

My Github Repo URL

## W13-P1: investigate jwt token using jwt.io

The screenshot shows the jwt.io interface with a valid JWT token decoded. The token is pasted into the 'Encoded' field:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3Mi0iJzdXBhYmFzZSIsInJlZiI6ImppZXdodHRrdHVzdm12Y31xbmtpIiwicm9sZSI6ImFub24iLCJpYXQiOjE2ODI0Dc5MTAsImV4cCI6MTk50DA2MzkxMH0.1KLlyU1Y9CF6jxbmnfgAuGNdufQ3TTXdnB1oPGOHMOA
```

The 'Decoded' section shows the structure of the JWT:

**HEADER: ALGORITHM & TOKEN TYPE**

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

**PAYOUT: DATA**

```
{  
  "iss": "sunahase",  
  "ref": "jiewhttktusvivcyqnki",  
  "role": "anon",  
  "iat": 1682487910,  
  "exp": 1998063910  
}
```

**VERIFY SIGNATURE**

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  eyJhbGciOiJIUzI1NiIsI  
) □ secret base64 encoded
```

A green box highlights the 'Signature Verified' message at the bottom left.

The screenshot shows the jwt.io interface with another valid JWT token decoded. The token is pasted into the 'Encoded' field:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3Mi0iJzdXBhYmFzZSIsInJlZiI6ImppZXdodHRrdHVzdm12Y31xbmtpIiwicm9sZSI6InNlcnZpY2VfcmsZSI6ImhdCI6MTY4MjQ4NzKxMCwiZXhwIjoxOTk4MDYzOTEwfQ.nKJxbEBE8gDFqEmxQyFHS-HzWGQxI4ER6qcnJhwipPgk
```

The 'Decoded' section shows the structure of the JWT:

**HEADER: ALGORITHM & TOKEN TYPE**

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

**PAYOUT: DATA**

```
{  
  "iss": "supabase",  
  "ref": "jiewhttktusvivcyqnki",  
  "role": "service_role",  
  "iat": 1682487910,  
  "exp": 1998063910  
}
```

**VERIFY SIGNATURE**

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  SIS5PIdpjgNj8tQYEclFSvi  
) □ secret base64 encoded
```

A green box highlights the 'Signature Verified' message at the bottom left.

# 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 two windows side-by-side. On the left is the Postman interface with a request to 'https://jiewhtktusvivcyqnki.supabase.co/rest/v1/shop2\_28?select=\*'. The 'Headers' tab shows 'apikey' and 'Authorization' headers. The 'Body' tab shows a JSON response with two products:

```
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 }
```

On the right is a browser window with the URL 'https://jiewhtktusvivcyqnki.supabase.co/rest/v1/shop2\_28?select=\*'. The 'Bash' tab shows the curl command used to make the request. The 'Response' tab shows the same JSON response as the Postman body.

## Get all products from shop\_xx (RLS enabled)

The screenshot shows two windows side-by-side. On the left is the Postman interface with a request to 'https://jiewhtktusvivcyqnki.supabase.co/rest/v1/shop2\_28?select=\*'. The 'Headers' tab shows 'apikey' and 'Authorization' headers. The 'Body' tab shows a JSON response with two products:

```
274   {
275     "pId": 40,
276     "pName": "Burgundy T-shirt",
277     "cat_id": 5,
278     "price": 25,
279     "remote_url": "https://i.ibb.co/mh3VM1f/polka-dot-shirt.png",
280   }
281 }
```

On the right is a browser window with the URL 'https://jiewhtktusvivcyqnki.supabase.co/rest/v1/shop2\_28?select=\*'. The 'anon' key is selected in the 'Project API key' dropdown. The 'Response' tab shows the same JSON response as the Postman body.

# 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 specific request titled "Get products of cat\_id=3 from shop2\_28" is selected. The URL is https://jiewhtktusvivcyqnki.supabase.co/rest/v1/shop2\_28?select=\*&cat\_id=eq.3. The "Query Params" table shows two entries: "select" with value "\*" and "cat\_id" with value "eq.3". The "Body" tab displays a JSON response object with one item, representing a product with ID 27, name "Timberlands", category ID 3, price 200, and URLs for remote and local images.

