

My Github URL

W11-P1: Implement route /api/shop_xx/:category in server

=> create tables category2_xx and shop2_xx using SQL and set foreign key of cat_id referencing to cid in category2_xx

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure with tables `category2_86` and `shop2_86`.
- Query Editor:** Contains the SQL command:


```
1 SELECT * FROM public.shop2_86
2 ORDER BY pid ASC
```
- Data Output:** Displays the results of the query, showing 35 rows of data from the `shop2_86` table.
- Create - Foreign key:** A modal window is open to define a foreign key constraint. It shows the following configuration:

Local	Referenced	Referenced Table
<code>cat_id</code>	<code>cid</code>	<code>public.category2_86</code>

=> update and delete constraints of the foreign key

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database structure with tables `category2_86` and `shop2_86`.
- Query Editor:** Contains the SQL command:


```
1 SELECT * FROM public.shop2_86
2 ORDER BY pid ASC
```
- Data Output:** Displays the results of the query, showing 35 rows of data from the `shop2_86` table.
- Create - Foreign key:** A modal window is open to define a foreign key constraint. It shows the following configuration:

On update	CASCADE
On delete	SET NULL

=> Chrome, show /api/shop_xx/womens with code (you need to use last two digits of your id)

The screenshot shows the Visual Studio Code interface with two open files: `server.js` and `database.js`. The `server.js` file contains code for an Express application, including routes for categories and products. The `database.js` file contains a connection string and a query for selecting products from the database. The browser window shows a JSON response for the `/api/shop_86/womens` route, listing 12 items with details like category ID (cid), name (cname), size, and price.

```

server_86 > JS server.js ...
1 import express from 'express';
2 import cors from 'cors';
3
4 const app = express();
5
6 import db from './utils/database.js';
7
8 app.use(cors());
9
10 app.use('/api/shop_86/:category', async (req, res, next) => {
11   console.log(`category: ${req.params.category}`);
12   const results = await db.query(
13     `select * from category2_86, shop2_86 where cname = $1 and
14     cid=cat_id`,
15     [req.params.category]
16   );
17   // console.log(`'next' is declared but its value is never read. ts(6133)`);
18   res.json(result);
19 })
20
21 app.use('/api/shop_86', async (req, res, next) => {
22   const results = await db.query('select * from shop2_86');
23   console.log(results, JSON.stringify(results.rows));
24   res.json(results.rows);
25 }
26
27 app.use('/', (req, res, next) => {
28   res.send('Kunslang Liao,213410128');
29 })
30
31 const port = process.env.PORT || 5000;
32 app.listen(port, () => {
33   console.log(`Server running on port ${port}`);
34 });
  
```

```

[{"cid": 4, "cname": "womens", "size": "large", "image_url": "https://i.ibb.co/GCCdy8t/womens.png", "remote_image_url": "/images/midterm/homepage/womens.png", "link_url": "/demo/shop_86/node/womens", "pid": 23, "pname": "Blue Tanktop", "cat_id": 4, "price": 25, "img_url": "/images/midterm/womens/blue-tank.png", "remote_img_url": "https://i.ibb.co/7COVJNm/blue-tank.png"}, {"cid": 5, "cname": "mens", "size": "large", "image_url": "https://i.ibb.co/R7QyBrQ/men.png", "remote_image_url": "/images/midterm/homepage/mens.png", "link_url": "/demo/shop_86/node/mens", "pid": 30, "pname": "Camo Down Vest", "cat_id": 5, "price": 325, "img_url": "/images/midterm/mens/camo-vest.png", "remote_img_url": "https://i.ibb.co/xISBT3Y/camo-vest.png"}]
  
```

=> Chrome, show `/api/shop_xx/mens` with code (you need to use last two digits of your id)

