Mahidol University
Faculty of Information
and Communication Technology

# Project Phase I and II report

By


| | | |
|---|---|---|
| Mr. Thitiwut | Harnphatcharapanukorn | ID 6488025 |
| Mr. Bhubodin | Somwhang | ID 6488085 |
| Mr. Sirasit | Puangpathanachai | ID 6488133 |
| Mr. Thanawat | Jarusuthirug | ID 6488178 |
| Mr. Kritchanapat | Junju | ID 6488206 |


A Report Submitted in Partial Fulfillment of the Requirements for

ITCS212 Web Programming

Faculty of Information and Communication Technology

Mahidol University

Semester 2, 2022

Table of contents

## Project Overview

Business domain: Food/drink (Bubble Tea)

Video Presentation:
https://drive.google.com/file/d/1Tht6aPDGqqwFvED54q45s9SyLCilPmeT/view?usp=sharing

The aim of this project is to develop a user-friendly bubble tea website for our bubble tea cafe “Chanom”. The website will provide a platform for customers to explore various recipes and flavors.

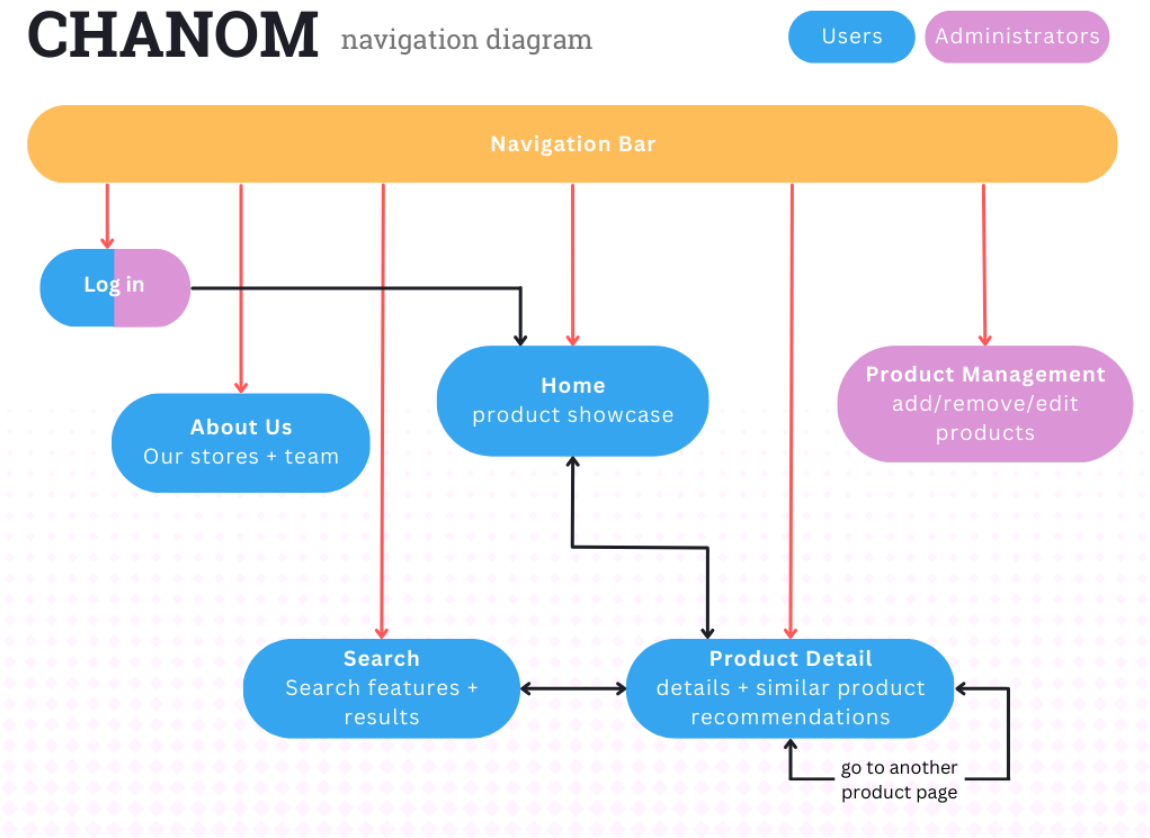
During Phase I, we developed a draft version of our website including the user interface in HTML and CSS. During this phase, all of the product pages are implemented by hand. No web services are included in displaying the product results. In this phase, the website is designed to suit two types of users which are normal users and administrators. In the styling process, we make the use of Bootstrap framework [1] to help us align the content of the page and use cool elements like cards and carousels as well as icons.

In Phase II, we shifted our to run on a web server. The web is now implemented using NodeJS with Express framework. A Chanom database is implemented to store the data of our website. Web services are implemented to handle requests regarding the database. The work from Phase I can now call web services from Phase II and our website is now fully connected to the database. The pages are now dynamic and can be changed according to the data on the database. The front-end (UI) and back-end (database) are separated into different servers. The front-end part runs on port 3030 while the back-end part runs on port 3000. To overcome cross-origin resource sharing problems, we used Axios [2] to help us call RESTful web services.

In addition to our implemented web services, we also added an external public web service called “fitness calculator API” on the about us page. This is a useful API to help calculate the ideal weight based on height and gender.

