization' headers. The Authorization header contains a JWT token. The response body is displayed in JSON format, showing two products:

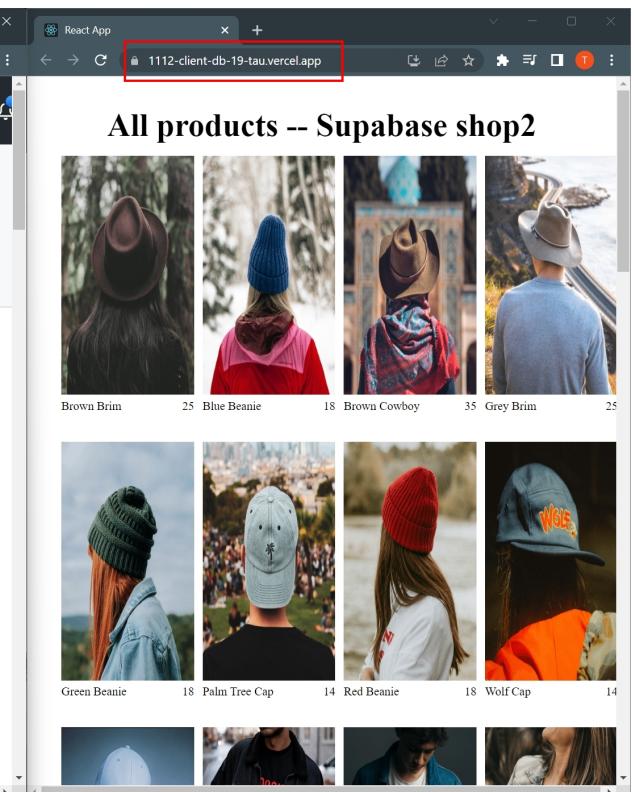
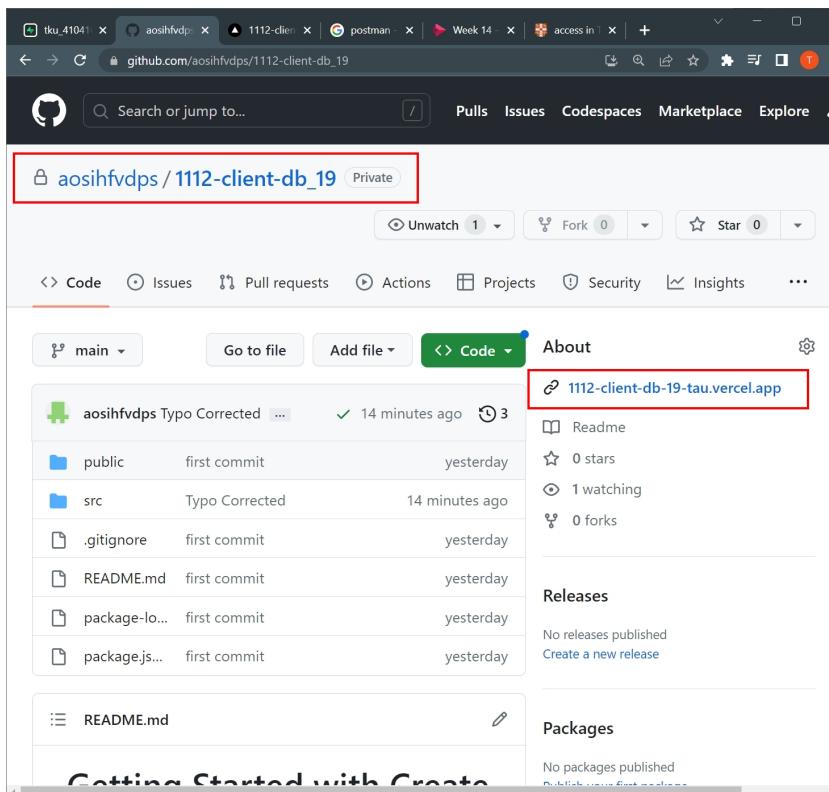
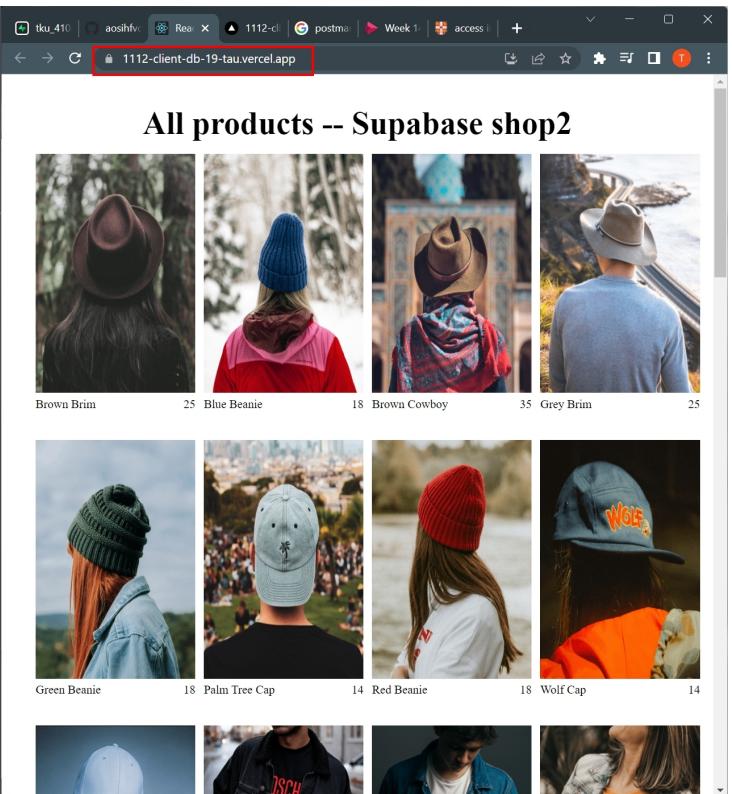
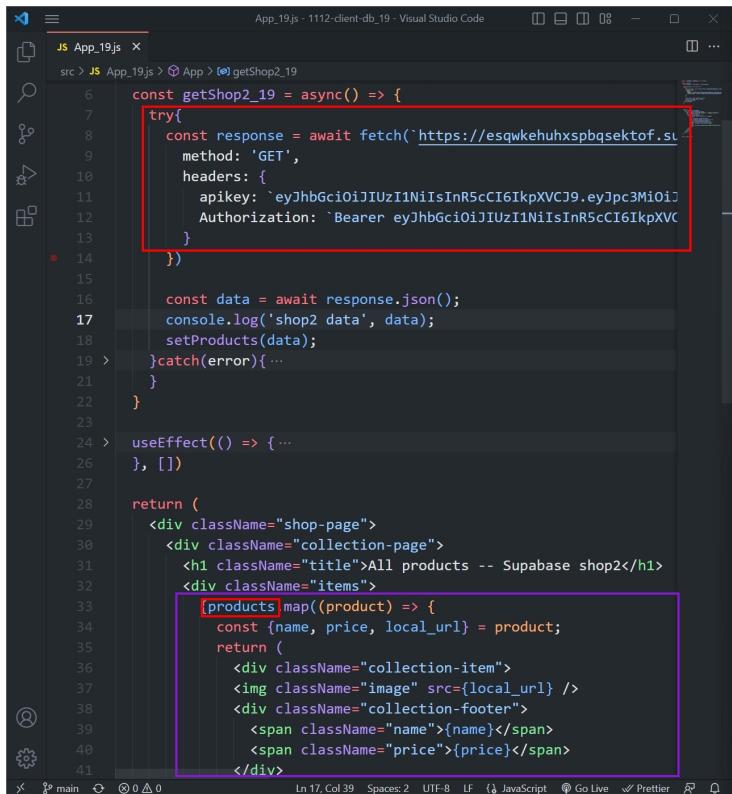
```
18:   "pname": "Floral Dress",
19:   "cat_id": 4,
20:   "price": 80,
21:   "img_url": "/img/womens/floral-skirt.png"
22: },
23: {
24:   "pid": 4,
25:   "pname": "Red Dots Dress",
26:   "cat_id": 4,
27:   "price": 80,
28:   "img_url": "/img/womens/red-polka-dot-dress.png"
29: }
```