## Get products cat\_id=4 from shop2\_xx (RLS not enabled)

The screenshot shows the Postman interface with the same setup as the previous screenshot. The "shop2\_28" table in the Supabase Authentication console has RLS enabled. The "Policies" section indicates "No policies created yet" and "RLS is enabled - create a policy to allow access to this table." The Postman response body is empty, showing only the number 1.

## Get products cat\_id=4 from shop\_xx (RLS enabled)

The screenshot shows two windows side-by-side. On the left is Postman, displaying a GET request to `https://jiewhttktusvivcyqnki.supabase.co/rest/v1/shop_28?select=*&cat_id=eq.3`. The Headers tab shows two entries: `apikey` with value `eyJhbGciOiJIUzI1NilsInR5cI6IkpxVCJ9.ey...` and `Authorization` with value `Bearer eyJhbGciOiJIUzI1NilsInR5cI6Ikpx...`. The Body tab shows a JSON response with one product item:

```
58: {  
59:   "pId": 27,  
60:   "pName": "Timberlands",  
61:   "cat_id": 3,  
62:   "price": 200,  
63:   "remote_url": "https://i.ibb.co/Mhh6wBg/timberlands.png",  
64:   "local_url": "/img/sneakers/timberlands.png"  
65:  
66: }
```

On the right is the jwt.io verifier, showing a decoded JWT with the following payload fields:

```
{  
  "iss": "supabase",  
  "ref": "jiewhttktusvivcyqnki",  
  "role": "service_role",  
  "iat": 1682487910,  
  "exp": 1998063910  
}
```

The VERIFY SIGNATURE section contains the HMACSHA256 formula and placeholder values for header, payload, and secret.

## Github logs of Week 13

```
User@LAPTOP-FDE6DRV T MINGW64 /d/database/1112-2A-db-demo-410410228 (main)  
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-9"  
68b80817      Daineair      Wed May 24 15:02:48 2023 +0800 w15 0524  
1e60f939      Daineair      Thu May 18 17:07:20 2023 +0800 w14 0518  
f9f93edf      Daineair      Wed May 17 17:43:53 2023 +0800 w14 0517  
6f430bfd      Daineair      Wed May 10 18:27:19 2023 +0800 w13 finish 0510  
c5d949f8      Daineair      Wed May 10 15:03:48 2023 +0800 w13 0510 p2 not finish
```

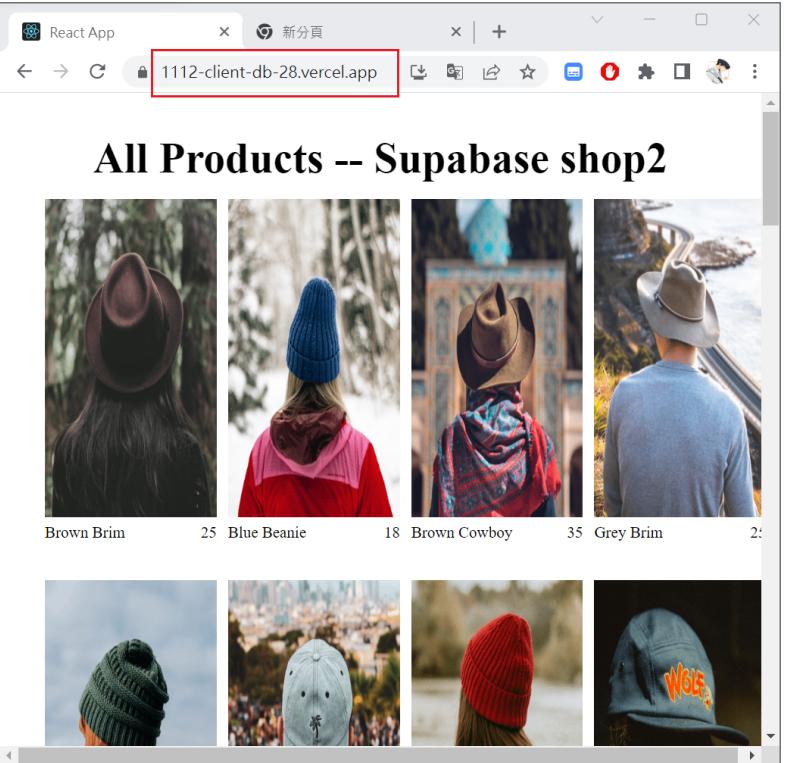
```
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-9"  
6f430bfd      Daineair      Wed May 10 18:27:19 2023 +0800 w13 finish 0510  
c5d949f8      Daineair      Wed May 10 15:03:48 2023 +0800 w13 0510 p2 not finish
```

