Faculty of Information and Communication Technology
Mahidol University

Project Phase II

By

Miss Russarin	Eaimrittikrai	6488021
Miss Chaninan	Phetpangun	6488061
Miss Penpitchapa	Pantaraksakul	6488175
Miss Warangkhana	Srichan	6188096

ITCS212 Web Programming
2023

Table of Contents

Link to Presentation Video:

https://drive.google.com/file/d/1qn28AZ4JhSeJ-rFWYabip6CL3TzvyVmv/view?usp=share_link

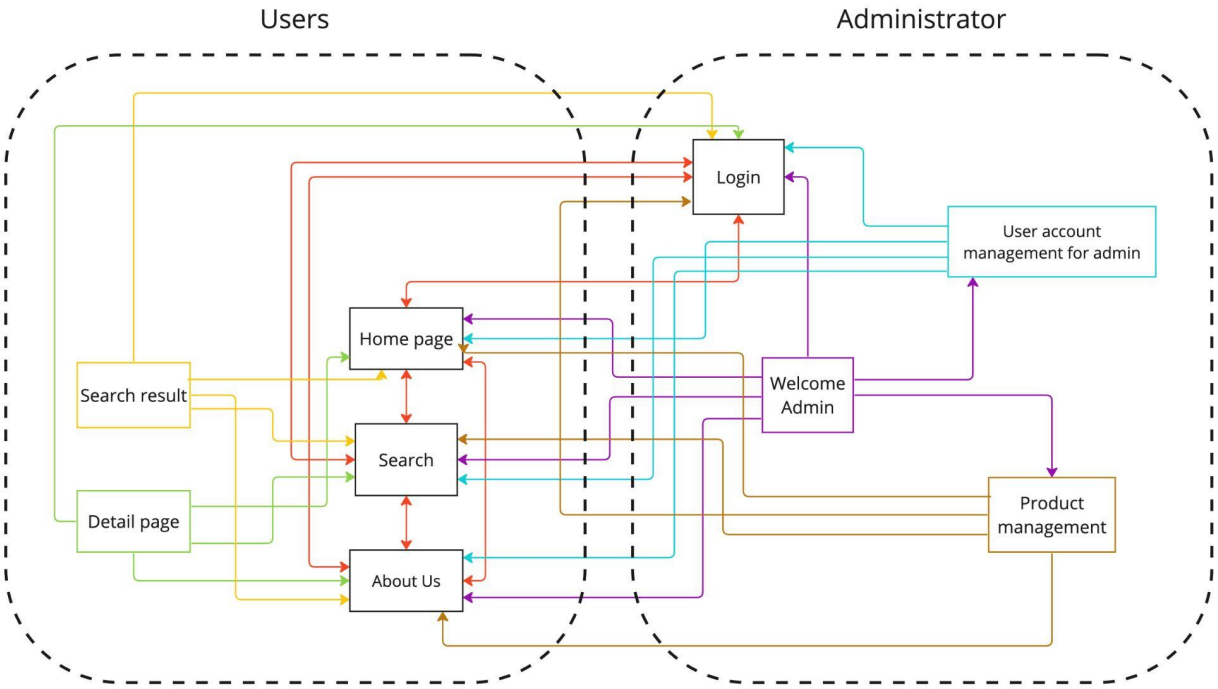
Project Overview

The business domain that was selected by the team members is an online clothes-shopping site that allows users to explore products from various brands; therefore, the name of this website is MultiClothes Shopping. In this project, the members are assigned to design and develop a front-ended section of the web application (UI) using the combination of HTML and CSS, and a back-ended section (web services) using the combination of Node.js and JavaScript.

The objective of the first phase of the project is to plan out the navigation pages and implement the pages by utilizing HTML and CSS to create front-end web pages for the business which consist of a Home Page, Search Page, Search Result Page, Product Detail Page, About Us Page, Log in for Administrator Page, Welcome Admin Page, Product Management for Administrator Page, and Admin Information Management for Administrator Page.

The objective of the second phase of the project is to create a database to keep the administrators' information, administrators' login information, and product information. Then, the members have to create web services to handle functions: authentication for administrators to log in, products modifications for administrators (search or view, insert, update, delete product), and administrators' information modification for administrators (search or view, insert, update, delete administrators' information). Finally, the members create web services interaction by connecting the web services to the interface of the web application and connecting the web application to a public API. In the implementation, the selected web API is Currency Data API to convert currency in real time (https://apilayer.com/marketplace/currency_data-api?e=Sign+Up&l=Success)

Navigation Diagram



The navigation diagram for this project is divided into two major sections: users and administrators. The users' section consists of a Home page, Search page, Search result, Details page of the product, and About Us. The administrators' section consists of a Login page, a Welcome Admin page, a Product Management page, and a User account management page for admin. Since the navigation bar is allocated on every page, the Home page, Search page, About Us page, and Login page for admin are accessible on every page. On the other hand, the Welcome Admin page, Product Management page, and User account management page are only accessible after the administrators successfully log in through the Login page.

Database

Name: multiclothes

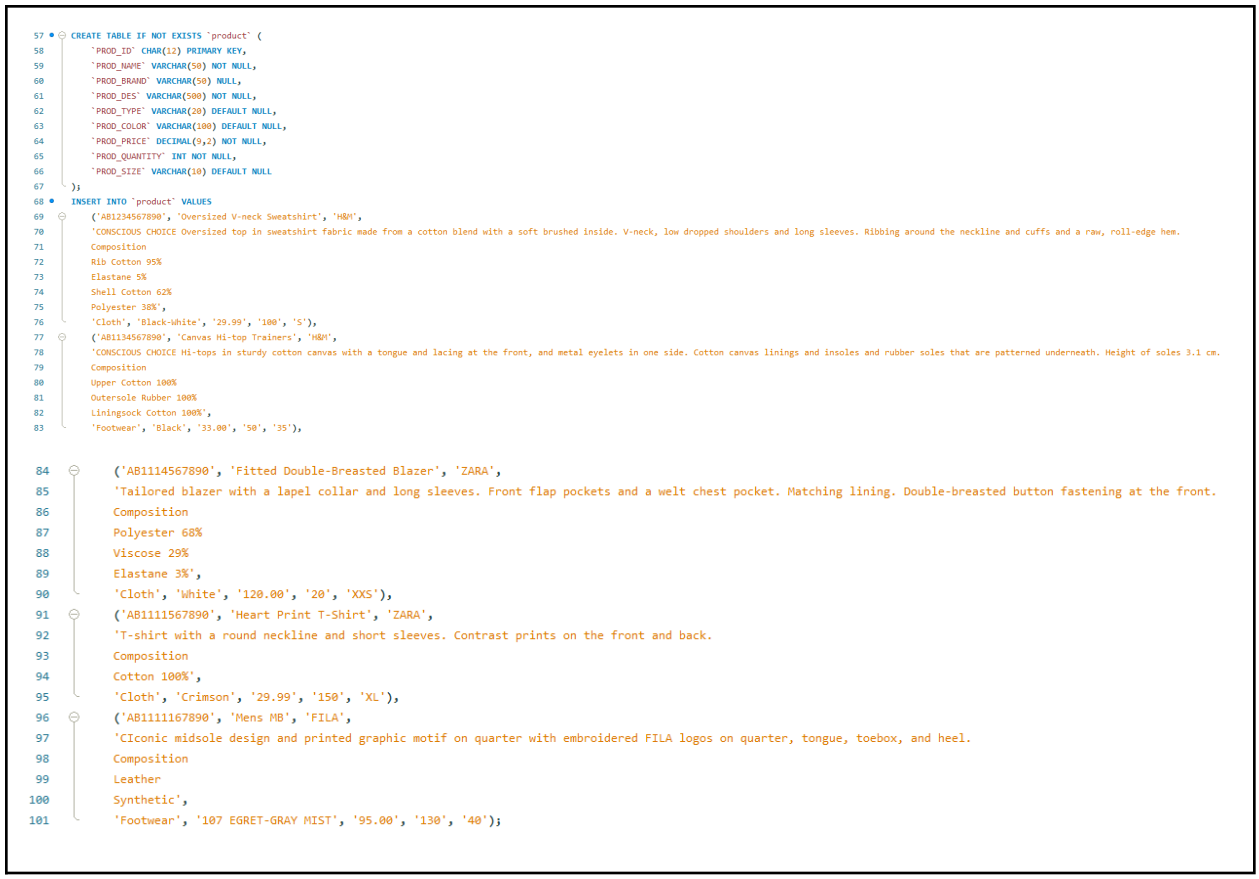
Table admin information: admin_info

```
CREATE TABLE IF NOT EXISTS `admin_info` (
    `AD_NUM` CHAR(9) PRIMARY KEY,
    `AD_FNAME` VARCHAR(50) NOT NULL,
    `AD_LNAME` VARCHAR(50) NOT NULL,
    `AD_ADDRESS` VARCHAR(500) NOT NULL,
    `AD_BD` DATE NOT NULL,
    `AD_EMAIL` VARCHAR(50) NOT NULL,
    `AD_PHONE` CHAR(10)
);

INSERT INTO `admin_info`VALUES
    ('000000001','Somsri','Mayang','999 Phuttamonthon 4 Road, Salaya, Nakhon Pathom 73170 Thailand','1988-11-22','somsri.may@multiclothes.com','0869144407'),
    ('006000202','Yang','Nonyu','23, Trok Rongnamkeang, Yotha Rd., Bangkok, 10100 Thailand','1991-01-17','yang.non@multiclothes.com','0972187739'),
    ('000020031','Suaysood','Naisoy','Khlong Luang 25 KhlongNung KhlongLuang, Chang Wat Pathum Thani 12120','2001-05-03','Suaysood.nai@multiclothes.com','0978394783'),
    ('000885600','Waimaii','Yaknon','17 Henri Dunant Rd, Khwaeng Pathum Wan, Khet Pathum Wan, Krung Thep Maha Nakhon 10330','1970-07-29','Waimaii.yak@multiclothes.com','0622785590'),
    ('000032191','Chanom','Kaimook','62 Moo 1, Rangsit-Ongkharak Road (Km.7) Thanyaburi, Pathum Thani','1999-09-12','Chanom.kai@multiclothes.com','0974328801');
-- -------------------------------------------------------
```

Table admin login information: admin_login

```
CREATE TABLE IF NOT EXISTS `admin_login` (
    `AD_USERNAME` VARCHAR(50) NOT NULL,
    `AD_PASSWORD` VARCHAR(20) NOT NULL,
    `AD_LOGIN_TIMESTAMP` DATETIME NOT NULL,
    `AD_NUM` CHAR(9) NOT NULL,
    CONSTRAINT `FK_adminAD_NUM` FOREIGN KEY (`AD_NUM`)
    REFERENCES `admin_info` (`AD_NUM`)
);

INSERT INTO `admin_login` VALUES
    ('somsri.may@multiclothes.com', 'Somchai555', '2022-11-08 22:02:22', '000000001'),
    ('yang.non@multiclothes.com', 'nyaungyooka', '2022-03-18 18:29:30', '006000202'),
    ('Waimaii.yak@multiclothes.com', 'Maiwai', '2022-01-04 10:20:35', '000885600'),
    ('Chanom.kai@multiclothes.com', 'Maiwann0', '2022-10-01 15:29:09', '000032191'),
    ('Suaysood.nai@multiclothes.com', 'blursabb', '2022-12-15 01:45:27', '000020031');
```

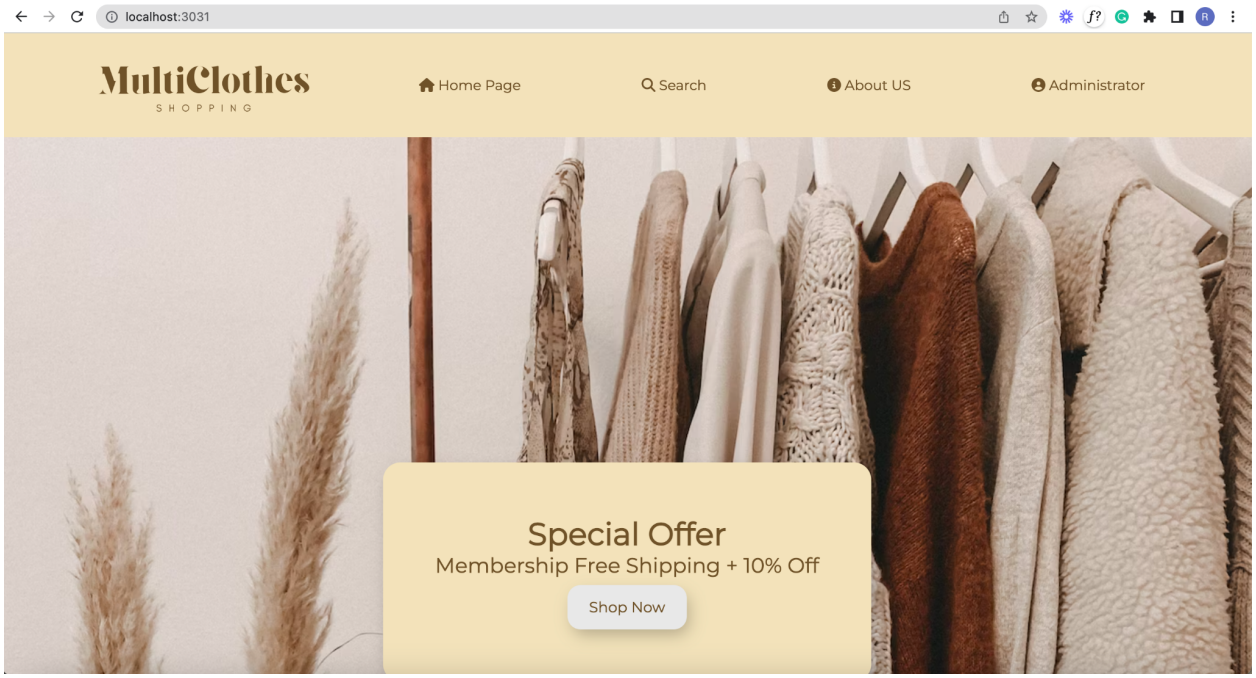
Table product information: product



```sql
CREATE TABLE IF NOT EXISTS `product` (
    `PROD_ID` CHAR(12) PRIMARY KEY,
    `PROD_NAME` VARCHAR(50) NOT NULL,
    `PROD_BRAND` VARCHAR(50) NULL,
    `PROD_DES` VARCHAR(500) NOT NULL,
    `PROD_TYPE` VARCHAR(20) DEFAULT NULL,
    `PROD_COLOR` VARCHAR(100) DEFAULT NULL,
    `PROD_PRICE` DECIMAL(9,2) NOT NULL,
    `PROD_QUANTITY` INT NOT NULL,
    `PROD_SIZE` VARCHAR(10) DEFAULT NULL
);
INSERT INTO `product` VALUES
    ('AB1234567890', 'Oversized V-neck Sweatshirt', 'H&M',
    'CONSCIOUS CHOICE Oversized top in sweatshirt fabric made from a cotton blend with a soft brushed inside. V-neck, low dropped shoulders and long sleeves. Ribbing around the neckline and cuffs and a raw, roll-edge hem.
    Composition
    Rib Cotton 95%
    Elastane 5%
    Shell Cotton 62%
    Polyester 38%',
    'Cloth', 'Black-White', '29.99', '100', 'S'),
    ('AB1134567890', 'Canvas Hi-top Trainers', 'H&M',
    'CONSCIOUS CHOICE Hi-tops in sturdy cotton canvas with a tongue and lacing at the front, and metal eyelets in one side. Cotton canvas linings and insoles and rubber soles that are patterned underneath. Height of soles 3.1 cm.
    Composition
    Upper Cotton 100%
    Outersole Rubber 100%
    Liningsock Cotton 100%',
    'Footwear', 'Black', '33.00', '50', '35'),

    ('AB1114567890', 'Fitted Double-Breasted Blazer', 'ZARA',
    'Tailored blazer with a lapel collar and long sleeves. Front flap pockets and a welt chest pocket. Matching lining. Double-breasted button fastening at the front.
    Composition
    Polyester 68%
    Viscose 29%
    Elastane 3%',
    'Cloth', 'White', '120.00', '20', 'XXS'),
    ('AB1111567890', 'Heart Print T-Shirt', 'ZARA',
    'T-shirt with a round neckline and short sleeves. Contrast prints on the front and back.
    Composition
    Cotton 100%',
    'Cloth', 'Crimson', '29.99', '150', 'XL'),
    ('AB1111167890', 'Mens MB', 'FILA',
    'CIconic midsole design and printed graphic motif on quarter with embroidered FILA logos on quarter, tongue, toebox, and heel.
    Composition
    Leather
    Synthetic',
    'Footwear', '107 EGRET-GRAY MIST', '95.00', '130', '40');
```

Details of web application and code

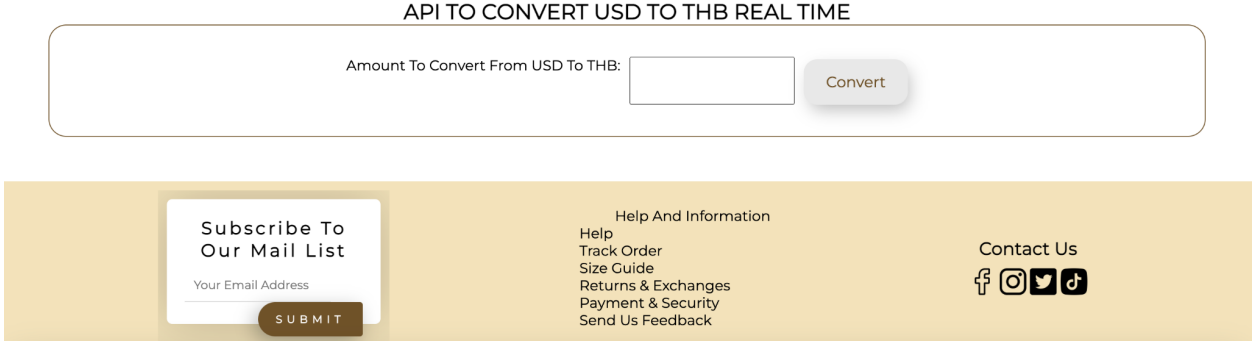
Web server of the homepage: http://localhost:3031/



Code that connects this page to the port:

```
//  Home Page
router.get('/', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/HomePage.html`))
    res.status(200)
```

This is the homepage of the web application which is running on port 3031. On this page, there is the connected public API at the end of the page that the users can convert USD to THB in real-time.

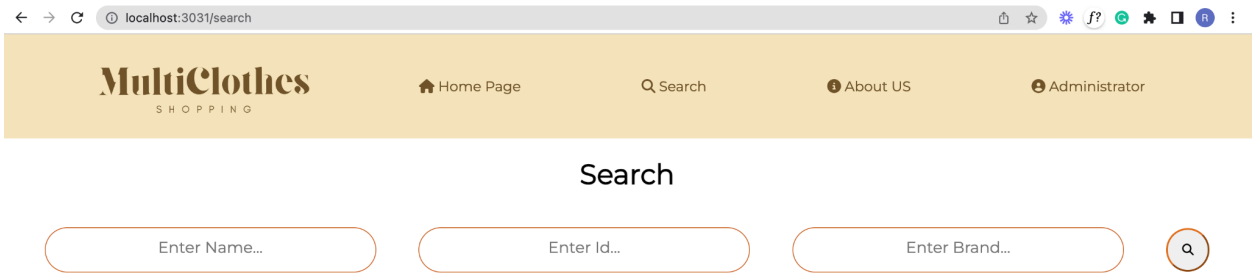


Code that connects the web application to the Currency Data API:

```
<script>
    // set endpoint and your access key
    const endpoint = 'convert'
    const access_key = 'ti3FllknYhicQb25iSRJrTVkEAXib9Gy';

    // define from currency, to currency, and amount
    const from = 'USD';
    const to = 'THB';
    const amount = document.getElementById('amount-input');
    const convertButton = document.getElementById('convert-btn');
    convertButton.addEventListener('click', async (event) => {
        event.preventDefault();
        var myHeaders = new Headers();
        myHeaders.append("apikey", "ti3FllknYhicQb25iSRJrTVkEAXib9Gy");
        const query = amount.value;

        var requestOptions = {
            method: 'GET',
            redirect: 'follow',
            headers: myHeaders
        };

        await fetch(`https://api.apilayer.com/currency_data/convert?to=${to}&from=${from}&amount=${query}`, requestOptions)
            .then(response => response.text())
            .then(result => alert(result))
            .catch(error => console.log('error', error));
    });
</script>
```

This code connects to the Currency Data API by using the Free API key from the website (The limitation of use is 100 a month), and then executes fetch to acquire the information from the link provided in the website to calculate the currency from USD to THB in real-time. The return data will alert on the web page in the form of JSON.