## Navigation diagram

- This is our navigation diagram from Phase I



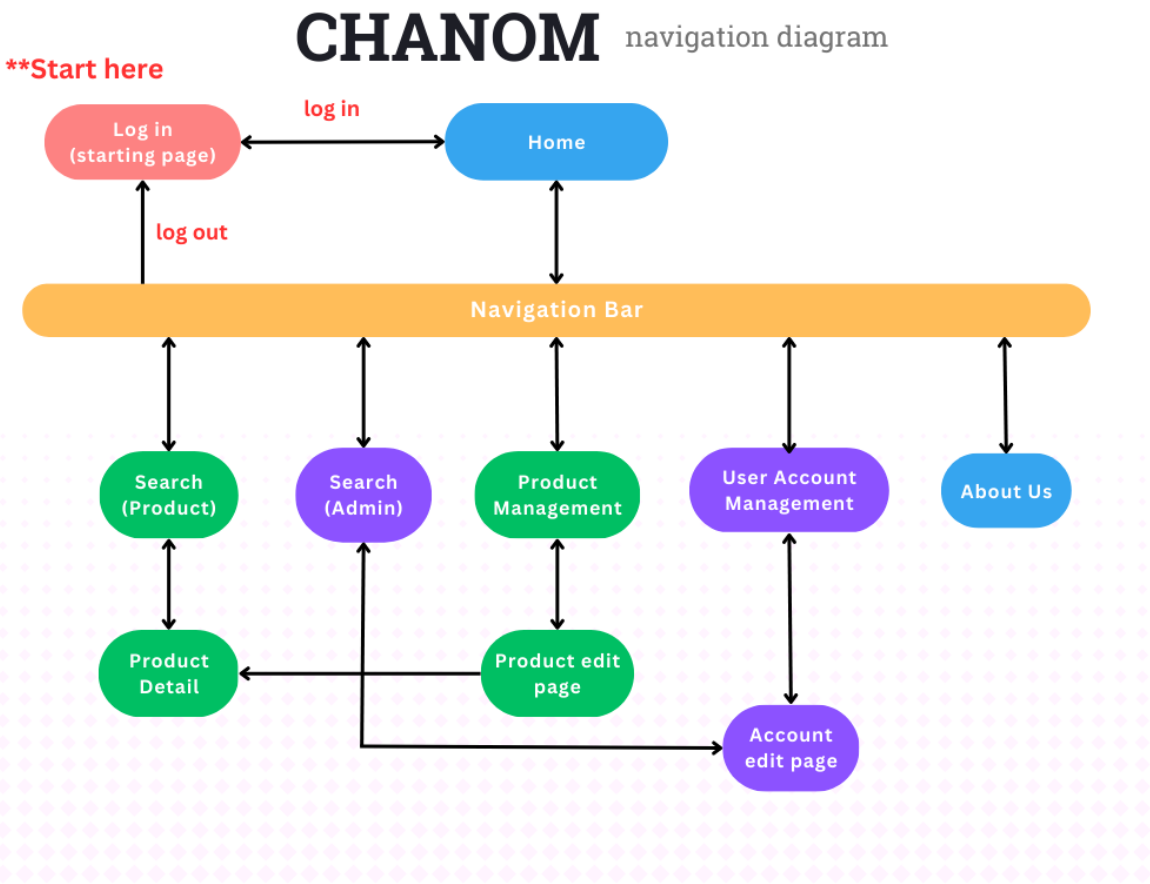

During this Phase, we did not include a user account management page. Each page is accessible both by going through the flow of the website, e.g. search page then product detail, and by using the navigation diagram.

The flow of the web is as follows

- Home is the starting page
- Every page is accessible by the navigation bar
- After the user clicks login on the login page, it will redirect to the home page
- The home page contains 5 recommendation products which can be redirected to the product details page
- Once the user clicks search on the search page, it will show the product results which can redirect to the product details page

- Each product detail page contains 3 recommendations which can be redirected to the product details page of that recommended product.

Now, let's move on to Phase II




During this phase, the website is completely for administrators. A user account management page is added along with an admin search page for searching admin data. An edit page where administrators can edit the data of each product and admin account is also added.

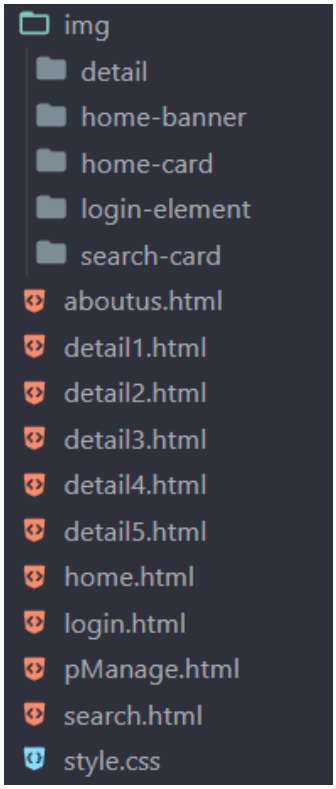
The flow of the web application is now changed to be the following:

- Login is the starting page. This page does not contain the navigation bar. Admins need to be authenticated before getting further into the website. After login, it will redirect to the home page.
- The home page and other pages have access to the navigation bar.

- The home page contains 5 recommendation products which can be redirected to the product details page
- Once the user clicks search on the search page, it will show the product results which can redirect to the product details page
- For the simplicity of the website and the development process, each product detail page no longer contains product recommendations.
- On the search admin page, the search results will appear as a table that can redirect the admin to the edit page which can be used to edit the admin information of that account.
- Both the user account management and product management page can redirect to the edit page of a particular product or account. However, the product management page can also redirect to the product details page of a particular product.

## Web Application and Code

### Phase I (Recap)




Here are the elements of our project in Phase I. All of the pages are styled using internal CSS, external CSS (style.css), and Inline CSS.

```
<style>
  .carousel-item img {
    height: 400px;
    object-fit: cover;
    border-radius: 30px;
  }

  div a {
    margin-right: 20px;
  }

  img.img-fluid {
    height: 320px;
    object-fit: cover;
  }
</style>
```

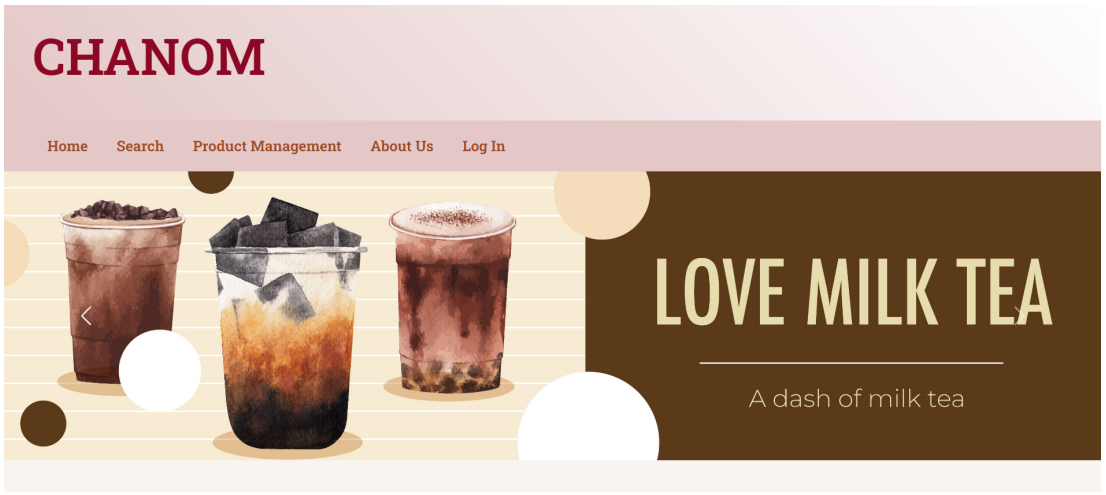
```
<p style="font-size: 22px;">Bubble Milk Tea</p>
<p style="font-size: 22px;">฿ 60</p>
<p style="font-size: 18px; text-align: justify;">"
  bubble tea!
  Made with premium tea leaves and topped with che
  this drink is the perfect treat to satisfy your
  such as classic milk tea or fruity passionfruit,
  including fresh fruit and jelly. With each sip,
  flavor and a delightful texture that's sure to l
</p>
```

