
ITCS212 Web Programming

Project Phase 3

Report

By


| | | |
|---|---|---|
| 6388063 | Nuttapat | Sumransilp |
| 6388113 | Poomrapee | Wareboutr |
| 6388133 | Pitchaya | Teerawongpairoj |
| 6388196 | Sasima | Srijanya |

Section 3 Group 9


Present

Dr. Wudhichart Sawangphol

Dr. Jidapa Kraisangka

Semester 2 Academic year 2021

## Table Of Contents

# Overview of Project

Our web programming project has been based on a retailer business module. First, we started working with our web diagram, then we moved on to start initializing each page until it looks functional. In the later stage of our project, we implemented the web service to handle different kinds of data operations methods on our web and database. In the final stage, we improve our web searching algorithm by implementing the react to our website.

**Overview For Phase I**

In phase I, we focus on designing and decorating the interface for the user side by using HTML language and CSS to style each webpage.

**Overview For Phase II**

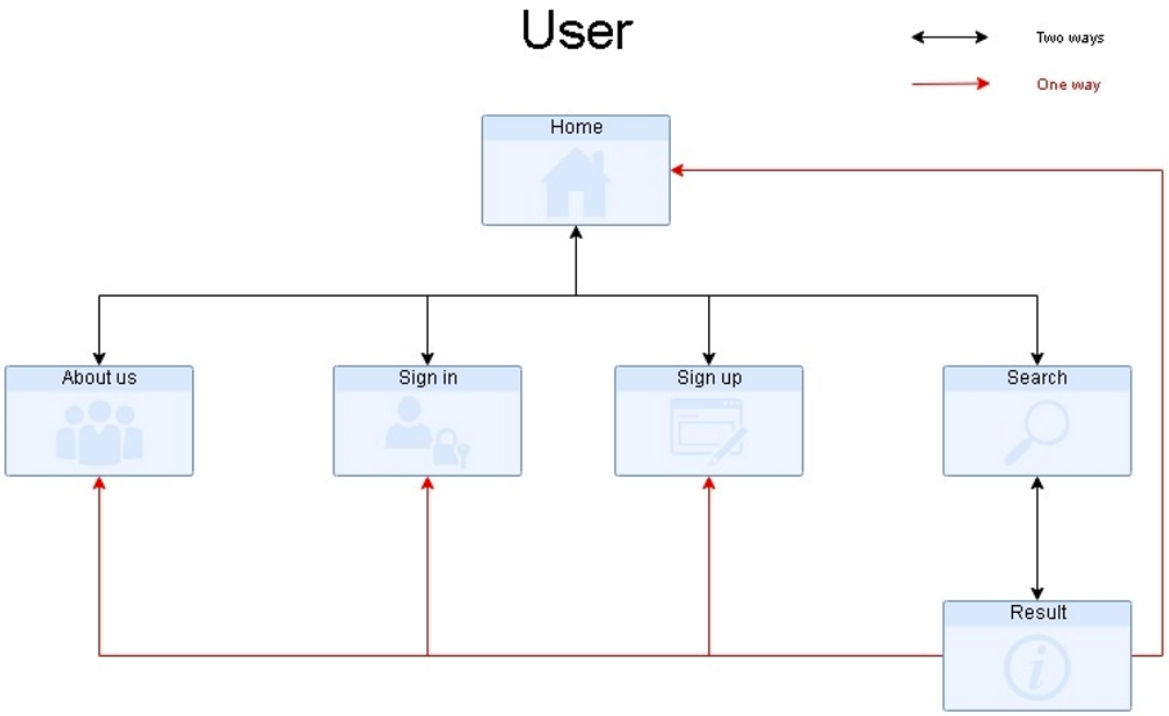
In phase II, we focus more on the web service to make the interaction between the user and the website. By implementing the search function to let the user input the data and then it will show the result.

**Overview For Phase III**

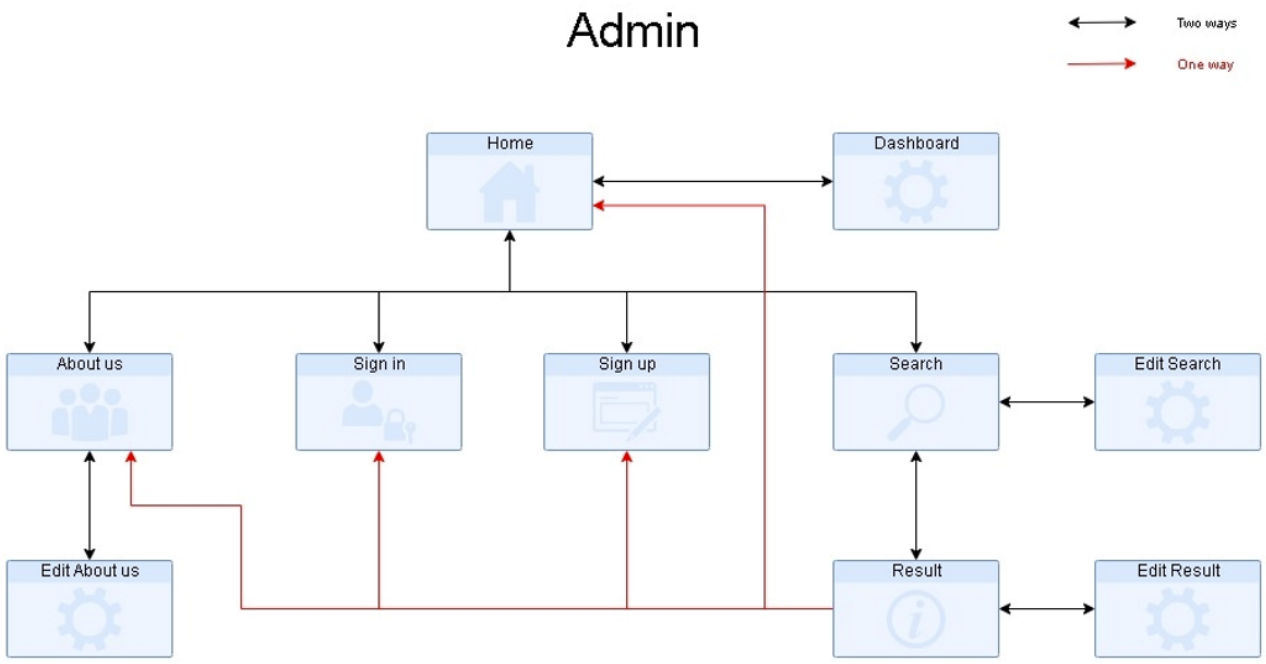
In phase III, we have a major goal to implement the web service from phase 2 to develop the handler page for administrators. This phase establishes all the project tasks using REACT and some other libraries and packages to achieve this final task.

## Phase I

### Navigation diagram



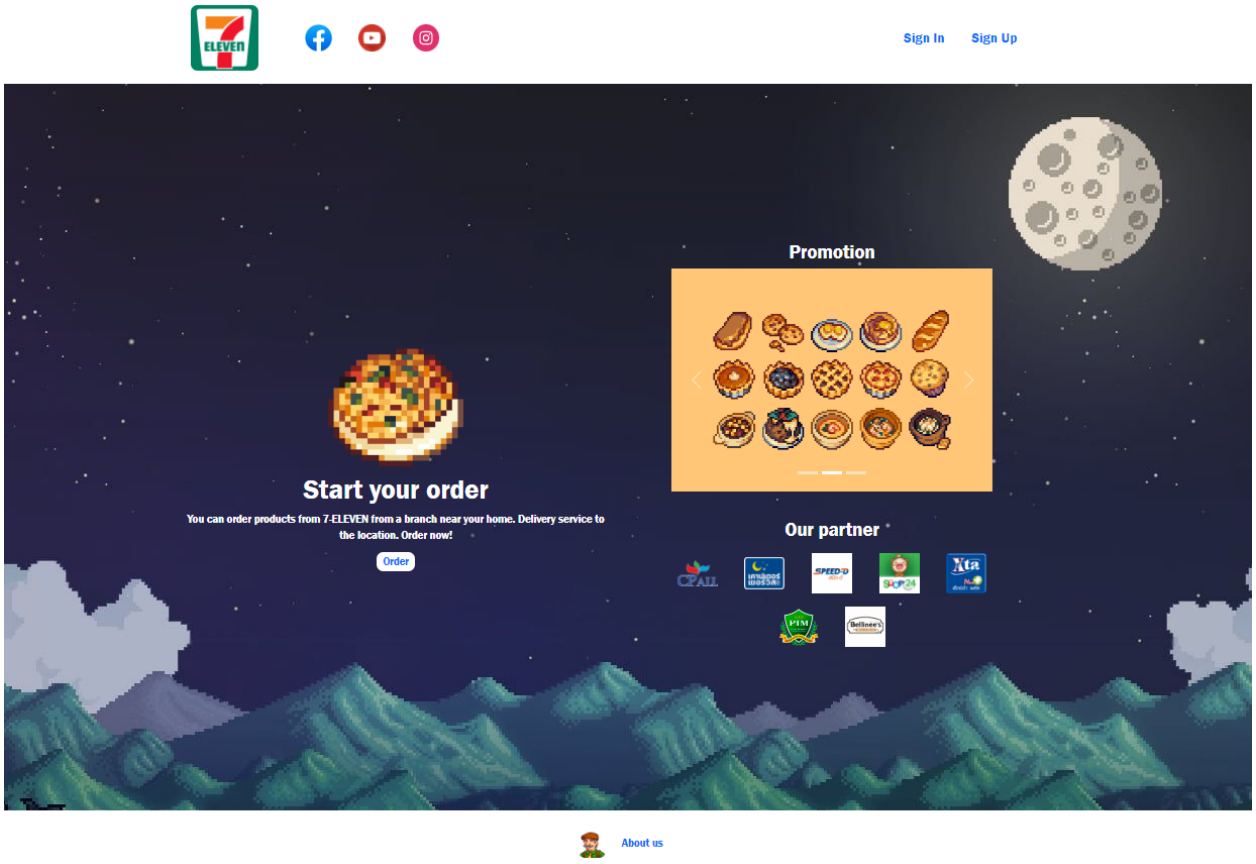

This is a navigation diagram for the user (customer).