My Github Repo URL

My Client DB Repo URL

My Vercel URL

# W14-P1: show all shop2\_xx products in Vercel



The screenshot shows a browser window displaying the URL [1112-client-db-28.vercel.app](https://1112-client-db-28.vercel.app). The page title is "All Products -- Supabase shop2". Below the title, there are four product cards:

- Brown Brim: A person wearing a brown fedora hat.
- Blue Beanie: A person wearing a blue knit beanie.
- Brown Cowboy: A person wearing a brown cowboy hat with a red bandana.
- Grey Brim: A person wearing a light grey fedora hat.

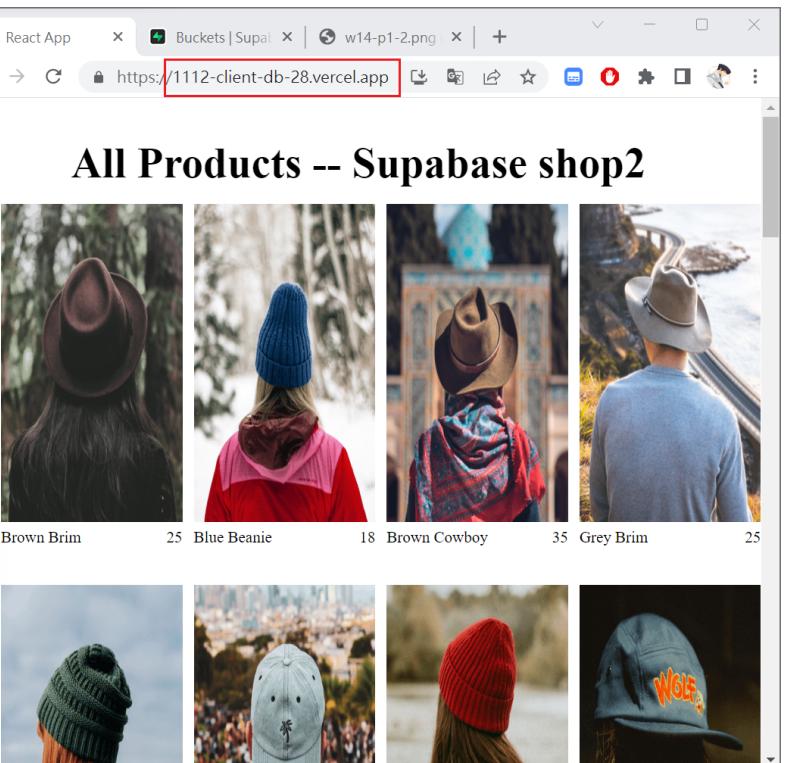
Below each card, the name and price are listed: Brown Brim (25), Blue Beanie (18), Brown Cowboy (35), and Grey Brim (25). The browser's address bar is highlighted with a red box.

On the left, a screenshot of the Visual Studio Code editor shows the code for the `App.js` file. A red box highlights the fetch request to the Supabase API endpoint `https://jiewhtktusivvcyqnnki.supabase.co/rest/v1/shop2_28?select=*`.

```
const getShop2_28 = async () => {
  try {
    const response = await fetch('https://jiewhtktusivvcyqnnki.supabase.co/rest/v1/shop2_28?select=*');
    const data = await response.json();
    console.log('shop2 data', data);
    setProducts(data);
  } catch(error) {
    console.error(error);
  }
}

useEffect(()=>{
  getShop2_28();
}, [ ]);

return (
  <div className="shop-page">
    <div className="collection-page">
      <h1 className="title">All Products -- Supabase shop2</h1>
      <div className="items">
        {products.map((product) => {
          const {name, price, local_url} = product;
          return (
            <div className="collection-item">
              <div className="image" style={{backgroundImage: `url(${local_url})`}}></div>
              <div className="collection-footer">
                <span className="name">{name}</span>
                <span className="price">{price}</span>
              </div>
              <button className="custom-button">Add to Cart</button>
            </div>
          );
        })}
      </div>
    </div>
  </div>
)
```



The screenshot shows a GitHub repository page for [Dainair / 1112-client-db\\_28](https://github.com/Dainair/1112-client-db_28). The repository is private and has 0 stars and 0 forks. It contains a single commit from "Dainair w14 0517" made 2 hours ago. The commit message is "2 hours ago".