To help us manage the content of the page, we also use the Bootstrap framework [2]. Another advantage of using Bootstrap is that we can have cool elements like cards and carousels to help us visualize our product better.

```
<div class="col-5">
  <div id="carouselExampleAutoplaying" class="carousel slide" data-bs-ride="carousel">
    <div class="carousel-inner">
      <div class="carousel-item active">
        <img
          src="img/detail/carousel-bubbleTea1.jpg"
          class="d-block w-100">
      </div>
      <div class="carousel-item">
        <img
          src="img/detail/carousel-bubbleTea2.jpg"
          class="d-block w-100">
      </div>
      <div class="carousel-item">
        <img src="img/detail/carousel-bubbleTea3.jpg" class="d-block w-100">
      </div>
    </div>
    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleAutoplaying"
      data-bs-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Previous</span>
    </button>
    <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleAutoplaying"
      data-bs-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Next</span>
    </button>
  </div>
</div>
```

```html
<div class="card mb-3" style="max-width: 540px;">
  <div class="row g-0">
    <div class="col-md-6">
      <img
        src="img/detail/card-chocolate.jpg"
        class="img-fluid rounded-start">
    </div>
    <div class="col-md-6">
      <div class="card-body">
        <h5 class="card-title">Chocolate Bubble Milk Tea</h5>
        <p class="card-text">Indulge in the rich and creamy taste of our chocolate bubble milk
          tea! Made with premium tea leaves and blended ...</p>
        <a class="btn btn-secondary" href="detail2.html">More Detail</a>
      </div>
    </div>
  </div>
</div>
```

Example of the use of the auto-playing Carousel to display product pictures on the product page

Example of Cards to display our products

## Search Page

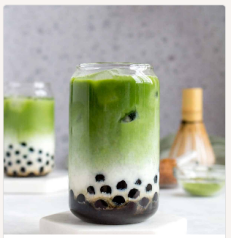
### Search Results

Classic Bubble Milk Tea

More Detail

Chocolate Bubble Milk Tea

More Detail



Matcha Bubble Green Tea

More Detail

Oolong Bubble Milk Tea

More Detail

Strawberry Yoghurt Frappe

More Detail

## Home Page

### CHANOM's Signatures

Classic Bubble Milk Tea

Our original classic Taiwanese Bubble Milk Tea

More Detail

Chocolate Bubble Milk Tea

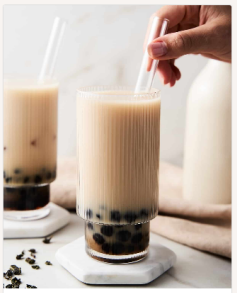
Our original classic Chocolate Bubble Milk Tea

More Detail

Matcha Bubble Green Tea

Our original classic Japanese Matcha Bubble Green Tea

More Detail



Oolong Bubble Milk Tea

Our original classic home made Oolong Bubble Milk Tea

More Detail

Strawberry Yoghurt Frappe

Our original classic Stawberry Yoghurt Frappe

More Detail

**Phase II**

Our web application (in the folder “sec1_gr6_src”) is run on port 3030. We implemented it using Express with help from Axios to do cross-origin resource sharing between client-side and web services.

Here is an example of how our web handles a request. This is how we load up the login page. (**All the code in this part can be found in “client-app.js”**)

```JavaScript
router.get("/login", (req, res) => {
  console.log("Request at /login");
  res.status(200).sendFile(path.join(__dirname, "/html/login.html"));
});
```

1. **Authentication**

To provide authentication, we also applied “session” to make sure that the user logs in before having access to other pages

```JavaScript
router.use(
  session({
    secret: "secret",
    resave: true,
    saveUninitialized: true,
  })
);
```

After successful authentication, the user will get a session that can be used to visit other pages

```javascript
if (code === 1) {
        console.log("Login successful!");
        req.session.user = req.body.username; // set session
        res.redirect("/home");
}
```

The other pages will check for a session before allowing a visit.

```javascript
router.get("/aboutus", (req, res) => {
  if (!req.session || !req.session.user) {  // check for session
    return res.redirect("/login");
  }
  console.log("Request at /aboutus");
res.status(200).sendFile(path.join(__dirname, "/html/aboutus.html"));
});
```

After the user logged out, remove the session

```javascript
router.get("/logout", (req, res) => {
  delete req.session.user; // delete the session user
  res.redirect("/login"); // get the user back to login page
});
```

## 2. Web services interaction

We make use of Axios to help us deal with cross-origin resource sharing (CORS) problems. Here is how we fetch product detail to the detail page.

```JavaScript
router.get("/detail/:id", (req, res) => {
  if (!req.session || !req.session.user) {
    return res.redirect("/login");
  } // users need to login first before going to this page
  const id = req.params.id;
  axios
    .get(`http://localhost:3000/selectchanom/${id}`, {responseType: "json"})
    .then((response) => {
      console.log(response.data);
      const data = response.data;
      .     // to be continued below
```

After that, we load up a detail page to modify with dom using fs (file system) and JSDOM

```JavaScript
fs.readFile(
        path.join(__dirname, "/html/detail.html"), "utf8", (err, html) => {
          if (err) {
            throw err;
          }
          const dom = new JSDOM(html);
          const output = dom.window.document.getElementById("output");   // now can use dom
            // to be continued
```

Now, we can use innerHTML to modify the element inside element id=”output”

JavaScript

```
          output.innerHTML = "";
          // add page content with the retrieved data
          output.innerHTML += `
                <div class="row">
                  <!-- Use Carousel to create preview images -->
                  <div class="col-5">
                    <div id="carouselExampleAutoplaying"
class="carousel slide" data-bs-ride="carousel">
                      <div class="carousel-inner">
                        <div class="carousel-item active">
                          <img
                            src="${data.pic1}"
                            class="d-block w-100">
                        </div>
                       and so on...
                `;
```

Finally, send the modified html file as a response

JavaScript

```
res.send(dom.serialize()); // sending the modified html file
        }
      );
    });
});
```

The same concept is used for the entire application, as well as for insert, update, delete and external public web service.