This is a navigation diagram for the admin.

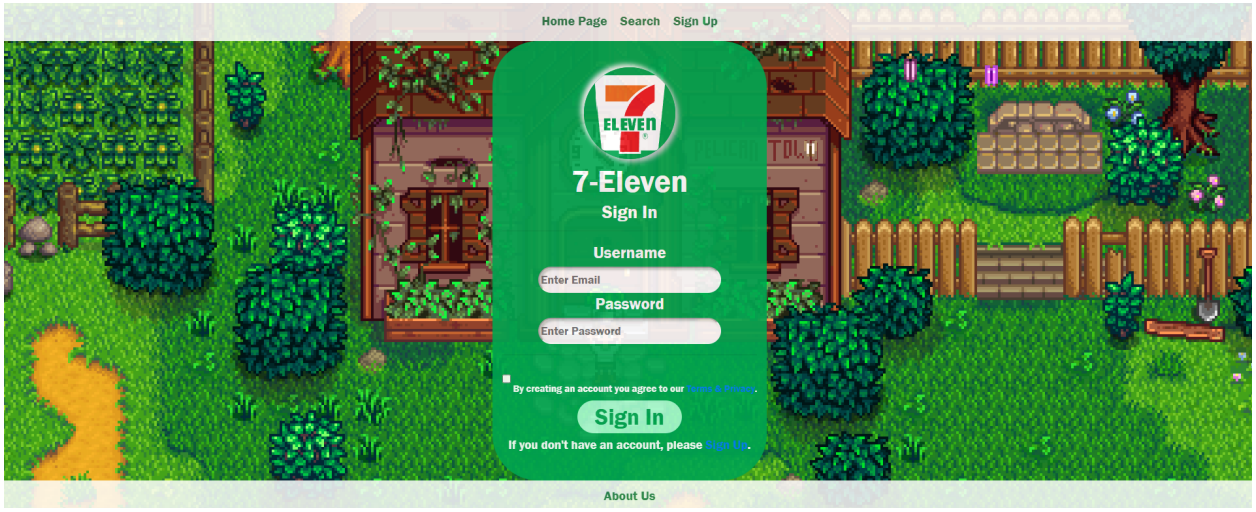
## Home Page

This page will show information about our company including promotion, partnerships, and social media. This page can also link to the sign-in page, sign up, about us, and order our product.



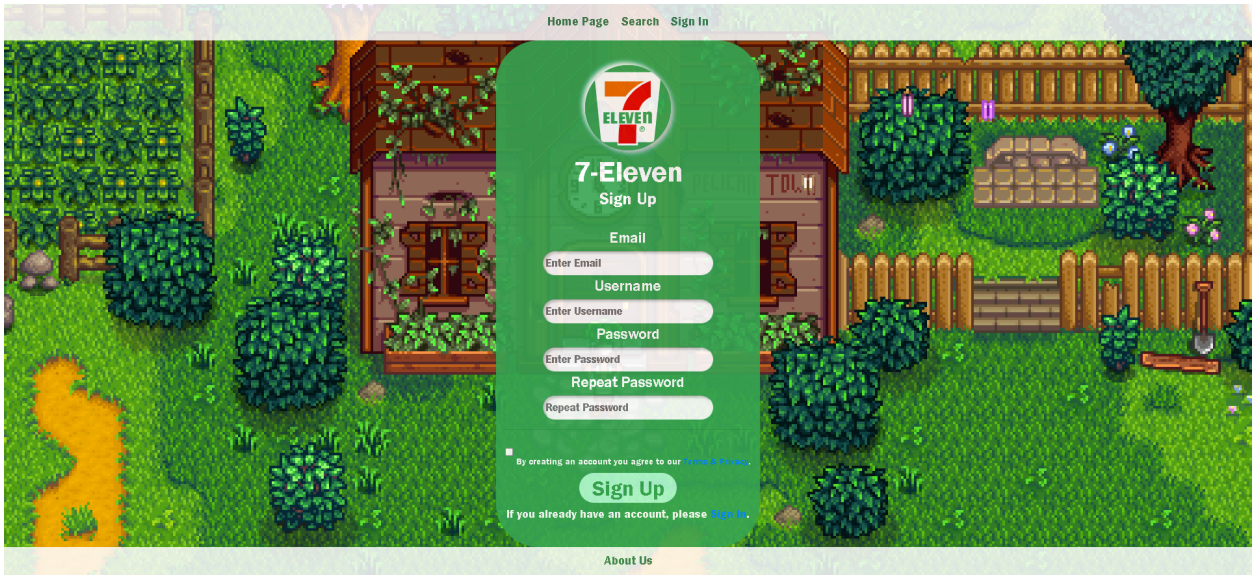
## Login Page

This page is for login into the website. If you don't have an account, you can click the link that links to the sign-up page.



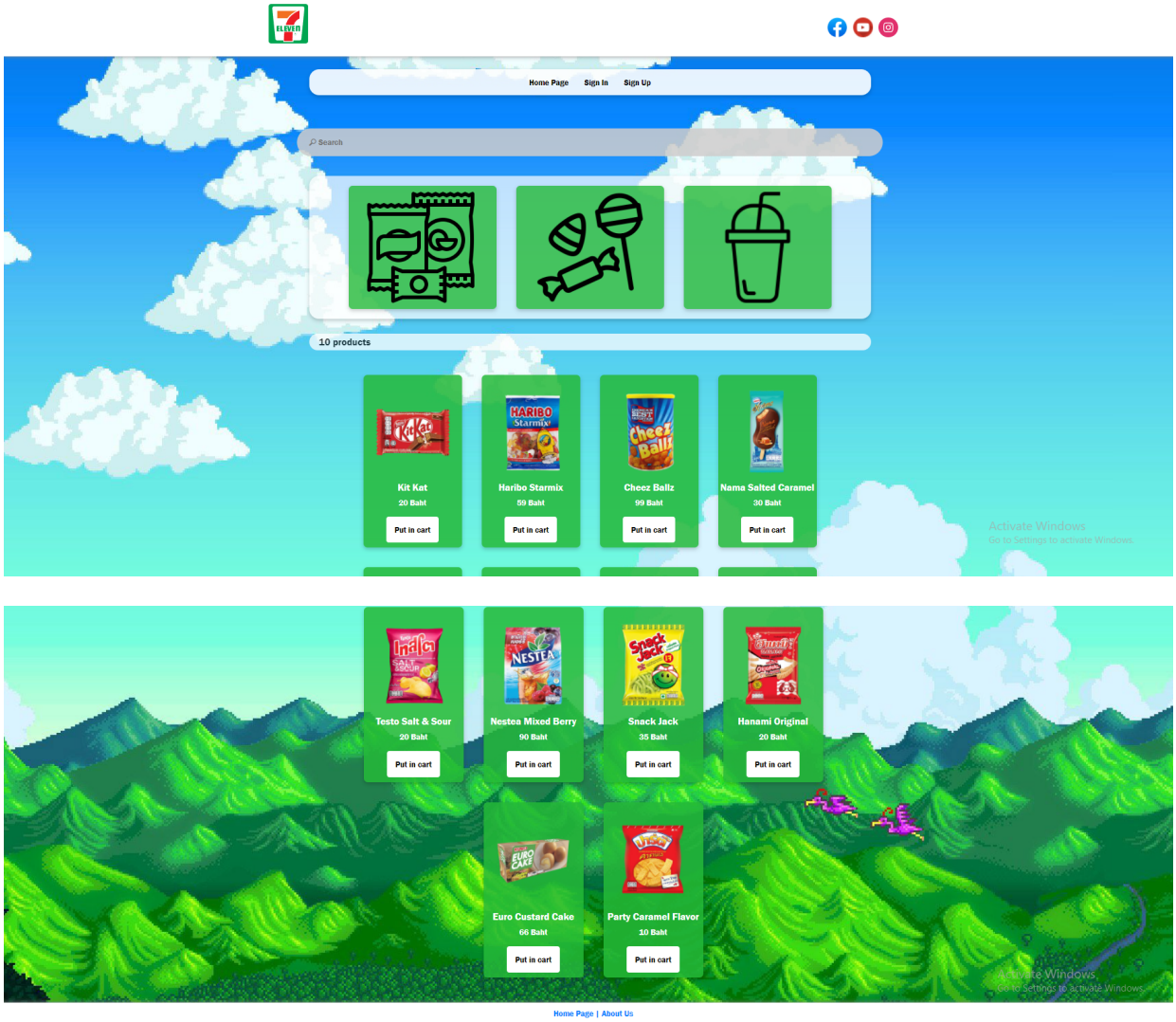

## Sign up Page

This page is for creating an account for being a member of our company.