On the right, a screenshot of a browser window shows the same "All Products -- Supabase shop2" page as the first screenshot, but with different product images and prices:

- Brown Brim: A person wearing a brown fedora hat.
- Blue Beanie: A person wearing a blue knit beanie.
- Brown Cowboy: A person wearing a brown cowboy hat with a red bandana.
- Grey Brim: A person wearing a light grey fedora hat.

Below each card, the name and price are listed: Brown Brim (25), Blue Beanie (18), Brown Cowboy (35), and Grey Brim (25). The browser's address bar is highlighted with a red box.

# W14-P2: Get all products from Supabase shop2\_xx, show in Vercel

The screenshot shows a developer's environment with two main windows. On the left is Visual Studio Code (VS Code) displaying the file `Shop2Page_28.js`. The code defines an `App_28` component using `react-router-dom` to handle routes for static pages and the Supabase shop. It includes a fetch call to get shop data from Supabase. On the right is a browser window showing the deployed application at `1112-client-db-28.vercel.app/supa_shop2_28`. The page title is "All Products -- Supabase shop2". Below the title are eight product cards, each showing a person wearing a different type of hat (Brown Brim, Blue Beanie, Brown Cowboy, Grey Brim, Green Beanie, Palm Tree Cap, Red Beanie, Wolf Cap) with their respective prices (25, 18, 35, 25, 18, 14, 14, 18).

## Github logs of Week 14

```
User@LAPTOP-FDE6DRV\ MINGW64 /d/database/1112-2A-db-demo-410410228 (main)
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-16"
68b80817      Daineair      Wed May 24 15:02:48 2023 +0800 w15 0524
1e60f939      Daineair      Thu May 18 17:07:20 2023 +0800 w14 0518
f9f93edf      Daineair      Wed May 17 17:43:53 2023 +0800 w14 0517
```

```
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-16"
1e60f939      Daineair      Thu May 18 17:07:20 2023 +0800 w14 0518
f9f93edf      Daineair      Wed May 17 17:43:53 2023 +0800 w14 0517
```

[My Github Repo URL](#)

## W15-P1: return json for route /api/crown2\_xx

The screenshot shows the Visual Studio Code interface with two panes. The left pane displays the file `crown2_28.js` containing Node.js code for a GET route. The right pane shows the resulting JSON response sent to the client.

**Code (crown2\_28.js):**

```
routes > api > JS crown2_28.js > router.get('/').callback
1 var express = require('express');
2 var router = express.Router();
3
4 let db = require('../utils/database');
5
6 router.get('/', async function(req, res, next){
7   try {
8     let results = await db.query('select * from category_28');
9     console.log('category data', JSON.stringify(results.rows));
10    res.json(results.rows);
11  } catch(error) {
12    console.log(error);
13  }
14 });
15
16 module.exports = router;
```