JavaScript

```
// insert an admin into the database
router.post("/admin-insert", (req, res) => {
  if (!req.session || !req.session.user) {
    return res.redirect("/login");
  }
  var data = req.body;
  console.log(data);
  axios.post("http://localhost:3000/insertadmin",
data).then((response) => {  // send the data along with the request
    res.redirect("/uManage");  // after insertion, go back to the
same page
  });
});
```

To summarize the concept of our web application, we use Router() from Express to handle each request (localhost:3030/xxxx). Then have Axios call the web services. After that, load an existing html page from our machine and modify it accordingly using JSDOM and send the result back to the user.

Our web application is also connected to an external public web service called “fitness calculator API”. This service is added to the about us page to help the user calculate the ideal weight based on height and gender. (So that they don’t drink too much bubble tea!)

More documentation on:
https://rapidapi.com/malaaddincelik/api/fitness-calculator

JavaScript

```
router.post("/getweight", (req, res) => {
  // console.log("button click", req.body.weight);

  if (req.body.gender === "") return;
  const options = {
    method: "GET",
    url: "https://fitness-calculator.p.rapidapi.com/idealweight",
    params: {
      gender: req.body.gender,
      height: parseInt(req.body.height),
    },
    headers: {
      "X-RapidAPI-Key":
"ec16420a96msh55f10f896eb4a5dp1ae895jsnabb435fbb8b6",
      "X-RapidAPI-Host": "fitness-calculator.p.rapidapi.com",
    },
  };

  axios.request(options).then((response) => {
    console.log(response.data);
    const data = response.data.data;
    console.log(data);
    fs.readFile(
      path.join(__dirname, "/html/aboutus.html"),
      "utf8",
      (err, html) => {
        if (err) {
          throw err;
        }
        const dom = new JSDOM(html);
        const output = dom.window.document.getElementById("weight_text");

        output.innerHTML = "";
        // add page content with the retrieved data
        output.innerHTML += `
                <div>
                  <p>Hamwi: ${data["Hamwi"]}
                  Devine: ${data["Devine"]}
                  Miller: ${data["Miller"]}
                  Robinson: ${data["Robinson"]}</p>
                </div>
                `;
        res.send(dom.serialize()); // sending the modified html file
      }
    );
  });
});
```

## Web Services and Code

The folder “sec1_gr6_ws_src” is our web services. The web services run on port 3000 and are linked to our Chanom database. In this project, we created a simple database with 2 tables which are product and administrator. Here is the relational schema.

Administrator

| aID | username | pass_word | fname | lname | birthdate | email |
|---|---|---|---|---|---|---|

Product

| pID | pName | pType | topping | rating | pDescription | pic1 | pic2 | pic3 | price |
|---|---|---|---|---|---|---|---|---|---|

**Administrator**

aID - administrator ID

username - administrator username

pass_word - administrator password

fname - administrator first name

lname - administrator last name

email - administrator email address

**Product**

pID - product ID

pName - product name

pType - product type

topping - product topping

rating - product rating from 1-5

pDescription - product description

pic1, pic2, pic3 - Image address of 3 product pictures

price - product price (Baht)

Unset

```
CREATE TABLE administrator (
    aID INT AUTO_INCREMENT,  // Administrator ID
    username VARCHAR(50),        // Username
    pass_word VARCHAR(50) NOT NULL,     // Password to log in to the web
    fname VARCHAR(50) NOT NULL,  // first name
    lname VARCHAR(50) NOT NULL,  // last name
    birthdate DATE NOT NULL,     // birthdate in the format of (YYYY-MM-DD)
    email VARCHAR(50) NOT NULL,  // email
    PRIMARY KEY (aID,username)// Composite primary key of aID and username
);
CREATE TABLE product (
    pID INT PRIMARY KEY AUTO_INCREMENT, // product ID
    pName VARCHAR(50) NOT NULL,  // product name
    pType VARCHAR(50) NOT NULL,  // product type
    topping VARCHAR(50),   // product topping
    rating INT NOT NULL,   // product rating
    pDescription TEXT NOT NULL,  // product description
    pic1 VARCHAR(200) NOT NULL,  //
    pic2 VARCHAR(200) NOT NULL,  // link to the picture of the product
    pic3 VARCHAR(200) NOT NULL,  //
    price INT NOT NULL,    // price
    CHECK (rating >= 0 AND rating <= 5)  // Rating is an integer between 1-5
);
```

Web services of product

1. Select by ID (GET)

This is an additional useful service for our website. It returns the product detail based on the provided product ID. E.g. when getting a request localhost:3000/selectchanom/1, the service will return the product detail of pID = 1.

```javascript
router.get('/selectchanom/:id', (req, res) => {

    let pID = req.params.id;

    if(!pID) {

        return res.status(400).send({error: true, message: 'Please provide product information'})

    }

    connection.query("SELECT * FROM product WHERE pID = ?", [pID], (error, results) => {

        if(error) throw error;

        res.json(results[0]);

        console.log(`Sending product result of pID = ${pID}`);

    })

})
```

2. Search (POST)

It accepts up to 4 criteria, which are pName (product name), pType (product type), topping, and rating. The main concept of this is that the service will check whether there is a value in each criteria sent by the request or not. If yes, it will modify the SQL statement and do the query to the database and return the response. Note that if there is no value in any criteria, it will return all product details (no-criteria search)

JavaScript

```
router.post('/searchchanom', (req, res) => {
    let pName = req.body.pName;
    let pType = req.body.pType;
    let topping = req.body.topping;
    let rating = req.body.rating;
    let queryParams = [];
    let queryString = 'SELECT * FROM product';  // initial SQL statement
    // check if each query attribute has a value or not
    if (pName) {
        queryParams.push(`pName LIKE '%${pName}%'`);
    }
    if (pType) {
        queryParams.push(`pType = '${pType}'`);
    }
    if (topping) {
        queryParams.push(`topping = '${topping}'`);
    }
    if (rating) {
        queryParams.push(`rating = ${rating}`);
    }
    if (queryParams.length > 0) {
        queryString += ` WHERE ${queryParams.join(' AND ')}`;
    }
    connection.query(queryString, (error, results) => {
        if (error) throw error;
        console.log(`${results.length} search result(s) found`);
        return res.json(results);
    });
});
```

3. Insert (POST)

The service accepts the product data from the request and then adds it into the database using SQL statement ‘INSERT INTO’

```javascript
router.post('/insertchanom', (req, res) => {

    let data = req.body;

    if(!data) {

        return res.status(400).send({error: true, message: 'Please provide product information'})

    }

    connection.query(`INSERT INTO product SET ? `, data, (error, results) => {

        if(error) throw error;

        console.log("Inserting a product");

        return res.send({error: false, data: results.affectedRows, message: 'New product has been added successfully'})

    })

})
```