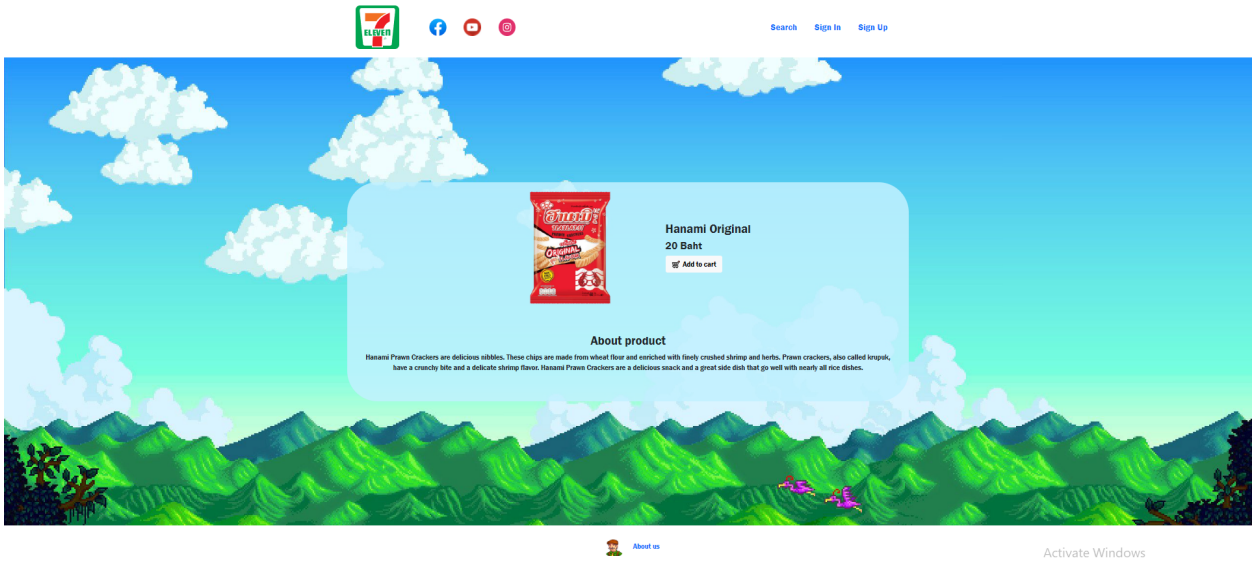
## Search Page

This page is for searching for a product. On the top side, there's a category button for navigating to a categorized search page. For phase one the implemented category are Snack, Sweet, and Drink. In addition, this previous module use a hand arrangement for all links on the categorized search page.



## Result Page

This page is the detail of the product after you click a product on the search page.




## About Us Page

This page is the information about our employees.

## Phase II

### Database

In this section of our database, we use the MySQL program to create our working database. There are some tables we've created in this project. For example Customer, Account, Employee, Product, and Location customer.

### Customer

Our customers will have various values in our database. They will have customer id, title, first name, last name, phone number, email address, and date of birth. Noted that our employees are also counted as one of our customers.

```
CREATE TABLE IF NOT EXISTS Customer(
    Customer_ID     VARCHAR(20)     PRIMARY KEY,
    Title           VARCHAR(5)      NOT NULL,
    Firstname       VARCHAR(50)     NOT NULL,
    Lastname        VARCHAR(50)     NOT NULL,
    Phone           CHAR(10)        NOT NULL,
    Email           VARCHAR(50)     NOT NULL,
    Date_Of_Birth   DATE            NOT NULL,
    CONSTRAINT Title CHECK (Title in ('Miss', 'Mrs', 'Mr'))
);

INSERT INTO Customer VALUES
('115342','Mr','Nititas','Noipalee','0954213674','nititas@gmail.com','1997-10-23'),
('116324','Mr','Narupon','Chokechatree','0845679856','narupon@yahoo.com','1990-02-18'),
('120031','Mr','Supachok','Chaichongmee','0836457251','supachook.chai@hotmail.com','2000-07-11'),
('125031','Mr','Nattapat','Sirilimprapan','0985473210','nattapat@outlook.co.th','1986-09-05'),
('137510','Mrs','Chotirot','Klinthong','0894718632','chotirot@gmail.com','2002-05-25'),
('132011','Miss','Chanakul','Chaliawklang','0812257320','chanakul2002@hotmail.com','2002-05-25'),
('143002','Mr','Pontapan','Polyiam','0957704913','pontapan.pol@yahoo.com','2001-11-07'),
('178001','Mrs','Wathanya','Tabanhan','0867739563','wathanya.tab@gmail.com','1990-01-02'),
('179520','Mr','Ariya','Malai','0985244453','ariya.mai@yahoo.com','1994-06-24'),
('204710','Mrs','Nitchamon','Aworn','0899886413','nitchamon2000@hotmail.com','2000-04-12'),
('0','Mr','Nuttapat','Sumransilp','0967182344','nuttapat.sum@student.mahidol.ac.th','2002-04-18'),
('1','Mr','Poomrapee','Wareeboutr','0957704913','pw12131.contact@gmail.com','2001-07-11'),
('2','Miss','Pitchaya','Teerawongpairoj','0819266322','pitchaya.tee@hotmail.com','2001-07-12'),
('3','Miss','Sasima','Srijanya','0922482545','sasima.sri@gmail.com','2002-05-25');
```

**Account**

For both customer and our employee will have their account, thus this is based on customer data. We will connect our customer data (included with our employees) with their given user names and password. For our employees, we will put the user role as admin to classify our employees from customers.

```
CREATE TABLE IF NOT EXISTS Account(
    Username        VARCHAR(50)     PRIMARY KEY,
    Customer_ID     VARCHAR(20)     NOT NULL,
    User_Password   VARCHAR(16)         NOT NULL,
    User_Role       VARCHAR(10)     NOT NULL CHECK(User_Role in ('admin', 'user')),
    CONSTRAINT FK_Customer_ID FOREIGN KEY (Customer_ID)
    REFERENCES Customer(Customer_ID)
);

INSERT INTO Account(Username,Customer_ID,User_Password,User_Role) VALUES
('nuttapat2344','0','poshinki416','admin'),
('pw12131','1','pw12131MU','admin'),
('pitchaya','2','Bentship33','admin'),
('sasima25','3','Sasima545','admin'),
('nitnoi97','115342','BsTD1hgH','user'),
('naruponcho','116324','Tvf4Wkfx','user'),
('ballsupachok','120031','qtpuQ6Fl','user'),
('0985473210','125031','daT6p0FX','user'),
('bamchotirot89','137510','H1CcsaG2','user'),
('chanakul2002','132011','BJO9w7DF','user'),
('pon11701','143002','ma1lyWBy','user'),
('tabanhan9563','178001','2ZMM6zNp','user'),
('AriyaZa','179520','daJD3DC5','user'),
('nitchamon','204710','b5lwkyUo','user');
```

## Employee

The employee table stores employee information with their employee id, title, first name, last name, phone number, email address, and date of birth.

```sql
CREATE TABLE IF NOT EXISTS Employee(
    Employee_ID     VARCHAR(20) PRIMARY KEY,
    Title           VARCHAR(5) NOT NULL,
    Firstname       VARCHAR(50) NOT NULL,
    Lastname        VARCHAR(50) NOT NULL,
    Phone           CHAR(10) NOT NULL,
    Email           VARCHAR(50) NOT NULL,
    Date_Of_Birth   DATE NOT NULL,
    CONSTRAINT chk_title CHECK (Title in ('Miss','Mrs','Mr'))
);

INSERT INTO Employee(Employee_ID,Title,Firstname,Lastname,Phone,Email,Date_Of_Birth) VALUES
('0638063','Mr','Nuttapat','Sumransilp','0967182344','nuttapat.sum@student.mahidol.ac.th','2002-04-18'),
('0638113','Mr','Poomrapee','Wareeboutr','0957704913','pw12131.contact@gmail.com','2001-11-07'),
('0633133','Miss','Pitchaya','Teerawongpairoj','0819266322','pitchaya.tee@hotmail.com','2001-07-12'),
('0633196','Miss','Sasima','Srijanya','0922482545','sasima.sri@gmail.com','2002-05-25');
```