Web server of the search page: http://localhost:3031/search

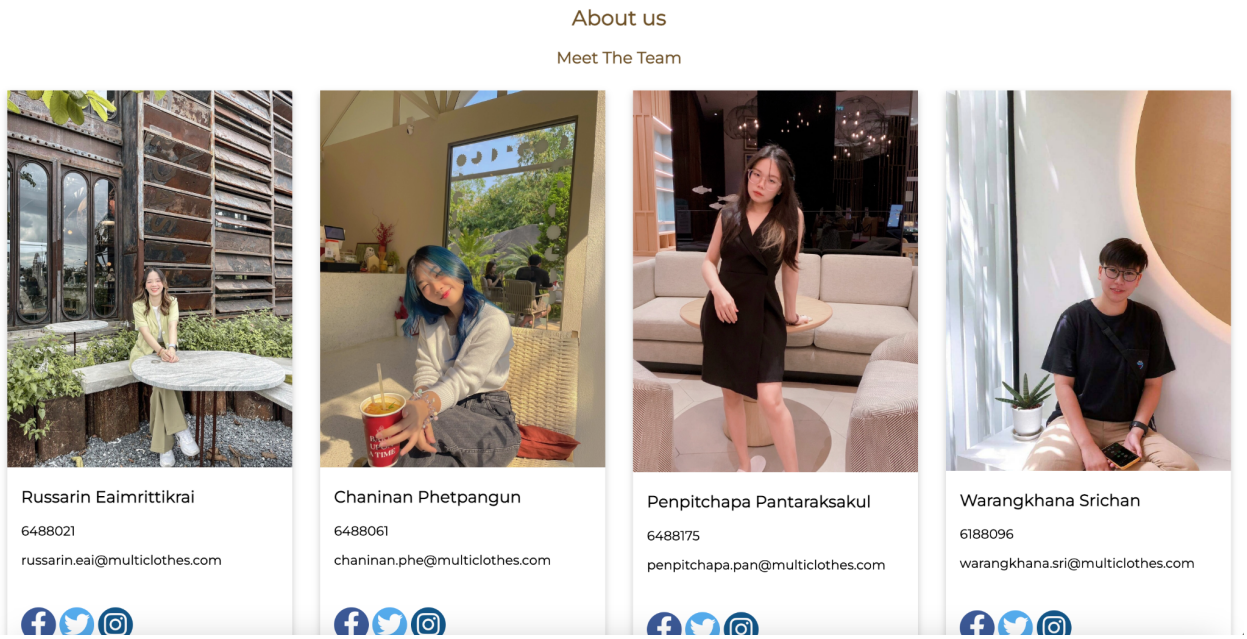


Code that connects this page to the port:

```
//  Search Page
router.get('/search', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/Search.html`))
    res.status(200)

});
```

This is the search page of the web application. On this page, the users can view all of the products and search for the products by entering either name, id, or brand of the product.

Web server of the about us page: http://localhost:3031/about-us

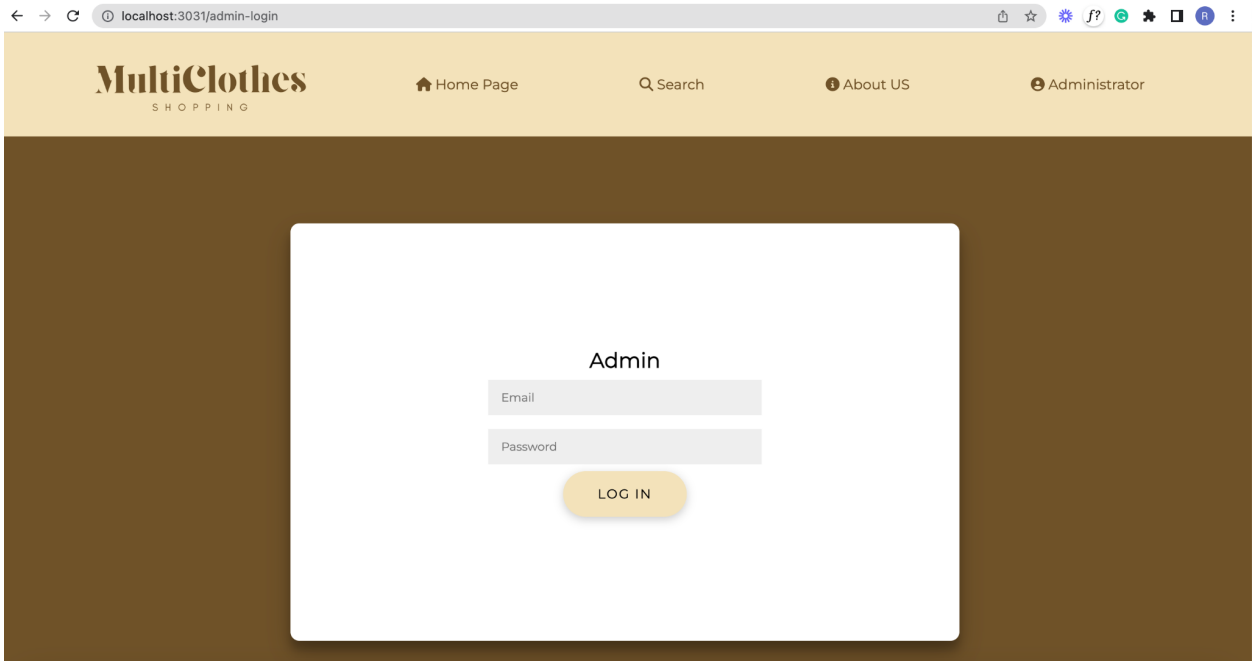


Code that connects this page to the port:

```
//  About Us Page
router.get('/about-us', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/About_us.html`))
    res.status(200)
});
```

This is the about us page of the web application. On this page, the users can view the name and contact information of the members of the team.

Web server of the administrator login page: http://localhost:3031/admin-login

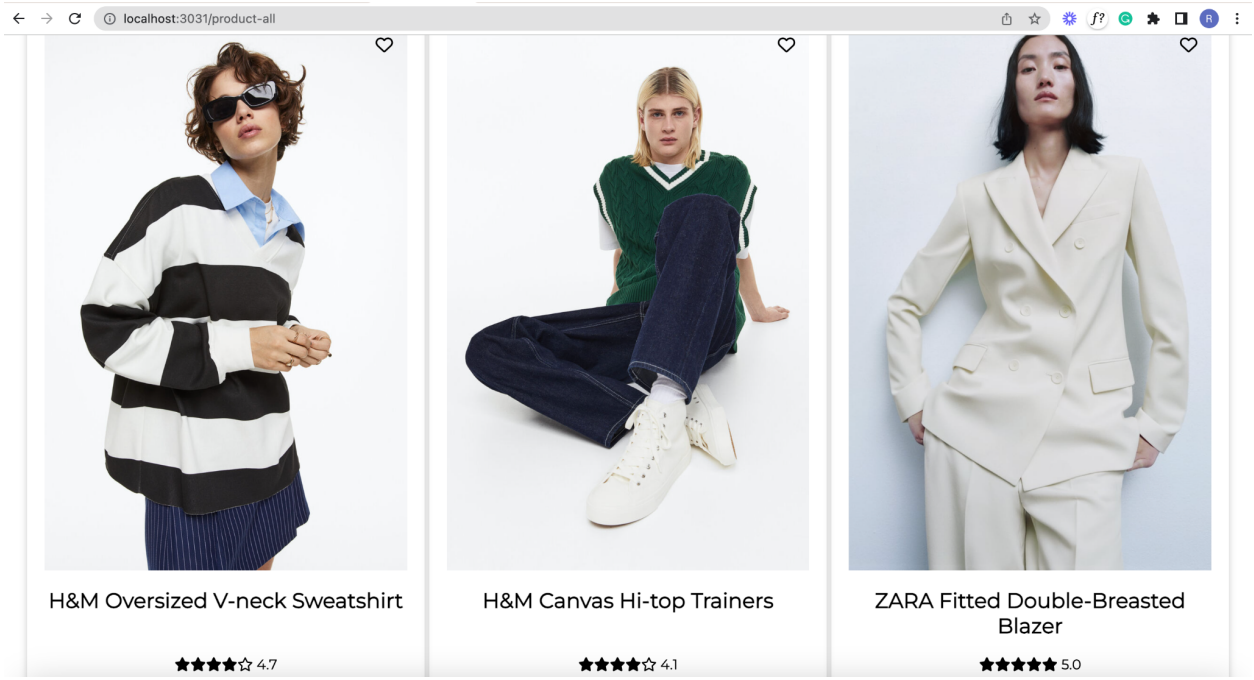


Code that connects this page to the port:

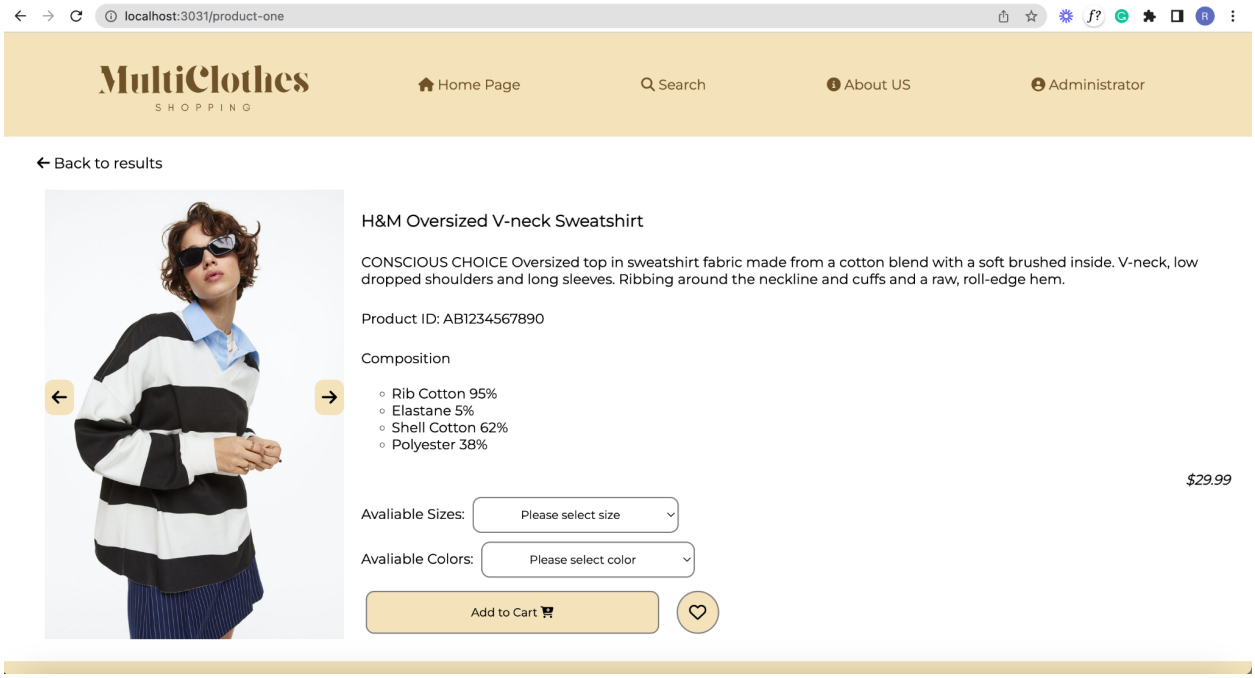
```
//  Admin login Page
router.get('/admin-login', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/Admin.html`))
    res.status(200)
});

router.post('/form-submit', function (req, res) {
    console.log(req.method);
    console.log(req.body.email);
    res.status(200);
});
```

This is the login page for administrators. The administrators who can log in to this page already have their email (username) and password information in the database. Otherwise, it will alert invalid login.

Web server of the login page for administrators: http://localhost:3031/product-all



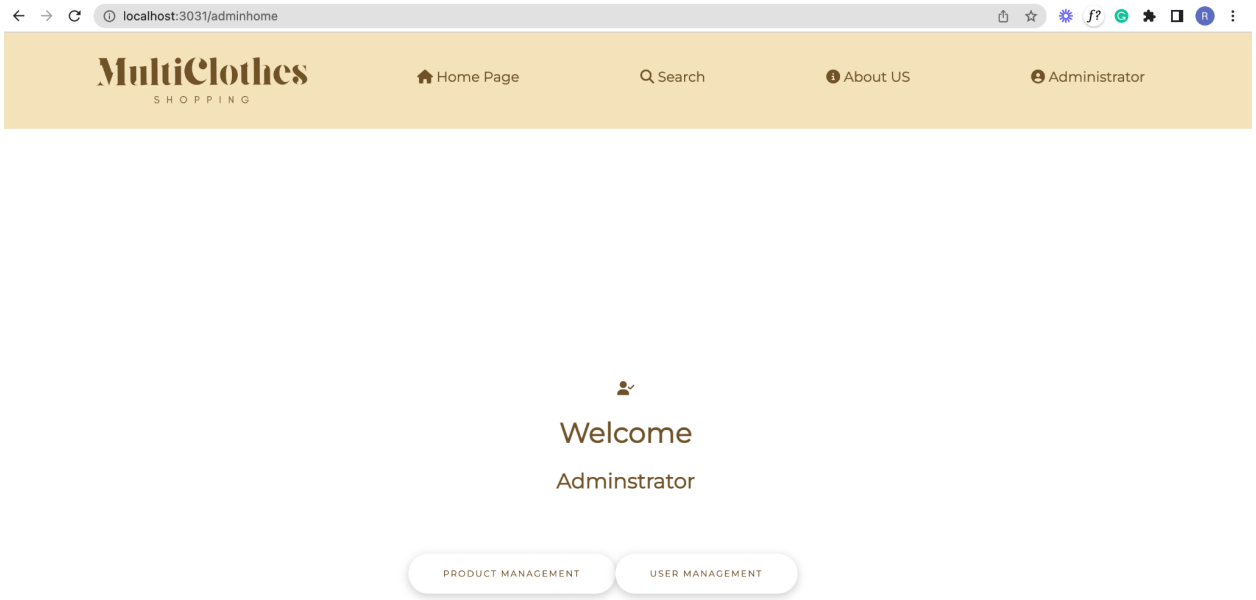
This is the page of the web application to show the list of products with pictures and information. On this page, the users can click the Buy Now button to see detailed information about each product. For instance, this page will show detailed information about the first product.



Code that connects this page to the port:

```
// page to show all product with picture
router.get('/product-all', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/DetailPage.html`))
    res.status(200)
});
// Detail page of product 1
router.get('/product-one', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/DetailPageOne.html`))
    res.status(200)
});
// Detail page of product 2
router.get('/product-two', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/DetailPageTwo.html`))
    res.status(200)
});
// Detail page of product 3
router.get('/product-three', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/DetailPageThree.html`))
    res.status(200)
});
// Detail page of product 4
router.get('/product-four', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/DetailPageFour.html`))
    res.status(200)
});
// Detail page of product 5
router.get('/product-five', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/DetailPageFive.html`))
    res.status(200)
});
// Detail page of product 6
router.get('/product-six', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/DetailPageSix.html`))
    res.status(200)
});
```

Web server of the page to welcome administrators: http://localhost:3031/adminhome

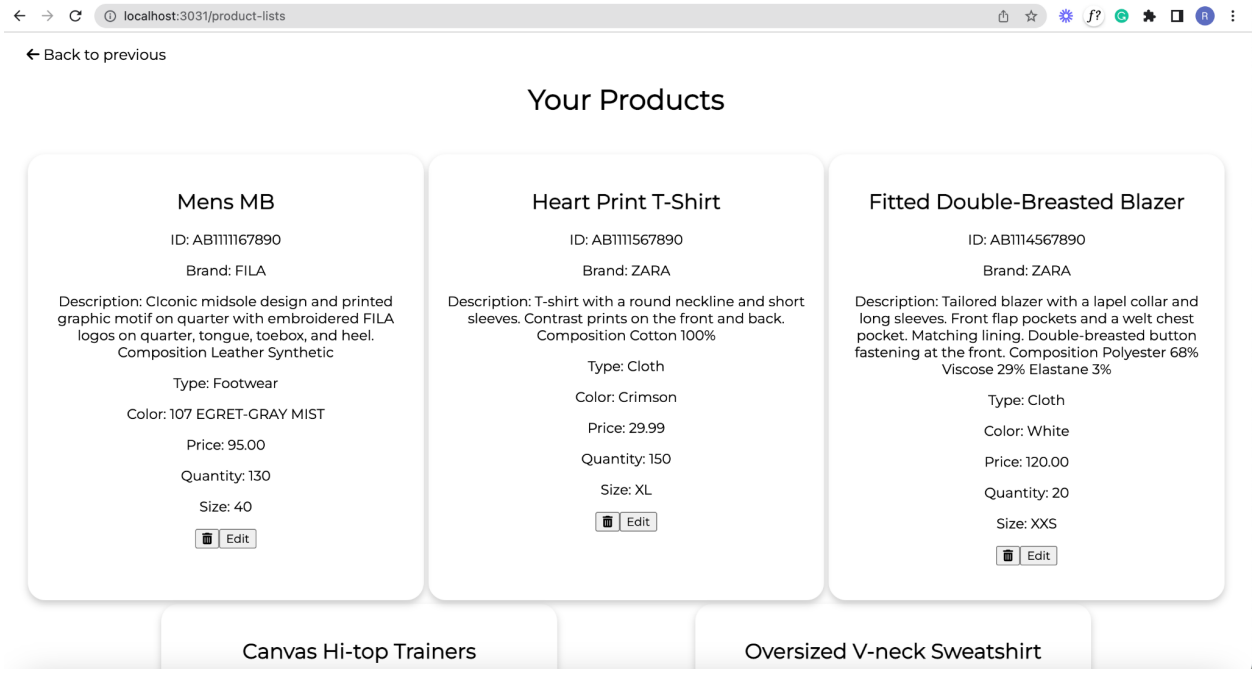


Code that connects this page to the port:

```
// Welcome Admin Page
router.get('/adminhome', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/WelcomePageAdmin.html`))
    res.status(200)
});
```

Once the administrator successfully logs in to the system, they will see this page that has two buttons that will navigate to the product management page and the user management page.