The screenshot shows the Visual Studio Code interface with two open files: `server.js` and `database.js`. The `server.js` file contains code for an Express application, including routes for categories and products. The `database.js` file contains a connection string and a query for selecting products from the database. The browser window shows a JSON response for the `/api/shop_86/mens` route, listing 12 items with details like category ID (cid), name (cname), size, and price.

```

server_86 > JS server.js ...
1 import express from 'express';
2 import cors from 'cors';
3
4 const app = express();
5
6 import db from './utils/database.js';
7
8 app.use(cors());
9
10 app.use('/api/shop_86/:category', async (req, res, next) => {
11   console.log(`category: ${req.params.category}`);
12   const results = await db.query(
13     `select * from category2_86, shop2_86 where cname = $1 and
14     cid=cat_id`,
15     [req.params.category]
16   );
17   // console.log(`results: ${JSON.stringify(results.rows)}`);
18   res.json(results.rows);
19 }
20
21 app.use('/api/shop_86', async (req, res, next) => {
22   const results = await db.query('select * from shop2_86');
23   console.log(results, JSON.stringify(results.rows));
24   res.json(results.rows);
25 }
26
27 app.use('/', (req, res, next) => {
28   res.send('Kunslang Liao,213410128');
29 })
30
31 const port = process.env.PORT || 5000;
32 app.listen(port, () => {
33   console.log(`Server running on port ${port}`);
34 });
  
```

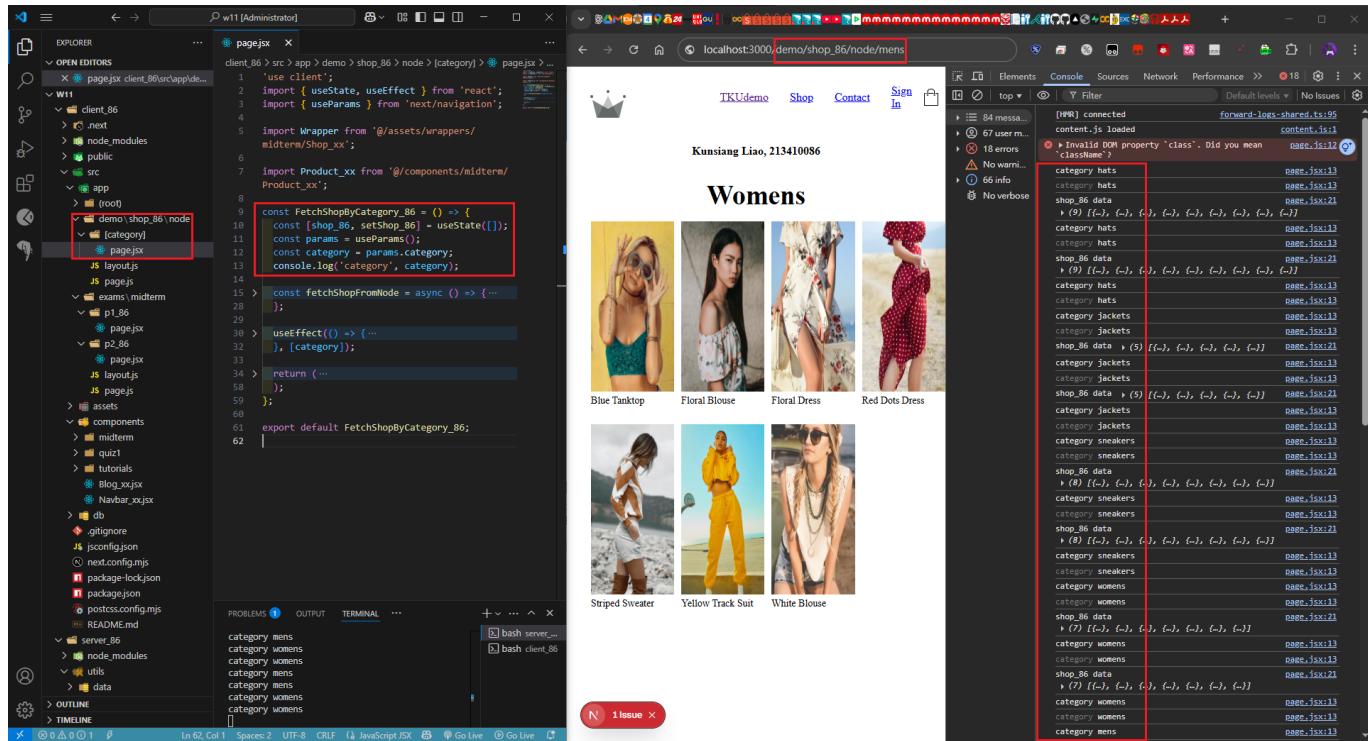
```

[{"cid": 5, "cname": "mens", "size": "large", "image_url": "https://i.ibb.co/R7QyBrQ/men.png", "remote_image_url": "/images/midterm/homepage/mens.png", "link_url": "/demo/shop_86/node/mens", "pid": 30, "pname": "Camo Down Vest", "cat_id": 5, "price": 325, "img_url": "/images/midterm/mens/camo-vest.png", "remote_img_url": "https://i.ibb.co/xISBT3Y/camo-vest.png"}]
  