## Product

The product table contains information about product id, name of the product, amount of member points, price, type of product, product description, and its image. Our available types of products, there are three kinds of them, sweet, snack, and drink.

```
61  CREATE TABLE IF NOT EXISTS Product(
62      Product_ID      VARCHAR(20) PRIMARY KEY,
63      Name_Of_Product VARCHAR(50) NOT NULL,
64      Member_Point    INT NOT NULL,
65      Price           DECIMAL(5, 2) NOT NULL,
66      Product_Type    VARCHAR(50)  NOT NULL,
67      Information     VARCHAR(500) NOT NULL,
68      Img             VARCHAR(200) NOT NULL
69  );
70
71  INSERT INTO Product VALUES
72  ('8891491947194','Kit Kat','2','20','Sweet','These snack-size KIT KAT® Candy Bars are individually wrapped
73  ('8851351151253','Haribo','5.9','59','Sweet','Starmix is a delicious voyage of discovery through the colou
74  ('8851362649514','Cheez Ball','9.9','99','Snack','Cheez Ballz are classically delicious cheese snacks that
75  ('8897461413658','Nama Salted Caramel','3','30','Sweet','The ultimate deliciousness from soft ice cream an
76  ('8894194134907','Testo Salt & Sour','2','20','Snack','Enjoy thin and crispy chips from TASTO Potato Chips
77  ('8869461941947','Nestea Mixed Berry','9','90','Drink','A refreshing mixed berries drink featuring Blackcu
78  ('8864596562626','Snack Jack','3.5','35','Snack','Snack Jack Vegetarian Green Pea Snack has been repackage
79  ('8865296563147','Hanami Original','2','20','Snack','Hanami Prawn Crackers are delicious nibbles. These ch
80  ('8850999321004','Euro Custard Cake','6.6','66','Sweet','Sweet & Soft cream puff cake comes with individua
81  ('8816169530314','Party Caramel Flavor','1','10','Snack','Party Fried Yam Chips Coated with Butter Caramel
```

## Location of customer

The location of the customer will keep information about our customer's location. There's an individual customer's id linked with a location number, house number, village, road, sub-district, district, province, and postal code.

```
CREATE TABLE IF NOT EXISTS Location_Customer(
    Customer_ID     VARCHAR(20) NOT NULL,
    Location_No     VARCHAR(20) PRIMARY KEY,
    House_No        VARCHAR(50) NOT NULL,
    Village         VARCHAR(50) NOT NULL,
    Road            VARCHAR(50),
    Sub_District    VARCHAR(50) NOT NULL,
    District        VARCHAR(50) NOT NULL,
    Province        VARCHAR(50) NOT NULL,
    Post_Code       CHAR(5) NOT NULL,
    CONSTRAINT FK_Customer_ID_2 FOREIGN KEY (Customer_ID)
    REFERENCES Customer (Customer_ID)
);

INSERT INTO Location_Customer(Location_No, Customer_ID, House_No, Village, Road, Sub_District, District, Province, Post_Code) VALUES
('00011','115342','71/2','Phrueksa Village','Daothong','Mahasawat','Phuttamonthon','Nakhon Pathom','73170'),
('00012','116324','61/67','Suralai Housing','Bang Kruai - Sai Noi','Bang Yai','Bang Yai','Nonthaburi','11140'),
('00013','120031','21/1','Ban Rim Suan','Yothathikan','Bang Rak Noi','Meuang','Nonthaburi','11000'),
('00014','125031','12/456','Kaeo Villa','Phuttha Monthon Sai 1','Bang Duan','Phasi Chareon','Bangkok','10160'),
('00015','137510','61/264','Butsaba Village','Phuttha Monthon Sai 2','Sala Thammasop','Thawi','Bangkok','10170'),
('00016','132011','56/23','Kawinthip Village','Borommaratchachonnani','Song Khanong','Sam Phran','Nakhon Pathom','73120'),
('00017','143002','653/09','Ladawan','Ratchaphruek','Taling Chan','Taling Chan','Bangkok','10170'),
('00018','178001','666/11','Royal Tier','Ratchaphruek','Taling Chan','Taling Chan','Bangkok','10170'),
('00019','179520','122/21','Parichart','Ratchaphruek','Bang Khu Wat','Mueang','Pathum Thani','12000'),
('00020','204710','666/09','Royal Tier','Ratchaphruek','Taling Chan','Taling Chan','Bangkok','10170'),
('00021','0','1150/1112','Mazalacha Village','Samlarn','Phra Sing',' Mueang Chiang Mai','Chiang Mai','50200'),
('00022','1','61/67','Pruksa 8','Salaya-Nakornchaisri','Lan Tak Fa','Nakornchaisri','Nakornpathom','73120'),
('00023','2','666/11','Royal Tier','Ratchaphruek','Taling Chan','Taling Chan','Bangkok','10170'),
('00024','3','35/17','Manthana Village','Sukhaphibal 2','Bang Phli Yai','Bang Phli','Samut Prakan','10540');
```

**Web service**

Our web service has 4 major methods to workaround. There're Select, Insert, Update, and Delete. For the Select method, there are 2 differences. One is only one object selected and another one is to select all of the available information.

Here is our web service that is used to connect, receive, and set information from the database. We have used it with customers, employees, products, and account entities.

For example, here is selecting all products from the product entity. We use the function GET to receive all of the products from the database.

```
// Select all products
router.get('/product', function (req, res) {
    console.log(`request at ${req.url}`);

    dbConnection.query(`select * from product`,
    function (error, results) {
        if(error) throw error;
        return res.send(results);
    });
});
```

For example, here is a select product from the product entity. We use the function GET to receive a product by the product ID from the database.

```
// Search a product by ID
router.get('/product/:id',cors(),function (req, res) {
    console.log(`request at ${req.url}`);
    let product_id = req.params.id;
    if (!product_id) {
        return res.status(400).send({ error: true, message: 'Please provide Product_ID.' });
    }
    dbConnection.query('SELECT * FROM product WHERE Product_ID = ?', product_id, function (error, results) {
        if (error) throw error;
        return res.send(results);
    });
});
```

For example, here is an insert product from the product entity. We use the function POST to set a new product by inserting all the product attributes into the database.

```
// Insert data of products
router.post('/product', function (req, res) {
    // res.json({msg: 'This is CORS-enabled for a single Route'})
    console.log(`request at ${req.url}`);
    let product = req.body.Product;
    console.log(product);
    if (!product) {
        return res.status(400).send({ error: true, message: 'Please provide product information' });
    }
    dbConnection.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.'});
    });
});
```

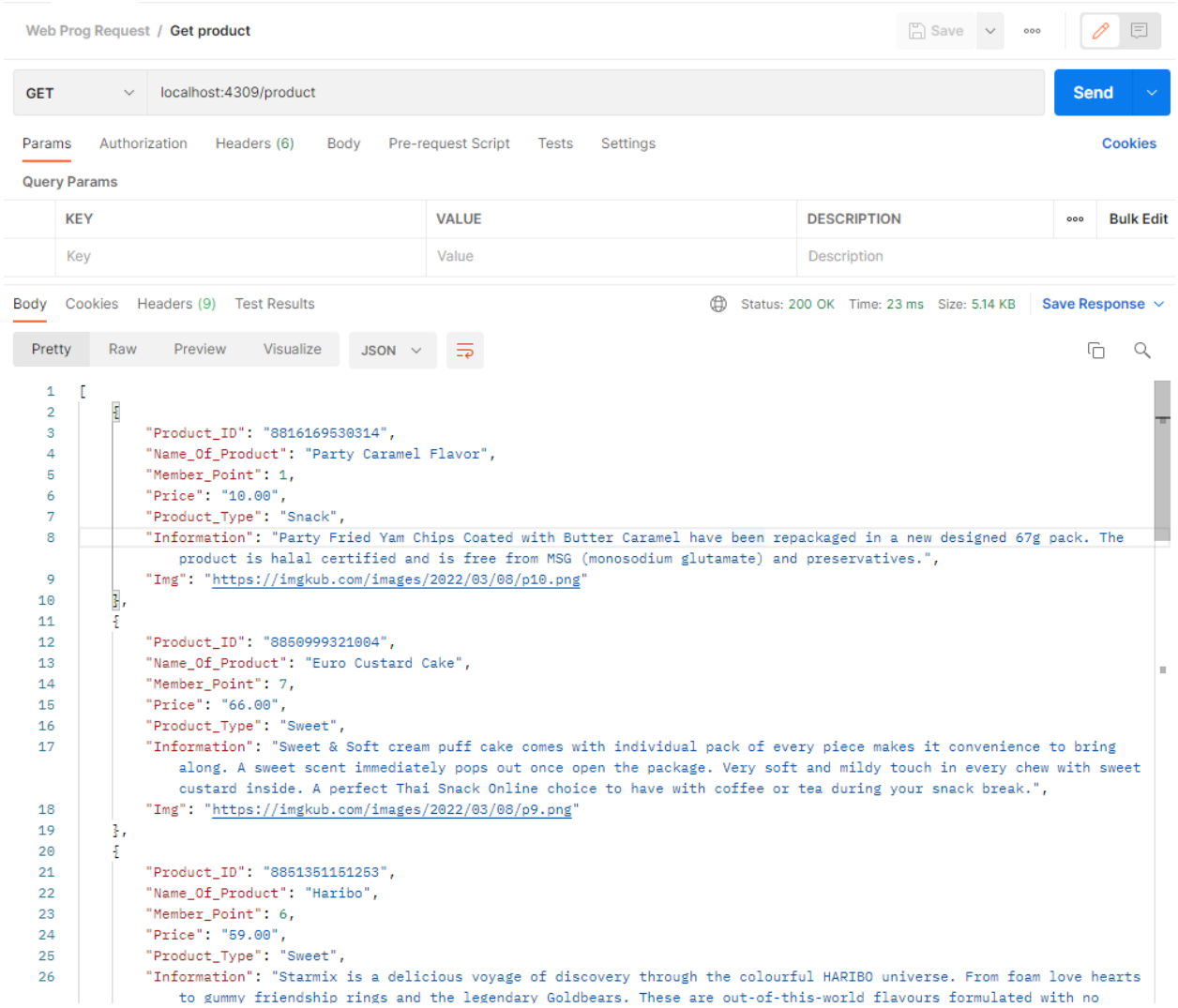
For example, here is an updated product from the product entity. We use the function PUT to update a product based on the product ID by putting the new product attributes into the database.

```
// Update data of product
router.put('/product', function (req, res) {
    console.log(`request at ${req.url}`);
    let product_id = req.body.Product.Product_ID; // <= This format (productID)
    let product = req.body.Product;
    if (!product_id || !product)
    {
        return res.status(400).send({ error: product, message: 'Please provide product information' });
    }
    dbConnection.query("UPDATE product SET ? WHERE Product_ID = ?", [product, product_id], function (error,results)
    {
        if (error) throw error;
        return res.send({error: false, data: results.affectedRows, message: 'Products has been updated successfully.'})
    });
});
```