Web server of the product management page:http://localhost:3031/product-lists

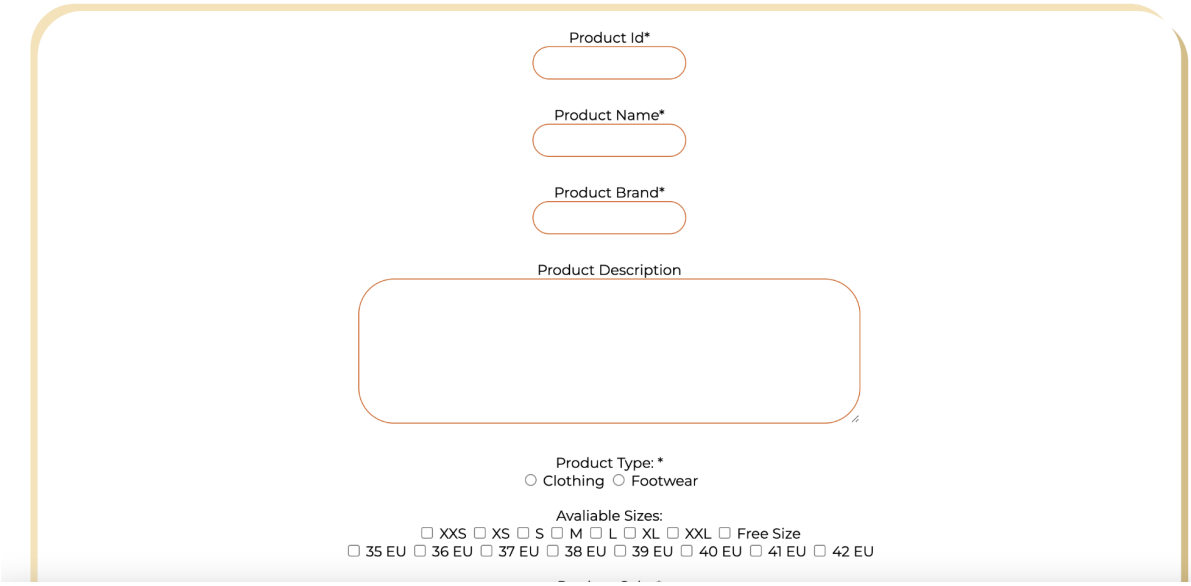


Code that connects this page to the port:

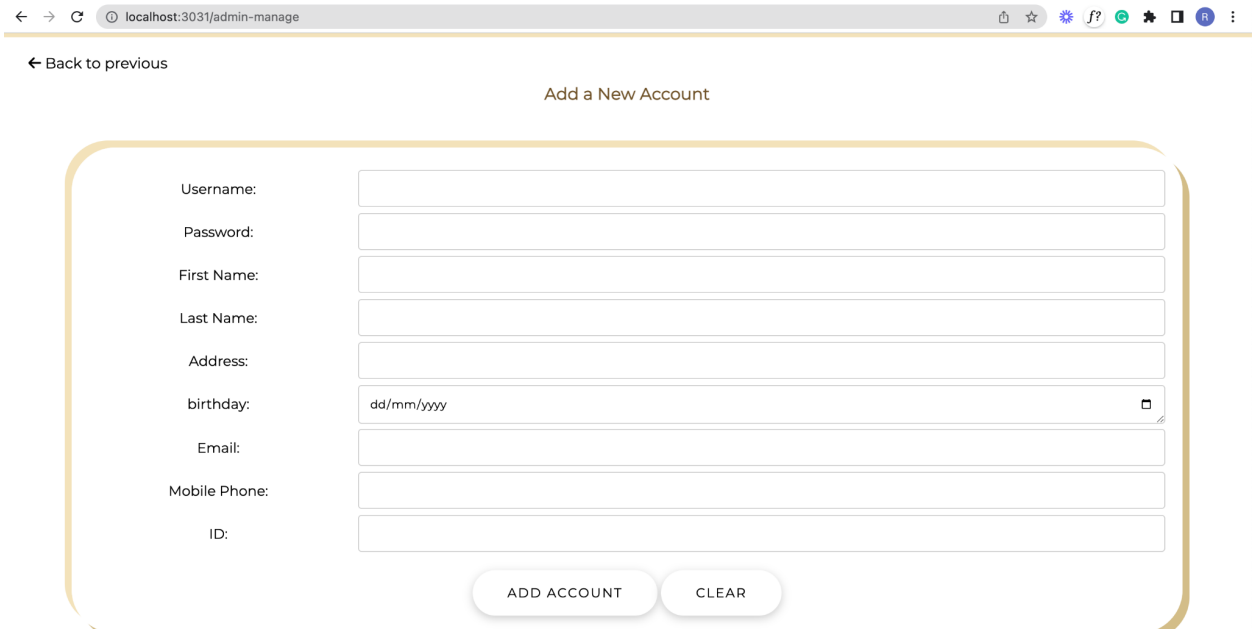
```
//  List of products in the store
router.get('/product-lists', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/ProductManage1.html`))
    res.status(200)
});
//  Form to add more product
router.get('/product-add', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/ProductManage.html`))
    res.status(200)
});
//  Form to update product
router.get('/product-update', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/ProductUpdate.html`))
    res.status(200)
});
```

After the administrator clicked on the Product management button, they will see this page that contains the information about the products. They can click the trash can button on the product to delete the product, click Edit to modify the product information, and click Add More which will lead to the form to fill in the detail of the new product (http://localhost:3031/product-add).

Product Management



Web server of the administrators' account management page:http://localhost:3031/admin-manage



Code that connects this page to the port:

```
//  Form to add more admin info
router.get('/admin-manage', (req, res) => {
    res.sendFile(path.join(`${__dirname}/html/User_Account_Management.html`))
    res.status(200)
});
```

After the administrator clicked on the Account management button, they will see this page that contains the form to add a new account. They can view all the administrators' details, search for the administrator's information, edit the administrator's information, and delete the administrator's account.

Details of web services and code

Port of the backend.js that keeps all the web services: http://localhost:3030
There is a total of three routers in the node:

- Router 1: product management service (http://localhost:3030/product)
- Router 2: login authentication service (http://localhost:3030/admin-login)
- Router 3: admin information management service (http://localhost:3030/admin)

1. Authentication web service for administrators

```
router2.post('/', (req, res) => {
    let admin_username = req.body.username;
    let admin_password = req.body.password;
    console.log('admin login:', admin_username, `at ${Date()} ${Date.now()}`);
    if (!admin_username || !admin_password) {
        return res.status(400).send({
            error: true,
            message: 'Please enter username and password'
        });
    }
    dbConn.query("SELECT AD_NUM FROM admin_login WHERE AD_USERNAME = ? AND AD_PASSWORD = ?", [admin_username, admin_password], function (error, results) {
        if (error) throw error;
        // If username or password doesn't match database
        if (results.length == 0) {
            return res.status(401).send({
                error: true,
                message: 'Invalid log in',
                status: 401
            });
        }
        else {
            return res.status(200).send({
                error: false,
                data: results,
                message: 'Successfully log in',
                status: 200
            });
        }

    });
});
```

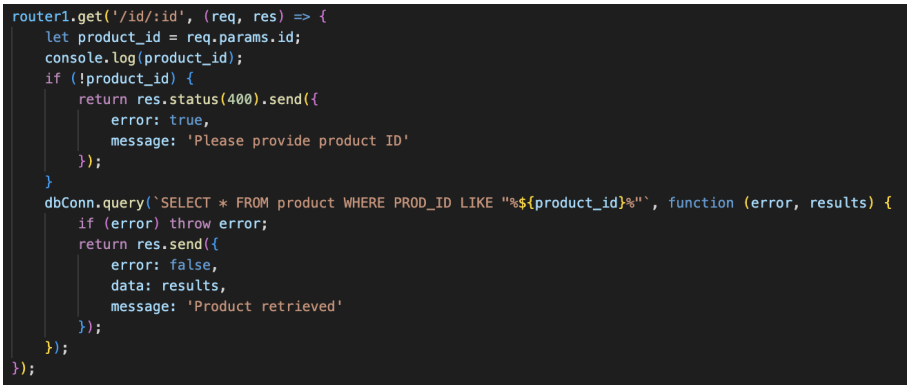
HTTP method for login authentication is POST. The elements that are required for authentication are username and password. The query checks if the username and password exist in the database and if they are matched.

2. Functions for administrators to search/view, insert, update, and delete products' information
    - View all products

```
// Select All (No criteria search)
router1.get('/products', (req, res) => {
    dbConn.query("SELECT * FROM product", function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results,
            message: 'Product list.'
        });
    });
});
```

Method: GET
URL: http://localhost:3030/product/products
Search all the products in the database.

- Search by three criteria: Product Id, Product Name, and Product Brand

Search by Id	
	Method: GET URL: http://localhost:3030/product/id/ Recieve ID as parameter and check if there is any product ID that contains the input ID.
Search by Name	
	Method: GET URL: http://localhost:3030/product/name/ Recieve Name as parameter and check if there is any product Name that contains the input Name.
Search by Brand	
	Method: GET URL: http://localhost:3030/product/brand/ Recieve Brand as parameter and check if there is any product Brand that contains the input Brand.

Search by Id and Name	
	Method: GET URL: http://localhost:3030/product/idname/ Recieve ID and Name as parameters and check if there is any product that contains the input ID or Name.
Search by Id and Brand	
	Method: GET URL: http://localhost:3030/product/idbrand/ Recieve ID and Brand as parameters and check if there is any product that contains the input ID or Brand.
Search by Name and Brand	
	Method: GET URL: http://localhost:3030/product/namebrand/ Recieve Name and Brand as parameters and check if there is any product that contains the input Name or Brand.
Search by Id, Name, and Brand	

```
// Search by id, name, and brand
router1.get('/all/:id/:name/:brand', (req, res) => {
    let product_id = req.params.id;
    console.log(product_id);
    let product_name = req.params.name;
    console.log(product_name);
    let product_brand = req.params.brand;
    console.log(product_brand);
    if (!product_id) {
        return res.status(400).send({
            error: true,
            message: 'Please provide product ID'
        });
    }
    if (!product_name) {
        return res.status(400).send({
            error: true,
            message: 'Please provide product name'
        });
    }
    if (!product_brand) {
        return res.status(400).send({
            error: true,
            message: 'Please provide product Brand'
        });
    }
    dbConn.query(`SELECT * FROM product WHERE PROD_ID LIKE "%${product_id}%" OR PROD_BRAND LIKE "%${product_brand}%" OR PROD_NAME LIKE "%${product_name}%"`,
        if (error) throw error;
        return res.send({
            error: false,
            data: results,
            message: 'Product retrieved'
        });
    });
});
```

Method: GET
URL: http://localhost:3030/product/all/

Recieve ID, Name and Brand as parameters and check if there is any product that contains the input ID or Name or Brand.

- Insert a product

```
// Insert
router1.post('/add', (req, res) => {
    let product = req.body.product;
    console.log(product);
    if (!product) {
        return res.status(400).send({
            error: true,
            message: 'Please provide product information'
        });
    }
    dbConn.query("INSERT INTO product SET ?", [product], function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results.affectedRows,
            message: 'New product has been created successfully'
        });
    });
});
```

Method: POST

URL: http://localhost:3030/product/add

Inert a product into the database. Receive the information in the body of product.

- Update a product

```
// Update
router1.put('/update', (req, res) => {
    let product_id = req.body.product.PROD_ID;
    let product = req.body.product;
    console.log(product_id);
    console.log(product);
    if (!product || !product_id) {
        return res.status(400).send({
            error: true,
            message: 'Please provide Product information'
        });
    }
    dbConn.query("UPDATE product SET ? WHERE PROD_ID = ?", [product, product_id], function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results.affectedRows,
            message: 'Product has been updated successfully'
        });
    });
});
```

Method: PUT

URL: http://localhost:3030/product/update

Update product information of the products that are already exist in the database based on the given product ID. Receive the information in the body of product.

- Delete a product

```
// Delete
router1.delete('/delete', (req, res) => {
    let product_id = req.body.product.PROD_ID;
    console.log(product_id);
    if (!product_id) {
        return res.status(400).send({
            error: true,
            message: 'Please provide Product ID'
        });
    }
    dbConn.query("DELETE FROM product WHERE PROD_ID = ?", [product_id], function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results.affectedRows,
            message: 'Product has been deleted successfully'
        });
    });
});
```

Method: DELETE

URL: http://localhost:3030/product/delete

Delete a product that are already exist in the database based on the given product ID. Receive the product ID in the body of product.

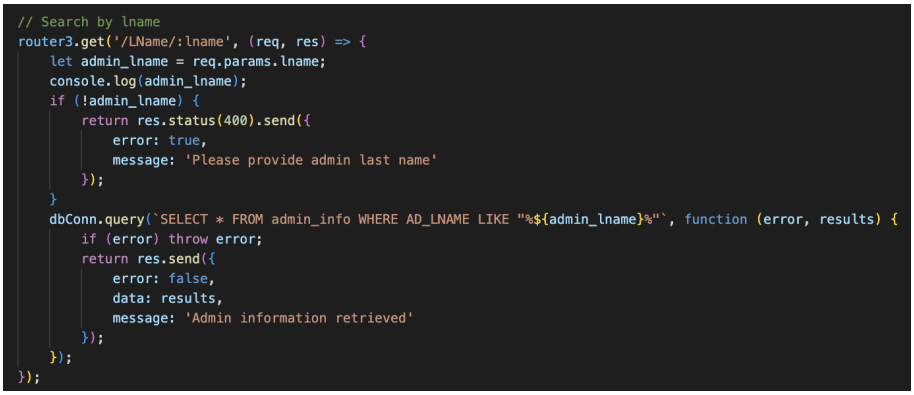
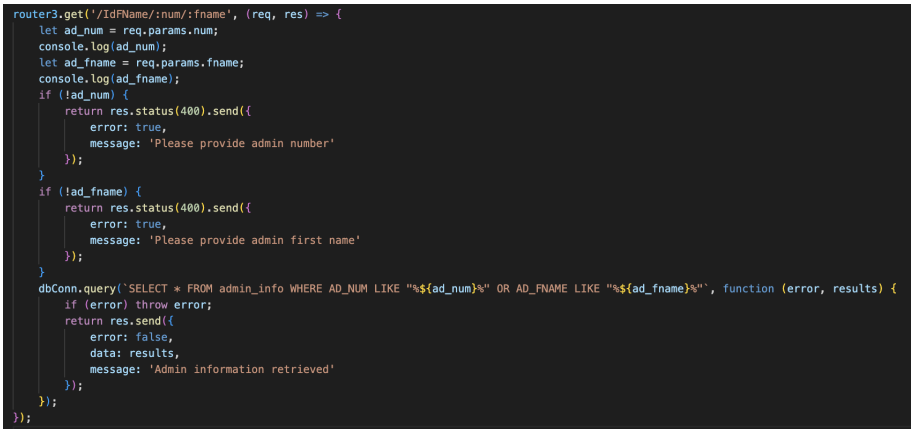
3. Functions for administrators to search/view, insert, update, and delete administrators' information
    - View all admin information

```
router3.get('/admins', (req, res) => {
    dbConn.query("SELECT * FROM admin_info", function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results,
            message: 'Administrator list.'
        });
    });
});
```

Method: GET
URL: http://localhost:3030/admin/admins
Search all the admin information in the database.

- Search by three criteria: Admin Number, Admin First Name, and Admin Last Name

Search by Number	
`router3.get('/Id/:num', (req, res) => { let ad_num = req.params.num; console.log(ad_num); if (!ad_num) { return res.status(400).send({ error: true, message: 'Please provide admin number' }); } dbConn.query(`SELECT * FROM admin_info WHERE AD_NUM LIKE "%${ad_num}%"`, function (error, results) { if (error) throw error; return res.send({ error: false, data: results, message: 'Admin information retrieved' }); }); });`	Method: GET URL: http://localhost:3030/admin/Id/ Recieve Number as parameter and check if there is any admin ID that contains the input ID.
Search by First Name	

	Method: GET URL: http://localhost:3030/admin/fname/ Recieve First Name as parameter and check if there is any admin First Name that contains the input First Name.
Search by Last Name	
	Method: GET URL: http://localhost:3030/admin/lname/ Recieve Last Name as parameter and check if there is any admin Last Name that contains the input Last Name.
Search by Num and First name	
	Method: GET URL: http://localhost:3030/admin/IdFName/ Recieve Num and First Name as parameters and check if there is any admin that contains the input Num or First Name.
Search by Num and Last name	

<table>
<tr>
<td>

```
router3.get('/IdLName/:num/:lname', (req, res) => {
    let ad_num = req.params.num;
    console.log(ad_num);
    let ad_lname = req.params.lname;
    console.log(ad_lname);
    if (!ad_num) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin number'
        });
    }
    if (!ad_lname) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin last name'
        });
    }
    dbConn.query(`SELECT * FROM admin_info WHERE AD_NUM LIKE "%${ad_num}%" OR AD_LNAME LIKE "%${ad_lname}%"`, function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results,
            message: 'Admin information retrieved'
        });
    });
});
```

</td>
<td>Method: GET
URL: http://localhost:3030/admin/IdLName/

Recieve Num and Last Name as parameters and check if there is any admin that contains the input Num or Last Name.</td>
</tr>
<tr>
<td>Search by First name and Last name</td>
<td></td>
</tr>
<tr>
<td>

```
router3.get('/FNameLName/:fname/:lname', (req, res) => {
    let ad_fname = req.params.fname;
    console.log(ad_fname);
    let ad_lname = req.params.lname;
    console.log(ad_lname);
    if (!ad_fname) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin first name'
        });
    }
    if (!ad_lname) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin last name'
        });
    }
    dbConn.query(`SELECT * FROM admin_info WHERE AD_FNAME LIKE "%${ad_fname}%" OR AD_LNAME LIKE "%${ad_lname}%"`, function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results,
            message: 'Admin information retrieved'
        });
    });
});
```

</td>
<td>Method: GET
URL: http://localhost:3030/admin/FNameLName/

Recieve First Name and Last Name as parameters and check if there is any admin that contains the input First Name or Last Name.</td>
</tr>
<tr>
<td>Search by Num, First name, and Last name</td>
<td></td>
</tr>
<tr>
<td>

```
router3.get('/IdFNameLName/:num/:fname/:lname', (req, res) => {
    let ad_num = req.params.num;
    console.log(ad_num);
    let ad_fname = req.params.fname;
    console.log(ad_fname);
    let ad_lname = req.params.lname;
    console.log(ad_lname);
    if (!ad_num) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin number'
        });
    }
    if (!ad_fname) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin first name'
        });
    }
    if (!ad_lname) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin last name'
        });
    }
    dbConn.query(`SELECT * FROM admin_info WHERE AD_NUM LIKE "%${ad_num}%" OR AD_FNAME LIKE "%${ad_fname}%" OR AD_LNAME LIKE "%${ad_lname}%"`,
        if (error) throw error;
        return res.send({
            error: false,
            data: results,
            message: 'Admin information retrieved'
        });
    });
});
```

</td>
<td>Method: GET
URL: http://localhost:3030/admin/IdFNameLName/

Recieve Num, First Name, and Last Name as parameters and check if there is any admin that contains the input Num or First Name or Last Name.</td>
</tr>
</table>

- Insert new admin

```
router3.post('/add', (req, res) => {
    let adminInfo = req.body.Admin_info;
    let adminLogin = req.body.Admin_login;
    if (!adminInfo || !adminLogin) {
        return res.status(400).send({
            error: true,
            message: 'Please provide new admin information'
        });
    }
    dbConn.query("INSERT INTO admin_info SET ?", [adminInfo], function (error, results) {
        if (error) throw error;
    });
    dbConn.query("INSERT INTO admin_login SET ?", [adminLogin], function (error, results) {
        if (error) throw error;
    });
    return res.send({
        error: false,
        message: 'New administrator information has been created successfully'
    });
});
```

Method: POST

URL: http://localhost:3030/admin/add

Inert a new administrator into the database. Receive the information in the body of Admin_info and Admin_login. The information have to be separate because the information of admin is divided into two database tables.