```

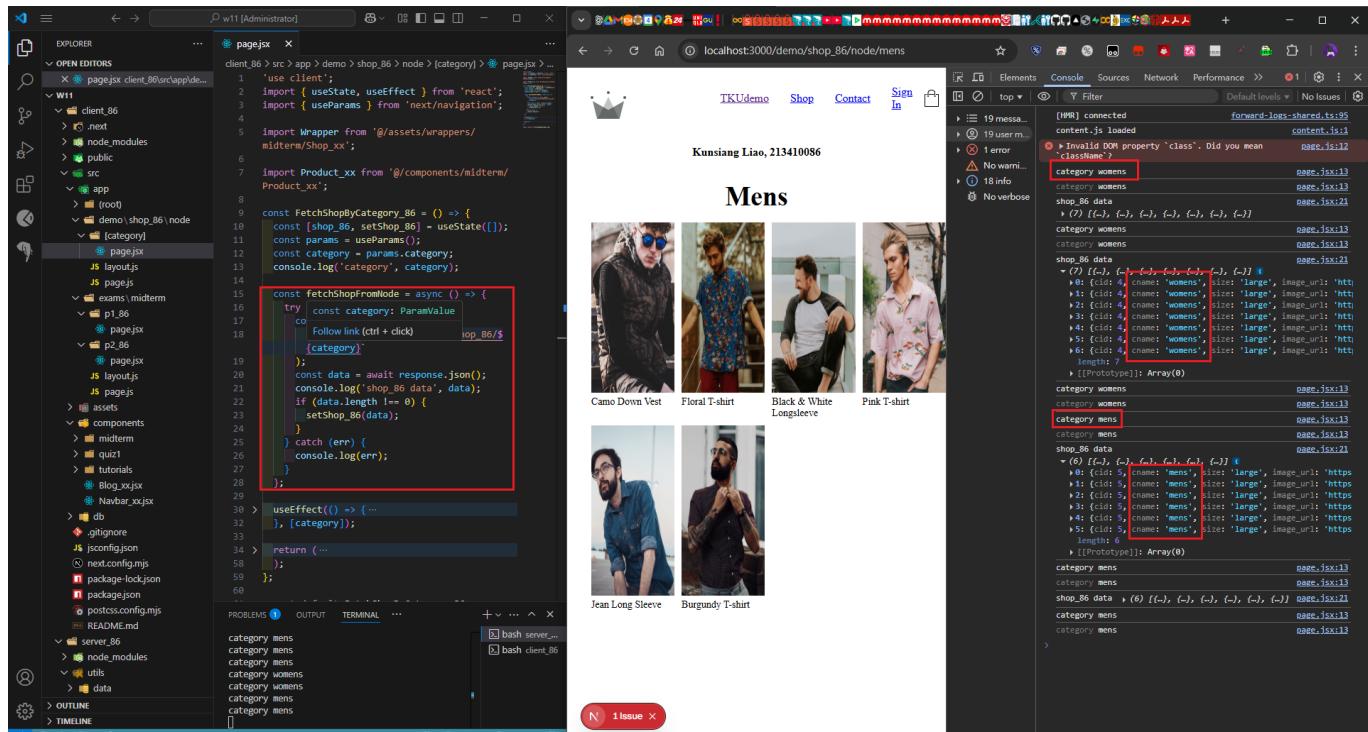
c0955c6 Littlei0409 Wed Dec 3 17:28:05 2025 +0800 W11-P1: Implement route
`/api/shop_xx/:category` in server