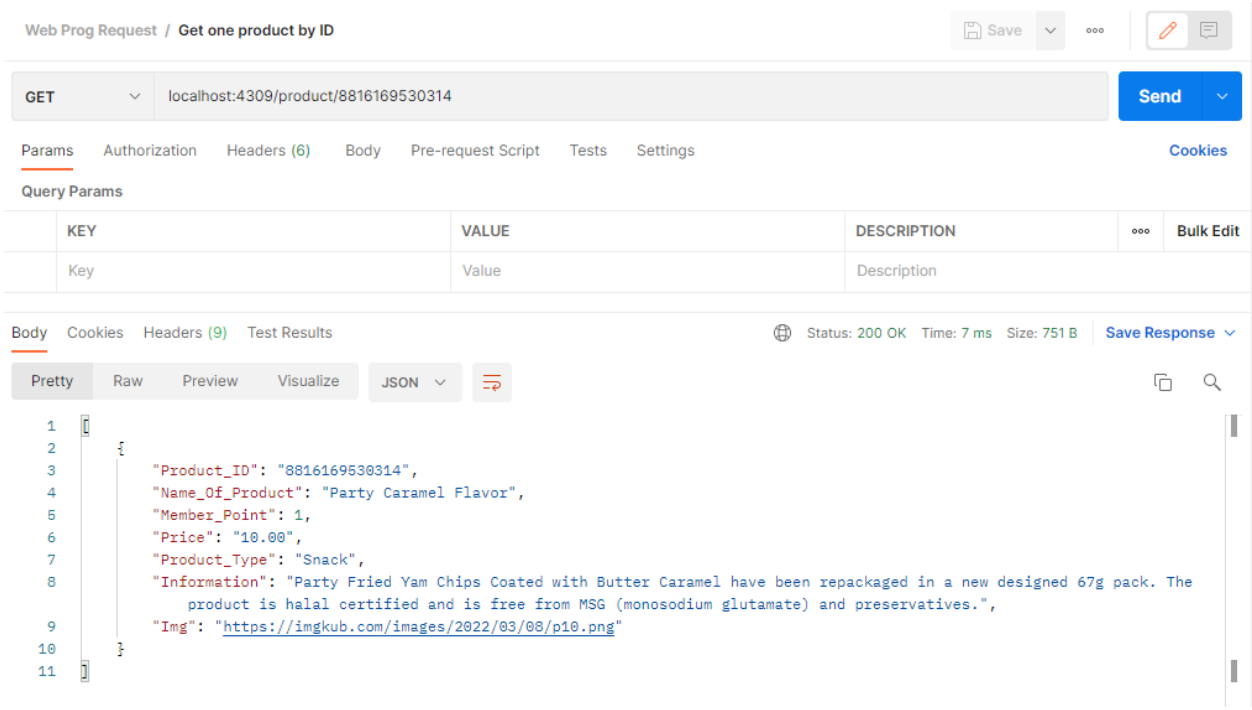
For example, here is a deleted product from the product entity. We use the function DELETE to delete a product based on the product ID from the database.

```
// Delete data of products
router.delete('/product', function (req, res) {
    console.log(`request at ${req.url}`);
    let product_id = req.body.Product_ID;
    if (!product_id) {
        return res.status(400).send({ error: true, message: 'Please provide Product_ID' });
    }
    dbConnection.query("DELETE FROM Product WHERE Product_ID = ?", [product_id], function (error, results)
    {
    if (error) throw error;
        return res.send({ error: false, data: results.affectedRows, message: 'Product has been deleted successfully.' });
    });
});
```