- Update an admin

```
// Update
router3.put('/update', (req, res) => {
    let admin_num = req.body.Admin_info.AD_NUM;
    let admin = req.body.Admin_info;
    console.log(admin_num);
    console.log(admin);
    if (!admin || !admin_num) {
        return res.status(400).send({
            error: true,
            message: 'Please provide admin information'
        });
    }
    dbConn.query("UPDATE admin_info SET ? WHERE AD_NUM = ?", [admin, admin_num], function (error, results) {
        if (error) throw error;
        return res.send({
            error: false,
            data: results.affectedRows,
            message: 'Administrator information has been updated successfully'
        });
    });
});
```

Method: PUT

URL: http://localhost:3030/admin/update

Update information of administrator that already existed in the database based on the given Admin Number. Receive the information in the body of Admin_info.

- Delete an admin

```
// Delete
router3.delete('/delete', (req, res) => {
    let admin_num = req.body.Admin.AD_NUM;
    console.log(admin_num);
    if (!admin_num) {
        return res.status(400).send({
            error: true,
            message: 'Please provide Admin number'
        });
    }
    dbConn.query("DELETE FROM admin_login WHERE AD_NUM = ?", [admin_num], function (error, results) {
        if (error) throw error;
    });
    dbConn.query("DELETE FROM admin_info WHERE AD_NUM = ?", [admin_num], function (error, results) {
        if (error) throw error;
    });
    return res.send({
        error: false,
        message: 'Administrator information has been deleted successfully'
    });
});
```

Method: DELETE

URL: http://localhost:3030/admin/delete

Delete information of administrator based on the given Admin Number. Receive the information in the body of Admin.

Javascript to call web services from backend to be interactive in the frontend	
	Function to call web service to execute select, insert, update, and delete.

```
async function callWS(url, method, sentData = {}) {
    console.log("Calling backend web service awdawd")
    let data;
    if (method == "select") {
        console.log("select")
        let response = await fetch(url, {
            method: 'GET'
        });
        data = await response.json();
    }
    else if (method == "insert") {
        let response = await fetch(url, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(sentData)
        });
        data = await response.json();

    }
    else if (method == "update") {
        let response = await fetch(url, {
            method: 'PUT',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(sentData)
        });
        data = await response.json();
    }
    else if (method == "delete") {
        let response = await fetch(url, {
            method: 'DELETE',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(sentData)
        });
        data = await response.json();
    }
    return data;
}
```

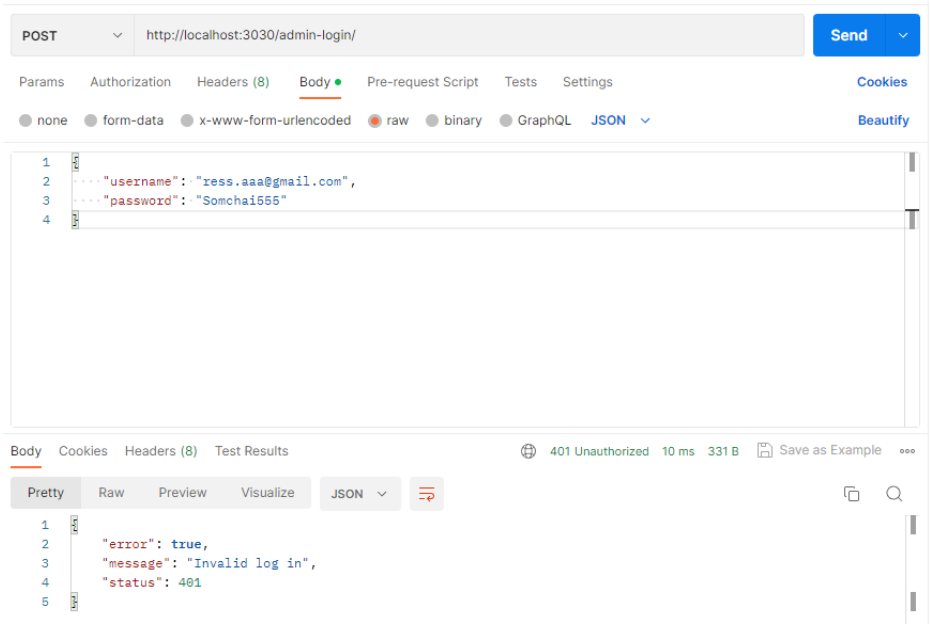
```
const rootURLAdminLogin = "http://localhost:3030/admin-login";
async function loginHandler() {
    const adminEmail = document.querySelector("#email").value;
    const adminPass = document.querySelector("#password").value;
    const data = {
        username: adminEmail,
        password: adminPass
    };
    const response =  await callWS(rootURLAdminLogin, "insert", data)
    if (response.status === 200) {
        console.log("Login successful");
        alert("Login successful");
        window.location.href = "/adminhome";
    } else if (response.status === 401) {
        console.log(response.message);
        alert('Invalid, Please try again');
    }
    else{
        alert('Please enter username and password');
    }
}
```

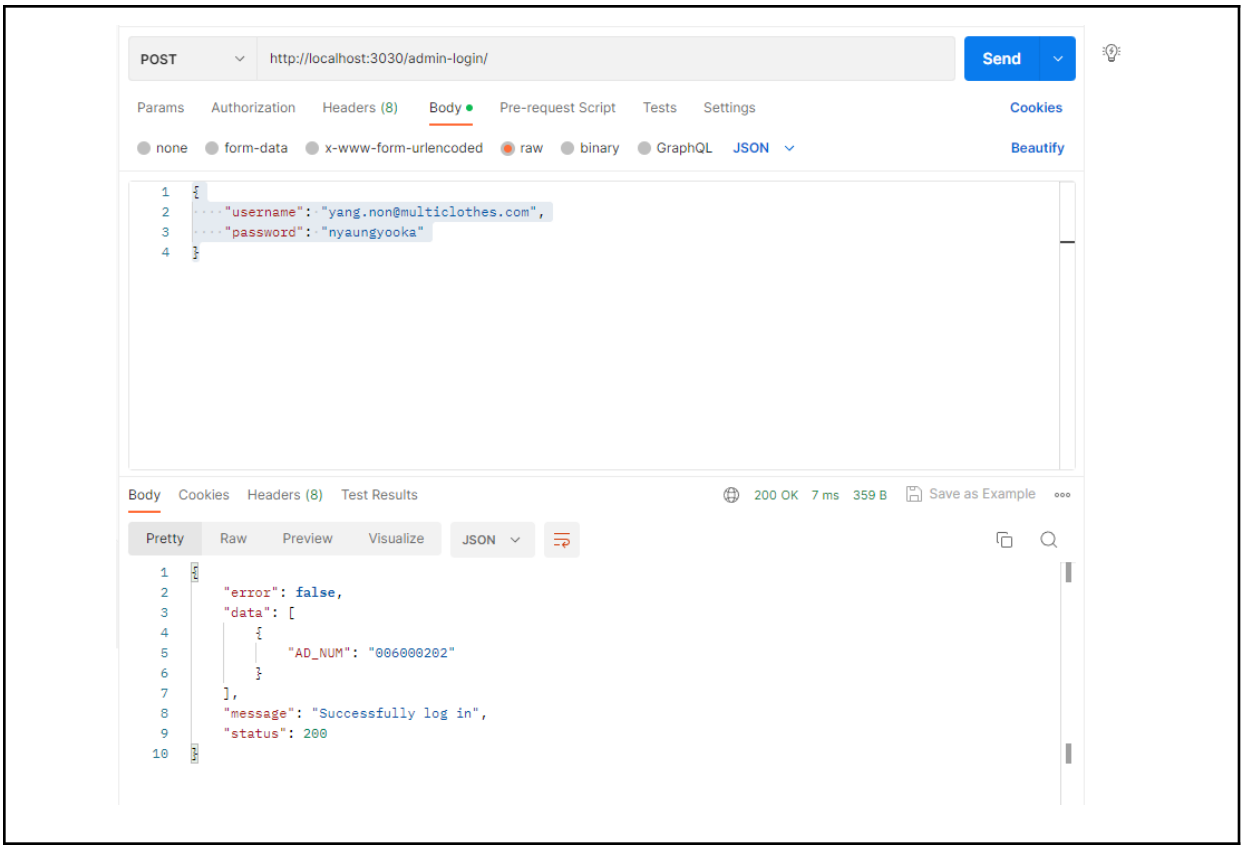
This is an example for using function callWS() to call web service to perform login authentication. Once the response status is sent from the bankend, it will check if the response is successful (200). If it is successful, it will alert "Login successful and direct to Welcome Admin Page". If not it will alert "Invalid, Please try again".

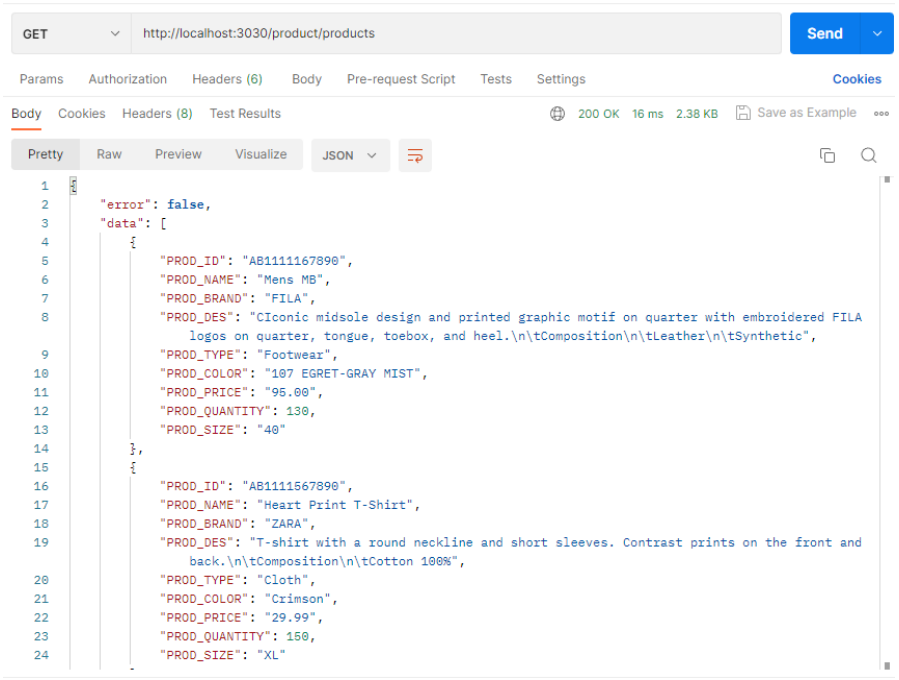
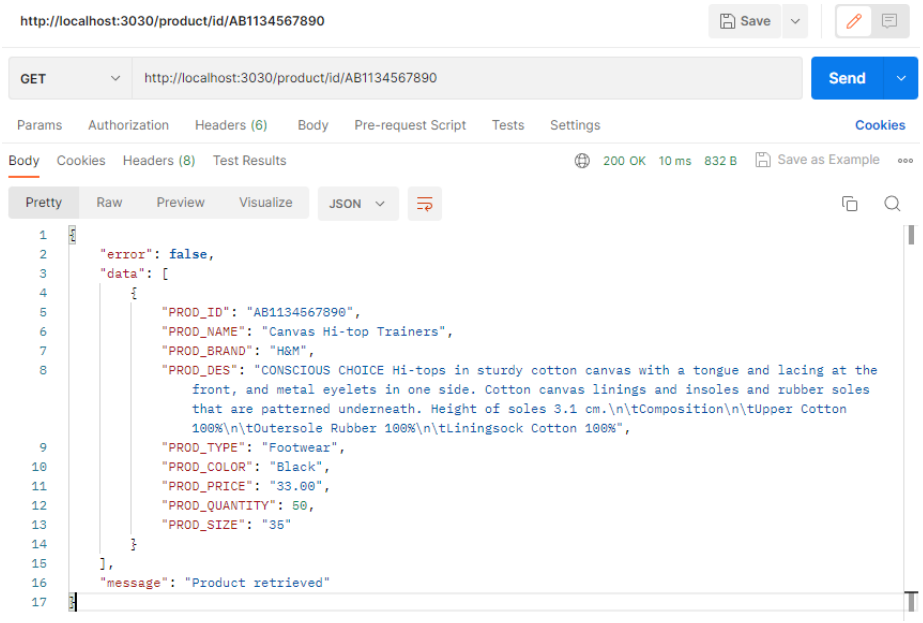
```
const rootURLPd = "http://localhost:3030/product";
let searchBtnRef = document.querySelector("#Submit");
searchBtnRef.addEventListener("click", () => {
    const prodId = document.querySelector("#Id").value;
    const prodName = document.querySelector("#Name").value;
    const prodBrand = document.querySelector("#Brand").value;
    const outputHtml = document.getElementById("output");
    // Select all
    if (!prodId && !prodName && !prodBrand) {
        callWS(rootURLPd + "/products", "select").then((data) => {
            console.log(data);
            outputHtml.innerHTML = '';
            if (data.data.length > 0) {
                alert(data.message);
                let output;
                output = "<h1 class='list'>Product List<h1>";
                output += "<article class='container'>";
                data.data.forEach((element) => {
                    output += "<article class='column'>";
                    output += "<h2>" + element.PROD_NAME + "</h2>";
                    output += "<p> ID: " + element.PROD_ID + "</p>";
                    output += "<p> Brand: " + element.PROD_BRAND + "</p>";
                    output += "<p> Description: " + element.PROD_DES + "</p>";
                    output += "<p> Type: " + element.PROD_TYPE + "</p>";
                    output += "<p> Color: " + element.PROD_COLOR + "</p>";
                    output += "<p class='price'>Price: " + element.PROD_PRICE + "</p>";
                    output += "<p> Quantity: " + element.PROD_QUANTITY + "</p>";
                    output += "<p> Size: " + element.PROD_SIZE + "</p>";
                    output += "</article>";
                })
                output += "</article>";
                console.log(data.data.length);
                outputHtml.innerHTML = output;
            }
        })
    }
```

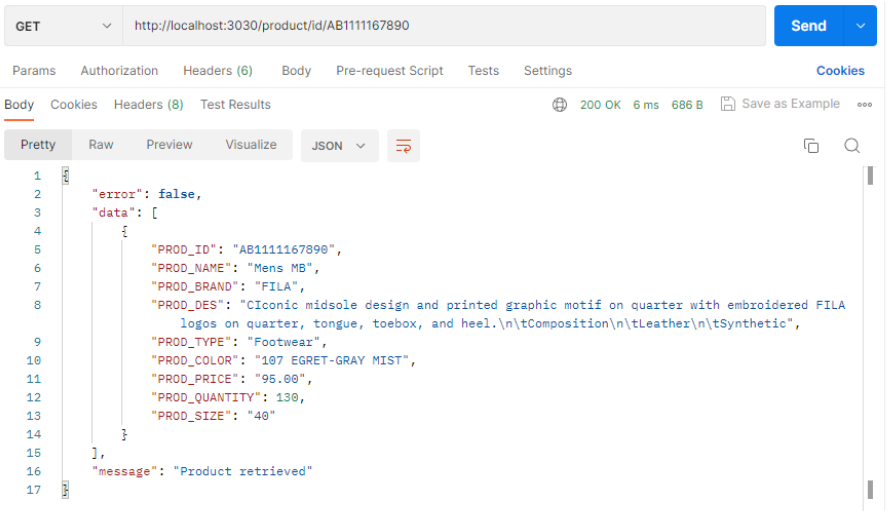
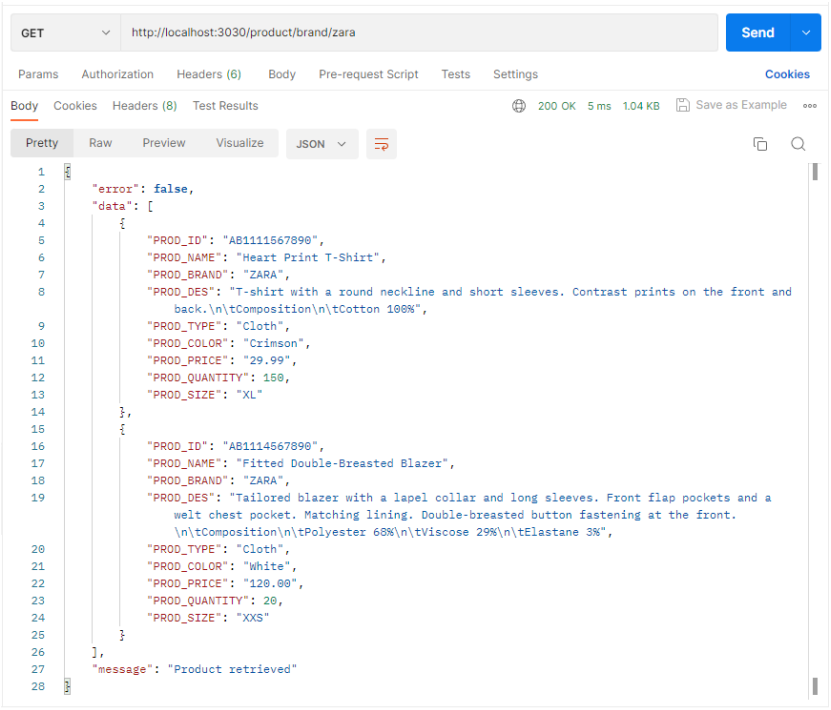
This is an example for using function callWS() to call web service to perform select. Once the response is sent from the bankend, it will appear in the web page by using innerHTML.