4. Update (PUT)

The service will update the product information based on the provided product ID and the provided information from the request. E.g. if the request is localhost:3000/updatechanom/1, it will update the information product pID = 1 based on the provided information from the request.

```JavaScript
router.put('/updatechanom/:id', (req, res) => {

    let pID = req.params.id;

    let product = req.body;

    if(!pID || !product) {

        return res.status(400).send({error: true, message: 'Please provide product information'})

    }

    connection.query("UPDATE product SET ? WHERE pID = ?", [product, pID], (error, results) => {

        if(error) throw error;

        console.log(`Updating product pID = ${pID}`);

        return res.send({error: false, data: results.affectedRows, message: 'Product has been updated successfully'})

    })

})
```

5. Delete (DELETE)

The service will delete a product based on the provided pID. E.g. If the request is from localhost/deletechanom/1, it will delete the product with pID = 1

JavaScript

```
router.delete('/deletechanom/:id', (req, res) => {

    let pID = req.params.id;

    if(!pID) {

        return res.status(400).send({error: true, message: 'Please provide
product information'})

    }

    connection.query("DELETE FROM product WHERE pID = ?", [pID], (error,
results) => {

        if(error) throw error;

        console.log(`Deleting product pID = ${pID}`);

        if(results.affectedRows === 0) {

            return res.send({error: false, deleted: results.affectedRows,
message: 'Product to be deleted does not exist'})

        }

        return res.send({error: false, deleted: results.affectedRows,
message: 'Product has been deleted successfully'})

    })

})
```

Web services of administrators

1. Authentication (POST)

The service will check provided username and password with the database. If the provided info matches with the one on the database, it will send status 1, if not, send status 0.

```javascript
router.post('/authenticate', (req, res) => {
    const username = req.body.username;
    const pass_word = req.body.password;
    if (!username || !pass_word) {
        return res.status(400).send({ error: true, message: 'Please provide admin information' });
    }
    try {
        connection.query("SELECT username, pass_word FROM administrator", (error, results) => {
            if (error) {
                console.error(error);
                return res.status(500).json({ error: true, message: 'Internal server error' });
            }
            for (var i = 0; i < results.length; i++) {
                if (results[i].username === username && results[i].pass_word === pass_word) {
                    console.log("Authentication: match");
                    return res.json({ "status": "match", "code": 1 });
// send status pass(1) back if match
                }
            }
            console.log("Authentication: no match");
            return res.json({ "status": "no match", "code": 0 });
// send fail(0) if username and password does not match
        });
    } catch (err) {
        console.error(err);
        return res.status(500).json({ error: true, message: 'Internal server error' });
    }
});
```

2. Select by ID (GET)

This is an additional useful service for our website. It returns the administrator detail based on the provided admin ID. E.g. when getting a request localhost:3000/selectadmin/1, the service will return the administrator detail of aID = 1.

```javascript
router.get('/selectadmin/:id', (req, res) => {
    let aID = req.params.id;
    if(!aID) {
        return res.status(400).send({error: true, message: 'Please provide
admin information'})
    }
    connection.query("SELECT * FROM administrator WHERE aID = ?", [aID],
(error, results) => {
        if(error) throw error;
        res.json(results[0]);
        console.log("Sending admin result");
    })
})
```

3. Search (POST)

It accepts up to 3 criteria, which are username, fname (first name), and lname (last name). The main concept of this is that the service will check whether there is a value in each criteria sent by the request or not. If yes, it will modify the SQL statement and do the query to the database and return the response. Note that if there is no value in any criteria, it will return all administrator details (no-criteria search)

JavaScript

```
router.post('/searchadmin', (req, res) => {
    let username = req.body.username;
    let fname = req.body.fname;
    let lname = req.body.lname;

    let queryParams = [];
    let queryString = 'SELECT * FROM administrator';

    // check if each query attribute has a value or not
    if (username) {
        queryParams.push(`username LIKE '%${username}%'`);
// add more to the query string if any
    }

    if (fname) {
        queryParams.push(`fname LIKE '%${fname}%'`);
    }

    if (lname) {
        queryParams.push(`lname LIKE '%${lname}%'`);
    }

    if (queryParams.length > 0) {
        queryString += ` WHERE ${queryParams.join(' AND ')}`;
    }

    connection.query(queryString, (error, results) => {
        if (error) throw error;
        console.log(`${results.length} search result(s) found`);
        return res.json(results);
    });
});
```

4. Insert (POST)

The service accepts the administrator data from the request and then adds it into the database using SQL statement ‘INSERT INTO’

JavaScript

```
router.post('/insertadmin', (req, res) => {
    let data = req.body;

    if(!data) {
        return res.status(400).send({error: true, message: 'Please
provide product information'})
    }

    connection.query(`INSERT INTO administrator SET ? `, data,
(error, results) => {
        if(error) throw error;
        return res.send({error: false, data: results.affectedRows,
message: 'New admin has been added successfully'})
    })

})
```

5. Update (PUT)

The service will update the administrator information based on the provided admin ID and the provided information from the request. E.g. if the request is localhost:3000/updateadmin/1, it will update the information administrator aID = 1 based on the provided information from the request.

```JavaScript
router.put('/updateadmin/:id', (req, res) => {
    let aID = req.params.id;
    let admin = req.body;

    if(!aID || !admin) {
        return res.status(400).send({error: true, message: 'Please
provide admin information'})
    }
    connection.query("UPDATE administrator SET ? WHERE aID = ?",
[admin, aID], (error, results) => {
        if(error) throw error;
        console.log(`Updating admin aID = ${aID}`);
        return res.send({error: false, data: results.affectedRows,
message: 'Admin has been updated successfully'})
    })
})
```

6. Delete (DELETE)

The service will delete an administrator based on the provided aID. E.g. If the request is from localhost/deleteadmin/1, it will delete the administrator with aID = 1

```JavaScript
router.delete('/deleteadmin/:id', (req, res) => {
    let aID = req.params.id;
    if(!aID) {
        return res.status(400).send({error: true, message: 'Please
provide product information'})
    }
    connection.query("DELETE FROM administrator WHERE aID = ?",
[aID], (error, results) => {
        if(error) throw error;
        console.log(`Deleting admin aID = ${aID}`);
        if(results.affectedRows === 0) {
            return res.send({error: false, deleted:
results.affectedRows, message: 'Admin to be deleted does not exist'})
        }
        return res.send({error: false, deleted: results.affectedRows,
message: 'Admin has been deleted successfully'})
    })
})
```