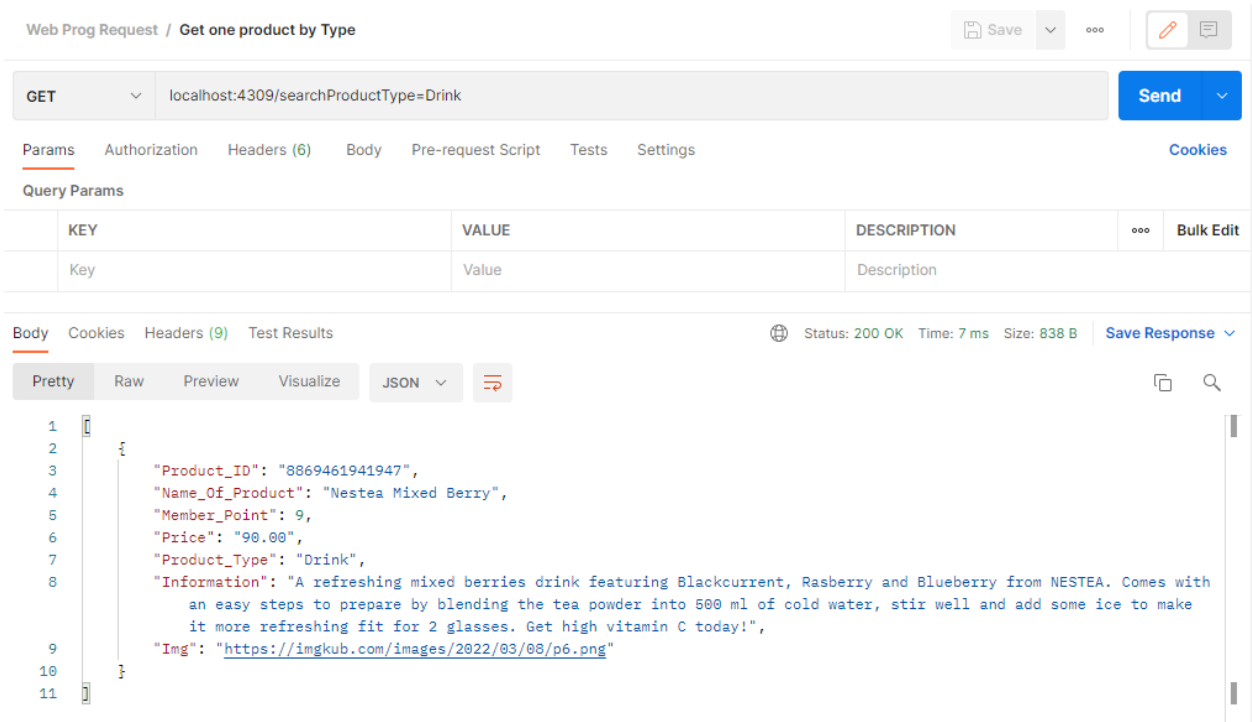
## Select all product



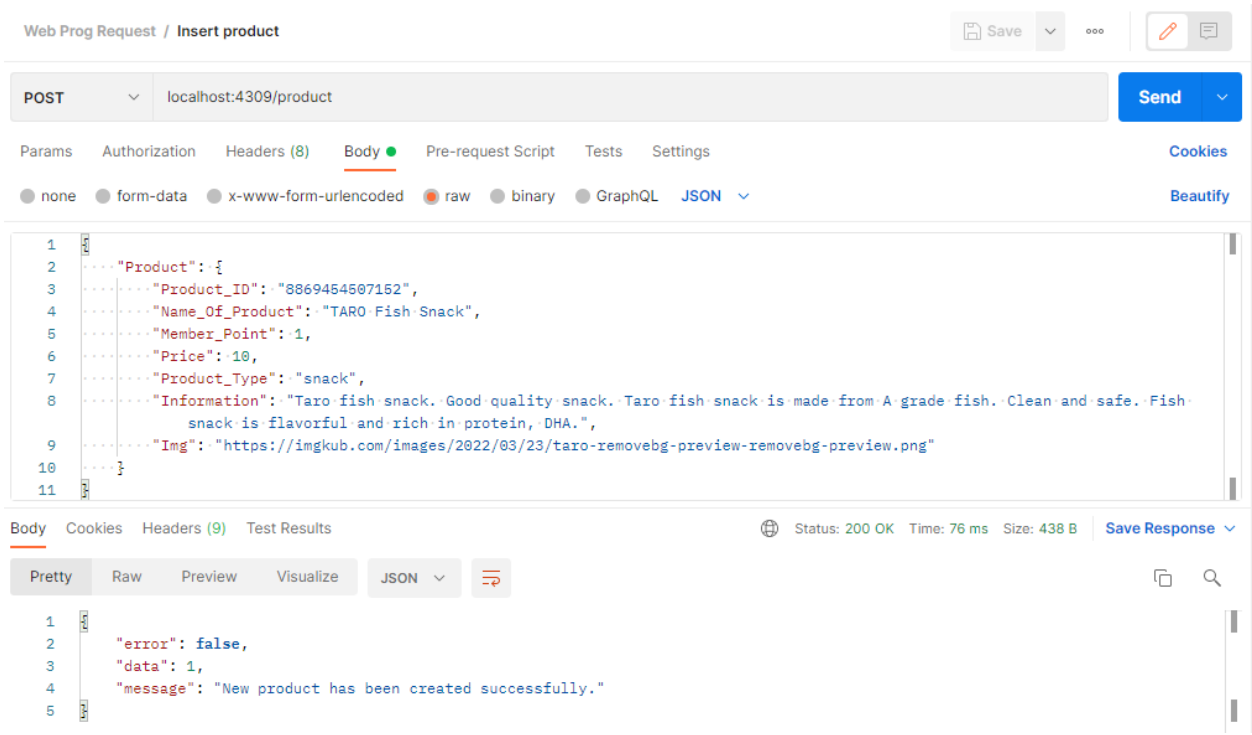
## Select product by ID



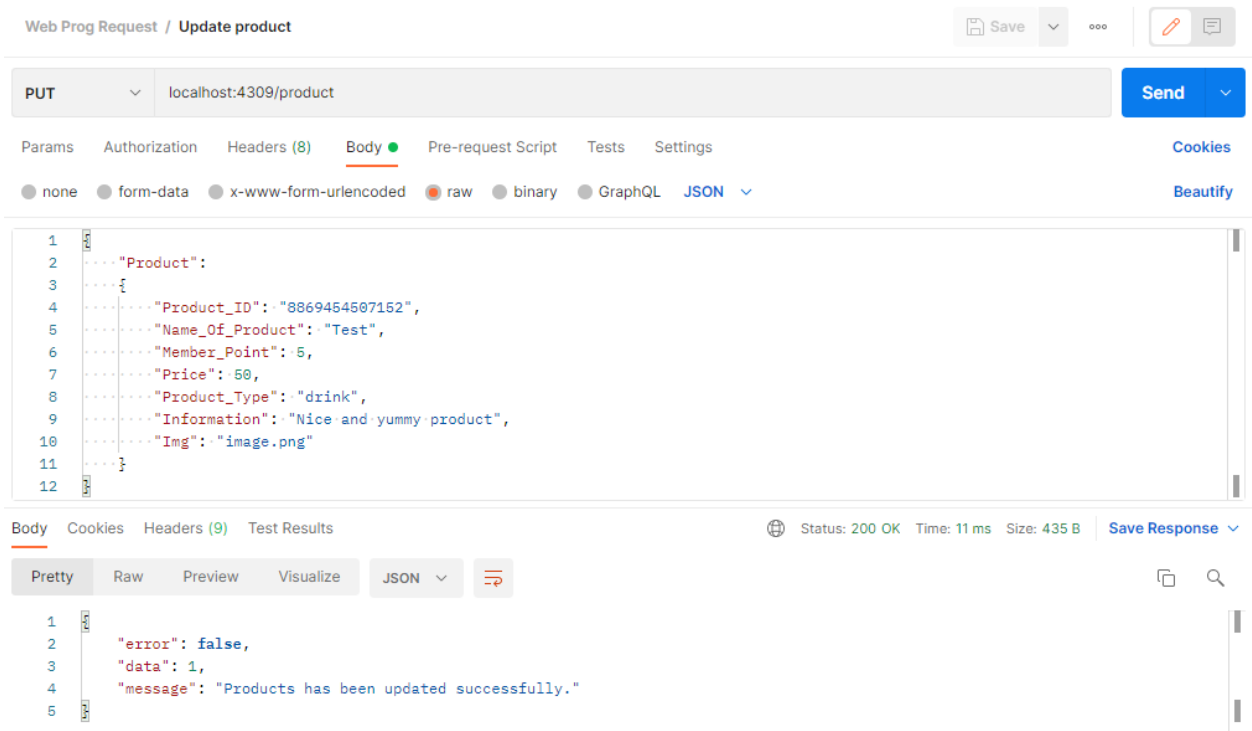

## Select product by Type



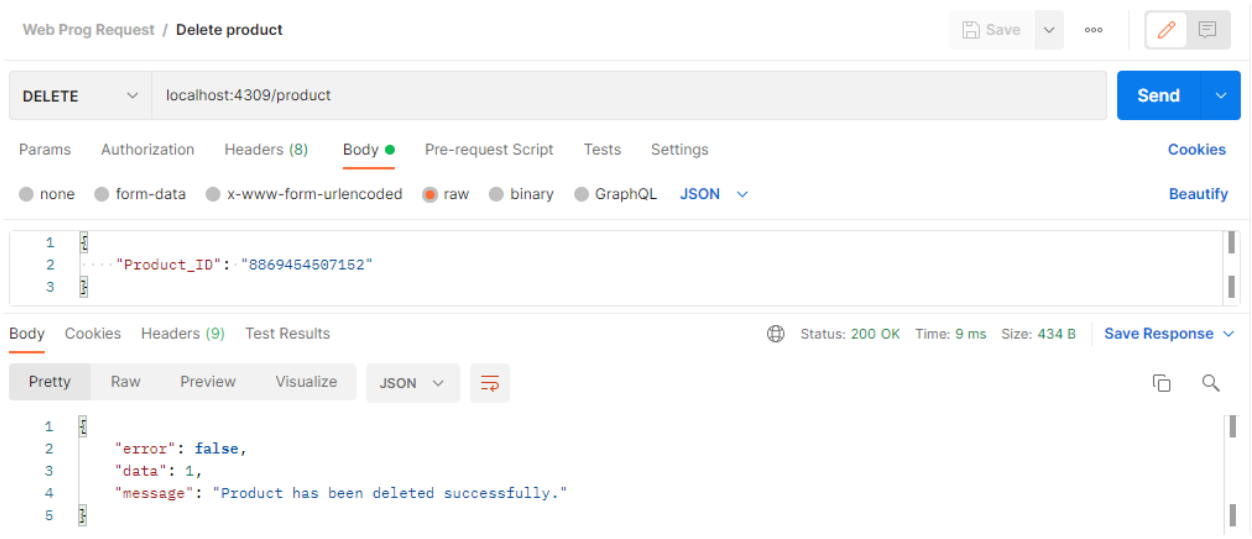
## Insert product




## Update product information



## Delete product



## Authentication

For authentication, we use a checking method to check whether the given username exists in our database or not. Then check whether both the user name and password are correct or not. If the user name does not exist or the given password is wrong, nothing will happen. If the user login successfully, the navigation will bring the user to our home page.

We also use the Json Web Token to authentication or check is it has this username and password in our database or not.  For the cookies we used to tell server where requests come from.

webproject > phase2 > JS cookieJwtAuth.js > ...

```
const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
    const jwtToken = req.cookies.jwtToken;
    try {
        const user = jwt.verify(jwtToken, process.env.SECRET);
        req.user = user;
        next();
    } catch(err) {
        res.clearCookie('jwtToken');
        // res.status(401).json({status: 'authentication failed' });
        res.send(`
            <form action="/login">
                <label>you have to login first</label>
                <button type="submit">sign in</button>
            </form>
        `);
    }
};
```

webproject > phase2 > JS generateAccessToken.js > ...

```
const jwt = require('jsonwebtoken');

module.exports = (user) => {
    return jwt.sign(user, process.env.SECRET, {expiresIn: '1m'});
};
```

## Web Services Interaction

This part is used to connect JSON Web Token with Login Page.

```
595  ////////////////////////////////////////// Login //////////////////////////////////////////////////
596  // This route is used to simple test route for authentication
597  router.get('/home', cookieJwtAuth, (req, res) => {
598      console.log(`request at ${req.url}`);
599      res.send('login successfully\n' + 'username: ' + req.user.username + '\nrole: ' + req.user.role);
600  });
601
602  // Route to the login page
603  router.get('/login', (req, res) => {
604      console.log(`request at ${req.url}`);
605      res.sendFile(path.join(__dirname, './login_admin/src/project_phase3/signin.js'));
606  });
607
608  // Route after the form is action
609  router.post('/login-submit', (req, res) => {
610      const username = req.body.username;
611      const password = req.body.password;
612
613      dbConnection.query(`
614      select username, user_password, user_role from account where username = ?
615      `,username,
616      (err, result) => {
617          if(err) throw err;
618          if(result.length == 0) {
619              res.redirect('/login');
620          } else {
621              if(password == result[0].user_password) {
622                  const jwtToken = generateAccessToken({username: result[0].username, role: result[0].user_role});
623                  res.cookie('jwtToken', jwtToken);
624                  res.redirect('/home');
625              } else {
626                  res.redirect('/login');
627              }
628          }
629      });
630  });
631  ////////////////////////////////////////////////////////////////////////////////////////////////////
```