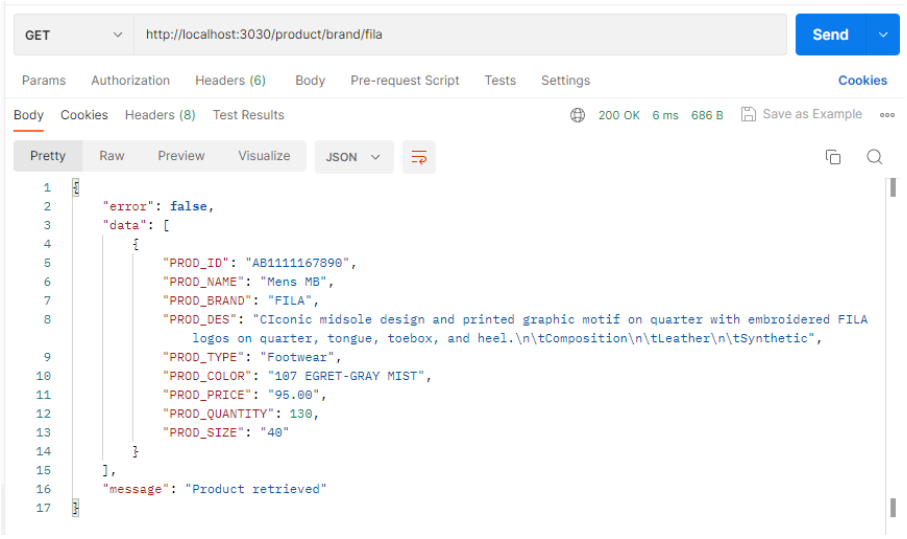
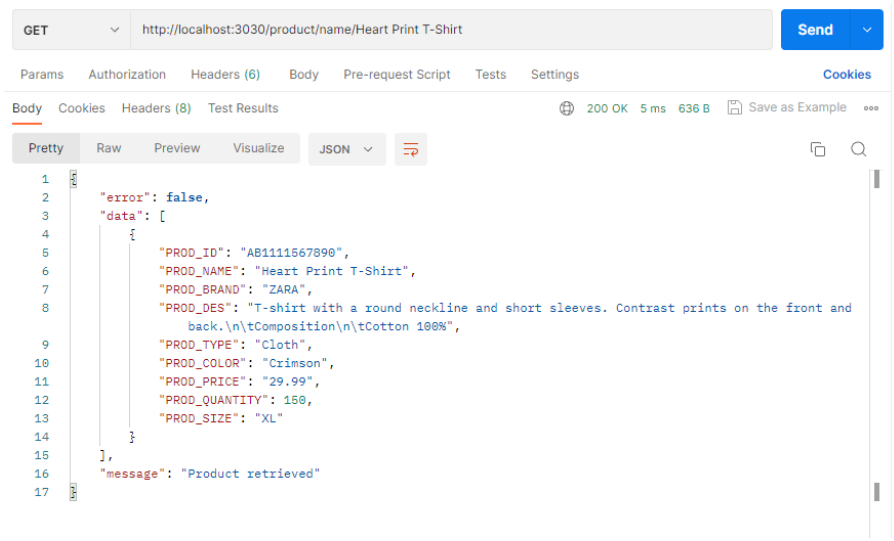
Testing results of web services

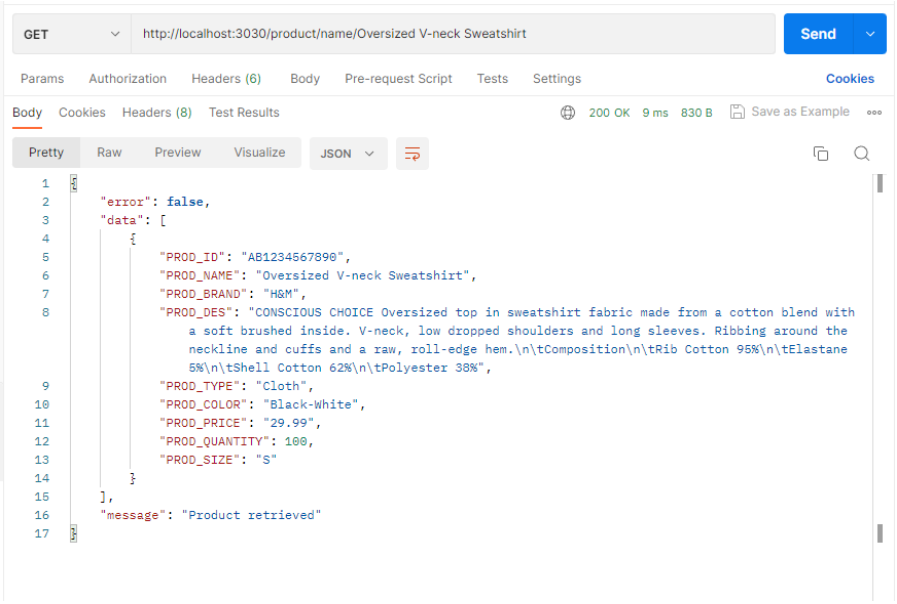
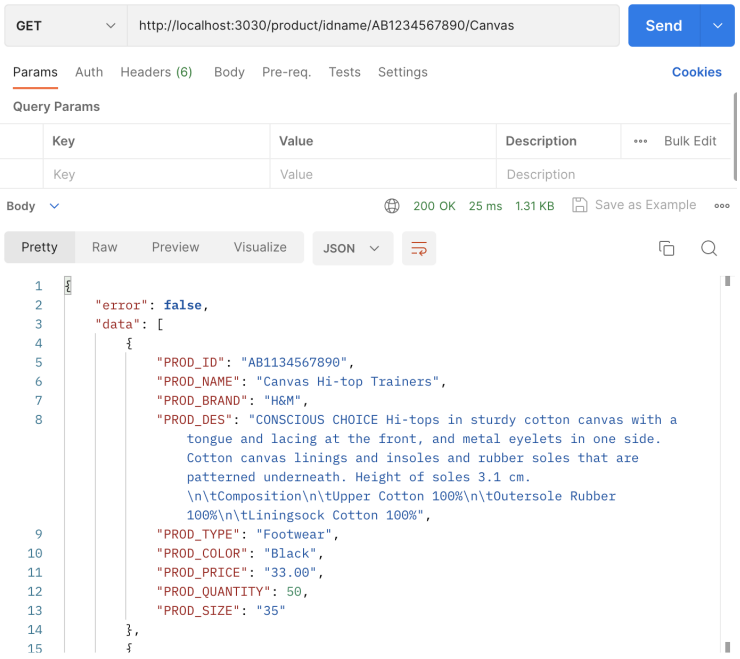
Test Cases for admin login authentication


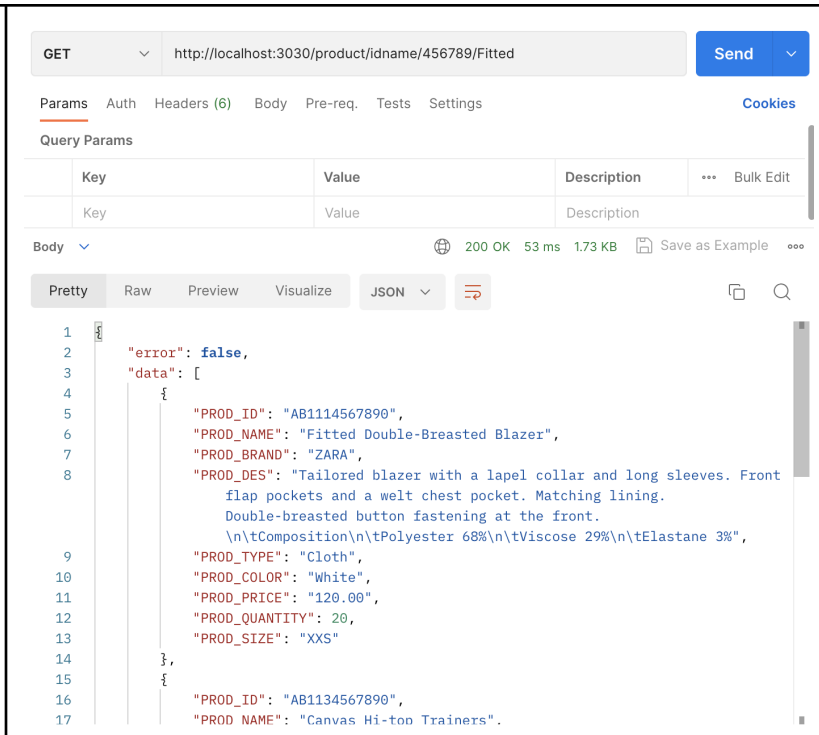

POST
http://localhost:3030/admin-login/
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"username": "yang.non@multiclothes.com",
"password": "nyaungyooka"
}
Body
Cookies
Headers (8)
Test Results
200 OK
7 ms
359 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"error": false,
"data": [
{
"AD_NUM": "006000202"
}
],
"message": "Successfully log in",
"status": 200
}

Test Cases for product management	
Search All	
Search by ID	

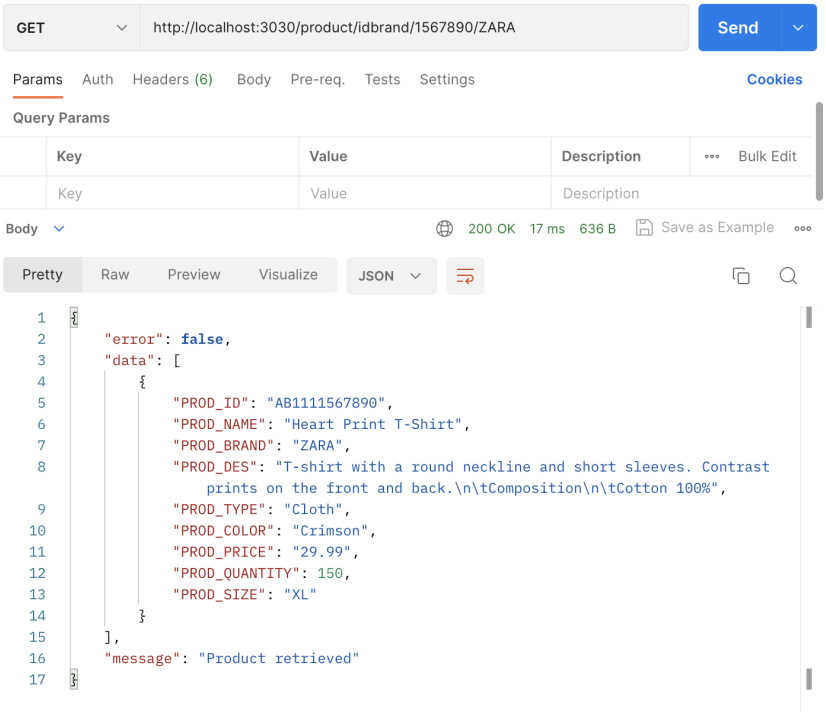
	
Search by Brand	

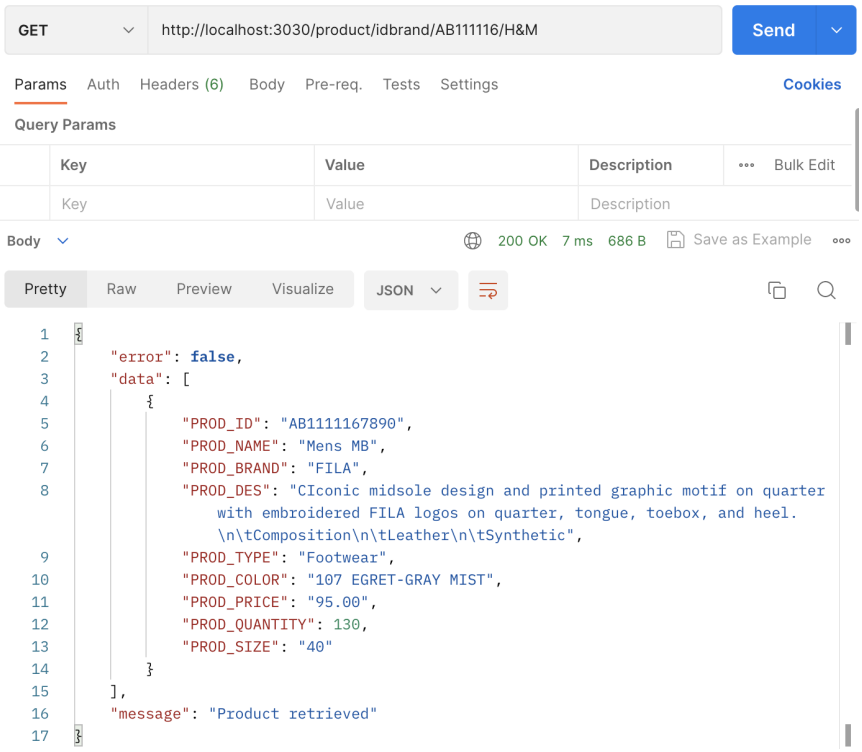
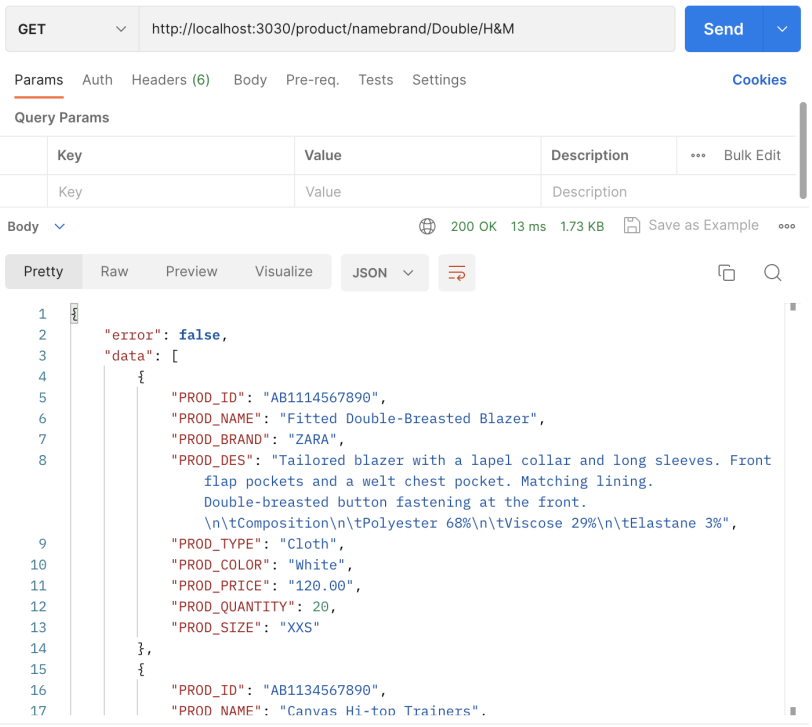
	
Search by name	

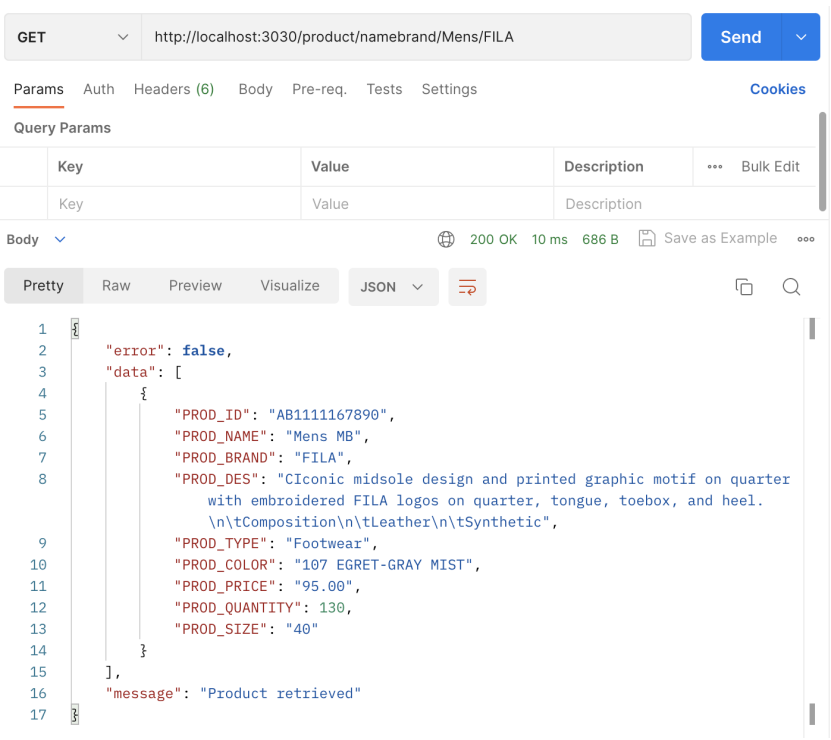
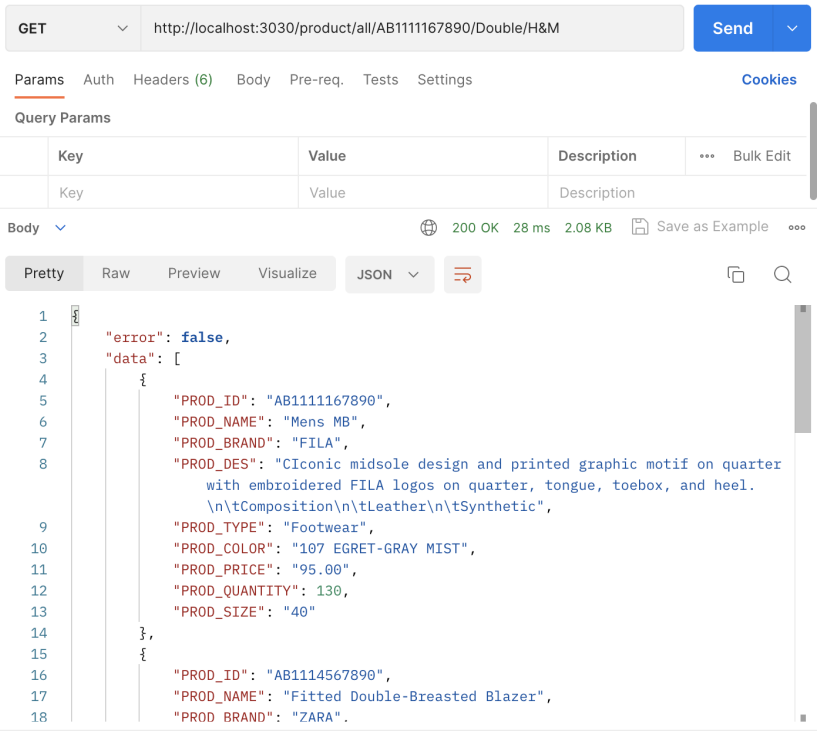
	