## Testing result

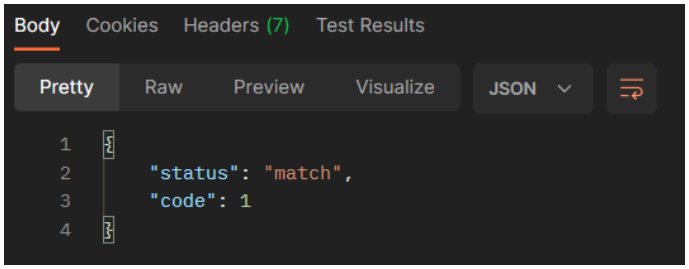
Here are the test cases and results of the web services via Postman

1. Authentication service

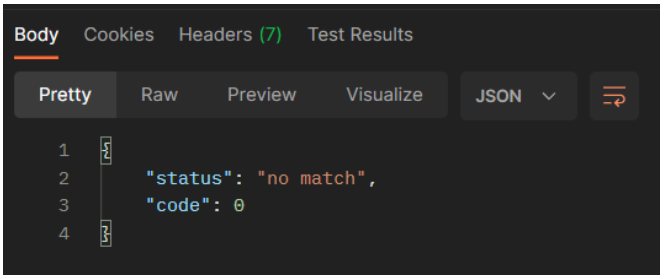
```
Unset
method: post

URL: http://localhost:3000/authenticate

body: raw JSON

Test case #1 (correct username, password)

{

    "username": "ict",

    "password": "ict555"

}

Test case #2 (incorrect username, password)

{

    "username": "acb",

    "password": "superBoss555"

}
```

Result:

Test case #1




Test case #2



## 2. Search product service

```
Unset
method: post

URL: http://localhost:3000/searchchanom

body: raw JSON

Test case #0 (no criteria search)

{

    "pName": "",

    "pType": "",

    "topping": "",

    "rating": ""

}

Test case #1 (search by name)

{

    "pName": "Classic Bubble",

    "pType": "",

    "topping": "",

    "rating": ""

}

Test case #2 (search by type)

{

    "pName": "",

    "pType": "Bubble Milk Tea",

    "topping": "",

    "rating": ""

}
```

Unset

```
Test case #3 (search by topping)
{
    "pName": "",
    "pType": "",
    "topping": "Black Pearl",
    "rating": ""
}
Test case #4 (search by rating)
{
    "pName": "",
    "pType": "",
    "topping": "",
    "rating": "5"
}
Test case #5 (combination of categories)
{
    "pName": "",
    "pType": "Bubble Milk Tea",
    "topping": "Brown Sugar",
    "rating": "5"
}
Test case #6 (combination of categories)
{
    "pName": "Taro",
    "pType": "Bubble Milk Tea",
    "topping": "",
    "rating": ""
}
```

Result of some of the cases (#0, #1, #6)

Test case #0 (return every product information)


```
[
    {
        "pID": 1,
        "pName": "Classic Bubble Milk Tea",
        "pType": "Bubble Milk Tea",
        "topping": "Brown Sugar",
        "rating": 5,
        "pDescription": "Indulge in the sweet and refreshing taste of our signature bubble tea! Made with premium tea leaves and topped with
            chewy tapioca pearls, this drink is the perfect treat to satisfy your sweet tooth. Choose from a variety of delicious flavors,
            such as classic milk tea or fruity passionfruit, and customize your drink with your choice of toppings, including fresh fruit
            and jelly. With each sip, you'll experience a burst of flavor and a delightful texture that's sure to leave you wanting more!",
        "pic1": "https://drive.google.com/uc?export=view&id=108iRXvoO25WyWhpmzpqOFhj7iMx8tOlv",
        "pic2": "https://drive.google.com/uc?export=view&id=1RyXXeWrmu6W2ZfF7ZLzpMIU5wEPypsz5",
        "pic3": "https://drive.google.com/uc?export=view&id=1Sx5t_ldu7ELiiLpOWrG9-4UFgcs-bcKj",
        "price": 60
    },
    {...
    },
    {...
    },
    {...
    },
    {
        "pID": 7,
        "pName": "Taro Bubble Milk Tea",
        "pType": "Bubble Milk Tea",
        "topping": "Black Pearl",
        "rating": 4,
        "pDescription": "Discover a new level of indulgence with our creamy and delicious taro bubble tea. Made with real taro root and
            blended to perfection, our bubble tea boasts a rich, velvety texture that is sure to please. Savor the subtle hints of vanilla
            and nuttiness with every sip, complemented by the satisfying chewiness of our tapioca pearls. Whether you're in need of a quick
            pick-me-up or simply craving a sweet treat, our taro bubble tea is the perfect choice.",
        "pic1": "https://drive.google.com/uc?export=view&id=1RY9Y5mHr2npnrsT-I0mquCzrWCoY79k0",
        "pic2": "https://drive.google.com/uc?export=view&id=1zhSWnfPiWld0Pz_hkv9YtFhNujciJ3O9",
        "pic3": "https://drive.google.com/uc?export=view&id=1P4-j2U45msCKT6iTXvTtdp2G6LX7hH9o",
        "price": 60
    }
]
```


Test case #1


```
[
    {
        "pID": 1,
        "pName": "Classic Bubble Milk Tea",
        "pType": "Bubble Milk Tea",
        "topping": "Brown Sugar",
        "rating": 5,
        "pDescription": "Indulge in the sweet and refreshing taste of our signature bubble tea! Made with premium tea leaves and topped with
            chewy tapioca pearls, this drink is the perfect treat to satisfy your sweet tooth. Choose from a variety of delicious flavors,
            such as classic milk tea or fruity passionfruit, and customize your drink with your choice of toppings, including fresh fruit
            and jelly. With each sip, you'll experience a burst of flavor and a delightful texture that's sure to leave you wanting more!",
        "pic1": "https://drive.google.com/uc?export=view&id=108iRXvoO25WyWhpmzpqOFhj7iMx8tOlv",
        "pic2": "https://drive.google.com/uc?export=view&id=1RyXXeWrmu6W2ZfF7ZLzpMIU5wEPypsz5",
        "pic3": "https://drive.google.com/uc?export=view&id=1Sx5t_ldu7ELiiLpOWrG9-4UFgcs-bcKj",
        "price": 60
    }
]
```