**Terminal:**

```
Node.js v18.13.0
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting `node ./bin/www`
```

**JSON Response (localhost:3000/api/crown2\_28):**

```
[{"id": 1, "name": "hats", "size": null, "remote_url": "https://i.ibb.co/cvpnL1/hats.png", "local_url": "/img/homepage/hats.png", "link_url": "/crown2_xx/shop_xx/hats"}, {"id": 2, "name": "jackets", "size": null, "remote_url": "https://i.ibb.co/px2tCc3/jackets.png", "local_url": "/img/homepage/jackets.png", "link_url": "/crown2_xx/shop_xx/jackets"}, {"id": 3, "name": "sneakers", "size": null, "remote_url": "https://i.ibb.co/0jgHpn/p/sneakers.png", "local_url": "/img/homepage/sneakers.png", "link_url": "/crown2_xx/shop_xx/sneakers"}, {"id": 4, "name": "womens", "size": "large", "remote_url": "https://i.ibb.co/GCCdy8t/womens.png", "local_url": "/img/homepage/womens.png", "link_url": "/crown2_xx/shop_xx/womens"}, {"id": 5, "name": "mens", "size": "medium", "remote_url": "https://i.ibb.co/55z32tw/long-sleeve.png", "local_url": "/img/mens/long-sleeve.png", "link_url": "/crown2_xx/shop_xx/mens"}, {"id": 6, "name": "mens", "size": "large", "remote_url": "https://i.ibb.co/RvwnBL8/pink-shirt.png", "local_url": "/img/mens/pink-shirt.png", "link_url": "/crown2_xx/shop_xx/mens"}, {"id": 7, "name": "mens", "size": "medium", "remote_url": "https://i.ibb.co/VpW4x5t/roll-up-jean-shirt.png", "local_url": "/img/mens/roll-up-jean-shirt.png", "link_url": "/crown2_xx/shop_xx/mens"}, {"id": 8, "name": "mens", "size": "small", "remote_url": "https://i.ibb.co/mh3VM1f/polka-dot-shirt.png", "local_url": "/img/mens/polka-dot-shirt.png", "link_url": "/crown2_xx/shop_xx/mens"}]
```

## 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_28.js` containing Node.js code for a GET route. The right pane shows the resulting JSON response sent to the client.

**Code (crown2\_28.js):**

```
routes > api > JS crown2_28.js > router.get('/shop2_28').callback
1 let db = require('../utils/database');
2
3 router.get('/', async function(req, res, next){
4   try {
5     let results = await db.query('select * from category_28');
6     console.log('category data', JSON.stringify(results.rows));
7     res.json(results.rows);
8   } catch(error) {
9     console.log(error);
10  }
11 });
12
13 router.get('/shop2_28', async function(req, res, next){
14   try {
15     let results = await db.query('select * from shop2_28');
16     console.log('shop2 data', JSON.stringify(results.rows));
17     res.json(results.rows);
18   } catch(error) {
19     console.log(error);
20   }
21 });
22
23 module.exports = router;
```

**Terminal:**

```
: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"}]
```

**GET /api/crown2\_28/shop2\_28 200 42.802 ms - 5484**

**JSON Response (localhost:3000/api/crown2\_28/shop2\_28):**

```
[{"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

All Products -- Node shop2

Brown Brim 25 Blue Beanie 18 Brown Cowboy 35 Grey Brim 25

Green Beanie 18 Palm Tree Cap 14 Red Beanie 18 Wolf Cap 14

Screenshot of Visual Studio Code showing the file Shop2NodePage\_28.js. The code uses useState and useEffect hooks to fetch data from 'http://localhost:5000/api/crown2\_28/shop2\_28'. It also includes a warning about img elements missing alt text.

```
import { useState, useEffect } from 'react';
const Shop2NodePage_28 = () => {
  const [products, setProducts] = useState([]);
  const getShop2_28 = async () => {
    try {
      const response = await fetch('http://localhost:5000/api/crown2_28/shop2_28');
      const data = await response.json();
      console.log('shop2 data', data);
      setProducts(data);
    } catch(error) {
      console.log(error);
    }
  }
  useEffect(()=>{
    getShop2_28();
  }, []);
  return (
    <div>
      <h1>All Products</h1>
      <ul>
        {products.map((product) => (
          <li>{product.name}</li>
        ))}
      </ul>
    </div>
  );
}
```

Line 29:13: img elements must have an alt prop, either with meaningful text, or an empty string for decorative images [jsx-a11y/alt-text](#)

src\pages\node\Shop2NodePage\_28.js

Line 35:13: img elements must have an alt prop, either with meaningful text, or an empty string for decorative images [jsx-a11y/alt-text](#)

webpack compiled with 1 warning

## Github logs of Week 15

```
User@LAPTOP-FDE6DRV7 MINGW64 /d/database/1112-2A-db-demo-410410228 (main)
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-23"
68b80817 Daineair Wed May 24 15:02:48 2023 +0800 w15 0524
```

```
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-23"
68b80817 Daineair Wed May 24 15:02:48 2023 +0800 w15 0524
```