Search by ID and name	

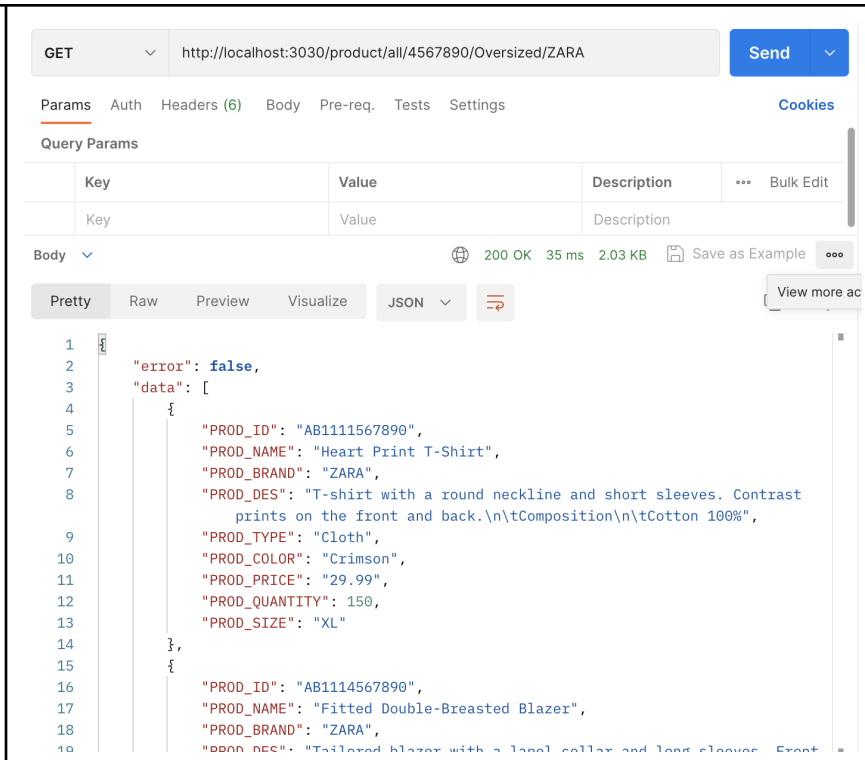


Search by Id and brand

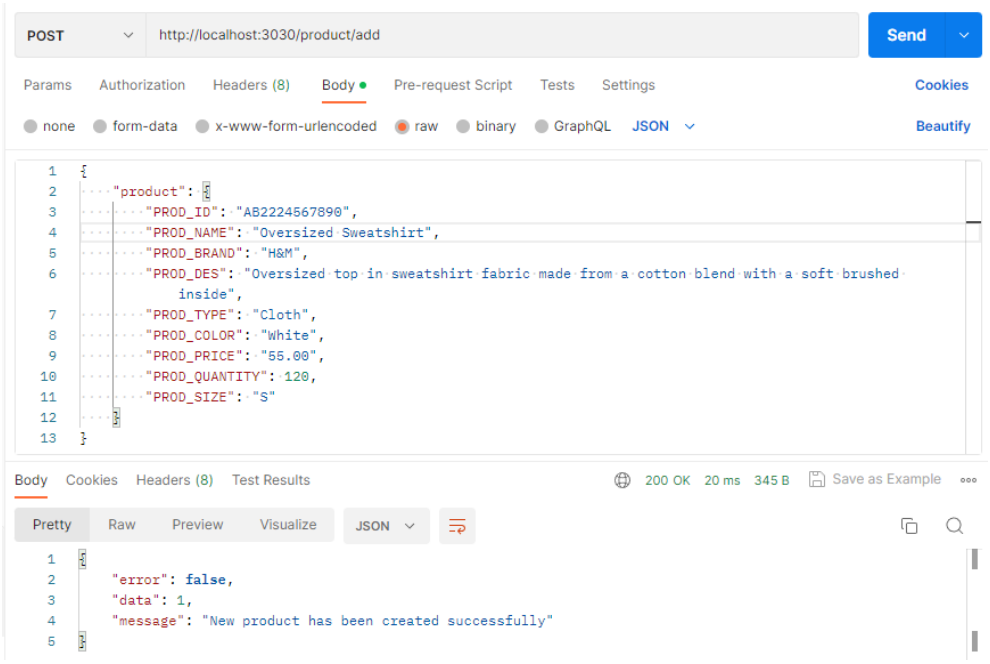


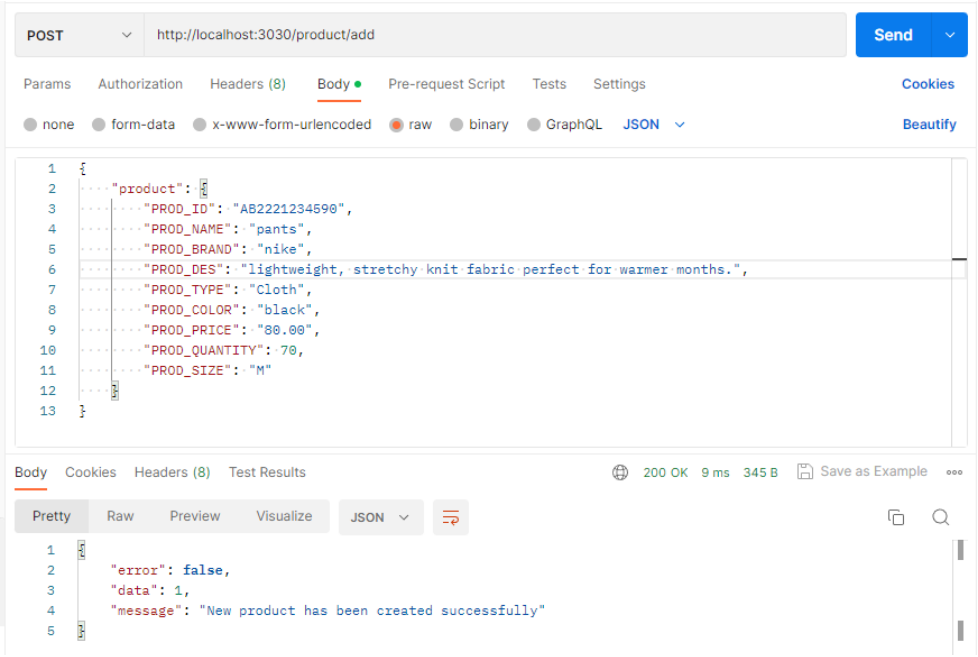
	
Search by name and brand	

	
Search by Id, name, and brand	

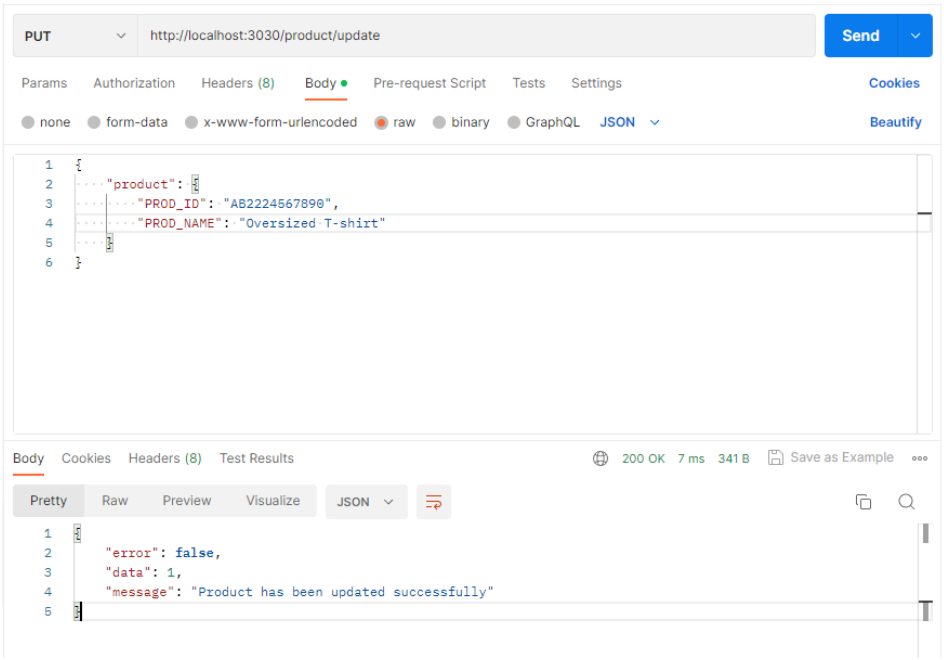

GET
http://localhost:3030/product/all/4567890/Oversized/ZARA
Send
Params Auth Headers (6) Body Pre-req. Tests Settings
Cookies
Query Params
Key Value Description Bulk Edit
Body
200 OK 35 ms 2.03 KB
Save as Example
Pretty Raw Preview Visualize JSON
View more ac
{
"error": false,
"data": [
{
"PROD_ID": "AB1111567890",
"PROD_NAME": "Heart Print T-Shirt",
"PROD_BRAND": "ZARA",
"PROD_DES": "T-shirt with a round neckline and short sleeves. Contrast prints on the front and back.\n\tComposition\n\tCotton 100%",
"PROD_TYPE": "Cloth",
"PROD_COLOR": "Crimson",
"PROD_PRICE": "29.99",
"PROD_QUANTITY": 150,
"PROD_SIZE": "XL"
},
{
"PROD_ID": "AB1114567890",
"PROD_NAME": "Fitted Double-Breasted Blazer",
"PROD_BRAND": "ZARA",

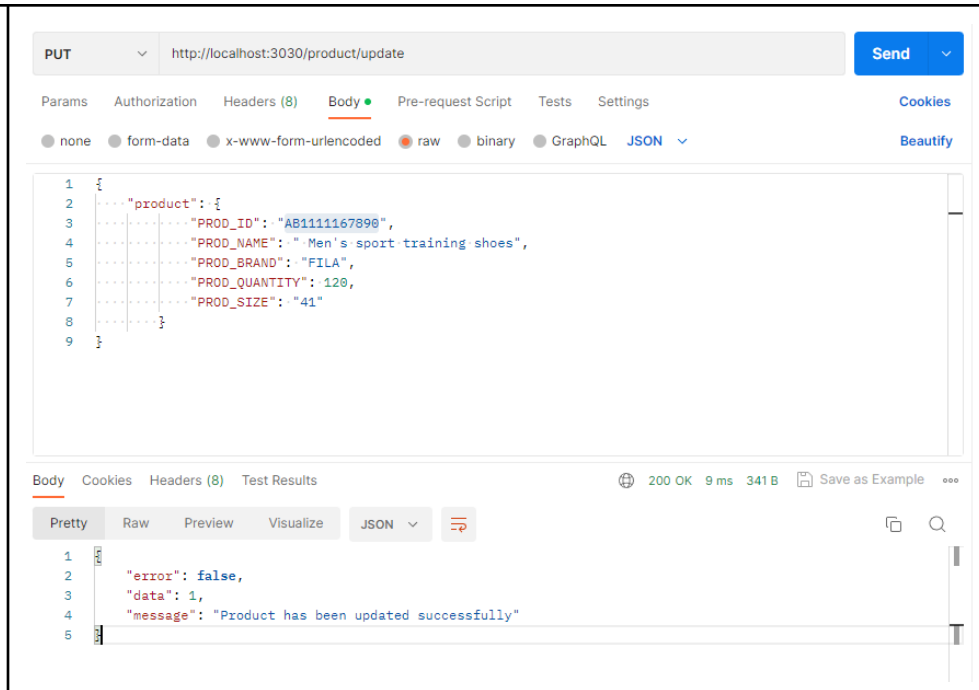
Insert


POST
http://localhost:3030/product/add
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
{
"product": {
"PROD_ID": "AB2224567890",
"PROD_NAME": "Oversized Sweatshirt",
"PROD_BRAND": "H&M",
"PROD_DES": "Oversized top in sweatshirt fabric made from a cotton blend with a soft brushed inside",
"PROD_TYPE": "Cloth",
"PROD_COLOR": "White",
"PROD_PRICE": "55.00",
"PROD_QUANTITY": 120,
"PROD_SIZE": "S"
}
}
Body Cookies Headers (8) Test Results
200 OK 20 ms 345 B
Save as Example
Pretty Raw Preview Visualize JSON
{
"error": false,
"data": 1,
"message": "New product has been created successfully"
}

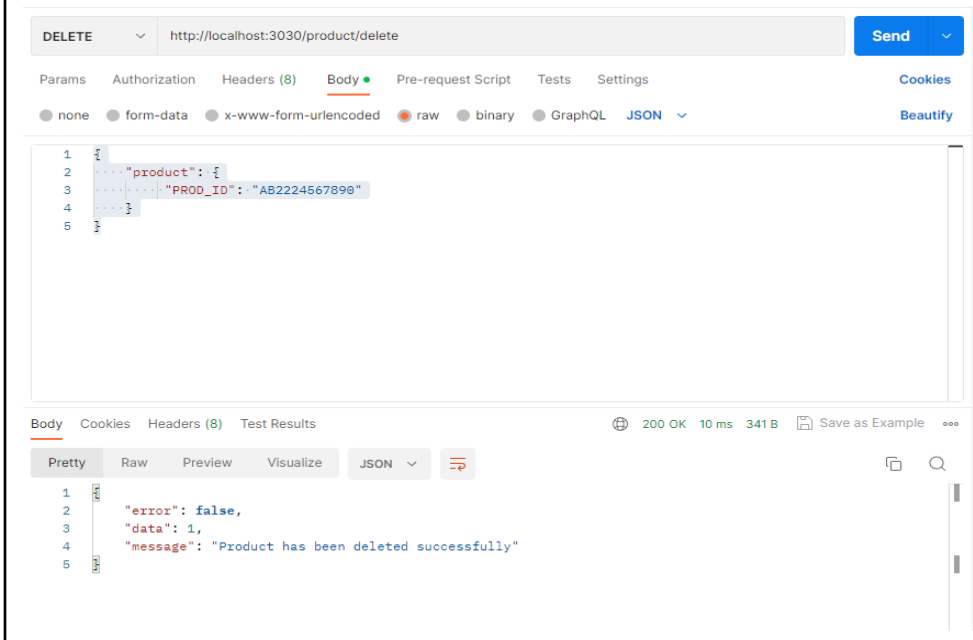

POST
http://localhost:3030/product/add
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"product": {
"PROD_ID": "AB2221234590",
"PROD_NAME": "pants",
"PROD_BRAND": "nike",
"PROD_DES": "lightweight, stretchy knit fabric perfect for warmer months.",
"PROD_TYPE": "Cloth",
"PROD_COLOR": "black",
"PROD_PRICE": "80.00",
"PROD_QUANTITY": 70,
"PROD_SIZE": "M"
}
}
Body
Cookies
Headers (8)
Test Results
200 OK
9 ms
345 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"error": false,
"data": 1,
"message": "New product has been created successfully"
}

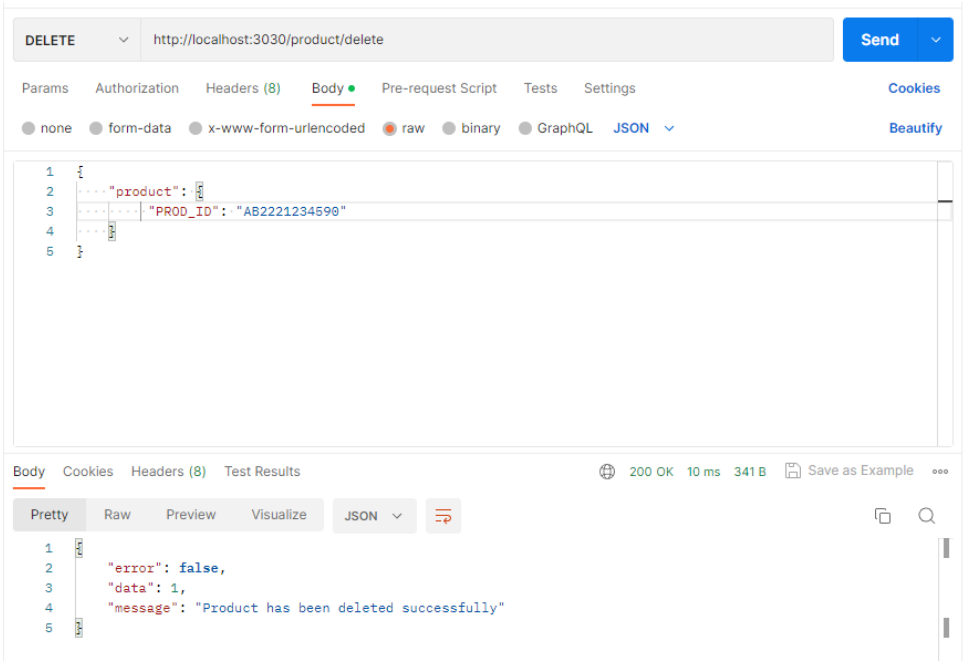
Update

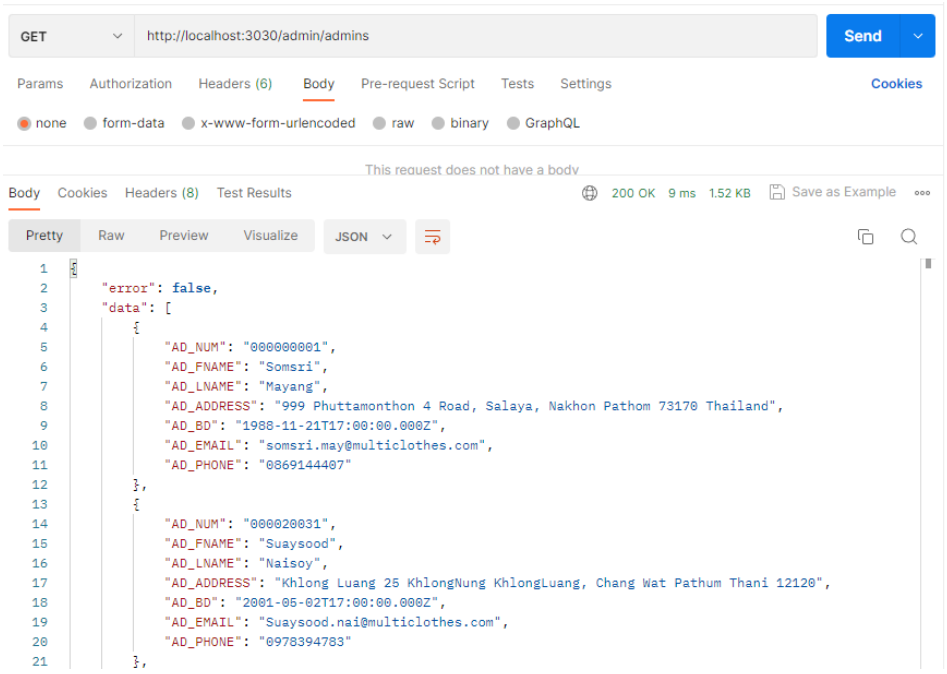
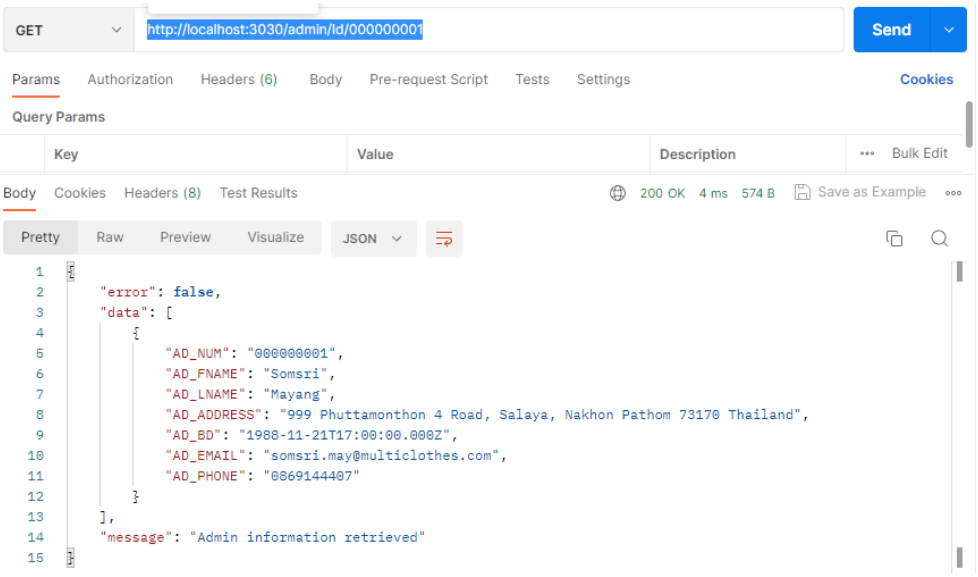

PUT
http://localhost:3030/product/update
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"product": {
"PROD_ID": "AB2224567890",
"PROD_NAME": "Oversized T-shirt"
}
}
Body
Cookies
Headers (8)
Test Results
200 OK
7 ms
341 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"error": false,
"data": 1,
"message": "Product has been updated successfully"
}

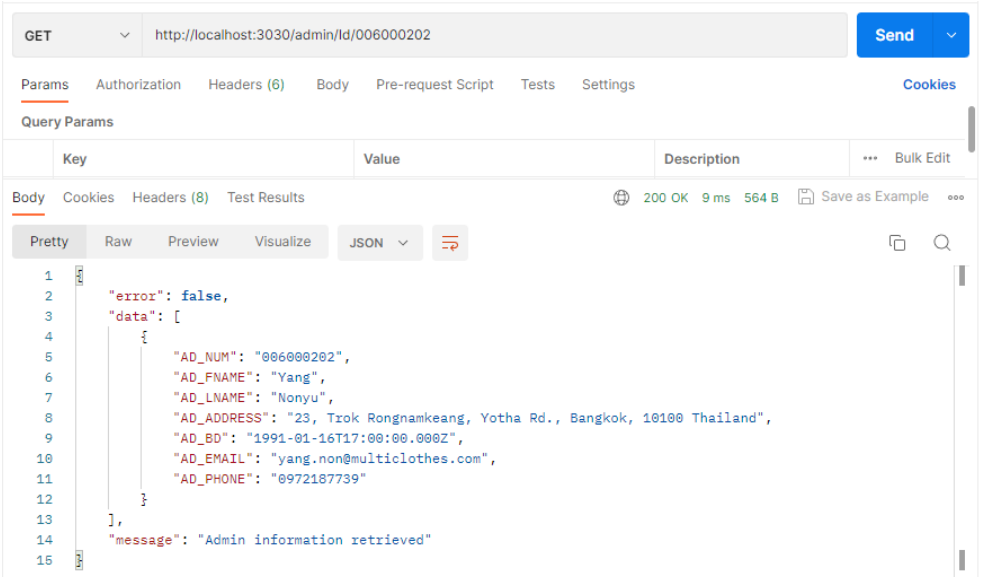
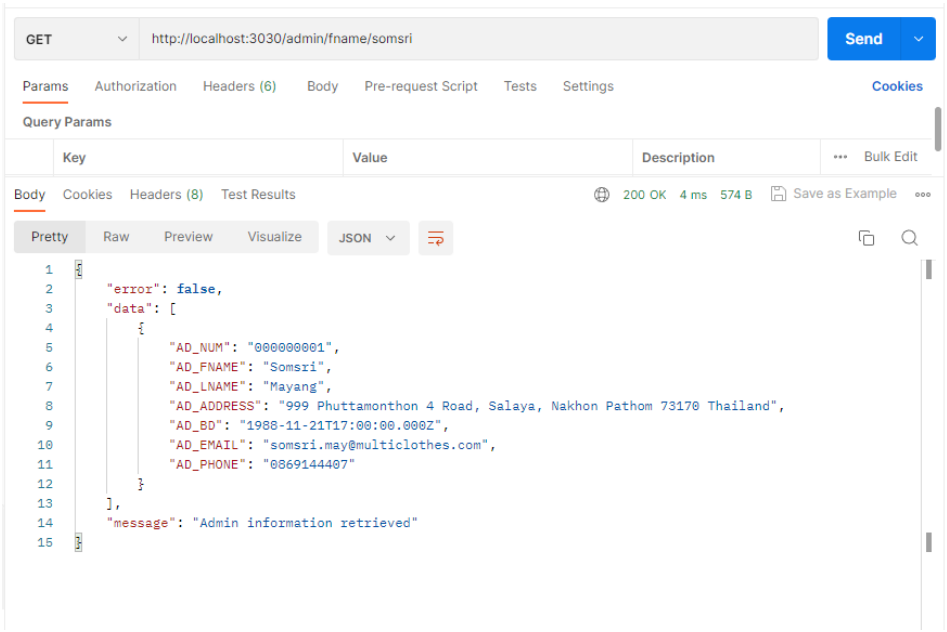