Test case #6


```
[
    {
        "pID": 7,
        "pName": "Taro Bubble Milk Tea",
        "pType": "Bubble Milk Tea",
        "topping": "Black Pearl",
        "rating": 4,
        "pDescription": "Discover a new level of indulgence with our creamy and delicious taro bubble tea. Made with real taro root and
            blended to perfection, our bubble tea boasts a rich, velvety texture that is sure to please. Savor the subtle hints of vanilla
            and nuttiness with every sip, complemented by the satisfying chewiness of our tapioca pearls. Whether you're in need of a quick
            pick-me-up or simply craving a sweet treat, our taro bubble tea is the perfect choice.",
        "pic1": "https://drive.google.com/uc?export=view&id=1RY9Y5mHr2npnrsT-I0mquCzrWCoY79k0",
        "pic2": "https://drive.google.com/uc?export=view&id=1zhSWnfPiWld0Pz_hkv9YtFhNujciJ3O9",
        "pic3": "https://drive.google.com/uc?export=view&id=1P4-j2U45msCKT6iTXvTtdp2G6LX7hH9o",
        "price": 60
    }
]
```


## 3. Insert products

Unset

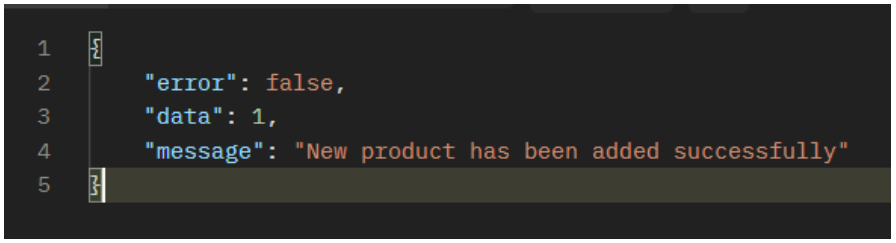
method: post
URL: http://localhost:3000/insertchanom
body: raw JSON
Test case #1

```
{
    "pName": "Oreo Milk Tea",
    "pType": "Bubble Milk Tea",
    "topping": "Black Pearl",
    "rating": 5,
    "pDescription": "The best oreo drink you will ever get to
drink!!",
    "pic1" :
"https://www.sunglowkitchen.com/wp-content/uploads/2022/12/cookies-an
d-cream-oreo-boba-tea-9.jpg",
    "pic2" :
"https://thelittlestcrumb.com/wp-content/uploads/oreo-milk-tea-featur
ed-image-1.jpg",
    "pic3" :
"https://images.squarespace-cdn.com/content/v1/5e8840afd65f745da4030c
a8/1616648671297-LW0OFQOWLIRCYIYPWWLO/National-Oreo-Day-Social-Media-
Posts---March-6-1.jpg",
    "price": 65
}
```

Unset

```
Test case #2
{
    "pName": "Jasmine Bubble Tea",
    "pType": "Bubble Milk Tea",
    "topping": "Black Pearl",
    "rating": 4,
    "pDescription": "The best Jasmine tea in town. You will never
find one better!!",
    "pic1" :
"https://cdn.shopify.com/s/files/1/0645/1401/articles/20220404225443-
untitled-1-932321_1280x.jpg?v=1660336156",
    "pic2" :
"https://yourcoffeeandtea.com/wp-content/uploads/2021/05/Jasmine-Milk
-Tea.jpg",
    "pic3" :
"https://theclassybaker.com/wp-content/uploads/2022/05/jasmine-milk-t
ea-11.jpg",
    "price": 60
}
```

## Result for both test cases



## 4. Update products

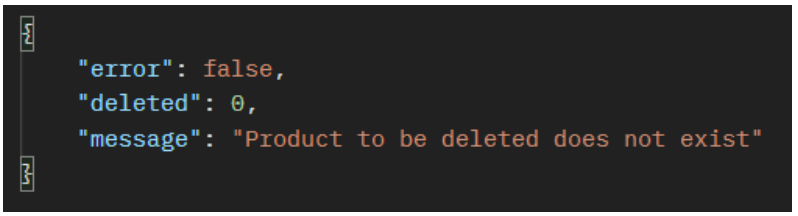
Unset

```
Update a product based on pID
method: put
Test case #1
URL: http://localhost:3000/updatechanom/1
body: raw JSON
{
    "pName": "Classic Bubble Milk Tea (Special Offer!)",
    "pType": "Bubble Milk Tea",
    "topping": "Grass Jelly",
    "rating": 5,
    "pDescription": "The best Bubble tea in town. You will never find one
better!!",
    "pic1" :
"https://drive.google.com/uc?export=view&id=108iRXvoO25WyWhpmzpqOFhj7iMx8tOl
v",
    "pic2" :
"https://drive.google.com/uc?export=view&id=1RyXXeWrmu6W2ZfF7ZLzpMIU5wEPypsz
5",
    "pic3" :
"https://drive.google.com/uc?export=view&id=1Sx5t_ldu7ELiiLpOWrG9-4UFgcs-bcK
j",
    "price": 50
}

Test case #2
URL: http://localhost:3000/updatechanom/2
body: raw JSON
{
    "pName": "Chocolate Milk Brown Sugar",
    "pType": "Bubble Milk",
    "topping": "Brown Sugar",
    "rating": 5,
    "pDescription" : "The best chocolate drink in town. You will never find
one better!!",
    "pic1" :
"https://drive.google.com/uc?export=view&id=1DgjSclqKldXjVF1n0MpeksyiC6e_a0k
3",
    "pic2" :
"https://drive.google.com/uc?export=view&id=1vEG_yn6NccFrDGmqZ-9WNbPs4r75pjA
g",
    "pic3" :
"https://drive.google.com/uc?export=view&id=1Jnh8uMgCIHRC4gTarR1rgu17bnj4IyF
I",
    "price": 65
}
```

Results for both cases


## 5. Delete products

```
Unset
Delete a product based on pID
method: delete
Test case #1 (Delete an existing product)
URL: http://localhost:3000/deletechanom/1
Test case #2 (Delete an unexisting product)
URL: http://localhost:3000/deletechanom/10
```

## 6. Search administrators

```
Unset
method: post
URL: http://localhost:3000/searchadmin
body: raw JSON
Test case #0 (no criteria search)
{
    "username": "",
    "fname": "",
    "lname": ""
}
Test case #1 (search by username)
{
    "username": "ict",
    "fname": "",
    "lname": ""
}
Test case #2 (search by fname)
{
    "username": "",
    "fname": "anna",
    "lname": ""
}
Test case #3 (search by lname)
{
    "username": "",
    "fname": "",
    "lname": "harn"
}
Test case #4 (Combination of criterias)
{
    "username": "",
    "fname": "anna",
    "lname": "robert"
}
Test case #5 (Combination of criterias)
{
    "username": "i",
    "fname": "ict",
    "lname": "use"
}
```