W11-P2: Implement `/demo/shop_xx/node` and `/demo/shop/node/:category` in client

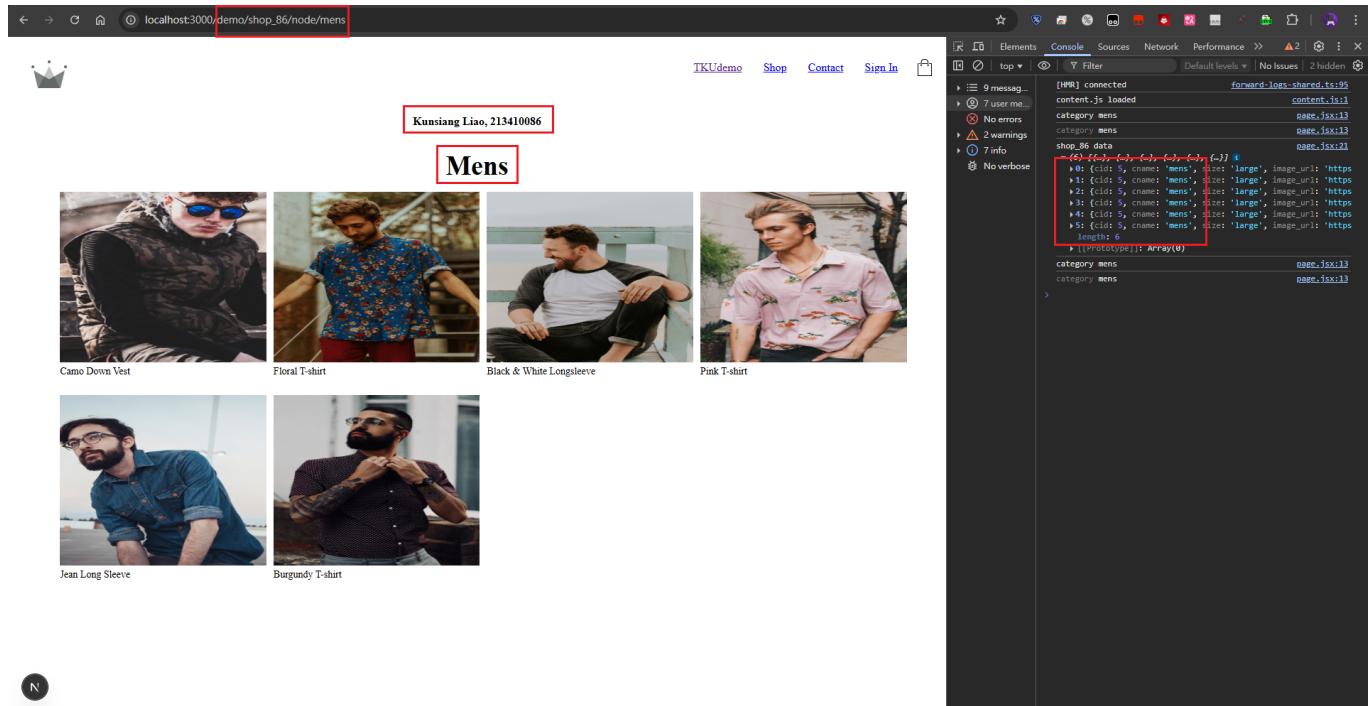
=> show how to get category from params



=> show how to fetch category from the category main page



=> Chrome, show FetchShopByCategory_xx.jsx by click Mens



=> relevant code for FetchShopByCategory_xx

```

File Edit Selection View Go Run Terminal Help
EXPLORER
GROUP 1
page.jsx client_86\src\app\...
GROUP 2
page.jsx client_86\src\app\...
W11
client_86
> .next
node modules
public
src
app
> (root)
demo\shop_86\node
> (category)
page.jsx
JS layout.js
JS page.js
JS exams\midterm
p1_86
page.jsx
p2_86
page.jsx
JS layout.js
JS page.js
> assets
components
midterm
quiz1
tutorials
Blog_xxxjs
Navbar_xxxjs
> db
gitignore
.json
config.json
package-lock.json
package.json
postcss.config.mjs
README.md
server_86
> OUTLINE
> TIMELINE
File Edit Selection View Go Run Terminal Help
File Edit Selection View Go Run Terminal Help
w11[Administrator]
page.jsx
client_86 > src > app > demo > shop_86 > node > [category] > page.jsx > ...
1 'use client';
2 import { useState, useEffect } from 'react';
3 import { useParams } from 'next/navigation';
4
5 import Wrapper from '@assets/wrappers/midterm/Shop_xx';
6
7 import Product_xx from '@components/midterm/Product_xx';
8
9 const FetchShopByCategory_86 = () => {
10   const [shop_86, setShop_86] = useState([]);
11   const params = useParams();
12   const category = params.category;
13   console.log('category', category);
14
15   const fetchShopFromNode = async () => {
16     try {
17       const response = await fetch(
18         `http://localhost:5000/api/shop_86/${category}`
19       );
20       const data = await response.json();
21       console.log('shop_86 data', data);