PUT
http://localhost:3030/product/update
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"product": {
"PROD_ID": "AB1111167890",
"PROD_NAME": "Men's sport training shoes",
"PROD_BRAND": "FILA",
"PROD_QUANTITY": 120,
"PROD_SIZE": "41"
}
}
Body
Cookies
Headers (8)
Test Results
200 OK
9 ms
341 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"error": false,
"data": 1,
"message": "Product has been updated successfully"
}

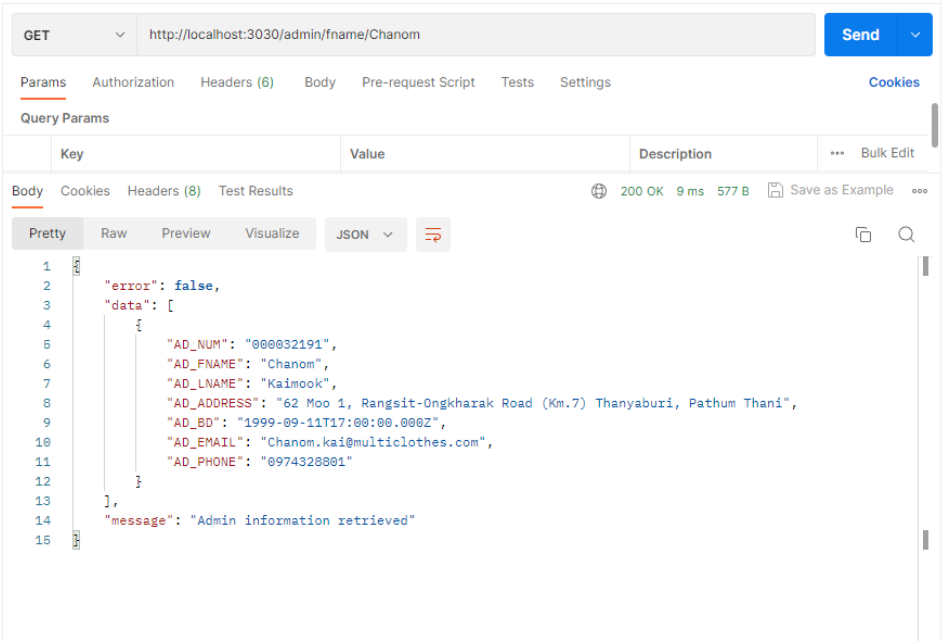
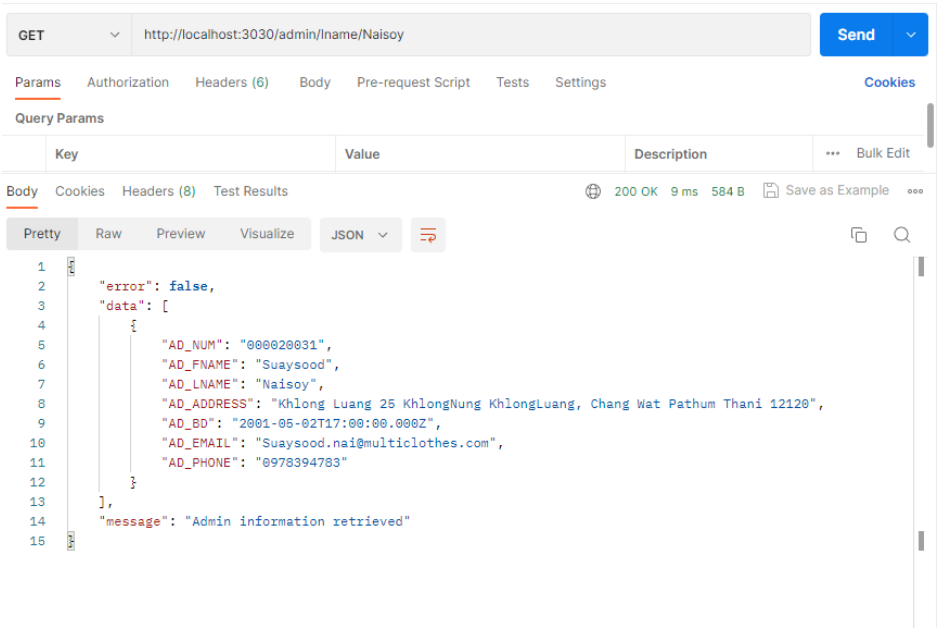
Delete

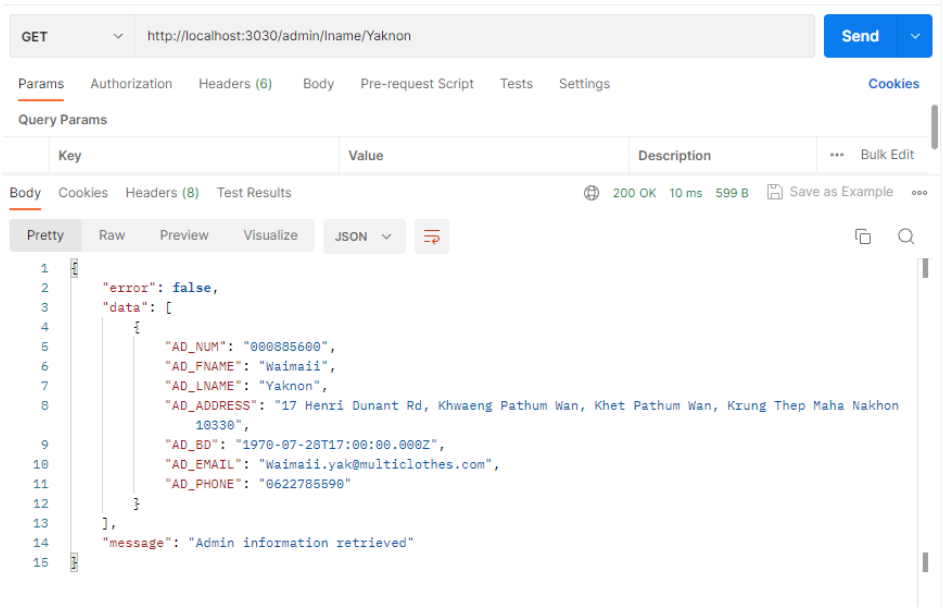
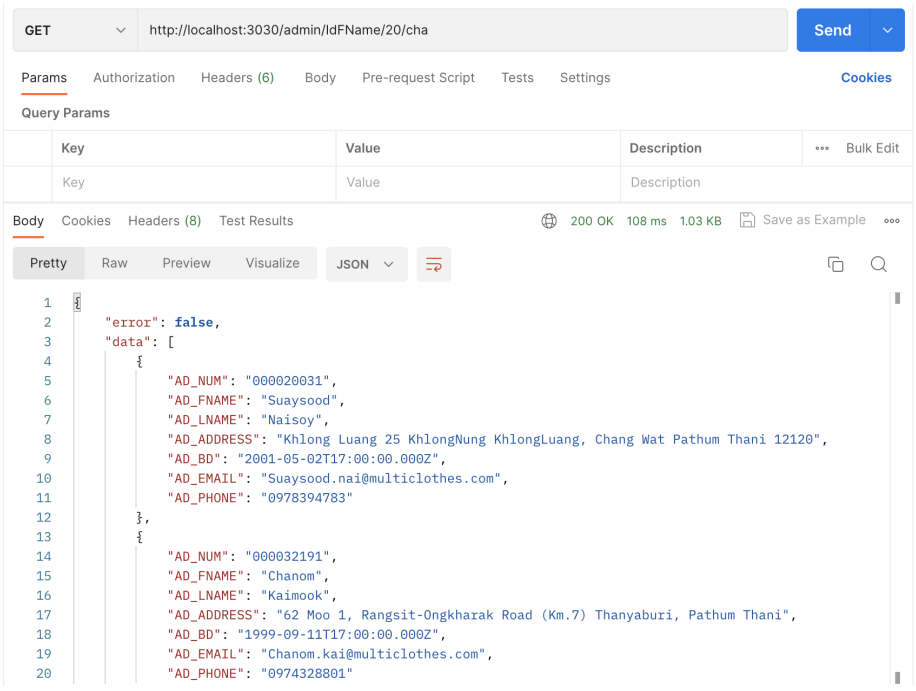

DELETE
http://localhost:3030/product/delete
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"product": {
"PROD_ID": "AB2224567890"
}
}
Body
Cookies
Headers (8)
Test Results
200 OK
10 ms
341 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"error": false,
"data": 1,
"message": "Product has been deleted successfully"
}

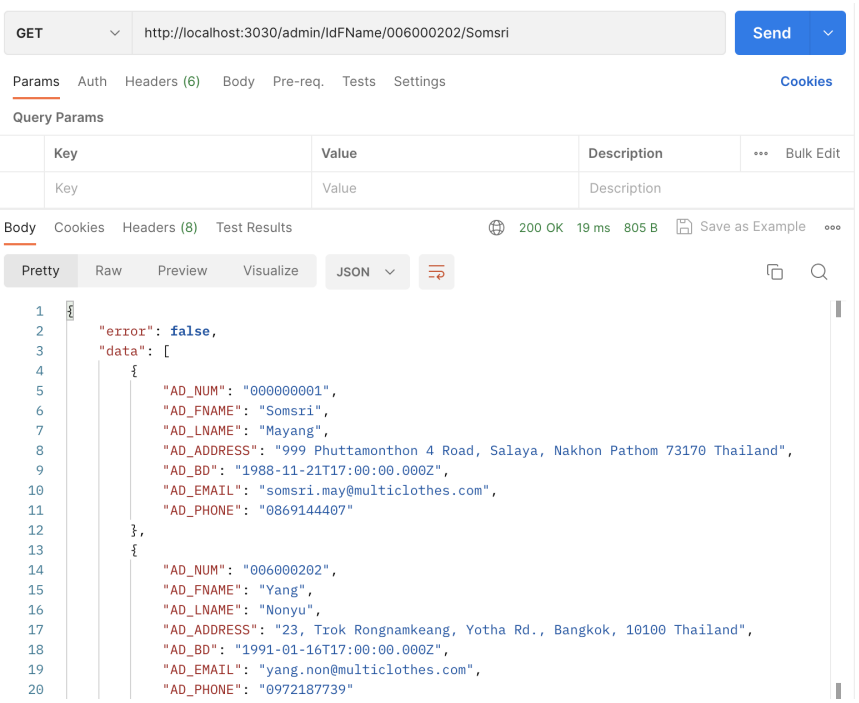
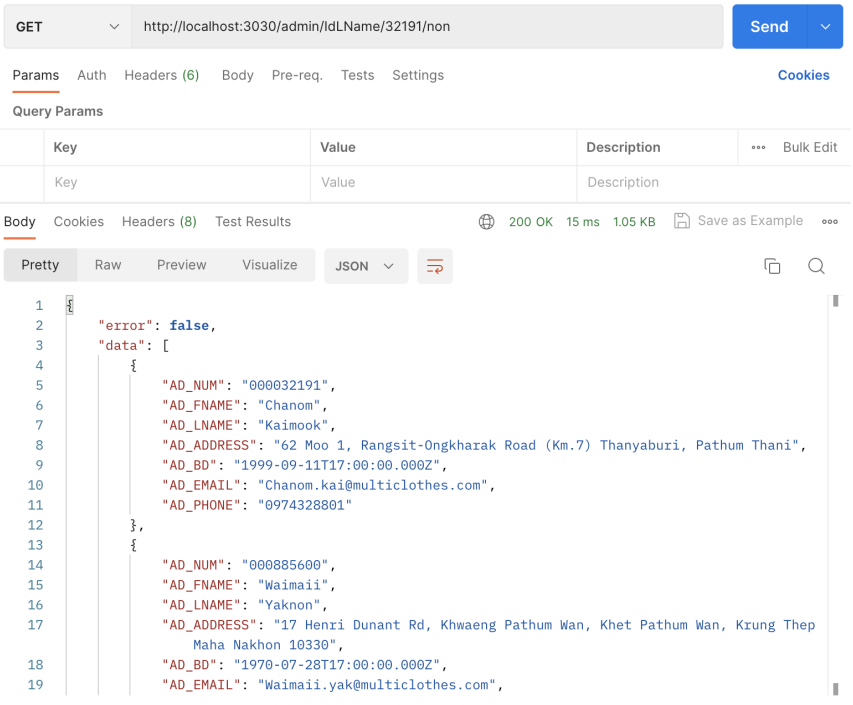

DELETE
http://localhost:3030/product/delete
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"product": {
"PROD_ID": "AB2221234590"
}
}
Body
Cookies
Headers (8)
Test Results
200 OK
10 ms
341 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"error": false,
"data": 1,
"message": "Product has been deleted successfully"
}

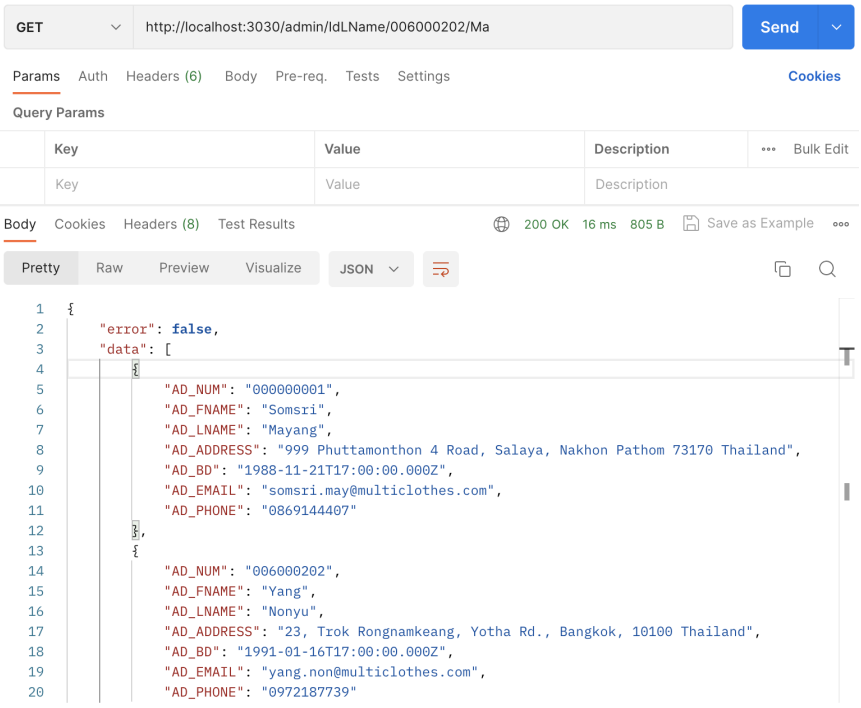
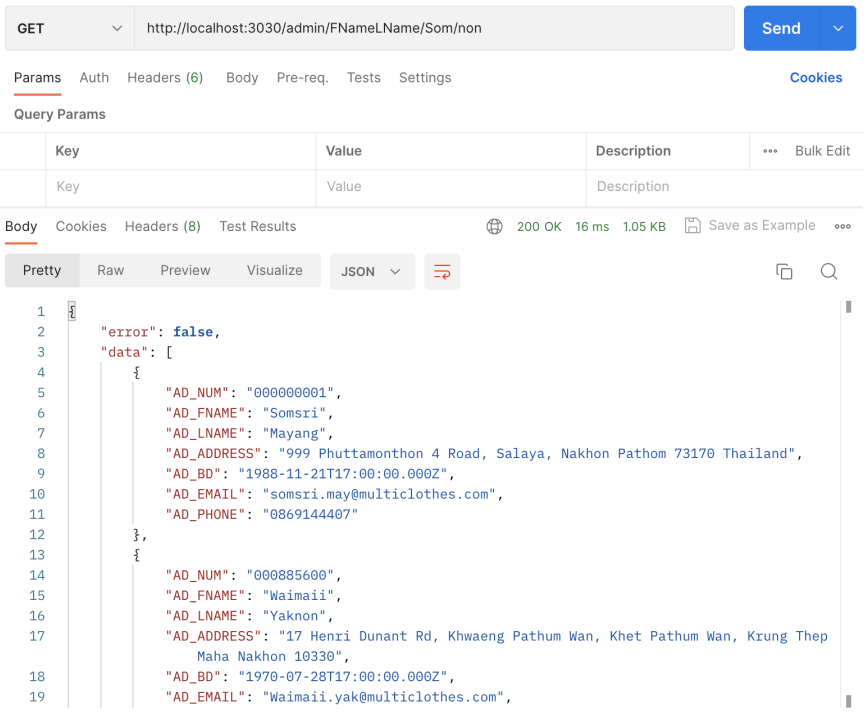
Test Cases for admin information management	
Search All	
Search by Num	