Test resultsResult of some of the cases (#0, #2, #4)

Test case #0 (no criteria search)

Return all admin information


```
[
    {
        "aID": 1,
        "username": "Kritchanapat",
        "pass_word": "Earthza007",
        "fname": "Kritchanapat",
        "lname": "Junju",
        "birthdate": "2003-05-25T17:00:00.000Z",
        "email": "kritchanapat.jun@student.mahidol.edu"
    },
    {…
    }…
    },
    {
        "aID": 8,
        "username": "AnaWooo",
        "pass_word": "DDAisReal55",
        "fname": "Anna",
        "lname": "Dupont",
        "birthdate": "1995-06-07T17:00:00.000Z",
        "email": "anna.dup@hotmail.com"
    }
```


Test case #2 (search by fname)


```
[
    {
        "aID": 7,
        "username": "Anababa",
        "pass_word": "abc1234",
        "fname": "Anna",
        "lname": "Robertson",
        "birthdate": "1998-04-05T17:00:00.000Z",
        "email": "anna.rob@hotmail.com"
    },
    {
        "aID": 8,
        "username": "AnaWooo",
        "pass_word": "DDAisReal55",
        "fname": "Anna",
        "lname": "Dupont",
        "birthdate": "1995-06-07T17:00:00.000Z",
        "email": "anna.dup@hotmail.com"
    }
]
```

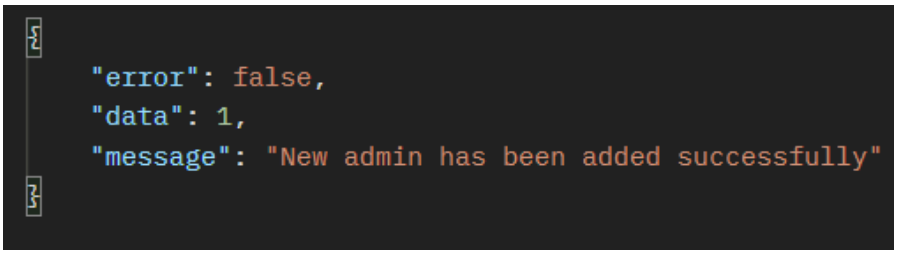

Test case #4 (fname and lname)


```
[
    {
        "aID": 7,
        "username": "Anababa",
        "pass_word": "abc1234",
        "fname": "Anna",
        "lname": "Robertson",
        "birthdate": "1998-04-05T17:00:00.000Z",
        "email": "anna.rob@hotmail.com"
    }
]
```

7. Insert administrators

```
Unset
method: post
URL: http://localhost:3000/insertadmin
body: raw JSON
Test case #1
{
    "username": "ictTest1",
    "pass_word": "ict123",
    "fname": "Lionel",
    "lname": "Messinho",
    "birthdate": "1987-06-24",
    "email": "testEmail@mail.com"
}
Test case #2
{
    "username": "iLoveCoding123",
    "pass_word": "cryInSide555",
    "fname": "Robert",
    "lname": "Dupont",
    "birthdate": "2001-06-04",
    "email": "robert@myMail.com"
}
```

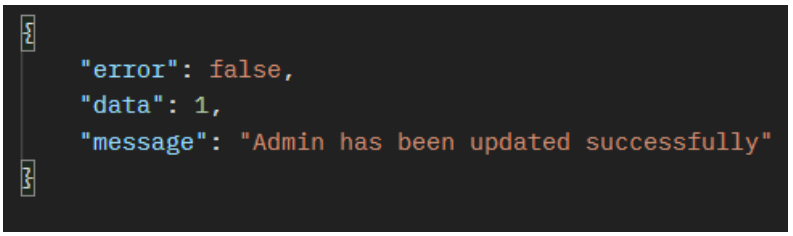
Result for both test cases



8. Update administrators

```
Unset
Update an admin based on aID
method: put
Test case #1
URL: http://localhost:3000/updateadmin/1
body: raw JSON
{
    "username": "KcnpEarth",
    "pass_word": "test5564",
    "fname": "Kritchanapat",
    "lname": "Junju",
    "birthdate": "2001-06-10",
    "email": "Kritchanapat@mail.com"
}
Test case #2
URL: http://localhost:3000/updateadmin/2
body: raw JSON
{
    "username": "Thitiwut",
    "pass_word": "boss5074",
    "fname": "Thitiwut",
    "lname": "Harn",
    "birthdate": "2001-06-04",
    "email": "thitiwutmcpeakz@gmail.com"
}
```

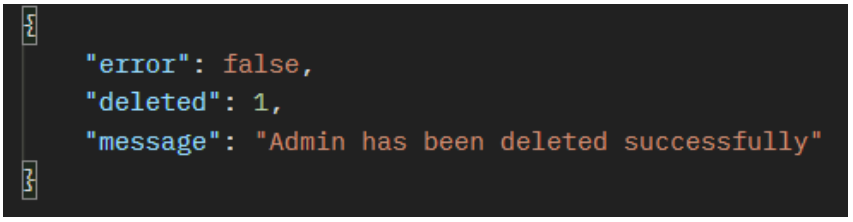
Test result for both cases



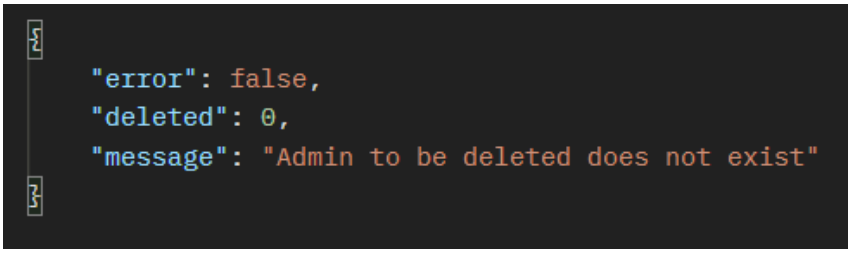
9. Delete administrators

```
Unset
Delete an admin based on aID
method: delete
Test case #1 (Delete an existing account)
URL: http://localhost:3000/deleteadmin/1
Test case #2 (Delete an unexisting account)
URL: http://localhost:3000/deleteadmin/10
```

Test case #1




Test case #2



## References

[1] https://getbootstrap.com/docs/5.3/getting-started/introduction/

[2] https://axios-http.com/docs/intro