## Fetch

This part is a fetch. Our group fetch from [https://spoonacular.com/food-api](https://spoonacular.com/food-api)

fetchSpoonacular.js

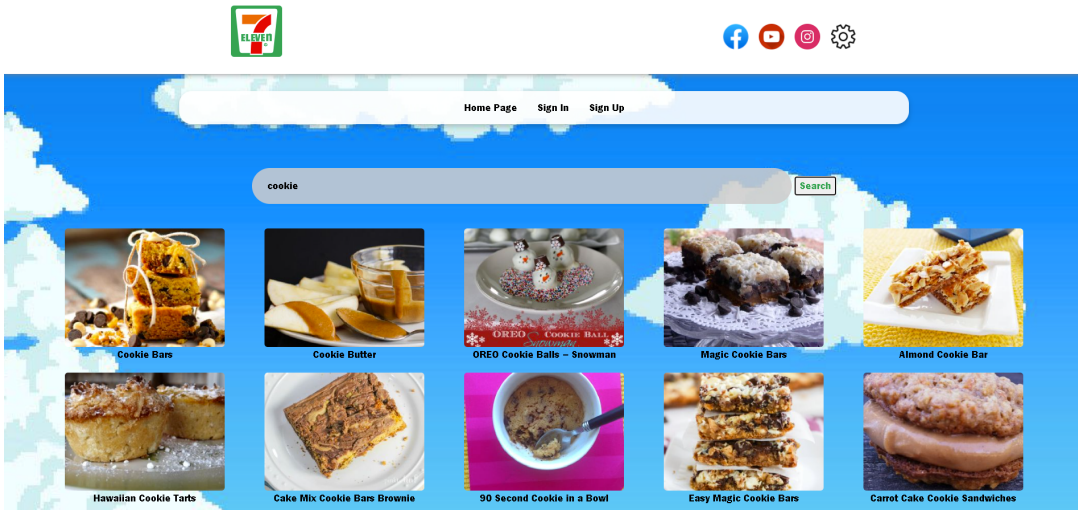
```
webproject > phase2 > JS fetchSpoonacular.js > searchMenu
function searchMenu(q) {
    const url = `https://api.spoonacular.com/recipes/complexSearch?apiKey=97474afb00144ea78526febd9948d2cf&query=${q}`;
    fetch(url)
    .then(response => response.json()) // Get JSON from the response
    .then(data => {
        const html = data.results.map(element => {
            return `
            <div class = "menu">
                <img src = ${element.image} style="width:80%"></img>
                <p style="color:black">${element.title}</p>
            </div>`;
        }).join("");
        document.querySelector("#resultMenu").innerHTML = html;
    })
    .catch((err) => console.log(err));
}

+
```

In search.html we connected to fetchSpoonacular.js to call APIs to our website.

search.html

```
    <script src="static/fetchSpoonacular.js"></script>
        <!-- This is the place where you can find your product items -->
        <div class="search bar">
        <input type="text"  class="search input" id="txtSearch" placeholder="Search">
        <button input="submit" onclick="searchMenu(document.getElementById('txtSearch').value)">Search</button>
        <h1 id="h1_title"></h1>
        <div id="resultMenu" style="padding-top:30px; display:flex; flex-direction:row;
            flex-wrap: wrap; justify-content: center; text-align: center;"></div>
    </div>
```

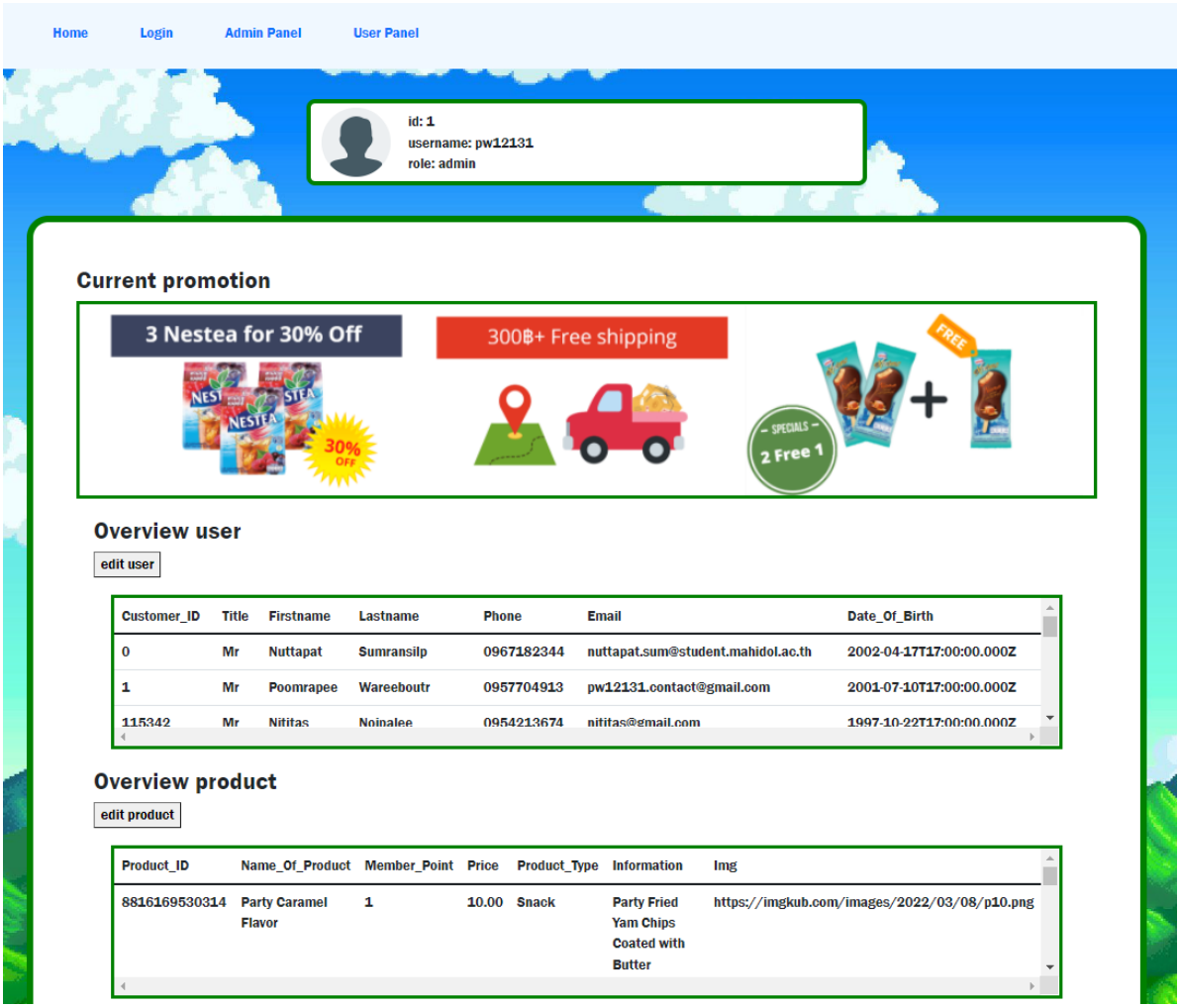
Here is our result of the fetch function that got the API data from the Spoonacular website.



# Phase III

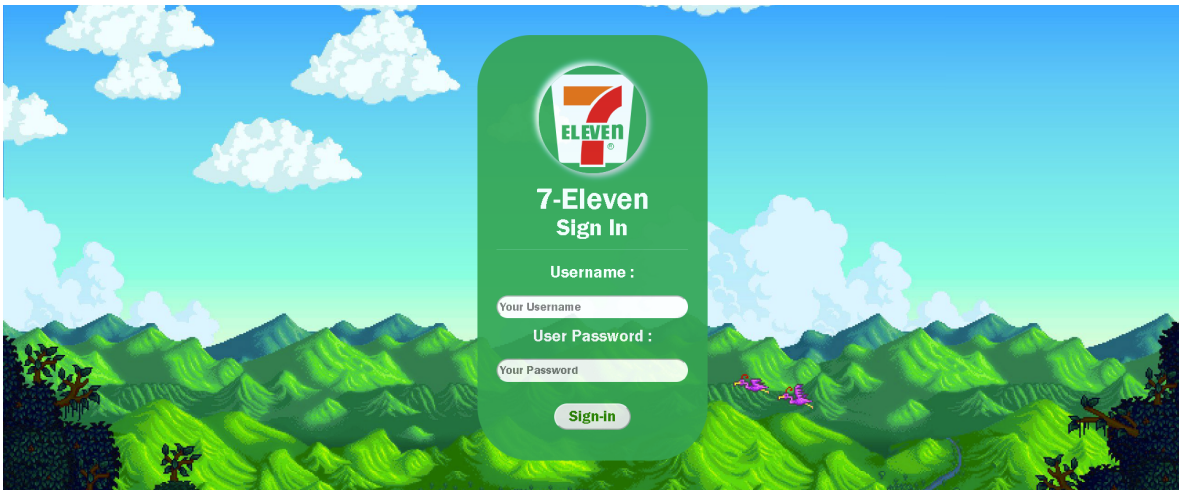
## Homepage

The homepage for the admin side consists of the admin's information to tell specific details about the username and role of that account, the promotion bar that would tell an admin what current promotions are related to the goods, and include 2 quick-look tables of the user and product tables in the database.