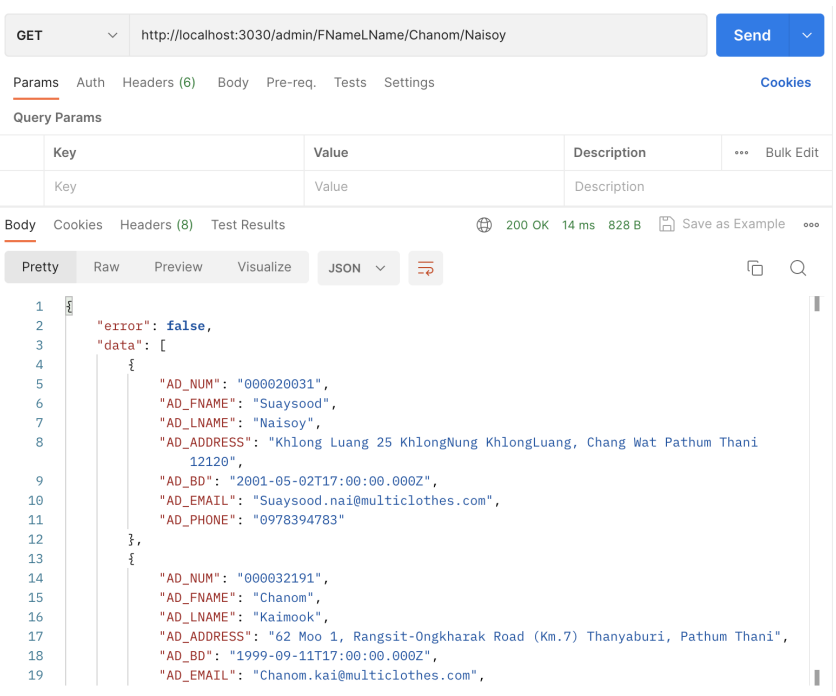
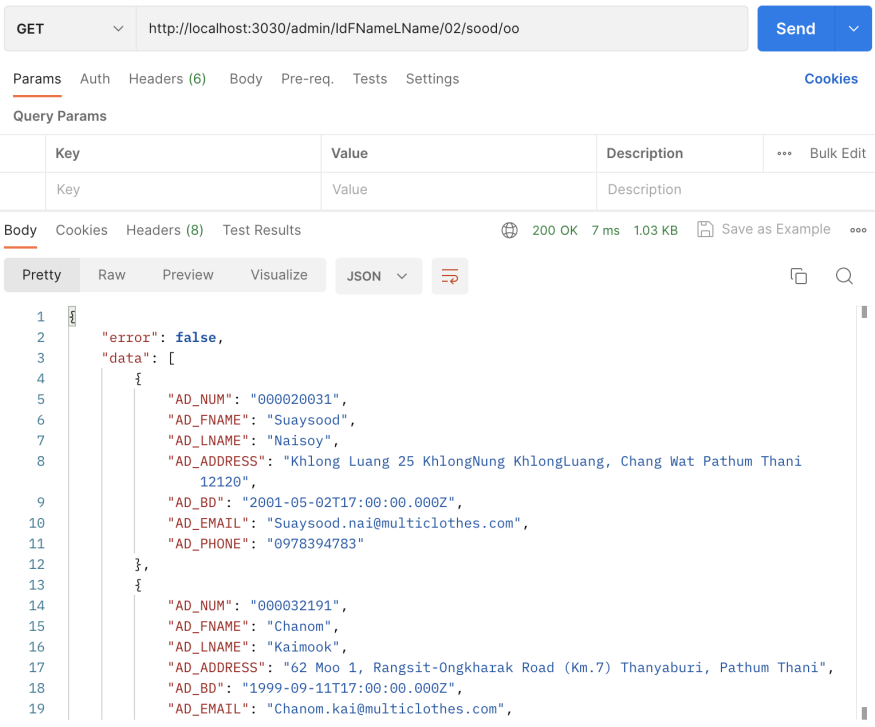
	
Search by First Name	

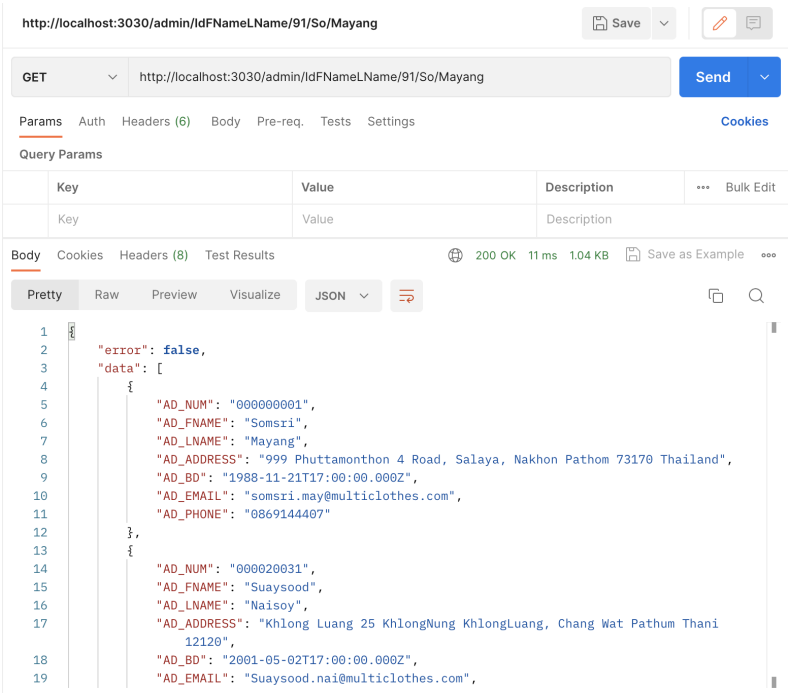
	
Search by Last Name	

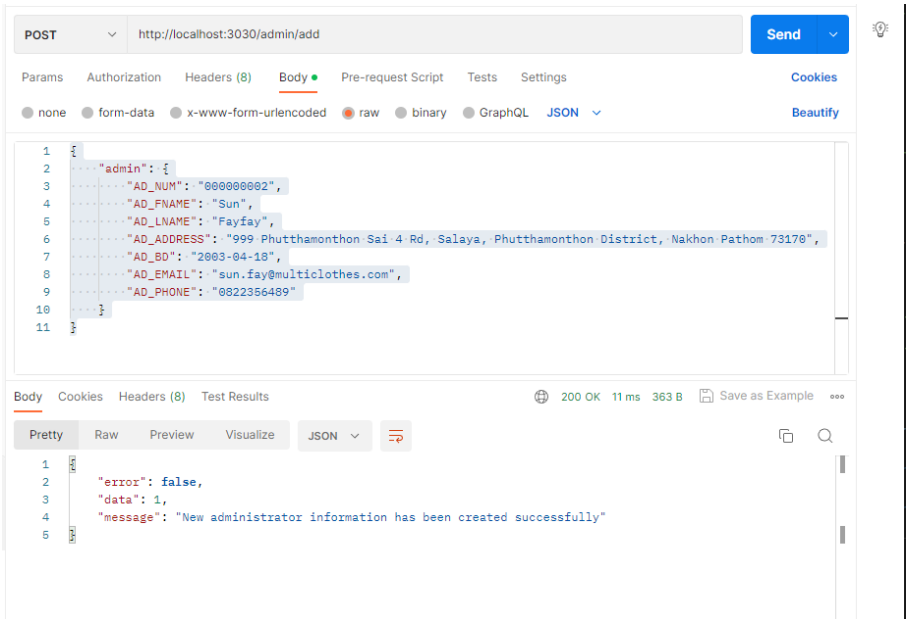
	
Search by ID and First Name	

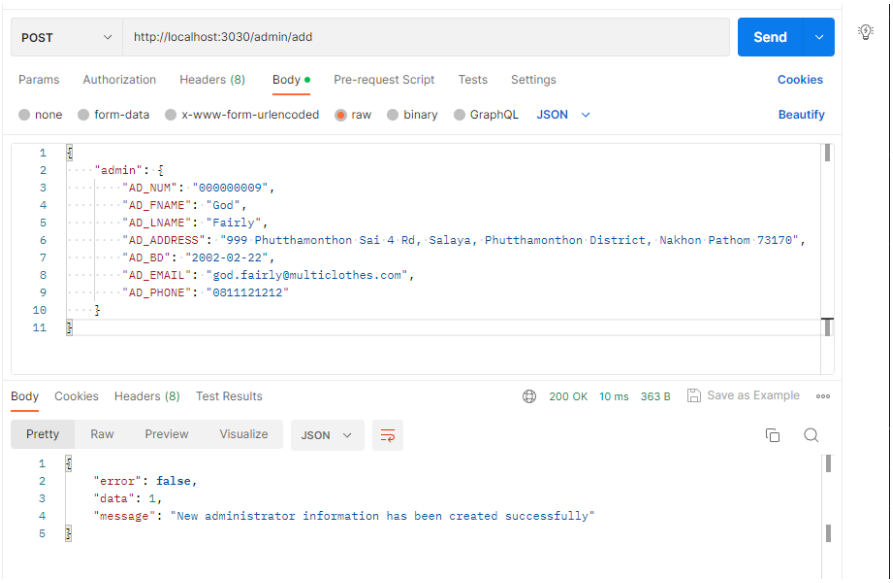
	
Search by ID and Last Name	

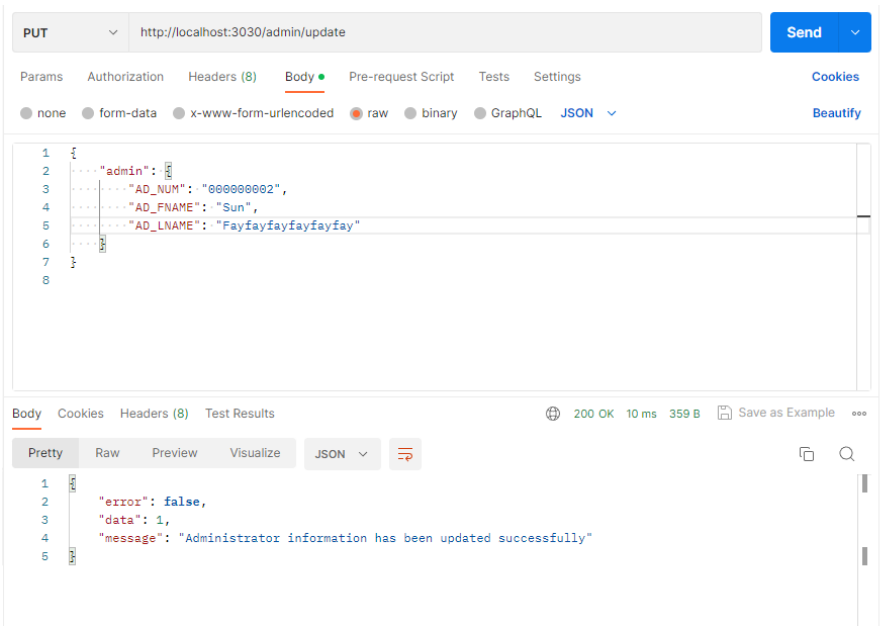
	
Search by First Name and Last Name	

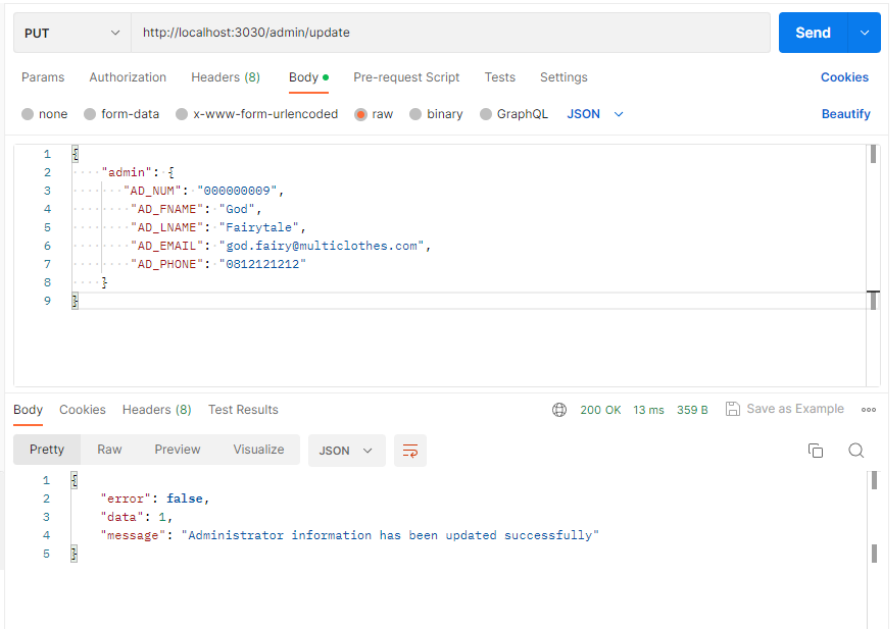
	
Search by ID, First Name, and Last Name	

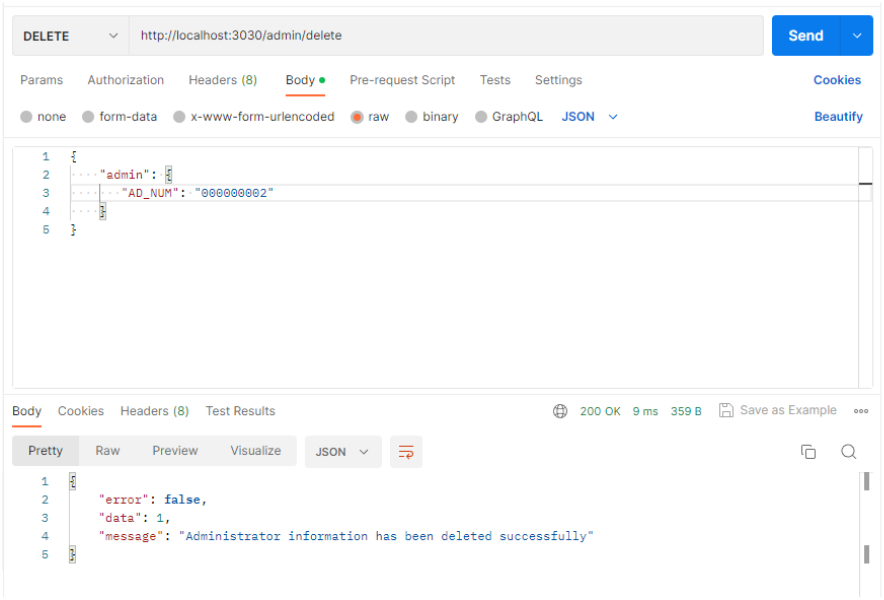
	 http://localhost:3030/admin/IdFNameLName/91/So/Mayang Save GET http://localhost:3030/admin/IdFNameLName/91/So/Mayang Send Params Auth Headers (6) Body Pre-req. Tests Settings Cookies Query Params Key Value Description Bulk Edit Body Cookies Headers (8) Test Results 200 OK 11 ms 1.04 KB Save as Example Pretty Raw Preview Visualize JSON { "error": false, "data": [{ "AD_NUM": "000000001", "AD_FNAME": "Somsri", "AD_LNAME": "Mayang", "AD_ADDRESS": "999 Phuttamonthon 4 Road, Salaya, Nakhon Pathom 73170 Thailand", "AD_BD": "1988-11-21T17:00:00.000Z", "AD_EMAIL": "somsri.may@multiclothes.com", "AD_PHONE": "0869144407" }, { "AD_NUM": "000020031", "AD_FNAME": "Suaysood", "AD_LNAME": "Naisoy", "AD_ADDRESS": "Khlong Luang 25 KhlongNung KhlongLuang, Chang Wat Pathum Thani 12120", "AD_BD": "2001-05-02T17:00:00.000Z", "AD_EMAIL": "Suaysood.nai@multiclothes.com",

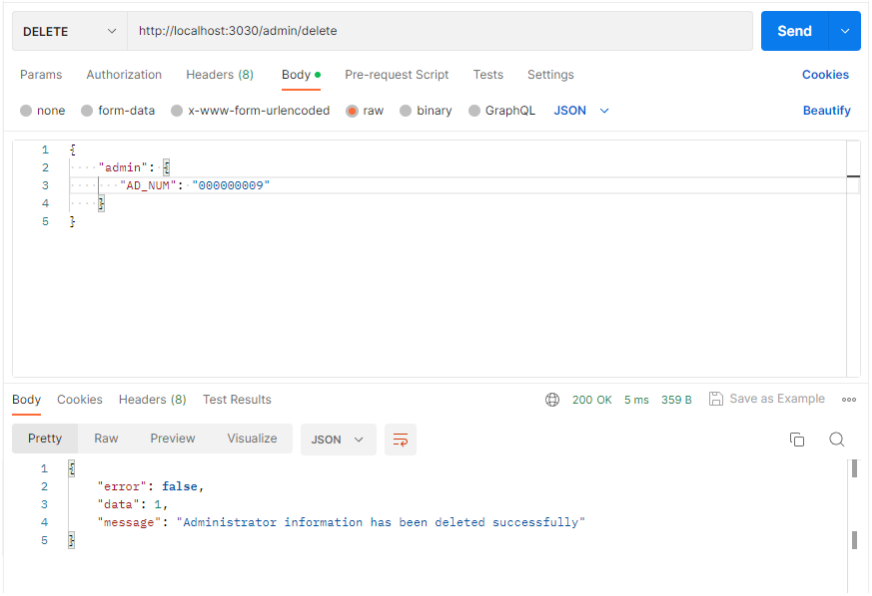
Insert	 POST
	http://localhost:3030/admin/add
	Send
	Params Authorization Headers (8) Body Pre-request Script Tests Settings
	Cookies
	none form-data x-www-form-urlencoded raw binary GraphQL JSON
	Beautify
	{
	"admin": {
	"AD_NUM": "000000002",
	"AD_FNAME": "Sun",
	"AD_LNAME": "Fayfay",
	"AD_ADDRESS": "999 Phutthamonthon Sai 4 Rd, Salaya, Phutthamonthon District, Nakhon Pathom 73170",
	"AD_BD": "2003-04-18",
	"AD_EMAIL": "sun.fay@multiclothes.com",
	"AD_PHONE": "0822356489"
	}
	}
	Body Cookies Headers (8) Test Results
	200 OK 11 ms 363 B Save as Example
	Pretty Raw Preview Visualize JSON
	{
	"error": false,
	"data": 1,
	"message": "New administrator information has been created successfully"
	}

	 POST
	http://localhost:3030/admin/add
	Send
	Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies
	none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
	{
	"admin": {
	"AD_NUM": "000000009",
	"AD_FNAME": "God",
	"AD_LNAME": "Fairly",
	"AD_ADDRESS": "999 Phutthamonthon Sai 4 Rd, Salaya, Phutthamonthon District, Nakhon Pathom 73170",
	"AD_BD": "2002-02-22",
	"AD_EMAIL": "god.fairly@multiclothes.com",
	"AD_PHONE": "0811121212"
	}
	}
	Body Cookies Headers (8) Test Results 200 OK 10 ms 363 B Save as Example
	Pretty Raw Preview Visualize JSON
	{
	"error": false,
	"data": 1,
	"message": "New administrator information has been created successfully"
	}

Update	 PUT
	http://localhost:3030/admin/update
	Send
	Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies
	none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
	{
	"admin": {
	"AD_NUM": "000000002",
	"AD_FNAME": "Sun",
	"AD_LNAME": "Fayfayfayfayfayfay"
	}
	}
	Body Cookies Headers (8) Test Results 200 OK 10 ms 359 B Save as Example
	Pretty Raw Preview Visualize JSON
	{
	"error": false,
	"data": 1,
	"message": "Administrator information has been updated successfully"
	}

	 PUT
	http://localhost:3030/admin/update
	Send
	Params Authorization Headers (8) Body Pre-request Script Tests Settings
	Cookies
	none form-data x-www-form-urlencoded raw binary GraphQL JSON
	Beautify
	{
	"admin": {
	"AD_NUM": "000000009",
	"AD_FNAME": "God",
	"AD_LNAME": "Fairytale",
	"AD_EMAIL": "god.fairy@multiclothes.com",
	"AD_PHONE": "0812121212"
	}
	}
	Body Cookies Headers (8) Test Results
	200 OK 13 ms 359 B
	Save as Example
	Pretty Raw Preview Visualize JSON
	{
	"error": false,
	"data": 1,
	"message": "Administrator information has been updated successfully"
	}

Delete	 DELETE
	http://localhost:3030/admin/delete
	Send
	Params Authorization Headers (8) Body Pre-request Script Tests Settings
	Cookies
	none form-data x-www-form-urlencoded raw binary GraphQL JSON
	Beautify
	{
	"admin": {
	"AD_NUM": "000000002"
	}
	}
	Body Cookies Headers (8) Test Results
	200 OK 9 ms 359 B
	Save as Example
	Pretty Raw Preview Visualize JSON
	{
	"error": false,
	"data": 1,
	"message": "Administrator information has been deleted successfully"
	}


DELETE
http://localhost:3030/admin/delete
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"admin": {
"AD_NUM": "000000009"
}
}
Body
Cookies
Headers (8)
Test Results
200 OK
5 ms
359 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"error": false,
"data": 1,
"message": "Administrator information has been deleted successfully"
}