22       if (data.length !== 0) {
23         setShop_86(data);
24       }
25     } catch (err) {
26       console.log(err);
27     }
28   };
29
30   useEffect(() => {
31     fetchShopFromNode();
32   }, [category]);
33
34   return (
35     <Wrapper>
36       <div>...
37         <h4>...</h4>
38         <h4>...</h4>
39         <div>...
40           <h1>...</h1>
41           <div>...
42             <shop_86.map(item => {
43               const { pid, pname, prize, img_url } = item;
44               return (
45                 <Product_xx
46                   key={pid}
47                   img_url={img_url}
48                   pname={pname}
49                   prize={prize}
50                 >
51                 ...
52               );
53             })
54           </div>
55         </div>
56       </div>
57     </Wrapper>
58   );
59 }
60
61 export default FetchShopByCategory_86;

```

14c41f1 Littlei0409 Thu Dec 4 11:40:46 2025 +0800 W11-P2: Implement /demo/shop_xx/node and /demo/shop/node/:category in client

W11-logs: git logs of 11

The screenshot shows a GitHub repository page for '1141-2N-kunslang-86'. The URL in the address bar is 'github.com/Little0409/1141-2N-kunslang-86/commits/main/'. The page displays a list of commits. A red box highlights the commit from December 4, 2025, which includes a task description: 'W11-P2: Implement /demo/shop_xx/node and /demo/shop/node/category in client'. This commit was made by Little0409 and has a commit hash of 8a9e5d7.

Date	Commit Message	Author	Commit Hash
Dec 4, 2025	W11-P2: Implement /demo/shop_xx/node and /demo/shop/node/category in client	Little0409	8a9e5d7
Dec 3, 2025	W11-P1: Implement route /api/shop_xx/category in server	Little0409	c0955c6