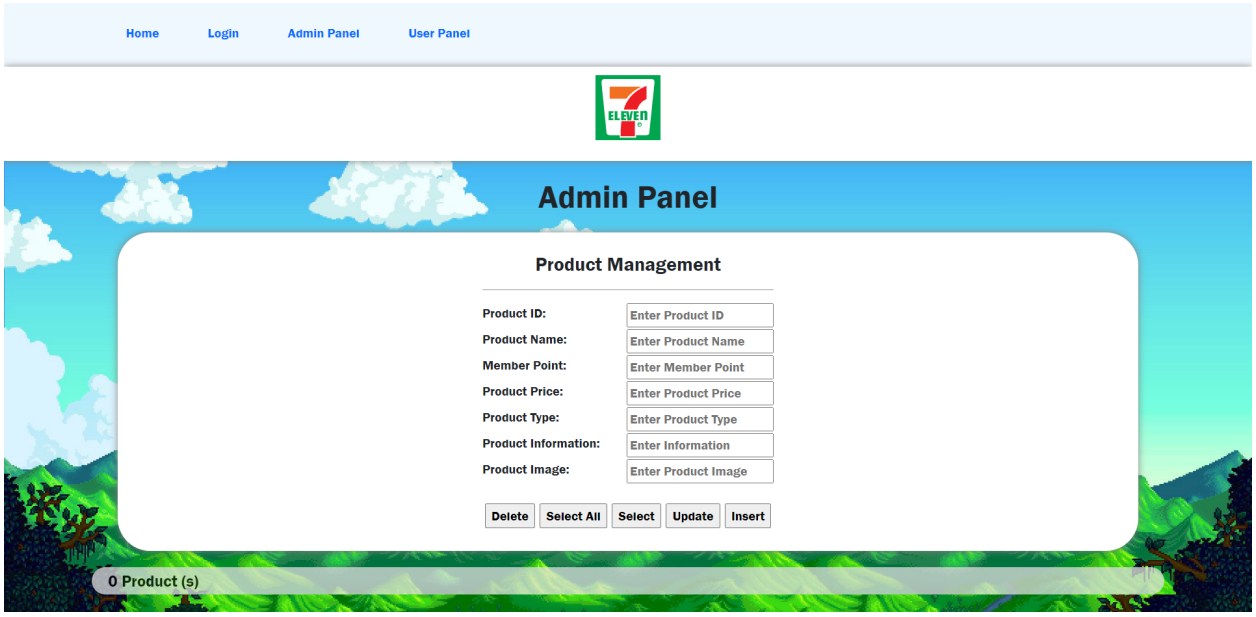
**Login Page**

The login page is for users to log in. The user has to fill in the username and password. If the user puts the correct username and password correctly, it will go to the home page. If the user puts the wrong username and password, it will stay on the login page.



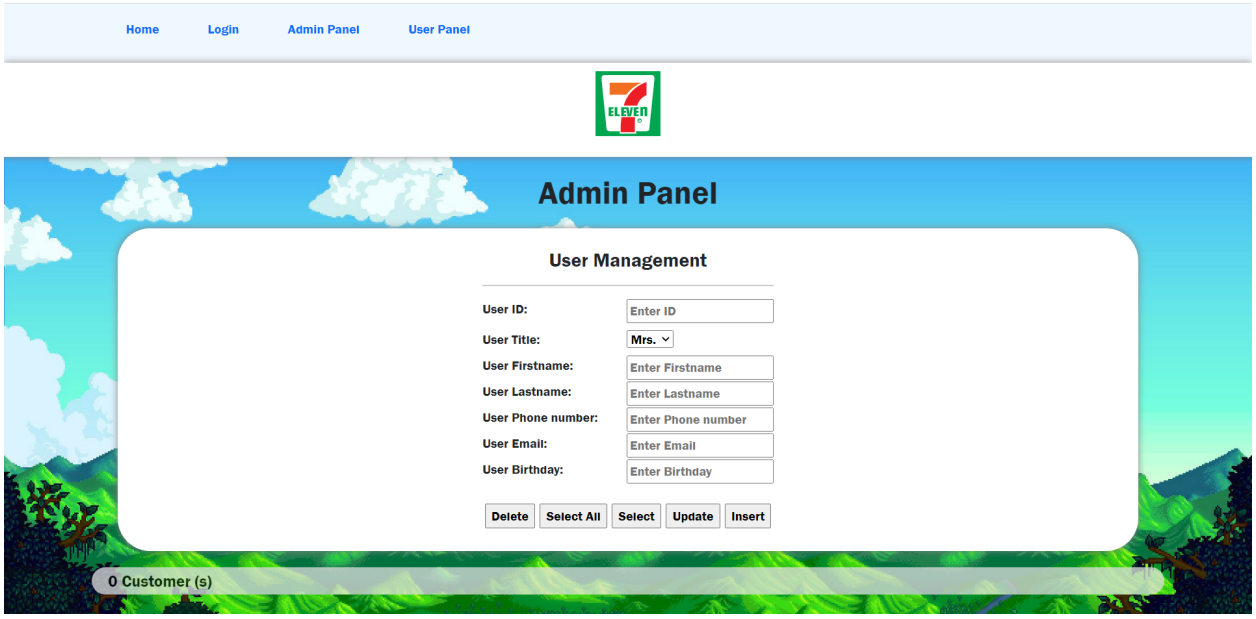
## Admin Panel for Product Management

The admin panel for product management is the place where the user can insert, update, delete, select all products, select a product by ID, select product by name, and select a product based on their type. Then it will show the result items below the control panel and if you click on each of the product images it will link to the result information page for each of the products to look for more insight information.



## Admin Panel for User Management

The admin panel for user management is the place where the user can insert, update, delete, select all users, select a user by ID, select a user by the first name, and select a user based on their title. Then it will show the result items below the control panel and if you click on the "Provide More Information" link it will connect to the result information page for you to see more information about each user.



### Select all

The select all function calls a “SELECTALL” to the database and returns every available data from the product data table.

```
selectall() { /* Call Web Service to select all product */
    axios.get("http://localhost:4309/product")
    .then((response) => {
        console.log(response.data);
        const products = response.data;
        this.setState({products, click: true, type: "selectall"});
    })
}
```

### Select

The selection function calls a “SELECT” function from the database based on what information has been given. The user can input up to 3 kinds of data, but our algorithm will look for only one from the given. This search priority will be arranged by ID, name, and product type. In the case of searching by name and type, the function will be calling “SELECTALL” which is related to the given name and type from the database instead.

```
select(e) { /* Call Web Service to select a product */
    e.preventDefault();
    if(this.state.pro_id !== ""){
        axios.get("http://localhost:4309/product/"+this.state.pro_id)
        .then((response) => {
            console.log(response.data);
            const products = response.data;
            this.setState({products, click: true, type: "select"});
        })
    } else if(this.state.pro_name !== ""){
        axios.get("http://localhost:4309/searchProductName="+this.state.pro_name)
        .then((response) => {
            console.log(response.data);
            const products = response.data;
            this.setState({products, click: true, type: "selectall"});
        })
    } else if(this.state.pro_type !== ""){
        axios.get("http://localhost:4309/searchProductType="+this.state.pro_type)
        .then((response) => {
            console.log(response.data);
            const products = response.data;
            this.setState({products, click: true, type: "selectall"});
        })
    }
}
```

**Insert**

The insert function works similarly to Update's structure and app/Jason format, but instead of "PUT" we'll be using "POST" to create new data into our database. If there's already existing data, the program will throw an error.

```
insert(e) { /* Call Web Service to insert a new product */
    e.preventDefault();
    fetch("http://25.13.241.170:4309/product", {
        "method": "POST",
        "headers": {
            "content-type": "application/json",
            accept: "application/json"
        },
        "body": JSON.stringify({
            Product: {
                Product_ID: this.state.pro_id,
                Name_Of_Product: this.state.pro_name,
                Member_Point: this.state.pro_mm,
                Price: this.state.pro_price,
                Product_Type: this.state.pro_type,
                Information: this.state.pro_info,
                Img: this.state.pro_img
            }
        })
    })
    .then((response) => response.json())
    .then((response) => {
        console.log(response);
        this.selectall();
    })
    .catch((err) => {
        console.log(err);
    });
}
```

**Delete**

The delete function deletes existing data by the product number (Product_ID). Call a “DELETE” function into our database, locate, and delete the existing data. If the given product number cannot be found, nothing will happen.

```
delete(e) { /* Call Web Service to delete a product */
    e.preventDefault();
    fetch("http://localhost:4309/product", {
        "method": "DELETE",
        "headers": {
            "content-type": "application/json",
            accept: "application/json"
        },
        "body": JSON.stringify({
            Product_ID: this.state.pro_id,
        })
    })
    .then((response) => response.json())
    .then((response) => {
        console.log(response);
        this.selectall();
    })
    .catch((err) => {
        console.log(err);
    });

}
```

**Update**

The update function will call “PUT” to the database, with the attached new data for the update. We’ll put all the information into the application/Jason’s body. With attribute Product_id, Name_of_product, Member_point, Price, Product_type, Information (detail text), and Img.

```
update(e) { /* Call Web Service to update a product */
    e.preventDefault();
    fetch("http://25.13.241.170:4309/product", {
        "method": "PUT",
        "headers": {
            "content-type": "application/json",
            accept: "application/json"
        },
        "body": JSON.stringify({
            Product: {
                Product_ID: this.state.pro_id,
                Name_Of_Product: this.state.pro_name,
                Member_Point: this.state.pro_mm,
                Price: this.state.pro_price,
                Product_Type: this.state.pro_type,
                Information: this.state.pro_info,
                Img: this.state.pro_img
            }
        })
    })
    .then((response) => response.json())
    .then((response) => {
        console.log(response);
        this.selectall();
    })
    .catch((err) => {
        console.log(err);
    });
}
```