My Github Repo URL

My Client DB Repo URL

My Vercel URL

W14-P1: show all shop2_xx products in Vercel



W14-P2: Get all products from Supabase shop2_xx, show in Vercel

The screenshot shows the development environment for a Next.js application. On the left, the code editor displays two files: `App_19.js` and `Shop2Page_19.js`. `App_19.js` contains the main application routing logic, including a `BrowserRouter` component that maps static routes to components like `WomensProductStaticPage_19` and `Shop2Page_19`. `Shop2Page_19.js` is highlighted with a red box and contains the logic for fetching products from a Supabase database. On the right, a browser window shows the deployed application at `1112-client-db-19-tau.vercel.app/supa.shop2_19`. The page title is "All products -- Supabase shop2". Below the title, there are four product cards: "Brown Brim" (25), "Blue Beanie" (18), "Brown Cowboy" (35), and "Grey Brim" (25). Further down the page, there are more product cards: "Green Beanie" (18), "Palm Tree Cap" (14), "Red Beanie" (18), and "Wolf Cap" (14).

W15-P1: return json for route /api/crown2_xx

The screenshot shows the development environment for a Node.js application. On the left, the code editor displays `crown2_19.js`, which is part of a larger project structure. The file contains an Express router configuration. A specific section of the code, where the router handles the root path ('/'), is highlighted with a red box. It uses `db.query` to select data from a database and then `JSON.stringify` to convert the results into a JSON object. The `res.json(results.rows)` line is also highlighted. On the right, a browser window shows the JSON response for the `/api/crown2_19` route. The response is a list of three categories: "hats", "jackets", and "sneakers". Each category object includes properties like `name`, `size`, `remote_url`, `local_url`, and `link_url`.

```

1 // 20230526151418
2 // http://localhost:3000/api/crown2_19
3
4 [
5   {
6     "id": 1,
7     "name": "hats",
8     "size": null,
9     "remote_url": "https://i.ibb.co/cvpnL1/hats.png",
10    "local_url": "/img/homepage/hats.png",
11    "link_url": "/crown2_19/shop2_19/hats"
12  },
13  {
14    "id": 2,
15    "name": "jackets",
16    "size": null,
17    "remote_url": "https://i.ibb.co/px2tCc3/jackets.png",
18    "local_url": "/img/homepage/jackets.png",
19    "link_url": "/crown2_19/shop2_19/jackets"
20  },
21  {
22    "id": 3,
23    "name": "sneakers",
24    "size": null,
25    "remote_url": "https://i.ibb.co/@qgHppn/sneakers.png",
26    "local_url": "/img/homepage/sneakers.png",
27    "link_url": "/crown2_19/shop2_19/sneakers"
28  }
]

```

W15-P2: return json for route /api/crown2_xx/shop2_xx

The screenshot shows the Visual Studio Code interface with two panes. The left pane displays the file `crown2_19.js` containing the code for the `/shop2_19` route. The right pane shows the browser output at `localhost:3000/api/crown2_19/shop2_19`, displaying a JSON array of products.

```

    routes.get('/shop2_19', async function (req, res) {
      try {
        let results = await db.query('select * from shop2_19');
        res.json(results.rows);
      } catch (error) {
        console.log(error);
      }
    });

    module.exports = router;
  
```

```

[{"id": 38, "name": "Pink T-shirt", "cat_id": 5, "price": 25, "remote_url": "https://i.ibb.co/55z32tw/long-sleeve.png", "local_url": "/img/mens/long-sleeve.png"}, {"id": 39, "name": "Jean Long Sleeve", "cat_id": 5, "price": 40, "remote_url": "https://i.ibb.co/vpW4x5t/roll-up-jean-shirt.png", "local_url": "/img/mens/roll-up-jean-shirt.png"}, {"id": 40, "name": "Burgundy T-shirt", "cat_id": 5, "price": 25, "remote_url": "https://i.ibb.co/mh3VM1f/polka-dot-shirt.png", "local_url": "/img/mens/polka-dot-shirt.png"}]
  
```

W15-P3: From React client, implement route /node_shop2_xx to get data from node server using route /api/crown2_xx/shop2_xx

The screenshot shows the Visual Studio Code interface with two panes. The left pane displays the file `ShopPage_19.js` containing the code for the `/node_shop2_xx` route. The right pane shows the browser output at `localhost:3000/node_shop2_19`, displaying a grid of product images with their names and prices.

```

    const getShop2_19 = async() => {
      try{
        const response = await fetch(`http://localhost:5000/api/crown2_19/shop2_19`)
        const data = await response.json();
        console.log('shop2 data', data);
        setProducts(data);
      }catch(error){
        console.log(error);
      }
    }

    useEffect(() => {
      getShop2_19();
    }, [])
  
```

Image	Name	Price
	Brown Brim	25
	Blue Beanie	18
	Brown Cowboy	35
	Grey Brim	35
	Green Beanie	18
	Palm Tree Cap	14
	Red Beanie	18
	Wolf Cap	18

```
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-05-01"
fbbecbe aosihfvdps      Fri May 26 22:01:18 2023 +0800  corc added
b0a80f7 aosihfvdps      Sat May 20 15:26:36 2023 +0800  w14 finished
cd51e9b aosihfvdps      Sun May 14 23:53:35 2023 +0800  May 15 2023, finished week 13
1423aff aosihfvdps      Sat May 6 12:31:30 2023 +0800  Today is May 6th, 12:31, homework w11-12 has been finishe
c9d69b9 aosihfvdps      Sat May 6 12:26:29 2023 +0800  Today is May 6th, 12:25, homework w11-12 has been finishe
9a4a124 aosihfvdps      Thu May 4 18:01:52 2023 +0800  May 4th after class
f300744 aosihfvdps      Thu May 4 17:27:41 2023 +0800  today is May 4th, 17:27
d4e6a78 aosihfvdps      Thu May 4 16:19:15 2023 +0800  today is May 4th, second time before the class
340380b aosihfvdps      Thu May 4 16:06:42 2023 +0800  today is May 4th, before the class
```